SIEMENS EDA

Calibre® nmSRAF™ User's and Reference Manual

Software Version 2021.2

SIEMENS

Unpublished work. © 2021 Siemens

This material contains trade secrets or otherwise confidential information owned by Siemens Industry Software, Inc., its subsidiaries or its affiliates (collectively, "Siemens"), or its licensors. Access to and use of this information is strictly limited as set forth in Customer's applicable agreement with Siemens. This material may not be copied, distributed, or otherwise disclosed outside of Customer's facilities without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This document is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made. Siemens disclaims all warranties with respect to this document including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement of intellectual property.

The terms and conditions governing the sale and licensing of Siemens products are set forth in written agreements between Siemens and its customers. Siemens' **End User License Agreement** may be viewed at: www.plm.automation.siemens.com/global/en/legal/online-terms/index.html.

No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

TRADEMARKS: The trademarks, logos, and service marks ("Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' trademarks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Table of Contents

Chapter 1
Introduction to Calibre nmSRAF 13

Calibre nmSRAF Workflow 13
Calibre nmSRAF Key Concepts 14
Calibre nmSRAF Requirements 14
Syntax Conventions 15

Chapter 2
Template Methodology for Applying SRAFs 17

Calibre cnSRAF Template Execution and Cleanup 17
Calibre cnSRAF Conflict Regions 20
Calibre cnSRAF and Calibre OPCsbar Integration 23

Chapter 3
Considerations for Writing Calibre nmSRAF Rules 25

Requirements and Assumptions 25
Layout Considerations 25
Template Recipe Priority 26
Calibre pxOPC Generation 27

Chapter 4
Calibre nmSRAF Command Reference 29

SVRF Commands 31
RET [EUV] MBSRAF 32
RET SBAR 36
LITHO FILE Statement and Commands 39
LITHO FILE 41
background 45
clone_transformed_cells 47
ddm 48
ddm_model_load 49
flare_longrange 51
image_options 53
layer 55
litho_model 57
modelpath 60
optical 61
optical_model_load 62
plane 63
processing_mode 64
progress_meter 65
resist | threshold 66

resist_model_load 67
setlayer cnsraf 68
setlayer cnsraf debug 69
setlayer mbsraf 75
setlayer mbsraf debug 77
setlayer tilegen 79
sraf_options 80
svrf_var_import 91
tilemicrons 93
LITHO FILE Statements for EUV 94
setlayer sraf_options Keywords 95
Layer Keywords 96
correctionlayer 97
offsetlayer 98
prioritylayer 100
sblayer 101
taglayer 102
Tag Keywords 103
newtag boolean 106
newtag external and internal 108
newtag layer 114
newtag length 116
newtag orientation 118
newtag pairtag 121
newtag recta and rectb 125
newtag square 127
Constraint Keywords 128
alignorigin 130
extension 131
maxlength 132
maxoffset 134
maxoffsetnegative 136
minarea 138
minareaextension 139
minfeaturespace 140
minlength 151
minlengthextension 152
minoffset 153
minoffsetlayerspace 158
minsquarearea 163
minsquarelength 164
minwidth 165
shielding 166
squarewidth 171
width 172
Templates 173
localmaxgradient 174
pairtemplate 177
rect 181

rect45 183
rect45corner 186
rectcorner 189
rectdiag 209
rectedge. 215
recthv 223
srafring 226
srafringhv 228
Style Keywords. 230
algorithm. 231
centerslit 233
centerwidth 234
collecttype. 235
cornerpref 237
cornerprefnegative 240
gradientfactornegative 242
gradientshift 243
image_options. 244
intersection 246
intersectionnegative 255
jogignore. 259
mode 261
negative. 262
peaktype 265
pw_condition 266
slopeweight. 268
symmetric 269
Conflict Keywords 270
centerthresh. 271
conflictmode 274
maxconflict 278
oversizesraf. 280
sbwidth 282
sraf 283
Cleanup Keywords 284
centermerge 285
cleanmode. 288
cleanupversion 290
cnsrafpromotion 294
dynamicsizing. 295
dynamicsizing_ambiguous 298
dynamicsizingrunlength 299
interpriorityalign. 301
intrapriorityalign. 304
lineendmerge 308
maxmove 311
reducedbarsplit 312
Priority Keywords. 314
priority 315

Chapter 5
Calibre nmSRAF Best Practices 323

Calibre nmSRAF Recipe Development 323
Calibre mbSRAF Recipe Development 333
Calibre cnSRAF Template Development Tips 343
Recommended Settings for Best cnsraf Results 345
Recommended Settings for Best mbsraf Results 348
Calibre nmSRAF Recipe LINT Messages 349

Appendix A
SRAF Cleanup 353

SRAF Cleanup 353

Appendix B
Running Calibre nmSRAF 357

calibre 358

Appendix C
Using the Calibre WORKbench RET Flow Tool With Calibre nmSRAF 359

RET Flow Tool and Calibre nmSRAF 360
Invoking the RET Flow Tool 360
Importing an Existing OPC Session to RFT 362
Extracting an mbsraf Session From an SVRF File 364
Setting RFT Application Options 367
Performing Gradient Mapping 371
Drawing Intensity Cutlines 374

Index

Third-Party Information

List of Figures

Figure 2-1. Typical Conflict Regions 21
Figure 2-2. Conflict Region 21
Figure 2-3. rectdiag With and Without conflictonly 23
Figure 4-1. INSIDE OF LAYER 35
Figure 4-2. INSIDE OF LAYER 38
Figure 4-3. lineend Debug Layers 73
Figure 4-4. Field and Template Output 73
Figure 4-5. Seedpixel Debug Output Layers 78
Figure 4-6. Tile and Hierarchy Effects on Tagging 104
Figure 4-7. mbsraf newtag external Usage 111
Figure 4-8. newtag fulledge Usage 112
Figure 4-9. newtag From and To Tag Usage 113
Figure 4-10. Newtag Orientation 120
Figure 4-11. The distance Keyword 121
Figure 4-12. An anchorlayer Application 122
Figure 4-13. The angle Keyword 122
Figure 4-14. The projection Keyword 122
Figure 4-15. Connector 123
Figure 4-16. Edge Selection With recta and rectb 126
Figure 4-17. maxlength 133
Figure 4-18. SRAF Limits with maxoffset 135
Figure 4-19. SRAF Limits with maxoffsetnegative 137
Figure 4-20. Valid and Invalid Line-ends 144
Figure 4-21. Protection Dimensions Relative to lineend2long 145
Figure 4-22. Protection Dimensions Relative to long2lineend 146
Figure 4-23. Using minfeaturespace and lineend 146
Figure 4-24. Using minfeaturespace With and Without projecting 146
Figure 4-25. Application of lineend Keyword Options 147
Figure 4-26. Using minfeaturespace and minoffset with mbsraf 148
Figure 4-27. Application of lineend Keyword with mbsraf 148
Figure 4-28. Square-Oriented Option Usage with Square Target 149
Figure 4-29. Square-Oriented Option Usage with Square SRAFs and Non-square Target . . 149
Figure 4-30. Using minoffset and lineend 155
Figure 4-31. Using minoffset With and Without projecting 156
Figure 4-32. Square-Oriented Option Usage with Square SRAF 156
Figure 4-33. Square-Oriented Option Usage with Square SRAFs 157
Figure 4-34. Application of minoffsetlayerspace 161
Figure 4-35. Keyword Application With Square offsetlayer 162
Figure 4-36. Keyword Application With Non-Square offsetlayer 162
Figure 4-37. minwidth Behavior Change 165

Figure 4-38. Positive SRAFs in Comb Pattern 167
Figure 4-39. SRAFs Responding to shielding seedpixel Option 167
Figure 4-40. Positive SRAFs in Stripe Pattern 168
Figure 4-41. Positive SRAFs in Irregular Pattern 168
Figure 4-42. Negative SRAFs in Irregular Pattern 169
Figure 4-43. Negative SRAFs in Irregular Patterns 170
Figure 4-44. SRAF Output Using Two Templates of Different Priorities With and Without maxoffset 176
Figure 4-45. Angle influence on SRAF placement 177
Figure 4-46. Offset less than half connector length 178
Figure 4-47. Offset greater than half connector length 178
Figure 4-48. The offset and angle keywords together 178
Figure 4-49. SRAFs oriented in vertical direction 178
Figure 4-50. rect Default Usage 182
Figure 4-51. rect45 Default Usage 184
Figure 4-52. Combination rect45 and recthv Usage 185
Figure 4-53. rect45corner Default Behavior 187
Figure 4-54. rect45corner angledalso Behavior 188
Figure 4-55. rectcorner Default SRAF Placement Behavior 195
Figure 4-56. rectcorner With Identical offset, width and length Values 195
Figure 4-57. rectcorner Usage With offset, width and length Variations 196
Figure 4-58. orient parallel and orient perpendicular 196
Figure 4-59. perpendicular and parallel Treatment of Square Target Shapes 197
Figure 4-60. rectcorner Using breaksym 198
Figure 4-61. scalefactor Index Effects 198
Figure 4-62. scalefactor offset and minoffset Effects 198
Figure 4-63. scalefactor Called by Multiple rectcorner Templates 199
Figure 4-64. scalefactor and orient 45 200
Figure 4-65. With protectionradius 201
Figure 4-66. Without protectionradius 202
Figure 4-67. scalefactor With Variable Offsets 203
Figure 4-68. center Without Options 203
Figure 4-69. center With angle Keyword 204
Figure 4-70. center With offset Keyword 204
Figure 4-71. Default Behavior of center 205
Figure 4-72. center strictcenter 206
Figure 4-73. Voronoi-Based SRAF Placement 207
Figure 4-74. Voronoi SRAF Placement Using offset 207
Figure 4-75. Voronoi SRAF Placement Using offsetmultiplier 208
Figure 4-76. rectdiag SRAF Placement Behavior 211
Figure 4-77. rectdiag Default Usage 211
Figure 4-78. rectdiag Usage With Identical X and Y Pitches 212
Figure 4-79. rectdiag Rectangle SRAF Placement 212
Figure 4-80. rectdiag SRAF Placement With breaksym 213
Figure 4-81. rectdiag SRAF Placement Using orient 214

Figure 4-82. rectedge Example 219
Figure 4-83. angledalso Example 219
Figure 4-84. center Without Value and spacelimit 220
Figure 4-85. center With Value and spacelimit 220
Figure 4-86. center Within a Notch Structure 220
Figure 4-87. extension45 Illustrations 221
Figure 4-88. strictcenter 222
Figure 4-89. recthv Identical-Pitch SRAF Placement 224
Figure 4-90. recthv Differing-Pitch SRAF Placement. 224
Figure 4-91. recthv Single Axis SRAF Placement 225
Figure 4-92. srafring Example 227
Figure 4-93. srafring Using ondiagonal 227
Figure 4-94. srafringhv Example 229
Figure 4-95. srafringhv Using Multiple Square SRAFs 229
Figure 4-96. Using centerwidth with mbsraf 234
Figure 4-97. Seed Pixel Collection 236
Figure 4-98. Some cornerpref Options and Results. 239
Figure 4-99. cornerprefnegative SRAFs From the Same Seed Pixel 241
Figure 4-100. Default Usage of intersection J 250
Figure 4-101. minjogwidth Modifies intersection J Output 250
Figure 4-102. Usage of minwidthoverlap and minwidthfree Parameters 251
Figure 4-103. Intersection L 252
Figure 4-104. Intersection P. 252
Figure 4-105. Intersection T 253
Figure 4-106. Intersection 45. 253
Figure 4-107. Intersection C 253
Figure 4-108. C-Segment SRAF 254
Figure 4-109. C-Segment Width Mismatch. 254
Figure 4-110. Disabling Negative SRAF. 257
Figure 4-111. intersectionnegative J and minjogwidth Interaction 258
Figure 4-112. intersectionnegative J With minjogwidth 258
Figure 4-113. With and Without jogignore 260
Figure 4-114. Negative SRAF 262
Figure 4-115. Positive and Negative SRAFs 263
Figure 4-116. Negative and Positive SRAFs in a Common File 264
Figure 4-117. SRAF Placed in the Center of the Conflict Region 272
Figure 4-118. SRAF Placed Along the Length of the Conflict Region. 272
Figure 4-119. Conflict Region Removed 273
Figure 4-120. conflictmode Violates MRC 275
Figure 4-121. conflictmode shift 276
Figure 4-122. Differences between conflictmode bbox and conflictmode merge. 277
Figure 4-123. Conflict Region Discarded by maxconflict. 279
Figure 4-124. Valid Conflict Region 279
Figure 4-125. oversizesraf Usage 281
Figure 4-126. centermerge Field SRAF Output. 286

Figure 4-127. centermerge Post-Cleanup Output 286
Figure 4-128. centermerge Disabled 287
Figure 4-129. Using cleanmode With recthv. 289
Figure 4-130. Differences Between Old and New cleanupversion Modes 293
Figure 4-131. Using dynamicsizing to Split SRAFs 296
Figure 4-132. Using dynamicsizing to Resize SRAFs 296
Figure 4-133. Ensuring Proper Sizing of Square SRAFs with dynamicsizing 297
Figure 4-134. dynamicsizingrunlength Disabled. 300
Figure 4-135. dynamicsizingrunlength Enabled 300
Figure 4-136. dynamicsizingrunlength and Negative SRAF. 300
Figure 4-137. interpriorityalign 303
Figure 4-138. intrapriorityalign 306
Figure 4-139. intrapriorityalign With spacerule 307
Figure 4-140. Using lineendmerge With rectedge. 309
Figure 4-141. lineendmerge Parameters and Their Behavior 310
Figure 4-142. reducedbarsplit Modes 313
Figure 5-1. Hybrid SRAF Generation 324
Figure 5-2. SRAF Coverage Varies by peaktype Mode 338
Figure 5-3. Indexed Templates by Priority 344
Figure 5-4. SRAF Generation With recthv and rectdiag 344
Figure 5-5. SRAF Results With recthv and rectdiag 344
Figure 5-6. SRAF Generation With srafringhv 345
Figure 5-7. SRAF Results Using srafringhv Alone 345
Figure C-1. RFT Window 361
Figure C-2. RFT Applications 362
Figure C-3. nmOPC Session 363
Figure C-4. Run nmOPC 363
Figure C-5. OPC Output on Design. 364
Figure C-6. Extraction Wizard. 365
Figure C-7. mbsraf Session 366
Figure C-8. RET Flow Tool For Running mbsraf 366
Figure C-9. SRAF Output 367
Figure C-10. mbsraf Options 368
Figure C-11. Example Output From OPC Run Options for cnSRAF 369
Figure C-12. Layers Generated by the mbsraf Run 370
Figure C-13. Edit Map/Contour Form 372
Figure C-14. SDF and Gradient Maps 373
Figure C-15. Drawing a Cutline 374
Figure C-16. Aerial Intensity Graph 375

List of Tables

Table 1-1. Syntax Conventions 15
Table 4-1. SVRF Commands 31
Table 4-2. LITHO FILE Statement and Commands 39
Table 4-3. sraf_options Keyword and Template Groups 87
Table 4-4. Hardware Requirements and tilemicrons Settings 93
Table 4-5. LITHO FILE Statements for EUV 94
Table 4-6. setlayer sraf_options Statement Keyword Groups 95
Table 4-7. Layer Keywords 96
Table 4-8. Tag Keywords 104
Table 4-9. Constraint Keywords 128
Table 4-10. Templates 173
Table 4-11. Style Keywords 230
Table 4-12. Candidate Generation and Cleanup by Option 238
Table 4-13. Conflict Keywords 270
Table 4-14. Conflict Region Processing 271
Table 4-15. Cleanup Keywords 284
Table 4-16. Priority Keywords 314
Table 5-1. Calibre nmSRAF Best Practices 328
Table 5-2. Calibre mbSRAF Recommended Optimization Settings 335
Table 5-3. Recommended Settings for Best cnsraf Results 346
Table 5-4. Recommended Settings for Best mbsraf Results 348
Table 5-5. Calibre nmSRAF LINT Messages 349
Table A-1. Cleanup Steps, cleanupversion Impact and Associated Debug Layers 354
Table C-1. Usage of the Text nmOPC RET Flow Tool 360

Chapter 1
Introduction to Calibre nmSRAF

Calibre® nmSRAF™ uses a template-defined method to insert sub-resolution assist features into a design. Calibre nmSRAF includes dimension- and model-based SRAF generation tools. Sub-Resolution Assist Features (SRAFs) are slivers of chrome strategically placed on the mask to sharpen printable images. The generated SRAFs remain too small to print.

Calibre nmSRAF is used for advanced nodes where stronger off-axis sources can induce proximity effects that are difficult to compensate for with Calibre® OPCsbar™. Calibre nmSRAF and Calibre OPCsbar can be used in tandem as required for design and illumination.

The template-based approach used in Calibre nmSRAF provides the flexibility to capture the shape requirements of Calibre® pxOPC™, an inverse lithography SRAF insertion engine. This flexibility of shape definition, along with the speed of template application, brings the lithographic power of inverse lithography to the manufacturing flow in a cost effective manner.

You combine Calibre nmSRAF with SRAF optimization tools available in Calibre® nmOPC™ to tune the SRAF to each local environment in a design. These optimizations use many of the same placement and sizing algorithms as Calibre pxOPC. The final SRAF placement and sizing are as effective as an initial Calibre pxOPC insertion, but in less time and with complete mask rule check (MRC) control included.

Calibre nmSRAF Workflow 13
Calibre nmSRAF Key Concepts 14
Calibre nmSRAF Requirements 14
Syntax Conventions 15

Calibre nmSRAF Workflow

The Calibre nmSRAF workflow enables successful generation of SRAFs.

1. Begin with an isolated target shape. If the target layer contains a variety of shapes, develop initial templates for each shape independently.
2. Run Calibre pxOPC on each isolated shape.
3. Create the template(s) for duplicating the Calibre pxOPC shapes of the inner ring (those SRAFs closer to the target). The templates can be created manually and optimized using the SVRF Optimizer tool.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

4. Next, create the templates for duplicating the Calibre pxOPC shapes of the next outer SRAF ring.
5. Run Calibre pxOPC on other training layouts (pitch patterns, real layout clips, targeted test patterns) and add templates as needed. Alternatively, run Calibre pxOPC to improve process metrics such as PV band.

   A PV band polygon shows how edge locations respond to process variations. It represents the area within which a feature prints as the process conditions vary. Generating PV bands involves simulating the printed image at various process conditions and combining the resulting printed images. PV band widths alter due to process variation.

Calibre nmSRAF Key Concepts

Calibre nmSRAF uses dimensional templates that are defined through specification of spacing, width and length parameters, among others.

The templates can be created manually. A template defines the resulting SRAF output with regard to its general structure of SRAF placement. Templates are additive and ordered, meaning that if a previous template does not fill the space, a subsequent template is inserted.

Assign a priority to these templates to indicate their processing significance. Keywords specifying dimensional and processing constraints can be set globally and locally in most cases. Keywords set outside any command context are global and those set within a command context are local. Local assignment of any keyword takes precedence over global assignment. Keywords set globally after local assignment or priority assignment result in a compilation error.

Calibre nmSRAF Requirements

Calibre nmSRAF requires executable paths, licenses and variables to be set prior to running the tools.

Before invoking the Calibre hierarchical engine, you must set your environment variables. Calibre tools require the CALIBRE_HOME environment variable to be set. See “Setting the CALIBRE_HOME Environment Variable” in the *Calibre Administrator’s Guide* for details.

If you intend to implement Calibre nmSRAF, you should be familiar with Calibre® nmDRC™, Calibre nmOPC, Calibre® RVE™, and Calibre® OPCverify™ procedures. An understanding of programming SVRF and TVF rule files is highly useful. To use Calibre nmSRAF, you must have the following items:

- Calibre version 2014.1 or higher

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- The Calibre nmSRAF license. For more information on licensing, refer to the *Calibre Administrator's Guide*.
- An SVRF file

  This rule file contains specification and operation statements with verification rules that are applied to your designs.
- A LITHO FILE

  The LITHO FILE statement contains commands used to configure Calibre nmSRAF for simulation. The LITHO FILE statement can be a separate file or it can be written within an SVRF file.
- A GDS or OASIS®[1]layout
- Model information
  - Optical model
  - Resist model

For help executing Calibre nmSRAF, refer to "Running Calibre nmSRAF" in Appendix A.

Syntax Conventions

The command descriptions use font properties and several metacharacters to document the command syntax.

Table 1-1. Syntax Conventions

Convention	Description
Bold	Bold fonts indicate a required item.
Italic	Italic fonts indicate a user-supplied argument.
`Monospace`	Monospace fonts indicate a shell command, line of code, or URL. A bold monospace font identifies text you enter.
Underline	Underlining indicates either the default argument or the default value of an argument.
UPPercase	For certain case-insensitive commands, uppercase indicates the minimum keyword characters. In most cases, you may omit the lowercase letters and abbreviate the keyword.
[ ]	Brackets enclose optional arguments. Do not include the brackets when entering the command unless they are quoted.

1. OASIS® is a registered trademark of Thomas Grebinski and licensed for use to SEMI®, San Jose. SEMI® is a registered trademark of Semiconductor Equipment and Materials International.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 1-1. Syntax Conventions (cont.)

{ }	Braces enclose arguments to show grouping. Do not include the braces when entering the command unless they are quoted.
‘ ’	Quotes enclose metacharacters that are to be entered literally. Do not include single quotes when entering braces or brackets in a command.
| or ||	Vertical bars indicate a choice between items. Do not include the bars when entering the command.
…	Three dots (an ellipsis) follows an argument or group of arguments that may appear more than once. Do not include the ellipsis when entering the command.

Example:

DEVice {***element_name*** [‘(’*model_name*‘)’]}
device_layer {***pin_layer*** [‘(’*pin_name*‘)’] …}
[‘<’*auxiliary_layer*‘>’ …]
[‘(’*swap_list*‘)’ …]
[BY NET | BY SHAPE]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 2
Template Methodology for Applying SRAFs

Calibre nmSRAF uses dimension-based and model-based methodologies for cnsraf and mbsraf, respectively.

Calibre® cnSRAF™ defines the placement of SRAFs around features by dimension, and the template defines the geometry and placement of the SRAFs. Calibre® mbSRAF™ defines placement of SRAFs by field analysis adhering to lithographic models.

You can use Calibre OPCsbar and Calibre cnSRAF together. In Calibre OPCsbar, rules define SRAF placement by classifying edges by their width and distance to other features. An SRAF of specified geometry is placed at an offset from each edge in every classification. To blend scattering bar and SRAF generation you use Calibre OPCsbar and Calibre cnSRAF together, respectively. Refer to "Calibre cnSRAF and Calibre OPCsbar Integration" on page 23 for more information.

The proper template specification is dependent upon the optical configuration of the lithographic system. Asymmetric optical source configurations typically require asymmetrical Calibre nmSRAF templates. Resolution enhancement processes ensure the most effective arrangement of SRAFs for all isolated vias are consistent. Templates are repeatedly applied to each of the features in a layout. However, vias are rarely isolated and SRAF placement must be determined by applying additional prioritized templates, SRAF mask rule checks (MRC), and conflict resolution schemes.

Calibre cnSRAF Template Execution and Cleanup 17
Calibre cnSRAF Conflict Regions 20
Calibre cnSRAF and Calibre OPCsbar Integration 23

Calibre cnSRAF Template Execution and Cleanup

A template is a specified arrangement of SRAFs around a contact feature based upon a pitch or distance from edges and corners.

Siemens EDA recommends you define Calibre cnSRAF templates by using Calibre pxOPC for:

- Model-based template generation with standard Thin Mask Approximation (TMA) models
- 3D Mask models such as Domain Decomposition Method (DDM)

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Hybrid Hopkins Abbe (HHA) models.

A model-based approach relies on SRAFs generated by a model-based solution. Defining templates requires model-based solutions on a number of selected contact configurations that represent the best cross section of possible configurations. It is important to take asymmetries in the optical configuration into account. Templates are defined such that they result in the best match of SRAFs to the model results.

For either thin mask or 3D mask models, Calibre pxOPC can be used to generate reference SRAFs for Calibre nmSRAF template generation. For more information see the *Calibre pxOPC User's and Reference Manual*.

Alternatively, Calibre cnSRAF templates can be extracted from an optimization procedure. In this methodology, the rules generating template SRAFs are changed algorithmically to minimize the process variation metric.

Calibre cnSRAF Template Execution

Calibre cnSRAF sequentially applies SRAF templates according to assigned priorities. Templates are applied to the layout in order of assigned priority. Each template has a different priority that controls how SRAFs are kept, cleaned, or modified during application. A priority of 1 has the highest priority (and the greatest influence) and is applied first, then a priority of 2 is applied, and so on. As SRAFs generated by subsequent templates are cleaned, only currently processed SRAFs that fit with the previously cleaned SRAFs remain.

After each template application, SRAFs that form conflicts or fail to fit are removed. Applying successive templates can fill the empty conflict regions. Defining the behavior of conflict regions is important to control the transition from one template to the next.

Exclusion Zones

Exclusion zones apply between SRAFs and target shapes, and between SRAFs with different priorities.

Typically, the templates must be defined for various densities. As the spacing between target shapes decreases, the placement of SRAFs may violate spacing rules with another shape. For example, the SRAF might be placed on a shape. In this scenario, the SRAF is considered to be within an exclusion zone defined by the geometry of the target and the manufacturing rule constraints (MRC). To ensure that the SRAFs are clean and do not violate MRC rules, they are automatically excluded from these areas.

Calibre mbSRAF Template Execution

Calibre mbSRAF applies SRAF templates with the same prioritization as Calibre cnSRAF.

Score-Based Cleanup

Calibre mbSRAF cleans up SRAFs by their score. The score is calculated as the integral of the SRAF area over the underlying gradient, and the best match has the highest score. Every SRAF

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

candidate receives a score and the highest score is placed with highest priority. The placement and cleanup of all subsequent SRAF candidates are performed in descending order, until all candidates have been either placed or discarded. All MRC restrictions are maintained while operating on each SRAF candidate. Some candidates are eliminated because of their failure to comply to MRC constraints.

Template Methodology Implementations

Two general methods of template implementation are currently supported:

- **Multiple localmaxgradient templates —** This method requires each localmaxgradient template to specify its own width keyword, without using the maxoffset keyword. This methodology is especially recommended for line layers:

```
localmaxgradient width 0.016 squarewidth 0.023 prior 1
localmaxgradient width 0.020 squarewidth 0.025 prior 2
...
```

- **Single localmaxgradient template —** This method requires one localmaxgradient template to specify a width keyword along with the maxoffset keyword:

```
localmaxgradient width 0.018 squarewidth 0.024
...
maxoffset 0.120
```

Multiple Template Interaction With maxoffset

Using multiple localmaxgradient templates, each template is specified with one maximum width for its generated SRAF (the localmaxgradient width parameter). Multiple localmaxgradient templates allow generation of multiple SRAF widths. The SRAFs are usually generated in a pattern of smaller widths placed closer to the target. SRAFs of increasingly larger widths are placed farther from the target. This pattern minimizes the risk of SRAF printing.

During multiple template use, each template places its SRAFs on the next available seed pixel radiating outward from the target shape. The maxoffset keyword specifies a radial limit for SRAF placement. All templates attempt to place SRAFs within the specified maxoffset value.

If multiple templates are used with maxoffset, the first template places SRAFs at the highest priority over all existing seed pixels outward until maxoffset is met, then the template with the next highest priority performs its placement, continuing until placement is completed by the lowest priority template. In most cases, the first template covers greater than 90 percent of all seed pixels. Subsequent attempts to place SRAFs of differing widths by lower priority templates fail.

Note

Although multiple templates may work jointly with maxoffset with specific recipes, Siemens EDA recommends against using the maxoffset keyword with multiple localmaxgradient templates requiring placement of varying SRAF widths.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Specify maxoffset when using a single localmaxgradient template. The maxoffset keyword determines the maximum distance from the target shape that SRAFs are generated. The number of SRAF rows to be generated is unknown at compile time.

Example 2-1. Use maxoffset With a Single Template

```
sraf_options {
   ...
   algorithm 1
   maxoffset 0.250
   localmaxgradient width 0.045 prior 1
}
```

You should not specify maxoffset when using multiple localmaxgradient templates. The templates determine the widths and distances from the target shape from which SRAF rows are generated.

Example 2-2. Do Not Use maxoffset With Multiple Templates

```
sraf_options {
   ...
   algorithm 1
   localmaxgradient width 0.050 prior 1
   localmaxgradient width 0.045 prior 2
   localmaxgradient width 0.040 prior 3
}
```

Related Topics

sraf_options

Calibre cnSRAF Conflict Regions

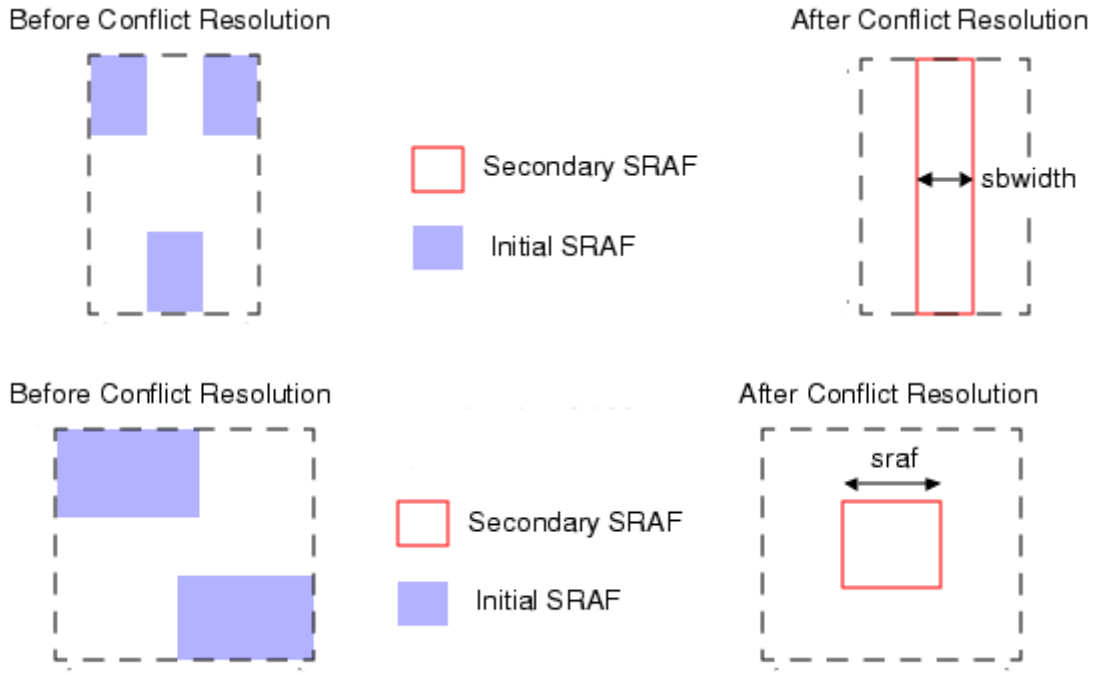
Conflict regions are processed by cnsraf to ensure adherence to MRC requirements and to enable resolution of SRAFs within conflict regions.

Conflict regions are formed by analysis of SRAFs that violate MRC rules. Such a region is depicted in the bounding box in Figure 2-2 where the spacing of generated SRAFs is less than the required MRC constraint.

A conflict is defined as a region containing two or more SRAFs of the same priority that violate MRC rules. The formation of conflict regions may sometimes be limited by the precision of the output database. SRAFs may be generated with different dimensions than specified in the recipe because of grid snapping, affecting dimensions of sbwidth (scattering bar width) and sraf (square SRAF size). Examples of typical conflict regions are shown in Figure 2-1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 2-1. Typical Conflict Regions



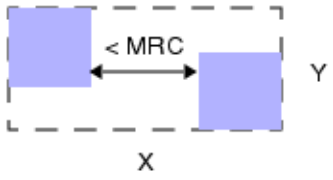
Rather than remove these SRAFs because of exclusion zone violations, conflict regions provide another means to process SRAF placement. The conflict regions are explicitly defined by the bounding box of all contributing SRAFs. Nearby conflict regions are merged, which may result in large areas or complex, non-rectangular areas.

Conflict regions are, by definition, larger than any contributing SRAF because overlapping SRAFs are not considered to be conflicting. Regions have processing options that provide flexibility over how SRAFs are placed. These processing options control SRAF cleanup, size, orientation, and position. Conflict regions can be ignored if the size constraint or the rectangle requirement is violated. If either of these constraints is violated, no SRAFs are placed within the area and the conflict region is discarded.

Note

Templates using the dynamicsizing keyword do not form conflict regions.

Figure 2-2. Conflict Region



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

SRAFs placed in conflict regions are removed allowing lower priority SRAFs to take precedence in those areas. Conflict regions, however, are either filled by a rectangular SRAF or are completely removed. The following regions are removed for these cases:

- Regions that are non-rectangular (as a result of merged conflict regions)
- Conflict regions that violate the MRC rules (to ensure that subsequent SRAFs generation conform to the rules)
- Conflict regions whose width or height exceed the specified maximum edge length (maxconflict)

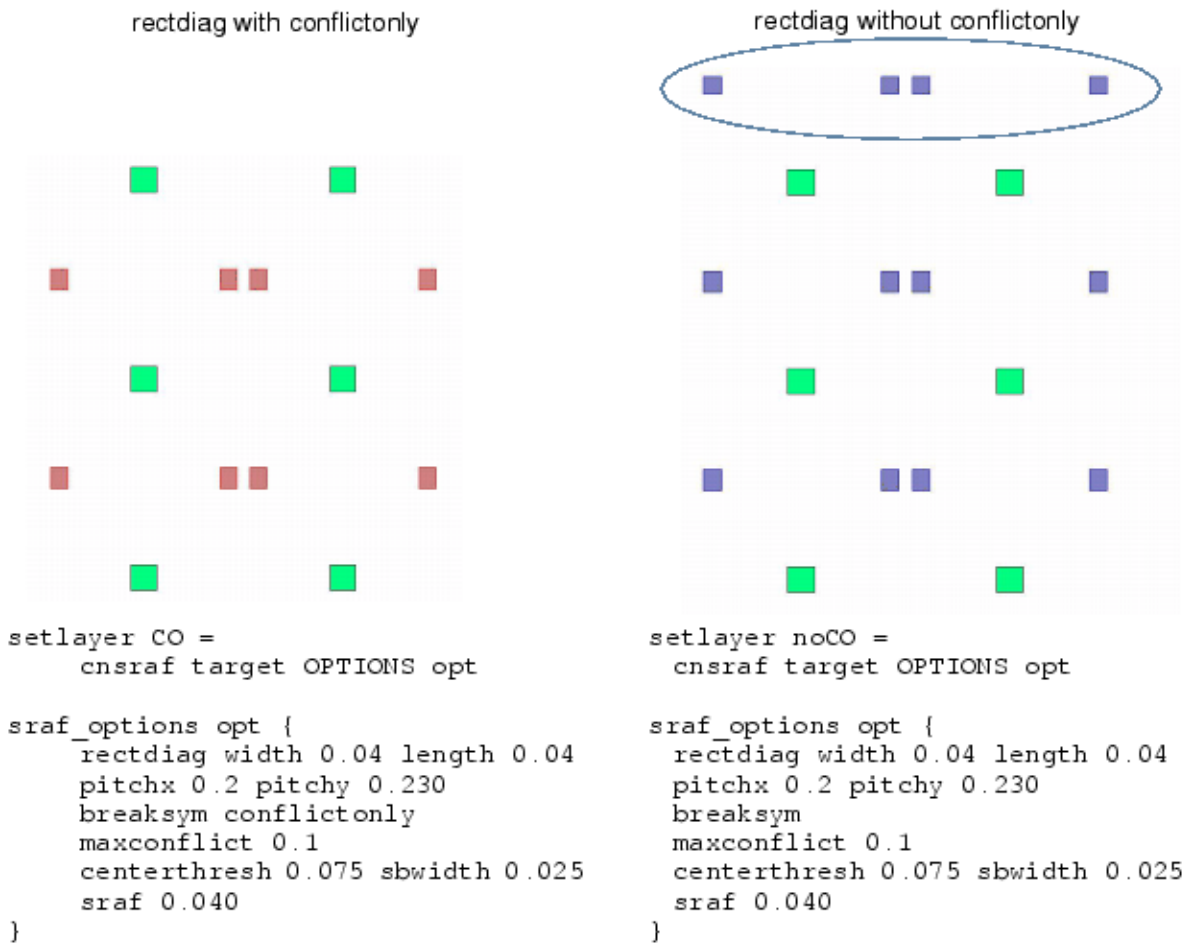
When you specify conflictonly within a template, SRAFs are placed if the target shapes are close enough such that conflict regions are formed by SRAFs of the same priority. This is a convenient option when SRAFs in densely-packed regions form higher priority templates than isolated regions. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using conflictonly only when conflict resolution shapes are relatively small and not using it for larger shapes such as line layer designs.

Using centerthresh specifies a threshold for the length and width of a conflict region determined by cnsraf. This keyword consolidates initial SRAFs that are too close together. For more information, see the centerthresh keyword.

Figure 2-3 shows a rectdiag template with and without the conflictonly argument.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 2-3. rectdiag With and Without conflictonly



The conflictonly argument permits the template to generate SRAFs only if they form conflicts with SRAFs of the same priority. The left side of Figure 2-3 uses conflictonly and does not have any such conflicts above the green target shapes, so the red SRAFs are not generated in that space. However, the right side of Figure 2-3 does not use conflictonly, so the blue SRAFs circled are generated in the upper portion of the illustration.

Further processing of the conflict region depends on the X and Y extents and the centerthresh setting. See centerthresh for more information.

Calibre cnSRAF and Calibre OPCsbar Integration

Calibre OPCsbar and Calibre cnSRAF can be used together. You can use output from Calibre OPCsbar as input to Calibre cnSRAF or conversely, Calibre cnSRAF as input to Calibre OPCsbar.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

To pass the output from Calibre OPCsbar into Calibre cnSRAF, you use the layer command to import SRAF data into Calibre cnSRAF. To pass the output from Calibre cnSRAF into Calibre OPCsbar, you use the SBLAYER command to import the SRAF data into Calibre OPCsbar. Refer to the OPCSBAR command in the *Calibre OPCsbar User's and Reference Manual* and setlayer cnsraf in this manual for more information.

The following example uses Calibre OPCsbar first and then passes its output into Calibre cnSRAF. This example only shows the relevant setup instructions and not the entire recipe. Refer to the *Calibre OPCsbar User's and Reference Manual* for more information.

```
// Create SRAFs using OPCSBAR.
// The output (assist_sbar) is used in setlayer cnsraf.

assist_sbar = OPCSBAR target

assist_CN = RET SBAR V1_target assist_sbar FILE cn_setup MAP assist_CN

LITHO FILE cn_setup [

   layer V1_target
   layer assist_sbar

   // Use output of OPCSBAR (assist_sbar) as sblayer input to cnsraf.

   setlayer assist_CN = cnsraf V1_target sblayer assist_sbar newtag x1 ...

   // output layers

   setlayer 1_field     = cnsraf debug assist_CN field        1
   setlayer 1_hard      = cnsraf debug assist_CN hardconflict 1
   setlayer 1_conflict  = cnsraf debug assist_CN conflict     1
   setlayer 1_iso       = cnsraf debug assist_CN isolate      1
   setlayer 1_processed = cnsraf debug assist_CN resolution   1

]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 3
Considerations for Writing Calibre nmSRAF Rules

Careful planning of the Calibre nmSRAF command flow ensures quality SRAF placement.

Requirements and Assumptions 25
Layout Considerations 25
Template Recipe Priority 26
Calibre pxOPC Generation. 27

Requirements and Assumptions

Specific tools and data are required for successful nmSRAF generation.

- **Optical models —** Required for running Calibre pxOPC and defining SRAFs.
- **Layout specifics —** Feature size, presence of multiple feature sizes, feature shapes, and typical feature pitches are required. An actual layout is not required.
- **MRC rules —** Required to generate MRC-clean SRAFs.
- **Calibre® WORKbench™ —** Use to create test suites with an array of varying pitches and offsets. Calibre WORKbench also measures shapes and displays processing results. The Calibre® RVE™ tool coupled with Calibre OPCverify is useful for OPC validation.
- **OPC rules —** Used to prepare layouts and enable validation of the original target.
- **PV band and area ratio calculation rules —** Used as validation and performance indicators.

Layout Considerations

The type of layout dictates the optimal Calibre nmSRAF template.

Template Applications

Apply pitch-based templates to square features only, and apply edge-based templates to all features meeting the edge length constraint on at least one edge.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Line-end Descriptions

Line-end criteria are specified by the minfeaturespace lineend keyword. Line-ends can be varied depending on their user definition. The line-end length, line-end leg length and adjacent vertices all become criteria for the selection of line-ends for SRAF generation. For more information, see the minfeaturespace keyword.

Template Recipe Priority

Calibre nmSRAF relies on proper ordering of templates within SRAF generation recipes.

Templates should be prioritized from regions closer to the target shape to regions farther from the target shape, having higher priorities through lower priorities, respectively.

Region Definitions

Isolated features are those placed at a distance greater from other features than any considered optical proximity where feature resolution would suffer through the optical interference of a neighboring shape. Dense features are those which interact with multiple features within their optical proximity.

Dense Regions

Dense SRAF regions are defined by spatial frequencies between features where SRAF directly overlaps the SRAFs of a neighboring shape. This is the starting point for determining the range of a conflict region for a template when the conflict resolution keywords are used. See “Calibre cnSRAF Conflict Regions” on page 20 for more information.

Set the value of the maxconflict and centerthresh parameters for this template to be large enough to allow conflict resolution to occur for a wide range of conflicts:

maxconflict = centerthresh

Where:

centerthresh <= SRAF_length

Alternatively as:

*centerthresh <= SRAF_width*2+MRC*

Use Calibre pxOPC to determine the valid conflict resolution range for the template by adjusting the target shape separation and checking that the primary feature remains covered by the Calibre nmSRAF output.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The target shape separation where conflict regions exist and generated SRAFs are no longer valid indicates that a new template is required. The point at which the first template is no longer valid defines the maxconflict parameter value (the point at which the SRAFs are removed).

Use Calibre pxOPC for horizontal, vertical, and off-axis features. Determine the centerthresh values for each template. Begin moving target shape positions laterally and determine applicable ranges of replacement templates.

Without Calibre pxOPC, you review the best PV band (or other chosen metric) for the conflict regions of interest.

Isolated Regions

Because no optically competing shapes cause optical interference with isolated shapes, isolated shapes provide solitary test vehicles for generic SRAF generation. They should remain as a lower priority during SRAF generation.

Calibre pxOPC Generation

The Calibre pxOPC SRAF results can facilitate the generation of Calibre nmSRAF templates. In simple cases, this is accomplished by matching Calibre pxOPC SRAFs with Calibre nmSRAF templates.

Start by generating Calibre pxOPC results for an isolated contact, identifying the SRAF configurations, and defining Calibre nmSRAF templates that generate matching SRAFs. As the process of defining templates continues, Calibre pxOPC can be used to capture the best SRAF placement.

3D Mask models are supported by Calibre pxOPC for Calibre nmSRAF template generation. To aid you with Calibre pxOPC setup, an application note is available on Support Center, entitled “Using pxOPC for generating reference SRAFs in the model-based nmSRAF Template Generation Flow when using 3D Mask Models” (application note number MG587480).

For more information on using Calibre pxOPC, see the *Calibre pxOPC User’s and Reference Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 4
Calibre nmSRAF Command Reference

You define Calibre nmSRAF dimensional- or model-based templates to insert sub-resolution assist features (SRAFs) into the layout to achieve the best process window.

The placement of SRAFs depends on how you specify their constraints, models, existing layout topology, and cleanup methods.

SVRF Commands **31**
RET [EUV] MBSRAF 32
RET SBAR 36

LITHO FILE Statement and Commands **39**
LITHO FILE 41
background 45
clone_transformed_cells 47
ddm 48
ddm_model_load 49
flare_longrange 51
image_options 53
layer 55
litho_model 57
modelpath 60
optical 61
optical_model_load 62
plane 63
processing_mode 64
progress_meter 65
resist | threshold 66
resist_model_load 67
setlayer cnsraf 68
setlayer cnsraf debug 69
setlayer mbsraf 75
setlayer mbsraf debug 77
setlayer tilegen 79
sraf_options 80
svrf_var_import 91
tilemicrons 93

LITHO FILE Statements for EUV **94**

setlayer sraf_options Keywords **95**
Layer Keywords 96
Tag Keywords 103

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Constraint Keywords 128
Templates 173
Style Keywords. 230
Conflict Keywords 270
Cleanup Keywords 284
Priority Keywords. 314

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

SVRF Commands

Calibre nmSRAF commands adhere to SVRF standards. A typical SVRF rule file containing cnsraf-related commands may also contain other SVRF commands that specify input layers, perform preliminary layer manipulation, and save output layers.

These commands are documented in the *Standard Verification Rule Format (SVRF) Manual*.

Table 4-1. SVRF Commands

Command	Description
RET [EUV] MBSRAF	Invokes Calibre mbSRAF.
RET SBAR	Invokes Calibre cnSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET [EUV] MBSRAF

SVRF Commands

Invokes Calibre mbSRAF.

Usage

output_layer = **RET** [EUV] **MBSRAF**
target_layer
[*correction_layer*]
[INSIDE OF LAYER *enclosing_layer*]
[*offset_layer*]
[*sb_layer* [*sb_layer2*]]
[*tag_layer*]
FILE {***filename*** | ***name***}
MAP *name*

Arguments

- ***output_layer***

  A required output layer name. The results of the RET MBSRAF command are written to this layer.

- EUV

  An optional keyword necessary for application of EUV (extreme ultraviolet) lithographic modeling. If EUV models (those with a 13.5 nm wavelength) are referenced, and the EUV keyword is not specified, a compile error results.

- ***target_layer***

  A required input layer consisting of target shapes. The target_layer must be specified as the first argument to RET MBSRAF and only one target_layer may be specified.

- *correction_layer*

  An optional polygon-based layer acting as a corrected target layer and used only to calculate the gradient. Siemens EDA recommends using the correction_layer for best SRAF performance. Only one correction_layer may be specified.

- INSIDE OF LAYER *enclosing_layer*

  An optional keyword specifying a layer of shapes that enclose target shapes for which SRAFs are generated. Generated SRAF shapes intersecting the enclosing_layer are truncated.

- *offset_layer*

  An optional polygon layer containing preexisting polygons that define areas where scattering bars cannot be created. The target layer should never be used as the offset_layer. Only one offset_layer may be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- *sb_layer* [*sb_layer2*]

  An optional polygon layer containing preexisting scattering bars to be factored into calculations when generating additional scattering bars. There is no default for this layer setting. If the sb_layer name is specified once, it may contain both positive and negative SRAFs. Alternatively, the sb_layer may be specified twice with no order dependency, one layer for positive SRAF and one layer for negative SRAF.

- *tag_layer*

  An optional polygon layer specifying a tag name representing imported layers or edges from SVRF. Any number of tag_layers may be specified.

- **FILE** {***filename*** | ***name***}

  A required keyword with one of the following arguments:

  filename — Indicates the path to the LITHO FILE statement contained within an external file. The filename parameter can contain environment variables. For information regarding the use of environment variables in the filename parameter, refer to "Environment Variables in Pathname Parameters" in the *Standard Verification Rule Format (SVRF) Manual*.

  name — The name of a LITHO FILE statement.

- **MAP *name***

  A required keyword and layer pair specifying the name of a layer that is mapped to the RET MBSRAF output layer. The layer name must be generated by the setlayer command within the LITHO FILE statement that is called by RET MBSRAF.

Description

The RET MBSRAF command inputs target layer shapes and the model file specified in the FILE section to output a new layer containing SRAFs. This command can be specified any number of times in the rule file.

RET EUV MBSRAF also supports black border models for simulation.

Note

RET EUV MBSRAF uses the Calibre pxOPC algorithm internally to generate its gradient. As of the 2018.1 release, this capability has been extended to RET MBSRAF for DUV. Known issues with the Calibre pxOPC interface in combination with Fullscale processing mode support are undergoing correction, but do not affect RET EUV MBSRAF integration with Fullscale.

Note

If both dimensional- and model-based templates are used for a given design, Siemens EDA recommends running the dimensional templates with RET SBAR first, and subsequently import the SRAF results to the sb_layer used by RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1

In the following example, the RET MBSRAF command is compiled, and the SVRF rule file is searched for the LITHO FILE statement named mbsraf_LF. The statement name is case-insensitive. The mbsraf_LF statement is used if found within the SVRF file. If not found, mbsraf_LF is searched for as an external file on disk.

The layer order specified within the LITHO FILE statement must match the order as specified in the RET MBSRAF command, and the layer saved by setlayer must appear as the MAP layer as demonstrated in this example:

```
LITHO FILE mbsraf_LF [
   # mbsraf keywords ...
   layer ca_target visible
   layer sb_layer visible
   ...
   sraf_options mbsraf_opts {
      algorithm 1
      width 0.025 ...
   }

   setlayer mbsb = mbsraf ca_target OPTIONS mbsraf_opts ...
]

mbSRAF = RET MBSRAF ca_target sb_layer FILE mbsraf_LF MAP mbsb
```

Example 2

In the following example, the RET EUV MBSRAF command calls a LITHO FILE block named mbsraf_EUV. This LITHO FILE block contains standard and EUV-specific statements:

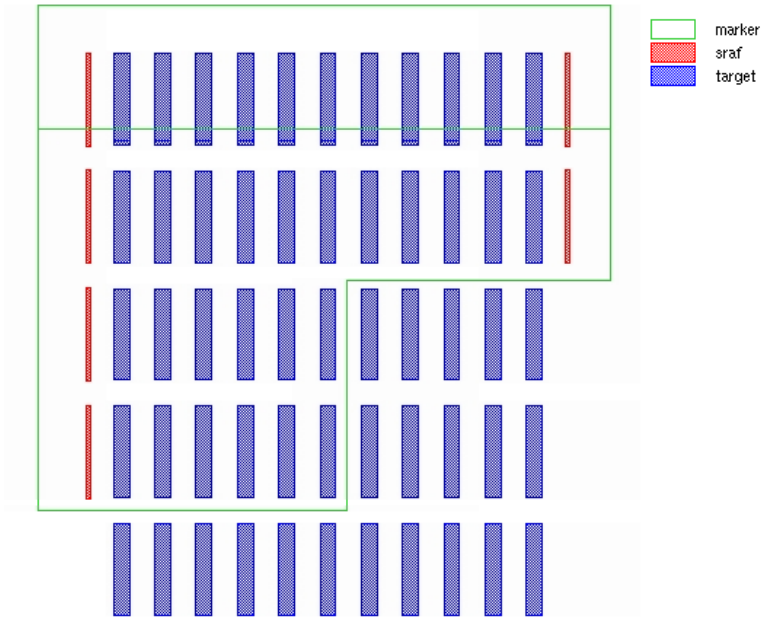
```
LITHO FILE mbsraf_EUV [
   # mbsraf keywords ...
   layer ca_target visible
   layer sb_layer visible
   flare_longrange euv_litho_modelconstant 0.06 png flare_file
   euv_slit_x_center 0.0054
   shadow_bias_model_load sh_mod euv_shadow_14
   ...
   sraf_options mbsraf_opts {
      algorithm 1
      width 0.025 ...
   }

   setlayer mbsb = mbsraf ca_target OPTIONS mbsraf_opts ...
]

mbSRAF_EUV = RET EUV MBSRAF ca_target sb_layer FILE mbsraf_EUV MAP mbsb
```

Example 3 — INSIDE OF LAYER

Specifying INSIDE OF LAYER provides a containment area for SRAF generation as shown in Figure 4-1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-1. INSIDE OF LAYER



sraf = RET SBAR target FILE litho_file MAP sraf INSIDE of LAYER marker

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET SBAR

SVRF Commands

Invokes Calibre cnSRAF.

Usage

output_layer = **RET SBAR**
target_layer
[INSIDE OF LAYER *enclosing_layer*]
[*offset_layer*]
[*sb_layer* [*sb_layer2*]]
[*tag_layer*]
FILE {***filename*** | ***name***}
MAP *name*

Arguments

- ***output_layer***

  A required output layer name. The results of the RET SBAR command are written to this layer.

- ***target_layer***

  A required input layer consisting of target shapes. The target_layer must be specified as the first argument to RET SBAR and only one target_layer may be specified.

- INSIDE OF LAYER *enclosing_layer*

  An optional keyword specifying a layer of shapes that enclose target shapes for which SRAFs are generated. Generated SRAF shapes intersecting the enclosing_layer are truncated.

- *offset_layer*

  An optional polygon layer containing preexisting polygons that define areas where scattering bars cannot be created. The target layer should never be used as the offset_layer. Only one offset_layer may be specified.

- *sb_layer* [*sb_layer2*]

  An optional polygon layer containing preexisting scattering bars to be factored into calculations when generating additional scattering bars. There is no default for this layer setting. If the sb_layer name is specified once, it may contain both positive and negative SRAFs. Alternatively, the sb_layer may be specified twice with no order dependency, one layer for positive SRAF and one layer for negative SRAF.

- *tag_layer*

  An optional polygon layer specifying a tag name representing imported layers or edges from SVRF. Any number of tag_layers may be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **FILE** {***filename*** | ***name***}

  A required keyword with one of the following arguments:

  filename — Indicates the path to the LITHO FILE statement contained within an external file. The filename parameter can contain environment variables. For information regarding the use of environment variables in the filename parameter, refer to "Environment Variables in Pathname Parameters" in the *Standard Verification Rule Format (SVRF) Manual*.

  name — The name of a LITHO FILE statement.

- **MAP *name***

  A required keyword and layer pair specifying the name of a layer that is mapped to the RET SBAR output layer. The layer name must be generated by the setlayer command within the LITHO FILE statement that is called by RET SBAR.

Description

The RET SBAR command inputs a target layer and templates specified in the LITHO FILE statement to output a new layer containing SRAFs. This command can be specified any number of times in the recipe.

Note

If both dimensional- and model-based templates are used for a given design, Siemens EDA recommends running the dimensional templates with RET SBAR first, and subsequently import the SRAF results to the sb_layer used by RET [EUV] MBSRAF.

Examples

Example 1 — LITHO FILE Organization

When the RET SBAR command is compiled, the SVRF rule file is searched for the LITHO FILE statement named cnsraf_LF. This search is case-insensitive. The cnsraf_LF LITHO FILE statement is used if found within the SVRF file. If it is not found, cnsraf_LF is searched for as an external file on disk.

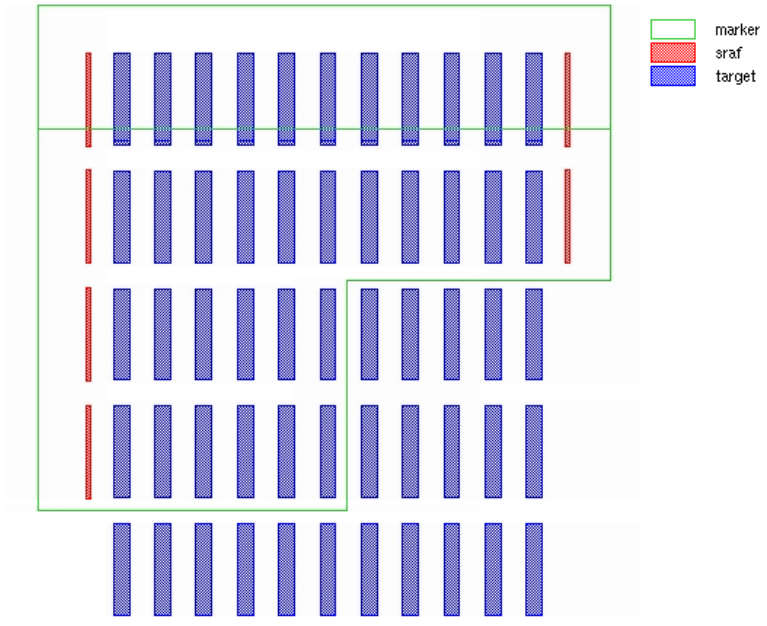
```
LITHO FILE cnsraf_LF [
   // setup commands ...
   layer target

   sraf_options cnsraf_opts {
      cleanupversion 1
      minoffset 0.025 ...
   }
   setlayer out_layer = cnsraf layer1 ...
   ...
]

SB_out = RET SBAR target FILE cnsraf_LF MAP out_layer OPTIONS cnsraf_opts
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 2 — INSIDE OF LAYER

Specifying INSIDE OF LAYER provides a containment area for SRAF generation:

Figure 4-2. INSIDE OF LAYER




```
sraf = RET SBAR target FILE litho_file MAP sraf INSIDE of LAYER marker
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LITHO FILE Statement and Commands

You write commands within the LITHO FILE statement to control setlayer cnsraf and setlayer mbsraf.

The commands shown in Table 4-2 are called by RET SBAR or RET [EUV] MBSRAF and are specified one command per line in the LITHO FILE statement. They do not appear in the sraf_options statement.

Table 4-2. LITHO FILE Statement and Commands

Command	Description
LITHO FILE	Specifies Calibre nmSRAF commands within the SVRF rule file. Used by RET SBAR and RET MBSRAF.
background	Defines background imaging characteristics. Used by RET MBSRAF only.
clone_transformed_cells	Copies cells with nonzero transforms so that simulation of asymmetric optical models is correct. Used by RET MBSRAF only.
ddm	Specifies the DDM model name. Used by RET MBSRAF only.
ddm_model_load	Loads DDM models. Used by RET MBSRAF only.
flare_longrange	Associates a precomputed long range flare convolution with a flare model contained inside a litho model. Used by RET EUV MBSRAF only.
image_options	A LITHO FILE command referencing the *Lithomodel* file and an sraf_options command referencing the litho_model name. Used by RET MBSRAF only.
layer	An input layer called by setlayer for generating SRAFs. Used by RET SBAR and RET MBSRAF.
litho_model	Specifies the directory name that contains files for the litho model format. Used by RET MBSRAF only.
modelpath	Specifies the directories to search for optical and resist models. Used by RET MBSRAF only.
optical	Specifies the optical model name. Used by RET MBSRAF only.
optical_model_load	Specifies an optical model. Alternate to litho model format. Used by RET MBSRAF only.
plane	Specifies the depth of the z-axis within the resist film where the image is computed. Used by RET MBSRAF only.
processing_mode	Enables various processing modes. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 4-2. LITHO FILE Statement and Commands (cont.)

Command	Description
progress_meter	Estimates the progress of the LITHO operation. Used by RET SBAR and RET MBSRAF.
resist | threshold	Specifies the resist model name or aerial threshold value. Used by RET MBSRAF only.
resist_model_load	Specifies a resist model. Alternate to litho model format. Used by RET MBSRAF only.
setlayer cnsraf	Generates SRAFs using dimensional templates. Used by RET SBAR only.
setlayer cnsraf debug	Generates and saves in-process debugging layers. Used by RET SBAR only.
setlayer mbsraf	Generates SRAFs using a model-based methodology. Used by RET MBSRAF only.
setlayer mbsraf debug	Generates and saves in-process debugging layers. Used by RET MBSRAF only.
setlayer tilegen	Creates a layer to display tile and cell boundaries. Used by RET SBAR and RET MBSRAF.
sraf_options	Allows Calibre nmSRAF keywords to be written in free format. The sraf_options statement is called by the setlayer command for SRAF generation. Used by RET SBAR and RET MBSRAF.
svrf_var_import	Imports variables from the SVRF into the LITHO FILE command. Used by RET SBAR and RET MBSRAF.
tilemicrons	Specifies the size of a processing tile. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LITHO FILE

LITHO FILE Statement and Commands

Specifies Calibre nmSRAF commands within the SVRF rule file. Used by RET SBAR and RET MBSRAF.

Usage

LITHO FILE ***name*** **'[**'
commands
']'

Arguments

- ***name***

  A required name allowing commands to be referenced by the RET SBAR or RET [EUV] MBSRAF commands. Each LITHO FILE statement must have a unique name.

- **'[' *commands* ']'**

  A required group of commands placed between brackets ([ ]). Any term on the same line as the brackets is ignored.

  Each command within the LITHO FILE statement must reside on its own line. You may span multiple lines with a line termination (backslash \), which must not be followed by any characters or whitespace.

  Note

  Siemens EDA recommends always using the sraf_options block to format and organize keywords within the LITHO FILE statement.

Description

The LITHO FILE statement contains cnsraf and mbsraf commands. The LITHO FILE statement is called by the FILE argument of the RET SBAR and RET MBSRAF commands. The LITHO FILE statement contents can be contained within the SVRF rule file or referenced as an external file, and in either case, called by a symbolic name for multiple instantiations.

You can use variables for any names or numbers in the SVRF or LITHO FILE statement, as described by the command "svrf_var_import" on page 91.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. These non-litho model commands are unsupported and are obsoleted as of release 2019.1:

- optical
- optical_model_load
- plane
- resist
- resist_model_load
- threshold

Examples

Example 1

The following code demonstrates the general format of the Calibre cnSRAF LITHO FILE statement using the sraf_options statement.

```
LITHO FILE cnsraf_1 [
   layer ref_layer
   sraf_options cnsraf_cmds {
      cleanupversion 1
      rectedge center length 0.025 width 0.015 ... prior 1
      priority 1 minoffset 0.025 minwidth 0.010 ...
   }
   setlayer out_layer = cnsraf ref_layer ... OPTIONS cnsraf_cmds \
      maxconflict size
   ...
]
```

Example 2

The following code demonstrates a Calibre mbSRAF LITHO FILE statement for the litho model format using the sraf_options statement.

```
LITHO FILE mb_setup_1 [
   modelpath ./models
   layer target
   image_options IOS {
      litho_model lithomodel.dir
      layer target visible
   }
   tilemicrons 40 adjust
   sraf_options mbsraf_cmds {
      algorithm 1
      image_options IOS
      width 0.025 localmaxgradient ...
   }
   setlayer mbsb = mbsraf target OPTIONS mbsraf_cmds ...
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 3

The following code demonstrates a Calibre mbSRAF discrete model statement. This format is deprecated. Siemens EDA recommends the litho model format used in Example 2.

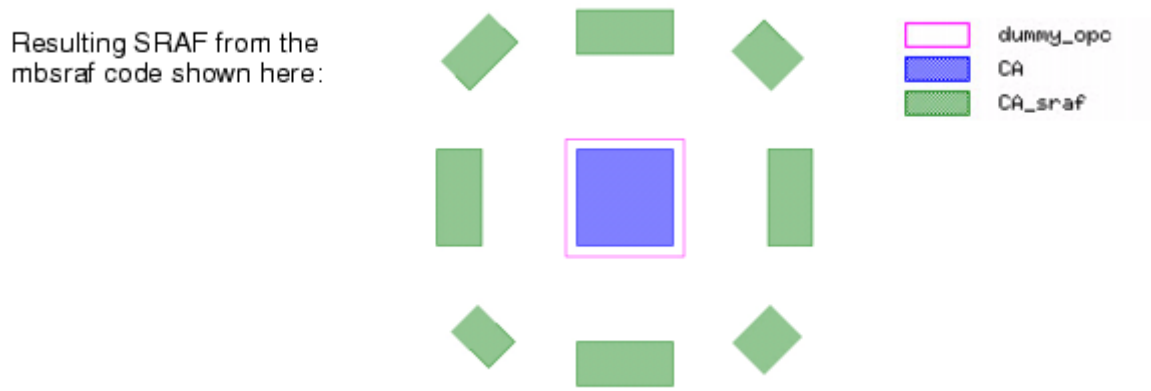
```
LITHO FILE setup_name [
   modelpath models
   optical_model_load op_mod
   resist_model_load res_model

   layer name type trans
   ...
   sraf_options mbsraf_opts {
      algorithm 1
      width 0.025 localmaxgradient ...
   }
   setlayer out_layer = mbsraf target_layer optical model_name \
      OPTIONS mbsraf_opts
]
```

Example 4

The following code demonstrates the use of mode nearest to generate SRAFs that are parallel to target shape edges and oriented 45 degrees at target shape corners.

```
LITHO FILE sample_lithofile [
   modelpath /user/opt/models/model1:/user/opt/models/mbsraf_model
   optical_model_load opticsfile e1_f+0.0000
   resist_model_load resistfile model1.mod
   background 0.0215 -0.0085
   layer CA visible 0.9785 0.0085
   layer dummy_opc visible 0.9785 0.0085
   layer sraf_ref_layer visible 0.9785 0.0085 25

   sraf_options mbOPT_statement {
      correctionlayer dummy_opc
      optical opticsfile
      algorithm 1
      resist resistfile
      minfeaturespace 0.035
      localmaxgradient width 0.030 prior 1
      priority 1 minoffset 0.040

   setlayer sb_sample = mbsraf CA dummy_opc OPTIONS mbOPT_statement
]
mbsraf_logic = RET MBSRAF CA dummy_opc FILE sample_lithofile MAP sb_sample
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.



Example 5

The following code demonstrates the implementation of a multi-patterning application.

```
LITHO FILE mp_mb_setup [
   modelpath ./models
   layer target_0
   layer target_1
   layer target_2
   image_options IOS {
      litho_model mp_litho_dir
      layer target_0 visible mask_layer 0
      layer target_1 visible mask_layer 1
      layer target_2 visible mask_layer 2
   }
   tilemicrons 40 adjust
   sraf_options mbsraf_cmds {
      algorithm 1
      image_options IOS
      width 0.025 localmaxgradient ...
   }
   setlayer mbsb = mbsraf target OPTIONS mbsraf_cmds ...
]
mp_mb_output = RET MBSRAF target_0 target_1 target_2
   FILE mp_mb_setup MAP layer1
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

background

LITHO FILE Statement and Commands

Defines background imaging characteristics. Used by RET MBSRAF only.

Usage

background {**dark** | **clear** | {**atten** ***factor***} | {***real imag***}}

Arguments

- **dark**

  A literal argument designating dark field imaging. Transmission added to the mask is equated as:

  layer transmission + 0

- **clear**

  A literal argument designating clear field imaging. Transmission added to the mask is equated as:

  layer transmission + 1

- **atten *factor***

  A literal and value designating attenuated phase-shifted background. The factor is specified as the normalized intensity that should transfer to the wafer for a clear exposure. The unit is a percentage; for an 8% attenuated phase shift mask, the factor should be 0.08. The maximum allowed value is 0.36 (36%).

  When Calibre LITHO tools encounter an attenuated factor, they translate the factor into a real imaginary pair. The basic principle behind transmission layers can be summarized as:

  actual transmission = background + layer transmission values

  When using the atten keyword to specify background transmission, it uses the following real imaginary pair:

  RE(bg) = -sqrt(percent/100)

  IM(bg) = 0

  For example:

  ```
  background atten 0.06
  ```

  is equivalent to

  ```
  background -0.245 0
  ```

  In this example, the factor of 6% (0.06) with a 180-degree phase-shifted transmission is applied to the background. This corresponds to an electric field (real, imaginary) pair of (-sqrt(0.06), 0) = (-0.245, 0).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

There are no limitations to the layer transmission values when the atten option is used for the background definition. Do not use an attenuated background with a phase180 layer, as this causes a user error.

- ***real imag***

  A pair of real numbers that specifies the layer transmission value and background value, respectively. Note that the atten syntax for the transmission is recommended over this syntax. The real imag syntax supports any arbitrary phase and transmission combination.

  If you specify background with the real imag pair for a specified layer transmission value, the resulting transmission value for the output layer with a real imag pair is resolved relative to the background pair. The layer transmission value is resolved according to the equation:

  background (real imag) + layer (real imag) = transmission_value

  For example:

```
background 1 0
#layer name    type     transmission
layer chrome visible 0 0
```

  The background is specified with a (1, 0) pair, while the layer, chrome, is defined with a transmission value of (0,0). With clear background (1,0) and a layer specified with (0,0), the resulting transmission layer is (1,0) + (0,0) = clear (1,0).

```
background 1 0
#layer name    type     transmission
layer p180   visible -1 0
```

  In this case, the layer is defined as (-1,0). With clear background (1,0) and a layer specified with (-1,0), the resulting transmission layer is:

  (1,0) + (-1,0) = dark (0,0)

Description

A required keyword appearing only once that specifies the background transmission level for each exposure. Background transmission is added to the ideal Kirchoff mask transmission for the layers in order to generate the overall mask transmission.

For more information regarding layer, see background in the *Calibre OPCVerify User's and Reference Manual*.

Examples

This example shows the combination of layer and background settings needed to define an attenuated phase shift mask. The background is attenuated and the layer type is clear because light is not obstructed when passing through the polygons in the VIA layer.

```
background atten 0.08
layer VIA visible clear
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

clone_transformed_cells

LITHO FILE Statement and Commands

Copies cells with nonzero transforms so that simulation of asymmetric optical models is correct. Used by RET MBSRAF only.

Usage

clone_transformed_cells {decide | no | yes}

Arguments

- decide

  The optical model is checked to decide if cell cloning is activated. More specifically, this option causes the Calibre engine to check if any asymmetric sources are present in the optical model. If so, the Calibre database constructor is forced to clone cells that have rotated or mirrored placements, so that hierarchical simulation of the rotated structures is correct. This is the default setting.

  Note

  Siemens EDA recommends setting clone_transformed_cells to decide.

- no

  Override source shape and force it to be considered symmetric (do not clone).

- yes

  Override source shape and force it to be considered asymmetric (force cloning).

Description

When using an asymmetrical source, cell orientation can have a significant impact on results. You can control how the simulations handle cell orientation when calculating the aerial image for a hierarchical design. The application can:

- Calculate the aerial image for the original cell, then adjust the orientation of the image to match each cell.
- Calculate a separate aerial image for each orientation in the cell. To do this, the application creates one clone of the cell for each orientation represented in the design and transforms the clone as necessary to match the design, and calculates a separate aerial image for each clone.

Examples

This example enables optical model checking for cell cloning:

```
clone_transformed_cells decide
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

ddm

LITHO FILE Statement and Commands

Specifies the DDM model name. Used by RET MBSRAF only.

Usage

ddm *model_name*

Arguments

- *model_name*

  The name of the DDM model.

Description

An optional keyword used to call a symbolic model name declared by ddm_model_load.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

ddm_model_load

LITHO FILE Statement and Commands

Loads DDM models. Used by RET MBSRAF only.

Usage

ddm_model_load *name* {*filename* | '{'*inline_modelname*'}'}

Description

Loads the Domain Decomposition Method (DDM) for Hybrid Hopkins Abbe (HHA) models contained in the filename, or defined by inline model text within braces ({ }). The DDM model is assigned the user-defined name, which you may later reference in the image command. Calibre mbSRAF supports crosstalk and corner chopping with DDM models.

The ddm keyword specified in the sraf_options statement calls the declared symbolic model name.

A DDM or HHA model is optional and only required if you use the image command with the ddm modifier. This command is ignored if a litho model is present; the DDM model file is set inside the *Lithomodel* file instead.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Note

For more information on DDM and HHA models, see the *Calibre WORKbench User's and Reference Manual*.

Arguments

- *name*

  An argument specifying the symbolic name of the DDM model referred to by subsequent DDM-related commands.

- *filename* | '{' *inline_modelname*'}'

  A required argument that specifies either a filename containing the DDM model or an inline file defining the DDM model. The inline DDM model definition must be enclosed in braces ({ }). One of the arguments must be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

```
LITHO FILE mbsraf_setup [
   tilemicrons 25
   modelpath ./models
   optical_model_load optical_ddm optical_ddm
   resist_model_load  RESIST cm1.mod
   ddm_model_load DDM_model ddm_model_named.ddm

   background -0.0259 0.0095
   layer target visible -0.9741 -0.0095
   layer dummy_opc visible -0.9741 -0.0095

   sraf_options opts {
      algorithm 1
      correctionLayer dummy_opc
      optical optical_ddm resist RESIST
      ddm DDM_model
      peaktype 3Dir
      ...
   }
   setlayer mb.sraf = mbsraf target OPTIONS opts
]

mb_sraf = RET MBSRAF target dummy_opc MAP mb.sraf FILE mbsraf_setup
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

flare_longrange

LITHO FILE Statement and Commands

Associates a precomputed long range flare convolution with a flare model contained inside a litho model. Used by RET EUV MBSRAF only.

Usage

flare_longrange *litho_model_dir*
{none | constant *value* | png *file_name* constant {average | minimum | maximum}}

Description

The flare_longrange command specifies how intra- and inter-field flare effects are handled. This version of flare_longrange is a reduced-feature set of the command and is tailored for RET EUV MBSRAF.

Arguments

- *litho_model_dir*

  An argument that specifies the name of a litho model directory that contains a flare model with long range flare. The model name must either be found in the modelpath or be a fully qualified path.

- none

  An argument that specifies that no pre-computed long range convolution is used when performing simulations with the specified model.

- constant *value* | png *file_name* constant {average | minimum | maximum}

  An argument set that specifies where the result of a pre-computed convolution of a fractal kernel (long range part) present in the loaded model and the currently processed layout is stored.

  - constant *value* — The long range flare is a constant percentage of clear field transmission given by value, where a value of 1 is equal to 100%. The intensity is calculated as:

    *I_constant_flare = kf * Iclear * value*

    where k_f is the coefficient from the flare model, and I_{clear} is the clear field value taking into account both optics and ddm mask transmission.

  - png *file_name* constant {average | minimum | maximum} — Make sure your Calibre session has access to the *.png* file. This file is generated using RET FLARE CONVOLVE. The file may be prefaced by a full or relative path, or found in the modelpath.

    If the constant parameter is specified, the long range flare is represented as a constant value that is a property of the long range flare map, and the constant flare

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

value is used for the entire simulated region, independent of the original flare map extent. Currently supported properties are:

- average — uses the average value in the flare map
- minimum — uses the smallest value anywhere in the flare map
- maximum — uses the largest value anywhere in the flare map

For more information, see flare_longrange in the *Calibre OPCVerify User's and Reference Manual*.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

image_options

LITHO FILE Statement and Commands and Style Keywords

A LITHO FILE command referencing the *Lithomodel* file and an sraf_options command referencing the litho_model name. Used by RET MBSRAF only.

Usage

Usage within the LITHO FILE statement:

image_options ***name*** '**{**'
  layer ***name*** **visible** [mask_layer *number*] ...
  litho_model ***litho_model_dir*** ...
  [layer *name* filter]
  '**}**'

Usage within the sraf_options statement:

image_options ***name***

Arguments

- ***name***

  Declares a required, unique user-defined name for the image_options statement. This name is also used the image_options statement from within the sraf_options statement or setlayer mbsraf.

- **layer** ***name*** **visible** [*mask_layer number*]

  Specifies a visible layer. At least one visible layer is required. The mask_layer parameter maps this layer to a corresponding mask_layer template inside the *Lithomodel* file. The mask_layer number defaults to 0 if not specified (in the case of single template litho models, for example). For multi-patterning masks, specify a mask_layer argument to additional layer commands.

- **litho_model** ***litho_model_dir***

  A required argument that specifies a directory name. This directory must contain optical and resist models, a *Lithomodel* file, and optionally contain an etch model. For more detail see "Lithomodel Format Usage for Calibre mbSRAF Execution" on page 341.

  This is an abbreviated description of the image_options command. For detailed information, consult the *Calibre OPCverify User's and Reference Manual*.

- layer *name* filter

  An optional filter layer limiting contour generation to the area of the layer. Only one filter layer may be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Description

Specifies a statement name containing a reference to the *Lithomodel* file and subsequently called by the setlayer mbsraf command. The image_options statement within sraf_options must be specified prior to any pw_condition keyword.

Examples

Example 1

The keyword is first defined within the LITHO FILE statement and then called within the sraf_options statement:

```
LITHO FILE M1_model [
   image_options MBS {
      litho_model 20nm_lithomodel
      layer target visible
   }
   ...
   sraf_options mb_OPT {
      algorithm 1
      correctionlayer correction_layer
      image_options MBS
      ...
      localmaxgradient width 0.026 prior 1
      pw_condition f40 focus -40nm dose 1.05 size -0.00025 weight 40
   }
   setlayer mbsb1 = mbsraf target OPTIONS mb_OPT
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

layer

LITHO FILE Statement and Commands

An input layer called by setlayer for generating SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

layer ***layer_name*** [hidden [*transmission*] | visible *transmission*]

Arguments

- ***layer_name*** [hidden [*transmission*] | visible *transmission*]

  A required argument specifying the name of the layer. Calibre nmSRAF uses this name throughout the LITHO FILE statement to refer to this layer.

  Layer names must consist of alpha-numeric characters, and form strings less than 32 characters in length. Any number of layer commands may be specified within a LITHO FILE statement. The layer type and transmission arguments are only used by RET MBSRAF.

  hidden [*transmission*]

  The hidden layer type is applied to all layers that are part of the layout, but not simulated. The transmission argument is optional for the hidden type.

  visible *transmission*

  The visible layer type is applied to all layers that are simulated. The transmission argument is required for the visible type.

Description

A required command that names an input layer. Every layer to be used in either setlayer cnsraf or setlayer mbsraf commands must be declared in the LITHO FILE statement must be declared using the layer command. Layer applications may include target layers, keepout areas (offsetlayer), marker layers (taglayer), preexisting SRAF layers (sblayer), or alternate simulation layers (for example, correctionlayer used by mbSRAF only). Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

Usage for setlayer mbSRAF

The target layer is a required layer for mbSRAF because the shapes provided on the target layer define the evaluation points for the gradient calculation. If a portion of the original target layer is being processed in mbSRAF (typically using mbSRAF as SRAF fill after cnsraf rules-based insertion), the selected region of the target layer must be large enough to provide an empiric topological environment for any subset of target shapes in order for the gradient to be representationally accurate. If a shape is artificially isolated, the gradient can be erroneous due to inadequate sampling of the empiric target environment.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The sblayer and correctionlayer are layers containing shapes that are simulated to provide the image from which the gradient is calculated. If neither of these layers are available, the target layer is used for the image simulation.

When using mbSRAF as fill or an SRAF safety net after rules-based insertion, all layers processed must contain a large enough region centered on the selected shape needing further lithographic validation through an optical diameter, but the generated SRAFs produced by mbSRAF must only be constrained to a smaller halo around the center target (using a value equal to or smaller than maxoffset).

Note

The layer command must appear before the first setlayer command. The order of layer commands in the LITHO FILE statement represents the order the declared layers will be used in RET MBSRAF and RET SBAR calls.

For more information regarding layer, see layer in the *Calibre OPCVerify User's and Reference Manual*.

Examples

The layer command is specified within the LITHO FILE statement:

```
LITHO FILE LF [
   #layer name    type     transmission
   layer M1 visible 0 0
   layer offset_layer hidden
   layer p180 visible -1 0
   layer sb_layer
   ...
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

litho_model

LITHO FILE Statement and Commands

Specifies the directory name that contains files for the litho model format. Used by RET MBSRAF only.

Usage

litho_model *dir_name*

Arguments

- ***dir_name***

  A required argument that specifies the litho model directory name.

Description

A required keyword that specifies the directory containing the *Lithomodel* file for mbsraf. The directory you specify must be located within the directory specified by the modelpath keyword. The litho_model keyword is written within an image_options block.

The directory referenced by the litho_model keyword contains a *Lithomodel* file, optical model directory and resist model file. For more information regarding best practice use of litho model format see "Lithomodel Format Usage for Calibre mbSRAF Execution" on page 341.

Note

Only the litho model format is supported for EUV, including models and all related keywords.

Examples

Example 1

The litho_model keyword and its directory name are shown inside an image_options block within a LITHO FILE statement. The model is declared and called by the image_options keyword:

```
LITHO FILE mbsraf_setup [
   modelpath .
   image_options M1_LMO {
      litho_model ./litho_model_dir
      layer M1 visible mask_layer 0
      layer correction_layer visible mask_layer 0
   }

   layer logicsraf_target
   layer correction_layer
   tilemicrons 25
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
      sraf_options mb_OPT {
         correctionlayer correction_layer
         image_options M1_LMO
         ...
         localmaxgradient width 0.026  prior 1
         priority 1 lineendmerge 0.06
      }

      setlayer mb_sraf = mbsraf logicsraf_target OPTIONS mb_OPT
      setlayer spx = mbsraf debug logicmb_sraf seedpixels 1
   ]

   mbsraf = RET MBSRAF M1 correction_layer MAP mb_sraf FILE mbsraf_setup
   seed_1 = RET MBSRAF M1 correction_layer MAP spx FILE mbsraf_setup
```

Example 2

For an inline litho model application, all model information is written locally within the LITHO FILE statement. The litho_model keyword and its symbolic name are followed by all data normally kept within the *Lithofile* file. The litho model is then referenced from the image_options block with its symbolic name and then subsequently called by the sraf_options block for mbsraf:

```
#ENCRYPT

LITHO FILE inline_LM [/*
   layer target
   layer opc
   modelpath ./LITHO_MODEL
   optical_model_load o1 optical.mod
   ddm_model_load d1 ddm.mod

   resist_model_load r1 {
      version 5
      modelType CM1
      ...
   }

   litho_model LITHO_MODEL {
      version 1
      resist r1
      mask 0 {
         optical o1
         background clear
         mask_layer 0 TRANS atten 0.06 DDM d1
      }
   }

   image_options A {
      layer opc visible
      litho_model LITHO_MODEL
   }
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
   sraf_options sraf_opts {
      image_options A
      correctionlayer opc
      ...
      localmaxgradient width 0.03 squarewidth 0.040 prior 1
   }
   setlayer c1 = mbsraf target OPTIONS sraf_opts
*/]

#ENDCRYPT

mbsraf = RET MBSRAF target correction_layer MAP mb_sraf FILE inline_LM
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

modelpath

LITHO FILE Statement and Commands

Specifies the directories to search for optical and resist models. Used by RET MBSRAF only.

Usage

modelpath ***directory*** [*:directory* ...]

Arguments

- ***directory*** [*:directory* ...]

  A required argument that specifies a colon-separated list of directories to search. The directories can be specified as either absolute or relative paths.

Description

A required keyword that specifies the directories to search for optical and resist models.

Examples

Example 1 — Absolute Path

```
modelpath /export/home/calibre_wrk/models
```

Example 2 — Relative Path

```
modelpath ./models
```

Example 3 — Multiple Paths Require a Colon Delimiter

```
modelpath ./models:../test_models:/usr/bin/opt/models
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

optical

LITHO FILE Statement and Commands

Specifies the optical model name. Used by RET MBSRAF only.

Usage

optical ***name*** [dose *value*] [weight *value*] [size *value*] [defocus *value*]

Arguments

- ***name***

  A required argument that specifies the name of the optical model.
- dose *value*

  An optional keyword that specifies the nominal dose.
- weight *value*

  An optional keyword that specifies the nominal weight of the EPE (edge placement error).
- size *value*

  An optional keyword that specifies the nominal mask size in microns.
- defocus *value*

  An optional keyword that specifies the nominal defocus.

Description

A required keyword that specifies the optical model name and various model attributes.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

optical_model_load

LITHO FILE Statement and Commands

Specifies an optical model. Alternate to litho model format. Used by RET MBSRAF only.

Usage

optical_model_load ***name filename***

Arguments

- ***name***

  A required argument that specifies a symbolic name other commands internal to the SVRF file uses to refer to this optical model. This name is used for either optical models as an in-line model or model file on disk.

- ***filename***

  The optical model filename on disk. If no filename is specified, an in-line optical model must be supplied instead.

Description

This is a required setup file parameter. At least one optical_model_load command must always be present for a non-litho model recipe. Multiple optical models may be loaded and used in the same SVRF run.

This command loads an optical model found in filename. The optical model is given a symbolic name which is used with all commands referencing the model in the simulation.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Examples

Shown is a typical introduction of a LITHO FILE statement that loads an optical model named *./models/opticalf0*:

```
LITHO FILE litho.in [
   modelpath models
   optical_model_load f0 opticalf0
   ...
```

All simulation commands in the example refer to the model by the name f0.

Related Topics

modelpath

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

plane

LITHO FILE Statement and Commands

Specifies the depth of the z-axis within the resist film where the image is computed. Used by RET MBSRAF only.

Usage

plane *value*

Arguments

- *value*

  An optional positive number that specifies the value of the z-axis within the resist film.

Description

An optional argument that specifies the z-axis within the resist film where the image will be computed. The value is relative to 0.0 at the bottom of the resist, and must be greater than or equal to 0.0 and less than or equal to the resist film thickness. If plane is not specified, the base focus plane from the optical model is used.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

processing_mode

LITHO FILE Statement and Commands

Enables various processing modes. Used by RET SBAR and RET MBSRAF.

Usage

processing_mode {flat | hierarchical | hybrid}

Arguments

- flat | hierarchical | hybrid

  Toggles the processing mode.

Description

An optional command that enables three processing modes:

- Flat mode enables litho-flat processing by splitting the design into tiles which are processed separately on remote machines.
- Hierarchical mode processes the design as cells and tiles on remote machines.
- Hybrid mode examines the layout hierarchy to determine on a per-cell basis whether default hierarchical or litho-flat processing would provide the best runtime performance.

For the RET SBAR command, LAYOUT ULTRAflex is invoked automatically when processing_mode flat is specified and only flat processing_mode flat may use LAYOUT ULTRAflex. See the *Calibre Post-Tapeout Flow User's Manual* for more information.

Note

If ULTRAflex mode is already enabled in your SVRF file, Siemens EDA recommends you also set processing_mode in the LITHO FILE statement, for either RET SBAR or RET MBSRAF, to achieve consistent SRAF results.

Examples

```
LITHO FILE cn_setup_1 [
   processing_mode flat
   ...
   sraf_options OPTS {
      cleanupversion 1
      minoffset 0.040 lineend 0.050 maxconflict 0.150
      rectedge tag All_Wire extension 0.020 width 0.03 offset 0.04 prior 1
   ...
   }
   setlayer cnsb1 = cnsraf M1 OPTIONS OPTS
   ...
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

progress_meter

LITHO FILE Statement and Commands

Estimates the progress of the LITHO operation. Used by RET SBAR and RET MBSRAF.

Usage

progress_meter {on | off}

Arguments

- on | off

  Toggles the progress meter status.

Description

Use this command to help estimate the progress of the LITHO operation in which it is used. It prints the completion percentage as tiles are run under the heading TIME FOR ALL CELLS in the console window. The progress_meter command is enabled by default.

This option may add up to 10 percent additional runtime because of the computation algorithm, depending on the size of the job. The cost of performing progress meter checks can be found in the logfile as TIME FOR PROGRESS METER.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

resist | threshold

LITHO FILE Statement and Commands

Specifies the resist model name or aerial threshold value. Used by RET MBSRAF only.

Usage

resist ***model_name*** | **threshold** ***aerial_value***

Arguments

- ***model_name*** | ***aerial_value***

  A required argument that specifies the name of the photoresist model or the aerial resist threshold. The threshold value must be a positive number.

Description

A required keyword that specifies either the name of the photoresist model, or the value of the aerial photoresist threshold. The threshold value can be obtained from the optical model.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

resist_model_load

LITHO FILE Statement and Commands

Specifies a resist model. Alternate to litho model format. Used by RET MBSRAF only.

Usage

resist_model_load *resist_model_name* {*filename* | '{' *inline_modelname* '}'}

Arguments

- *resist_model_name*

  A required argument that specifies a name used by other commands to refer to this resist model.

- *filename* | '{' *inline_modelname* '}'

  A required argument that specifies either a filename containing the resist model or an inline file defining the resist model. The inline resist model definition must be enclosed in braces ({ }). One of the arguments must be specified.

Description

Loads a resist model specified by filename, or defined by inline model keywords. The resist model is assigned a user-defined symbolic name, which you may later reference in the image command. Resist model specification is optional for mbsraf.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. This non-litho model command is not supported and is obsoleted as of release 2019.1.

Examples

This LITHO FILE statement loads a resist model named *./models/poly65.mod*:

```
LITHO FILE litho.in [
   modelpath models
   resist_model_load resistfile poly65.mod
   ...
]
...
```

Subsequent commands must refer to this particular model with the name *resistfile*.

Related Topics

modelpath

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer cnsraf

LITHO FILE Statement and Commands

Generates SRAFs using dimensional templates. Used by RET SBAR only.

Usage

setlayer ***output_layer*** = **cnsraf** ***target_layer*** **OPTIONS** ***statement_name***

Arguments

- ***output_layer***

  A required keyword that specifies the name of the new output target layer. The new layer is created in GDS or OASIS format, and the results of the operation are written to the layer.

- ***target_layer***

  A required input layer that specifies the target layer.

- **OPTIONS** ***statement_name***

  A required keyword and name pair that calls the previously defined sraf_options statement. For more information regarding cnsraf keywords, see "sraf_options" on page 80.

Description

This command is used inside a LITHO FILE statement to generate SRAFs as defined in the sraf_options statement.

Examples

Example 1

These lines execute the setlayer command and reference the sraf_options statement containing all desired options:

```
sraf_options cnsraf_opts {
   cleanupversion 1
   newtag tag1 length > 0.01
   maxconflict 0.150
   rectedge tag tag1 extension 0 width 0.03 offset 0.04 prior 1
   priority 1 minfeaturespace 0.040 minoffset 0.020
   }
setlayer cnsb1 = cnsraf target OPTIONS cnsraf_opts ...
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer cnsraf debug

LITHO FILE Statement and Commands

Generates and saves in-process debugging layers. Used by RET SBAR only.

Usage

setlayer *output_layer* = cnsraf debug *layer debug_keyword priority* [template *index*]

Arguments

- setlayer *output_layer*

  A required command and layer name to which SRAFs are output.

- *layer*

  A required name of the SRAF layer to which all debugging layers apply.

- *debug_keyword*

  You must specify one of the following keywords. They are listed in the order they are processed by cnsraf debug:

  a. tags — Outputs edges of target shapes used to generate SRAFs. These edges are selected using a tagging keyword. These edges highlight those target layer edges contributing to a template during recipe development.

  b. candidates — Outputs SRAFs formed from selected seedpixels prior to selection and placement by priority. Applicable to mbSRAF only.

  c. initialfield — Outputs SRAFs generated by pairtemplate prior to further processing.

  d. field — Outputs all SRAFs by priority prior to cleaning.

  e. lineend — Outputs the line-end keepout areas derived by the minfeaturespace lineend keyword. If the offsetlayer keyword is specified, shapes derived for offsetlayer are merged and output with the lineend layer.

  f. preclean — Outputs SRAFs after the first step of cleaning only for cleanupversion 0 (specifically, removing SRAF violations against target shapes), unless cleanmode postsizing is specified. For cleanupversion 1, preclean outputs a copy of field.

  g. align — Outputs all SRAFs generated by interpriorityalign and intrapriorityalign keywords, after lineendmerge (as applicable), but prior to any cleaning.

  h. process — Outputs SRAFs which are processed by dynamicsizing prior to conflict resolution. This debug mode is accumulative and includes the output of previous priorities.

  i. hardconflict — Outputs the conflict regions prior to filtering. These regions include non-rectangular shapes and may be larger than the maxconflict constraint. Non-rectangular shapes may be converted to rectangular regions by conflictmode bbox

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

for potential processing. Conflict regions are either merged or not merged depending on conflictmode settings merge or nomerge, respectively.

j. conflictbbox — Outputs rectangular bounding boxes of conflict regions as determined by conflictmode bbox, and is used only for the conflictmode bbox keyword. All bounding boxes are processed separately and are not merged, even though they may overlap.

k. conflict — Outputs regions less than or equal to maxconflict in both x- and y-directions that are filtered by cnsraf.

l. conflictbar — Outputs conflict resolution bars (formed as rectangles or squares) prior to resolving conflicts between them and or any other shapes. Conflict bar shapes may overlap each other or overlap target or SRAF shapes. Further cleanup or removal is performed to resolve these conflicts.

m. resolution — Outputs the SRAFs generated by processing conflict regions.

n. isolate — Outputs the final SRAF result by priority.

These additional debug output keywords are listed by function:

Dynamicsizing Related

- dynamic — Outputs SRAFs after symmetric dynamic sizing.
- unsizabledynamic — Outputs SRAFs that are unsizable after symmetric dynamic sizing.

Barsplitting Related

- barsplit — Outputs all SRAFs after barsplitting.
- unsizablebarsplit — Outputs SRAFs that are unsizable after barsplitting.
- barsplit1 — Outputs sizable SRAFs after successful barsplitting.
- unsizablebarsplit1 — Outputs SRAFs that fail barsplitting.
- ubarsplitpostcr — Outputs unsizable SRAFs barsplit after conflict resolution.

Symmetry Related

- asymmetric — Outputs SRAFs after asymmetric dynamicsizing.
- unsizableasymmetric — Outputs SRAFs that are unsizable after asymmetric dynamicsizing.

Line-end Related

- lemu — Outputs SRAFs after initial lineendmerge of overlapping SRAFs.
- lem — Outputs SRAFs after lineendmerge.

Centermerge Related

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- centermerge0 — Outputs SRAFs after initial centermerge with dynamicsizing or trimming.
- centermerge — Outputs SRAFs after centermerge with dynamicsizing or trimming.

Intersection Related

- intersection — Outputs SRAFs after adjustment of existing intersections and construction of new intersections.
- unsizable_intersection — Outputs unsizable SRAFs after intersection construction is complete.
- post_intersection — Outputs SRAFs after construction of intersections and subsequent cleanup operations.
- unsizable_post_intersection — Outputs unsizable SRAFs after construction of all intersections and subsequent cleanup operations.

Final-Stage Related

- oprotected — Outputs keep-out regions around final SRAF shapes.
- positive — Outputs positive SRAFs (in Boolean terms, SRAFs NOT Target).
- negative — Outputs negative SRAFs (in Boolean terms, SRAFs AND Target).
- maxlength — Outputs SRAFs after maxlength splitting is performed, either during per priority cleanup or after all priorities have been processed.
- sraf_p*n* — Outputs a debug layer containing SRAFs only of the specified priority. For a priority 1 template, the sraf_p*n* debug keyword is specified as sraf_p1; for a template of priority 2, sraf_p2, and so on.

- *priority*

  A required integer specifying the priority declared by any template. All SRAFs generated by the template are output by the specified priority. Exceptions include unsizable_barsplit, positive, negative, sraf_p*n*, and others.

- template *index*

  An optional keyword and non-negative integer argument pair that specifies a template to be output for any given priority. This argument pair is valid only for the field debug keyword. This argument permits the debug output of SRAFs limited to a single priority and template combination.

Description

The debug command receives output data from the setlayer cnsraf command and, during cnsraf processing, generates layers for analysis that are viewable in Calibre WORKbench. This intermediate data helps you develop rules and debug cnsraf processing.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

You must specify setlayer cnsraf with a target layer. Generated SRAF layers must be output to the layout database for any associated debug layers to be saved and subsequently viewed. No error message is reported when generated SRAF layers are not saved.

Note

Debug layers may be clipped at tile boundaries, giving unexpected results. Siemens EDA recommends using setlayer tilegen for the generation of tile boundaries to observe debug layer interaction.

Examples

Example 1

This example contains a partial list of debug keys and their respective layers.

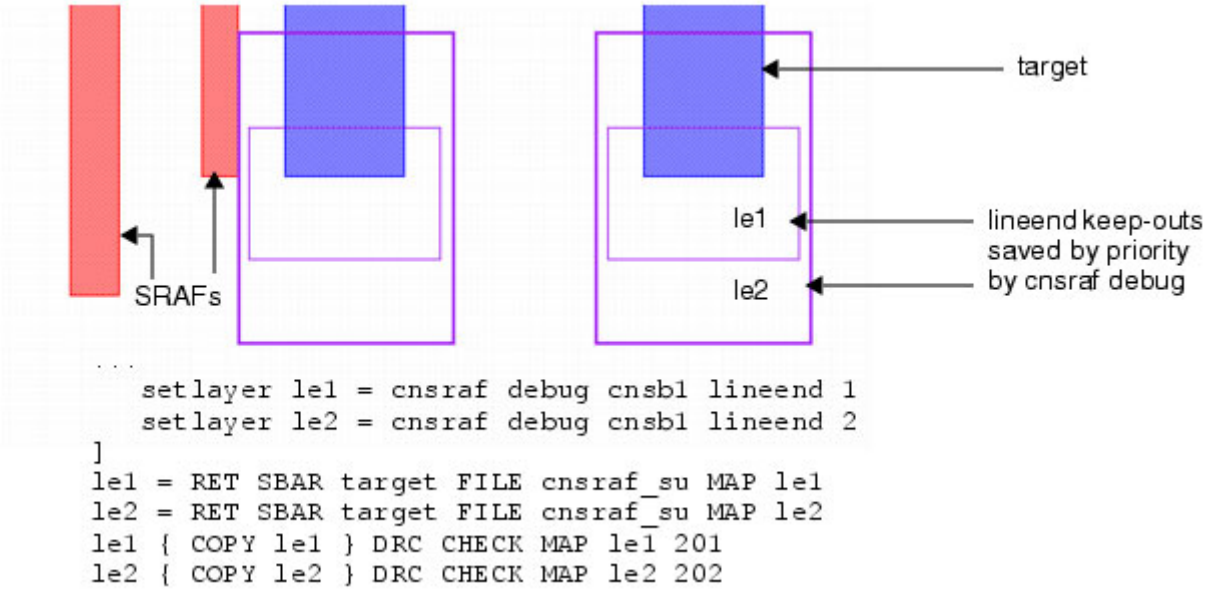
```
LITHO FILE cnsraf_su [
   layer layer1
   ...
   sraf_options OPTS {... prior 1 ... prior 2 ...}
   setlayer cnsb1 = cnsraf target OPTIONS OPTS

   setlayer field_layer        = cnsraf debug cnsb1 field        1
   setlayer hardconflict_layer = cnsraf debug cnsb1 hardconflict 1
   setlayer isolate_layer      = cnsraf debug cnsb1 isolate      1
   setlayer tags_layer         = cnsraf debug cnsb1 tags         1
   setlayer lineend_layer      = cnsraf debug cnsb1 lineend      1
]
field_layer        = RET SBAR target FILE cnsraf_su MAP field_layer
hardconflict_layer = RET SBAR target FILE cnsraf_su MAP hardconflict_layer
isolate_layer      = RET SBAR target FILE cnsraf_su MAP isolate_layer
tags_layer         = RET SBAR target FILE cnsraf_su MAP tags_layer
lineend_layer      = RET SBAR target FILE cnsraf_su MAP lineend_layer
```

Example 2

This example demonstrates lineend protection established by keepout shapes based on priority and then saved to the layout by RET SBAR and COPY.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-3. lineend Debug Layers



Example 3

This example demonstrates usage of the field debug keyword (in red) with priority 60 (in green). The code snippet shown generates the example output in Figure 4-4.

```
LITHO FILE cnsraf_su [
   layer target
   sraf_options TEMPLATE {
      rectedge tag sram width 0.03 offset 0.04 prior 60
      rectedge tag sram_2 width 0.05 offset 0.08 prior 60
   }
   setlayer cn_sraf = cnsraf target OPTIONS TEMPLATE

   setlayer 60_field_t1 = cnsraf debug cn_sraf field 60 template 1
   setlayer 60_field_t2 = cnsraf debug cn_sraf field 60 template 2

]
60_field_t1 = RET SBAR target FILE cnsraf_su MAP 60_field_t1
60_field_t2 = RET SBAR target FILE cnsraf_su MAP 60_field_t2
```

Figure 4-4. Field and Template Output

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4 — Saving Tags for Debugging

These code examples output tags for cnsraf and mbsraf debugging purposes. The tag name is shown in red. The debug keyword required to output tags is green. The rulecheck and database layer names must be identical. They are shown in magenta.

Note

Siemens EDA recommends never using the output of debug layers for SVRF-related post-processing. The only purpose for using the cnsraf debug capability is for debugging SRAF-generated intermediate layers within small regions. A high possibility exists that generated debug layers may be missing or not MRC clean near tile boundaries. Therefore, Siemens EDA does not guarantee clean generation of debug-specific SRAF layers for large designs or hierarchical runs.

```
LITHO FILE cnsraf_setup [
   ...
   sraf_options cnsraf_opts {
      cleanupversion 1
      newtag tag_name external < 0.20
      ...
   }
   setlayer cn_sraf = cnsraf target OPTIONS cnsraf_opts ...
   setlayer tag_name = cnsraf debug cn_sraf tags 1
]
tag_name = RET SBAR target MAP tag_name FILE cnsraf_setup
tag_out {copy tag_name} drc check map tag_out 10

LITHO FILE mbsraf_setup [
   ...
   sraf_options mbsraf_opts {
      algorithm 1
      newtag tag_name external < 0.20
      ...
   }
   setlayer mb_sraf = mbsraf target OPTIONS mbsraf_opts ...
   setlayer tag_name = mbsraf debug mb_sraf tags 1
]
tag_name = RET MBSRAF target MAP tag_name FILE mbsraf_setup
tag_out {copy tag_name} drc check map tag_out 10
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer mbsraf

LITHO FILE Statement and Commands

Generates SRAFs using a model-based methodology. Used by RET MBSRAF only.

Usage

setlayer ***output_layer*** = **mbsraf** ***target_layer*** **OPTIONS** ***statement_name***

Arguments

- ***output_layer***

  The required name of the new output layer. The new layer is created and the results of the operation are written to the layer. This layer name must be used in subsequent operations to output layer information.

- ***target_layer***

  A required input layer that specifies the target layer.

- **OPTIONS** ***statement_name***

  A required keyword and name pair that calls the previously defined sraf_options statement. For more information with mbsraf options, see sraf_options statement in this manual.

Description

This command is used inside of a LITHO FILE statement to generate SRAFs as defined by keywords in the sraf_options statement.

Examples

The following example creates one row of SRAFs that are parallel to the edges and another row of SRAFs that are rotated 45-degrees at the corners of the target layer.

Example 4-1. Use of setlayer mbsraf

```
LITHO FILE sample_setup [

# -------------------- Model path and names --------------------
   modelpath /user/opt/models/model1:/user/opt/models/mbsraf_model
   optical_model_load opticsfile e1_f+0.0000
   resist_model_load resistfile model1.mod

# -------------------- Layer info --------------------
   background 0.0215 -0.0085
   layer sraf_target hidden 0.9785 0.0085
   layer dummy_opc hidden 0.9785 0.0085
   layer sraf_ref_layer hidden 0.9785 0.0085
   layer no_asst_marker hidden 0.9785 0.0085

# ------ specify max tile size to control memory use ------
   tilemicrons 25
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
# ------ sraf_options statement ------
   sraf_options mbsraf.opt {
      correctionlayer dummy_opc
      offsetlayer no_asst_marker
      optical opticsfile dose 1.0
      resist resistfile
      algorithm 0
      peaktype 4dir
      mode nearest
      localmaxgradient width 0.030 prior 1
      priority 1 dynamicsizing 0.03 minoffset 0.010 minfeaturespace 0.035
   }

# ------ setlayer mbsraf call referencing sraf_option statement ------
   setlayer sb_sample = mbsraf sraf_target OPTIONS mbsraf.opt
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer mbsraf debug

LITHO FILE Statement and Commands

Generates and saves in-process debugging layers. Used by RET MBSRAF only.

Usage

setlayer *output_layer* = mbsraf debug *layer debugkey priority*

Arguments

- *output_layer*

  A required output layer name. The results of the debug operation are written to this layer. This layer name must be used in subsequent operations to output debug layer information.

- *layer*

  A required mbsraf layer from which the debug information is extracted.

- *debugkey*

  One of the following keywords must be specified:

  candidates — Outputs the SRAF candidate shapes prior to cleanup. These SRAF candidate shapes are subsequently processed by any cleanup keywords specified. All candidate shapes are output as merged data by mbsraf debug.

  lineend — Outputs the line-end keepout areas derived by the minfeaturespace lineend option. If the offsetlayer keyword is specified, all shapes derived for offsetlayer are merged with the lineend layer.

  negative — Outputs the negative SRAF shapes generated for all priorities.

  positive — Outputs the positive SRAF shapes generated for all priorities.

  seedpixels — All pixels with the minimum number of directions with local maxima in the gradient are displayed by the seedpixels debug layer.

- *priority*

  A number required to specify the priority of the mbsraf debug operation.

Description

The debug command receives output data from the setlayer mbsraf command and, during mbsraf processing, generates layers for analysis that are viewable in Calibre WORKbench. This intermediate data helps you develop rules and debug mbsraf processing.

You must first specify setlayer mbsraf with a target layer for debug use to have any meaning. If not, mbsraf stops with an error message. Additionally, generated SRAF layers must be output to the database for any associated debug layers to be saved and subsequently viewed. No error message is reported when generated SRAF layers are not saved.

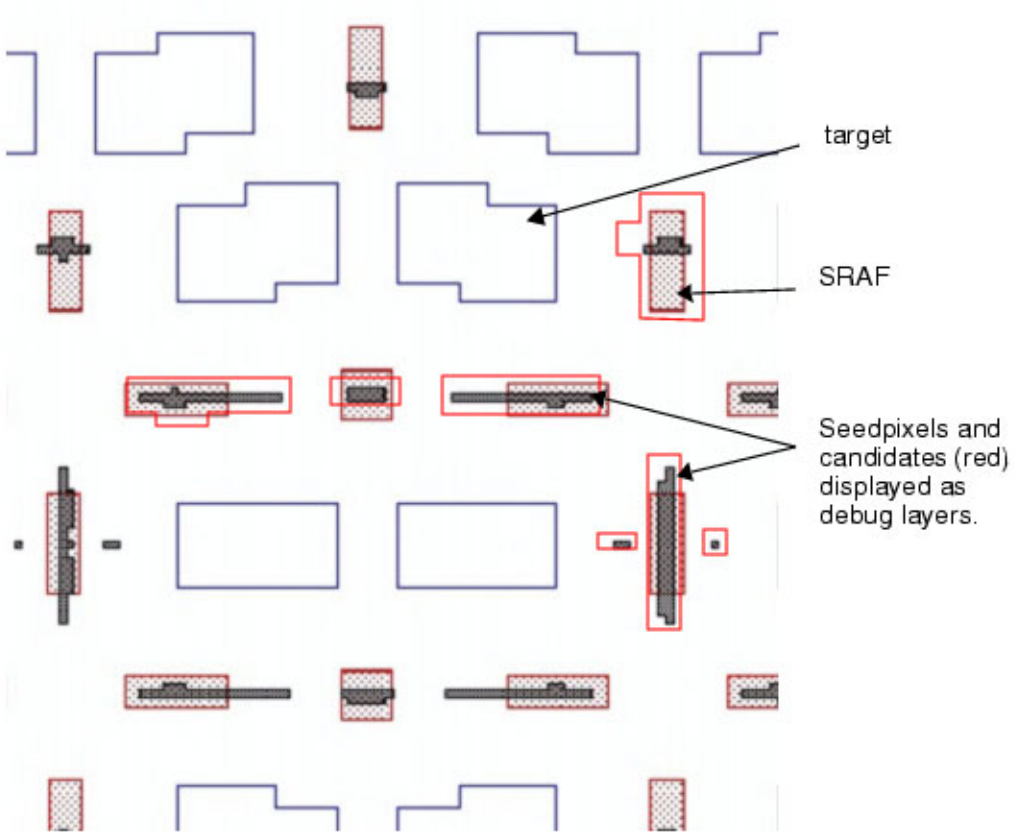
Examples

```
LITHO FILE ca [
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
   sraf_options mbsraf.opt {
      algorithm 1
      minoffset 0.025 ...
   }
   setlayer ca_sbar = mbsraf debug CA OPTIONS mbsraf.opt
   setlayer seedpx = mbsraf debug ca_sbar seedpixels 1
   setlayer candid = mbsraf debug ca_sbar candidates 1
]

ca_sbar = RET SBAR target FILE ca MAP ca_sbar
seedpx = RET SBAR target FILE ca MAP seedpx
candid = RET SBAR target FILE ca MAP candid

ca_sbar {COPY ca_sbar} DRC CHECK MAP ca_sbar 3
seedpx {COPY seedpx} DRC CHECK MAP seedpx 5
candid {COPY candid} DRC CHECK MAP candid 6
```

Figure 4-5. Seedpixel Debug Output Layers



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer tilegen

LITHO FILE Statement and Commands

Creates a layer to display tile and cell boundaries. Used by RET SBAR and RET MBSRAF.

Usage

setlayer *layer* = tilegen *value*

Arguments

- ***layer***

  A required output layer name containing the generated representation of the tiles.

- ***value***

  If the specified value is negative, the actual tiles are sized by the value. If the value is positive, the tile boundaries between valid tile regions are sized, not the actual tiles. The value specified must be between -0.1 and 0.1 microns.

Description

When performing optical process correction, RET SBAR processes the data arrayed in tiles. This command generates a layer that displays the tile boundaries used when processing your layout.

The output layer contains polygons representing every other tile. Therefore, when you display the output layer using a fill pattern, you see a checkerboard pattern.

For a detailed description of tilegen, see the *Calibre OPCVerify User's and Reference Manual*.

Examples

```
setlayer t1 = tilegen -0.05
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

sraf_options

LITHO FILE Statement and Commands

Allows Calibre nmSRAF keywords to be written in free format. The sraf_options statement is called by the setlayer command for SRAF generation. Used by RET SBAR and RET MBSRAF.

Usage

For RET MBSRAF:

sraf_options ***statement_name*** **'{'**

Layer Keywords

[correctionlayer *name*]
[offsetlayer *name*]
[prioritylayer *name*]
[sblayer *name* [sblayer *name*]]
[taglayer *name*]

Tag Keywords

```
[newtag name
   {boolean {...} |
    external {...} |
    internal {...} |
    layer {...} |
    length {...} |
    orientation {...} |
    recta {...} |
    rectb {...} |
    square {...}
   }
]
```

Constraint Keywords

minoffset ***value***
[lineend *value*]
[projecting *value*]
[lineend2long *value*]
[long2lineend *value*]
[corner2sqcorner *value*]
[lineend2sqedge *value*]
[long2sqedge *value*]
[sqcorner2corner *value*]
[sqcorner2sqcorner *value*]
[sqedge2lineend *value*]
[sqedge2long *value*]
[sqedge2sqedge *value*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

width *value*
[alignorigin {0 | 1}]
[extension *value*]
[maxlength *value*]
[maxoffset *value*]
[maxoffsetnegative *value*]
[minarea *value*]
[minfeaturespace *value*
    [lineend *value*
        [lemaxadjacent *value*]
        [lemaxlength *value*]
        [leminadjacent *value*]
        [leminlength *value*]
        [leprojectionin *value*]
        [leprojectionside *value*]
    ]
    [lineend2long *value*]
    [long2lineend *value*]
    [projecting *value*]
    [corner2sqcorner *value*]
    [lineend2sqedge *value*]
    [long2sqedge *value*]
    [sqcorner2corner *value*]
    [sqcorner2sqcorner *value*]
    [sqedge2lineend *value*]
    [sqedge2long *value*]
    [sqedge2sqedge *value*]
]
[minlength *value*]
[minoffsetlayerspace *value*]
    [lineend *value*]
    [lineend2long *value*]
    [long2lineend *value*]
    [projecting *value*]
    [corner2sqcorner *value*]
    [lineend2sqedge *value*]
    [long2sqedge *value*]
    [sqcorner2corner *value*]
    [sqcorner2sqcorner *value*]
    [sqedge2lineend *value*]
    [sqedge2long *value*]
    [sqedge2sqedge *value*]
[minsquarearea *value*]
[minsquarelength *value*]
[minwidth *value*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

[shielding {distance | no | seedpixel}]
[squarewidth *value*]

Templates

localmaxgradient {...}

Style Keywords

algorithm {**0** | **1**}
image_options ***name*** '**{**'
  layer ***name*** **visible** [*mask_layer n*] ...
  litho_model ***litho_model_dir***
  [layer *name* filter]
'**}**'
[centerslit {0 | 1}]
[centerwidth *value*]
[collecttype {0 | 13}]
[cornerpref {all | horizontal | no | square | squareonly | squareonlyscore | vertical}]
[cornerprefnegative {all | horizontal | no | square | squareonly | squareonlyscore | vertical}]
[gradientfactornegative {-1 | 1}]
[gradientshift *value* [autobright]]
[intersection J [minjogwidth *value*]]
[intersectionnegative {J | N} [minjogwidth *value*]]
[jogignore *value*]
[mode {collectno45 | nearest | nearestno45}]
[negative]
[peaktype {1dir | 2dir | 3dir | 4dir}]
[pw_condition *name*
  {[optical *name*]
   [dose *value*]
   [plane *value*]
   [weight *value*]
   [size *value*]
   [defocus *value*]
  }
]
[slopeweight 1 | {verticalweight *value* narrowweight *value* slopeweight *value*}]

Cleanup Keywords

[dynamicsizing *value*]
[dynamicsizingrunlength {0 | 1}]
[interpriorityalign *value*]
[intrapriorityalign *value*]
[lineendmerge *value*]

Priority Keywords

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

[priority *value*
[dynamicsizing *value*]
[interpriorityalign *value*]
[intersection J [minjogwidth *value*]]
[intersectionnegative {J | N} [minjogwidth *value*]]
[intrapriorityalign *value*]
[lineendmerge *value*]
[minarea *value*]
[minfeaturespace *value* [lineend *value*] [projecting *value*]]
[minlength *value*]
[minoffset *value* [lineend *value*] [projecting *value*]]
[minoffsetlayerspace *value*]
[minsquarearea *value*]
[minsquarelength *value*]
[negative]
[shielding {no | seedpixel | distance}]
]

'}'

For RET SBAR:

sraf_options *statement_name* '{'

Layer Keywords

[offsetlayer *name*]
[sblayer *name* [sblayer *name*]]
[taglayer *name*]

Tag Keywords

[newtag *name*
{boolean {...} |
external {...} |
internal {...} |
layer {...} |
length {...} |
orientation {...} |
pairtag {...} |
recta {...} |
rectb {...} |
square {...}
}
]

Constraint Keywords

minoffset *value*
[lineend *value*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
        [projecting value]
        [lineend2long value]
        [long2lineend value]
        [corner2sqcorner value]
        [lineend2sqedge value]
        [long2sqedge value]
        [sqcorner2corner value]
        [sqcorner2sqcorner value]
        [sqedge2lineend value]
        [sqedge2long value]
        [sqedge2sqedge value]
    [alignorigin {0 | 1}]
    [maxlength value]
    [minarea value]
    [minfeaturespace value
        [lineend value
            [lemaxadjacent value]
            [lemaxlength value]
            [leminadjacent value]
            [leminlength value]
            [leprojectionin value]
            [leprojectionside value]
        ]
        [lineend2long value]
        [long2lineend value]
        [projecting value]
        [corner2sqcorner value]
        [lineend2sqedge value]
        [long2sqedge value]
        [sqcorner2corner value]
        [sqcorner2sqcorner value]
        [sqedge2lineend value]
        [sqedge2long value]
        [sqedge2sqedge value]
    ]
    [minlength value]
    [minoffsetlayerspace value]
        [lineend value]
        [lineend2long value]
        [long2lineend value]
        [projecting value]
        [corner2sqcorner value]
        [lineend2sqedge value]
        [long2sqedge value]
        [sqcorner2corner value]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

[sqcorner2sqcorner *value*]
[sqedge2lineend *value*]
[sqedge2long *value*]
[sqedge2sqedge *value*]
[minsquarearea *value*]
[minsquarelength *value*]
[minwidth *value*]

Templates

{pairtemplate {...} |
rect {...} |
rect45 {...} |
rect45corner {...} |
rectcorner {...} |
rectdiag {...} |
rectedge {...} |
recthv {...} |
srafring {...} |
srafringhv {...}
}

Style Keywords

[centerwidth *value*]
[intersection {J | L | N | P | T} [minjogwidth *value*]]
[intersectionnegative {J | L | N | P | T} [minjogwidth *value*]]
[negative]
[symmetric]

Conflict Keywords

maxconflict *value*
[centerthresh *value* [sbwidth *value*]]
[conflictmode {merge | nomerge | shift}]
[oversizesraf {no | yes}]
[sbwidth *value*]
[sraf *value*]

Cleanup Keywords

cleanupversion {**0** | **1**}
[centermerge *max_distance*]
[cleanmode {immediate | postsizing}]
[cnsrafpromotion {0 | 1}]
[dynamicsizing *value*]
[dynamicsizing_ambiguous {0 | 1}]
[dynamicsizingrunlength {0 | 1}]
[interpriorityalign *value*]
[intrapriorityalign *value*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

[lineendmerge *value*]
[maxmove *value*]
[reducedbarsplit {0 | 1 | 2 | 3}]

Priority Keywords

[priority *value*
[centerthresh *value* [sbwidth *value*]]
[cleanmode {immediate | postsizing}]
[conflictmode {merge | nomerge | shift}]
[dynamicsizing *value*]
[interpriorityalign *value*]
[intersection {J | L | N | P | T} [minjogwidth *value*]]
[intersectionnegative {J | L | N | P | T} [minjogwidth *value*]]
[intrapriorityalign *value*]
[lineendmerge *value*]
[maxconflict *value*]
[minarea *value*]
[minfeaturespace *value* [lineend *value*] [projecting *value*]]
[minlength *value*]
[minoffset *value* [lineend *value*] [projecting *value*]]
[minoffsetlayerspace *value*]
[minsquarearea *value*]
[minsquarelength *value*]
[negative]
[oversizesraf {no | yes}]
[sbwidth *value*]
[sraf *value*]
]

'}'

Arguments

- ***statement_name***

  A required argument that is a unique name for the sraf_options statement. This statement is passed by name to the setlayer cnsraf and setlayer mbsraf commands for processing. Currently, there is a 15-character limit on the statement name. For example:

```
sraf_options template {
   ...
}
sraf_options priority {
   ...
}
setlayer cn_sb = cnsraf target OPTIONS template OPTIONS priority \
   ...
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- ‘{’ *sraf_options_keywords* ‘}’

  Any number of required keywords placed between braces ({ }). The keywords may share the same line or span multiple lines. You may format the keywords within the sraf_options statement with white space. Some keywords are exclusive to either cnsraf or mbsraf. The keyword descriptions within their respective group listings state their application.

Table 4-3. sraf_options Keyword and Template Groups

Group	Function
Layer Keywords	Calibre nmSRAF requires input layers for SRAF generation.
Tag Keywords	Tags direct templates where to generate SRAFs. The newtag family of keywords select edges and refine locations of SRAF placement.
Constraint Keywords	Specifies dimensional limits of generated SRAFs.
Templates	Specifies the template for SRAF generation.
Style Keywords	Provides management of SRAF output style.
Conflict Keywords	Provides commands for managing conflicts formed during SRAF generation.
Cleanup Keywords	Invokes cleanup after SRAF generation.
Priority Keywords	Specifies keywords by priority.

Description

The sraf_options statement is used to provide keywords to the setlayer cnsraf and setlayer mbsraf commands for SRAF generation. The setlayer commands do not permit use of white space or comments. The sraf_options statement, however, allows a textual free format for organizing, reading, documenting, and maintaining setlayer cnsraf and mbsraf keywords.

You specify the sraf_options statement inside the LITHO FILE statement and a setlayer command with an OPTIONS argument to call the sraf_options statement. The LITHO FILE statement is read by the RET SBAR or RET MBSRAF commands with the FILE argument:

```
LITHO FILE cn_LF [
   sraf_options cn_statement {
      cleanupversion 1 ...
   }
   setlayer cn_out = cnsraf M1 OPTIONS cn_statement
]
sb_out = RET SBAR M1 FILE cn_LF MAP cn_out
```

The keywords within the sraf_options statement are parsed once to evaluate all variables before runtime. The sraf_options statement supports recursion. This permits you to reference an OPTIONS statement from within the sraf_options statement. Transcripts contain two iterations

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

of the sraf_options keywords: one during parsing, and another with variable substitution completed.

Note

If the specified values of any keyword are incompatible with each other, Calibre nmSRAF issues a warning message and no values are adjusted. Additionally, a minarea value is not assumed from the product of minwidth and minlength if no minarea specification is made.

Examples

Example 1 — setlayer cnsraf

The following example demonstrates an abbreviated sraf_options statement implementation. The target layer is defined in the LITHO FILE statement and then called by setlayer cnsraf.

Example 4-2. Simple sraf_options Statement for setlayer cnsraf

```
LITHO FILE cn_setup_1 [
   layer M1
   tilemicrons 25

   # sraf_options is named cnsraf.opt inside the LITHO FILE statement
   # The statement is passed to setlayer cnsraf

   sraf_options cnsraf.opt {
      cleanupversion 1
      newtag All_Tag length < 0.170
      rectedge tag All_Tag extension 0.03 width 0.04 offset 0.06
      maxconflict 0 sraf 0.030
      centerthresh 0 sbwidth 0.030
   }

   setlayer cnsb1 = cnsraf M1 OPTIONS cnsraf.opt

   setlayer 1_pre = cnsraf debug cnsb1 preclean 1
]
```

Example 2 — setlayer cnsraf

This example demonstrates the use of multiple sraf_options statements organized by function and subsequently called by setlayer cnsraf. The sraf_options statements use white space and comments. To prevent confusion between cleanup keywords and templates always order priorities after templates.

Note

The order of sraf_options statements are arbitrary. The OPTIONS arguments that call the sraf_options statements determine their order.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4-3. Organizational sraf_options Statement for setlayer cnsraf

```
LITHO FILE cn_setup_1 [
   layer target

   sraf_options tags {
      newTag tag1 length > 0.01
   }
   sraf_options general {
      cleanupversion 1
      minoffset 0.020
      maxconflict 0.150
   }
   sraf_options templates {
      rectedge tag tag1 extension 0 width 0.03 offset 0.04 prior 1
      rectedge tag tag1 extension 0.050 width 0.04 offset 0.14 prior 2
   }
   sraf_options priorities {
      priority 1
         # SRAF edge movement
         dynamicsizing 0.1
         # SRAF lineend keepouts
         minfeaturespace 0.040
            projecting 0.040
            lineend 0.070
      priority 2
         lineendmerge 0.040
         minfeaturespace 0.040
            projecting 0.040
            lineend 0.140
   }

   setlayer cnsb1 = cnsraf target OPTIONS tags OPTIONS general \
      OPTIONS templates OPTIONS priorities
]
```

Example 3 — setlayer mbsraf

The following example demonstrates an abbreviated sraf_options statement setup and implementation. The name assigned to the sraf_options statement, and the name called by setlayer mbsraf are green. The target layer is red, and is defined in the LITHO FILE statement and called by setlayer mbsraf.

Example 4-4. sraf_options Statement for setlayer mbsraf

```
LITHO FILE v0sraf.in [
   modelpath /user/demo/IC/20nm/models/exp02
   optical_model_load opt1 e1_f+0.0000 resist_model_load res exp02.mod \
      background 0.0215 -0.0085
   layer sraf_target visible 0.9785 0.0085
   layer cntctopen visible 0.9785 0.0085
   layer sraf_ref_layer visible 0.9785 0.0085
   ...
   tilemicrons 25
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
# sraf_options is named mbsraf.opt inside the LITHO FILE statement
# The statement is passed to setlayer mbsraf

   sraf_options mbsraf.opt {
      correctionlayer cntctopen
      sblayer sraf_ref_layer
      optical opt1 dose 1.0
      resist res
      algorithm 0
      peaktype 4dir
      localmaxgradient width 0.028
   }

   setlayer srafout = mbsraf sraf_target OPTIONS mbsraf.opt
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

svrf_var_import

LITHO FILE Statement and Commands

Imports variables from the SVRF into the LITHO FILE command. Used by RET SBAR and RET MBSRAF.

Usage

svrf_var_import ***var1*** *...* ***varN***

Arguments

- ***var1*** *...* ***varN***

  A required argument that specifies SVRF variables to be imported into the LITHO FILE statement.

Description

Imports each of the listed variables from SVRF (defined using the VARIABLE command), which creates Tcl variables of the same name in the global scope of the setup command block. If a listed variable name is not defined in SVRF, it results in an error. If a listed variable name already exists in the setup, this overrides the variable value.

Examples

Variable substitution is demonstrated using Tcl inside an SVRF file, within a LITHO FILE block, and by declaring environmental variables prior to running Calibre cnSRAF. The use of uppercase is not required but can be used to differentiate between user variables and keywords.

Example 4-5. Using Variables With TCL

```
VARIABLE MINOFFSET 0.020
...
LITHO FILE cn_setup [
   ...
   svrf_var_import MINOFFSET

   sraf_options cnsraf_opts {
      minoffset {use $MINOFFSET} ...
   }
   setlayer cnsb = cnsraf ref_contact_layer OPTIONS cnsraf_opts \
       ...
   ...
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4-6. Using Variables Within a LITHO FILE Statement

```
LITHO FILE mb_setup [
   ...
   set width 0.035
   ...
   sraf_options mbsraf_opts {
      width $width ...
   }
   setlayer mbsb = mbsraf m1_target OPTIONS mbsraf_opts ...
]
```

Example 4-7. Environmental Variable Method

In a csh shell, execute:

```
setenv WIDTH 0.025
```

After the variable is assigned in the shell, you pass it to the LITHO FILE statement:

```
LITHO FILE mb_sraf [
   ...
   sraf_options mbsraf_opts {
      width $env(WIDTH) ...
   }
   setlayer mbsb = mbsraf m1_target OPTIONS mbsraf_opts \
   ...
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

tilemicrons

LITHO FILE Statement and Commands

Specifies the size of a processing tile. Used by RET SBAR and RET MBSRAF.

Usage

tilemicrons *value* [adjust | exact]

Arguments

- *value*

  A number specifying the tile size in microns. The default value is 100 um.

- adjust

  An option adjusting the tilemicrons setting slightly to make it closer to a full multiple of simulation frames if possible. A simulation frame is a fast Fourier transform (FFT) frame, which is roughly six to ten microns in size. This is the default setting.

- exact

  An option using the value as specified.

Description

Specifies the size of a processing tile. Large tiles are broken into tiles of the value specified.

Examples

```
tilemicrons 35
```

Recommended Settings for tilemicrons per Node

The selection of tile size is critical for optimum processing. If the tile size is too large or too small, primary or remote memory consumption may go into swap and slow performance. The empirical, optimum tile size settings correlate to the technology node and balance the size of data per tile. Siemens EDA recommends that you set tilemicrons to the following values based on the node listed in Table 4-4. For double patterning, use the smaller values listed in the table.

Table 4-4. Hardware Requirements and tilemicrons Settings

Technology Node	Hardware Requirement [1]	tilemicrons Setting
>=45 nm	2 GB per core	45 to 55 um
32 nm	3 GB per core	35 to 40 um (30 um for M1)
22 nm	4 GB per core	25 to 30 um (20 um for M1)
EUV	> 4 GB per core	5 um

1. In each case, double the memory per core to support hyperthreading. For example, use an 8 GB per core at 22 nm, which is 64 GB per host on 8-core remotes and 96 GB per host on 12-core remotes.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

LITHO FILE Statements for EUV

You write statements within the LITHO FILE block to control RET EUV MBSRAF commands.

The statements shown in Table 4-5 are called by RET [EUV] MBSRAF only, and are specified one statement per line in the LITHO FILE block. They do not appear in any sraf_options statement.

Table 4-5. LITHO FILE Statements for EUV

Command	Description
bbm_options	Sets parameters for a black border model, overriding the values in a loaded model.
euv_field_center	Specifies the center of an EUV field.
euv_slit_x_center	Sets the X-coordinate for the center of an EUV through-slit litho model extension.
flare_longrange	Loads long range flare information from a litho model.
flare_model_load	Loads a flare model through an inline model.
shadow_bias_model_load	Loads a shadow bias through an inline model.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

setlayer sraf_options Keywords

Specify keywords within the sraf_options statement to control setlayer cnsraf and setlayer mbsraf commands.

The groups shown in Table 4-6 are comprised of keywords specified within the sraf_options statement residing in the LITHO FILE statement. Some keywords are used by both setlayer cnsraf and setlayer mbsraf, some are exclusive to either. The keyword short descriptions state their respective application.

Table 4-6. setlayer sraf_options Statement Keyword Groups

Group	Description
Layer Keywords	Calibre nmSRAF requires input layers for SRAF generation.
Tag Keywords	Tags direct templates where to generate SRAFs. The newtag family of keywords select edges and refine locations of SRAF placement.
Constraint Keywords	Calibre nmSRAF requires the specification of some minimum SRAF dimensions for application of MRC rules.
Templates	Calibre nmSRAF requires dimensional- or model-based templates to pattern and prioritize SRAF generation.
Style Keywords	Provides management of SRAF output style.
Conflict Keywords	Provides commands for managing conflicts formed during SRAF generation.
Cleanup Keywords	Calibre nmSRAF invokes SRAF cleanup after generation.
Priority Keywords	Calibre nmSRAF provides many keywords to be called by priority.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Layer Keywords

Calibre nmSRAF requires input layers for SRAF generation.

For organizational keyword information, refer to “sraf_options” on page 80.

Table 4-7. Layer Keywords

Keyword	Description
correctionlayer	Specifies a polygon layer for gradient calculation. Used by MBSRAF only.
offsetlayer	Specifies a layer name representing areas within which to prohibit positive and negative SRAF generation. Used by RET SBAR and RET MBSRAF.
prioritylayer	Specifies shapes or edges prioritized as targets during SRAF generation. Used by RET MBSRAF only.
sblayer	Specifies a layer name containing preexisting SRAFs. Used by RET SBAR and RET MBSRAF.
taglayer	Specifies a tag name representing imported layers or edges from SVRF. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

correctionlayer

Layer Keywords

Specifies a polygon layer for gradient calculation. Used by MBSRAF only.

Usage

correctionlayer *name*

Arguments

- *name*

  Specifies the name of the polygon correction layer.

Description

An optional keyword and polygon layer name used to calculate the gradient. The correctionlayer may only be specified once. If multiple correctionlayer keywords are specified, only the last one defined is used, and all other correctionlayer keywords are ignored.

The correctionlayer for mbsraf can be a sized-up target layer, or an OPC layer (resized by several iterations using nmOPC). The image generated using the correctionlayer should be within plus or minus 20 percent of the expected nominal contour image.

These modifications provide enlarged shapes that lead to an improved gradient. The correctionlayer is used during simulation to define the initial image that forms the basis of the gradient calculation.

The sblayer and correctionlayer are layers containing shapes that are simulated to provide the image from which the gradient is calculated. If neither of these layers are available, the target layer is used for the image simulation.

Additionally, when correctionlayer is specified, it must appear in either the options block or the setlayer command, but never within both. If the correctionlayer is required in other areas of the recipe, you must derive it again under a different name prior to processing it.

Note

Siemens EDA recommends using the correctionlayer for reliable, better performing SRAF, especially with bright field masks. For more information regarding correctionlayer, see the section entitled “Calibre mbSRAF Recipe Development” on page 333 in the Best Practices chapter.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

offsetlayer

Layer Keywords

Specifies a layer name representing areas within which to prohibit positive and negative SRAF generation. Used by RET SBAR and RET MBSRAF.

Usage

offsetlayer *name*

Arguments

- *name*

  Specifies the name of the layer.

Description

An optional keyword and layer name that contains preexisting polygons or edges that define areas where positive and negative scattering bars cannot be generated. SRAFs can abut but not overlap offsetlayer shapes.

The offsetlayer keyword should only be used as an alternative when the lineend parameters provided by minfeaturespace do not provide sufficient keepout areas. To avoid processing redundancies, the target layer and taglayer should never be used as the offsetlayer. Instead, the offsetlayer can always be derived with Calibre SVRF commands and then input to setlayer cnsraf. Only one offsetlayer may be specified. Only manhattan- and 45-degree input layer geometries are supported. Additionally, no skew-edged geometries are valid offsetlayer shapes for SRAFs.

If offsetlayer is specified, it must appear in either the sraf_options statement or the setlayer command to avoid duplication. If the offsetlayer is required in other areas of the recipe, you must derive it again under a different name prior to processing it.

The minoffsetlayerspace keyword may be specified per priority or globally to check the offsetlayer.

Note

As of release 2013.4 and later, SRAFs may abut the offsetlayer shape. In past releases, all SRAFs were kept at a spacing equal to minfeaturespace from the offsetlayer. To model previous SRAF-offsetlayer spacing behavior, use minoffsetlayerspace to enforce this space.

Examples

The code shown here demonstrates manually-derived keepout areas that prohibit placement of SRAFs. After derivation of the line-end keepout area, the layer is then input to the cnsraf and RET SBAR commands. When the offsetlayer keyword and the lineend debug keyword in the cnsraf debug command are specified, the offsetlayer output is merged into the lineend output layer. See “setlayer cnsraf debug” on page 69 for more information.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
offset_edge = CONVEX EDGE target
   ANGLE1 == 90 LENGTH1 > 0.100 //leg1 gt value ( leminadjacent)
   ANGLE2 == 90 LENGTH2 > 0.100 //leg2 gt value (leminadjacent)
   WITH LENGTH >= 0.090 < 0.700 //le range (leminlength, lemaxlength)
offset_layer = EXPAND EDGE offset_edge
   EXTEND BY 0.030 //le side extension (leprojectingside)
   OUTSIDE BY 0.2  //le outward extension (lineend)
   INSIDE BY 0.1   //le inward projection (leprojectingin)
LITHO FILE cn_setup_1 [
   layer target
   layer offset_layer
   setlayer cnsb1 = cnsraf target offsetlayer offset_layer ... \
]
cnsb1 = RET SBAR target offset_layer FILE cn_setup_1 MAP cnsb1
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

prioritylayer

Layer Keywords

Specifies shapes or edges prioritized as targets during SRAF generation. Used by RET MBSRAF only.

Usage

prioritylayer *name* [prioritylayer *name*...]

Arguments

- *name*

  The name of a shape or edge.

Description

An optional keyword and layer name specifying original or derived polygons or edges that take priority during SRAF generation. Where possible, SRAFs are placed first near shapes in the specified prioritylayer within the mbsraf minfeaturespace distance from those shapes. You may specify multiple prioritylayer keywords.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

sblayer

Layer Keywords

Specifies a layer name containing preexisting SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

sblayer *name* [sblayer *name*]

Arguments

- *name*

  The name of a polygon layer containing SRAFs. If the sblayer name is specified once, it may contain both positive and negative SRAFs. Alternatively, the sblayer may be specified twice with no order dependency, one layer for positive SRAF and one layer for negative SRAF.

Description

An optional keyword used to import previously generated SRAF layers. All cnsraf templates can reference these layers by name. No SRAFs are placed within a distance of minoffset to the shapes imported on the sblayer.

If sblayer is specified twice, to differentiate between positive and negative SRAFs:

- Positive SRAFs are completely outside of any target shape (no logical interaction including butting edges).
- Negative SRAFs are completely inside target shapes (logical enclosure and no coincident edges).

The MRC keywords used for checking the sblayer may be specified per priority or globally:

- lineend
- minoffset
- projecting

If sblayer is specified, it must appear in either the options block or the setlayer command, but never within both. If sblayer is required in other areas of the recipe, you must derive it again under a different name prior to processing it.

Only manhattan- and 45-degree input layer geometries are supported. No SRAFs with skew-edges are valid as sblayer names.

The sblayer and correctionlayer are the layers containing shapes that are simulated to provide the image from which the gradient is calculated. If neither of these layers are available, the target layer is used for the image simulation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

taglayer

Layer Keywords

Specifies a tag name representing imported layers or edges from SVRF. Used by RET SBAR and RET MBSRAF.

Usage

taglayer *name* [taglayer *name*...]

Arguments

- *name*

  Specifies the name of the taglayer.

Description

An optional keyword used to import polygons or edges into templates from SVRF, derived as needed and named for subsequent tag use. With specific exceptions, all cnsraf templates may reference taglayers by name. Multiple taglayers can be referenced in a single template. The input taglayer is internally ANDed with the target layer, and the resulting edges are then used as the operational taglayer. The taglayer must not also be used as an offsetlayer.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

Examples

This example demonstrates one possible layer flow using the taglayer keyword.

```
LAYER target 0
x1 = EXPAND EDGE by 0.001 (EXT [target] > 0.1 < 0.5 opposite)

LITHO FILE cn_setup_1 [
   layer target
   layer x1
   ...
   sraf_options template {
      cleanupversion 1
      taglayer x1
      rectEdge tag x1 extension 0.01 width 0.03 offset 0.04 prior 1
   }
   setlayer cnsb1 = cnsraf target OPTIONS template
]
sraf = RET SBAR target x1 FILE cn_setup_1 MAP cnsb1
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Tag Keywords

Tags direct templates where to generate SRAFs. The newtag family of keywords select edges and refine locations of SRAF placement.

Although tags are optional, they provide significant capabilities for SRAF generation for each template. The templates reference tags by a tag keyword, followed by a tag name. Multiple tags can be specified for a template. If a template uses multiple tags, the tags will be combined through a logical OR operation. All templates of the same priority must use the same tags.

In-place Tags

If the name of a tag is defined only for the use of another tag, the first tag name explicitly defined can be replaced by the body of the tag enclosed in a pair of parenthesis. The two lines of the newtag commands can be merged into one. For example:

```
newtag tag1 recta > 0.01
newtag tag2 tag1 length > 0.02
```

can be described as:

```
newtag tag2 ( recta > 0.01 ) length > 0.02
```

An automatically generated tag name is assigned to each parenthesized in-place tag. This tag name is used to manage the tag internally and is not accessible outside Calibre.

The components in a boolean tag, as well as the predefined tag specified for some types of tags may be a fully parenthesized boolean expression. All components of these expressions must be previously defined tags. The allowed boolean operators are OR, AND, NOT and XOR.

Calibre nmSRAF does not define precedence rules between the operators. Therefore, you must use parentheses to explicitly define the order of logic. The nested boolean expression must be enclosed in an outermost pair of parentheses. For example:

```
newtag tag1 length < 0.5
newtag tag2 recta > 0.01
newtag tag3 external > 0.2
newtag tagfinal tag1 OR (tag2 AND tag3)
```

can be described as:

```
newtag tagfinal (length < 0.5) OR ((recta > 0.01) AND (external > 0.2))
```

Calibre cnSRAF Template Behavior

Tags usually select some parts of target edges. The rectedge and rectcorner templates generate their candidates for these edges.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

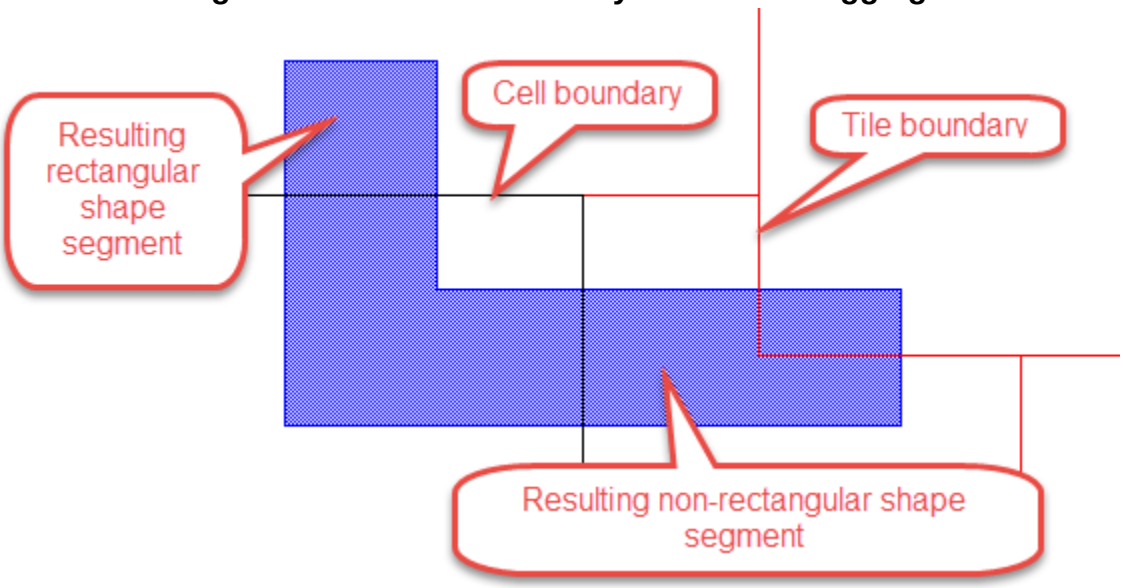
The rectcorner and rectcorner45 templates generate SRAFs for corners of the target layer if the corner is part of a tagged partial edge. They do not generate candidates for end points of tagged partial target edges that are not tagged.

All Calibre cnSRAF templates other than rectedge, rectcorner, and rect45corner are pitch based. They generate candidates only for tagged target rectangles. A target rectangle is referred to as tagged if any part of the rectangle boundary coincides with a tagged edge.

How Design Hierarchy or Tiling Partitions Can Affect Tagging Behavior

Flat designs have no hierarchy-based effects on polygon tagging, but are still susceptible to effects of tile boundaries. Tagging shapes in hierarchical designs is susceptible to hierarchy- and tile boundary-effects. Figure 4-6 provides examples of potential tagging issues that may render rectangle-based tagging keywords recta and rectb to not function as expected.

Figure 4-6. Tile and Hierarchy Effects on Tagging



For organizational keyword information, refer to “sraf_options” on page 80.

Table 4-8. Tag Keywords

Keyword	Description
newtag boolean	Specifies a tag name representing boolean combinations of layers or edges. Used by RET SBAR and RET MBSRAF.
newtag external and internal	Specifies a tag name representing the inner or outer edges of layers for which to generate SRAFs. Used by RET SBAR and RET MBSRAF.
newtag layer	Specifies a tag name representing all target edges interacting with the specified tag layer. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 4-8. Tag Keywords (cont.)

Keyword	Description
newtag length	Specifies a tag name representing all target edges qualified by the length specified. Used for RET SBAR and RET MBSRAF.
newtag orientation	Specifies a tag name representing orientations for edges, corners and angles. Used by RET SBAR and RET MBSRAF.
newtag pairtag	Specifies a tag name, exclusively for pairtemplate, that represents layers or edges for which to generate SRAFs. Used by RET SBAR only.
newtag recta and rectb	Specifies a tag name representing all edges of rectangular target shapes matching the specified constraint. Used by RET SBAR and RET MBSRAF.
newtag square	Specifies a tag name representing the edges of square target shapes matching the constraint. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag boolean

Tag Keywords

Specifies a tag name representing boolean combinations of layers or edges. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name tagname1 boolean tagname2* [fulledge]

newtag *name boolean tagname1 tagname2* [*tagnameN*...] [fulledge]

Arguments

- *name*

  Specifies the name of the tag.

- *tagname0*, *tagname1*, *tagname2*

  Previously derived tags generated by taglayer or newtag commands.

- *boolean*

  Specifies a boolean operator used to derive tags:

  AND — Outputs overlapping portions of the input tags.

  NOT — Outputs portions of the first input tag not covered by the second tag. If NOT is used in unary form, the output consists of portions of the target layer not covered by the second tag.

  OR — Outputs a combination of both input tags.

  XOR — Outputs mutually exclusive portions of the input tags.

- fulledge

  An optional keyword forcing the entire length of a tagged edge to be used even when a shorter portion of the edge is originally tagged. This option is compatible with all constraint types.

Description

Selects edges processed by boolean operations to combine the results of other tags. The first usage syntax is for operations between two tags. The next usage defines boolean operations for two or more tags. All boolean operations are performed on sets of partial edges defined by previously defined tags, and select coinciding edges or merge the sets of edges.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect.

Examples

Example 1 — Newtag Boolean Usage

```
taglayer M1
newtag sq square <= 0.05
newtag sh length < 0.010
newtag small_square sq AND sh
newtag long NOT sh

newtag or2  t1 OR t2
newtag or4  OR t1 t2 t3 t4
newtag not1 NOT t1
newtag xor3 XOR t1 t2 t3
newtag fuzzy (t1 AND t2) OR (t3 XOR (t4 AND t5))
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag external and internal

Tag Keywords

Specifies a tag name representing the inner or outer edges of layers for which to generate SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name* {[*predefined_tag*] | [*from_tag to_tag*]}
{[external *constraint*] [internal *constraint*]} [fulledge] [*metric*] [projecting]

Arguments

- *name*

  Specifies the name of the tag.

- *predefined_tag*

  Previously derived tags generated by taglayer or newtag commands. Mutually exclusive with from_tag and to_tag.

- *from_tag*

  A previously derived tag from which to measure to the to_tag. Mutually exclusive with predefined_tag.

- *to_tag*

  A previously derived tag from which to measure to the from_tag. Mutually exclusive with predefined_tag.

- external *constraint*

  A keyword applied to the outside edges of target shapes used for spacing. The constraint takes several forms:

  - <= *value*
  - <= *value1* >= *value2*
  - <= >= *value*

- internal *constraint*

  A keyword applied to the inside edges of target shapes used for widths. The internal constraints use the same forms as external. As a default, the promotion distance is used for the upper bound.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Caution

Upper bounds for the internal keyword are critical:

- When using the internal keyword to generate tags used in the context of rectedge center strictcenter template, you must provide an upper bound to the internal constraint to enable limits during edge collection.
- When applying the internal keyword to any pitch-based template, you must provide an upper bound to the constraint to enable limits during edge collection.

- fulledge

  An optional keyword forcing the entire length of a tagged edge to be used even when a shorter portion of the edge is originally tagged. This option is compatible with all constraint types.

  Note

  Partial edge tagging with Calibre mbSRAF may lead to unexpected results because of differences between intended SRAF and MRC enforcement. Siemens EDA recommends supplying the entire edge to mbsraf when using tagging.

- *metric*

  An optional metric used to qualify the constraint. The euclidean, opposite and opposite_extended metrics are mutually exclusive. Any of these three metric constraints may be used with projecting.

  euclidean — The external and internal constraint is measured radially from edges. This is the default metric.

  opposite — The external and internal constraint is measured only between mutually facing edges.

  opposite_extended *value* — The external and internal constraint is measured between mutually facing edges, plus a positive extension value added to each end of the edge pair.

- projecting

  An optional keyword using only those edges that face each other for an edge length that satisfies the constraint. The projecting keyword is compatible with all metrics.

  Note

  Partial edge tagging with Calibre mbSRAF may lead to unexpected results because of differences between intended SRAF and MRC enforcement. Siemens EDA recommends supplying the entire edge to mbsraf when using tagging.

Description

An optional keyword used to input tags by name to Calibre nmSRAF templates. Multiple tags can be referenced in a single template. The newtag keyword uses external and internal to

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

measure distance between tags and target for selective edge pair isolation. Alternatively, newtag can be used to measure and isolate edges between two tags. See the examples for further information.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

When using tagging, positive SRAFs may be generated parallel and perpendicular (given template constraints) relative to tagged edges and negative SRAFs are generated only parallel to tagged edges. See the example in Figure 4-115 on page 263 for more information.

Shielding is implicit for all internal and external newtag operations and is equivalent to EXCLUDE_SHIELDED 4 for the SVRF INTERNAL/EXTERNAL commands.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect.

Examples

Example 1 — Specifying Tags as a Combination of Space and Width

The first method demonstrates proper use of internal and external keywords setting width and space constraints, respectively, with two newtag commands, the first newtag tag (mapped in red) passed to the second newtag command:

```
newtag tag0 internal > 0.1 < 0.25
newtag tag1 tag0 external > 0.04 < 0.080
```

The second tag demonstrates valid usage, but when combining the two keywords in a single newtag command, the internal filter is ignored and the output of tag_50_100 is equal to all of the target edges that satisfy only the external constraint:

```
newtag tag_50_100 internal >= 0.1 < 0.25 external > 0.05
```

Example 2 — Applying Metrics to Tags

```
newtag ltag external >= 0.1 opposite

newtag ltag external >= 0.1 opposite_extended 0.040 projecting
```

Example 3 — Extracting Projecting Edges by Run Length

```
newtag external <= 0.070 opposite length > 0.10
```

Example 4 — Compounding length Tags

```
newTag tag1 l1 length > 0.10
newTag tag2 l2 length < 0.05
newTag tag3 external <0.13
newTag final tag3 and (tag1 or tag2)
```

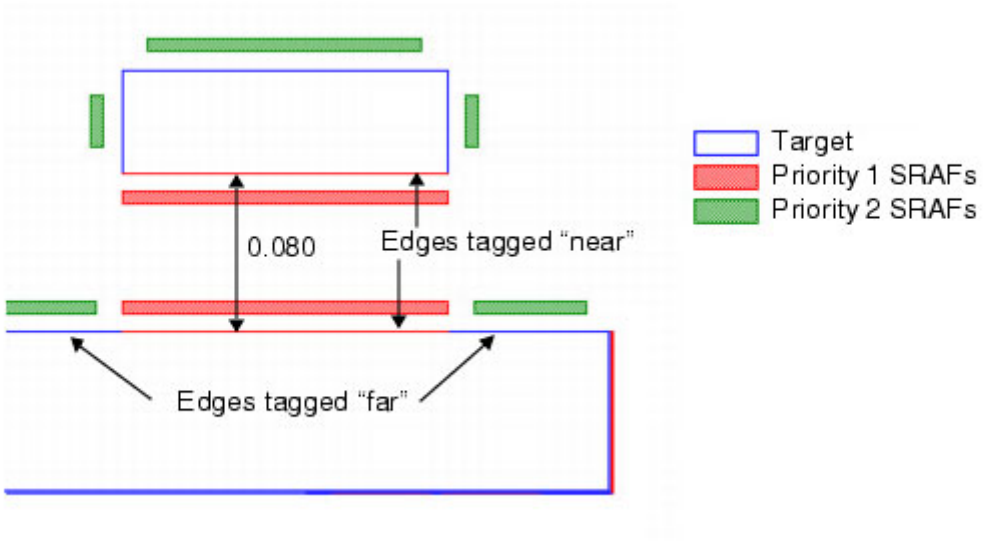

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 5 — Newtag external Keyword

Figure 4-7 and the code snippet shown illustrate newtag near applied to external edges with spaces greater than 0.060, but less than or equal to 0.090. Only the red SRAFs of priority 1 are output between these edges. Newtag far is applied to external edges with spacing greater than 0.090. Green SRAFs of priority 2 are output on these edges.

```
newtag near external > 0.060 <= 0.090
newtag far external > 0.090
localmaxgradient tag near width 0.024 prior 1
localmaxgradient tag far width 0.024 prior 2
```

Figure 4-7. mbsraf newtag external Usage



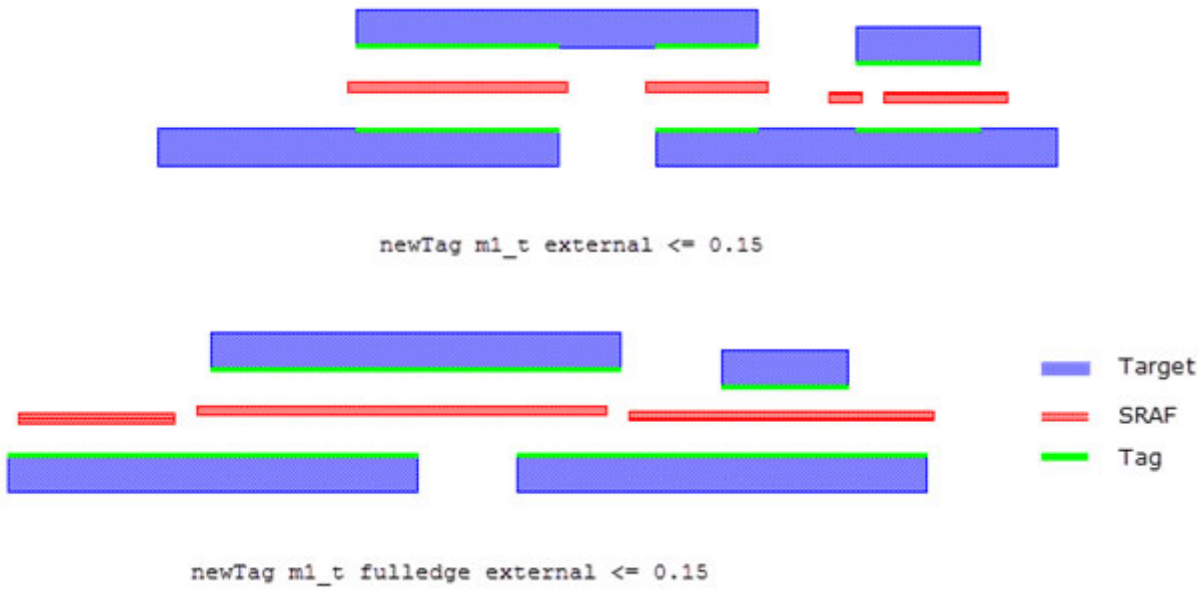
Example 6 — Newtag fulledge Keyword

In the first illustration of Figure 4-8 without fulledge, newtag external is applied to edges farther apart than 0.15nm. Only portions of the target edges are selected, and the generated SRAFs adhere to these tags. In the second illustration with fulledge, when external is used in conjunction with fulledge, the entire edge of the target shapes is selected, changing the SRAF output.

```
newTag m1_t fulledge external <= 0.15
localmaxgradient tag m1_t width 0.018 extension 0.015
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-8. newtag fulledge Usage



Example 6 — Saving Tags for Debugging

These code examples output tags for cnsraf and mbsraf debugging purposes. The tag name is shown in red. The debug keyword required to output tags is green. The rulecheck and database layer names must be identical. They are shown in magenta.

```
LITHO FILE cnsraf_setup [
   ...
   newtag tag_name external < 0.20
   ...
   setlayer cn_sraf = cnsraf target ...
   setlayer tag_name = cnsraf debug cn_sraf tags 1
]
tag_name = RET SBAR target MAP tag_name FILE cnsraf_setup
tag_out {copy tag_name} drc check map tag_out 10

LITHO FILE mbsraf_setup [
   ...
   newtag tag_name external < 0.20
   ...
   setlayer mb_sraf = mbsraf target ...
   setlayer tag_name = mbsraf debug mb_sraf tags 1
]
tag_name = RET mbsraf target MAP tag_name FILE mbsraf_setup
tag_out {copy tag_name} drc check map tag_out 10
```

Example 7 — Implicit Boolean AND Equivalence

This code example demonstrates the equivalence of the implicit AND during tag construction. The output of both v_internal and v_internal_and are identical.

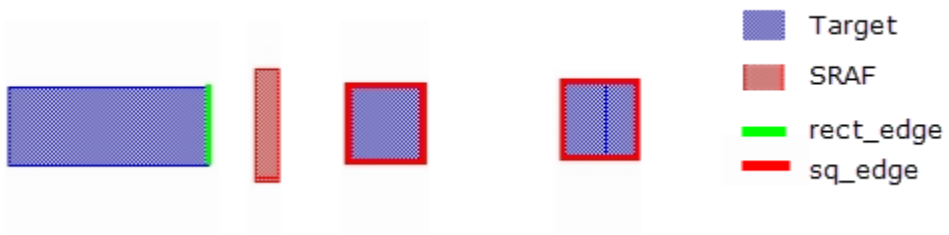
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
sraf_options tag {
   newtag v (vertical)
   newtag v_internal v internal >= 0.230 < 0.315
   newtag v_internal_and v and (internal >= 0.230 < 0.315)
}
```

Example 8 — Measurements Between from_tag and to_tag

In this example, SRAFs must be placed between the rectangular and square target shapes, but not between the two square target shapes. The measurement is performed between two tags that were previously generated, rect_edge and sq_edge, from the rectangular target, and square target shapes, respectively.

Figure 4-9. newtag From and To Tag Usage




```
# The rectangle edge is tagged (this is positioned as the from_tag):
newtag rect_edge recta < 0.085 rectb > 0.085

# The square edge is tagged (this is positioned as the to_tag):
newtag sq_edge square < 0.085

# An external measurement is performed between the from_tag and
# to_tag, rect_edge and sq_edge, respectively, and named sq2rect:
newtag sq2rect rect_edge sq_edge external < 0.160

# Finally, tag sq2rect is specified in a rectedge template:
rectedge tag sq2rect extension 0.02 width 0.025 offset 0.05 prior 2
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag layer

Tag Keywords

Specifies a tag name representing all target edges interacting with the specified tag layer. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name* {layer *taglayer_name*} [asis] [fulledge]

Arguments

- *name*

  Specifies the name of the tag.

- layer *taglayer_name*

  A required keyword and layer pair that selects all target edges interacting with the specified tag layer. A taglayer is defined in the setlayer command using the taglayer keyword. The layer may be a polygon or an edge.

- asis

  The asis keyword specifies edges of the taglayer used in the tag. The fulledge and asis keywords are mutually exclusive. Layer tags using the asis keyword must not be mixed with other tags, and they cannot be used as boolean tags. If templates specify multiple tags, asis layer tags and other tags must not be mixed. You must perform appropriate boolean operations outside of Calibre nmSRAF.

- fulledge

  An optional keyword forcing the entire length of a tagged edge to be used even when a shorter portion of the edge is originally tagged. This option is compatible with all constraint types.

  Note

  Partial edge tagging with Calibre mbSRAF may lead to unexpected results because of differences between intended SRAF and MRC enforcement. Siemens EDA recommends supplying the entire edge to mbsraf when using tagging.

Description

An optional keyword used to input tags by name to Calibre nmSRAF templates. Multiple tags can be referenced in a single template.

When using tagging, positive SRAFs may be generated parallel and perpendicular (given template constraints) relative to tagged edges and negative SRAFs are generated only parallel to tagged edges. See the example in Figure 4-115 on page 263 for more information.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect.

Examples

Example 1 — Managing Tag Names

```
taglayer tagl1
newtag tagl2 layer tagl1 fulledge
newtag tagl3 layer tagl1 asis
```

Example 2 — Compounding length Tags

```
newTag tag1 l1 length > 0.10
newTag tag2 l2 length < 0.05
newTag tag3 external <0.13
newTag final tag3 and (tag1 or tag2)
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag length

Tag Keywords

Specifies a tag name representing all target edges qualified by the length specified. Used for RET SBAR and RET MBSRAF.

Usage

newtag *name* [*predefined_tag*] {length *constraint*} [fulledge]

Arguments

- *name*

  Specifies the name of the tag.

- *predefined_tag*

  Previously derived tags generated by taglayer or newtag commands.

- length *constraint*

  A constraint applied to the edges of target shapes and existing tags used for classifying edge length and projecting edge run length. When applied to target shapes, length defaults to inputting the entire target layer. The length type can also accept up to one existing tag. Use the opposite metric with length to find the projecting edge pair run length. To measure lengths of existing tags, specify the existing tag immediately preceding the length type.

  Note

  When applying the length keyword to any pitch-based template, you must provide an upper bound to the constraint to enable limits during edge collection.

- fulledge

  An optional keyword forcing the entire length of a tagged edge to be used even when a shorter portion of the edge is originally tagged. This option is compatible with all constraint types.

  Note

  Partial edge tagging with Calibre mbSRAF may lead to unexpected results because of differences between intended SRAF and MRC enforcement. Siemens EDA recommends supplying the entire edge to mbsraf when using tagging.

Description

An optional keyword used to input tags by name to Calibre nmSRAF templates. Multiple tags can be referenced in a single template.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

When using tagging, positive SRAFs may be generated parallel and perpendicular (given template constraints) relative to tagged edges and negative SRAFs are generated only parallel to tagged edges.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect.

Examples

Example 1 — Newtag length Usage

```
newtag medEdge      length >  0.050 <= 0.065
newtag longEdge     length >  0.065
```

Example 2 — Newtag length Used with Boolean Keywords

```
taglayer M1
newtag sq square <= 0.05
newtag sh length < 0.010
newtag small_square sq AND sh
newtag long NOT sh
```

Example 3 — Specifying a Portion of an Existing Tag with length

```
taglayer M1_tag
newtag long_tag M1_tag length > 0.10
```

Example 4 — Compounding length Tags

```
newTag tag1 l1 length > 0.10
newTag tag2 l2 length < 0.05
newTag tag3 external <0.13
newTag final tag3 and (tag1 or tag2)
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag orientation

Tag Keywords

Specifies a tag name representing orientations for edges, corners and angles. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name* ‘(’*orientation*‘)’

Arguments

- *name*

  Specifies the name of the tag.

- *orientation*

  Specifies the orientation keyword which must be enclosed in parenthesis when accompanied by constraints:

  angled — Selects all edges not orthogonal to the X- or Y-axes.

  eastedge — Selects all edges to the right side of shapes.

  horizontal — Selects all edges parallel to the X-axis.

  northeastcorner — Selects and outputs 5dbu of the two edges forming the corresponding corner of the upper-right corner of target shapes.

  northedge — Selects all edges to the upper side of shapes.

  northwestcorner — Selects and outputs 5dbu of the two edges forming the corresponding corner of the upper-left corner of target shapes.

  southeastcorner — Selects and outputs 5dbu of the two edges forming the corresponding corner of the lower-right corner of target shapes.

  southedge — Selects all edges to the lower side of shapes.

  southwestcorner — Selects and outputs 5dbu of the two edges forming the corresponding corner of the lower-left corner of target shapes.

  vertical — Selects all edges parallel to the Y-axis.

  westedge — Selects all edges to the left side of shapes.

Description

The orientation tag selects target edges of the specified orientation or a certain side in an axis-aligned polygon. Excepting the angled orientation, target polygons containing angled edges are not selected. All of the orientations are mutually exclusive. Use boolean tags or specify multiple tags in the templates to combine different orientations.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

All corner-based tags are generated with two adjacent edges of 5 dbu in length. These two adjacent edges are imported when specified using the tag argument by the rectedge, rect45corner, and rectcorner templates.

Examples

Example 1 — Basic Newtag Orientation Usage

```
newtag tag1 vertical
newtag tag2 angled
newtag tag3 northedge
```

Example 2 — Dimensional Usage

```
newtag AllShortEdges length <  0.1
newtag AllLongEdges length >= 0.1
```

Example 3 — Combined Boolean and Dimensional Usage

Note

In-place assignment by orientation keywords requires their parenthetic enclosure. Two general forms of boolean syntax apply for tags, however, only the latter form is valid for boolean operations with keywords:

- <operator> <tag> <tag>
- <tag> <operator> <tag>

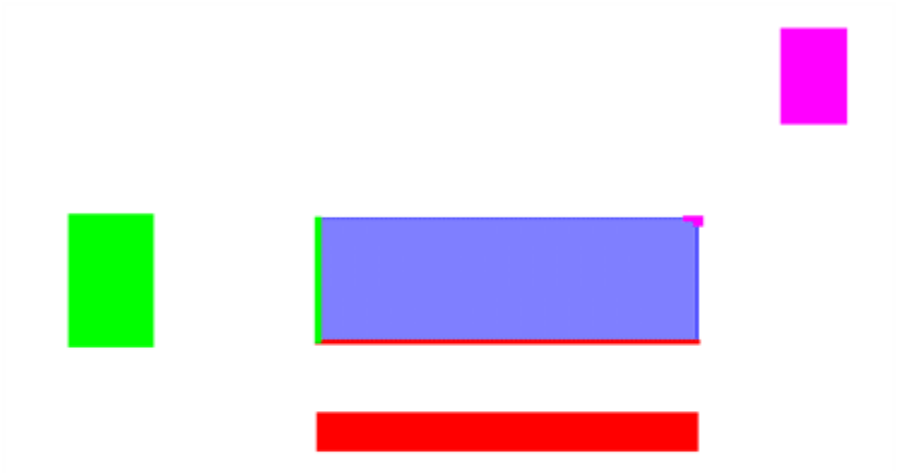
```
newtag AllBottomEdge southedge
newtag ShortBottomEdge (southedge) length <  0.1
newtag LongBottomEdge (southedge) length >= 0.1
newtag AllHorizontalEdge (northedge) OR AllBottomEdge
newtag AllHorizontalShortEdge AllHorizontalEdge length < 0.1

newtag bottomEdge southedge
newtag bottomLeftCorner southwestcorner
newtag LL_or OR bottomEdge bottomLeftCorner
newtag LL_and AND bottomEdge bottomLeftCorner
```

Newtag Orientation Example

This example shows the code and its topological results.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-10. Newtag Orientation



```
newtag ShortBottomEdge1 (southEdge) length <  0.1
newTag leftEdge WestEdge
newTag necorner northeastcorner

rectEdge   tag ShortBottomEdge1 width 0.008   extension 0.000 offset 0.015 prior 1
rectEdge   tag leftEdge width 0.010   extension 0.001 offset 0.020 prior 2
rectCorner tag necorner offsetX 0.01 offsetY 0.02 width 0.008 length 0.020 breaksym prior 3
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag pairtag

Tag Keywords

Specifies a tag name, exclusively for pairtemplate, that represents layers or edges for which to generate SRAFs. Used by RET SBAR only.

Usage

newtag ***name*** **pairtag**
distance *range*
[anchorlayer *name*]
[angle *range*]
[projection *range*]
[shieldinglayer *name*]
[shielding {line | none}]

Arguments

- ***name***

  A required name specifying the name of the tag.

- **pairtag**

  A required keyword invoking the pairtag tag. This tag may only be used for pairtemplate.

- **distance *range***

  A required keyword specifying the length of generated connectors. A connector is defined as the distance between the centers of shapes on the named anchor or target layer. The range must contain an upper bound. You specify the range in microns. In Figure 4-11, the target shapes fall within the range specified as greater than 30nm, but less than 80 nm.

Figure 4-11. The distance Keyword

```
newtag sraf pairtag distance >0.030 <0.080
```

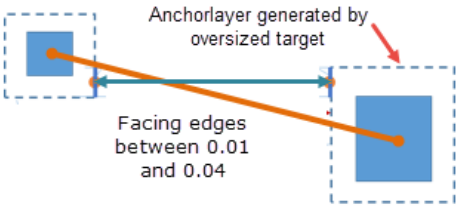
- anchorlayer *name*

  An optional keyword specifying the name of the anchor layer used to generate connectors. By default, the anchor layer shapes are formed from bounding boxes derived from polygons on the target layer. The anchorlayer must contain rectangle polygons only. If anchorlayer is specified, it is the only layer used during pairtag generation.

  You must declare the anchorlayer within the sraf_options block prior to specifying it in newtag pairtag. In Figure 4-12, the target shapes have been oversized and declared as an anchorlayer. In this example, the edges fall within the range specified by the projection keyword.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-12. An anchorlayer Application

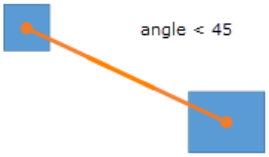



```
newtag sraf pairtag distance 0.080
   anchorlayer os_target projection > 0.010 <= 0.040
```

- angle *range*

  Specifies a range of values selecting the minimum angle between the connector and X-axis in degrees (irrespective of the angle's sign). You must specify a range between 0 and 90 degrees. These figures represent typical SRAF generation possible by pairtemplate seeded by the connectors generated by pairtag.

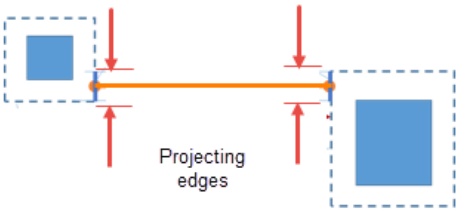
Figure 4-13. The angle Keyword



- projection *range*

  Specifies a range of mutual projection of rectangular bounding boxes on the anchorlayer. The connector is not generated if mutual projection of the corresponding rectangles is outside the specified range. You use projection less than 0 to generate connectors between rectangles of the anchorlayer which are not projected on each other. In Figure 4-14, the target layer has been oversized to provide for the anchorlayer. The anchorlayer edges provide an acceptable projection range.

Figure 4-14. The projection Keyword



- shieldinglayer *name*

  An optional name that specifies the name of the shielding layer. The shieldinglayer contains shapes used to hide potential paths of connectors. By default, the target layer is used as the shielding layer.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

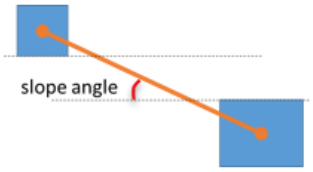
- shielding {line | none}

  An optional keyword that specifies the shielding mode. The line option allows connectors to be generated only if its segment does not intersect any shape on the shielding layer. This is the default behavior. None indicates that no shielding is performed.

Description

An optional keyword used to input tags by name to the pairtemplate template. The pairtag keyword generates connectors that extend from the centers of two target shapes or shapes on the anchorlayer. Connectors are qualified by their lengths, degree of angle from the X-axis, and local shielding. Figure 4-15 shows a connector, fixing the centers of neighboring shapes (within the specified constraint), ready to be passed to pairtemplate for SRAF processing. The connector has an associated slope and length.

Figure 4-15. Connector



Only manhattan- and 45-degree input layer geometries are supported by pairtag. No skew-edged geometries are valid target shapes for SRAFs.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times.

Examples

Example 1 — Usage with pairtemplate

```
newtag pt1 pairtag distance >=0.1 <0.3 projection >= 0.04
newtag pt2 pairtag distance >=0.1 <0.2 projection >= 0.02

pairtemplate tag pt1 width 0.040 length 0.040 offset 0.120
   angle 45 orient ortho prior 10
pairtemplate tag pt2 width 0.020 length 0.070 offset 0.085
   angle 0 orient ortho prior 20
```

Example 2 — Pairtag using angle.

```
#Generates connectors between shapes on the anchorlayer with distances
   between 100 and 150nm and an angle of less than 45 degrees

newtag c2ctag pairtag anchorlayer anchor_l distance >0.1 <0.15 angle <45
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 3 — Comprehensive Flow using shieldinglayer and anchorlayer.

```
pair_CN = RET SBAR target anchor_lay shield_lay FILE cn_setup MAP
pair_CNLITHO FILE cn_setup [
   layer target
   layer anchor_lay
   layer shield_lay
   tilemicrons 10

   sraf_options sraf_setup {
      cleanupversion 1
      anchorlayer anchor_lay
      shieldinglayer shield_lay
      maxconflict 0.03
      minoffset 0.01
      minfeaturespace 0.01
      centerthresh 0.03
      dynamicsizing 0.01

      newtag  short_pt pairtag distance > 0.01 <=0.300
         anchorlayer anchor_lay
         shieldinglayer shield_lay shielding Line
      pairtemplate tag short_pt width 0.017 length 0.050
         angle 0 orient normal prior 1
      priority 1 minfeaturespace 0.040 sbwidth 0.0175
         sraf 0.038 cleanmode postsizing
   }  setlayer pair_CN = cnsraf target OPTIONS sraf_setup

   setlayer 1_tags = cnsraf debug pair_CN tags 1
   setlayer 1_pre = cnsraf debug pair_CN preclean 1
   setlayer 1_conflict = cnsraf debug pair_CN conflict 1
   setlayer 1_processed = cnsraf debug pair_CN resolution 1
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag recta and rectb

Tag Keywords

Specifies a tag name representing all edges of rectangular target shapes matching the specified constraint. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name* recta *constraint* [rectb *constraint*]

Arguments

- *name*

  Specifies the name of the tag.

- recta *constraint*

  A keyword to select rectangular target shapes with at least one edge conforming to the specified constraint. Edge-based templates are applied to the recta edge only.

- rectb *constraint*

  An optional keyword name identifying orthogonal edges adjacent to edges selected with recta. The rectb keyword used in conjunction with recta to qualify edges of interest in rectangular target shapes. Only the edges selected by recta are applied to the edge-based template.

Note

When applying the recta and rectb keywords to any pitch-based template, you must provide an upper bound to their constraints to enable limits during edge collection.

Description

The recta and rectb tags select the edge of interest, and any adjacent orthogonal edges, respectively. The selected edges of rectangular target shapes must satisfy the constraints of the specified edge lengths. No consideration of orientation of the rectangle is made. The constraints may define a lower and an upper bound on the length of the long and short edges of the rectangles. The two sets of constraints for the long and short edges may overlap.

Only manhattan- and 45-degree input layer geometries are supported. No skew-edged geometries are valid target shapes for SRAFs.

When using tagging, positive SRAFs may be generated parallel and perpendicular (given template constraints) relative to tagged edges and negative SRAFs are generated only parallel to tagged edges. See the example Figure 4-115 on page 263 for more information.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect. In addition, if geometries extend past tile boundaries, recta and rectb cannot determine the rectilinearity of those portions of the geometry extending beyond the tile boundary.

Examples

Example 1 — recta Usage

```
newtag recta_edge recta > 0.1 <= 0.4
```

Example 2 — recta and rectb Usage

In this figure, the edge named recta_edge is selected by recta, but is first filtered to meet qualification of the edges named orthogonal_edge, selected by rectb.

Figure 4-16. Edge Selection With recta and rectb

```
newtag recta_edge recta > 0.1 <= 0.4 rectb > 1 < 2
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

newtag square

Tag Keywords

Specifies a tag name representing the edges of square target shapes matching the constraint. Used by RET SBAR and RET MBSRAF.

Usage

newtag *name* square *constraint*

Arguments

- *name*

  Specifies the name of the tag.

- square *constraint*

  A keyword that specifies the constraint. For edge-based templates, the constraint may be a range.

Note

When applying the square keyword to any pitch-based template, you must provide an upper bound to the constraint to enable limits during edge collection.

Description

The square tag selects all edges of square target shapes matching the specified edge length constraint.

Caution

Using constraint values that exceed cell or tile boundaries may cause flattening and increase run times. Siemens EDA recommends using the recta and rectb newtag types on relatively shorter geometries such as contacts, vias and local interconnect.

Examples

Example 1 — recta and rectb Usage

```
newtag sq square == 0.2
newtag smallSquares square >= 0.090 < 0.120
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Constraint Keywords

Calibre nmSRAF requires the specification of some minimum SRAF dimensions for application of MRC rules.

For organizational keyword information, refer to "sraf_options" on page 80.

Table 4-9. Constraint Keywords

Keyword	Description
alignorigin	Interprets canonical cell and tile origins in alignment with global database origins. Used by RET SBAR and RET MBSRAF.
extension	Specifies SRAF extension length. Used by RET MBSRAF only.
maxlength	Specifies the maximum length of generated SRAFs, and the combined maximum length of generated SRAFs of same priorities during cleanup. Used by RET SBAR and RET MBSRAF.
maxoffset	Specifies the maximum distance from an SRAF to a target shape. Used by RET MBSRAF only.
maxoffsetnegative	Specifies the maximum distance from a negative SRAF to the enclosing target shape edge. Used by RET MBSRAF only.
minarea	Specifies the minimum area for SRAFs. Used by RET SBAR and RET MBSRAF.
minareaextension	Specifies the minimum area for 45-degree SRAF extensions. Used by RET SBAR and RET MBSRAF.
minfeaturespace	Specifies the minimum distance from target shapes to an SRAF. Used by RET SBAR and RET MBSRAF.
minlength	Specifies the minimum length allowed for a rectangular SRAF. Used by RET SBAR and RET MBSRAF.
minlengthextension	Specifies the minimum length for 45-degree SRAF extensions. Used by RET SBAR and RET MBSRAF.
minoffset	Specifies the minimum distances between generated SRAFs. Used by RET SBAR and RET MBSRAF.
minoffsetlayerspace	Specifies the minimum distance to SRAF shapes from the offsetlayer. Used by RET SBAR and RET MBSRAF.
minsquarearea	Specifies the minimum area of a square SRAF. Used by RET SBAR and RET MBSRAF.
minsquarelength	Specifies the minimum edge length for square SRAFs. Used by RET SBAR and RET MBSRAF.
minwidth	Specifies the minimum width allowed for square SRAFs. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 4-9. Constraint Keywords (cont.)

Keyword	Description
shielding	Uses edge tag shielding to characterize SRAF generation. Used by RET MBSRAF only.
squarewidth	Specifies the width of square SRAFs. Used by RET MBSRAF only.
width	Specifies the default width of generated SRAFs. Used by RET MBSRAF only.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

alignorigin

Constraint Keywords

Interprets canonical cell and tile origins in alignment with global database origins. Used by RET SBAR and RET MBSRAF.

Usage

alignorigin {0 | 1}

Arguments

- 0 | 1

  Disables alignorigin. The default is 1 (enabled).

Description

The origin of canonical representations of cells and tiles are aligned by default to the database global origins to reduce tile-to-tile variability of grid values.

Note

The alignorigin keyword was disabled by default in releases previous to 2014.1.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

extension

Constraint Keywords

Specifies SRAF extension length. Used by RET MBSRAF only.

Usage

extension *value*

Arguments

- *value*

  A positive or negative floating point number specified in microns.

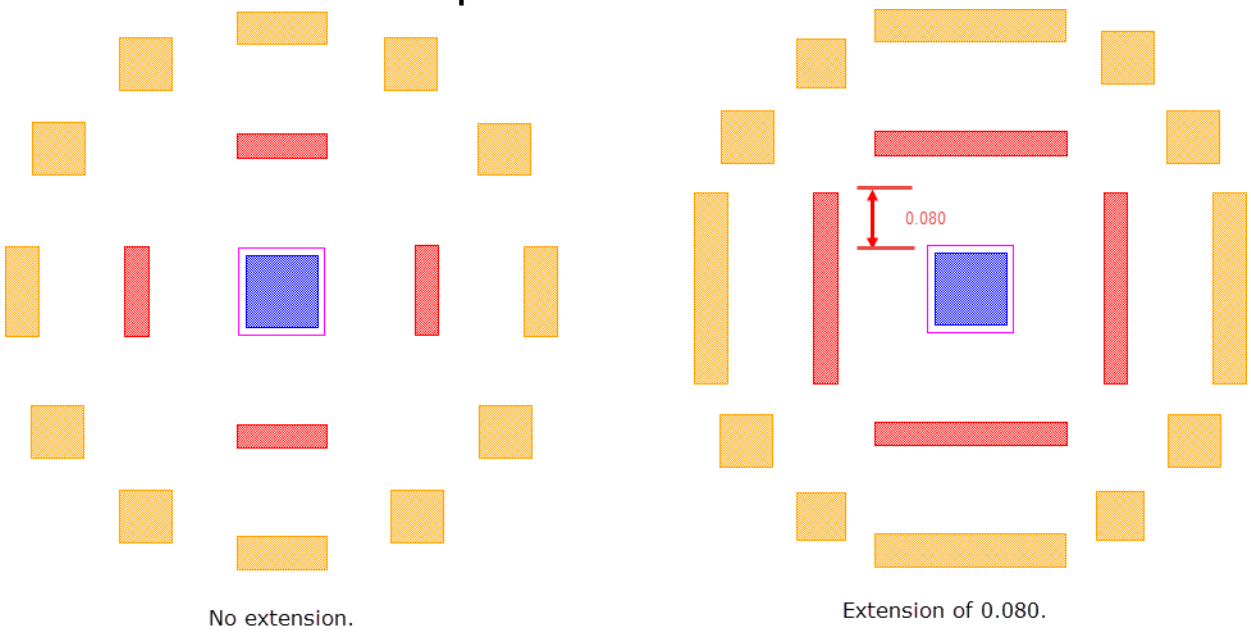
Description

An optional keyword that adds to or subtracts from the length of a rectangular SRAF end by the specified value based on the neighboring target edge. The extension keyword works only with the localmaxgradient modes nearest and nearestno45. A negative extension reduces SRAF length if the extension is less than half of the target edge length and the resulting SRAF does not violate MRC constraints. The extension keyword may be specified globally or by priority.

Examples

This figure shows the effects of using extension.

Example 4-8. Effects of extension



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

maxlength

Constraint Keywords

Specifies the maximum length of generated SRAFs, and the combined maximum length of generated SRAFs of same priorities during cleanup. Used by RET SBAR and RET MBSRAF.

Usage

maxlength *value*

Arguments

- *value*

  A required positive floating point number. The value is the maximum length of generated SRAFs, or the combined length of generated SRAFs of the same priority processed by intrapriorityalign during cleanup.

Description

An optional keyword that measures and splits generated SRAFs to a maximum of the specified length. The maxlength keyword may only be specified globally. You may use maxlength in conjunction with intrapriorityalign to limit the combined length of aligned and merged SRAFs. The maxlength keyword may be applied to positive and negative SRAF generation.

You may use maxlength for the following templates:

- localmaxgradient
- rect
- rectcorner
- rectdiag
- rectedge
- recthv

Examples

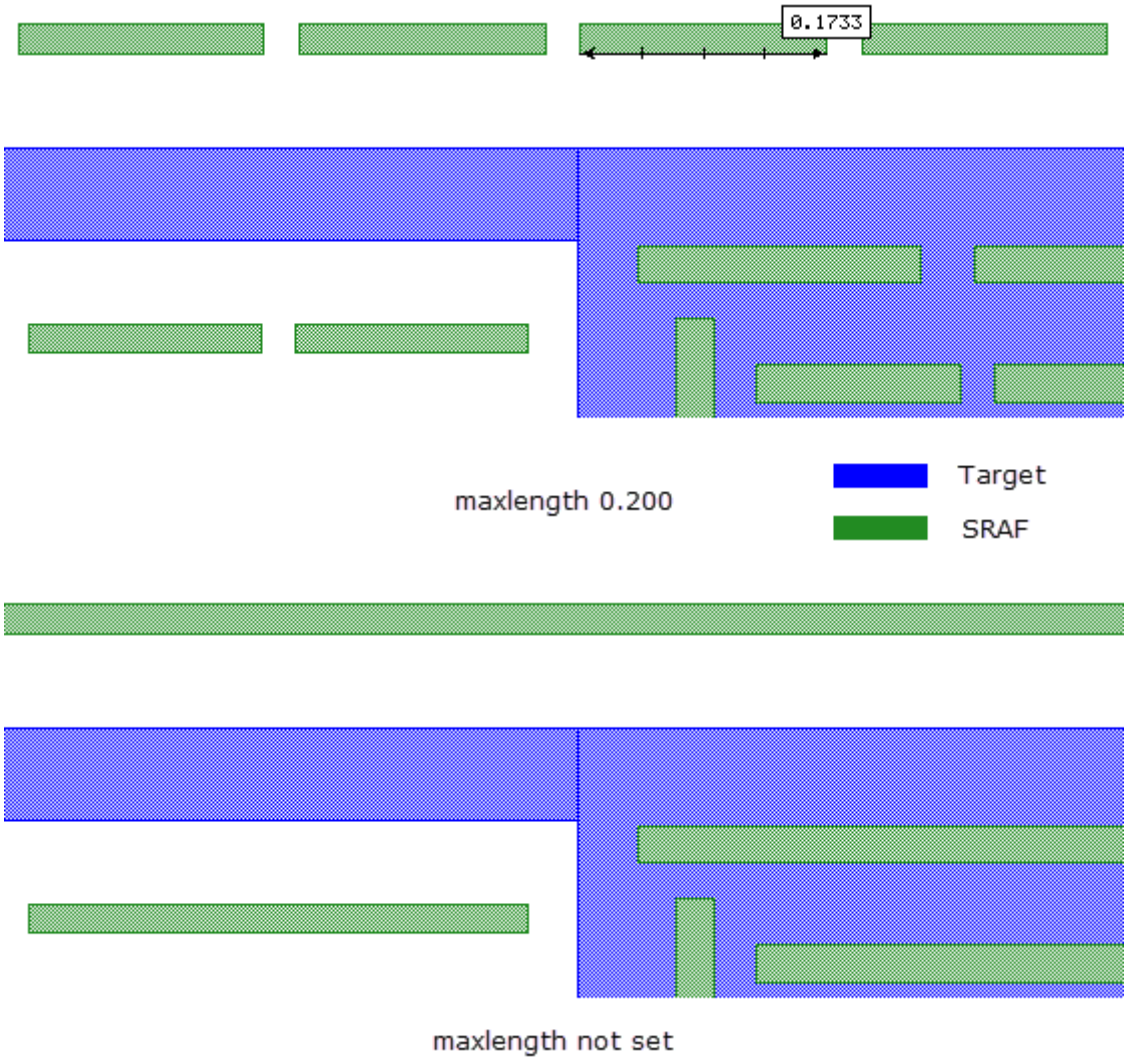
Example 1 — maxlength Code and Results

In the first illustration of Figure 4-17, SRAFs generated by the localmaxgradient template specifying prior 1 generate positive SRAFs and prior 2 generates negative SRAFs. The maxlength keyword as specified splits continuous SRAFs into segments evenly divided into lengths less than or equal to the maxlength value. The second illustration shows SRAF generation without maxlength.

```
maxlength 0.200
localmaxgradient width 0.020 squarewidth 0.025 prior 1
localmaxgradient width 0.026 squarewidth 0.045 prior 2
priority 2 negative
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-17. maxlength



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

maxoffset

Constraint Keywords

Specifies the maximum distance from an SRAF to a target shape. Used by RET MBSRAF only.

Usage

maxoffset *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the maximum distance from target shapes that SRAFs can be placed. The maxoffset keyword may be specified only once in a LITHO FILE, for either priority or global assignment, and must be specified before all cnsraf or mbsraf templates.

Note

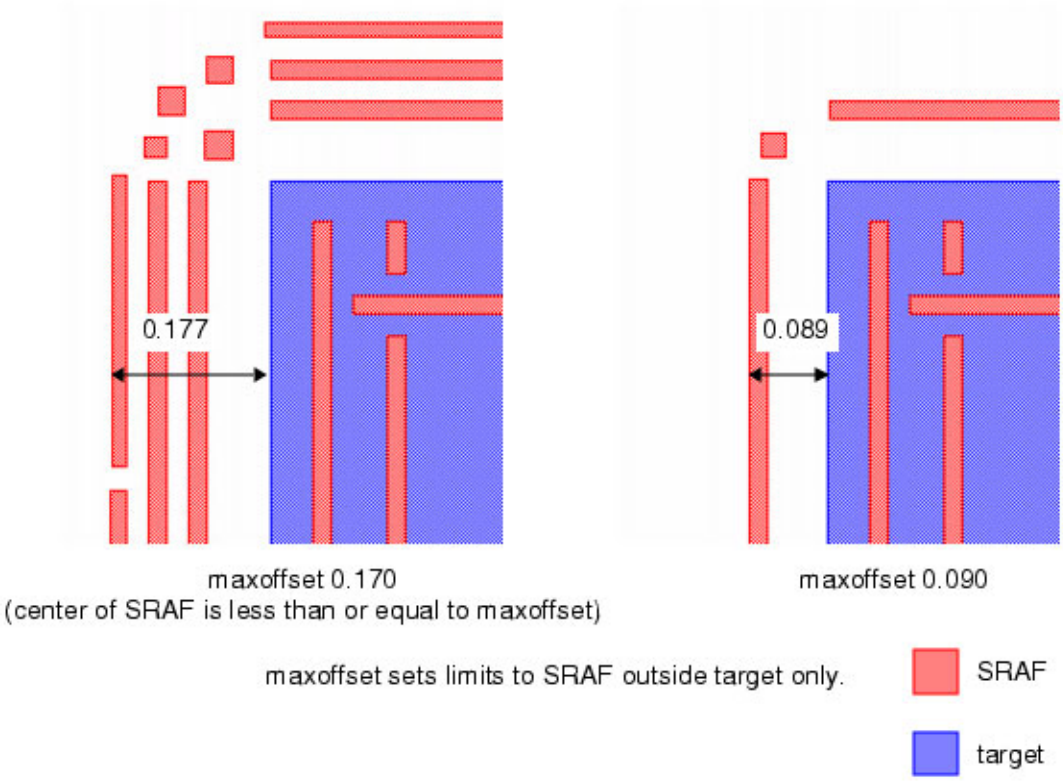
For recipes requiring placement of varying SRAF widths by multiple localmaxgradient templates, Siemens EDA recommends against using the maxoffset keyword.

Examples

Figure 4-18 illustrates the results establishing limits with maxoffset.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-18. SRAF Limits with maxoffset



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

maxoffsetnegative

Style Keywords

Specifies the maximum distance from a negative SRAF to the enclosing target shape edge. Used by RET MBSRAF only.

Usage

maxoffsetnegative *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

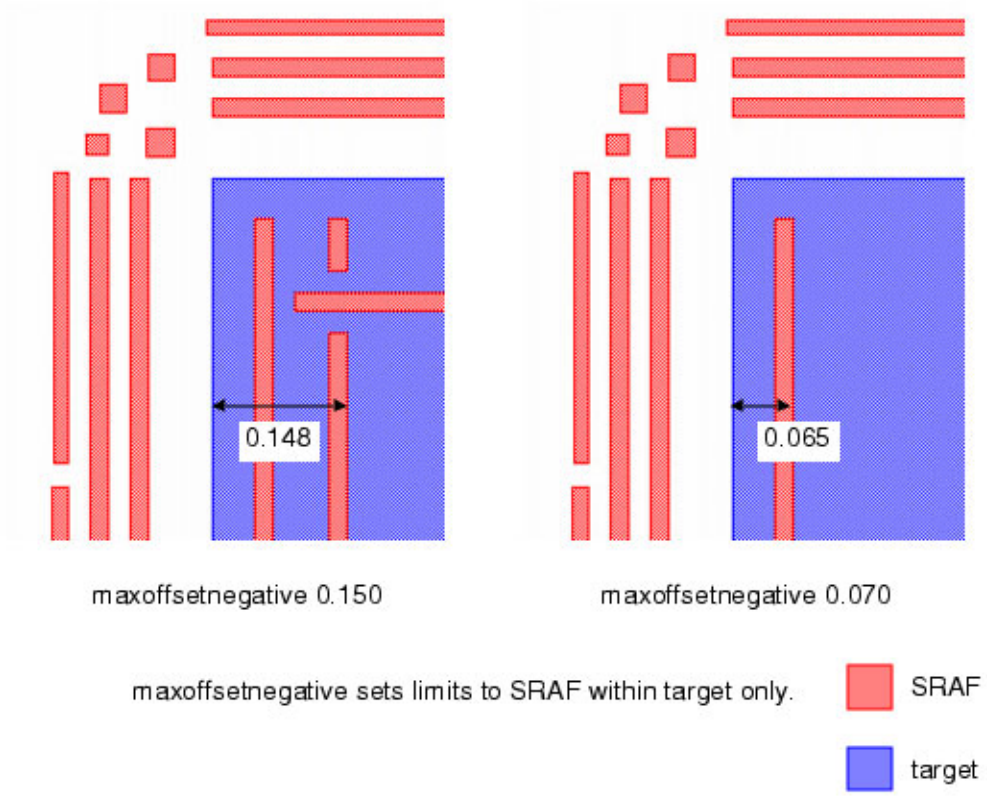
An optional keyword that specifies the maximum distance allowed to generate negative SRAFs (slots) within a target shape. The maxoffsetnegative keyword is used only with the negative keyword. The maxoffsetnegative keyword may be specified only once in a LITHO FILE, for either priority or global assignment.

Examples

Figure 4-19 illustrates the results establishing limits with maxoffsetnegative.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-19. SRAF Limits with maxoffsetnegative



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minarea

Constraint Keywords

Specifies the minimum area for SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

minarea *value*

Arguments

- *value*

  A floating point number specified in square microns. The minarea keyword may be specified globally or by priority. The default value is 0.

Description

An optional keyword that specifies the minimum area allowed for any SRAF. SRAFs with an area smaller than minarea are eliminated.

Although minarea applies to all SRAFs, if minsquarearea or minsquarelength are specified, all square SRAFs are checked against these two parameters but not minarea.

For information regarding interactions between minwidth, minarea and minlength, see "minarea, minlength, and minwidth Keyword Interaction" on page 325.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minareaextension

Constraint Keywords

Specifies the minimum area for 45-degree SRAF extensions. Used by RET SBAR and RET MBSRAF.

Usage

minareaextension *value*

Arguments

- *value*

  A floating point number specified in square microns. The minareaextension keyword may be specified globally or by priority. The default value is 0.

Description

An optional keyword that specifies the minimum area allowed for any 45-degree SRAF extension. 45-degree SRAF extensions with an area smaller than minareaextension are eliminated during cleanup.

For information regarding minareaextension interactions with extension45, see “rectedge” on page 215.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minfeaturespace

Constraint Keywords

Specifies the minimum distance from target shapes to an SRAF. Used by RET SBAR and RET MBSRAF.

Usage

minfeaturespace *value*
[lineend *value*
[lemaxadjacent *value*]
[lemaxlength *value*]
[leminadjacent *value*]
[leminlength *value*]
[leprojectionin *value*]
[leprojectionside *value*]
]
[projecting *value*]
[lineend2long *value*]
[long2lineend *value*]
[corner2sqcorner *value*]
[lineend2sqedge *value*]
[long2sqedge *value*]
[sqcorner2corner *value*]
[sqcorner2sqcorner *value*]
[sqedge2lineend *value*]
[sqedge2long *value*]
[sqedge2sqedge *value*]

Arguments

- ***value***

  A required positive floating point number specifying the basic minimum distance from target shapes to SRAFs. If not specified, minfeaturespace defaults to minoffset. The minfeaturespace command enforces a distance based on a fraction of the optical pitch if the specified value is smaller than the calculated value. For recipes using algorithm 1, no internally calculated value is used; only explicitly specified values are enforced.

- lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the line-end of a target shape to any SRAF. The lineend distance is measured from the edge declared as the line-end of a rectangular target shape. Square target shapes are not considered by the lineend keyword. The lineend keyword forms a shape that acts as a keepout area for subsequent OPC generation. The lineend value must be greater than or equal to the minfeaturespace value. The default value for lineend is equal to the value specified for minfeaturespace.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Additional line-end keywords are available to refine the qualifying criteria of line-ends and line-end keepout areas. To use these keywords you must first specify lineend:

leminlength *value* — Specifies the lower limit of line-end length. Defaults to 0.

lemaxlength *value* — Specifies the upper limit of line-end length. Defaults to 150 nm.

leminadjacent *value* — Qualifies line-ends by the shortest leg sharing a convex corner of the line-end. This is the lower limit of the leg length. Defaults to lemaxlength.

lemaxadjacent *value* — Qualifies line-ends by the longest leg sharing a convex corner of the line-end. This is the upper limit of the leg length. There is no default.

leprojectionin *value* — Extends the keepout area into the line-end by the value specified. Defaults to 0.

leprojectionside *value* — Extends the keepout area from the sides of the line-end by the value specified. Defaults to 0.

- projecting *value*

  An optional keyword and positive floating point number that specifies the minimum side projection distance measured along the long edge of the target shape. In cases of square target shapes, projection is measured on all edges. In cases of angled target edges, projection is measured asymmetrically, from target shapes toward SRAFs, and symmetrically between SRAFs. The value of projecting must be greater than or equal to the value of minfeaturespace, and the projecting value may be greater than or equal to the specified lineend value. The projection default value is equal to the value specified for minfeaturespace.

- lineend2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the long side of the target shape to the line-end of an SRAF. The measurement is performed during SRAF cleanup. The lineend2long default value is equal to the value specified for projecting. The behavior is depicted in Figure 4-21.

- long2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the target shape line-end to the long side of the an SRAF. The measurement is performed during SRAF cleanup. The long2lineend default value is equal to the value specified for lineend. The behavior is depicted in Figure 4-22.

- corner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of an SRAF to the corner of a square target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- lineend2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the line-end of an SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-28.

- long2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the longer edge of an SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-28.

- sqcorner2corner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of a target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-29.

- sqcorner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of a square target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

- sqedge2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the line-end of a target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-29.

- sqedge2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the longer edge of a target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-29.

- sqedge2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

MRC parameters such as lineend2long, long2lineend, square2long, etc., must be specified by priority for minfeaturespace. Do not define these conditional parameters globally. Although parsed as valid, global specification of these parameters degrades performance.

Description

An optional keyword that specifies the minimum distance SRAFs are placed from target shapes. You can specify minfeaturespace by priority or globally. All global keywords must be specified first.

Line-End Descriptions

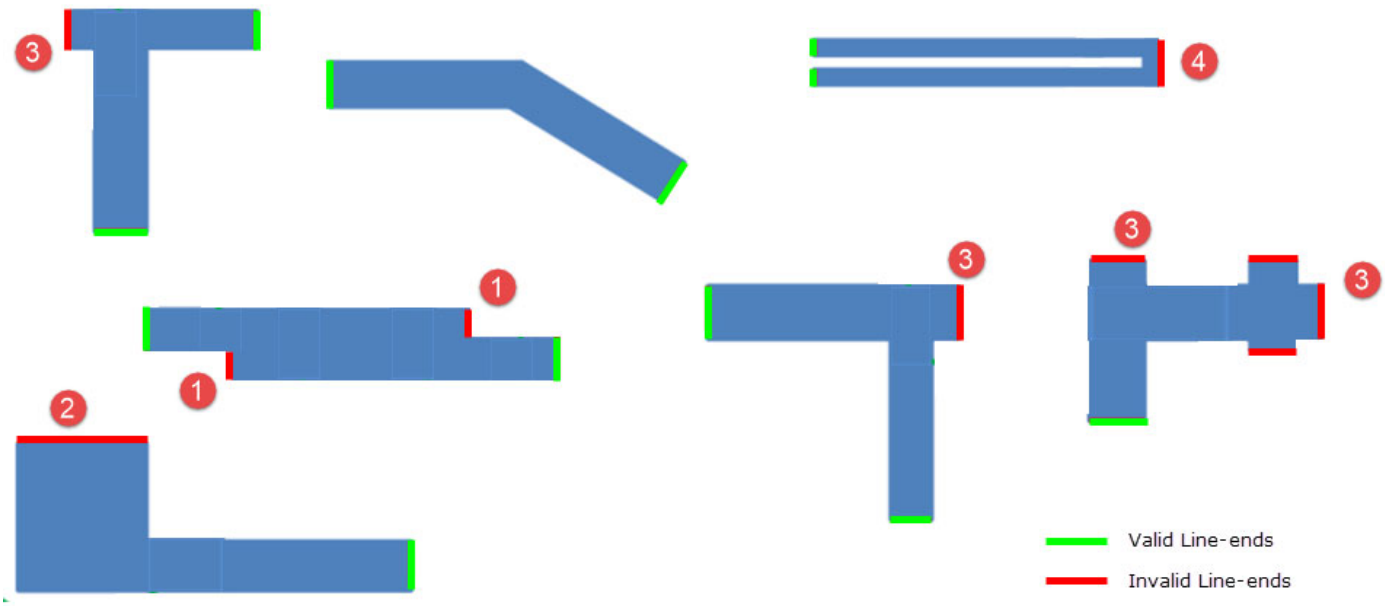
Line-end criteria are specified by the minfeaturespace lineend keyword. Line-ends can be varied depending on their user definition. The line-end length, line-end leg length and adjacent vertices all become criteria for the selection of line-ends for SRAF generation.

The following criteria applies to the various line-end forms depicted in Figure 4-20, where green line-ends are valid and red line-ends are invalid. The default dimension is minoffset. The line-end includes specific criteria:

- **Convex Corners —** An edge series having only two convex corners. The edges labeled 1 do not have two convex corners and are invalid line-ends.
- **Line-End Length —** The line-end must have a length between the constraints specified by the leminlength and lemaxlength keywords. The edge labeled 2 exceeds lemaxlength and is an invalid line-end.
- **Line-End Legs —** The line-end must have two adjacent perpendicular edges (legs), with both lengths between the constraints specified by the leminadjacent and lemaxadjacent keywords. Other perpendicular edges within the distance of the specified leg length invalidates the line-end. The edges labeled 3 do not exceed leminadjacent for both legs and are invalid line-ends.
- **Serpentine Shapes —** External edges that are part of any serpentine or U-shaped target, or those of rectangles with holes are invalid line-ends. The edge labeled 4 exceeds lemaxlength and is an invalid line-end.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-20. Valid and Invalid Line-ends



Conditional MRC

Conditional cleanup enforces MRCs that act conditionally for minfeaturespace. The conditional enforcement of MRCs are applied to a subset of tagged target edges and target shapes defined by specified values for a given template. The conditional values override global minfeaturespace values, normally enforced for all target shapes. The condition applies to the target shapes that are found near local SRAFs. During cleanup, the MRC values are applied to a pair of node and contender SRAFs. Where contending SRAFs interact with the tag or meet the specified constraint, the conditional values in the priority statement are enforced.

This code snippet shows the same priority using two separate conditional methods, one using tags of wide width shapes, shown in red, and the other using tags of standard width shapes, shown in green:

```
newtag wide internal > 0.1
newtag std internal <= 0.1

minfeaturespace 0.030 projecting 0.050
minoffset 0.002 projecting .040
bmaxconflict 0.01 centerthresh 0.15 sbwidth 0.03 sraf 0.05

priority 1 tag wide
   minfeaturespace 0.008 projecting 0.08 lineend 0.07 lineend2long 0.04
priority 1 tag std
   minfeaturespace 0.01 projecting 0.08 lineend 0.07 lineend2long 0.04
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1 - lineend2long

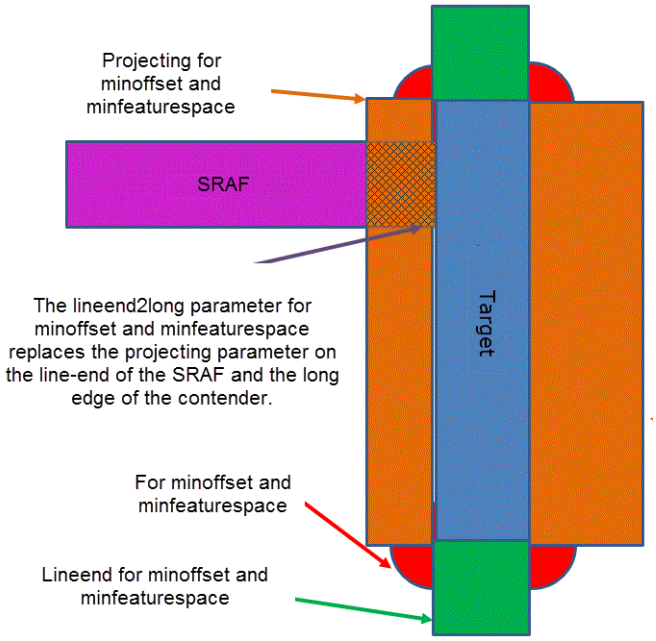
For lineend2long, the parameters define the protection at the line-end of the SRAF and the long edge of the contender during SRAF cleanup. This code:

```
priority 2 minoffset 0.01 lineend2long 0.035
```

implies these defaults:

```
minoffset 0.01 projecting 0.01 lineend2long 0.035 lineend 0.01 long2lineend 0.01
```

Figure 4-21. Protection Dimensions Relative to lineend2long



Example 2 — long2lineend

For long2lineend, the parameters define the protection at the long edge of the SRAF and the line-end of the contender during SRAF cleanup. This code:

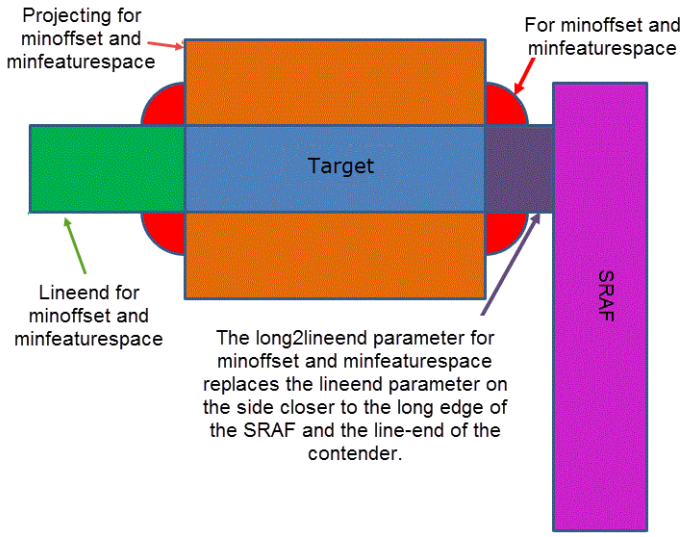
```
priority 2 minoffset 0.01 long2lineend 0.035
```

implies these defaults:

```
minoffset 0.01 projecting 0.01 lineend2long 0.01 lineend 0.01 long2lineend 0.035
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

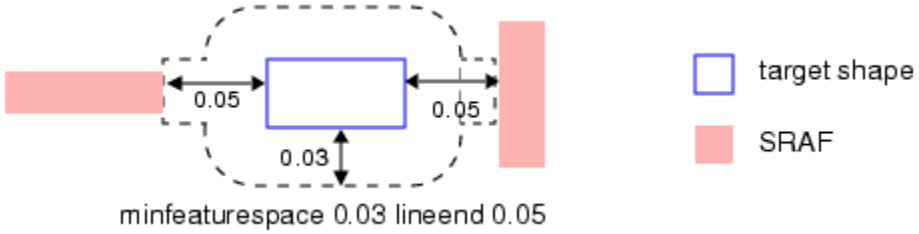
Figure 4-22. Protection Dimensions Relative to long2lineend



Example 3

You specify a distance used only for the line-end projected direction of target shapes to SRAFs with the lineend keyword. SRAFs within the dashed perimeter are cleaned.

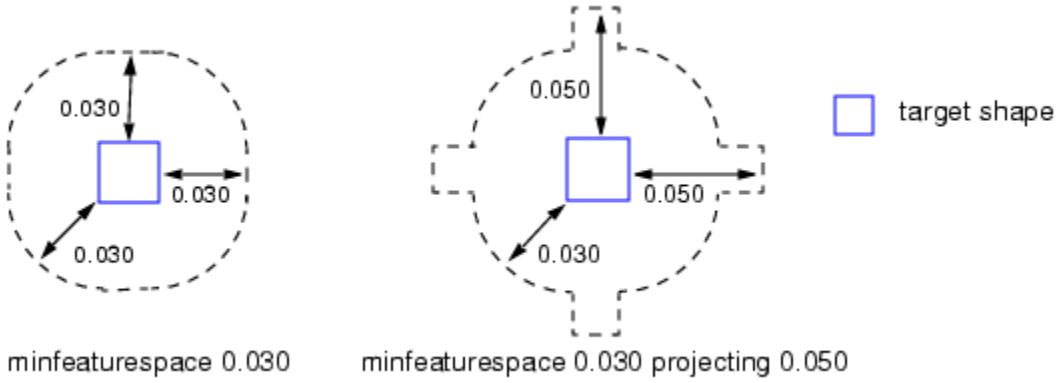
Figure 4-23. Using minfeaturespace and lineend



Example 4

The projecting keyword allows you to specify a different distance in the horizontal and vertical projections of the target shape.

Figure 4-24. Using minfeaturespace With and Without projecting

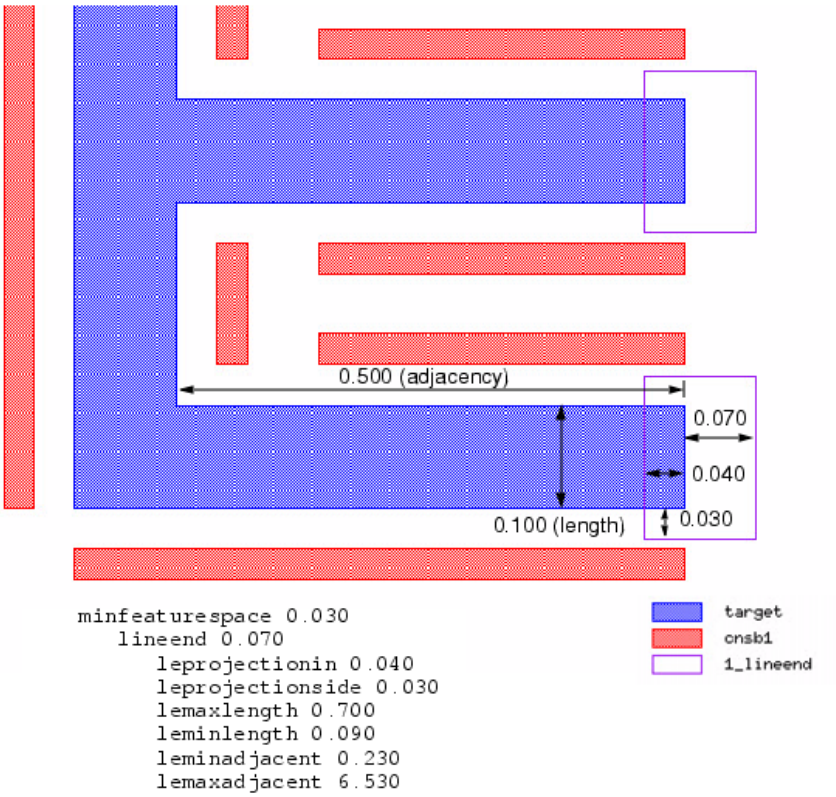


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 5

In this figure, the output layer representing the line-end keepout area is 1_lineend. The pictured line-end was qualified because it had a line-end edge that fell between 0.090 (leminlength) and 0.700 (lemaxlength). It was then further qualified because its legs were between 0.230 (leminadjacent) and 6.530 (lemaxadjacent). The SRAF layer is cnsb1. The line-end keepout layer was saved using the lineend debug keyword. See setlayer cnsraf debug for more detail regarding saving the lineend keepout layer.

Figure 4-25. Application of lineend Keyword Options

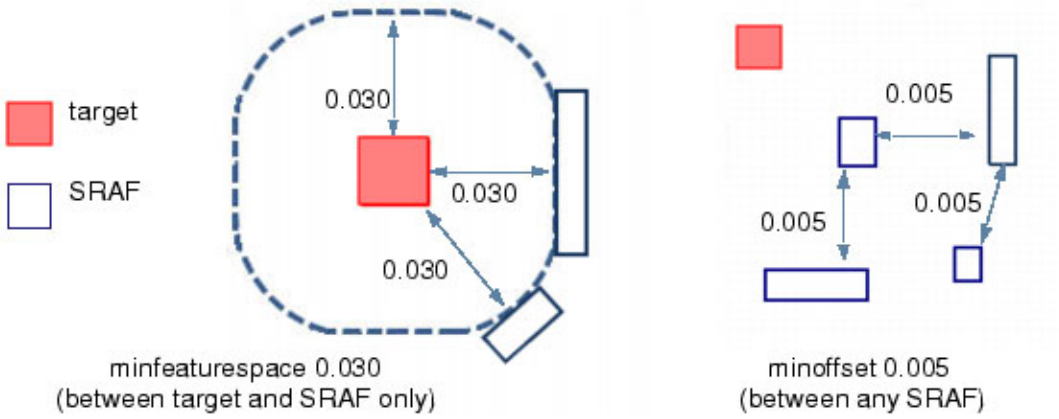


Example 6

Figure 4-26 illustrates one application of minfeaturespace and minoffset.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

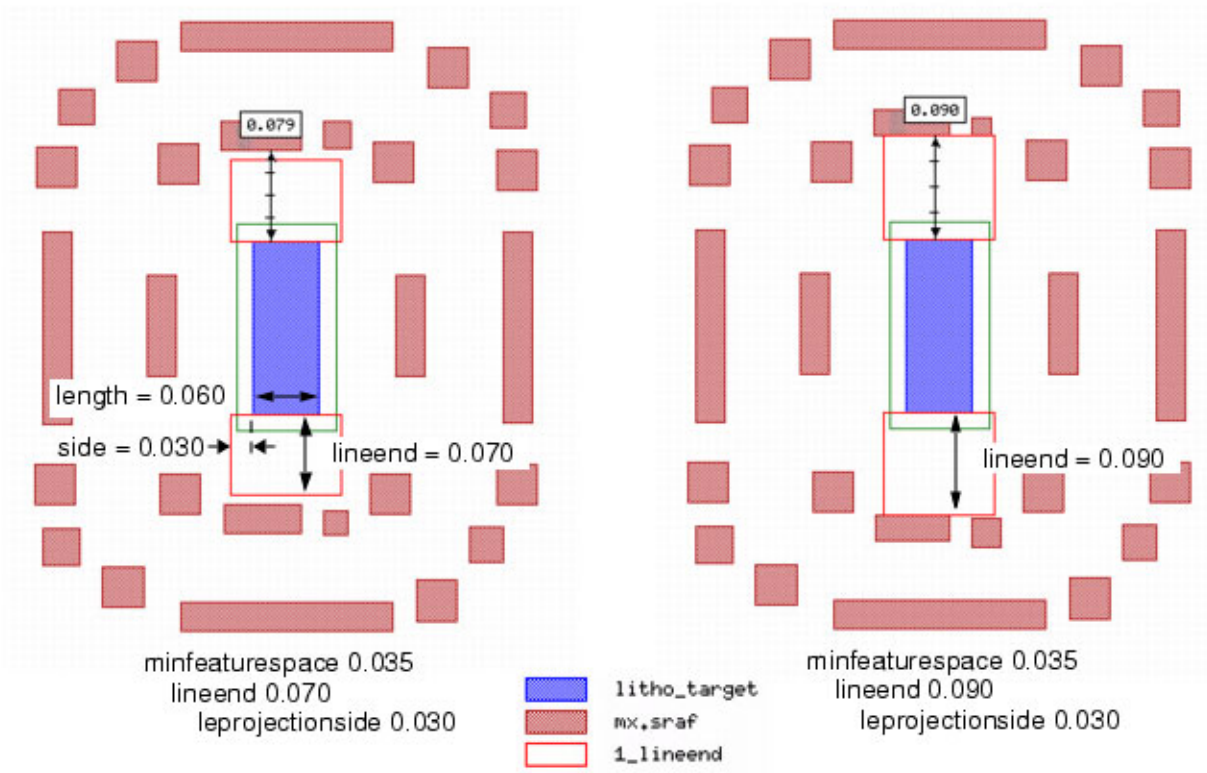
Figure 4-26. Using minfeaturespace and minoffset with mbsraf



Example 7

Figure 4-27 illustrates the lineend keyword applied to mbsraf. The lineend value is 70nm in the left picture and does not affect SRAFs. Although lineend is extended 20nm in the right picture, the SRAF still prints but is adjusted until butting the lineend keepout. The pictured line-end qualifies because it is less than the default lemaxlength of 150nm. The line-end keepout layer (1_lineend) is saved using setlayer mbsraf debug.

Figure 4-27. Application of lineend Keyword with mbsraf



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 8

Figure 4-28. Square-Oriented Option Usage with Square Target

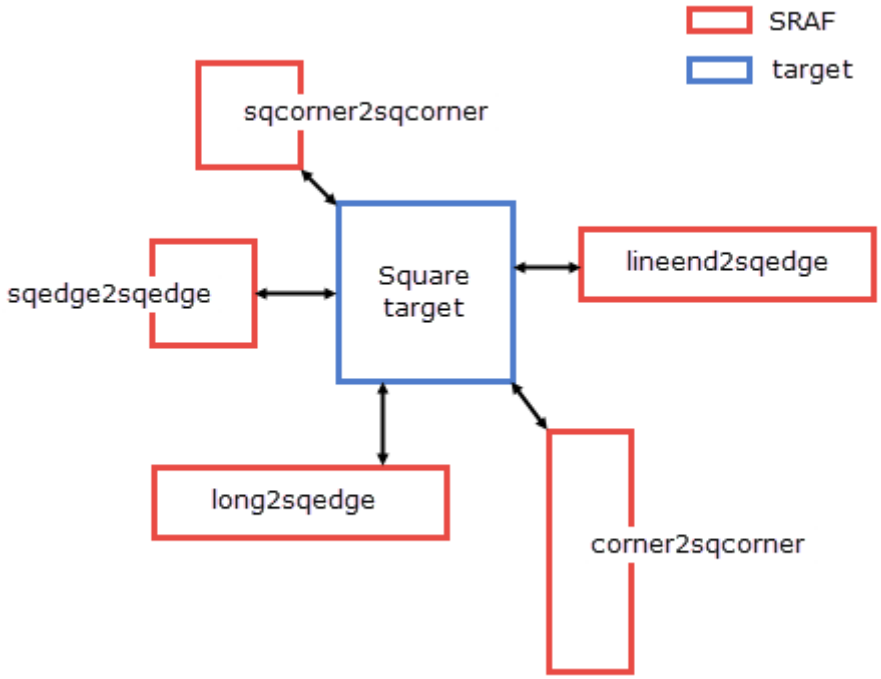
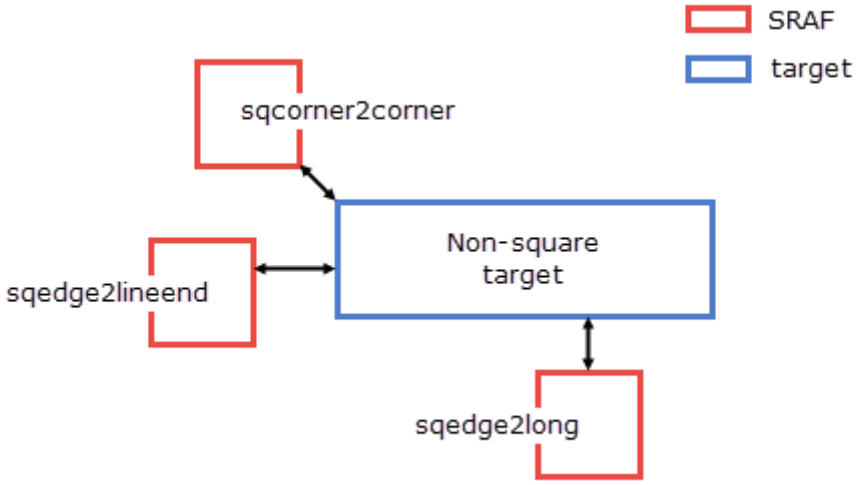


Figure 4-28 illustrates the minimum distance measurement behavior of all keywords involving square target shapes and square or non-square SRAFs.

Example 9

Figure 4-29. Square-Oriented Option Usage with Square SRAFs and Non-square Target



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-29 illustrates the minimum distance measurement behavior of all keywords involving square SRAFs and non-square target shapes.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minlength

Constraint Keywords

Specifies the minimum length allowed for a rectangular SRAF. Used by RET SBAR and RET MBSRAF.

Usage

minlength *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the minimum length allowed for a rectangular SRAF. SRAFs with any side of a length smaller than minlength are eliminated after cleanup is performed. You can specify minlength globally or by priority.

For information regarding interactions between minwidth, minarea and minlength, see "minarea, minlength, and minwidth Keyword Interaction" on page 325.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minlengthextension

Constraint Keywords

Specifies the minimum length for 45-degree SRAF extensions. Used by RET SBAR and RET MBSRAF.

Usage

minlengthextension *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the minimum length allowed for a 45-degree SRAF extension. All 45-degree SRAF extensions with a side length smaller than minlengthextension are eliminated during cleanup. You can specify minlengthextension globally or by priority.

For information regarding minlengthextension interactions with extension45, see “rectedge” on page 215.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minoffset

Constraint Keywords

Specifies the minimum distances between generated SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

minoffset *value*
[lineend *value*]
[projecting *value*]
[lineend2long *value*]
[long2lineend *value*]
[corner2sqcorner *value*]
[lineend2sqedge *value*]
[long2sqedge *value*]
[sqcorner2corner *value*]
[sqcorner2sqcorner *value*]
[sqedge2lineend *value*]
[sqedge2long *value*]
[sqedge2sqedge *value*]

Arguments

- ***value***

  A required positive floating point number that specifies the minimum distance allowed between SRAFs.

- lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance an SRAF is placed from another SRAF. The projection is measured from the short edge of the rectangular SRAF. Line-end distance is measured between SRAFs. The lineend value must be greater than or equal to minoffset. Square SRAFs have no line-end projection.

- projecting *value*

  An optional keyword and positive floating point number that specifies the minimum distance allowed between projections of SRAFs. The projecting distance is measured orthogonally to the edges of the target shapes. SRAF to SRAF projecting distances are enforced symmetrically. The projecting value may be greater than or equal to the specified lineend value, and must be greater than or equal to minoffset.

- lineend2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the long side of the target shape to the line-end of an SRAF. If lineend2long and long2lineend are uniquely specified, the lesser of the two arguments is enforced and a warning message is issued to the transcript. The measurement is performed during SRAF

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-21 on page 145.

- long2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the target shape line-end to the long side of the an SRAF. If lineend2long and long2lineend are uniquely specified, the lesser of the two arguments is enforced and a warning message is issued to the transcript. The measurement is performed during SRAF cleanup. The default is the specified lineend value. The behavior is depicted in Figure 4-22 on page 146.

- corner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of an SRAF to the corner of a square target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-32.

- lineend2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the line-end of an SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-32.

- long2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the longer edge of an SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-32.

- sqcorner2corner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of a target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-33.

- sqcorner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of a square target shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-32.

- sqedge2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the line-end of a target shape. The measurement

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-33.

- sqedge2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the longer edge of a target shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-33.

- sqedge2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the edge of a square target shape. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-32.

Note

Conditional MRC parameters such as lineend2long, long2lineend, square2long, etc., must be specified by priority only for minoffset. Do not define these conditional parameters globally. Although parsed as valid, global specification of these parameters degrades performance.

Description

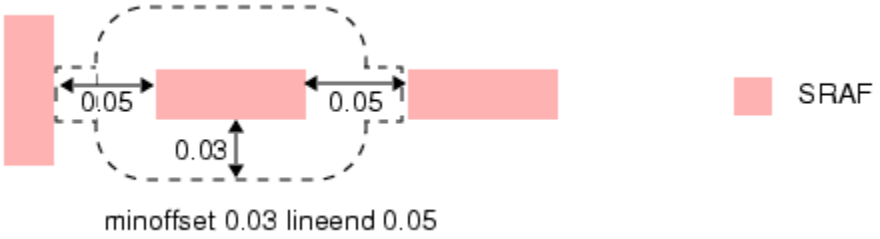
A required keyword that specifies the minimum distance allowed between all SRAF shapes. Any target shapes that fall within this distance are treated as conflicts and are processed according to the elimination and secondary bar rules. This keyword can be set globally or by priority using the priority keyword.

Examples

Example 1

Figure 4-30 illustrates minoffset behavior with the lineend keyword. The lineend keyword allows you to specify a different distance from the line-end projected directions of the SRAF to another SRAF. SRAFs within the dashed perimeter are cleaned.

Figure 4-30. Using minoffset and lineend

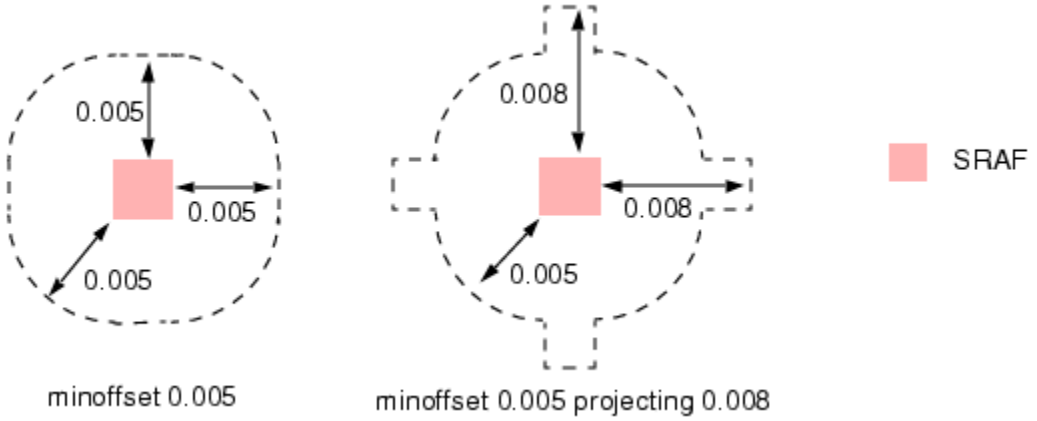


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 2

Figure 4-31 illustrates the minimum distance with and without the projecting keyword. The projecting keyword allows you to specify a different distance in the horizontal and vertical projections of the target shape. SRAFs within the dashed perimeter are cleaned.

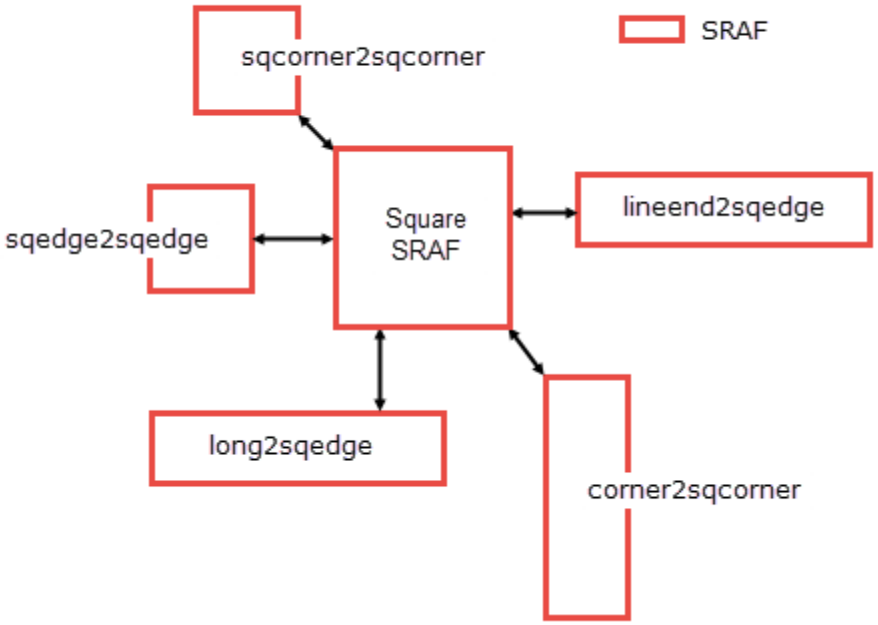
Figure 4-31. Using minoffset With and Without projecting



Example 3

Figure 4-32 illustrates the minimum distance measurement behavior of all keywords involving square SRAFs to square or non-square SRAFs.

Figure 4-32. Square-Oriented Option Usage with Square SRAF

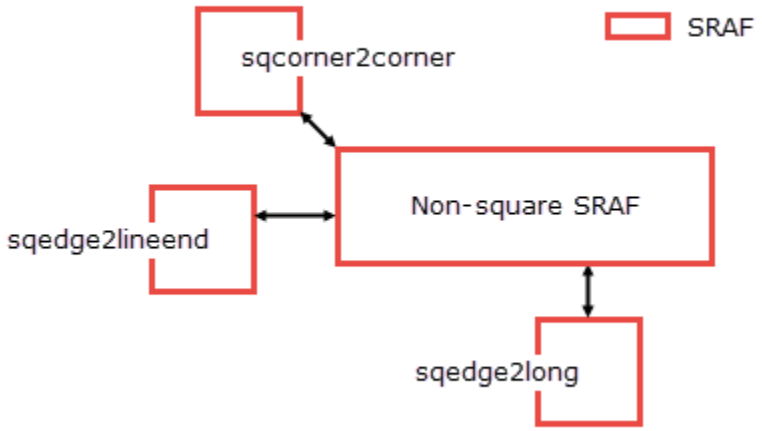


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4

Figure 4-33 illustrates the minimum distance measurement behavior of all keywords involving square SRAFs and non-square SRAFs.

Figure 4-33. Square-Oriented Option Usage with Square SRAFs



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minoffsetlayerspace

Constraint Keywords

Specifies the minimum distance to SRAF shapes from the offsetlayer. Used by RET SBAR and RET MBSRAF.

Usage

minoffsetlayerspace *value*
[lineend *value*]
[projecting *value*]
[lineend2long *value*]
[long2lineend *value*]
[corner2sqcorner *value*]
[lineend2sqedge *value*]
[long2sqedge *value*]
[sqcorner2corner *value*]
[sqcorner2sqcorner *value*]
[sqedge2lineend *value*]
[sqedge2long *value*]
[sqedge2sqedge *value*]

Arguments

- ***value***

  A required positive floating point number specified in microns.

- lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance an SRAF is placed from another SRAF. The projection is measured from the short edge of the rectangular SRAF. Line-end distance is measured between SRAFs. The lineend value must be greater than or equal to minoffset. Square SRAFs have no line-end projection.

- projecting *value*

  An optional keyword and positive floating point number that specifies the minimum distance allowed between projections of SRAFs. The projecting distance is measured orthogonally to the edges of the offsetlayer shapes. SRAF to SRAF projecting distances are enforced symmetrically. The projecting value may be greater than or equal to the specified lineend value, and must be greater than or equal to minoffset.

- lineend2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the long side of the offsetlayer shape to the line-end of an SRAF. The measurement is performed during SRAF cleanup. The default is the specified value of projecting. The behavior is depicted in Figure 4-29.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- long2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the offsetlayer shape line-end to the long side of the an SRAF. The measurement is performed during SRAF cleanup. The default is the specified value of lineend. The behavior is depicted in Figure 4-29.

- corner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of any SRAF to the corner of a square offsetlayer shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

- lineend2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the line-end of an SRAF to the edge of a square offsetlayer shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-28.

- long2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the longer edge of an SRAF to the edge of a square offsetlayer shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-28.

- sqcorner2corner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of any offsetlayer shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-29.

- sqcorner2sqcorner *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the corner of a square SRAF to the corner of a square offsetlayer shape (non-projecting cases only). All corner-to-corner parameters must be a smaller value than any edge-related parameter. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

- sqedge2lineend *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the line-end of an offsetlayer shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-29.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- sqedge2long *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the longer edge of an offsetlayer shape. The measurement is performed during SRAF cleanup. The default is the specified projecting value. The behavior is depicted in Figure 4-29.

- sqedge2sqedge *value*

  An optional keyword and positive floating point number that specifies the minimum distance from the edge of a square SRAF to the edge of a square offsetlayer shape. The measurement is performed during SRAF cleanup. The default is the specified minoffset value. The behavior is depicted in Figure 4-28.

Note

Conditional MRC parameters such as lineend2long, long2lineend, square2long, etc., must be specified by priority only for minoffsetlayerspace. Do not define these conditional parameters globally. Although parsed as valid, global specification of these parameters degrades performance.

Description

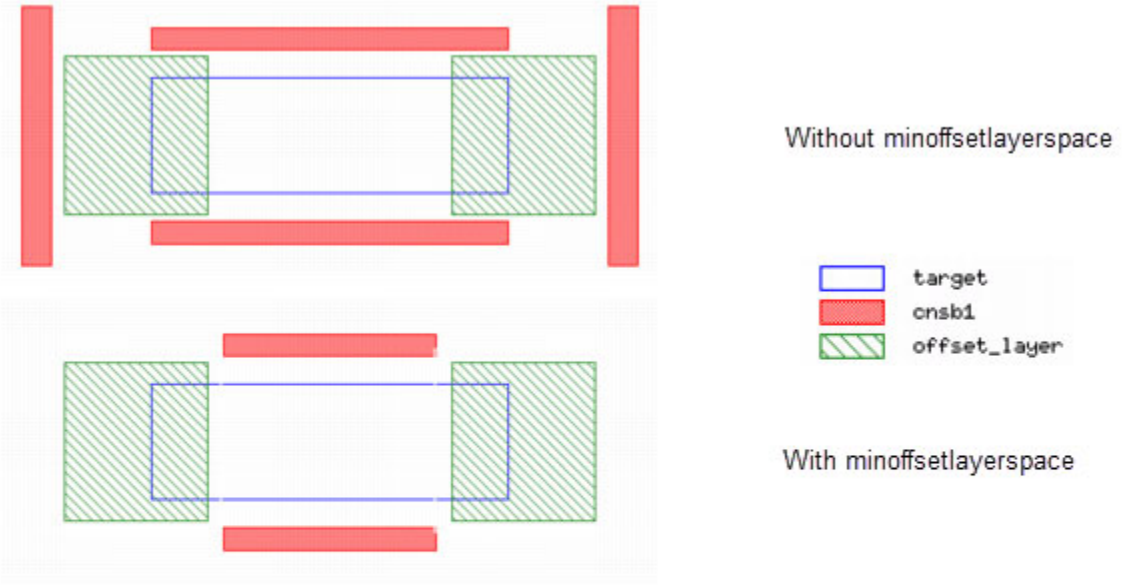
An optional keyword that specifies the minimum distance from any SRAF shape to the offsetlayer. You derive the offsetlayer for additional line-end keepout areas. This keyword can also be applied as a priority. For extended application information, see offsetlayer and minfeaturespace.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example1

Figure 4-34. Application of minoffsetlayerspace




```
LITHO FILE cn_setup_1 [
   layer target
   layer offset_layer
   minoffsetlayerspace 0.030
   setlayer cnsb1 = cnsraf target offsetlayer offset_layer ... \
]
cnsb1 = RET SBAR target offset_layer FILE cn_setup_1 MAP cnsb1
```

In Figure 4-34 the application of minoffsetlayerspace expands the offset_layer by 0.030 nm. This effectively expands the SRAF keepout area. Where minoffsetlayerspace and minfeaturespace are equivalent, the offset behavior reverts to releases prior to 2013.4.

Example 2

Figure 4-28 illustrates the minimum distance measurement behavior of all keywords involving square offsetlayer shape and square or non-square SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-35. Keyword Application With Square offsetlayer

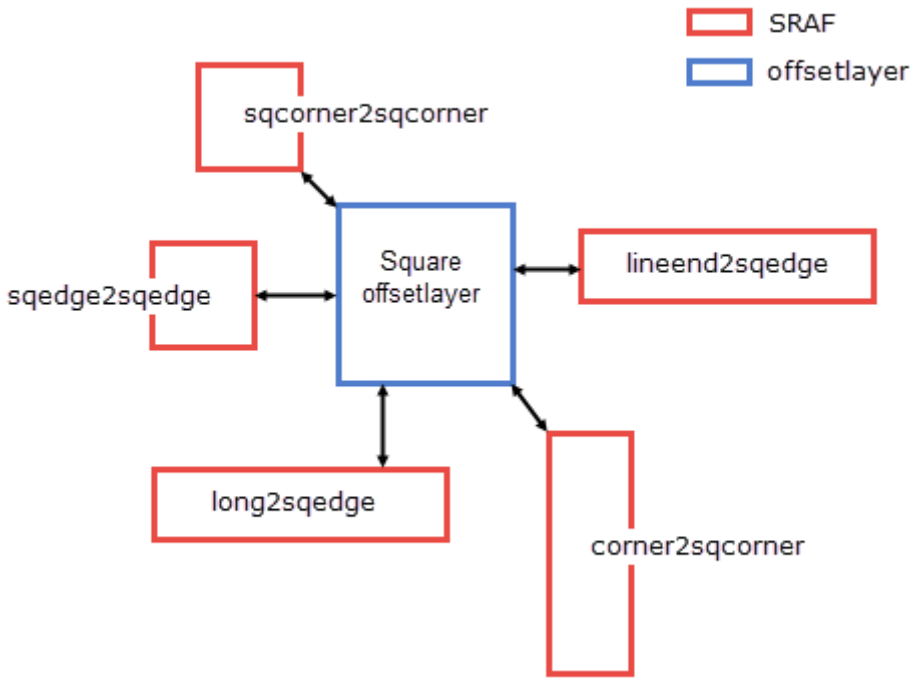
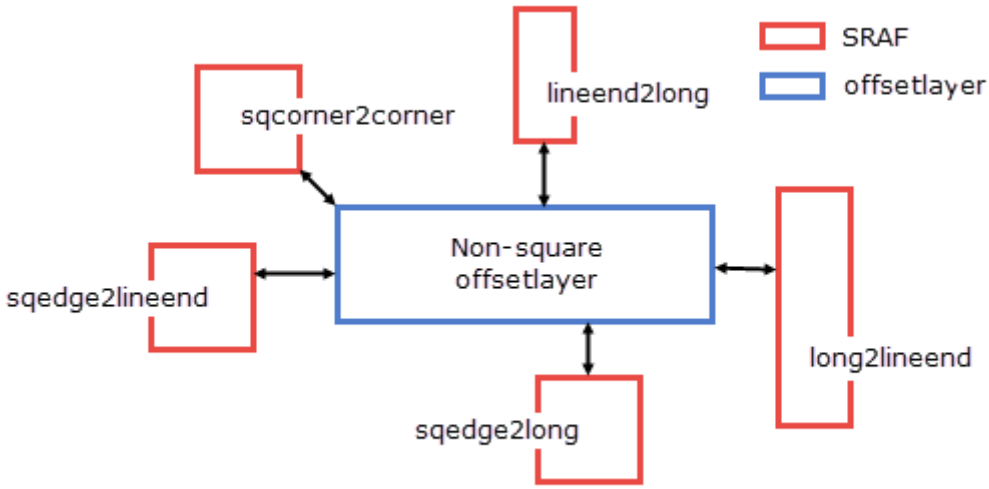


Figure 4-29 illustrates the minimum distance measurement behavior of all keywords involving SRAFs and non-square offsetlayer shapes.

Figure 4-36. Keyword Application With Non-Square offsetlayer



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minsquarearea

Constraint Keywords

Specifies the minimum area of a square SRAF. Used by RET SBAR and RET MBSRAF.

Usage

minsquarearea *value*

Arguments

- *value*

  A positive floating point number specified in square microns.

Description

An optional keyword that specifies the minimum area allowed for any square SRAF. Square SRAFs with an area smaller than minsquarearea are eliminated. If minsquarearea is not specified, the minimum area allowed for square SRAFs defaults to minarea. Although minarea applies to all SRAFs, if minsquarearea or minsquarelength are specified, all square SRAFs are checked against these two parameters but not minarea.

You can specify minsquarearea globally or by priority.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minsquarelength

Constraint Keywords

Specifies the minimum edge length for square SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

minsquarelength *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the minimum edge length allowed for any square SRAF. Square SRAFs with an edge length smaller than minsquarelength are deleted. The minsquarelength keyword does not apply to non-square SRAFs. Although minarea applies to all SRAFs, if minsquarearea or minsquarelength are specified, all square SRAFs are checked against these two parameters but not minarea.

You can specify minsquarelength globally or by priority.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minwidth

Constraint Keywords

Specifies the minimum width allowed for square SRAFs. Used by RET SBAR and RET MBSRAF.

Usage

minwidth *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the minimum width for square SRAFs. SRAFs with a width smaller than minwidth are eliminated. The minwidth keyword can be set globally or by priority.

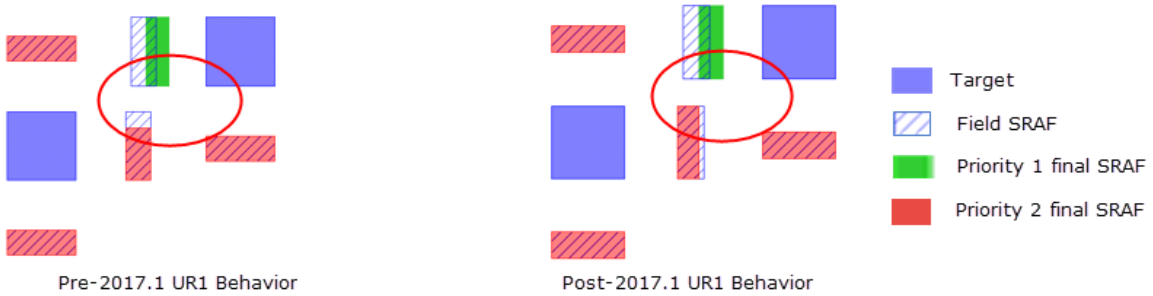
As of Calibre release 2017.1 UR1, when using cleanupversion 1 with minwidth, SRAFs can be trimmed by width only if minwidth is specified. When minwidth is specified, the trimming of an SRAF width or length is determined by satisfying the largest calculated SRAF area.

For information regarding interactions between minwidth, minarea and minlength, see "minarea, minlength, and minwidth Keyword Interaction" on page 325.

Examples

In the following example, to avoid the minoffset projecting violation between the priority 1 and priority 2 SRAFs and the target shape (circled areas), the priority 2 SRAF is trimmed in the length direction (pre-2017.1 UR1 behavior). However, this potentially causes a loss of significant SRAF area. Because SRAF is permitted to be trimmed by minwidth in the width direction as of 2017.1 UR1, the minoffset projecting violation is resolved and consequently more SRAF area may now be retained.

Figure 4-37. minwidth Behavior Change




```
minoffset 0.030 projecting 0.050
minwidth 0.024
rectEdge tag t1 extension 0.00 offset 0.045 width 0.027  prior 1
rectEdge tag t2 extension 0.00 offset 0.060 width 0.030  prior 2
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

shielding

Constraint Keywords

Uses edge tag shielding to characterize SRAF generation. Used by RET MBSRAF only.

Usage

shielding {distance | no | seedpixel}

Arguments

- distance

  This option only generates SRAFs in regions closer to tagged edges or corners than to other target edges.

- no

  This option turns shielding of tagged edges off. This reverts behavior back to Calibre release 2015.4.

- seedpixel

  This option determines whether a tagged edge or corner is visible by projection from any seed pixel. It ignores other target edges and corners that are not tagged, even if they are closer to the seed pixel.

Description

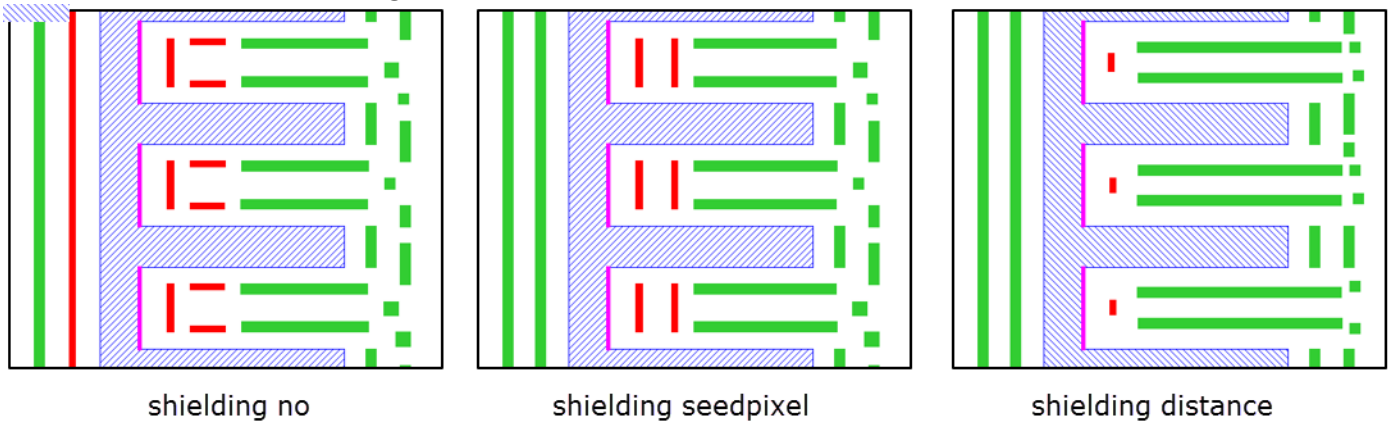
An optional keyword that specifies shielding of tagged edges to alter SRAF generation relative only to the tagged edges. Shielding may be specified globally or by priority. Shielding is compatible with positive and negative SRAF generation.

Examples

These illustrations show the three shielding options, their code snippets in context, and their respective results in each layout pattern.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

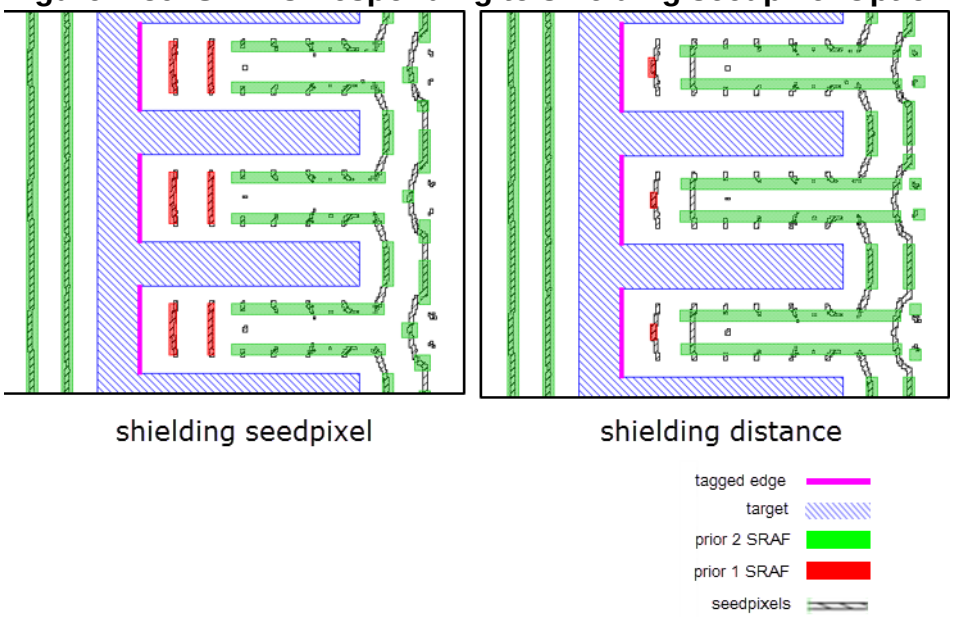
Figure 4-38. Positive SRAFs in Comb Pattern




```
newTag t1 length >0.2< 0.3
minoffset 0.04
shielding {no|seedpixel|distance}
LocalMaxGradient tag t1 width 0.015 SquareWidth 0.015  prior 1
LocalMaxGradient width 0.025 SquareWidth 0.025  prior 2
priority 1 minfeaturespace 0.04
priority 2 minfeaturespace 0.042
```

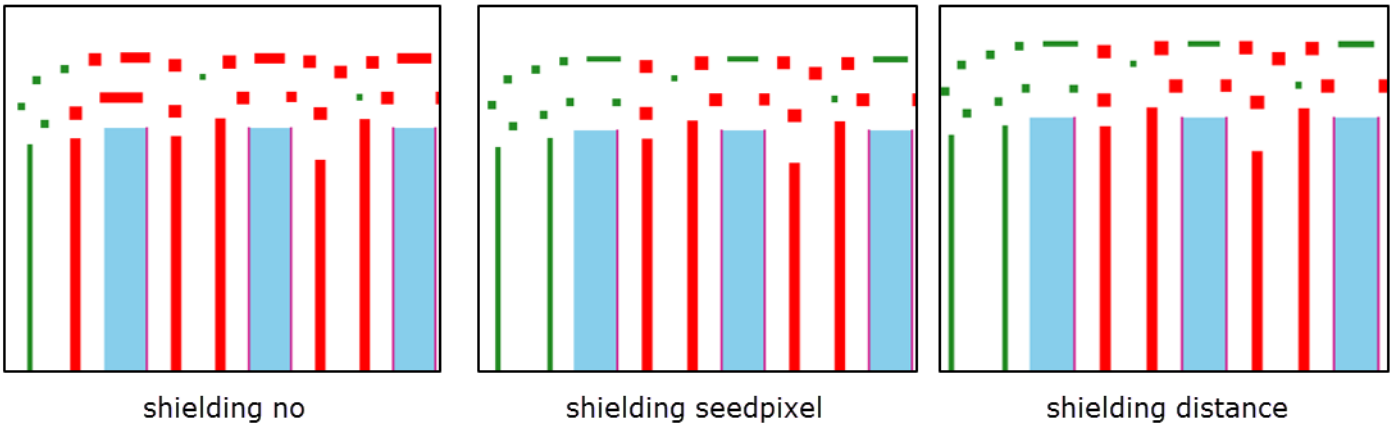
tagged edge
target
prior 2 SRAF
prior 1 SRAF

Figure 4-39. SRAFs Responding to shielding seedpixel Option



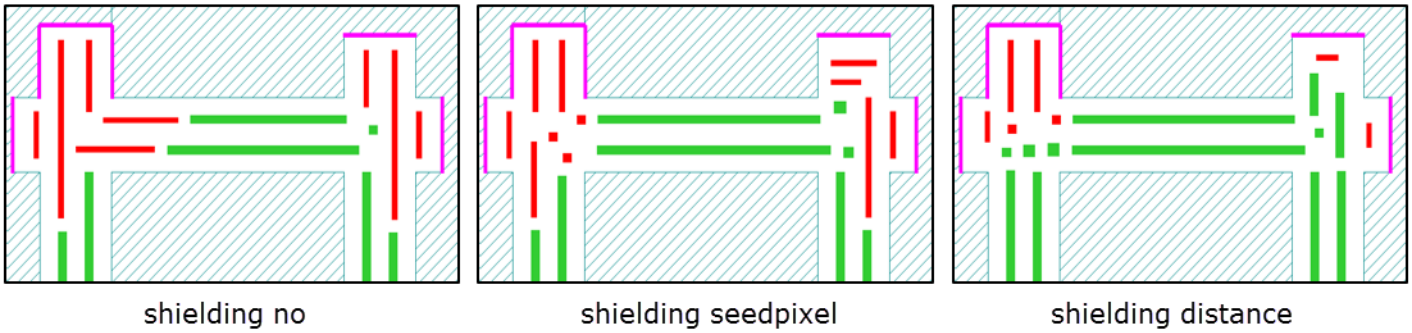
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-40. Positive SRAFs in Stripe Pattern

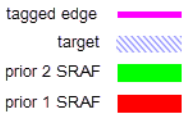



```
newTag t1 external >0.2< 0.3
minoffset 0.04
shielding {no|seedpixel|distance}
LocalMaxGradient tag t1 width 0.025 SquareWidth 0.025  prior 1
LocalMaxGradient width 0.015 SquareWidth 0.015  prior 2
priority 1 minfeaturespace 0.04
priority 2 minfeaturespace 0.05
```

Figure 4-41. Positive SRAFs in Irregular Pattern

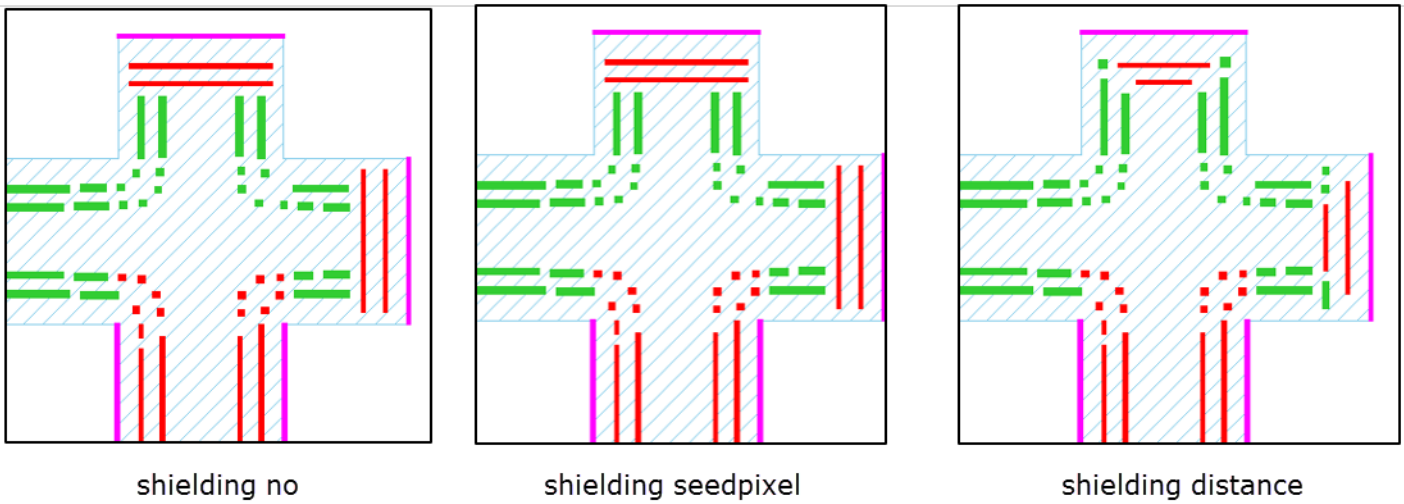



```
newTag t1 length >0.1< 0.15
minoffset 0.04
shielding {no|seedpixel|distance}
LocalMaxGradient tag t1 width 0.015 SquareWidth 0.015  prior 1
LocalMaxGradient width 0.025 SquareWidth 0.025  prior 2
priority 1 minfeaturespace 0.04
priority 2 minfeaturespace 0.042
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-42. Negative SRAFs in Irregular Pattern

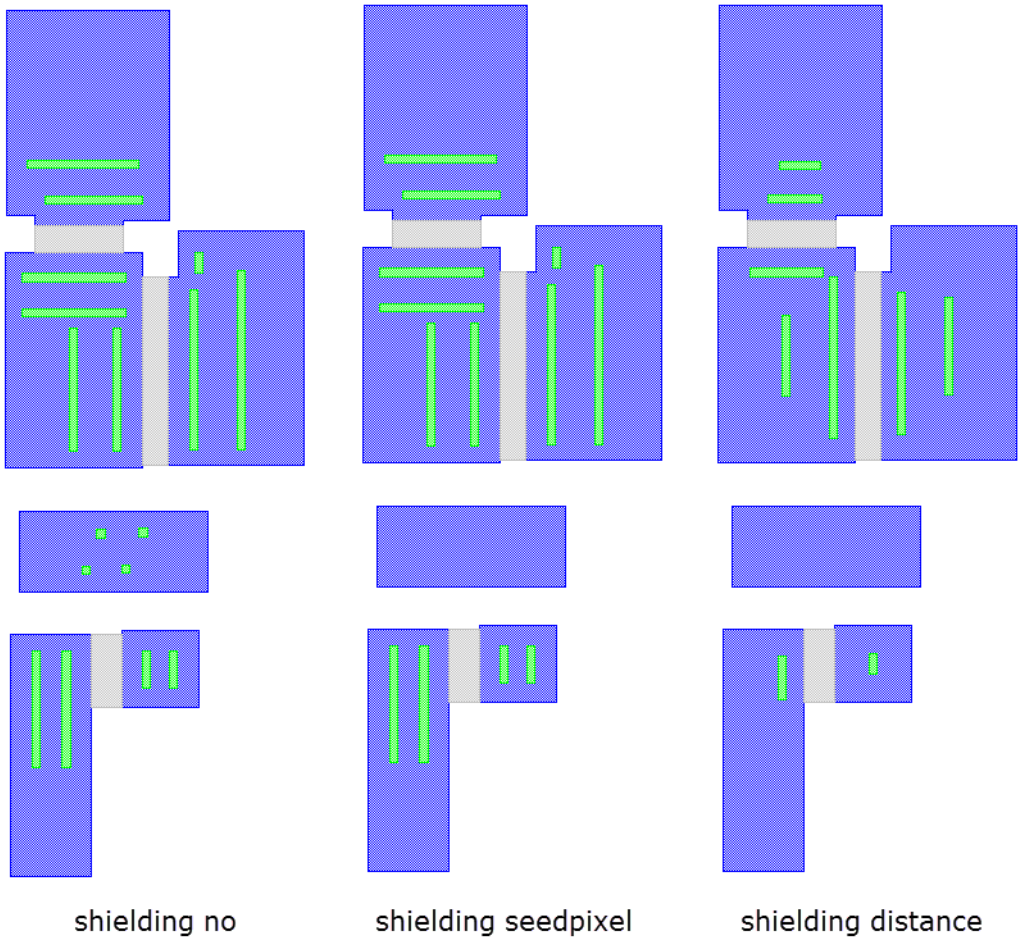



```
newTag t1 length >0.4 <0.5
newTag t2 length <1.1>0.9
minoffset 0.04
maxOffsetNegative 0.2
negative
LocalMaxGradient tag t1 tag t2 width 0.015 SquareWidth 0.025  prior 1
LocalMaxGradient width 0.025 SquareWidth 0.030  prior 2
priority 1 minfeaturespace 0.05  shielding {no|seedpixel|distance}
```

tagged edge
target
prior 2 SRAF
prior 1 SRAF

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-43. Negative SRAFs in Irregular Patterns




```
Taglayer vertical
newTag tightspace < 0.075
newTag v_tight tightspace and vertical
localMaxGradient tag v_tight width 0.021 squarewidth 0.02 prior 1
priority 1 negative shielding {no | seedpixel | distance}
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

squarewidth

Constraint Keywords

Specifies the width of square SRAFs. Used by RET MBSRAF only.

Usage

squarewidth *value*

Arguments

- *value*

  A positive floating point number specified in microns.

Description

An optional keyword that specifies the width of square SRAFs. The squarewidth keyword may be specified globally or within the localmaxgradient template.

SRAFs generated by squarewidth are placed near corners and have no projection to any target edge. The placement style of square SRAFs at target corners is controlled by cornerpref. The squarewidth keyword may also be specified in the localmaxgradient template.

If the squarewidth parameter is greater than the width parameter, then a square SRAF with a width equaling the squarewidth value is placed around the local maximum pixels.

The squarewidth and width parameters can be simultaneously specified, generating two sizes of square SRAFs. Where width-based square SRAFs are placed in projection with a target edge, they are potentially converted to MRC-compliant rectangular SRAFs. SRAFs generated by squarewidth are never converted to rectangles.

Although minarea applies to all SRAFs, if minsquarearea or minsquarelength are specified, all square SRAFs are checked against these two parameters and not minarea. If squarewidth^2 is less than minarea, no square SRAFs are generated, unless either minsquarearea or minsquarelength are specified.

You can specify squarewidth globally or by priority.

Note

In Calibre 2014.4 and later, squarewidth is no longer automatically calculated. Instead, square SRAFs of an area equal to width^2 are generated, given they do not violate either minsquarelength or minsquarearea parameters.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

width

Constraint Keywords

Specifies the default width of generated SRAFs. Used by RET MBSRAF only.

Usage

width value

Arguments

- value

  A positive floating point number specified in microns.

Description

A required keyword that specifies the default SRAF width. The width keyword can be specified globally or within the localmaxgradient template.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Templates

Calibre nmSRAF requires dimensional- or model-based templates to pattern and prioritize SRAF generation.

For organizational template information, refer to "sraf_options" on page 80.

Table 4-10. Templates

Keyword	Description
localmaxgradient	Creates SRAFs around the local maxima of the gradient. Used by RET MBSRAF only.
pairtemplate	Places SRAFs at specified angles and offsets from target shape center. Used by RET SBAR only.
rect	Places a rectangular SRAF relative to the center of the target shape. Used by RET SBAR only.
rect45	Places four rectangular SRAFs rotated 45 degrees relative to the target shape center. Used by RET SBAR only.
rect45corner	Places four rectangular SRAFs rotated 45 degrees at target shape corners. Used by RET SBAR only.
rectcorner	Places SRAFs with four methods relative to corners of target shapes. Used by RET SBAR only.
rectdiag	Places four or eight rectangular SRAFs diagonally offset relative to the center of the target shape. Used by RET SBAR only.
rectedge	Places rectangular SRAFs offset from each edge of the target shape. Used by RET SBAR only.
recthv	Places four rectangular SRAFs at orthogonal X,Y pitches relative to the target shape center. Used by RET SBAR only.
srafring	Places square SRAFs radially around a target shape. Used by RET SBAR only.
srafringhv	Places rectangular and square SRAFs radially around a target shape. Used by RET SBAR only.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

localmaxgradient

Templates

Creates SRAFs around the local maxima of the gradient. Used by RET MBSRAF only.

Usage

localmaxgradient
[tag *name* ...]
width *value*
[centerwidth *value*]
[extension *value*]
[minwidth *value*]
[prior *value*]
[squarewidth *value*]

Description

A required template and argument set that creates SRAFs around the local maxima of the gradient. All possible SRAF placements are computed. SRAFs are accepted in the order of decreasing integral values, unless an SRAF overlaps an already accepted SRAF with a larger integral value.

Multiple specification of localmaxgradient templates permit generation of increasingly larger SRAFs outwardly through varied width and squarewidth parameters. Larger SRAFs placed at greater distances from the target benefit the resulting image and reduce SRAF printing.

Note

For recipes requiring placement of varying SRAF widths by multiple localmaxgradient templates, Siemens EDA recommends against using the maxoffset keyword. For more information, see Example 3 — Use of Multiple localmaxgradient Templates and "Calibre cnSRAF Template Execution and Cleanup" on page 17.

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **width *value***

  A required keyword (specified in the localmaxgradient template or globally) that specifies the width of rectangle SRAFs that are generated by the localmaxgradient template. If width is not specified globally or in the template, a parser error is output. All localmaxgradient templates of the same priority must specify the same width value if centerwidth is also specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- centerwidth *value*

  An optional keyword which generates SRAFs equidistant from input target shapes. The centerwidth value must be greater than or equal to the width value. All localmaxgradient templates of the same priority must specify the same centerwidth value.

- extension *value*

  An optional keyword that specifies the length of extension added to both ends of the original SRAF.

  The extension keyword supports negative extensions as long as the extension is less than half of the SRAF length and the resulting SRAF does not violate MRC constraints.

- minwidth *value*

  An optional keyword that specifies the minimum width allowed for a square SRAF generated by the localmaxgradient template.

- prior *value*

  An optional keyword that specifies the priority of the SRAFs placed by localmaxgradient.

  The priorities are applied sequentially: a priority of 1 is applied first, then a priority of 2 is applied, and so on. The default priority is 1.

- squarewidth *value*

  An optional keyword that specifies the width of square SRAFs generated by the localmaxgradient template. The squarewidth keyword may also be specified globally.

  If squarewidth and width are specified together, and squarewidth is greater than or equal to width, then a square SRAF with edges of squarewidth is placed around the local maximum pixels in conformance to the cornerpref setting. All localmaxgradient templates of the same priority must specify the same squarewidth value.

  As of Calibre release 2014.4, squarewidth is no longer automatically computed.

Examples

Example 1 — Contextual Use of localmaxgradient

```
LITHO FILE mb_setup [
   ...

   sraf_options s_opt {
      localmaxgradient width 0.035 squarewidth 0.040 prior 3
      ...
   }

   setlayer sb = mbsraf target OPTIONS s_opt optical f0 resist res

]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

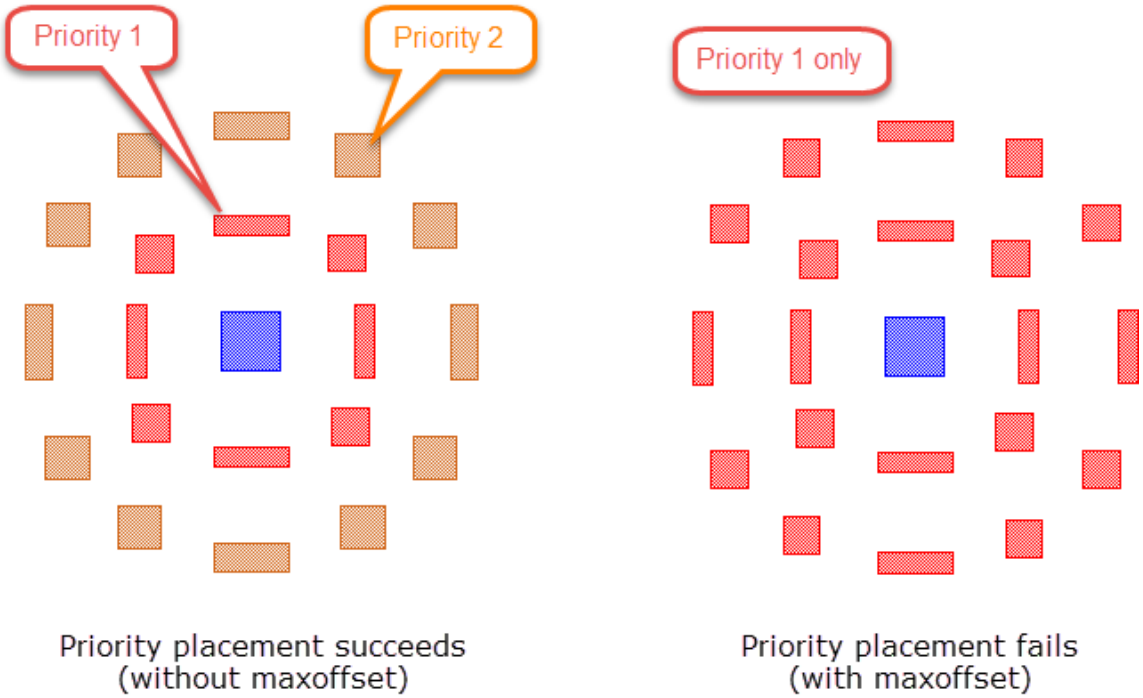
Example 2 — Use of a Single localmaxgradient Template With maxoffset

```
# Assignment of a maximum SRAF offset for a single template
# generating multiple SRAF rings.

maxoffset 0.250
localmaxgradient width 0.016 squarewidth 0.023
```

Example 3 — Use of Multiple localmaxgradient Templates

```
# Assignment of priority for templates with different widths.

localmaxgradient width 0.016 squarewidth 0.023 prior 1
localmaxgradient width 0.020 squarewidth 0.025 prior 2
```

Figure 4-44. SRAF Output Using Two Templates of Different Priorities With and Without maxoffset



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

pairtemplate

Templates

Places SRAFs at specified angles and offsets from target shape center. Used by RET SBAR only.

Usage

pairtemplate
tag *name* [tag *name* ...]
{**angle *value*** | **offset *value*** | {**angle *value* offset *value***}}
width *value*
[length *value*]
[orient {horizontal | ortho | vertical}]
[pitchfilter *value*]
[prior *value*]

Arguments

- **tag *name*** ...

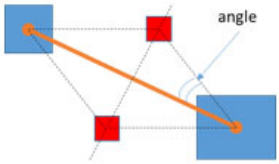
  A required keyword specifying one or more tag names generated only by the pairtag keyword of the newtag command. The tag name identifies a group of connectors to the centers of shapes to be input to the template. Tags must appear as the first arguments in the template.

- **angle *value*** | **offset *value*** | {**angle *value* offset *value***}

  One of two required keyword or keyword pairs specifying the angle and/or the offset SRAFs are placed from target shapes. The connectors used in the measurements must be generated by the pairtag keyword.

  The angle keyword specifies the minimum allowed angle between the connector and axis of the proposed SRAF from the end point of the connector. You must specify the value in degrees, between 0 and 90 degrees, inclusive.

Figure 4-45. Angle influence on SRAF placement



  The offset keyword specifies the minimum distance between the center of the proposed SRAF to the end point of the connector.You must specify a positive value in microns. When the offset value is less than half the connector length, SRAFs are placed on the connector at the offset distance from the connector end, as depicted in Figure 4-46.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-46. Offset less than half connector length

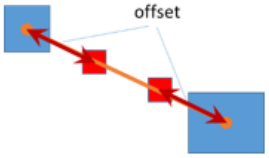
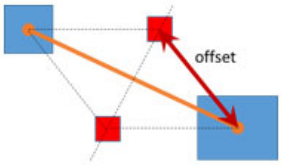


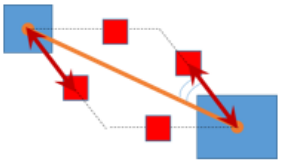
Figure 4-47 depicts use of the offset keyword where the offset value exceeds half the length of the connector. These SRAFs are placed on either side of the connector.

Figure 4-47. Offset greater than half connector length



You may specify the angle and offset keywords together as required as depicted in Figure 4-48.

Figure 4-48. The offset and angle keywords together



- **width *value***

  A required keyword that specifies the width of SRAFs.

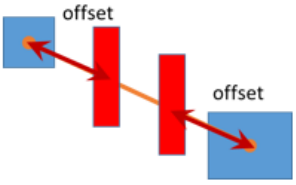
- length *value*

  An optional keyword that specifies the length of rectangular SRAFs to be generated. If not specified, length defaults to width, generating square SRAFs.

- orient {horizontal | <u>ortho</u> | vertical}

  An optional keyword that orients the generated SRAFs relative to the cell. The horizontal and vertical keywords generate SRAFs parallel to the X-axis, and parallel to the Y-axis, respectively. The ortho keyword generates SRAFs with horizontal orientation for connectors inclined more than 45 degrees, or vertical orientation for connectors inclined 45 degrees or less. The default mode is ortho. Figure 4-49 depicts a vertical orientation.

Figure 4-49. SRAFs oriented in vertical direction



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- pitchfilter *value*

  An optional keyword and parameter pair that allows SRAF generation if no target shapes are found closer to the proposed SRAFs than the specified value. The distance is measured on center-to-center between target shapes and SRAFs. Shape centers at connector end-points are not considered. If the specified value is -1, then the distance is defined between the center of the proposed SRAF and the end point of the connector. The value must be a positive number in microns. It specifies the minimum distance between target shapes and SRAFs for generation of SRAFs to be performed. The distance is measured between centers of the bounding boxes of target shapes and centers of SRAFs. The target shape of origin (target shape with its center at the connector end point) is not considered.If the value is negative (-1), the distance is then defined as between the center of the SRAF and end point of the pairtag connector which generates the SRAF.

  Pitchfilter filters out the field SRAFs generated by pairtemplate based on their center-to-center distance between the SRAF and other target shapes (not those associated with pairtag) that did not create these SRAF fields.

  Filtering is disabled when pitchfilter is not specified or the specified value is 0. This is the default behavior.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a *value* of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that places rectangular SRAFs at the specified angles and offsets from each target center. The pairtemplate template must be specified with a tag generated by the pairtag keyword of the newtag command.

Examples

Example 1 — Basic Usage

```
# Generates square SRAFs located 150 nm from connector ends
# (pTag shape centers) with an angle of 30 degrees from the connector.

pairtemplate TAG pTag WIDTH 0.02 offset 0.15 angle 30 prior 1
```

Example 2 — Comprehensive Usage

```
# Generates oblong SRAFs with two priorities located at 85 nm and 120 nm
# from connector ends (pt1 and pt2, respectively) with  angles of 20
# and 40 degrees from connectors.

newtag pt1 pairtag distance >=0.1 <0.2 projection >= 0.02
newtag pt2 pairtag distance >=0.1 <0.3 projection >= 0.04
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

```
pairtemplate tag pt1 width 0.020 length 0.070 offset 0.085
   angle 20 prior 10
pairtemplate tag pt2 width 0.030 length 0.090 offset 0.120
   angle 40 prior 20
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rect

Templates

Places a rectangular SRAF relative to the center of the target shape. Used by RET SBAR only.

Usage

rect
[tag *name* ...]
{**length** ***value*** **width** ***value***}
{**pitchx** ***value*** **pitchy** ***value***}
[conflictonly]
[orient {horizontal | vertical}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value*** **width** ***value***

  Two required keywords, **length** and **width**, that specify the longer and shorter edges of a rectangular SRAF, respectively. The **length** and **width** values can be identical.

- **pitchx** ***value*** **pitchy** ***value***

  A required keyword pair that specifies the horizontal and vertical distance from the center of the target shape to the center of the SRAF, respectively.

- conflictonly

  An optional keyword that outputs SRAFs within conflict regions only; no other SRAFs of this template are generated. Conflict regions are formed where neighboring SRAFs generated by this template violate the MRC rules you specify in the template. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using the conflictonly argument only when conflict resolution shapes are relatively small and not using conflictonly for larger shapes such as line layer designs.

- orient horizontal | vertical

  An optional keyword that enables asymmetric templates to be placed and oriented based on the orientation of non-square rectangles. Specifying orient horizontal applies SRAFs to target shapes as if the SRAFs were oriented horizontally (the long edge of the rectangle is parallel to the x-axis). Specifying orient vertical applies templates to rectangular target shapes as if they were oriented vertically (the long edge of the rectangle is parallel to the y-axis).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See "priority" on page 315 for more information.

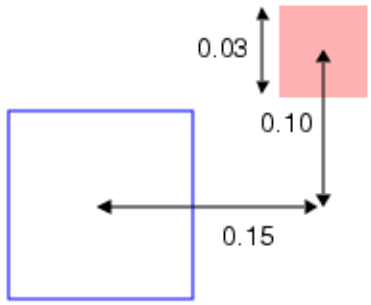
Description

An optional template and argument set that places a single rectangular SRAF the specified distance from the center of a target shape. The center of the rectangular SRAF is placed (pitchx, pitchy) from the center of the target shape. This is an asymmetric template that is disabled with the symmetric keyword. This template cannot generate negative SRAFs.

Examples

Figure 4-50. rect Default Usage

```
rect
   width 0.030 length 0.030
   pitchx 0.15 pitchy 0.10
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rect45

Templates

Places four rectangular SRAFs rotated 45 degrees relative to the target shape center. Used by RET SBAR only.

Usage

rect45
[tag *name* ...]
{**length** ***value*** **width** ***value***}
{**pitchx** ***value*** **pitchy** ***value***}
[conflictonly]
[orient {horizontal | vertical}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value*** **width** ***value***

  Two required keywords, length and width, that specify the longer and shorter edges of a rectangular SRAF, respectively. The length and width values can be identical.

- **pitchx** ***value*** **pitchy** ***value***

  A required keyword pair that specifies the horizontal and vertical distance away from the center of the target shape to the center of placed SRAFs.

- conflictonly

  An optional keyword that outputs SRAFs within conflict regions only. No other SRAFs of this template are generated. Conflict regions are formed where neighboring SRAFs generated by this template violate the MRC rules you specify in the template. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using the conflictonly argument only when conflict resolution shapes are relatively small and not using conflictonly for larger shapes such as line layer designs.

- orient horizontal | vertical

  An optional keyword that enables asymmetric templates to be placed and oriented based on the orientation of non-square rectangles. Specifying orient horizontal applies templates to rectangular target shapes as if they were oriented horizontally (the long edge of the rectangle is parallel to the x-axis). Specifying orient vertical applies templates to rectangular target shapes as if they were oriented vertically (the long edge of the rectangle is parallel to the y-axis).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See "priority" on page 315 for more information.

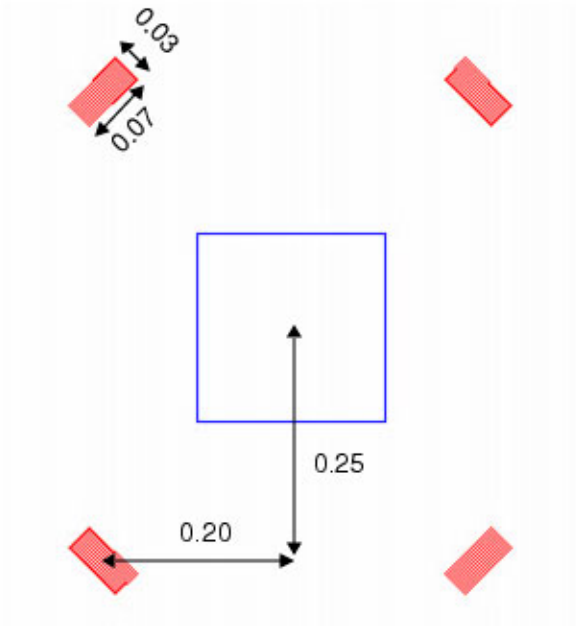
Description

An optional template and argument set that places four rectangular SRAFs rotated 45 degrees. The center of the rectangular SRAFs are located symmetrically and diagonally from the center of a target shape. This template cannot generate negative SRAFs.

Examples

Figure 4-51. rect45 Default Usage

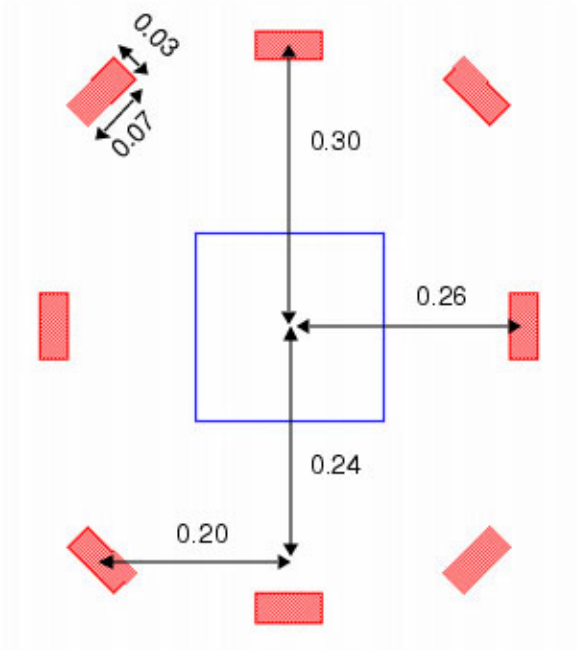
```
rect45
   width 0.03 length 0.07
   pitchx 0.20 pitchy 0.25
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-52. Combination rect45 and recthv Usage

```
rect45
   width 0.03 length 0.07
   pitchx 0.20 pitchy 0.24

recthv
   width 0.03 length 0.07
   pitchx 0.26 pitchy 0.30
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rect45corner

Templates

Places four rectangular SRAFs rotated 45 degrees at target shape corners. Used by RET SBAR only.

Usage

rect45corner
[tag *name* ...]
{**length** ***value*** **width** ***value***}
{**offsetx** ***value*** **offsety** ***value***}
[angledalso]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value*** **width** ***value***

  Two required keywords, length and width, that specify the longer and shorter edges of a rectangular SRAF, respectively. The length and width values can be identical.

- **offsetx** ***value*** **offsety** ***value***

  A required keyword pair that specifies the horizontal and vertical distance away from the corner of the target shape to the corner of the SRAF, respectively. By default, offsetx is measured from the Y-axis aligned target edge and offsety is measured from the X-axis aligned target edge. The offsetx and offsety values can be identical.

- angledalso

  An optional keyword that additionally generates axis-aligned SRAFs at corners of 45-degree oriented target shapes only for the rect45corner template.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that specifies the placement of four SRAFs rotated 45 degrees relative to target shape corners. The SRAF is placed by the center point of its edge closest to the target corner at the corresponding values of offsetx and offsety.

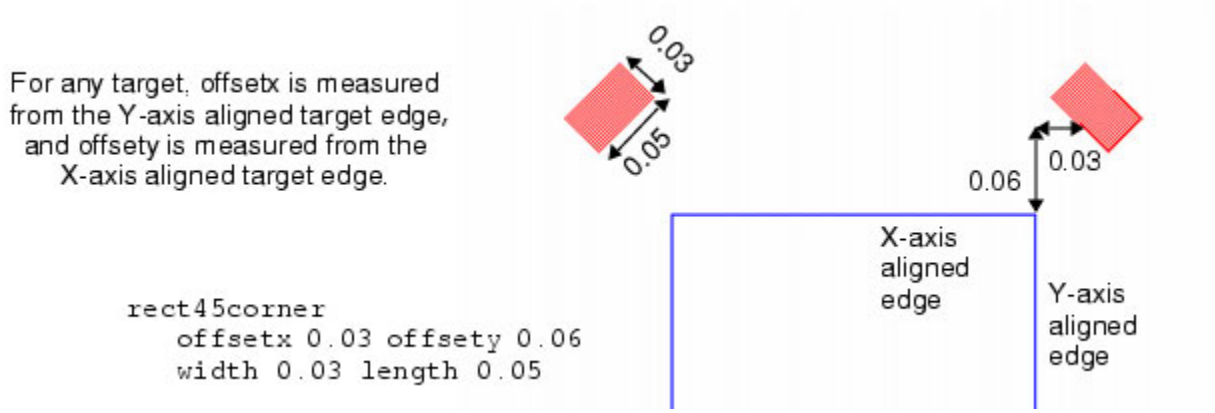
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1

In Figure 4-53 the four required keywords shown here determine the size of the SRAFs and their X- and Y-offsets.

Figure 4-53. rect45corner Default Behavior

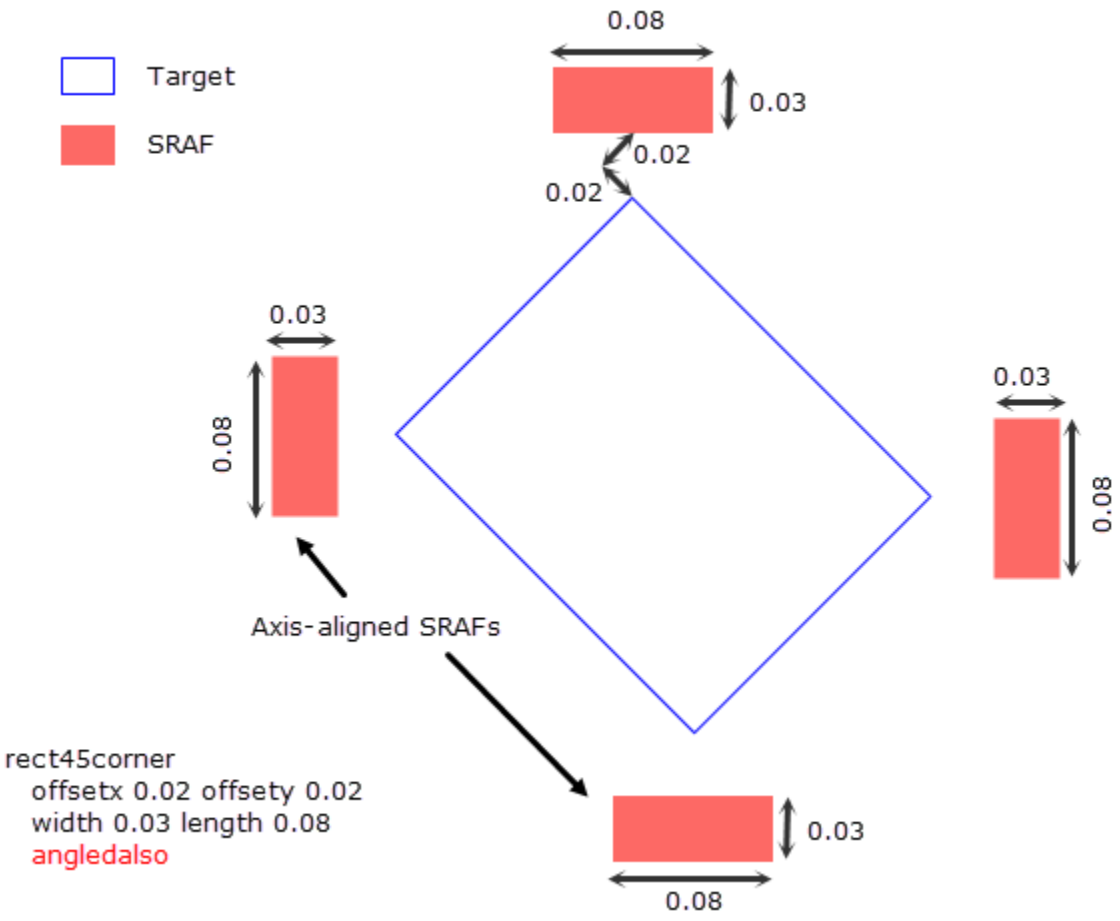


Example 2

In Figure 4-54, the effects of the angledalso option on SRAFs generated by rect45corner are shown here. The angledalso option permits the generation of axis-aligned SRAFs at corners of target shapes that have been rotated to 45 degrees.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-54. rect45corner angledalso Behavior



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rectcorner

Templates

Places SRAFs with four methods relative to corners of target shapes. Used by RET SBAR only.

Usage

center Method

rectcorner
[tag *name* ...]
{**center spacelimit *value***
{**length *value*** | **width *value***}
{**angle *degree*** | **offset *distance***}}
[corner {both | center | pair}]
[strictcenter]
[prior *priority*]

center multicorner Method

rectcorner
[tag *name* ...]
{**center multicorner spacelimit *value***
{**length *value*** | **width *value***}
{**angle *degree*** | **offset *distance*** [offsetmultiplier *factor*]}}
[strictcenter]
[prior *priority*]

Offset Method

rectcorner
[tag *name* ...]
{**offsetx *value*** | **offsety *value***}
{**length *value*** | **width *value***}
[breaksym]
[orient {parallel | perpendicular}]
[prior *priority*]

scalefactor Method

rectcorner
[tag *name* ...]
{**scalefactor *index* offset *distance***
{**length *value*** | **width *value***}}
[edgediff 0 | 1]
[{ext1 *value* ext2 *value* [off1 *value*] off2 *value*}]
[orient 45]
[protectionradius *radius*]
[prior *priority*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Arguments

Optional Keywords Shared by Methods

Optional keywords used with various methods described in this section.

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template. All rectcorner usages accept tag names.

- prior *priority*

  An optional keyword, applicable to all three usages, that specifies a numeric priority for placed SRAFs. The priority keyword must be specified after the tag within the template. The default priority is 1. See "priority" on page 315 for more information.

- strictcenter

  An optional keyword controlling center SRAF placement between tagged corners. Specifying strictcenter restricts SRAFs to be generated only between two tagged corners. By default, where tags are specified within the rectcorner template using the center method, center SRAFs are placed between one or two tagged corners. If no tags are specified within the rectcorner template, center SRAFs are placed between any qualifying corner, and strictcenter has no effect. This keyword is used only by center and center multicorner methods. See Figure 4-72 for more information.

center Method

- **center spacelimit *value*** {**length *value*** | **width *value***} {**angle *degree*** | **offset *distance***} [corner {both | center | pair}]

  The center method places SRAFs equidistant from qualifying corners provided by tagged edges. At least one tag must be specified for center. The center keyword requires cleanupversion 1. For detailed illustrations, see "Example 8 — center offset and angle Effects" on page 203. The center usage is mutually exclusive with offset and scalefactor usages.

  The center keyword boundary of corner and edge collection is derived from spacelimit which, in the following example, is 0.8. However, spacelimit never overrides any specified newtag external values. In the following case, the boundary for edge collection is established by tag2 (0.9) because its value is greater than spacelimit (0.8):

```
cleanupversion 1
newtag tag1 external <= 0.5
newtag tag2 external <= 0.9
rectcorner tag tag1 tag tag2 center spacelimit 0.8
```

  Positive SRAF generation using the center option requires specifying newtag of external type for selection of the outside edges of target shapes for correct SRAF placement. Negative SRAF generation using the center option requires specifying newtag internal for selection of the inside edges of target shapes for correct SRAF placement.

  spacelimit *value*

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The spacelimit keyword and value define the maximum distance between corners that can generate corner SRAFs. The spacelimit keyword establishes the boundary by euclidean measurement.

Note

Excessively large spacelimit values trigger larger corner search areas and significantly increase run time.

length *value* | width *value*

A required keyword argument to specify longer and shorter edges of rectangular SRAFs. Either length or width must be specified.

angle *degree* | offset *distance*

One of two required, mutually exclusive keywords that must be specified with the center keyword. The angle keyword specifies the position of generated SRAFs by the angle extended from the corners of target shapes. For two-corner position, only positive values may be specified for degree. SRAF placement defaults to angle 45. See Figure 4-69 for more information.

The offset keyword specifies the offset position of generated SRAFs by an orthogonal distance from the center of the line connecting corners of target shapes. Only non-negative values may be specified for distance. There is no default for offset. See Figure 4-70 for more information.

corner {both | center | pair}

An optional keyword that controls output of corner SRAFs. The corner keyword is used only by the center method.

both

Outputs center and pair SRAFs.

center

Outputs a center SRAF. This is the default.

pair

Outputs a pair of SRAFs.

Note

The center method only works with cleanupversion 1; parse errors are issued if used with cleanupversion 0.

center multicorner Method

- **center multicorner spacelimit *value* length *value* width *value***
  {**angle *degree* | offset *distance*** [offsetmultiplier *factor*]}

  The center multicorner method places SRAFs equidistant from qualifying corners provided by tagged edges. At least one tag must be specified for center. The center keyword requires cleanupversion 1. For detailed illustrations, see “Example 8 — center offset and angle

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Effects" on page 203. The center multicorner method is mutually exclusive with offset and scalefactor methods.

spacelimit *value*

The spacelimit keyword and value define the maximum distance between corners that can generate corner SRAFs. The spacelimit keyword establishes the boundary by euclidean measurement.

Note

Excessively large spacelimit values trigger larger corner search areas and significantly increase run time.

length *value* width *value*

Two required keywords that specify the longer and shorter edges of rectangular SRAFs, respectively. The length and width values can be identical.

angle *degree* | offset *distance* [offsetmultiplier *factor*]

One of two mutually exclusive keywords that must be specified with the center keyword. The angle keyword specifies the position of generated SRAFs by the angle extended from the corners of target shapes. For two-corner position, only positive values may be specified for degree. SRAF placement defaults to angle 45. See Figure 4-69 for more information.

The offset keyword specifies the offset position of generated SRAFs by an orthogonal distance from the center of the line connecting corners of target shapes. Only non-negative values may be specified for distance. There is no default for offset. See Figure 4-70 for more information.

offsetmultiplier *factor*

The offsetmultiplier keyword specifies a positive integer as the offset multiplier for SRAFs generated with the offset keyword. Multiple SRAFs may be generated with their centers offset as the specified multiple of the specified offset distance. The default factor is 1. The offsetmultiplier keyword is only used by the center multicorner method. See Figure 4-70 for more information.

Note

The center multicorner method only works with cleanupversion 1; parse errors are issued if used with cleanupversion 0.

Offset Method

- **offsetx *value* offsety *value* length *value* width *value*** [breaksym] [orient {parallel | perpendicular}]

  The offset method places SRAFs from qualifying corners at the specified offset values. For detailed illustrations, see "Example 1 — Identical offset, width, and length Values" on page 195. The offset usage is mutually exclusive with center and scalefactor usages.

  offsetx *value* offsety *value*

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Two required keywords that specify the orthogonal distances from the target corner, in the X- and Y-axes, respectively, to the closest corner of generated SRAFs.

length *value* width *value*

Two required keywords that specify the longer and shorter edges of rectangular SRAFs, respectively. The length and width values can be identical.

breaksym

An optional keyword, applied only to the offset usage, that outputs SRAF with a designated length parallel to the Y-axis (vertically-oriented SRAF), that breaks the symmetric SRAF output.

orient {parallel | perpendicular}

An optional keyword that filters SRAFs oriented either parallel or perpendicular to the tags input to rectcorner. For square target shapes, orient perpendicular places SRAFs with edges designated by length perpendicular to the X-axis; orient parallel places SRAFs with edges designated by length parallel to the X-axis. For non-square target shapes, orient parallel places SRAFs with edges designated by length parallel to the longest edge of the target shape; orient perpendicular places SRAFs with edges designated by width perpendicular to the longest edge of the target shape. Application of orient is consistent irrespective of the target shape orientation. All parallel and perpendicular SRAFs are output by default.

scalefactor Method

- **scalefactor *index* offset *distance*** {**length *value*** | **width *value***} [breaksym] [edgediff 0 | 1] [{ext1 *value* ext2 *value* [off1 *value*] off2 *value*}] [orient 45] [protectionradius *radius*]

  The scalefactor method generates one-quarter circles of square SRAFs around target corners. Every generated SRAF placed on the arc is square and MRC-clean. For detailed illustrations, see "Example 7 — scalefactor Indexes, offset Effects, and orient 45" on page 198. The scalefactor usage is mutually exclusive with center and offset usages.

  scalefactor *index*

  A scalefactor index must be specified within the range of 0.7 through 1.4. Indexes less than 1 pull the SRAFs toward the target; indexes greater than 1 push the SRAFs away from the target.

  offset *distance*

  A required argument that specifies the distance from target that SRAFs are placed. This value is the distance along the edge at the corner.

  length *value* | **width *value***

  Required keywords that specify length or width of generated SRAFs. When applied to the scalefactor keyword, only one of width or length may be specified. Both keywords render identical results because scalefactor generates square SRAFs.

  edgediff <u>0</u> | 1

  An optional keyword that enables placement of SRAFs around non-orthogonal corners. If edgediff is set to 0, no SRAFs are placed around non-orthogonal corners.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The edgediff keyword can be used with ext1, ext2, off1, and off2 keywords. The default is 0.

ext1 *value*

An optional keyword that determines the orthogonal extension from off1, the offset of the longer of the two edges of the corner. There is no default.

ext2 *value*

An optional keyword that determines the orthogonal extension from off2, the offset of the shorter of the two edges of the corner. There is no default.

off1 *value*

An optional keyword that determines the orthogonal offset from the longer of the two edges of the corner. The default for off1 is the value set for offset of the rectcorner template.

off2 *value*

An optional keyword that determines the orthogonal offset from the shorter of the two edges of the corner. There is no default.

orient 45

An optional keyword that places SRAFs at a 45 degree angle to the target corner. Application of orient 45 is consistent irrespective of the target shape orientation.

protectionradius *radius*

An optional keyword that maintains the specified radius between SRAFs placed by scalefactor and neighboring SRAFs generated by other templates. The protectionradius keyword only works with the scalefactor method of the rectcorner template. The default is 1/2 of minoffset.

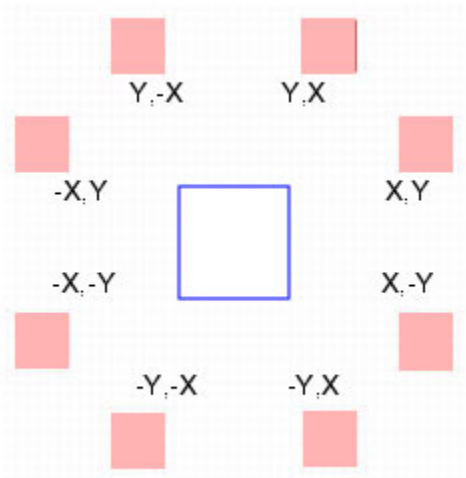
Description

An optional template and argument set that, by default, places eight rectangular SRAFs symmetrically, and orthogonally by X- and Y-offsets, from each corner of a target shape or from a tagged edge.

The corners of each SRAF are placed symmetrically by the coordinate system as shown in Figure 4-55.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-55. rectcorner Default SRAF Placement Behavior



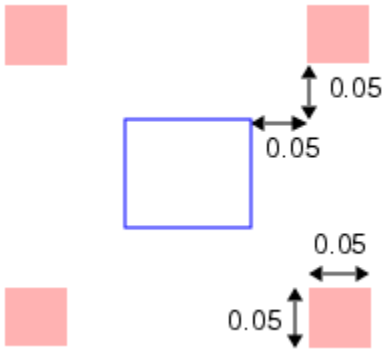
Examples

Example 1 — Identical offset, width, and length Values

Usage of rectcorner with identical specifications for X- and Y-offsets and SRAF length and width is shown. Because of a single offset value, only one SRAF is output at each corner.

Figure 4-56. rectcorner With Identical offset, width and length Values

```
rectcorner
   offsetx 0.05 offsety 0.05
   width 0.05 length 0.05
```

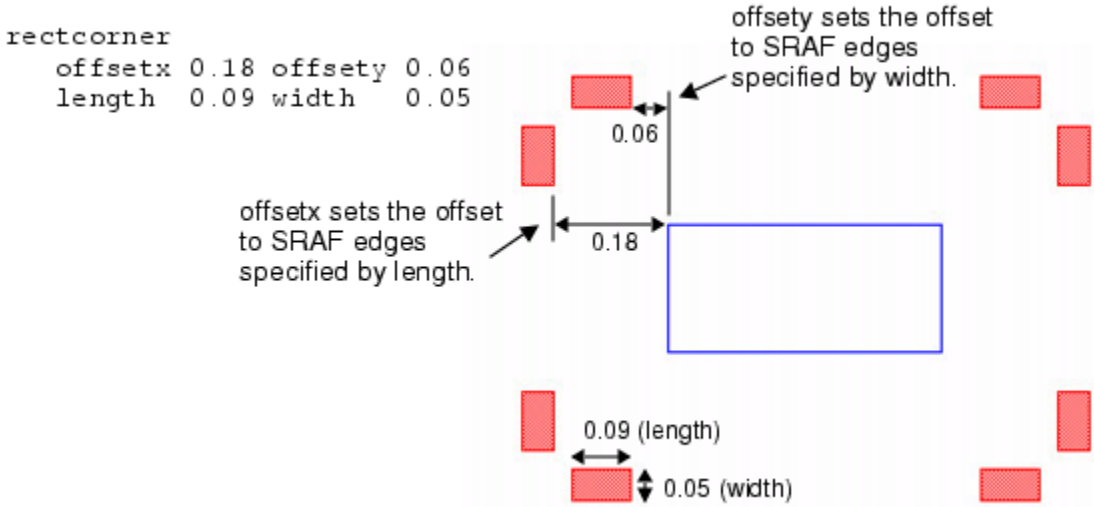


Example 2 — Different offset, width, and length Values

The default usage of rectcorner supports varied specifications for all parameters. By default, SRAFs with both orientations (perpendicular and parallel) are placed as seen in the following figure.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

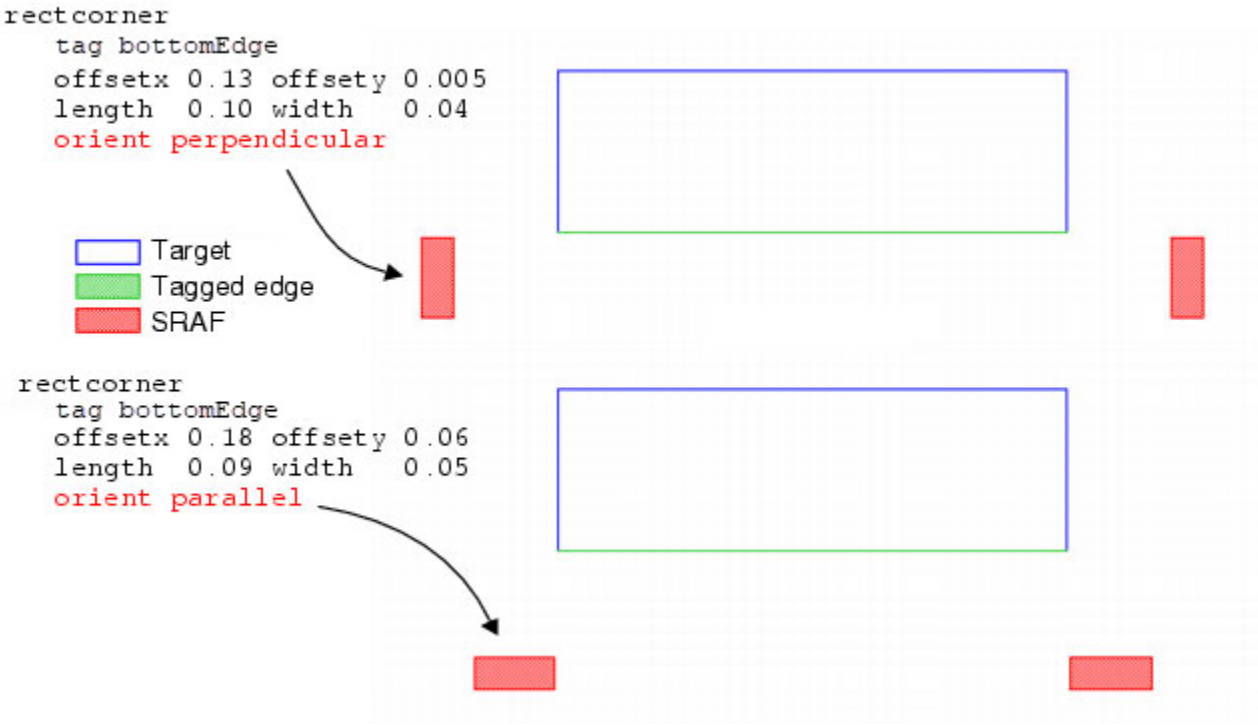
Figure 4-57. rectcorner Usage With offset, width and length Variations



Example 3 — Orient Keywords

The orient parallel and orient perpendicular keywords output SRAFs oriented by their length relative to the tagged target edges that you specify. This is depicted in Figure 4-58.

Figure 4-58. orient parallel and orient perpendicular

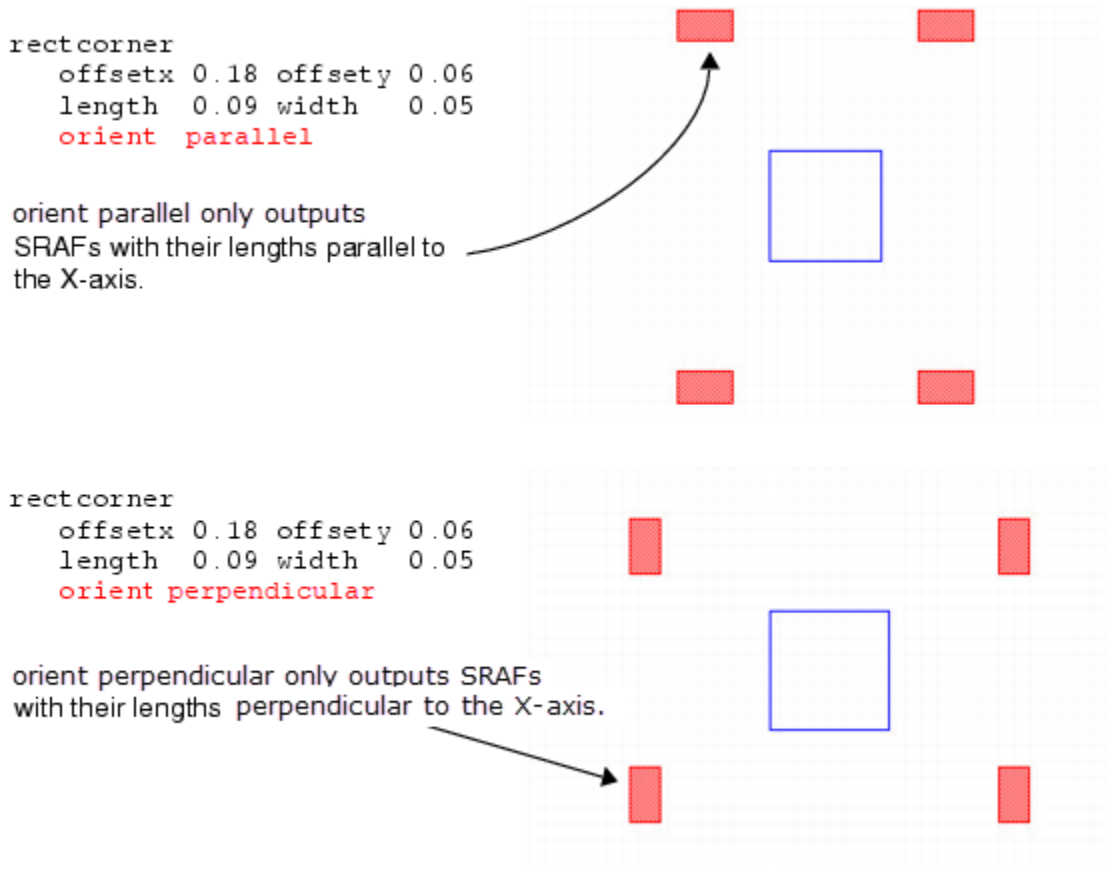


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4 — Orient for Square Target Shapes

The behavior of the orient keyword for treatment of square target shapes results in SRAFs placed with the orientations shown in Figure 4-59. The parallel option outputs SRAFs parallel to the X-axis. The perpendicular option outputs SRAFs perpendicular to the X-axis.

Figure 4-59. perpendicular and parallel Treatment of Square Target Shapes

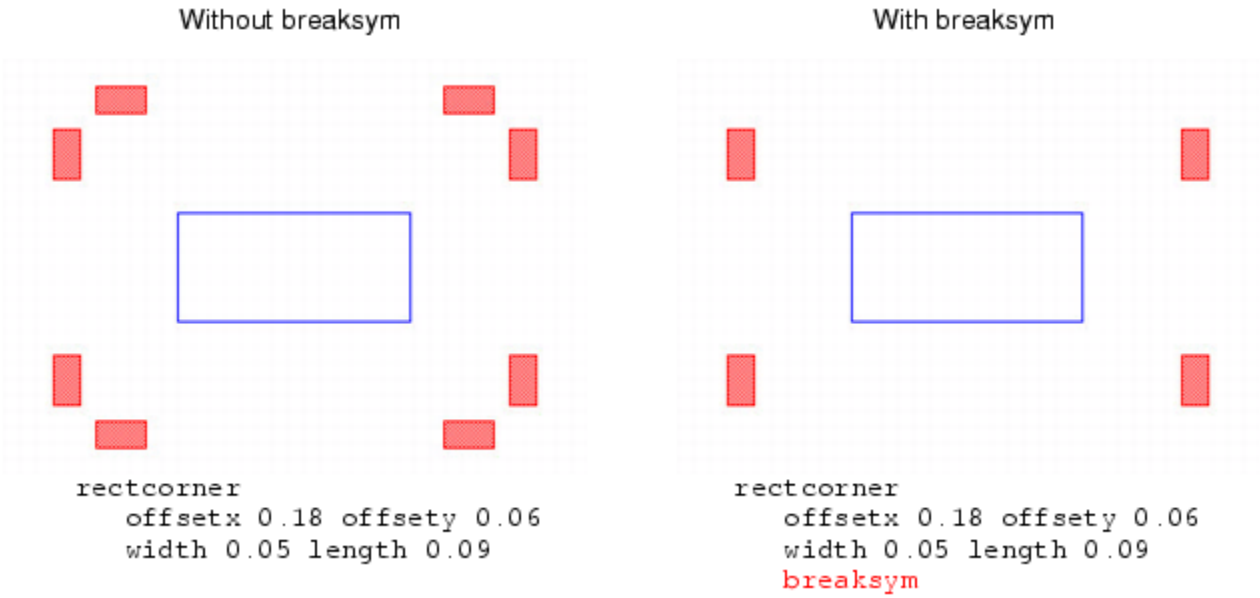


Example 5 — breaksym Usage

The breaksym option outputs only the vertically-oriented SRAF of the default SRAF placement.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-60. rectcorner Using breaksym



Example 7 — scalefactor Indexes, offset Effects, and orient 45

The rectcorner scalefactor usage outputs an array of square SRAFs around each target corner.

Figure 4-61. scalefactor Index Effects

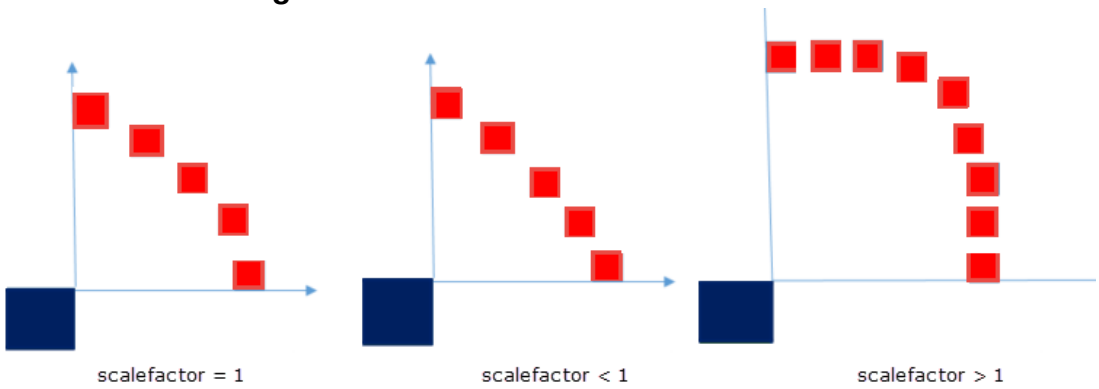
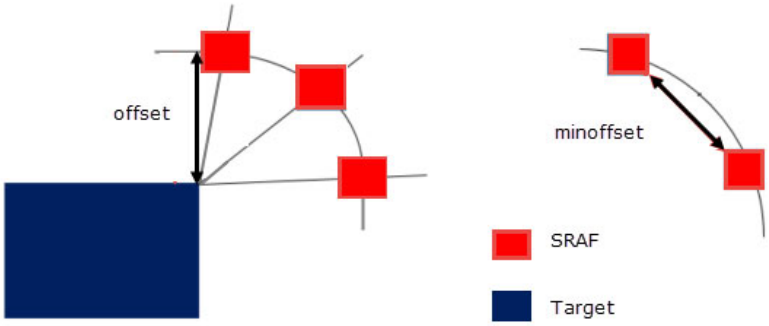


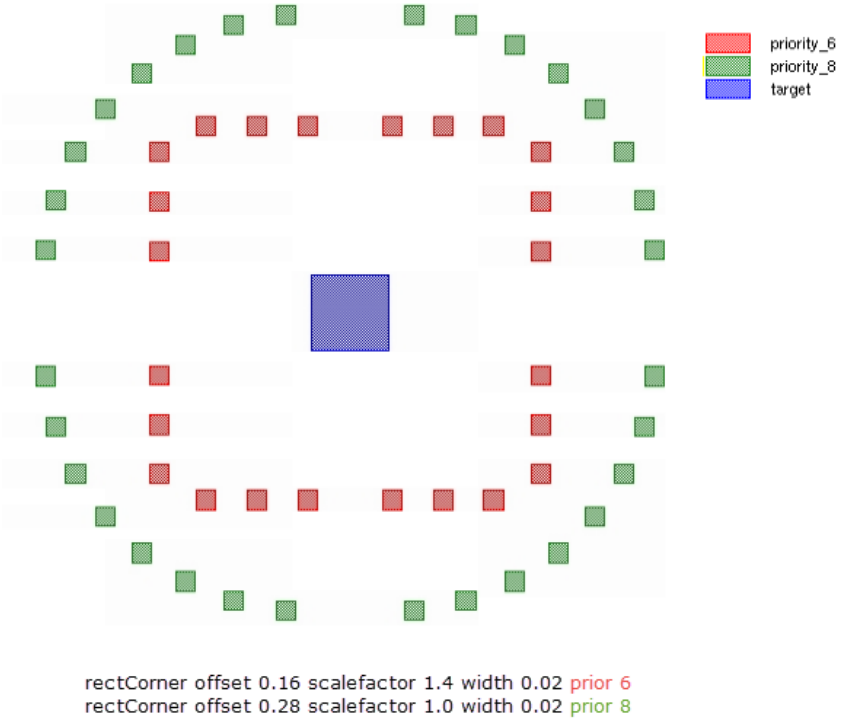
Figure 4-62. scalefactor offset and minoffset Effects



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

In the following illustration, scalefactor is called twice using two rectcorner templates with different scalefactor indexes, offsets, and priorities to differentiate the output.

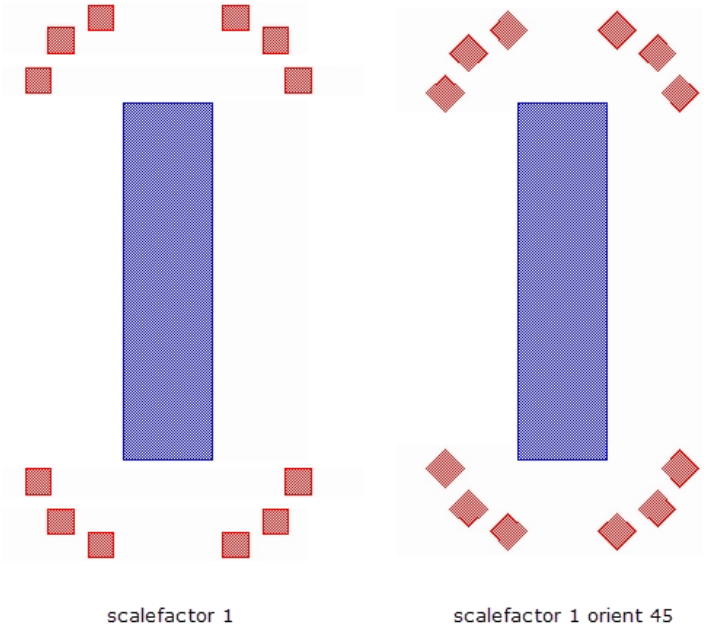
Figure 4-63. scalefactor Called by Multiple rectcorner Templates



In the following figure, scalefactor results are shown on the left illustration without using orient 45 and the right with orient 45. The output SRAFs in the right illustration are all rotated 45 degrees with respect to the target shape. The offset is from the target corner to the center of the rotated SRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

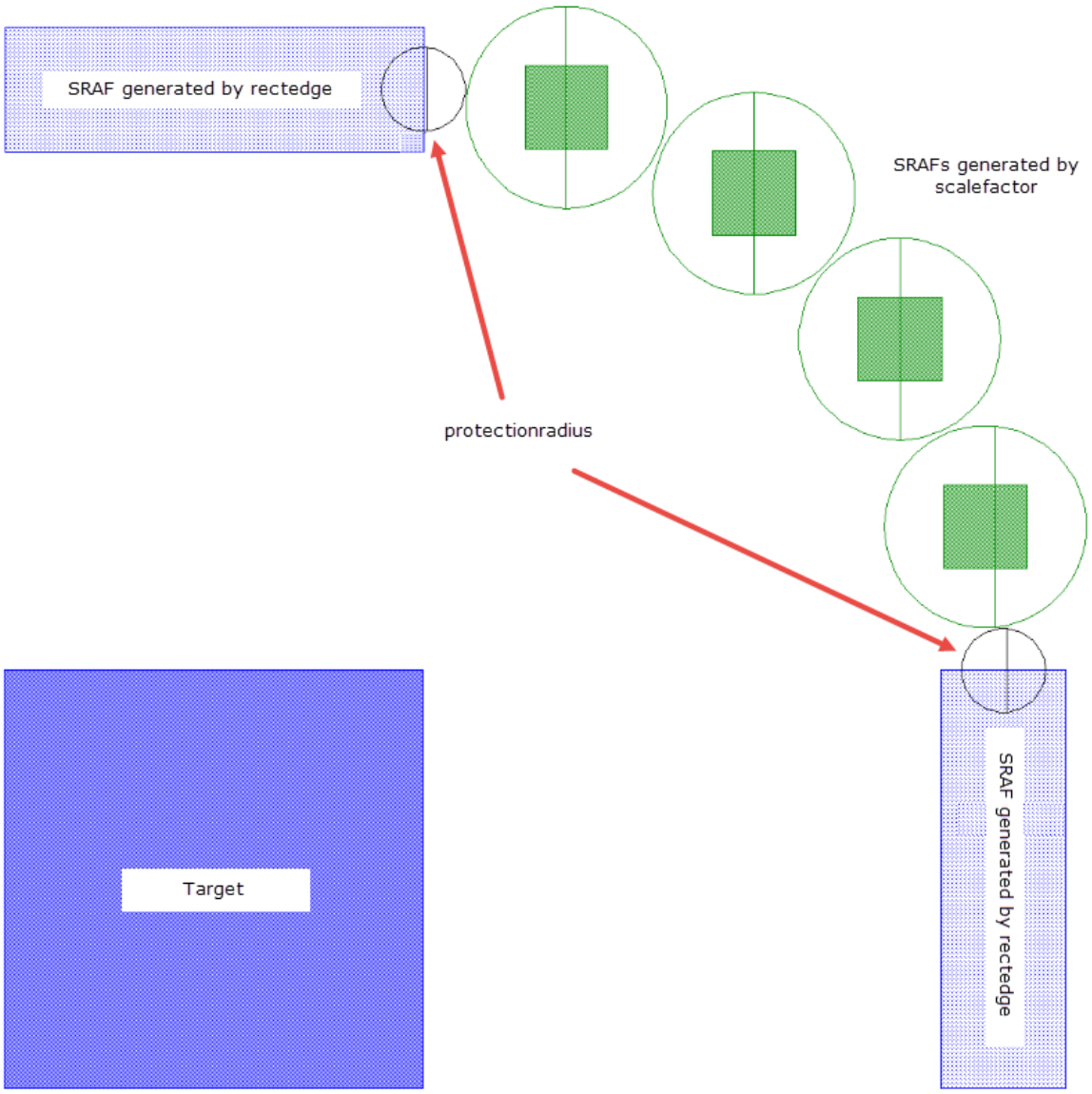
Figure 4-64. scalefactor and orient 45



The following illustration depicts the implementation of protectionradius which keeps scalefactor SRAFs MRC-clean.

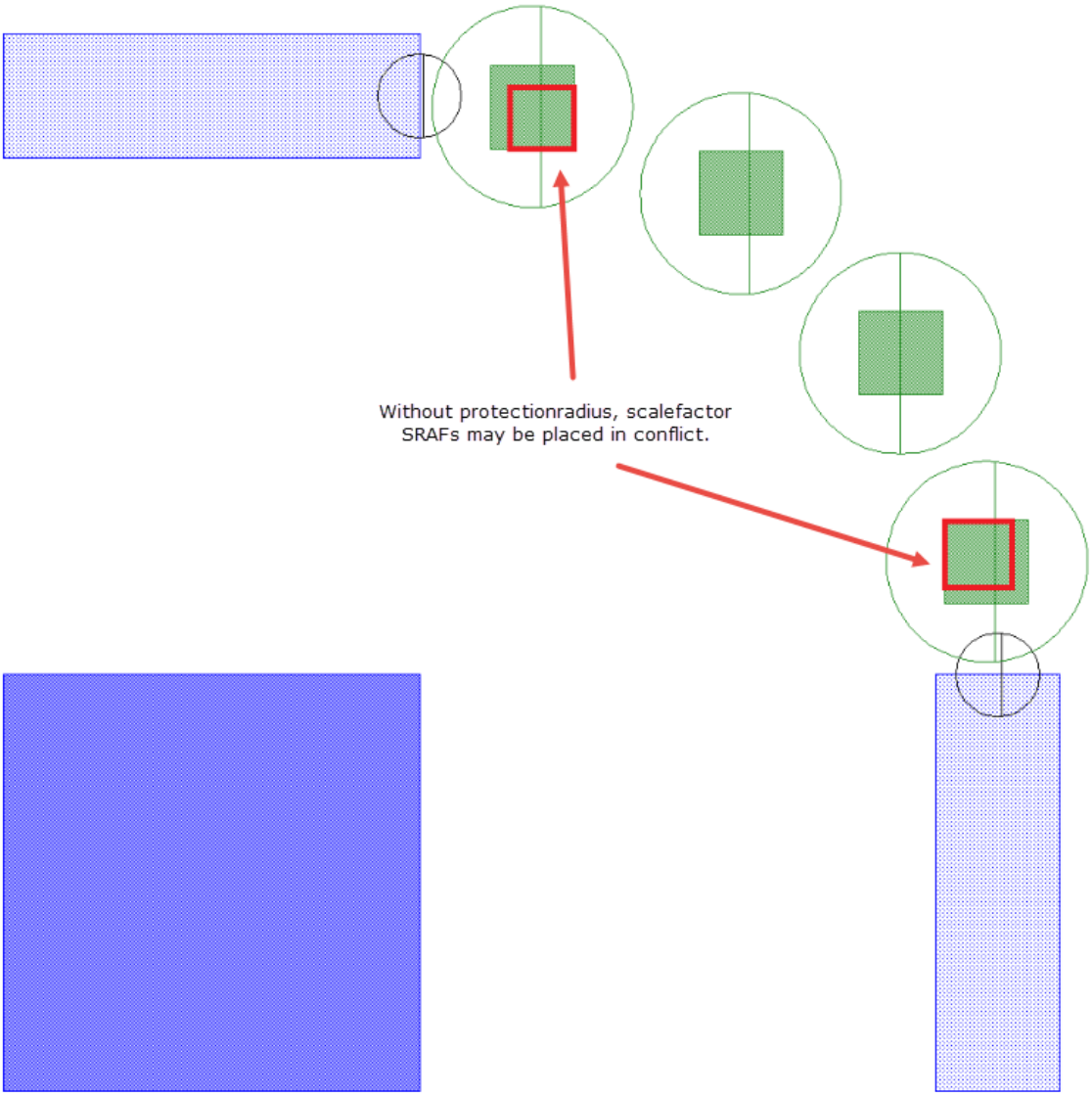
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-65. With protectionradius



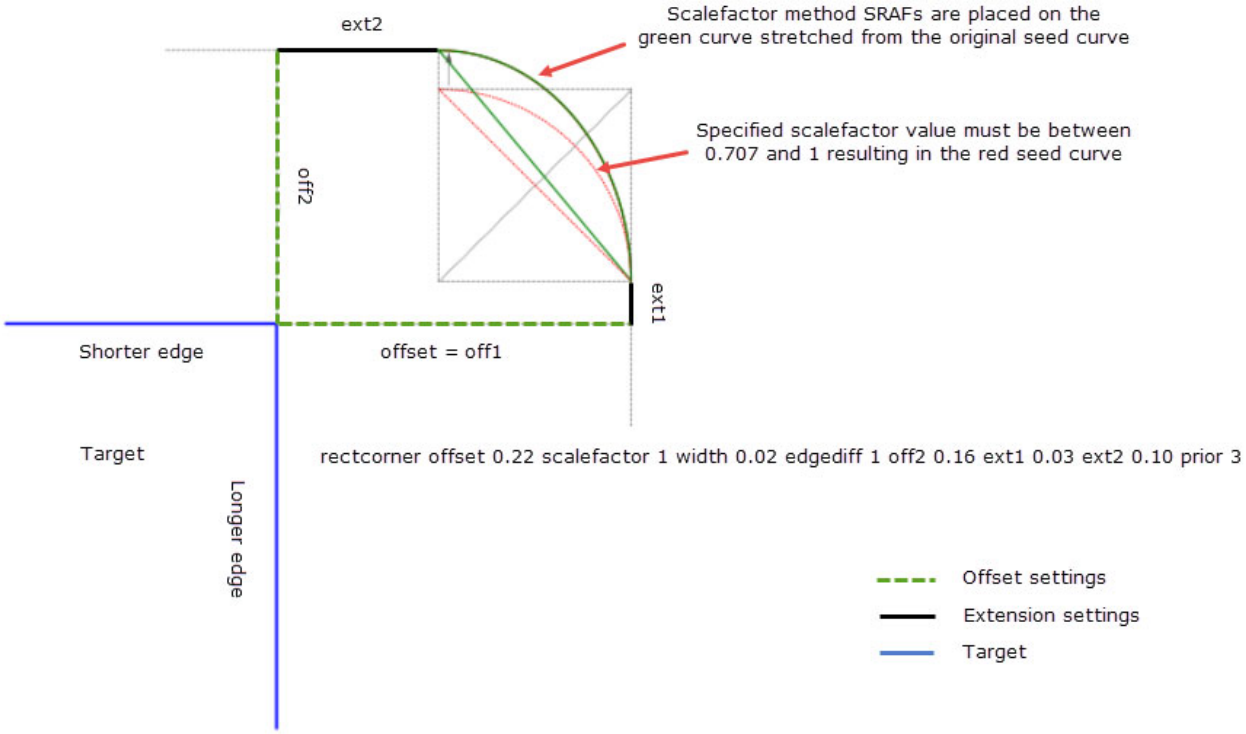
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-66. Without protectionradius



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

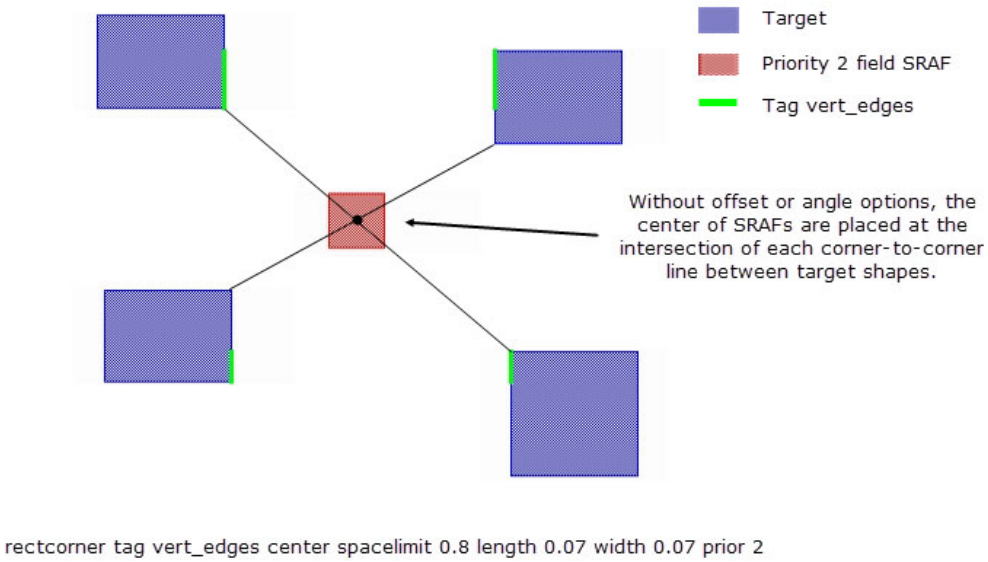
Figure 4-67. scalefactor With Variable Offsets



Example 8 — center offset and angle Effects

The rectcorner center usage generates SRAFs between the edges of qualifying tagged corners. Figure 4-68 shows an SRAF placed as a function of spacelimit:

Figure 4-68. center Without Options



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-69 demonstrates how the angle option generates SRAFs from a two-corner position:

Figure 4-69. center With angle Keyword

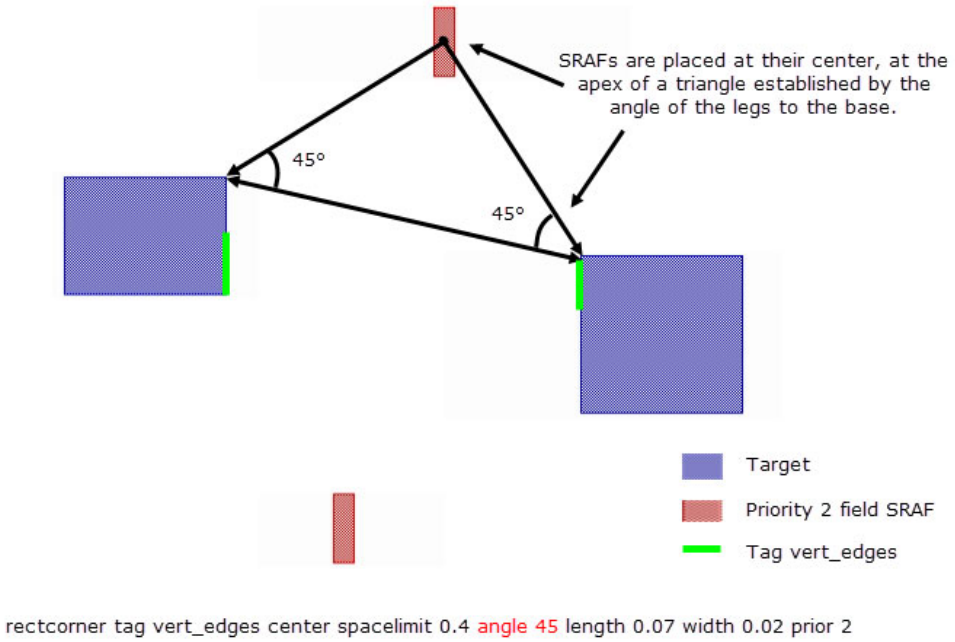


Figure 4-70 demonstrates how the offset option generates SRAFs from a two-corner position:

Figure 4-70. center With offset Keyword

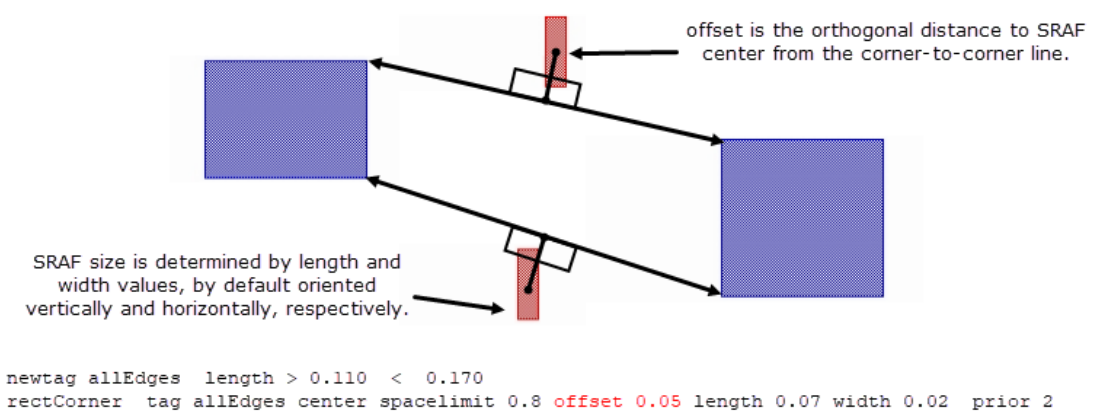
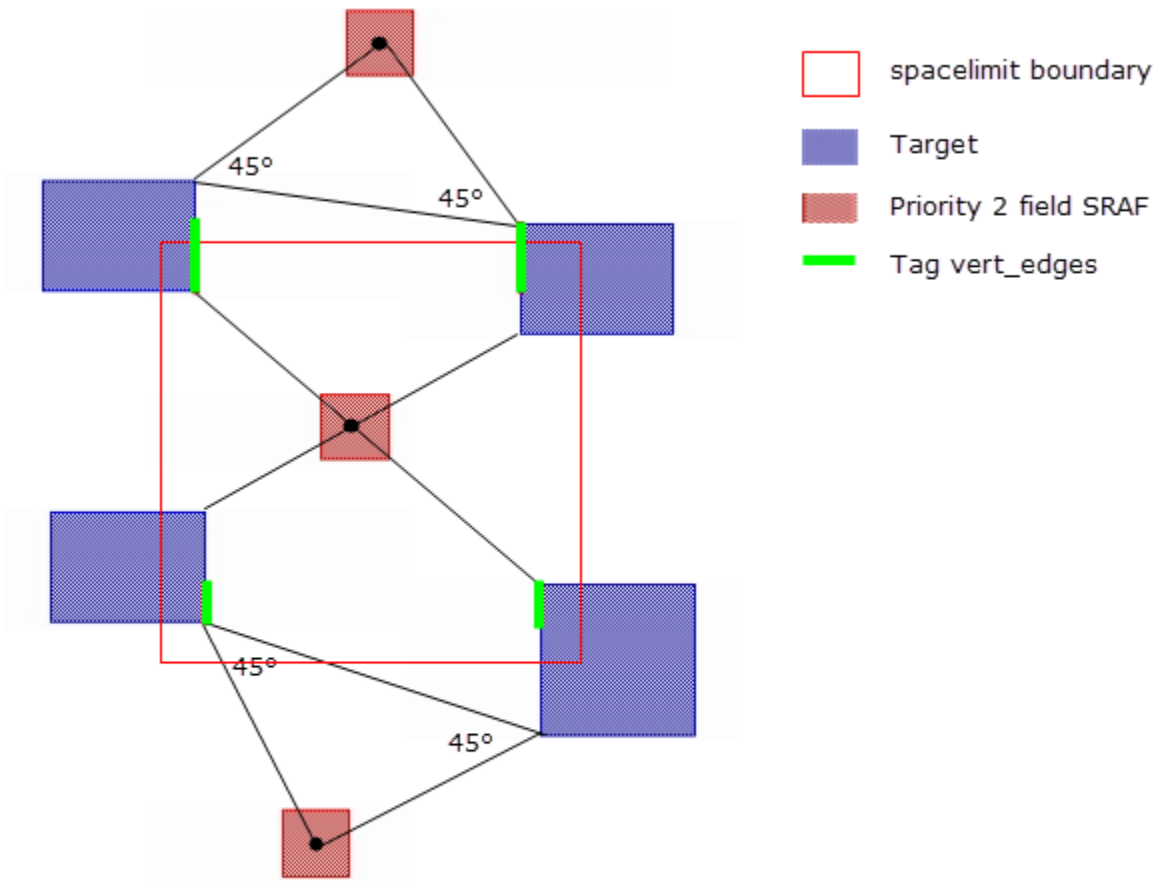


Figure 4-71 demonstrates that angle 45 is default behavior: both rectcorner templates in the illustration perform the same SRAF generation. By default, spacelimit selects all corners within its boundary and places a centered SRAF equidistant to all corners.

However, when strictcenter is specified in conjunction with angle 22, only the tagged portions of target shape edges at corners are used to place SRAFs as seen in Figure 4-72. In this case, SRAFs are placed at 22 degrees from the tagged corners.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-71. Default Behavior of center



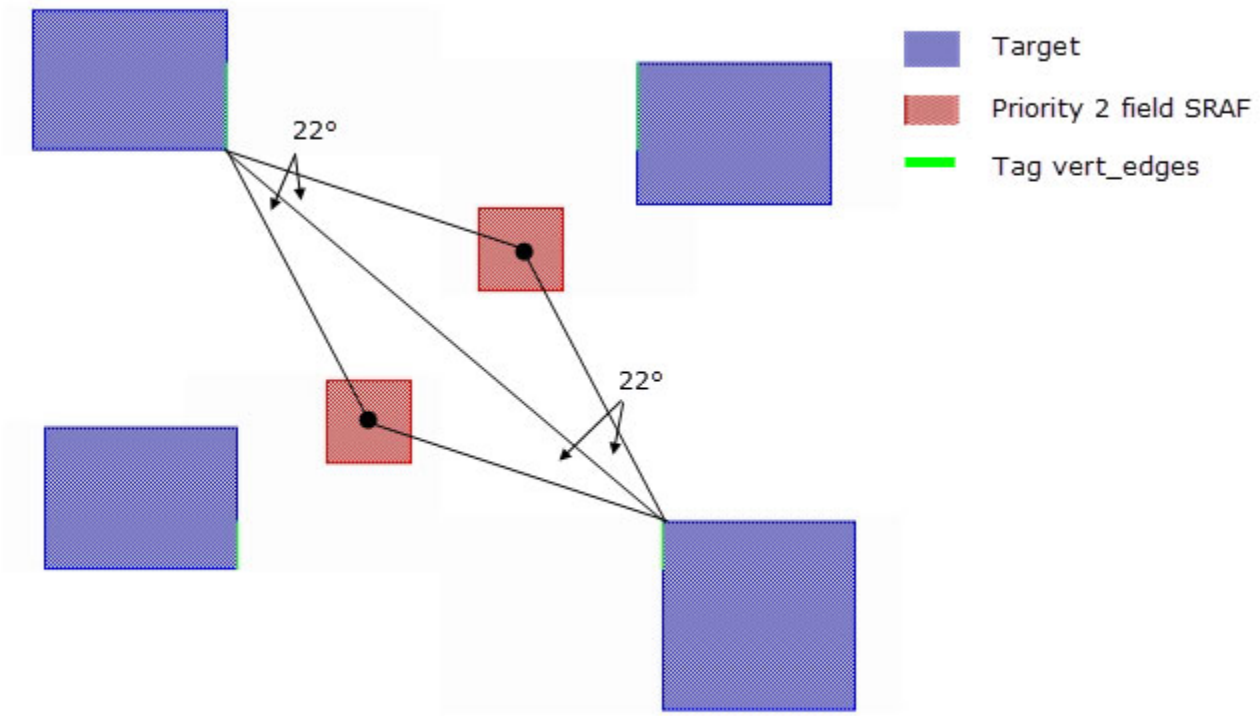

```
newtag vert_edges external > 0.1 < 0.5
rectcorner tag vert_edges center spacelimit 0.8 length 0.07 width 0.07 prior 2
```

or

```
rectcorner tag vert_edges center spacelimit 0.8 angle 45 length 0.07 width 0.07 prior 2
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-72. center strictcenter




```
rectcorner tag vert_edges center spacelimit 0.4 twotagcenter angle 22  width 0.07 length 0.07 prior 2
```

Figure 4-73 shows the valid space in which SRAFs can be placed relative to the bounding edges projected by the two target corners. The voronoi edge is established as a bisecting line 90 degrees to a line connecting the two target corners. Using the center multicorner method, all generated SRAFs are placed on the voronoi edge. Where offset is set to 0, the center of the SRAF is also placed on the target corner connecting line, as seen in the right-hand figure. The SRAF generated in the left-hand figure is outside the placement space and is deleted.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-73. Voronoi-Based SRAF Placement

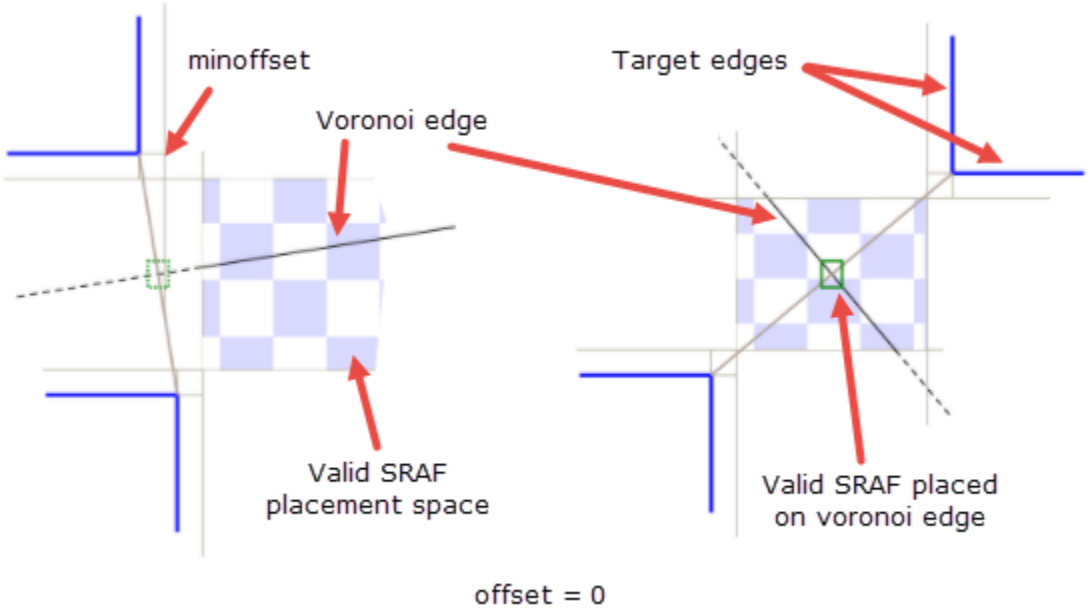


Figure 4-74 shows the center of the generated SRAF placed at an offset of 0.02 from the target corner connecting line. The right-hand figure shows the voronoi SRAF space accommodating two SRAFs equidistant from the target corner connecting line. The left-hand figure also shows two generated SRAFs, but only one within the placement space is kept.

Figure 4-74. Voronoi SRAF Placement Using offset

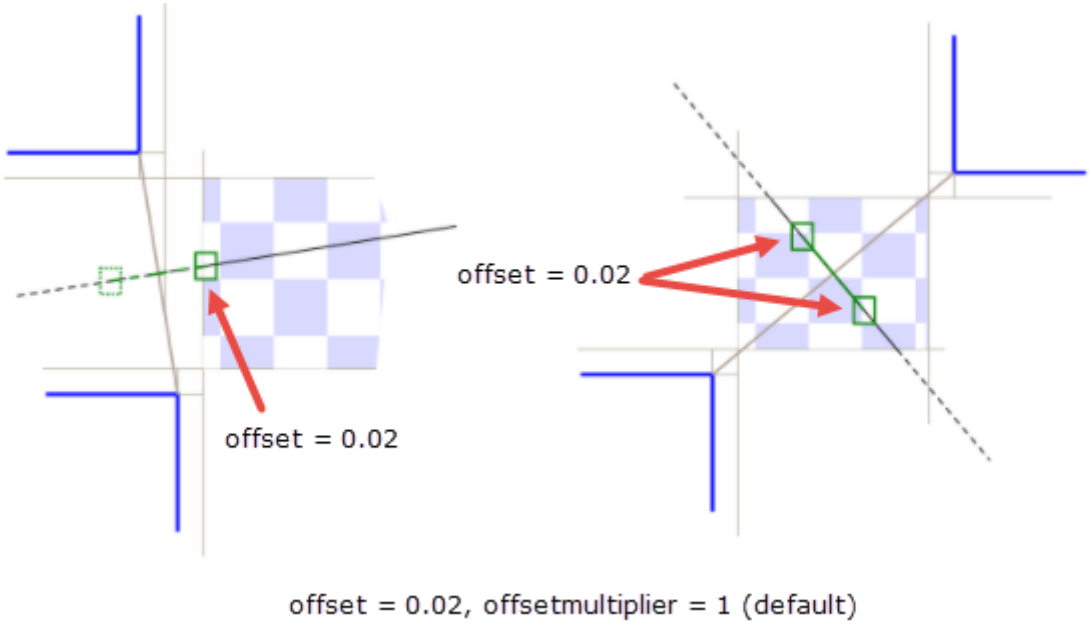
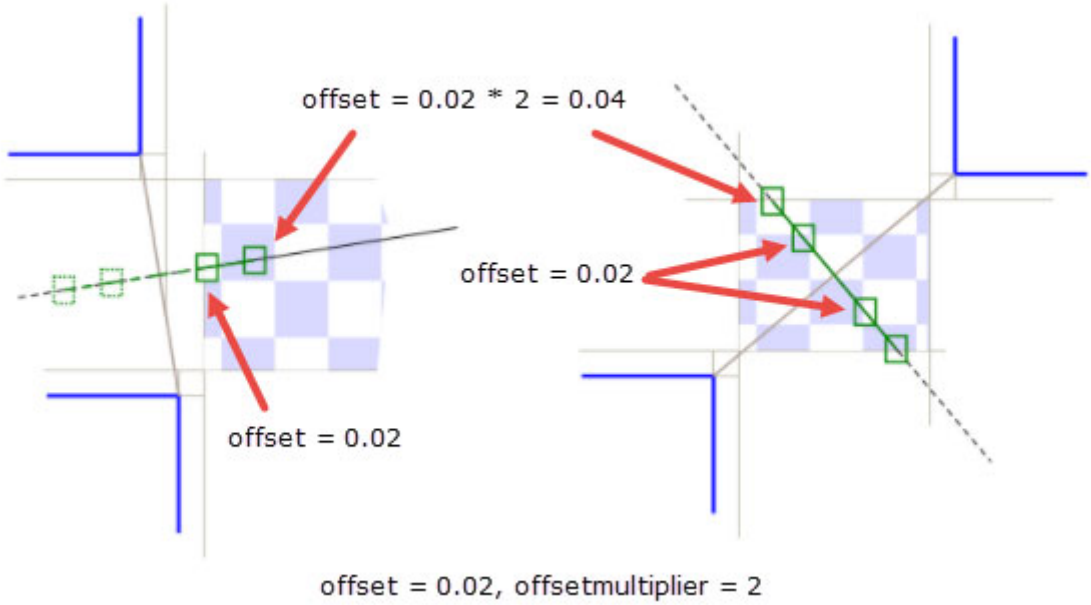


Figure 4-75 shows results when using offsetmultiplier specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-75. Voronoi SRAF Placement Using offsetmultiplier



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rectdiag

Templates

Places four or eight rectangular SRAFs diagonally offset relative to the center of the target shape. Used by RET SBAR only.

Usage

rectdiag
[tag *name* ...]
{**length** ***value*** **width** ***value***}
{**pitchx** ***value*** **pitchy** ***value***}
[breaksym]
[conflictonly]
[orient {horizontal | vertical}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value*** **width** ***value***

  Two required keywords, length and width, that specify the longer and shorter edges of a rectangular SRAF, respectively. The length and width values can be identical.

- **pitchx** ***value*** **pitchy** ***value***

  A required keyword pair that specifies the horizontal and vertical distance from the center of the target shape to the center of the SRAF, respectively.

- breaksym

  An optional keyword that prevents the template from generating symmetric target shapes. By default, the following rectdiag generates symmetric SRAFs:

  ```
  rectdiag width 0.030 length 0.100 pitchx X pitchy Y
  ```

  but also generates SRAFs at these symmetric locations (where X and Y are reversed):

  ```
  rectdiag width 0.030 length 0.100 pitchx Y pitchy X
  ```

  To prevent this symmetric generation, use breaksym (which also works for reversed X and Y pitches):

  ```
  rectdiag width 0.030 length 0.100 pitchx X pitchy Y breaksym
  ```

- conflictonly

  An optional keyword that outputs SRAFs within conflict regions only; no other SRAFs of this template are generated. Conflict regions are formed where neighboring SRAFs

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

generated by this template violate MRC rules you specify in the template. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using the conflictonly argument only when conflict resolution shapes are relatively small and not for larger shapes such as line layer designs.

- orient horizontal | vertical

  An optional keyword that enables the rectdiag template to place SRAFs oriented to the aspect of rectangular target shapes. Specifying orient horizontal applies templates to rectangular target shapes as if they were oriented horizontally (the long edge of the rectangle is parallel to the x-axis). Specifying orient vertical applies templates to rectangular target shapes as if they were oriented vertically (the long edge of the rectangle is parallel to the y-axis).

  Note

  If a template is symmetric or symmetry is not broken by the breaksym keyword, then there is no need to specify the orientation. For additional details regarding the usage of the orient keyword, see Figure 4-81.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that symmetrically places four or eight rectangular SRAFs, depending on pitch value, located diagonally from the center of the target shape. This template cannot generate negative SRAFs.

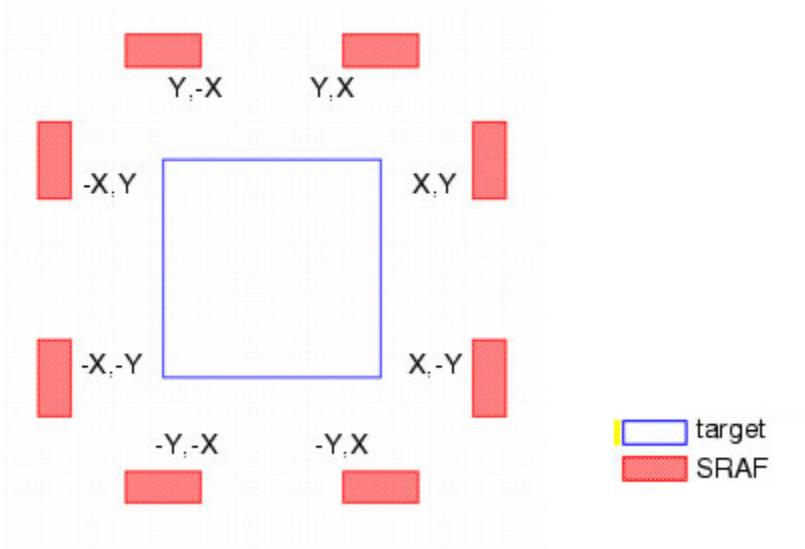
Examples

Example 1

Each SRAF is placed at the relative coordinates shown in Figure 4-76. All SRAFs are placed by specifying only two pitch values. The coordinate permutations are established automatically.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

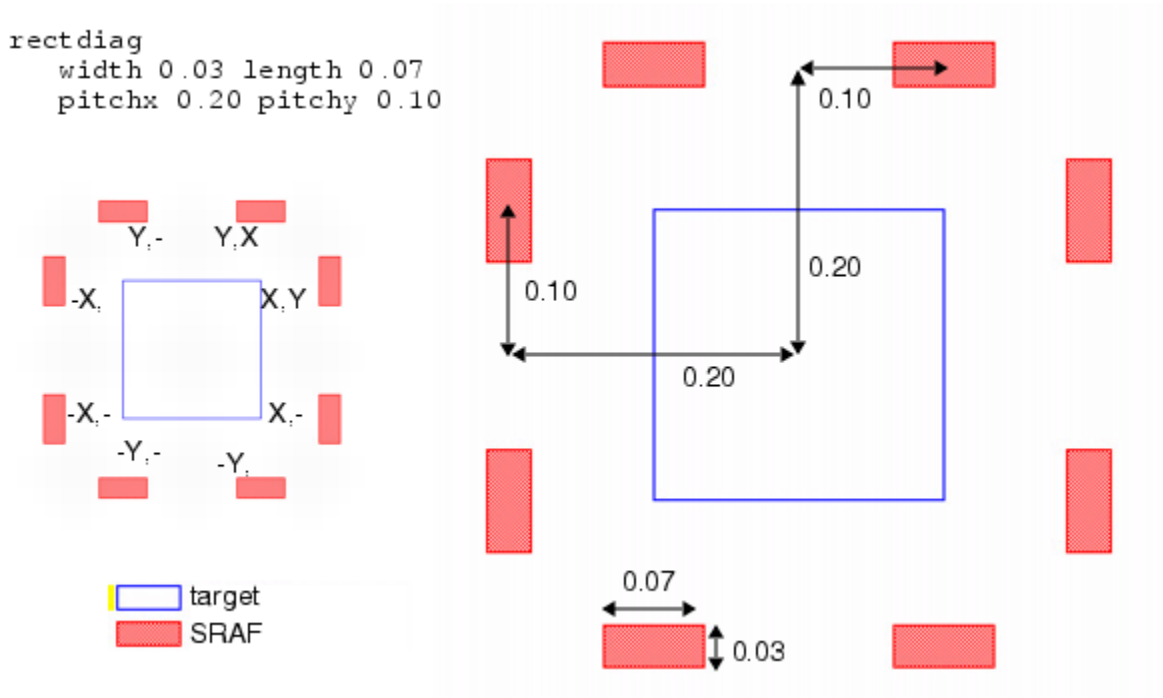
Figure 4-76. rectdiag SRAF Placement Behavior



Example 2

The SRAFs shown are placed by specifying pitchx and pitchy values of 0.20 and 0.10, respectively. The coordinate permutations are applied to the pitch values automatically for the rectdiag template.

Figure 4-77. rectdiag Default Usage

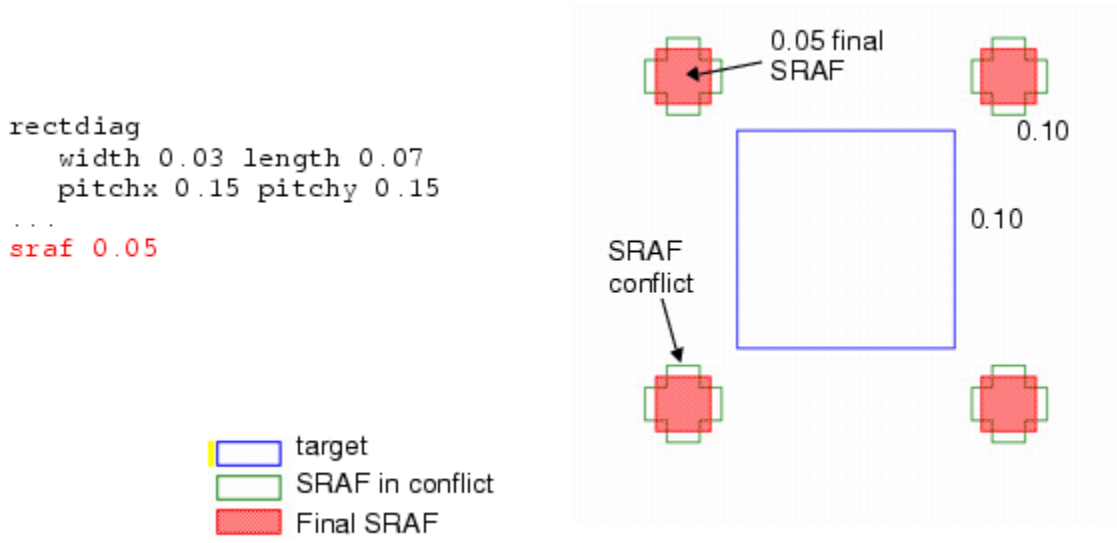


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 3

Setting identical pitchx and pitchy values results in SRAF conflicts. Subsequent square SRAFs can be output in those conflict regions by the specification of the sraf keyword. Generating four SRAFs with rectdiag requires either identical pitches in conjunction with the sraf keyword for conflict cleanup, or the use of the rectdiag breaksym option. Resulting SRAFs differ between these two approaches. These differences are shown in Figure 4-78 and Figure 4-80 on page 213, respectively.

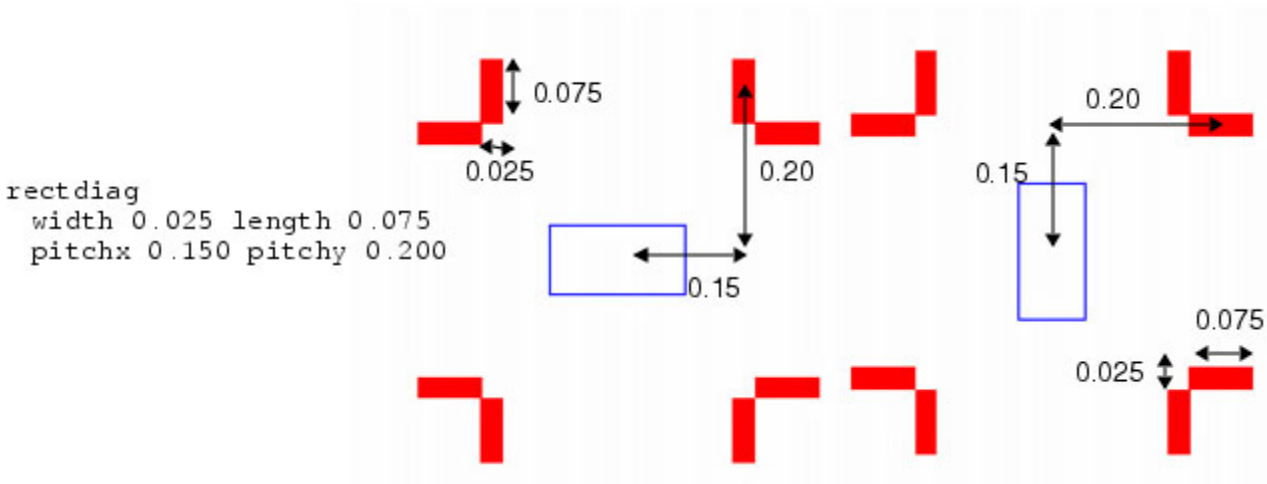
Figure 4-78. rectdiag Usage With Identical X and Y Pitches



Example 4

The default usage of the rectdiag template outputs SRAFs at all orientations irrespective of axis alignment or target shape aspect ratio.

Figure 4-79. rectdiag Rectangle SRAF Placement

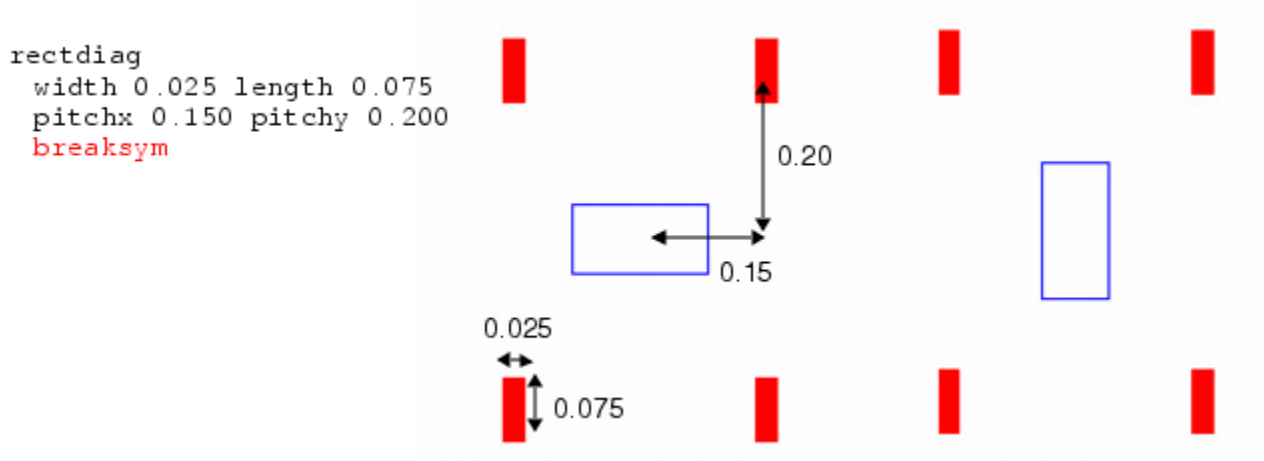


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 5

The breaksym option only outputs SRAFs with Y-axis alignment, as shown in Figure 4-80.

Figure 4-80. rectdiag SRAF Placement With breaksym

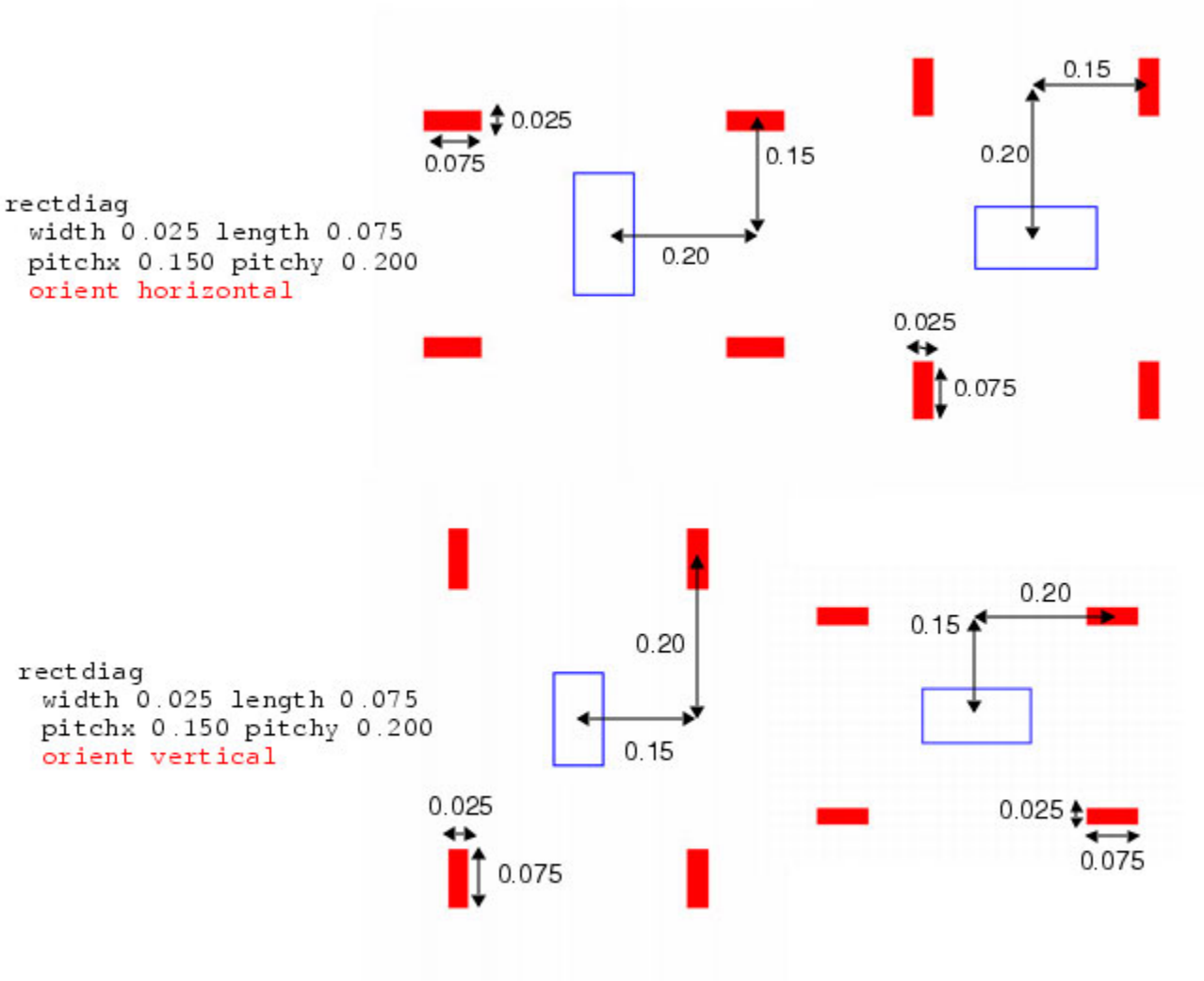


Example 6

As depicted in Figure 4-81, the orient horizontal option places SRAFs perpendicular relative to the aspect ratio of the target shape, and orient vertical places SRAFs parallel relative to the aspect ratio of the target shape. These relative SRAF orientations remain consistent regardless of the orientation of the target shapes. If the target shape is square, orient places SRAFs either horizontally or vertically relative to the target shape.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-81. rectdiag SRAF Placement Using orient



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rectedge

Templates

Places rectangular SRAFs offset from each edge of the target shape. Used by RET SBAR only.

Usage

rectedge
[tag *name* ...]
{**length** ***value*** | **extension** ***value*** [extension45 *value* [width45 *value*]]}
{**center** [*value*] [spacelimit *value*] |
offset ***value*** |
{**offsetx** ***value*** **offsety** ***value***}}
width ***value***
[angledalso]
[conflictonly]
[strictcenter {0 | 1}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value***

  One of two required keywords (length or extension) that must specify either the length or the extension of rectangular SRAFs, respectively.

  The length keyword determines the length of rectangular SRAFs.

  The length keyword is mutually exclusive with the center keyword.

- **extension** ***value*** [extension45 *value* [width45 *value*]]

  One of two required keywords (length or extension) that must specify either the length or the extension of rectangular SRAFs, respectively.

  - ***value***

    The extension keyword and value pair adds to, or subtracts from, the length of rectangular SRAF ends based on the original target edge. Negative extensions that reduce SRAF length are valid if the specified extension is less than half of the target edge length and the resulting SRAF does not violate MRC constraints.

  - extension45 *value*

    The optional extension45 keyword and value pair controls the extension of SRAFs that are 45 degrees relative to target shapes. To avoid SRAF removal, the minimum

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

value of extension45 must be greater than or equal to width45/sqrt(2). For example, where width45 equals 20 nm:

extension45 length = 20/sqrt(2) = 14.14 nm

For additional control of 45-degree SRAF extensions, use the minareaextension and minlengthextension keywords.

- width45 *value*

  The optional width45 keyword and value pair controls the widths of those SRAFs that are 45 degrees relative to target shapes for extension45. The width45 option defaults to width. The width45 keyword adheres to upper and lower bounding values, between width/sqrt(2) and width*sqrt(2). For example, where width equals 18 nm:

  width45 range = [12.73, 25.4]

Note

Regarding extension45 keyword and cleanupversion interactions:

- Siemens EDA does not yet recommend use of intersection J or lineendmerge with the extension45 option. Unexpected results may occur.
- The rectedge extension45 keyword is incompatible with cleanupversion 0 and results in a compile-time error.

- **center** [*value*] [spacelimit *value*]

  One of three mutually exclusive, required keyword or keyword pairs (center, or offset, or offsetx and offsety) that must be specified for the distance SRAFs are placed from the edge of a target shape.

  The center keyword forces the output SRAFs to be placed equidistant from both qualifying edges provided by tags or the target shape edges. See Figure 4-84 on page 220. If the tags input to rectedge do not provide a pair of qualifying edges, no SRAFs are output when specifying the center keyword.

  The center and length keywords are mutually exclusive.

  - *value*

    A center value may be optionally specified as a positive floating-point number, specifying the spacing from the centerline between facing target edges to the inner edges of the centered pair of parallel SRAFs that are generated. For more information, see Figure 4-85 on page 220.

  - spacelimit *value*

    An optional keyword specified to establish the upper limit for center SRAF generation between one tagged edge and one untagged edge. The spacelimit keyword is compatible only with the euclidean metric.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

Excessively large spacelimit values trigger larger edge search areas and significantly increase run time.

Caution

Regarding center keyword and cleanupversion interactions:

- The rectedge center keyword and its options are incompatible with cleanupversion 0 and result in a compile-time error.
- The spacelimit option is compatible only with the euclidean metric. If other metrics are required, use external to derive tags.
- If cleanupversion 1 is specified, any external or internal newtag upper boundary specified overrides spacelimit, irrespective of the values specified.
- An upper boundary must be specified for newtag internal when implementing it for rectedge center applications.

For cleanupversion 1 shown in the example, the upper limits, 0.5 and 0.7, both override spacelimit which is 0.8, and the largest upper limit, 0.7, overrides all other tag upper limits in the rectedge template. The spacelimit specification never overrides any newtag upper limit:

```
cleanupversion 1
newtag tag1 external <= 0.5
newtag tag2 external <= 0.7
rectedge tag tag1 tag tag2 center ... spacelimit 0.8
```

For cleanupversion 1, the spacelimit specification of 0.8 remains in effect for any newtag lower limits, which are 0.15 and 0.09:

```
cleanupversion 1
newtag tag1 external > 0.15
newtag tag2 external > 0.09
rectedge tag tag1 tag tag2 center ... spacelimit 0.8
```

Positive SRAF generation using the center option requires specifying newtag of external type for selection of the outside edges of target shapes for correct SRAF placement. Negative SRAF generation using the center option requires specifying newtag of internal type for selection of the inside edges of target shapes for correct SRAF placement.

- **offset *value***

  One of three mutually exclusive, required keyword or keyword pairs (center, or offset, or offsetx and offsety) that must be specified for the distance SRAFs are placed from the edge of a target shape.

  The offset keyword specifies the minimum orthogonal distance from both horizontal and vertical edges of a target shape. All values specified must be positive floating point values.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **offsetx *value* offsety *value***

  One of three mutually exclusive, required keyword or keyword pairs (center, or offset, or offsetx and offsety) that must be specified for the distance SRAFs are placed from the edge of a target shape.

  The offsetx and offsety keywords specify the minimum orthogonal distance from the horizontal and vertical edges of a target shape, respectively, from the midpoint of the target edge to the midpoint of the nearest SRAF edge. All values specified must be positive floating point values.

- **width *value***

  A required keyword that specifies the width of the SRAF.

- angledalso

  An optional keyword that additionally generates SRAFs at 45-degree target edges for the rectedge template.

- conflictonly

  An optional keyword that outputs SRAFs within conflict regions only. No other SRAFs of this template are generated. Conflict regions are formed where neighboring SRAFs generated by this template violate MRC rules you specify in the template. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using the conflictonly argument only when conflict resolution shapes are relatively small and not for larger shapes such as line layer designs.

- strictcenter {0 | 1}

  An optional keyword controlling center SRAF placement between tagged edges. Specifying strictcenter 1 allows SRAFs to be generated only between tagged edges. Specifying strictcenter 0 ignores tags and allows SRAFs to be generated between all target edges, and also permits center SRAFs to be placed with reference to a single tagged edge and a facing target edge.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that places rectangular SRAFs at an offset from each edge of the target or tagged edges.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

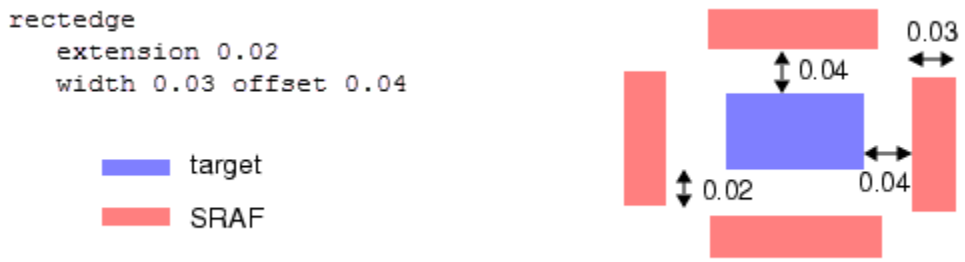
Regarding cleanupversion interactions:

- Siemens EDA recommends always specifying cleanupversion 1 for new recipes.
- The rectedge center and extension45 keywords are incompatible with cleanupversion 0.
- Using the rectedge template with cleanupversion 0 may result in SRAF output differences from recipes using cleanupversion 1. Using cleanupversion 1 implements enhanced algorithms performing more robust classification and identification of tagged edges.

Examples

Example 1 — Basic Usage

Figure 4-82. rectedge Example



Example 2 — Angled SRAF

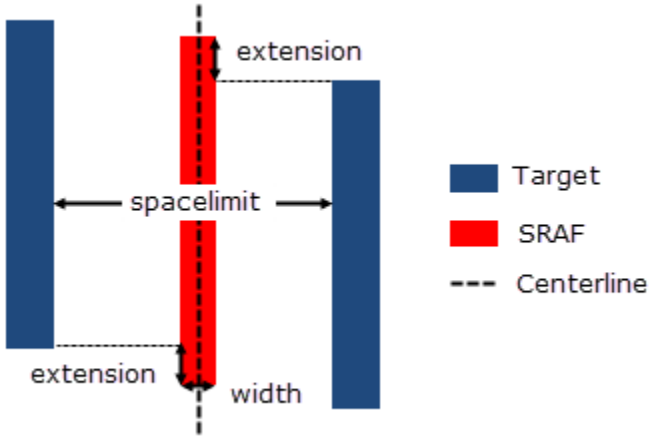
Figure 4-83. angledalso Example

Example 3 — center Usages

Figure 4-84 shows the center keyword used without a value specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

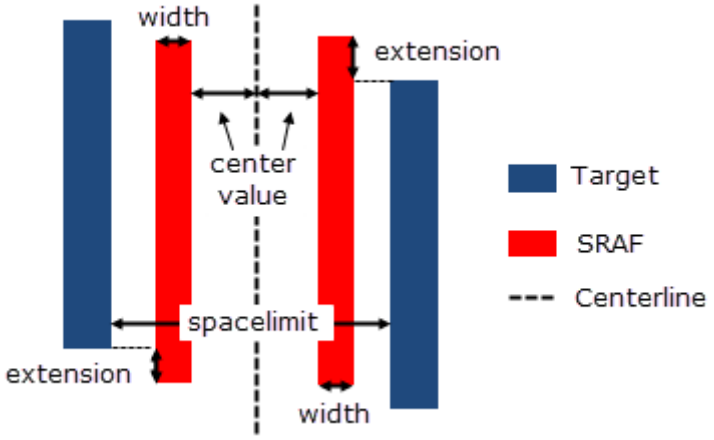
Figure 4-84. center Without Value and spacelimit




```
rectedge width 0.025 center spacelimit 0.5 ...
```

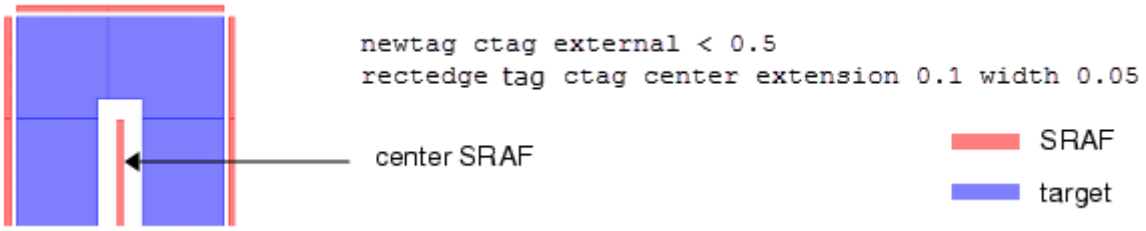
Figure 4-85 shows the center keyword used with the value specified.

Figure 4-85. center With Value and spacelimit




```
rectedge width 0.025 center 0.035 spacelimit 0.5 ...
```

Figure 4-86. center Within a Notch Structure

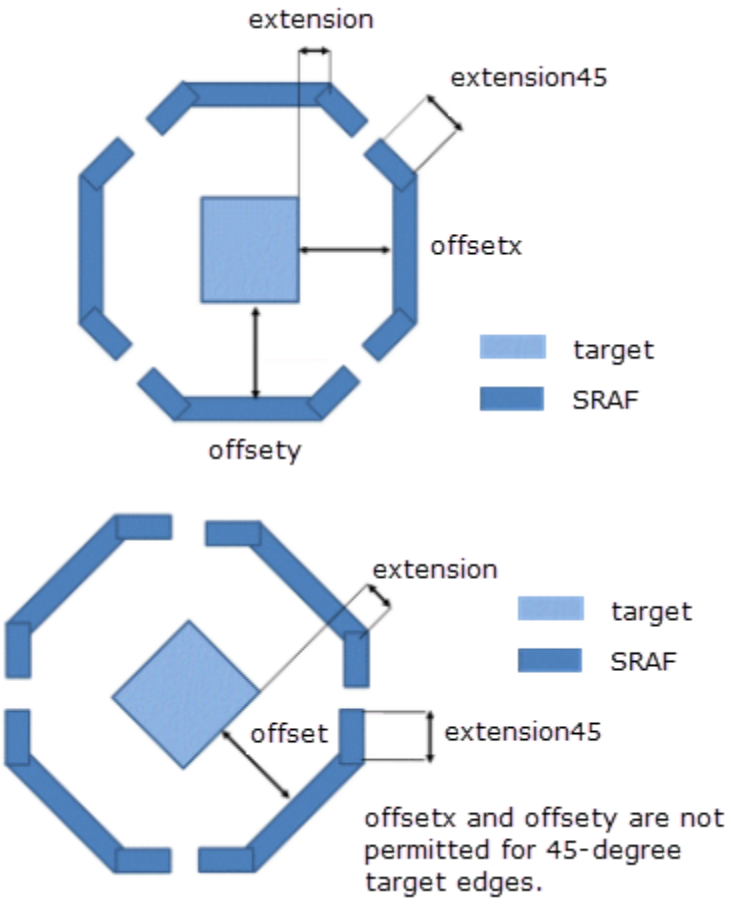


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4 — extension45 Usage

The SRAF portion generated by rectedge extension45 is part of the entire SRAF, and the 45-degree portion is measured with reference from the long edge of the parallel portion of the SRAF towards the target.

Figure 4-87. extension45 Illustrations

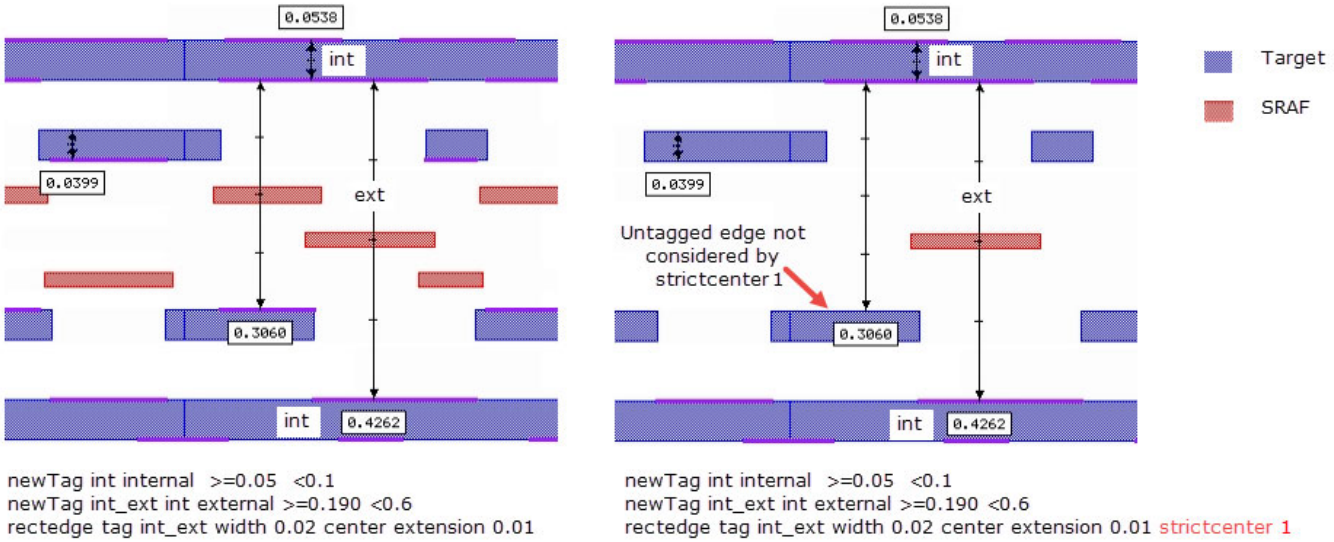


Example 5 — strictcenter

Specifying strictcenter 1 recognizes tagged edges only, those qualified by the newtag command, and does not consider untagged shape edges. The following figure demonstrates the internal tag, int, established and being passed to the second newtag forming an external tag, int_ext, a compound tag. Specifying strictcenter 1 reduces the number of generated SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-88. strictcenter




```
newTag int internal  >=0.05  <0.1
newTag int_ext int external >=0.190 <0.6
rectedge tag int_ext width 0.02 center extension 0.01
```

```
newTag int internal  >=0.05  <0.1
newTag int_ext int external >=0.190 <0.6
rectedge tag int_ext width 0.02 center extension 0.01 strictcenter 1
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

recthv

Templates

Places four rectangular SRAFs at orthogonal X,Y pitches relative to the target shape center. Used by RET SBAR only.

Usage

recthv
[tag *name* ...]
{**length** ***value*** **width** ***value***}
{**pitchx** ***value*** **pitchy** ***value***}
[conflictonly]
[orient {horizontal | vertical}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **length** ***value*** **width** ***value***

  Two required keywords, length and width, that specify the longer and shorter edges of a rectangular SRAF, respectively. The length and width values can be identical.

- **pitchx** ***value*** **pitchy** ***value***

  A required keyword pair that specifies pitchx and pitchy for the horizontal and vertical distance from the center of the target shape to the center of the SRAF, respectively.

- conflictonly

  An optional keyword that outputs SRAFs within conflict regions only. No other SRAFs of this template are generated. Conflict regions are formed where neighboring SRAFs generated by this template violate the MRC rules you specify in the template. The conflictonly argument and dynamicsizing keyword cannot be used within the same priority. Siemens EDA recommends using the conflictonly argument only when conflict resolution shapes are relatively small and not for larger shapes such as line layer designs.

- orient horizontal | vertical

  An optional keyword that enables asymmetric templates to be placed and oriented based on the orientation of non-square rectangles. Specifying orient horizontal applies templates to rectangular target shapes as if they were oriented horizontally (the long edge of the rectangle is parallel to the x-axis). Specifying orient vertical applies templates to rectangular target shapes as if they were oriented vertically (the long edge of the rectangle is parallel to the y-axis).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See "priority" on page 315 for more information.

Description

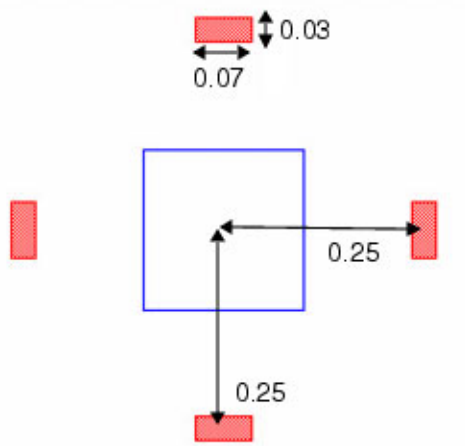
An optional template and argument set that places four rectangular SRAFs at locations of (pitchx,0), (-pitchx,0), (0,pitchy) and (0,-pitchy) relative to the center of a target shape. This template cannot generate negative SRAFs.

Examples

Example 1

Figure 4-89. recthv Identical-Pitch SRAF Placement

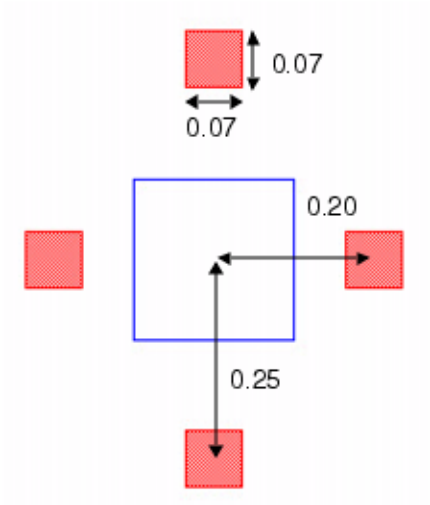
```
recthv
   width 0.03 length 0.07
   pitchx 0.25 pitchy 0.25
```



Example 2

Figure 4-90. recthv Differing-Pitch SRAF Placement

```
recthv
   width 0.07 length 0.07
   pitchx 0.20 pitchy 0.25
```



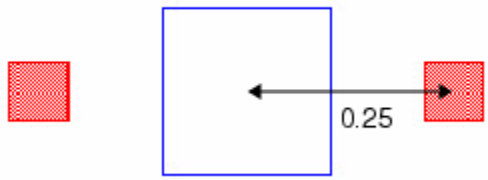
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 3

SRAF placement is shown along the Y-axis aligned (vertical) edges with a pitchy equal to 0.

Figure 4-91. recthv Single Axis SRAF Placement

```
recthv
   width 0.07 length 0.07
   pitchx 0.25 pitchy 0.0
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

srafring

Templates

Places square SRAFs radially around a target shape. Used by RET SBAR only.

Usage

srafring
[tag *name* ...]
circumoffset *value*
pitch *value*
width *value*
[ondiagonal {no | yes}]
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **circumoffset *value***

  A required keyword that specifies the center-to-center distance between square SRAFs. This distance is measured along the arc of the ring, not the straight line distance between square SRAFs.

- **pitch *value***

  A required keyword that specifies the distance from the center of the target shape to the center of the square SRAFs.

- **width *value***

  A required keyword that specifies the width of the SRAFs placed around the square target shape.

- ondiagonal no | yes

  An optional keyword that places the initial SRAFs on 45-degree angles. By default, the initial SRAF is placed on the Y-axis.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that places square SRAFs radially around a target shape. The number of square SRAFs is determined by the circumoffset distance. The center of the

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

SRAFs are placed the specified distance (pitch) from the center of the target shape. This template cannot generate negative SRAFs.

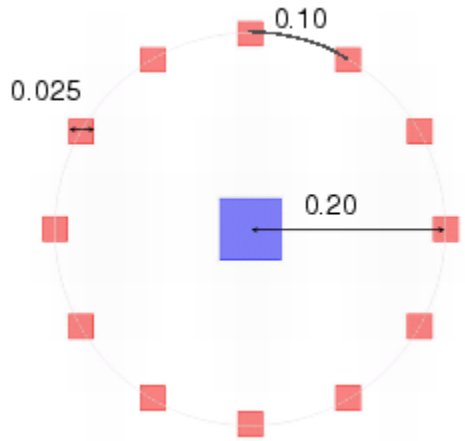
Examples

Example 1

The srafring template places SRAFs by pitch starting at 0 degrees and then computes the circumoffset for all remaining SRAFs.

Figure 4-92. srafring Example

```
srafring
   pitch 0.20 width 0.025
   circumoffset 0.10
```

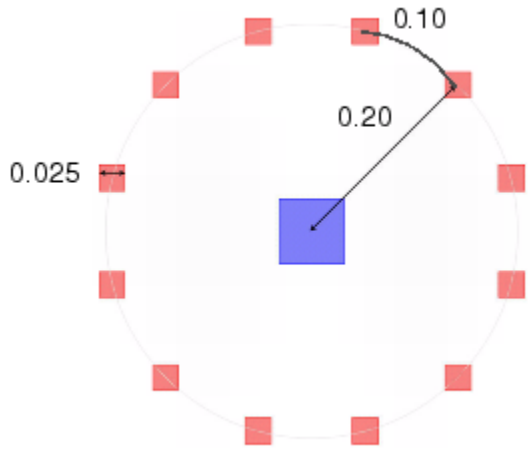


Example 2

The ondiagonal option places one SRAF on each 45-degree angle from the target and then computes the circumoffset for all remaining SRAFs.

Figure 4-93. srafring Using ondiagonal

```
srafring
   pitch 0.20 width 0.025
   circumoffset 0.10
   ondiagonal yes
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

srafringhv

Templates

Places rectangular and square SRAFs radially around a target shape. Used by RET SBAR only.

Usage

srafringhv
[tag *name* ...]
pitch *value* {**rectlength *value* rectwidth *value***}
srafcount *value*
width *value*
[prior *value*]

Arguments

- tag *name* ...

  An optional keyword specifying one or more tag names used in the template. The tag name identifies a group of edges or shapes to be used with the template. If specified, tags must appear as the first arguments in the template.

- **pitch *value***

  A required keyword that specifies the distance from the center of the target shape to the center of the rectangular SRAF.

- **rectlength *value* rectwidth *value***

  A required keyword pair that specifies the length and width, respectively, of rectangular SRAFs.

- **srafcount *value***

  A required keyword that specifies the number of square SRAFs placed between rectangular SRAFs.

- **width *value***

  A required keyword that specifies the width of the square SRAFs placed between rectangular SRAFs.

- prior *value*

  An optional keyword that specifies a numeric identifier for placed SRAFs. Identifiers are applied sequentially, so a value of 1 is applied first, then 2 is applied, and so on. The default priority is 1. See “priority” on page 315 for more information.

Description

An optional template and argument set that places rectangular and square SRAFs radially around a target shape. The center of the SRAFs are placed the specified distance (pitch) from the center of the target shape. This template cannot generate negative SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Figure 4-94. srafringhv Example

```
srafringhv
   pitch 0.220 width 0.035
   srafcount 1
   rectlength 0.160 rectwidth 0.025
   prior 1
```

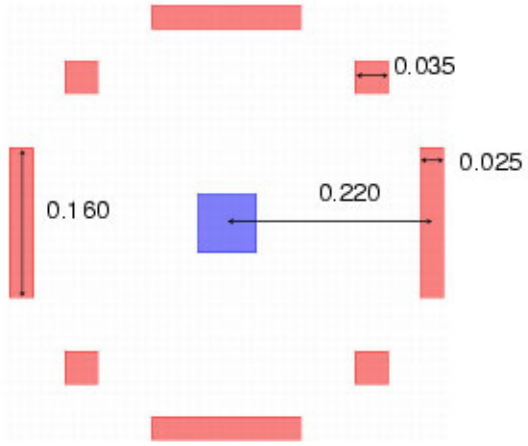
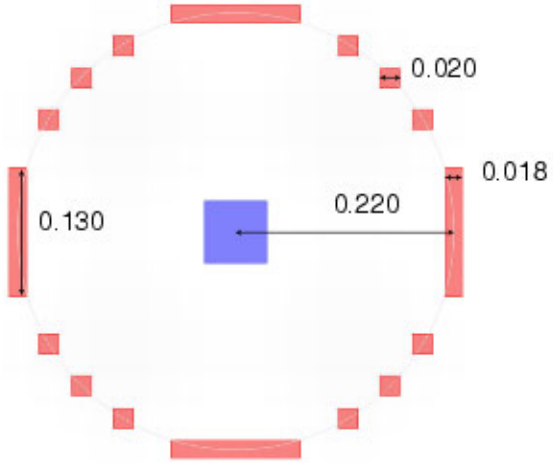


Figure 4-95. srafringhv Using Multiple Square SRAFs

```
srafringhv
   pitch 0.220 width 0.020
   srafcount 3
   rectlength 0.130 rectwidth 0.018
   prior 1
```



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Style Keywords

Provides management of SRAF output style.

For organizational keyword information, refer to "sraf_options" on page 80.

Table 4-11. Style Keywords

Keyword	Description
algorithm	Modifies geometric consistency of generated SRAF candidates. Used by RET MBSRAF only.
centerslit	Disables slit interpolation. Used by RET EUV MBSRAF only.
centerwidth	Enables the generation of SRAFs equidistant from target edges. Used by RET MBSRAF.
collecttype	Collects seed pixels near target corners to generate square SRAFs. Used by RET MBSRAF only.
cornerpref	Specifies the SRAF generation method at target shape corners. Used by RET MBSRAF only.
cornerprefnegative	Specifies the negative SRAF generation method at target shape corners. Used by RET MBSRAF only.
gradientfactornegative	Specifies a gradient factor value for negative SRAFs. Used by RET MBSRAF only.
gradientshift	Specifies a gradient shift value. Used by RET MBSRAF only.
image_options	A LITHO FILE command referencing the *Lithomodel* file and an sraf_options command referencing the litho_model name. Used by RET MBSRAF only.
intersection	Enables valid SRAFs to be merged at various junctions. Used by RET SBAR and RET MBSRAF.
intersectionnegative	Enables valid negative SRAFs to be merged at various junctions. Used by RET SBAR and RET MBSRAF.
jogignore	Overrides jogs in generated SRAFs. Used by RET MBSRAF only.
mode	Specifies a generated SRAF orientation. Used by RET MBSRAF only.
negative	Enables negative SRAF output. Used by RET SBAR and RET MBSRAF.
peaktype	Specifies the method of SRAF propagation. Used by RET MBSRAF only.
pw_condition	Specifies a process window condition. Used by RET MBSRAF only.
slopeweight	Specifies the gradient slope weight. Used by RET MBSRAF only.
symmetric	Ensures all templates generate SRAFs with symmetry. Used by RET SBAR only.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

algorithm

Style Keywords

Modifies geometric consistency of generated SRAF candidates. Used by RET MBSRAF only.

Usage

algorithm 0 | 1

Arguments

- **0**

  Forces model-based SRAF to adhere to the original (pre-2016.3) algorithm.

- **1**

  Invokes model-based SRAF with an updated algorithm.

Description

A required keyword that modifies the geometric consistency of generated SRAF candidates. This is a global setting and cannot be specified per template or per priority.

Note

Regarding algorithm 1 requirements and interactions:

- The algorithm keyword is required as of Calibre release 2018.1. Siemens EDA recommends always using algorithm 1 for new mbSRAF recipes to increase geometric consistency.
- The algorithm 1 operation modifies several internal parameters. The internally-set keywords peaktype, and modes nearestno45 and collectno45, negate performance of algorithm 1 and must not be specified within the same recipe.
- Prior to release 2018.1, the default for maxmove where algorithm 1 was specified, was 0nm. The maxmove default as of Calibre release 2018.1 is 1nm. Therefore, existing recipes using algorithm 1 must explicitly specify maxmove 0 to obtain the previously generated SRAF results.

Setting algorithm 1 requires further explicit specification of two keywords:

- minoffset
- minfeaturespace

The algorithm keyword also sets the defaults for three keywords:

- peaktype (set internally and cannot be modified)
- mode (set internally and cannot be modified)

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- cornerpref (set to no, but may be changed depending on the type of SRAFs required for given models)

Examples

You must specify algorithm as a global keyword.

Example 1 — algorithm 1

```
sraf_options mbopt1 {
   ...
   algorithm 1
   minoffset 0.025
   minfeaturespace 0.040
   ...
}
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

centerslit

Style Keywords

Disables slit interpolation. Used by RET EUV MBSRAF only.

Usage

centerslit *value*

Arguments

- 0 | 1

  Either 0 or 1, where 0 enables and 1 disables slit interpolation, respectively.

Description

The centerslit keyword disables slit interpolation for an EUV global litho model. When centerslit is set to 1, it allows only the center slit to be used during the gradient calculation. The default setting is 0, enabling slit interpolation and through-slit shadowing effects. The centerslit parameter allows use of a litho model that does not use multiple slits, and does not require preparation of a special litho model for Calibre mbSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

centerwidth

Style Keywords

Enables the generation of SRAFs equidistant from target edges. Used by RET MBSRAF.

Usage

centerwidth *value*

Arguments

- *value*

  A positive, floating-point number specified in microns. The value you specify must be greater than the width parameter value.

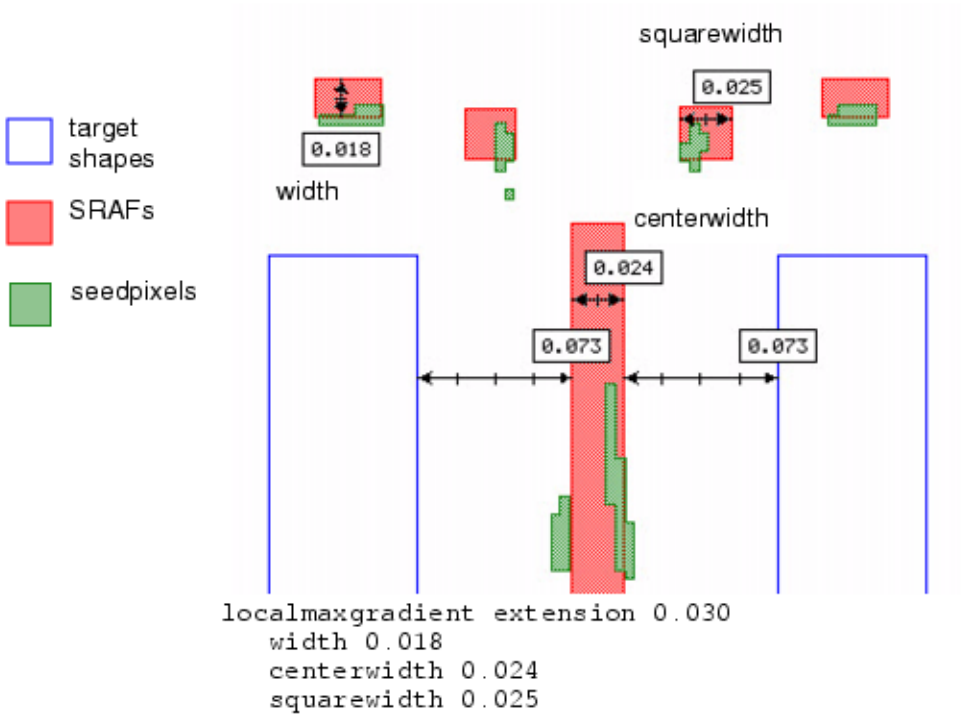
Description

An optional keyword which generates SRAFs equidistant from input target shapes. The centerwidth keyword may be specified globally or in the localmaxgradient template. The centerwidth value must be greater than or equal to width. All localmaxgradient templates of the same priority must specify the same centerwidth value.

Examples

Figure 4-96 illustrates the differentiation of SRAF values possible with the mbsraf localmaxgradient template using centerwidth. Each of width, squarewidth and centerwidth are assigned different values.

Figure 4-96. Using centerwidth with mbsraf



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

collecttype

Style Keywords

Collects seed pixels near target corners to generate square SRAFs. Used by RET MBSRAF only.

Usage

collecttype {0 | 13}

Arguments

- 0

  No seed pixels are collected at corners. This is the default setting for standard SRAF generation.

- 13

  Seed pixels are collected at target corners for square SRAF generation.

Description

An optional keyword, enabled by mode collectno45, which performs seed pixel collection for SRAF generation at corners.

Note

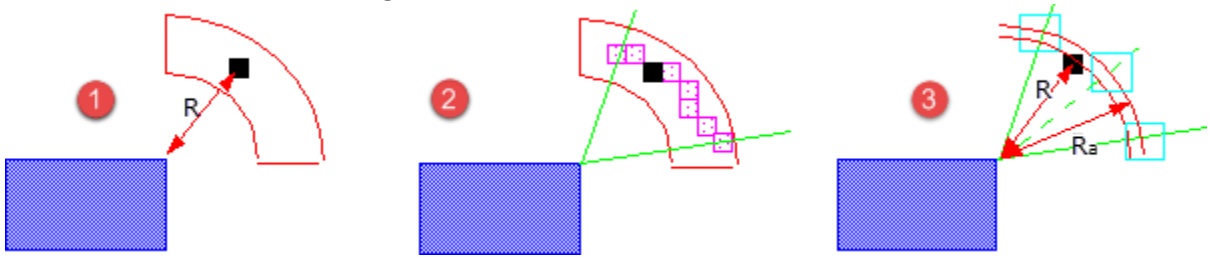
The collecttype keyword is deprecated as of Calibre release 2018.4 and will be obsoleted as of release 2019.2.

If a seed pixel is found near a target corner (not a target edge), an arc of seed pixels is collected around that corner and axis-aligned SRAFs are generated for the entire arc. The collection process for collecttype 13 has several steps demonstrated in Figure 4-97:

1. A 45 degree angle from a target corner is chosen as a reference for the initial SRAF placement.
2. Within this quarter ring, the minimum and maximum angle of all pixels with a local maximum facing the corner are determined.
3. The angular range of the arc (R_a) is then covered by SRAFs maintaining the established minoffset spacing from each other, and the integral of the gradient over each SRAF is computed.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-97. Seed Pixel Collection



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cornerpref

Style Keywords

Specifies the SRAF generation method at target shape corners. Used by RET MBSRAF only.

Usage

cornerpref {all | horizontal | no | square | squareonly | squareonlyscore | vertical}

Arguments

- all

  Generates rectangular SRAFs aligned to all target edge axes. Rectangular SRAFs take priority over square SRAFs with this argument.

- horizontal

  Generates rectangular SRAFs aligned to horizontal target edge axes.

- no

  This method inhibits SRAF generation at corners. If algorithm 1 is set, no is the default setting.

- square

  Generates a square SRAF at seed pixels near the corner of a target shape by the area product of either localmaxgradient keywords width^2 or squarewidth^2. As the localmaxgradient template determines, square SRAFs may be changed to rectangular SRAFs, given necessary alignment criteria. SRAF candidates generated by square are allowed to project from corners into target edges. Cleanup of SRAFs generated by square is score-based (see "Calibre cnSRAF Template Execution" on page 18 for more information regarding score-based cleanup). If algorithm 0 is set, square is the default setting.

- squareonly

  If localmaxgradient keyword squarewidth is defined, squareonly generates square SRAFs at seed pixels near corners or jogs with an area of squarewidth^2. The squareonly option does not allow extension of square SRAF candidates to project into target edge space. Unlike SRAFs generated by the argument square, no SRAFs generated by squareonly can be changed to rectangles. Cleanup of SRAFs generated by squareonly is performed first on square SRAF candidates.

- squareonlyscore

  The squareonlyscore option performs SRAF generation in the same way as squareonly, but performs cleanup in the same way as square. Like squareonly, if localmaxgradient keyword squarewidth is defined, squareonlyscore generates square SRAFs at seed pixels near corners or jogs with an area of squarewidth^2. The squareonlyscore option does not allow extension of square SRAF candidates to project into target edge space. Like SRAFs generated by the argument squareonly, no SRAFs generated by squareonlyscore can be changed to

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

rectangles. Cleanup of SRAFs generated by squareonlyscore is score-based, the same as square.

- vertical

  Generates rectangular SRAFs aligned to vertical target edge axes.

Note

Regarding algorithm interactions:

- If algorithm 0 is set, square is the default cornerpref setting.
- If algorithm 1 is set, no is the default cornerpref setting.

Description

An optional keyword that specifies a single SRAF generation method at target shape corners. This keyword may be specified globally, by priority, or within the localmaxgradient template.

Table 4-12. Candidate Generation and Cleanup by Option

	SRAF Candidate Generation	SRAF Cleanup
square	Allow square candidates from the corners to grow into the edges.	Cleanup by score.
squareonly	Disallow extension of square candidates into edge space.	Cleanup square candidates first not by score.
squareonlyscore	Disallow extension of square candidates into edge space.	Cleanup by score.

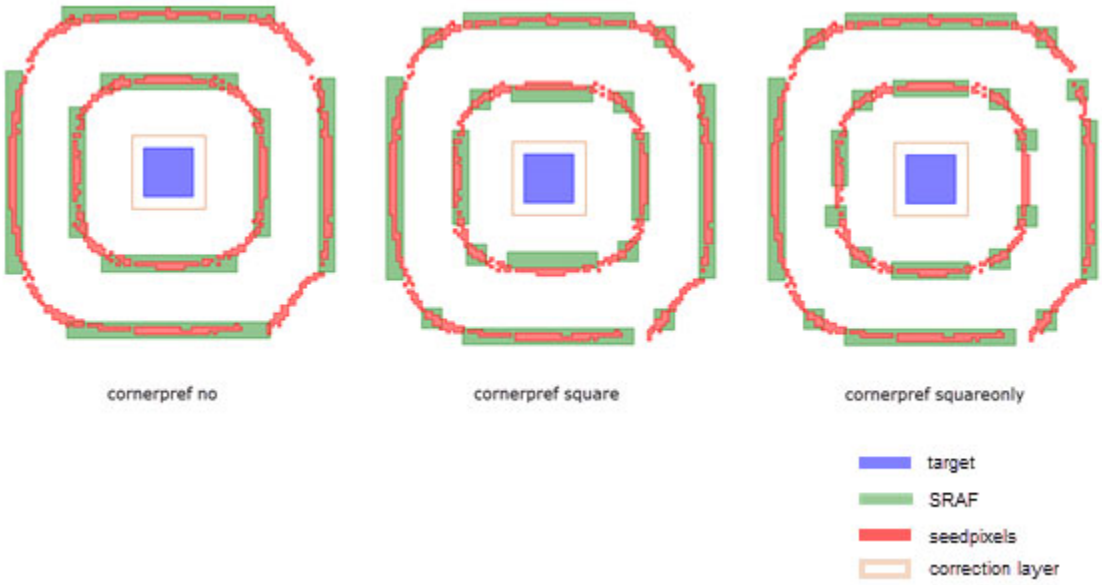
Examples

Some cornerpref Options and Their Results

Some options are shown in Figure 4-98 with their respective SRAF result on an isolated contact.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-98. Some cornerpref Options and Results



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cornerprefnegative

Style Keywords

Specifies the negative SRAF generation method at target shape corners. Used by RET MBSRAF only.

Usage

cornerprefnegative {all | horizontal | no | square | squareonly | squareonlyscore | vertical}

Arguments

- all

  Generates rectangular SRAFs inside target shapes and aligned to all target edge axes. Rectangular SRAFs take priority over square SRAFs with this argument.

- horizontal

  Generates rectangular SRAFs inside target shapes and aligned to horizontal target edge axes.

- no

  This method inhibits SRAF generation inside target shapes at corners. If algorithm 1 is set, no is the default setting.

- square

  Generates a square SRAF at seed pixels inside target shapes near their corners. As the localmaxgradient template determines, square SRAFs may be changed to rectangular SRAFs, given necessary alignment criteria. If algorithm 0 is set, square is the default setting.

- squareonly

  If localmaxgradient keyword squarewidth is defined, squareonly generates square SRAFs at seed pixels inside target shapes near their corners or jogs. Unlike SRAFs generated by argument square, no SRAFs output by squareonly can be changed to rectangles.

- squareonlyscore

  The squareonlyscore option performs SRAF generation in the same way as squareonly, but performs cleanup in the same way as square. Like squareonly, if localmaxgradient keyword squarewidth is defined, squareonlyscore generates square SRAFs at seed pixels inside target shapes near corners with an area of $squarewidth^2$. Like SRAFs generated by the argument squareonly, no SRAFs generated by squareonlyscore can be changed to rectangles. Cleanup of SRAFs generated by squareonlyscore is score-based, the same as square.

- vertical

  Generates rectangular SRAFs inside target shapes and aligned to vertical target edge axes.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

Regarding algorithm interactions:

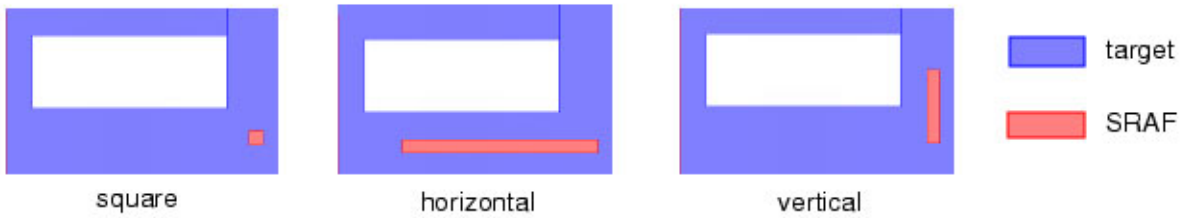
- If algorithm 0 is set, square is the default cornerprefnegative setting.
- If algorithm 1 is set, no is the default cornerprefnegative setting.

Description

An optional keyword that specifies the negative SRAF generation method at target shape corners. The cornerprefnegative keyword is applicable only to the negative keyword and may be specified globally, by priority, or within the localmaxgradient template.

Examples

Figure 4-99. cornerprefnegative SRAFs From the Same Seed Pixel



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

gradientfactornegative

Style Keywords

Specifies a gradient factor value for negative SRAFs. Used by RET MBSRAF only.

Usage

gradientfactornegative {-1 | 1}

Arguments

- -1 | 1

  Either negative or positive 1.

Description

An optional keyword that specifies the placement of seeds on valleys or peaks for negative SRAFs. Setting the value to 1 places the SRAF seeds on gradient peaks and setting it to -1 places the SRAF seeds on gradient valleys. Siemens EDA recommends setting gradientfactornegative to 1 only for DDL simulations.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

gradientshift

Style Keywords

Specifies a gradient shift value. Used by RET MBSRAF only.

Usage

gradientshift *value* [autobright]

Arguments

- *value*

  A positive or negative floating point number.

- autobright

  An optional keyword to invoke gradientshift for bright field masks or negative tone resists for obtaining gradientshift values during SRAF recipe development.

Description

An optional keyword that specifies the amount to shift the gradient during SRAF generation. Siemens EDA recommends using gradientshift only to identify the necessary shift value and the explicitly specify this value in subsequent batch runs.

The autobright option calculates the shift value needed to move the gradient under all of the seed pixels to be positive in the first processed frame. It applies this shift in the current run and then prints the value in the transcript for the user to specify during subsequent runs. The autobright option is intended to assist with early stages of SRAF recipe development.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

image_options

LITHO FILE Statement and Commands and Style Keywords

A LITHO FILE command referencing the *Lithomodel* file and an sraf_options command referencing the litho_model name. Used by RET MBSRAF only.

Usage

Usage within the LITHO FILE statement:

image_options ***name*** '**{**'
  layer ***name*** **visible** [mask_layer *number*] ...
  litho_model ***litho_model_dir*** ...
  [layer *name* filter]
  '**}**'

Usage within the sraf_options statement:

image_options ***name***

Arguments

- ***name***

  Declares a required, unique user-defined name for the image_options statement. This name is also used the image_options statement from within the sraf_options statement or setlayer mbsraf.

- **layer** ***name*** **visible** [*mask_layer number*]

  Specifies a visible layer. At least one visible layer is required. The mask_layer parameter maps this layer to a corresponding mask_layer template inside the *Lithomodel* file. The mask_layer number defaults to 0 if not specified (in the case of single template litho models, for example). For multi-patterning masks, specify a mask_layer argument to additional layer commands.

- **litho_model** ***litho_model_dir***

  A required argument that specifies a directory name. This directory must contain optical and resist models, a *Lithomodel* file, and optionally contain an etch model. For more detail see "Lithomodel Format Usage for Calibre mbSRAF Execution" on page 341.

  This is an abbreviated description of the image_options command. For detailed information, consult the *Calibre OPCverify User's and Reference Manual*.

- layer *name* filter

  An optional filter layer limiting contour generation to the area of the layer. Only one filter layer may be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Description

Specifies a statement name containing a reference to the *Lithomodel* file and subsequently called by the setlayer mbsraf command. The image_options statement within sraf_options must be specified prior to any pw_condition keyword.

Examples

Example 1

The keyword is first defined within the LITHO FILE statement and then called within the sraf_options statement:

```
LITHO FILE M1_model [
   image_options MBS {
      litho_model 20nm_lithomodel
      layer target visible
   }
   ...
   sraf_options mb_OPT {
      algorithm 1
      correctionlayer correction_layer
      image_options MBS
      ...
      localmaxgradient width 0.026 prior 1
      pw_condition f40 focus -40nm dose 1.05 size -0.00025 weight 40
   }
   setlayer mbsb1 = mbsraf target OPTIONS mb_OPT
]
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

intersection

Style Keywords

Enables valid SRAFs to be merged at various junctions. Used by RET SBAR and RET MBSRAF.

Usage

intersection {45 | C | J | L | N | P | T}
[maxjogwidth *value*]
[minjogwidth *value*]
[minstickout *value*]
[minwidthfree *value*]
[minwidthoverlap *value*]
[widthc *value*]

Arguments

- 45

  Enables valid, generated SRAFs to be merged at 45-degree junctions. Only 45-degree SRAFs of the same widths are merged. A 45-degree merge requires axis-aligned SRAFs; the convex corner is the common point, and the intersection of the two SRAFs defines the concave corner.

  The intersection 45 keyword only supports applications for the rectedge extension45 keyword. The intersection 45 keyword is incompatible with cleanupversion 0 and requires specification of cleanupversion 1.

  Note

  Siemens EDA does not recommend using intersection 45 where combined with intersection J and lineendmerge. This may allow 45-degree SRAFs that are not aligned to merge with a jog defined by minjogwidth. Although Siemens EDA does not recommend this combination, if intersection 45 is used with lineendmerge, all SRAFs to be merged must be the same width and must be specified with cleanupversion 1.

- C

  Enables valid, generated SRAFs to be merged at 90-degrees with an intermediate 45-degree junction. The C option can be used with either positive or negative SRAFs. The C option may be specified with the widthc option, providing a target width for the C-segment. Where the C and L options are specified together, the C option is given priority. If application of the C option fails, an attempt is made to apply the L option. Application of the C option fails if the orthogonal SRAFs to be joined are too close together, resulting in corners of the 45-degree SRAF extending past the orthogonal SRAF edges. Intersection C is incompatible with cleanupversion 0.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- J

  Enables valid, generated SRAFs to be merged at jog point intersections. The J option can be used with either positive or negative SRAFs. When using intersection J with lineendmerge, jogs are merged only when SRAF widths match. This is the default behavior with cleanupversion 1.

  Note

  Siemens EDA recommends using intersection J with the lineendmerge keyword set to a value greater than minoffset. This provides more continuous SRAF results. If no lineendmerge parameter is set, its value is automatically set. This value is reported to the transcript during processing. During Calibre nmOPC and SRAF Print Avoidance (SPA) after SRAF generation, fragments near concave corners cannot be aligned and eliminated. SPA cannot resolve printing at or near jogs because of the inability to align fragments near concave corners.

- L

  Enables valid, generated SRAFs to be merged at 90-degree junctions, irrespective of orientation angle. The L option can be used with either positive or negative SRAFs. Where the C and L options are specified together, the C option is given priority. If application of the C option fails, an attempt is made to apply the L option. Intersection L is incompatible with cleanupversion 0.

- N

  Disables SRAF merging. The N option can be used with either positive or negative SRAF generation. The N option is default.

- P

  Enables valid, generated SRAFs to be merged at orthogonal, non-45-degree junctions (forming four-way intersections or multiple T-junctions). The P option can be used with either positive or negative SRAFs. Intersection P is incompatible with cleanupversion 0.

- T

  Enables valid, generated SRAFs to be merged where lineend and long edges of SRAFs meet (forming T-junctions). The T option can be used with either positive or negative SRAFs. Intersection T is incompatible with cleanupversion 0.

- maxjogwidth *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Sets the maximum overlap of a jog. It can be set globally or by priority. The maxjogwidth parameter does not apply when smaller than the overlap between line-ends. The default for maxjogwidth is the maximum of the width parameters for all templates at the applied priority.

- minjogwidth *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. This keyword overrides the default jog width (equal to the template-based width

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

keyword). The default for minjogwidth is minwidth (if specified) or the minimum width parameter of any template of any priority. This default ensures that no MRC violations can occur when SRAFs of different priorities are jogged.

- minstickout *value*

  An optional keyword and argument that defines the minimum length a leg can extend from an intersection formed by intersection L, T, or P. The minstickout keyword controls both existing overlapping SRAFs and subsequently generated intersections. Existing overlapping SRAFs not meeting the minimum value specified by minstickout are cleaned. The default for minstickout is minwidth (if specified) or the minimum width parameter of any template of any priority. The minstickout value must be specified as a positive, floating-point number in microns.

- minwidthfree *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Specifies the minimum non-overlapping portions of jog intersections. See Figure 4-102 on page 251 for more information.

- minwidthoverlap *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Specifies the minimum overlap of line widths forming jog intersections. See Figure 4-102 on page 251 for more information.

- widthc *value*

  An optional keyword and argument, applied only to the C option, that specifies the width of C-segment SRAFs joining two orthogonal SRAFs. The larger orthogonal segment width must be less than or equal to 2 times the smaller orthogonal width for the C-segment to be correctly placed. By default, the width of the inserted segment is equal to the larger orthogonal SRAF segment. The widthc parameter can be used to specify a different width within a valid range of two orthogonal SRAF widths:

  *smaller_width * sqrt(2) <= widthc <= larger_width / sqrt(2)*

  If the specified widthc value is greater than the range, the maximum value is used; if the specified value is less than the valid range, the minimum value is used. See Figure 4-109 for more information. The parameter must be specified as a positive, floating-point number in microns.

Description

An optional keyword that enables valid, generated SRAFs to be joined at various specified intersections. The intersections allowed are represented by characters approximating their final shape: 45 (for diagonal SRAFs), C (for 45s joined with L SRAFs), J (for jogs), L, T, and P (for plus, an orthogonal cross at any orientation). The intersection keyword can be used with either positive or negative SRAFs. Optionally, the intersectionnegative keyword can be used globally to control negative SRAF intersections.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Formation of Intersection Candidates

The search range to locate candidates for the formation of intersections is defined by the parameter lineendmerge. If lineendmerge is not specified, the maximum of one of the minoffset parameter lineend, or the intersection parameters minjogwidth, or maxjogwidth, plus an additional 15% of the maximum value, defines the search distance. All SRAFs within the search distance are considered for the formation of a legal intersection by constructing one or two extensions of the existing SRAFs. For intersection J, three attempts are made: one symmetric attempt by extending both SRAFs, and two asymmetric attempts by extending either one of the SRAFs. Attempts to construct intersections between SRAFs fail if the resulting SRAF candidate conflicts with any existing SRAFs within the search distance.

When using intersection J with lineendmerge, jogs are merged only when SRAF widths match. This is the default behavior with cleanupversion 1.

Note

Regarding intersection usage:

- Siemens EDA recommends using intersection J with the lineendmerge keyword set to a value greater than minoffset. This provides more continuous SRAF results. If no lineendmerge parameter is set, its value is automatically set for you. This value is reported to the transcript during processing. During Calibre nmOPC and SRAF Print Avoidance (SPA) after SRAF generation, fragments near concave corners cannot be aligned and eliminated. SPA cannot resolve printing at or near jogs because of the inability to align fragments near concave corners.
- The SRAF merging order for multi-option intersection keywords is always: J, L, T, and then P. The specified order of the intersection keyword option letters has no effect on SRAF merging. In all other cases where one option is specified, only merging consistent with that option can be performed.

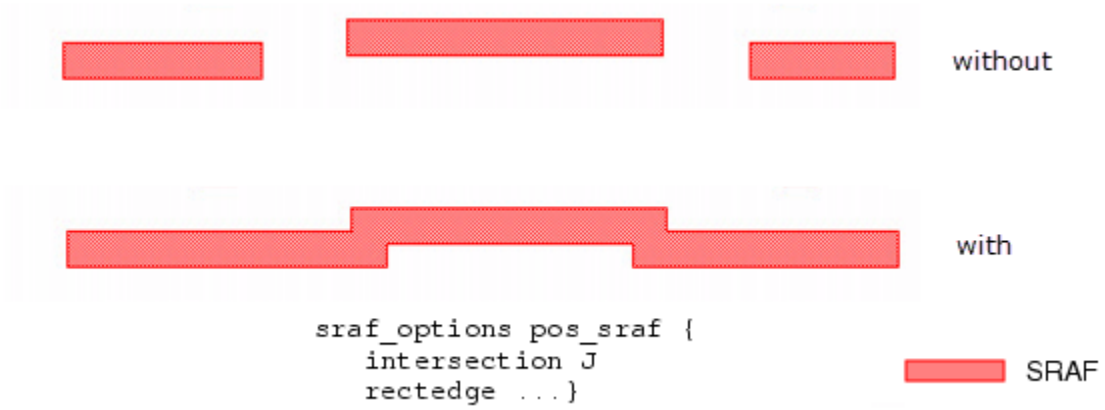
Examples

Example 1 — J Option

The first and second pictures show SRAFs generated without and with intersection J, respectively.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

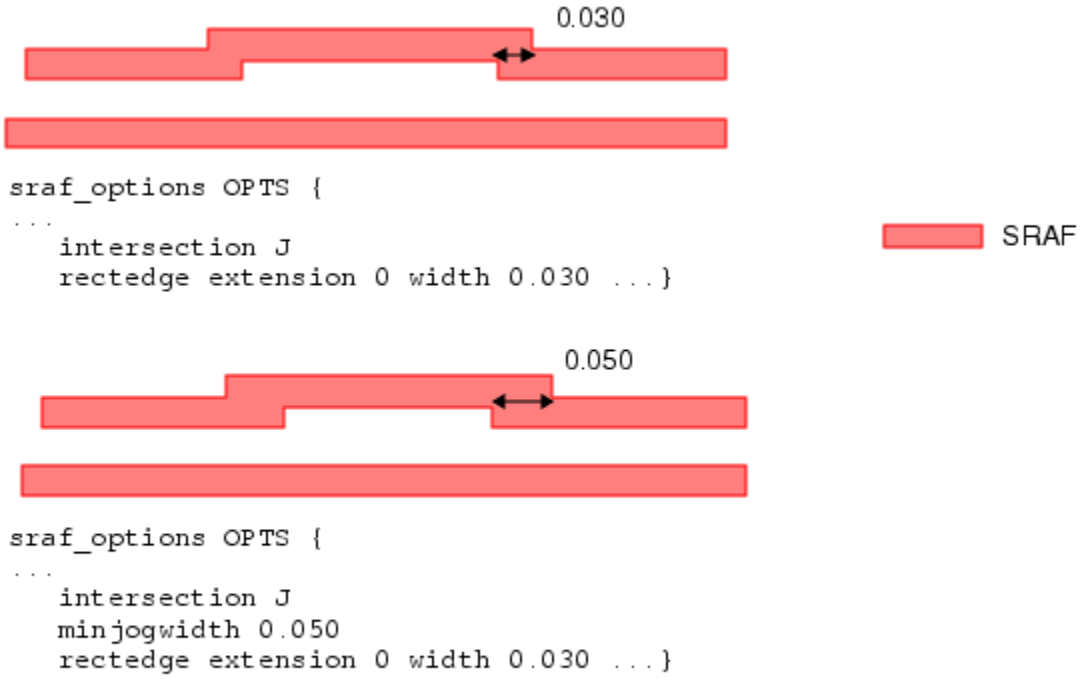
Figure 4-100. Default Usage of intersection J



Example 2 — J Option

The first picture shows SRAFs generated with intersection J using the default jog width. The second picture shows SRAFs generated with intersection J and the default jog width overridden by minjogwidth.

Figure 4-101. minjogwidth Modifies intersection J Output

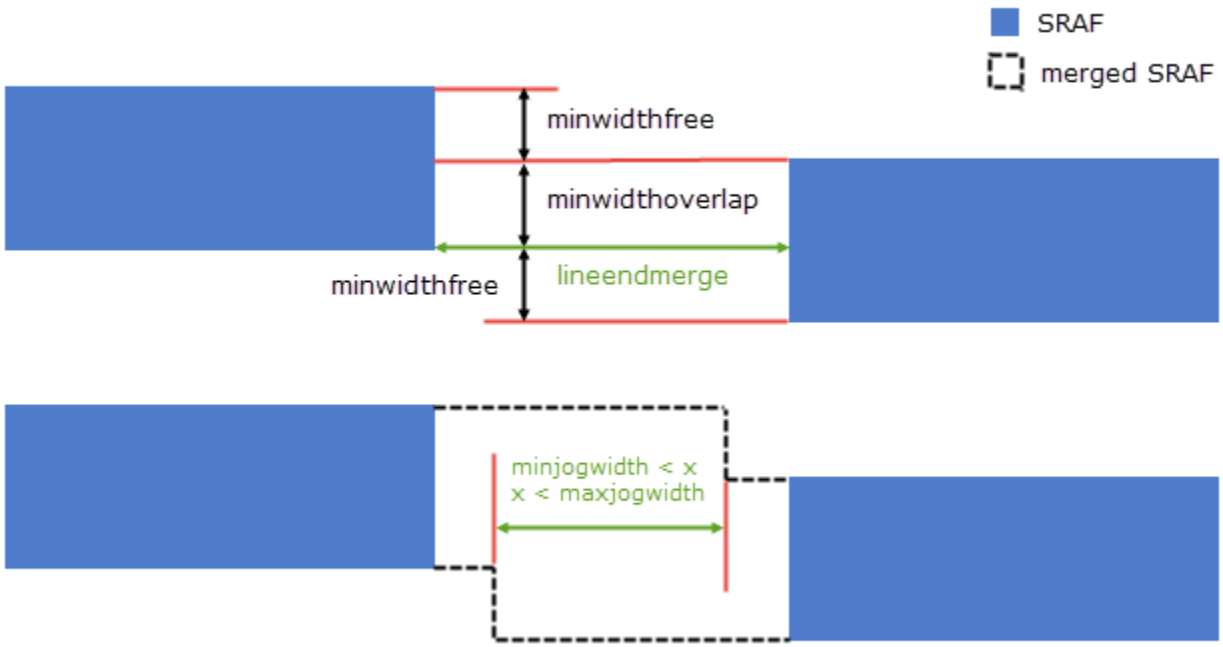


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 3 — J Option

The first and second pictures show SRAFs generated without and with intersection J, respectively.

Figure 4-102. Usage of minwidthoverlap and minwidthfree Parameters

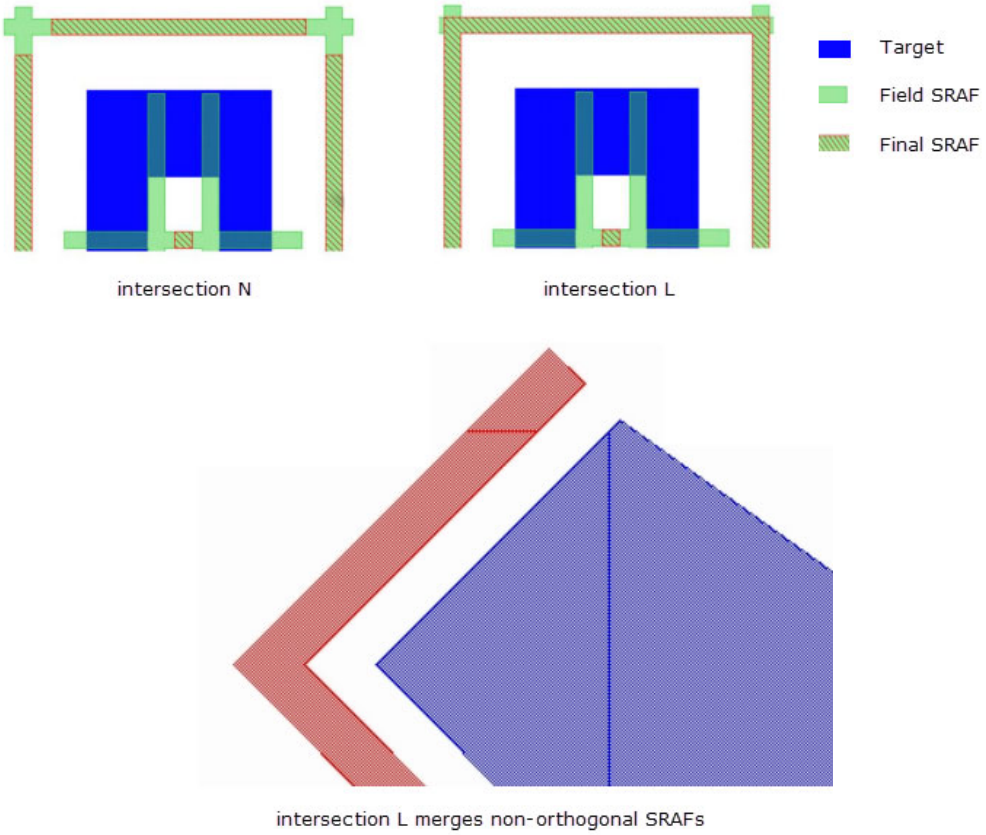


Example 4 — L Option

The left and right pictures show SRAFs generated without and with intersection L, respectively. Intersection L can be applied in various orientations and proximities.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

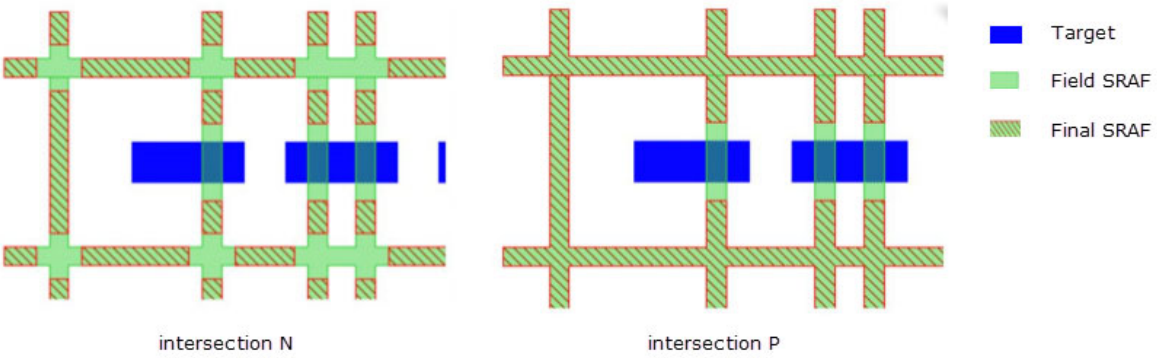
Figure 4-103. Intersection L



Example 5 — P Option

The left and right pictures show SRAFs generated without and with intersection P, respectively. Intersection P can be applied with various results depending on topology.

Figure 4-104. Intersection P

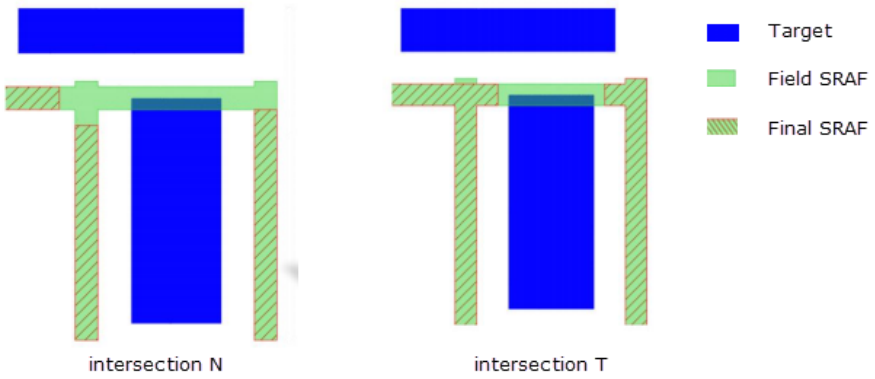


Example 6 — T Option

The left and right pictures show SRAFs generated without and with intersection T, respectively. Intersection T can be applied with various results depending on topology.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-105. Intersection T



Example 7 — 45 Option

The left and right pictures show SRAFs generated without and with intersection 45, respectively.

Figure 4-106. Intersection 45

Example 8 — C Option

Figure 4-107 shows SRAFs generated with intersection C.

Figure 4-107. Intersection C

Figure 4-108 shows the individual SRAF segments and the C-segment joining the two orthogonal SRAF segments.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-108. C-Segment SRAF

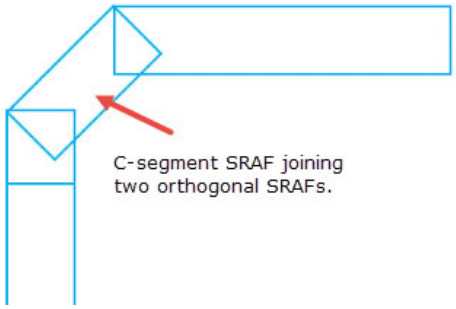


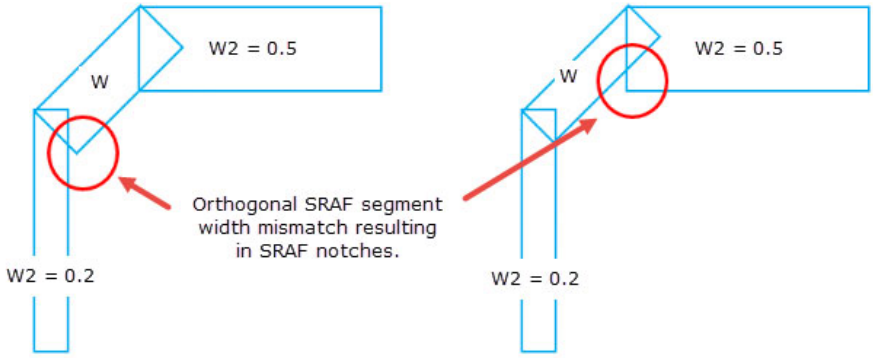
Figure 4-109 shows a failure of a C-segment to join two orthogonal SRAF segments, forming notches. The larger orthogonal segment width must be less than or equal to 2 times the smaller orthogonal width for successful application of the C-segment. In the illustration, where W1 and W2 are the widths of the wider and narrower SRAFs respectively, a C-segment can be formed only if:

*W1 <= 2 * W2*

Within this range, the possible C-segment width within a minimum to maximum range, if widthc is specified, is W:

*W2 * sqrt(2) <= W <= W1 / sqrt(2)*

Figure 4-109. C-Segment Width Mismatch



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

intersectionnegative

Style Keywords

Enables valid negative SRAFs to be merged at various junctions. Used by RET SBAR and RET MBSRAF.

Usage

intersectionnegative {J | L | N | P | T} [minjogwidth *value*]

Arguments

- J

  Enables valid, generated SRAFs to be merged at jog point intersections. The J option can be used with either positive or negative SRAFs. When using intersectionnegative J with lineendmerge, jogs are merged only when SRAF widths match. This is the default behavior with cleanupversion 1.

  Note

  Siemens EDA recommends using intersectionnegative J with the lineendmerge keyword set to a value greater than minoffset. This provides more continuous SRAF results. If no lineendmerge parameter is set, its value is automatically set. This value is reported to the transcript during processing. During Calibre nmOPC and SRAF Print Avoidance (SPA) after SRAF generation, fragments near concave corners cannot be aligned and eliminated. SPA cannot resolve printing at or near jogs because of the inability to align fragments near concave corners.

- L

  Enables valid, generated SRAFs to be merged at 90-degree junctions, irrespective of orientation angle. The L option can be used with either positive or negative SRAFs. intersectionnegative L is incompatible with cleanupversion 0.

- N

  Disables SRAF merging. The N option can be used with either positive or negative SRAF generation. The N option is default.

- P

  Enables valid, generated SRAFs to be merged at orthogonal, non-45-degree junctions (forming four-way intersections or multiple T-junctions). The P option can be used with either positive or negative SRAFs. intersectionnegative P is incompatible with cleanupversion 0.

- T

  Enables valid, generated SRAFs to be merged where lineend and long edges of SRAFs meet (forming T-junctions). The T option can be used with either positive or negative SRAFs. intersectionnegative T is incompatible with cleanupversion 0.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- maxjogwidth *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Sets the maximum overlap of a jog. It can be set globally or by priority. The maxjogwidth parameter does not apply when smaller than the overlap between line-ends. The default for maxjogwidth is the maximum of the width parameters for all templates at the applied priority.

- minjogwidth *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. This keyword overrides the default jog width (equal to the template-based width keyword). The default for minjogwidth is minwidth (if specified) or the minimum width parameter of any template of any priority. This default ensures that no MRC violations can occur when SRAFs of different priorities are jogged.

- minstickout *value*

  An optional keyword and argument that defines the minimum length a leg can extend from an intersection formed by intersectionnegative L, T, or P. The minstickout keyword controls both existing overlapping SRAFs and subsequently generated intersections. Existing overlapping SRAFs not meeting the minimum value specified by minstickout are cleaned. The default for minstickout is minwidth (if specified) or the minimum width parameter of any template of any priority. The minstickout value must be specified as a positive, floating-point number in microns.

- minwidthfree *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Specifies the minimum non-overlapping portions of jog intersections. See Figure 4-102 on page 251 for more information.

- minwidthoverlap *value*

  An optional keyword and argument specified as a positive, floating-point number in microns. Specifies the minimum overlap of line widths forming jog intersections. See Figure 4-102 on page 251 for more information.

Description

An optional keyword that enables valid, generated negative SRAFs to be joined at jog point intersections. The intersectionnegative keyword is used only for negative SRAF generation and should only be set globally. The default jog width (established by the template width keyword) can be overridden with minjogwidth. Although describing positive SRAF, see the intersection keyword for more examples of L, P, and T SRAF merging interaction.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

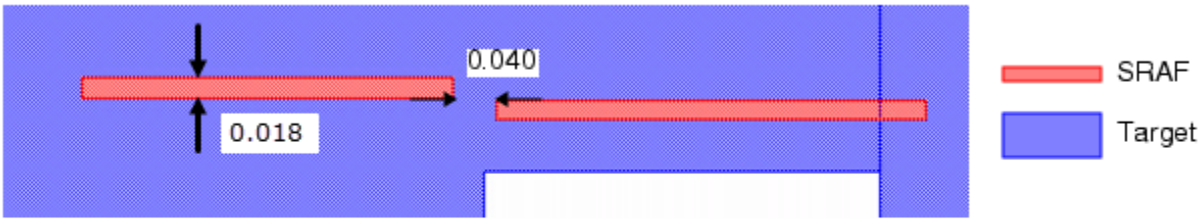
Siemens EDA recommends using intersection J by priority to control jogs with both positive and negative SRAFs within a single run. See the negative keyword for details regarding this procedure. During a Calibre nmOPC run with SRAF Print Avoidance (SPA) after SRAF generation, fragments near concave corners cannot be aligned and eliminated. SPA cannot resolve printing at or near jogs because of the inability to align fragments near concave corners.

Examples

Example 1 — With intersectionnegative N (or without intersectionnegative)

With intersectionnegative N, negative SRAFs are disabled and not merged at jog points. Their spacing defaults to minoffset.

Figure 4-110. Disabling Negative SRAF



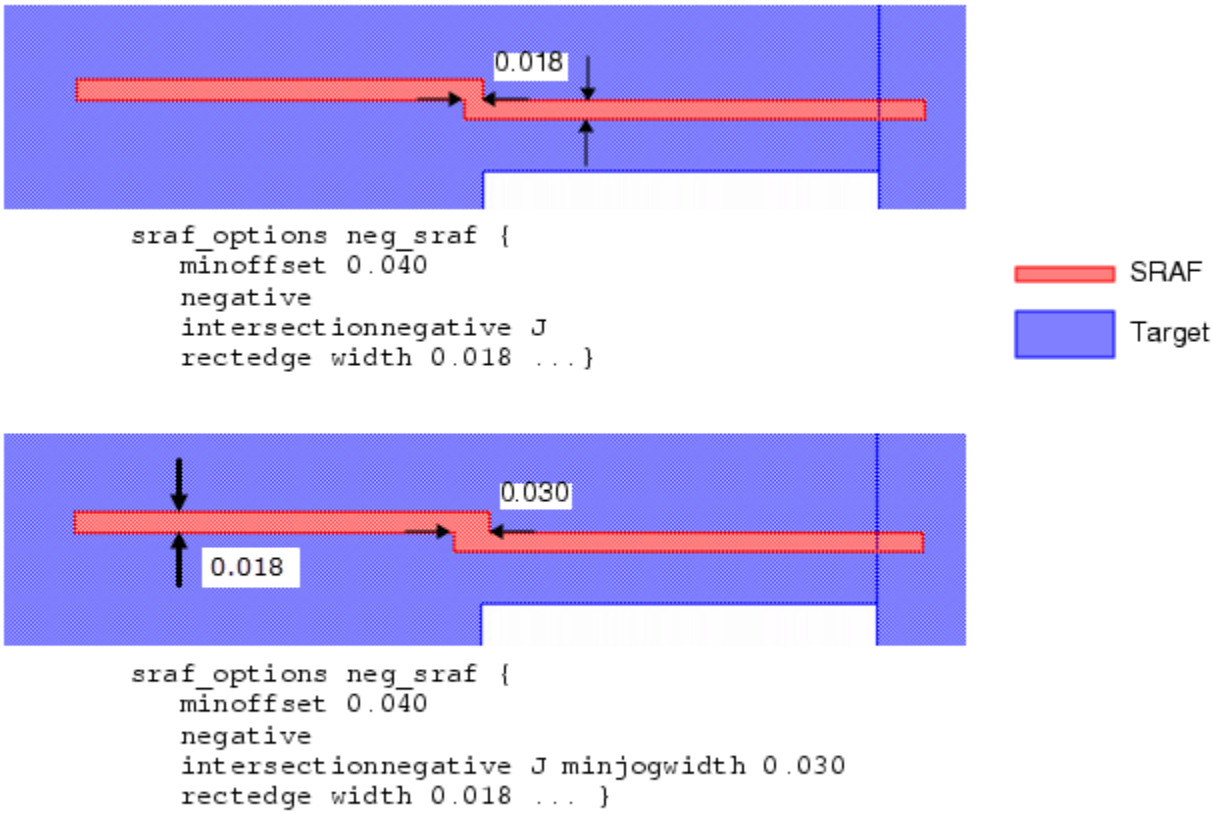

```
sraf_options neg_sraf {
   minoffset 0.040
   negative intersectionnegative N
   rectedge width 0.018 ...}
```

Example 2

With intersectionnegative J, negative SRAFs are merged at jog points and their jog width defaults to the template width. With minjogwidth added to intersectionnegative J, negative SRAFs are merged and their jog width increases to minjogwidth.

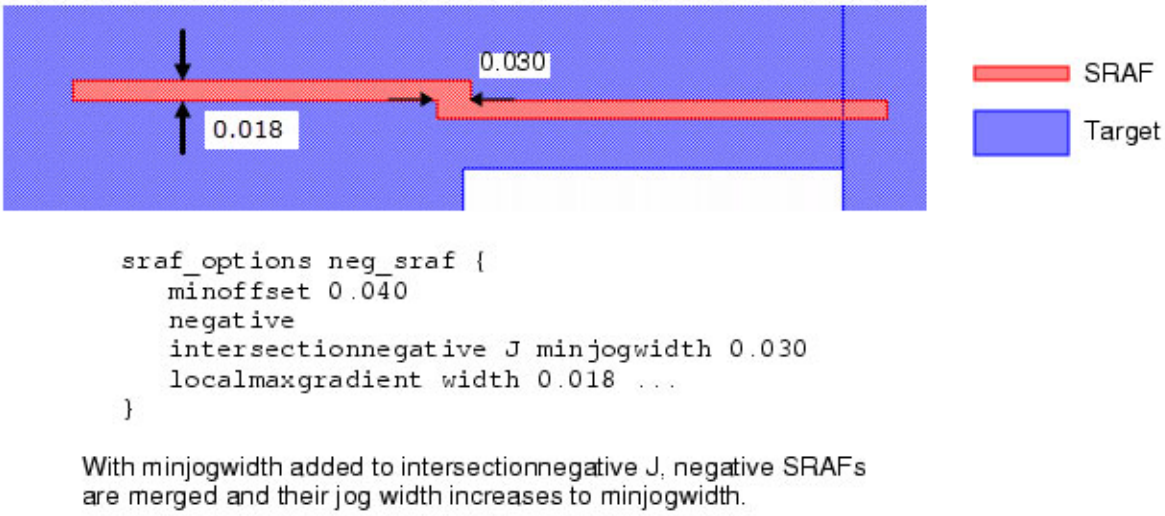
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-111. intersectionnegative J and minjogwidth Interaction



Example 3 — With intersectionnegative J and minjogwidth Specified

Figure 4-112. intersectionnegative J With minjogwidth



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

jogignore

Style Keywords

Overrides jogs in generated SRAFs. Used by RET MBSRAF only.

Usage

jogignore *value*

Arguments

- *value*

  An optional argument specified as a positive, floating-point number in microns.

Description

An optional keyword that enables mbsraf to override jogs in generated SRAFs that are less than the specified value. Because algorithm 1 typically generates long, unbroken SRAFs when jogs are present, it does not require jogignore.

Note

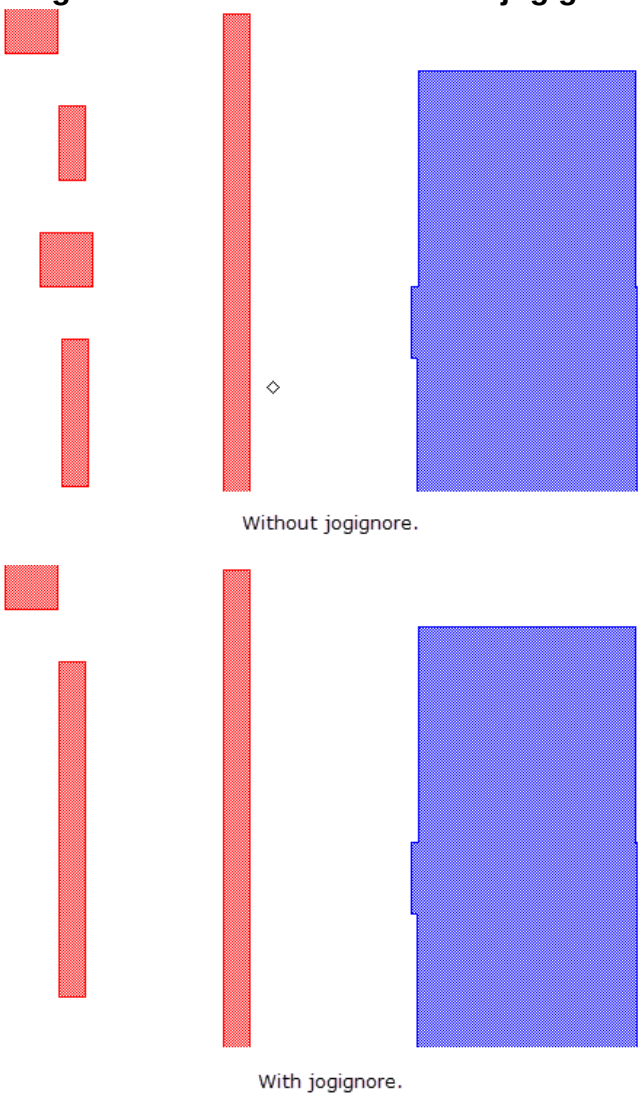
Siemens EDA recommends always using jogignore with mode collectno45 to achieve expected results.

Examples

The example illustrates a clip with and without jogignore.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-113. With and Without jogignore



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

mode

Style Keywords

Specifies a generated SRAF orientation. Used by RET MBSRAF only.

Usage

mode {**collectno45** | **nearest** | **nearestno45**}

Arguments

- **collectno45**

  This mode begins with seed pixels and follows the gradient to define the final SRAF. The width is then defined by mapping nearby target shapes, and length is established in an orthogonal direction to width but limited by the gradient. This mode does not generate 45 degree SRAFs.

- **nearest**

  For each local maximum, this mode obtains the nearest edge(s) and corner(s) of the target layer and then adjusts the length and orientation of the SRAF accordingly. SRAFs tilted 45 degrees are only constructed near tilted target edges or near corners.

- **nearestno45**

  For each local maximum, SRAF length and orientation is determined by the length and orientation of the nearest target edge or corner. This mode does not create SRAFs tilted 45 degrees. If algorithm 0 is set, nearestno45 is the default setting.

Description

An optional keyword that specifies which SRAF orientation is used for all templates. Only one mode must be set for an mbsraf run. The mode keyword can only be specified globally.

Note

Regarding algorithm interactions:

- If algorithm 0 is set, nearestno45 is the default mode setting.
- If algorithm 1 is set, mode is set internally and must not be set explicitly.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

negative

Style Keywords

Enables negative SRAF output. Used by RET SBAR and RET MBSRAF.

Usage

negative

Arguments

None.

Description

An optional keyword that specifies the generation of negative SRAFs, also referred to as slots in target shapes. The negative keyword cannot be used with pitch-based templates.

Examples

Example 1

Negative SRAFs are generated as holes in input target shapes.

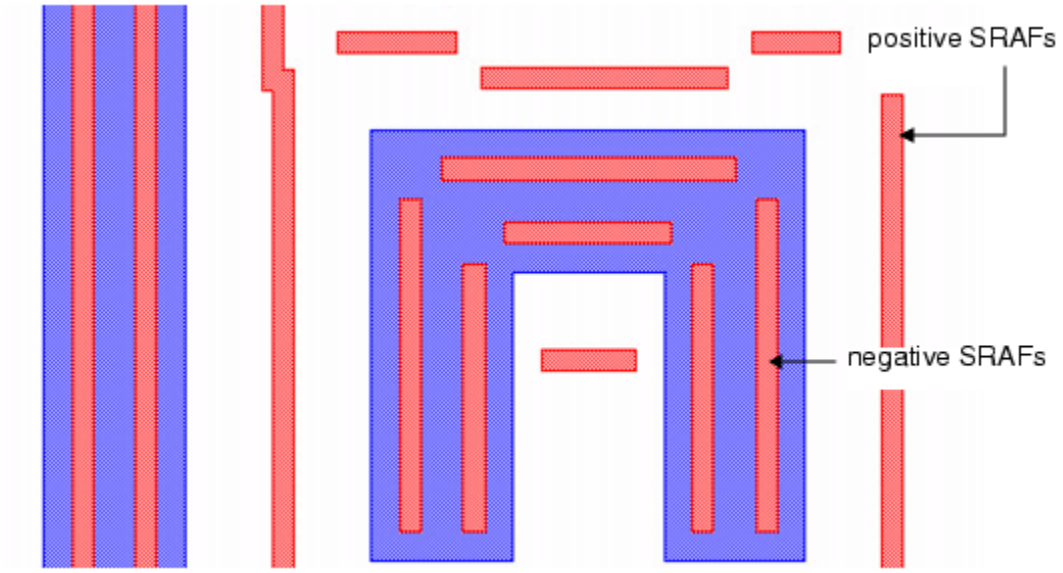
Figure 4-114. Negative SRAF

Example 2

Negative and positive SRAFs should be generated with different priorities for segregation during subsequent processing and debugging.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-115. Positive and Negative SRAFs



Example 3

Negative and positive SRAFs can be generated within the same LITHO FILE statement, using different priorities. The following code snippet positive SRAFs as priority 1 and negative SRAFs as priority 2.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-116. Negative and Positive SRAFs in a Common File

```
LITHO FILE cn_setup_1 [

   layer target

   sraf_options constraints {
      minoffset 0.020
      minfeaturespace 0.030
      maxconflict 0.150
   }

   sraf_options templates {
      rectedge extension 0.0 width 0.04 offset 0.04 prior 1
      rectedge extension -0.06 width 0.04 offset 0.03 prior 2
   }

   sraf_options priorities {
      priority 1
         lineendmerge 0.120
      priority 2
         negative
         lineendmerge 0.100
   }

   setlayer cnsb1 = cnsraf target \
      OPTIONS constraints OPTIONS templates OPTIONS priorities
]

cnsb1 = RET SBAR target FILE cn_setup_1 MAP cnsb1
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

peaktype

Style Keywords

Specifies the method of SRAF propagation. Used by RET MBSRAF only.

Usage

peaktype {1dir | 2dir | 3dir | 4dir}

Arguments

- 1dir

  Propagates SRAFs if the localmaxgradient is found in at least one direction from the seed pixel. The direction can be N-S, E-W, NE-SW, or NW-SE.

- 2dir

  Propagates SRAFs if the localmaxgradient is found in at least two directions from the seed pixel.

- 3dir

  Propagates SRAFs if the localmaxgradient is found in at least three directions from the seed pixel. If algorithm 0 is set, 3dir is the default setting.

- 4dir

  Propagates SRAFs if the localmaxgradient is found in all four directions from the seed pixel.

Note

Regarding algorithm interactions:

- If algorithm 0 is set, 3dir is the default peaktype setting.
- If algorithm 1 is set, peaktype is set internally and must not be set explicitly.

Description

An optional keyword that specifies the method with which SRAFs are propagated.

Qualification of Seed Pixels

The qualification of seed pixels is considered in four directions: vertical (available for display through debug layer seedv), horizontal (seedh), northwest-southeast (seednwse), and northeast-southwest (seednesw).

For arguments 1dir, 2dir, 3dir and 4dir, the four directions are used to determine whether the seed pixel under analysis contains a gradient value peak in one or more directions. If the pixel demonstrates the desired number of directions that locate peaks, that pixel is selected as a seed pixel (available for display through seedpixels debug layer).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

pw_condition

Style Keywords

Specifies a process window condition. Used by RET MBSRAF only.

Usage

pw_condition *name*
{[optical *name*]
[dose *value*]
[plane *value*]
[weight *value*]
[size *value*]
[defocus *value*]
}

Arguments

- *name*

  The name of the process window condition.

- optical *name*

  An optional keyword that specifies the name of the optical model for this process window condition.

- dose *value*

  An optional keyword that specifies the dose for the specified process window condition.

- plane *value*

  An optional keyword that specifies the value of the z-position within the resist film.

- weight *value*

  An optional keyword that specifies the weight for the EPE for the specified process window condition.

- size *value*

  An optional keyword that specifies the mask size, in microns, for this process window condition.

- defocus *value*

  An optional keyword that specifies the defocus for the specified process window condition.

Description

An optional keyword that specifies the name and various attribute arguments of a process window condition. At least one attribute argument must be specified.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

Siemens EDA recommends using the litho model format for all model-based SRAF recipes. These non-litho model arguments to pw_condition are not supported and are obsoleted as of release 2019.1:

- optical
- plane
- size

Note

You can specify multiple process window condition statements. All process window condition statements must be specified after the nominal optical model and any correctionlayer specifications.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

slopeweight

Style Keywords

Specifies the gradient slope weight. Used by RET MBSRAF only.

Usage

slopeweight 1 | {verticalweight *value* narrowweight *value* slopeweight *value*}

Arguments

- verticalweight *value*, narrowweight *value*, slopeweight *value*

  Alternative keywords specifying the vertical, narrow and slope weight values, respectively, of the gradient slope. Each value you specify must be a whole number between 0 and 1, inclusive.

Description

Optional keywords that specify the weight associated with each gradient type. The default usage is slopeweight 1.

If the alternative usage in which all weight keywords are specified, the values of slopeweight, narrowweight and verticalweight must be summed to equal 1. Setting narrowweight or verticalweight to 1 is discouraged. The summed values for verticalweight, narrowweight, and slopeweight must be set between 0 and 1, respectively, such that:

$$vweight + nweight + sweight = 1$$

If you use DDM models with HHA approximation for 3D mask modeling, mbSRAF does not use narrowweight. Only slopeweight and verticalweight are applied to the final gradient value. For more information regarding the pxopc gradient computing use of mbsraf slope values, see the Release Notes for Calibre Release 2015.4.

Note

Siemens EDA recommends relying on the default gradient slope weight, slopeweight 1, whenever possible.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

symmetric

Style Keywords

Ensures all templates generate SRAFs with symmetry. Used by RET SBAR only.

Usage

symmetric

Arguments

None.

Description

An optional keyword used to ensure all templates generate SRAFs symmetrically. If any template does not specify identical pitch, SRAF width and length values, or offsetx and offsety values, a parse error results.

Examples

These code portions show both non-identical offsetx and offsety, and width and length values. When the symmetric keyword is specified, and the recipe has asymmetrical SRAF output specified, the run fails with a parsing error. The keywords in red constitute the parse failures when symmetric is specified.

```
...
symmetric
rectedge
   width 0.050 length 0.53
   offsetx 0.2 offsety 0.1
```

ERROR: nmSRAF: Template is not symmetric: rectEdge template

```
...
symmetric
rectCorner tag bEdge
   offsetx 0.02 offsety 0.04 width 0.02 length 0.02
   prior 1
rectCorner tag vEdge center spacelimit 0.8
   length 0.05 width 0.07
   prior 2
```

ERROR: nmSRAF: Template is not symmetric: rectCorner template

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Conflict Keywords

Provides commands for managing conflicts formed during SRAF generation.

For organizational keyword information, refer to “sraf_options” on page 80.

Table 4-13. Conflict Keywords

Keyword	Description
centerthresh	Specifies the threshold of a conflict region. Used by RET SBAR only.
conflictmode	Specifies the means of conflict region processing. Used by RET SBAR only.
maxconflict	Specifies the maximum size of a conflict region. Used by RET SBAR only.
oversizesraf	Allows SRAFs within conflict regions to exceed the conflict region size. Used by RET SBAR only.
sbwidth	Specifies the width of secondary SRAFs. Used by RET SBAR only.
sraf	Enables and specifies the size of SRAF output in conflict regions. Used by RET SBAR only.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

centerthresh

Conflict Keywords

Specifies the threshold of a conflict region. Used by RET SBAR only.

Usage

centerthresh *threshold* [sbwidth *width*]

Arguments

- *threshold*

  A positive floating point number that specifies a threshold value for a conflict region.

- sbwidth *width*

  An optional keyword and positive floating point number that specifies the width of secondary scattering bars invoked by centerthresh. The minimum value you can specify is 2 database units. If sbwidth is not specified, no width-based secondary SRAFs are generated. The sbwidth keyword can be set by priority alone for all cnsraf templates.

Description

An optional keyword that specifies a threshold for the length and width of a conflict region determined by setlayer cnsraf. The centerthresh keyword is used to consolidate SRAFs initially generated that violate MRC constraints.

If centerthresh is not specified, one square SRAF is placed in the center of the conflict region. For more detail regarding the size of the SRAF placed in the conflict region, see the sraf and oversizesraf keywords.

The centerthresh keyword can be set globally and by priority for all cnsraf templates.

Results of processing the conflict region depend on the value of centerthresh compared to the conflict region extents as shown in the table (where maxconflict is greater than or equal to the conflict region):

Table 4-14. Conflict Region Processing

Conflict Region Extent Evaluation	Conflict Region Resolution
Both extents are less than centerthresh	A square SRAF is placed in the center of the conflict region; see Figure 4-117.
Only one extent is less than or equal to centerthresh	A rectangular SRAF is centered along the longer extent exceeding centerthresh; see Figure 4-118.
Neither extent is less than centerthresh	The conflict region is removed; see Figure 4-119.

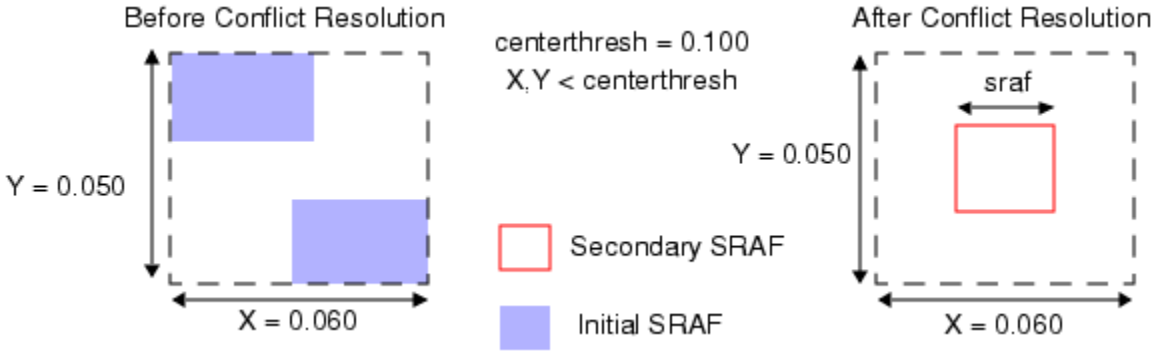
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1

A conflict region is shown where X and Y extents are less than centerthresh, so a square SRAF is placed at the center of the conflict region. The size of the square SRAF is specified by the sraf keyword.

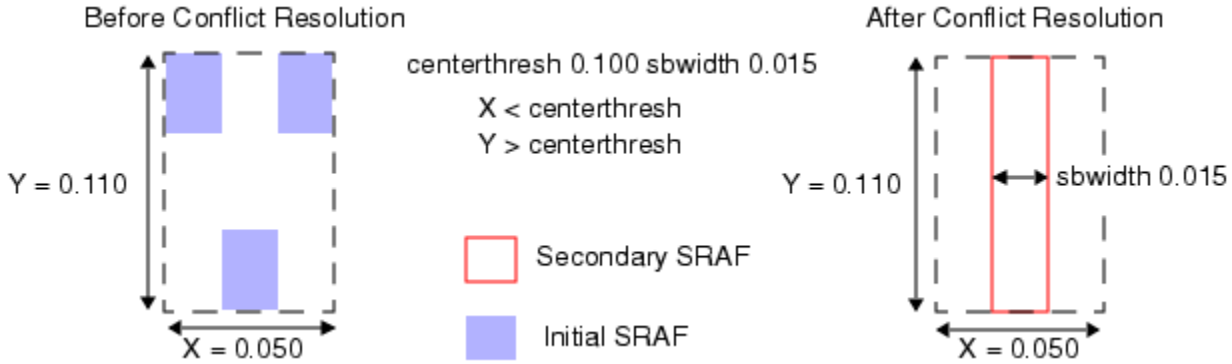
Figure 4-117. SRAF Placed in the Center of the Conflict Region



Example 2

A conflict region is shown where only the X extent is less than centerthresh. The initial SRAFs forming the conflict region are resolved to a single SRAF placed along the longest length of the conflict region. The width of the SRAF is set by the sbwidth keyword.

Figure 4-118. SRAF Placed Along the Length of the Conflict Region

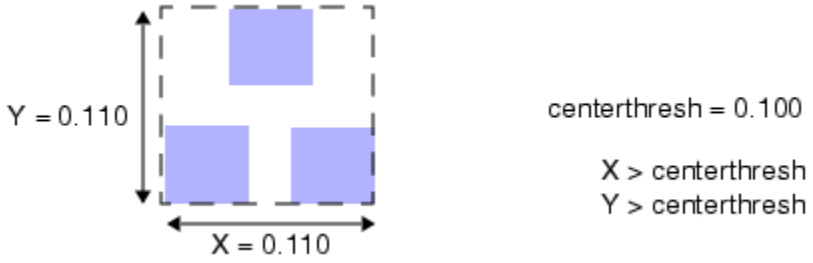


Example 3

A conflict region is shown where both X and Y extents are greater than centerthresh. The conflict region is removed and initial SRAFs are discarded.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-119. Conflict Region Removed



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

conflictmode

Conflict Keywords

Specifies the means of conflict region processing. Used by RET SBAR only.

Usage

conflictmode [bbox | merge | nomerge | shift]

Arguments

- bbox

  Generates a bounding box around conflict regions. For each pair of SRAFs in conflict, a bounding box is generated. These interacting bounding boxes are merged into polygons. The bounding box of the resulting polygons is used as the conflict region to be processed as specified by the maxconflict and centerthresh rules. These resulting conflict regions are always rectangular. The conflictbbox debug layer helps you visualize this bounding box.

  This mode is only supported in cleanupversion 1. Use of this mode in cleanupversion 0 results in warning messages.

- merge

  Merges all rectangular conflict regions only. For each pair of SRAFs in conflict, a bounding box is generated. These interacting bounding boxes are merged into polygons. Merging may result in the generation of non-rectangular polygons. Resulting polygons are checked with the maxconflict and centerthresh controls and processed. Non-rectangular polygons and rectangles with at least one side exceeding maxconflict are discarded. The hardconflict debug layer helps you visualize these polygons.

  This is the default mode for cleanupversion 0 and 1.

- nomerge

  Does not merge the conflict regions, but processes each one individually. Inserted SRAFs are checked for MRC violations.

- shift

  Does not merge the conflict regions, but processes each one individually. This method attempts to shift common SRAFs (including 45-degree SRAFs) to the center of the conflict region. If the orientations are not matched, the conflict region is considered for regular conflict resolution.

Tip

Siemens EDA recommends using conflictmode bbox for best results.

Description

An optional keyword that specifies the method by which conflict regions are constructed. This keyword can be set globally for all cnsraf templates and it can be assigned on a priority basis using the priority keyword. Setting conflictmode on a priority basis overrides the global

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

assignment. These options outputs conflict regions that can be displayed by various debug keywords. See "setlayer cnsraf debug" on page 69 for more information.

Examples

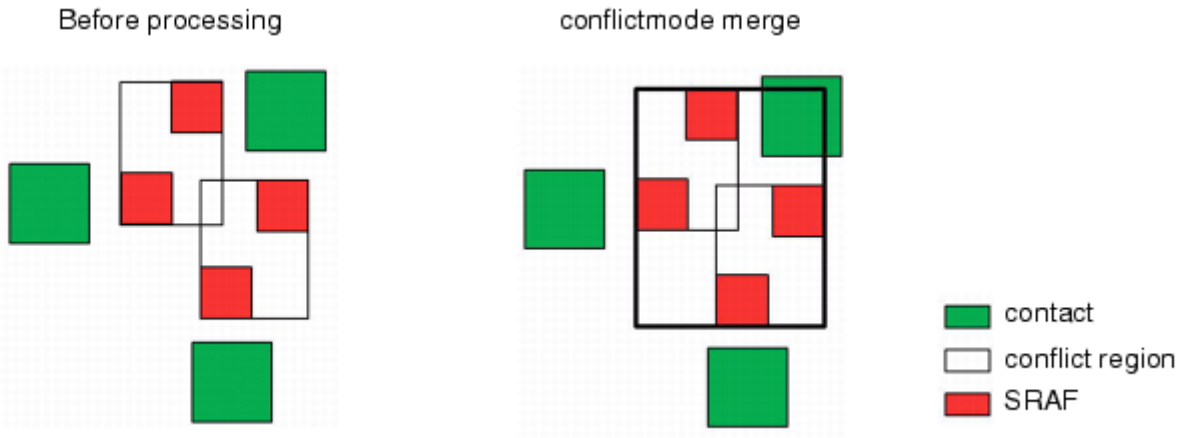
Example 1

The merge option merges all rectangular conflicts into one conflict region. Because the conflict region is MRC clean, it is processed, and a new SRAF is added. The nomerge option processes each conflict region individually. Any new SRAFs violating MRC are removed.

Example 1

Figure 4-120 shows a case where the merged conflict region is not MRC clean and, therefore, is not processed.

Figure 4-120. conflictmode Violates MRC

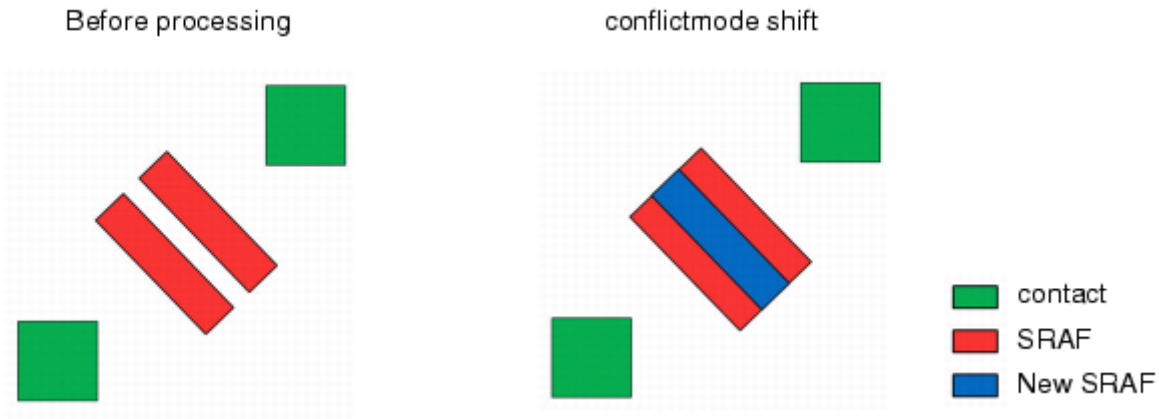


Example 2

The shift option processes each conflict region individually and shifts common SRAFs to the center of the conflict region. If the SRAFs are not aligned, then shift is not applied, and nomerge is used instead.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-121. conflictmode shift

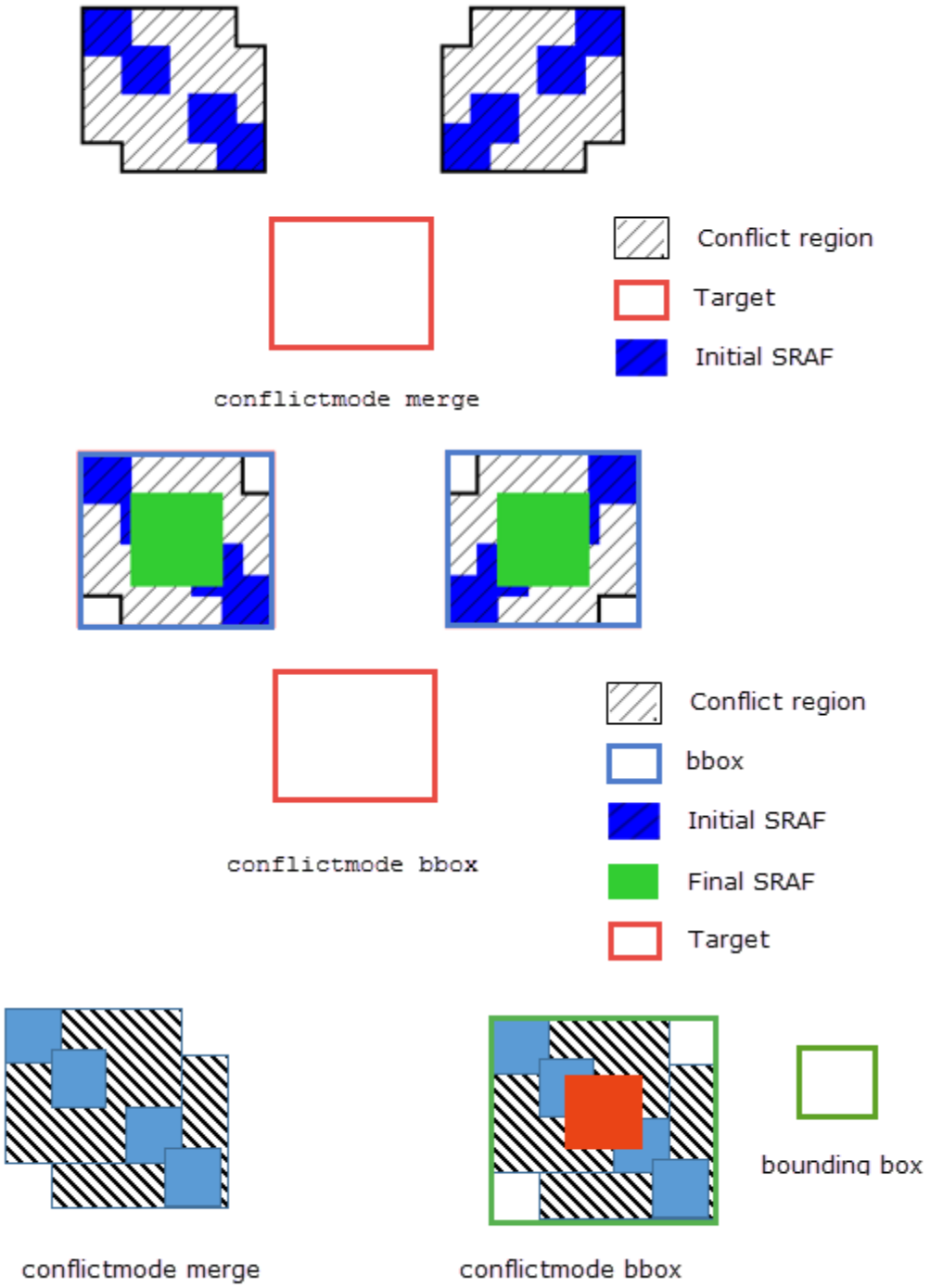


Example 3

The bbox option merges each conflict region that may overlap and applies a bounding box around the periphery of the merged regions. This bounding box permits a final SRAF to be saved, because the bounding box is always rectangular, whereas the merged conflict regions output by merge mode may not afford the required space. Any resulting non-rectangular conflict regions are discarded with their field SRAFs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-122. Differences between conflictmode bbox and conflictmode merge



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

maxconflict

Style Keywords

Specifies the maximum size of a conflict region. Used by RET SBAR only.

Usage

maxconflict *value*

Arguments

- ***value***

  A required positive floating point number that specifies the maximum size of a conflict region to process. Siemens EDA recommends setting this value less than 0.5 um. If a larger value is needed, create an additional template instead.

Description

A required keyword that sets the limit on the size of a conflict region to process. The maxconflict keyword is critical for promotion during hierarchical processing. MRC conflicts typically result with rule-based SRAF generation in complex topologies. For this reason, maxconflict requires a value for SRAF placement, likely targeting a significant portion of your design.

Conflict regions larger than maxconflict are eliminated. If maxconflict is set to 0, no conflict regions are calculated, and no secondary SRAFs are generated. Setting maxconflict greater than 0 keeps qualifying regions and processes them according to the rules specified by centerthresh. If centerthresh is not specified, one square SRAF is placed in the center of the conflict region. For additional details regarding the size of the SRAF placed in the conflict region, see the sraf and oversizesraf keywords.

This keyword can be set globally and by priority for all cnsraf templates. If no global or priority specification of maxconflict is provided, a compile-time error is issued.

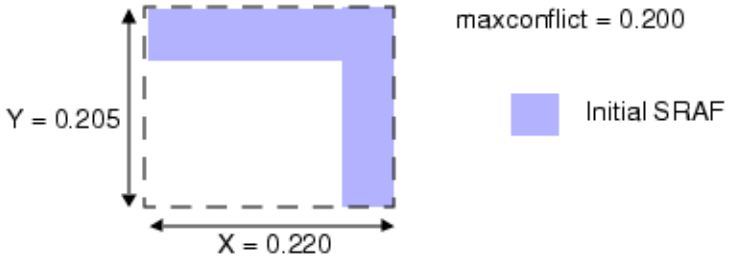
Examples

Example 1

The dimensions of the conflict region (0.220, 0.205) are greater than maxconflict (0.200) so the conflict region is removed and all secondary SRAF discarded.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

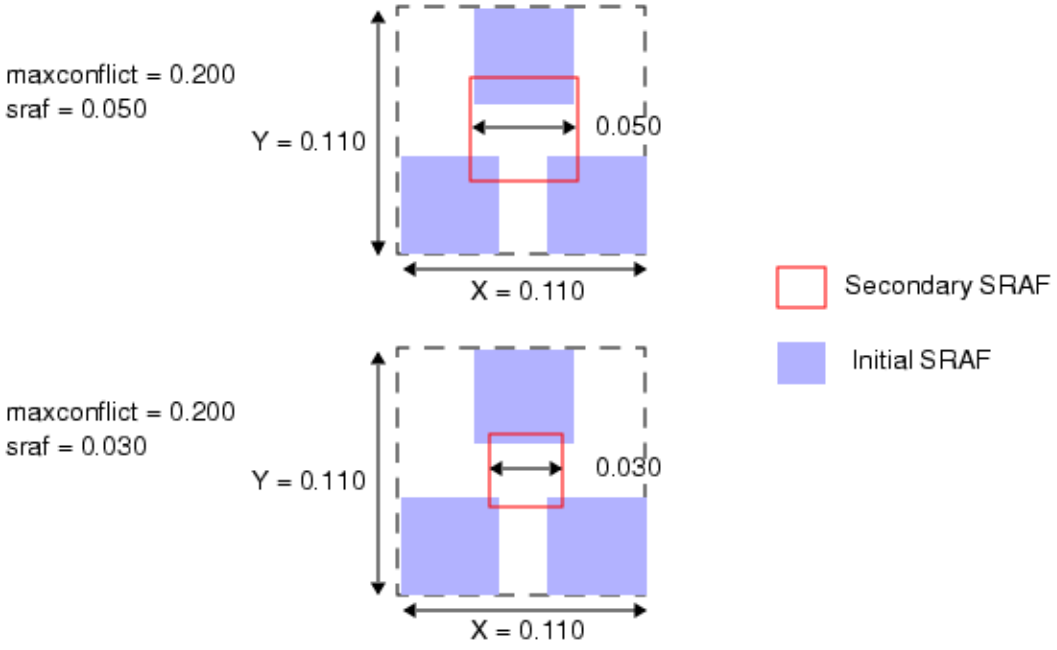
Figure 4-123. Conflict Region Discarded by maxconflict



Example 2

The X,Y dimensions of the conflict region (0.110, 0.110) are less than maxconflict (0.200) so the conflict region is processed according to the rules specified by centerthresh. If centerthresh is not specified, a secondary square SRAF, as specified by the cnsraf template, is placed in the center of the conflict region. If the sraf keyword is specified, a secondary square SRAF is generated with the sraf value you specify, not exceeding the boundary of the conflict region. To override the conflict region extent during secondary SRAF generation, use oversizesraf.

Figure 4-124. Valid Conflict Region



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

oversizesraf

Conflict Keywords

Allows SRAFs within conflict regions to exceed the conflict region size. Used by RET SBAR only.

Usage

oversizesraf {no | yes}

Arguments

- yes

  Enables SRAFs to exceed the conflict region size.

- no

  Prohibits SRAFs from exceeding the conflict region size. When using oversizesraf no, Siemens EDA recommends you specify the minimum SRAF size with minwidth to prevent output of arbitrarily small SRAFs. This is the default value.

Description

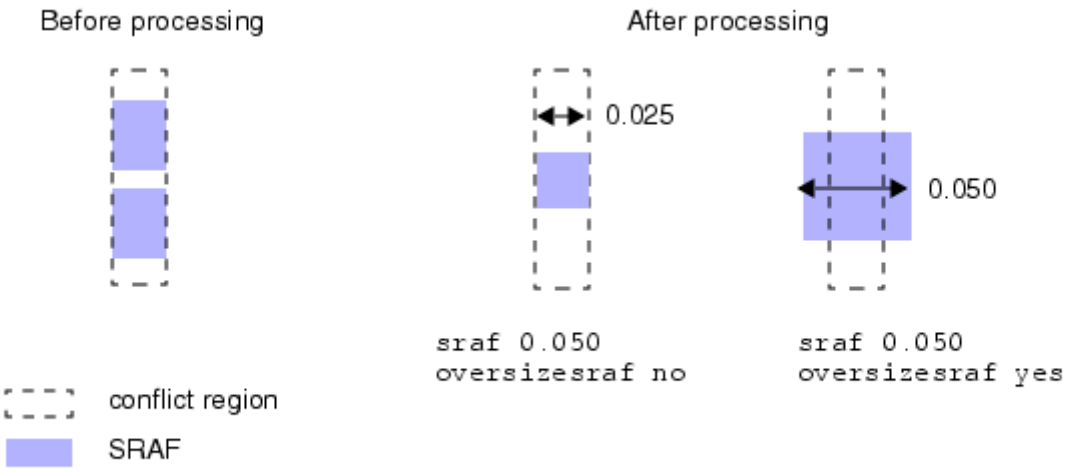
An optional keyword, typically used as an option with the sraf keyword, that allows SRAFs in conflict regions to be larger in size than the conflict region itself. By default, SRAFs are cropped to the size of the conflict region. The oversizesraf keyword can be set globally and by priority for all cnsraf templates.

Examples

Figure 4-125 shows an example where an SRAF is larger than the conflict region. When oversizesraf no is specified as shown in the center, the square SRAF is generated no larger than the value specified by the sraf keyword to fit inside the conflict region. The SRAF is generated, and maintains its square shape. When oversizesraf yes is specified, shown at right, the original square SRAF maintains its original size.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-125. oversizesraf Usage



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

sbwidth

Conflict Keywords

Specifies the width of secondary SRAFs. Used by RET SBAR only.

Usage

sbwidth *width*

Arguments

- *width*

  An optional positive floating point number. The minimum value you can specify is 2 database units.

Description

An optional keyword that specifies the width of secondary scattering bars as invoked by centerthresh. If sbwidth is not specified, no width-based secondary SRAFs are generated. See centerthresh for more information regarding its use with sbwidth. This keyword can be set globally or by priority for all cnsraf templates.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

sraf

Conflict Keywords

Enables and specifies the size of SRAF output in conflict regions. Used by RET SBAR only.

Usage

sraf *value*

Arguments

- *value*

  A positive floating point number. The minimum value is two database units.

Description

An optional keyword that enables secondary SRAF output in conflict regions. The sraf keyword also specifies the square size of the SRAFs added to the conflict region if the length of the region edges is less than or equal to maxconflict and less than centerthresh.

The SRAF is output at the size specified by the sraf keyword up to a maximum of the conflict region extent. If the sraf keyword is not specified, no SRAFs are generated in the conflict region. The minimum sraf keyword value specified must be two database units (DBU).

By default, during data promotion where conflict regions are split between tiles, conflict region-placed SRAFs may be generated less than the sraf value when trimmed to a smaller conflict region boundary. Set oversizesraf to yes to override this behavior.

The sraf keyword can be set globally or by priority for any setlayer cnsraf template.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Cleanup Keywords

Calibre nmSRAF invokes SRAF cleanup after generation.

For organizational keyword information, refer to "sraf_options" on page 80.

Table 4-15. Cleanup Keywords

Keyword	Description
centermerge	Merges two parallel SRAFs within the specified distance. Used by RET SBAR only.
cleanmode	Specifies the method of SRAF removal. Used by RET SBAR only.
cleanupversion	Specifies the version of SRAF cleanup. Used by RET SBAR only.
cnsrafpromotion	Disables default promotion for cnsraf. Used by RET SBAR only.
dynamicsizing	Sizes SRAFs during generation. Used by RET SBAR and RET MBSRAF.
dynamicsizing_ambiguous	Manages the sizing of SRAFs near tile boundaries during generation. Used by RET SBAR and RET MBSRAF.
dynamicsizingrunlength	Disables centermerge emphasis on long SRAFs. Used by RET SBAR only.
interpriorityalign	Aligns SRAFs of different priorities during cleanup. Used by RET SBAR and RET MBSRAF.
intrapriorityalign	Aligns SRAFs of same priorities during cleanup. Used by RET SBAR and RET MBSRAF.
lineendmerge	Merges SRAF line-ends during SRAF cleanup. Used by RET SBAR and RET MBSRAF.
maxmove	Restricts SRAF movement during cleanup. Used by RET MBSRAF only.
reducedbarsplit	Splits or merges SRAFs in conflict. Used by RET SBAR only.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

centermerge

Cleanup Keywords

Merges two parallel SRAFs within the specified distance. Used by RET SBAR only.

Usage

centermerge *max_distance*

Arguments

- *max_distance*

  A positive real number indicating the maximum distance between which to merge two SRAFs. A negative value turns off centermerge capability.

Description

The centermerge keyword controls merging of two parallel SRAFs within the specified distance. Specifically, two initial SRAFs are replaced, for their common runlength, by a single SRAF centered between them. The default distance is the value specified by the minoffset projecting keyword, and where no projecting value is specified, by the minoffset value. The centermerge keyword may be specified globally or by priority. The centermerge behavior can be modified by the dynamicsizingrunlength keyword.

The centermerge capability can be performed before or after dynamicsizing during the cleanup process. By default, centermerge is performed after dynamicsizing. Siemens EDA recommends specifying dynamicsizingrunlength 0. This forces centermerge to be performed before dynamicsizing, cleaning up SRAFs having a common runlength of more than one third of their length.

Global and Priority Behavior Interactions

There are value-based interactions between global and priority specifications with centermerge. When centermerge is specified globally with a value of less than zero (< 0), centermerge specified per priority is only enabled when specified with a value of greater than zero (> 0). When centermerge is specified globally with a value of greater than zero (> 0), centermerge for all specified priorities is enabled except when any centermerge priority is specified with a value of less than to zero (< 0).

Examples

Example 1 — Application of centermerge Keyword

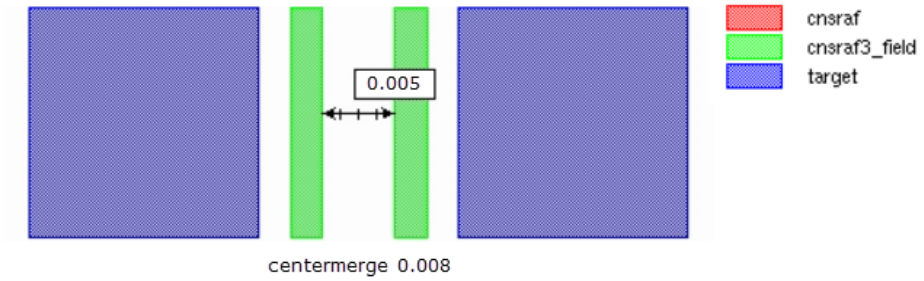
The code shown in Example 4-9 generates SRAFs initially reserved as field shapes, as shown in Figure 4-126, pending further cleanup processing by centermerge. The max_distance specified by centermerge must exceed the distance between initial SRAFs, in this case, 0.030 and 0.0222, respectively.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 4-9. centermerge Code

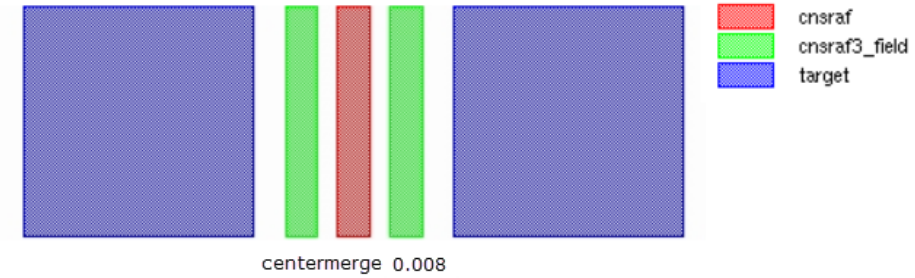
```
sraf_options OPT {
   dynamicSizing 0.032 cleanmode postSizing centermerge 0.030
   minOffset 0.020 cleanupversion 1 conflictmode merge maxconflict 0.200
   minlength 0.032 minWidth 0.010 minfeaturespace 0.010
   rectEdge tag tagext extension 0 width 0.01 offset 0.010 prior 3
}
setlayer cnsraf = cnsraf target OPTIONS OPT
```

Figure 4-126. centermerge Field SRAF Output



centermerge field prior to final SRAF output.

Figure 4-127. centermerge Post-Cleanup Output



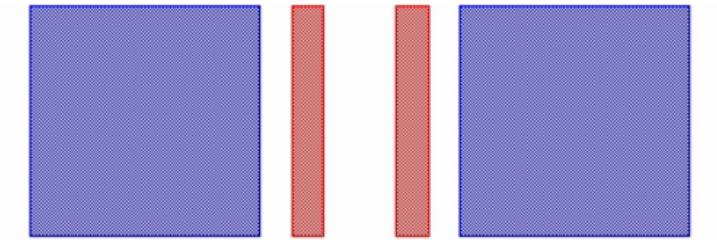
centermerge outputs merged SRAF when specified with a maximum distance of 0.008.

Example 2 — Disabling centermerge

To disable centermerge capability, specify any negative value as shown in Figure 4-128. The initial field SRAFs are not cleaned by centermerge and remain.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-128. centermerge Disabled



centermerge -1

centermerge specified with any negative value
disables center merging and outputs field SRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cleanmode

Cleanup Keywords

Specifies the method of SRAF removal. Used by RET SBAR only.

Usage

cleanmode [immediate | postsizing]

Arguments

- immediate

  Removes all SRAFs that violate MRC with input layers. For recipes specifying cleanupversion 0, this is the default behavior.

- postsizing

  For SRAFs generated using dynamicsizing, postsizing allows SRAFs that violate MRC constraints to be resized prior to cleanup instead of being removed. For recipes specifying cleanupversion 1, this is the default behavior.

Description

An optional keyword that specifies the method in which SRAFs violating MRC are removed. This keyword can be set globally for all setlayer cnsraf templates, and it can be assigned on a priority basis using the priority keyword. Setting cleanmode on a priority basis overrides the global assignment.

Tip

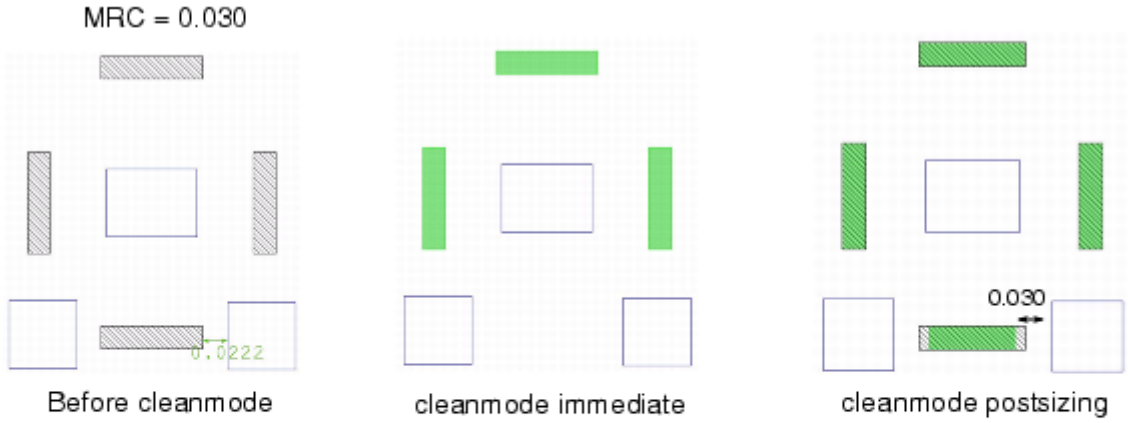
Siemens EDA recommends specifying cleanupversion 1 for all new recipes, which invokes cleanmode postsizing by default, also recommended by Siemens EDA. This mode allows the SRAFs violating MRC rules to be dynamically sized instead of immediately removed.

Examples

Figure 4-129 illustrates a recthv template using the cleanmode keyword to remove SRAFs violating MRC. The code uses cleanmode with the postsizing option. The illustration shows the output for cleanmode postsizing and also the output for cleanmode immediate. The immediate option removes any SRAFs that violate MRC. The postsizing option keeps the SRAFs that were generated using dynamicsizing and gives them the opportunity to be dynamically sized.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-129. Using cleanmode With recthv




```
setlayer cn 1 = cnsraf target newtag e1 length > 0.060 OPTIONS opt
sraf_options opt {
   cleanupversion 1
   priority 20 dynamicsizing 0.01
   recthv tag e1 width 0.02 length 0.09 pitchx 0.1 pitchy 0.12 prior 20
   minoffset 0.030 minfeaturespace 0.030 cleanmode postsizing
}
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cleanupversion

Cleanup Keywords

Specifies the version of SRAF cleanup. Used by RET SBAR only.

Usage

cleanupversion 0 | 1

Arguments

- **0**

  Forces cleanup to adhere to the original (pre-2016.3) algorithm.

- **1**

  Invokes cleanup with an updated algorithm.

Description

A required keyword that specifies the algorithmic version in which SRAFs are removed. This keyword can be set globally for all setlayer cnsraf templates.

Note

The cleanupversion 0 setting has been deprecated and is to be obsoleted in a future Calibre release.

Note

The cleanupversion keyword is required as of Calibre release 2018.1. Siemens EDA recommends always setting cleanupversion 1 for new cnSRAF recipes of line-layer and 2D topologies.

Siemens EDA also recommends always setting cleanmode postsizing with cleanupversion 1 to avoid excessive field SRAF. For existing recipes using Calibre versions prior to 2016.3, where cleanupversion 1 is specified, Siemens EDA also recommends specifying cleanmode immediate globally.

The cleanupversion 1 keyword provides a number of algorithmic improvements for SRAF generation:

- Promotion is automatically enabled.
- The dynamicsizing algorithm is automatically invoked, which aids with more intelligent conflict resolution.
- The number of templates required to achieve equivalent SRAF generation coverage is significantly reduced, which requires less tuning of cleanup parameters.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note

Various SRAF output differences may be apparent depending on the value of cleanupversion you select. Many recipes produce significantly different results when migrating from cleanupversion 0 to 1 and require cleanup tuning. Use cleanupversion 1 when developing new recipes to obtain equivalent or improved SRAF generation coverage. Other particulars regarding cleanupversion include:

- When cnsraf cleanupversion 1 is enabled, preclean debug output may not be useful, because the preclean cleanup step is not performed for cleanupversion 1.
- Promotion is enabled by default for both modes of cleanupversion as of Calibre release 2016.2.
- Using cleanupversion 1, SRAFs can be trimmed by width only if minwidth is specified. When minwidth is specified, the determination of whether an SRAF width or length is trimmed is determined by the greatest resulting SRAF area.
- For any cleanupversion, all SRAFs retain their original priority throughout all cleanup phases. SRAFs of each successive priority are cleaned with regard to the SRAFs of previously cleaned priorities.

Examples

Example 1 — Template Requirements Using the Old Cleanup Method

In the first example, templates using conflictonly and dynamicsizing by priority are separate. The conflictonly argument is used in within the template, and dynamicsizing is specified by priority in the second example, eliminating the need to exercise careful tuning of cleanup parameters.

```
rectedge tag med_end extension 0.035 width 0.019 offset 0.075
   conflictonly prior 1
rectedge tag med_end extension 0.035 width 0.0175 offset 0.075
   prior 2
rectedge tag med_end extension 0.035 width 0.0175 offset 0.075
   prior 3
rectedge tag med_end extension -0.035 width 0.0175 offset 0.075
   prior 4

priority 1 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250
   centerthresh 0.080  sbwidth 0.018 sraf 0.038 conflictmode nomerge
priority 2 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250
   centerthresh 0.080  sbwidth 0.0175 sraf 0.038 conflictmode nomerge
   cleanmode postsizing dynamicsizing 0.030
priority 3 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250
   centerthresh 0.080 sbwidth 0.0175 sraf 0.038 conflictmode nomerge
   cleanmode postsizing
priority 4 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250
   centerthresh 0.080 sbwidth 0.0175 sraf 0.038 conflictmode nomerge
   cleanmode postsizing
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

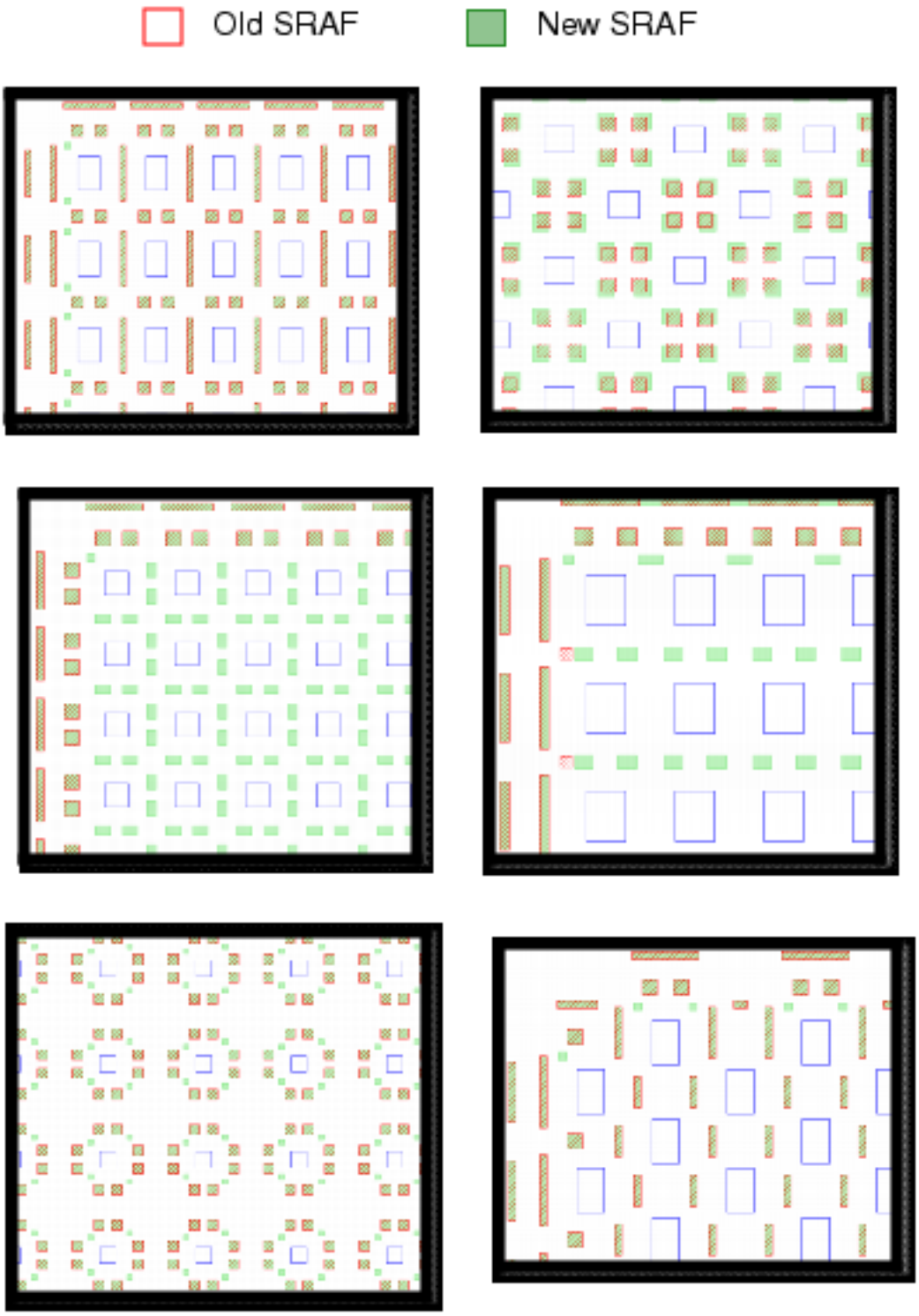
Example 2 — Equivalent Code to Achieve Coverage Using the New Cleanup Method

Output differences are shown in Figure 4-130 between two independent SRAF generation runs, using half the number of templates when using cleanupversion 1. The red SRAFs were generated using cleanupversion 0. The green SRAFs, requiring half the templates and reduced cleanup tuning parameters, were generated using cleanupversion 1.

```
rectedge tag med_end extension 0.035 width 0.0175 offset 0.075 prior 1

priority 1 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250
   centerthresh 0.080 sbwidth 0.0175 sraf 0.038 conflictmode nomerge
   cleanmode postsizing dynamicsizing 0.025
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-130. Differences Between Old and New cleanupversion Modes



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

cnsrafpromotion

Cleanup Keywords

Disables default promotion for cnsraf. Used by RET SBAR only.

Usage

cnsrafpromotion {0 | 1}

Arguments

- 0 | 1

  Disables or enables default promotion, respectively.

Description

An optional keyword that disables hierarchical promotion for cnsraf. Disabling promotion changes SRAF generation results. This keyword adopts the standard Calibre promotion scheme. A performance impact may occur depending on your design hierarchy. This keyword can only be set globally and is enabled by default.

Note

Prior to Calibre release 2016.2, cnsrafpromotion was disabled by default.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

dynamicsizing

Cleanup Keywords

Sizes SRAFs during generation. Used by RET SBAR and RET MBSRAF.

Usage

dynamicsizing *value*

Arguments

- *value*

  A positive floating point number specifying the minimum length of the trimmed SRAF.

Description

An optional keyword used globally or by priority to resize SRAFs according to local conditions and the minimum SRAF length you specify by its value (you cannot specify dynamicsizing within templates). With cnsraf, candidate SRAFs are sized with respect to other SRAFs until they are conflict-free or are removed. If you use dynamicsizing globally, all applicable templates perform dynamicsizing as specified. When you use dynamicsizing by priority, all applicable templates must specify that priority to perform dynamicsizing as specified.

Square target shapes are sized down uniformly. Rectangular target shapes are reduced in size along the long axis from either end as required. Refer to cleanmode postsizing for more information regarding MRC-clean generation of SRAFs when using dynamicsizing.

Note

Regarding dynamicsizing requirements and interactions:

- The dynamicsizing and conflictonly keywords cannot be used in the same priority.
- Siemens EDA recommends outputting intermediate debug layers for each priority when using dynamicsizing during migration processes from any earlier release to 2013.2 and later releases.

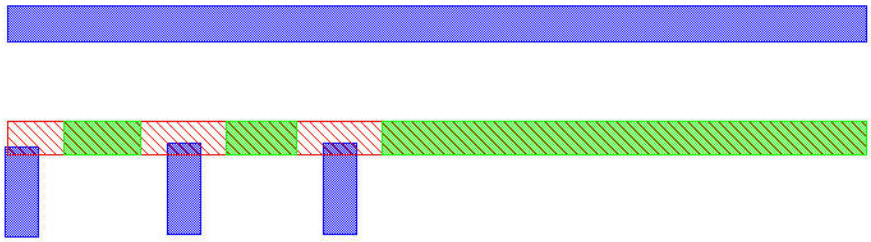
Examples

Example 1

The dynamicsizing keyword can split long SRAFs into multiple SRAFs if reasonable conditions apply as shown in Figure 4-131. The green SRAF was originally one continuous bar. In the presence of the three smaller blue bars and the use of dynamicsizing, the sections highlighted in red are removed, splitting the green SRAF into three pieces.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

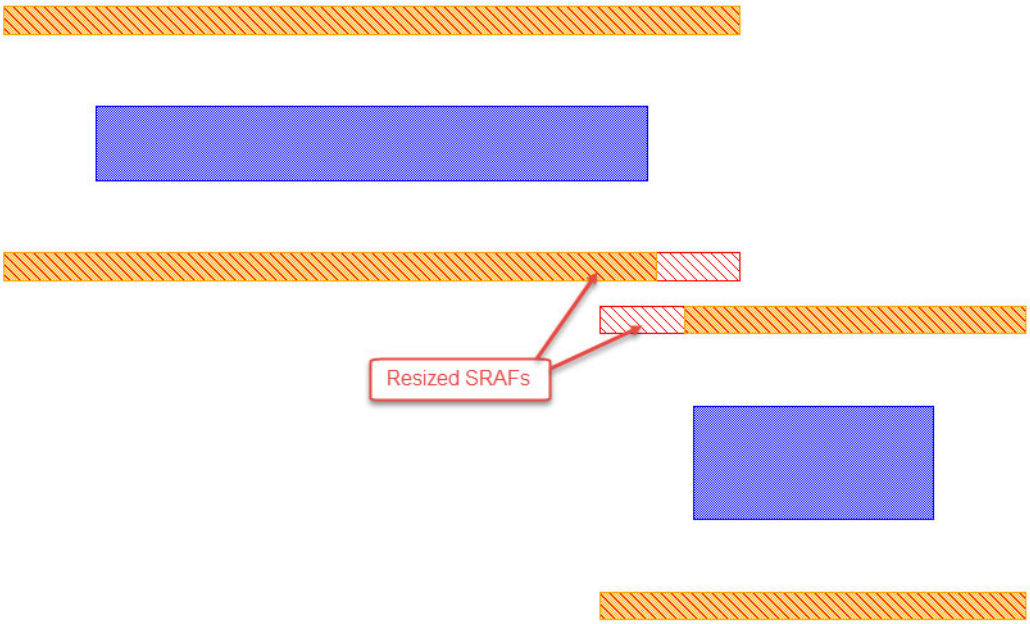
Figure 4-131. Using dynamicsizing to Split SRAFs



Example 2

The dynamicsizing keyword resizes SRAFs no shorter than the specified value as pictured in Figure 4-132. The SRAFs are reduced in length as necessary to resolve conflicts. The SRAF is either fit to its location or is removed.

Figure 4-132. Using dynamicsizing to Resize SRAFs

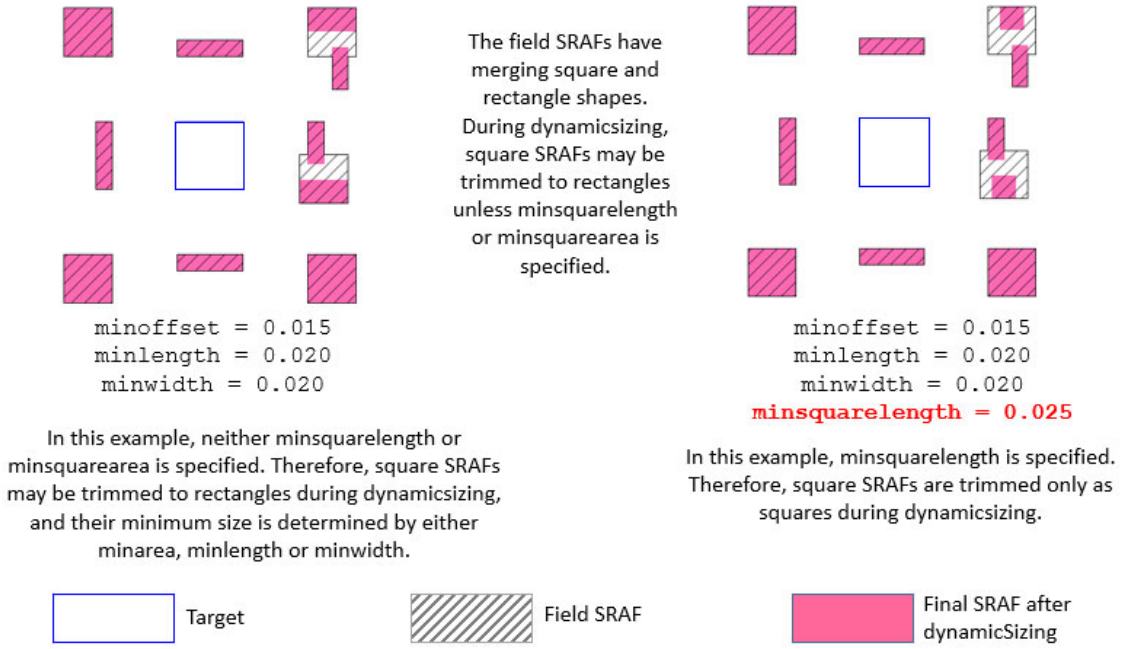


Example 3

To retain square SRAFs specified by the template after dynamicsizing, use minsquarearea or minsquarelength in your recipe. Pictured in Figure 4-133 are two recipes and their results, the one on the left without minsquarelength or minsquarearea, and the one on the right using minsquarelength.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-133. Ensuring Proper Sizing of Square SRAFs with dynamicsizing



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

dynamicsizing_ambiguous

Cleanup Keywords

Manages the sizing of SRAFs near tile boundaries during generation. Used by RET SBAR and RET MBSRAF.

Usage

dynamicsizing_ambiguous 0 | 1

Arguments

- 0 | 1

  A positive floating point number specifying the minimum length of the trimmed SRAF. The default setting is 0.

Description

An optional keyword used globally or by priority to resize SRAFs according to potentially uncertain local tile boundary conditions and the minimum SRAF length you specify by its value (you cannot specify dynamicsizing_ambiguous within templates). With cnsraf, candidate SRAFs are sized with respect to other SRAFs until they are conflict-free or are removed. If you use dynamicsizing_ambiguous globally, all applicable templates perform dynamicsizing as specified. When you use dynamicsizing_ambiguous by priority, all applicable templates must specify that priority to perform dynamicsizing as specified.

Square target shapes are sized down uniformly. Rectangular target shapes are reduced in size along the long axis from either end as required. Refer to cleanmode postsizing for more information regarding MRC-clean generation of SRAFs when using dynamicsizing.

Note

To use dynamicsizing_ambiguous with cleanupversion 1, dynamicsizing_ambiguous must be set to 1.

Note

The dynamicsizing_ambiguous and conflictonly keywords cannot be used in the same priority.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

dynamicsizingrunlength

Cleanup Keywords

Disables centermerge emphasis on long SRAFs. Used by RET SBAR only.

Usage

dynamicsizingrunlength *value*

Arguments

- 1 | 0

  Either 1 or 0, disabling or enabling dynamicsizingrunlength, respectively.

Description

The dynamicsizingrunlength keyword controls centermerge emphasis over long SRAFs. A long run length is determined by the common length of SRAFs in conflict that exceed one-third the length of each.

By default, dynamicsizingrunlength is set to 1, and centermerge emphasis is disabled, consistent with SRAF generation behavior prior to Calibre release 2017.3. Setting dynamicsizingrunlength 1 more often results in two unmerged long SRAFs normally processed by dynamicsizing during cleanup. Setting dynamicsizingrunlength 0 enables emphasis of centermerge during cleanup of long SRAFs, typically resulting in a single long SRAF output in the area of conflict. This keyword may only be specified globally.

Note

Siemens EDA recommends using dynamicsizingrunlength 0 for line layers when invoking cleanupversion 1.

Examples

When dynamicsizingrunlength is Applied

Given SRAF run lengths must exceed one-third of the length of each SRAF in conflict, these examples demonstrate when dynamicsizingrunlength is and is not applied. In Figure 4-134, SRAF results are identical with dynamicsizingrunlength set to 1 or 0, because the common run length of the SRAFs in conflict is less than the threshold required for centermerge to be enabled. In Figure 4-135, the default setting of dynamicsizingrunlength 1 emphasizes standard dynamicsizing, while dynamicsizingrunlength 0 emphasizes cleanup using centermerge.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-134. dynamicsizingrunlength Disabled

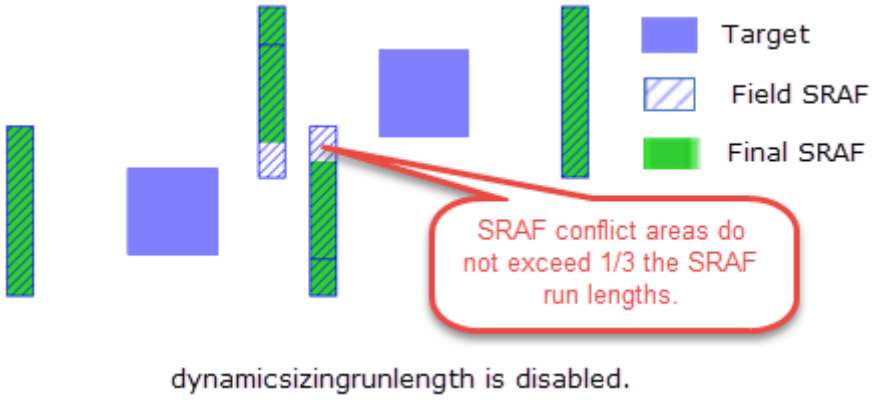
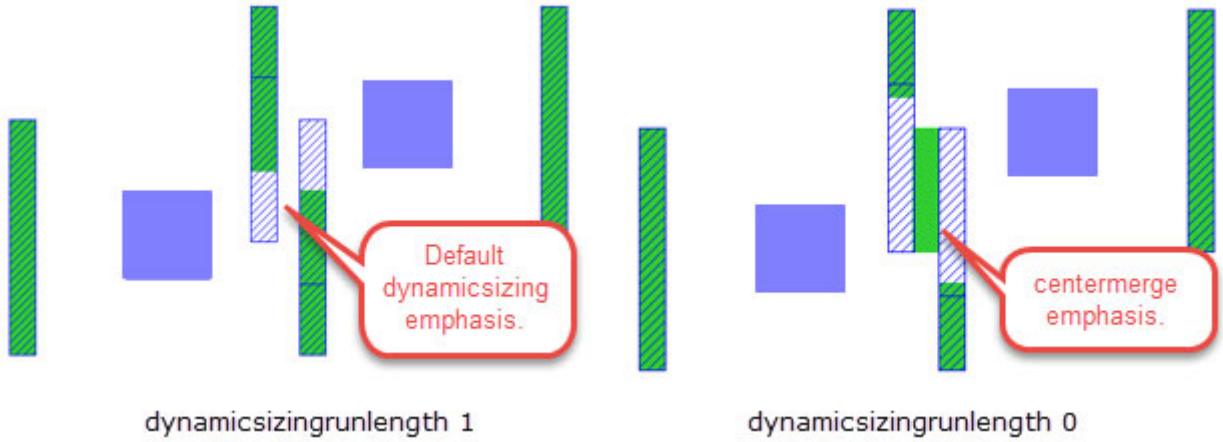


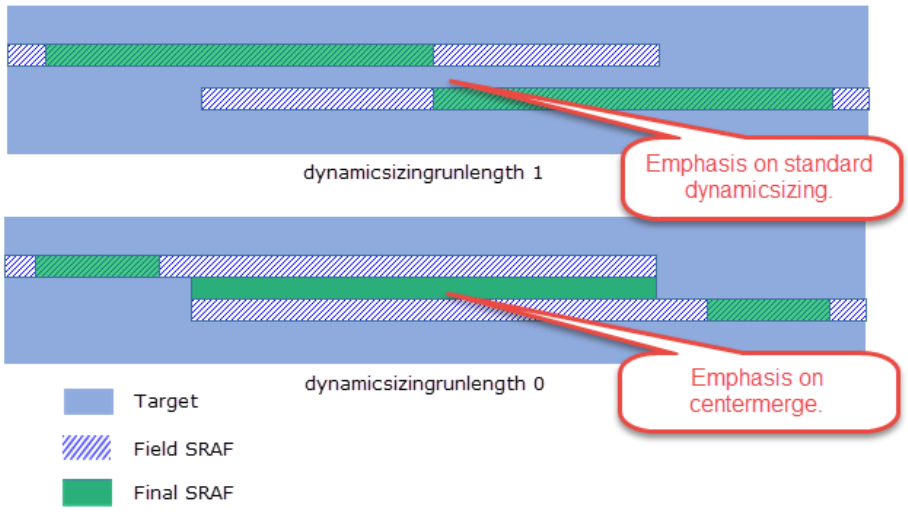
Figure 4-135. dynamicsizingrunlength Enabled



Negative SRAF Application

Figure 4-136 demonstrates dynamicsizingrunlength behavior with negative SRAFs.

Figure 4-136. dynamicsizingrunlength and Negative SRAF



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

interprioritycalign

Cleanup Keywords

Aligns SRAFs of different priorities during cleanup. Used by RET SBAR and RET MBSRAF.

Usage

interprioritycalign *value* [centermove *value*] [spacerule 1 | -1]

Arguments

- ***value***

  A required positive floating point number. The value is the maximum distance lower-priority SRAFs can be moved to align with higher priority SRAFs.

- centermove *value*

  An optional positive floating point number. This is the maximum amount of movement allowed for center SRAFs. The default value is the interprioritycalign value.

- spacerule 1 | -1

  An optional argument, either 1 or -1. A value of -1 means alignment will drive SRAFs towards target shapes. A value of 1 drives alignment away from target shapes. The default value is 1.

Description

An optional keyword specified only by priority to align and optionally merge SRAFs of different priorities during SRAF cleanup. To simultaneously permit merging, you must specify the lineendmerge option. When you specify the interprioritycalign keyword for a given priority, all SRAFs of that priority are aligned with SRAFs of higher priorities only.

Note

The interprioritycalign keyword is incompatible with cleanupversion 0 and results in a compile-time error. The interprioritycalign keyword is incompatible with maxlength.

The fundamental alignment location rules include:

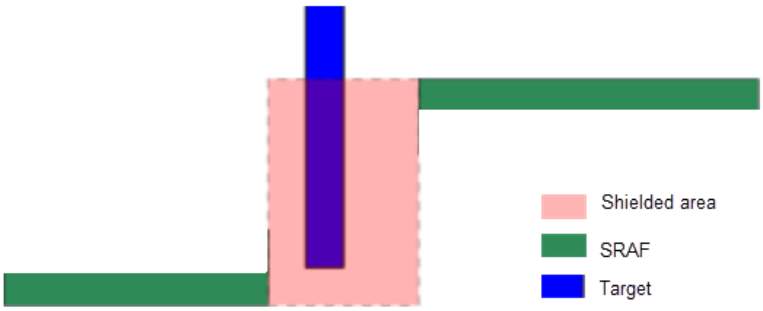
- The initial position of SRAFs are given greater emphasis during alignment processing.
- Symmetric alignment solutions are given greater emphasis during alignment processing.
- The widths of SRAFs may be adjusted by interprioritycalign during alignment and line-end merging.

For alignment of SRAFs with the different priority, these conditions must be satisfied:

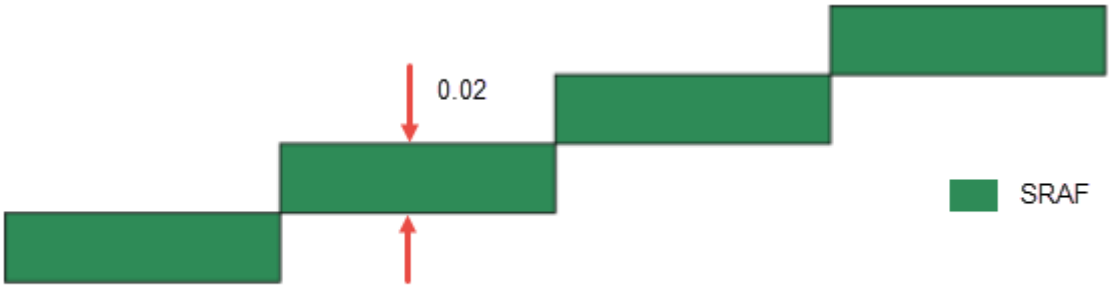
- The orientation must be the same.
- Diagonal or square SRAFs are not supported.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- A minimum distance is established from SRAFs to target shapes through specification of minfeaturespace.
- SRAFs cannot align if they are shielded by a target shape. SRAFs are not shielded where their long-end projections are overlapping. The shielded area is determined by the bounding box of the two SRAFs shown; two edges of the shielded area touch the lineend of SRAFs and the other two edges touch the boundary of the shielded area. In the example, the red box is the shielded area of the two green SRAFs. The SRAFs are shielded by the blue target shape where it appears within the shielded area, and therefore no alignment is performed.



- In the example, widths of SRAFs are 0.02 um. The first three SRAFs ascending from the left can be aligned and the three SRAFs descending from the right can be aligned together, but all four of them cannot be aligned into a single SRAF. No alignment scenarios exceeding the SRAF widths would be handled by interpriorityalign, but other cleanup steps may be available to resolve this situation.



priority lineendmerge 0.04 interpriorityalign 0.02

- The lineend distance between two SRAFs must be less than or equal to the specified lineendmerge value. The lineendmerge keyword is required for interpriorityalign to work. A parsing error is issued if lineendmerge is not specified. There is no default value for lineendmerge.
- Siemens EDA recommends values of interpriorityalign to be less than maximum width of its associated priority. If the value exceeds this amount, a warning message is issued.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The interprioritycalign keyword demonstrates these behavioral characteristics:

- The value specified for interprioritycalign acts as the search distance for the alignment of eligible SRAF.
- Only two SRAFs of the same width can be merged.
- The maximum movement for a center-placed SRAF is equivalent to the value specified for interprioritycalign.
- During alignment, SRAFs are moved away from target shapes.
- The widths of SRAFs may be adjusted by intraprioritycalign during alignment and line-end merging.

Note

Siemens EDA recommends always specifying interprioritycalign by priority. A warning message is issued to the transcript when interprioritycalign is specified globally.

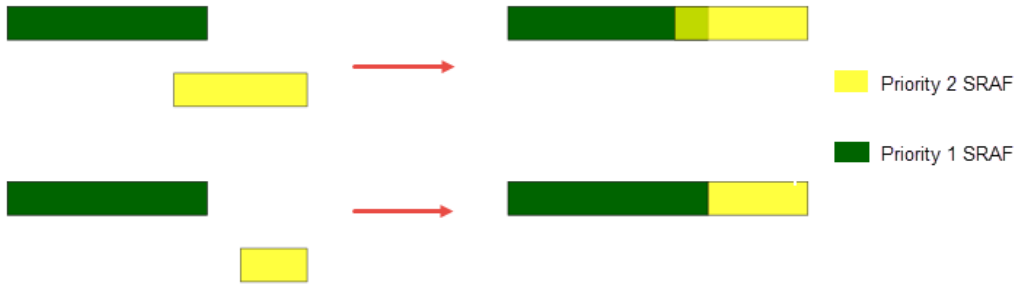
Examples

In Figure 4-17, all SRAFs generated by the rectedge template specifying prior 2 would be moved to align with the SRAFs generated by prior 1 (because lineendmerge is also specified, the prior 2 SRAFs would also be merged with those of prior 1).

Example 1 — interprioritycalign Code

```
rectEdge tag tag1 extension 0.020 width 0.040 offset 0.060 prior 1
rectEdge tag tag2 extension 0.020 width 0.040 offset 0.080 prior 2
priority 2 lineendmerge 0.020 interprioritycalign 0.030
```

Figure 4-137. interprioritycalign



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

intrapriorityalign

Cleanup Keywords

Aligns SRAFs of same priorities during cleanup. Used by RET SBAR and RET MBSRAF.

Usage

intrapriorityalign *value* [centermove *value*] [spacerule 1 | -1]

Arguments

- *value*

  A required positive floating point number. The value is the maximum distance lower priority SRAFs can be moved to align with higher priority SRAFs.

- centermove *value*

  An optional positive floating point number. This is the maximum amount of movement allowed for center SRAFs. The default value is the intrapriorityalign value.

- spacerule 1 | -1

  An optional argument, either 1 or -1. A value of -1 moves SRAFs away from target shapes during alignment. A value of 1 moves SRAFs toward target shapes during alignment. The default value is -1.

Description

An optional keyword that enables valid, generated SRAFs of same priorities to align and merge during SRAF cleanup. This option can be set globally or per priority. If set globally and not specified for any priority, the global setting is adopted. Before alignment, SRAFs are trimmed with respect to target shapes because the projecting distance and shielding are not well defined when SRAFs overlap targets. You may use maxlength in conjunction with intrapriorityalign.

Note

The intrapriorityalign keyword is incompatible with cleanupversion 0 and results in a compile-time error.

The fundamental alignment location rules include:

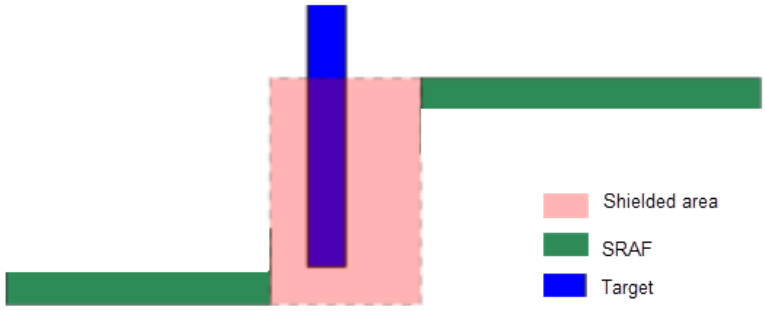
- The initial position of SRAFs are given greater emphasis during alignment processing.
- Symmetric alignment solutions are given greater emphasis during alignment processing.
- The widths of SRAFs may be adjusted by intrapriorityalign during alignment and line-end merging.

For alignment of SRAFs with the different priority, these conditions must be satisfied:

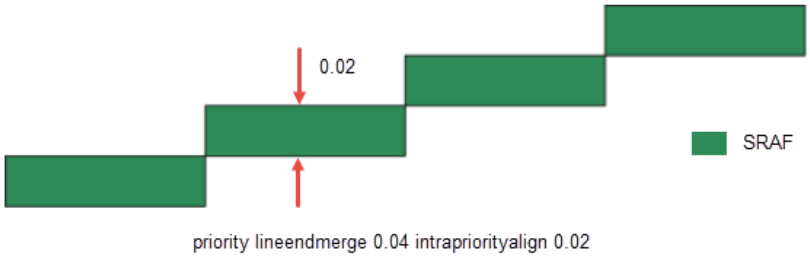
- The orientation must be the same.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Diagonal or square SRAFs are not supported.
- A minimum distance is established from SRAFs to target shapes through specification of minfeaturespace.
- SRAFs cannot align if they are shielded by a target shape. SRAFs are not shielded where their long-end projections are overlapping. The shielded area is determined by the bounding box of the two SRAFs shown; two edges of the shielded area touch the lineend of SRAFs and the other two edges touch the boundary of the shielded area. In the example, the red box is the shielded area of the two green SRAFs. The SRAFs are shielded by the blue target shape where it appears within the shielded area, and therefore no alignment is performed.



- In the example, widths of SRAFs are 0.02. The first three SRAFs ascending from the left can be aligned and the three SRAFs descending from the right can be aligned together, but all four of them cannot be aligned into a single SRAF. No alignment scenarios exceeding the SRAF widths would be handled by intrapriorityalign, but other cleanup steps may be available to resolve this situation.



- The lineend distance between two SRAFs must be less than or equal to the specified lineendmerge value. The lineendmerge keyword is required for intrapriorityalign to work. A parsing error is issued if lineendmerge is not specified. There is no default value for lineendmerge.
- Siemens EDA recommends values of intrapriorityalign to be less than maximum width of its associated priority. If the value exceeds this amount, a warning message is issued.

The intrapriorityalign keyword demonstrates these behavioral characteristics:

- The value specified for intrapriorityalign acts as the search distance for the alignment of eligible SRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Only two SRAFs of the same width can be merged.
- The maximum movement for a center-placed SRAF is equivalent to the value specified for intrapriorityalign.
- During alignment, SRAFs are moved away from target shapes.
- The widths of SRAFs may be adjusted by intrapriorityalign during alignment and line-end merging.

Note

Siemens EDA recommends always specifying intrapriorityalign by priority. A warning message is issued to the transcript when intrapriorityalign is specified globally.

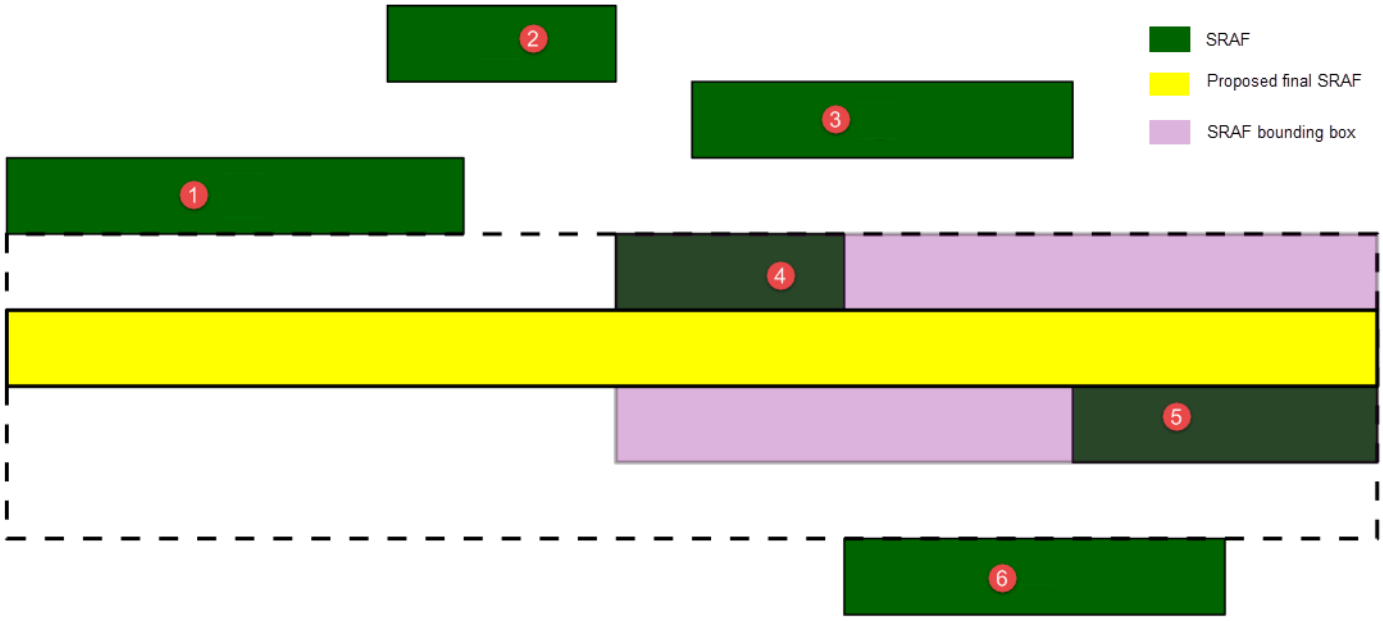
Examples

Example 1 — intrapriorityalign Code

In Figure 4-138, all SRAFs generated by the rectedge template specifying an offset of 0.050 would be moved to align with the SRAFs generated by the rectedge template specifying an offset of 0.080. The dashed line represents the feasible range of merged SRAFs. If no SRAFs are found within the range, SRAFs are merged in the middle of the range. The pink area is the bounding box of the SRAFs found in the range. The merged SRAF (shown in yellow) is generated between the SRAFs forming the bounding box.

```
rectEdge tag tag1 extension 0.020 width 0.030 offset 0.050 prior 1
rectEdge tag tag1 extension 0.020 width 0.040 offset 0.080 prior 1
priority 1 lineendmerge 0.020 intrapriorityalign 0.030
```

Figure 4-138. intrapriorityalign



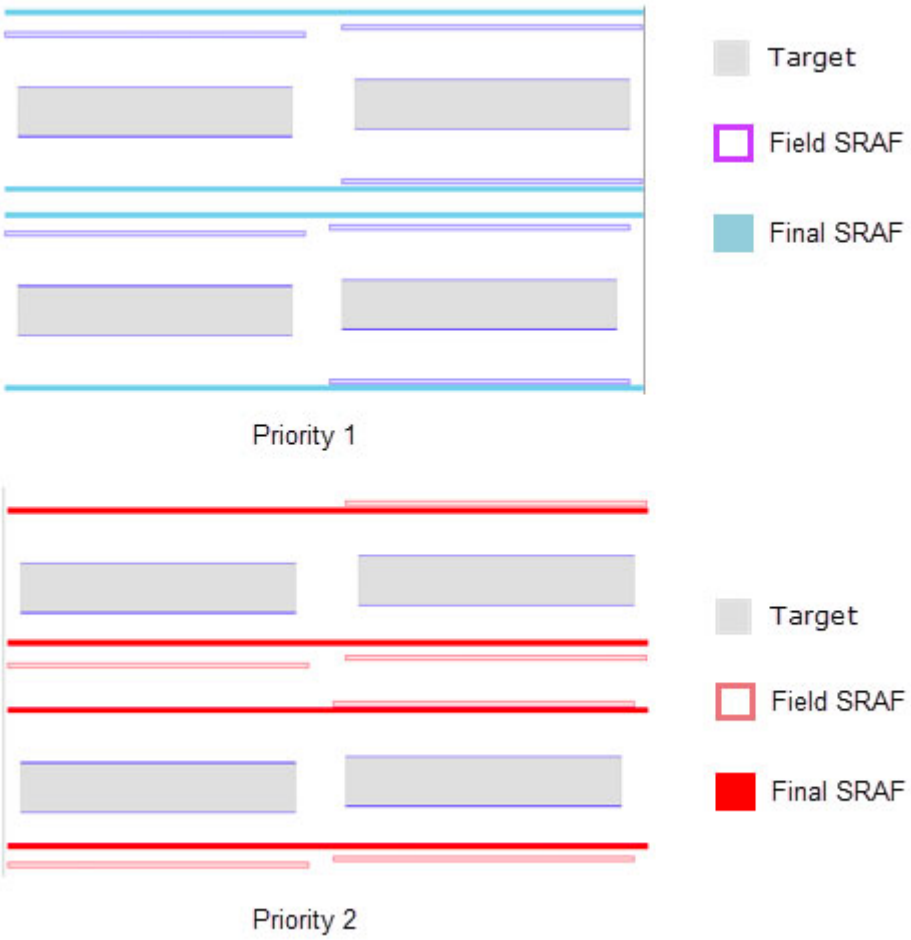
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Example 2 — intrapriorityalign Using spacerule

In Figure 4-139, field SRAFs for priority 1 and priority 2 are identical. In the illustration, intrapriorityalign with a priority -1 moves SRAFs away from the target shape (the default behavior), and intrapriorityalign with a priority 2, using spacerule, moves SRAFs toward the target shape.

```
rectEdge tag longEdges extension 0.01 width 0.004 offset 0.040 prior 1
rectEdge tag longEdges extension 0.01 width 0.004 offset 0.040 prior 1
priority 1 lineendmerge 0.07 intrapriorityalign 0.030
priority 2 lineendmerge 0.07 intrapriorityalign 0.030 spacerule 1
```

Figure 4-139. intrapriorityalign With spacerule



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

lineendmerge

Cleanup Keywords

Merges SRAF line-ends during SRAF cleanup. Used by RET SBAR and RET MBSRAF.

Usage

lineendmerge {*value* | no | 0}

Arguments

- *value*

  A positive floating point number establishing a limit for line-end merging.

- no

  Disables all line-end merging. Existing overlapping SRAFs are separated. Use of lineendmerge no is incompatible with the interpriorityalign, intrapriorityalign, and intersection keywords. The no option is incompatible with cleanupversion 0 and results in a compile-time error.

- 0

  Allows line-end merging only of field (preliminary) SRAFs that align or overlap. Use of lineendmerge 0 is incompatible with the interpriorityalign, intrapriorityalign, and intersection keywords. The 0 option is incompatible with cleanupversion 0 and results in a compile-time error.

Description

An optional keyword used to merge SRAFs at line-ends. The keyword can be specified globally or by priority. Conditions and behaviors include:

- SRAFs must be exactly aligned, so that no jogs result after merging.
- Distances between two SRAFs are less than or equal to the specified value. There is no default value for lineendmerge.
- If the recipe specifies cleanupversion 0, dynamicsizing may need to be used to achieve intended results of lineendmerge in some cases.

Note

When lineendmerge is specified globally, line-ends of different priorities may be merged. Therefore, Siemens EDA recommends only specifying lineendmerge by priority.

The intrapriorityalign and interpriorityalign keywords cannot be used with lineendmerge 0 or lineendmerge no. The lineendmerge keyword requires a floating-point value to be used with intrapriorityalign and interpriorityalign. If selective SRAF merging is required using tags, Siemens EDA recommends using unique tags and templates for those cases and setting lineendmerge by priority to continue using intrapriorityalign and interpriorityalign keywords globally.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Examples

Example 1

This example shows the lineendmerge keyword called by a rectedge template to merge SRAFs.

Figure 4-140. Using lineendmerge With rectedge

```
setlayer rect_edge = cnsraf sraf_target
 newtag CA recta > 0 <= 0.050 OPTIONS opt

sraf_options opt {
    minoffset 0.040 maxconflict 0.5
    rectedge tag CA width 0.028 offset 0.150 extension 0.01 prior 4
    priority 4 lineendmerge 0.120
    ...
}
```

Example 2

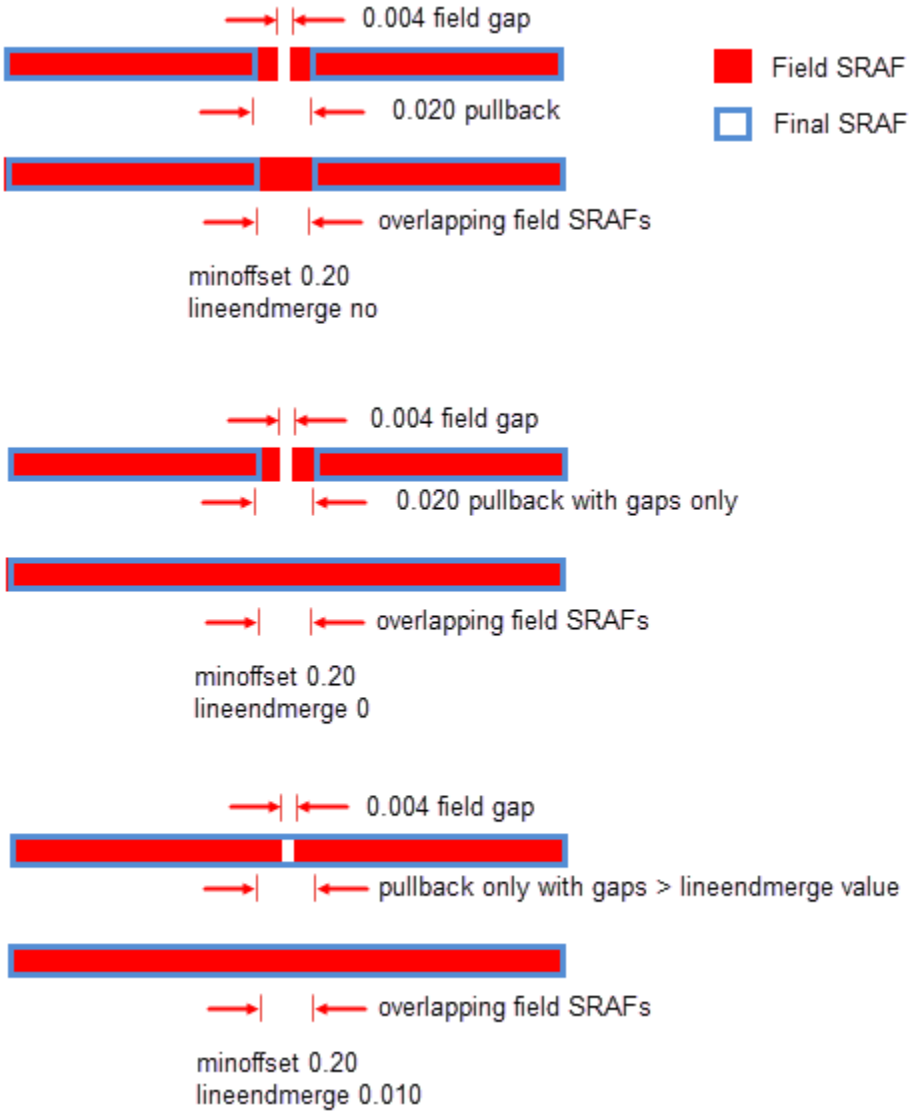
These examples show various lineendmerge parameter settings and their respective behavior. The first picture demonstrates lineendmerge no, where uniform line-end pullback at the specified minoffset value is applied for gaps of any value, or overlapping field SRAFs.

The second picture demonstrates lineendmerge no, where line-end pullback at the specified minoffset value is applied only for gaps of any value in field SRAFs.

The third picture demonstrates lineendmerge with a specified value (in this case 0.010), where line-end pullback is only applied to field SRAFs exceeding the specified value.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-141. lineendmerge Parameters and Their Behavior



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

maxmove

Cleanup Keywords

Restricts SRAF movement during cleanup. Used by RET MBSRAF only.

Usage

maxmove *value*

Arguments

- *value*

  A positive floating point number.

Description

An optional keyword used to restrict SRAF movement during cleanup. For example, maxmove is used to limit the distance an SRAF can be moved from its initially generated position during cleanup to comply to MRC constraints. Where maxmove is set to 0nm, no movement of SRAFs is permitted during cleanup. For recipes where algorithm 0 is specified, the default value is the specified width value divided by 2. Where algorithm 1 is specified, the default value is 1nm.

Note

Prior to release 2018.1, the default for maxmove where algorithm 1 was specified, was 0nm. The maxmove default as of Calibre release 2018.1 is 1nm. Therefore, existing recipes using algorithm 1 must explicitly specify maxmove 0 to obtain the previously generated SRAF results.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

reducedbarsplit

Cleanup Keywords

Splits or merges SRAFs in conflict. Used by RET SBAR only.

Usage

reducedbarsplit *mode*

Arguments

- *mode*

  Four different modes can be specified:

  - 0 — Processes SRAF candidates perpendicular to one another. Mode 0 splits the top and prioritizes the stem of the T formed between perpendicular SRAF candidates. Siemens EDA recommends setting a value of 0.
  - 1 — Processes SRAF candidates perpendicular to one another. Mode 1 splits the stem and prioritizes the top of the T formed between perpendicular SRAF candidates. This is the default value.
  - 2 — Processes all non-parallel SRAFs. Mode 2 splits the top and prioritizes the stem of the T formed between non-parallel SRAF candidates.
  - 3 — Disables bar-splitting of SRAFs.

Description

The reducedbarsplit keyword removes conflicts between perpendicular SRAFs. Bar splitting is the only cleanup process that breaks existing SRAF candidates into multiple segments. The reducedbarsplit keyword controls which SRAFs are selected for bar splitting.

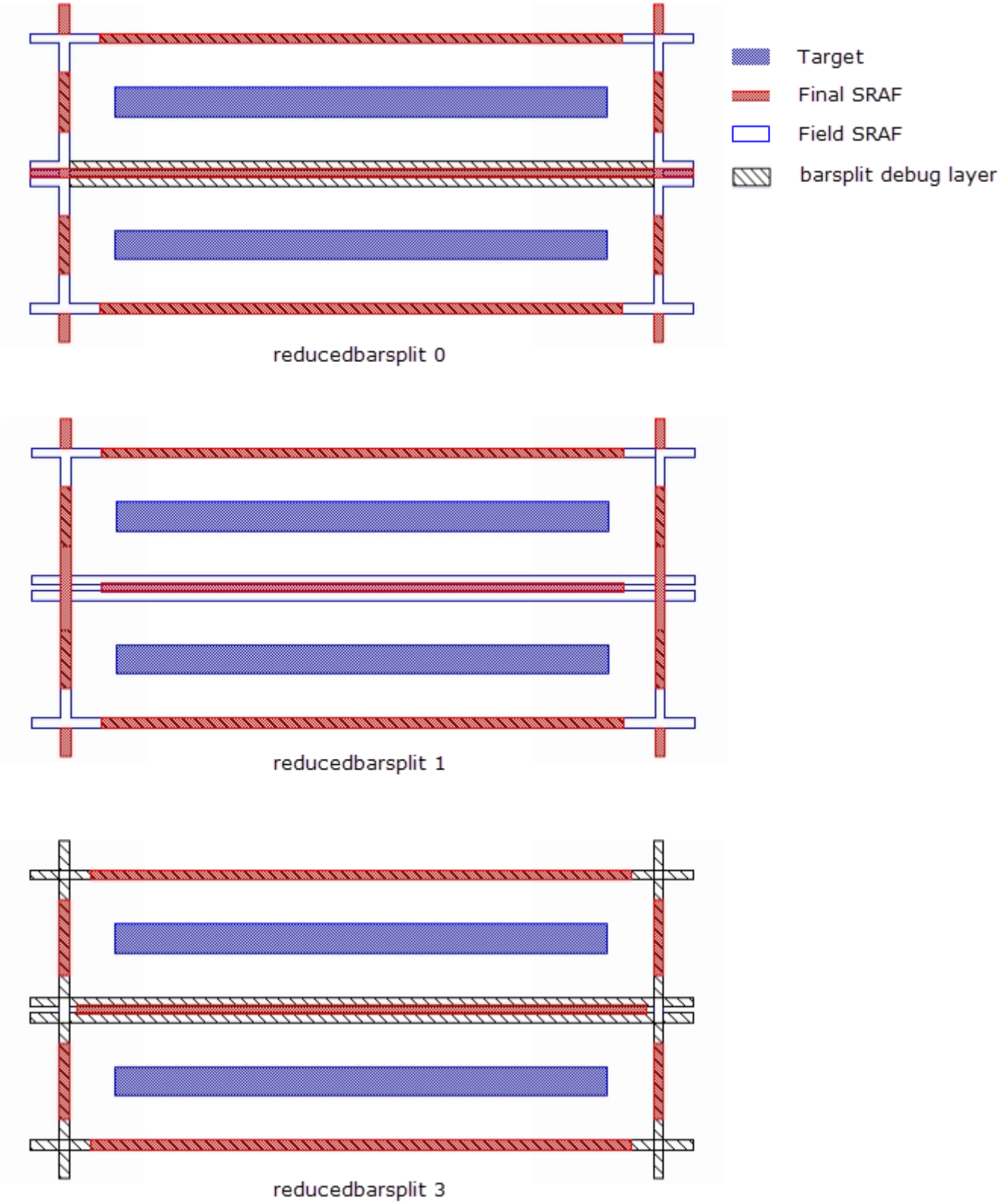
Examples

Differences Between reducedbarsplit Settings

The following figure demonstrates how various modes of reducedbarsplit are applied.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 4-142. reducedbarsplit Modes



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Priority Keywords

Calibre nmSRAF provides many keywords to be called by priority.

For organizational keyword information, refer to "sraf_options" on page 80.

Table 4-16. Priority Keywords

Keyword	Description
priority	Overrides global parameters by priority as specified by each template. Used by RET SBAR and RET MBSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

priority

Priority Keywords

Overrides global parameters by priority as specified by each template. Used by RET SBAR and RET MBSRAF.

Usage

For RET MBSRAF:

priority *number* [to *number*] [with *number* [to *number*]]
[*conditional_cleanup_sets*]
[dynamicsizing *value*]
[interpriorityalign *value*]
[intersection J [minjogwidth *value*]]
[intersectionnegative {J | N} [minjogwidth *value*]]
[intrapriorityalign *value*]
[lineendmerge *value*]
[minarea *value*]
[minfeaturespace *value* [lineend *value*] [projecting *value*]]
[minlength *value*]
[minoffset *value* [lineend *value*] [projecting *value*]]
[minoffsetlayerspace *value*]
[minsquarearea *value*]
[minsquarelength *value*]
[negative]
[shielding {no | seedpixel | distance}]

For RET SBAR:

priority *number* [to *number*] [with *number* [to *number*]]
[*conditional_cleanup_sets*]
[centermerge *value*]
[centerthresh *value* [sbwidth *value*]]
[cleanmode {immediate | postsizing}]
[conflictmode {merge | nomerge | shift}]
[dynamicsizing *value*]
[interpriorityalign *value*]
[intersection J [minjogwidth *value*]]
[intersectionnegative {J | N} [minjogwidth *value*]]
[intrapriorityalign *value*]
[lineendmerge *value*]
[maxconflict *value*]
[minarea *value*]
[minfeaturespace *value* [lineend *value*] [projecting *value*]]
[minlength *value*]
[minoffset *value* [lineend *value*] [projecting *value*]]
[minoffsetlayerspace *value*]

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

[minsquarearea *value*]
[minsquarelength *value*]
[negative]
[oversizesraf {no | yes}]
[sbwidth *value*]
[sraf *value*]

Arguments

- *number* [to *number*]

  The first number argument specifies the priority of SRAFs to which all subsequent keywords are applied, and is required. An optional second number following the 'to' keyword is a lower priority limit establishing a range of priorities to which all subsequent keywords are applied. The priorities are always applied sequentially: a priority of 1 is applied first, then a priority of 2 is applied. The default priority is 1.

- with *number* [to *number*]

  Optional keywords and respective values specifying a single priority or a range of priorities for conflicts between SRAFs. The first number argument specifies the priority of SRAFs to which all subsequent keywords are applied, and is required. An optional second number following the 'to' keyword is a lower priority limit establishing a range of priorities to which all subsequent keywords are applied. The specified keywords following 'with' are applied only to SRAFs of the specified priority. The priority specified for 'with' must be a lower number than those specified for 'to'. SRAF cleanup is performed from the lowest priority number sequentially through the highest priority number. When cleanup has been performed for any priority *P*, those MRC-clean SRAFs for priority *1* through *P-1* remain unmodified. SRAFs of priorities *P+1* and upwards are not considered during cleanup for priority *P*.

- *conditional_cleanup_sets*

  One of eight optional keyword sets specifying a constraint and range or name, and the MRC keyword and value pair with optional modifier keywords and values. The sets must be specified with ordered keywords:

  constraint argument MRC value modifier value

  constraint argument

  One of eight optional keyword and argument pairs:

  area *range*

  A constraint measuring the area of the contending SRAF. The range may be bounded or unbounded.

  aspect *range*

  A constraint measuring the aspect ratio of the contending SRAF. The range may be bounded or unbounded.

  commonrunlength *range*

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

A constraint measuring the run length of a continuous edge of the contending SRAF. The range may be bounded or unbounded.

length *range*

A constraint measuring the length of the contending SRAF. The range may be bounded or unbounded.

squarearea *range*

A constraint measuring the area of any square contending SRAF. The range may be bounded or unbounded.

squarelength *range*

A constraint measuring the length of an edge of any square contending SRAF. The range may be bounded or unbounded.

tag *name*

A constraint specifying the name of tagged edges of the contending SRAF.

width *range*

A constraint measuring the width of the contending SRAF. The range may be bounded or unbounded.

MRC value

One of four optional MRC keywords and respective values:

minfeaturespace *value*

An optional MRC determining the minimum space between SRAF and target shapes. The SRAF shape is always the contending shape.

minfeaturespace *value*

An optional MRC determining the minimum space between SRAF and target shapes. The SRAF shape is always the contending shape.

minoffset *value*

An optional MRC determining the minimum space between SRAFs shapes.

minoffsetlayerspace *value*

An optional MRC determining the space between SRAFs and any shape designated by offsetlayer. The SRAF shape is always the contending shape.

seedoffset *value*

An optional MRC determining the minimum space between SRAF seeds and target shapes. The SRAF seed is always the contending shape. Used only by mbSRAF.

modifier value

An optional modifier and its respective value specified after the MRC keyword. See minoffset for comprehensive definitions of all modifiers:

corner2sqcorner *value*

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

lineend *value*

lineend2long *value*

lineend2sqedge *value*

long2lineend *value*

long2sqedge *value*

projecting *value*

sqcorner2corner *value*

sqcorner2sqcorner *value*

sqedge2lineend *value*

sqedge2long *value*

sqedge2sqedge *value*

Conditional cleanup sets specified within the priority keyword provide great flexibility for SRAF cleanup. Combinations of constraints for a single priority as seen here provide complex cleanup methods:

```
priority 10 tag cover minfeaturespace 0.010
priority 10 tag cover minoffset 0.060
priority 11 width > 0.1 tag cover minoffset 0.060
priority 12 width > 0.1 minoffset 0.040
priority 13 tag cover width > 0.1 minoffset 0.040 length < 0.2
   tag cover minfeaturespace 0.050
```

- centermerge *value*

  An optional keyword that specifies the maximum distance within which to merge parallel SRAFs. For more information, see centermerge in the sraf_options statement.

- centerthresh *value* [sbwidth *value*]

  An optional keyword that specifies the minimum length of a conflict region where an SRAF is added. For more information, see centerthresh in the sraf_options statement.

- cleanmode [immediate | postsizing]

  An optional keyword that specifies the method by which SRAFs violating MRC are removed. For more information, see cleanmode in the sraf_options statement.

- conflictmode [merge | nomerge | shift]

  An optional keyword that specifies the method in which conflict regions are processed. For more information, see conflictmode in the sraf_options statement.

- dynamicsizing *value*

  An optional keyword used globally or by priority to resize SRAFs according to local conditions and the minimum SRAF length specified by the value. You may not specify dynamicsizing within a template. The dynamicsizing keyword and conflictonly argument cannot be used within the same priority. For more information, see dynamicsizing in the sraf_options statement.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- interpriorityalign *value*

  An optional keyword specified only by priority to align and optionally merge SRAFs of different priorities during SRAF cleanup. To simultaneously permit merging, you must specify the lineendmerge option. When you specify the interpriorityalign keyword for a given priority, all SRAFs of that priority are aligned with SRAFs of higher priorities only. For more information, see interpriorityalign in the sraf_options statement.

- intersection J [minjogwidth *value*]

  An optional keyword that enables valid, generated SRAFs to be joined at jog point intersections. The intersection J keyword can be used with either positive or negative SRAFs. The default jog width (established by the template width keyword) can be overridden with minjogwidth. For more information, see intersection in the sraf_options statement.

- intersectionnegative {J | N} [minjogwidth *value*]

  An optional keyword that enables valid, generated negative SRAFs to be joined at jog point intersections. The intersectionnegative keyword is used only for negative SRAF generation and should only be set globally. The default jog width (established by the template width keyword) can be overridden with minjogwidth. For more information, see intersectionnegative in the sraf_options statement.

- intrapriorityalign *value*

  An optional keyword specified only by priority to align and optionally merge SRAFs of same priorities during SRAF cleanup. To simultaneously permit merging, you must specify the lineendmerge option. When you specify the intrapriorityalign keyword for a given priority, all SRAFs of that priority are aligned with SRAFs of the same priorities only. For more information, see intrapriorityalign in the sraf_options statement.

- lineendmerge *value*

  An optional keyword used to merge two SRAFs at line-ends that are exactly aligned and are less than the lineendmerge value specified. For more information, see lineendmerge in the sraf_options statement.

- maxconflict *value*

  An optional keyword that specifies the limit on the size of a conflict region to process. For more information, see maxconflict in the sraf_options statement.

- minarea *value*

  An optional keyword that specifies the minimum area allowed for any SRAF. SRAFs with an area smaller than minarea are eliminated. For more information, see minarea in the sraf_options statement.

- minfeaturespace *value* [lineend *value*] [projecting *value*]

  An optional keyword that specifies the minimum distance an SRAF is placed from an input layer target shape. For more information, see minfeaturespace in the sraf_options statement.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- minlength *value*

  An optional keyword that specifies the minimum length allowed for rectangular SRAFs. SRAFs with any side of a length smaller than minlength are eliminated during cleanup. The minlength keyword does not apply to square SRAFs. For more information, see minlength in the sraf_options statement.

- minoffset *value* [lineend *value*] [projecting *value*]

  An optional keyword that specifies the minimum distance allowed between SRAFs. Any SRAFs that fall within this distance are treated as conflicts and are processed according to keywords listed in "Cleanup Keywords" on page 284. For more information, see minoffset in the sraf_options statement.

- minoffsetlayerspace *value*

  An optional keyword that specifies the minimum distance from any SRAF shape to the offsetlayer. You derive the offsetlayer for additional line-end keepout areas. For more information, see minoffsetlayerspace in the sraf_options statement.

- minsquarearea *value*

  An optional keyword that specifies the minimum area allowed for any square SRAF in square microns. Square SRAFs with an area smaller than minsquarearea are eliminated. If minsquarearea is not specified, the minimum area allowed for square SRAFs defaults to minarea. For more information, see minsquarearea in the sraf_options statement.

- minsquarelength *value*

  An optional keyword that specifies the minimum length allowed for any square SRAF. Square SRAFs with a length smaller than minsquarelength are deleted. The minsquarelength keyword does not apply to non-square SRAFs. For more information, see minsquarelength in the sraf_options statement.

- negative

  An optional keyword that specifies the generation of negative SRAFs or slots. Using the negative keyword as a priority allows cnsraf to generate positive and negative SRAFs in a single run. For more information, see negative in the sraf_options statement.

- oversizesraf [no | yes]

  An optional keyword that allows SRAFs in conflict regions to be larger in size than the conflict region itself. By default, this is set to no, cropping SRAFs to the size of the conflict region. You can use oversizesraf in conjunction with the sraf keyword, or you can use oversizesraf separately by priority. For more information, see oversizesraf in the sraf_options statement.

- sbwidth *value*

  An optional keyword that specifies the width of the secondary scattering bars. You can use sbwidth in conjunction with centerthresh, or you can use sbwidth separately by priority. For more information, see centerthresh in the sraf_options statement.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- sraf *value*

  An optional keyword that specifies the square size of the SRAFs added in the conflict region. For more information, see sraf in the sraf_options statement.

- shielding {distance | no | seedpixel}

  An optional keyword that specifies shielding of tagged edges to alter SRAF generation relative only to the tagged edges. Shielding may be specified globally or by priority. Shielding is compatible with positive and negative SRAF generation. For more information, see shielding in the sraf_options statement.

Description

An optional keyword used to override conflict resolution and MRC parameters specified globally. The priority keyword must be followed by number and any additional keywords.

If the prior keyword is not specified in the template, the global keyword and assigned parameter value is used. For complete information regarding these keywords, see their description in the sraf_options statement.

Note

When using priority-based keywords, global keywords and their respective options must be specified first, otherwise Calibre nmSRAF issues warnings and, possibly, compile-time error messages. Calibre assumes any keyword specified after the priority keyword to be priority based. For any parameter to be considered global, it must be specified before the priority keyword.

Examples

The following examples demonstrate priority usage in context.

Establishes negative SRAF generation for priority 34:

```
rectEdge tag intEdges extension 0.08 offset 0.17 width 0.07  prior 34
priority 34 negative
```

Applies oversized SRAFs and a minimum space from the offset layer to SRAFs generated by priorities 3 through 5:

```
rectCorner offsetx 0.16 offsety 0.16 length 0.07 width 0.02 prior 3
rectCorner offsetx 0.18 offsety 0.18 length 0.07 width 0.02 prior 4
rectCorner offsetx 0.20 offsety 0.20 length 0.07 width 0.02 prior 5
priority 3 to 5 oversizesraf yes minoffsetlayerspace 0.057
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Chapter 5
Calibre nmSRAF Best Practices

Siemens EDA provides several tools for SRAF insertion, each taking different approaches and working together to provide optimized results.

This chapter outlines several best practices when using Calibre nmSRAF.

Calibre nmSRAF Recipe Development 323
Calibre mbSRAF Recipe Development 333
Calibre cnSRAF Template Development Tips 343
Recommended Settings for Best cnsraf Results 345
Recommended Settings for Best mbsraf Results 348
Calibre nmSRAF Recipe LINT Messages 349

Calibre nmSRAF Recipe Development

When developing recipes, Siemens EDA recommends you start from isolated layout features and transition to layouts with greater feature density.

Priority Usage

Rules should be prioritized to apply closer SRAFs first and secondary SRAFs later. The secondary SRAFs should be placed at a minimum distance from SRAFs with the highest priority.

Global and Priority Precedence

When using priority, global keywords and their respective options must be specified first, otherwise nmSRAF issues warning messages. The specification of any keyword after the priority keyword forces the specification to be priority-based and not global. For any parameter to be considered global, it must be specified before the priority keyword.

Use Priority for Improved Processing and Treatment

Although formulating recipes is often easier when specifying keywords globally, a number of important considerations may impact processing and treatment efficacy:

- **Runtime —** In general, Siemens EDA recommends using lineendmerge 0 or no, and intrapriorityalign and interpriorityalign, and conditional MRCs such as lineend2long, long2lineend, and square2long per priority (restricted to required applications only), not globally, to avoid runtime impact.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **Consistency —** The consistency of generated SRAFs may be negatively impacted using priority-related keywords such as interpriorityalign, intrapriorityalign, maxlength, and others.
- **Predictability —** Results of intrapriorityalign, and particularly, interpriorityalign may be unpredictable at tile boundaries.

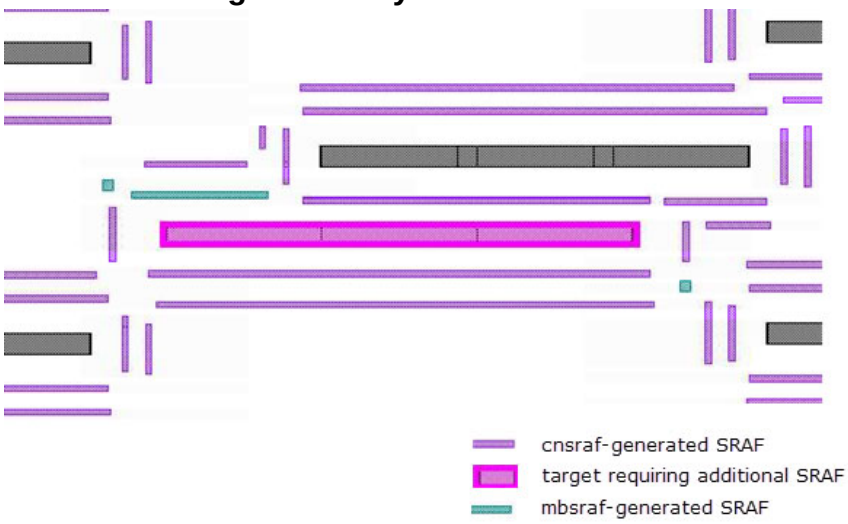
Hybrid Recipes

Hybrid recipes are defined as combinations of Calibre cnSRAF- and Calibre mbSRAF-generated SRAFs applied to the same design (whether executed together or separately). Calibre mbSRAF provides an SRAF generation safety net for ensuring SRAF coverage that may be potentially missed. If the initial SRAF generation is rule-based, the secondary model-based SRAF generation provides complete SRAF coverage.

This requires that Calibre mbSRAF process the entire design, achieving complete coverage, with an expected increase in associated simulation runtime. Alternatively, if the rule-based SRAF generation can be defined exclusively for a large portion of the design (for example, SRAM) or a complete rules-based recipe can be generated, SVRF data-processing rules can be used to isolate regions which do not require SRAF coverage within a specific distance of the region. These isolated shapes and edges can be passed as targets to Calibre mbSRAF, with the rules-based SRAFs as an sblayer. Because of the limited target areas requiring simulation, this method provides complete SRAF coverage with potentially no additional mbsraf runtime.

In Figure 5-1, after cnsraf-generated SRAF is output, the pink target still needs additional coverage for good fidelity. The purple cnsraf-generated SRAF is then passed to sblayer, imported to mbsraf and the green mbsraf-generated SRAF is finally added.

Figure 5-1. Hybrid SRAF Generation



Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

minarea, minlength, and minwidth Keyword Interaction

The minwidth, minlength, and minarea keywords define the minimum size of SRAFs. Width and length always refer to the length of the shorter and longer edges of rectangular SRAFs, respectively, regardless of SRAF orientation; width is the distance between the two long edges and length is the distance between the two short edges in a rectangular shape.

If any two of the keyword values are specified but not the third, the value of the third is not derived, and must be explicitly defined. Siemens EDA recommends that all these keyword values be specified. In addition, all values and priorities of these keywords are printed to the session transcript, providing useful information for debugging purposes.

If the specified values of any keyword are incompatible with each other, Calibre nmSRAF issues a warning message and no values are adjusted. Additionally, a minarea value is not assumed from the product of minwidth and minlength if no minarea specification is made. Incompatible keyword values may include:

minlength < *minwidth* or *minlength* * *minwidth* > *minarea*

A warning is issued if width or length are specified for a template and are incompatible with the specified keywords. For example, where:

width < *minwidth* or *length* < *minlength* or *width* * *length* < *minarea*

No corrective measures are taken in these cases, but the template does not generate any valid shapes.

Additionally, the dynamicsizing, sbwidth, and sraf keywords define minimum constraints for dynamic sizing and conflict resolution operations. They do not define constraints implemented by minarea, minlength and minwidth and they do not override those keyword values. However, during parsing they are checked against the specified min... keywords. If they are found incompatible, a warning is given and the values of dynamicsizing, sbwidth, and sraf are increased to comply with min... keyword constraints.

If min* keyword constraints for square SRAFs are specified, the sraf keyword that defines the size of square SRAFs inserted into certain conflict regions is checked against the square SRAF-related keyword values only. The sraf keyword is not checked against minarea, minlength and minwidth keyword values.

If minarea, minlength, and minwidth are not specified, the dynamicsizing, sbwidth, and sraf keywords and the specified width and length values of the templates are checked to define the appropriate limits for the SRAF sizes. For each priority, the minimum of any of the specified keywords dynamicsizing, sbwidth, and sraf and the nominal SRAF width and length is used as the minwidth constraint. Calibre nmSRAF then issues an informational message stating the actual constraints it uses.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

SRAFs do not change orthogonal orientation if the square SRAF-related keywords are specified. If separate square keywords are not specified, minwidth and minarea are used as constraints, and square SRAFs may become rectangular in shape and are placed at either orientation during cleanup operations.

Use the sraf_options Statement

Rules should be formatted within an sraf_options statement. This insures developers free-format whitespace for comments and easier template and keyword manipulation than using the setlayer command alone.

Layer Handling

Most Calibre nmSRAF templates should require only one setlayer statement. Additional setlayer statements do not cause additional runtime, unlike additional templates within a previous setlayer. However, additional setlayers only process the output of previously-generated SRAFs.

In general, no input layers may be used with the same name more than once. If the same layer is required, it must be derived and output under a different name prior to its subsequent use.

You must specify the targetlayer first in the argument list to RET MBSRAF. You must specify only one targetlayer. If the targetlayer is required in other areas of the recipe, before you process it you must derive it again under a different name.

Using the offsetlayer can help manage SRAF placement. Never use the target layer as the offsetlayer. Additionally, if sblayer or offsetlayer is specified, they must appear in either the sraf_options statement or the setlayer command, but never both.

Siemens EDA recommends using the correctionlayer for best SRAF-generating performance. The correctionlayer is a polygon-based layer acting as a corrected target layer and is used to calculate the gradient. Only one correctionlayer may be specified.

Debug Layers

Siemens EDA recommends you enable debug layers so you can see what happens in each step of the SRAF generation process. If SRAFs are not placed where you expect them, it may be because the minoffset or minfeaturespace distance checks are being violated. You can examine the field debug layer for any affected templates.

Note

Siemens EDA recommends never using the output of debug layers for SVRF-related post-processing. The only purpose for using the cnsraf debug capability is for debugging SRAF-generated intermediate layers within small regions. A high possibility exists that generated debug layers may be missing or not MRC clean near tile boundaries. Therefore, Siemens EDA does not guarantee clean generation of debug-specific SRAF layers for large designs or hierarchical runs.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Multiple Mapping of Debug Layers

Multiple mapping and output of debug layers within Calibre nmSRAF is not permitted as of Calibre release 2019.2. A compile-time error message is generated if multiple mapping is attempted. The following code fails and generates the error message shown:

```
setlayer sraf = cnsraf target OPTIONS all
setlayer p10_iso = cnsraf debug sraf isolate 10
setlayer p10_iso_2 = cnsraf debug sraf isolate 10
p10_iso = RET SBAR target FILE sraf_gen MAP p10_iso
p10_iso_2 = RET SBAR target FILE sraf_gen MAP p10_iso_2

Trying to use the debug layer "isolate" for priority 10 in more than one
setlayer command
```

Tags, Templates and Conflict Resolution

Upper bounds for the newtag internal keyword are critical:

- When using the internal keyword to generate tags used in the context of rectedge center strictcenter template, you must provide an upper bound to the internal constraint to enable limits during edge collection.
- When applying the internal keyword to any pitch-based template, you must provide an upper bound to the constraint to enable limits during edge collection.
- As a default, the promotion distance is used for the upper bound.

Templates using the conflictonly keyword are generally placed at a higher priority than other templates.

When defining multiple tags in the setup file, Siemens EDA recommends that no gaps are left between the individual tagging bins. This ensures that all the target edges in the design are accounted for.

SRAF interaction with optics are localized effects and effective control requires localized movement. If SRAF fragmentation is not allowed during co-optimization of OPC, conflict resolution shapes should be restricted in size.

Match MRC and OPC-MRC Specifications

The values assigned to options minoffset, minfeaturespace, minwidth, sbwidth, and sraf within the Calibre nmSRAF LITHO FILE should be set in accordance with the final MRC values defined for SRAFs in the Calibre nmOPC setup file. If the SRAFs input to nmOPC are not MRC clean, nmOPC cannot guarantee MRC clean SRAF output after the sraf_print_avoidance operation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Disable SMT

Simultaneous multi-threading (SMT) treats each physical core as two logical CPU cores (one physical and one virtual). The processor causes the operating system to behave as if there are twice as many CPU cores. This halves the available memory per core.

To turn off SMT, set the environment variable CALIBRE_MT_SMT_FACTOR to 0.

For detailed information regarding disabling SMT, see the Calibre Local Printability Enhancement User's and Reference Manual.

AVX and SOCS Recommendations

Use SOCS shared memory for better memory management during SRAF generation. As of Calibre release 2016.4, processes running on the same physical host can share the directory that contains SOCS calculations. To enable SOCS memory management, you set LITHO_SOCS_KERNELS_SHARED to 1 in your environment.

Use AVX for performance benefits, enabled at the command line.

Caution

Both recipe development and production execution must be performed using identical AVX and non-AVX environments to eliminate potential SRAF XOR differences resulting from varied gradient values between these respective environments.

For usage information regarding both AVX and SOCS, see the Calibre Post-Tapeout Flow User's Manual.

Best Practices for cnSRAF and mbSRAF

Table 5-1. Calibre nmSRAF Best Practices

Best Practice
2019.4
When passing tags in to the newtag command, Siemens EDA recommends always applying an upper bound to the internal constraint. **Reason:** For predictable output of SRAFs using tags.
2019.2
Siemens EDA recommends always using minareaextension and minlengthextension with 45-degree SRAFs. **Reason:** For comprehensive control of 45-degree SRAF extensions.
2019.2

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-1. Calibre nmSRAF Best Practices (cont.)

Best Practice
Siemens EDA recommends always specifying cleanupversion 1. The rectedge center and extension45, centermerge, intrapriorityalign, interpriorityalign, lineendmerge no, lineendmerge 0, and intersection LTP keywords are all incompatible with cleanupversion 0, and no new features will be compatible with cleanupversion 0 going forward. **Reason:** For predictable, high quality SRAF generation.
2019.2
Siemens EDA recommends never using the output of debug layers for SVRF-related post-processing. The only purpose for using the cnsraf debug capability is for debugging SRAF-generated intermediate layers within small regions. A high possibility exists that generated debug layers may be missing or not MRC clean near tile boundaries. Therefore, Siemens EDA does not guarantee clean generation of debug-specific SRAF layers for large designs or hierarchical runs. **Reason:** Prevents perpetuation of unnecessary errors from improperly generated debugging layers.
2018.2
Siemens EDA recommends using minoffset, minfeaturespace, projecting and lineend keywords for global MRCs. **Reason:** Provides for predictable SRAF generation.
2018.2
Siemens EDA recommends limiting conditional MRC keywords such as lineend2long, sqedge2long, and sqcorner2corner to be used by priority only or where required. **Reason:** To avoid increased runtime due to additional measurement and cleanup with use of these extra MRC keywords.
2018.2
Siemens EDA recommends setting lineendmerge no and lineendmerge 0 by priority. **Reason:** Enables use of interpriorityalign and intrapriorityalign globally.
2018.2
Siemens EDA recommends using strictcenter for rectedge-center SRAF generation. **Reason:** Two tagged target edges provide predictable SRAF generation.
2018.1

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-1. Calibre nmSRAF Best Practices (cont.)

Best Practice
Siemens EDA recommends always using cleanupversion 1 for all new cnSRAF recipes. **Reason:** For predictable, high quality SRAF generation.
2018.1
Siemens EDA recommends always using algorithm 1 for all new mbSRAF recipes. **Reason:** For predictable, high quality SRAF generation.
2017.4
Siemens EDA recommends using dynamicsizingrunlength 0 for line layers when invoking cleanupversion 1. **Reason:** For predictable, high quality SRAF generation.
2017.4
Siemens EDA recommends using reducedbarsplit 0 for line layers when invoking cleanupversion 1. **Reason:** For predictable, high quality SRAF generation.
2017.3
Siemens EDA recommends always explicitly specifying minlength, minwidth and minarea in Calibre nmSRAF recipes. **Reason:** For predictable, high quality SRAF generation.
2017.3
Specify dynamicsizing values to be greater than or equal to minlength. If they are not specified as such, a warning message is issued. **Reason:** To maintain syntactically dependable recipes.
2017.3
Specify sbwidth values to be greater than or equal to minwidth. If they are not specified as such, a warning message is issued. **Reason:** To maintain syntactically dependable recipes.
2017.3
Specify sraf values to be greater than or equal to minsquarelength or the square root of minsquarearea. If they are not specified as such, a warning message is issued. **Reason:** To maintain syntactically dependable recipes.
2017.2

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-1. Calibre nmSRAF Best Practices (cont.)

Best Practice
For EUV-specific nodes, Siemens EDA recommends using maxoffset instead of multiple localmaxgradient templates to control the number of SRAF rings. **Reason:** For high quality SRAF generation at EUV nodes.
2016.4
It is imperative that the Advanced Vector Extensions (AVX) environment is identical for recipe development and production runs; AVX environmental differences may cause slight variations in gradient maps which could result in final SRAF differences. **Reason:** For consistent SRAF generation between recipe development and production treatment.
2016.4
The interprioritiyalign and intraprioritiyalign keywords require you to set cleanupversion 1. **Reason:** For improved SRAF cleanup.
2016.3
Siemens EDA recommends using the algorithm 1 keyword as a part of all Calibre mbSRAF recipes. **Reason:** For improved SRAF consistency of generated SRAFs.
2016.1
All layers input to Calibre nmSRAF recipes must be unique. When attempting to reuse any input layer as originally named in subsequent operations within a Calibre nmSRAF recipe, the layer name must be changed through a second derivation. This best practice affects input layers for both cnsraf and mbsraf: target layer, offsetlayer, sblayer and correctionlayer. **Reason:** Failure to re-derive the input layer may potentially result in erratic results.
2015.4
A comprehensive table has been added to the Best Practices chapter of the Calibre nmSRAF User's and Reference Manual outlining defaults, recommendation and user notes for important keywords. **Reason:** Provides users with better control over Calibre nmSRAF for generating optimal SRAFs.
2014.4

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-1. Calibre nmSRAF Best Practices (cont.)

Best Practice
Specify cleanupversion keyword with a value of 1. **Reason:** Provides for more precise MRC enforcement and better support for line layers.
2014.3
Specify lineendmerge keyword with a value greater than minoffset when using in conjunction with intersection J and intersectionnegative J. **Reason:** Provides for more continuous SRAF.
Specify minwidth. **Reason:** Insures no SRAFs below MRC values are inadvertently placed.
2014.2
Use cleanmode postsizing. **Reason:** Provides consistent SRAF cleanup.
Use the Lithofile model format instead of individual calls to background, optical_model_load and resist_model_load. **Reason:** Provides better model organization, and reduced specification set in the LITHO FILE statement and sraf_options block.
2014.1
Use mbSRAF to fill SRAF areas instead of the family of fillmaxoffset keywords. **Reason:** fillmaxoffset keywords have been deprecated.
2013.1
Use correctionLayer for mbSRAF **Reason:** Provides better performing scattering bars.
rectedge space-based option **Reason:** Provides proficient spaced-based scattering bars.
2012.3
Template indexing capability added. **Reason:** Facilitates new template additions.
Match cnsraf specifications with OPC MRC specification. **Reason:** Ensures MRC-clean SRAF output after SRAF print avoidance (SPA).
Siemens EDA recommends use of srafringhv template. **Reason:** Template simplifies the recipe.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-1. Calibre nmSRAF Best Practices (cont.)

Best Practice
Siemens EDA recommends use of GUI-based Litho File Tool to generate debug layer information. **Reason:** Simplifies and speeds SRAF debugging.
General tips for mbSRAF recipe development and control. **Reason:** Simplifies and speeds recipe development and improves results of SRAF output.
2012.2
As of release 2012.2, Siemens EDA recommends the use of cnsraf_fill instead of RET SRAF_FILL. **Reason:** Provides optimal SRAF.
A table of technology node, hardware requirements, and the respective tilemicrons settings was added. **Reason:** Optimizes implementation.

Related Topics

sraf_options

Calibre mbSRAF Recipe Development

Application of best practices aid in recipe development.

Role of Input Layers During Gradient Calculation

These input layers affect gradient map generation for mbsraf:

- correctionlayer
- offsetlayer
- sblayer
- targetlayer

Although taglayer and prioritylayer are input layers, their role is limited to SRAF candidate selection, after seed pixel generation is complete, using the computed gradient. Therefore, these input layers are not shown in this list. Various input layers specified to mbSRAF can be classified as one of three types:

- **Gradient —** This layer is used for determining pixel evaluation points for computing the gradient. Currently, only the targetlayer is used for deciding pixel evaluation points for computing the gradient.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- **Light —** This layer is used to inform where the light passes through. The gradient computer uses this layer to compute the gradient at each pixel evaluation point. It is not used during seed pixel generation, candidate generation, or MRC checks. If the correctionlayer and sblayer are defined, they are used to determine where light passes through the mask. If not, the targetlayer is used.
- **MRC —** This layer is used to perform enforcement of MRC checks and define keepout areas. The targetlayer, sblayer, and offsetlayer are all used during the MRC cleanup process. The correctionlayer is not used during MRC checking. Its role is limited only to the gradient calculation.

The target layer is a required layer because the shapes provided on the target layer define the evaluation points for the gradient calculation. If a portion of the original target layer is being processed in mbSRAF (typically using mbSRAF as SRAF fill after cnsraf rules-based insertion), the selected region of the target layer must be large enough to provide an empiric topological environment for any subset of target shapes in order for the gradient to be representationally accurate. If a shape is artificially isolated, the gradient can be erroneous due to inadequate sampling of the empiric target environment.

The sblayer and correctionlayer are layers containing shapes that are simulated to provide the image from which the gradient is calculated. If neither of these layers are available, the target layer is used for the image simulation.

When using mbSRAF as fill or an SRAF safety net after rules-based insertion, all layers processed must contain a large enough region centered on the selected shape needing further lithographic validation through an optical diameter, but the generated SRAFs produced by mbSRAF must only be constrained to a smaller halo around the center target (using a value equal to or smaller than maxoffset).

Using algorithm 1

Added in Calibre release 2016.3, algorithm 1 significantly enhances geometric consistency during SRAF generation. Siemens EDA recommends always enabling algorithm 1 within all Calibre mbSRAF recipes. Enabling algorithm 1 requires these keyword settings:

- The minfeaturespace keyword must be explicitly set.
- The minoffset keyword must be explicitly set.
- Because algorithm 1 provides optimal SRAF generation through internal settings, Siemens EDA recommends that none of these keywords be modified:
  - cornerpref
  - cornerprefnegative
  - mode
  - peaktype

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Calibre mbSRAF Settings

Without OPC and Calibre OPCverify scripts, assessing Calibre mbSRAF optimization results do not provide clear indicators of successful optimization. The Calibre OPCverify script you use must contain figures of merit that SRAFs are intended to address such as Depth of Focus (DOF) or PVband width.

Table 5-2. Calibre mbSRAF Recommended Optimization Settings

Command	Default	Recommended	Notes
maxmove	width/2 for algorithm 0, 1 nm for algorithm 1	For finite control of SRAF movement during cleanup	As of 2018.1, the previous default setting of maxmove 0 has changed, and can now be explicitly specified. However, recipes previously using algorithm 1 must now explicitly specify maxmove 0 to obtain identical SRAF results prior to 2018.1..
jogignore	Disabled	Use with variable etch bias (VEB) models	Siemens EDA recommends always using jogignore with mode collectno45 to achieve expected results.
algorithm	Required	algorithm 1	Siemens EDA recommends using the algorithm 1 keyword within all Calibre mbSRAF recipes for improved geometric consistency during SRAF generation. The minfeaturespace parameter must be explicitly set. Additionally, algorithm 1 sets peaktype internally, and sets cornerpref to no; Siemens EDA recommends against overriding these internally-set values.
target	None	Litho target	Target should be based on what the litho processing step attempts to achieve and not the etch target.
correctionlayer	None	Defined, pass a single layer	The correctionlayer shapes should be 80% to 120% of the area of their respective target shapes at the defined process conditions. If multiple layers combine to serve as a correctionlayer, they should be passed as a single layer to mbsraf after using an SVRF logical OR operation to combine them.
sblayer	None	Required if they exist	When combining rules-based with model-based SRAF, you must provide the rules-based as optical elements. If positive and negative SRAF exist, you must pass them separately in two different layers. Additionally, when sblayer is specified, it must appear in either the options block or the setlayer command, but never both.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-2. Calibre mbSRAF Recommended Optimization Settings (cont.)

Command	Default	Recommended	Notes
offsetlayer	None	Pass a single layer	Offsetlayer should never be a copy of the target layer or correctionlayer. It should be an additional, single layer for which SRAFs must honor spacing restrictions. If multiple layers need to be merged for this layer, it is best to OR them using an SVRF operation prior to running mbsraf. **Note:** When offsetlayer is specified, it must appear in either the options block or the setlayer command, but never both.
Model specification	None	Litho model format to insure proper specifications	Organized models prevent entry errors, especially for layer modifications.
minfeaturespace	None	minCD/2	The tool enforces a distance based on a fraction of the optical pitch if the specified value is smaller than the calculated value. For recipes using algorithm 1, however, no such internally calculated value is used; only explicitly-specified values are enforced.
minoffset	None	minCD/2, but optimize	More SRAFs are inserted when minoffset is smaller, and the risk of SRAF printing increases when minoffset is smaller.
minoffsetlayerspace	0	Desired offset from layer shapes	Only needed when offset layers are used, and a non-zero separation between SRAF and offsetlayer is desired.
gradientshift	None	Larger than reported need in transcript	When negative values are near target they can lead to smaller SRAFs. The transcript reports the value immediately adjacent to the target. An application note (MG585966) is available on Support Center, entitled "*Identifying the gradientShift value to use in mbSRAF*".
width	None	minCD/4 to minCD/2	The maximum width that does not print.
squarewidth	None	Up to 2x width	The maximum square that does not print.
minarea/ minsquarearea/ minwidth/ minlength/ minsquarelength	None	Defined	Explicit specification provides optimal results.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-2. Calibre mbSRAF Recommended Optimization Settings (cont.)

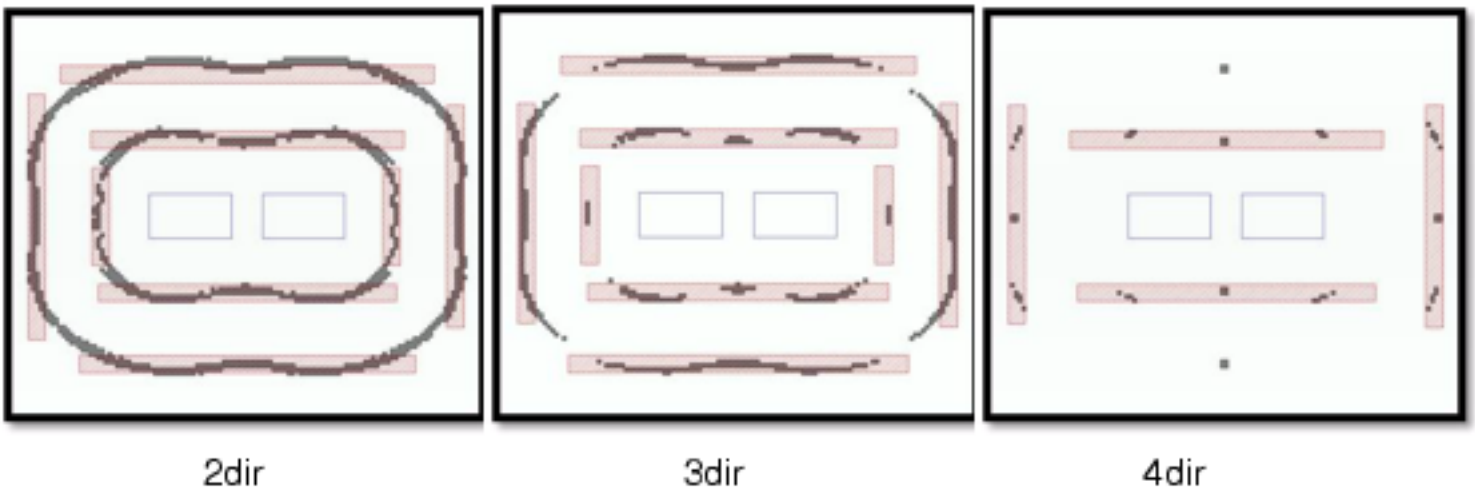
Command	Default	Recommended	Notes
peaktype	3dir for algorithm 0, internally set for algorithm 1	Use algorithm 1, internally set	Lower numbers (1,2)dir may lead to more seed pixels and candidates. Use of (1,2,3)peak should be limited to processes which exhibit saddle-points in the gradient on some layouts. peaktype behavior is demonstrated in Figure 5-2.
mode	nearestno45 for algorithm 0, internally set for algorithm 1	Use algorithm 1, internally set	Collectno45 creates more SRAFs not immediately adjacent to the target, which is needed in some processes; nearestno45 limits SRAF size to the target plus extension.
pw_conditions	None	Conditions that match the degraded image in Calibre OPCverify simulations	Additional conditions add runtime, but they can help correct imaging failures identified in verification.
cornerpref	square for algorithm 0, no for algorithm 1	Use algorithm 1, cornerpref set to no	Some processes are dependent on corner SRAFs while others show no benefit (where cornerpref no lowers mask costs).
cornerprefnegative	square for algorithm 0, no for algorithm 1	Use algorithm 1, cornerprefnegative set to no	Some processes are dependent on corner SRAFs while others show no benefit (where cornerprefnegative no lowers mask costs).
prioritylayer	None	Use for specific layouts	Small shapes orthogonal to large shapes benefit the most (for example, M1 signal lines in the triple-patterning process).
maxlength	None	Use as advised	Results at tile boundaries may vary, but this is a cost effective way to reduce SRAF printing. If printing is a minor concern, Siemens EDA recommends not using this feature.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-2. Calibre mbSRAF Recommended Optimization Settings (cont.)

Command	Default	Recommended	Notes
maxoffset	None	Specify a value to place enough of the required SRAF rows to benefit the process window	Use Calibre OPCverify results to help define how far SRAFs should be placed from any target (which affects the number of rows of SRAFs). For certain layers, such as metal, do not specify maxoffset. Instead, control the number of rows by multiple localmaxgradient templates that permit control of SRAF widths per row. The combination of multiple localmaxgradient templates with maxoffset is not recommended.
intersection J	None	Use as advised	Allowing jogs provides more continuous SRAFs, but sraf_print_avoidance in Calibre nmOPC is unable to correct any printing induced at jogs because of control fragment misalignment.

Figure 5-2. SRAF Coverage Varies by peaktype Mode



Calibre mbSRAF Optimization Steps

The foundation of Calibre mbSRAF is the gradient calculation for a process condition that defines SRAF placement locations. In order for the gradient to be as robust as the process can support, it is important that the print image at the desired process conditions roughly matches the target shape dimensions. One empirically consistent rule used for many customer designs and processes is the area of image should be 80% to 120% of the target area for any given process condition. A correction layer derived from a SIZE operation or from a few iterations of OPC may be required to achieve images of this range.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

The next value to specify for proper gradient-to-SRAF conversion is gradientshift. The optimal amount can best be measured on an isolated target. An application note is available on Support Center, entitled *Application note MG585866* which details methods for choosing this setting. The gradientshift keyword may be required in resist processes using negative tone development (NTD) where dark areas in a bright mask are the desired target locations.

When the gradient is largely shifted to a positive range, the SRAFs generated from seed pixels will be larger, and therefore, more effective in image correction. The number of SRAFs is closely related to the prevalence of the seed pixels, and the number of seed pixels can be increased by reducing the number of directions a local maxima must satisfy. Siemens EDA recommends that you measure the effectiveness of SRAF with the default 3dir keyword before modifying it. Use the verification script to help determine if more or less SRAFs are needed. Both the seedpixels and candidates debug layers are useful for analysis of SRAFs that are placed.

Once the spatial frequency and density of SRAFs seems reasonably effective, optimize the values of width and squarewidth to maximize positive image benefit while maintaining a low level of SRAF printing. The largest process window advantage comes from leaving some limited SRAF printing at the insertion stage and relying on sraf_print_avoidance in OPC to resolve this printing. A more conservative choice would be to keep the SRAF size smaller, but it is still possible for OPC edge movement to combine with these smaller SRAFs to induce printing, so sraf_print_avoidance is not optional for a production solution.

In combination with some experimental runs yielding varying SRAF width, this is the appropriate time in the recipe development to experiment with the mode. As a rough rule, collectno45, which bases SRAF length on the gradient peaks rather than target size, has proven effective on both island layers (generally contacts) as well as line layers (generally metal). However, as the relative shape of layers continue to change as additional interconnect solutions are adopted, there is no way to state that either mode will work better without experimentation.

In more classical line layer topologies, the square SRAFs at line ends may be unnecessary if line end pullback requirements are not very strict. The corner SRAFs help significantly with formation of 2D images, which is why cornerpref square is default. For many illuminators, the square SRAFs are so important that the squareonly option is required so that square SRAFs are placed at a higher priority than edge-oriented SRAFs.

Specific case solutions

The following solutions are provided as additional guidance for specific layer and illuminator scenarios.

- **Line layers —** As mentioned, square SRAFs may be unnecessary, depending on the image requirement specifications. In addition, the restriction of a single width for each localmaxgradient can diminish the achievable process window. The solution is to not specify maxoffset (maximum distance from targets that SRAFs are placed) and instead have additional localmaxgradient templates, which have (generally) increasing SRAF widths (and square widths if desired). The benefit of the larger SRAFs at the larger

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

distance to targets is more image influence, while the separation from OPC movement reduces the chance of SRAF printing (still controlled in OPC by sraf_print_avoidance).

- **Layers of mixed size targets in close proximity —** Several layers include a mix of smaller and larger shapes in close proximity to each other. An example would be the smaller signal lines and larger power lines in a standard cell metal layer. For these environments, the gradient typically provides a grid of seed pixels that can be transformed into SRAFs which are parallel to either shape. If small shapes are marked as members of the prioritylayer, the SRAF output provides better support of small images at a modest expense of large shape imaging. Because the tolerance of the larger shapes to image imperfections is typically larger, this allows a general method of directing SRAFs to focus the benefit on the smaller target shapes.
- **Inadequate process window after optimization —** In the majority of layouts and processes, simulation with a single optical condition (best focus or defocus) provides adequate SRAF insertion. However, at times the competing forces on proper image placement may require that the gradient consider more than one condition. Using Calibre OPCverify conditions for process windows, identify the conditions that lead to image failure. If these (or slightly more off-nominal) conditions are provided to the gradient calculations, the placement of SRAFs will shift and usually resolve the insufficient solution. At times, the weight of these pw_conditions may need to be in the 10-100 range, so that they more fully dominate the gradient calculation. Of course, if a single condition is needed to achieve the process window that condition can be provided as the nominal model (defined by the mbsraf optical command).
- **Negative SRAF insertion —** In large target shapes, which are near tight spaces, imaging can be improved by inserting negative SRAFs, which are SRAFs that remove material inside the target (typically generating slots) in the vicinity of a tight space, increasing the amount of light that can form the outer edges. To enable negative SRAF generation with Calibre mbSRAF, several steps are required.

  a. Define a localmaxgradient template with a unique prior value. In the associated priority definition, specify the keyword negative along with any other priority level controls desired (for example minfeaturespace).

  b. Define styles specific to negative SRAF such as maxoffsetnegative, cornerprefnegative, and intersectionnegative, which refine negative SRAFs you generate.

  c. Introduce negative SRAFs into all shapes large enough for MRC-legal insertion. While it is mathematically valid that inserting negative SRAFs on isolated shapes improves many process variation artifacts, including PVbands, many customers desire to only have negative SRAFs inserted near tight spaces where they have the most significant imaging benefit. To control SRAF placement in these spaces, use a tag (from taglayer or using a tag script) to define the space edges in the localmaxgradient template used for negative SRAF. If the space where this is needed is unknown, it is probably better to choose a much larger space than suspected, and

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

to adjust it tighter as the process matures. Again, the critical imaging metrics should be the guide for the definition of spaces that have restricted access.

Miscellaneous Recipe Development Aids

Using the available Calibre mbSRAF keywords and constraints properly speed recipe development.

1. Start by defining the MRC constraints to match those used by OPC.
2. Use the mode keyword to define required SRAF length controlled by target length or length controlled by gradient.
3. Optimize width and squarewidth values based on process window printing.

Lithomodel Format Usage for Calibre mbSRAF Execution

Siemens EDA recommends using litho model format in the development and execution of Calibre mbSRAF because:

- The litho model format provides a more organized repository for model data.
- The litho model format is becoming more widely used for many Siemens EDA litho modeling tools.

The litho model format has several operational constituents:

- **image_options —** Specifies the image_options block name.
- **litho_model —** Specifies the directory containing the *Lithomodel* data.
- ***Lithomodel* file —** References optical and resist models contained in the directory specified by litho_model.

In this *Lithomodel* file, a DDM model is specified:

```
version 1
mask 0 {
   optical optical_0
   optical optical_-40 focus -40nm
   optical optical_+40 focus 40nm
   background 1 0
   mask_layer 0 TRANS -.9785 -.0085 DDM omog.dark.quad.ddm
   mask_layer 1 TRANS -.9785 -.0085 DDM omog.dark.quad.ddm
   dose 1
}
resist resist.mod
image_threshold 0.246
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

This *Lithomodel* file specifies a non-DDM model:

```
version 1
mask 0 {
   optical opt
   background atten 0.06
   mask_layer 0 TRANS clear CATEGORY visible
}
resist ctr.mod
```

This example presents a complete LITHO FILE statement, implementing the litho model format, for generating SRAF with Calibre mbSRAF:

```
LITHO FILE mbsraf_setup [
   modelpath ./models
   layer M1
   layer M1_sraf
   layer M1_OPC
   tilemicrons 25
   progress_meter on


   image_options LMO {
      lithomodel MX1
      layer M1 visible mask_layer 1
      layer M1_sraf visible mask_layer 2
      layer M1_OPC visible mask_layer 406
   }

   sraf_options opts {
      algorithm 1
      sblayer M1_sraf
      correctionlayer M1_OPC
      image_options LMO
      minfeaturespace 0.045 minoffset 0.025 maxoffset 0.220
      minwidth 0.015 minlength 0.030
      minsquarelength 0.022 minsquarearea 0.000529
      gradientshift 0
      localmaxgradient prior 1 width 0.02 squarewidth 0.03
   }

   setlayer mb_sraf = mbsraf M1 IMAGE_OPTIONS LMO OPTIONS opts
   setlayer seedpixels = mbsraf debug mb_sraf seedpixels 1
]

mb_sraf = RET mbsraf M1 M1_sraf M1_OPC FILE mbsraf_setup MAP mb_sraf
```

For detailed information regarding the *Lithomodel* file and litho model format, see the *Calibre nmOPC User's and Reference Manual.*

correctionlayer Usage for Calibre mbSRAF

The correctionlayer keyword is used during simulation to define the initial image that forms the basis of the gradient calculation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Siemens EDA recommends using the correctionlayer argument for better performing SRAF, especially with bright field masks. The correctionlayer for mbsraf can be a sized-up target layer or an OPC layer (a layer that has been resized by several iterations of nmOPC). The image generated using the correctionlayer should be within plus or minus 20 percent of the expected nominal contour image.

Priorities for Calibre mbSRAF Tagging

Priorities are vital for the segregation of SRAFs generated by different templates. Priorities also apply to tag different groups, and ensure that minfeaturespace and minoffset enforcement for these various tag groups do not conflict. Additionally, localmaxgradient templates with different tags and the same priority generate Calibre warning messages.

Example

This code sample:

```
localmaxgradient tag mytag1 width 0.018 squarewidth 0.025 prior 1
localmaxgradient tag mytag2 width 0.018 squarewidth 0.025 prior 1
```

produces this warning in the transcript:

/-- WARN: nmSraf: Priority 1, templates 0 and 1 do not have the same tags. Check tags or move one template to a different priority.

Setting different priorities for each template with differing tags parses correctly:

```
localmaxgradient tag mytag1 width 0.018 squarewidth 0.025 prior 1
localmaxgradient tag mytag2 width 0.018 squarewidth 0.025 prior 2
```

Calibre cnSRAF Template Development Tips

Careful setup during cnsraf template development insures optimal SRAF generation and best image fidelity.

Template Indexing

Using an indexing method with the priority keyword in conjunction with the prior keyword of any template facilitates the addition of templates during the development of your command file. For example, when priorities numbered in tens are first assigned you permit more development flexibility by retaining nine additional spots to add subsequent templates without modifying past priority values. Figure 5-3 demonstrates this capability.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 5-3. Indexed Templates by Priority

```
setlayer CN_SRAF = cnsraf sraf_target \
                   newTag      sqr             rectA <= 0.060 rectB >= 0.060 \
                   newTag      short_end       length > 0.06 <= 0.090 \
                   newTag      med_end         length > 0.09 <= 1.000 \
rectEdge        tag med_end     extension  0.035 width 0.017 offset 0.075 conflictOnly                  prior 10
rectEdge        tag med_end     extension  0.035 width 0.017 offset 0.075 dynamicSizing 0.030           prior 20 \
rectCorner      tag short_end   offsetX -0.015 offsetY 0.065 width 0.038 length 0.038 conflictOnly      prior 30 \
priority 10 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250 centerthresh 0.080  sbwidth 0.018 sraf 0.038 \
priority 20 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250 centerthresh 0.080  sbwidth 0.017 sraf 0.038 \
priority 30 minoffset 0.030 minfeaturespace 0.040 maxconflict 0.250 centerthresh 0.080  sbwidth 0.017 sraf 0.038
```

Substituting recthv and rectdiag for srafringhv

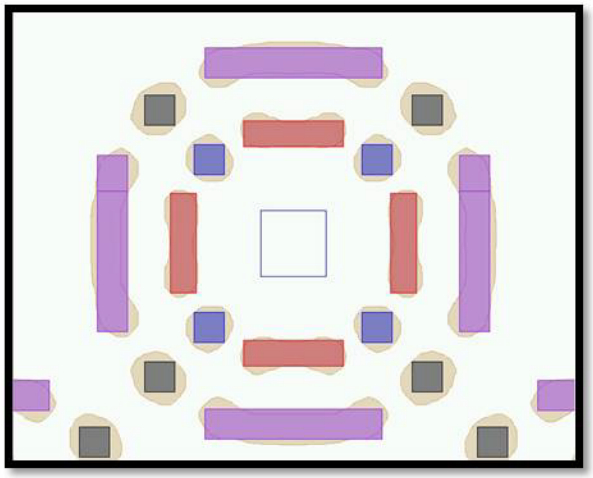
The following code snippets represent the coding efficiency of using srafringhv in place of both recthv and rectdiag. As the accompanying illustrations show, results are nearly identical.

Example 5-4 uses recthv and rectdiag only. The code colors represent the SRAF generated by that specific line of code shown in Figure 5-5.

Figure 5-4. SRAF Generation With recthv and rectdiag

```
setlayer sraf_out = \
   cnsraf sraf_target newtag e1 recta <= 0.076 rectb < 0.1 \
   recthv tag e1 pitchx 0.130 pitchy 0.130 \
      width 0.030 length 0.116 prior 1 \
   rectdiag tag e1 pitchx 0.090 pitchy 0.090 \
      width 0.035 length 0.035 prior 2 \
   recthv tag e1 pitchx 0.210 pitchy 0.210 \
      width 0.035 length 0.216 prior 3 \
   rectdiag tag e1 pitchx 0.190 pitchy 0.190 \
      width 0.035 length 0.035 prior 4 \
```

Figure 5-5. SRAF Results With recthv and rectdiag



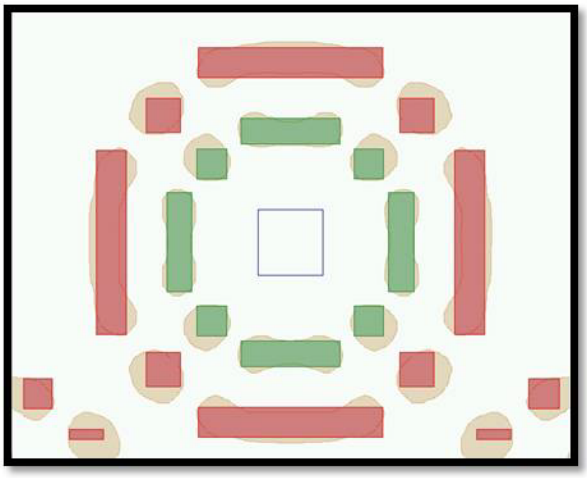
Example 5-6 uses srafringhv alone. The code colors match the SRAF colors generated by the respective lines of code, as shown in Figure 5-7.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure 5-6. SRAF Generation With srafringhv

```
setlayer ass_CN_0 =
   cnsraf sraf_target newtag e1 recta <= 0.076 rectb < 0.1 \
   srafringhv tag e1 width 0.035 pitch 0.130 srafcount 1 \
      rectlength 0.116 rectwidth 0.030 prior 1 \
   srafringhv tag e1 width 0.040 pitch 0.210 srafcount 1 \
      rectlength 0.216 rectwidth 0.035 prior 2 \
```

Figure 5-7. SRAF Results Using srafringhv Alone



Use RET Flow Tool for cnsraf Template Development

The RET Flow Tool greatly improves efficiency and ease-of-use for debugging and optimizing cnsraf templates. The RET Flow Tool interface has several parts: Textual extension and layer mapping window, Options forms with debug layer selection, and Calibre WORKbench layout editor and viewer.

You access the RET Flow Tool for applications such as OPC and SRAF generation tools from Calibre WORKbench.

For detailed information regarding the use of Calibre nmSRAF with RET Flow Tool, see “RET Flow Tool and Calibre nmSRAF” on page 360.

Recommended Settings for Best cnsraf Results

Recommended settings improve results produced by the cnsraf command.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-3. Recommended Settings for Best cnsraf Results

Option	Recommendation/Notes
minareaextension minlengthextension rectedge	For additional control of 45-degree SRAF extensions with the rectedge template, use minareaextension and minlengthextension keywords.
minoffset minoffsetlayerspace minfeaturespace	Conditional MRC parameters such as lineend2long, long2lineend, square2long, etc., must be specified by priority only for these keywords. Do not define conditional parameters globally. Although parsed as valid, global specification of these parameters degrades performance.
minoffset minoffsetlayerspace minfeaturespace	Siemens EDA recommends always setting cleanupversion 1 when using the square-oriented keyword options (long2sqedge, corner2sqcorner, etc.) for minoffset, minoffsetlayerspace, and minfeaturespace, or square2square keywords.
interpriorityalign intrapriorityalign	Siemens EDA recommends always setting interpriorityalign and intrapriorityalign per priority. If these keywords are set globally, a warning is issued to the transcript.
lineendmerge	Siemens EDA recommends using the lineendmerge keyword in conjunction with the intersection and intersectionnegative keywords to provide more continuous SRAF output. If no lineendmerge parameter is explicitly set, then lineendmerge is automatically set and its resulting value reported to the transcript. When lineendmerge is specified globally, line-ends of different priorities may be merged. Therefore, Siemens EDA recommends only specifying lineendmerge by priority.
algorithm	Siemens EDA recommends setting algorithm to 1 for new recipe development.
cleanupversion	Siemens EDA recommends setting cleanupversion 1 for new recipe development only, and always used in conjunction with cleanmode postsizing to avoid excessive field SRAF and consequent runtime impact during cleaning. Using cleanupversion 1 provides best results for line and space layers, and 2D structures. For existing recipes using Calibre versions prior to 2016.3, if cleanupversion 1 is specified, also specify cleanmode immediate globally. This setting provides more precise MRC enforcement, invokes promotion by default, and improves SRAF coverage for line layers and 2D structures.
minwidth	Siemens EDA recommends using the minwidth keyword to avoid the possibility of outputting SRAFs of a size less than the minimum MRC constraint.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-3. Recommended Settings for Best cnsraf Results (cont.)

Option	Recommendation/Notes
cleanmode	Siemens EDA recommends setting cleanupversion 1 and using cleanmode postsizing when specifying dynamicsizing. This allows SRAFs that are violating MRC constraints to be sized down instead of being immediately removed.
maxconflict	Setting maxconflict too large (greater than 0.5 um) leads to large interaction distances during hierarchical processing. A large maxconflict value also reduces control over the SRAF generated through conflict resolution and rarely provides the desired match to Calibre pxOPC. Siemens EDA recommends keeping the maxconflict setting in the template specification and create additional templates if necessary. Use the priority override feature to set maxconflict values for subsequent templates.
maxlength	Use this keyword as advised. Results at tile boundaries may vary, but this is a cost-effective way to reduce SRAF printing. If printing is of minor concern, Siemens EDA recommends not using this keyword.
minfeaturespace	Set this sufficiently large so that no primary or secondary SRAFs are created too close to the contacts. The primary minfeaturespace should be smaller than the secondary minfeaturespace to help enforce their respective placement with neighboring target shapes.
newtag external and internal, and newtag recta and rectb	Use the external keyword to isolate edges of specific spaces when using the rectedge template. For recta and rectb, use smaller values to avoid flattening data and producing inconsistent results.
rectedge	If you are using the rectedge template that generates long assist features, specify dynamicsizing within the template's priority to improve coverage. The rectedge template can generate SRAFs that are longer than the interaction distance (the SRAFs can also be longer than the tilesize). If dynamicsizing is not used, then a conflict at one end of the bar can cause the entire assist feature to be removed. Use dynamicsizing to allow SRAFs to be resized so they can be preserved. For more efficient rectedge template application, use the external newtag keyword to specify dimensional conditions for scattering bar generation.
tilemicrons	The selection of tile size is critical for optimum processing. If the tile size is too large or too small, primary or remote memory consumption may go into swap and slow performance. The optimum tile size settings found by Siemens EDA correlate to the technology node and balance the size of data per tile. It is recommended that you set tilemicrons to the values based on the node listed in Table 4-4.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Recommended Settings for Best mbsraf Results

Recommended settings improve the results produced by the mbsraf command.

Table 5-4. Recommended Settings for Best mbsraf Results

Option	Recommendation/Notes
litho_model	Only the litho model format is supported for EUV, including models and all related keywords.
maxmove	As of 2018.1, the previous default setting of maxmove 0 has changed, and can now be explicitly specified. However, recipes previously using algorithm 1 must now explicitly specify maxmove 0 to obtain identical SRAF results prior to 2018.1.
algorithm	Siemens EDA recommends always setting algorithm 1 for improved SRAF consistency of generated SRAFs.
interpriorityalign intrapriorityalign	Siemens EDA recommends always setting interpriorityalign and intrapriorityalign per priority. If these keywords are set globally, a warning is issued to the transcript.
gradientfactornegative	Siemens EDA recommends setting gradientfactornegative to -1 for single exposure simulations.
lineendmerge	Siemens EDA recommends using the lineendmerge keyword in conjunction with the intersection and intersectionnegative keywords to provide more continuous SRAF output. If no lineendmerge parameter is explicitly set, it is automatically set and its resulting value reported to the transcript. When lineendmerge is specified globally, line-ends of different priorities may be merged. Therefore, Siemens EDA recommends only specifying lineendmerge by priority.
minwidth	Siemens EDA recommends using the minwidth keyword to avoid the possibility of outputting SRAFs of a size less than the minimum MRC constraint.
correctionlayer	The correction layer can be a modified target layer (a layer that has been resized by a few iterations of Calibre nmOPC, for example). The modification provides enlarged shapes that lead to an improved gradient. The correctionlayer keyword is used during simulation to define the initial image that forms the basis of the gradient calculation. Siemens EDA recommends using correctionlayer so it leads to a simulated contour within plus or minus 20 percent of the target area.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-4. Recommended Settings for Best mbsraf Results (cont.)

Option	Recommendation/Notes
tilemicrons	The selection of tile size is critical for optimum processing. If the tile size is too large or too small, primary or remote memory consumption may go into swap and slow performance. The optimum tile size settings found by Siemens EDA correlate to the technology node and balance the size of data per tile. It is recommended that you set tilemicrons to the values based on the node listed in Table 4-4.
clone_transformed_cells	Siemens EDA recommends you set clone_transformed_cells to the decide option. The optical model is checked if cell cloning is activated. The decide option causes the Calibre engine to check if any asymmetric sources are present in the optical model. If so, Calibre clones cells that have rotated or mirrored placements so that hierarchical simulation of the rotated structures is correct.

Calibre nmSRAF Recipe LINT Messages

Understanding informational messages and correcting warning messages aid in improving results generated by Calibre nmSRAF.

During compilation and validation of the setlayer input, a series of LINT messages are generated. These include warnings. Some layout-dependent warnings can only be found at runtime and not during compile-time. In addition to warnings, a brief overview of cleanup parameters is provided. Some messages inform the user of specified or default parameters that are in compliance with Siemens EDA recommended settings. If a parameter deviates from Siemens EDA-recommended settings, the specified value is reported as a warning.

Note

Some specifications listed in Table 5-5 contain formatted output values of the parameters and do not appear the same in the log file generated at runtime. The messages shown are not listed alphabetically.

Table 5-5. Calibre nmSRAF LINT Messages

LINT Messages
nmSraf: global intrapriorityalign is %g. It is not being used. nmSraf: Priority %d CenterThresh 0. This is the recommended value.
nmSraf: global intrapriorityalign is %g. It will be applied to any priority that doesn't have intrapriorityalign specified. intrapriorityalign is not recommended to be used globally. nmSraf: global CenterThresh is %g. Recommended value is 0.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-5. Calibre nmSRAF LINT Messages (cont.)

LINT Messages
nmSraf: global maxConflict is %g. It is not being used. nmSraf: Priority %d CenterThresh %g. Recommended value is 0.\nIf needed, more templates should be added to avoid nonzero CenterThresh
nmSraf: global maxConflict is %g nmSraf: global CenterThresh is 0. This is the recommended value.
nmSraf: global maxConflict is %g. Recommended value in [0,0.5) nmSraf: global CenterThresh is %g. It is not being used.
nmSraf: Priority %d maxConflict %g. nmSraf: Priority %d cleanmode is %d (not found)
nmSraf: Priority %d maxConflict %g. Recommended value is <0.5. nmSraf: priority %d cleanmode is %s
nmSraf: global cleanmode is %s. Recommended setting: %s nmSraf: priority %d cleanmode is %s. Recommended setting: %s
nmSraf: global cleanmode is %d (not found) nmSraf: global cleanmode is %s
nmSraf: global interpriorityalign is %g. It will be applied to any priority that doesn't have interpriorityalign specified. interpriorityalign is not recommended to be used globally nmSraf: global interpriorityalign is %g. It is not being used.
nmSraf: Priority %d minFeatureSpace %g. nmSraf: global minFeatureSpace is %g
nmSraf: global minFeatureSpace is %g. It is not being used. nmSraf: Priority %d minWidth %g.
nmSraf: Priority %d minLength %g. nmSraf: Priority %d minArea %g.
nmSraf: Priority %d minSquareLength %g. nmSraf: Priority %d minSquareArea %g.
nmSraf: Priority %d minimum sraf size %g. nmSraf: Plane value %s cannot be translated to value.
nmSraf: Plane value %s translates to %g nmSraf: max optical radius: %gum
nmSraf:flare radius (short range only): %gum nmSraf:flare radius: %gum

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table 5-5. Calibre nmSRAF LINT Messages (cont.)

LINT Messages
nmSraf: \"tilemicrons\" has not been specified%s nmSraf: \"tilemicrons\" value is %gum%s
nmSraf: EUV simulation with zero flare radius nmSraf: \"tilemicrons\" value is %gum%s
nmSraf: clone_transformed_cells %s. Recommended \"decide\" nmSraf: clone_transformed_cells is %s
nmSraf: clone_transformed_cells %s. Recommended \"decide\" nmSraf: clone_transformed_cells is %s nmSraf: print mrc from lint
Node:>=45nm - 45...50 Node:32nm - 35...40 Node:22nm - 25...30 Node:14nm - 15...21 Node:10nm - 10...15 Node:7nm - 6.2...11.2

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Appendix A
SRAF Cleanup

Optimal placement of SRAFs require an iterative cleanup process during generation.

SRAF Cleanup 353

SRAF Cleanup

Calibre nmSRAF uses multiple steps to clean generated SRAFs for their optimal placement.

Calibre nmSRAF generates SRAFs within each tile of the input design. The tiles are then processed sequentially. For each tile, SRAFs are generated in a loop over all specified priorities from the lowest to highest priority number. For each priority specified by each template, field SRAFs are initially generated. These SRAFs follow the rules and tags specified within the templates. Because of the interrelationship of various priorities and rules, generated field SRAFs may violate MRCs (mask rule checks), or overlap other SRAFs or target shapes.

To resolve these violations, after field SRAFs are generated, cleanup is performed to remove MRC conflicts between SRAFs of the current priority and SRAFs of other priorities, and SRAF overlap of target shapes. All cleanup steps are performed for each priority. MRC parameters and cleanup options may be used to tune cleanup steps for individual priorities.

SRAF cleanup is ordered by priority, and all SRAFs retain their original priority throughout all cleanup phases. SRAFs are prioritized by specifying the ‘prior’ keyword in the generating template; a ‘prior’ value of 1 is the highest priority (greater values representing lower priorities), and these SRAFs take cleanup precedence. After SRAFs of highest priority are MRC-clean, the next lower-priority SRAFs are processed. Subsequently processed SRAFs of lower priorities respect all previously cleaned SRAFs of higher priorities. In this example, the template specifying ‘prior 2’ takes precedence, followed by ‘prior 3’, and lastly by ‘prior 4’:

```
rectedge tag tag2
   width 0.025 extension 0.01 offset 0.10 prior 2
rectcorner tag tag1
   width 0.03 length 0.03 offsetx 0.12 offsety 0.12 prior 3
rectedge tag tag1
   width 0.03 extension 0.01 offset 0.16 prior 4
```

After each tile has been processed, some SRAFs near tile boundaries may be hierarchically promoted to provide spatial context, allowing those SRAFs across boundaries to be cleaned and meet MRCs.

SRAF cleanup processes are detailed in the following table, consisting of cleanup steps, the respective impact of available cleanupversion settings and their associated debug layers that are

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

generated. Each cleanup step is shown in the order in which it is processed, and as a function of the cleanupversion parameter setting, 0 or 1. Each step generates their respective debug layers (when specifying the corresponding setlayer cnsraf debug command) shown in the last column. See the table footnote for step name explanations.

Table A-1. Cleanup Steps, cleanupversion Impact and Associated Debug Layers

Step Name	CV0	CV1	Debug Layer(s)
[1]INI	Always run.	Always run.	field, candidates, lineend, protected, extra2, maxoffset, tags, cornerrange, edgerange, initialfield, densefilter
[2]PRE	Always run.	Always run.	clipears, preclean
[3]IPA	Run if IPA parameters are specified.	Run if IPA parameters are specified.	align, ambiguous
[4]BSP	Run if dynamicsizing is specified.	Always run unless cleanmode immediate is specified.	barsplit, unsizablebarsplit
[5]LEM1	N/A	Always run.	lem1, lem1u
[6]CM0	Run if centermerge is specified.	Run if centermerge is specified or if dynamicsizingrunlength equals 0.	centermerge0raw, centermerge0
[7]SDS	Run if dynamicsizing is greater than 0.	Always run unless cleanmode immediate is specified.	dynamic, unsizabledynamic
[8]ADS	Run if dynamicsizing is greater than 0 and intersection is specified.	Always run.	asymmetric, unsizableasymmetric, process
[9]ADS1	Run if dynamicsizing is greater than 0.	Always run.	asymmetric1, unsizable_asymmetric1
[10]BSP1	Run if reducedbarsplit is equal to 3.	Run if reducedbarsplit is equal to 3.	barsplit1, unsizablebarsplit1
[11]CM	Run if centermerge is specified.	Always run.	centermergeraw, centermerge, process
[12]CR	Run if maxconflict is greater than 0.	Run if maxconflict is greater than 0.	unsizable, hardconflict, conflict, conflictbbox, conflictbar, resolution, conflictres, unsizableconflictres
[13]LEM	Always run.	Always run.	lem, lemu
[14]UBAR	N/A	Always run.	ubarsplitpostcr

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Table A-1. Cleanup Steps, cleanupversion Impact and Associated Debug Layers (cont.)

Step Name	CV0	CV1	Debug Layer(s)
[15]ISO	Always run, controlled by intersection and maxlength.	Always run, controlled by intersection and maxlength.	maxlength, intersection, unsizable_intersection, post_intersection, unsizable_post_intersection, final, oprotected, positive, negative, sraf_pn

1. INI - Initialization. SRAF candidates are generated.
2. PRE - Pre-clean. Removes duplicate SRAFs and SRAFs found within other SRAFs of the same priority.
3. IPA - Intra- and inter-priority alignment. Aligns SRAFs by width, then merges line-ends of same and different priorities, respectively.
4. BSP - Preliminary bar splitting. Removes initial conflicts between perpendicular SRAFs.
5. LEM1 - Preliminary line-end merging. Merges line-ends after preliminary bar splitting.
6. CM0 - Preliminary center merging. Generates an SRAF centered between a pair of parallel SRAFs still in need of cleanup.
7. SDS - Symmetric dynamic sizing. Removes conflicts in the length direction of SRAFs.
8. ADS - Asymmetric dynamic sizing. Reduces the length of regular SRAFs to make space to place unsizable SRAFs.
9. ADS1 - Final asymmetric dynamic sizing. Performs dynamic sizing for unsizable SRAFs
10. BSP1 - Final bar splitting. Bar splitting subsequent to the previous asymmetric dynamic sizing.
11. CM - Final center merge. Generates an SRAF centered between a pair of parallel SRAFs.
12. CR - Conflict resolution. Identifies conflict regions and inserts a single square or rectangular SRAF into the conflict region.
13. LEM - Final line-end merging. Merges all line-ends.
14. UBAR - Unsizable bar splitting. Final priority-based cleanup step; performs bar splitting for all unsizable SRAFs.
15. ISO - SRAF isolation. Outputs MRC-clean SRAFs for all priorities at the end of the cleanup process.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Appendix B
Running Calibre nmSRAF

Calibre nmSRAF runs in batch mode, and requires an SVRF file. This input file is processed by the Calibre nmDRC-H hierarchical database engine.

calibre 358

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

calibre

Linux®[1]command

Executes SVRF commands.

Usage

$CALIBRE_HOME/bin/calibre
-drc [-hier] ***svrf_filename***
[-turbo [*number_of_processors*]] [-turbo_litho [*number_of_processors*]]
[-remotefile *filename*]

Arguments

- **-drc** [-hier]

  Selects hierarchical Calibre nmDRC checking.

- ***svrf_filename***

  Specifies the SVRF filename.

- -turbo [number_of_processors]

  Directs Calibre nmDRC-H to use multi-threaded parallel processing for all stages except LITHO operations.

- -turbo_litho [number_of_processors]

  Directs Calibre nmDRC-H to use multi-threaded parallel processing when performing Litho operations. The optional number_of_processors argument is a positive integer that specifies the number of CPUs to use in the processing. If you do not specify a number, Calibre nmDRC-H runs on the maximum number of CPUs available.

  You can specify the -turbo and the -turbo_litho options concurrently in a single command line, and the respective number_of_processors strings can vary between the two options.

- -remotefile *filename*

  Enables multi-threaded operations on remote hosts of a distributed network. This option is part of the Calibre MTflex multi-threaded, parallel processing architecture. The remote hosts do not have to be of the same platform type when using this option.

  The -remotefile option must be specified in conjunction with the -turbo or -turbo_litho option. Any of these commands may specify the number of processors you are using, including those on the remote hosts. The filename specifies the pathname of a configuration file containing information for the primary and remote hosts. You must have the required number of licenses for your job.

  The -remotefile option only applies to hierarchical applications. For more details, see the *Calibre Administrator's Guide*.

1. Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Appendix C
Using the Calibre WORKbench RET Flow Tool With Calibre nmSRAF

The RET Flow Tool provided in Calibre WORKbench provides a GUI interface for generating and running application setup files.

The RET Flow Tool simplifies management of the setup file. Fine tuning of simulation settings has been condensed with these tools to enable rapid recipe development. In Calibre WORKbench simulations, the setup parameters are loaded directly into Calibre WORKbench from the RET Flow Tool.

RET Flow Tool and Calibre nmSRAF 360
Invoking the RET Flow Tool 360
Importing an Existing OPC Session to RFT 362
Extracting an mbsraf Session From an SVRF File 364
Setting RFT Application Options 367
Performing Gradient Mapping 371
Drawing Intensity Cutlines 374

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

RET Flow Tool and Calibre nmSRAF

The RET Flow Tool is a productive GUI-based tool used with Calibre nmSRAF.

Table C-1. Usage of the Text nmOPC RET Flow Tool

Step	Description
Invoking the RET Flow Tool	The RET Flow Tool provides a convenient, productive method for executing a number of GUI-based applications with Calibre nmSRAF.
Importing an Existing OPC Session to RFT	You import the Calibre nmOPC session into the RET Flow Tool.
Extracting an mbsraf Session From an SVRF File	You extract the Calibre nmSRAF portions of your SVRF file, and import it into the RET Flow Tool. This session is added to the existing nmOPC session already in place.
Setting RFT Application Options	You can set a variety of options within the RET Flow Tool for analysis.
Performing Gradient Mapping	You can display gradient maps superimposed on your layout using Calibre WORKbench.
Drawing Intensity Cutlines	You can draw cutlines and display resulting aerial intensity plots using Calibre WORKbench.

Invoking the RET Flow Tool

The RET Flow Tool provides a convenient, productive method for executing a number of GUI-based applications with Calibre nmSRAF.

Prerequisites

- Install Calibre WORKbench and licensing.
- Install Calibre nmOPC, Calibre nmSRAF, and licensing.
- Set the environment variable CALIBRE_HOME to the location of your Siemens EDA tree. See "*Setting the CALIBRE_HOME Environment Variable*" in the Calibre Administrator's Guide for additional details.

Procedure

1. In a shell, invoke Calibre WORKbench from the command line:

```
calibrewb
```


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

As an alternative, you invoke RFT directly from the command line by specifying the -rft option with any previously-created flow:

```
calibrewb -rft myFlow
```

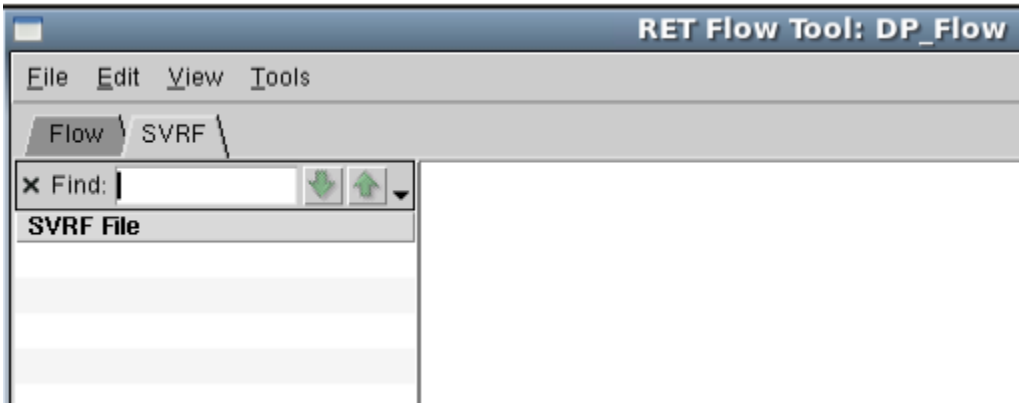
In either case, the prompt in the shell changes to “%” and Calibre WORKbench appears. For a complete description, refer to the *Calibre WORKbench User’s Manual* for all other available invocation command line options.

2. Load a design layout into Calibre WORKbench and zoom into a small portion of the design for subsequent OPC and SRAF generation.

3. If you did not invoke RFT directly from the command line (alternative Step 2), start RFT from the **Litho** menu in Calibre WORKbench (**Litho** > **RET Flow Tool**):

   The RET Flow Tool appears.

Figure C-1. RFT Window

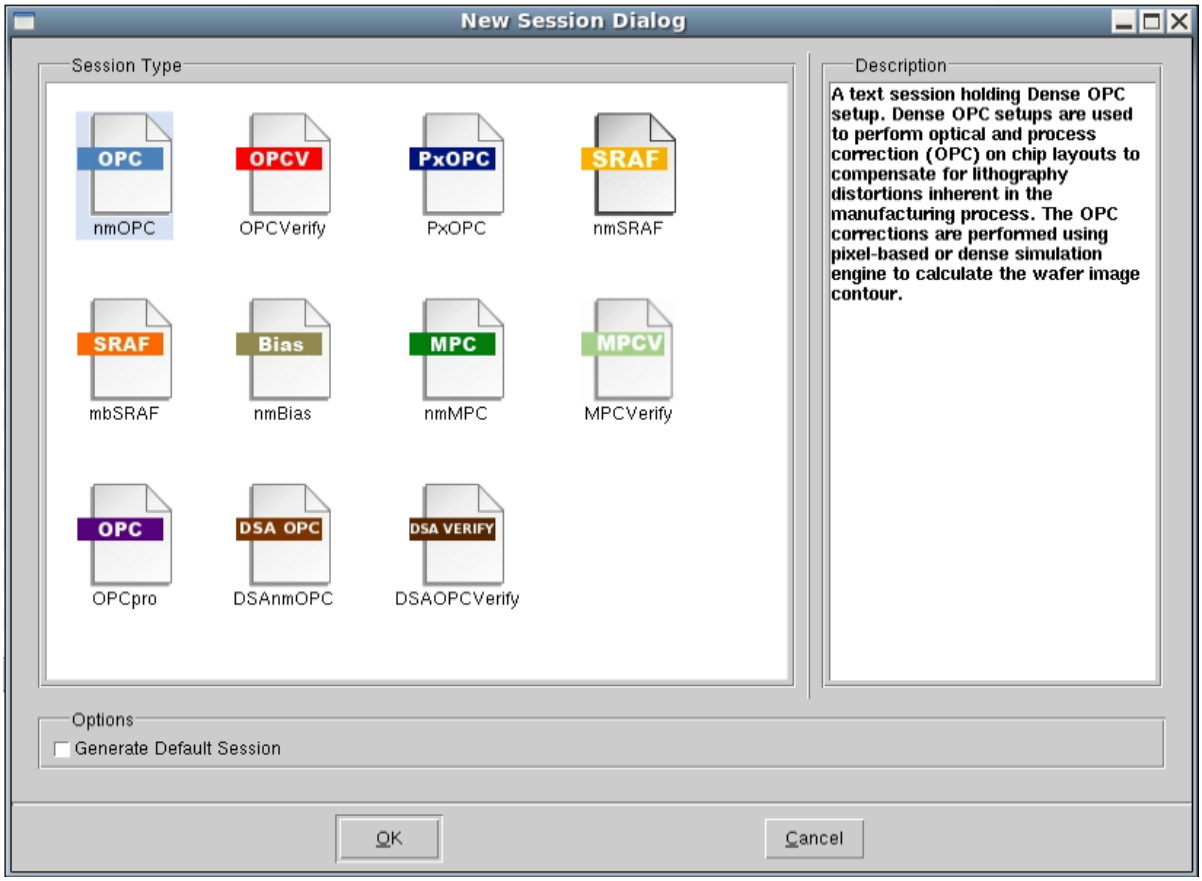


4. To add new sessions of various applications, choose **File** > **Add New Session**.

   The New Session form appears with all available session applications.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-2. RFT Applications



Choosing any one of these applications impacts what options are available from the menus available in RFT. You can start from scratch by choosing a new session, but it is faster to import existing sessions as shown in the next section.

Results

You opened a design in calibre WORKbench, started RFT, and gained access to add new applications for the RET Flow Tool.

Importing an Existing OPC Session to RFT

You import the Calibre nmOPC session into the RET Flow Tool.

Prerequisites

- An nmOPC session file containing Calibre nmOPC simulation commands

Procedure

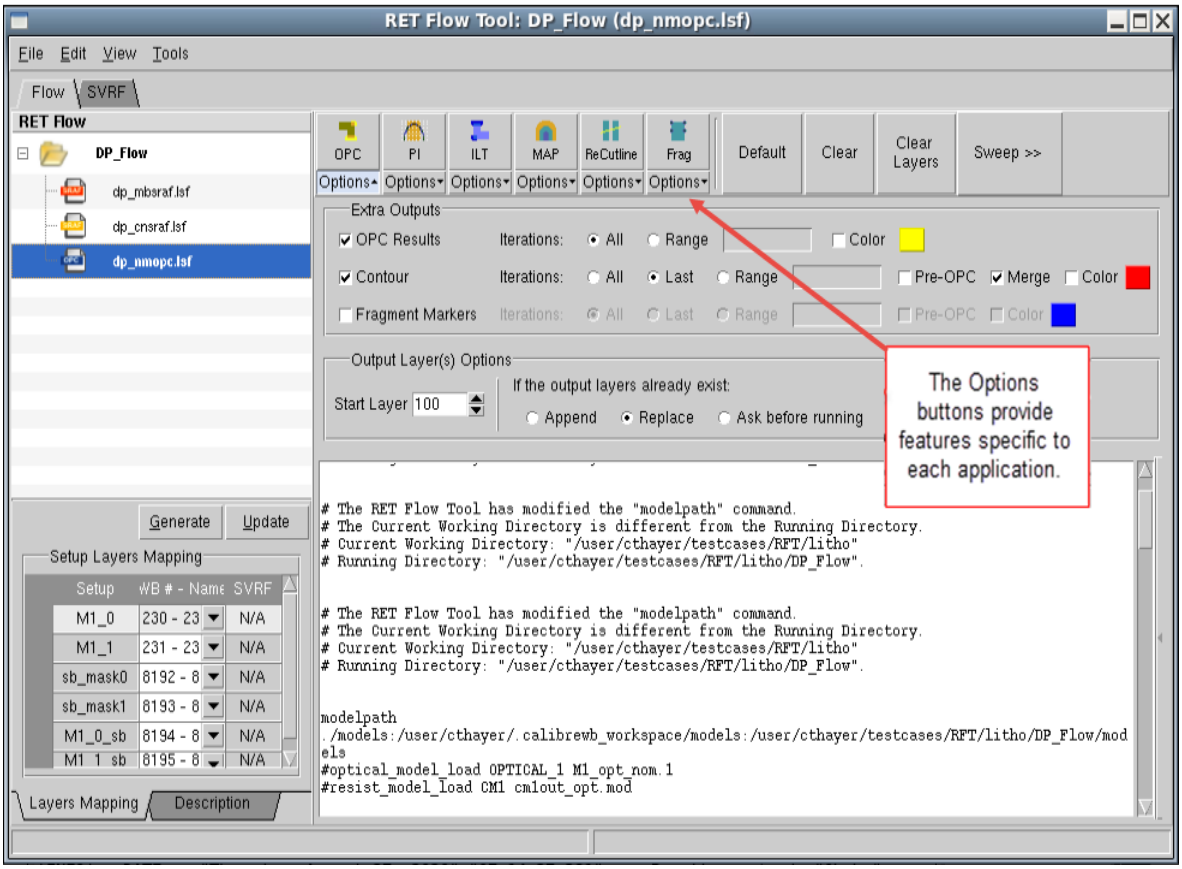
1. In the RET Flow Tool, add an existing nmOPC session file by choosing **File > Add Existing Session**.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

2. In the dialog box, choose the *.lsf* file of interest and click **OK**.

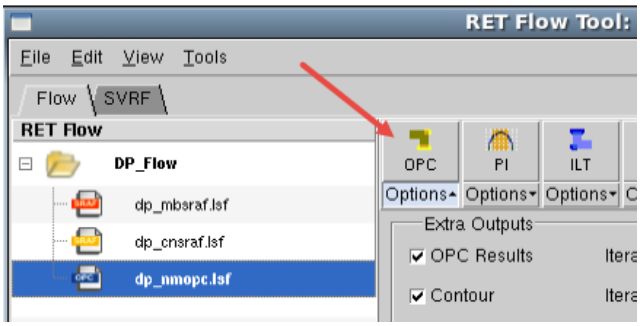
   The session is loaded into RFT with some default settings.

Figure C-3. nmOPC Session



3. Click the **OPC** button in RFT to generate OPC only on the small area in which you previously zoomed in.

Figure C-4. Run nmOPC



OPC is generated and layers are added to the design.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-5. OPC Output on Design

Extracting an mbsraf Session From an SVRF File

You extract the Calibre nmSRAF portions of your SVRF file, and import it into the RET Flow Tool. This session is added to the existing nmOPC session already in place.

Prerequisites

- An SVRF file containing Calibre nmSRAF commands

Procedure

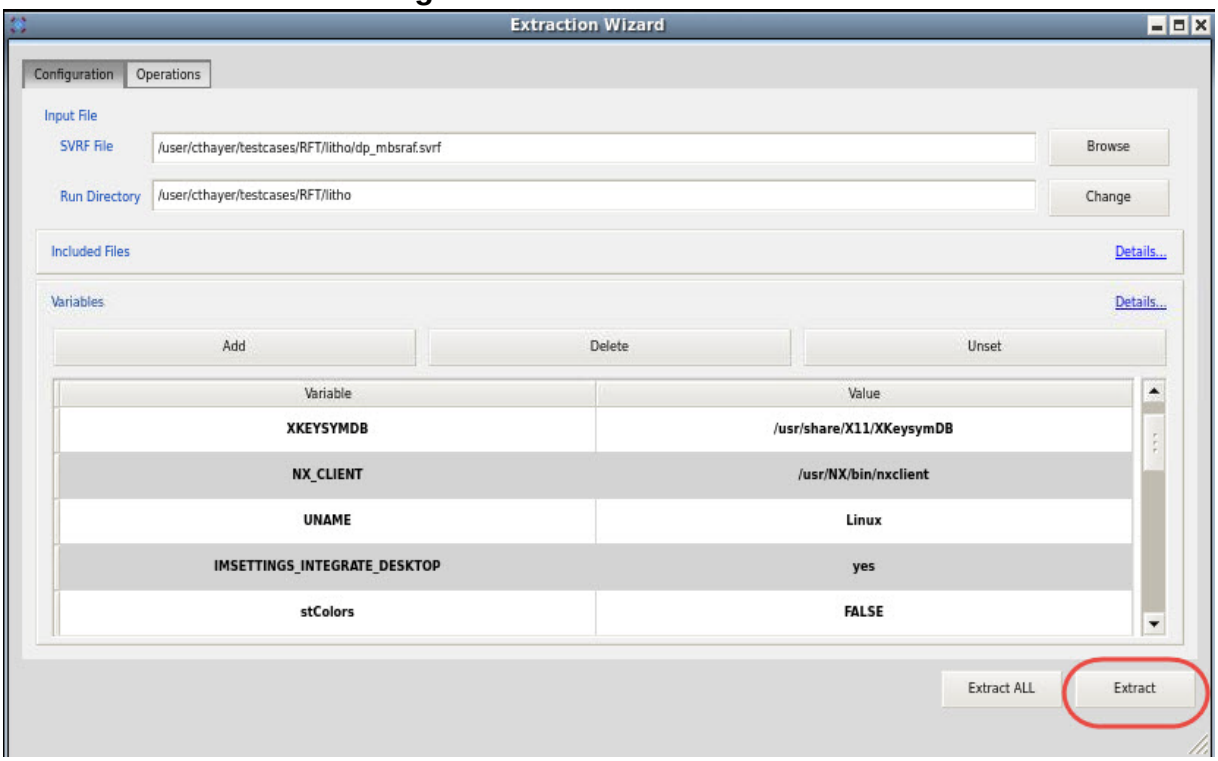
1. In the RET Flow Tool, extract the mbsraf portions of an SVRF file by choosing **File > Extract from SVRF**, and in the dialog box that appears, open the SVRF file of interest.

   The Extraction Wizard appears with the variables expanded in the left pane and their respective values in the right pane.

2. Click **Extract** (in the lower-right corner of the Extraction Wizard window) as shown in Figure C-6.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-6. Extraction Wizard

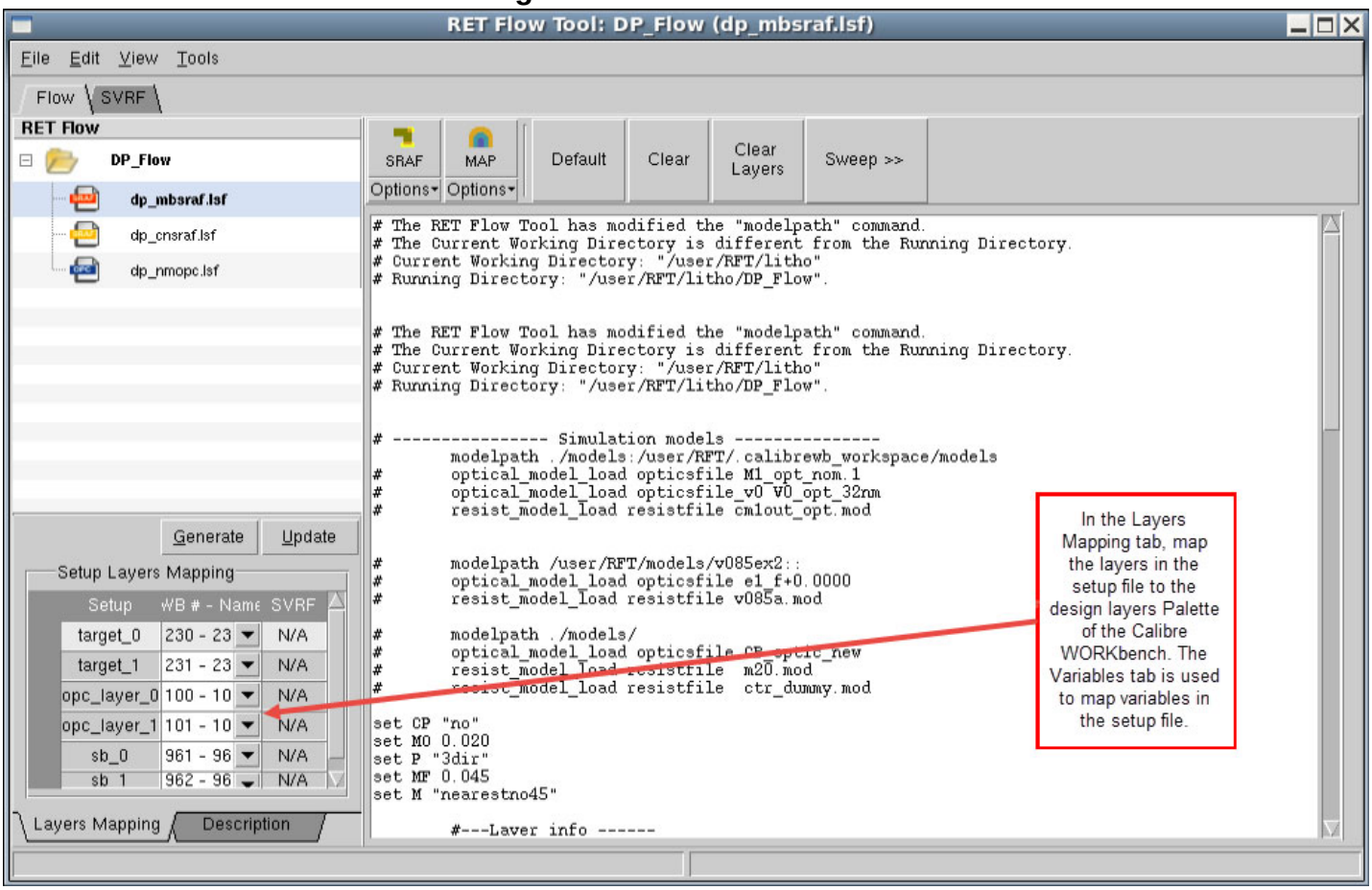


The mbsraf session window appears with the mbsraf session map expanded in the left pane and SVRF code in the right pane.

3. Map the input layers from the setup file to layers displayed in the Layers Palette of Calibre WORKbench as shown in Figure C-7. Layer numbers are matched by order to layer statements.

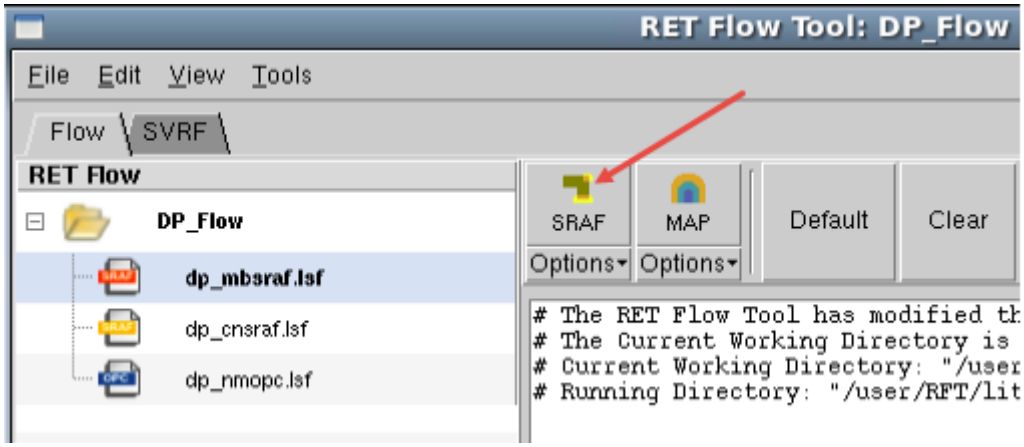
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-7. mbsraf Session



4. Click the **SRAF** button as shown in Figure C-8 to start Calibre mbSRAF.

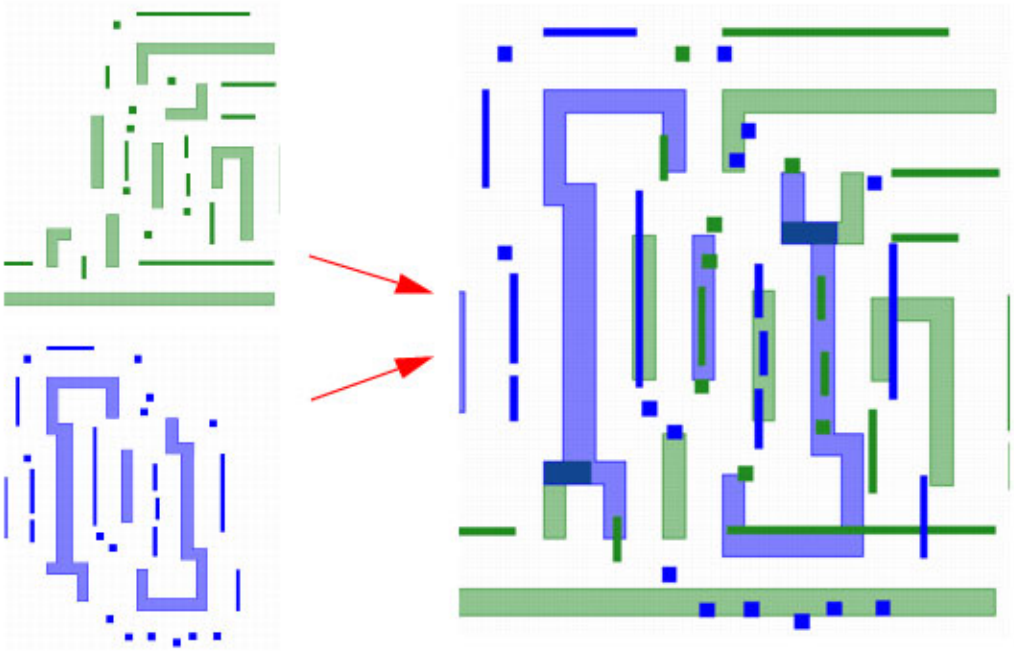
Figure C-8. RET Flow Tool For Running mbsraf



The SRAF is generated and output on the design for all layers that were input (in this case, both metal masks of a double-patterning design).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-9. SRAF Output



Results

You have imported your Calibre nmOPC and nmSRAF commands into the RET Flow Tool and performed OPC and SRAF generation.

Setting RFT Application Options

You can set a variety of options within the RET Flow Tool for analysis.

Prerequisites

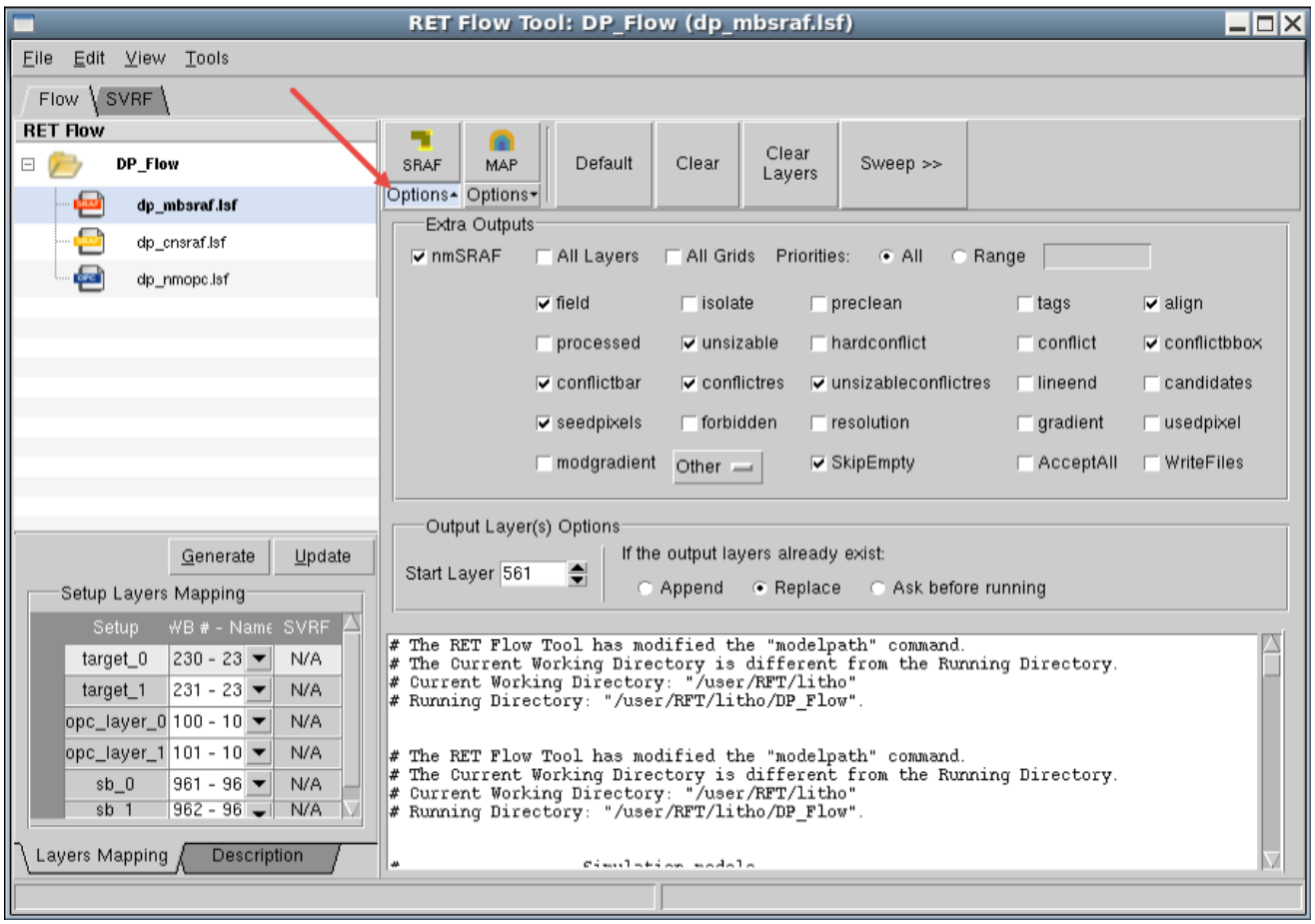
- A previously loaded application such as Calibre nmOPC or Calibre nmSRAF

Procedure

1. In Calibre WORKbench, click the **Options** button as shown in Figure C-10. The illustration shows that Calibre nmSRAF has been enabled and outputs the selected layers.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-10. mbsraf Options



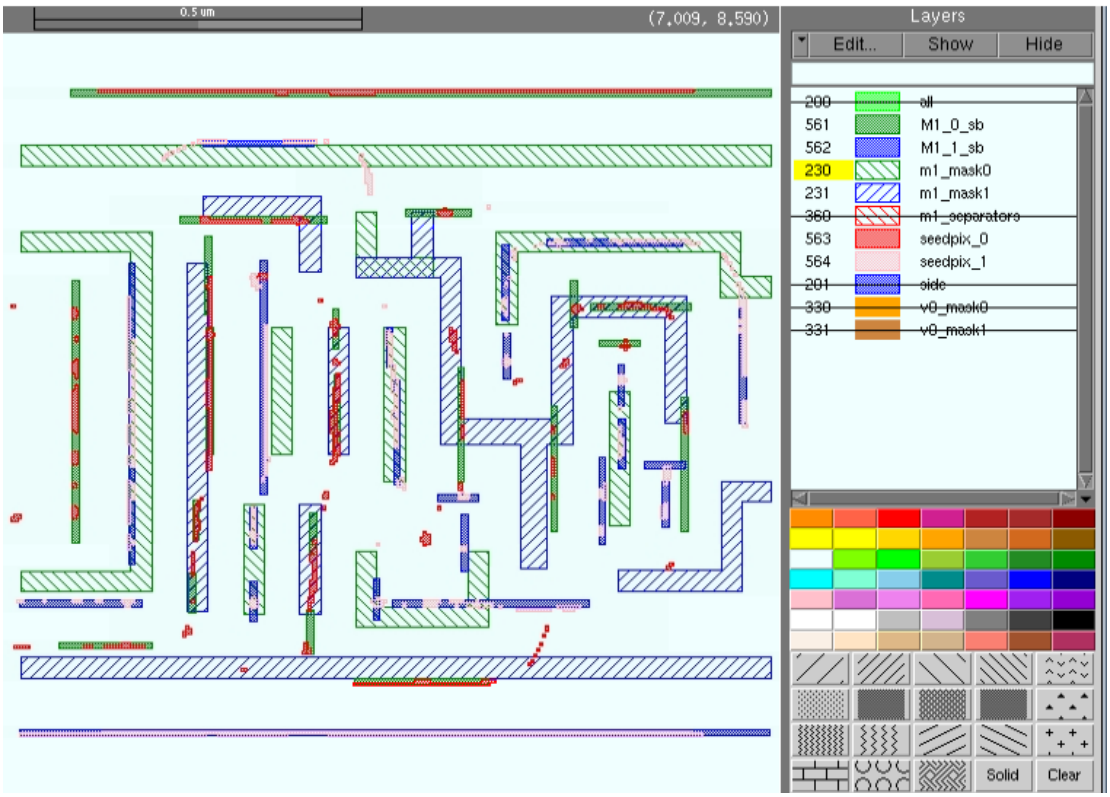
This form allows you to specify different types of information to output any iteration of the SRAF generation run, including gradient maps, debug layers, and SRAF. Only the visible region is simulated and written to the selected layers.

2. You can set the initial output layer number in the Output Layer(s) Options section. The output layers are incremented by 1 for each new output layer per iteration produced.

3. Click **SRAF** in this form to assert the options and run nmSRAF.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-11. Example Output From OPC Run Options for cnSRAF

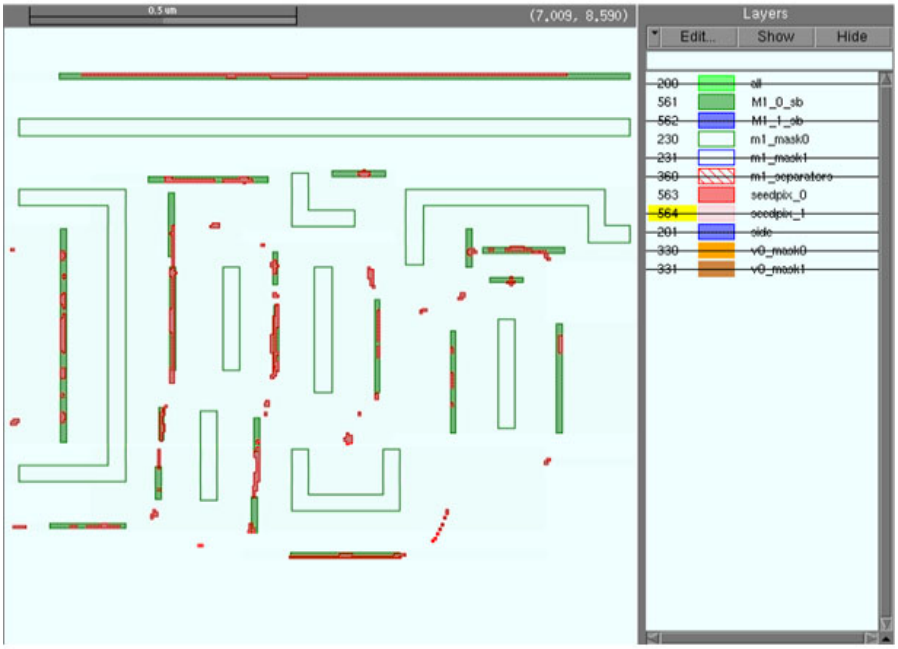


These are several output layers produced by the mbsraf run. Layers 100 through 109 were generated in the initial run. Layers above 500 are the layers generated by the RFT-based mbsraf run. They include the initial seed pixels (red and pink, respectively) and

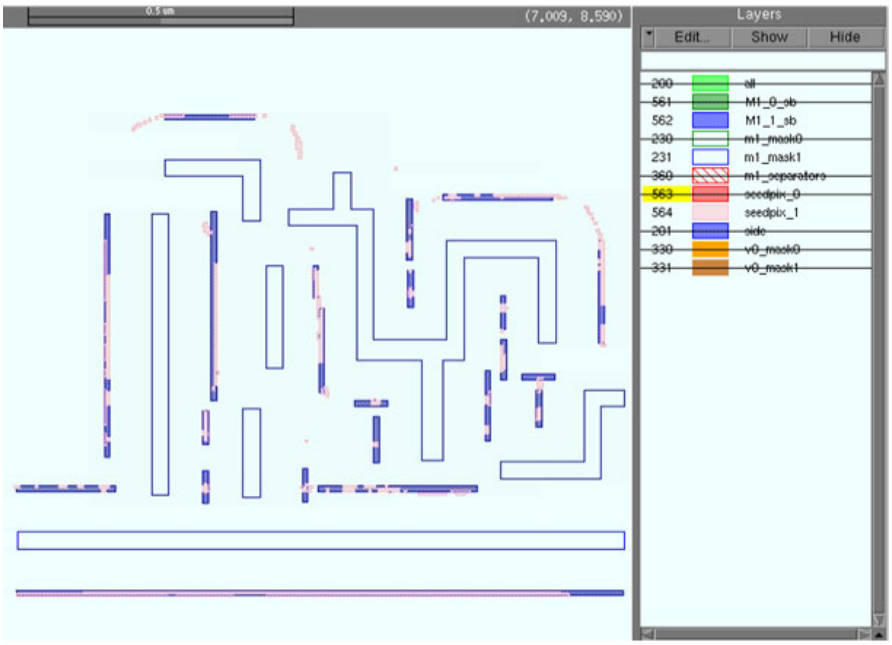
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

final model-based SRAF (green and blue, respectively). Masks 0 and 1, with their respective SRAFs, are shown for clarity with two illustrations in Figure C-12:

Figure C-12. Layers Generated by the mbsraf Run



Mask 0



Mask 1

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Results

You have generated model-based SRAF and the selected gradient maps for metal layer. You ran the mbSRAF engine within the RET Flow Tool.

Performing Gradient Mapping

You can display gradient maps superimposed on your layout using Calibre WORKbench.

Prerequisites

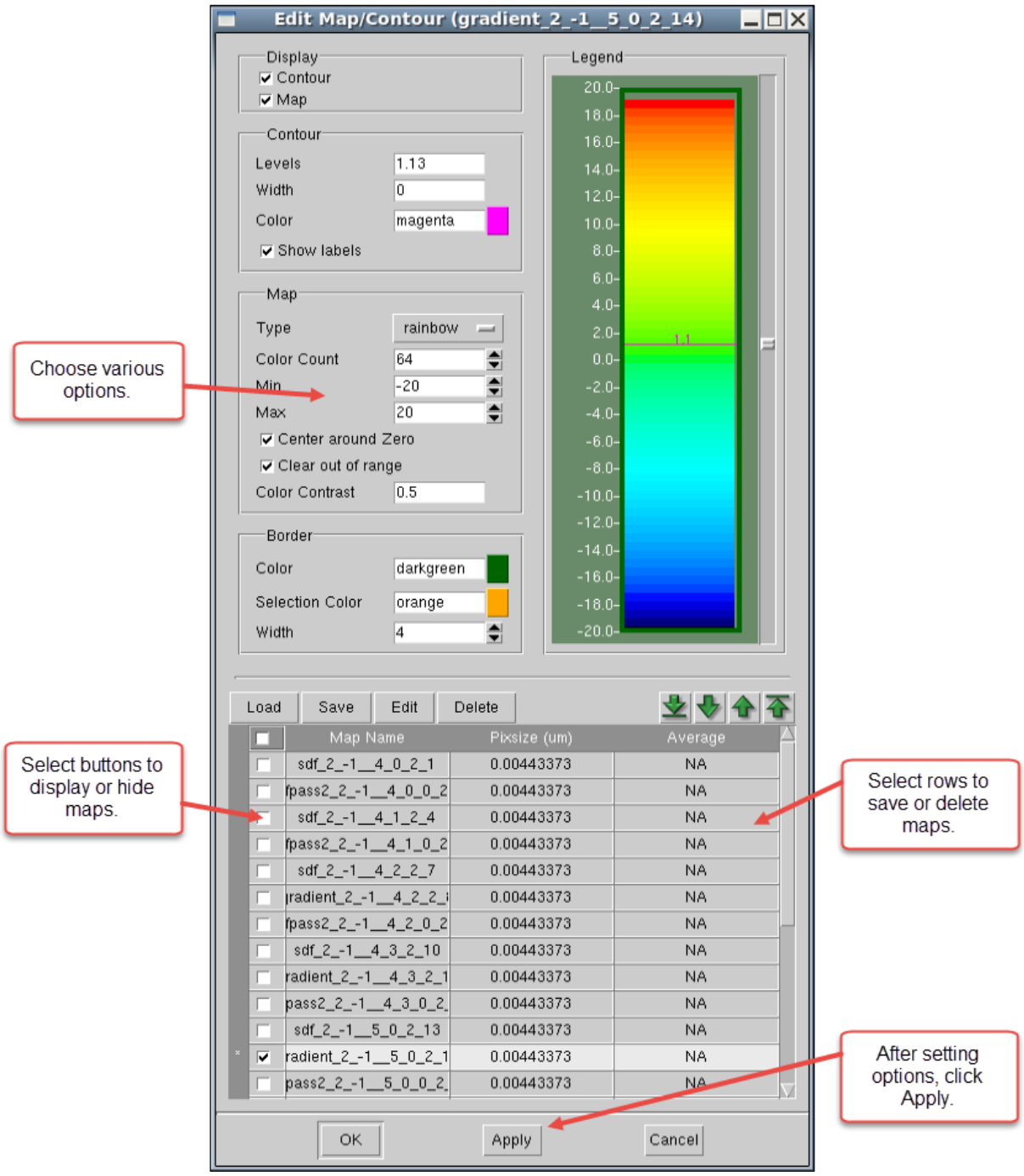
- A previously performed SRAF and gradient map run

Procedure

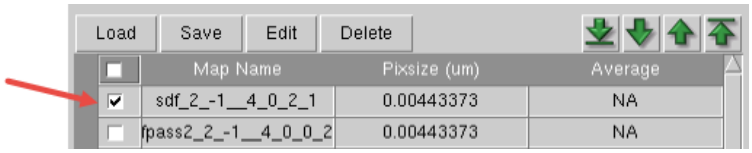
1. In the Calibre WORKbench layout window double-click the visible gradient map (or select **Edit** > **Maps/Contours**). This invokes the Edit Map/Contour form. (see Figure C-13).

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Figure C-13. Edit Map/Contour Form



2. Turn off the gradient map and leave the sdf map on.

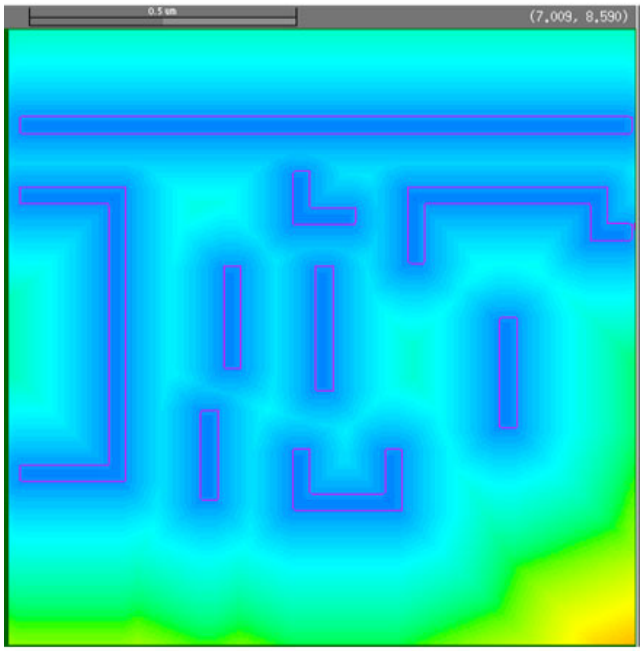


Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

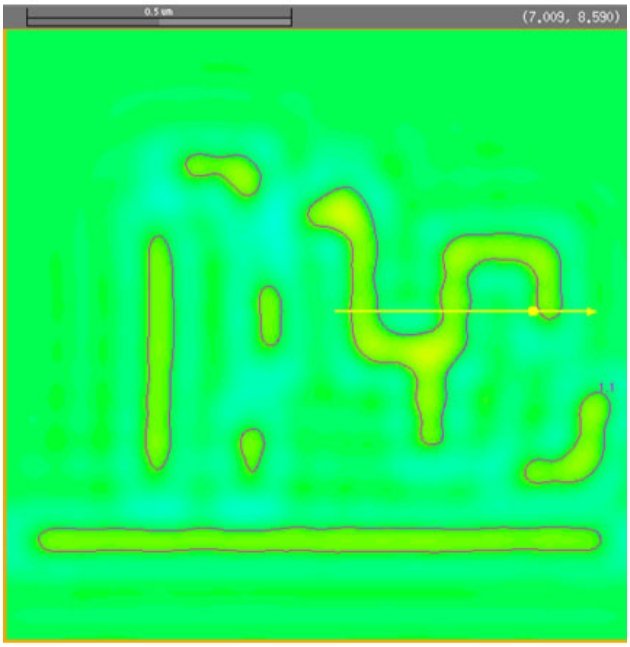
3. Click **Apply**.

   The sdf map appears as you have changed it.

Figure C-14. SDF and Gradient Maps



SDF map.



Gradient map with contour and cutline.

4. Turn the gradient map back on for the next procedure, “Drawing Intensity Cutlines” on page 374.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

5. Click **Apply** or **OK**.

Results

You have modified the display of the gradient map for a metal layer.

Drawing Intensity Cutlines

You can draw cutlines and display resulting aerial intensity plots using Calibre WORKbench.

Prerequisites

- A previous mbsraf run with RFT
- Generated gradient maps

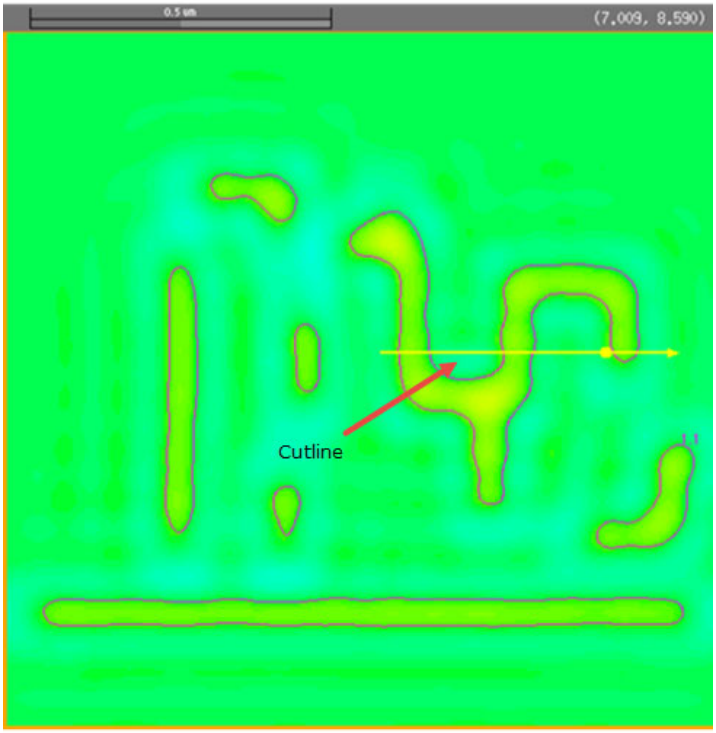
Procedure

1. Click the map.

   The map border turns orange.

2. In the Calibre WORKbench icon bar, choose Cutline.
3. Draw a cutline through a feature on the map by clicking once for the cutline start and once for the cutline end.

Figure C-15. Drawing a Cutline



Gradient map with contour and cutline.

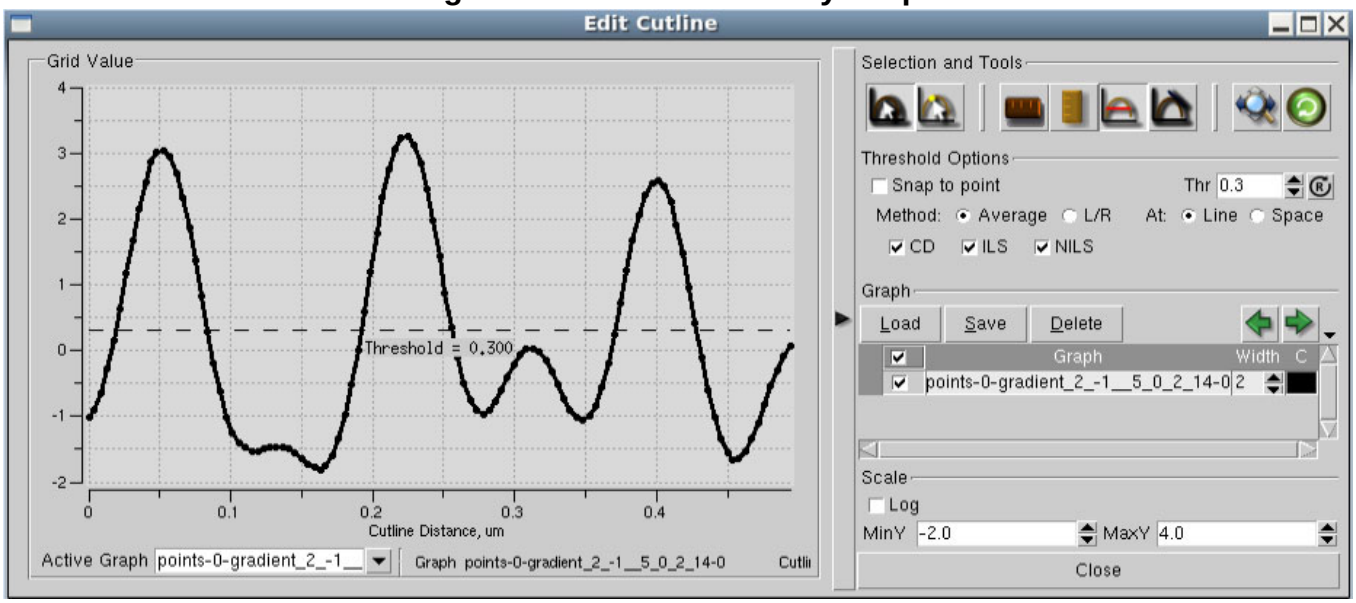
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

After clicking the cutline end, the Cutline Run Options form appears.

4. Click **OK** in the Cutline Run Options form.

   The Edit Cutline form appears displaying the cutline graph. The graph shows the cutline distance on the X-axis and the aerial intensity on the Y-axis.

Figure C-16. Aerial Intensity Graph



Results

You have drawn a cutline across the gradient map for a metal layer target.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Index

— Symbols —

[], 16
{}, 16
\, 41
|, 16

— B —

Backslash (\), 41
Bold words, 15

— C —

Calibre tools
    LITHO FILE statements, 41
centerwidth, 234
clear
    background, 45
Cloning cells, 47
cnsraf, 68
cnsraf debug, 69
Command reference, 15
Configuration
    litho_model command, 57
    modelpath command, 60
    optical_model_load command, 62
    resist_model_load command, 67
Configurations
    Dense, 26
    Isolated, 26
Conflict
    region, 18, 21, 22
    resolution, 226
conflictOnly, 23
Courier font, 15

— D —

Dark
    background, 45
Dense configurations, 26
Double pipes, 16
dynamicsizing, 295, 298

— E —

Exclusion zones, 18

— F —

flare_longrange command, 51

— H —

Heavy font, 15

— I —

image_options, 53, 244
Inline file, 41
Isolated configurations, 26
Italic font, 15

— L —

Line continuation, 41
lineendmerge, 308, 311
LITHO FILE command, 41
litho_model, 57
localmaxgradient
    centerwidth, 175

— M —

MBSRAF, 32
mbSRAF
    LITHO FI LE statements, 43
mbsraf, 75
mbsraf debug, 77
Minimum keyword, 15
minLength, 158, 171
minoffset, 155
Model-based template extraction, 18
modelpath setup command, 60

— N —

negative, 262

— O —

optical_model_load setup command, 62
Optimization-based template extraction, 18
oversizesraf, 280

— P —

Parentheses, 16
Pipes, 16
processing_mode, 64
PV band, 14

— Q —

Quotation marks, 16

— R —

rectedge
    angledalso, 177, 215
Region
    conflict, 21, 22
    semi-dense, 26
resist_model_load setup command, 67
RET MBSRAF, 32
RET SBAR, 36

— S —

SBAR, 36
Semi-dense
    region, 26
setlayer
    cnsraf, 68
    cnsraf debug, 69
    mbsraf, 75
    mbsraf debug, 77
Share SVRF variables, 91
Slanted words, 15
Square parentheses, 16
sraf, 283
sraf_options, 80
SVRF
    LITHO FILE, 41

— T —

Template
    prioritization, 18
Template extraction
    model-based, 18
    optimization-based, 18

— U —

Underlined words, 15

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.