



SIEMENS EDA

Calibre® Local Printability Enhancement User's and Reference Manual

Software Version 2021.2

Unpublished work.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 LPE	9
Calibre LPE Overview	9
Calibre LPE Workflow	10
Calibre LPE Terminology	11
Calibre LPE Requirements	12
Syntax Conventions	13
Calibre LPE Modes of Operation	14

Chapter 2

Calibre LPE Command Reference	17
calibre -lpe	19
SVRF Commands	26
LITHO FILE	27
RET REFINE	29
Litho Setup File Format for RET REFINE	32
Litho Setup File Commands	35
denseopc_context_parameters	38
denseopc_copy_fragments	41
denseopc_fragment_coincident	42
denseopc_fragment_nearly_coincident	43
denseopc_init_mode	44
denseopc_insert_correction	45
denseopc_limit_excess	46
denseopc_pre_notchfill	47
denseopc_smooth_corrections	48
denseopc_use_correction_map	49
fragment_optimize... reset_displacement	50
job 4 finetune	51
layer	53
min_frag_len	56
min_space	57
min_width	58
mrc_min_length	59
mrc_min_rect_length	61
mrc_min_rect_width	62
mrc_min_square	63
mrc_small_feature_size	65
pick_region	67
pxopc_init_mode	68
pxopc_transition_region	69
pxopc_weight_layer	70

rearrangement_disable	72
refine_distance	73
refine_exec	76
refine_map	83
refine_markers	88
refine_markers_limit	89
refine_markers_style	90
refine_patterns	92
refine_process	96
refine_prune_markers	97
refine_regions	98
refine_stop_iterations	99
refine_stop_layer	100
refine_update_layer	102
refine_visible_sim	103
reject_duplicate_layers	105
smart_cut_disable	106
Chapter 3	
Calibre LPE Best Practices	107
LPE Setup Best Practices	107
Tool Interaction Best Practices	108
Memory Management Best Practices	109
Pre-Existing MRC Errors	110
Browsing LPE Results	111
Index	
Third-Party Information	

List of Figures

Figure 1-1. Calibre LPE as Part of the Post-Tapeout Flow	9
Figure 1-2. High-Level Calibre LPE Workflow	10
Figure 1-3. Recommended Calibre LPE Workflow	11
Figure 1-4. Calibre LPE Regions.	12
Figure 2-1. Valid Small Features for MRC - mrc_min_length	59
Figure 2-2. Valid Small Features for MRC - mrc_min_rect_length	61
Figure 2-3. Valid Small Features for MRC - mrc_min_rect_width	62
Figure 2-4. Valid Small Features for MRC - mrc_min_square	63
Figure 2-5. Selecting Markers	67
Figure 2-6. Region Size Measurements.	74
Figure 2-7. refine_map iteration 0	86
Figure 2-8. refine_map iteration 1	86
Figure 2-9. refine_map iteration all	87
Figure 2-10. Transformations for Pattern Matching	92
Figure 2-11. Matching Repair Regions	93
Figure 2-12. Visible Simulation Region	104
Figure 3-1. LPE Browser Panels	112
Figure 3-2. LPE Browser Region Decisions	113

List of Tables

Table 1-1. Syntax Conventions	13
Table 2-1. Calibre LPE SVRF Commands	26
Table 2-2. Required Commands for RET REFINE Setup Files	32
Table 2-3. Calibre LPE Litho Setup File Commands	35
Table 2-4. Supported SVRF Operations	79
Table 2-5. Predefined LPE Layers for Regions	80
Table 2-6. Outputs by Iteration	85
Table 3-1. Calibre LPE Best Practices	107
Table 3-2. Calibre LPE Dense OPC Best Practices	108
Table 3-3. Consistency Best Practices	108

Chapter 1

Introduction to Calibre LPE

This chapter introduces Calibre® Local Printability Enhancement (Calibre LPE), and contains the following sections related to using the product:

Calibre LPE Overview	9
Calibre LPE Workflow	10
Calibre LPE Terminology	11
Calibre LPE Requirements	12
Syntax Conventions	13
Calibre LPE Modes of Operation	14

Calibre LPE Overview

Calibre LPE corrects identified OPC hotspots using advanced pixel-based optimization. Pixel-based optimization offers the best process window results but is too computationally intensive for full-chip layouts. Calibre LPE allows you to quickly run OPC on the initial layout and then refine the results only where needed. It then smoothly integrates the refined regions with the main mask results.

The Calibre LPE software makes use of the Calibre® pxOPC™ third-generation OPC algorithm, the Calibre® nmOPC™ dense OPC tool, the Calibre® nmBIAS™ rule-based OPC tool, the Calibre® OPCverify™ verification engine, and ULTRAfex processing mode. It is able to refine both main features and SRAFs.

Like other Calibre OPC tools, Calibre LPE is called from within an SVRF rule file.

Figure 1-1. Calibre LPE as Part of the Post-Tapeout Flow



Related Topics

[Calibre nmOPC Users and Reference Manual](#)

[Calibre OPCverify Users and Reference Manual](#)

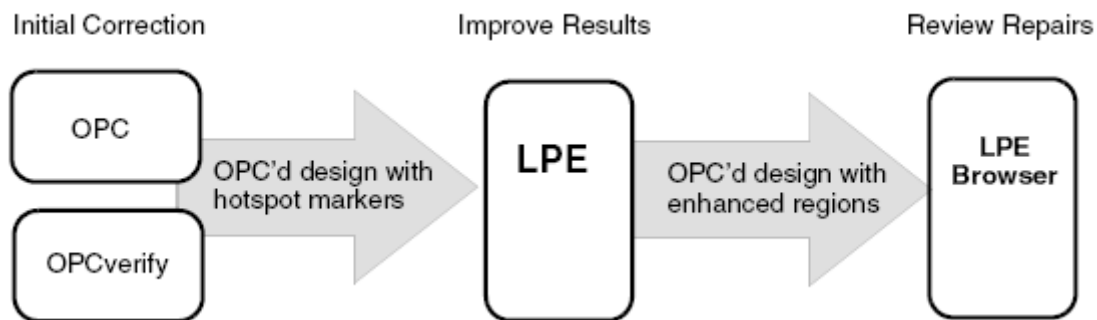
[Calibre pxOPC Users and Reference Manual](#)

Calibre LPE Workflow

Calibre LPE is run after other tools have identified and marked the particular areas in a layout that do not meet requirements.

You then run Calibre from the command line upon the particular layout areas. For a Calibre LPE run, you use an SVRF rule file that includes an LPE setup file. The LPE setup file calls an OPC repair tool on the areas to further improve them, and then Calibre OPCverify to identify any new hotspots. When the layout meets criteria, the results are passed back to the main Calibre flow and saved to a layout database. After the run, you can view the results in Calibre WORKbench with the LPE Browser.

Figure 1-2. High-Level Calibre LPE Workflow

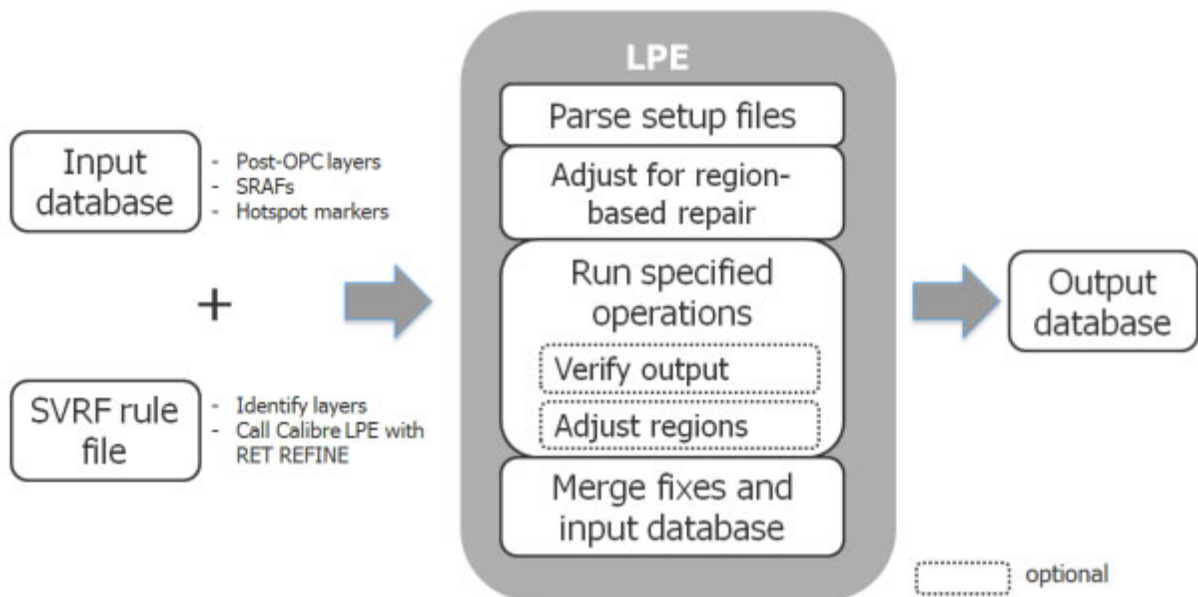


For Calibre LPE, the post-OPC layout should already be mostly acceptable and MRC clean. Remaining problems should be mostly localized and few. It is not recommended to run on an initial design that has many areas of poor quality.

Figure 1-3 shows the stages that occur during a Calibre LPE run. When Calibre LPE parses the setup files, it reads its own specifications and then adjusts the defaults for the tools it runs based on those settings. The exact operations that are run depend on the `refine_exec` commands in the setup file.

The workflow includes optional stages for verifying the fixes before merging, and adjusting regions. These can be combined in an *iterative flow*. In the iterative flow, you define an upper limit on iterations using `refine_stop_iterations` and an earlier limit, should no hotspots be found on re-verification, with `refine_stop_layer`.

Figure 1-3. Recommended Calibre LPE Workflow



By default, however, Calibre LPE runs one pass (a single iteration). The results are then merged with the layers from the input database. Rule checks in the SVRF rule file determine which layers are written to a new database.

Related Topics

[SVRF Commands](#)

[Litho Setup File Commands](#)

[refine_exec](#)

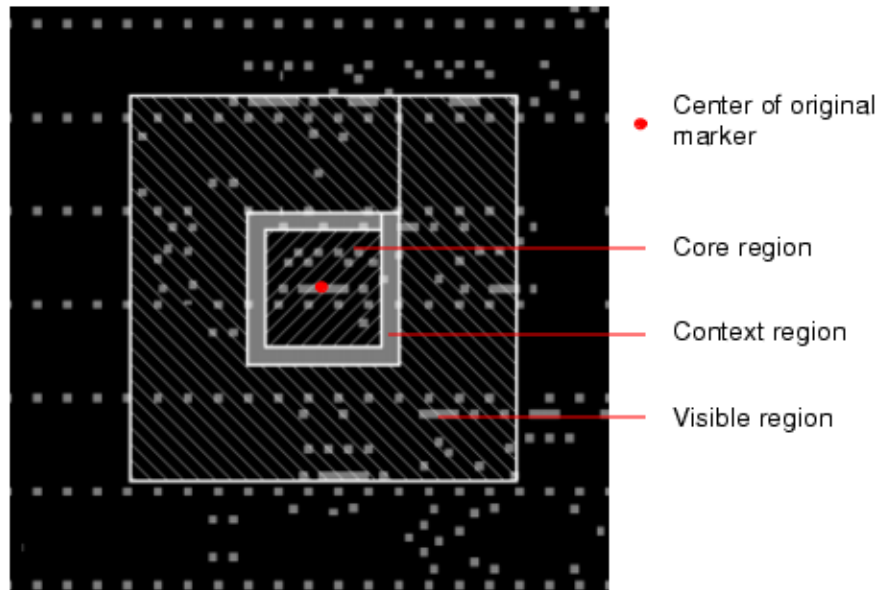
[refine_stop_iterations](#)

[refine_stop_layer](#)

Calibre LPE Terminology

To more easily understand how to set up a Calibre LPE rule file, you need to be familiar with *regions*. Calibre LPE creates three regions in a layout: core, context, and visible. These three regions are referred to collectively as the *repair region*.

Figure 1-4. Calibre LPE Regions



The *core region* is expanded around the center of a hotspot marker. The marker must be a polygon. The size of the core region is based on the optical radius. It can be explicitly set with `refine_distance0`, or forced to encompass the entirety of long markers with `refine_markers_style`. Inside the core region, full re-optimization occurs. Original shapes are replaced with shapes from Calibre pxOPC or Calibre nmOPC.

The *context region* is a zone outside the core region. Its size is also based on the optical radius, or can be explicitly set with `pxopc_transition_region` or `refine_distance1`. The context region ensures smooth transitions between the core region's new shapes from the OPC tool and the original OPC output mask.

The *visible region* extends outside the context region. While geometries in the core and context regions can be changed by Calibre LPE, the geometries in the visible region are used only for simulation. Shapes are visible for core and context simulation, but frozen. Its size is set to the optical radius plus 0.2 microns, or it can be explicitly set with `refine_distance2`.

Related Topics

[refine_map](#)

[pxopc_transition_region](#)

[refine_distance](#)

[refine_markers_style](#)

Calibre LPE Requirements

Before using Calibre LPE, you need the correct environment, required files, and input.

To use Calibre LPE, you must have the following:

- The CALIBRE_HOME environment variable pointing to a Calibre version 2013.2 or higher installation.
- Licenses for Calibre nmDRC, Calibre LPE, and any other Calibre tools called in the setup file. These are typically Calibre OPCverify and either Calibre pxOPC or Calibre nmOPC. A Calibre® DESIGNrev™ or Calibre® WORKbench™ license is also recommended for viewing output.
- An SVRF file that specifies the input and output, as well as including at least one RET REFINE statement and the associated setup file. The SVRF file must also contain the statement LAYOUT ULTRA FLEX YES.
- GDS or OASIS®¹ layout file with hotspots indicated by markers.

You may also need additional setup files and models to use Calibre pxOPC, Calibre nmOPC, and Calibre OPCverify. These are described in their respective manuals.

Note

The setup files must be modified from their non-LPE forms. You should not do this modification explicitly; it is handled by Calibre LPE.

Calibre LPE makes use of correction layers, whereas the non-LPE original forms typically use target layers. Calibre LPE automatically makes the adjustments necessary to use correction layers.

Related Topics

[Calibre LPE \[Calibre Administrator's Guide\]](#)

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.
<i>Italic</i>	Italic fonts indicate a user-supplied argument.

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.

Table 1-1. Syntax Conventions (cont.)

Convention	Description
Monospace	Monospace fonts indicate a shell command, line of code, or URL. A bold monospace font identifies text you enter.
<u>Underline</u>	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.
{ }	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 { <i>element_name</i> [‘(<i>model_name</i>)’]} <i>device_layer</i> { <i>pin_layer</i> [‘(<i>pin_name</i>)’] ...} [‘< <i>auxiliary_layer</i> >’ ...] [‘(<i>swap_list</i>)’ ...] [<u>BY NET</u> BY SHAPE]	

Calibre LPE Modes of Operation

Calibre LPE runs in batch mode, which requires an SVRF file to be processed by the Calibre database constructor (calibre -drc). The processing is invoked from a shell command line. It requires Calibre® MTflex™ mode in order to support ULTRAfex processing. It does not run on the Calibre® FullScale™ platform.

The invocation is similar to the following:

```
calibre -drc -hier -turbo svrf | tee lpe.log
```

where the items in italics represent filenames that you provide, and the bold keywords are required.

After the initial run, you can review results using the LPE Browser in Calibre WORKbench. See “[Browsing LPE Results](#)” on page 111.

Chapter 2

Calibre LPE Command Reference

Calibre LPE is launched from a standard Calibre SVRF file. A litho setup file contains the tool-specific settings.

This chapter contains the following sections:

calibre -lpe	19
SVRF Commands	26
LITHO FILE	27
RET REFINE	29
Litho Setup File Format for RET REFINE	32
Litho Setup File Commands	35
denseopc_context_parameters	38
denseopc_copy_fragments	41
denseopc_fragment_coincident	42
denseopc_fragment_nearly_coincident	43
denseopc_init_mode	44
denseopc_insert_correction	45
denseopc_limit_excess	46
denseopc_pre_notchfill	47
denseopc_smooth_corrections	48
denseopc_use_correction_map	49
fragment_optimize... reset_displacement	50
job 4 finetune	51
layer	53
min_frag_len	56
min_space	57
min_width	58
mrc_min_length	59
mrc_min_rect_length	61
mrc_min_rect_width	62
mrc_min_square	63
mrc_small_feature_size	65
pick_region	67
pxopc_init_mode	68
pxopc_transition_region	69
pxopc_weight_layer	70
rearrangement_disable	72
refine_distance	73
refine_exec	76

refine_map	83
refine_markers	88
refine_markers_limit	89
refine_markers_style	90
refine_patterns	92
refine_process	96
refine_prune_markers	97
refine_regions	98
refine_stop_iterations	99
refine_stop_layer	100
refine_update_layer	102
refine_visible_sim	103
reject_duplicate_layers	105
smart_cut_disable	106

calibre -lpe

Shell command

A utility that creates a starting setup suitable for LPE-driven dense opc from a Calibre nmOPC rule file.

Usage

calibre -lpe *output.svrf input.svrf*

Arguments

- ***output.svrf***
A required argument that specifies the name of the file to create. If ***output.svrf*** already exists, the command exits with an error. The supplied filename does not need to have the file suffix *.svrf* though this is recommended.
- ***input.svrf***
A required argument that specifies the existing rule file. The file must contain at least one valid LITHO DENSEOPC operation.

Description

The LPE script generator is a utility that takes an input rule file that contains LITHO DENSEOPC operations and converts it to one that uses RET REFINE operations to direct a denseopc-based repair process. The input rule file can contain Tcl code, although Tcl inside the denseopc_options block is ignored.

The ***output.svrf*** file contains general DRC statements such as LAYOUT PATH and DRC RESULTS DATABASE. It also adds LAYER statements for the additional layers, RET REFINE layer operations, DRC CHECKMAP statements for all new layers, and litho setup files for both LPE and Calibre nmOPC.

The generated rules only serve as a starting point. To complete the final LPE script, you will need to perform the following:

- Verify all necessary general statements.
- Reassign the layer numbers in accordance with the real design database. The LPE script generator uses an arbitrary set of layer numbers for the LAYER definition as well as for the DRC CHECK MAP statements it adds. The LPE script generator also adds layers such as markers and initial correction layers, which need to be properly mapped.
- Correct the MRC constraints. The LPE script generator uses arbitrary hard-coded values for min_frag_len, min_width, and min_space. The values are likely wrong for your process.

- Add refine_exec ... opcverify calls. The generated setup only contains the minimum calling sequence for refine_exec ... denseopc. It does not contain any calls to Calibre OPCverify, and these are needed to define hotspots on subsequent iterations.
- Set up iterations. The generated setup runs only a single LPE iteration.

Examples

Example 1: Specifying a Particular Setup File

When there are multiple DENSEOPC operations with different setup files, you can indicate which one should be converted with the environment variable LPE_SELECT_SETUP. If this is not set, calibre -lpe converts the first encountered setup file for Calibre nmOPC.

Consider an SVRF flow that uses different LITHO DENSEOPC setups for different parts of the design. One operation calls for a setup file named “analog” and a different operation uses a setup file named “digital”. In a C shell terminal, you would enter the following commands to be sure of basing the LPE script generator’s output on the digital setup file:

```
setenv LPE_SELECT_SETUP digital
calibre -lpe lpe.svrf original.svrf
```

where *original.svrf* is the name of the original rule file containing multiple LITHO DENSEOPC operations.

Example 2: Translation of an SVRF File

The following example shows the original input and the resulting output from a call to calibre -lpe. Notice that the RESOLUTION and LAYOUT INPUT EXCEPTION SEVERITY statements are not in the output.

Example 2-1. Input SVRF for LPE Script Generator

```
LAYOUT PATH      "test.oas"
LAYOUT PRIMARY   "*"
LAYOUT SYSTEM    OASIS
PRECISION        4000
RESOLUTION       1

LAYOUT MAGNIFY   AUTO

DRC RESULTS DATABASE "ml_nmopc.repair.oas" OASIS PSEUDO
DRC SUMMARY REPORT  "ml_nmopc_repair.sum"

DRC MAXIMUM RESULTS  ALL
LAYOUT ERROR ON INPUT YES
DRC MAXIMUM VERTEX   199
DRC KEEP EMPTY       YES
```

```
LAYOUT INPUT EXCEPTION SEVERITY POLYGON_DEGENERATE 1
LAYOUT INPUT EXCEPTION SEVERITY PRECISION_RULE_FILE 1
LAYOUT INPUT EXCEPTION SEVERITY MISSING_REFERENCE 1
LAYOUT RECOGNIZE DUPLICATE CELLS YES
LAYOUT ALLOW DUPLICATE CELL YES

layer M1_OPC_ORG_1 61

M1_OPC_ORG_1 { copy M1_OPC_ORG_1 } drc check map M1_OPC_ORG_1 61

LAYER PINCH_MARKER_1 237

M1_OPC = LITHO DENSEOPC FILE M1_TDOPC_RULE_1 M1_OPC_ORG_1 PINCH_MARKER_1
MAP M1_OPC_1

M1_OPC { copy M1_OPC } drc check map M1_OPC 62

LITHO FILE M1_TDOPC_RULE_1 [
  modelpath models
  optical_model_load OPT opt_4
  resist_model_load RESIST cm1_4.mod
  tilemicrons 25 adjust

  background atten 0.065
  layer M1_OPC_ORG_1 hidden clear
  layer PINCH_MARKER_1 hidden clear

  progress_meter off

  denseopc_options td_option {
    version 1
    algorithm 2

    background atten 0.065

    layer M1_OPC_ORG_1 opc clear
    layer PINCH_MARKER_1_size hidden clear

    image optical OPT dose 1 resist RESIST

    fragment_inter -interdistance 0.01 -ripplelen 1 -num 0 -shield 1
    fragment_min 0.04
    fragment_corner convex 1
    fragment_corner concave 1
    fragment_max 5.0

    mrc_rule external M1_OPC_ORG_1 { projecting use 0.015 euclidean \
      not_projecting use 0.013 euclidean length 0.02 use 0.015 0.01}
    mrc_rule internal M1_OPC_ORG_1 { projecting use 0.028 opposite \
      length 0.02 use 0.025 opposite not_projecting use 0.028 opposite}

    NEWTAG all M1_OPC_ORG_1 -out tgall

    ##pinch

    DISPLACEMENT tgall -fixedOffset 0.3
```

```
    }  
    setlayer M1_OPC_1 = denseopc M1_OPC_ORG_1 PINCH_MARKER_1 \  
                                MAP M1_OPC_ORG_1 OPTIONS td_option  
]
```

Example 2-2. Output SVRF From LPE Script Generator

```
//-----  
// LPE script generation starts here  
//-----  
LAYOUT PATH "m1_nmopc.repair.oas"  
LAYOUT PRIMARY "*"   
LAYOUT SYSTEM OASIS  
DRC RESULTS DATABASE "m1_nmopc.repair.oas.lpe" OASIS PSEUDO  
DRC SUMMARY REPORT "m1_nmopc_repair.sum.lpe"  
PRECISION 4000  
DRC MAXIMUM RESULTS ALL  
DRC MAXIMUM VERTEX 199  
LAYOUT ERROR ON INPUT YES  
LAYOUT ALLOW DUPLICATE CELLS YES  
LAYOUT RECOGNIZE DUPLICATE CELLS YES  
DRC KEEP EMPTY YES  
LAYOUT ULTRA FLEX YES  
LAYER M1_OPC_ORG_1 61  
LAYER PINCH_MARKER_1 240  
// Note: the above layer is not present in DRC CHECK MAPs of  
// the input rules file  
LAYER markers 243 // This is the marker layer for LPE  
LAYER M1_OPC__init 62 // init correction layer for LPE  
M1_OPC_ORG_1 {COPY M1_OPC_ORG_1} DRC CHECK MAP M1_OPC_ORG_1 61  
PINCH_MARKER_1 {COPY PINCH_MARKER_1} DRC CHECK MAP PINCH_MARKER_1 240  
// Note: the above check is not present in DRC CHECK MAPs of  
// the input rules file  
markers {COPY markers} DRC CHECK MAP markers 243  
M1_OPC__init {COPY M1_OPC__init} DRC CHECK MAP M1_OPC__init 62  
//M1_OPC = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers M1_OPC__init  
FILE LPE_setup.in MAP M1_OPC_1  
//hotspots = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers M1_OPC__init  
FILE LPE_setup.in MAP hotspots  
M1_OPC__corr = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers M1_OPC__init  
FILE LPE_setup.in MAP M1_OPC__init  
core_region = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers M1_OPC__init  
FILE LPE_setup.in MAP core_region  
context_region = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers  
M1_OPC__init FILE LPE_setup.in MAP context_region  
visible_region = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers  
M1_OPC__init FILE LPE_setup.in MAP visible_region  
repair_region = RET REFINE M1_OPC_ORG_1 PINCH_MARKER_1 markers  
M1_OPC__init FILE LPE_setup.in MAP repair_region  
//M1_OPC {COPY M1_OPC} DRC CHECK MAP M1_OPC 62  
M1_OPC__corr {COPY M1_OPC__corr} DRC CHECK MAP M1_OPC__corr 245  
core_region {COPY core_region} DRC CHECK MAP core_region 246  
context_region {COPY context_region} DRC CHECK MAP context_region 247  
visible_region {COPY visible_region} DRC CHECK MAP visible_region 248  
repair_region {COPY repair_region} DRC CHECK MAP repair_region 249  
  
//-----  
// LPE setup  
//-----  
LITHO FILE LPE_setup.in [/*  
layer M1_OPC_ORG_1 opc  
layer PINCH_MARKER_1 hidden
```

```
layer markers hidden
layer M1_OPC__init correction
refine_markers markers
#refine_stop_layer hotspots
refine_stop_iterations 1
min_frag_len 0.015
min_width 0.015
min_space 0.015
refine_regions update
denseopc_retargert yes
denseopc_insert_correction yes
denseopc_copy_fragments yes
denseopc_context_parameters
    displacement limit off feedback off
    displacement limit -0.012 0.012 feedback -keep -0.15
    displacement limit -0.007 0.007 feedback -keep -0.07
    displacement limit -0.003 0.003 feedback -keep -0.03
    displacement limit -0.001 0.001 feedback -keep -0.01
refine_exec M1_OPC = DENSEOPC M1_OPC_ORG_1 PINCH_MARKER_1 \
FILE M1_TDOPC_RULE_1 MAP M1_OPC_1
refine_map M1_OPC map M1_OPC_1
refine_map M1_OPC__init -from M1_OPC map M1_OPC__init
refine_map -region core map core_region
refine_map -region context map context_region
refine_map -region visible map visible_region
refine_map -region all map repair_region
*/]
//-----
// DENSEOPC setup
//-----
LITHO FILE M1_TDOPC_RULE_1 [/*
modelpath models
optical_model_load OPT opt_4
resist_model_load RESIST cm1_4.mod
tilemicrons 25 adjust
background atten 0.065
layer M1_OPC_ORG_1 hidden clear
layer PINCH_MARKER_1 hidden clear
progress_meter off
denseopc_options td_option {
version 1
algorithm 2
background atten 0.065
layer M1_OPC_ORG_1 opc clear
layer PINCH_MARKER_1_size hidden clear
image optical OPT dose 1 resist RESIST
fragment_inter -interdistance 0.01 -ripplelen 1 -num 0 -shield 1
fragment_min 0.04
fragment_corner convex 1
fragment_corner concave 1
fragment_max 5.0
mrc_rule external M1_OPC_ORG_1 { projecting use 0.015 euclidean
not_projecting use 0.013 euclidean length 0.02 use 0.015 0.01}
mrc_rule internal M1_OPC_ORG_1 { projecting use 0.028 opposite length 0.02
use 0.025 opposite not_projecting use 0.028 opposite}
NEWTAG all M1_OPC_ORG_1 -out tgal1
##pinch
DISPLACEMENT tgal1 -fixedOffset 0.3
```



```
}  
setlayer M1_OPC_1 = denseopc M1_OPC_ORG_1 PINCH_MARKER_1 MAP M1_OPC_ORG_1  
OPTIONS td_option  
*/]  
//-----  
// LPE script generation completed  
//-----
```

SVRF Commands

There are two SVRF commands used by Calibre LPE.

Table 2-1. Calibre LPE SVRF Commands

Command	Description
LITHO FILE	Specifies a Litho setup file in the SVRF rule file for use by Calibre Litho/RET tools.
RET REFINE	Invokes Calibre LPE.

LITHO FILE

Type: [SVRF Commands](#)

Specifies a Litho setup file in the SVRF rule file for use by Calibre Litho/RET tools.

Usage

LITHO FILE *name* '[' /*
 inline_file
 */ '['

Arguments

- *name*

A required name allowing the inline setup file to be referenced by [RET REFINE](#), LITHO DENSEOPC, and RET PXOPC statements. If there is more than one LITHO FILE statement in the file, no two LITHO FILE statements may use the same *name*.

- '['
 inline_file
 ']'

A required setup file placed between brackets ([]). Characters that occur within the brackets on the same line as the brackets are ignored. The comment characters (/* and */) are not part of the syntax, but are generally used to prevent the SVRF compiler from warning about the non-SVRF contents of the *inline_file*.

For Calibre LPE, the setup file has the general format of

```
LITHO FILE setup_name [  
# Layers  
layer ...  
  
# Parameters  
  
# Execution  
refine_markers...  
  
refine_exec...  
  
# Mapping  
refine_map...  
]
```

These commands are specific to Calibre LPE. A complete list is provided in “[Litho Setup File Commands](#)”. For the setup files for other tools, see the appropriate manual.

Calibre LPE runs do not support environment variables or SVRF variables within the setup file.

Examples

Example 1 — Basic Call

The following code shows an RET REFINE command calling the *lpe.in* setup file.

```
LITHO FILE lpe.in [  
    setup commands  
]  
  
RET REFINE layer1 layer2 layer3 FILE lpe.in MAP outlayer
```

Example 2 — Calling Other LITHO FILES From a Calibre LPE LITHO FILE

The Calibre LPE setup specifies various settings and then calls Calibre pxOPC with a Calibre pxOPC setup file. While these setup files do not need to be part of the SVRF rule file, the practice is recommended as it reduces errors.

In the following code, there are two LITHO FILE statements: one for Calibre LPE and one for Calibre pxOPC. The RET REFINE command calls *lpe.in*, which in turn calls *pxopc.in*.

```
LITHO FILE lpe.in [  
    ...  
    refine_exec x1 = pxopc target_layer file pxopc.in map out1  
    ...  
]  
  
LITHO FILE pxopc.in [  
    ...  
]  
  
RET REFINE layer1 layer2 layer3 FILE lpe.in MAP outlayer
```

Related Topics

[Litho Setup File Format for RET REFINE](#)

[Calibre nmOPC Users and Reference Manual](#)

[Calibre OPCverify Users and Reference Manual](#)

[Calibre pxOPC Users and Reference Manual](#)

RET REFINE

Type: [SVRF Commands](#)

Invokes Calibre LPE.

Usage

[EUV] **RET REFINE** *layer1 layer2 layer3* [*layer ...*] **FILE** {*filename* | *name*} **MAP** *output*

Arguments

- **EUV**

An optional keyword that specifies to use the EUV keyword as required for refine_exec operations. These will require the appropriate EUV licenses.

The EUV keyword forces “rearrangement_disable yes” and “refine_process flat” even if they are explicitly set in the Calibre LPE setup file.

- ***layer1 layer2 layer3*** [*layer ...*]

Required and optional layer names. You must specify at least three layer names, representing the initial OPC layer, the target layer, and the markers. These can be listed in any order; they are mapped to the setup file’s layer commands, which assign function.

Layers can be original layers, derived polygon layers, or edge layers.

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

Required keyword, followed by one of the following:

A *filename* indicating the path to the external setup 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*.

The *name* parameter of a LITHO FILE specification statement. The setup file is specified in the Litho File statement. This is the format used in all examples in this manual.

- **MAP** *output*

A required keyword, followed by the map name of a refine_map command from the setup file. The name specified after the refine_map map keyword and the name specified after the RET REFINE MAP keyword must be identical, including case. This layer is then mapped to the layer in the SVRF rule file for output.

Description

Specifies to run Calibre LPE during Calibre nmDRC-H. Calibre LPE takes as input an MRC-clean, post-OPC layout with hotspot markers, and returns an improved layout. It calls Calibre pxOPC, Calibre nmOPC, or Calibre nmBIAS to optimize the hotspots and merges the optimized regions smoothly with the rest of the layout.

This command can be specified any number of times in your rule file. Command instances that differ only by the **MAP *layername*** are run concurrently.

Examples

Example 1 — Layer Passing

The following example shows how layers are mapped between the RET REFINE command and the setup file. (The contents of the setup file show only the parts relevant to this example. A working setup file would also specify several other settings.)

```
m1_best = RET REFINE m1_opc m1_err m1_drawn FILE "setup.in" MAP L1
m1_best {COPY m1_best} DRC CHECK MAP OASIS m1_best 20
...
LITHO FILE setup.in [
    ...
    layer init_opc correction # m1_opc, because it is first
    layer markers hidden      # m1_err
    layer target opc          # m1_drawn
    ...
    refine_markers markers    # the "m1_err" of RET REFINE call

    refine_exec x1 = pxopc target ... # runs pxOPC on m1_drawn
    ...

    # Pass merged form back
    refine_map init_opc -from x1 map L1

    # These also pass a layer, but need a different RET REFINE MAP name
    # to receive the output.
    refine_map x1 map x1      # the pxOPC output
    refine_map target map target
]
```

The RET REFINE call has three layers: m1_opc, m1_err, and m1_drawn. These are passed in order to the three layer commands in the setup file. The layer commands provide names to use inside the setup file and also identify the layer function. After the layers have been acted upon by refine_markers and refine_exec, the refine_map command passes a layer back. The layer names after the map keyword in both refine_map and RET REFINE must match, like L1 in the example code.

Example 2 — Alternate Forms

In Example 1, the RET REFINE command assigns the output, L1, to a layer available to the rest of Calibre, m1_best. This makes it available for further processing. Alternatively, the RET REFINE operation might be enclosed in a rule check and have its output written directly to a GDS or results database, as shown here:

```
m1_best {RET REFINE m1_opc m1_err m1_drawn FILE "setup.in" MAP L1}
DRC CHECK MAP OASIS m1_best 20
```

Here, “m1_best” is a rule check name. While this name is given to the layer in the results database, the rest of the Calibre run cannot access it.

Related Topics

[LITHO FILE](#)

[layer](#)

[refine_markers](#)

[refine_exec](#)

[refine_map](#)

Litho Setup File Format for RET REFINE

Input for: Calibre LPE's RET REFINE command only.

The Litho setup file for Calibre LPE has some differences from a standard OPC Litho setup file, as described here. Litho setup files are used by most Calibre post-tapeout tools to specify local settings and encapsulate product-specific commands.

Format

A RET REFINE Litho setup file has the following formatting and syntax rules:

- All keywords are case insensitive. This does not apply to Litho setup files for other LITHO operations that may be in the same rule file.
- The file must contain only supported commands. In particular, an RET REFINE Litho setup file does not support environment variables or SVRF variables. This does not apply to Litho setup files for other LITHO operations.
- Carriage returns are ignored except for comments. Commands that span multiple lines do not require a line continuation character. This also is not typical for most Litho setup files.
- Comments begin with the pound sign (#) and run until the end of line. This is typical behavior.
- Litho setup files can be independent files or part of an SVRF rule file. When they are in an SVRF rule file, they must be enclosed in a LITHO FILE statement.

Parameters

Table 2-2. Required Commands for RET REFINE Setup Files

Keyword	Example Value	Description
layer	layer init_opc correction	Identifies layer name and function for use within the Litho setup file. There should be one instance for each layer passed in.
refine_exec	refine_exec check = or bridging pinching	Derives a new layer by running the specified operation. Operations include pxopc and opcoverify as well as DRC-like Boolean operations.
refine_map	refine_map -region all map fixed	Specifies the output.
refine_markers	refine_markers hotspots	Specifies the layer with markers identifying the regions to repair. Must appear only once.

For other commands that can be used in a Calibre LPE Litho setup file, see “[Litho Setup File Commands](#)” on page 35.

Examples

Example 1 — Minimum Calibre LPE Setup File

The setup file following includes all the required elements. It names the three input layers, identifies the layer containing the hotspots markers, and then calls Calibre pxOPC followed by Calibre OPCverify to perform the correction. The contour layer created by Calibre OPCverify is passed as output from the run.

```
layer target opc
layer init1 correction
layer markers hidden

refine_markers markers

refine_exec x1=pxopc target file pxopc.in map mx.pxopc.e1
refine_exec pvbandwidth = opcverify x1 file opcv.in map pvbandwidth

refine_map pvbandwidth map pvbandwidth
```

Most Calibre LPE runs involve many more refine_exec and refine_map statements.

Example 2 — Calibre LPE Setup File for Iterative Flow

Calibre LPE can be set to perform multiple correct-and-check passes, known as the iterative flow. The example below shows a larger Calibre LPE setup file that includes the commands for this.

```
layer target opc
layer init1 correction
layer markers hidden

refine_markers markers

refine_stop_iterations 3
refine_stop_layer hot_spots
refine_regions update

refine_exec x1 = pxopc target file pxopc.in file pxopc2.in \
    file pxopc3.in map mx.pxopc.e1

refine_exec violation_s = mrc x1 map spacing
refine_exec violation_h = mrc x1 map hole

refine_exec contour_nominal = opcverify x1 \
    file opcv.in map contour_nominal
refine_exec contour_pvband = opcverify x1 file opcv.in map contour_pvband
refine_exec bridging = opcverify x1 file opcv.in map bridging
refine_exec pinching = opcverify x1 file opcv.in map pinching
refine_exec pvbandwidth = opcverify x1 file opcv.in map pvbandwidth

refine_exec hot_spots = or bridging pinching violation_s violation_h
```

```
refine_map target map target
refine_map init1 -from x1 map ex1
refine_map x1 map x1
refine_map hot_spots map hot_spots
refine_map contour_nominal map contour_nominal
refine_map contour_pvband map contour_pvband
refine_map bridging map bridging
refine_map pinching map pinching
refine_map violation_s map violation_s
refine_map violation_h map violation_h
refine_map pvbandwidth map pvbandwidth

refine_map -region core map core_region
refine_map -region context map context_region
refine_map -region visible map visible_region
refine_map -region all map repair_region
```

Related Topics

[LITHO FILE](#)

[Litho Setup File Commands](#)

Litho Setup File Commands

This section lists commands that can appear in a Calibre LPE setup file.

A Calibre LPE setup file is referenced by the *name* argument to [RET REFINE](#). The file itself is typically contained within a [LITHO FILE](#) statement. The table lists the commands used by LPE.

For the rules governing an LPE setup file, see “[Litho Setup File Format for RET REFINE](#)”.

Table 2-3. Calibre LPE Litho Setup File Commands

Command	Description
denseopc_context_parameters	Recommended for nmOPC runs. Controls the gradient of correction in the transition region by creating zones with DISPLACEMENT -limit and FEEDBACK constraints on fragment movement.
denseopc_copy_fragments	Recommended for nmOPC runs. Specifies that fragment information should be copied from the opc layers to the correction layers for runs using Calibre nmOPC.
denseopc_fragment_coincident	Ensures fragmentation of long edges at region boundaries by adding fragment_coincident to the Calibre nmOPC setup file.
denseopc_fragment_nearly_coincident	Improves fragmentation on long lines.
denseopc_init_mode	Sets up the Calibre nmOPC run to use shapes from either the correction layers, the target layers, or intermediate result layers in the core region.
denseopc_insert_correction	Recommended for nmOPC runs. Converts a Calibre nmOPC setup file that operates on OPC layers to one that operates on correction layers.
denseopc_limit_excess	Recommended for nmOPC runs. Restricts notch fragments on the correction layer to a smaller displacement for Calibre nmOPC runs.
denseopc_pre_notchfill	Processes setlayer notchfill statements in a Calibre nmOPC setup file. Used with denseopc_copy_fragments.
denseopc_smooth_corrections	Recommended for nmOPC runs. Removes notches and nubs in the initial correction layer for Calibre nmOPC.
denseopc_use_correction_map	Specifies which Calibre nmOPC NEWTAG command to use when copying information from the opc layer to the correction layer.
fragment_optimize... reset_displacement	Recommended for nmOPC runs that use fragment_optimize. Sets fragment_optimize to only compute fragment information, and not move fragments.

Table 2-3. Calibre LPE Litho Setup File Commands (cont.)

Command	Description
<code>job 4 finetune</code>	A Calibre pxOPC job option that must be used instead of refine when invoking Calibre pxOPC from an RET REFINE run.
<code>layer</code>	Required. Specifies input layers that correspond to the layers in the RET REFINE command.
<code>min_frag_len</code>	Specifies the minimum fragment length for MRC.
<code>min_space</code>	Specifies the minimum spacing for MRC.
<code>min_width</code>	Specifies the minimum width for MRC.
<code>mrc_min_length</code>	Sets the minimum distance between two internal edges.
<code>mrc_min_rect_length</code>	Sets the minimum distance between two internal edges.
<code>mrc_min_rect_width</code>	Sets the minimum distance between two internal edges.
<code>mrc_min_square</code>	Sets the minimum distance between two internal edges.
<code>mrc_small_feature_size</code>	Removes features smaller than a specified size from Calibre pxOPC output. This should be set smaller than the <code>mrc_min_square</code> value.
<code>pick_region</code>	Restricts the LPE run to only the picked regions. Can be specified multiple times.
<code>pxopc_init_mode</code>	Sets up the Calibre pxOPC run to use shapes from either the correction layers or the opc layers in the core region.
<code>pxopc_transition_region</code>	Specifies an override for the size of the context region.
<code>pxopc_weight_layer</code>	Adjusts the weight to give a previous pxOPC output in the next iteration.
<code>rearrangement_disable</code>	Controls whether repair regions that overlap in the visible region are adjusted to separate them.
<code>refine_distance</code>	Specifies an override for the size of all regions.
<code>refine_exec</code>	Required. Calls the various operations to enhance printability in the marked regions. This command must be used instead of “setlayer pxopc” or “setlayer denseopc”.
<code>refine_map</code>	Required. Identifies the output to return from an RET REFINE command.
<code>refine_markers</code>	Required. Specifies the layer with hotspot markers. Must appear only once.
<code>refine_markers_limit</code>	Specifies the maximum number of markers to process. Suggested use is to prevent excessive runtime due to bad input.

Table 2-3. Calibre LPE Litho Setup File Commands (cont.)

Command	Description
refine_markers_style	Enables core regions to encompass large markers.
refine_patterns	Identifies repair regions with identical input and processes them as one instance. Requires refine_process hier.
refine_process	Preserves the layout hierarchy in repair regions.
refine_prune_markers	Controls what is included in the active region of iterative repairs.
refine_regions	Alters the repair region definition for multiple iterations.
refine_stop_iterations	Specifies how many iterations at most to perform. The default is 1.
refine_stop_layer	Identifies the refine_exec output layer that contains any hot spots after each iteration.
refine_update_layer	Copies the output from a refine_exec iteration to a specified layer for use in the next iteration when “ refine_regions update” is set.
refine_visible_sim	Checks the inner part of the visible region for new hotspots.
reject_duplicate_layers	Permits duplicate layer names in refine_exec operations.
smart_cut_disable	Controls whether additional compensation is done to avoid MRC violations.

denseopc_context_parameters

Type: [Litho Setup File Commands](#)

Recommended for nmOPC runs. Controls the gradient of correction in the transition region by creating zones with DISPLACEMENT -limit and FEEDBACK constraints on fragment movement.

Usage

denseopc_context_parameters {**displacement_limit** {**off** | *inner outer*} **feedback spec**}...

Arguments

- **displacement_limit** {**off** | *inner outer*}

A required argument that specifies values to use for the “DISPLACEMENT *tag* -limit” statement that Calibre LPE inserts in the Calibre nmOPC setup file. The tag set identifies the zone affected.

off — A required literal that causes the DISPLACEMENT line to be a comment.

inner outer — A required pair of arguments that constrains the displacement. See “[DISPLACEMENT](#)” in the *Calibre nmOPC User’s and Reference Manual*.

Only one of **off** and “*inner outer*” may be specified.

- **feedback spec**

A required argument that specifies a feedback factor used to determine edge movement. Calibre LPE inserts a “FEEDBACK [-keep] *tagvalue*...” statement in the Calibre nmOPC setup file for each zone. See “[FEEDBACK](#)” in the *Calibre nmOPC User’s and Reference Manual*.

The parameter *spec* may have one of four forms:

off — A required literal that causes the FEEDBACK line to be a comment.

[-keep] *value*... — An optional keyword and a required list of values specifying a feedback factor to use for each iteration. If you specify -keep, the last *value* will be used for iterations beyond the length of the list. Typically *value* is between -1 and 0.

auto — A required literal that is the equivalent of **duplicate 6 sub -0.001**.

duplicate num sub alt_value — A required keyword set that copies part of the feedback statement from the original Calibre nmOPC setup file to insert as a FEEDBACK statement. The syntax translates as the last *num* specifications of *value*, and use *alt_value* for any others.

For example:

```
feedback -0.5 -0.4 -0.3 -0.2 -0.1 #from original file
feedback duplicate 3 sub -0.8      #from denseopc_context_parameters
```

is the same as

```
feedback -0.5 -0.4 -0.3 -0.2 -0.1
feedback -0.8 -0.8 -0.3 -0.2 -0.1
```

Description

An optional (but strongly recommended) command used when executing Calibre nmOPC for the OPC portion. This option lets you control the “dumping strength” of the OPC correction. It can be useful to improve the convergence of the OPC process for all regions.

The repair flow primarily tries to repair geometry in the core region. Geometry in the visible region is frozen. Between core and visible is the context region, which you can further divide with this command. For each **displacement_limit/feedback** pair you specify, Calibre LPE creates a zone. Zones are ordered from next to the core region out to the visible region. To prevent corrections in the context region from degrading results in the visible region, restrict adjustments more in the outer zones.

The **displacement_limit** and **feedback** arguments can be specified multiple times, provided they are always specified together.

The recommended setting for the denseopc_context_parameters command is

```
denseopc_context_parameters
displacement limit off          feedback off
displacement limit -0.012 0.012 feedback -keep -0.15
displacement limit -0.007 0.007 feedback -keep -0.07
displacement limit -0.003 0.003 feedback -keep -0.03
displacement limit -0.001 0.001 feedback -keep -0.01
```

Examples

The following command divides the context region into five zones.

```
denseopc_context_parameters
displacement limit off feedback off
displacement limit -0.012 0.012 feedback -0.15
displacement limit -0.007 0.007 feedback -0.07
displacement limit -0.003 0.003 feedback -0.03
displacement limit -0.001 0.001 feedback -0.01
```

Calibre LPE expands this into the following lines, which are added to the Calibre nmOPC setup file:

```
NEWTAG topological not_outside_edge init1 zone_1 -out 1markers_frag_1
NEWTAG topological not_outside_edge init1 zone_2 -out 1markers_frag_2
NEWTAG topological not_outside_edge init1 zone_3 -out 1markers_frag_3
NEWTAG topological not_outside_edge init1 zone_4 -out 1markers_frag_4
NEWTAG topological not_outside_edge init1 zone_5 -out 1markers_frag_5

DISPLACEMENT 1markers_frag_5 -limit -0.001 0.001
DISPLACEMENT 1markers_frag_4 -limit -0.003 0.003
DISPLACEMENT 1markers_frag_3 -limit -0.007 0.007
DISPLACEMENT 1markers_frag_2 -limit -0.012 0.012
#DISPLACEMENT 1markers_frag_1 -limit -0.020 0.020
```

```
FEEDBACK -keep 1markers_frag_5 -0.01  
FEEDBACK -keep 1markers_frag_4 -0.03  
FEEDBACK -keep 1markers_frag_3 -0.07  
FEEDBACK -keep 1markers_frag_2 -0.15  
#FEEDBACK -keep 1markers_frag_1 -0.20
```


denseopc_copy_fragments

Type: [Litho Setup File Commands](#)

Recommended for nmOPC runs. Specifies that fragment information should be copied from the opc layers to the correction layers for runs using Calibre nmOPC.

Usage

denseopc_copy_fragments {**no** | **yes**}

Arguments

- **no**
A required argument indicating that fragment information is not copied. This is the default behavior.
- **yes**
A required argument indicating that fragment information should be copied from the layer of type opc to the layer of type correction.

Description

This optional statement can be useful for iterative repair flow. It is ignored if the refine_exec statement does not call Calibre nmOPC.

Setting **denseopc_copy_fragments yes** is recommended.

Related Topics

[refine_exec](#)

denseopc_fragment_coincident

Type: [Litho Setup File Commands](#)

Ensures fragmentation of long edges at region boundaries by adding `fragment_coincident` to the Calibre nmOPC setup file.

Usage

denseopc_fragment_coincident {no | yes}

Arguments

- **no**
A required argument that specifies that `fragment_coincident` is not to be added to the setup file. This is the default behavior.
- **yes**
A required argument that directs Calibre LPE to add `fragment_coincident` to the Calibre nmOPC setup file.

Description


This optional command directs Calibre LPE to insert a `fragment_coincident` statement at the end of a `fragment_layer` block as follows:

```
fragment_layer opc_layer {  
    ...  
    fragment_coincident correction_layer -overlay 0  
}
```

The Calibre nmOPC setup file must contain a `fragment_layer` block for the OPC layer or the run exits with the following message:

```
RET REFINE - denseopc_fragment_coincident is yes, but no fragment_layer is  
present
```

Note

 This command is only designed to work on OPC layers and is skipped for `fragment_layers` that use SRAF layers. In other words, if an SRAF layer is specified for `fragment_layer`, it will not have a `fragment_coincident` statement added to its block.

Examples

The following line enables long-line fragmentation at boundaries:

```
denseopc_fragment_coincident yes
```

denseopc_fragment_nearly_coincident

Type: [Litho Setup File Commands](#)

Improves fragmentation on long lines.

Usage

denseopc_fragment_nearly_coincident {no | yes}

Arguments

- **no**
A required argument that indicates that fragment_nearly_coincident is not to be added to the setup file. This is the default behavior.
- **yes**
A required argument that directs Calibre LPE to add fragment_nearly_coincident to the Calibre nmOPC setup file.

Description

This optional command directs Calibre LPE to increase the amount of fragmentation by interfeature fragmentation from the correction layer to the opc layer, and copying fragmentation from the opc layer to the correction layer. It can improve convergence when the initial geometries have long fragments.

The denseopc_fragment_nearly_coincident command should be used with denseopc_copy_fragments and denseopc_fragment_coincident like this:

```
denseopc_copy_fragments yes  
denseopc_fragment_coincident yes  
denseopc_fragment_nearly_coincident yes
```

denseopc_init_mode

Type: [Litho Setup File Commands](#)

Sets up the Calibre nmOPC run to use shapes from either the correction layers, the target layers, or intermediate result layers in the core region.

Usage

denseopc_init_mode { **target** | **org** | **last** | **target_last** }

Arguments

One of the following arguments must be set.

- **target**
The initial correction layer is the target layer inside the core regions and the original correction layer outside the core regions. This is the default behavior if the command is not set.
- **org**
The initial correction layer is the original correction layer across all repair regions.
- **last**
The initial correction layer of the first iteration is the original correction layer, as with **org**. Subsequent iterations use the previous iteration's result.
- **target_last**
The initial correction layer of the first iteration is the target layer inside the core regions and the original correction layer outside the core regions, as with **target**. Subsequent iterations use the previous iteration's result.

Description

The optional `denseopc_init_mode` command specifies the initial correction layer for `refine_exec ... denseopc` iterations. Certain commands, such as `refine_prune_markers`, require this command also be set.

Within the RET REFINE setup, target layers are identified by being in a “layer ... opc” statement and correction layers are identified by a “layer ... correction”, “layer ... sraf”, or “layer ... asraf” statement. Any of these layers may be the initial correction layer for the `refine_exec ... denseopc` iterations.

Related Topics

[Calibre LPE Terminology](#)

[layer](#)

[refine_exec](#)

[refine_prune_markers](#)

denseopc_insert_correction

Type: [Litho Setup File Commands](#)

Recommended for nmOPC runs. Converts a Calibre nmOPC setup file that operates on OPC layers to one that operates on correction layers.

Usage

denseopc_insert_correction {no | yes}

Arguments

- **no**
A required argument that does not convert the setup file. This behavior is the default.
- **yes**
A required argument that causes Calibre LPE to convert the Calibre nmOPC setup file.

Description

An optional command that directs Calibre LPE to convert a dense OPC setup file that is based on layers of type `opc` to one that is based on layers of type `correction`. To convert, Calibre LPE changes the following:

- Adds correction layers to the layer definitions.
- Adds the corresponding layer names to the `setlayer denseopc` statements.
- Substitutes the `opc` layer names with the correction layer counterparts in `image`, `pw_condition`, `mrc_rule`, and `OUTPUT_OPC` commands in the Calibre nmOPC setup file.
- Inserts the correction layer version of the tags that are based on the `opc` layer in `DISPLACEMENT`, `FRAGMENT`, and `FRAGMENT_MOVE` command.
- Inserts `fragment_layer CORRECTION_LAYER {fragment_inter off}` and `fragment_coincident`.

Setting **denseopc_insert_correction yes** is recommended.

Related Topics

[refine_exec](#)

[denseopc_use_correction_map](#)

[Calibre nmOPC Users and Reference Manual](#)

denseopc_limit_excess

Type: [Litho Setup File Commands](#)

Recommended for nmOPC runs. Restricts notch fragments on the correction layer to a smaller displacement for Calibre nmOPC runs.

Usage

denseopc_limit_excess {**no** | **yes**}

Arguments

- **no**
A required argument that does not change the displacement limit from the setup file. This behavior is the default.
- **yes**
A required argument that directs Calibre LPE to add restrictions to notch fragment movement.

Description

This optional command controls whether fragments on notches have additional displacement limits enforced.

When `denseopc_limit_excess` is set to `yes`, certain fragments have their OPC movement restricted to no more than 0.05 microns outward, and no inward movement. These fragments meet the following criteria:

- On the correction layer
- Length of `min_space`
- Depth of at least `min_space`
- In the context region, and at least 0.1 microns outside the core region

Fragments on a feature that meet these criteria have their displacement limited in the Calibre nmOPC setup file with an outer value of 0.05.

Setting **denseopc_limit_excess yes** is recommended.

Related Topics

[DISPLACEMENT \[Calibre nmOPC User's and Reference Manual\]](#)

denseopc_pre_notchfill

Type: [Litho Setup File Commands](#)

Processes setlayer notchfill statements in a Calibre nmOPC setup file. Used with `denseopc_copy_fragments`.

Usage

denseopc_pre_notchfill {no | yes}

Arguments

- **no**
A required argument that does not specially process setlayer notchfill statements. This is the default behavior.
- **yes**
A required argument that directs Calibre LPE to handle setlayer notchfill operations on target layers in such a way that the `denseopc_copy_fragments` command succeeds.

Description

An optional command that performs special handling of the setlayer notchfill statements in a dense OPC setup file. It has no affect if the dense OPC setup file does not contain setlayer notchfill statements.

When yes is specified, the following changes occur:

1. When setting up the initial correction layers, any setlayer notchfill statements are executed to produce the target layers.
2. The target layers are used for setting up the initial correction layers in the core region.
3. During the `refine_exec` denseopc operation,
 - If the full setup file contains setlayer `curve_target`, `curve_target` runs on the original target layers.
 - All commands from the setup file execute on the original targets (before notchfill) and the initial correction layers.

This procedure ensures that the `opc` and correction layers are coincident inside the core region, which then allows `denseopc_copy_fragments` to work successfully.

Related Topics

[denseopc_copy_fragments](#)

denseopc_smooth_corrections

Type: [Litho Setup File Commands](#)

Recommended for nmOPC runs. Removes notches and nubs in the initial correction layer for Calibre nmOPC.

Usage

denseopc_smooth_corrections {[no](#) | yes}

Arguments

- [no](#)
A required argument that retains the fragment placement provided to Calibre LPE. This is the default behavior.
- **yes**
A required argument that directs Calibre LPE to perform the smoothing process when preparing the initial correction layer.

Description

An optional command that adjusts the correction layer in the outer context region before executing Calibre nmOPC. It has no effect when re-OPC is performed with Calibre pxOPC.

Calibre LPE identifies notches and nubs in the core region and removes them. It identifies these unwanted features as those with length of min_space and depth of at least min_space (notches), or length of min_width and height of at least min_width or more (nubs). It adjusts the central fragment to be collinear with its adjacent neighbor(s). Fragments more than 0.1 micron from the core region are not adjusted. This option is ignored if denseopc_init_mode org is set.

Setting **denseopc_smooth_corrections yes** is recommended.

Related Topics

[min_space](#)

[min_width](#)

[denseopc_limit_excess](#)

[denseopc_init_mode](#)

denseopc_use_correction_map

Type: [Litho Setup File Commands](#)

Specifies which Calibre nmOPC NEWTAG command to use when copying information from the opc layer to the correction layer.

Usage

denseopc_use_correction_map {**no** | **yes**}

Arguments

- **no**
A required argument indicating to use NEWTAG enclose for copying. This is the default behavior.
- **yes**
A required argument indicating to use NEWTAG correction_map for copying.

Description

This optional keyword controls whether Calibre LPE uses the Calibre nmOPC command NEWTAG correction_map or NEWTAG enclose when copying tags from the opc layer to the correction layer. Copying tags to the correction layer only happens when [denseopc_insert_correction](#) yes is set.

Examples

```
denseopc_insert_correction yes  
denseopc_use_correction_map yes
```

Related Topics

[NEWTAG correction_map \[Calibre nmOPC User's and Reference Manual\]](#)

[NEWTAG enclose \[Calibre nmOPC User's and Reference Manual\]](#)

fragment_optimize... reset_displacement

Type: [Litho Setup File Commands](#) (denseopc_options only)

Recommended for nmOPC runs that use fragment_optimize. Sets fragment_optimize to only compute fragment information, and not move fragments.

Usage

fragment_optimize [*standard_options*] [reset_displacement]

Arguments

- *standard_options*

All supported fragment_optimize arguments are supported, as described in “[fragment_optimize](#)” in the *Calibre nmOPC User’s and Reference Manual*.

- reset_displacement

An optional argument that resets the displacement of all fragments to 0 when optimization is complete.

Note



Calibre LPE runs must specify reset_displacement.

Description

The fragment_optimize keyword optimizes model-based fragmentation on an OPC layer. When used without reset_displacement, it is likely to move fragments on the OPC layer, which can lead to incorrect results.

The LPE script generator automatically adds reset_displacement to any fragment_optimize commands it encounters in the input.

Related Topics

[calibre -lpe](#)

job 4 finetune

Type: [Litho Setup File Commands](#) (pxopc_options only)

A Calibre pxOPC job option that must be used instead of refine when invoking Calibre pxOPC from an RET REFINE run.

Usage

job 4 finetune

Arguments

None.

Description

When Calibre pxOPC is started from within an RET REFINE setup file, pxopc_options must use “job 4 finetune” instead of “job 4 refine.” If refine is used instead, the transcript includes the following errors:

```
PXOPC PARSING ERROR:
on line N in pxopc.setup
    PXOPC: job "refine" cannot be used in reopc mode

...

RET REFINE - parsing error for run time setup file for PXOPC at iteration

ERROR: Error LTH12 on line N of svrf - RET or MDP... operation FILE error.
```

For details on the job command, see the [Calibre pxOPC User's and Reference Manual](#). For the finetune job, these are the defaults:

- iterations: 10
- pw_select: all defined process conditions

The finetune job fills the same function as the refine job, fine-grained optimization for best EPE RMS and to remove extra printing. Like refine, it represents the mask as a contour.

Although the finetune job does not have any arguments, there are several pxopc_options keywords that may be specified immediately after it to change the defaults or global settings for the finetune job only.

Examples

```
LITHO FILE pxopc.in [
    ...
```

```
pxopc_options LPE-invoked {  
    # general options  
    background ...  
    layer ...  
  
    pw_condition COND1 ...  
    pw_condition COND2 ....  
  
    # job sections  
    job 1 open  
    job 2 decorate  
    job 3 correct  
    job 4 finetune  
        pw_select COND1 COND2  
    job 5 lmrc  
} # end of pxopc_options block  
  
setlayer opcout = pxopc ... OPTIONS LPE-invoked  
] # end of LITHO FILE
```

Related Topics

[pxopc_options Keywords \[Calibre pxOPC User's and Reference Manual\]](#)

layer

Type: [Litho Setup File Commands](#)

Required. Specifies input layers that correspond to the layers in the RET REFINE command.

Usage

layer *name type*

Arguments

- **name**

A required argument specifying a name to use within the setup file when referring to the layer.

- **type**

A required argument specifying the layer type. Must be one of the following:

correction — The initial OPC corrections, including SRAFs if using Calibre pxOPC for re-OPC. Shapes in this layer may be changed, and are also used in simulation.

hidden — Auxiliary layers, such as the marker layer. Layers of this type are neither changed nor simulated.

opc — The target layer. The shapes on the opc layer represent the actual geometry that you intend to transfer onto the wafer. All EPEs are calculated with respect to the opc layer, but it is not itself simulated.

sraf — Only valid when using Calibre nmOPC for re-OPC. Use for the layer containing SRAF shapes in the initial design. Layers of this type are corrected.

asraf — Only valid when using Calibre nmOPC for re-OPC. Use for the layer containing negative SRAF shapes in the initial design. Layers of this type are corrected.

retarget — Only valid when using both Calibre nmOPC and Calibre nmBIAS. The “refine_exec nmbias” setup file receives the retarget layer in place of the correction layer. The retarget layer should contain the initial layout’s pre-biased OPC target.

Description

A required command identifying the layers for the run. At least one each of types **correction** and **opc** must be specified. Layers are matched by the order in which they are passed in from the RET REFINE command; the names do not need to be the same.

Some lithography processes require additional layers. For example, double dipole lithography (DDL) requires two correction layers.

Note that unlike the layer command of most other LITHO tools, the form used by Calibre LPE does not require transmission information.

Examples

Example 1 — Passing Layers

The following code shows a simple set of layers. Notice that the initial layer is named “init_shapes” in the SVRF section but “init1” in the LPE setup file because layers are matched by order.

```
//SVRF Layer operation
LAYER target      45
LAYER init_shapes 100
LAYER markers     200

//A Calibre LPE operation
contour = RET REFINE target init_shapes markers FILE setup MAP nominal

LITHO FILE setup [/*
layer target    opc
layer init1     correction
layer markers   hidden
...
*/ ]
```

Example 2 — Retargeting

Both Calibre nmOPC and Calibre nmBIAS setups require a correction layer, which can be modified by the repairs. To keep the pre-biasing target and the post-biasing/pre-OPC results separate, use a retarget layer as the correction layer for refine_exec ... nmbias.

```
# LPE setup file
layer CONTACT_RETARGET retarget
layer CONTACT_TGT      opc
layer marker           hidden
layer CONTACT_OPC      correction

...
# Notice CONTACT_RETARGET is passed automatically, not CONTACT_OPC
refine_exec x1_nmbias = nmbias CONTACT_TGT FILE nmbias.in MAP preclean

...
refine_exec x1_reopc = denseopc x1_nmbias marker FILE nmopc.in \
                        MAP CONTACT_OPC

refine_map CONTACT_RETARGET -from x1_nmbias  map et1
refine_map CONTACT_OPC     -from x1_reopc    map ex1
refine_map x1_nmbias        map x1_nmbias
refine_map x1_reopc         map x1
```

```
# nmbias.in (Calibre nmBIAS setup file)
...
layer le_bias_all hidden
options opt {
    layer le_bias_all opc
    ...}
...
setlayer preclean = nmbias le_bias_all MAP le_bias_all OPTIONS opt
...

# nmopc.in (Calibre nmOPC setup file)
...
layer CONTACT_TGT visible clear 0 1.0
layer marker hidden clear 0 1.0
...
denseopc_options opc_opt {
    ...
    layer CONTACT_TGT opc clear
    layer marker hidden clear
    ...}
...
setlayer CONTACT_OPC = denseopc CONTACT_TGT marker MAP ... OPTIONS opc_opt
```

Related Topics

[Calibre LPE Terminology](#)

[LITHO FILE](#)

[RET REFINE](#)

min_frag_len

Type: [Litho Setup File Commands](#)

Specifies the minimum fragment length for MRC.

Usage

min_frag_len *value*

Arguments

- *value*

A required argument specifying the minimum fragment length for MRC operations in user units, typically microns.

Description

This is an optional command that may appear at most once in a Calibre LPE setup file, but must appear at least once in either the Calibre LPE setup file, or as `mrc_min_edge` in the setup file of the operations called by [refine_exec](#).

The following rules apply for determining the default value:

1. If the value is in the Calibre LPE setup file, it takes precedence.
2. If the value is *not* in the Calibre LPE setup file, it is set equal to `mrc_min_edge` in the called setup file. If there are multiple setup files for that particular operation, the last setup file's value is used.
3. If neither the Calibre LPE or the operation setup file specifies the value, the run exits with the following error message:

```
RET REFINE - min_frag_len not specified

ERROR: Error LTH12 on line N of rules - RET or MDP... operation FILE
error.
```

The setting used for `min_frag_len` is printed in the transcript section “RET REFINE - setting summary”.

Related Topics

[min_space](#)

[min_width](#)

[mrc_min_edge](#) [Calibre pxOPC User's and Reference Manual]

min_space

Type: [Litho Setup File Commands](#)

Specifies the minimum spacing for MRC.

Usage

min_space *value*

Arguments

- *value*

A required argument specifying the minimum spacing for MRC operations in user units, typically microns.

Description

This is an optional command that may appear at most once in a Calibre LPE setup file, but must appear at least once in either the Calibre LPE setup file, or as `mrc_min_external` in the setup file of the operations called by [refine_exec](#).

The following rules apply for determining the default value:

1. If the value is in the Calibre LPE setup file, it takes precedence.
2. If the value is *not* in the Calibre LPE setup file, it is set equal to `mrc_min_external` in the called setup file. If there are multiple setup files for that particular operation, the last setup file's value is used.
3. If neither the Calibre LPE or the operation setup file specifies the value, the run exits with the following error message:

```
RET REFINE - min_space not specified

ERROR: Error LTH12 on line N of rules - RET or MDP... operation FILE
error.
```

The setting used for `min_space` is printed in the transcript section “RET REFINE - setting summary”.

Related Topics

[min_frag_len](#)

[min_width](#)

[mrc_min_external](#) [Calibre pxOPC User's and Reference Manual]

min_width

Type: [Litho Setup File Commands](#)

Specifies the minimum width for MRC.

Usage

min_width *value*

Arguments

- *value*

A required argument specifying the minimum width for MRC operations in user units, typically microns.

Description

This is an optional command that may appear at most once in a Calibre LPE setup file, but must appear at least once in either the Calibre LPE setup file, or as `mrc_min_internal` in the setup file of the operations called by [refine_exec](#).

The following rules apply for determining the default value:

1. If the value is in the Calibre LPE setup file, it takes precedence.
2. If the value is *not* in the Calibre LPE setup file, it is set equal to `mrc_min_internal` in the called setup file. If there are multiple setup files for that particular operation, the last setup file's value is used.
3. If neither the Calibre LPE or the operation setup file specifies the value, the run exits with the following error message:

```
RET REFINE - min_width not specified

ERROR: Error LTH12 on line N of rules - RET or MDP... operation FILE
error.
```

The setting used for `min_width` is printed in the transcript section “RET REFINE - setting summary”.

Related Topics

[min_frag_len](#)

[min_space](#)

[mrc_min_internal](#) [Calibre pxOPC User's and Reference Manual]

mrc_min_length

Type: [Litho Setup File Commands](#)

Sets the minimum distance between two internal edges.

Usage

mrc_min_length *min_length_val*

Arguments

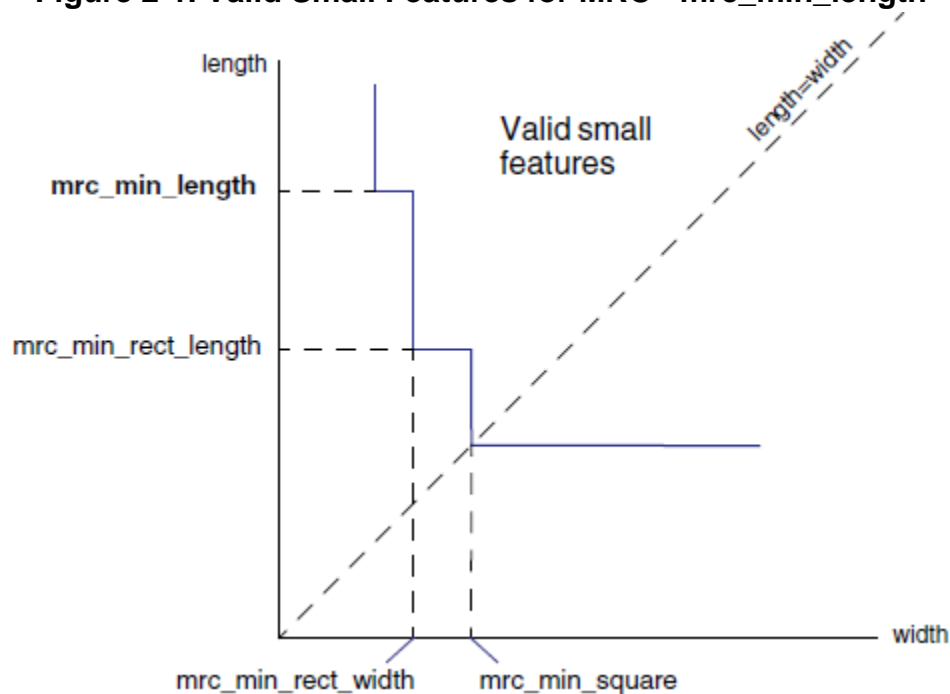
- *min_length_val*

A required argument specifying a minimum distance in microns between the shorter facing edges of a single polygon during MRC. The default is 0.015 um. If the polygon is square, the pair are chosen arbitrarily.

Description

An optional command that specifies the distance between two internal edges of a shape. Use this control for secondary features such as SRAFs.

Figure 2-1. Valid Small Features for MRC - mrc_min_length



Related Topics

[mrc_min_rect_length](#)

[mrc_min_rect_width](#)

[mrc_min_square](#)

[mrc_small_feature_size](#)

mrc_min_rect_length

Type: [Litho Setup File Commands](#)

Sets the minimum distance between two internal edges.

Usage

mrc_min_rect_length *min_val*

Arguments

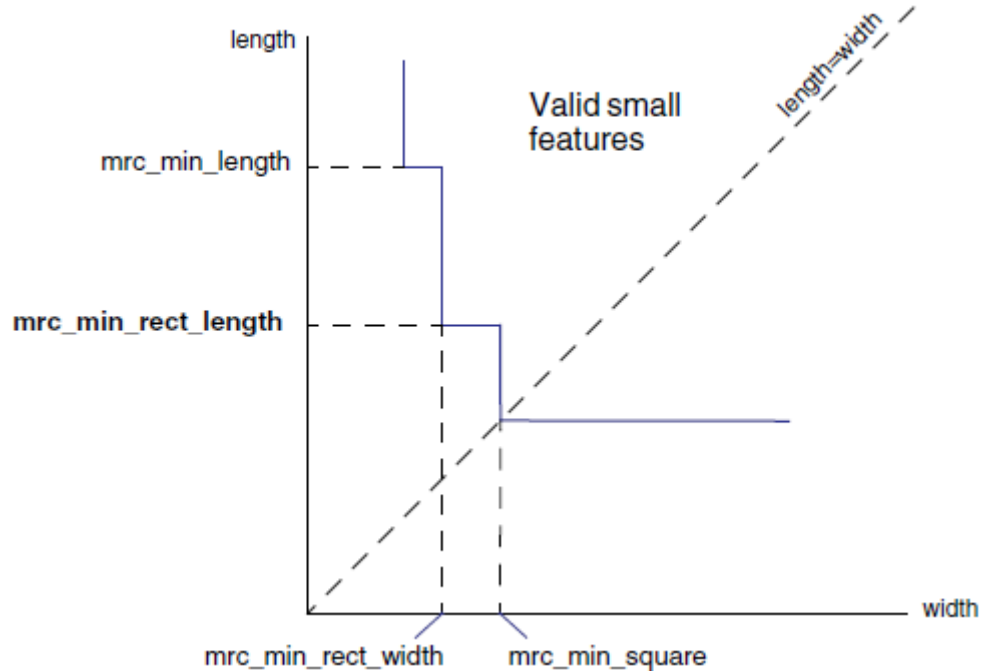
- *min_val*

A required argument specifying a minimum distance in microns between the shorter facing edges of a single polygon during MRC. The default is 0.015 um.

Description

An optional command that specifies the distance between two internal edges of a shape. Use in conjunction with [mrc_min_rect_width](#) for controlling secondary features such as SRAFs. The length is the longer edge of a rectangular SRAF. When the length and width are equal, [mrc_min_square](#) applies.

Figure 2-2. Valid Small Features for MRC - mrc_min_rect_length



Related Topics

[mrc_min_length](#)

[mrc_small_feature_size](#)

mrc_min_rect_width

Type: [Litho Setup File Commands](#)

Sets the minimum distance between two internal edges.

Usage

mrc_min_rect_width *min_val*

Arguments

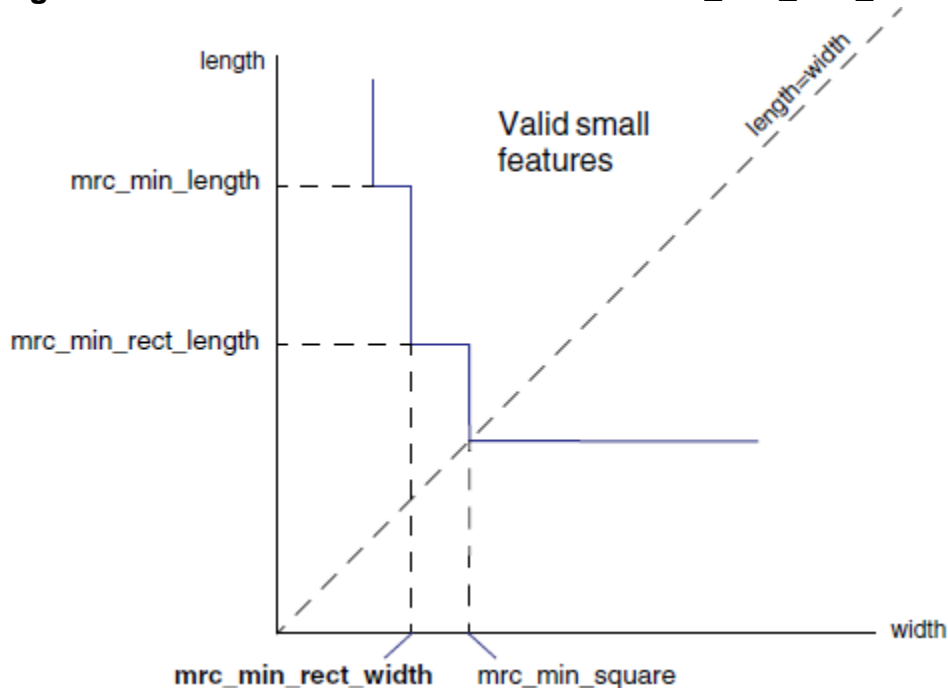
- *min_val*

A required argument specifying a minimum distance in microns between the longer facing edges of a single polygon during MRC. The default is 0.015 um.

Description

An optional command that specifies the distance between two internal edges of a shape. Use in conjunction with [mrc_min_rect_length](#) to control small secondary features such as SRAFs.

Figure 2-3. Valid Small Features for MRC - mrc_min_rect_width



Related Topics

[mrc_min_length](#)

[mrc_min_rect_length](#)

[mrc_min_square](#)

[mrc_small_feature_size](#)

mrc_min_square

Type: [Litho Setup File Commands](#)

Sets the minimum distance between two internal edges.

Usage

mrc_min_square *min_square_val*

Arguments

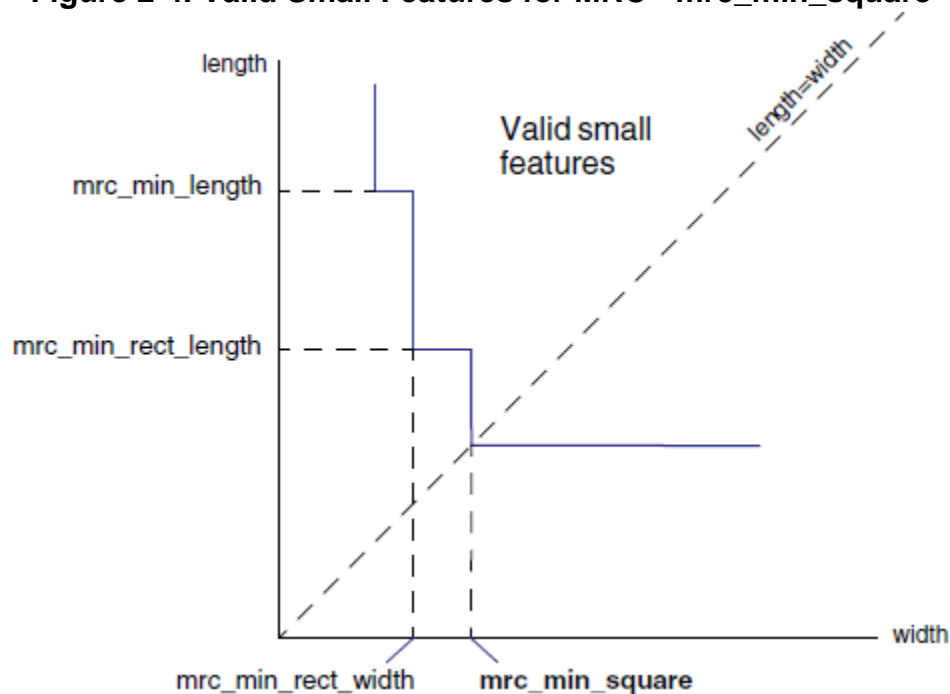
- *min_square_val*

A required argument specifying a minimum distance in microns between facing edges of a square during MRC. The default is 0.015 um.

Description

An optional command that specifies the distance between two internal edges of a shape. This value should be smaller than [mrc_min_length](#). Use to control small secondary features such as SRAFs. For primary square features such as contacts, use [min_width](#).

Figure 2-4. Valid Small Features for MRC - mrc_min_square



Related Topics

[mrc_min_length](#)

[mrc_min_rect_length](#)

[mrc_min_rect_width](#)

[mrc_small_feature_size](#)

mrc_small_feature_size

Type: [Litho Setup File Commands](#)

Removes features smaller than a specified size from Calibre pxOPC output. This should be set smaller than the `mrc_min_square` value.

Usage

mrc_small_feature_size *min_val*

Arguments

- *min_val*

A required argument specifying a minimum distance in microns between facing edges of a single polygon during MRC. The default is 0.015 um.

Description

An optional command that specifies the distance between two internal edges of a shape. Features returned by [refine_exec](#) pxopc that are smaller than *min_val* are not processed; they disappear from the output. The default value is 0.015 microns.

This value can be set in either the LPE or a pxOPC setup file.

1. If the value is in the Calibre LPE setup file, it takes precedence.
2. If the value is *not* in the Calibre LPE setup file, it is set equal to the pxOPC setup file's `mrc_small_feature_size` value. If there are multiple setup files for that particular operation, the last setup file's value is used.
3. If it is not specified in either the LPE or a pxOPC setup file, 0.015 um is used throughout.

The setting used for `mrc_small_feature_size` is printed in the transcript section “RET REFINE - setting summary”.

To determine a good *min_val*, use these guidelines:

- Do not exceed the value of [mrc_min_square](#).
- Add the following line to the pxOPC setup file:

```
setlayer finetune.out = pxopc in_layer... MAP layer \  
    OPTIONS pxopc.options LASTJOB 4
```

The LASTJOB number should be the same as that of the “[job 4 finetune](#)” command, if you have a modified job set.

Examine the SRAF sizes in the finetune.out layer and determine the minimum SRAF size to keep based on litho quality. Generally, the SRAFs closest to the main feature

should be kept as they have the largest contribution to litho quality. Use this minimum for *min_val*.

Related Topics

[mrc_min_length](#)

[mrc_min_rect_length](#)

[mrc_min_rect_width](#)

[mrc_min_square](#)

pick_region

Type: [Litho Setup File Commands](#)

Restricts the LPE run to only the picked regions. Can be specified multiple times.

Usage

pick_region *x y*

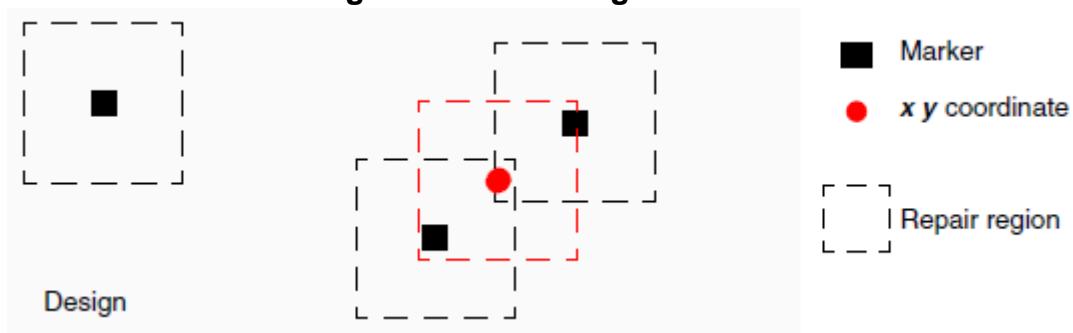
Arguments

- *x y*
A required argument specifying a layout coordinate in dbu.

Description

This optional command directs LPE to run only on markers that intersect with a repair region constructed around the *x y* coordinates. A coordinate can select multiple markers, as shown in [Figure 2-5](#). (Both black markers interacting with the red repair region are selected.) When multiple `pick_region` statements occur, the results are additive. All selected markers are used.

Figure 2-5. Selecting Markers



After markers are selected, the run proceeds as normal. LPE constructs repair regions around the selected markers. The temporary repair region around *x y* is not used.

Note

Also include “[refine_process flat](#)” in the setup file. The `pick_region` coordinates are interpreted as top-level coordinates.

This command can be particularly useful for reducing test cases for debugging.

pxopc_init_mode

Type: [Litho Setup File Commands](#)

Sets up the Calibre pxOPC run to use shapes from either the correction layers or the opc layers in the core region.

Usage

pxopc_init_mode {**org** | **target**}

Arguments

- **org** | **target**

A required argument specifying the layer by type to use as the initial layer within the core region. The area outside the core region always uses the layer of type “correction”.

org — This is the default behavior. The correction layer(s) are used.

target — The initial layer(s) inside the core region are of type “opc”. The area outside the core region uses the original correction shapes.

Description

Although the default is **org** (use correction layers), the **target** setting frequently produces better results, especially when doing multiple iterations. As described in the layer command, the shapes you want to print are on the opc layer while the prior OPC output is on the correction layer.

Related Topics

[Calibre LPE Terminology](#)

[layer](#)

pxopc_transition_region

Type: [Litho Setup File Commands](#)

Specifies an override for the size of the context region.

Usage

pxopc_transition_region *value*

Arguments

- *value*

A required argument in user units (typically microns) indicating the halo size around the core region. The value must be positive. The default is 10 times the larger of [min_frag_len](#) and [min_width](#).

Description

An optional command that may appear at most once in a Calibre LPE setup file. It adjusts the area considered for Calibre pxOPC repairs to the initial input.

As explained in “[Calibre LPE Terminology](#)” on page 11, Calibre LPE partitions the layout into four zones: a core region, a context region, a visible region, and the rest of the design. You can output these regions using syntax 2 of [refine_map](#). This variable sets the size of the context region, which is used to transition between the re-optimization of the core shapes and the original optimization in the visible region. This also shifts the visible region outward.

Generally it is recommended to accept the default value of the repair region or set it slightly larger, about the size of the optical radius. Do not go below $3 \times \text{mrc_min_edge}$ or $3 \times \text{min_frag_len}$.

Related Topics

[refine_distance](#)

[mrc_min_edge](#) [[Calibre pxOPC User's and Reference Manual](#)]

pxopc_weight_layer

Type: [Litho Setup File Commands](#)

Adjusts the weight to give a previous pxOPC output in the next iteration.

Usage

pxopc_weight_layer *layer weight*

Arguments

- **layer**
A required argument specifying the name of a layer derived by a [refine_exec](#) command. The **layer** must match the refine_exec **layer_out** exactly.
- **weight**
A required argument specifying the weight to use for the layer in the next pxOPC run.

Description

This command is for iterative repair flows. It directs Calibre LPE to save the specified refine_exec output from the previous iteration and add it to the current iteration's pxOPC setup as a weight layer.

Although you can use any refine_exec output layer as a weight layer, it is most useful to use the OPCverify output.

For the first iteration, the weight layer for the pxOPC operation is empty.

The pxOPC setup file does not need to explicitly include the weight layer. It is added at run time by Calibre LPE.

Examples

The following Calibre LPE setup runs Calibre pxOPC on a target layer. The output from the OPCverify phase is passed to the second pxOPC iteration with a weight of 20.

```
layer target opc
layer init1 correction
layer markers hidden

refine_stop_layer hotspots
refine_stop_iterations 2

pxopc_weight_layer w 20

refine_exec x1 = pxopc target file pxopc1.in file pxopc2.in map x1
refine_exec hotspots = opcverify x1 file opcv.in map hotspots
refine_exec w = opcverify x1 file opcv.in map w
```

Related Topics

[refine_stop_iterations](#)

[refine_stop_layer](#)

rearrangement_disable

Type: [Litho Setup File Commands](#)

Controls whether repair regions that overlap in the visible region are adjusted to separate them.

Usage

rearrangement_disable {yes | no}

Arguments

- **yes** | **no**

A required argument specifying whether to rearrange the internal representation of regions.

yes — Do not perform the rearrangement step.

no — Rearrange the internal representation of repair regions that overlap in the visible region. This is the default.

Description

The optional `rearrangement_disable` command controls whether Calibre LPE performs rearrangement when setting up regions for re-OPC. The rearrangement process internally relocates the repair regions to separate those repair regions that overlap in the visible region.

The typical Calibre LPE run takes a relatively small number of markers. Time spent on setting up repair regions is negligible compared to time spent in `refine_exec`. However, if the markers are such that the majority of the design area is covered with repair regions, the setup time can be prohibitive. To alleviate slow setup time, you can turn off region rearrangement.

The other case in which the setup file should disable region rearrangement is when the optical model is EUV. EUV models include flare maps, which are location dependent.

Setting `rearrangement_disable yes` also sets the following, potentially overriding explicit settings:

- [refine_visible_sim](#) no
- [denseopc_smooth_corrections](#) no
- [denseopc_limit_excess](#) no

Note that `rearrangement_disable yes` can change the repair region count in the transcript. By default, each separated region is counted individually. When the overlapping regions are not rearranged, they count as a single region.

refine_distance

Type: [Litho Setup File Commands](#)

Specifies an override for the size of all regions.

Usage

Syntax 1

refine_distance *value*

Syntax 2

refine_distance0 *value*

refine_distance1 *value*

refine_distance2 *value*

Arguments

- *value*

A required value in microns specifying the size of the core, context, and visible regions.

The visible region size must be greater than or equal to the largest optical radius of the optical models used in all operations called by [refine_exec](#). The optical radius is half the model's hoodpix parameter.

The default values are as follows:

refine_distance0: optical radius

refine_distance1: optical radius


refine_distance2: optical radius plus 0.2 um, or (EUV) optical radius plus 0.05 um

Description

An optional command or set of commands that specifies the sizes for the core, context, and visible regions. Use `refine_distance` (Syntax 1) to set a single size to be used for each region, or Syntax 2 to set all three individually. [Figure 2-6](#) shows the mapping between numbers and regions and how *value* is applied.

You can verify the settings your run uses by either checking the transcript or outputting the regions. The transcript shows distance0, distance1, and distance2 settings in the “RET REFINE - setting summary” section. To view the regions in the results, use `refine_map -region`, as shown in the Examples.

Note

 If you set the distances in such a way that the repair region is larger than the OPC tile size, Calibre LPE issues the following warning:

```
RET REFINE Warning: a large repair region detected in the following
coordinate window in the cell <cellname>:
<bottom left coordinate> -> <upper right coordinate>
```

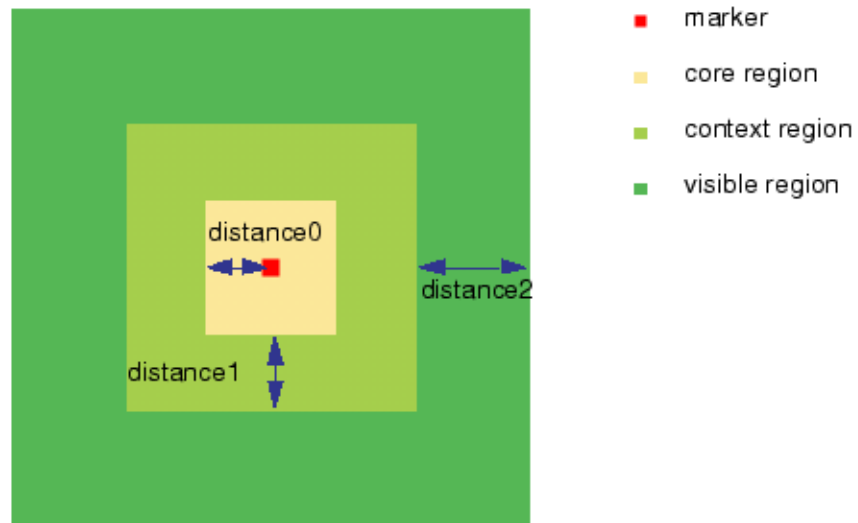
It is recommended to keep the sum of distance0, distance1, and distance2 less than the tilemicrons settings of all referenced setup files for faster processing.

A setup file may contain both `refine_distance` and `pxopc_transition_region`. The transition region size is set by `pxopc_transition_region`. If `refine_distance1` is explicitly specified, it must set the same value as `pxopc_transition_region` or the run exits with the following error:

```
RET REFINE - distance1=value1; distance1 must be the same as repairRegion
(value2) for the PXOPC-only mode
```

where *value2* is the value set by `pxopc_transition_region`.

Figure 2-6. Region Size Measurements



Examples

These lines set the different distances:

```
refine_distance0 1.5
refine_distance1 2.0
refine_distance2 1.5
```

The same effect can be achieved using

```
refine_distance 1.5  
pxopc_transition_region 2.0
```

To see the regions in the resulting layout, add these lines to the LPE setup file:

```
refine_map -region core map core_region      #distance0  
refine_map -region context map context_region #distance1  
refine_map -region visible map visible_region #distance2
```

In the SVRF file, copy the layers to the rule file. The layer names in the MAP arguments of RET REFINES and refine_map must match.

```
core_region = RET REFINES input1 input2... FILE setup MAP core_region  
core_region {COPY core_region} DRC CHECK MAP core_region 400
```

The lines for context_region and visible_region are similar.

Related Topics

[Calibre LPE Terminology](#)

[refine_map](#)

refine_exec

Type: [Litho Setup File Commands](#)

Required. Calls the various operations to enhance printability in the marked regions. This command must be used instead of “setlayer pxopc” or “setlayer denseopc”.

Usage

Syntax 1 (Primary Operations):

refine_exec *layer_out* = *primary_operation* *layer_in* ... {*file filename*}... **map** *mapname*

Syntax 2 (Secondary Operations):

refine_exec *layer_out* = *secondary_operation* *layer_in* ... [map *operation_option*]

Arguments

- **layer_out**

A required argument specifying a name for the layer created by the refine_exec command. The **layer_out** name can then be used in other commands in the RET REFINE setup file.

- **primary_operation**

For Syntax 1, a required argument specifying the Calibre product to execute. Supported values are

denseopc — Executes Calibre nmOPC. See “[Additional Information on denseopc Setups](#)” on page 78.

nmbias — Executes Calibre nmBIAS. See “[Additional Information on nmbias Setups](#)” on page 78.

opcverify — Executes Calibre OPCverify. See “[Additional Information on opcverify Setups](#)” on page 79.

pxopc — Executes Calibre pxOPC. See “[Additional Information on pxopc Setups](#)” on page 79.

svrf — Executes a modified SVRF rule file. See “[Additional Information on svrf Setups](#)” on page 79.

- **secondary_operation**

For Syntax 2, a required argument specifying the operation to execute. Supported values are

and — Performs a Boolean AND on the polygons. At least two **layer_in** layers must be specified.

copy — Copies a single layer. This operation does not accept multiple **layer_in** layers.

interact — Selects all polygons from the first **layer_in** that either share some or all of their area with a polygon from the second **layer_in**. This includes coincident edge segments. If either input layer is empty, there is no output. Exactly two **layer_in** layers must be specified.

interact ... map not — Returns the complement of the **interact** operation: only polygons from the first *layer_in* that do not touch or overlap polygons from the second *layer_in*.

mrc — Performs a pre-defined set of MRC checks. This operation does not accept multiple *layer_in* layers. The *layer_out* shapes can be restricted using the map option.

The MRC checks can be described with the SVRF following. The VIOLATIONS rule check corresponds to output from this operation.

```
W = INT layer_in < min_width REGION EXTENTS
S = EXT layer_in < min_space REGION EXTENTS

SHORT_FRAG = LENGTH layer_in < min_frag_len min_frag_len
X = INT [SHORT_FRAG] < min_frag_len INTERSECTING ONLY ABUT==90
Y = EXT [SHORT_FRAG] < min_frag_len INTERSECTING ONLY ABUT==90
Z = (EXPAND EDGE X OUTSIDE BY 0.005 CORNER FILL) OR
    (EXPAND EDGE Y INSIDE BY 0.005 CORNER FILL)

H = (HOLES layer_in INNER EMPTY) NOT INTERACT
    (HOLES target INNER EMPTY)
VIOLATIONS {W COPY S COPY Z COPY H}
```

The following values are supported for “map operation_option” with **mrc**:

- **map hole** — Return only the violations corresponding to layer H.
- **map spacing** — Return only the violations corresponding to layers W and S.

If no map argument is supplied, *layer_out* receives all three types of violations.

not — Performs a Boolean NOT on the polygons. Two *layer_in* layers must be specified.

or — Performs a Boolean OR on the polygons. At least two *layer_in* layers must be specified.

The *primary_operation* keywords of Syntax 1 are also considered operations for the remaining argument discussions.

- **layer_in**

A required argument that passes in the named layer or layers to the operation. At least one layer must be specified. The operation receives the layers in the order listed, which can affect results.

Layers are *assigned by order* to layer commands in the setup file, and are not matched by name.

- **file filename**

For Syntax 1, a required argument specifying a setup file for the primary operations. If it appears in Syntax 2, it is ignored. Calibre searches for *filename* first as a LITHO FILE in the SVRF file and then as a separate file.

When `refine_stop_iterations` is set to a value greater than 1 and the operation is **svrf**, **pxopc** or **denseopc**, the **file** argument must be specified as many times as the iterations. For example:

```
refine_stop_iterations 2
refine_exec final = pxopc target file pxopc.1 file pxopc.2 map final
```

- **map *mapname***

For Syntax 1, a required argument that indicate which layer created by the *operation* should be assigned to *layer_out*. The value of *mapname* must match one of the layer names generated by a setlayer operation in the *filename*, or by a DRC check for **svrf**.

For Syntax 2, an optional argument for the **interact** and **mrc** operations, as described in the information for those keywords.

This argument may be specified only once per statement. To assign multiple layers from a single *operation*, use multiple **refine_exec** statements that are identical except for *layer_out* and *mapname*. The statements are executed concurrently.

Description

A required command that may appear multiple times in a Calibre LPE setup file. It specifies the execution body of the repair flow operations.

The `refine_exec` command must be used to invoke the primary operations instead of setlayer commands because Calibre LPE is a driver that executes these operations for constructed regions around the markers. In contrast, the setlayer commands are tile-level operations and read in the entire layout extent.

Additional Information on denseopc Setups

The setup file may contain Tcl and TVF except in the options block. This allows the use of loop constructs for creating concurrent setlayer calls.

Additional Information on nmbias Setups

The setup file does not need to explicitly identify the core, context, visible, or prebiased layers; this is handled automatically. The drawn layer should be passed in the `refine_exec` call.

If the setup file does not include **BIASRULE**, the pre-OPC results are not available in the context and visible regions. This prevents LPE results from being merged smoothly into the rest of the design and could cause new errors.

For **nmbias** operations on double patterning layouts, there must be two OPC layers and two correction layers. The layers in the LPE setup file must be listed in order of exposure, like so:

```
layer drawn0 opc
layer drawn1 opc
layer prebiased0 correction
layer prebiased1 correction
```

In the Calibre nmBIAS setup, the OPC layers and the BIASRULE operation need to also be specified in order of exposure for proper mapping:

```
LITHO FILE nmbias.in [
    layer drawn0 hidden clear
    layer drawn1 hidden clear

    options td_option {
        layer drawn0 opc clear pattern 0
        layer drawn1 opc clear pattern 1
    }
    ...

    BIASRULE pattern 0 ... -table {...} -interpolate no
    pattern 1 ... -table {...} -interpolate no
```

The setup file may contain Tcl. This allows the use of loop constructs for creating concurrent setlayer calls. Unlike the other Tcl-supporting setup files, the nmbias setup file may not contain TVF.

Additional Information on opcverify Setups

If the command generating the marker layer uses a classify block, be sure the classify block includes “keep_no_context duplicates”. The default is to suppress duplicates, which results in only one version of a repeated violation being passed to Calibre LPE.

The setup file may contain Tcl and TVF. This allows the use of loop constructs for creating concurrent setlayer calls.

Additional Information on pxopc Setups

The setup file may contain Tcl and TVF except in the options block. This allows the use of loop constructs for creating concurrent setlayer calls.

Additional Information on svrf Setups

With the **svrf** keyword, refine_exec can run a subset of SVRF operations contained in a LITHO FILE setup file.

Table 2-4. Supported SVRF Operations

AND	ANGLE
AREA	[NOT] COINCIDENT EDGE
[NOT] COINCIDENT INSIDE EDGE	[NOT] COINCIDENT OUTSIDE EDGE
CONVEX EDGE	COPY
ENCLOSURE	EXPAND EDGE
EXTENTS	EXTERNAL
HOLES	[NOT] INSIDE
[NOT] INSIDE EDGE	[NOT] INTERACT

Table 2-4. Supported SVRF Operations (cont.)

INTERNAL	[NOT] LENGTH
NOT	OR
[NOT] OUTSIDE	[NOT] OUTSIDE EDGE
[NOT] RECTANGLE	RECTANGLES
SIZE	[NOT] TOUCH
[NOT] TOUCH EDGE	[NOT] TOUCH INSIDE EDGE
[NOT] TOUCH OUTSIDE EDGE	[NOT] WITH EDGE
[NOT] WITH WIDTH	XOR

DRC Checks¹

1. DRC checks must have only one COPY operation. DRC CHECK MAP statements are allowed but ignored.

LAYER statements with names that match those passed in by refine_exec are required. The value of the layer numbers does not matter. Additionally, the input and output layers must be polygon layers. Edge layers are allowed as intermediate layers, but error layers are not allowed at all.

SVRF setup files can access the LPE regions using predefined layer names listed in the following table. The LPE layer names must be mapped using an SVRF LAYER statement.

Table 2-5. Predefined LPE Layers for Regions

region::core	region::repair
region::context	region::visible
region::core_context	

Note that the results of SVRF operations within Calibre LPE may not match standard SVRF runs. This is because Calibre LPE clips geometries when creating repair regions, similar to LAYOUT WINDOW.

Examples

Example 1 — Iterations

```
refine_stop_iterations 3
refine_exec x1 = pxopc target file setup1.in
                                file setup2.in
                                file setup3.in map e1
```

In the example above, Calibre LPE runs Calibre pxOPC for the first iteration with *setup1.in*, for the second iteration with *setup2.in*, and for the third iteration with *setup3.in*. In other words, it is unnecessary to literally repeat refine_exec multiple times for the purpose of iterations. The multiple file specifications, however, are not required for Calibre LPE when running Calibre

OPCverify. This is because the same set of verification rules should be always used, regardless of how many iterations.

Also note that the second line does not use the line continuation character “\”. It is not necessary in Calibre LPE setup files.

Example 2 — Multiple Layers

```
refine_exec combined = or layer1 layer2 layer3
```

Notice that there are no parentheses when supplying multiple layers to an operation. If the order of combination is important, use separate `refine_exec` statements.

Example 3 — Using SVRF for Retargeting

With `refine_exec svrf`, you can perform retargeting inside Calibre LPE. Retargeting modifies the target shapes and uses the modified target layer as the `opc` layer for `refine_exec denseopc`. The original target layers before modification must be defined as `opc` layers in the LPE setup file.

```
LITHO FILE svrf.in [ /*
    LAYER POLY 1
    E = CONVEX EDGE POLY ANGLE1==90 ANGLE2==90 LENGTH1>0.01 LENGTH2>0.01
        WITH LENGTH<=0.02
    X = EXPAND EDGE E INSIDE BY 0.01
    X { COPY X }
*/ ]

LITHO FILE lpe.in [ /*
    layer POLY correction
    layer target opc
    layer markers hidden
    ...
    refine_exec ADJ_X = svrf POLY file svrf.in map X
    ...
    refine_exec outlayer = denseopc ADJ_X target markers file nmopc.in \
        map outlayer
    ...
*/ ]

RET REFINE Poly Poly.Target Marker FILE lpe.in MAP outlayer
```

Example 4 — Repairing Biased Layers

With `refine_exec nmbias`, Calibre LPE ensures the previously biased shapes are preserved in the visible region while new bias rules are applied in the core region based on the original target. The setup for `nmbias` needs to identify the drawn target (which should be of type `opc`) but the prebiased layer is passed automatically, as in the lines following:

```
layer drawn opc
layer prebiased correction
layer markers hidden
```

refine_exec

```
refine_markers markers
refine_stop_iteration 1
refine_regions update

refine_exec x = nmbias drawn setup nmbias.in map x

refine_map x map x
refine_map prebiased -from x map corrected
refine_map -region core map core
refine_map -region context map context
refine_map -region visible map visible
```

Although the post-OPC layer, prebiased, is included in the list of layers for the LPE setup file, the `refine_exec` command only lists the target layer, drawn. The `refine_map` lines pass the corrected layers (one with corrections only, and the other with merged results) and region markers as output to corresponding RET REFINE statements.

If the Calibre LPE setup includes both `refine_exec nmbias` and `refine_exec denseopc`, identify the prebiased layer as type `retarget` instead of `correction`. If only `refine_exec nmbias` appears, identify the prebiased layer as type `correction`.

Related Topics

[refine_map](#)

[refine_stop_iterations](#)

refine_map

Type: [Litho Setup File Commands](#)

Required. Identifies the output to return from an RET REFINE command.

Usage

Syntax 1:

refine_map *layername* [-from *layer_out*] [-iter {*value* | all}] **map** *mapname*

Syntax 2:

refine_map [-iter {*value* | all}] **-region area** **map** *mapname*

Arguments

- **layername**

For syntax 1, a required argument that can be either the output (*layer_out*) of a [refine_exec](#) operation or an input [layer](#).

When **layername** is an input layer, the behavior depends on the layer type.

- For hidden and opc layers, the geometries within the repair region are output. These geometries are clipped, and do not show the portion of polygons outside the region.
- For correction layers, the refine_map statement must include the “-from *layer_out*” option. The output includes both the correction layer geometries from outside the repair region and the corrected *layer_out* shapes from inside the repair region.

Note



If a correction layer is specified without “-from *layer_out*” the Calibre LPE output is not merged into the original layer.

- **-region area**

For syntax 2, a required argument where area is one of the following values:

- all** — Returns the geometries in the core, context, and visible regions; that is, all the shapes considered by Calibre pxOPC when called by Calibre LPE.
- context** — Returns the geometries of the transitional zone, where repaired polygons are adjusted to minimize discontinuities with the visible region.
- core** — Returns the inner area around the marker shape that received correction. The corrected shapes, particularly for SRAFs, can be significantly different from the input.
- visible** — Returns the geometries in the zone that was used for simulation, but not corrected.

The core, context, and visible regions should not be used for filtering. Their main use in the output is when interpreting results: the regions indicate the amount of correction to expect. The regions should not be used in post-LPE output for further operations.

- **map *mapname***

A required argument that indicates which RET REFINE statement receives the output. This is matched by *mapname*. For example,

```
A = RET REFINE target init_shapes markers FILE setup MAP inner
B = RET REFINE target init_shapes markers FILE setup MAP outer
C = RET REFINE target init_shapes markers FILE setup MAP all
...

LITHO FILE setup [ /*
...

refine_map -region visible map outer
refine_map -region core map inner
refine_map -region all map all
...
*/ ]
```

maps A to the second refine_map statement (-region core), B to the first refine_map statement (-region visible), and C to the third (-region all) based on the map names in bold font.

- -from *layer_out*

For syntax 1, an argument required when *layername* is a correction layer. The layer name for *layer_out* must match a [refine_exec layer_out](#).

Note



At least one statement must have a -from argument.

- -iter { *value* | all }

An optional argument specifying which refine_exec iteration to output.

value — Zero or a positive integer value less than the value of refine_stop_iterations.

For example, if refine_stop_iterations is 3, then *value* can be 0, 1, or 2. This outputs the results of the refine_exec operation for the specified iteration.

all — The default whenever *layername* or *area* is the output of a refine_exec operation.

This outputs the combined converged results from all iterations plus the unconverged results of the final iteration.

Description

A required command that is typically specified multiple times per Calibre LPE setup.

There are two modes to refine_map: Syntax 1 constructs output from input layers and refine_exec results. Syntax 2 can be used to see local repair regions. The local repair regions are for information only and should not be used in layout processing outside of Calibre LPE.

For corrected results to be passed to the SVRF run, two conditions must be met:

- There must be at least one statement that passes the results of a correction, sraf, or asraf layer with a -from argument.
- The names in the map arguments of **RET REFINE** and the “refine_map ...-from” must match. If they are not the same, the refine_map statement output is ignored.

Examples

To show how iterations and mapping work, consider the following case:

```
1 layer target opc
2 layer init_opcl correction
3 layer hot_spots hidden

4 refine_stop_iterations 3

5 refine_exec x = pxopc target file F0 file F1 file F2 map x

6 refine_map -iter 0 x map x0
7 refine_map -iter 1 x map x1
8 refine_map -iter 2 x map x2
9 refine_map -iter all x map xall
10 refine_map init_opcl -from x map ex
```

The iteration count starts at 0:

Table 2-6. Outputs by Iteration

Property	Iteration 0	Iteration 1	Iteration 2
pxopc Setup File	F0	F1	F2
Converged Results	C0	C1	C2
Unconverged Results	U0	U1	U2
Refine_Map Output	C0 + U0	C1 + U1	C2 + U2

The output for xall (line 9) would be C0 + C1 + C2 + U2, and the output for ex (line 10) would be xall plus the initial geometries outside the repair region.

The start of each iteration is marked in the transcript with lines similar to the following:

```
// RET REFINE - transcript for iteration #0
//      The repair region count = 4
```

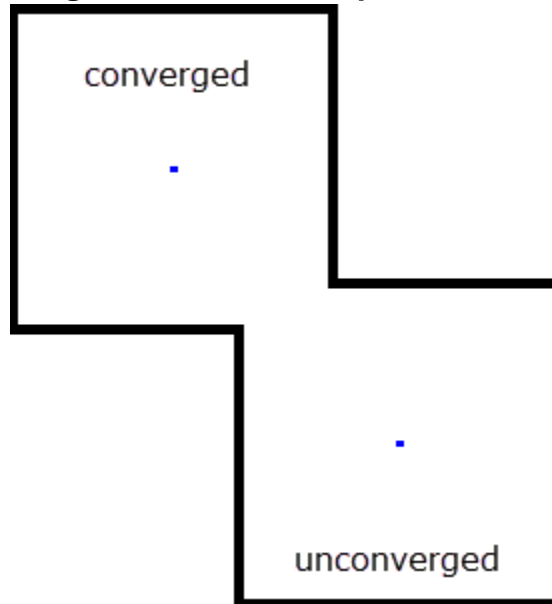
The end of each iteration is marked with a progress report, similar to the following:

```
// RET REFINE - progress report at the end of iteration #0
// CPU TIME = 8 + 1125 REAL TIME = 332 LVHEAP = 1137/1230/1231
//      The initial repair region count for iteration #0 = 4
//      The unconverged repair region count at the end of iteration #0 = 4
```

When the unconverged repair region count is zero, the repair process has completed.

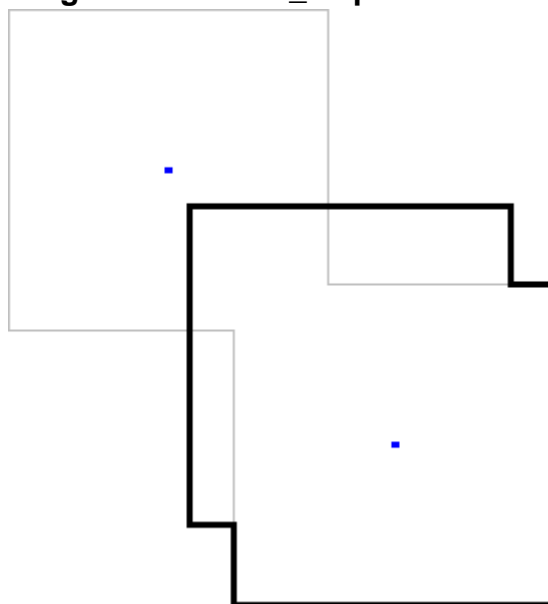
For example, if on iteration 0 one region converged and a second region did not, x0 (line 6) might show the result in [Figure 2-7](#).

Figure 2-7. refine_map iteration 0



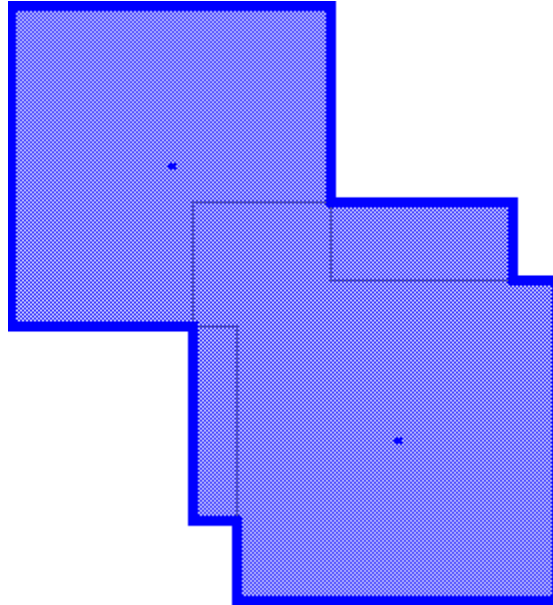
The converged region above is not passed to iteration 1, but the unconverged one is. An additional hot spot was detected nearby, and the output from iteration 1 (line 7) shows a slightly different repair region in the lower right. (The lighter line is the iteration 0 result for reference.)

Figure 2-8. refine_map iteration 1



Assuming iteration 1 fixed both hotspots, iteration 2 does not occur. The final repair region of xall would include the areas of both iteration 0 and iteration 1:

Figure 2-9. refine_map iteration all



Related Topics

[Calibre LPE Terminology](#)

[refine_stop_iterations](#)

refine_markers

Type: [Litho Setup File Commands](#)

Required. Specifies the layer with hotspot markers. Must appear only once.

Usage

refine_markers *name*

Arguments

- *name*
A required argument specifying the name of the layer that has shapes identifying the regions for optimization. The *name* should appear in a [layer](#) statement as a hidden layer.

Description

A required command that specifies the layer that contains the hotspot markers. Calibre LPE defines the repair regions around the *centerpoint* of each marker by default; the actual size and shape of the marker polygons does not matter, although small shapes are recommended. To use the actual marker itself, set “[refine_markers_style](#) wide”.

Marker shapes should be kept as small as possible, on the order of minimum feature size. This reduces the chance of the hotspot being on the edge of the repair region. For example, if the marker is created by selecting the entire polygon and the hotspot is on one corner of the polygon, the actual hotspot may be in the outer area or even outside the repair region. This is because the region is constructed outwards from the point at the center of the marker.

Another reason to keep the markers as small as possible is to reduce marker overlap. When marker shapes overlap, they are merged. The centerpoint is calculated from the merged shape, and might not cover the full area intended.

refine_markers_limit

Type: [Litho Setup File Commands](#)

Specifies the maximum number of markers to process. Suggested use is to prevent excessive runtime due to bad input.

Usage

refine_markers_limit *limit* [stop | skip]

Arguments

- **limit**
A required argument specifying the maximum number of polygons on the marker layer that should be processed. If neither stop nor skip are specified, the first **limit** polygons go through the Calibre LPE operation and the remaining markers are ignored.
When **limit** is 0, there is no upper bound on the number of markers to process. (This is the default behavior if refine_markers_limit is not specified.) Set **limit** > 0 to control run time.
- stop
An optional argument that causes the entire run to exit if the number of polygons on the marker layer exceeds **limit**. Cannot be specified with “skip.”
- skip
An optional argument that causes the run to move to the next operation after RET REFINE if the number of polygons on the marker layer exceeds **limit**. Cannot be specified with “stop.”

Description

When Calibre LPE is part of a fully integrated production flow, the number of markers is typically fairly low. Calibre LPE is used for unexpected failures, as production flow recipes should be well tested. A large number of markers indicate a problem in the setup, such as a mismatch between the layout and OPC rules or dropped data. To prevent hundreds of CPU hours being spent on an incorrect setup, add the refine_markers_limit command to the recipe.

Examples

```
refine_markers_limit 600 stop
```

refine_markers_style

Type: [Litho Setup File Commands](#)

Enables core regions to encompass large markers.

Usage

refine_markers_style { points | size | wide }

Arguments

- **points**

A required argument indicating that markers are treated as center points of their extents. This is the default setting.

- **size**

A required argument indicating to use the marker as the core region. Use when markers contain holes that should be maintained.

The size setting derives the context region by sizing up the marker by distance1 and then subtracting the marker. If the marker contains a hole, the area inside the hole is part of the context region. The visible region remains the outer perimeter of the context region sized up by distance2.

- **wide**

A required argument indicating that markers control the region sizes. When this is set, the core region always includes the full marker shape.

If the marker is smaller than distance0 (whether the default or set by refine_distance), the core region behaves like **points**. If the marker is greater, the core region is resized to at least the marker extent. There is no requirement that the width and length of the core region be the same, but the shorter edge is at least distance0.

The context and visible regions are determined as usual using distance1 and distance2.

Description

Wide markers can be useful when an entire SRAF represents a hotspot, or in other situations where an entire region is difficult to resolve across process conditions. Generally, however, it is best to place a marker at a relatively small bridging or pinching location and use the default (centerpoint of marker) setting.

Large markers can result in large repair regions, which cause slower runs. When a repair region is larger than a tile, Calibre LPE produces the following message:

```
RET REFINE Warning: a large repair region detected in the following
coordinate window in the cell cellname:
bottom left coordinate -> upper right coordinate
```

The tile size is set in the OPC or verify setup file.

Related Topics

[refine_distance](#)

refine_patterns

Type: [Litho Setup File Commands](#)

Identifies repair regions with identical input and processes them as one instance. Requires `refine_process` hier.

Usage

refine_patterns {**layer** *layer_in*}... [*transform*]

Arguments

- **layer** *layer_in*

A required argument indicating the input layer. At least one input layer must be specified. The first layer must be of type “correction” or “opc”. (The opc layer is recommended.) Subsequent layers can be of type correction, opc, or hidden.

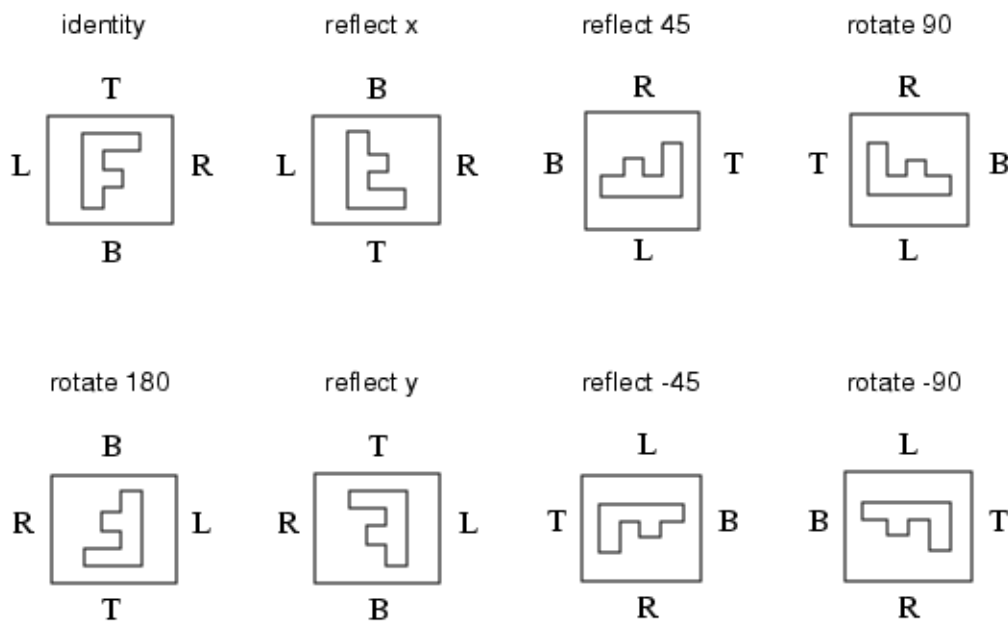
- *transform*

An optional argument specifying valid transformations for a region to be considered a duplicate. The following keywords are valid:

<code>all</code>	<code>reflect x</code>	<code>reflect 45</code>	<code>rotate 90</code>	<code>identity</code>
<code><u>auto</u></code>	<code>reflect y</code>	<code>reflect -45</code>	<code>rotate -90</code>	<code>rotate 180</code>

The keywords `all` and `auto` cannot be specified with any other *transform* keywords. The other eight keywords can be specified in combination, and are defined in [Figure 2-10](#).

Figure 2-10. Transformations for Pattern Matching



The default behavior is “auto”. This setting automatically determines applicable transformations based on the optical model of the first pxopc or opcverify [refine_exec](#) statement. The “all” keyword specifies all transformations, even if they are not valid with the particular optical model. The recommended setting is “auto”.

Description

An optional command that activates pattern classification to recognize repair regions with identical input layers. It requires hierarchical repair regions, which you enable with [refine_process](#) hier. If `refine_patterns` is in the recipe without `refine_process` hier, the run exits with the following error:

```
RET REFINE - Specify refine_process hier to use refine_patterns.
```

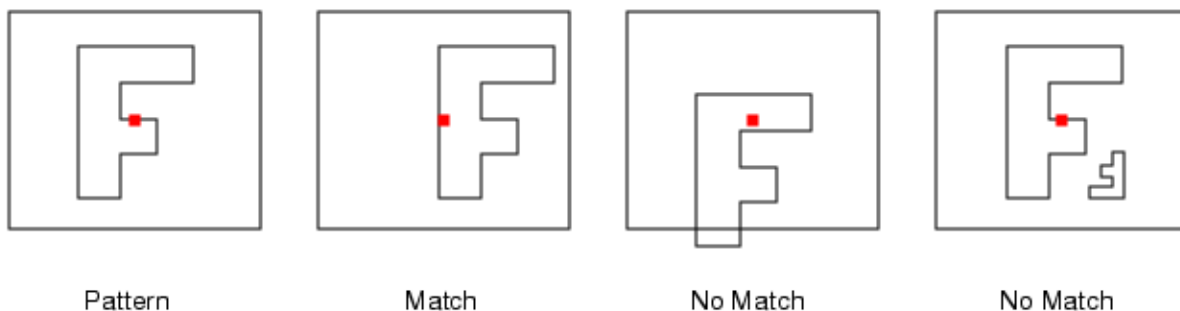
Matching regions are processed once, instead of each being processed independently. For designs with repetitive patterns such as memory, pattern classification reduces run time and memory consumption.

After optimization, the same core and context results are pasted into all matching repair regions. In order to catch any individual MRC issues, it is best to run `refine_patterns` as part of an iterative repair flow. (See “[Example 2 — Recommended Combination.](#)”)

The `refine_patterns` command does not require the marker to be identically placed relative to the pattern. The pattern is constructed from the first unique region the algorithm encounters, generally the bottom left of the layout. It maps the relative position of the vertices of the shapes on *layer_in* within the repair region.

All other repair regions are compared to discovered patterns. If all vertices within the region on the layer have the same geometrical configuration, the regions are considered a match. If the marker is significantly offset, parts of the *layer_in* shapes may project beyond the repair region. In this case a match will not be made, even if the overlapping regions are otherwise identical.

Figure 2-11. Matching Repair Regions



Double Pattern Classification

If the *layer_in* is a target layer (that is, it is of type “opc”), pattern matching is done with double pattern classification. This method uses the target layer in pxOPC operations, and the correction

layer for other refine_exec operations. The reason that the verification operations use the correction layer is that shapes in the visible region could cause MRC violations.

Automatic MRC repair is automatically applied when results are copied to the original matching regions, before running the verification refine_exec operations. The repair is based on the correction layer of the initial pattern. Automatic MRC repair is written to the transcript as the “reassemble” process.

Transcript Additions

When refine_patterns is included in the LPE setup file, the following sections are added to the transcript.

```
refine_patterns enabled for the following transforms:
IDENTITY (TRANSLATION)
...
```

Depending on the specified *transforms*, there may be additional transformations listed.

The next section depends on whether the **layer_in** is a correction layer (regular pattern matching) or an opc layer (double pattern classification). For regular pattern matching, the transcript shows the following:

```
refine_patterns activated with correction layer:
The correction layer is to be used for the pattern matching
There is no second pattern matching in this case
```

For double pattern classification, the transcript shows instead:

```
refine_patterns activated with target layer:
The target layer is to be used for the first pattern matching
The correction layer is to be used for the second pattern matching
The first pattern matching will be used for pxopc execution
The second pattern matching will be used for the execution of the rest
of the operations.
```

Last is a report of the time required for pattern matching.

```
RET REFINE the pattern matching finished: CPU TIME ...
```

Examples

Example 1 — Minimum Command Syntax

```
refine_patterns layer input1
refine_process hier
```

The minimum command specifies one input layer, and reads the optical model to determine what types of transformation are valid. Optical model information is reported in the transcript in a line beginning “/-- INFO: WORST OPTICAL MODEL SOURCE SYMMETRY”.

Example 2 — Recommended Combination

When using `refine_patterns`, it is best to include it in an iterative flow, as shown in the following code.

```
LITHO FILE lpe.setup [/*

# Layers
layer target   opc           # drawn target
layer init1    correction    # layer with earlier OPC corrections
layer markers hidden        # hotspot indicators
...
refine_patterns target
refine_process hier
...
# Commands for iterative flow
refine_stop_iterations 2
refine_stop_layer hotspots

# Execution
refine_exec iter1 = pxopc target file pxopc1.in file pxopc2.in map x1
refine_exec space = mrc iter1 map spacing
refine_exec bridge = opcverify target iter1 file opcv.in map bridge
refine_exec pinch = opcverify target iter1 file opcv.in map pinch
refine_exec hotspots = or bridge pinch space
...
*/]
```

Related Topics

[refine_process](#)

refine_process

Type: [Litho Setup File Commands](#)

Preserves the layout hierarchy in repair regions.

Usage

refine_process {**flat** | **hier**}

Arguments

- **flat**
A required argument that specifies Calibre LPE flattens repair regions before operating on them. The layout outside the repair regions is not affected.
- **hier**
A required argument that specifies Calibre LPE attempts to preserve any cell or layer hierarchy in the repair regions. This is the default behavior.

Description

This optional command causes Calibre LPE to prepare hierarchical repair regions. For highly hierarchical input such as memory designs, hierarchical repair regions can improve memory consumption and processing speed.

When “refine_process flat” is used, Calibre LPE places all repair regions at the top level of the design hierarchy. This results in each repair region being treated independently. It provides more accurate results as any contextual differences such as rotation are included in the optimization and simulation.

In the default or “refine_process hier” mode, Calibre LPE places the repair regions at the lowest possible level of the design hierarchy. The lowest possible level includes all layers related to a particular repair region. For highly repetitive hierarchical designs such as memory, this produces more compact output and possibly reduces run time.

If the setup file uses pick_region, set refine_process to flat. The pick_region coordinates are interpreted as flat.

The RET REFINE operation is not affected by this setting. It always runs in the Calibre hierarchical (multithreading) mode. The processing_mode settings of the operations called by refine_exec are also not affected.

Related Topics

[pick_region](#)

refine_prune_markers

Type: [Litho Setup File Commands](#)

Controls what is included in the active region of iterative repairs.

Usage

refine_prune_markers {**yes** | **no**}

Arguments

- **yes** | **no**

A required argument indicating how to handle markers in unconverged repair regions.

yes — Prune markers of unconverged repair regions. The next iteration considers only newly formed hotspots. This requires “refine_regions update” and “denseopc_init_mode last” or “denseopc_init_mode target_last” also be set.

no — Keep markers of unconverged repair regions and include the regions in the next iteration. (Markers in converged repair regions are never included in the next iteration.) This is the default.

Description

This optional command controls whether markers in unconverged regions are kept for the next iteration, or only newly formed hotspots are optimized.

This command is ignored except for “refine_exec denseopc” iterations.

Examples

The following commands would perform up to three iterations to repair hotspots created by the first re-OPC with a Calibre nmOPC setup.

```
denseopc_init_mode last
refine_stop_iterations 4
refine_regions update
refine_prune_markers yes
```

Related Topics

[denseopc_init_mode](#)

[refine_regions](#)

refine_regions

Type: [Litho Setup File Commands](#)

Alters the repair region definition for multiple iterations.

Usage

refine_regions {**keep** | **update**} [keep | update]...

Arguments

- **keep** | **update**

A required argument indicating whether to keep the original hotspots (the default behavior) or generate new ones after an iteration. The first iteration, iteration 0, always uses the regions supplied by the marker layer.

This argument can be specified up to (refine_stop_iterations - 1) times. For example, if refine_stop_iterations is 3, the syntax is “refine_regions {keep | update} [keep | update]”. If the iterations exceed the number of values, the last setting continues to be used for the remaining iterations.

Description

An optional command that may appear at most once in a Calibre LPE setup file. After the first iteration, Calibre LPE removes the converged regions from further operations. If refine_regions is “keep”, the same hotspots and repair regions are used. If refine_regions is “update”, new hotspots and regions are calculated using refine_stop_layer and refine_exec... opcverify.

Examples

```
refine_stop_iterations 4
refine_regions keep update
```

In the lines above, for the first iteration the input markers from refine_markers are used to calculate the repair regions. In the second iteration, “keep” is applied. The third iteration updates the repair regions. In the final, fourth iteration, as there is no keyword specified, the behavior remains “update”.

Related Topics

[refine_stop_iterations](#)

[refine_stop_layer](#)

[refine_exec](#)

[refine_markers](#)

refine_stop_iterations

Type: [Litho Setup File Commands](#)

Specifies how many iterations at most to perform. The default is 1.

Usage

refine_stop_iterations *num*

Arguments

- *num*
A required positive integer indicating how many iterations to perform.

Description

An optional command that may appear at most once in a Calibre LPE setup file. It specifies the maximum number of iterations to perform. If the results for all regions converge before the *num* iteration, the remaining iterations are not performed.

Convergence rules are specified using `refine_stop_layer` and OPCverify simulation results.

Related Topics

[refine_exec](#)

[refine_regions](#)

[refine_stop_layer](#)

refine_stop_layer

Type: [Litho Setup File Commands](#)

Identifies the refine_exec output layer that contains any hot spots after each iteration.

Usage

refine_stop_layer *name*

Arguments

- *name*
A required argument specifying a layer name. This layer is derived by refine_exec, generally from Calibre OPCverify results.

Description

An optional command that may appear at most once in a Calibre LPE setup file. It is used with iterative flows that update the repair regions.

If the name specified does not match a refine_exec output, the run exits with the following:

```
RET REFINE - the refine_stop_layer name is not present in the refine_exec
outputs

ERROR: Error LTH12 on line N of file - RET or MDP... operation FILE error.
```

Examples

Example 1 — Matching Names

The lines following show how the refine_stop_layer name matches that of a refine_exec layer:

```
refine_stop_layer hot_spots
refine_exec hot_spots = opcverify x1 file opcv.in map bridging
```

Example 2 — An RET REFINE Setup With Multiple Iterations

The excerpt from an RET REFINE setup file show a layer derivation for an iterative flow. It does not show all settings that might be needed, but just those that relate to using refine_stop_layer.

```
refine_stop_layer hot_spots
refine_stop_iterations 3
refine_regions update      # Without this, hot_spots is never checked
                           # to see if there are new problems.

# Call Calibre pxOPC to do the refining. The setup files are not shown.
refine_exec x1 = pxopc target      file pxopc.in    file pxopc1.in
                                file pxopc2.in    map opc_out
refine_exec bridge = opcverify x1 file opcv.in    map bridging
refine_exec pinch  = opcverify x1 file opcv.in    map pinching

# OR together the OPCverify checks to see if there are new spots.
# If this layer is empty, no more iterations are performed.
refine_exec hot_spots = or bridging pinching
```

Related Topics

[refine_exec](#)

[refine_regions](#)

[refine_stop_iterations](#)

refine_update_layer

Type: [Litho Setup File Commands](#)

Copies the output from a refine_exec iteration to a specified layer for use in the next iteration when “refine_regions update” is set.

Usage

refine_update_layer *layer_in* with *layer_out*

Arguments

- ***layer_in***

A required argument specifying a layer name. The ***layer_in*** must be named in a [layer](#) command with type **hidden**. It should not be a marker layer.

If ***layer_in*** is not a hidden layer, the run exits with the following message:

```
The first layer in refine_update_layer <layer> with <layer> is not found
among the hidden input layers.
```

- **with *layer_out***

A required argument specifying a layer created by [refine_exec](#) ***layer_out***. Its content is transferred to ***layer_in***.

Description

The optional refine_update_layer command transfers the refine_exec output from an iteration to be used as input in the next iteration when “refine_regions update” is also specified. (When “refine_regions keep” is specified, refine_update_layer has no effect.)

This command can be specified multiple times; however, each ***layer_in*** may only be paired with one ***layer_out***. That is, a ***layer_in*** cannot receive data from multiple layers. The data from ***layer_out*** replaces the data on ***layer_in*** within the repair regions.

This command has no effect on the first iteration, as there is no ***layer_out*** to copy.

Examples

This example includes just the lines that need to be set for refine_update_layer. The matching names are highlighted in the same color.

```
layer mW hidden

refine_exec out = opcverify mla mlb file setup.in map class_2

refine_update_layer mW with out
refine_regions update
```

Related Topics

[refine_regions](#)

refine_visible_sim

Type: [Litho Setup File Commands](#)

Checks the inner part of the visible region for new hotspots.

Usage

refine_visible_sim {yes | no}

Arguments

- yes | no

A required value indicating whether the visible region should be checked for hotspots.

yes — This is the default behavior. Check the inner part of the visible region.

no — The visible region is not checked for new hotspots.

You must specify either yes or no.

Description

When doing multiple iterations, Calibre LPE can check for changes to hotspots either only in the core and context regions or across the entire repair region.

The default is to check across the entire repair region. This works well for large repair regions that can be formed by clustered hotspot markers, as are often found in real layouts. You may want to set this keyword to no if your hotspots are separated enough that the repair regions do not interact.

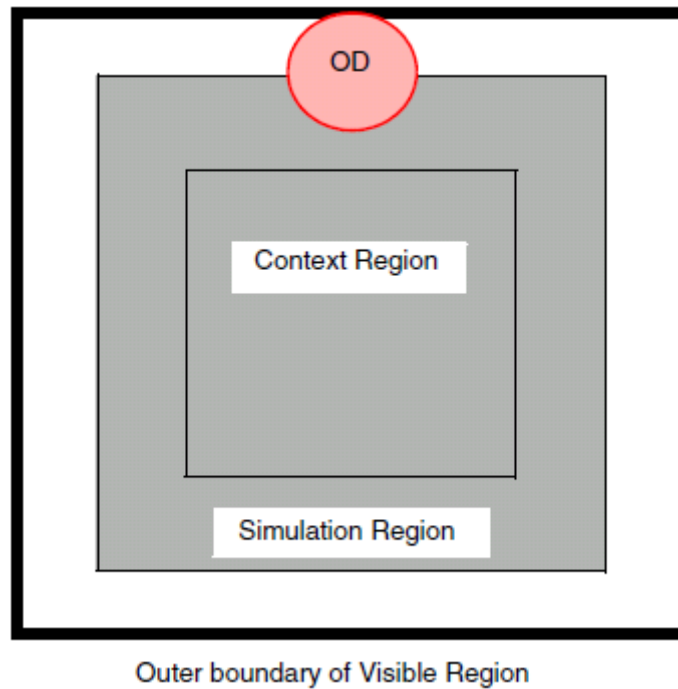
If the visible region width is less than the optical diameter, the images used for calculating interaction with the inner regions changes significantly. Additionally, the number of iterations may increase because of new hotspots when the visible region is also checked.

If you do not set this keyword to no, the setup file also needs to specify the following:

- Specify a visible region greater than the optical diameter using refine_distance2.
In [Figure 2-12](#), the visible region extends from the edge of the context region to the heavy outer border. The simulation zone in the visible region is constructed by shrinking the outer boundary by an optical radius (half the optical diameter, shown as a red circle).
- Include the hotspots from the visible simulation region by setting refine_regions update.

If refine_regions update is not set, Calibre LPE detects the hotspots in the visible simulation region but does not shift the repair region to include them.

Figure 2-12. Visible Simulation Region



Examples

If the optical radius is 0.64 μm , then to check for hotspots in the visible region the recipe adds the following lines:

```
refine_distance2 1.64
...
refine_visible_sim yes
...
refine_regions update
```

The `refine_distance2` setting leaves a visible simulation region of 1.0 μm .

Related Topics

[refine_regions](#)

[refine_distance](#)

reject_duplicate_layers

Type: [Litho Setup File Commands](#)

Permits duplicate layer names in refine_exec operations.

Usage

reject_duplicate_layers {yes | no}

Arguments

- yes | no

A required value indicating whether duplicate layer names should be rejected.

yes — This is the default behavior. Duplicate layer names are treated as a syntax error.

no — Duplicate layer names are allowed.

You must specify either yes or no.

Description

The reject_duplicate_layers command controls whether duplicate layer names in refine_exec statements are syntax errors. The check should only be disabled to enable triple patterning with Calibre pxOPC.

Examples

This will not exit with an error:

```
...
reject_duplicate_layers no
...
refine_exec X = opcverify layer1 layer1 layer2 layer3 file setup.in map X
...
```

Related Topics

[refine_exec](#)

smart_cut_disable

Type: [Litho Setup File Commands](#)

Controls whether additional compensation is done to avoid MRC violations.

Usage

smart_cut_disable {yes | **no**}

Arguments

- yes | **no**

A required argument specifying whether smart cuts should be used.

yes — Do not perform the smart cut step.

no — Perform smart cut. This is the default.

Description

The optional `smart_cut_disable` command controls whether Calibre LPE uses “smart cuts.” Smart cuts smooth the transition between LPE corrections and frozen geometry when using Calibre nmOPC for repair.

The typical Calibre LPE run takes a relatively small number of markers. Time spent on setting up repair regions is negligible compared to time spent in `refine_exec`. However, if the markers are such that the majority of the design area is covered with repair regions, the setup time can be prohibitive. To alleviate slow setup time, you can turn off smart cuts.

If you set `smart_cut_disable` yes, it is recommended that you set the last line of `denseopc_context_parameters` to “displacement limit 0 0 feedback 0.” This line prevents fragments close to the border between context and visible layers from and that the setup file for the `refine_exec` operations should include all MRC settings.

Related Topics

[denseopc_context_parameters](#)

Chapter 3

Calibre LPE Best Practices

Best practices are proven settings that Siemens EDA generally recommends, but does not require.

LPE Setup Best Practices	107
Tool Interaction Best Practices	108
Memory Management Best Practices	109
Pre-Existing MRC Errors	110
Browsing LPE Results	111

LPE Setup Best Practices

The Calibre LPE work flow is different from other OPC tools and places different constraints on the input and settings.

Table 3-1 lists general best practices.

Table 3-1. Calibre LPE Best Practices

Practice	Reason
The input to LPE should reflect OPC and contain SRAFs.	Although Calibre LPE adjusts the main features and generates new SRAFs, the shapes in the visible region are not modified. They need to reflect the shapes expected to be manufactured on the mask so that simulation is more accurate.
Do not create large markers.	Markers are treated as points. Large polygons do not affect the repair region size, but may cause it to be placed other than where expected. Point-like squares of a size similar to the minimum feature size are best.
The value of mrc_small_feature_size should be smaller than mrc_min_square .	When this is violated, square SRAFs that are acceptable to the foundry are removed.
Use the SVRF INCLUDE statement to include separate setup files.	While it is allowed for Litho setup files to be outside the SVRF rule file, it causes bad output if the setup file is edited while a run that calls it is executing. Because post-tapeout runs typically take significant time, it is best that the main rule file include all Litho setup files.

Table 3-2 lists recommended command settings specific to setting up Calibre LPE to run Calibre nmOPC for re-opc.

Table 3-2. Calibre LPE Dense OPC Best Practices

Command	Default	Recommendation
calibre -lpe	Not applicable	Use the LPE script generator to perform initial modifications to a Calibre nmOPC rule file for Calibre LPE use.
denseopc_context_parameters	Not set (context region fragments are treated identically in entire region)	Five zones, defined as follows: <pre>displacement_limit off feedback off displacement_limit -.012 0.012 feedback -keep -0.015 displacement_limit -.007 0.007 feedback -keep -0.07 displacement_limit -.003 0.003 feedback -keep -0.03 displacement_limit -.001 0.001 feedback -keep -0.01</pre>
denseopc_copy_fragments	no	yes
denseopc_insert_correction	no	yes
denseopc_limit_excess	no	yes
denseopc_smooth_corrections	no	yes

Tool Interaction Best Practices

Because Calibre LPE can call multiple setup files, it is best to check that all tools are using consistent settings. The tool does not enforce these.

Table 3-3. Consistency Best Practices

Practice	Reason
Do not specify small tiles.	The entire repair region should fit inside a single tile. When the visible region (or worse, the context region) intersects tile boundaries MRC problems may occur.
Ensure minimum edge and spacing values are the same in all tools.	When Calibre pxOPC and Calibre OPCverify use different settings, solutions appear to not converge.
The value of mrc_min_edge should be no smaller than the value of mrc_min_internal.	When this is violated, Calibre pxOPC cannot improve the PV band.

Table 3-3. Consistency Best Practices (cont.)

Practice	Reason
In the pxopc_options, use “ scatter_offset 1.0 ” for metal layers and “ scatter_offset 0.5 ” for contact and via layers.	Reduces SRAF inconsistency in the transition region.
In Calibre OPCverify setup files, include “ keep_no_context duplicates ” in the classification blocks.	Calibre LPE requires unclassified violation markers from Calibre OPCverify.
In Calibre OPCverify setup files used with Calibre nmOPC correction, set checks about 10% or 2 dbu tighter.	Tighter verification settings prior to and during the Calibre LPE run catch possible errors so that they can be corrected. This makes the final results less susceptible to the detection of possible errors during final verification with Calibre OPCverify outside of Calibre LPE.

Additionally, some things to be aware of when checking for tool-to-tool consistency include the following:

- The transition regions in Calibre LPE are not the same as those used in the Calibre pxOPC run. The PXOPC engine for Calibre LPE defines the transition region as the boundary of the visible region sized down by a certain amount. There are several cases in which this differs from the transition regions shown by the Calibre LPE framework. For example, in Calibre LPE transition regions may connect two different core regions whereas the Calibre pxOPC engine would consider the two as one core.
- Calibre LPE does not support the setlayer tilegen command. The command does run, but the tiles it outputs are not the same as the tiles constructed inside a run.

Related Topics

[Classification Block \[Calibre OPCverify User's and Reference Manual\]](#)

Memory Management Best Practices

A common cause of a hung or failed run is not enough memory on the remote hosts. The settings summarized here make the best use of memory on remote hosts, allowing even triple-patterning LPE-pxOPC runs to complete.

Disable SMT

Simultaneous multithreading (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. See the cautions in “[Simultaneous Multithreading With Virtual CPUs](#)” in the *Calibre Administrator's Guide*.

To turn off SMT, set the environment variable `CALIBRE_MT_SMT_FACTOR` to 0.

If you are using hyperscaling, you must additionally disable SMT in the remote configuration file. Ensure the remote configuration file contains the following line:

```
HYPER SMTFACTOR 0
```

C Shell

```
% setenv CALIBRE_MT_SMT_FACTOR 0
```

Bourne/Korn Shell

```
$ CALIBRE_MT_SMT_FACTOR=0  
$ export CALIBRE_MT_SMT_FACTOR
```

Use SOCS Shared Memory

Manufacturing processes that require multi-patterning typically use advanced optical models that include SOCS kernels. From Calibre version 2016.4 and forward, processes running on the same physical host can share the directory that contains SOCS calculations.

See “[LITHO_SOCS_KERNELS_SHARED](#)” in the *Calibre Post-Tapeout Flow User’s Manual* for details on monitoring shared memory.

To enable, set the environment variable `LITHO_SOCS_KERNELS_SHARED` to 1.

C Shell

```
% setenv LITHO_SOCS_KERNELS_SHARED 1
```

Bourne/Korn Shell

```
$ LITHO_SOCS_KERNELS_SHARED=1  
$ export LITHO_SOCS_KERNELS_SHARED
```

Related Topics

[Shared Memory Guidelines for Distributed Calibre \[Calibre Administrator's Guide\]](#)

[Calibre Post-Tapeout Flow User’s Manual](#)

Pre-Existing MRC Errors

Most rule files run MRC and verification after Calibre LPE, and sometimes report errors in the repair region that are not a result of Calibre LPE. The example SVRF shows how to ignore errors that do not interact with the corrected region.

Calibre LPE corrects all issues in the core region, smooths the corrections to the pre-existing shapes in the context or transition region, and uses the visible region to inform the simulation. Shapes in the visible region are unchanged by Calibre LPE and may have MRC violations. Use lines like the following example to ignore the visible region.

```

refine_exec mrc_hotspot_W = svrf M0 M0_OPC region::core_context \
                                FILE mrccheck.in MAP mrc_hotspot_W
refine_exec mrc_hotspot_S = svrf M0 M0_OPC region::core_context \
                                FILE mrccheck.in MAP mrc_hotspot_S
refine_exec mrc_hotspot_H = svrf M0 M0_OPC region::core_context \
                                FILE mrccheck.in MAP mrc_hotspot_H

//SVRF setup file
LITHO FILE mrccheck.in [
    LAYER M0                      1
    LAYER M0_OPC                  2
    LAYER CORE_CONTEXT_REGION    3

    FILTER_pre = SIZE M0 by 1
    FILTER      = SIZE FILTER_pre by -1.5
    W_pre       = INT M0_OPC < 0.01 REGION EXTENTS
    S_pre       = EXT M0_OPC < 0.01 REGION EXTENTS
    H_pre       = HOLES M0_OPC
    W = (INTERACT W_pre FILTER) INTERACT CORE_CONTEXT_REGION
    S = (INTERACT S_pre FILTER) INTERACT CORE_CONTEXT_REGION
    H = (INTERACT H_pre FILTER) INTERACT CORE_CONTEXT_REGION
    mrc_hotspot_W { COPY W }
    mrc_hotspot_S { COPY S }
    mrc_hotspot_H { COPY H }
]

```

The `refine_exec` keyword `region::core_context` refers to an internal layer created by the LPE process. It is passed to the SVRF setup file as a polygon layer with shapes marking the core and context regions.

Related Topics

[refine_exec](#)

Browsing LPE Results

The LPE Browser provides a simple way to compare the original OPC results and the LPE output, and then decide which to keep in the design. It can run LPE to fix merged regions or generate an SVRF file you can run in batch mode.

Restrictions and Limitations

- When hotspots lie inside merged regions and you undo only part of the merged results, you must re-run Calibre LPE. It is not possible to undo one part of a merged region without disturbing the results on the joining areas.

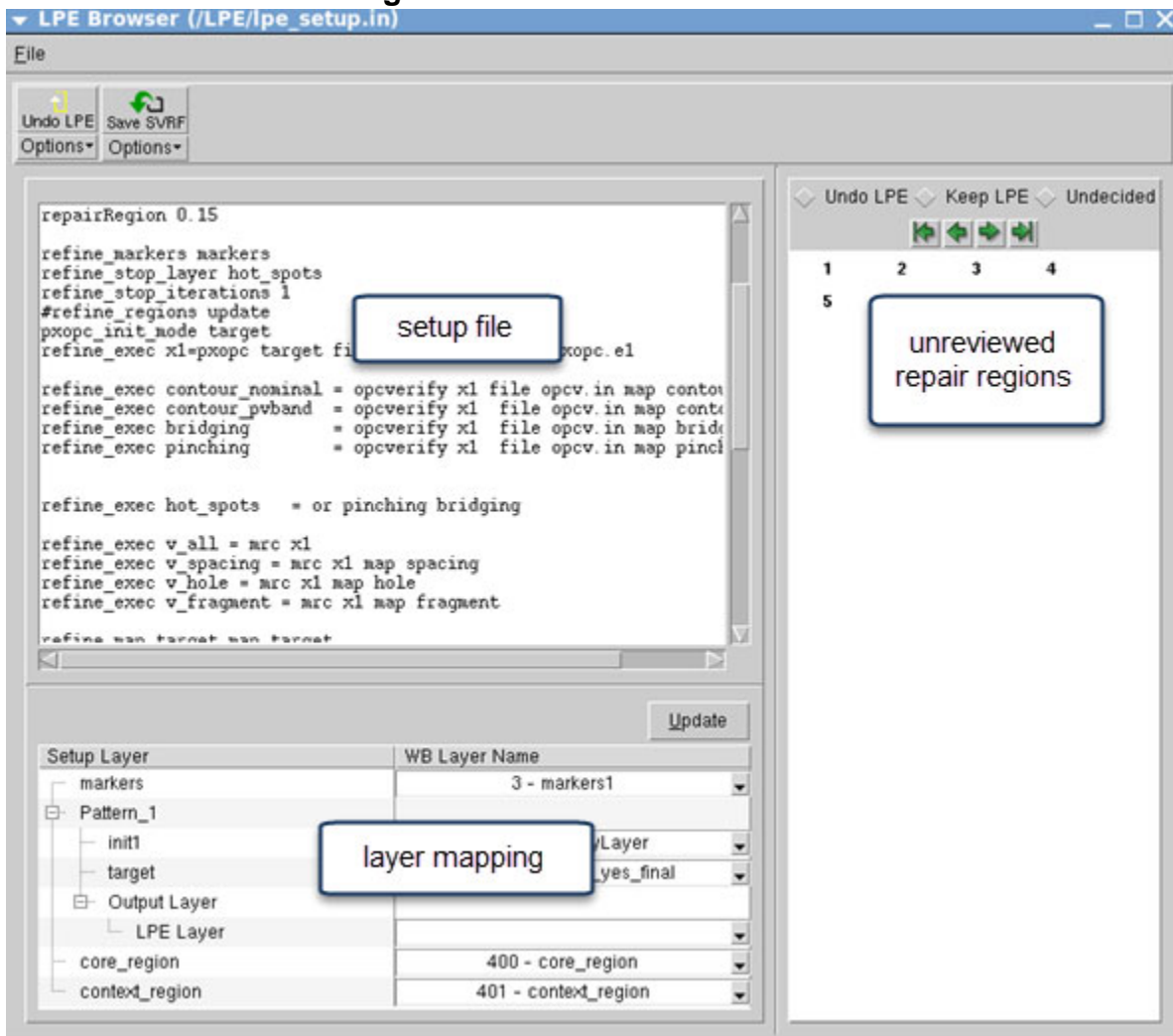
Prerequisites

- A layout file with all layers that were involved in the Calibre LPE run, both input and output. If the layout has only some of the input layers, you can still browse results but not undo them.

Procedure

1. Open the layout file saved from the Calibre LPE run in Calibre WORKbench.
2. Select **Litho > LPE Browser** to open the LPE Browser window.
3. In LPE Browser, select **File > Open LPE setup file** and navigate to the setup file used by the RET REFINE operation.

Figure 3-1. LPE Browser Panels

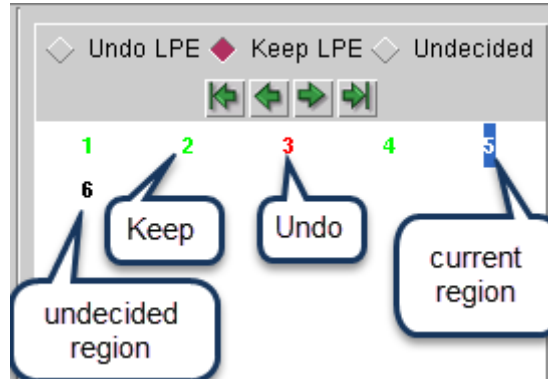


4. In the layer mapping pane, identify the layout layer for each setup file layer.
If you only want to browse results without using the other LPE Browser capabilities, you can skip this step. A full layer mapping is required, though, to undo LPE changes or re-run LPE.
5. Start reviewing the results. You can either click a number in the panel to the right or use the green arrow keys to move between them.

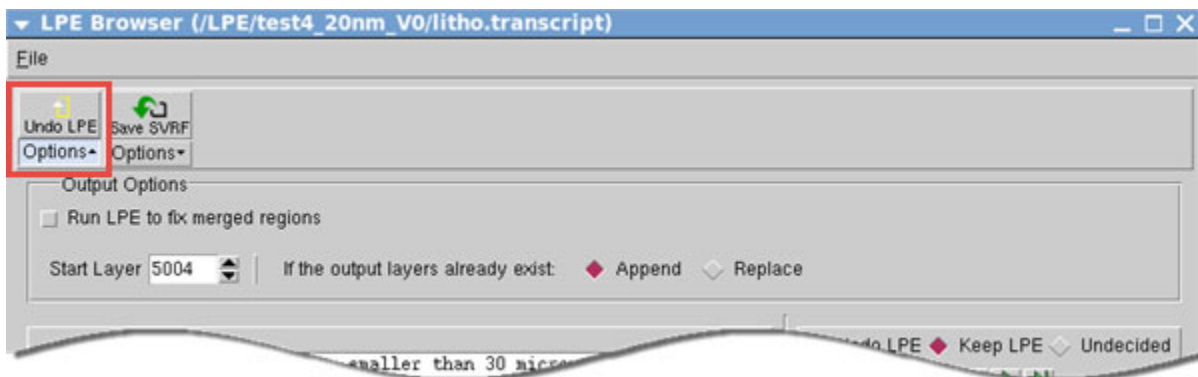
As you move between the numbers, the Calibre WORKbench view updates to show the region in the center. Zoom in and adjust layers as required to compare original and LPE results.

6. Use the radio buttons above the navigation controls to mark each region with “Undo LPE” or “Keep LPE.” The default is “Undecided”.

Figure 3-2. LPE Browser Region Decisions



7. If there are any regions to undo, choose one of the following methods:
 - Create an SVRF file to run later.
 - i. Open the Save Options pane by selecting the **Options** button under **Save SVRF**.
 - ii. If there are merged regions, decide whether you want to have the browser set up the RET REFINE operations to repair any changes. You may prefer to modify the original rules.
 - iii. Set the Start Layer. The browser uses it as the lowest layer number to which it can assign the layer derivations it creates.
 - iv. Click Save SVRF and enter a name for the file. You can run this in batch mode.
 - Have the LPE Browser undo the regions now.
 - i. Open the Output Options pane by selecting the **Options** button under the **Undo LPE** button.



- ii. Select Run LPE to fix merged regions if any of the regions you are undoing are part of a merged region.
- iii. Set the Start Layer. Append is recommended to avoid accidentally overwriting any layers.
- iv. Click the **Undo LPE** button. You will be prompted to save the results to a new file.

The generated SVRF will be executed from the directory you save it in. If the setup file uses relative paths such as “*../models*” choose a directory where these paths resolve correctly.

— Symbols —

`[]`, [14](#)

`{}`, [14](#)

`|`, [14](#)

— A —

About LPE, [9](#)

AND operation, [77](#)

— B —

Best practices, [89](#), [107](#)

Bold words, [13](#)

— C —

Calibre LPE invocation, [15](#)

Calibre LPE licenses, [13](#)

Calibre LPE prerequisites, [12](#)

Calibre LPE stages, [10](#)

CALIBRE_HOME variable, [13](#)

Classify blocks, [79](#)

Command line, [15](#)

Command list, [17](#)

Command syntax, [13](#)

Comments, [27](#)

Context regions, [12](#), [39](#)

Convergence, [100](#)

Core regions, [12](#)

Correction layers, [53](#)

Courier font, [14](#)

— D —

Denseopc setup, [19](#)

denseopc_context_parameters command, [38](#),
[108](#)

denseopc_copy_fragments command, [41](#)

denseopc_fragment_coincident command, [42](#)

denseopc_init_mode command, [44](#)

denseopc_insert_correction command, [45](#)

denseopc_limit_excess command, [46](#)

denseopc_pre_notchfill command, [47](#)

denseopc_smooth_corrections command, [48](#)

denseopc_use_correction_map, [49](#)

Double dipole lithography (DDL), [53](#)

Double pipes, [14](#)

— E —

Error messages

distance1, [74](#)

duplicate layers, [105](#)

min_frag_len, [56](#)

min_space, [57](#)

min_width, [58](#)

pxOPC, [51](#)

refine pattern, [93](#)

refine_stop_layer, [100](#)

refine_update_layer, [102](#)

— F —

Filter visible region, [110](#)

Finetune command, [51](#)

Font conventions, [13](#)

Full-chip best practices, [109](#)

— H —

Heavy font, [13](#)

Hidden layers, [53](#)

Hierarchical commands, [92](#), [96](#)

Hyperscaling, [109](#)

— I —

Improve results, [103](#), [107](#)

Input requirements, [10](#), [107](#)

Italic font, [13](#)

Iterative flow, [85](#)

Iterative flow commands

denseopc_copy_fragments, [41](#)

denseopc_init_mode, [44](#)

pxopc_weight_layer, [70](#)

refine_prune_markers, [97](#)

refine_region, [98](#)

refine_stop_iterations, [99](#)

- refine_stop_layer, [100](#)
- refine_update_layer, [102](#)
- refine_visible_sim, [103](#)

— J —

Job command, [51](#)

— L —

Large markers, [90](#)

Launch, [17](#)

Layer command, [53](#)

Layer commands

- relationship, [30](#)

Layer types, [53](#)

Layers

- for denseopc, [19](#)
- hotspots, [100](#)
- metal, [109](#)
- outputting, [30](#)
- passing, [30](#)

Litho File command, [27](#)

LPE commands, [17](#)

LPE components, [9](#)

LPE inputs, [10](#)

LPE SVRF, [29](#)

LPE switch, [19](#)

— M —

Marker layers, [53](#)

Marker specification, [88](#), [89](#)

Markers, [90](#), [98](#), [107](#)

Maximum iterations, [99](#)

Memory best practices, [109](#)

Merged regions, [72](#)

Min_frag_len command, [56](#)

Min_space command, [57](#)

Min_width command, [58](#)

Minimum keyword, [14](#)

MRC

- ignore visible region, [110](#)
- internal distance, [59](#), [61](#), [62](#), [63](#)
- minimum fragment length, [56](#), [108](#)
- minimum spacing, [57](#)
- minimum width, [58](#)
- recommendation, [107](#)
- small feature filter, [65](#)

MRC operation, [77](#), [106](#)

mrc_min_length keyword, [59](#)

mrc_min_rect_length keyword, [61](#)

mrc_min_rect_width keyword, [62](#)

mrc_min_square keyword, [63](#)

Multi-patterning, [105](#), [109](#)

— N —

nmBIAS setup files, [78](#)

nmOPC commands

- convert layers, [45](#)
- copy fragments, [41](#)
- displacement, [38](#), [46](#)
- execute, [76](#)
- feedback, [38](#)
- generate setup, [19](#)
- long edges, [42](#), [43](#)
- notchfill, [47](#)
- recommendations, [108](#)
- remove jogs, [48](#), [106](#)
- use correction map, [49](#)

nmOPC setup files, [78](#)

NOT operation, [77](#)

Notch features, [46](#), [48](#)

Nub features, [48](#)

— O —

OPC layers, [53](#)

OPCverify command, [76](#)

OPCverify setup files, [79](#)

OPCverify tips, [109](#)

Operations list, [77](#)

Optimize run, [107](#)

OR operation, [77](#)

Output, [109](#)

Output command, [83](#)

Overlapped regions, [72](#)

— P —

Parentheses, [14](#)

Pattern matching, [92](#)

Pick_region command, [67](#)

Pipes, [14](#)

pxOPC command, [76](#)

Pxopc_init_mode command, [68](#)

Pxopc_transition_region command, [69](#), [74](#)

Pxopc_weight_layer command, 70

— Q —

Quotation marks, 14

— R —

rearrangement_disable command, 72

Reduce testcases, 67

Refine_distance command, 73

Refine_exec command, 76

Refine_map command, 83

Refine_markers command, 88, 97

Refine_markers_limit command, 89

Refine_markers_style command, 90

Refine_patterns keyword, 92

Refine_process keyword, 96

Refine_prune_markers command, 97

Refine_regions command, 98

Refine_stop_iterations command, 99

Refine_stop_layer command, 100

Refine_update_layer command, 102

Refine_visible_sim keyword, 103

Region count, 72

Regions, 11, 84

Reject_duplicate_layers keyword, 105

Repair region size, 69, 73

Repair regions, 11

RET Refine command, 29

Retargeting, 81

— S —

Setlayer commands, 76

Setup files

- called tools, 108

- comments, 32

- general format, 27

- including, 107

- list of commands, 35

- Operation specific, 79

- required commands, 32

- SVRF, 27, 79

- syntax, 32

Setup optimization, 72, 89, 106

Slanted words, 13

smart_cut_disable command, 106

Speed, 72, 90, 106

Square parentheses, 14

SRAFs

- by region, 107

- layer type, 53

SVRF commands

- Litho File, 27

- RET Refine, 29

- supported, 79

Syntax conventions, 13

— T —

Target layers, 53

Tcl commands, 78

Tiles, 108, 109

Transition regions, 109

— U —

Ultraflex command, 13

Underlined words, 14

Usage syntax, 13

— V —

Verification settings, 109

Via layers, 109

Virtual cores, 109

Visible regions, 13, 107, 110

— W —

Warning messages

- large repair region, 74, 90

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.

