SIEMENS EDA

Calibre® pxOPC™ User's and Reference Manual

Software Version 2021.2

SIEMENS

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

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction to Calibre pxOPC

Calibre® pxOPC™ is the Siemens EDA inverse lithography tool.

Calibre pxOPC Overview

Calibre pxOPC is a third-generation mask optimization tool for small and medium-sized layouts. It optimizes every pixel on the mask for maximum process window, taking into consideration depth of focus (DOF), exposure latitude, and mask error enhancement factors (MEEF) for both shapes and sub-resolution assist features (SRAFs).

Process recipes are also easier to create with Calibre pxOPC. SRAFs are automatically inserted; you do not need to run additional tools or create rules for them. Optical proximity correction (OPC) uses pixels instead of fragments, so you do not need to set up fragmentation rules.

With Calibre pxOPC, you can investigate the best possible OPC and SRAF solutions to determine what design rules are feasible. It allows exploration of challenging convergence issues encountered in a foundry environment as new processes are brought on line.

Calibre pxOPC uses the Standard Verification Rule Format (SVRF) and Tcl Verification Format (TVF) controls and command language formats common to all Calibre tools. This manual describes the key concepts, setup file commands, and scripting commands for Calibre pxOPC.

Calibre pxOPC Workflow

Calibre pxOPC can be used in several ways: on its own to develop design rules, to improve the OPC results of other products, or to create SRAF templates.

To improve OPC results, use it as part of Calibre® LPE as documented in the *Calibre LPE User's and Reference Manual*. For SRAF templates, see "SRAF Template Example" on page 252.

Within a Calibre pxOPC run, the work is split into a sequence of "jobs". The jobs run in the order you specify to perform the following specific tasks:

1. Reshape the layout features by growing the target shapes. (OPEN job)
2. Add initial SRAFs to improve the litho quality. (DECORATE or DECORATEDARKSRAF job)
3. Correct the image by suppressing extra printing and improving the edge placement error (EPE) distribution on the main features. (CORRECT job)
4. Refine the corrected image and improve the process window. (REFINE job)
5. Convert contours into MRC-clean manufacturable mask shapes. (LMRC job)
6. Recover potential degradation in quality caused by the conversion to Manhattan polygons. (DETAIL job)
7. Optimize depth of focus with a fixed or floating focus center, without modifying the optical source. (CDOF job)

Typically after the run you examine the results in the Calibre® WORKbench™ or Calibre® LITHOview™ viewer.

Calibre pxOPC Key Concepts

The feature that differentiates Calibre pxOPC from the other Calibre OPC and SRAF offerings is the sequential execution of jobs. Each job, because it performs a specific task, can be held to a "contract". A contract is a measurable result that must be met before advancing to the next job. The final pxOPC result is defective if and only if one of the contracts was not satisfied.

Because of jobs and contracts, Calibre pxOPC recipes are much easier to tune than other Calibre products. Debugging is part of the process: when you identify an unmet contract, the debugging steps in the "Calibre pxOPC Configuration and Tuning" chapter suggest how to fix it.

Calibre pxOPC Requirements

To use Calibre pxOPC, you must set up your environment and files properly.

The following are the requirements for using Calibre pxOPC:

- The CALIBRE_HOME environment variable pointing to a Calibre version 2011.3 or higher installation. Refer to "Setting the CALIBRE_HOME Environment Variable" in the *Calibre Administrator's Guide* for information on setting this variable.
- Licenses for Calibre pxOPC and Calibre® nmDRC™/Calibre® nmDRC-H™. Licenses for Calibre WORKbench and Calibre OPCverify are also recommended. For more information on licensing, refer to the *Calibre Administrator's Guide*.

- SVRF file that specifies the input layout database and results database along with necessary DRC specifications and the RET PXOPC statement.
- Litho setup file that configures Calibre pxOPC. The file can be a separate file or inside the SVRF file in a Litho File statement.
- GDS or OASIS®[1] layout.
- Optical model, as described in the *Calibre WORKbench User's and Reference Manual*. Litho model format is preferred, and required for runs that include a CDOF job.

Syntax Conventions

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

Table 1-1. Syntax Conventions

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

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

Calibre pxOPC Modes of Operation

Calibre pxOPC is usually run in batch mode, which requires an SVRF file to be processed by the Calibre hierarchical platform (calibre -drc). The processing is invoked from a shell command line.

Typically Calibre pxOPC is run multithreaded. The invocation is similar to the following:

```
calibre -drc -hier svrf -turbo -remotefile remote > drc.log
```

where the items in italics represent files.

Calibre pxOPC can also be run within the Calibre WORKbench GUI. Both batch and GUI methods are described in detail in “Running Calibre pxOPC” on page 31.

Calibre pxOPC does not run on the Calibre® FullScale™ platform.

Chapter 2
Calibre pxOPC Configuration and Tuning

The following key sections are covered in this chapter.

Setting Up the SVRF and Litho Setup Files

Calibre pxOPC is run through a litho setup file with pxOPC commands. This can be called either from an SVRF file or from the Calibre WORKbench RET Flow Tool. Setup and SVRF files are created with a text editor. This section provides example files at the end that you can copy into your editor.

Prerequisites

- Name and type of the layout.
- Layer number of the layer you will run Calibre pxOPC on.
- Location of optical models.

Procedure

1. Copy an SVRF file into the text editor. See “Minimum SVRF File Example” on page 17 if you do not have one you use for other OPC runs.
2. Update Layout Path, Layout Primary, and Layout System with the layout name, top cell name, and database type respectively.
3. Make sure the number of the layer you intend to run Calibre pxOPC upon is listed in the Layer statements. (The name does not have to match the name in the database.) If you are using any additional layers as correction or weight layers, they should also be listed.
4. (Optional) Copy the layer into the results database.
5. (Optional) Smooth the target layer using setlayer curve. This is not required, but smoothed targets converge more quickly than rectilinear targets. Copy the smoothed layer into the results database.

   When creating the smoothed target, verify it touches the drawn polygons. Avoid gaps greater than 2 dbu along edges, as it can cause excessive EPE in the results.
6. Add the Calibre pxOPC command, RET PXOPC. It has the following form:

   ```
   out_layer = RET PXOPC in_layers FILE setup_file MAP setup_file_layer
   ```

   where

   - *out_layer* is a new layer that will get a copy of *setup_file_layer*.
   - *in_layers* are the target, smoothed target, correction, and weight layers.
   - *setup_file* is the name of the Litho setup file you will be creating next. In the example SVRF file, it is “litho.in”.
   - *setup_file_layer* is the name of the layer in the setup file that is passed back when the run is complete. In the example setup file, it is “opcout”.
7. Create the setup file. Paste in the contents of Example 2-2 on page 18 or any other Calibre pxOPC setup file. Do not use a setup file for other RET/OPC tools as some commands interfere with the Calibre pxOPC tool.

   - If the SVRF file you are working with uses in-line setup files, add a LITHO FILE statement for the Calibre pxOPC setup file.
   - If you would prefer an external file, open an additional text editor.

   Be sure the name you give this file matches the *setup_file* in step 6.
8. Update the modelpath and optical_model_load statements with the models you intend to use for this simulation.

9. For each layer passed in as *in_layers*, include a corresponding layer statement in the main section of the setup file. The layers should be declared as type “hidden”. See “layer (for main setup file)” on page 81.
10. If necessary, modify the pxOPC-specific commands in the pxopc_options block and the setlayer pxopc statement.
    - If you added or removed layers in Step 9, also add or remove them here.
    - Verify the pw_condition statements are set for the process windows you intend to simulate. If you are using the example file, you will probably want to add process conditions for an inner (conditions leading to smaller shapes) and an outer (conditions leading to larger shapes) process corner.
    - If this file already has job-specific overrides in the job sections, comment them out. The optimization problem is sensitive to initial conditions, and the later instructions for tuning assume you are starting with initial conditions.
    - In the setlayer pxopc statement, make sure that the name of the derived layer is the same as *setup_file_layer* in Step 6 and that the name of the layer in MAP is the name of the target layer in pxopc_options.
11. Save the file(s).

Results

You have two files resembling the ones shown in Example 2-1 and Example 2-2.

Example 2-1. Minimum SVRF File Example

```
//Comments begin with //. Blue text links to the reference page.
//Standard input definitions.
LAYOUT PATH "input.oas"
LAYOUT PRIMARY '*'
LAYOUT SYSTEM OASIS
LAYOUT USE DATABASE PRECISION YES

//Only required for hierarchical runs, as used in this book.
LAYOUT ULTRA FLEX YES

//Specify output database (RDB).
DRC RESULTS DATABASE "result.oas" OASIS

//Identify the layer in "input.oas" that you care about.
LAYER target 1

//Copy it into the RDB (optional)
target {COPY target} DRC CHECK MAP target 1

//Use OPCverify to smooth the target layer before pxOPC (optional).
trgtsmth = LITHO OPCVERIFY FILE "smooth.in" target MAP trgtsmth
trgtsmth {COPY trgtsmth} DRC CHECK MAP trgtsmth 2
```

```
//This is the OPCverify Litho Setup File.
LITHO FILE smooth.in [ /*
  layer CONTACT
  setlayer trgtsmth = curve CONTACT
*/ ]

//PXOPC creates the mrc_opc layer for the optimized mask.
mrc_opc = RET PXOPC target trgtsmth FILE "litho.in" MAP opcout
mrc_opc {COPY mrc_opc} DRC CHECK MAP mrc_opc 20

//End of File
```

Example 2-2. Litho Setup File for pxOPC

```
# Litho setup files mark comments with the number sign (#).
# If this were in the SVRF file, the next line would not be commented:
# LITHO FILE litho.in [ /*

# Identify the model directory and any models. This example only
# uses one optical model, but most real runs use several.
modelpath models
optical_model_load f0 opticalf0
resist_model_load resist_mod resist.mod

# It is strongly recommended to set tilemicrons because the default of
# 100 microns is roughly 5 times what is optimal for pxOPC process nodes.
tilemicrons 19 exact

# These layers are "hidden" so you do not need a background here. There
# needs to be a layer declared for every layer passed by RET PXOPC.
layer target hidden clear
layer trgtsmth hidden clear

# The processing_mode command is only required when the invocation
# includes "-hier".
processing_mode flat

# The pxopc_options block is like a setup file inside the setup file.
# It is called by the setlayer pxopc command after the block. It specifies
# the conditions and layer properties that Calibre pxOPC will use.
pxopc_options SIMULATION_OPTIONS {
   #general section; defines settings for all jobs.
   background 0.0215 -0.0085
   layer target CORRECTION 0.9785 0.0085
   layer trgtsmth TARGET 0.9785 0.0085
   pw_condition NOMINAL optical f0 dose 1 aerial 0.21 weight 1.0

   # job-specific sections; can specify overrides.
   job 1 open
   job 2 decorate
   job 3 correct
   job 4 refine
   job 5 lmrc
   job 6 detail
}
```

```
# This command runs pxopc and creates the "opcout" layer. The "MAP
# opcout" in the RET PXOPC line of the SVRF file means this layer
# will be passed back to the SVRF level where it is called "mrc_opc".
setlayer opcout = pxopc target trgtsmth MAP target \
      OPTIONS SIMULATION_OPTIONS

# The backslash (\) in the setlayer command allows it to wrap lines.
# If this setup file were in-line in the SVRF, the next line would
# not begin with a "#":
# */  ]
```

Related Topics

Calibre pxOPC Best Practices

Standard Verification Rule Format (SVRF) Manual

LAYOUT ULTRA FLEX [Calibre Post-Tapeout Flow User's Manual]

Rule Check Statements [Calibre Verification User's Manual]

LITHO OPCVERIFY [Calibre OPCverify User's and Reference Manual]

Setting Up the SVRF File Using the Calibre pxOPC Template Wizard

The Calibre pxOPC Template Wizard creates an initial pxOPC recipe that includes typical main optimization metrics such as MRC and extra print control. The generated pxOPC recipe is a good starting point that the user can tune for optimal results.

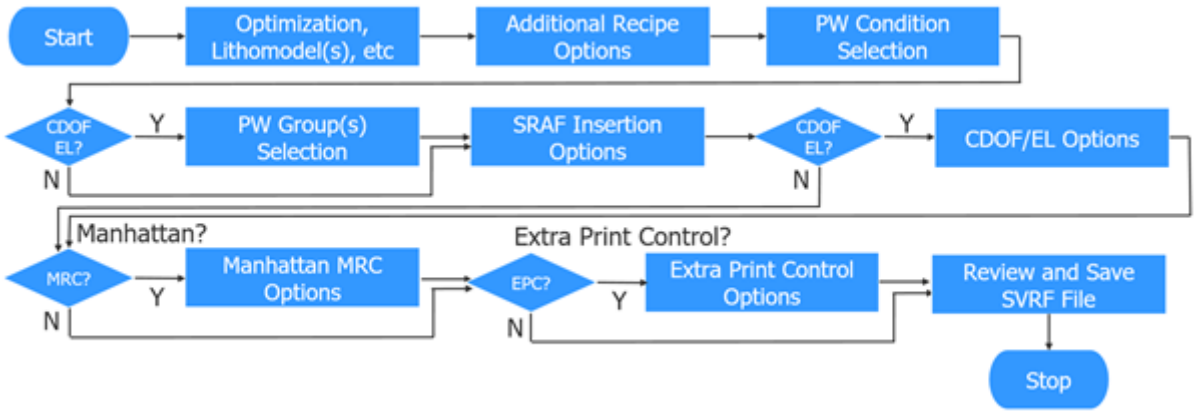
Prerequisites

- Name and type of the layout.
- Layer number of the layer you will run Calibre pxOPC on.
- Location of optical models.

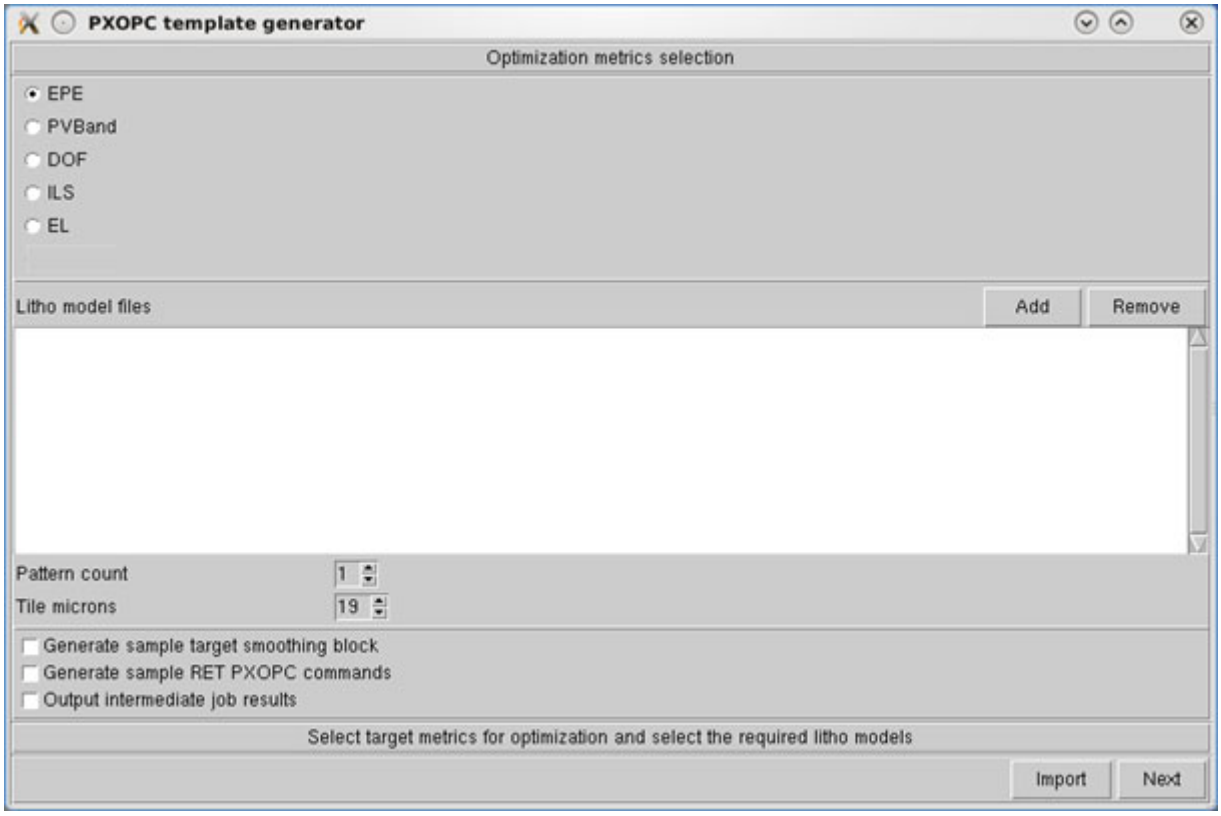
Procedure

1. In Calibre WORKbench, invoke the Calibre pxOPC Template Wizard from the **Macros > PXOPC template wizard** menu item.

Figure 2-1. Calibre pxOPC Template Wizard Workflow



2. Select the optimization metric and load the litho model file(s).



a. Select the optimization metric:

- **EPE** — Optimizes to minimize edge placement error (EPE).
- **PVBand** — Optimizes to minimize the process variation band (PV band).
- **DOF** — Optimizes the process window to maximize common depth of focus (CDOF).

- **ILS** — Optimizes the process window to maximize image log slops (ILS).
- **EL** — Optimizes the process window to maximize exposure latitude (EL).
- **MEEF** — This feature is unavailable in the current release.

b. Load the litho model files.

   i. Click **Add** and specify the folder containing the *Lithomodel* file.

   ii. For multi-patterning, increase the **Pattern count** to match the number of patterning steps and load litho model files.

   iii. (Optional) Adjust the tile size. See Recommended tilemicrons Settings in "tilemicrons" on page 108 for tile size recommendations.

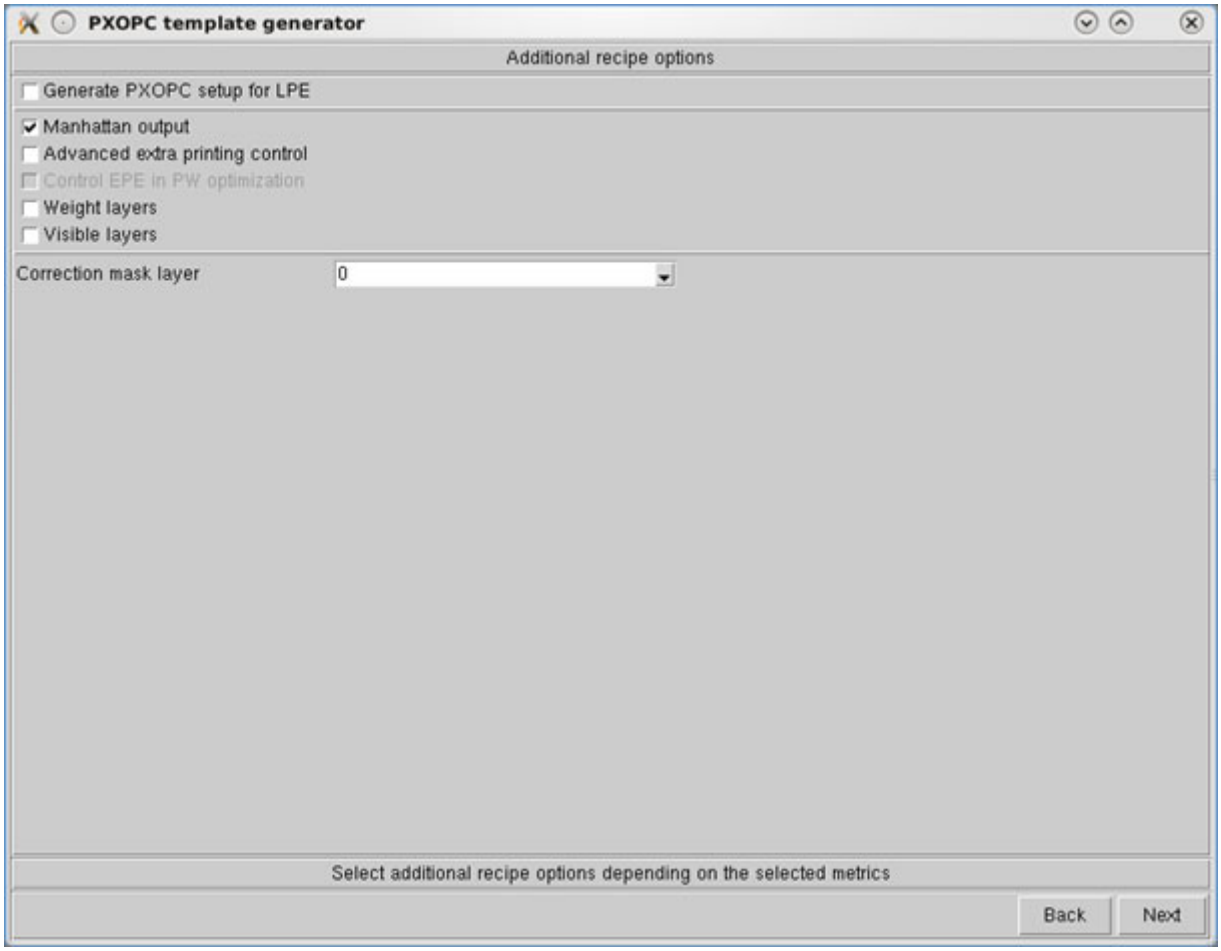
c. (Optional) If desired, enable one or more of the following options:

   - **Generate sample target smoothing block** — Adds initial code to create a smoothed target layer.
   - **Generate sample RET PXOPC commands** — Adds initial code to invoke Calibre pxOPC.
   - **Output intermediate job results** —Adds LASTJOB commands to each job's setlayer statements to report intermediate results.

d. (Optional) To load an existing SVRF file as the template, click **Import** and select the file.

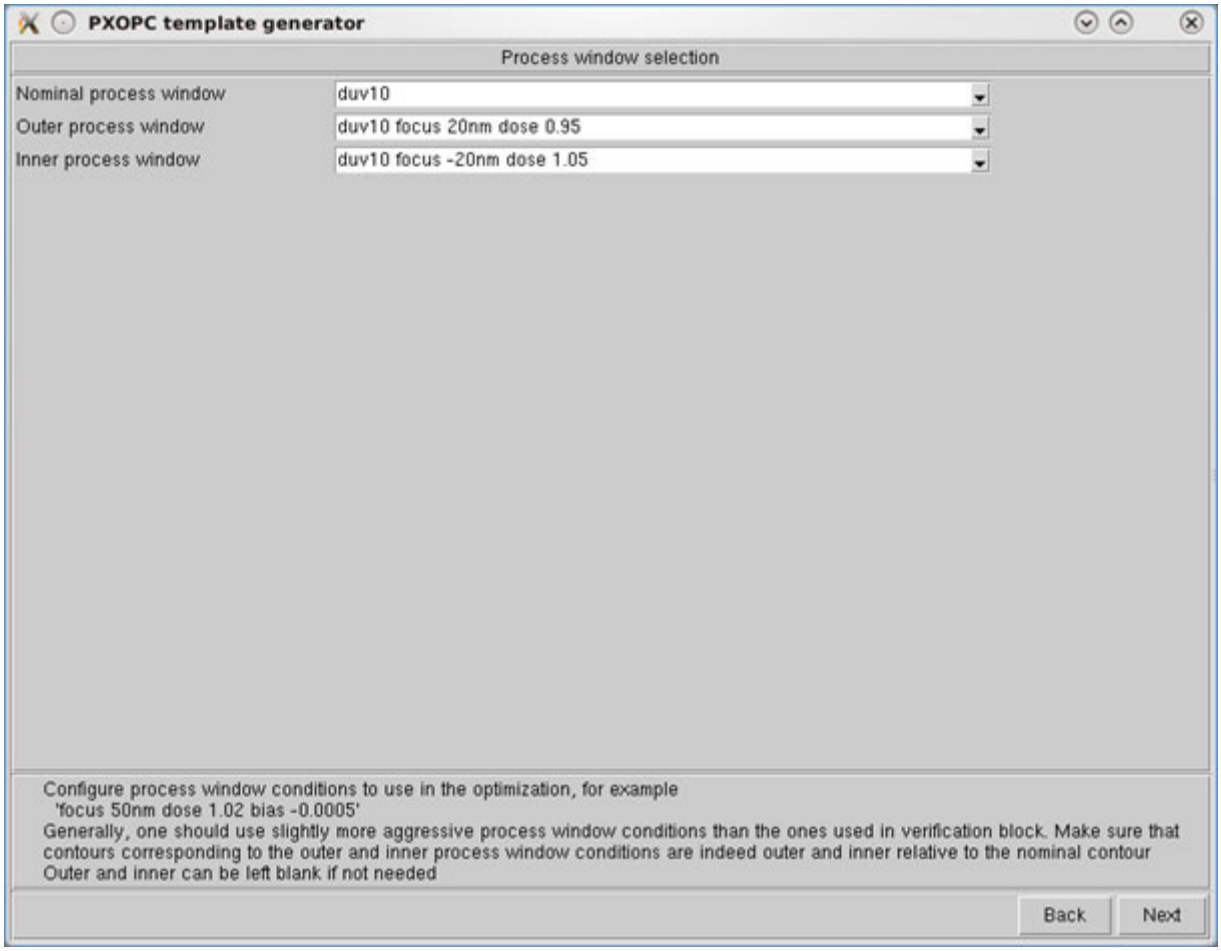
e. Click **Next**.

3. Select additional recipe options:



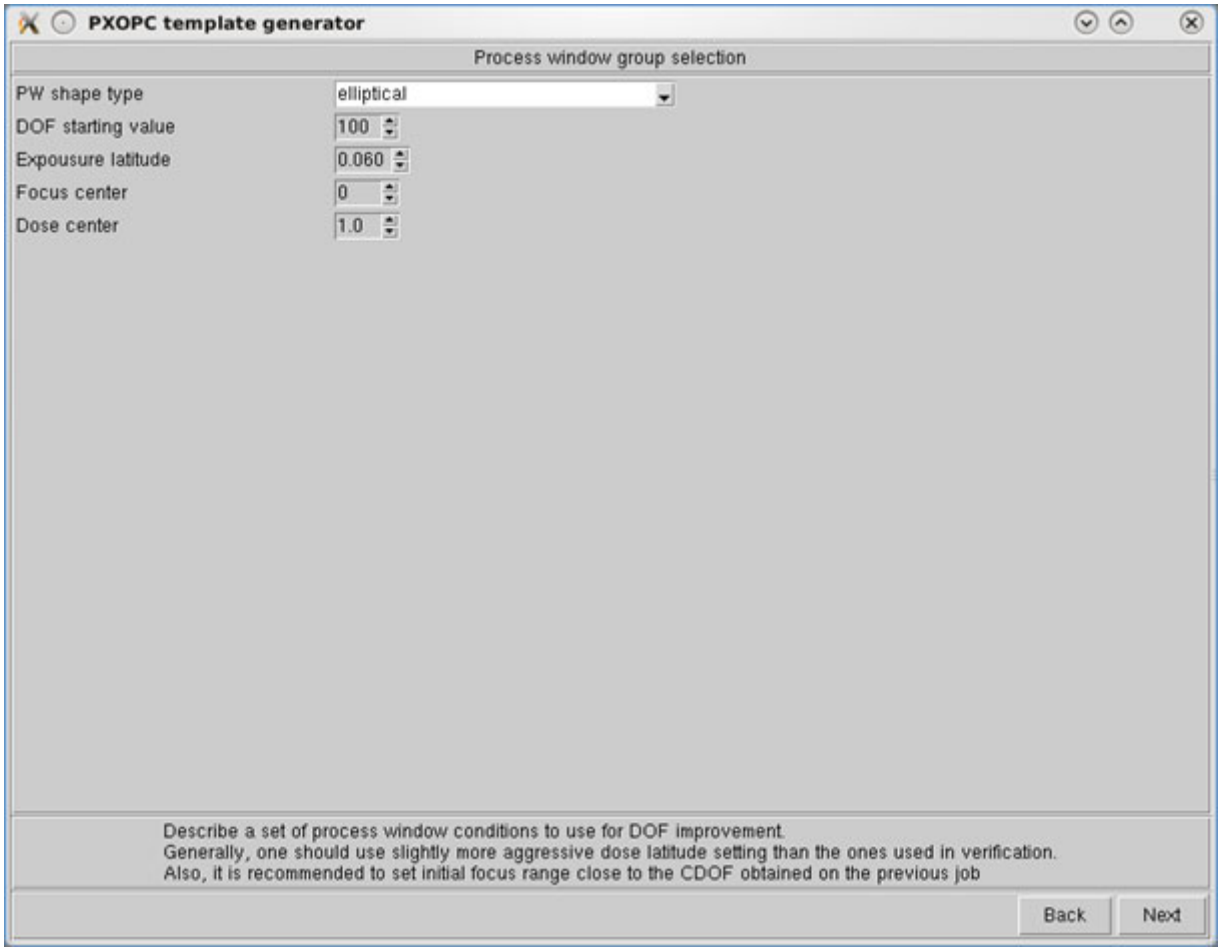
- **Generate PXOPC setup for LPE** — Adjusts pxOPC setup options to generate a recipe compatible with LPE.
- **Manhattan output** — Adds Lmrc and Detail jobs to produce an output mask with Manhattan rectilinear polygons.
- **Advanced extra printing control** —Adds pw_condition and pw_select statements to the Correct and Refine jobs to eliminate SRAF printing.
- **Weight layers** — This feature is unavailable in the current release.
- **Visible layers** — This feature is unavailable in the current release.
- **Correction mask layer** —Specifies a layer number for the correction mask layer.

4. Select process window conditions:



Select the nominal, outer, and inner process window conditions. The process window conditions are read from the imported lithomodel file. Click **Next**.

5. (Optional) If DOF or EL is selected as the optimization metric, then configure the process window group selection options:



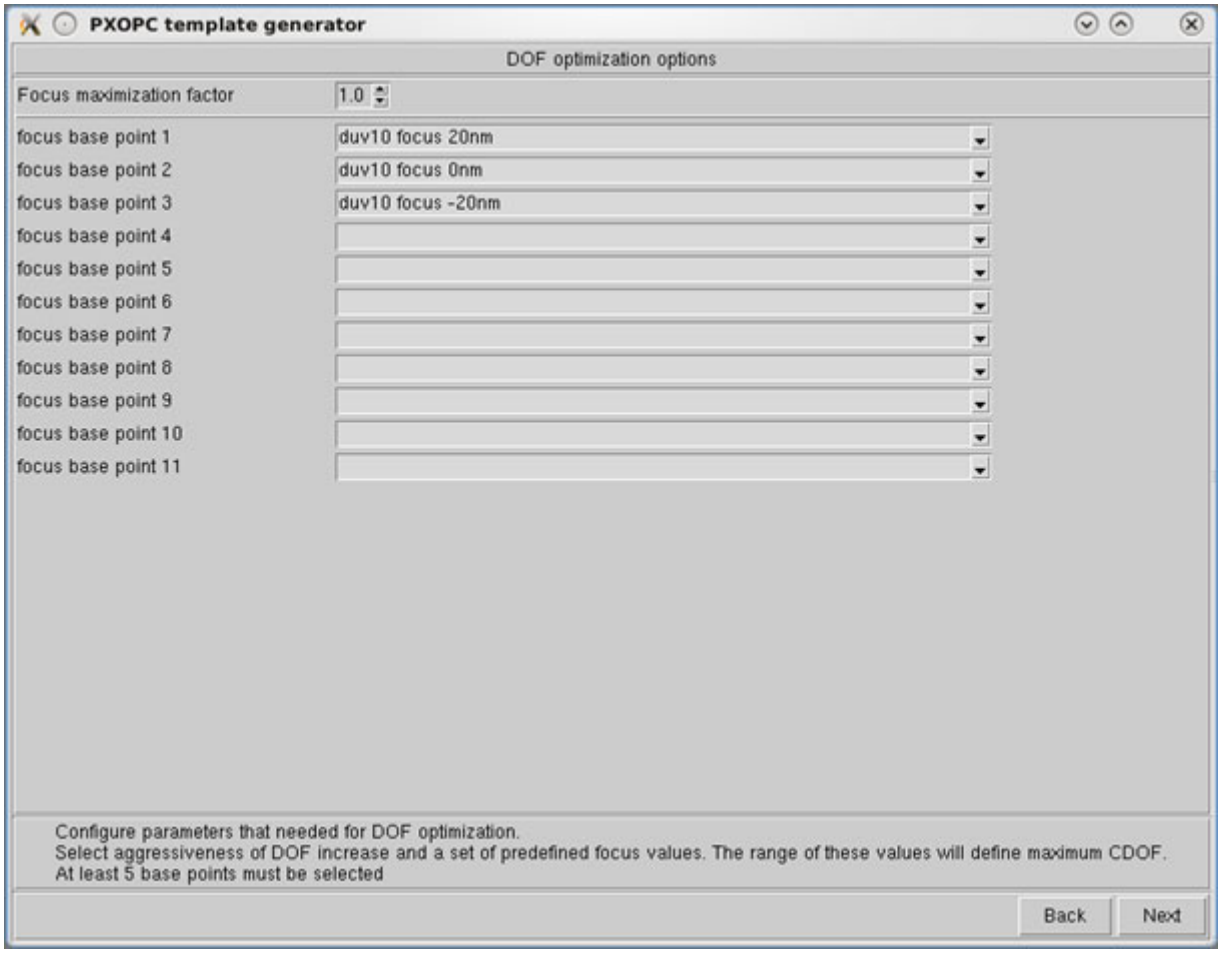
Depending on the optimization metric, the following options appear:

- **PW Shape type** — Specifies the shape inscribed inside the common process window.
- **DOF starting value** and **Exposure latitude starting value** — Specifies a minimum starting value for DOF or EL optimization.
- **DOF value** and **Exposure latitude** — Specifies a fixed DOF or EL that meets the common process window requirements.
- **Focus center** and **Dose center** — Sets the center of the process window shape.

6. Configure the SRAF insertion options:

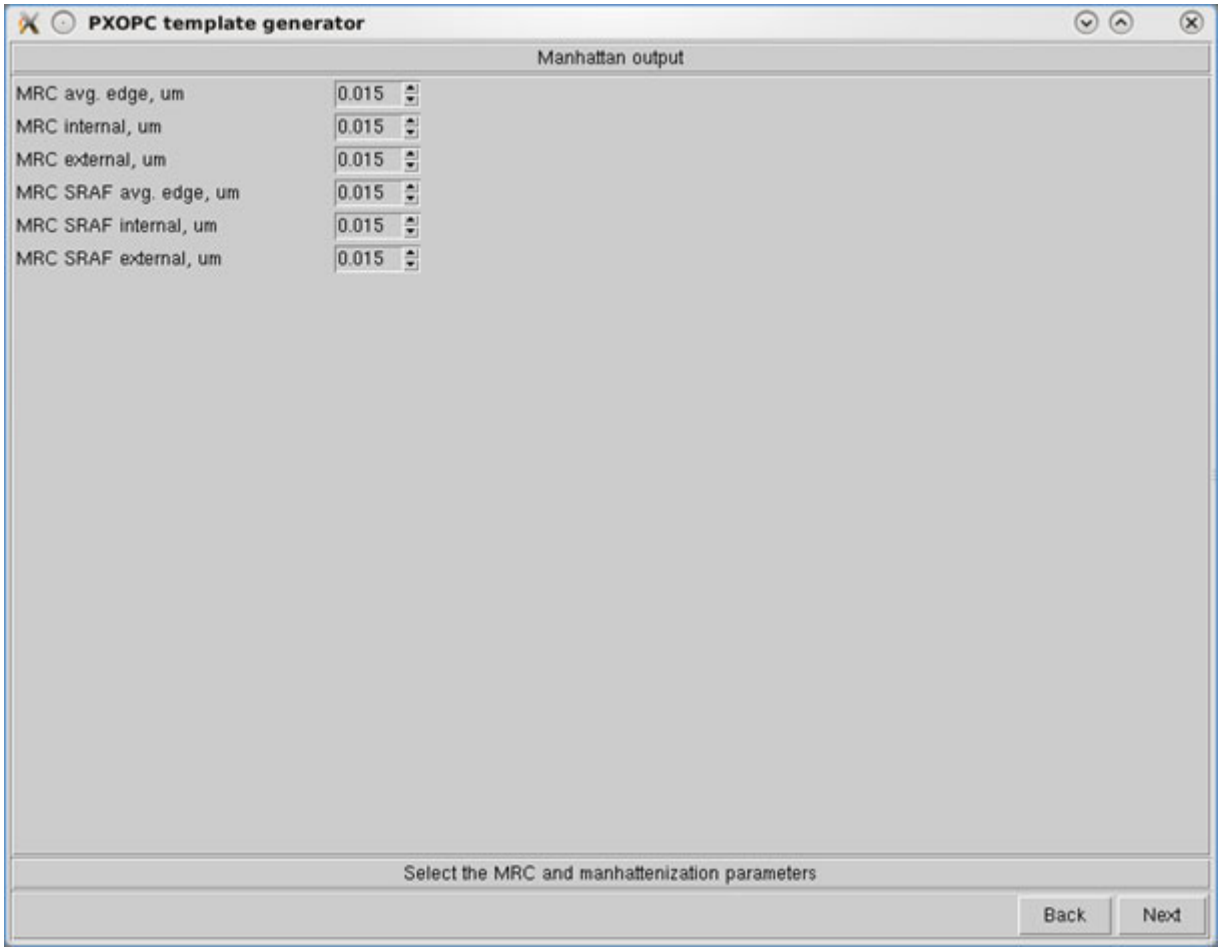
- **Min. distance from the main features** — Specifies the minimum distance between the main feature and positive SRAFs. Adds scatter_offset to the Decorate job.
- **Max. distance from the main features** —Specifies how far from the target the SRAF rings should extend. Adds scatter_belt to the Decorate job.
- **SRAF insertion iterations** —Specifies how many iterations the optimization algorithm in the Decorate job should run.

7. (Optional) If DOF or EL is selected as the optimization metric, then configure the DOF or EL optimization options.



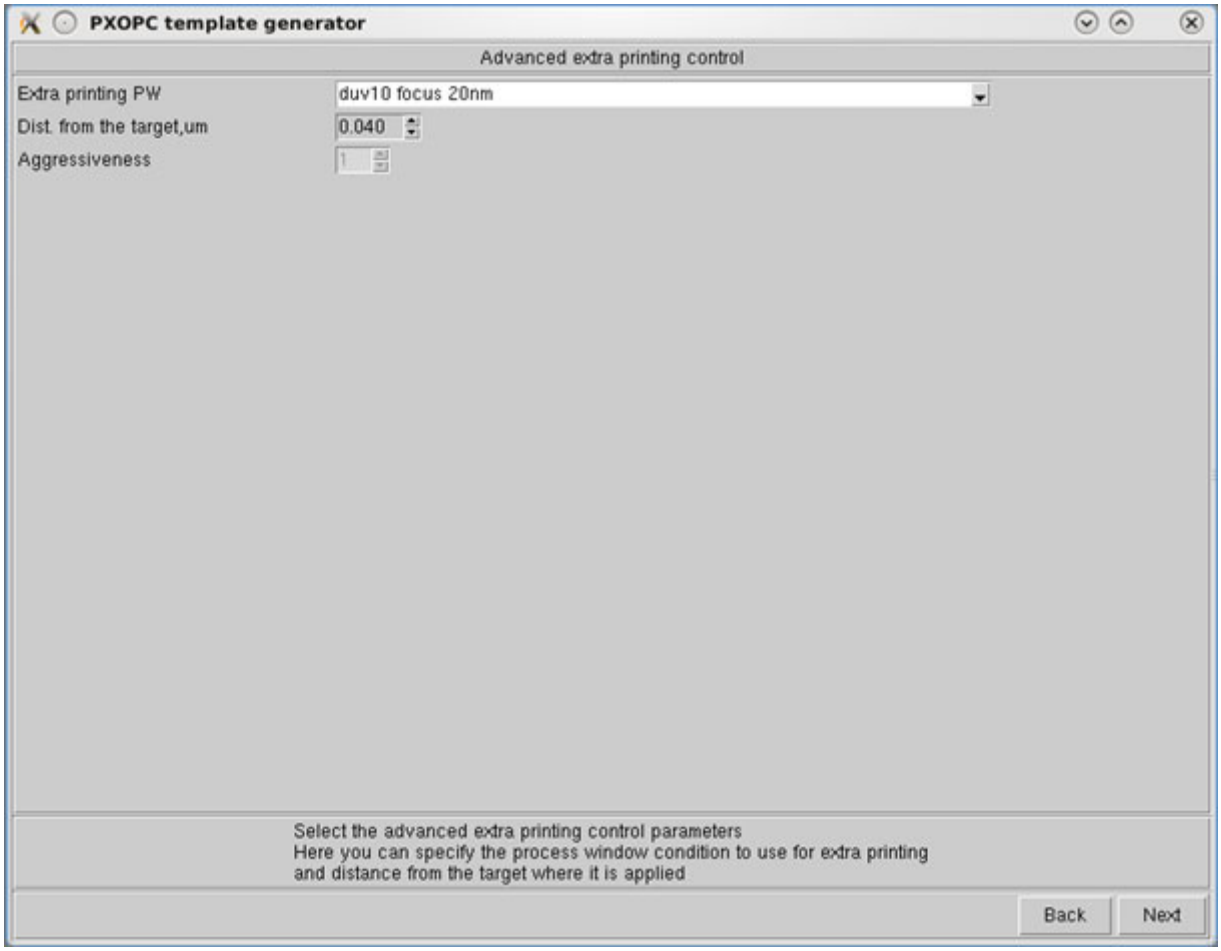
Select focus base points from the litho model that will be used in the optimization. At least 5 base points must be selected. A wide range of focus base points define the maximum DOF or EL. To increase the DOF or EL from the optimization, tune the value of the **Focus maximization factor** or **Exposure latitude** factor.

8. Select the MRC options for Manhattan output:



- **MRC avg. edge, um** — Sets a target edge length for MRC geometries. Adds mrc_min_edge to the Lmrc job.
- **MRC internal, um** — Sets the minimum distance between two internal edges. Adds mrc_min_internal to the Lmrc job.
- **MRC external, um** — Constrains external spacing during MRC. Adds mrc_min_external to the Lmrc job.
- **MRC SRAF avg. edge, um** — Sets a target SRAF edge length for MRC geometries. Adds mrc_min_assist_edge to the Lmrc job.
- **MRC SRAF internal, um** — Constrains internal spacing during MRC for SRAFs only. Adds mrc_min_internal_sraf to the Lmrc job.
- **MRC SRAF external, um** — Constrains external spacing during MRC for SRAFs only. Adds mrc_min_external_sraf to the Lmrc job.

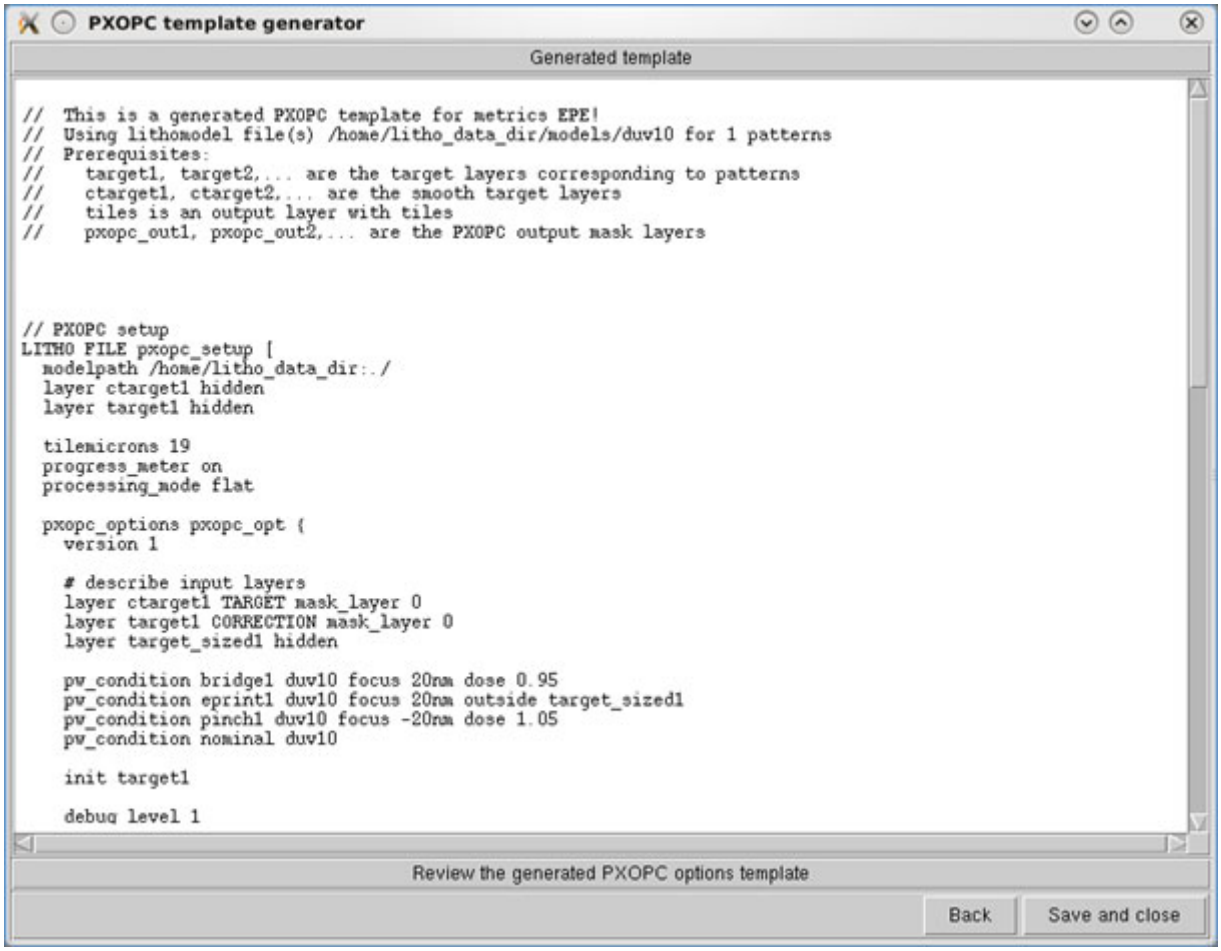
9. (Optional) Select the options for Extra Print Control:



The following options are available if advanced extra printing control is enabled in **Additional recipe options**:

- **Extra printing PW** — Specifies the process window condition that extra print control is applied to.
- **Dist. from the target, um** — Specifies the distance from the main feature edge where extra print control is applied to.
- **Aggressiveness** — (Optional) Specifies the emphasis on extra print control when the optimization metric is set to DOF or EL.

10. Review and save the generated template:



Click **Save and close** to name and save the SVRF file.

11. Add layer processing information to the saved template:

    a. Open the SVRF file in a text editor. See “Minimum SVRF File Example” on page 17 if you do not have one you use for other OPC runs.

    b. Add Layout Path, Layout Primary, and Layout System with the layout name, top cell name, and database type respectively.

    c. Make sure the number of the layer you intend to run Calibre pxOPC upon is listed in the Layer statements. (The name does not have to match the name in the database.) If you are using any additional layers as correction or weight layers, they should also be listed.

    d. (Optional) Copy the layer into the results database.

    e. (Optional) Smooth the target layer using setlayer curve. This is not required, but smoothed targets converge more quickly than rectilinear targets. Copy the smoothed layer into the results database.

When creating the smoothed target, verify it touches the drawn polygons. Avoid gaps greater than 2 dbu along edges, as it can cause excessive EPE in the results.

f. Add the Calibre pxOPC command, RET PXOPC. If **Generate sample RET PXOPC commands** was enabled during optimization metrics selection, then edit the PXOPC RET commands section of the file. The section has the following form:

```
out_layer = RET PXOPC in_layers FILE setup_file MAP
setup_file_layer
```

where

- *out_layer* is a new layer that will get a copy of *setup_file_layer*.
- *in_layers* are the target, smoothed target, correction, and weight layers.
- *setup_file* is the name of the Litho setup file you will be creating next. In the example SVRF file, it is “litho.in”.
- *setup_file_layer* is the name of the layer in the setup file that is passed back when the run is complete. In the example setup file, it is “opcout”.

g. Save the file.

Running Calibre pxOPC

You can run Calibre pxOPC files three ways, two of them from the command line and one from Calibre WORKbench.

In general, when optimizing a Calibre pxOPC recipe, running on a section of layout is the most efficient. This is described in "Using the RET Flow Tool in Calibre WORKbench".

Using the RET Flow Tool in Calibre WORKbench

The RET Flow Tool can provide a template setup file for you to start with. You can iteratively tune settings on a small section of layout without the need of an SVRF file.

Restrictions and Limitations

- Do not use this method on large layouts. The RET Flow Tool always runs on a single core, even when Calibre WORKbench was launched in multithreaded mode.

Prerequisites

- A GDS or OASIS setup file
- An optical model

Procedure

1. Open the layout in Calibre WORKbench.

   ```
   calibrewb layout
   ```

2. Zoom into a portion of the layout that contains a good variety of different geometries.
3. Open the RET Flow Tool by clicking the **Litho** button, or selecting **Litho > RET Flow Tool**.
4. Select **File > Add New Session**, and in the New Session Dialog choose **PxOPC** and Generate Default Session then click **OK**.

   Note

   If you have a setup file you would prefer to start from, right-click *Untitled_PxOPC.lsf* and select **Open Setup** to load the setup file.

5. In the **Layers Mapping** tab at the bottom left, use the dropdown menu in the WB # - Name column to map design layers to the setup file.

- The setup “target” is the layer to adjust. In the template setup file, it is of type “correction”, which means it is adjusted.
- The setup “trgtsmth” represents the achievable target, normally created by setlayer curve_target before running Calibre pxOPC. If your design does not have a curved target, map it to the same layer as the setup “target”.

Generally, a Calibre pxOPC setup file has a correction layer to receive adjustments and a target layer that represents the ideal shape. If you are using your own recipe, map the layers appropriately.

6. Click the **Options** button under the **PXOPC** button and adjust the Output Layer(s) Options as needed.

Caution

The recipe shown in the text area is overwritten when you click **Default**; any changes you have made to the text file are lost.

Additionally, once you open the Options pane under Default, the template recipe even in new Calibre WORKbench sessions uses the values shown in the Options pane. For example, the initial template recipe includes job 5 but the recipe generated by Options does not.

7. (Optional) Click **Options** under **Default**, and consider whether any of the Jobs Options or Curve Options may be helpful. To try them in the template recipe, make changes and click **Default**.

   Alternatively, consult the commands listed in “Litho Setup File Commands” on page 65 and edit the setup file manually.

8. To run the setup file, click **PXOPC** in either the RET Flow Tool window or the Flow pane on the left of the main Calibre WORKbench window.

9. Continue to edit the file and test settings iteratively. When you are satisfied with the results, in the RET Flow Tool window select **File > Save Setup File (<*session name*>)**.

Results

The Calibre WORKbench main window shows “pxOPC…” in the bottom border, and the terminal window shows the typical Calibre transcript being written.

After Calibre WORKbench completes the pxOPC operation, the additional layers are written to the layout area. If you used the file in “Setting Up the SVRF and Litho Setup Files” Example 2-2, the layer is “opcout.”

Related Topics

Calibre WORKbench: RET Flow Tool User’s Manual

Running Calibre pxOPC From the Command Line

You can run Calibre pxOPC from the command line, as with any other Calibre product. This is the fastest method if you want to run on the full layout.

Prerequisites

- GDS or OASIS layout.
- SVRF file to run Calibre pxOPC upon a layer in the file.
- Calibre pxOPC setup file, called by the SVRF file. It may be a separate file or inside the SVRF file.

Procedure

1. Open a Linux terminal window.
2. Run Calibre pxOPC with the following invocation:

   ```
   calibre -drc -hier svrf -turbo -remotefile remote > drc.log
   ```

   where

   - "calibre -drc -hier" runs hierarchical Calibre nmDRC.
   - *svrf* is the name of the SVRF file you created in "Setting Up the SVRF and Litho Setup Files".
   - "-turbo" instructs Calibre nmDRC-H to use multithreaded parallel processing for both the SVRF operations and the Litho operations.
   - "-remotefile *remote*" provides configuration information necessary for parallel processing on hosts that are not of the same hardware as the computer on which you invoked the run. Remote configuration files are the same for all Calibre tools. See the *Calibre Administrator's Guide* for more information.
   - "> *drc.log*" redirects the output to a log file, *drc.log*.

Results

While the run is processing, you will see a transcript being written to the terminal. When the run is complete, it shows the following lines:

```
--- CALIBRE::DRC-F EXECUTIVE MODULE COMPLETED. CPU TIME = ...
--- TOTAL RULECHECKS EXECUTED = ...
--- TOTAL RESULTS GENERATED = ...
--- DRC RESULTS DATABASE FILE = ...

--- CALIBRE::DRC-F COMPLETED - <date>
--- TOTAL CPU TIME = ... REAL TIME = ...
--- SUMMARY REPORT FILE =
```

The results of the run are in the file named in the DRC Results database file.

Running Calibre pxOPC From Calibre LPE

Calibre Local Printability Enhancement, or Calibre LPE, runs Calibre pxOPC on marked regions of a full-chip layout.

Prerequisites

- OASIS layout with markers identifying regions to correct.
- SVRF file including the Calibre LPE command RET REFINE.
- Licenses for all products called by Calibre LPE.
- Calibre pxOPC setup file.

Procedure

1. Modify the Calibre pxOPC setup file to meet Calibre LPE requirements.
   a. Make a copy of the setup file.
   b. In the setup file to be called by Calibre LPE, change “refine” to “finetune”.
   c. Verify that any MRC settings are consistent with other products called in the RET REFINE setup file.
2. Modify the SVRF file to include the pxOPC setup file. This can be done with the INCLUDE SVRF command, or by pasting the setup file into a LITHO FILE statement.
3. Remove any RET PXOPC calls from the SVRF file. Calibre pxOPC will be called by “refine_exec …pxopc” in the RET REFINE setup file.
4. Save the files and invoke Calibre LPE:

   ```
   calibre -drc -hier svrf -turbo -remotefile remote > drc.log
   ```

   The meaning of the switches is documented in “Running Calibre pxOPC From the Command Line”.

Results

While the run is processing, you will see a transcript being written to the terminal. When the run is complete, it shows the following lines:

```
--- CALIBRE::DRC-F EXECUTIVE MODULE COMPLETED. CPU TIME = ...
--- TOTAL RULECHECKS EXECUTED = ...
--- TOTAL RESULTS GENERATED = ...
--- DRC RESULTS DATABASE FILE = ...

--- CALIBRE::DRC-F COMPLETED - <date>
--- TOTAL CPU TIME = ... REAL TIME = ...
--- SUMMARY REPORT FILE =
```

The results of the run are in the file named in the DRC Results database file.

Related Topics

Calibre LPE Users and Reference Manual

Checking Initial Results

Calibre pxOPC is designed to provide reasonable results for many situations with no or minute customizations. To determine where the customizations are needed, each step (job) is checked against its contract. To best know where to tune, first you must see where the contracts are not being met by default settings.

Prerequisites

- GDS or OASIS layout containing the run results.
- SVRF file to run Calibre pxOPC upon a layer in the file.
- Calibre pxOPC setup file, called by the SVRF file. It may be a separate file or inside the SVRF file.
- Calibre WORKbench license.

Procedure

1. After the run completes, open the results database in a Calibre viewer. In the example, the results are written to *result.oas*. To open this file in Calibre WORKbench:

   ```
   calibrewb result.oas
   ```

2. Hide all but the final layer. It should show post-OPC features and SRAFs. To simulate the likely final image:

   a. If the layout is large, zoom into a region.

   b. Select **Litho > RET Flow Tool**.

   c. Add required settings by either extracting them from an SVRF file (**File > Extract from SVRF**), opening an external Litho setup file, or by clicking **Default** in the middle of the toolbar.

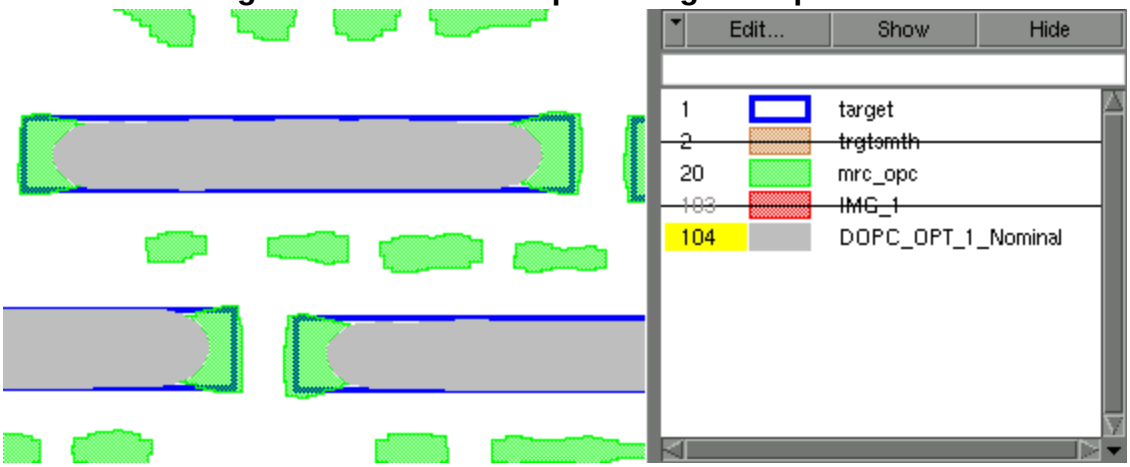
   d. Click **PI** to run print image.

3. Compare the PI layer and the target layer for any issues.

Results

Your layout now shows both Calibre pxOPC output and a simulated print image, as shown in Figure 2-2. In the figure, the Calibre pxOPC output is on layer 20 and the simulated print image is on layer 104.

Figure 2-2. Default Output Using Example Files



Fixing Initialization Problems

If you notice that the wrong layer received OPC correction, or that the process window conditions were not aggressive enough, you need to fix the general section of the pxopc_options block.

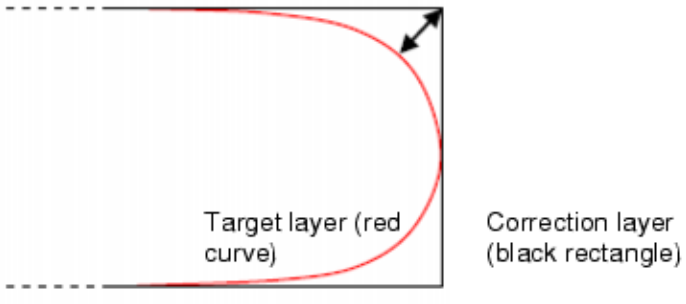
Prerequisites

- Calibre pxOPC setup file
- Optical models for multiple dose conditions

Procedure

1. Verify the correct geometries are being passed:
   - The Layer statement in the SVRF file is set to the correct design layer.
   - If you have smoothed the target layer, the RET PXOPC call passes the smoothed layer to the setup file as the target layer, and the original design layer (the design intent) as a correction layer. A smooth target is preferred over a rectilinear target.
   - If you have smoothed the target layer, check the amount. Measure as shown in Figure 2-3. The offset of the target from the corner should be in the range of 1⁄6 to 1⁄4 of the critical distance (CD). For line ends, it should be slightly greater than 1⁄4; for example, 15 to 17 nm for a 60 nm CD. For contacts, it should be closer to 1⁄6; for example, 10 to 12 nm for a 60 nm CD.

Figure 2-3. Where to Measure Offset on Smoothed Shapes

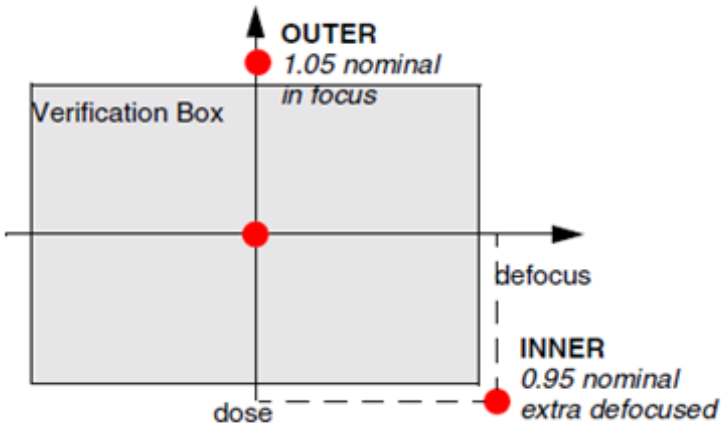


- In the setlayer pxopc call, the order of layers matches the order of layer statements in the pxopc_options block. The layers passed by setlayer pxopc are mapped in order to the layer statements, just as the RET PXOPC input layers are mapped in order to the layer statements in the main setup file.
- Any SRAFs are on a separate layer from the main features.
- If multiple layers are passed, set the init command to use the design intent layer.

2. Increase the process window. For best results, use at least three process conditions: nominal, underexposed, and overexposed.

   - **Bright Features** (more common)

Figure 2-4. Recommended Process Conditions for Bright Features




```
pw_condition NOMINAL \
    optical f0 dose 1 aerial 0.21

pw_condition OUTER \
    optical f0 dose 1.05 aerial 0.21

pw_condition INNER \
    optical f40 dose 0.95 aerial 0.21
```

Tip

To prevent SRAFs from printing after MRC is performed, the inner and outer process corners should use 1 to 2% larger exposure latitude and 10% larger focus latitude than the similar process corners in the Calibre OPCverify or other verification tool. Using the verification box of Figure 2-4, this means the OPCverify conditions should be inside the box, but the pxOPC conditions are outside the box (shown as red dots).

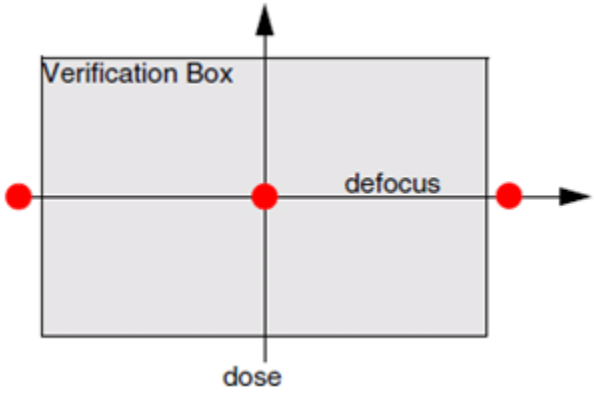
If there is an obligatory mask bias, add it to the OUTER condition and subtract it from the INNER condition. For example, for a mask bias of 1 nm, the statements become:

```
pw_condition NOMINAL optical f0  dose 1    aerial 0.21
pw_condition OUTER   optical f0  dose 1.05 aerial 0.21 \
   mask_size  0.001
pw_condition INNER   optical f40 dose 0.95 aerial 0.21 \
   mask_size -0.001
```

- **Dark Features**

  When the absolute value of the background is greater than the absolute value of the transmission of the correction layer, try these process conditions.

Figure 2-5. Recommend Process Conditions for Dark Features




```
pw_condition NOMINAL \
    optical f0 dose 1 aerial 0.21

pw_condition FOC_NEG \
    optical f-60 dose 1 aerial 0.21

pw_condition FOC_POS \
    optical f+60 dose 1 aerial 0.21
```

3. Re-run Calibre pxOPC and check the output. If the correct geometries are being simulated but the output still needs improvement, proceed to "Fixing Open Job Problems".

Results

The layers are correctly configured. Process corners are considered in the optimization.

Fixing Open Job Problems

The Open job reshapes (or "opens") the main layout features so that they print under nominal conditions. The contract of the Open job is to deliver printed shapes at nominal conditions. Contacts deviate no more than 30% and lines no more than 40% from the target critical dimension (CD).

For dipole illuminations and rectangular contacts, the main features are elongated in one axis and shrunk in the other. This is expected and is not an error.

Prerequisites

- Calibre pxOPC setup file.

- Nominal condition is defined in a pw_condition statement.

Procedure

1. To see the “job 1 open” output, add a setlayer pxopc command with “LASTJOB 1” to your pxOPC setup file. Using the layer names from “Litho Setup File for pxOPC” on page 18, the command looks as follows:

   ```
   setlayer px.open = pxopc target trgtsmth MAP target \
      OPTIONS SIMULATION_OPTIONS LASTJOB 1
   ```

   Also add a rule check in the SVRF file to copy it to the results:

   ```
   open_output {RET PXOPC target trgtsmth FILE "pxopc.in" MAP px.open }
   DRC CHECK MAP open_output 21
   ```

   Note

   Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. If the print image-to-design difference is too large (for contacts, greater than 0.3 CD; for lines, greater than 0.4 CD), try one of these methods:

   - Increase the number of iterations in increments of two.

     For example, the default number of iterations for the Open job is four. Start by increasing it to six:

     ```
     job 1 open
        iterations 6
     ```

   - Reduce the rate to less than 1 (the default). It can be as small as 0.1.

3. If the mask is not rectilinear, verify that the original design layer (the design intent) is present in both the main setup file and the pxopc_options block as a *correction* layer.

4. Re-run Calibre pxOPC and check the output by running print image on the layer created by the Open job. If the images are large enough but the output still needs improvement, proceed to “Fixing Decorate Job Problems”.

Results

A print image created from the output of the Open job shows main features that are within at least 60% of the target for lines and complex geometries, and within 70% for contacts.

Fixing Decorate Job Problems

The Decorate job generates SRAFs intended to deliver the maximum process window. If the optimization is limited to one process condition, then the dose latitude is optimized. The contract of the Decorate job is that it generates at least two rows of generally correct assist

features. The SRAFs are robust and in dark areas. (Dark areas are outside the target for dark backgrounds, and inside the target for clear backgrounds.)

If the Decorate contract is met, all main features print. As much as 25% of the total SRAF area may also print; this is repaired in the Correct job.

When the absolute value of the background is greater than the absolute value of the transmission of the correction layer, or when the mask is dark and the background is clear, use a Decoratedarksraf job instead of a Decorate job. It performs the same function but has different default settings.

Prerequisites

- Calibre pxOPC setup file.
- Open job is working correctly.

Procedure

1. First, determine if there is a problem. A fairly typical mask with printing contours is shown in Figure 2-6. The red curves represent printing areas; they appear not only over the design features but also between (or even occasionally within) SRAFs. But because the printing SRAF area is less than 25% of the SRAF area, the contract is still met.

   To see the “job 2 decorate” output, add a setlayer pxopc command with “LASTJOB 2” to your file. Using the layer names from “Litho Setup File for pxOPC” on page 18, the command looks as follows:

   ```
   setlayer px.deco = pxopc target trgtsmth MAP target \
      OPTIONS SIMULATION_OPTIONS LASTJOB 2
   ```

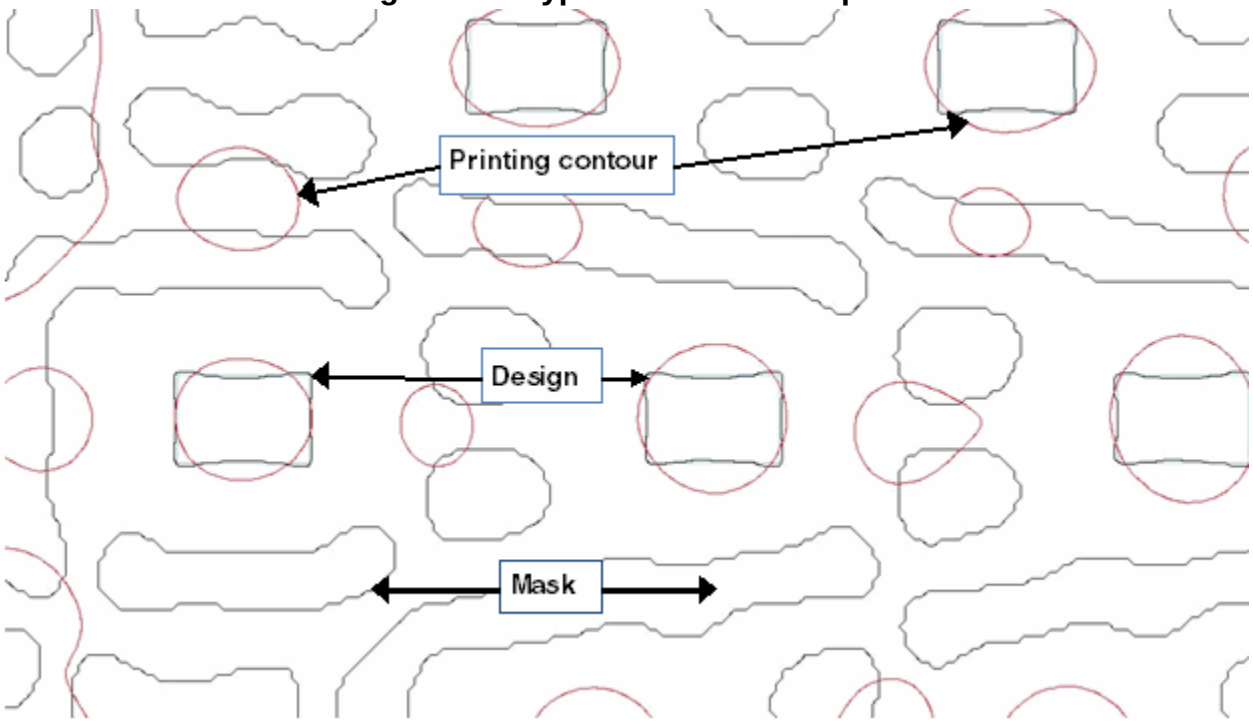
   Also add a rule check in the SVRF file to copy it to the results:

   ```
   dcrt_output {RET PXOPC target trgtsmth FILE "pxopc.in" MAP px.deco }
   DRC CHECK MAP dcrt_output 22
   ```

   Note

   Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

Figure 2-6. Typical Decorate Output



2. If you are running Calibre LPE and the generated SRAFs are not compatible with the ones from the original mask, tune scatter_offset. Incompatible SRAFs may appear as “wavy” SRAFs in the transition region.
3. If there are too many rows of SRAFs, try one of these methods:
   - Decrease scatter_belt by CD⁄4. The default is 0.256 um.
   - Decrease iterations in increments of two.
   - For negative SRAFs, set scatter_belt_negative. Start with 0.256 um and if needed decrease by CD/4. (Negative SRAFs are not affected by scatter_belt.)
4. If large areas are missing SRAFs, try one of these methods:
   - If there are too few SRAF rings, increase scatter_belt by CD/4.
   - If rings are present but not complete (weak or inconsistent SRAFs), try one of these methods:
     - Increase iterations in increments of two. The default is eight.
     - Check the exposure type. If the features are dark (that is, the absolute value of the background transmission is greater than the absolute value of the correction layer’s transmission), decrease the slope weight with the option metric_impact slope factor 0.05.

- To improve image slope, increase the slope weight by adding "metric_impact slope factor *value*" to the Decorate job. Start tuning using a value greater than the default 0.1.

5. Re-run Calibre pxOPC and check the output by running print image on the layer created by the Decorate job. If the images are large enough but the output still needs improvement, proceed to "Fixing Correct Job Problems".

Results

The output from the Decorate job shows consistent SRAFs and good feature printing. While there is extra printing, it does not exceed 25% of the total SRAF area.

Fixing Correct Job Problems

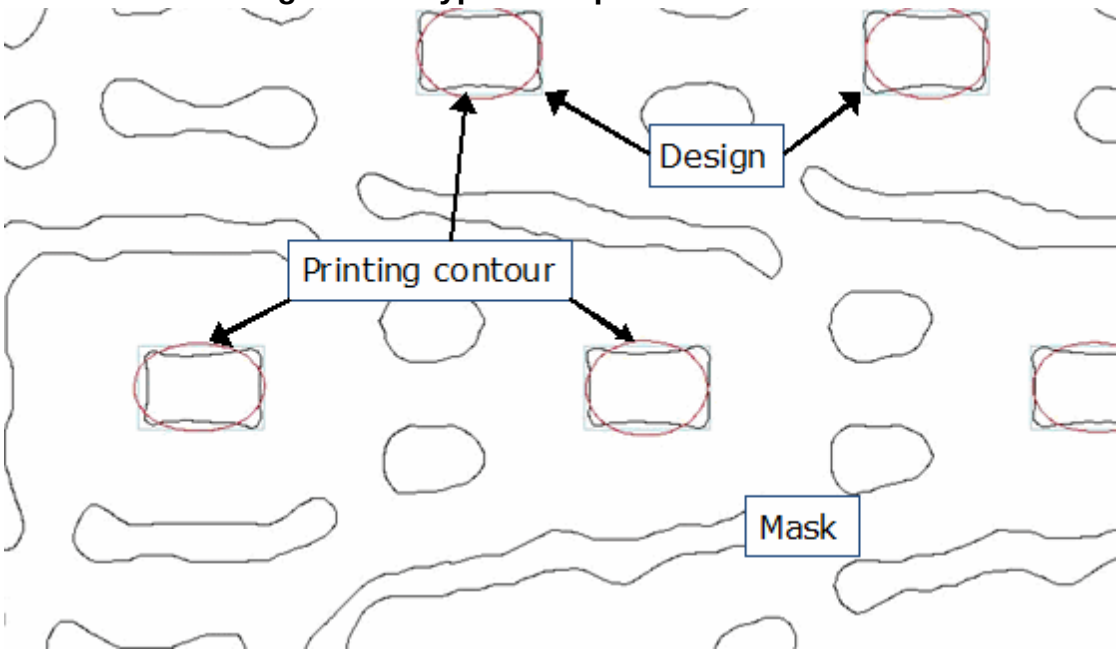
The Correct job corrects the printing and EPE distribution. It reduces PV bands and brings contours closer to design targets. After Correct runs, there may still be line end pullback, but there are no catastrophic failures.

The contract for this job is

- For all process window conditions within the verification box of Figure 2-4 of "Checking Initial Results," extra printing has been almost completely eliminated.
- Printing shapes are on target within 8% of CD.

Figure 2-7 shows typical output. Notice how compared to Figure 2-6 of "Fixing Decorate Job Problems" there are far fewer printing contours.

Figure 2-7. Typical Output of Correct Job



Prerequisites

- Decorate job is working correctly.

Procedure

1. To see the "job 3 correct" output, add a setlayer pxopc command with "LASTJOB 3" to your file. Using the layer names from "Litho Setup File for pxOPC" on page 18, the command looks as follows:

```
setlayer px.Correct = pxopc target trgtsmth MAP target \
   OPTIONS SIMULATION_OPTIONS LASTJOB 3
```

2. Add a rule check in the SVRF file to copy the output to the results:

```
correct_output {
   RET PXOPC target trgtsmth FILE "pxopc.in" MAP px.Correct }
DRC CHECK MAP correct_output 23
```

Note

Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

3. Determine if there is a problem. It is expected that there is some line-end pullback, and that contours may be as much as 8% of CD away from the target. Unless the difference is excessive, the contract is still met. Additionally, it is expected that there might be occasional SRAFs printing.
4. To improve main feature contours, try the following techniques:
   - If too many SRAFs are printing
     - Check whether the pw_condition settings are the same or slightly more aggressive than those used by the verification process window conditions.
     - For negative SRAFs, add an Unclose job after the Decorate job.
     - Increase iterations by two.
   - If EPE is too large, or some features are not opened in all process conditions
     - Verify that the selected process window conditions include nominal, inner, and outer.
     - If the process window conditions used an aerial resist model, try a CM1 model.
     - Increase iterations by two.
   - If contours ripple excessively
     - Add "straight_contour 1" to the Correct job.
     - Use a weight_layer.

- Tune the smoothed target to use less sharp corners.

- To improve image slope, adjust constraint.
  - If you do not have the constraint command in the job, add “constraint ils greater *val*”, where val is the target ILS, and “outer_iterations 5”.
  - If the constraint command is already in the recipe, try adding the “initial_weight *w*” option. The default *w* is 0.01.
  - If ILS is still insufficient, change constraint to “constraint ils maximize initial_weight *w*”. The “maximize” setting should only be used with set_on_target and suppress_extra_printing.

5. If the adjustments were not sufficient, adjust the process window condition. The process window conditions set in pw_condition should be somewhat more aggressive than the conditions set in the verification file.
6. Re-run Calibre pxOPC and check the output by running print image on the layer created by the Correct job. If the image meets the contract but the final output still needs improvement, proceed to “Fixing Refine Job Problems”.

Results

The output from the Correct job shows a contour-like mask and the printing is almost entirely main features, which match acceptably with the design layer.

Fixing Refine Job Problems

The Refine job improves the output of the Correct job by doing pixel-based optimization. Because some areas are more important to resolve than others, critical EPE sites can be emphasized with weighted shapes.

The contract of the Refine job is that printing shapes are on target within fractions of a nanometer in those locations that are identified with weight_layer markers. Additionally, no extra printing is occurring.

Calibre LPE requires substituting “finetune” for “refine”. The information in this procedure also applies to Finetune jobs.

Prerequisites

- Output from the Correct job is acceptable.
- Critical sites identified.

Procedure

1. To see the “job 4 refine” output, add a setlayer pxopc command with “LASTJOB 4” to your file and a rule check in the SVRF file to copy it to the results.

Note

Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. To improve critical sites, add markers in a weight layer:

   a. Use SVRF commands to create a layer or layers with marker shapes over the critical sites on the original target layer (not the smoothed target). The shapes should meet the following conditions:

      - Marker shapes should be long and narrow, and must overlap both the contour and the target edge. A good size for 20 nm and below is a width of 3 nm and a length of 30, but this may be too small if your contours ripple excessively.
      - The marker's long edge should extend equally on either side of the target edge.
      - Markers that will have different weights should be on different layers.

   b. Add the marker layers to the run.

      At the SVRF level, assuming the name is wt1:

      ```
      RET PXOPC target trgtsmth wt1 FILE "pxopc.in" MAP opcout
      ```

      In the main portions of the setup file:

      ```
      LITHO FILE pxopc.in [ ...
         layer trgtsmth hidden clear
         layer wt1 hidden clear
         ...
         setlayer opcout = pxopc target trgtsmth wt1 MAP target \
            OPTIONS SIMULATION_OPTIONS
      ```

      In the pxOPC options block of the setup file:

      ```
      ...
      layer trgtsmth TARGET 0.9785 0.0085
      layer wt1 HIDDEN 0.9785 0.0085
      ...
      weight_layer wt1 set 100
      ```

      Start with a weight of 100. The default weight (that is, the weighting of unmarked areas) is 1.

3. To improve overall fit, use only the nominal process window condition and increase iterations by two. The default for Refine is all defined pw_conditions and six iterations.

4. If contours ripple excessively, use one of these three methods:

   - Add a weight layer with markers for the problematic edges.
   - Adjust the setlayer curve command to be more loose near corners.

- Add “straight_contour 1” to the Refine job. This option can degrade EPE, so increase the value slowly when tuning.

5. If some EPEs are still very bad, increase the length of the markers. If markers are too short, they might not cover the printing contour. If they do not overlap the contour, their weighting has no effect.
6. If the region contains dissimilar patterns and the design constraints are tight, reduce rate and increase iterations. This also increases the run time; adjust gradually.
7. To improve image slope, adjust constraint.
   - If you do not have the constraint command in the job, add “constraint ils greater *val*”, where val is the target ILS, and “outer_iterations 5”.
   - If the constraint command is already in the recipe, try adding the “initial_weight *w*” option. The default *w* is 0.01.
   - If ILS is still insufficient, change constraint to “constraint ils maximize initial_weight *w*”. The “maximize” setting should only be used with set_on_target and suppress_extra_printing.
8. Re-run Calibre pxOPC and check the output by running print image on the layer created by the Refine job. If the image meets the contract but the final output still needs improvement, proceed to “Fixing LMRC Job Problems”.

Results

The output looks very good but the mask is still non-Manhattan. The contours are converted to Manhattan shapes in the next job, LMRC.

Fixing LMRC Job Problems

The Litho-Mask Rule Constraint (LMRC) job converts the mask back from contours to Manhattan shapes of predefined edge lengths. It then performs litho simulations to detect assist features that print and adjusts the mask to fix this.

The contract for LMRC is that you have a manufacturable mask:

- The output mask is strictly Manhattan.
- No extra printing occurs.
- MRC rules are obeyed.

Obeying all MRC constraints usually results in a slight degradation of OPC quality.

Prerequisites

- Output from the Refine job is acceptable.

Procedure

1. To see the “job 5 lmrc” output, add a setlayer pxopc command with “LASTJOB 5” to your file and a rule check in the SVRF file to copy it to the results.

   Note

   Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. Set mrc_min_edge, mrc_min_external, and mrc_min_internal to the minimums supported by your process. Using minimum values allows the Manhattanized contours to more closely match the curved shapes.
3. To improve extra printing, specify a pw_condition specific to LMRC that is more aggressive than the other process window conditions, and use only it within the LMRC job. For example, for dark backgrounds increase the dose by 0.1 to 0.5%. (For clear backgrounds, decrease the dose similarly.) Use only this condition.
4. If negative SRAFs are printing, set “lmrc_inside_assist_cuts_enabled yes” for the LMRC job.
5. If you still have printing SRAFs, increase mrc_iterations in increments of 10.
6. If running with Calibre LPE and there are MRC violations near the repair region boundary, make sure in both setup files that the mrc_min_edge option is not larger than the original OPC fragment length and that the original mask does not have MRC violations.

Results

A manufacturable mask that meets your goal. Occasionally the conversion of contours to polygons degrades results too much; in this case, run a Detail job after the LMRC job.

Fixing Detail Job Problems

The Detail job runs fine-grained optimization for the best EPE RMS, meaning that EPE distribution is centered and shrunken while maintaining an MRC-clean mask. The mask is represented as Manhattan polygons.

The contract of the Detail job is that you have a manufacturable mask:

- The output mask is strictly Manhattan.
- No extra printing or catastrophic failures occur.
- MRC rules are obeyed.

Prerequisites

- Output from the LMRC job is acceptable.

Procedure

1. To see the “job 6 detail” output, add a setlayer pxopc command with “LASTJOB 6” to your file and a rule check in the SVRF file to copy it to the results.

   Note

   Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. Set mrc_min_edge, mrc_min_external, and mrc_min_internal to the values used in the LMRC job. This prevents SRAFs from printing that were not printing after LMRC.
3. To eliminate extra printing, add “assist_outside_dose_multiplier 1.02”. If you are using pw_bridging and pw_pinching, decrease their weight.
4. If contours ripple excessively, use one of these three methods:
   - Add a weight layer with markers for the problematic edges.
   - Adjust the setlayer curve command to be more loose near corners.
   - Add “straight_contour 1” to the Detail job. This option can degrade EPE, so increase the value slowly when tuning.
5. To improve litho quality at critical sites, add markers in a weight layer. See “Fixing Refine Job Problems” on page 44 for detailed instructions.
6. If there is soft pinching or bridging in some of the PV bands, try these techniques:
   - Increment the weight on pw_bridging or pw_pinching by 1.
   - Check the distance between the process window contour and the verification contour and adjust the process window condition or the constraint accordingly.
   - If you are running Calibre pxOPC through Calibre LPE, additionally verify that the initial contours in Calibre LPE do not violate the process window constraints within the visible region.
7. If enclosure is violated, add a pw_enclose objective to the job. Do not use pw_enclose if the incoming contours show bad pinching or bridging, or the enclosure has already been violated.
8. To improve image slope, adjust constraint.
   - If you do not have the constraint command in the job, add “constraint ils greater *val*”, where val is the target ILS, and “outer_iterations 5”.
   - If the constraint command is already in the recipe, try adding the “initial_weight *w*” option. The default *w* is 0.01.

- If ILS is still insufficient, change constraint to “constraint ils maximize initial_weight *w*”. The “maximize” setting should only be used with set_on_target and suppress_extra_printing.

9. If running with Calibre LPE and there are MRC violations near the repair region boundary, make sure in both setup files that the mrc_min_edge option is not larger than the original OPC fragment length and that the original mask does not have MRC violations.

Results

A manufacturable mask that meets your goal.

Fixing CDOF Job Problems

The CDOF job optimizes the mask to improve the common process window, a lithographic measure of manufacturing robustness. A good common process window should allow some variation in dose and a wide range of defocus while still keeping EPE within tolerance.

The CDOF job is *optional*. It requires a litho model with multiple defocus conditions. If you can optimize the optical source for this mask, consider using Calibre® pxSMO™ source-mask optimization. See the *Calibre SMO RET Selection User’s and Reference Manual*.

The contract of the CDOF job is that you have a manufacturable mask that fulfills the Detail job contract and offers an optimal depth of focus.

Restrictions and Limitations

- The example litho setup file provided in Figure 2-2 on page 18 of “Setting Up the SVRF and Litho Setup Files” does not use litho models, so cannot be used for experimenting with a CDOF job.
- Calibre pxOPC does not provide common process window information in its transcript. To see the outcome of these fixes, simulate and plot the results in another tool such as Calibre WORKbench. See “Creating A Process Window in PW Plot” in the *Calibre WORKbench User’s and Reference Manual*.

Prerequisites

- Output from the Detail job is acceptable.
- Detail job included at least three process window conditions.

Procedure

1. If the CDOF job is not the final mask, add a setlayer pxopc command with “LASTJOB 7” to your file to see the “job 7 cdof” output, and a rule check in the SVRF file to copy it to the results.

Note

Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. Check the process window plot.
   - If the expected focus range was less than 50 nm or greater than 200 nm and the process window plot shows an inscribed shape is not near those values, set the initial pw_condition_group … focus_range closer to the expected result. The default is 100 nm.
   - Adjust the tolerance band identified by pw_tolerance_bound. Verify that along straight edges it is always at least as wide as the CD tolerance, and significantly wider at line ends, space ends, and corners.
   - If the inscribed shape is bounded only by its height (dose):
     - Decrease pw_focus_range_factor in increments of 0.2. The default is 1.
     - Increase pw_condition_group … dose_latitude in increments of 0.005. The default is 0.06. Also verify that shape_type is vertical_segment.
   - If the inscribed shape is bounded by its width:
     - Increase pw_focus_range_factor in increments of 2. The default is 1.
     - Increase iterations in increments of 20. The default is 100.
3. Generate a print image to see how well the mask achieves the design intent.
   - If the EPE is too large (that is, the contour is too far from the edge), add or adjust "set_on_target nominal." Remember that improving EPE is likely to decrease the common depth of focus.
   - If the simulated contours show hard bridging, decrease pw_focus_range_factor in increments of 0.2. The default is 1. Alternatively, shrink the tolerance band contour within the target (to maintain the focus range). It is important that near gauges the tolerance band be exactly CD tolerance.
   - If there is extra printing, verify that all process conditions associated with it are listed in the job's pw_select and suppress_extra_printing keywords. If so, increase the weight of suppress_extra_printing by 1. This is likely to reduce the focus range.
4. Re-run Calibre pxOPC.

Results

The output from the CDOF job has no extra printing, hard bridging, or hard pinching. Additionally, all MRC rules are obeyed and the mask allows the best possible common depth of focus.

Fixing EL Job Problems

The EL job optimizes the mask to improve the exposure latitude (EL). Exposure latitude is also referred to as dose latitude, and affects image-log slope (ILS).

The EL job is *optional*. It requires a litho model with multiple defocus conditions and two process window groups. If you can optimize the optical source for this mask, consider using Calibre® pxSMO™ source-mask optimization. See the *Calibre SMO RET Selection User's and Reference Manual*.

The contract of the EL job is that you have a manufacturable mask that fulfills the CDOF job contract and offers an optimal dose latitude without impairing DOF.

Restrictions and Limitations

- The example litho setup file provided in Figure 2-2 on page 18 of "Setting Up the SVRF and Litho Setup Files" does not use litho models, so cannot be used for experimenting with an EL job.
- Calibre pxOPC does not provide common process window information in its transcript. To see the outcome of these fixes, simulate and plot the results in another tool such as Calibre WORKbench. See "Creating A Process Window in PW Plot" in the *Calibre WORKbench User's and Reference Manual*.

Prerequisites

- Output from the previous job is acceptable.

Procedure

1. If the EL job is not the final mask, add a setlayer pxopc command with "LASTJOB 8" to your file to see the "job 8 el" output, and a rule check in the SVRF file to copy it to the results.

   Note

   Do not forget to remove the lines for extra output when you are ready to use the SVRF and pxOPC setup files for production runs.

2. Check the process window plot. If the dose latitude is not large enough, try the following adjustments. Keep in mind, however, that as Calibre pxOPC only optimizes the mask, improvements to dose latitude often reduce depth of focus.
   - Increase pw_dose_latitude_factor. The default value is 10; increase by 2 at a time. (EPE may become worse.)
   - Increase the tolerance band width (pw_tolerance_bound). Straight edges should be at least as wide as the CD tolerance, and line ends, space ends, and corners should be around twice as wide.
   - Increase iterations in increments of 20. The recommended value is 100.

3. Generate a print image to see how well the mask achieves the design intent.
   - If the EPE is too large (that is, the contour is too far from the edge), add or adjust "set_on_target nominal." Remember that improving EPE is likely to decrease the dose latitude and ILS.
   - If there is extra printing, verify that all process conditions associated with it are listed in the job's pw_select and suppress_extra_printing keywords. If so, increase the weight of suppress_extra_printing by 1. This is likely to reduce the dose latitude.
4. Re-run Calibre pxOPC.

Results

The output from the EL job has no extra printing, hard bridging, or hard pinching. Additionally, all MRC rules are obeyed.

Reconciling Results with Calibre OPCverify

Occasionally Calibre OPCverify may show pinching, bridging, or SRAF printing that Calibre pxOPC simulations do not. Generally this is because the two setup files use different parameters.

Prerequisites

- Results from Calibre OPCverify.
- Intermediate results from Calibre pxOPC, produced with the LASTJOB keyword.

Procedure

1. Verify that the process window conditions defined in the two setup files match, or that the Calibre pxOPC ones are more stringent as described in Figure 2-4 of "Fixing Initialization Problems."
2. If Calibre OPCverify has a process window condition is different in dose, add a dose multiplier to the Correct, Refine, Finetune, and Detail jobs.
   a. Divide the Calibre OPCverify dose by the dose used for checking bridging conditions in Calibre pxOPC. This determines the minimum size for the multiplier.
   b. Add the dose multiplier to any of the specified jobs in the recipe with the following keyword:

      ```
      assist_outside_dose_multiplier multiplier
      ```

      where *multiplier* is slightly larger than the value calculated from the doses.
3. If Calibre OPCverify has a process window condition that uses a different resist model or defocus value, add a similar condition to the Calibre pxOPC recipe file. Specify the new condition in the Correct, Refine, Finetune, and Detail jobs.

4. If Calibre OPCverify has a process window condition that uses mask sizing, change the Calibre pxOPC recipe as follows:

   a. Derive a layer by sizing up the target by a similar value:

   ```
   setlayer newLayer = size target by value
   ```

   b. Add a process window condition similar to the Calibre OPCverify condition, but using "outside *newLayer*" with pw_condition. This makes the new condition active in only the SRAF region.

   c. Add this new condition to the Correct, Refine, Finetune, Detail, and LMRC jobs.

Results

The Calibre pxOPC and Calibre OPCverify results match.

Improving Calibre pxOPC Run Time

When working with complex designs, you may desire a faster result. This procedure offers some tips for improving execution time.

Prerequisites

- Correctly working SVRF and pxOPC setup files.
- An understanding of the purpose of each job type.

Procedure

1. The most basic way to speed up your run is to work on a simpler layout database. If possible, try the following:
   - Flatten the layout using Expand Cell "*".
   - Reduce the area, ideally to a single frame, about 3 microns per side.
2. Simplify the SVRF file.
   - Comment out rule checks you are not using, such as those for copying intermediate job layers.
   - Do not include connectivity information.
   - If you are not debugging, be sure the setup file does not print intermediate job layers (LASTJOB).
3. If the main features already print on target, omit the Open job.
4. If print image quality is satisfactory, omit the Refine job.
5. For all jobs except Refine and Detail, use process conditions that use an aerial threshold instead of a resist model.

6. Run fewer process conditions in general. Increased speed is traded off for quality of results, however.

   The Decorate job requires two process conditions in order to optimize the process window. Except as noted below, the other jobs can run with one process condition.

   - When the recipe includes a CDOF job, the Detail job should use at least three process conditions.
   - For recipes that use an additional Detail job for pw_bridging and pw_pinching, use just the pw_bridging and pw_pinching Detail job. Keep their width constraints under 60 nm.

7. Balance the iterations. If you needed to increase iterations in one job, see if you still get acceptable results by reducing the iterations in the one after. (Recall that the LMRC job uses mrc_iterations rather than iterations.)
8. Adjust the rate. The recommended rate is slightly above the average for all reported rates for a particular job type. The rates for different jobs do not relate to each other.

   a. Add the line "debug_level 1" to your setup file.

   b. Perform several runs, capturing the rates in a log.

   c. For each job type calculate the average rate multiplied by 1.1. This is the recommended value of rate.

   d. Remove the line "debug_level 1".

   e. For each job, add a rate setting.

Results

The SVRF and pxOPC setup files are streamlined for faster runs.

Related Topics

Analyzing Transcripts

Analyzing Transcripts

The pxOPC transcript analyzer is available in the Calibre® DESIGNrev™, Calibre® MDPview™, Calibre LITHOview, and Calibre WORKbench viewers. It calls Gnuplot to display graphs of the rate and objective.

Restrictions and Limitations

- The transcript analyzer is only supported for transcripts created by the same version of Calibre.
- The computer session must have Gnuplot in the $PATH.

Prerequisites

- The transcript must have “debug_level 1” output.
- An active Calibre viewer (see the viewer list in the introduction).

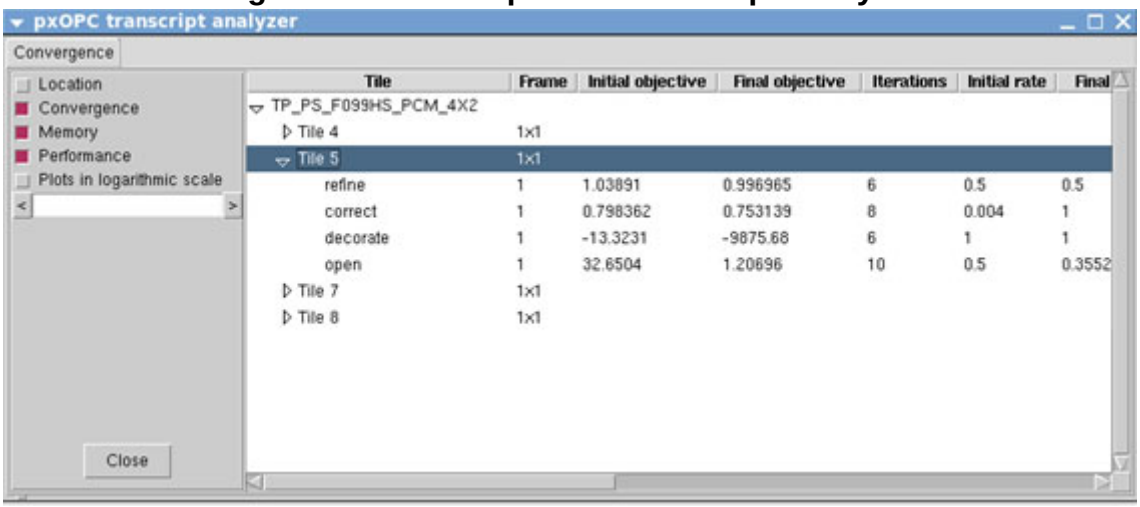
Procedure

1. In the Calibre viewer, select **Macros > PXOPC transcript analysis**.
2. Use the file browser to select the log file from the Calibre pxOPC run.

   The file browser automatically filters on *.transcript*. If you do not see your log file, change the “Files of type” setting.

   If the pxOPC Transcript Analyzer window does not show any data in the **Convergence** tab, the run was either from an incompatible version of Calibre or was done without debug_level 1 being set.
3. Expand one of the tile listings to show the job entries. (See Figure 2-8.) Each entry lists the job name, final objective, and final rate.

Figure 2-8. Calibre pxOPC Transcript Analyzer



The data shown depends on your selections to the left. The Convergence option includes the initial and final rate, which can be useful for tuning recipes. If the recipe includes any check keywords, the Convergence option also displays whether the checks found any violations.

4. To plot the data, select a job entry and right-click.

   For steeply changing plots, select the “Plots in logarithmic scale” option on the left and plot again. You can have multiple plot views open at the same time.

   You can zoom into sections of the plot by dragging with the left mouse button over a section. Go back to the previous view by typing “u” in the plot window. “p” and “n” go forward and back through the view history.

Results

The plots of the rate and objective help you find convergence problems and tune your run.

- If either rate or objective do not flatten at the end of the graph, try adding more iterations to the job.
- If the objective rises, it means the settings are preventing the job from converging on a solution. Try decreasing the rate or loosening some constraints.

Related Topics

debug_level

enable_default_checks

Chapter 3
Calibre pxOPC Command Reference

This chapter contains descriptions for the commands used by Calibre pxOPC. The commands are grouped by their location in a rule file.

SVRF Commands

There are two SVRF commands used by Calibre pxOPC.

Table 3-1. Calibre pxOPC SVRF Commands

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

The rule file may use many other SVRF commands. The rule file syntax and other SVRF commands are described in the *Standard Verification Rule Format (SVRF) Manual*.

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 in-line setup file to be referenced by 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***.

  Each statement in the setup file must be on a line of its own. Statements can span multiple lines by ending the line with a backslash (\). The backslash character must not be followed by any other characters, even whitespace.

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

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

   layer name type trans ...

pxopc_options opt_name {
   // general section
   layer name type trans
   ...
   // job 1 section
   // job 2 section
...
}

setlayer out_layer = pxopc name MAP name OPTIONS opt_name

*/ ]
```

Different Calibre RET tools allow different commands in the ***inline_file***, so Litho File statements cannot generally be shared between tools. The setup file for Calibre pxOPC runs may include Calibre OPCverify commands without causing an error but some commands may be ignored by the pxOPC run.

Description

The Litho File statement specifies the FILE parameter of the RET PXOPC operation to be inline separate from the operation itself, yet still in the SVRF rule file. The FILE parameter of an operation making a call to a Litho File may indicate a Litho Filename.

Examples

```
LITHO FILE pxopc.in [ /*
   setup commands
*/ ]

RET PXOPC layer1 FILE pxopc.in MAP layer1
```

When the RET PXOPC operation is compiled, the rule file is searched for the Litho File specification statement with a name of *pxopc.in*. This search is case-insensitive. If “LITHO FILE pxopc.in” is found, then it is used. If it is not found, then *pxopc.in* is assumed to be an external file.

No Litho File specification statements may share the same ***name***.

RET PXOPC

Type: SVRF Commands

Invokes Calibre pxOPC.

Usage

RET [EUV] **PXOPC** ***layer*** [*layer*…] **FILE** {***filename*** | ***name*** | '**[**'
inline_file
'**]**'} **MAP** ***name***

Parameters

- EUV

  An optional keyword that is required when the optical model has a wavelength of 13.5 nm or shorter. To use EUV libraries you must have an EUV license.

- ***layer***

  A required original layer, derived polygon layer, or edge layer upon which to perform Calibre pxOPC operations. You can specify one or more ***layer***s in one statement. The layer statements in the litho setup file determine the function of the ***layer***s.

- **FILE** {***filename*** | ***name*** | '**[**'

  inline_file

  '**]**'}

  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.

  An ***inline_file*** between brackets ([ ]). Characters within the brackets on the same line as them are ignored. in-line files contain setup instructions for OPC and span multiple lines. The instructions may not use environment variables.

- **MAP** ***name***

  A required keyword, followed by the name of a layer from the setup file. This layer is then mapped to the layer in the SVRF rule file for output.

Description

Specifies to run Calibre pxOPC during Calibre nmDRC or Calibre nmDRC-H. Calibre pxOPC performs full mask optimization, including OPC and SRAF insertion and optimization, upon the layers you specify. The command takes in a target layer and outputs a new layer with OPC-corrected shapes and optimized SRAFs.

This command can be specified any number of times in your rule file.

Examples

Example 1 - Layer Passing

The following example shows how layers are mapped between the RET PXOPC 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.)

```
L1 = RET PXOPC m1a m1_smooth FILE "setup.in" MAP L1
L1 {COPY L1} DRC CHECK MAP L1 20
...
LITHO FILE setup.in [
   ...
   layer orig hidden clear //"global" part of setup file; pxopc layers
   layer smoothed hidden clear    //hidden to just pass through
   ...
   pxopc_options px.options {  //pxopc-specific commands
      layer orig CORRECTION clear
      layer smoothed TARGET clear
      ...
   }
   ...
   setlayer mask_out = pxopc orig smoothed MAP orig OPTIONS px.options
   ...
]
```

The RET PXOPC command passes the m1a and m1_smooth layers to the setup file, where they are mapped to the orig and smoothed layers respectively. These names are then used throughout the setup file.

Example 2 - Alternate Forms

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

```
L1 {RET PXOPC m1a m1_smooth FILE "setup.in" MAP L1} DRC CHECK MAP L1 20
...
```

The line shown here would replace the two lines from before:

```
L1 = RET PXOPC m1a m1_smooth FILE "setup.in" MAP L1
L1 {COPY L1} DRC CHECK MAP L1 20
```

Litho Setup File Commands

The ***inline_file*** argument to RET PXOPC specifies an in-line Litho File setup file containing a command initialization block. Commands in this block are lowercase only. The following commands may appear in the setup file, specified one per line.

Some commands have a different syntax if using a litho model in place of optical, ddm, and resist models. A setup file must either use the litho model syntax for all commands, or not at all. Standard syntax and litho model syntax cannot be mixed.

Table 3-2. RET PXOPC Litho Setup File Commands

Command	Description
ddm_model_load	Loads a domain decomposition method (DDM) model.
direct_input	Enables direct data entry (DDE), which can improve run time in some cases. Requires litho flat processing mode.
direct_output	Enables direct data output (DDO), which can improve run time in some cases. Requires litho flat processing mode.
euv_field_center	Centers the field for EUV black border models that include the field_size keyword.
euv_slit_x_center	Sets a placement X-coordinate for EUV through-slit and shadow bias models.
flare_longrange	Associates a precomputed long range flare convolution with a flare model contained inside a litho model.
layer (for main setup file)	Defines layer characteristics such as name and type.
litho_model	Permits in-line definition of a litho model.
modelpath	Specifies the directories to search for litho, optical and resist models.
optical_model_load	Loads an optical model.
processing_mode	Flattens geometries when the database was constructed hierarchically.
progress_meter	Estimates the progress of the LITHO operation.
pxopc_options	Command block within a setup file that defines the Calibre pxOPC optimization settings.
resist_model_load	Specifies a resist model.
setlayer copy	Copies the specified layer to a new layer.
setlayer curve	Creates “smoothed” shapes based on the input layer.
setlayer curve_target	Creates a target that mimics the final image.
setlayer pxopc	Creates and derives layers.

Table 3-2. RET PXOPC Litho Setup File Commands (cont.)

Command	Description
setlayer tilegen	Creates an annotated layer to display tile and cell boundaries with properties.
tilemicrons	Specifies the size of a tile in microns. Recommended.

ddm_model_load

Type: Litho Setup File Commands

Loads a domain decomposition method (DDM) model.

Usage

ddm_model_load ***ddm_model_name*** {***filename*** |'**{**' ***inline*** '**}**' }

Arguments

- ***ddm_model_name***

  A required argument specifying a name that other commands will use to refer to this model.

- ***filename***

  An argument specifying a filename to load containing the DDM model. If you do not specify a filename, an in-line model must be supplied instead.

- '**{**' ***inline*** '**}**'

  An argument defining a DDM model. The entire model specification must be enclosed in braces ({ }).

Description

An optional command that loads the DDM model contained in the file ***filename*** or defined by specifying the complete model text within braces ({ }). The DDM model is assigned the user-defined ***ddm_model_name*** for later command references.

A DDM model is a type of mask model that can simulate 3D transmission by including edge effects. Its use is generally limited to systems with low k1 and high NA. DDM models are distinct from optical and resist models. You can generate DDM models in Calibre WORKbench as described in the section "3D Mask Modeling With DDM and HHA" in the *Calibre WORKbench User's and Reference Manual*.

If you are using a litho model, this command is ignored. The DDM information is included in the litho model.

Examples

The following example shows the lines related to models in a Calibre pxOPC setup file.

```
modelpath /central/models
optical_model_load nominal  opt1_nom_ddm
optical_model_load minus_40 opt1_nom_ddm-40
optical_model_load plus_40  opt1_nom_ddm+40
resist_model_load  resist   cm1.mod

ddm_model_load     ddm.mod  approx.ddm
...

pxopc_options OPT {
   ...
   ddm ddm.mod
   pw_condition NOMINAL optical nominal dose 1.0
   pw_condition PINCH optical minus_40 dose 0.96
   ...
}
```

Related Topics

ddm

direct_input

Type: Litho Setup File Commands

Enables direct data entry (DDE), which can improve run time in some cases. Requires litho flat processing mode.

Usage

direct_input '**{**'
{ **vboasis_path** ***filename***
{[input *input_number*] **layer** ***number*** [*datatype*]}…
[vboasis_precision_multiplier {*n*/*d* | AUTO}]
} …

'**}**'

Arguments

- **vboasis_path** ***filename***

  A required argument supplying the absolute path to an OASIS filename to be used as the design source. A run-time error is returned if the input file is invalid. The filename must not be in single or double quotes.

  Multiple **vboasis_path** sets can be specified in a direct_input statement.

- [input *input_number*] **layer** ***number*** [*datatype*]

  A required keyword set that specifies one or more layers. At least one layer declaration is required.

  input *input_number* — An optional keyword set that specifies which input layer from the setup file to map. The index starts at 1 for the first layer listed in the setup file. The specified layer is mapped to the VB:OASIS layer ***number***, allowing you to use the input layers out of order. The value you specify for *input_number* cannot exceed the number of setup layers. If this keyword set is not specified, layers are assumed to be in the same order as the setup file.

  Either all layers must use input *input_number*, or none of them.

  layer ***number*** — A required keyword set defining a layer that is present in ***filename***.

  datatype — An optional argument that limits the datatypes that map onto Calibre pxOPC layers. If *datatype* is absent, all datatypes for that layer are used.

- vboasis_precision_multiplier {*n*/*d* | AUTO}

  An optional argument that modifies the PRECISION of ***filename***. It may be specified once per vboasis_path. Either AUTO or a ratio must be specified.

  n/*d* — Multiplies the PRECISION by *n*/*d*, and then scales the data from the file by the same amount to retain the same absolute scale as the original PRECISION setting.

  AUTO — Attempt to infer the correct ratio. AUTO exits from the parser with an error if the ratio is not a rational number or would cause arithmetic overflow.

Description

This optional command enables direct data entry (DDE). It works in conjunction with litho flat processing mode and automatically activates it when a direct_input block is found, even if processing_mode is set to hierarchical.

This statement can appear only once per LITHO FILE. Use multiple **vboasis_path** sets in a single statement to access multiple files.

This command causes the layers in LAYOUT PATH to be left unused. They are replaced with layers read directly from the OASIS database specified with the **vboasis_path** argument. If direct_output is not also specified, only the OASIS data within the coordinates of the HDB extent is directly read. Therefore, it is not enough to pass in dummy layers through the RET PXOPC statement; at least one DRC RDB layer must contain a square large enough to cover the given OASIS file in order to ensure that everything in it gets processed.

There is a one-to-one mapping between the layers specified in this command and the layers specified using the layer (for main setup file) command. This mapping is based on the layer order.

Examples

The following example of an SVRF file declares an empty OASIS file for the traditional (HDB) input, and then directly reads the *test.oas* and *test2.oas* OASIS files.

```
LAYOUT PATH "dummy.oas"

LAYER layout_window 1
LAYER dummy1 998
LAYER dummy2 999

# Write layout_window to DRC RDB to ensure the OASIS data is read
layout_window { COPY layout_window } DRC CHECK MAP layout_window 1000

M1 = RET PXOPC layout_window dummy1 dummy2 FILE setup MAP M1_copy
M1_OPC = RET PXOPC layout_window dummy1 dummy2 FILE setup MAP M1_OPC

LITHO FILE setup [ /*

   direct_input {
      vboasis_path test.oas
         input 1 layer 43
         input 3 layer 46
      vboasis_path "test2.oas"
         input 2 layer 33 100
   }

   modelpath models
   optical_model_load op_mod
   resist_model_load res_model

   processing_mode flat     # Optional when using direct_input.

   layer M1 visible clear   # Without direct_input, this would have been
                            # "layout_window." With direct_input this is
                            # layer 43 in the file "test.oas".
   layer sraf visible clear # Without direct_input, this would have been
                            # "dummy1." With direct_input this is layer
                            # 33, datatype 100, in the file "test2.oas".
   layer CONT hidden clear  # Without direct_input, this would have been
                            # "dummy2." With direct_input this is layer
                            # 46 in the file "test.oas".

   pxopc_options px.opt {
   ...
   }

setlayer M1_copy = PXOPC ...
setlayer M1_OPC = PXOPC ...
*/]
```

Related Topics

Direct Data Access [Calibre Post-Tapeout Flow User's Manual]

direct_output

processing_mode

direct_output

Type: Litho Setup File Commands

Enables direct data output (DDO), which can improve run time in some cases. Requires litho flat processing mode.

Usage

direct_output '**{**'
{**vboasis_path** ***filename*** [append]
{**output** ***name*** **layer** ***number*** [*datatype*]}…
[vboasis_output_primary *source*] } …
'**}**'

Arguments

- **vboasis_path** ***filename*** [append]

  A required argument supplying the absolute path to an OASIS filename. All remote hosts must be able to access the file.

  Multiple **vboasis_path** sets can be specified in a direct_output statement.

  Use the optional append keyword if ***filename*** already exists. This causes the output to be added to ***filename*** rather than overwriting it.

- **output** ***name*** **layer** ***number*** [*datatype*]

  A required keyword set that specifies one or more layers. At least one layer declaration is required.

  output ***name*** — A required argument specifying a layer name created with a setlayer command elsewhere in the Litho setup file.

  layer ***number*** — A required argument specifying the layer number to use when writing to ***filename***.

  datatype — An optional argument that can be used to specify a datatype sublayer.

- vboasis_output_primary *source*

  An optional argument that can be specified once per direct_output command. It specifies the source of the top cell name in ***filename***. Use one of the following:

  HDB — The topcell name from the HDB.

  DIRECTIN — The top cell name from the first **vboasis_path** input.

  FILE — Same as DIRECTIN.

  explicit *name* — Explicitly names the top cell.

Description

This optional command enables direct data output (DDO). To prevent the compiler optimizing away "empty" layers, layers written to ***filename*** must also be passed to the SVRF level with the RET PXOPC MAP keyword and written to a results database.

The direct_output command works in conjunction with litho flat processing mode and automatically activates it. You must set the CALIBRE_MTFLEX_LOCAL_HOST_DIR environment variable and the LOCAL HOST DIR configuration statement, as described in the *Calibre Administrator's Guide*.

This statement can appear only once per LITHO FILE. Use multiple **vboasis_path** arguments in a single statement to output to multiple files.

Examples

The example SVRF file in Example 3-1 combines direct_input and direct_output. It shows the use of fake input and output files (*dummy_in.oas* and *dummy_out.oas*) and layer mapping.

Example 3-1. Direct Data Instead of HDB

```
LAYOUT PATH "dummy_in.oas"
LAYER target 5 100
LAYER contour 1006

DRC RESULTS DATABASE "dummy_out.oas" OASIS
POLYGON 0 0 1000 1000 target
LAYOUT ULTRA FLEX YES

X = RET PXOPC target contour MAP final FILE setup.in
ilt_out { COPY X }
DRC CHECK MAP ilt_out 101 2

LITHO FILE setup.in [
   direct_input {
    vboasis_path "/net/machine1/export/input.oas"
    input 1 layer 5 100
    input 2 layer 1006
   }
   layer target hidden clear
   layer contour hidden clear

   pxopc_options opt [
      ...
      layer target correction
      ...
   ]

   setlayer final = pxopc target contour MAP target OPTIONS opt

   direct_output {
    vboasis_path "/net/machine1/export/output.oas"
    output final layer 101
  }
]
...
```

Related Topics

Direct Data Access [Calibre Post-Tapeout Flow User's Manual]

direct_input

processing_mode

euv_field_center

Type: Litho Setup File Commands

Centers the field for EUV black border models that include the field_size keyword.

Usage

euv_field_center ***x_um y_um***

Arguments

- ***x_um y_um***

  A required pair of arguments specifying the X, Y coordinates in microns of the center of the EUV field. Use the coordinate system for the input design file for the current run.

Description

This optional command specifies the center of an EUV field to center the addition of black border models. It is only used for black border models that contain the field_size keyword. Other EUV field offsets such as those for the flare maps are specified separately.

The EUV field is defined as the region surrounded by the REMA blades. The size of the EUV field may not correspond to the EUV slit position; see “Examples”.

If the black border model contains “field_size”, this command is required.

Examples

One case when the euv_field_center and euv_slit_x_center may appear unrelated is when there are multiple chiplets being imaged. For example, if the setup uses the masking blades to image chiplet 1 on the right side of the reticle, the field center should also be on the right side and the black border model should include field_size. The slit center would still be in the center of the reticle, to the left of the field center.

If the black border model has “field_size 100 100” and the setup file has “euv_field_center 500 1000”, the EUV field for this chiplet would extend along X from 400 to 600 along Y from 900 to 1100. The black border would be outside this field.

euv_slit_x_center

Type: Litho Setup File Commands

Sets a placement X-coordinate for EUV through-slit and shadow bias models.

Usage

euv_slit_x_center [-field] ***x_um***

Arguments

- -field

  An optional argument that indicates the shift is relative to the setting of euv_field_center.

- ***x_um***

  A required argument specifying the X-coordinate center of the EUV field in microns. It must use the coordinate system of the run.

Description

This command is required for usage of EUV through-slit litho models and shadow bias models. The coordinate is used only for offset of through-slit and shadow bias models. Other EUV field offsets (for example, for flare maps) must be specified separately.

Note

Specifying euv_slit_x_center is required for all usages of shadow bias models, even if this keyword is set to 0.

flare_longrange

Type: Litho Setup File Commands

Associates a precomputed long range flare convolution with a flare model contained inside a litho model.

Usage

flare_longrange ***litho_model_name***
{**none** | **constant** ***value*** | {**png** ***pngfile*** [*png_options*]}} [name *string*]

Arguments

- ***litho_model_name***

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

Note

The arguments "**none**", "**constant** ***value***", or "**png** ***pngfile***" are mutually exclusive. One of the arguments must be specified to include long range calculations.

- **none**

  A required argument that specifies not to use a pre-computed long range convolution when doing simulations. If the flare_longrange statement is not specified, this is the default behavior.

- **constant** ***value***

  A required argument that specifies that the long range flare is a constant percentage of clear field transmission given by *value*, where a value of 1 is equal to 100%. The intensity is calculated as: I_constant_flare = kf * Iclear * *value*, where k_f is the coefficient from the flare model, and I_{clear} is the clear field value taking into account both optics and ddm mask transmission.

- **png** ***pngfile*** [*png_options*]

  A required argument set that specifies to use the long-range convolution pre-computed from the fractal kernel of the loaded model and the current layout. The convolution is stored in ***pngfile***, which may be in the current working directory, an explicit path or in the modelpath. The file is generated using RET FLARE_CONVOLVE and must be in PNG format.

  The following additional options can be specified:

  - constant {average | minimum | maximum}

    An option to the **png** ***pngfile*** argument; cannot be specified with delta *value*.

If the constant option is specified, it replaces flare values from the PNG file with a constant flare approximation whose value is computed directly from the flare map based on the setting:

average — Computes the average value in the flare map.

minimum — Returns the smallest value anywhere in the flare map.

maximum — Returns the large value anywhere in the flare map.

Note

When the constant argument is used, flare is added to the entire simulation and does not consider the flare map's extent.

The values which are used by the "constant" option are seen by using the Calibre Workbench getflare command: "getflare range -pngin *png_file*".

- delta *value*[%]

  An option to the **png *pngfile*** argument; cannot be specified with constant.

  If the delta option is specified, the values in the flare map are adjusted. If the specified *value* causes the flare value to fall locally below zero, it is truncated to zero at that point.

  "delta *value*%" — The adjustment is by a percentage, up or down. The *value* must be greater than -100.

  "delta *value*" (no %) — A constant offset, interpreted as a fraction of the clear field transmission, is added. The *value* must be greater than or equal to zero. It is calculated as in **constant *value***.

- flare_map_shift [-field] *x_um y_um*

  An option to the **png *pngfile*** argument.

  If the flare_map_shift option is specified, the flare map is shifted by the amounts specified for *x_um* and *y_um* in microns before being used in the run. Set this to the negative of any coordinate shifts applied to the layout before creating the flare map. The default value for flare_map_shift is 0 0.

  When -field is specified, the adjustment is relative to the EUV field center, specified by euv_field_center. By default, the adjustment is relative to the global coordinates stored in the flare map.

  Note

  The flare_map_shift change is *only* used for flare maps. Other EUV field offsets (such as for through-slit models) must be specified separately.

- name *string*

  An optional argument that defines a named instance of a flare_longrange command. Named instances allow you to have multiple flare_longrange configurations in the same litho model. The string must be unique for each flare_longrange command specified in the setup file.

  Named flare specifications can be used with pw_condition statements by specifying “flare *string*” in the pw_condition command.

Description

The flare_longrange command specifies how intra- and inter-field flare effects are handled. It associates a precomputed EUV long range flare convolution with a flare model contained inside a litho model. Note that Calibre simulation models treat the EUV black border effect separately from flare effects.

As of 2017.2, this command is optional. The default is to not load a precomputed flare model even if it exists in the litho model unless it is explicitly specified with flare_longrange.

Examples

The following example shows the beginning of an EUV setup file. The EUV model is in a litho model; the optical and resist models are not explicitly loaded. The flare model is also in the litho model, but does need to be loaded.

```
LITHO FILE pxopc.setup [/*
   modelpath models
   flare_longrange test_euv none

   layer target hidden
   layer smoothed hidden

   processing_mode flat
   tilemicrons 9 exact

   pxopc_options px.options {
      ...
```

The litho model directory, *models/test_euv*, contains the flare PNG file, a *Lithomodel* file, an EUV optical model directory for each focus point, and other models that the *Lithomodel* file refers to such as resist model(s), DDM models, and so on. Notice that *Lithomodel* does not contain a reference to the flare PNG file in the following example.

Example 3-2. Lithomodel file for an EUV Process

```
version 1
resist cm1.mod
mask 0 {
   optical euv_f+0.0000
   optical euv_f-0.0300 focus -30nm
   optical euv_f+0.0300 focus 30nm
   ...
   optical euv_f-0.0670 focus -67nm
   optical euv_f+0.0670 focus 67nm
   background dark
   mask_layer 0 TRANS clear CX 0.01 CC 0.01 DDM euv.ddm
}
flare test_euv.fmf
```

layer (for main setup file)

Type: Litho Setup File Commands

Defines layer characteristics such as name and type.

Note

Calibre pxOPC primarily uses the layer information defined in the pxopc_options Keywords. Layer properties set outside pxopc_options are only used by commands at the main level, such as setlayer curve.

Usage

layer ***name mtype mtrans*** [*mask dose*]

Arguments

- ***name***

  A required argument specifying the name of the layer. Layer names must be alphanumeric strings less than 32 characters in length. The ***name*** is used in the rest of the main setup file to refer to the layer.

- ***mtype***

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

  hidden — Hidden layers are used to identify a layer that will also be referred to in the pxopc_options block.

  visible — Visible layers are used in optical simulations at the main setup file level. If you specify a layer as visible, you must also include a background specification in the main setup file.

  Note that these are not the same types as the command “layer” on page 157, which is for use inside the pxopc_options block.

- ***mtrans***

  A required argument specifying the transmission value. For hidden layers, the value is ignored. Legal values for this argument are the following:

dark	**clear**
atten ***value***	***real imag***

  If you plan on performing optical simulations using **visible** layers, see the documentation for the appropriate product to understand how ***mtrans*** values will affect the results.

- *mask dose*

  An optional argument pair used with phase shifting masks to allow the simulators to support multiple masks.

- *mask* — The number of the mask to which the layer belongs. The default is mask 0. When using multiple masks, they must be numbered sequentially beginning with 0.
- *dose* — The layer's light energy dose, expressed as a percentage of the full energy dose for the mask. For example, 0.5 indicates 50% of the full energy dose.

Description

Names and describes the optical properties of the input layer(s). Every layer used in the pxOPC run must be declared both at the main setup file level and in the pxopc_options block.

Examples

```
LITHO FILE litho.in [
   modelpath models

   optical_model_load f0 opticalf0

   background dark
   layer target hidden clear   # These layer types and transmissions are
   layer trgtsmth hidden clear # overridden by the pxopc_options layers.

   pxopc_options SIMULATION_OPTIONS {
      background 0.0215 -0.0085
      layer target CORRECTION 0.9785 0.0085 # These values are USED
      layer trgtsmth TARGET 0.9785 0.0085   # by Calibre pxOPC.

      pw_condition NOMINAL optical f0 dose 1 aerial 0.21 weight 1.0

   job 1 open
   job 2 decorate
   job 3 correct
   job 4 refine
   job 5 lmrc
   }

setlayer opcout = pxopc target trgtsmth MAP target \
OPTIONS SIMULATION_OPTIONS
]
```

litho_model

Type: Litho Setup File Commands

Permits in-line definition of a litho model.

Usage

litho_model ***name*** '**{**' ***inline_model_text*** '**}**'

Arguments

- ***name***

  A required argument that specifies the name of the litho model. Once defined, this name can be used anywhere in the setup file where litho models are supported.

- ***inline_model_text***

  A required argument that specifies the litho model text in the same format as the contents of a Lithomodel file. However, instead of defining component models by specifying file names within the litho model directory, these must be replaced by the symbolic names of the models. These names must have previously been defined by "*_model_load" commands that load each component model. The individual "*_model_load" commands themselves can either be file-based or include the model definition between brackets.

Description

This command permits in-line definition of a litho model. It is intended only for use in creating encrypted setup files. Do not use this in standard Calibre OPCverify or Calibre nmOPC setup files. The litho_model command is only permitted if either the setup file is encrypted, or if a special environment variable is set:

```
LITHO_PERMIT_INLINE_LITHO_MODEL_FOR_ENCRYPTED_SETUP_DEVELOPMENT=1
```

modelpath

Type: Litho Setup File Commands

Specifies the directories to search for litho, optical and resist models.

Usage

modelpath *dir*

Arguments

- ***dir***

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

Description

Specifies the directories to search for optical and resist models.

Examples

Absolute path:

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

Relative path:

```
modelpath ../models
```

optical_model_load

Type: Litho Setup File Commands

Loads an optical model.

Usage

optical_model_load ***optical_model_name*** {***filename*** | '**{**' ***inline*** '**}**'}

Arguments

- ***optical_model_name***

  A required argument specifying a name that other commands use to refer to this model.

- ***filename***

  An argument specifying a filename to load containing the optical model. If you do not specify a filename, an in-line optical model must be supplied instead.

- '**{**' ***inline*** '**}**'

  An argument defining an optical model. The entire optical model specification must be enclosed in braces ({ }).

Description

This is a required setup file parameter. At least one optical_model_load command must always be present. Multiple optical models may be loaded and used.

This command loads an optical model found in ***filename***, or defined by specifying model text in-line inside braces. The optical model is given a user-defined ***optical_model_name*** that you use with simulation operations.

If you are using a litho model instead of optical, ddm, and resist models, this command is ignored.

Examples

Here is a typical beginning of a Litho setup file that loads an optical model *./models/opticalf0*:

```
LITHO FILE litho.in [
   modelpath models

   optical_model_load f0 opticalf0
 ...
```

The simulation commands will refer to it by the name "f0".

Related Topics

modelpath

resist_model_load

processing_mode

Type: Litho Setup File Commands

Flattens geometries when the database was constructed hierarchically.

Usage

processing_mode {**hierarchical** | **flat**}

Arguments

- **hierarchical**

  A required argument that specifies the default mode of processing input. In this mode, the layer is split into cells and then tiles. This is more efficient when the layout is extremely hierarchical.

- **flat**

  A required argument that turns on litho-flat processing. This has the following limitations:

  - The amount of data generated may be larger because of pushing flat output into a hierarchical database. This could slow down subsequent SVRF operations.
  - When using flat mode, the run that creates the HDB must use ULTRA FLEX processing or Calibre pxOPC terminates with an error.

  In litho-flat mode, the full chip is split into tiles, which are then processed separately on remote machines. This is more efficient than hierarchical mode when the layout is almost flat.

Description

An optional command that must be used when the run is invoked with calibre -drc -hier. When flat Calibre nmDRC is invoked, this command is ignored.

The processing_mode setting is also ignored when using direct_input or direct_output.

Related Topics

Calibre Data Construction [Calibre Post-Tapeout Flow User's Manual]

progress_meter

Type: Litho Setup File Commands

Estimates the progress of the LITHO operation.

Usage

progress_meter {**on** | **off**}

Arguments

- **on**| **off**

  A required argument that controls whether the progress_meter is on or not. When the command is not specified, it behaves as “on”. Specify either “on” or “off” but not both.

Description

An optional command that controls the progress meter. By default, the progress meter is active (on) and prints the completion percentage as a bar under the heading TIME FOR ALL CELLS in the console window.

This option may add up to 10 percent additional run time depending on the size of the job. The additional time required is listed in the log file as TIME FOR PROGRESS METER.

pxopc_options

Type: Litho Setup File Commands

Command block within a setup file that defines the Calibre pxOPC optimization settings.

Usage

pxopc_options ***name*** '**{**'
options_list
'**}**'

Arguments

- ***name***

  A required argument naming the pxopc_options block so that it may be referred to by other commands.

- ***options_list***

  A required list of parameters for the Calibre pxOPC run surrounded by required brackets ({ }). The parameters must appear each on their own line. The pound sign character (#) comments out the rest of the line.

  The options list is organized as a general section followed by one or more job sections. A dictionary of valid parameters is provided in the section "pxopc_options Keywords".

Description

Creates a uniquely named sub-command block inside a LITHO FILE statement to set pxOPC options. This block is called by a setlayer pxopc command.

Examples

```
LITHO FILE litho.in [
   ...
   pxopc_options opts {
      # general options
      background 0.0215 -0.0085
      layer target CORRECTION 0.9785 0.0085
      layer trgtsmth TARGET 0.9785 0.0085

      pw_condition NOMINAL optical f0 dose 1 aerial 0.21 weight 1.0

      # job sections
      job 1 open
      job 2 decorate
      job 3 correct
      job 4 refine
      job 5 lmrc
   } # end of pxopc_options block

setlayer opcout = pxopc target trgtsmth MAP target OPTIONS opts
] # end of LITHO FILE
```

resist_model_load

Type: Litho Setup File Commands

Specifies a resist model.

Usage

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

Arguments

- ***resist_model_name***

  A required argument specifying a name that other commands will use to refer to this resist model.

- ***filename***

  A required argument specifying a filename to load containing the resist model. If you do not specify a filename, an in-line resist model must be supplied instead.

- '**{**' ***inline_model***'**}**'

  A required argument defining a resist model. The entire specification must be enclosed in braces. Either this argument or a ***filename*** argument must be supplied.

Description

Loads the resist model contained in the file ***filename***, or defined by specifying model text inside braces ({ }). The resist model is assigned the user-defined ***resist_model_name*** that you use with simulation operations such as pw_condition.

Calibre pxOPC requires CM1-type resist models. These can be created in the Calibre WORKbench CM1 Center, as described in the *Calibre WORKbench User's and Reference Manual*.

If you are using a litho model, this command is ignored.

Examples

In this example, the code specifies a resist model location of *models/nom.mod* (modelpath and resist_model_load), which is then used in the NOMINAL pw_condition:

```
LITHO FILE px.setup [ /*
   modelpath models
   optical_model_load f0    f+0.0000
   resist_model_load resist_mod nom.mod
   ...

   pxopc_options px.options {
      ...
```

```
        pw_condition NOMINAL optical f0 dose 1.0 resist resist_mod
        ...
    }
*/ ]
```

setlayer copy

Type: Litho Setup File Commands

Copies the specified layer to a new layer.

Usage

setlayer ***out_layer*** = **copy** ***layer***

Arguments

- ***out_layer***

  A required layer name for the new layer. The name must be a valid Calibre layer name.

- ***layer***

  A required argument that specifies the layer to copy. Only names of layers within the main setup file are recognized, not names from the options block or the main SVRF file.

Examples

Copies the layer t1 to the new layer tx. Layer t1 is defined earlier in the setup file.

```
LITHO FILE pxopc.in[
   ...
   layer t1 hidden clear
   ...
   setlayer tx = copy t1
]
```

setlayer curve

Type: Litho Setup File Commands

Creates “smoothed” shapes based on the input layer.

Usage

setlayer *output_layer_name* = curve *input_layer_name*
[order {linear | quadratic | cubic | *num*}] [cpdist *cpdist_value*] [pts_per_cp *value*]
[maxdist *maxdist_value*] [midpt *midpt_value*] [scale1 *value*] [scale2 *value*]
[midpt_offset *value*] [endpt_offset *value*] [active_layer *active_layer*]
[jog_tol *tol_val* jog_adj *adj_val*] [jog_cleanup_dist *value*] [jog_extra *value*]
[output_cpts *val*] [output_clean_target]
[concave [*set_of_options*]] [convex [*set_of_options*]]
[line_end [*set_of_options*] length *value*]
[space_end [*set_of_options*] length *value*]
[jog [*set_of_options*] length *value*]
[new_cp *corner1 corner2* length *constraint* [length1 *constraint*] [length2 *constraint*]
offset *value* [perp *value*] [delete_corner] [delete_midpt]]

where *set_of_options* is the following set of optional arguments:
[cpdist *cpdist_value*] [maxdist *maxdist_value*] [midpt *midpt_value*][midpt_offset *value*]
[endpt_offset *value*]

Arguments

- ***output_layer_name***

  A required keyword specifying the name of the new output target layer. The new layer is created and the results of the operation are written to the layer. This layer must be a valid Calibre layer containing polygon data.

- ***input_layer_name***

  A required keyword specifying the name of the input layer.

- order {linear | quadratic | cubic | *num*}

  An optional parameter that specifies the B-spline order number. You can either specify a value *num* between 2 and 6, or specify linear (equivalent to a *num* of 2), quadratic (3), or cubic (4). The default value is quadratic.

- cpdist *cpdist_value*

  Specifies the distance between additional control points inserted to the left and right of an original polygon point. Longer distances increase rounding. The default value is 50 nm.

- pts_per_cp *value*

  Specifies the number of vertices for each control point in the output. The default is 10 vertices.

- maxdist *maxdist_value*

  Specifies that the last control point should be at exactly *maxdist_value* away from the original polygon vertex. Though the spline curve will not trace out the same shape as the polygon, at *maxdist_value*, the curve will touch the polygon.

- active_layer *active_layer*

  If the active_layer argument is specified, the output produces control points that overlap between the specified input layer and *active_layer*. This is implemented by making sure that no additional control points are placed within the *active_layer*.

  The effect of this option is to force the output edge to conform to the drawn gate edge while still over the active layer before flaring out at the 90 degree polygon bend. This squares out any 90 degree turns in poly or metal layers that lie near an active region, minimizing any gate flare effect.

- midpt *midpt_value*

  Specifies how many additional control points to add in the middle of a notch/nub that would have had no additional control points otherwise.

- [scale1 *value*] [scale2 *value*]

  If an adjacent edge is a notch or nub and is smaller than *cpdist_value* * scale1, control points on adjacent edges are put at a *dist_value* * scale2 distance, instead of the *cpdist_value*. This is done for better curvature control for various line widths. Both scale1 and scale2 default to 1 if undeclared.

- midpt_offset *value*

  All control points created by the midpt parameter (in the middle of line ends/space ends) are moved to increase the curvature (towards the outside of the polygon for line ends and towards the inside of the polygon for space ends).

- endpt_offset *value*

  All control points situated at the polygon’s corners are moved to increase the curvature (towards the outside of the polygon for convex corners and towards the inside of the polygon for concave corners).

- jog_tol *tol_val* jog_adj *adj_val*

  When specified, this option will ignore all jogs less than or equal to *tol_val* that have neighbor edges smaller or equal to *adj_val*. This is to prevent small jogs from affecting the results of target smoothing prior to OPC.

- jog_cleanup_dist *value*

  In the first stage of generating a spline, target jogs are removed according to the jog removal parameters jog_tol and jog_adj.

If a value for jog_cleanup_dist is set:

- Edges with their length <= jog_tol and with at least one neighbor with length <= jog_adj (if jog_adj is present), and with both neighbors' lengths > jog_tol, and parallel to each other, are classified as jogs.
- Edges classified as jogs are removed.
- Unconnected edges (which are parallel by definition) are shortened by up to jog_cleanup_dist then connected by a line.

Note

This algorithm has a finite interaction distance, thus will cause no hierarchical differences.

If jog_cleanup_dist is not specified, setlayer curve defaults to 0.5 microns (the maximum value possible).

- jog_extra *value*

  An optional argument that specifies to generate additional control points when edges are longer than jog_adj *adj_val*. The control point is inserted *value* from the jog along the edge.

- output_cpts *val*

  If this parameter is present, squares with a half-width equal to the *val* are output instead of the actual curve. The squares represent the curve's control points.

- output_clean_target

  If this parameter is present, the "clean" (after jog removal) target is output instead of the spline.

- concave

  If this parameter is present, the global values set by the previous arguments are overridden in the case of concave corners.

- convex

  If this parameter is present, the control points are set for convex corners.

- line_end

  If this parameter is present, the control points are set to line ends.

- space_end

  If this parameter is present, the control points are arranged for space ends.

- jog

  If this parameter is present, the control points are set for jogs.

- length *value*

  An optional argument (required if line_end, space_end, or jog are used) that specifies the fragment length of the feature.

Note

If concave, convex, line_end, space_end, and jog are not used, or if they are not fully defined, then the values used will be the values set by the original keywords. If those values are not set, then the default values are used.

- new_cp *corner1 corner2* length *constraint* [length1 *constraint*] [length2 *constraint*] offset *value* [perp *value*] [delete_corner] [delete_midpt]

  An optional argument set that specifies when to manually add new control points. When a point could be added by either new_cp or jog_extra, the one from new_cp is inserted.

  corner1 corner2 length *constraint* — Specifies the edge to receive the control point. The corners may be “convex,” “concave,” “noncorner,” or “any.” Noncorner refers to the shallow corners inserted near jogs when jog_tol is specified. If jog_tol is not set, no corners match noncorner.

  length1 *constraint* — An optional constraint that applies to the preceding edge.

  length2 *constraint* — An optional constraint that applies to the edge after *corner2*.

  The length constraints are in microns.

  If the edge specification is symmetric (for example, new_cp convex convex length…) two control points are inserted, because it matches both corners.

  offset *value* [perp *value*] — Specifies where to place the new control point. The offset value must be between -1.0 and 1.0, inclusive, measured inward along the edge from *corner1*.

  perp *value* — An optional perpendicular offset. Positive values move the control point outward from the polygon. Specifying “offset 0.05 perp -0.05” will offset the control point 50 nm away from the first corner and 50 nm into the polygon’s region.

  delete_corner — An optional argument to new_cp that removes an old corner control point to prevent interference with the new control point.

  delete_midpt — An optional argument to new_cp that removes an old midpoint control point to prevent interference with the new control point.

Description

The setlayer curve command allows you to create a “smoothed” target based on the original target. The smoothed target allows you to prevent overcorrection on 2D structures and jogs.

The setlayer curve command implements the curve using a series of B-splines. B-splines are also known as basis splines, a spline function with minimal support with respect to a given degree, smoothness, and domain partition. Spline functions of a given degree, smoothness, and domain partition can be represented as a linear combination of B-splines of that same degree and smoothness over that same partition.

This command generates a spline or spline-derived output in the following sequence:

1. Target jogs are removed according to the jog removal parameters jog_tol and jog_adj. The results are affected by whether you set a value for jog_cleanup_dist. If jog_tol and jog_adj are not set, all line segments are considered edges.
2. Spline control points are generated according to the following parameters:

order	cpdist	pts_per_cp
maxdist	midpt	scale1
scale2	midpt_offset	endpt_offset
active_layer	jog_extra	new_cp

   Control points are derived from existing polygon vertices; additional points are added before and after corners. At least one point is added in the center of short segments.
3. The output is generated according to the control points and the following parameters:
   - pts_per_cp, output_clean_target, output_cpts

Warnings

If pts_per_cp is set to a value higher than five, this warning appears in the transcript:

```
"setlayer curve": Using a pts_per_cp value larger than 5 can cause
unnecessary slowdown in curve and subsequent ops
```

Examples

```
L1 {RET PXOPC m1a FILE "setup.in" MAP L1} DRC CHECK MAP L1 20
...
LITHO FILE setup.in [
...
   layer orig hidden clear           //Note only one layer declared
...
   setlayer smoothed = curve orig    //This creates a smoothed target
   ...
   //Both layers get passed to pxOPC...
   setlayer mask_out = pxopc orig smoothed MAP orig OPTIONS px.options
   ...
   // ... so both need to be listed in the pxopc_options block.
   pxopc_options px.options {  //pxopc-specific commands
      layer orig CORRECTION clear
      layer smoothed TARGET clear
      ...
   }
]
```

setlayer curve_target

Type: Litho Setup File Commands

Creates a target that mimics the final image.

Usage

setlayer *output_layer_name* = curve_target *input_layer_name* [criticalDistance *val*]
[cornerRadius *val*] [cornerRadiusConcave *val*] [cornerRadiusConvex *val*]
[colinearAngleTolerance *degrees*]
[lineEndRatio *val*] [linearRatio {true | false}] [spaceEndRatio *val*]
[lineEndWidth *value*] [spaceEndWidth *value*] [maxJog *val*]
[roundCorner {true | false} [roundedCornerRadius *value*]]
[preferMid {true | false} [midSpanExt *val*] [maxMidSpan *val*]] [enclose *layer* by *val*]
[taglayer *layer overrides*]

Arguments

- ***output_layer_name***

  A required keyword specifying the name of the new output target layer. The new layer is created and the results of the operation are written to the layer. This layer must be a valid Calibre layer containing polygon data.

- ***input_layer_name***

  A required keyword specifying the name of the input layer.

- criticalDistance *value*

  The smallest feature that must be resolved. The default value is (Ro/3), where Ro is the resolution value obtained from the optical model parameters (lambda/NA).

  This parameter is typically specified, rather than using the default value. You can optimize the value around the default value by plus or minus 50% to obtain the best results for your applications.

- cornerRadius *val*

  An optional argument that is used at corners to estimate the achievable target. You should optimize this parameter around the default value by plus or minus 50% to obtain the best results for their applications.

  The value must be greater than 0. The default is criticalDistance.

- cornerRadiusConcave *val*

  The radius of curvature used to estimate the achievable target for concave corners only. This value is also used for space ends. The value must be greater than 0. The default value is cornerRadius, which in turn defaults to criticalDistance.

- cornerRadiusConvex *val*

  The radius of curvature used to estimate the achievable target for convex corners only. This value is also used for line ends. The value must be greater than 0. The default value is cornerRadius, which in turn defaults to criticalDistance.

  Note

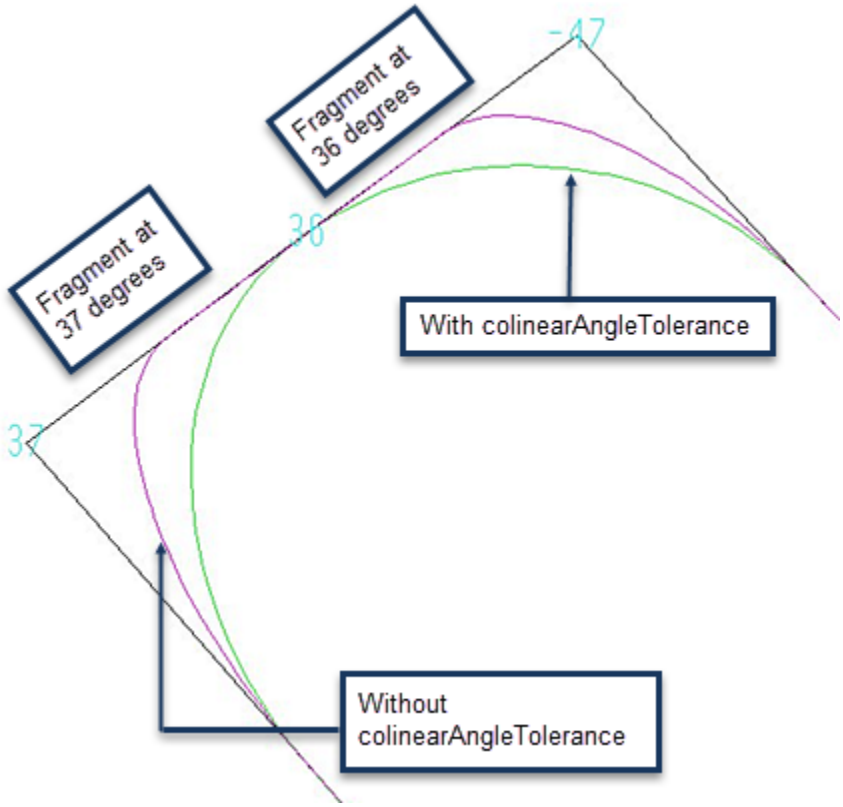
  In jogs and s-shapes that have overlapping convex and concave corners, the larger of cornerRadiusConcave or cornerRadiusConvex is used.

- colinearAngleTolerance *degrees*

  An optional parameter that specifies in degrees how different two adjacent fragments may be and still be considered as collinear. The value must be an integer. The default value is 0 degrees.

  This option is useful when a design contains non-45 skew angles as it can allow the algorithm to better recognize a straight edge, as shown in the figure below. The sharper outer contour (maroon) occurred because the two fragments of the edge were treated independently. Adding “colinearAngleTolerance 1” produced the smoother green contour.

Figure 3-1. Effect of colinearAngleTolerance 1



- lineEndRatio *val*

  Allows for the specification of an elliptical curve at line ends. The curve is tangent to the side at a distance *val* from the corner (where *val* is computed by multiplying the width of the

line by the ratio). The default ratio is 0.5 (resulting in a circular curve). This applies to narrow line ends whose width is less than two times the corner radius.

- linearRatio {true | false}

  An optional keyword that, if set to false, Calibre makes the following adjustments to the curvature of the smooth target:

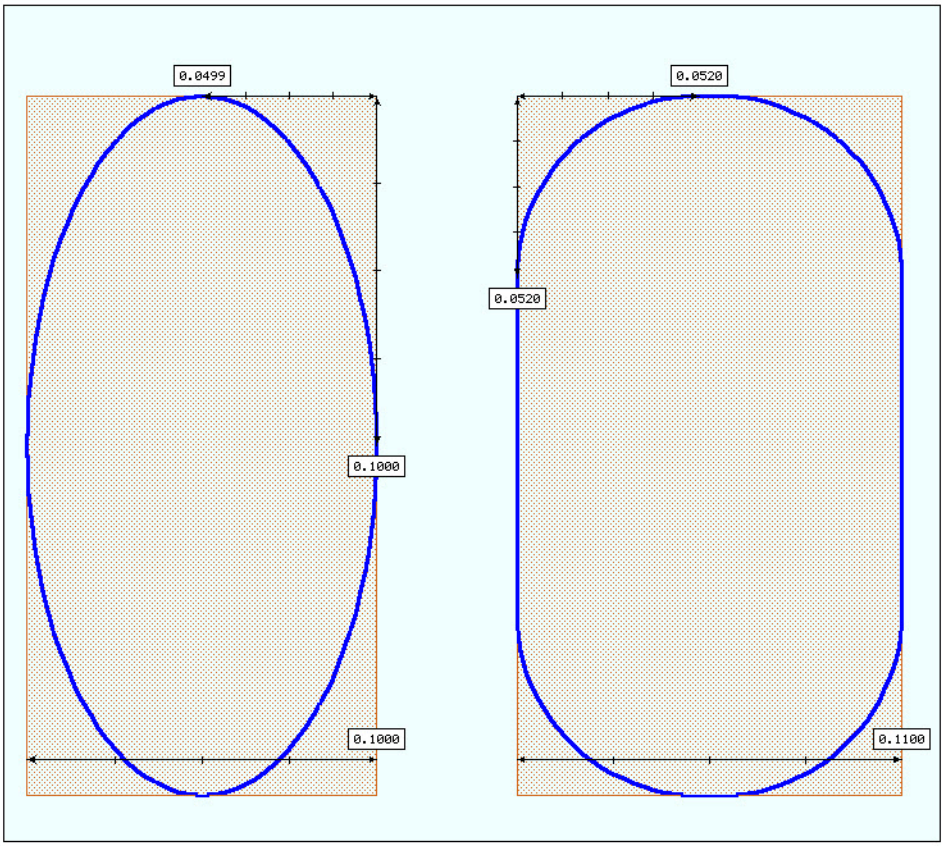
  - For the line ends smaller than lineEndWidth (or if not defined, smaller than 2 * cornerRadius):
    - On the line end (LE) fragment, the tangent point of the curve is set at the middle of the LE fragment.
    - On the LE adjacent fragment, the tangent point of the curve is set at a distance specified by that particular (LE CD) * lineEndRatio away from the corner.
  - For line ends larger than lineEndWidth (or if not defined, smaller than 2 * cornerRadius) the tangent point of the curve is set at a cornerRadius distance away from the corner for both LE and LE adjacent fragments.

  Figure 3-2 illustrates an example using the following settings:

  ```
  criticalDistance 0.050 cornerRadius 0.052 lineEndRatio 1.0
  ```

  Note that sharp transitions in curvature radius can occur when the lineEndWidth is reached.

Figure 3-2. Curvature Example



If linearRatio is set to true, Calibre makes the following adjustments to the tangent point of the curvature at the LE adjacent fragment:

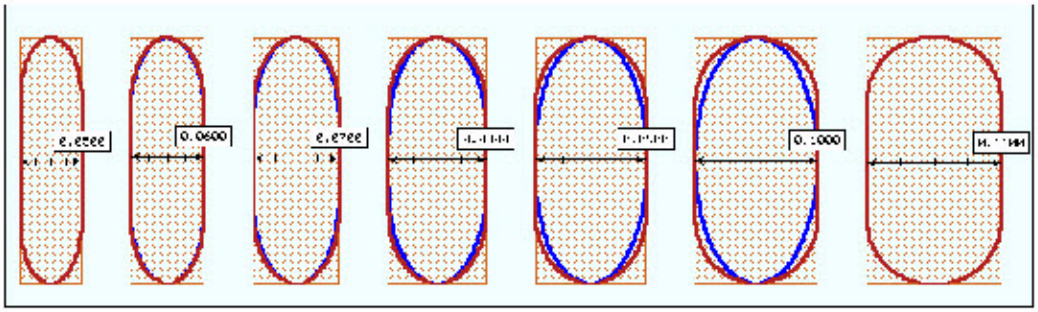
- No adjustment occurs for line ends smaller than cornerRadius.
- For the line ends larger than cornerRadius but smaller than 2 * cornerRadius, the tangent point of the curve is set at a distance specified by that particular (LE CD) * lineEndRatio * multiplier.
- The multiplier is an additional scaling parameter that changes linearly between cornerRadius and 2 * cornerRadius.
- No adjustment occurs for the line ends larger than 2 * cornerRadius.

Figure 3-3 shows a comparison of smooth curves with linearRatio set to false (in blue) and true (in red) with the rest of the parameters as follows:

```
criticalDistance 0.050 cornerRadius 0.052 lineEndRatio 1.0
```

Setting linearRatio allows for smoother transitions.

Figure 3-3. Comparisons of linearRatio True Versus False



Red contours: linearRatio set to true
Blue contours: linearRatio set to false

In cases where the lineEndWidth is defined and is <= 2 * cornerRadius:

- No adjustment occurs for line ends smaller than lineEndWidth.
- For the line ends larger than lineEndWidth but smaller than 2 * cornerRadius, the tangent point of the curve is set at a distance specified by that particular (LE CD) * lineEndRatio * multiplier.
  - The multiplier is an additional scaling parameter that changes linearly between lineEndWidth and 2 * cornerRadius.
- No adjustment occurs for line ends larger than 2 * cornerRadius.

In cases where the lineEndWidth is defined and is > 2 * cornerRadius:

- If the curve's tangent point on the line end is farther from the corner than the cornerRadius value, the point is moved closer to the corner (by changing the lineEndRatio value).

  This means that the tangent point on the line end with a width equal to lineEndWidth (maximal) is set at a distance (the cornerRadius value) from the corner. This provides a smoother transition.
- No adjustment occurs for line ends larger than lineEndWidth.

The default value is false.

- spaceEndRatio *val*

  An optional flag that allows for the specification of an elliptical curve at space-ends. This option otherwise performs similarly to the lineEndRatio option. The default value is the lineEndRatio setting.
- lineEndWidth *value*

  The minimum width at which an edge with two convex corners is not classified as a line end. Edges with length less than *value* are treated as line ends. The units for *value* are user units. The default value is 2 * cornerRadiusConvex.

- spaceEndWidth *value*

  The minimum width at which an edge with two concave corners is not classified as a space end. Edges with length less than *value* are treated as space ends. The units for *value* are user units. The default value is 2 * cornerRadiusConcave.

- maxJog *val*

  The maximum length at which an edge is considered a jog and not a feature. Smaller values cause fewer edges to be considered jogs. The units for *val* are user units. The default value is approximately 0.292893*radius, where radius is the minimum of cornerRadiusConcave, cornerRadiusConvex, and cornerRadius.

- roundCorner {true | false} [roundedCornerRadius *value*]

  An optional parameter that causes Calibre to round concave corners that are shorter than cornerRadiusConcave. When set to true, the target is moved out towards the printed image by roundedCornerRadius. You can use the optional roundedCornerRadius keyword to increase the distance from the original corner that the target is moved. The default value for this parameter is false.

- preferMid {true | false}

  If set to true, this parameter specifies that the midpoint handler should be used where possible, instead of S-steps or quarter circles. This means that instead of a curve being drawn, a straight line will be used. For some curve targets, this is may be a more appropriate fit than curves that are typically used. The default value for this parameter is false.

- midSpanExt *val*

  Specifies a floating point value between 0 and 10 that controls the slope of the line drawn when preferMid is enabled. The default value is 1, which draws a 45-degree angle line through the mid point of a span. Increasing the value of midSpanExt increases the length of the line drawn through the span and results in a shallower slope. By default, the extent of the line drawn is equal to the length of the span (*val* is set to 1). Increasing the value to 2 means that the extent of the line drawn will now be twice as long as the span, and the slope will be much shallower.

- maxMidSpan *val*

  Specifies a length in user units (0 to 1 micron) that determines the maximum size of a span to be handled by the midpoint handler when preferMid is enabled. Spans longer than this value will be represented by curves. The default value is 2/3 of the critical distance (CD).

- enclose *layer* by *val*

  An optional flag that allows the target to be drawn around contacts and vias, where *val* specifies the amount of enclosure or internal distance requested for OPC.

- taglayer *layer overrides*

  An optional keyword set that specifies a marker layer (*layer*) and setting overrides for fragments that interact with *layer*. The following arguments can appear as *overrides*:

  cornerRadiusConcave cornerRadiusConvex lineEndRatio

linearRatio	spaceEndRatio	lineEndWidth
spaceEndWidth	roundCorner	roundedCornerRadius
preferMid	midSpanExt	maxMidSpan

Description

The purpose of the setlayer curve_target command is to create a target that mimics what the final image is going to look like. Using this command, you look at the differences between the image and the curve target and adjust the parameters to more closely match what the final image looks like. By having a feasible target (one that can actually be printed on silicon), the optimization can be simplified, resulting in better OPC results.

Examples

```
setlayer c_target = curve_target target_fill \
            criticalDistance 0.040 \
            cornerRadiusConcave 0.020 \
            cornerRadiusConvex 0.040 \
            lineEndRatio 1.0
```

setlayer pxopc

Type: Litho Setup File Commands

Creates and derives layers.

Usage

setlayer ***out_layer*** = **pxopc** ***in_layer*** [*in_layer*…] **MAP** ***name*** **OPTIONS** ***block_name*** [LASTJOB *id*]

Arguments

- ***out_layer***

  A required layer name to identify the layer containing the optimized mask. The new layer is created and the results of the operation (polygons) are written to the layer. The name must be a valid Calibre layer name.

- ***in_layer***

  The layer or layers to use in the PXOPC procedure. At least one layer is required, but any number of layers may be specified. Layers must appear in the setup file as “layer *in_layer* hidden” or “layer *in_layer* visible”.

- **MAP** ***name***

  A required keyword and layer name indicating the init layer of the exposure.

- **OPTIONS** ***block_name***

  A required keyword and argument indicating the name of the pxopc_options block to use.

- LASTJOB *id*

  An optional keyword and argument for debugging. Valid values for *id* are numbers corresponding to the job numbers. This argument causes the output of the job specified to be written to the ***out_layer***.

  Note

  LASTJOB should only be used on small layouts, not a full chip. Remove LASTJOB in production rule files.

Description

This command creates and derives layers, depending on the options used. One or more of the created layers is eventually passed as output to the RET PXOPC call in your SVRF file.

The order of ***in_layer***s should match the definitions of layers declared using RET PXOPC.

Examples

```
//standard call, suitable for production runs
setlayer px.mask = pxopc px.target px.weight \
                     MAP px.target OPTIONS px.options

//debugging call - checking output of job 1
setlayer px.mask = pxopc px.target px.weight \
                     MAP px.target OPTIONS px.options LASTJOB 1
```

setlayer tilegen

Type: Litho Setup File Commands

Creates an annotated layer to display tile and cell boundaries with properties.

Usage

setlayer *out_layer* = tilegen *value* [property '{'
keywords
'}']

Arguments

- ***out_layer***

  A required layer name for the new layer. This layer must be used in subsequent processing in the SVRF file for the operation to occur.

- ***value***

  A required numeric argument that specifies the style of output. The ***value*** specified must be between -0.1 and 0.1.

  -0.1 to <0 — The tile is sized by ***value*** and written to ***out_layer*** as polygons.

  0 to 0.1 — The boundaries between valid tile regions are sized, and the boundaries are written to ***out_layer***.

- property {

  keywords

  }

  An optional property block specifying tile properties to write to an RDB database for viewing with Calibre RVE.

  Note

  When using a property block, the ***value*** must be negative.

  The following are valid values for keywords:

  location — Outputs the geographical location of the tile as a property. The information may include the cell name, tile index, or coordinates.

  remote — Outputs the name of the host that processed the tile.

  lvheap_current — Outputs the amount of memory used in processing the tile, in MB.

  lvheap_max — Outputs the maximum amount of memory in MB.

  rss_max — Outputs the maximum amount of RAM used by Calibre in MB.

  This argument follows the formatting rules of Calibre OPCverify property blocks, as described in "Property Block" in the *Calibre OPCverify User's and Reference Manual*.

Description

Calibre pxOPC breaks input into smaller areas, referred to as tiles. The setlayer tilegen command creates a supplementary output layer that displays the tile and cell boundaries that were used in the run.

Examples

Example 1 — Grid Output

To display the tile boundaries as a grid:

```
setlayer t1 = tilegen 0
```

Example 2 — Polygon Output

To display the tiles with 0.1 microns between edges:

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

Example 3 — Property Blocks

To output a tile layer with information to aid debugging:

```
setlayer t2 = tilegen -0.1 property {
   location
   remote
}
```

Be sure to include it in a rule check and DRC CHECK MAP statement to write the data to the RDB; for example,

```
T2 = RET PXOPC ... MAP t2
T2_OAS { COPY T2 } DRC CHECK MAP T2 300
T2_RDB { DFM RDB T2 tiles.rdb ALL CELLS CHECKNAME "tile_info" }
```

The RDB file will include data similar to this:

```
...
p 1 4
cell ghdac2b
tile 190
remote node1001
-8272 -8272
-400 -8272
-400 -400
-8272 -400
```

tilemicrons

Type: Litho Setup File Commands

Specifies the size of a tile in microns. Recommended.

Usage

tilemicrons *tile_size* exact

Arguments

- ***tile_size***

  A required argument that specifies the tile size in microns. The default is approximately 100 microns. (Calibre adjusts the size slightly to make it a full multiple of simulation frames if possible.)

Description

This command is recommended. The default tile size across Calibre RET tools is 100 microns, which is typically significantly larger than the optimal value for Calibre pxOPC processing. The mismatch can cause excessive run time and memory usage, which cause warning messages.

The best value to use is influenced by the number of process window conditions in the last job. For sub-45 nm process nodes, the recommended values are shown in Table 3-3.

Table 3-3. Recommended tilemicrons Settings

Number of Process Window Conditions	Recommended Tile Size (microns)
1	24
2	19
3	For EUV: 5 exact For DUV: • Single patterning: 19 exact • Double patterning: 10 exact • Triple patterning: 6 or 7 exact

Examples

```
tilemicrons 19 exact
```

pxopc_options Keywords

This section lists commands that can appear in a pxopc_options block.

The OPTION block_name argument in the setlayer pxopc command specifies a pxopc_options parameter block within the LITHO FILE block. Keywords in this block are case-insensitive.

Many keywords do not accept arithmetic expressions as values. These values must be specified as a number. The error message for this condition is

```
ERROR: PXOPC: option "<keyword>" value <expression> is invalid: real value
is expected
```

The following keywords may appear in the pxopc_options block, specified one per line:

Table 3-4. pxopc_options Summary

Keyword	Description
assist_inside_dose_multiplier	Scales aerial image from negative assist features.
assist_outside_dose_multiplier	Scales aerial image from positive assist features.
assist_inside_threshold_multiplier	Scales print image from negative assist features.
assist_outside_threshold_multiplier	Scales print image from positive assist features.
auto_grid_size_pick	Enables automatic calculation of tile size.
background	Defines background transmission characteristics for each exposure. Up to two exposures can be defined.
check contours_dump	Writes the contours for the job.
check contours_extraprinting	Checks for contours that are not associated with a target polygon.
check contours_hardbridging	Checks if there are contours that correspond to more than one target polygon.
check contours_nesting	Checks for nesting process window contours.
check contours_notprinting	Checks if there are target polygons with no corresponding contours.
check contours_overlap	Checks that contours on different patterns overlap as intended.
check mask_empty	Checks if the initial mask layer is empty for the current job.
check mask_hardbridging	Checks if any polygon on the mask interacts with more than one polygon on the target layer.
check mask_is_broken	Checks if any polygon on the target layer interacts with more than one polygon on the mask layer.

Table 3-4. pxopc_options Summary (cont.)

Keyword	Description
check mask_isG45	Checks for edges on the mask that are not a multiple of 45 degrees.
check mask_isG90	Checks for edges on the mask that are at an angle to the axes.
check mask_withSRAFs	Checks whether any areas of the mask layer lack SRAFs.
check once	Controls the running of other checks.
check target_empty	Checks if the target layer is empty for the current job.
constraint	Adds image slope, ILS, or NILS optimization to the default objective of the job.
constraint contour	Adds EPE optimization to the default objective of the job.
ddm	Calls the DDM model.
debug_level	Writes optimization rates to stdout.
enable_default_checks	Runs a set of default checks for each job.
enable_visible_layer	Allows layers of type “visible”. The default is to exit with an error for visible layers.
generate_g45	Enables generation of 45-degree edges in the Detail job output.
init	Identifies the initial mask.
iterations	Specifies how many iterations the optimization algorithm should run.
job	Calls the various tasks and allows local overrides of settings.
layer	Defines layer characteristics such as name and type.
lmrc_inside_assist_cuts_enabled	Enables extra print checking for SRAFs cut into polygons.
mask_periodicity	Forces the output to repeat patterns periodically.
mask_symmetry	Forces the output to the specified symmetry. Should only be used with symmetric optical models and input layers.
metric_impact	Customizes the job objective.
mrc_align_edges	Specifies what edges to attempt to align.

Table 3-4. pxopc_options Summary (cont.)

Keyword	Description
mrc_enabled	Activates MRC constraints for curvilinear output.
mrc_iterations	Specifies how many iterations the MRC algorithm should run.
mrc_max_notch_height	Sets the maximum height allowed for a notch before it is considered to be an MRC violation.
mrc_max_nub_height	Sets the maximum height allowable for a nub before it is considered an MRC violation.
mrc_min_area	Constrains the minimum SRAF area during MRC.
mrc_min_assist_edge	Sets a target SRAF length for MRC geometries.
mrc_min_edge	Sets a target edge length for MRC geometries.
mrc_min_external	Constrains external spacing during MRC.
mrc_min_external_main2sraf	Constrains external spacing during MRC between a main feature and an SRAF.
mrc_min_external_main2visible	Constrains external spacing during MRC between a main feature and a feature on the visible layer only.
mrc_min_external_sraf	Constrains external spacing during MRC for SRAFs only.
mrc_min_external_sraf2visible	Constrains external spacing during MRC between an SRAF and a feature on a visible layer only.
mrc_min_internal	Sets the minimum distance between two internal edges.
mrc_min_internal_sraf	Constrains internal spacing during MRC for SRAFs only.
mrc_min_length	Sets the minimum distance between two internal edges.
mrc_min_notch_length	Sets the minimum length allowed for a notch before it is considered to be an MRC violation.
mrc_min_nub_length	Sets the minimum length allowed for a nub before it is considered to be an MRC violation.
mrc_min_rect_length	Sets the minimum distance between two internal edges.
mrc_min_rect_width	Sets the minimum distance between two internal edges.

Table 3-4. pxopc_options Summary (cont.)

Keyword	Description
mrc_min_square	Sets the minimum distance between two internal edges.
mrc_min_small_feature_area_excess	Sets the minimum area for features on the mask.
mrc_small_feature_size	Sets the minimum distance between two internal edges.
objective_region	Optimizes a region using specified objectives.
pw_bridging	Performs additional bridging correction for shapes within a specific pw_condition.
pw_condition	Creates print image conditions for dose, focus, resist, and aerial threshold.
pw_condition_group	Creates a named set of automatically generated conditions based on specified process window parameters. The conditions are only accessible as a group; not individually.
pw_dose_latitude_factor	Controls the relative emphasis given to the dose latitude during process window optimization.
pw_enclose	Identifies a layer that should remain enclosed during pw_pinching and pw_bridging corrections.
pw_focus_base_points	Defines the defocus conditions to use for interpolation of aerial images.
pw_focus_range_factor	Controls the relative emphasis given to a process window condition group during process window optimization.
pw_pinching	Performs additional pinching correction for shapes within a specific pw_condition.
pw_select	Restricts simulation to only the specified processes.
pw_tolerance_bound	Defines inner and outer contours for a process window condition group in CDOF and EL jobs.
pw_vertical_constraint	Optimizes contours to be inside or outside of the shapes of the specified layer for a certain process window condition or group.
rate	Sets the rate of mask change per iteration.
scatter_assist_type	Specifies where to place SRAFs relative to the target layer.

Table 3-4. pxopc_options Summary (cont.)

Keyword	Description
scatter_belt	Identifies how far from the target the SRAF rings should extend.
scatter_belt_negative	Identifies how far inside the target negative SRAFs should extend.
scatter_offset	Specifies the minimum distance between the main feature and positive SRAFs.
scatter_offset_negative	Specifies the minimum distance between the main feature edge and negative SRAFs.
scatter_offset_type	Sets whether scatter_offset and scatter_offset_negative are calculated based on mask or target shapes.
set_on_target	Improves EPE for the specified process window condition.
size_by	Resizes the input layer polygons.
size_max_shift_edge	Limits the resizing of input layer polygons by “size_by percent”.
straight_contour	Reduces ripples on contours.
suppress_extra_printing	Controls the area and aggressiveness for suppressing SRAF printing.
vary_pw_group	Determines which process window condition group to optimize. Required when two or more process window condition groups are defined.
weight_background	Sets the default weight.
weight_layer	Identifies polygon layers which define more important areas for optimization.

Note

A setup file must either use the litho model syntax for all commands, or not at all. Standard syntax and litho model syntax cannot be mixed.

assist_inside_dose_multiplier

Type: pxopc_options Keywords - any section

Scales aerial image from negative assist features.

Usage

assist_inside_dose_multiplier ***mult_value***

Arguments

- ***mult_value***

  A required argument giving the value by which to scale the assist features. The default is 1.

Description

An optional command that scales the aerial image near assist features *inside of the target* shapes (for example, negative scattering bars) by the given ***mult_value***.

By default, assist features are only placed outside of the target, so this command has no effect. To place shapes inside the target set scatter_assist_type to “inside” or “both”.

Examples

```
scatter_assist_type both
assist_inside_dose_multiplier 0.980
```

assist_outside_dose_multiplier

Type: pxopc_options Keywords - any section

Scales aerial image from positive assist features.

Usage

assist_outside_dose_multiplier ***mult_value***

Arguments

- ***mult_value***

  A required argument giving the value by which to scale the assist features. The default is 1.

Description

An optional command that scales the aerial image near assist features *outside of the target* shapes (for example, SRAFs) by the given ***mult_value***. If scatter_assist_type is set to “inside” this command has no effect.

Examples

```
assist_outside_dose_multiplier 1.1
```

assist_inside_threshold_multiplier

Type: pxopc_options Keywords - any section

Scales print image from negative assist features.

Usage

assist_inside_threshold_multiplier ***mult_value***

Arguments

- ***mult_value***

  A required argument giving the value by which to scale the assist features. The default is 1.

Description

An optional command that scales the print image near assist features *inside of the target* shapes (for example, negative scattering bars) by the given ***mult_value***.

By default, assist features are only placed outside of the target, so this command has no effect. To place shapes inside the target set scatter_assist_type to “inside” or “both”.

Before version 2013.4, this functionality was known as “assist_inside_dose_multiplier.” This change was made to improve consistency across different dose values. Generally it is better to scale by aerial image (light exposure) than print image (simulated resist reaction).

Examples

```
scatter_assist_type both
assist_inside_threshold_multiplier 0.980
```

assist_outside_threshold_multiplier

Type: pxopc_options Keywords - any section

Scales print image from positive assist features.

Usage

assist_outside_threshold_multiplier *mult_value*

Arguments

- ***mult_value***

  A required argument giving the value by which to scale the assist features. The default is 1.

Description

An optional command that scales the print image near assist features *outside of the target* shapes (for example, SRAFs) by the given ***mult_value***. If scatter_assist_type is set to “inside” this command has no effect.

Before version 2013.4, this functionality was known as “assist_outside_dose_multiplier.” This change was made to improve consistency across different dose values. Generally it is better to scale by aerial image (light exposure) than print image (simulated resist reaction).

Examples

```
assist_outside_threshold_multiplier 1.1
```

auto_grid_size_pick

Type: pxopc_options Keywords - general section only

Enables automatic calculation of tile size.

Usage

auto_grid_size_pick {**yes** | **no**}

Arguments

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

  A required argument specifying whether to have Calibre pxOPC calculate an optimal tile size (yes) or use the settings specified in the setup file (no, the default).

Description

When this keyword is set to yes, Calibre pxOPC partitions the layout in such a way as to reduce run time.

background

Type: pxopc_options Keywords - general section only

Litho Setup File Commands

Defines background transmission characteristics for each exposure. Up to two exposures can be defined.

Usage

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

Arguments

- **dark**

  A literal argument designating dark field imaging. Transmission added to the mask will be {layer ***mtrans*** + 0}.

- **clear**

  A literal argument designating clear field imaging. Transmission added to the mask will be {layer ***mtrans*** + 1}.

- **atten *factor***

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

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

  ```
  actual transmission = background + layer transmission values
  ```

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

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

  IM(bg) = 0
  ```

  For example:

  ```
  background atten 0.06
  ```

  is equivalent to

  ```
  background -0.245 0
  ```

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

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

- ***real imag***

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

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

  background (***real imag***) + layer (***real imag***) = transmission value

  For example:

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

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

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

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

Description

This keyword specifies the background transmission level for one or two exposures. Background transmission is added to the ideal Kirchoff mask transmission for the layers in order to generate the overall mask transmission.

If you are using a litho model instead of optical, ddm, and resist models, this command is ignored. If you are not using a litho model, this command is *required*. If a ddm model specifies a different background transmission, the ddm value is used instead of the background-specified value.

Note

Calibre pxOPC uses only the background setting specified in the pxopc_options block. If a background statement appears in the setup file outside the pxopc_options block it is ignored.

Examples

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

```
background atten 0.08
layer VIA target clear
```

Related Topics

layer

layer (for main setup file)

check contours_dump

Type: pxopc_options Keywords — job sections only

Writes the contours for the job.

Usage

check contours_dump {**yes** | **no**} [dir_name *absolute_path*]

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

- dir_name *absolute_path*

  An optional argument indicating the directory to write the GDS files in. If this option is not specified, a directory named *check_dump* is created in the directory in which the run was started.

  If the specified directory does not already exist, it is created.

Description

An optional keyword that writes each contour to two GDS files. The GDS filenames are of the form

contour_<location>_job_<number>_pattern_<index>_pw_<cond_name>.gds

and

contour_<location>_job_<number>_pattern_<index>_pw_<cond_name>_shifted.gds

The *shifted.gds* shifts the coordinates to (0, 0) but is otherwise the same.

When the “check once” or “enable_default_checks” keywords are set to “yes” along with “check contours_dump,” the contours are written at the beginning of the job and the location part of the filename ends in “_before”.

Related Topics

check once

enable_default_checks

check contours_extraprinting

Type: pxopc_options Keywords — job sections only

Checks for contours that are not associated with a target polygon.

Usage

check contours_extraprinting {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check for contours that do not correspond to any target polygons. If the check finds such, it prints the following warning to the transcript:

```
WARNING: Extra printing, job <number>, location <string>, pattern <index>,
pw name <cond_name>
```

For hierarchical runs, the check can also issue the warning as “Extra printing in the tile filter.”

If you receive such a warning for the Correct or Finetune job, increase the number of iterations in the Correct job. If you receive the warning for the Detail job, check previous jobs for options that control extra printing.

check contours_hardbridging

Type: pxopc_options Keywords — job sections only

Checks if there are contours that correspond to more than one target polygon.

Usage

Syntax 1: Target-Based

check contours_hardbridging {**yes** | **no**}

Syntax 2: Contour to Contour

check contours_hardbridging ***pw_name pattern pw_name pattern***

Arguments

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

  A required argument for Syntax 1 indicating whether to run the target-based check (yes) or not (no, the default).

- ***pw_name pattern***

  A required pair of arguments for Syntax 2 identifying a contour. Two contours must be supplied.

  The ***pw_name*** must match a **pw_select** ***cond***. The ***pattern*** must match a valid pattern ***index***. For single patterning runs, set both pattern values to 0.

Description

An optional keyword that checks for hard bridging. It has two forms.

Target-Based Contour Bridging Check

Syntax 1 checks if any contour corresponds to two or more target polygons. If this is identified, the check prints the following warning to the transcript:

```
WARNING: Hard bridging, job <number>, location <string>, pattern <index>,
pw name <cond_name>
```

For hierarchical runs, the check can also issue the warning as “Hard bridging in the tile filter.”

If you receive this warning for the Correct, Refine, or Finetune job, increase the number of iterations in the Correct job. If you receive this warning for the Detail job, increase the number of iterations in the Refine or Finetune job.

Contour to Contour Bridging Check

Syntax 2 checks that the contours corresponding to the process window conditions and patterns do not touch. If they do, the check prints the following warning to the transcript:

```
WARNING: Hard bridging between two contours in the tile filter, job
<number>, location <string>, first contour: pattern <index>, pw name
<cond_name> second contour: pattern <index>, pw name <cond_name>
```

If you receive this warning for the Detail job, increase the number of iterations in the Refine or Finetune jobs.

The contour-to-contour bridging check is not run by enable_default_checks.

check contours_nesting

Type: pxopc_options Keywords — job sections only

Checks for nesting process window contours.

Usage

check contours_nesting ***pw_name_1 pattern_1 pw_name_2 pattern_2***

Arguments

- ***pw_name_1 pattern_1***

  pw_name_2 pattern_2

  Two required pairs of arguments identifying two contours.

  The ***pw_name*** must match a **pw_select** ***cond***. The ***pattern*** must match a valid pattern ***index***. For single patterning runs, set both pattern values to 0.

Description

An optional keyword that checks that the contour corresponding to the process window condition and pattern specified first is inside the contour corresponding to the process window condition and pattern specified second. If the second contour is inside the first contour, the check prints the following warning to the transcript:

```
WARNING: Contour with pattern <index>, pw name <cond_name> is not inside
contour with pattern <index>, pw name <cond_name>
```

This check is not run by enable_default_checks. It requires that the job specify two pw_condition statements at a minimum.

If you receive this warning for the Detail job, try using different process window conditions or making the conditions more aggressive.

Examples

The following code identifies two process window conditions to check:

```
pxopc_options PXOPC{
  ...

  pw_condition NOMINAL optical opt0 dose 1.000 mask_size 0.0000 resist RESIST
  pw_condition outer   optical opt1 dose 1.450 mask_size 0.0025 resist RESIST
  pw_condition inner   optical opt2 dose 0.52 mask_size -0.0025 resist RESIST
...

  job 2 decorate
     pw_select NOMINAL outer inner
     check contours_nesting inner 0 outer 0
...}
```

Related Topics

enable_default_checks

check contours_notprinting

Type: pxopc_options Keywords — job sections only

Checks if there are target polygons with no corresponding contours.

Usage

check contours_notprinting {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that checks if there are target polygons that do not correspond to any contours in the current job. If any unmatched target polygons are found, it prints the following warning:

```
WARNING: Hard pinching, job <number>, location <string>, pattern <index>,
pw name <cond_name>
```

For hierarchical runs, the check can also issue the warning as “Hard pinching in the tile filter.”

If you see this warning for the jobs, try these changes:

- **Decorate job**
  - Ensure that the Open and Decorate jobs are using the same process window conditions.
  - Increase the number of iterations in the Open job.
  - Tune the rate in the Decorate job.
- **Correct job**
  - Ensure that the Open and Decorate jobs are using the same process window conditions.
  - If the Decorate job generates negative SRAFs, add an Unclose job before the Correct job.
  - If the conditions are using an aerial or CTR model instead of a CM1 model, try one of these changes:

    Increase the number of iterations in increments of 2.

    Add “inner” and “outer” process window conditions to the Correct job.

    Use a CM1 model instead of the aerial or CTR model.

- **Refine or Finetune job**
  - Reduce the rate of the Correct job
- **Detail job**
  - Decrease the rate of the Refine or Finetune job.

check contours_overlap

Type: pxopc_options Keywords — job sections only

Checks that contours on different patterns overlap as intended.

Usage

check contours_overlap *pw_name pattern pw_name pattern*

Arguments

- ***pw_name pattern***

  A required pair of arguments identifying a contour. Two contours must be supplied.

  The ***pw_name*** must match a **pw_select *cond***. The ***pattern*** must match a valid pattern ***index***. The two ***pattern*** values should be different.

Description

This optional keyword is intended for use with multi-patterning runs. It checks that when target polygons on different patterns overlap (that is, there is a “stitch”), the corresponding contours interact with each other. When the contours do not interact near a stitch, the check prints the following warning to the transcript:

```
WARNING: Hard pinching between two contours in the tile filter, job
<number>, location <string>, first contour: pattern <index>, pw name
<cond_name> second contour: pattern <index> pw name <cond_name>
```

If you see this warning for the Detail job, increase the number of iterations in the Refine or Finetune job.

check mask_empty

Type: pxopc_options Keywords — job sections only

Checks if the initial mask layer is empty for the current job.

Usage

check mask_empty {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the mask layer at the start of the job. If the check determines the initial mask is empty, it prints the following warning to the transcript:

```
WARNING: Mask inside the tile filter is empty, job <number>, location
<string>, pattern <index>, exposure <number>
```

If a tile filter is defined, the check runs only on the region within the filter.

If you receive this warning for the Open job, check whether the correction layer is set to the layer with the intended design.

check mask_hardbridging

Type: pxopc_options Keywords — job sections only

Checks if any polygon on the mask interacts with more than one polygon on the target layer.

Usage

check mask_hardbridging {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the mask layer. If the check determines that a mask polygon interacts with multiple target polygons, it prints the following warning to the transcript:

```
WARNING: Single mask polygon interacts multiple target shapes, job
<number>, location <string>, pattern <index>
```

If you receive this warning, try these changes:

- **Open job —** Check whether the correction layer is set to the layer with the intended design.
- **Decorate job —** Reduce the rate of the Open job. (Note that for the Decorate job it may be acceptable for the mask to interact with multiple target shapes.)
- **Correct job —** Reduce the rate of the Correct job.
- **Refine or Finetune job —** Increase the number of iterations of the Correct job.
- **Detail job —** Increase the number of iterations of the Refine or Finetune job.

check mask_is_broken

Type: pxopc_options Keywords — job sections only

Checks if any polygon on the target layer interacts with more than one polygon on the mask layer.

Usage

check mask_is_broken {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the mask layer. If the check determines that more than one mask polygon interacts with a single target polygon, it prints the following warning to the transcript:

```
WARNING: Mask is broken, job <number>, location <string>, pattern <index>
```

If you receive this warning, try these changes:

- **Open job —** Check whether the correction layer is set to the layer with the intended design.
- **Correct job —** Increase the number of iterations for the Open job.
- **Refine or Finetune job —** Increase the number of iterations of the Correct job.
- **Detail job —** Increase the number of iterations of the Refine or Finetune job.

Related Topics

check mask_hardbridging

check mask_isG45

Type: pxopc_options Keywords — job sections only

Checks for edges on the mask that are not a multiple of 45 degrees.

Usage

check mask_isG45 {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the mask layer. If the check determines the mask contains edges at an angle other than 0, 45, or 90 degrees, it prints the following warning to the transcript:

```
WARNING: Mask is non-45 degree, job <number>, location <string>, pattern
<index>, exposure <number>
```

If you receive this warning, try these changes:

- **Open job —** Check whether the correction layer is set to the layer with the intended design.
- **Decorate job —** Reduce the rate of the Open job.
- **Detail job —** If possible, reduce the customizations in your recipe. This warning should not occur in a typical recipe.

check mask_isG90

Type: pxopc_options Keywords — job sections only

Checks for edges on the mask that are at an angle to the axes.

Usage

check mask_isG90 {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the mask layer. If the check determines the mask contains non-orthogonal edges, it prints the following warning to the transcript:

```
WARNING: Mask is non-orthogonal, job <number>, location <string>, pattern
<index>, exposure <number>
```

If you receive this warning, try these changes:

- **Open job —** Check whether the correction layer is set to the layer with the intended design.
- **Detail job —** If possible, reduce the customizations in your recipe. This warning should not occur in a typical recipe.

check mask_withSRAFs

Type: pxopc_options Keywords — job sections only

Checks whether any areas of the mask layer lack SRAFs.

Usage

check mask_withSRAFs {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that checks if the mask layer contains SRAFs for the current job. SRAFs are mask polygons that do not interact with any target polygons. If a tile is found without SRAFs, it prints a warning to the transcript:

```
WARNING: No SRAFs inside the tile filter, job <number>, location <string>,
pattern <index>, exposure <number>
```

If you receive this warning, try these changes:

- **Correct job**
  - Increase iterations for the Decorate job in increments of 2.
  - Verify that you have specified at least three process window conditions for the Decorate job.
- **Refine or Finetune job**
  - Reduce the value of assist_outside_dose_multiplier, if used.
  - Make the process window conditions for controlling extra printing less aggressive.
- **Detail job**
  - Make sure that options for controlling extra printing are consistent for all previous jobs.
  - Make sure that all SRAF MRC constraints were fulfilled by the previous job.

check once

Type: pxopc_options Keywords — job sections only

Controls the running of other checks.

Usage

check once {**yes** | **<u>no</u>**}

Arguments

- **yes** | **<u>no</u>**

  A required argument indicating whether to run checks only one time (yes) or on all iterations (no, the default).

Description

An optional keyword that can be set once per job. If “check once yes” is added to a job, all checks apply only on the first iteration.

When no check keywords are present, Calibre pxOPC runs certain default checks on the first iteration. (See “enable_default_checks” on page 145 for the checks. Note that enable_default_checks sets check once to yes.)

- If any custom checks are added to jobs without specifying enable_default_checks or check_once, the custom checks are run on only the first iteration.
- If “check once no” is set in any job, the checks for that job run on all iterations.
- If “enable_default_checks no” is set and some jobs set custom checks, those checks run on all iterations unless “check once yes” is also listed in the job.

Within a job, all checks are either run on the first iteration only or on all iterations. There is no way to specify that a subset of checks within a job run on only the first iteration and the rest run on all iterations.

check target_empty

Type: pxopc_options Keywords —job sections only

Checks if the target layer is empty for the current job.

Usage

check target_empty {**yes** | **no**}

Arguments

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

  A required argument indicating whether to run this check (yes) or not (no, the default).

Description

An optional keyword that turns on a check of the target layer. If the check determines the target is empty, it prints the following warning to the transcript:

```
WARNING: Target inside the tile filter is empty, job <number>, location
<string>, pattern <index>, exposure <number>
```

If a tile filter is defined, the check runs only on the region within the filter.

If you receive this warning for the Open job, check whether the target layer is set to the layer with the intended design.

constraint

Type: pxopc_options Keywords — job sections only

Adds image slope, ILS, or NILS optimization to the default objective of the job.

Usage

constraint {**slope** | **ils** | **nils**} {**greater** ***value*** | **maximize**} [on *layer*]
[initial_weight *value*]
[pw *pw_cond_name* [pattern *index*] [*pw_cond_name* [pattern *index*]]…]

[outer_iterations *N*]

Arguments

- **slope** | **ils** | **nils**

  A required argument specifying which image value to optimize:

  - **slope** — Optimizes the image slope.
  - **ils** — Optimizes the image log slope, commonly referred to as ILS.
  - **nils** — Optimizes the normalized image log slope, commonly referred to as NILS.

  When an image is exactly on target, the values calculated by **ils** and **slope** are the same.

- **greater** ***value***

  An argument that specifies a value along the contour that the optimization tries to exceed. Units are in 1/um for ils.

  Either **greater** ***value*** or **maximize** must be specified, but not both.

- **maximize**

  An argument that specifies the optimization should try for the greatest possible value along the contour.

- on *layer*

  An optional argument that specifies a layer name. Optimization runs on the contours or shapes contained on the specified layer. If this argument is not specified, optimization is done upon the printed contour.

- initial_weight *value*

  An optional argument specifying an initial weight for the constraint optimization versus the job objective. The default is 0.01. Larger values may help achieve the constraint value.

- pw *pw_cond_name* [pattern *index*] [*pw_cond_name* [pattern *index*]]…

  An optional set of arguments restricting the constraint optimization to a particular process window. When specifying multiple process windows, the pw keyword should only appear once.

The *pw_cond_name* must also appear in a pw_select keyword for the job and a pw_condition definition.

- outer_iterations *N*

  An optional separate command that allows the constraint weight to be adjusted. The worse the constraint is relative to the user-specified value, the more weight is given to the constraint optimization. If outer_iterations is not included in the job keywords, the optimization uses a fixed weight and the change to ILS is generally slight.

  The default value of outer_iterations is 1. A good initial value for experimenting with this keyword is in the range of 5 to 10. The total number of iterations that will be performed is iterations * outer_iterations.

Description

The optional constraint keyword works to maximize the minimum image slope, image log-slope (ILS), or normalized image log slope (NILS) value along a target. It is typically used in the final job of the setup file. Optimizing the image slope, ILS or NILS can improve dose latitude (also referred to as exposure latitude, or EL) and line edge roughness.

The equation for image slope (slope keyword) is dI/dx, where I is image intensity.

The equation for image log-slope (ils keyword) is d(ln(I))/dx.

The equation for normalized image log-slope (nils keyword) is w*d(ln(I))/dx, where w is the feature width.

These values are measured on a *contour*. If the contour is not close to the target, the ILS or NILS measured by the algorithm and the ILS or NILS measured on the target can be significantly different. Use set_on_target to keep the contour close to the target and to also minimize EPE degradation. The **ils** and **nils** options produce similar results, and the user can select their preferred metric.

To measure ILS, use the intensity_ilscheck setlayer command, documented in the *Calibre OPCverify User's and Reference Manual*. To measure NILS, use the intensity_nilscheck setlayer command, documented in the *Calibre OPCverify User's and Reference Manual*.

Examples

The following lines show how to properly add ILS optimization to a job. It uses set_on_target to keep the nominal contour near the target layer (smoothed_target) and suppress_extra_printing to ensure that attempts to improve ILS do not cause SRAF printing.

```
pw_select nominal ...
constraint ils greater 65 on smoothed_target initial_weight 0.03
set_on_target nominal weight 16
suppress_extra_printing nominal
```

Depending on the job, the pw_select statement can include additional conditions.

Related Topics

set_on_target

suppress_extra_printing

constraint contour

Type: pxopc_options Keywords — job sections only

Adds EPE optimization to the default objective of the job.

Usage

constraint contour {**inside** | **outside** *layer1*} [region *layer2*] [initial_weight *value*] [pw *pw_cond_name* [pattern *index*] [*pw_cond_name* [pattern *index*]]…]

Arguments

- **inside** | **outside** *layer1*

  A required argument set specifying the location of the image contour relative to the polygons of the specified layer.

- region *layer2*

  An optional argument set that specifies a layer name. Optimization runs on the contours or shapes contained on the specified layer. If this argument is not specified, optimization is done upon the printed contour.

- initial_weight *value*

  An optional argument set specifying an initial weight for the constraint optimization versus the job objective. The default is 0.01. Larger values may help achieve the constraint value.

- pw *pw_cond_name* [pattern *index*] [*pw_cond_name* [pattern *index*]]…

  An optional set of arguments restricting the constraint optimization to a particular process window. When specifying multiple process windows, the pw keyword should only appear once.

  The *pw_cond_name* must also appear in a pw_select keyword for the job and a pw_condition definition.

Description

The optional constraint keyword works to minimize the edge placement error (EPE) value along a target. It is typically used in the final job of the setup file.

To measure EPE, use the measure_epe setlayer command, documented in the *Calibre OPCverify User's and Reference Manual*.

ddm

Type: pxopc_options Keywords - general section only

Calls the DDM model.

Usage

ddm ***ddm_model_name***

Multi-patterning Usage

ddm ***ddm_model_name*** [*ddm_model_name*]

Arguments

- ***ddm_model_name***

  A required argument identifying the DDM model to load. The ddm_model_name must be the same as the name used in the ddm_model_load command.

Description

Calls the DDM model for use with Calibre pxOPC runs. The effects of DDM are most likely to be detectable in areas that have had issues with biasing.

This command requires the ddm_model_load command be specified in the main Litho setup file.

The number of DDM models should be equal to the number of decomposed layers.

This command is not allowed in runs that use litho models instead of traditional models. The DDM information is included in the litho model itself.

Related Topics

job

litho_model

debug_level

Type: pxopc_options Keywords - general section only

Writes optimization rates to stdout.

Usage

debug_level {**0** | **1**}

Arguments

- **0**

  A required argument indicating no extra information should be printed. This is the default.

- **1**

  A required argument that causes optimization rates to be printed to standard output (stdout).

Description

The debug_level command controls whether optimization rates are output. This can be useful for tuning recipes.

Note

Do not use this command in a production setup file. It makes the log file very large, and slows down the run.

Examples

```
debug_level 0
```

Related Topics

rate

enable_default_checks

enable_default_checks

Type: pxopc_options Keywords — general section only

Runs a set of default checks for each job.

Usage

enable_default_checks {**yes** | **no**}

Arguments

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

  A required argument determining whether the checks are run (yes, the default) or not (no).

Description

Unless the default checks are disabled, Calibre pxOPC checks certain properties for each job. If problems are found, the check prints a warning to the transcript.

Table 3-5. Default Checks for Each Job Type

Job	Default Checks
Open	check mask_empty check target_empty check mask_hardbridging check mask_is_broken check mask_isG90 check mask_isG45
Decorate Decoratedarksraf	check contours_notprinting check mask_empty check mask_hardbridging check mask_is_broken check mask_isG45
Correct	check contours_hardbridging check contours_notprinting check mask_hardbridging check mask_withSRAFs

Table 3-5. Default Checks for Each Job Type (cont.)

Job	Default Checks
Refine Finetune	check contours_extraprinting check contours_hardbridging check contours_notprinting check mask_hardbridging check mask_is_broken check mask_withSRAFs
LMRC	No extra checks.
Detail	check contours_extraprinting check contours_hardbridging check contours_notprinting check mask_hardbridging check mask_is_broken check mask_isG45 check mask_isG90 check mask_withSRAFs

The default checks are run once, on the first iteration of the job, unless “check once no” is specified. (The algorithm defining the default checks also sets “check once yes” for each job.)

You can also set checks individually for a job. Some checks, such as check contours_overlap, are not run by default and need to be individually added to jobs.

Related Topics

check contours_extraprinting

check contours_hardbridging

check contours_notprinting

check mask_empty

check mask_hardbridging

check mask_is_broken

check mask_isG45

check mask_isG90

check mask_withSRAFs

check once

check target_empty

enable_visible_layer

Type: pxopc_options Keywords - general section only

Allows layers of type “visible”. The default is to exit with an error for visible layers.

Usage

enable_visible_layer {**yes** | **no**}

Arguments

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

  A required argument determining whether visible layers are allowed (yes) or not (no, the default).

Description

When this optional keyword is set to yes, Calibre pxOPC setup files may use layers of type visible. Visible layers are not changed, but they are included in the mask for simulation and affect the print image.

The only job in which the mask shapes may touch visible layer shapes is Open; for the Open job all visible layers are ignored. For other jobs, Calibre pxOPC tries to keep the mask from intersecting shapes on the visible layers and to satisfy the mrc_min_external distance between the mask and visible layer shapes.

It is the responsibility of the user to manage any printing that may occur from the visible layer.

- If you do not care whether visible layers print, specify a weight layer that includes everything except the visible layer sized up by some amount.
- If the visible layer should print to a specific contour, modify the target accordingly and supply an initial mask that produces this contour.

Visible layers are the recommended method for handling SRAFs generated outside of Calibre pxOPC and shapes that should print but are not sensitive enough to warrant detailed correction, such as fill.

By default, Calibre pxOPC tries to keep visible layers from printing if there are no nearby target shapes.

Examples

```
enable_visible_layer yes
...
layer M1_fill visible atten 0.06
```

generate_g45

Type: pxopc_options Keywords - Detail job only

Litho Setup File Commands

Enables generation of 45-degree edges in the Detail job output.

Usage

generate_g45 {**yes** | **no**}

Arguments

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

  A required argument controlling the functionality. Specify only **yes** or **no**. By default, it is off (no).

Description

This option enables the generation of 45-degree edges in the output of the Detail job. By default, this option is turned off. It is useful for packed via arrays, in which the corner-to-corner distance is small.

To generate 45-degree edges, it is recommended that the recipe include two consecutive Detail jobs, using the same set of MRC constraints. The second Detail job sets generate_g45 as in the example following.

Examples

This example illustrates the recommended usage for generate_g45. The MRC settings are the same between jobs.

```
...
job 6 detail
   mrc_min_edge 0.020
   mrc_min_internal 0.020
   mrc_min_external 0.020

job 7 detail
   mrc_min_edge 0.020
   mrc_min_internal 0.020
   mrc_min_external 0.020
   generate_g45 yes
```

init

Type: pxopc_options Keywords - general section only

Identifies the initial mask.

Usage

init *layer1* [*layer2*]

Multi-patterning Syntax

init {***layer1*** [*layer2*]}…

Arguments

- ***layer1***

  A required argument identifying a layer to be used for the initial mask (mask0 if using two exposures). The ***layer1*** name should match that of a layer statement within the pxopc_options block. The layer must be of type **correction** or **target**. Names are not case-sensitive.

- *layer2*

  An optional layer that initializes mask1 for a dual exposure mask. The layer must be of type **correction** or **target** and listed in a layer statement in the pxopc_options block.

Description

This optional statement initializes the mask using the specified input layer(s). The ***layer1*** layer is used for the first exposure and the *layer2* layer for the second exposure.

If this command is not used in the setup file, then the starting mask is the correction layer. If the correction layer is not present or is not unique, the starting mask is the target layer.

For multi-patterning setups, this command is required. The number of layers specified should be equal to the number of decomposed layers multiplied by the number of exposures.

Examples

The following example shows an explicitly declared mask layer:

```
LITHO FILE px.setup [ /*
   modelpath models
   ...

   layer px.target hidden  +0.9785 +.0.0085 //these layer statements are
   layer px.visible hidden +0.9785 +.0.0085 //ignored
   ...
```

```
      pxopc_options px.options {
         background +0.0215 -0.0085
         layer px.target correction +0.9785 + 0.0085 //these layers must be
         layer px.smooth target     +0.9785 +.0.0085 //the right type
         ...
         init px.target
         ...
      }
   */ ]
```

In this case, the init command specifies the same layer that would have been used as an initial mask without the command. If the setup file had included another correction layer (for example, px.target2) then the default initial mask would have instead been px.smooth.

iterations

Type: pxopc_options Keywords - any section

Specifies how many iterations the optimization algorithm should run.

Usage

iterations ***number_of_iterations***

Arguments

- ***number_of_iterations***

  A required positive integer that specifies how many optimization passes are performed. The default value depends on the job type and whether the run is controlled by Calibre LPE:

Table 3-6. Default Iterations

Job	Standalone pxOPC	LPE-Driven pxOPC
Open	10	10
Decorate Decoratedarksraf	6	6
Unclose	10	10
Correct	8	16
Refine	6	Not used
Finetune	Not used	10
Detail Detailhybrid Detailrectsraf	10	24
CDOF	100	Not used

The LMRC job does not use an optimization algorithm and iterations has no effect.

Description

This command specifies the iteration count of optimization algorithms. It can be set either in the general section, in which case all jobs run through the same number of optimization passes, or on a per-job basis.

Examples

```
iterations 3
```

Related Topics

job

mrc_iterations

job

Type: pxopc_options Keywords - job section header

Calls the various tasks and allows local overrides of settings.

Usage

job ***number type***

Arguments

- ***number***

  A required integer identifying the job. Jobs are numbered sequentially starting at 1.

- ***type***

  A required job type in lowercase letters. Must be one of the following, listed in the order typically used in a recipe:

  open — Ensures that there is a printing contour in the vicinity of the target so that the main feature is printable. If this step is not executed, assist features are used more to increase feature size than to improve process window. Omitting the Open job may result in weak SRAFs or ones that are placed too close to the main features.

  decorate — Generates the assist features. The placement and sizes of the SRAFs are set to deliver maximum process latitude for the given process corners. This optimization step runs quickly at the cost of some print quality. The print quality is restored by the Correct and Refine jobs.

  decoratedarksraf — Like Decorate, but with default settings tuned for when the absolute value of the background transmission is greater than the absolute value of the correction layer's transmission.

  Note

  The Decoratedarksraf job type is deprecated. Use a Decorate job with metric_impact instead.

  unclose — Restores the contour of main features associated with negative SRAFs if the contours do not print after a Decorate or Decoratedarksraf job.

  The Unclose job is typically not part of a recipe. It should only be used if the Decorate job sets scatter_assist_type to inside or both, and some significant portion of the main features is not printing after a Decorate or Decoratedarksraf job.

  correct — Optimizes the mask for the best EPE RMS and removes extra printing.

  cdof_curvilinear — Optimizes a contoured representation of the mask for the best common depth of focus (CDOF) while suppressing extra printing.

  el_curvilinear — Optimizes a contoured representation of the mask for the best exposure latitude (EL) while maintaining depth of focus and suppressing extra printing. The EL_Curvilinear job can be used to optimize image-log slope (ILS).

refine — Performs fine-grained optimization for the best EPE RMS and removes extra printing. Cannot be used in Calibre LPE runs. See “Refine and Finetune Jobs” on page 155.

finetune — For use with Calibre LPE only. The function of the job is the same as the Refine job but uses different algorithms.

lmrc — Ensures that the final mask is MRC-clean and that SRAFs do not print.

detail — Optimizes over all indicated process conditions for both litho quality and MRC compliance. It centers and shrinks EPE distribution and keeps the mask MRC clean. The output mask has Manhattan main features and SRAFs. Unlike other jobs, process conditions should always be explicitly set with pw_select.

detailhybrid — Similar to the Detail job type, except this job type produces curvilinear main features with Manhattan SRAFs.

detailrectsraf — Similar to the Detail job type, except this job type produces Manhattan main features with rectangular SRAFs.

cdof — Improves the common depth of focus (CDOF) while maintaining litho quality and MRC compliance. The mask is represented with rectilinear polygons.

el — Improves the exposure latitude (EL) while suppressing extra printing and maintaining CDOF and MRC compliance. The mask is represented with rectilinear polygons.

size — Resizes the input layer polygons. This job type does not work with other job types.

A typical recipe does not include all job types, but does typically include Open, Decorate, Correct, Refine or Finetune, LMRC, and Detail.

Description

The job command modularizes the Calibre pxOPC run into discrete units whose output is verifiable. The ***number*** causes each job to be called in the proper sequence; the ***type*** tells the tool which task to run.

Keywords specified between two job commands apply to the first job. That is, first you declare the job and then supply any value overrides. See “Example 2 — Job Options”.

The Refine and Detail jobs both work to ensure quality of results. You may get acceptable output using only one, depending on your process window requirements.

Refine and Finetune Jobs

When Calibre pxOPC is started from within a Calibre LPE setup file, the pxopc_options must use “finetune” instead of “refine”. If refine is used, 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
```

EL, EL_Curvilinear, CDOF, and CDOF_Curvilinear Jobs

These job types require litho models and that the definitions for process windows use the litho model syntax. Because they prioritize the focus and dose rather than EPE, you must also indicate acceptable placement for the simulated contours using pw_tolerance_bound.

If the job list includes CDOF or CDOF_Curvilinear, the litho model must include optical models for the full focus range (±100 nm by default).

If the job list includes EL or EL_Curvilinear, the setup file must define at least two process window condition groups using pw_condition_group. The first should set focus_range, and may set other options. These options are held constant. The second should define the dose latitude parameters. Failure to have two process window groups generates the following error:

```
ERROR: PXOPC: job (null): the number of pw condition groups set in global
section is 1. It can not be less than 2
```

Examples

Example 1 — Minimum pxopc_options Block

The following example code represents a minimum pxopc_options block:

```
pxopc_options SIMULATION_OPTIONS {
   background 0.0215 -0.0085

   layer target CORRECTION 0.9785 0.0085
   layer trgtsmth TARGET 0.9785 0.0085

   pw_condition NOMINAL optical f0 dose 1 aerial 0.21 weight 1.0

   job 1 open
   job 2 decorate
   job 3 correct
   job 4 refine
   job 5 lmrc
}
```

Example 2 — Job Options

This example shows a job overriding the default settings:

```
job 3 correct
pw_select nominal
```

By default, job 3 would simulate all process window conditions. With the pw_select command, it simulates only the nominal condition. Other jobs still simulate all conditions unless similarly constrained.

Related Topics

pw_condition_group

scatter_assist_type

layer

Type: pxopc_options Keywords - general section only

Defines layer characteristics such as name and type.

Usage

Standard Format:

layer *name mtype mtrans* [mask *mask_num*]

Standard Multi-patterning Format:

layer *name mtype mtrans* [mask *mask_num*] [pattern *index*]

Litho Model Format:

layer *name mtype* [*mask*] [mask_layer *lm_num*]

Multi-patterning Litho Model Format:

layer *name mtype* [*mask*] [mask_layer *lm_num*] [pattern *index*]

Arguments

- ***name***

  A required argument specifying the name of the layer. Layer names must be alphanumeric strings less than 32 characters in length. The ***name*** is used in the rest of the pxopc_options block to refer to the layer.

- ***mtype***

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

  target — The optimal layout. The output contours from simulation are compared to this layer and the difference minimized.

  correction — An initial mask.

  hidden — Weight layers. Layers of this type are neither changed nor simulated in pxOPC runs.

  visible — Fill layers. Layers of this type are not changed, but are included in simulations. The setup file must include enable_visible_layer yes.

  Layers referenced by the init keyword must have an ***mtype*** of target or correction and, if using a DDM model, the polygons should only have vertical or horizontal edges.

- ***mtrans***

  A required argument for the standard syntax specifying the transmission value. The mask transmission is defined layer by layer, with the sum of all layers added to the background transmission to generate the ideal mask transmission. Layers which have the same transmission are automatically merged if they overlap; hence overlapping layers will not affect simulation. Layers with different transmissions are not merged, hence their

transmissions will add if they overlap. The transmission values are ignored for hidden layer types. Legal values for this argument are the following:

dark — The transmission added to the mask is {0 - background}.

clear — The transmission added to the mask is {1.0 - background}.

atten *value* — The transmission added to the mask is {1 + value - background}.

real imag —The transmission added to the mask is {real, imag}.

For more information on how transmission is calculated, see the discussion in the arguments for "background".

- mask *mask_num*

  An optional argument indicating if there are multiple exposures. For multiple exposures, the mask number must be entered separately for layers on each exposure.

  0 — The default value. Indicates the simulation and mask are single exposure.

  1 — The simulation is multiple exposure.

- pattern *index*

  An optional argument for multi-patterning setup files that is used in the standard syntax only. This argument defines the decomposed mask to which the layer belongs.

  The *index* values are 0, 1, 2, or 3 and should be defined in order. That is, if a layer has "pattern 1", there must also be a layer with "pattern 0".

- mask_layer *lm_num*

  An optional argument used in the litho model syntax only. Although optional, it is recommended. This argument maps the layer to the mask block in the litho model. The default is 0.

Description

Names and describes the optical properties of the input layer(s). Every layer used in the pxOPC run must be declared in the pxopc_options block using this command.

Note

The Calibre pxOPC algorithm only uses transmission values specified by this keyword. Layer transmission values set outside of the pxopc_options block are ignored. If a ddm model is used, either by explicit call or as part of a litho model, the ddm transmission overrides any layer-specified transmission.

Notice that the allowed arguments for this placement of the layer command are different from the valid values for layer (for main setup file).

Note

A setup file must either use the litho model syntax for all commands, or not at all. Standard syntax and litho model syntax cannot be mixed.

Examples

Example 1 — Standard Format

```
LITHO FILE litho.in [
   modelpath models

   optical_model_load f0 opticalf0

   background dark
   layer target hidden clear   # These layer values are IGNORED
   layer trgtsmth hidden clear # by Calibre pxOPC.

   pxopc_options SIMULATION_OPTIONS {
      background 0.0215 -0.0085
      layer target CORRECTION 0.9785 0.0085 # These values are USED
      layer trgtsmth TARGET 0.9785 0.0085   # by Calibre pxOPC.

      pw_condition NOMINAL optical f0 dose 1 aerial 0.21 weight 1.0

   job 1 open
   job 2 decorate
   job 3 correct
   job 4 refine
   job 5 lmrc
   job 6 detail
   }

setlayer opcout = pxopc target trgtsmth MAP target \
                                  OPTIONS SIMULATION_OPTIONS
]
```

Example 2 — Litho Model Format

```
LITHO FILE litho.in [
   modelpath models

   layer target hidden clear   # These layer values are IGNORED
   layer trgtsmth hidden clear # by Calibre pxOPC.

   pxopc_options SIMULATION_OPTIONS {
      layer target CORRECTION # These values are USED
      layer trgtsmth TARGET   # by Calibre pxOPC.

      pw_condition NOMINAL lm_model dose 1 aerial 0.21 weight 1.0

   job 1 open
   job 2 decorate
   job 3 correct
   job 4 refine
   job 5 lmrc
   job 6 detail
   }

setlayer opcout = pxopc target trgtsmth MAP target \
                                  OPTIONS SIMULATION_OPTIONS
]
```

lmrc_inside_assist_cuts_enabled

Type: pxopc_options Keywords - LMRC jobs only

Enables extra print checking for SRAFs cut into polygons.

Usage

lmrc_inside_assist_cuts_enabled {**yes** | **no**}

Arguments

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

  A required argument controlling the functionality. Specify only **yes** or **no**. By default, it is off (no).

Description

An optional keyword for use in LMRC jobs. It improves SRAF printing suppression when the layout uses negative SRAFs.

Related Topics

job

mask_periodicity

Type: pxopc_options Keywords - general section only

Forces the output to repeat patterns periodically.

Usage

mask_periodicity yes | no

Arguments

- yes | no

  An optional argument enabling mask periodicity enforcement. The default behavior is no periodicity enforcement.

Description

An optional parameter specifying periodicity in the mask output. The periodicity in the X and Y directions is detected from the target layer (usually a curved target). To apply mask periodicity, the repeating pattern must be replicated at least twice in at least one direction. Extract the desired layout window to maximize replication of the repeating pattern.

Correct, Refine, and Finetune jobs support mask periodicity enforcement for curvilinear mask output.

Note

In the current release, LMRC and Detail jobs do not support mask periodicity enforcement for rectilinear mask output.

Examples

The following setting enables mask periodicity:

```
mask_periodicity yes
```

mask_symmetry

Type: pxopc_options Keywords — general section only

Forces the output to the specified symmetry. Should only be used with symmetric optical models and input layers.

Usage

mask_symmetry ***type***

[mask_symmetry_tolerance *dbus*]

Arguments

- ***type***

  A required argument indicating the type of symmetry to enforce.

  x — Symmetric across the horizontal axis.

  y — Symmetric across the vertical axis.

  xy — Symmetric across both axes.

  detect — Automatically determine appropriate symmetry on a per-tile basis.

- mask_symmetry_tolerance *dbus*

  An optional separate command to allow a slight amount of non-symmetry in the input layout. This can be useful when curved shapes might be snapped to the grid.

  The default is no tolerance, 0 dbus.

  This command can only be specified in the general section. All jobs must have the same tolerance.

Description

An optional argument that forces the Calibre pxOPC output to follow the selected symmetry type. Symmetrical output is guaranteed for Correct, Refine, and Finetune jobs.

Correct, Refine, and Finetune jobs support mask symmetry enforcement for curvilinear mask output.

Note

In the current release, LMRC and Detail jobs do not support mask symmetry enforcement for rectilinear mask output.

The typical uses for this command are memory arrays with adjacent symmetric patterns and well-separated clips typical of source-mask optimization and model calibration.

There are two limitations on the use of this setting:

- All shapes should lie completely within a tile. You may need to adjust tilemicrons.

- When you select a symmetry type, all input layers (including the weight and visible layers) must adhere to that symmetry type. The optical model should also be symmetric in the same way. If there are any parts that do not display the specified symmetry, the run exits with an error message.

```
pxopc_exec: target layer does not match the selected symmetry type
```

The “detect” setting is used primarily for memory arrays and well-separated clips that may each have their own symmetry type, including no symmetry at all. The output (assuming the optical model is symmetric) is a mix of regular Calibre pxOPC output and symmetric masks.

By default, symmetry is not enforced.

Examples

This setting indicates all input is symmetric in the X axis:

```
mask_symmetry x
```

Because no tolerance is specified, any grid snapping is likely to cause the run to exit with an error message.

metric_impact

Type: pxopc_options Keywords - job sections only

Customizes the job objective.

Usage

metric_impact {**slope** | **epe** | **dof**} **factor** ***value***
[pw *cond* [*cond*…] [pw_nominal *cond*] [pattern *index*]]

Arguments

- **slope** | **epe** | **dof**

  A required argument that specifies to which quality the settings apply. Specify only one of the following keywords per statement:

  slope — The image slope, defined as the derivative of image intensity over location.

  epe — Edge placement error, defined as the distance between the desired placement and the simulated contour.

  dof — Depth of focus, defined as the range of focus values for which the image still resolves well.

  Typically when you specify metric_impact in a job there is a metric_impact specification for each of these keywords.

- **factor** ***value***

  A required argument specifying the relative weight to give the metric. Values range from 0 to 1.

  Caution

  If only one metric_impact statement is used, do not set the factor value to 0. If more than one metric_impact statement is used, the metric with a factor value of zero is not used during optimization.

- pw *cond* [*cond*…] [pw_nominal *cond*] [pattern *index*]

  An optional set of arguments restricting metric_impact to the named process window conditions only. If pw is not included, all process window conditions for the job are used. The pw keyword must be followed by at least one *cond*, which must also be listed in a pw_select statement for the job. Process window condition groups are not accepted.

  pw_nominal *cond* — An optional argument that can only be used with **dof**. Specifies the process window condition corresponding to a defocus of 0. Defaults to the condition named “nominal.”

  pattern *index* — An optional argument that restricts metric_impact to the indicated multi-patterning set. The layer statements must also use pattern. If this argument is not specified, all patterns are used.

Description

This optional command allows you to create your own custom objectives for a job. If any metric_impact statement appears in a job section, that job's default objectives are ignored and only the metric_impact statement(s) are used for optimization.

This statement can be specified multiple times per job section. Results may change depending on the order of the statements. For example,

```
metric_impact epe factor 0.5 pw nominal
metric_impact dof factor 0.5 pw positive negative
metric_impact slope factor 0.1 pw nominal positive negative
```

produces slightly different output than this order:

```
metric_impact slope factor 0.1 pw nominal positive negative
metric_impact dof factor 0.5 pw positive negative
metric_impact epe factor 0.5 pw nominal
```

You can set different weights for different process window conditions using separate statements:

```
metric_impact epe factor 0.8 pw bridge_condition
metric_impact epe factor 0.2 pw pinch_condition
```

The error

```
ERROR: pxopcFrameProblem::pxopcFrameProblem: Zero number of pw conditions
was passed to frame problem.
```

indicates that the job has only process window condition groups selected. Add at least one process window condition to the pw_select statement for the job.

Examples

Example 1 — Simple Usage

This shows a minimal recipe:

```
version 1

layer tgt        correction
layer tgt.smooth target

pw_condition nominal    lmod
pw_condition inner      focus  50nm dose 1.025
pw_condition outer      focus   0nm dose 0.975
...
```

```
...
job 2 decorate
metric_impact epe factor 0.5
metric_impact dof factor 0.5
metric_impact slope factor 0.1
...
```

These settings would cause the second job to use all defined process window conditions when calculating EPE, DOF, and image slope because there is no pw_select statement for the job or pw option for the metric_impact keywords. The job would give EPE and DOF optimization five times the importance of image slope.

Example 2 — Typical Usage

This shows a more typical usage in that it selects particular process window conditions for the metrics to improve performance.

```
version 1

layer tgt        correction
layer tgt.smooth target

pw_condition nominal    lmod
pw_condition inner      focus  50nm dose 1.025
pw_condition outer      focus   0nm dose 0.975
pw_condition positive   focus  50nm dose 1
pw_condition negative   focus -50nm dose 1

...
job 2 decorate
metric_impact epe factor 0.5 pw nominal
metric_impact dof factor 0.5 pw positive negative
metric_impact slope factor 0.1 pw nominal positive negative
...
```

mrc_align_edges

Type: pxopc_options Keywords - any section

Specifies what edges to attempt to align.

Usage

mrc_align_edges {**none** | **sraf** | **main** | **all**}

Arguments

- **none** | **sraf** | **main** | **all**

  A required argument specifying which type of mask edges to attempt to align.

  none — Do not attempt to align edges. This is the default.

  sraf — Attempt to align only edges on SRAF features.

  main — Attempt to align only edges on intentionally printed features.

  all — Attempt to align edges for all features.

Description

This optional keyword instructs the MRC system to align adjacent mask edges, provided it does not violate MRC constraints. The reason to align edges is to reduce shot count.

Examples

```
job 5 lmrc
mrc_align_edges all
```

mrc_enabled

Type: pxopc_options Keywords - any section

Activates MRC constraints for curvilinear output.

Usage

mrc_enabled [no | yes]

Arguments

- { no | yes }

  An optional argument specifying activation of MRC constraints for curvilinear output.

Description

An optional argument specifying activation of MRC constraints for curvilinear output in Correct or Refine jobs.

Examples

Example 1

The following example activates MRC constraints for curvilinear output and permits curvilinear shapes as close as 12 nanometers and as narrow as 12 nanometers wide.

```
job 3 correct
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
```

mrc_iterations

Type: pxopc_options Keywords - any section

Specifies how many iterations the MRC algorithm should run.

Usage

mrc_iterations ***iteration_number***

Arguments

- ***iteration_number***

  A required positive integer specifying how many iterations the MRC algorithm makes. The default is 30 for standard runs, and 0 when invoked from Calibre LPE.

Description

An optional argument that specifies the number of main MRC iterations to perform. It only has an effect in the LMRC job.

If after running LMRC the process window is significantly worse, increase mrc_iterations. A good step size is 10. (That is, the first pass would increase the default of 30 to 40 iterations; the next would set mrc_iterations to 50.)

Examples

```
job 5 lmrc
mrc_iterations 40
```

Related Topics

iterations

mrc_max_notch_height

Type: pxopc_options Keywords - any section

Sets the maximum height allowed for a notch before it is considered to be an MRC violation.

Usage

mrc_max_notch_height ***microns***

Arguments

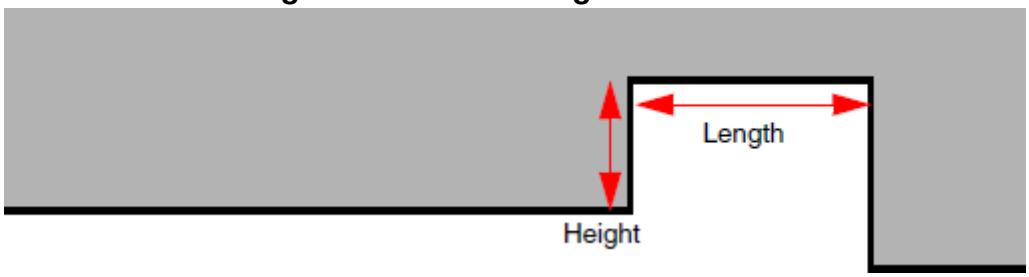
- ***microns***

  A required argument that, in conjunction with mrc_min_notch_length, specifies the largest allowable notch in microns. The height must be a positive value. The default is 0.

Description

Use this command to restrict minimum-length edges from having significant displacement from their neighbors. Only “notches” that are either longer than or equal to mrc_min_notch_length (by default, mrc_min_internal) and shallower than mrc_max_notch_height (by default, 0) pass MRC checks. Notches with shorter length or greater depth are adjusted away if possible.

Figure 3-4. Notch Height Definition



This keyword only changes LMRC and Detail job results. If you do not specify mrc_max_notch_height, mrc_min_notch_length has no effect.

Related Topics

mrc_min_notch_length

mrc_max_nub_height

mrc_max_nub_height

Type: pxopc_options Keywords - any section

Sets the maximum height allowable for a nub before it is considered an MRC violation.

Usage

mrc_max_nub_height ***microns***

Arguments

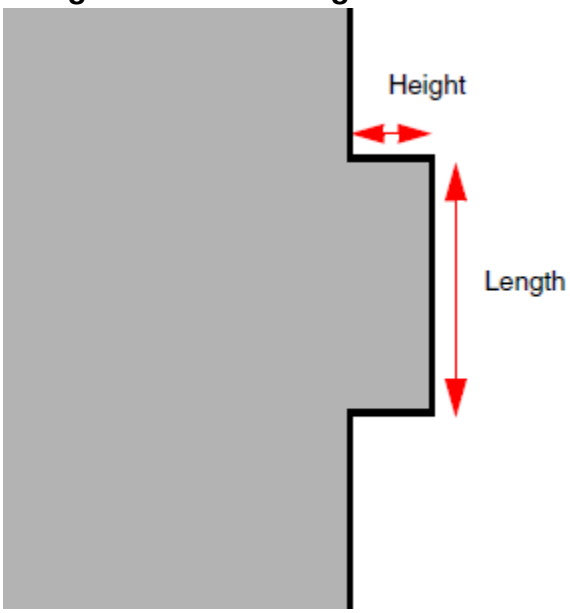
- ***microns***

  A required argument that, in conjunction with mrc_min_nub_length, specifies the largest allowable nub in microns. The height must be a positive value. The default is 0.

Description

Use this command to restrict minimum-length edges from having significant displacement from their neighbors. Only “nubs” that are either longer than or equal to mrc_min_nub_length (by default, mrc_min_external) and shallower than mrc_max_nub_height (by default, 0) pass MRC checks. Nubs with shorter length or greater displacement are adjusted away if possible.

Figure 3-5. Nub Height Definition



This keyword only changes LMRC and Detail job results. If you do not set mrc_max_nub_height, mrc_min_nub_length has no effect.

Related Topics

mrc_min_nub_length

mrc_max_notch_height

mrc_min_area

Type: pxopc_options Keywords - any section

Constrains the minimum SRAF area during MRC.

Usage

mrc_min_area ***min_sraf_area_val***

Arguments

- ***min_sraf_area_val***

  A required argument specifying a minimum area for SRAFs in square microns during MRC. The default value is 0.

Description

An optional command that specifies the minimum SRAF size during Lmrc jobs for Manhattan shapes. For curvilinear SRAFs, use the command in both the Correct and Refine jobs, and specify "mrc_enabled yes" in each job to activate curvilinear MRC. For SRAFs that violate the minimum area check, the command increases their size to meet the constraint.

Examples

Example 1— Manhattan MRC

The following example ensures Manhattan SRAFs larger than 0.0015 square microns:

```
job 5 lmrc
   mrc_min_area 0.0015
   ...
```

Example 2— Curvilinear MRC

The following example ensures curvilinear SRAFs larger than 0.0015 square microns:

```
job 3 correct
   mrc_enabled yes
   mrc_min_area 0.0015
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_area 0.0015
   ...
```

mrc_min_assist_edge

Type: pxopc_options Keywords - any section

Sets a target SRAF length for MRC geometries.

Usage

mrc_min_assist_edge ***min_assist_edge_val***

Arguments

- ***min_assist_edge_val***

  A required argument specifying the preferred length in microns of SRAF edges for MRC-simplified geometries. The default is 0.015 um.

Description

This command sets the preferred SRAF length for edges once contours are converted to shapes with fragmented Manhattan edges. This command has no effect in the early contour-driven stages.

Notice that the length is preferred; this means some edges may be smaller or larger if required.

If using LMRC or Detail jobs, set the same value for mrc_min_assist_edge for both. A difference in values can cause SRAFs to print.

Examples

```
# set the mrc_min_assist_edge command where it matters:
job 5 lmrc
mrc_min_assist_edge 0.015
```

mrc_min_edge

Type: pxopc_options Keywords - any section

Sets a target edge length for MRC geometries.

Usage

mrc_min_edge ***min_edge_val***

Arguments

- ***min_edge_val***

  A required argument specifying the preferred length in microns of the edges for MRC-simplified geometries. The default is 0.015 um.

Description

Sets a preferred length for edges once contours are converted to shapes with fragmented Manhattan edges. This command has no effect in the early contour-driven stages.

Notice that the length is *preferred*; this means some edges may be smaller or larger if required.

If using LMRC and Detail jobs, set the same value for mrc_min_edge for both. A difference in values can cause SRAFs to print.

Examples

```
# set the mrc_min_edge command where it matters:
job 5 lmrc
mrc_min_edge 0.015
```

Related Topics

job

mrc_min_external

Type: pxopc_options Keywords - any section

Constrains external spacing during MRC.

Usage

mrc_min_external ***min_external_val***

Arguments

- ***min_external_val***

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

Description

An optional command that specifies the distance between two exterior edges. For Manhattan shapes, use the command in Lmrc jobs. For curvilinear shapes, use the command in both the Correct and Refine jobs, and specify "mrc_enabled yes" in each job to activate curvilinear MRC. The edges can occur on the same polygon, such as a notch.

Examples

Example 1 — Manhattan MRC

In the following example, the mrc_min_external statement permits Manhattan shapes as close as 0.012 microns; the mrc_min_internal statement permits Manhattan features 0.012 microns and wider:

```
job 5 lmrc
   mrc_min_external 0.012 # permits shapes closer together than the
                          # default
   mrc_min_internal 0.012 # also permits narrower minimum feature
```

Example 2 — Curvilinear MRC

In the following example, the mrc_min_external statement permits curvilinear shapes as close as 0.012 microns; the mrc_min_internal statement permits curvilinear features 0.012 microns and wider:

```
job 3 correct
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
   ...
```

The mrc_enabled statement is required to activate curvilinear MRC.

Related Topics

mrc_min_internal

job

mrc_min_external_main2sraf

Type: pxopc_options Keywords - any section

Constrains external spacing during MRC between a main feature and an SRAF.

Usage

mrc_min_external_main2sraf ***min_external_val***

Arguments

- ***min_external_val***

  A required argument specifying a minimum distance in microns between the exterior sides of edges during MRC. The default is the maximum of mrc_min_external and mrc_min_external_sraf.

Description

An optional command that specifies the exterior distance between main features and SRAF edges during MRC. For Manhattan shapes, use the command in Lmrc and Detail jobs. For curvilinear shapes, use the command in both the Correct and Refine jobs, and specify "mrc_enabled yes" in each job to activate curvilinear MRC. If mrc_min_external_main2sraf is not set, the spacing between main features and SRAFs is checked against the larger of mrc_min_external and mrc_min_external_sraf.

Examples

Example 1 — Manhattan MRC

The following example permits 14 nanometers between Manhattan shapes and 14 nanometers between SRAFs; the main-to-SRAF distance is set to 15 nanometers:

```
job 5 lmrc
   mrc_min_external 0.014 # permits shapes closer together than the
                          # default
   mrc_min_external_sraf 0.014 # also permits closer SRAFs
   mrc_min_external_main2sraf 0.015 # sets a larger main-to-SRAF distance
```

Example 2— Curvilinear MRC

The following example permits 14 nanometers between curvilinear shapes and 14 nanometers between SRAFs; the main-to-SRAF distance is set to 15 nanometers:

```
job 3 correct
   mrc_enabled yes
   mrc_min_external 0.014
   mrc_min_external_sraf 0.014
   mrc_min_external_main2sraf 0.015
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_external 0.014
   mrc_min_external_sraf 0.014
   mrc_min_external_main2sraf 0.015
   ...
```

mrc_min_external_main2visible

Type: pxopc_options Keywords - any section

Constrains external spacing during MRC between a main feature and a feature on the visible layer only.

Usage

mrc_min_external_main2visible *min_external_val*

Arguments

- ***min_external_val***

  A required argument specifying a minimum distance in microns between the exterior sides of edges during MRC. The default is the maximum of mrc_min_external and mrc_min_external_sraf.

Description

An optional command that specifies the exterior distance between the edges of a main feature and features on a visible layer during Lmrc jobs. The offset between main and visible layer features is checked against the larger of mrc_min_external and mrc_min_external_sraf.

mrc_min_external_sraf

Type: pxopc_options Keywords - any section

Constrains external spacing during MRC for SRAFs only.

Usage

mrc_min_external_sraf *min_external_val*

Arguments

- ***min_external_val***

  A required argument specifying a minimum distance in microns between the exterior sides of edges during MRC. The default is the maximum value used for mrc_min_external and mrc_min_external_sraf.

Description

An optional command that specifies the exterior distance between SRAF edges during MRC. For Manhattan shapes, use the command in Lmrc jobs. For curvilinear shapes, use the command in both the Correct and Refine jobs, and specify "mrc_enabled yes" in each job to activate curvilinear MRC.

This command can be used to set different MRC limits for the distance between the main features and SRAFs. If mrc_min_external_main2sraf is not set, then the offset between main features and SRAFs is checked against the larger of mrc_min_external and mrc_min_external_sraf.

mrc_min_external_sraf2visible

Type: pxopc_options Keywords - any section

Constrains external spacing during MRC between an SRAF and a feature on a visible layer only.

Usage

mrc_min_external_sraf2visible *min_external_val*

Arguments

- ***min_external_val***

  A required argument specifying a minimum distance in microns between the edge of an SRAF and a feature on a visible layer during MRC. The default is the maximum of mrc_min_external and mrc_min_external_sraf.

Description

An optional command that specifies the exterior distance between the edges of an SRAF and features on a visible layer during Lmrc jobs. The offset between main and visible layer features is checked against the larger of mrc_min_external and mrc_min_external_sraf.

mrc_min_internal

Type: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

Usage

mrc_min_internal ***min_internal_val***

Arguments

- ***min_internal_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 during MRC. For Manhattan shapes, use the command in Lmrc jobs. For curvilinear shapes, use the command in both the Correct and Refine jobs, and specify "mrc_enabled yes" in each job to activate curvilinear MRC. Use this keyword for controlling the size of main features. Use mrc_min_rect_width or mrc_small_feature_size for controlling the size of SRAFs.

Examples

Example 1 — Manhattan MRC

In the following example, the mrc_min_internal statement permits Manhattan features 0.012 microns and wider; the mrc_min_external statement permits Manhattan shapes as close as 0.012 microns:

```
job 5 lmrc
mrc_min_external 0.012 # permits shapes closer together than the default
mrc_min_internal 0.012 # also permits narrower minimum feature
```

Example 2 — Curvilinear MRC

In the following example, the mrc_min_internal statement permits curvilinear features 0.012 microns and wider; the mrc_min_external statement permits curvilinear shapes as close as 0.012 microns:

```
job 3 correct
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_external 0.012
   mrc_min_internal 0.012
   ...
```

The mrc_enabled statement is required to activate curvilinear MRC.

Related Topics

job

mrc_min_external

mrc_min_nub_length

mrc_min_internal_sraf

Type: pxopc_options Keywords - any section

Constrains internal spacing during MRC for SRAFs only.

Usage

mrc_min_internal_sraf *min_internal_val*

Arguments

- ***min_internal_val***

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

Description

This optional command allows setting the internal MRC constraint for SRAFs explicitly. This may be simpler to use than the other options for applying internal MRC checks to SRAFs:

- mrc_small_feature_size
- A combination of any of mrc_min_rect_length, mrc_min_rect_width, mrc_min_square, and mrc_min_length
- A combination of mrc_min_small_feature_area_excess and mrc_min_edge

For Manhattan SRAF shapes, specify the command in LMRC and detail jobs. For curvilinear SRAFs, use the command in both the correct and refine jobs, and specify "mrc_enabled yes" in each job to activate MRC constraints for curvilinear output.

Examples

Example 1 - Manhattan SRAFs

This combination allows Manhattan SRAF shapes to be narrower than main features:

```
job 5 lmrc
   mrc_min_internal      0.02
   mrc_min_internal_sraf 0.015
```

Example 2 - Curvilinear SRAFs

This combination allows curvilinear SRAFs to be narrower than main features:

```
job 3 correct
   mrc_enabled yes
   mrc_min_internal      0.02
   mrc_min_internal_sraf 0.015
   ...
job 4 refine
   mrc_enabled yes
   mrc_min_internal      0.02
   mrc_min_internal_sraf 0.015
   ...
```

The mrc_enabled statement is required to activate curvilinear MRC.

mrc_min_length

Type: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

Usage

mrc_min_length ***min_length_val***

Arguments

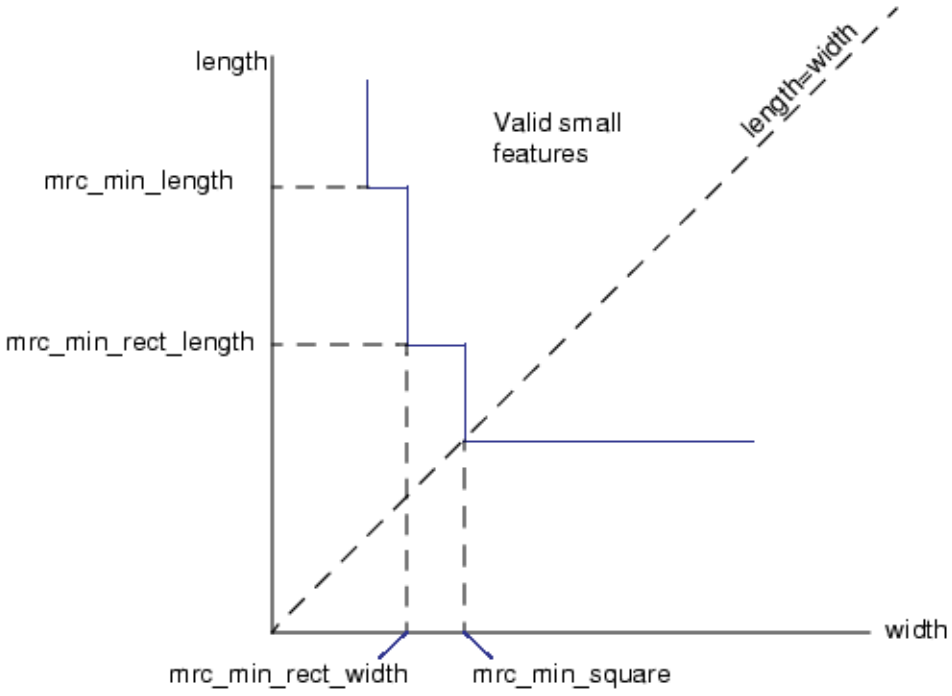
- ***min_length_val***

  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 of edges are chosen arbitrarily.

Description

An optional command that specifies the distance between two internal edges of a shape during LMRC jobs. Use this control for secondary features.

Figure 3-6. Valid Small Features for MRC - mrc_min_length



Related Topics

mrc_min_rect_length

mrc_min_rect_width

mrc_min_square

mrc_min_notch_length

mrc_min_nub_length

mrc_small_feature_size

mrc_min_notch_length

Type: pxopc_options Keywords - any section

Sets the minimum length allowed for a notch before it is considered to be an MRC violation.

Usage

mrc_min_notch_length ***microns***

Arguments

- ***microns***

  A required argument that, in conjunction with mrc_max_notch_height, specifies the largest allowable notch in microns. The default is mrc_min_internal.

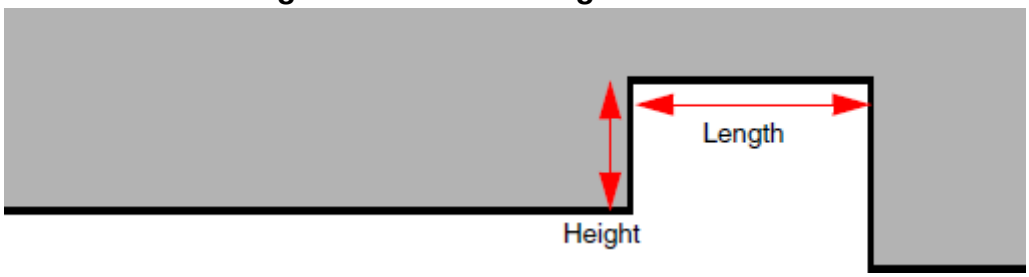
Description

Use this command to restrict minimum-length edges from having significant displacement from their neighbors. Only “notches” that are either longer than or equal to mrc_min_notch_length (by default, mrc_min_internal) and shallower than mrc_max_notch_height (by default, 0) pass MRC checks. Notches with shorter length or greater depth are adjusted away if possible.

Note

If mrc_max_notch_height is not set to a positive value, this keyword has no effect.

Figure 3-7. Notch Length Definition



This keyword only changes LMRC and Detail job results.

Related Topics

mrc_max_notch_height

mrc_min_nub_length

mrc_min_nub_length

Type: pxopc_options Keywords - any section

Sets the minimum length allowed for a nub before it is considered to be an MRC violation.

Usage

mrc_min_nub_length ***microns***

Arguments

- ***microns***

  A required argument that, in conjunction with mrc_max_nub_height, specifies the largest allowable nub in microns. The default is mrc_min_external.

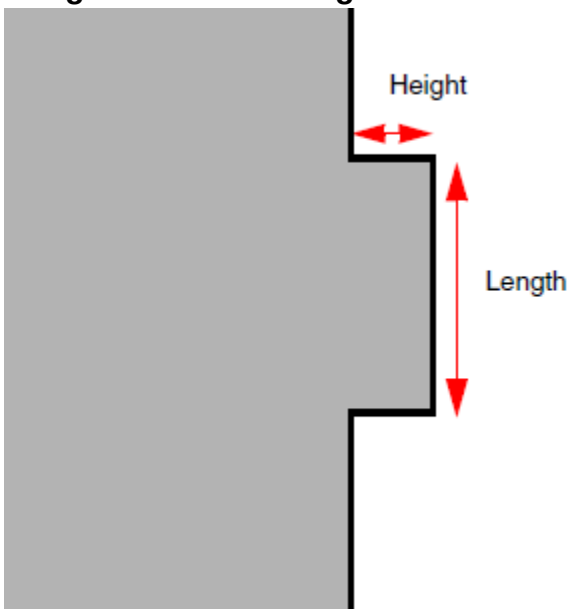
Description

Use this command to restrict minimum-length edges from having significant displacement from their neighbors. Only “nubs” that are either longer than or equal to mrc_min_nub_length (by default, mrc_min_external) and shallower than mrc_max_nub_height (by default, 0) pass MRC checks. Nubs with shorter length or greater displacement are adjusted away if possible.

Note

If mrc_max_nub_height is not set to a positive value, this keyword has no effect.

Figure 3-8. Nub Length Definition



This keyword only changes LMRC and Detail job results.

Related Topics

mrc_max_nub_height

mrc_min_notch_length

mrc_min_rect_length

Type: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

Usage

mrc_min_rect_length ***min_val***

Arguments

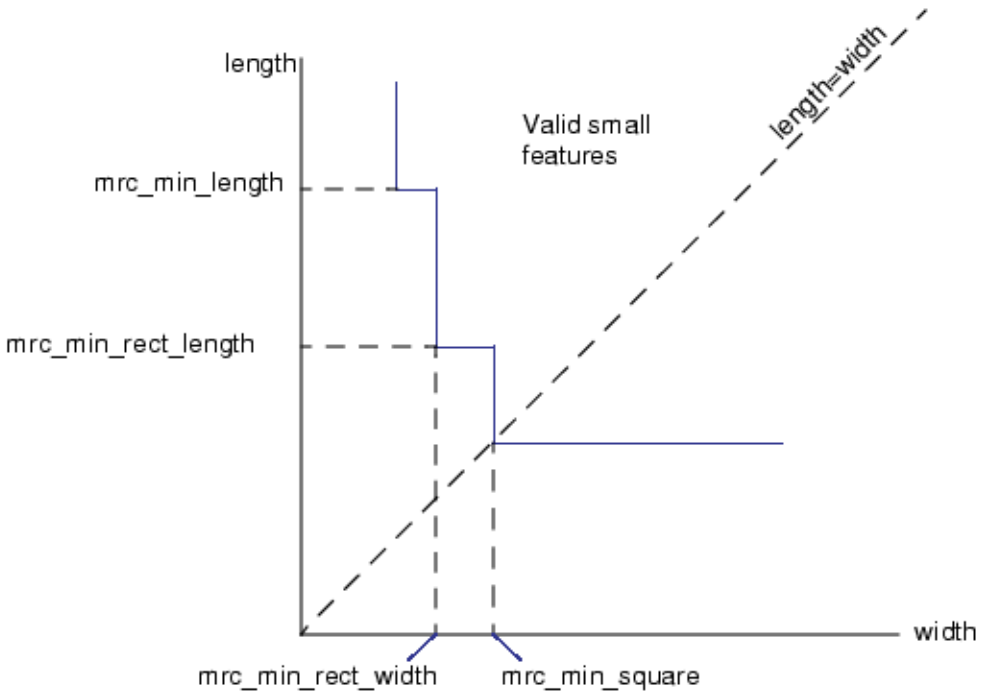
- ***min_val***

  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 during LMRC jobs. Use in conjunction with mrc_min_rect_width for controlling secondary features such as SRAFs.

Figure 3-9. Valid Small Features for MRC - mrc_min_rect_length



Related Topics

mrc_min_length

mrc_min_rect_width

mrc_min_square

mrc_small_feature_size

mrc_min_rect_width

Type: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

Usage

mrc_min_rect_width ***min_val***

Arguments

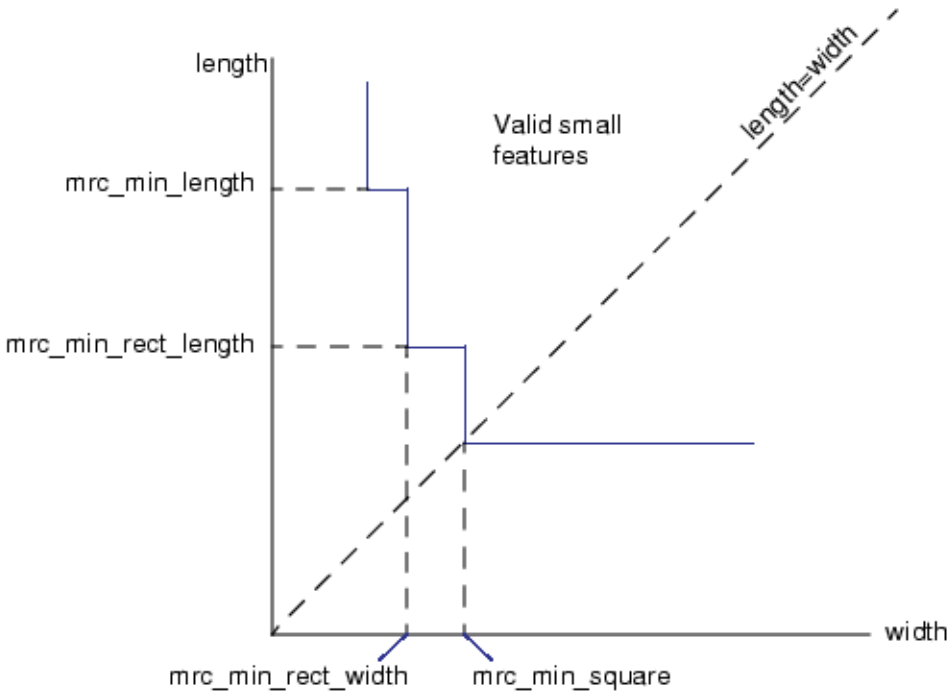
- ***min_val***

  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 during LMRC jobs. Use in conjunction with mrc_min_rect_length to control small secondary features such as SRAFs.

Figure 3-10. 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: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

Usage

mrc_min_square ***min_square_val***

Arguments

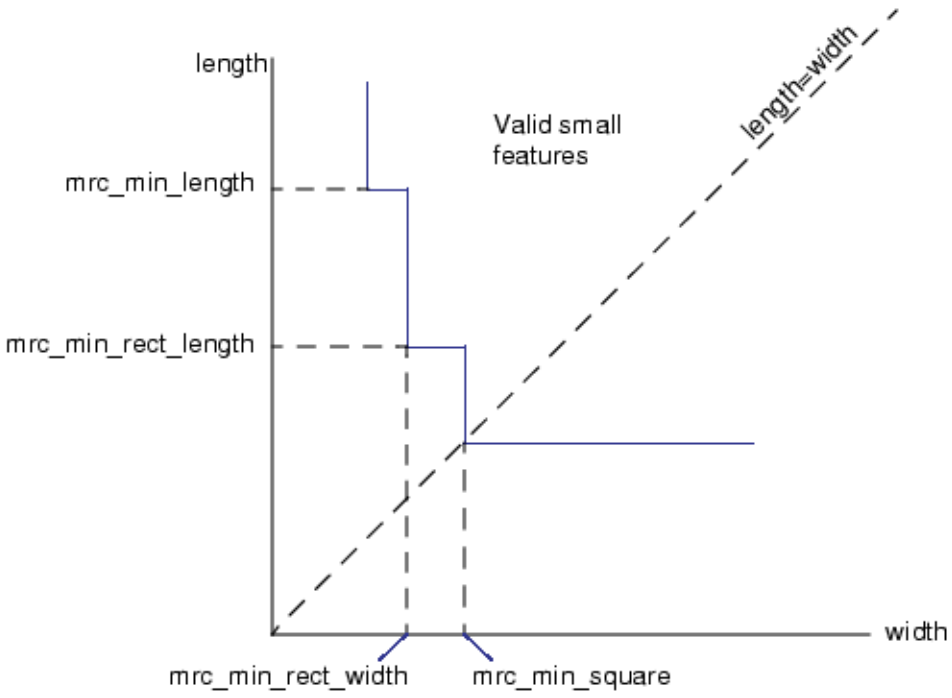
- ***min_square_val***

  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 during LMRC jobs. 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 mrc_min_internal.

Figure 3-11. 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_min_small_feature_area_excess

Type: pxopc_options Keywords - any section

Sets the minimum area for features on the mask.

Usage

mrc_min_small_feature_area_excess *min_val*

Arguments

- ***min_val***

  A required argument specifying a value in microns.

  Features that are smaller than the minimum area are either deleted so they do not appear in the output mask, or increased in size until larger than the minimum area.

  0 <= ***min_val*** < 3 * mrc_min_edge.

Description

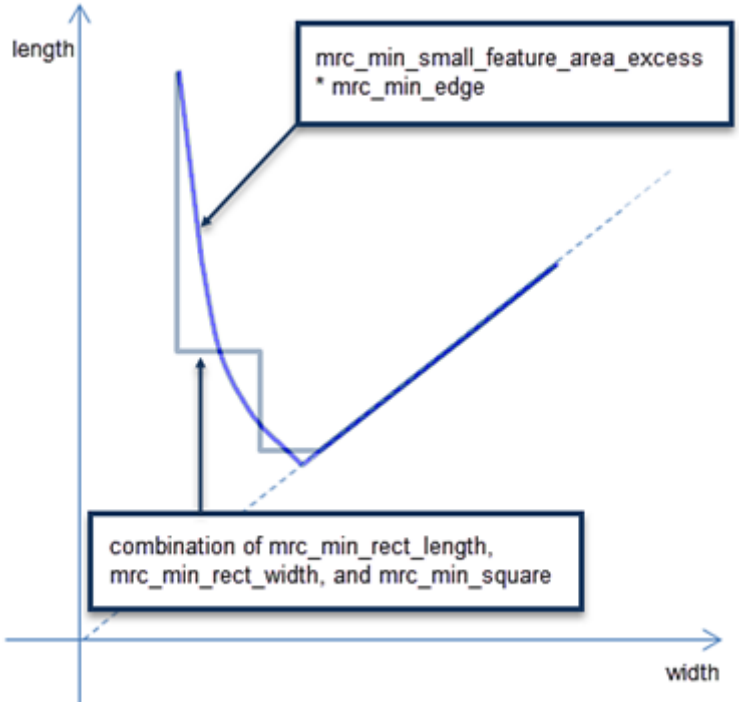
An optional command that specifies the minimum area for features on a mask. This can be useful for preventing undersized SRAFs.

The minimum area is determined by ***min_val*** * mrc_min_edge. This does not restrict either edge's length; only the SRAF area is constrained. The default is to not have a minimum size.

The benefit of mrc_min_small_feature_area_excess is that unlike mrc_min_square, the edges are not the same size. The mrc_min_small_feature_area_excess keyword also scales with mrc_min_edge, unlike the combination of mrc_min_rect_width and mrc_min_rect_length.

The following figure contrasts the different threshold conditions for valid small features using the two methods.

Figure 3-12. Comparison of SRAF MRC Methods



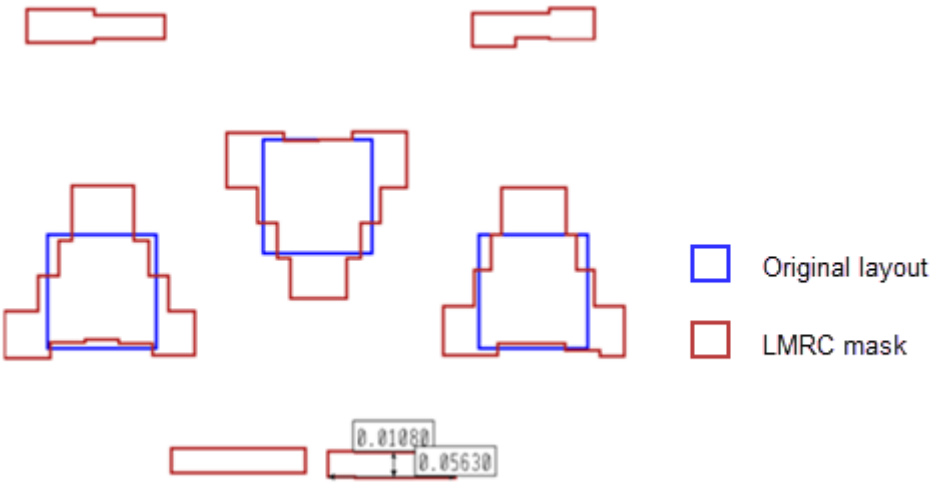
Examples

Example 1 — Manhattan MRC

The figure following was produced from a rule file that used the settings shown, which gave a minimum SRAF area of 0.0006 um^2:

```
job 6 lmrc
  mrc_min_edge                        0.015
  mrc_min_small_feature_area_excess 0.040
```

Figure 3-13. Minimum Area SRAFs



In the mask output from LMRC (red), there are SRAFs with edges 0.0108 um. This is smaller than the minimum edge (0.015) but the area is 0.0108 * 0.0563. The product, 0.00060804, is greater than the minimum SRAF area of 0.0006, so the SRAF is not removed.

Related Topics

mrc_min_edge

mrc_small_feature_size

Type: pxopc_options Keywords - any section

Sets the minimum distance between two internal edges.

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 during LMRC jobs. Features smaller than this size are not processed; they disappear from the output. The default value is 0.015 microns.

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 refine.out = pxopc in_layer... MAP layer \
                            OPTIONS px.options LASTJOB number
  ```

  The LASTJOB number should be the same as that of the Finetune or Refine command.

  Examine the SRAF sizes in the refine.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

objective_region

Type: pxopc_options Keywords - job section only

Optimizes a region using specified objectives.

Usage

objective_region {***region_layer*** | **backdrop**}
{{**set** | **add**} {***objective obj_factor***} [{*objective obj_factor*}...]}
[pw *cond_name* [*cond_name*...]] [pattern *index*]

Arguments

- ***region_layer***

  A required argument specifying the input layer to which the objectives are applied. Either ***region_layer*** or **backdrop** must be specified.

- **backdrop**

  A required argument that applies the specified objective to the entire layout except for regions of the layer where the objectives are specified in other objective_region statements using the **set** argument. Either ***region_layer*** or **backdrop** must be specified.

  When used with the **set** argument, the specified objective replaces the existing objective. When used with the **add** argument, the specified objective is applied in addition to the pre-existing objective.

- **set** | **add** ***objective obj_factor***

  A required argument that optimizes ***region_layer*** using only a specified objective or set of objectives.

  - **set** — Optimizes the ***region_layer*** using only the objectives specified in the set statement.
  - **add** — Optimizes the ***region_layer*** by combining the existing objectives with the objectives specified in the add statement.
  - ***objective*** — A required objective used for optimization. At least one objective must be specified, and one factor value must follow each objective. The following objective values are supported:
    - **epe** — Optimizes the mask to minimize the sum of edge placement error (EPE).
    - **eperms** — Optimizes the mask to minimize the total area of edge placement error (EPE). Useful for layers with line-end features.
    - **slope** — Optimizes the mask to maximize the image slope.
    - *<job name>* — Applies the default objectives for the specified job name.
  - ***obj_factor*** — A required non-negative coefficient applied to the specified objective. Higher values emphasize the corresponding objective.

- pw *cond_name*

  An optional argument specifying the process window conditions to simulate. The contour_diff objective requires at least two process window conditions. The process window condition name must be used as the *cond_name* in a pw_condition statement. The process window condition name must also be specified in the pw_select statement for the job. The default behavior is to simulate all PW conditions.

- pattern *index*

  An optional argument specifying the pattern (mask, in multi-patterning runs) to which the objective applies. The *index* must match that of a layer statement. If this argument is not specified, then the objective is applied to all patterns.

Description

An optional statement that applies a custom objective to optimize a marker layer. The custom objective can be a single objective or a linear combination of multiple objectives.

Examples

Example 1

The following example uses a combination of objectives for optimizing the lineends1 mask layer:

```
objective_region lineends1 set epe 0.99 extraprint 0.99 slope 0.02
```

Example 2

The next example overrides the default Detail job objectives in the lineend_region for all process window conditions. The default Detail job objectives are restored for regions outside of the lineend_region only on the nominal process window condition.

```
job 6 detail
 pw_select nominal pw1 pw2
 objective_region lineend_region set epe 10
 objective_region backdrop set detail 1 pw nominal
```

Related Topics

layer

pw_condition

pw_select

pw_bridging

Type: pxopc_options Keywords - job sections only

Performs additional bridging correction for shapes within a specific pw_condition.

Usage

Conventional Syntax

pw_bridging ***cond_name*** **width** ***microns*** [weight *weight_val*]

Multi-patterning Syntax

pw_bridging ***cond_name*** **pattern** ***index***
cond_name2 **pattern** ***index2*** **width** ***microns*** [weight *weight_val*]

Arguments

- ***cond_name***

  A required argument specifying the name of a process window condition. The name must be defined in a pw_condition statement. If the job specifies conditions, name must also be specified in pw_select.

- **pattern** ***index***

  For the multi-patterning syntax only, a required keyword and value identifying the pattern to which the ***cond_name*** applies. The ***index*** must match that of a layer statement.

- **width** ***microns***

  A required argument specifying the distance in microns for the spacing by which contours must be separated. When two contours within ***cond_name*** are closer than ***microns***, it is considered to be a bridging violation.

  Run time is affected by the **width** setting. Smaller values take less time. A large value such as 1 micron may noticeably add to overall run time.

- ***cond_name2*** **pattern** ***index2***

  For the multi-patterning syntax only, a required keyword and value identifying the condition and pattern for the second layer. The ***index2*** must match that of a different layer statement than ***index***. All restrictions on ***cond_name2*** are the same as for ***cond_name***.

- weight *weight_val*

  An optional argument that specifies the weight to give to the bridging violation. Use this to balance between competing objectives. The default is 1.0.

Description

The pw_bridging keyword turns on bridging control for the specified process window condition. You can specify multiple pw_bridging conditions per job. It can be specified for any

job type, but is only recommended for the Detail job. Specifying pw_bridging outside of a job causes a run-time error:

```
ERROR: PXOPC: option "pw_bridging": should be specified in per-job section
```

Calibre pxOPC always attempts to optimize the layer in such a way that no pinching or bridging occurs, but cannot always achieve completely hotspot-free layouts. Using pw_bridging and pw_pinching, you can control which areas and process conditions should receive particular attention.

Bridging checks occur *within the process window condition.* If a bridge occurs between two different process window conditions, or if ***cond_name*** is defined to filter layout shapes so that a neighbor is not part of the condition, pw_bridging does not detect a violation.

The algorithm assumes that the contours it is checking are close to the target. This means that in the case of hard bridging or contours with a large displacement from their target, violations are not detected.

Process Window Constraints

Lithographic processes impose constraints on process windows that usually can be satisfied by putting print image contours on the target. When this is not the case, use pw_pinching and pw_bridging to specify additional constraints.

Typically pw_pinching and pw_bridging are used with an additional Detail job. That is, the recipe would include two Detail jobs in sequence. The second Detail job should repeat the parameters of the first Detail job and add pw_pinching and pw_bridging constraints.

These conditions should apply before using process window constraints:

- The contours should not have hard bridging or printing.
- The maximum EPE should not exceed 8 nm.
- When part of a Calibre LPE run, the print image contours should satisfy or nearly satisfy the constraints within the visible region.

Examples

The following lines show using pw_pinching and pw_bridging with a PV band.

```
job 5 detail
   pw_select nominal bridge pinch
   pw_pinching pinch width 0.005
   pw_bridging bridge width 0.010 weight 2
```

Related Topics

pw_pinching

job

pw_condition

Type: pxopc_options Keywords - general section only

Required statement. May appear more than once, but no more than 16 times.

Creates print image conditions for dose, focus, resist, and aerial threshold.

Usage

Standard format:

pw_condition *cond_name* {**optical *opt_name*** [dose *dose1*] [mask_size *bias*]}…
{**aerial *threshold*** | **resist *res_name***} [plane *height*] [weight *weight_val*]
[{inside | outside} *layer*]

Litho model format:

pw_condition *cond_name* [*litho_model_name*]
{[mask*n*] [focus *f*nm] [dose *dose1*] [bias *b*]}…
[aerial *threshold* | {model [thr_delta *offset*[%]]}]
[stochastic {high | low}] [stochastic_sigma *multiplier*]
[plane {*height* | SRAF_POSITIVE | SRAF_NEGATIVE | TOPLOSS_PLANE}]
spec *spec_name* [{inside | outside | not_inside | not_outside} *layer*]
[weight *weight_val*] [no_flare | flare *name*]

Arguments

- ***cond_name***

  A required argument that specifies the name of the process window condition. The first pw_condition statement in the setup file should have the name “nominal”.

- *litho_model_name*

  An optional argument that loads the litho model. It is required for the nominal process window condition when using litho models instead of separate optical and resist models.

  Different pw_condition statements can use different litho models. If *litho_model_name* is not specified, the litho model for the nominal condition is used. The user is responsible for making sure the litho models differ in only the areas intended to vary.

- **optical *opt_name***

  A required argument for standard format that specifies the name of the optical model and optionally its dose for this process condition. The pw_condition may also include a second optical model and dose.

- dose *dose1*

  An optional argument whose default is 1.0. For the standard format, it replaces the optical model dose. For litho model format, *dose1* is multiplied by the base dose specified for the mask in the litho model.

- **aerial** ***threshold***

  Specifies the value of the aerial threshold for this process condition. Either **aerial** or **resist** must be specified for the standard format. Generally **aerial** is preferred for faster run time.

  For litho model format, this argument is optional. If it is not specified, the resist model defined in the litho model is used.

  For litho models only, the ***threshold*** argument can take the value **model**. This sets the aerial threshold to the image_threshold value from the litho model. If image_threshold is not set, it uses the resist model value.

- **resist** ***res_name***

  Specifies the name of the resist model as given in resist_model_load for this process condition. Either **aerial** or **resist** must be specified for the standard format. Although **aerial** is preferred for run time, the Refine job benefits significantly from the more accurate modeling obtained when **resist** is specified.

- mask_size *bias*

  An optional argument that resizes the mask shapes for this process window condition only. Use it for mask bias. The default value is 0.0 (no biasing). Units are in microns.

- stochastic {high | low}

  An optional argument that includes the EUV stochastic model. It accounts for EUV shot noise by modifying the simulation threshold for critical dimension (CD).

  high — Use positive threshold deviation.

  low — Use negative threshold deviation.

  By default, stochastic model effects are not included in simulation even if the model is in the litho model.

- stochastic_sigma *multiplier*

  An optional argument that sets the confidence interval of the CD threshold change with the EUV stochastic model. It is ignored unless stochastic low or stochastic high is specified. The default for *multiplier* is 3.0. Values must be positive.

- plane *height*

  An optional argument that is used for 3D resist imaging. It specifies the z-position within the resist film where the image is computed. The position (*height*) can take one of the following four forms when using litho models. (The standard format only allows a positive number.)

  positive number — A number that is interpreted to be in microns relative to the bottom of the resist. Its value must be between 0 and the resist film thickness, inclusive.

  A positive number can be used with either an aerial threshold or a CM1 3D resist model.

  SRAF_POSITIVE — A keyword that indicates the pw_condition models the printing of positive SRAFs.

SRAF_NEGATIVE — A keyword that indicates the pw_condition models the printing of negative SRAFs.

TOPLOSS_PLANE — A keyword that indicates the pw_condition models resist toploss.

All values require an optical model that uses the imageplanes keyword. If a resist model is specified, it must be a CM1 3D resist model. The TOPLOSS_PLANE, SRAF_POSITIVE and SRAF_NEGATIVE keywords cannot be used with aerial, and also require that the litho model contain a zplanes model.

For SRAF_POSITIVE and SRAF_NEGATIVE, the actual plane value calculated depends on whether the resist is negative or positive tone (NTD or PTD), and whether the mask is darkfield or clearfield.

- weight *weight_val*

  An optional argument that specifies the weight to give to the EPE defined for this condition. The default weight is 1.0.

- mask*n*

  An optional argument for the litho model format. The value *n* is run into the keyword “mask”, for example, “mask0”. This argument specifies the mask number from the litho model to which the other arguments apply. Mask0 is the default.

- focus *f*nm

  An optional argument for the litho model format. The value *f* must be appended with “nm”, for example, “focus 30nm”. This argument specifies the optional focus condition to be used for the mask specified by the mask*n* argument. It maps to the optical model defined by “optical *file* focus *f*nm” in the litho model. If not specified, the nominal focus from the litho model is used.

- bias *b*

  An optional argument for the litho model format specifying bias in microns. It is used for sizing the mask in simulation. Bias values are not used in OPC mask movements. If any biases were specified for individual mask_layers in the litho model, those biases are added to the value specified here. The default is 0.

  Note

  Only mask biasing for individual mask layers is added. Mask-block level biasing using the XSHIFT or YSHIFT keywords in the litho model generates an error.

- thr_delta *offset*[%]

  An optional argument for the litho model format that can only be used in conjunction with “aerial model”. This argument modifies the image_threshold parameter in the litho model. The offset can be given as a numeric shift (for example, thr_delta -0.001) or a percentage change (for example, -0.1%).

- spec *spec_name*

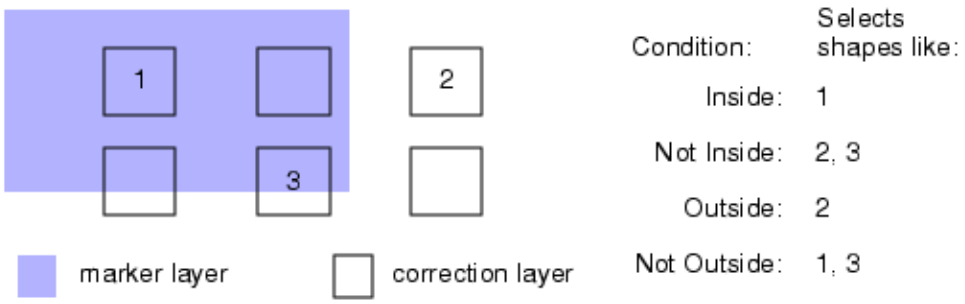
  An optional argument set that specifies an optical model labeled *spec_name* in the *Lithomodel* file to use for the process window condition. This argument set is commonly used to specify optical models that include aberrations.

- {inside | outside | not_inside | not_outside} *layer*

  An optional argument that applies the defined process window condition only to polygons that meet one of the following conditions for the specified layer: completely “inside” the layer, completely “outside” the layer, not completely inside the layer, or not completely outside the layer. The default behavior is to apply the process window condition everywhere.

  The not_inside and not_outside forms can only be used with litho models.

Figure 3-14. Inside, Not Inside, Outside, and Not Outside



- no_flare

  An optional argument for the litho model format that can only be used with ***cond_name*** “nominal”. When specified with an EUV optical model, no process window conditions use the flare model during simulation.

  The no_flare argument cannot be specified with flare *name*.

- flare *name*

  An optional argument for the litho model format that sets this particular process window condition to use a named flare_longrange specification instead of the default unnamed flare_longrange. The *name* value must match that of a “flare *string*” argument on a flare_longrange specification.

  The flare *name* argument cannot be specified with no_flare.

Description

Note

A setup file must either use the litho model syntax for all commands, or not at all. Standard syntax and litho model syntax cannot be mixed.

A required command to specify process window conditions for optimization. When using litho model format, this command also loads the litho model. The EPE is measured for all defined process window conditions, unless restricted by pw_select. At least one condition must be defined in the pxopc_options block.

Tip

Use 1 to 2% larger exposure latitude and 10% larger focus latitude for Calibre pxOPC runs than for Calibre OPCverify runs.

Examples

Example 1 — Basic Process Window Conditions

The following code shows only the necessary lines for defining process window conditions. There must be a LITHO FILE statement that loads models. These models are used inside the pxopc_options block. In this case, three process window conditions are created: NOMINAL, OUTER, and INNER. All three use a resist model. OUTER and INNER also show using mask_size to handle mask bias.

```
LITHO FILE px.setup [ /*
   modelpath models
   optical_model_load f0    e1_f+0.0000
   optical_model_load fplus e1_f+0.0570
   resist_model_load resist_mod nom.mod
...

   pxopc_options px.options {
      ...
      pw_condition NOMINAL optical f0 dose 1.0 resist resist_mod
      pw_condition OUTER optical f0 dose 1.04 mask_size +0.001 \
         resist resist_mod
      pw_condition INNER optical fplus dose 0.94 mask_size -0.001 \
         resist resist_mod
      ...
   }
*/ ]
```

Example 2 — Balancing Speed and Detail

The following code shows how to define overlapping process window conditions and how to use pw_select within the jobs to run just the most relevant conditions.

```
LITHO FILE px.setup [ /*
   modelpath models
   optical_model_load f0    e1_f+0.0000
   optical_model_load fplus e1_f+0.0570
   resist_model_load resist_mod nom.mod
...
```

```
   pxopc_options px.options {
      ...
      pw_condition NOMINAL_CM1 optical f0 dose 1.0 resist resist_mod
      pw_condition NOMINAL_AER optical f0 dose 1.0 aerial 0.21
      pw_condition OUTER_AER optical f0 dose 1.04 aerial 0.21
      pw_condition INNER_AER optical fplus dose 0.94 aerial 0.21
      ...

      job 1 open
         pw_select NOMINAL_AER

      job 2 decorate
         pw_select INNER_AER OUTER_AER

      job 3 correct
         pw_select NOMINAL_AER

      job 4 refine
         pw_select NOMINAL_CM1

      job 5 lmrc
         pw_select OUTER_AER
   }
*/ ]
```

Generally, runs are faster if each job uses only one process condition. This can be done for all jobs except Decorate; the Decorate job requires at least two process conditions in order to optimize PV bands as functions of created assist features. If this causes a contour to not print for a job, set the previous job to use all process conditions.

For example, if the settings in the LITHO FILE above resulted in the Decorate job not printing the outer contour, you would set the Open job with all conditions:

```
      job 1 open
         pw_select NOMINAL_AER OUTER_AER INNER_AER
```

Another way to improve speed is to use the aerial (CTR model) instead of a resist model. Most of the time this does not cause significant differences. The Refine job, which performs detailed optimization, is the exception. If possible, use a resist model with Refine.

Example 3 — Multiple Litho Models

The code in Example 1 can also be represented with litho models:

```
LITHO FILE px.setup [ /*
   modelpath models
   # Notice there are no models explicitly loaded in the general section.
   # The NOMINAL pw_condition statement loads the main litho model.
...
```

```
    pxopc_options px.options {
       ...
       pw_condition NOMINAL nominal_lm dose 1.0
       pw_condition OUTER dose 1.04 bias +0.001
       pw_condition INNER overexpose_lm dose 0.94 bias -0.001
       ...
    }
*/ ]
```

The NOMINAL and OUTER conditions use the same litho model, nominal_lm, for optical and resist models. The INNER condition uses a different model, overexpose_lm, to demonstrate how you would use multiple litho models. Both of these litho model directories must be in the modelpath. In all three cases, the optical and resist models are implicitly defined by the litho model.

pw_condition_group

Type: pxopc_options Keywords - any section

Creates a named set of automatically generated conditions based on specified process window parameters. The conditions are only accessible as a group; not individually.

Usage

pw_condition_group *pw_group_name* [shape_type *type*] [focus_range *fr_nm*] [focus_center *fc_nm*] [dose_latitude *dose_l*] [dose_center *dose_c*]

Arguments

- ***pw_group_name***

  A required argument that specifies the name of the process window condition group for use in other commands.

- shape_type *type*

  An optional argument that specifies the shape inscribed in the process window. The following values are supported:

  elliptical — Inscribes an ellipse in the common process window. This is the default.

  rectangular — Inscribes a rectangle in the common process window.

  vertical_segment — Maximizes vertical range (dose) without regard to horizontal range (focus). This is recommended for dose latitude optimization (EL or EL_Curvilinear jobs).

- focus_range *fr_nm*

  An optional argument that specifies a value in nanometers for the focus range. This is the initial width of the common process window shape. The default is 100.

- focus_center *fc_nm*

  An optional argument that specifies a value in nanometers for the focus center. This is the initial center of the common process window shape. The default is 0.

- dose_latitude *dose_l*

  An optional argument that specifies a value for the exposure latitude. This is the initial height of the common process window shape. The default is 0.06.

- dose_center *dose_c*

  An optional argument that specifies a value for the dose center. This is the initial center of the common process window shape. The default is 1.0.

Description

An optional command that creates a group of process window conditions that would create a common process window[1] satisfying the shape, focus, and dose settings. Use this command

instead of pw_condition when you are more interested in a range of process window conditions than in simulating specific values.

The pw_condition_group command can appear more than once, but each ***pw_group_name*** must be unique.

Examples

Example 1

This statement creates a group named “PWG” that has an initial focus range of 100 nm and a dose latitude of 5%.

```
pw_condition_group PWG dose_latitude 0.05
```

Example 2

These are the recommended settings for EL jobs:

```
pw_condition_group PWG_focus shape_type elliptical focus_range 100
pw_condition_group PWG_dose shape_type vertical_segment
```

The job section should select them in the following order:

```
pw_select PWG_focus PWG_dose
```

Each process window group should also have a tolerance bound, for a total of four pw_tolerance_bound statements.

1. A common process window is a composite graph composed of simulated CD measurements for a variety of pitches as dose and focus are varied. The area inside all the focus-dose curves represents the common process window.

pw_dose_latitude_factor

Type: pxopc_options Keywords - job sections only

Controls the relative emphasis given to the dose latitude during process window optimization.

Usage

pw_dose_latitude_factor *pw_group_name value*

Arguments

- ***pw_group_name***

  A required argument specifying the name of a process window condition group. The name must also be used as the ***pw_group_name*** in a pw_condition_group statement as well as in a pw_select statement for the job.

- ***value***

  A required argument indicating how aggressively Calibre pxOPC tries to increase the dose latitude. The default value is 1.0.

Description

The pw_dose_latitude_factor keyword is only recommended for the EL job.

Because process window conditions specified with pw_condition have set dose and focus they cannot be used with pw_dose_latitude_factor. Using a process window condition instead of a process window condition group returns this error:

```
ERROR: PXOPC: job <n>: option "pw_dose_latitude_factor": group <cond_name>
was not found among list of process window groups.
```

Examples

This example shows the necessary statements for optimizing dose latitude:

```
pxopc_options px.options {
   ...
   pw_condition_group PWG ...

   ...
   job 7 el
      pw_select PWG ...
      pw_dose_latitude_factor PWG 1.01
}
```

pw_enclose

Type: pxopc_options Keywords - job sections only

Identifies a layer that should remain enclosed during pw_pinching and pw_bridging corrections.

Usage

pw_enclose *cond_name* [pattern *index*] **width *microns* layer *name*** [weight *weight_val*]

Arguments

- ***cond_name***

  A required argument specifying the name of a process window condition. The name must be defined in a pw_condition statement and also in pw_select.

  The pw_condition inside and outside definitions are respected; if the shape to be enclosed and the contour are on different sides of the boundary, violations will not be detected.

- pattern *index*

  An optional keyword and value identifying the pattern (mask, in multi-patterning runs) to which the enclosure constraint applies. The *index* must match that of a layer statement.

- **width *microns***

  A required keyword and value that define the enclosure. The distance is measured between the shapes on **layer *name*** and the ***cond_name*** contours of the layer being adjusted. The value must be greater than 0.

- **layer *name***

  A required keyword and value identifying the layer containing the shapes to be enclosed. It is recommended to supply a curved layer to better represent the manufacturable target.

  The algorithm assumes that the shapes on the layer being adjusted already fully enclose the shapes on the layer. If this is not the case, or if as initially drawn the two layers have coincident edges, use a copy of the drawn layer with sized down targets.

- weight *weight_val*

  An optional argument that specifies the weight to give to the enclosure violation. Use this to balance between competing objectives. The default is 1.0.

Description

This optional keyword causes the selected process window condition to perform bridging checks between the identified layer (***name***) and the layer currently undergoing OPC. It ensures the distance between the layer's contours at that condition and the shapes on ***name*** are included in the job's objective so that layers interconnect well. Use the weight keyword to indicate how important enclosure is relative to the other objectives specified for the job.

The pw_enclose statement can be specified multiple times for different conditions.

The algorithm assumes that the contour and the enclosed shape are close (within about 8 nm) and overlapping. The enclosed layer shapes should be large enough (typically, not less than 8 nm) to avoid interfering with the measurement. If any of this is not the case (for example, line end pullback partially uncovering a via), the check may fail.

Like pw_bridging and pw_pinching, pw_enclose increases memory requirements and run time. It is best used in the Detail job.

Suggested Method

To most likely ensure that all conditions are met, take these precautions:

1. Generate a smoothed version of the enclosed layer using setlayer curve.
2. Shrink the contours on the smoothed version by some amount (referred to as *s*). Do not shrink so far as to make the shapes more narrow than about 8 nm.
3. Have pw_enclose use the shrunk, smoothed layer and specify **width** as *s* + required constraint.

Examples

The following example specifies to check the enclosure between the inner contour and the shapes on via_all_both_curve_sized is at least 11 nanometers. This objective is weighted four times as heavily as the overall objective of the Detail job, optimizing the overall process window and keeping the mask MRC clean.

```
job 6 detail
   pw_select nominal inner outer
   pw_enclose inner width 0.011 layer via_all_both_curve_sized weight 4
   iterations 20
```

Notice the Detail job iterations were also increased in order to reach a better solution.

Related Topics

pw_bridging

pw_condition

pw_select

pw_focus_base_points

Type: pxopc_options Keywords - job sections only

Defines the defocus conditions to use for interpolation of aerial images.

Usage

pw_focus_base_points ***focus focus focus focus*** [*focus*…]

Arguments

- ***focus focus focus focus***

  A required argument listing four or more defocus values. The units are in nanometers.

  The default is -100 nm to 100 nm, in 20 nm increments.

Description

The optional pw_focus_base_points keyword lists defocus values to use when interpolating aerial image throughfocus. The litho model must contain the pre-computed defocus models or the run exits with the following error:

```
pxopcOptionPwConditionLM::getTccName: model not available
```

The list of focus values should slightly exceed the minimum and maximum hoped for common depth of focus.

Examples

This example overrides the default focus base points:

```
pw_focus_base_points -120 -90 -60 -30 0 30 60 90 120
```

pw_focus_range_factor

Type: pxopc_options Keywords - job sections only

Controls the relative emphasis given to a process window condition group during process window optimization.

Usage

pw_focus_range_factor *pw_group_name value*

Arguments

- ***pw_group_name***

  A required argument specifying the name of a process window condition group. The name must also be used as the ***pw_group_name*** in a pw_condition_group statement as well as in a pw_select statement for the job.

- ***value***

  A required argument indicating how aggressively Calibre pxOPC tries to increase the focus range for the specified process window condition group. The default value is 1.0.

Description

The pw_focus_range_factor keyword is only recommended for the CDOF job. This optional keyword controls how much the optimization emphasizes increasing the focus range.

Because process window conditions specified with pw_condition have set dose and focus they cannot be used with pw_focus_range_factor. Using a process window condition instead of a process window condition group returns this error:

```
ERROR: PXOPC: job <n>: option "pw_focus_range_factor": group <cond_name>
was not found among list of process window groups.
```

Examples

This example demonstrates emphasizing a dose-based process window group (PWG) over a process window condition (SRAF_PRINT) when optimizing for depth of focus:

```
pxopc_options px.options {
   ...
   pw_condition nominal litho.model
   pw_condition SRAF_PRINT dose 1.1 aerial 0.15
   ...
   pw_condition_group PWG dose_latitude 0.05

   ...
   job 7 cdof
      pw_select SRAF_PRINT PWG
      pw_focus_range_factor PWG 3
   ...
}
```

Related Topics

pw_condition_group

pw_select

pw_pinching

Type: pxopc_options Keywords - job sections only

Performs additional pinching correction for shapes within a specific pw_condition.

Usage

Conventional Syntax

pw_pinching *cond_name* width *microns* [weight *weight_val*]

Multi-patterning Syntax

pw_pinching *cond_name* pattern *index*
cond_name2* pattern *index2* width *microns [weight *weight_val*]

Arguments

- ***cond_name***

  A required argument specifying the name of a process window condition. The name must be defined in a pw_condition statement. If the job specifies conditions, name must also be specified in pw_select.

- **pattern *index***

  For the multi-patterning syntax only, a required keyword and value identifying the pattern to which the ***cond_name*** applies. The ***index*** must match that of a layer statement.

- **width *microns***

  A required argument specifying the minimum inner distance for the contour in microns.

  Run time is affected by the **width** setting. Smaller values take less time. A large value such as 1 micron may noticeably add to overall run time.

- ***cond_name2* pattern *index2***

  For the multi-patterning syntax only, a required keyword and value identifying the condition and pattern for the second layer. The ***index2*** must match that of a different layer statement than ***index***. All restrictions on ***cond_name2*** are the same as for ***cond_name***.

- weight *weight_val*

  An optional argument that specifies the weight to give to the pinching violation. Use this to balance between competing objectives. The default is 1.0.

Description

The pw_pinching keyword turns on pinching control for the specified process window condition. You can specify multiple pw_pinching conditions per job. It can be specified for any job type, but is only recommended for the Detail job. Specifying pw_pinching outside of a job causes a run-time error:

```
ERROR: PXOPC: option "pw_pinching": should be specified in per-job section
```

Calibre pxOPC always attempts to optimize the layer in such a way that no pinching or bridging occurs, but cannot always achieve hotspot-free layouts. Using pw_bridging and pw_pinching, you can control which areas and process conditions should receive particular attention.

Pinching checks occur *within the process window condition*. If a ***cond_name*** is defined to filter layout shapes so that the facing contour is not part of the condition, pw_pinching does not detect a violation.

The algorithm assumes that the contours it is checking are close to the target. This means that in the case of hard pinching or contours with a large displacement from their target, violations are not detected.

Process Window Constraints

Lithographic processes impose constraints on process windows that usually can be satisfied by putting print image contours on the target. When this is not the case, use pw_pinching and pw_bridging to specify additional constraints.

Typically pw_pinching and pw_bridging are used with an additional Detail job. That is, the recipe would include two Detail jobs in sequence. The second Detail job should repeat the parameters of the first Detail job and add pw_pinching and pw_bridging constraints.

These conditions should apply before using process window constraints:

- The contours should not have hard bridging or printing.
- The maximum EPE should not exceed 8 nm.
- When part of a Calibre LPE run, the print image contours should satisfy or nearly satisfy the constraints within the visible region.

Examples

The following lines show using pw_pinching and pw_bridging with a PV band.

```
job 5 detail
   pw_select nominal bridge pinch
   pw_pinching pinch width 0.005
   pw_bridging bridge width 0.010 weight 2
```

Related Topics

pw_bridging

job

pw_select

Type: pxopc_options Keywords - any section

Restricts simulation to only the specified processes.

Usage

Standard Syntax

pw_select *cond* [*cond*…]

Multi-Patterning Syntax

pw_select *cond* [*cond*…] [pattern *index*]

Arguments

- ***cond***

  A required name of at least one process condition or process window condition group. The name must also be used as either the ***cond_name*** in a pw_condition statement or the ***pw_group_name*** in a pw_condition_group statement. Names are not case-sensitive.

  Only use process window condition groups within CDOF jobs. When a pw_select statement that includes a process window condition group is used in the general section, the run exits with this error:

  ```
  ERROR: PXOPC: option "pw_select": pw condition group isn't allowed
  to be selected in the global section
  ```

  Either remove the process window condition group from the instance in the general section or copy the pw_select statement to each job section as applicable.

- pattern *index*

  An optional argument for multi-patterning layouts that specifies the pattern (mask or color) for which the listed process conditions should be used.

- weight *value*

  An optional argument that specifies the weight to give to this process condition group when simulating.

Description

An optional statement that restricts simulation to the process window group or condition ***cond***(s) listed. The process window conditions are defined in the pw_condition statements. Process window condition groups are defined in the pw_condition_group statements. Only the specified process conditions are simulated.

By default, only the nominal process condition is simulated for the Open and LMRC jobs, and all process conditions for the Decorate, Decoratedarksraf, Correct, Refine, and Finetune jobs. You must use pw_select for Detail and CDOF jobs.

This statement can be used to restrict conditions for the entire pxOPC run by placing it in the general section of the pxopc_options block. It can also be used to override the conditions on a per-job basis by placing it in a job section. In that case, only the job where the statement appears is affected; the next job runs with the general setting.

Examples

The following example defines three process conditions but runs only the nominal process condition:

```
pxopc_options px.options {
   ...
   pw_condition NOMINAL optical ...
   pw_condition OUTER optical...
   pw_condition INNER optical ...

   pw_select nominal
   ...
}
```

pw_tolerance_bound

Type: pxopc_options Keywords - job sections only

Defines inner and outer contours for a process window condition group in CDOF and EL jobs.

Usage

pw_tolerance_bound ***pw_group_name*** {**outside** | **inside**} ***layer***

Arguments

- ***pw_group_name***

  A required argument specifying the name of a process window condition group. The name must also be used as the ***pw_group_name*** in a pw_condition_group statement as well as in a pw_select statement for the job.

- **outside** | **inside**

  A required argument specifying the type of contour. Use “outside” for the contour representing the smallest acceptable shape, and “inside” for the contour representing the largest acceptable shape.

- ***layer***

  A required argument specifying the name of a layer. The name must also be used as the ***name*** in a layer statement.

  The layer should contain a contour to act as either the inner or outer contour of the tolerance band.

Description

The optional pw_tolerance_bound keyword identifies the layers comprising the tolerance band. The tolerance band indicates what constitutes acceptable edge placement when optimizing for conditions other than EPE. All print image contours of the process window condition group must be between the outside and inside contours. By default, the CDOF job constructs a tolerance band by scaling the target layer 10%.

The most important property of the tolerance band is that its width (that is, the difference between the inner and outer contour) should never be less than the CD tolerance. The width should be even larger at line ends and corners as shown by the inner and outer red curves in the following figure.

Near the depth of focus gauges, the band should be exactly equal to CD tolerance. Where features are very close, adjusting the smaller contour on just those features can help avoid hard bridging.

Examples

This example identifies the inner and outer contours of the tolerance band for use in a CDOF job:

```
pw_tolerance_bound PWG inside tolband_outer
pw_tolerance_bound PWG outside tolband_inner
```

pw_vertical_constraint

Type: pxopc_options Keywords - job sections only

Optimizes contours to be inside or outside of the shapes of the specified layer for a certain process window condition or group.

Usage

pw_vertical_constraint ***pw_name*** {{**outside** | **inside**} ***layer***}
[weight *weight_val*]

Arguments

- ***pw_name***

  A required argument specifying the name of a process window condition or process window condition group. If specifying a process window condition, the name must also be used as the ***cond_name*** in a pw_condition statement as well as in a pw_select statement for the job. If specifying a process window condition group, the name must also be used as the ***pw_group_name*** in a pw_condition_group statement as well as in a pw_select statement for the job.

- {**outside** | **inside**} ***layer***

  A required argument set specifying whether the image must print outside or inside a specified layer. The ***layer*** must be among the input layers. In most cases, the layer type is hidden.

- weight *weight_val*

  An optional argument set specifying the aggressiveness of the optimization. Use this to balance between competing objectives. The default is 1.

Description

The pw_vertical_constraint keyword optimizes contours to be inside or outside of the shapes of the specified layer for a certain process window condition or group. Multiple pw_vertical_constraint statements can be specified.

Examples

The following optimizes contours at the nominal process window condition to be inside the target layer shapes:

```
job 6 detail
   pw select nominal
   ...
   pw_vertical_constraint nominal inside target weight 25
   ...
```

rate

Type: pxopc_options Keywords - any section

Sets the rate of mask change per iteration.

Usage

rate ***rate_val*** [*rate_val* …]

Arguments

- ***rate_val***

  An argument specifying how much the mask is changed during one iteration. The default value is tuned for specific job types.

  Multiple rates are separated by spaces. As of version 2014.4, commas are not allowed.

Description

This keyword sets a list of rates. The ***rate_val*** specifies how much the mask is changed during one iteration. Careful rate selection can improve run time for the Open and Decorate jobs.

Note

The rates have been carefully tuned for each job. It is not recommended you override the default rate for the Correct, Refine, or Detail jobs. Lmrc jobs are not affected by rate because they do not perform optimization.

At the first iteration, the first ***rate_val*** is read from the list. As long as this rate decreases the objective function (improves the EPE), the rate remains the same. If the objective function instead increases, the algorithm uses the next *rate_val* in the list. If the objective function increases and all *rate_val* entries have been used, the algorithm uses a reduced rate, typically 0.5 or 0.7 of the last used rate. To see the rates used, set debug_level to 1.

The length of the ***rate_val*** list is unrestricted.

Examples

```
rate 0.9 0.7 0.65 0.6
```

scatter_assist_type

Type: pxopc_options Keywords - any section

Specifies where to place SRAFs relative to the target layer.

Usage

scatter_assist_type {**outside** | **inside** | **both**}

Arguments

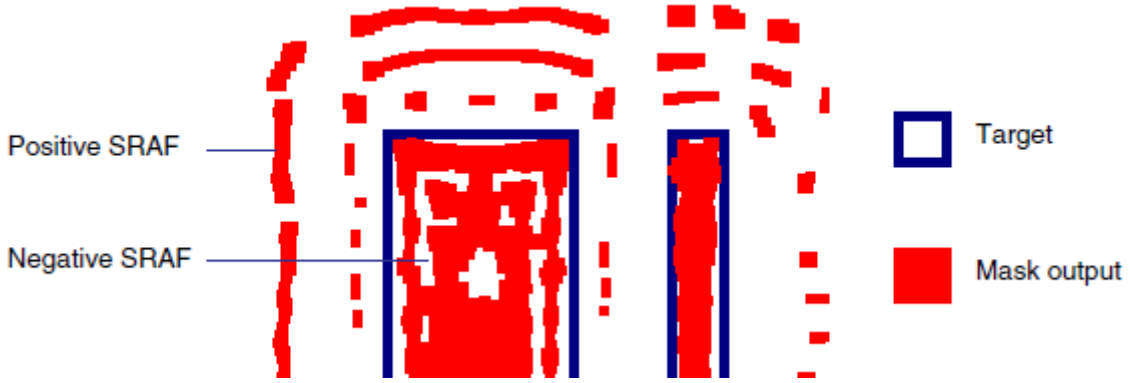
- **outside** | **inside**| **both**

  A required argument specifying whether SRAFs go outside the target layer, inside the target layer, or both inside and outside the target layer. Only one of the keywords may be specified.

Description

By default, SRAFs are placed outside the target layer shapes. Use the **scatter_assist_type** command to create negative scattering bars instead of or in addition to the positive SRAFs.

Figure 3-15. Positive and Negative SRAFs



Calibre pxOPC combines the SRAF shapes and OPC corrections on the same layer, unlike when OPC correction and SRAF generation are run separately. Negative SRAFs cause the main feature shape to increase significantly in complexity.

Examples

```
scatter_assist_type outside
```

Related Topics

assist_inside_dose_multiplier

assist_outside_dose_multiplier

scatter_belt

scatter_offset

scatter_belt

Type: pxopc_options Keywords - any section

Identifies how far from the target the SRAF rings should extend.

Usage

scatter_belt *belt_value*

Arguments

- ***belt_value***

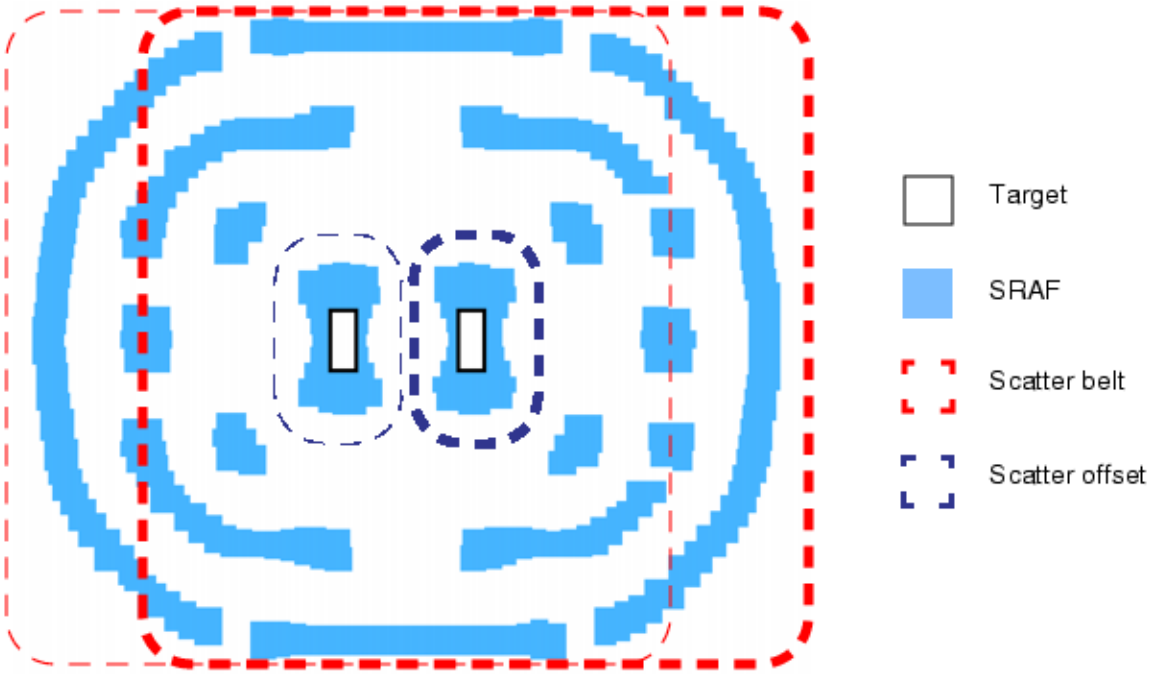
  A required argument specifying the distance in microns from the mask to the outer edge of the SRAFs.

Description

Specifies how far SRAFs extend. The distance is measured from the mask returned from the previous job. The default if the command is not specified is 0.256 um.

This command is most often used in the Decorate job. If there are too few SRAF rings, increase the ***belt_value*** by adding 0.25*CD (one quarter the critical distance).

Figure 3-16. scatter_belt



Related Topics

scatter_offset

scatter_belt_negative

Type: pxopc_options Keywords - any section

Identifies how far inside the target negative SRAFs should extend.

Usage

scatter_belt_negative *belt_value*

Arguments

- ***belt_value***

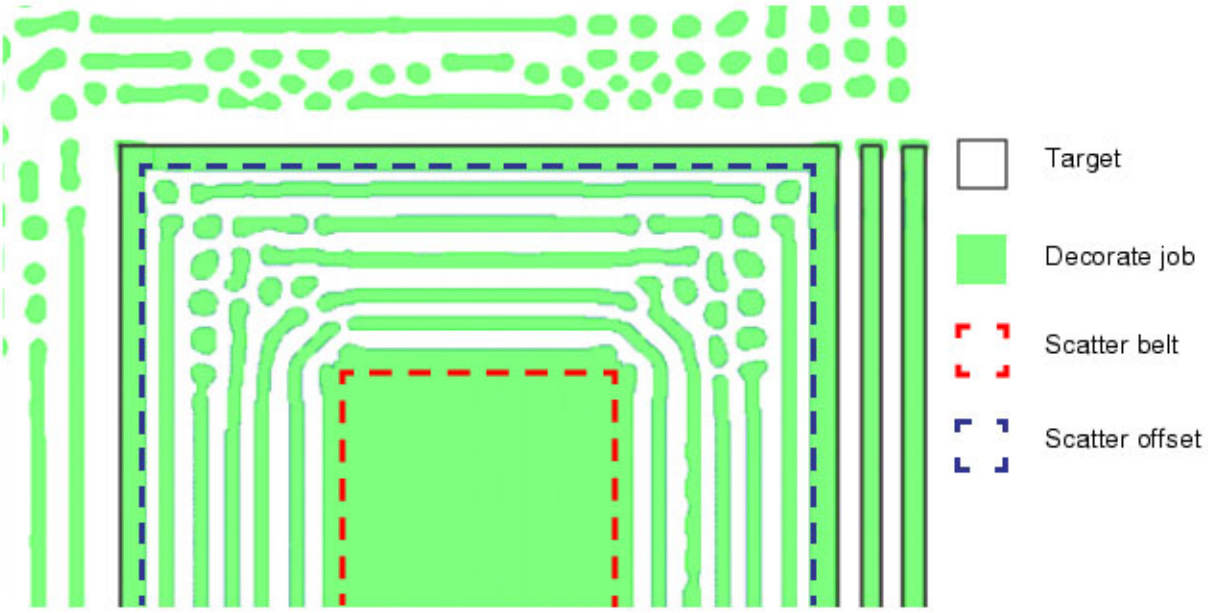
  A required argument specifying the distance in microns from the shapes on the output mask layer to the edge of the internal SRAFs. The negative SRAF belt width is the ***belt_value*** minus the scatter_offset_negative ***value***, both being applied to the inside of the polygon.

Description

By default, Calibre pxOPC creates only positive SRAFs (that is, SRAFs outside the target shape). To change this behavior, you must set "scatter_assist_type both" or "scatter_assist_type inside" to create negative (within target) SRAFs. If negative SRAFs are enabled and this command is not set, by default the negative SRAFs are allowed to fill the polygon if needed.

This command is most often used in the Decorate job. If there are too few SRAF rings, increase the ***belt_value*** by adding 0.25*CD (one quarter the critical distance).

Figure 3-17. Negative Scatter Belt



Examples

```
scatter_belt_negative 0.256
```

scatter_offset

Type: pxopc_options Keywords - general section only

Specifies the minimum distance between the main feature and positive SRAFs.

Usage

scatter_offset *value*

Arguments

- ***value***

  A required argument specifying the distance from the main feature edge to the closest SRAFs. The distance must be greater than 0.

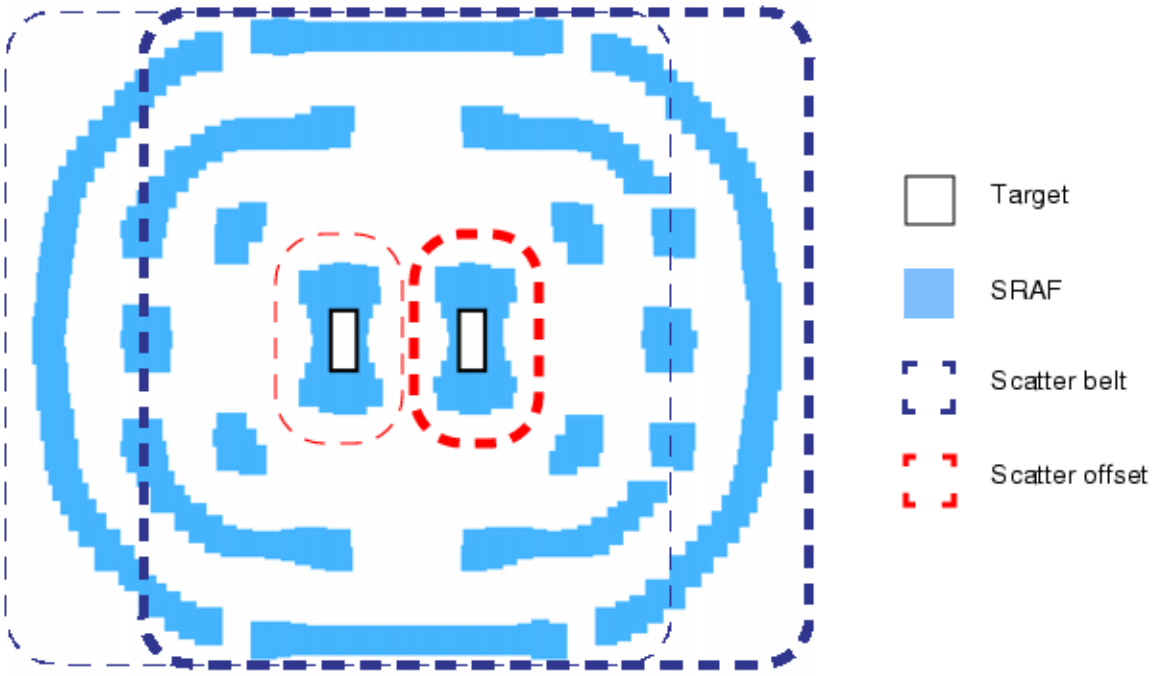
  As of 2018.4, the units are in microns. Previously the units were equal to 0.125*lambda/NA, where NA is numerical aperture and lambda is the wavelength. The default is based on the optical model.

Description

Specifies how close SRAFs may be to the main feature edge. The maximum distance is set by scatter_belt. The offset value affects only positive SRAFs. To set an offset for negative SRAFs, set scatter_offset_negative.

By default, the offset is calculated between the output mask main feature and SRAF. In some cases, the difference between mask and target is large enough to allow SRAFs to be placed in undesired locations. To calculate the distance instead using the target layer and the output mask SRAF, set “scatter_offset_type target”.

Figure 3-18. scatter_offset



Examples

```
scatter_offset 1.3
```

Related Topics

scatter_offset_negative

scatter_offset_type

scatter_offset_negative

Type: pxopc_options Keywords - any section

Specifies the minimum distance between the main feature edge and negative SRAFs.

Usage

scatter_offset_negative ***value***

Arguments

- ***value***

  A required argument specifying the distance from the main feature edge to the closest SRAFs inside the drawn shape. The distance must be greater than 0.

  As of 2018.4, the units are in microns. Previously the units were equal to 0.125*lambda/NA, where NA is numerical aperture and lambda is the wavelength. The default is based on the optical model.

Description

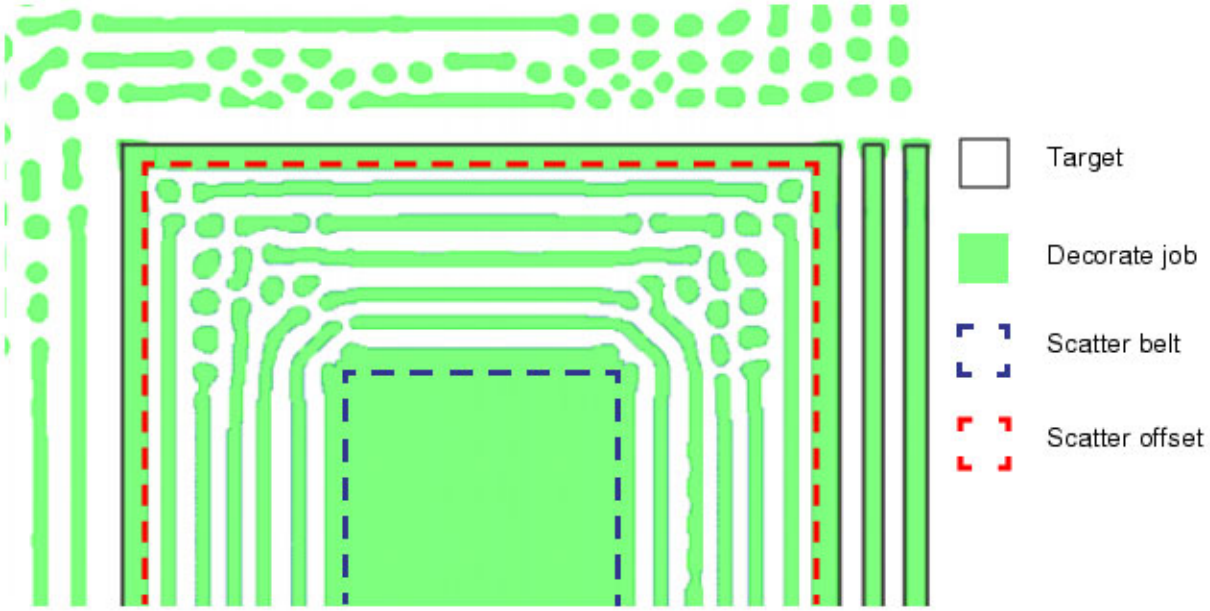
Specifies how close SRAFs may be to the main feature edge. The maximum distance inward is set by scatter_belt_negative. (See Figure 3-19.) The offset value affects only negative SRAFs.

By default, Calibre pxOPC creates only positive SRAFs (that is, SRAFs outside the shape). To change this behavior, you must set “scatter_assist_type both” or “scatter_assist_type inside” to create negative (within shape) SRAFs.

The offset is calculated using the output mask by default. In some cases, the difference between mask and target is large enough to allow SRAFs to be placed in undesired locations. To calculate the distance instead using the target layer and the output mask SRAF, set “scatter_offset_type target”.

This command can appear in a job or the general section, but is most often used in the Decorate job.

Figure 3-19. Negative Scatter Offset



Examples

```
scatter_offset_negative 1.3
```

Related Topics

scatter_assist_type

scatter_belt_negative

scatter_offset_type

scatter_offset_type

Type: pxopc_options Keywords - any section

Sets whether scatter_offset and scatter_offset_negative are calculated based on mask or target shapes.

Usage

scatter_offset_type {**mask** | **target**}

Arguments

- **mask** | **target**

  A required argument specifying whether to use the mask layer (the layer with type "correction") or target layer when calculating the offset of assist features. The default is mask.

Description

By default, the scatter parameters are calculated using the layer of type correction, the output mask. Sometimes this can result in weak or inconsistent SRAFs. In these cases, the target layer may provide better SRAF placement. You can try this out by adding "scatter_offset_type target" to the recipe.

The scatter_offset_type setting applies to both positive and negative SRAFs.

Examples

```
scatter_offset_type target
```

Related Topics

scatter_offset

set_on_target

Type: pxopc_options Keywords - job sections only

Improves EPE for the specified process window condition.

Usage

set_on_target *cond_name* [*cond_name*…] [pattern *index*] [weight *value*]

Arguments

- ***cond_name***

  A required argument specifying the names of one or more process window conditions. If the job uses a pw_select statement, the same ***cond_name*** must be listed in it.

- pattern *index*

  An optional argument for multi-patterning setup files. The index values are 0, 1, 2, or 3, and should also be defined on a layer statement. If this argument is not specified, all patterns are used.

- weight *value*

  An optional argument that controls the aggressiveness of EPE optimization. The default is 1.

Description

This optional command can improve EPE for the specified process window conditions, but at some cost to other metrics such as overall process variation or SRAF printing. At high weights, the run time may increase significantly. The set_on_target keyword is supported in the following job types: CDOF, Open, Correct, Refine, Finetune, and Detail.

Examples

This is the recommended usage for CDOF jobs:

```
job 7 cdof
   set_on_target nominal
```

Related Topics

layer

pw_select

size_by

Type: pxopc_options Keywords — size job section only

Resizes the input layer polygons.

Usage

size_by {***resize_um*** **um**} | {***resize_pct*** **percent**}

Arguments

- ***resize_um*** **um**

  A required argument specifying the amount in microns to resize the input layer polygons. Either ***resize_um*** **um** or ***resize_pct*** **percent** must be specified. Positive and negative numbers are supported.

- ***resize_pct*** **percent**

  An required argument specifying the percentage of resize to apply to the input layer polygons. Either ***resize_um*** **um** or ***resize_pct*** **percent** must be specified. Positive and negative percentages are supported.

Description

This optional command allows you to resize the input layer polygons in the Size job type. You must specify either an absolute amount or a percentage.

Examples

The following example shows a job reducing the size of the input layer polygons by 10 percent:

```
job 1 size
size_by -10 percent
```

Related Topics

job

size_max_shift_edge

size_max_shift_edge

Type: pxopc_options Keywords — size job section only

Limits the resizing of input layer polygons by “size_by percent”.

Usage

size_max_shift_edge *max_shift_um*

Arguments

- *max_shift_um*

  An optional argument specifying the maximum edge shift in microns applied to the input layer polygons. The value must be a positive number.

Description

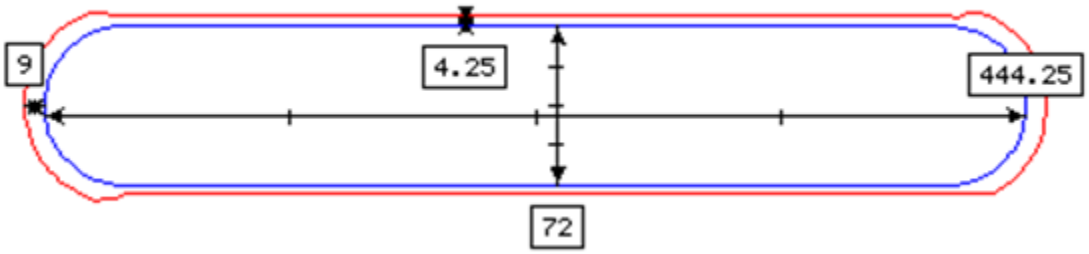
This optional command allows you to limit the resizing of input layer polygons by the “size_by percent” command in the Size job type.

Examples

The following example shows a job increasing the size of the input layer polygons by 6 percent and allowing a maximum edge shift of 9 nm:

```
job 1 size
size_by 6 percent
size_max_shift_edge 0.009
```

Figure 3-20. Polygon Resized by 6 Percent and Limited to 9 nm Maximum Edge Shift



Related Topics

job

size_by

straight_contour

Type: pxopc_options Keywords - any section

Reduces ripples on contours.

Usage

straight_contour ***weight***

Arguments

- ***weight***

  A required positive value that controls the relative importance of straight contours to overall optimization. The default is 0.

Description

The straight_contour command causes the jobs to analyze solutions for straightness in addition to minimizing the overall EPE. It can be useful when contours show unacceptable rippling, but should be used carefully as it can increase EPE. A good initial value is 1.

This command only affects output from the Open, Decorate, Correct, Refine, and Detail jobs. It can be specified in the main pxopc_options block, in which case the specified weight is applied in all applicable jobs, or inside a job section.

Examples

```
straight_contour 1
```

Related Topics

job

weight_layer

suppress_extra_printing

Type: pxopc_options Keywords - job sections only

Controls the area and aggressiveness for suppressing SRAF printing.

Usage

suppress_extra_printing *cond* [*cond*…] [pattern *index*] [weight *value*]

Arguments

- ***cond***

  A required argument specifying the name of at least one process window condition or process window condition group. The name must also be used as either the ***cond_name*** in a pw_condition statement or ***pw_group_name*** in a pw_condition_group statement as well as in a pw_select statement for the job.

- pattern *index*

  An optional argument for multi-patterning setup files. The index values are 0, 1, 2, or 3, and should also be defined on a layer statement. If this argument is not specified, all patterns are used.

- weight *value*

  An optional argument to define the aggressiveness of SRAF print suppression. The default value is 1.

Description

The optional suppress_extra_printing keyword controls the details of SRAF print suppression. By default, all jobs attempt to minimize SRAF printing, but balance it against other goals, such as improved EPE or depth of focus. Because of this, increase the weight by +1 each trial when tuning this command. The suppress_extra_printing keyword is supported in the following job types: Cdof, Correct, Refine, Finetune, and Detail.

When this command is specified, only the conditions listed are used for SRAF print suppression; the default print suppression algorithm is not used.

Examples

This example shows using suppress_extra_printing in a Cdof job:

```
job 7 cdof
   pw_select PWG outer
   suppress_extra_printing PWG outer weight 2
```

Related Topics

layer

vary_pw_group

Type: pxopc_options Keywords - EL and EL_Curvilinear job sections only

Determines which process window condition group to optimize. Required when two or more process window condition groups are defined.

Usage

vary_pw_group ***pw_group_name*** {**dose** | **focus**}

Arguments

- ***pw_group_name***

  A required argument specifying the name of the process window condition group to use. It must be defined with a pw_condition_group statement.

- **dose** | **focus**

  A required argument that specifies which variable to optimize.

  dose — Vary dose_latitude and dose_center. Generally used for EL and EL_Curvilinear jobs.

  focus — Vary focus_range.

  Specify either “dose” or “focus” but not both.

  If the job also includes variable settings such as pw_dose_latitude_factor or pw_focus_range_factor, ensure that the choice of dose or focus matches.

Description

This keyword is used primarily with the lithographic jobs EL and EL_Curvilinear. It identifies to the job which process window condition group should have its dose or focus optimized.

Note

This command initializes the focus and dose factors; place before any job-specific settings for dose latitude, dose center, and focus range.

Specifying vary_pw_group when a job has only one process window condition group selected causes the run to exit with this error:

```
ERROR: PXOPC: job 'cdof': option vary_pw_group is set if there are at
least two pw-condition groups selected
```

Although you can specify dose or focus, the keyword should be used only once per job.

Examples

The following example sets the EL job to optimize the dose of PWG2. PWG1 provides the focus_range.

```
job 7 EL
   pw_select PWG1 PWG2
   vary_pw_group PWG2 dose
```

weight_background

Type: pxopc_options Keywords - any section

Sets the default weight.

Usage

weight_background ***bgweight***

Arguments

- ***bgweight***

  A required non-negative value that sets the initial weight. The default is 1.

Description

Adjusts the default weight used by weight_layer. This statement may only appear once in the rule file.

Examples

```
weight_background 1.0
```

weight_layer

Type: pxopc_options Keywords - any section

Identifies polygon layers which define more important areas for optimization.

Usage

weight_layer ***layer operation weight***

Arguments

- ***layer***

  A required argument specifying the name of the layer containing weighting information. This layer must be declared as type **hidden** in the layer statement. Within a job, ***layer*** must be unique.

- ***operation***

  A required keyword indicating how the weighting is to be performed.

  add — The ***weight*** is added to the existing weight of the region.

  set — The ***weight*** replaces the existing weight of the region.

- ***weight***

  A required argument specifying the weighting associated with the polygons on ***layer***. Values can be negative, positive, or zero. Values between 10 and 200 provide acceptable results.

Description

Use the **weight_layer** command to mark certain target areas as more important to optimize than others. For example, to fix line-end pullback use a weight layer with rectangles drawn over the extent the lines must reach.

Tip

Only place weight layer rectangles over the edge sections that can print. Extending the rectangles to the target's corners causes longer run times because Calibre pxOPC attempts to match the mask more closely, and little improvement is possible.

Weight layer rectangles should be centered on the target; that is, the markers should extend equally on both sides of the edge as shown by the blue rectangles in Figure 3-21. The recommended width is 3 to 5 nm, with a length of 30 to 40 nm. If markers are too short they may not overlap contours with particularly bad EPE.

Figure 3-21. Weight Layer Marker

When **weight_layer** is in the general section, it applies to all jobs. **Weight_layer** statements in the job sections are combined for that job with any general statements. An options block may have any number of **weight_layer** statements but within a job, each ***layer*** must be unique.

When a region is covered by multiple **weight_layer**… **set** statements, the last **weight_layer**… **set** value is used.

The final weighted value of a region is never negative. If the accumulated weight is negative it is rounded up to 0.

Examples

```
weight_layer wt1 set 100
```

Related Topics

layer

weight_background

Chapter 4
Examples

This chapter provides examples for some common situations. Each example includes the SVRF lines that call the Calibre RET tools.

Setlayer Curve Example

The setlayer curve command is commonly used to create a smoothed target for post-tapeout operations.

The following lines of code are used with a 20 nm test at Siemens EDA. They are part of a larger file which also provides the typical SVRF specifications such as Layer, Precision, preliminary DRC processing, and so on. All three blocks are within a single file.

Example 4-1. SVRF Calls

```
px.smooth =  RET PXOPC px.target
               FILE px.curve.setup MAP px.smooth
px.mask   =  RET PXOPC px.target px.smooth px.weight
               FILE px.setup MAP px.mask
```

In the code in Example 4-1, here are some key things to note:

- SVRF statements do not require a line continuation character. Each of the RET PXOPC calls uses two lines.
- The px.smooth layer cannot run concurrently with px.mask. It uses a different setup file (Example 4-2) and passes different layers. The px.smooth layer is processed first.

Example 4-2. First LITHO FILE Statement (Curve Target)

```
LITHO FILE px.curve.setup [/*
   layer px.target  hidden +0.9985 +0.0085

   setlayer px.smooth = curve px.target cpdist 0.05 pts_per_cp 5 \
      maxdist 0.05 midpt 3 endpt_offset -0.004 order 6 scale1 1.5 \
      scale2 0.6  concave cpdist 0.060 maxdist 0.060 midpt 3 \
      midpt_offset 0.0011 endpt_offset -0.005
*/]
```

In the code in Example 4-2, notice how short the file is – yet it is a complete LITHO setup file. The only two commands are a layer declaration and the setlayer curve operation. It generates the target for the main pxOPC operation.

Example 4-3. Second LITHO FILE Statement (Main pxOPC)

```
LITHO FILE px.setup [/*
# These commands are in the main section.
   modelpath models/via20
   optical_model_load f+000 sym_f+0.0000
   optical_model_load f+050 sym_f+0.0500

   resist_model_load resist_mod e085a.mod
   background               +0.0215 -0.0085
   layer px.target  hidden  +0.9985 +0.0085
   layer px.smooth  hidden  +0.9985 +0.0085
   layer px.weight  hidden  +0.9985 +0.0085
   # for this layer command, see layer (for main setup file)

   tilemicrons 19 exact
   processing_mode flat

   pxopc_options px.options {
   # litho settings
      background                   +0.0215 -0.0085
      layer px.target  correction  +0.9985 +0.0085
      layer px.smooth  target      +0.9985 +0.0085
      layer px.weight  hidden      +0.9985 +0.0085

      # The first three are aerial models, and the second three CM1.
      pw_condition nominal     optical f+000 dose 1.000 \
            mask_size +0.0000 aerial 0.135
      pw_condition inner       optical f+050 dose 0.950 \
            mask_size -0.0005 aerial 0.135
      pw_condition outer       optical f+000 dose 1.050 \
            mask_size +0.0005 aerial 0.135
      pw_condition cm1_nominal optical f+000 dose 1.000 \
            mask_size +0.0000 resist resist_mod weight 1.0
      pw_condition cm1_inner   optical f+050 dose 0.950 \
            mask_size -0.0005 resist resist_mod weight 1.0
      pw_condition cm1_outer   optical f+000 dose 1.050 \
            mask_size +0.0005 resist resist_mod weight 1.0

      init px.target
```

```
        job 1 open
        pw_select nominal

        job 2 decorate
        pw_select nominal inner outer

        job 3 correct
        pw_select cm1_nominal cm1_inner cm1_outer

        job 4 refine
        pw_select cm1_nominal cm1_inner cm1_outer

        job 5 lmrc
        pw_select cm1_outer
        mrc_min_edge 0.025
        mrc_min_external 0.015
        mrc_min_internal 0.015
        mrc_align_edges all

        job 6 detail
        pw_select cm1_nominal cm1_inner cm1_outer
        mrc_min_edge 0.025
        mrc_min_external 0.015
        mrc_min_internal 0.015
        mrc_align_edges all

    //end of pxopc_options block
    }

setlayer px.mask   = pxopc px.target px.smooth px.weight \
                         MAP px.target OPTIONS px.options
*/]
```

Example 4-3 is a complete Litho setup file for a Calibre pxOPC run. Things to note include the following:

- Most settings for the jobs are left at their default values. In most cases, you should not need to vary values.
- Each job specifies particular process windows. Recommendations for which conditions to use are included in “Improving Calibre pxOPC Run Time” on page 53.

Calibre pxOPC With Calibre OPCverify Example

Foundries commonly combine different Calibre operations such as biasing, OPC, and ORC within a single SVRF rule file. The different litho setup files can be either external or included in the SVRF rule file with each in its own LITHO FILE statement.

The next three example blocks are embedded in one SVRF file, demonstrating one way multiple RET tools can use the same rule deck.

Example 4-4 invokes Calibre pxOPC. It uses a different format, DRC rule checks, for calling the tool and saving output to an OASIS file compared to Example 4-1 in "Setlayer Curve Example," which used derived layers.

Example 4-4. Alternate Form of SVRF Calls

```
//Exposure1 Debug Layers
px.job1 { RET PXOPC si_target dd.visible FILE pxopc_setup MAP px.job1   }
   DRC CHECK MAP px.job1   OASIS 31 0 "$ORC_OAS"

px.job2 { RET PXOPC si_target dd.visible FILE pxopc_setup MAP px.job2   }
   DRC CHECK MAP px.job2   OASIS 32 0 "$ORC_OAS"
...
px.job5 { RET PXOPC si_target dd.visible FILE pxopc_setup MAP px.job5   }
   DRC CHECK MAP px.job5   OASIS 35 0 "$ORC_OAS"

//OPCverify calls
cont.job1 { LITHO OPCVERIFY px.job1 FILE opcv_setup MAP cont.job }
   DRC CHECK MAP cont.job1   OASIS 31 1 "$ORC_OAS"
cont.job2 { LITHO OPCVERIFY px.job2 FILE opcv_setup MAP cont.job }
   DRC CHECK MAP cont.job2   OASIS 32 1 "$ORC_OAS"
...
cont.job5 { LITHO OPCVERIFY px.job5 FILE opcv_setup MAP cont.job }
    DRC CHECK MAP cont.job5   OASIS 35 1 "$ORC_OAS"
```

In the px.job rule checks, note that the pxOPC operations can run concurrently because they all supply the same set of layers to the same setup file (*pxopc_setup*, provided in Example 4-5). The cont.job rule checks, which call OPCverify, also can run concurrently. (The OPCverify setup file is in Example 4-6.)

Example 4-5. First LITHO FILE

```
LITHO FILE pxopc_setup [/*
   processing_mode flat
   tilemicrons 19 exact

   modelpath models/m200c
   optical_model_load nominal_opt si_f+0.0000
   optical_model_load inner_opt   si_f-0.0450
   optical_model_load outer_opt   si_f+0.0000
   resist_model_load resist_mod m200c_resist.mod

   background              +1.0000 +0.0000
   layer od.target  hidden -0.9785 -0.0085
   layer od.visible hidden -0.9785 -0.0085

   pxopc_options od.options {
      # litho settings
      background                  +1.0000 +0.0000
      layer od.target  correction -0.9785 -0.0085
      layer od.smooth  target     -0.9785 -0.0085
      layer od.visible hidden     -0.9785 -0.0085
```

```
        #Outer nominal + Underdose 4.3% (opcv = 3.3% + 1% extra underdose)
        #Inner f-45 (opcv = F-40 +10% larger)
        #Focus latitude  + Overdose  3.8% (opcv = 2.8% + 1% extra overdose)

        pw_condition nominal          optical nominal_opt dose 1.000 \
              mask_size +0.00000 aerial 0.194
        pw_condition inner            optical inner_opt   dose 1.038 \
              mask_size +0.00000 aerial 0.194
        pw_condition outer            optical outer_opt   dose 0.957 \
              mask_size +0.00000 aerial 0.194
        pw_condition cm1.nominal      optical nominal_opt dose 1.000 \
              mask_size +0.00000 resist resist_mod
        pw_condition cm1.inner        optical inner_opt   dose 1.038 \
              mask_size +0.00000 resist resist_mod
        pw_condition cm1.outer        optical outer_opt   dose 0.957 \
              mask_size +0.00000 resist resist_mod

        init od.target

        job 1 open
        pw_select nominal

        job 2 decorate
        pw_select nominal inner outer

        job 3 correct
        pw_select cm1.nominal cm1.inner cm1.outer

        job 4 refine
        pw_select cm1.nominal cm1.inner cm1.outer

        job 5 lmrc
        pw_select cm1.outer
        mrc_min_edge 0.015
        mrc_min_internal 0.015
        mrc_min_external 0.015

        job 6 detail
        pw_select cm1.nominal cm1.inner cm1.outer
        mrc_min_edge 0.015
        mrc_min_internal 0.015
        mrc_min_external 0.015

    #end of pxopc_options block
    }

# construct the smoothed target the same way it is done in nmOPC recipes
    setlayer od.smooth1 = curve od.target order 6 cpdist 0.050 \
                          smooth 0.05 maxdist 0.040 midpt 3
    setlayer od.smooth2 = curve od.target order 6 cpdist 0.060 \
                          smooth 0.05 maxdist 0.060 midpt 3
    setlayer od.smooth = or od.smooth1 od.smooth2

    setlayer od.out    = pxopc od.target od.smooth od.visible \
                         MAP od.target OPTIONS od.options
```

```
   setlayer px.job1   = pxopc od.target od.smooth od.visible \
                        MAP od.target OPTIONS od.options LASTJOB 1
   setlayer px.job2   = pxopc od.target od.smooth od.visible \
                        MAP od.target OPTIONS od.options LASTJOB 2
   setlayer px.job3   = pxopc od.target od.smooth od.visible \
                        MAP od.target OPTIONS od.options LASTJOB 3
   setlayer px.job4   = pxopc od.target od.smooth od.visible \
                        MAP od.target OPTIONS od.options LASTJOB 4
   setlayer px.job5   = pxopc od.target od.smooth od.visible \
                        MAP od.target OPTIONS od.options LASTJOB 5

//end of LITHO FILE statement
*/]
```

In Example 4-5, notice the names of the derived layers for the setlayer pxopc calls. The layers are named "px.job*N*". This is the same layer name used with the MAP keyword of the RET PXOPC statements in the SVRF file (Example 4-4). It is important that these two layers have the same name, including case, or the output will not be passed from Calibre pxOPC to the results database (RDB). Empty layers in the output are a symptom that the layer names did not match.

The name "px.job*N*" is also used to name the rule checks, but this is not required to match anything in the LITHO setup file.

Example 4-6. LITHO FILE for OPCverify

```
LITHO FILE opcv_setup [/*
   modelpath models/m200c
   optical_model_load nominal_opt si_f+0.0000
   resist_model_load  resist_mod  m200c_resist.mod
   tilemicrons 20

   background           +1.0000 +0.0000
   layer px.job  hidden -0.9785 -0.0085

   setlayer cont.job = image mask0 optical  nominal_opt background  1 0 \
      layer px.job -0.9785 -0.0085 dose 1.000 size +0.0000 \
      resist_model resist_mod
*/]
```

SRAF Template Example

Calibre pxOPC can be used in place of Calibre® PixBAR™ to create templates for use in Calibre® nmSRAF™.

Note

The recipe shown in the example follows best practices for Calibre nmSRAF templates but is generic. Better results can be achieved by tuning the recipe for specific cases.

In this example, Calibre pxOPC is used to generate SRAFs around a target polygon. The output is separated into SRAFs and main features for use in a later run by Calibre nmSRAF. Because the run is producing a template rather than a manufacturable mask, the recipe does not include the LMRC or Refine jobs.

This example also demonstrates a consistent set of commands in litho model format.

Example 4-7. SVRF and Setup File for SRAFs

```
// Standard SVRF setup
LAYOUT SYSTEM OASIS
LAYOUT PATH   "via_test_pattern.oas"

LAYOUT PRIMARY "*"
LAYOUT ALLOW DUPLICATE CELL YES
DRC MAXIMUM RESULTS all
PRECISION 10000
DRC RESULTS DATABASE "sraf_out.oas" oasis USER MERGED
DRC SUMMARY REPORT out_report

// The drawn target for OPC
LAYER target 0
target { COPY target } DRC CHECK MAP target 1604

// Run Calibre pxOPC
pxopc_results = RET PXOPC target FILE pxopc_setup MAP pxopc_results
pxopc_results {COPY pxopc_results } DRC CHECK MAP pxopc_results 92

// Split the pxOPC results into main feature and SRAF layers
pxopc.main = pxopc_results INTERACT target
pxopc.sraf = pxopc_results NOT INTERACT target

pxopc_opc {COPY pxopc.main } DRC CHECK map pxopc_opc 80
pxopc_sraf {COPY pxopc.sraf } DRC CHECK map pxopc_sraf 81

LITHO FILE pxopc_setup [/*

   modelpath ./models
   # This example uses a litho model model directory, so it does not
   # separately load optical, resist, or ddm models.

   layer target visible atten 0.06

   processing_mode flat
   progress_meter on

   tilemicrons 10

   pxopc_options pxopc_opts {
      debug_level 0

      # These use litho-model format
      layer target correction     mask_layer 0
      layer curve  target         mask_layer 0
```

```
      pw_condition nominal  ddm.models
      pw_condition inner focus 30nm dose 1.02 bias -0.00025
      pw_condition outer            dose 0.98 bias  0.00025

      init target

      job 1 open
         pw_select nominal
         rate 0.01
         iterations 8

      job 2 decorate
         scatter_belt 0.275
         iterations 6

      job 3 correct
         assist_outside_dose_multiplier 0.85
         iterations 8

      job 4 detail
         pw_select nominal inner outer
         assist_outside_dose_multiplier 0.85
         iterations 10
   }

   # Smooth the target, then use in OPC
   setlayer curve         = curve_target target criticalDistance 0.04 \
                                         cornerRadius 0.03
   setlayer pxopc_results = pxopc target curve MAP target \
                                         OPTIONS pxopc_opts
*/]
```

Double Patterning Example

This example shows a setup file you might use to start optimizing a recipe for double patterning. There are two target layers, and different optical and DDM models for each.

This setup file uses standard syntax instead of litho model syntax.

Example 4-8. Setup File for Multi-Patterning

```
modelpath ./Models_per_pattern
optical_model_load  OPTIC_blue       DDM_mask0
optical_model_load  OPTIC_n30_blue   DDM_n30_mask0
optical_model_load  OPTIC_green      DDM_mask1
optical_model_load  OPTIC_n30_green  DDM_n30_mask1

resist_model_load   RESIST           MF22_DS25.mod
ddm_model_load      ddm1_blue        bright_mask0.ddm
ddm_model_load      ddm1_green       bright_mask1.ddm
```

```
tilemicrons 19 exact
background clear

processing_mode flat

#Two curved targets, and two drawn targets.
layer ctarget0 hidden atten 0.06
layer ctarget1 hidden atten 0.06
layer target0 hidden atten 0.06
layer target1 hidden atten 0.06

pxopc_options opts {
   version 1

# The pattern keyword defines which layers go to blue (0) or green (1)
# mask.
   layer ctarget0 TARGET atten 0.06 pattern 0
   layer ctarget1 TARGET atten 0.06 pattern 1
   layer target0 CORRECTION atten 0.06 pattern 0
   layer target1 CORRECTION atten 0.06 pattern 1

# Process window conditions for first mask
   pw_condition nominal optical OPTIC_blue dose 1 resist RESIST
   pw_condition outer_blue optical OPTIC_blue dose 0.94 resist RESIST
   pw_condition inner_blue optical OPTIC_blue dose 1.06 resist RESIST

# Process window conditions for second mask
   pw_condition nominal_green optical OPTIC_green dose 1 resist RESIST
   pw_condition outer_green optical OPTIC_green dose 0.94 resist RESIST
   pw_condition inner_green optical OPTIC_green dose 1.06 resist RESIST

   background  clear

# Two layers for initial target
   init target0 target1

# Blue models are used for target0 and green for target1
   ddm ddm1_blue ddm1_green

# Each job selects conditions for both masks
   job 1 open
      pw_select nominal pattern 0
      pw_select nominal_green pattern 1

   job 2 decorate
      pw_select nominal inner_blue outer_blue pattern 0
      pw_select nominal_green inner_green outer_green pattern 1

   job 3 correct
      pw_select nominal inner_blue outer_blue pattern 0
      pw_select nominal_green inner_green outer_green pattern 1

   job 4 refine
      pw_select nominal inner_blue outer_blue pattern 0
      pw_select nominal_green inner_green outer_green pattern 1
```

```
   job 5 lmrc
      pw_select outer_blue pattern 0
      pw_select outer_green pattern 1

   job 6 detail
      pw_select nominal pattern 0
      pw_select nominal_green pattern 1

      pw_bridging nominal pattern 0 nominal_green pattern 1 \
         width 0.0298 weight 10
      pw_pinching nominal pattern 0 nominal_green pattern 1 \
         width 0.05 weight 10
      pw_bridging nominal pattern 0 width 0.0485 weight 10
      pw_pinching nominal pattern 0 width 0.0388 weight 10
      pw_bridging nominal_green pattern 1 width 0.0485 weight 10
      pw_pinching nominal_green pattern 1 width 0.0388 weight 10
  }

# Setlayer pxopc needs to be called once for each mask (blue and green).
setlayer pxopc_out0 = pxopc ctarget0 ctarget1 target0 target1 \
      MAP target0 OPTIONS opts
setlayer pxopc_out1 = pxopc ctarget0 ctarget1 target0 target1 \
      MAP target1 OPTIONS opts
]
```

EUV Example

EUV Calibre setups are similar to DUV setups, requiring only a few extra commands and an EUV license.

You will need EUV optical models and typically a DDM library as well. Most EUV processes also model long-range effects such as flare and sometimes through-slit. You can generate the flare model yourself using RET FLARE_CONVOLVE, but the other files are independent of the design and provided by the foundry.

One of the primary differences is the run mode itself. EUV causes severe loss of hierarchy, so use ULTRAflex. For comparable processing time to a DUV setup, use more memory on the remote CPUs.

The following example SVRF file with embedded litho setup files highlights the lines required for EUV with bold font.

Example 4-9. SVRF and Litho Setup File for EUV

```
LAYOUT SYSTEM OASIS
LAYOUT PATH "input/test_pattern.oas"
LAYOUT PRIMARY "*"
PRECISION 20000

LAYOUT ULTRA FLEX YES
```

```
DRC MAXIMUM RESULTS  ALL
DRC MAXIMUM VERTEX  199
DRC RESULTS DATABASE "output/test_pattern_results.oas" OASIS
DRC SUMMARY REPORT  "output/test_pattern.rep"

LAYER target 1501
target { COPY target } DRC CHECK MAP target 1

//Create curved target layer and copy to results
trgtsmth =  LITHO OPCVERIFY FILE "smooth.in" target MAP curve
trgtsmth { copy trgtsmth } drc check map trgtsmth 2

LITHO FILE "smooth.in" [
    layer target
    setlayer curve = curve_target target criticalDistance 0.03 \
                         cornerRadius 0.009
]

//Call Calibre pxOPC and copy final output to results
pxopc.full = RET EUV PXOPC target trgtsmth FILE pxopc.full.setup MAP pxopc
pxopc.full {COPY pxopc.full } DRC CHECK map pxopc.full 10

//Because this is a test setup, also copy tiles to results.
pxopc.full.tiles = RET EUV PXOPC target trgtsmth FILE pxopc.full.setup
                                                    MAP pxopc.tiles
pxopc.full.tiles {COPY pxopc.full.tiles} DRC CHECK map pxopc.full.tiles 11

LITHO FILE pxopc.full.setup [/*
   # This example uses a litho model and litho model syntax.

   modelpath /models:./models
   flare_longrange LITHOMODEL_ALLDDM png ./models/flare.png
   euv_slit_x_center 0.0

   layer target
   layer trgtsmth hidden

   progress_meter on
   processing_mode flat
   tilemicrons 19 exact

   pxopc_options px.options {
      version 1
      debug_level 0

      layer target  correction mask_layer 0
      layer trgtsmth target     mask_layer 0

      pw_condition nominal LITHOMODEL_ALLDDM
      pw_condition pw1     LITHOMODEL_ALLDDM

      init target

      job 1 open
             pw_select nominal
```

```
        job 2 decorate

        job 3 correct

        # The other jobs used defaults, but EUV processes typically
        # have smaller MRC settings than the default.
        job 4 lmrc
              mrc_min_edge 0.007
              mrc_min_internal 0.007
              mrc_min_external 0.007

        job 5 detail
              pw_select nominal
              mrc_min_edge     0.007
              mrc_min_internal 0.007
              mrc_min_external  0.007
     }

    setlayer pxopc = pxopc target trgtsmth MAP target OPTIONS px.options

    setlayer pxopc.tiles = tilegen -0.0005 property {
         location
         remote
       }
*/]
```

Related Topics

ULTRAflex Data Construction [Calibre Post-Tapeout Flow User's Manual]

RET FLARE_CONVOLVE [Calibre nmOPC User's and Reference Manual]

euv_slit_x_center

flare_longrange

Common Depth of Focus Example

CDOF jobs can adjust a mask to improve the range of focus conditions at which the design is printed properly. They do not change the source or the optical model.

When the setup calls a CDOF job, the supporting models must contain precomputed defocus models and other supporting models packaged as a litho model. Some commands such as “layer” have different syntax for traditional and litho models. All commands must use litho model syntax.

The first part of the example shows constructing a tolerance band in the SVRF using a simple SIZE operation.

Example 4-10. SVRF for Creating a Tolerance Band

```
//The target and targetSmooth layers are already in the layout.
LAYER target       1
LAYER targetSmooth 2

//The inner and outer tolerance bands are constructed by sizing the
//smoothed target.
inner   = SIZE targetSmooth BY -0.004
outer   = SIZE targetSmooth BY  0.004
xtarget = SIZE target       BY  0.020

//Not shown: lines copying the layers to the output GDS and DRC Check Map

pxopc_out =
   RET PXOPC FILE "pxopc.in" targetSmooth target inner outer xtarget
   MAP pxopc_out
opcv_out =
   LITHO OPCVERIFY FILE "opcverify.in" target pxopc_out
   MAP nominal
```

The *pxopc.in* litho setup file uses litho model syntax throughout.

Example 4-11. Setup File Using CDOF Job

```
modelpath       ./models
tilemicrons  19 exact

layer targetSmooth hidden clear
layer target       hidden clear
layer inner        hidden clear
layer outer        hidden clear
layer xtarget      hidden clear

pxopc_options pxopc_opts {
   version       1

# There are four process window conditions, and one
#     process window condition group
   pw_condition nominal lm                          # loads litho model
   pw_condition PW1              dose 1.05 aerial 0.15
   pw_condition PW10 focus 100nm dose 0.95 aerial 0.15
   pw_condition PW_SPRINT        dose 1.1 aerial 0.15
   pw_condition_group PWG        dose_latitude 0.05

# These are the layer definitions that get used.
# inner and outer are the tolerance band.
   layer targetSmooth TARGET      mask_layer 0
   layer target       CORRECTION  mask_layer 0
   layer inner        HIDDEN
   layer outer        HIDDEN
   layer xtarget      HIDDEN

   init                    target
```

```
   job 1 open
      pw_select NOMINAL PW10
   job 2 decorate
      pw_select NOMINAL PW1 PW10
   job 3 correct
      pw_select NOMINAL PW1 PW10
      iterations 20
   job 4 refine
      pw_select NOMINAL PW1 PW10
      iterations 20
   job 5 lmrc
      pw_select PW_SPRINT
   job 6 detail
      pw_select NOMINAL PW1 PW10
      iterations 20

   job 7 cdof
# Specify the process windows to optimize focus for
      pw_select PW_SPRINT PWG
# Extend the focus range possibilities to +/- 120 nm
      pw_focus_base_points -120 -100 -80 -60 -40 0 40 60 80 100 120
# Specify the tolerance bands created in the SVRF
      pw_tolerance_bound PWG outside inner
      pw_tolerance_bound PWG inside  outer
}

setlayer pxopc_out = pxopc targetSmooth target inner outer xtarget \
                     MAP target OPTIONS pxopc_opts
]
```

Chapter 5
Calibre pxOPC Best Practices

Calibre pxOPC has been tuned so that in most cases only small customizations are needed. The best practice recommendations are built around typical sequences of jobs.

In each job, the Calibre pxOPC engine solves a particular mask optimization problem. Mask optimization is sensitive to initial conditions and solutions are not unique. This means any mask can be improved, even if only by a small amount. It is important, therefore, to establish the success criteria before beginning optimization.

General Recommendations

Calibre pxOPC can be run either standalone (for example, when developing an optimization recipe or SRAF template) or as part of a larger optimization flow (for example, when called from Calibre LPE). The best practices have been tested with both types of flow.

Table 5-1. Best Practices for All Calibre pxOPC Runs

Practice	Reason
For DUV, use a tilemicrons size of 19 or 20, and litho flat processing mode. (For other cases, see “tilemicrons” on page 108.)	Keeps memory consumption under 4 GB per CPU when processing large areas, unless models are complicated.
Use a smoothed target layer.	Rectilinear targets take longer to converge, increasing run time.
Define process window conditions (dose, defocus) to be 10% more aggressive than Calibre OPCverify process window conditions.	Reduces the likelihood of a soft pinch or soft bridge. Prevents SRAFs from printing.
Use litho models instead of separate optical, resist, and DDM models.	Ease of maintenance. Some command features are only available with litho models.
Process window constraints (pinching, bridging, MRC) should be *slightly* more aggressive than verification constraints.	Calibre pxOPC tries to keep contours in the vicinity of the target. Constraints that are too aggressive not only take longer but can result in larger EPE.

Table 5-1. Best Practices for All Calibre pxOPC Runs (cont.)

Practice	Reason
To improve Calibre pxOPC run time when using pw_bridging and pw_pinching, keep the constraints to less than 60 nm.	Larger values require more iterations to satisfy all objectives. Testing has shown that 60 nm provides a good balance between run time and quality.
If the Detail job does not satisfy process window constraints, make sure the Lmrc job does not show hard bridging or very bad pinching. If also using pw_enclose, the Lmrc output should keep the enclosed layer fully covered by the target.	Improving the input allows the Detail job to converge faster.

Related Topics

tilemicrons

processing_mode

Litho Flat Processing Mode [Calibre Post-Tapeout Flow User's Manual]

setlayer curve

pw_condition

pw_enclose

Job Contracts

For a correctly optimized mask, every job must deliver certain measurable results (the contract). A bad result can be traced back to the first job that does not fulfill its contract.

Table 5-2. Contracts Summary

Job	Contract
Open	• Printed shapes at nominal conditions. • Contacts deviate no more than 30% and lines no more than 40% from the target critical dimension (CD). If the contract is not met, see “Fixing Open Job Problems” on page 38.
Decorate Decoratedarksraf	• At least two rows of assist features are generated. • SRAFs are robust and in dark areas. • Main features print, and no more than 25% of SRAFs print. If the contract is not met, see “Fixing Decorate Job Problems” on page 39.
Unclose	This job should only be used if Decorate is not meeting its contract and using negative SRAFs. • Main features print, though holes in the contours are allowed because negative SRAFs are present. • Printing shapes are on target within 40% of target CD.
Correct	• Printing shapes are on target within 8% of CD. • Printing SRAFs from Decorate input have been mostly eliminated. If the contract is not met, see “Fixing Correct Job Problems” on page 42.
CDOF_Curvilinear EL_Curvilinear	• The best possible common depth of focus or dose latitude. • No extra printing occurs. • No catastrophic failures on main features.
Refine Finetune (LPE)	• No SRAFs print. • Areas marked with weight layers are printing on target within fractions of a nanometer. If the contract is not met, see “Fixing Refine Job Problems” on page 44
LMRC Detail	• The output mask is strictly Manhattan. • No extra printing occurs. • MRC rules are obeyed. If the contract is not met, see “Fixing LMRC Job Problems” on page 46 or “Fixing Detail Job Problems” on page 47.

Table 5-2. Contracts Summary (cont.)

Job	Contract
Detailhybrid Detailrectsraf	Same as Detail job contract except the output mask. • The Detailhybrid output mask has curvilinear main features with Manhattan SRAFs. • The Detailrectsraf output mask has Manhattan main features and rectangular SRAFs.
CDOF EL	• The best possible common depth of focus or dose latitude. • The output mask is strictly Manhattan. • No extra printing occurs. • No catastrophic failures on main features. • MRC rules are obeyed. If the contract is not met, see "Fixing CDOF Job Problems" on page 49 or "Fixing EL Job Problems" on page 51.

To debug results, check intermediate layers to identify the first job that does not deliver the required output. (See the "Fixing…" sections in "Calibre pxOPC Configuration and Tuning" for step-by-step debugging instructions.)

The sections that follow summarize recommendations covered in the procedures mentioned in Table 5-2.

Techniques to Control Extra Printing

Extra printing happens when assist features such as SRAFs that are not meant to appear on the wafer generate a contour. The method to control extra printing depends on the associated process window condition.

Table 5-3. Methods to Control Extra Printing

Printing Condition	Recommended Change
Exactly the same as bridging condition	None.

Table 5-3. Methods to Control Extra Printing (cont.)

Printing Condition	Recommended Change

Different in dose from the bridging condition	Add the SRAF printing condition to the recipe:
	1. Divide the dose used by the Calibre OPCverify check by the dose used by the Calibre pxOPC bridging condition.
	2. Round this value up, and use as the value for assist_outside_dose_multiplier in the Correct, Refine, Finetune, and Detail jobs.
	3. Add a new process window condition similar to the bridging condition except for the dose and add to the LMRC job:
	`pw_condition lmrc_bridge dose 1.05 bias 0.001`
	`...`
	`job 5 lmrc`
	`pw_select lmrc_bridge`

Different resist model, mask sizing, or defocus	Add a condition specifically for the SRAF area:
	1. Create a new layer with larger target shapes. In the SVRF portion, this might be
	`t_large = SIZE target BY 30`
	2. Add the new layer to the RET PXOPC statement and to the litho setup file as a hidden layer.
	3. Use the layer to create a new pw_condition. Use the outside keyword to make it active only in the SRAF area.
	`pw_condition sraf_printing outside t_large`
	If this has already been done but problems persist, add additional settings to make the new condition slightly more aggressive than the one used in verification.
	4. Add the new condition to the pw_select statement in the Correct, Refine, Finetune, Detail, and LMRC jobs.
	Alternatively, in the Correct, Refine, and Finetune jobs, use assist_outside_dose_multiplier. After tuning the value, this may provide better performance.

Related Topics

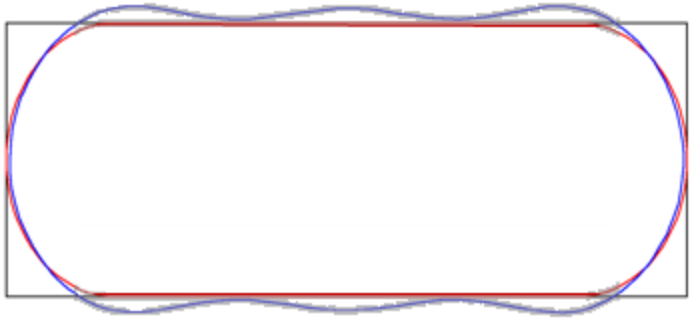
assist_inside_dose_multiplier

pw_condition

Techniques to Reduce Ripples

Ripples, or oscillating contours, sometimes appear when too much emphasis is placed on reducing corner rounding. The smoothed target has straight edges and round corners, but the printing contour has wavy edges and looser corners.

Figure 5-1. Oscillating Contours



The following recommendations are ordered from simplest to most effort. They can safely be combined.

- **Weighted Markers** — Weighted markers use shapes on a weight layer to emphasize meeting EPE on the straight edges over reaching corners. The marker shapes should only be placed on problematic edges, and not extend to where the smoothed target begins to curve.
- **Curve Tuning** — For polygons with concave corners, adjust the parameters in setlayer curve or setlayer curve_target such that the smoothed target is closer to the printing contour. This typically can be done for setlayer curve by reducing midpt_offset or endpt_offset values.
- **Straight Objective** — The straight_contour option instructs Calibre pxOPC to include contour straightness in evaluating solutions. It effects the Correct, Refine, and Detail jobs. Because it adds an extra objective to check, it may increase run time. Additionally, high values can increase EPE, so it must be tuned carefully.

Related Topics

setlayer curve

setlayer curve_target

straight_contour

weight_layer

Index

— E —

— F —

— G —

— H —

— I —

— J —

— L —

— M —

— N —

— O —

— P —

— Q —

— R —

— S —

— T —

— U —

— V —

— W —

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.