SIEMENS EDA

Calibre® LSG for Synthetic Layout Generation User's Manual

Software Version 2021.2
Document Revision 2

SIEMENS

Unpublished work. © 2021 Siemens

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

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

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

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

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

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

Table of Contents

Chapter 4

List of Figures

List of Tables

Revision History

Revision	Changes	Status/ Date
2	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released April 2021
1	Modifications to improve the readability and comprehension of the content. Approved by Lucille Woo. All technical enhancements, changes, and fixes listed in the *Calibre Release Notes* for this products are reflected in this document. Approved by Michael Buehler.	Released February 2021

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Siemens EDA documentation source. For specific topic authors, contact the Siemens Digital Industries Software documentation department.

Revision History: Released documents maintain a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation which are available on https://support.sw.siemens.com/.

Chapter 1
Introduction to Calibre LSG

Calibre® LSG Synthetic Layout Generator (Calibre LSG) is a software tool used in conjunction with Calibre® WORKbench™ to create user-defined pattern definitions for generated random layout clips. These layout clips are suitable for pattern analysis and other early design phase applications for specific layers and technologies.

An overview of the purpose and the workflow is provided, followed by the requirements, the modes of operation, and the basic concepts for using this software tool.

Overview of Calibre LSG

The Calibre LSG tool creates realistic design-like layout clips that can be used for the initial creation and analysis of pattern libraries and early design phase applications and flows.

Detection of possible layout litho hotspots from problematic patterns is essential early in the design cycle, when actual design layout samples may not be available. Calibre LSG enables the early analysis of challenging patterns by generating random layout clips from user specified input such as design pitch, pattern shape definitions, and rules.

The generated layout clips can be used by Computer Aided Design (CAD) groups as design samples for the development of early pattern libraries for specific technologies and layers that require critical shapes analysis. While the output layout clips are not guaranteed to be 100% DRC clean, they provide meaningful information about potentially problematic patterns that can result in litho hotspots. Full DRC checking is not part of this functionality, but other tools and flows can also be used to run checks on the sample layouts generated by Calibre LSG. Applications for using the generated layout clips include and are not limited to the following:

- Early pattern library development and optimization
- OPC and RET recipe optimization
- Test chips, patterns for new technologies, layers, and nodes

- DRC kit testing and optimization

To further enhance the pattern information in the generated layout clips, the Calibre LSG tool has several definitions, rule formats, and command options. These options include user-defined layout clip definitions, clip template generation, and custom unit pattern creation.

Calibre LSG functionality supports optional flows for post-layout DRC fixing (colored and non-colored patterns), systematic pattern generation, via shape analysis, and standard cells generation (colored and non-colored cells). In addition, command options and rules for off-grid (ungrid) pattern variation provide solutions for performing realistic critical pattern analysis using Calibre LSG generated layout clips.

Note

See your Siemens EDA representative for more information on the available Calibre LSG flow and command options.

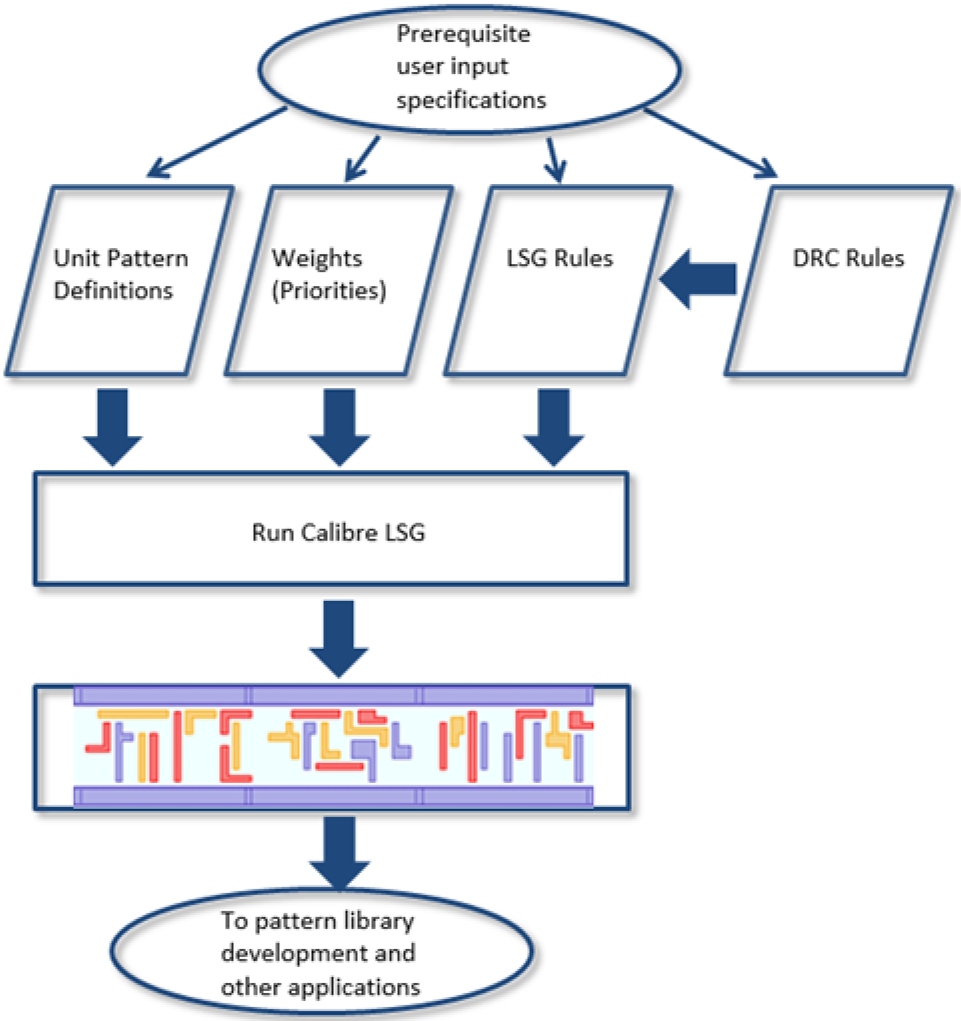
Calibre LSG Workflow

Calibre LSG fits into the early design phase flow by providing CAD engineers with a method for initial pattern library development and other applications.

The following figure shows the basic Calibre LSG flow.

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

Figure 1-1. Calibre LSG Workflow


Prerequisite user input specifications
Unit Pattern Definitions
Weights (Priorities)
LSG Rules
DRC Rules
Run Calibre LSG
To pattern library development and other applications

Calibre LSG Requirements

To run the Calibre LSG flow, you must provide user-defined pattern input information (for non-default pattern configuration), required licenses, and any necessary environment variable settings—see your Siemens EDA representative for specific details.

The requirements for performing the basic Calibre LSG flow are as follows:

- **User-defined input** — Specify your user-defined input configuration files.

  Note
  Default and random configurations apply if user-defined input files are unspecified.

  - Unit pattern definitions
  - Weights and priorities for pattern definitions

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

    - Calibre LSG rules
    - DRC rules and statements
    - Additional rule files as needed for Calibre LSG optional flows

- **Licensing** — See your Siemens EDA representative for licensing information.
- **Environment variables** — Set the necessary environment variables.
    - Set the CALIBRE_HOME or MGC_HOME environment variable to the path of your Calibre software tree.

Syntax Conventions

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

The following table identifies the syntax conventions used for Calibre LSG.

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.
<u>Underline</u>	Underlining indicates either the default argument or the default value of an argument.
UPPercase	For certain case-insensitive commands, uppercase indicates the minimum keyword characters. In most cases, you may omit the lowercase letters and abbreviate the keyword.
[ ]	Brackets enclose optional arguments. Do not include the brackets when entering the command unless they are quoted.
{ }	Braces enclose arguments to show grouping. Do not include the braces when entering the command unless they are quoted.
‘ ’	Quotes enclose metacharacters that are to be entered literally. Do not include single quotes when entering braces or brackets in a command.
| or ||	Vertical bars indicate a choice between items. Do not include the bars when entering the command.
…	Three dots (an ellipsis) follows an argument or group of arguments that may appear more than once. Do not include the ellipsis when entering the command.

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

Table 1-1. Syntax Conventions (cont.)

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

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

Calibre LSG Run Modes

Calibre LSG software is performed by the Calibre® hierarchical engine. You can invoke and run Calibre LSG from Calibre WORKbench in the graphical user interface (GUI) mode, or from a Tcl shell call, or from a command line shell.

- **Calibre WORKbench GUI** — You can use the Calibre LSG GUI in Calibre WORKbench from the **Litho** menu item **LSG Layout Generation** to interactively specify pattern input information and then run Calibre LSG.
- **Calibre WORKbench Tcl shell call** — Alternatively, you can also execute a Tcl shell call from the Calibre WORKbench command line to enable the same Calibre LSG GUI functionality.
- **Linux command-line** — You can invoke Calibre LSG functionality from a Linux[1] command line and specify a Calibre LSG options file. See "calibre -lsg" on page 26.

Invoking Calibre LSG from the GUI

You can invoke the Calibre LSG GUI from the Calibre WORKbench **Litho** menu.

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.

Procedure

1. Invoke the Calibre WORKbench GUI by specifying the executable name on the command line.

   ```
   calibrewb
   ```

2. Choose the **Litho** > **LSG Layout Generation** menu item.

Results

The Calibre LSG GUI is displayed in the Calibre LSG Layout Generation window.

Invoking Calibre LSG from a Tcl Shell Call

You can invoke the Calibre LSG GUI with a Tcl shell call from Calibre WORKbench.

[1] Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

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

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.

Procedure

Invoke Calibre WORKbench and the Tcl interpreter with either of the following methods:

a. Use the Calibre WORKbench executable with the *tclsh* executable to start Calibre WORKbench with the Tcl command line interactive interpreter.

```
calibrewb tclsh
```

Specify the command line arguments to open the Calibre LSG Layout Generation window.

```
dpe gui
```

b. Use the Calibre WORKbench executable with the -s option to designate a script file.

```
calibrewb -s scriptfile
```

Specify the arguments to open the Calibre LSG Layout Generation window in a script file, for example:

```
#!/bin/sh
#\
exec tclsh "$@" "$0"
    dpe gui
```

Results

The Calibre LSG GUI is displayed in the Calibre LSG Layout Generation window.

Invoking Calibre LSG from the Command Line

You can invoke Calibre LSG batch functionality from a Linux command line shell and specify your arguments in an LSG options file.

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.
- Prepare an input LSG options file that contains the argument options used by Calibre LSG to configure and generate the layout clips. See "calibre -lsg" on page 26 for information on how to specify these arguments.

Procedure

1. Invoke Calibre with LSG and specify your LSG options file from a Linux command line as follows:

```
calibre -lsg -turbo number_of_processors lsg_options_file
```

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

Specifying the -turbo option enables multithreaded parallel processing. See "Calibre Command Line" in the *Calibre Administrator's Guide* for more information.

2. Review the output layout clips using your layout browser of choice.

Results

Calibre LSG generates the configured layout clips to the specified output directory. You can re-configure the generated layout clips as needed by editing the arguments in your LSG options file.

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

Calibre LSG Concepts

Certain key concepts are important for understanding the input specifications for using Calibre LSG. These concepts include unit pattern definitions, weights (priorities), and rules used to generate output layout clips that are representative of the pattern sensitivities of your design.

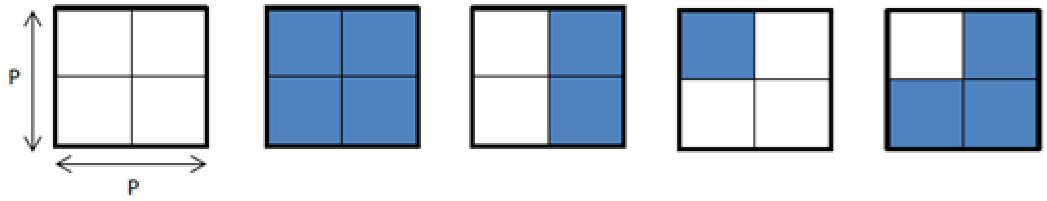
Calibre LSG Unit Patterns

Calibre LSG uses specific input pattern definitions that are combined into layout clips for early pattern library generation and other early design phase analysis.

The unit pattern (or unit cell) is the smallest defined pattern of a layout based on the minimum design pitch (P). The unit pattern is a square with sides equal to the minimum design pitch of the design technology. The minimum design pitch is specified as a user input to the Calibre LSG run.

There are five basic unit patterns defined, with each having four possible rotations of 90 degrees (0 (no rotation), 90, 180, 270).

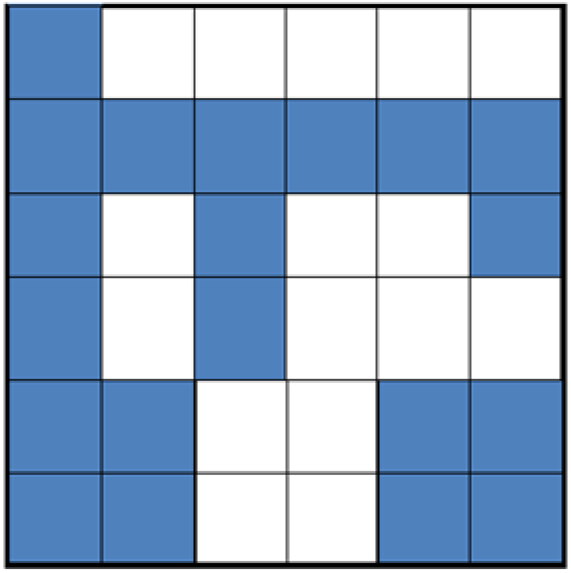
Figure 1-2. Basic Unit Patterns



Unit patterns can be used to create larger user-defined patterns that represent actual design layout shapes. The unit patterns are the building blocks of the Calibre LSG layout clips. An evaluation layout clip is an *N* x *N* frame of unit patterns generated by a selection of predefined unit patterns placed in a grid. *N* = 3 in the layout evaluation clip shown in Figure 1-3.

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

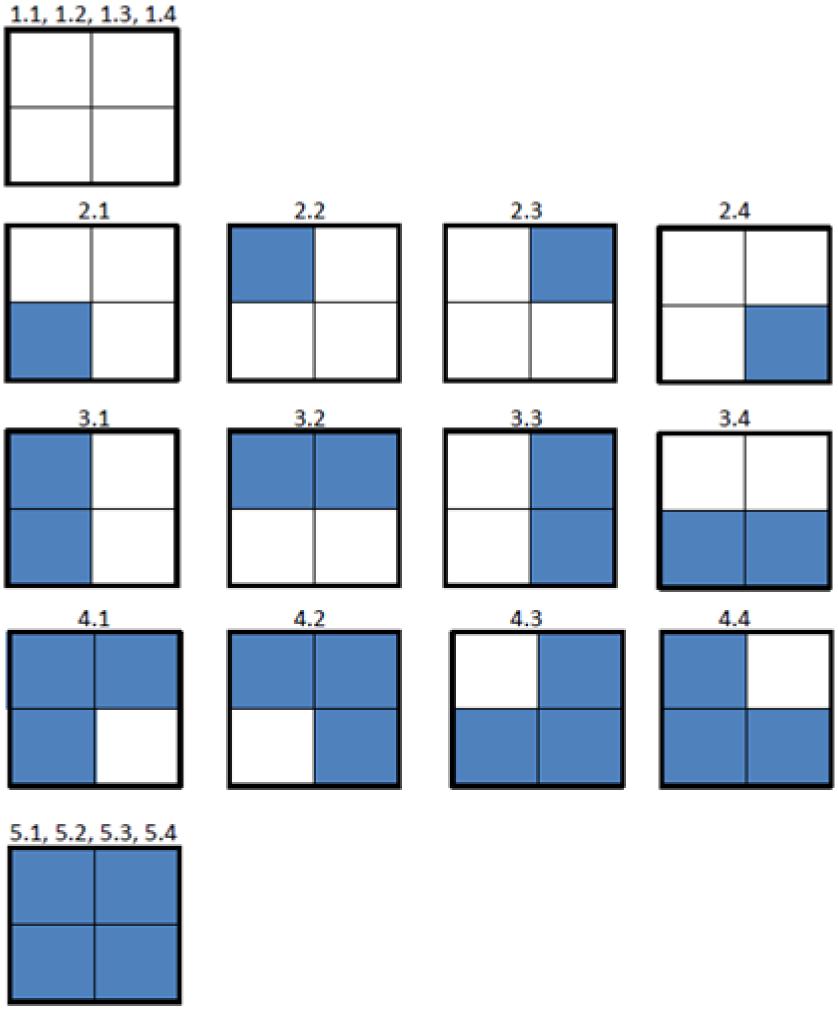
Figure 1-3. *N* = 3 Evaluation Clip



Calibre LSG allows pattern customization through the input of unit patterns that are defined by pitch and coordinate points. Figure 1-4 shows the five basic unit patterns and the four possible rotations. Each unit pattern has a side equal to P. Unit patterns are specified as “*p_id.r*”, where “*p_id*” is the base unit pattern identifier, and “*r*” is the rotation index, where *r* is 1, 2, 3, or 4, corresponding to 0, 90, 180, or 270 degrees of rotation.

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

Figure 1-4. Defined Unit Patterns



Selectivity of the unit patterns and rotations can be done by assigning optional weights and priority to specific unit patterns and rotations.

Calibre LSG Weights

Weights can be applied to influence and prioritize the selection of unit patterns that make up the layout clips. A Calibre LSG priority file is used to apply a weight to a unit pattern and each of its possible rotations.

Calibre LSG allows the input assignment of different weights (priorities) for unit patterns to vary the output design clips as typically seen in horizontal and vertical routing. The random selection of unit patterns is based on the relative priorities between the unit patterns. Each rotation for a pattern can be assigned a different priority. If a unit pattern has a priority equal to zero, then that unit pattern is not used in the layout clip generation. A priority file of all zero specifications is not allowed.

You can use the GUI in the Calibre LSG Layout Generation window of Calibre WORKbench to assign the weights for each unit pattern in the priority file, or you can manually specify the priority file syntax. The assignment of weights is used to increase or decrease the unit pattern percentage (usage) within the generated layout clips. By default, all unit patterns have the same priority or weight.

Related Topics

Defining Priorities and Weights

Priority File Format

Calibre LSG Rules

Calibre LSG uses a rule file for specifying additional unit pattern configurations and neighboring constraints and for translating DRC rules to LSG rules during placement of the unit patterns inside the layout clips.

After selecting the unit patterns and their percentage of use, you can define the rules for placing the unit patterns in the layout clips for a given layer and technology in one of two ways:

- Manually create a rule file using LSG syntax to specify unit pattern and neighboring unit pattern definitions, weights, and constraints.
- Use the LSG GUI in the Calibre LSG Layout Generation window of Calibre WORKbench to graphically define or edit a rule file (preferred method).

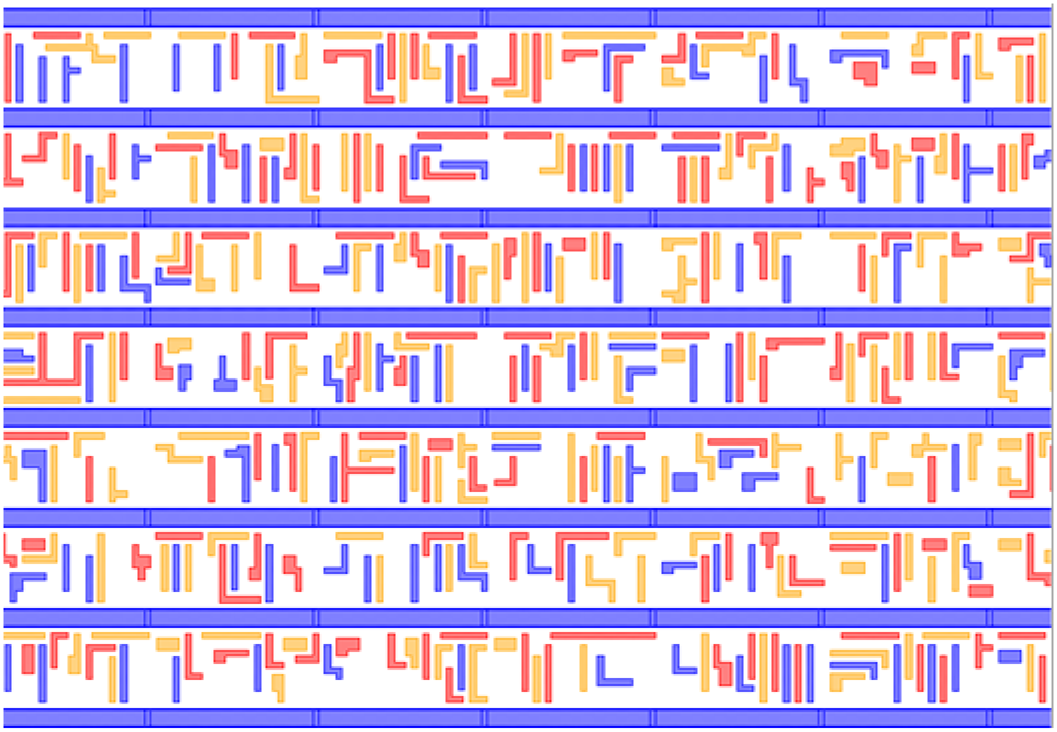
Calibre LSG Layout Clips

Calibre LSG randomly generates output layout clips that can be user-configured as realistic layouts clips for early pattern library generation and other design analysis activities.

The Calibre LSG layout clips contain unit patterns defined as multiples of design critical dimensions and symmetric shapes. Unit patterns are placed in the layout clips from the top-left corner to the bottom-right corner using basic DRC and user-defined rules in a grid formation.

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

Figure 1-5. Output Layout Clips



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

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

Chapter 2
Calibre LSG Command and File Reference

Specific syntax, command arguments, and file formats are used to configure the Calibre LSG tool flows and output.

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

calibre -lsg

The Calibre LSG software tool is run from a Linux command line with an LSG options file.

Calibre LSG generates layout clips that can be used for creating initial pattern libraries and other early design phase layout applications.

Usage

calibre -lsg [-turbo [*number_of_processors*]] ***lsg_options_file***

Description

Calibre LSG argument options can be used to customize the layout clips to represent more realistic design structures that are nearly DRC clean.

You run Calibre LSG from a command line and specify an LSG options file (*lsg_options_file*) that contains the argument options specified as *option*=*value*. The format of the command syntax in the LSG options file is one argument option per line as shown:

```
pattern_count=10
pitch_size=0.1
pattern_height=8
pattern_width=15
output_folder=LSG_OUT
```

Arguments

- **-lsg**

  A required argument specifying to run Calibre LSG from the command line.

- ***lsg_options_file***

  A required argument specifying an input file containing the argument options for running Calibre LSG. The options in this file are formatted as *option*=*value*. Specify this argument from the Calibre LSG command line.

- -turbo [*number_of_processors*]

  An optional argument and value enabling multithreaded parallel processing. Specify this argument from the Calibre LSG command line.

  The optional number_of_processors value is a positive integer that specifies the number of CPUs to use. If you do not specify number_of_processors, Calibre LSG runs on the maximum number of CPUs available for which you have licenses. See “Calibre Command Line” in the *Calibre Administrator’s Guide* for more information.

- **pattern_count=*pattern_count***

  A required keyword and argument specifying the number of the layout clips that are output. This keyword must be specified if pattern_row_count and pattern_col_count are not specified. Specify this number as a positive integer.

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

This keyword is not used if the “systematic” keyword is specified. See “SPG Flow Options” on page 77 for more information.

- **pattern_row_count=*pattern_row_count***

  A required keyword and argument specifying the number generated rows of layout clips. This keyword is used in conjunction with pattern_col_count and must be specified if pattern_count is not specified. Specify this number as a positive integer.

- **pattern_col_count=*pattern_col_count***

  A required keyword and argument specifying the number of generated columns of layout clips. This keyword is used in conjunction with pattern_row_count and must be specified if pattern_count is not specified. Specify this number as a positive integer.

- **pitch_size=*pitch_size***

  A required keyword and argument that defines the edge length of a square unit pattern with symmetric pitch. This keyword must be specified if pitch_width and pitch_height are not specified. Specify this number as a positive floating-point number in microns. The minimum value is 0.002 microns. For example, for a pitch of a 64 nm, specify pitch_size as follows:

  ```
  pitch_size=0.064
  ```

- **pitch_width=*pitch_width***

  A required keyword and argument that defines the width for a unit pattern with asymmetric pitch. Use this keyword in conjunction with the pitch_height keyword. This keyword must be specified if pitch_size is not specified. Specify this number as a positive floating-point number in microns. The minimum value is 0.002 microns.

- **pitch_height=*pitch_height***

  A required keyword and argument that defines the height of a unit pattern with asymmetric pitch. Use this keyword in conjunction with the pitch_width keyword. This keyword must be specified if pitch_size is not specified. Specify this number as a positive floating-point number in microns. The minimum value is 0.002 microns.

- **pattern_size=*pattern_size***

  A required keyword and argument that specifies the size of one layout clip in terms of the number of unit patterns used to form the clip side. A layout clip is made up of multiples of unit patterns. A pattern_size of 10 means a layout clip pattern of 10 x 10. This keyword must be specified if pattern_width and pattern_height are not specified. Specify this value as a positive integer number.

- **pattern_width=*pattern_width***

  A required keyword and argument that specifies the width of a layout clip in terms of the number of unit patterns used to form the layout clip width. Use this keyword in conjunction with the pattern_height keyword to create a rectangular layout clip. This keyword must be specified if pattern_size is not specified. Specify this value as a positive integer number.

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

- **pattern_height=*pattern_height***

  A required keyword and argument that specifies the height of a layout clip in terms of the number of unit patterns used to form the layout clip height. Use this keyword in conjunction with the pattern_width keyword to create a rectangular layout clip. This keyword must be specified if pattern_size is not specified. Specify this value as a positive integer number.

- **output_folder=*output_dir***

  A required keyword and argument defining the name of the output directory for temporary files and the generated layout clips.

- output_format={OASIS | GDSII}

  An optional keyword and argument specifying the format of the output layout clip database. If not specified, format defaults to OASIS.

- priority_file=*priority_file_name*

  An optional keyword and argument specifying the name of the priority or weight file. This file contains the weights for selecting the unit patterns and rotations.

- custom_unit_patterns=*./unit_def.conf*

  An optional keyword and argument specifying the path and filename of a unit pattern definitions file. This file may contain both default and custom unit pattern definitions. See "Unit Patterns Definition File Format" on page 45.

- margin=*margin_size*

  An optional keyword and argument specifying the margin between the generated layout clips. Specify this space as a floating-point number in microns. The margin keyword is overridden if specified with the margin_vertical or margin_horizontal keywords. The default margin space is 2 * pitch_size.

- margin_vertical=*margin_vert*

  An optional keyword and argument specifying the vertical margin between layout clips. Specify this space as a positive or negative floating-point number in microns. You can control the spacing between the layout clips as follows:

  - Specify a negative floating-point number for overlapping layout clips.
  - Specify a zero for no space between layout clips.
  - Specify a positive floating-point number for a space between layout clips.

  The margin_vertical keyword can be specified with the margin_horizontal keyword and overrides the margin keyword. The default vertical margin space is 2 * pitch_height.

- margin_horizontal=*margin_horiz*

  An optional keyword and argument specifying the horizontal margin space between the layout clips. Specify this space as a positive or negative floating-point number in microns. You can control the output as shown for the margin_vertical keyword.

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

The margin_horizontal keyword can be specified with the margin_vertical keyword and overrides the -margin keyword. The default horizontal margin space is 2 * pitch_width.

- layer=*layer_number*

  An optional keyword and argument specifying the layout layer number for the generated layout clips. Specify this number as a positive integer. The default for layer_number is 200.

- output_file=*output_file_name*

  An optional keyword and argument specifying the name of the output database file for the generated layout clips. A file extension of *.oas* or *.gds* is added to output_file_name, depending on the database format. The default database filename is *lsg_output.oas* for OASIS databases (default format) and *lsg_output.gds* for GDS format databases.

- rules=*rules_file*

  An optional keyword and argument specifying additional unit pattern configuration and neighboring unit pattern rules. For more information, contact your Siemens EDA representative for rules that may be specific to your Calibre LSG run.

- drc=*postfix_vars_file*

  An optional keyword and argument specifying a postfix variables filename. This file is used with other required input options to run a DRC-fixing flow on the generated layout clips. See “Calibre LSG DRC-Based Postfix” on page 66 and “Postfix Options” on page 70 and your Siemens EDA representative for more information on this flow.

- multi_pattern=*multi_pattern_number*

  An optional keyword and argument specifying the number of patterns (mask layers) to be represented in the generated layout clips. Specify this number as a positive integer. If the specified number is not equal to one (1), the default behavior is to define the group of unit patterns for each layer.

  For example, a double pattern ten sample layout clip is specified by the following command arguments in an LSG options file:

  ```
  pattern_count=10
  pitch_size=0.064
  pattern_size=15
  multi_pattern=2
  priority_file=my_dp_weights
  rules=my_dp_rules.conf
  output_folder=my_out_dir
  ```

  Unit patterns from 1 to 5 belong to the first layer, and unit patterns from 6 to 10, which are a copy from unit pattern 1 to 5, belong to the second layer.

  You can assign a specific layer number to each layer by defining layer IDs and assigning each layer ID a layer number using the layers_numbers argument. See the layers_numbers argument for format specifications. If specific layer numbers are not assigned to each layer, the layer numbers are assigned randomly.

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

You can define your own custom unit patterns (rather than using the basic unit patterns definitions) and generate multi-pattern layout clips by specifying a unit patterns definition file and the multi_patterning and layers_numbers arguments. The unit_patterns_layer argument can also be specified with these options to define the unit patterns for specific layers. See "Unit Patterns Definition File Format" on page 45 and the unit_patterns_layer argument for format specifications.

- layers_numbers=*layers_numbers_file*

  An optional keyword and argument specifying the name of the SVRF file used to set a layer number to each specified layer ID. If layer numbers are not specified in this file, Calibre LSG assigns them randomly. The format specifies the LAYERS_IDS variable followed by the variables for the layer number assignment.

  Constraints for the clip template definitions are as follows:

  For example, assigning layer ID 1 to layer number 100, and layer ID 2 to layer number 20.5, respectively, is specified as follows:

  ```
  VARIABLE LAYERS_IDS "1" "2"
  VARIABLE layer_1 100
  VARIABLE layer_2 20.50
  ```

- unit_pattern_layers=*unit_pattern_layer_file*

  An optional keyword and argument specifying the name of the SVRF file used to map layer IDs to unit patterns. For example, this specifies to map layer ID 1 to unit pattern IDs 1 through 5 and layer ID 2 to unit pattern IDs 6 through 10:

  ```
  VARIABLE UNIT_PATTERNS_layer_1 "1" "2" "3" "4" "5"
  VARIABLE UNIT_PATTERNS_layer_2 "6" "7" "8" "9" "10"
  ```

- cell_name_prefix=*cell_name_prefix*

  An optional keyword and argument specifying a cell name prefix for each layout clip. Each layout clip is saved in the output directory when cell_name_prefix is specified.

  For example, if cell_name_prefix LSG is specified, the top cell is named LSG and the individual layout clips are named LSG_0, LSG _1, LSG _2, …. The layout clip files *LSG_0.gds*, *LSG _1.gds*, *LSG _2.gds*, … are saved in the output directory.

  By default, the top cell is named "cell" and the individual layout clips are named "cell_0", "cell_1", "cell_2" …. The individual layout clips are *not* saved by default.

- clip_template=*clip_template_file*

  An optional keyword and argument specifying the name of a rule file used to generate power rail-like or cell boundary-like structures in the layout clip. You define the structures by filling rows or columns with a unit pattern or by placing unit patterns at specific positions. Any undefined positions are filled with random patterns.

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

Constraints for the clip template definitions are as follows:

- Definitions for rows, columns, and positions can be combined as long as a position is not defined more than once. An error is issued if clip template definitions specify the same position.
- Keyword “n” can be used to define a last row, column, or position enabling clip template definitions to update with layout clip size changes.
- Rows or columns can be defined along with a range of unit pattern positions to facilitate specification.

The clip template file contains SVRF Variable statements for the following parameters:

Row Fill

NUMBER_OF_DEFINED_ROWS *J* — Specifies the number *J* of row definition variables (DEFINED_ROWS_*j*) that are provided, where *j* goes from 1 to *J*.

DEFINED_ROWS_*j* “*p_id*” “*row*” [“*row*” …] — Fills one or more rows with a unit pattern *p_id*. Row numbers “*row*” start with 1 at the top.

You specify a “*row*” definition using the following definitions:

integer — A positive integer value.

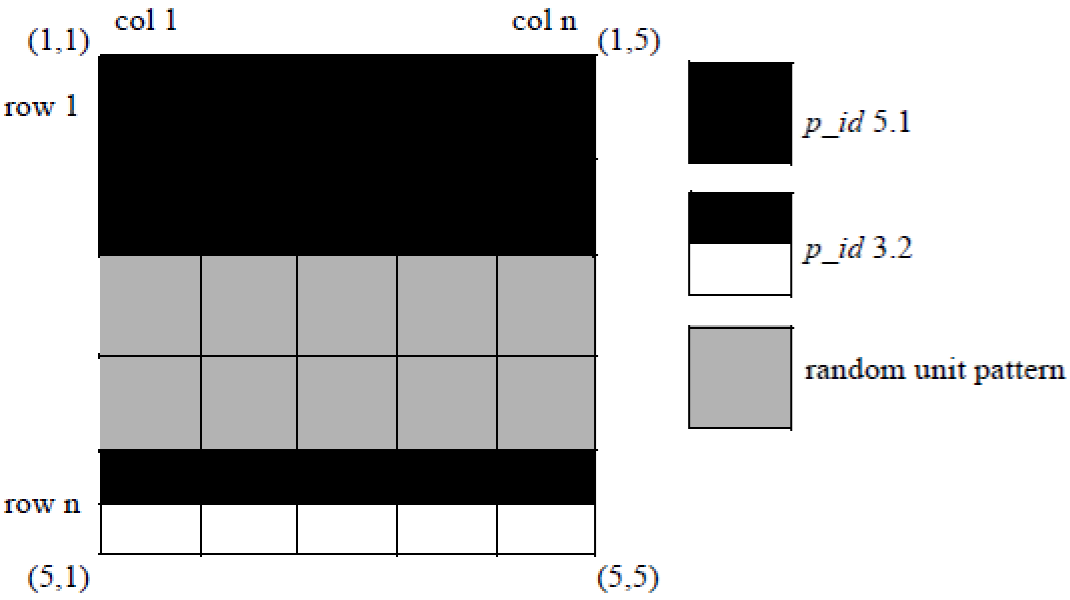
n — A keyword “n” that specifies the last row number.

n - *x* — An expression that specifies a row number relative to the last row, where *x* is a positive integer less than n.

For example, for a generated clip that is 5 x 5 with two rows of unit pattern 5.1 for the top rail and one row of unit pattern 3.2 for the bottom power rail (last row), the clip template file looks like this:

```
VARIABLE NUMBER_OF_DEFINED_ROWS 2
// two rows of pattern 5.1 at the top
VARIABLE DEFINED_ROWS_1 "5.1" "1" “2”
// one row of pattern 3.2 in row 5
VARIABLE DEFINED_ROWS_2 "3.2" "n"
```

Figure 2-1. VARIABLE DEFINED_ROWS



Column Fill

NUMBER_OF_DEFINED_COLUMNS *K* — Specifies the number *K* of column definition variables (DEFINED_COLUMNS_*k*) that are provided, where *k* goes from 1 to *K*.

DEFINED_COLUMNS_*k* "*p_id*" "*col*" ["*col*" …] — Fills one or more columns with a unit pattern *p_id*. Column numbers "*col*" start with 1 at the left side.

You specify a "*col*" definition using the following:

integer — A positive integer value.

n — A keyword "n" that specifies the last column number.

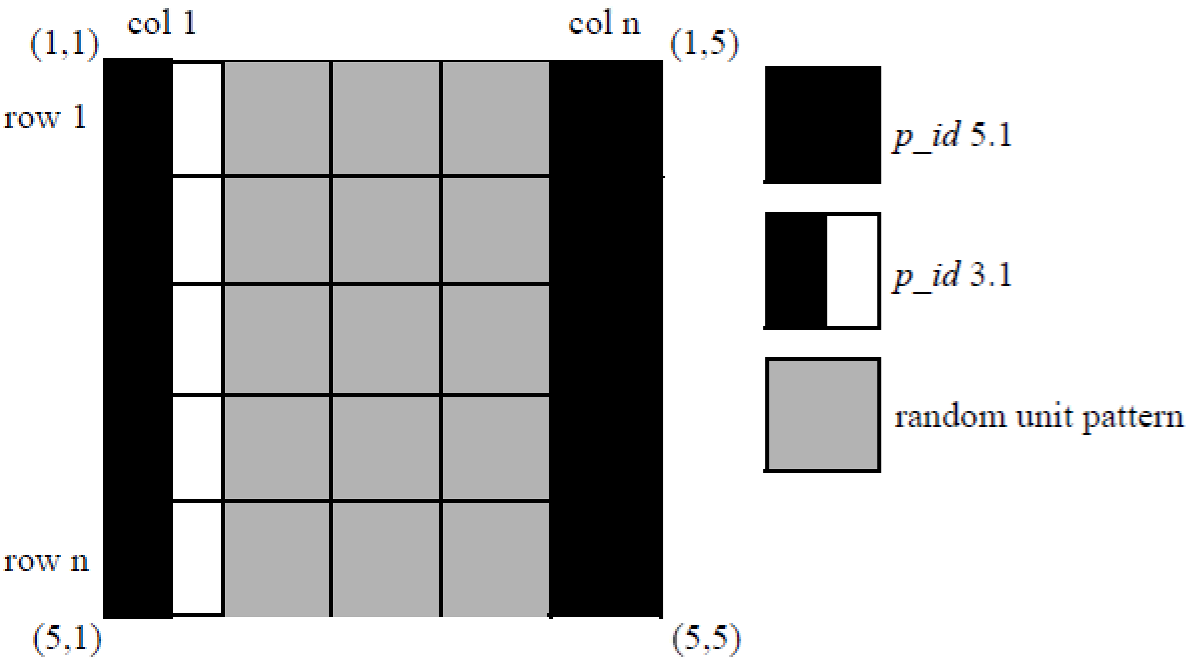
n - *x* — An expression that specifies a column number relative to the last column, where *x* is a positive integer less than n.

For example, for a generated clip that is 5 x 5 with last column of unit pattern 5.1 and first column of unit pattern 3.1, the clip template file looks like this:

```
VARIABLE NUMBER_OF_DEFINED_COLUMNS 2
// last column of pattern 5.1 at the right
VARIABLE DEFINED_COLUMNS_1  "5.1" "n"
//first column of pattern 3.1 at the left
VARIABLE DEFINED_COLUMNS_2  "3.1" "1"
```

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

Figure 2-2. VARIABLE DEFINED_COLUMNS



Position Definitions

NUMBER_OF_DEFINED_POSITIONS *P* — Specifies the number *P* of position definition variables (DEFINED_POSITIONS_*p*) that are provided, where *p* goes from 1 to *P*.

DEFINED_POSITIONS_*p* "*p_id*" "*row,col*" ["*row,col*" …] — Fills the one or more position with unit pattern *p_id*. Positions are defined with row and column numbers, starting with "1,1" at the top left.

You specify a "*row,col*" position definition using the following:

integer — A positive integer value.

n — A keyword "n" that specifies a last row or column number.

n - *x* — An expression that specifies a position relative to a last row or column, where *x* is a positive integer less than n.

***integer*:n** — A range from a positive integer value to n.

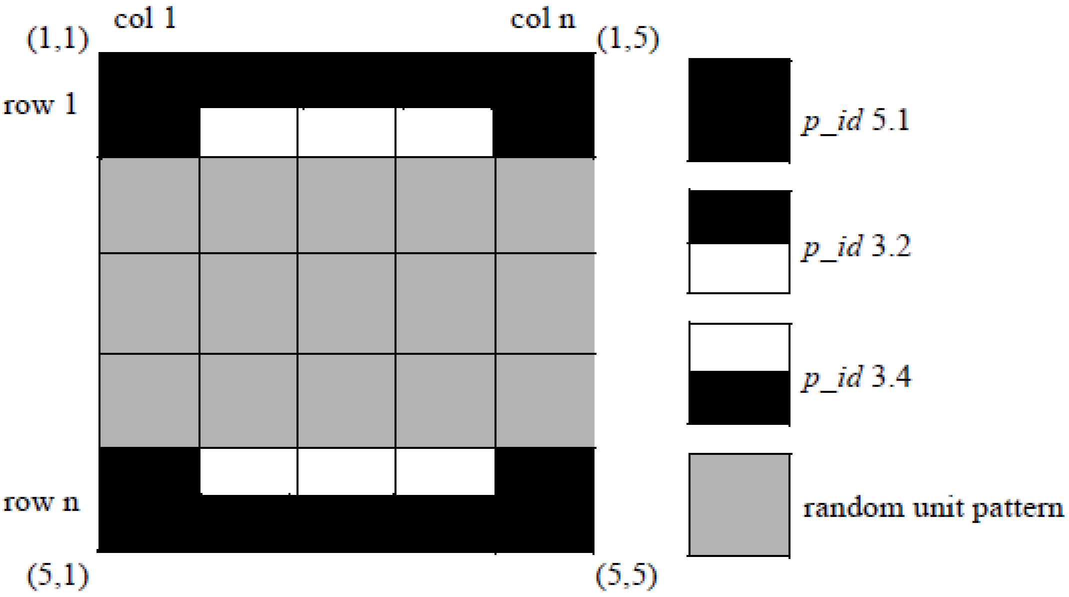
integer*:n - *x — A range from a positive integer value to a position relative to a last row or column, where *x* is a positive integer less than n.

For example, for a 5 x 5 layout clip, the following clip template file places unit pattern 5.1 at the four corners, unit pattern 3.2 in the remaining positions in the top row, and unit pattern 3.4 in the remaining positions in the bottom (last) row.

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

```
VARIABLE NUMBER_OF_DEFINED_POSITIONS 3
// define four corners of 5 x 5 grid using specified positions
VARIABLE DEFINED_POSITIONS_1 "5.1" "1,1" "1,5" "5,1" "5,5"
// define row 1, columns 2, 3, and 4 using a range of positions
VARIABLE DEFINED_POSITIONS_2 "3.2" "1,2:4"
// define row 5, and columns 2, 3, and 4 using a range of positions
 VARIABLE DEFINED_POSITIONS_3 "3.4" "n,2:4"
```

Figure 2-3. VARIABLE DEFINED_POSITIONS



- highlight_layer={*layer_number* | *layer_number.datatype*}

  An optional keyword and argument specifying the layout layer number for a highlight layer that covers the pattern extents in the generated layout clips. Specify a positive integer for layer_number (default datatype 0) or a layer number and datatype combination for layer_number.datatype. Valid layer numbers range from 0 to 65535. The highlight_layer option can be specified in conjunction with the highlight margin options.

- grid_layer={*layer_number* | *layer_number.datatype*}

  An optional keyword and argument generating a grid layer for each output clip. Specify a positive integer for layer_number (default datatype 0) or a layer number and datatype combination for layer_number.datatype. Valid layer numbers range from 0 to 65535. For example,

  ```
  grid_layer=50.2
  ```

  The grid is generated in the form of separated rectangles with the size assigned to the cells. The size of each grid rectangle is equal to the size assigned to the cell under it. The grid rectangles are sized down by 1 dbu to create a separation between the cells. The grid size follows ungrid_values if specified.

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

The behavior of the grid_layer option is as follows:

- In the case of postfix (drc option), the grid output layer is preserved after the postfix run.
- In the case of run_post_fix_only, on a previously generated layout with grid_layer, you must define the generated grid layer in the configuration file of the stand-alone postfix run. This preserves the grid layer and includes it in the resultant layout.
- In case of margin specifications with negative values, the grid layer overlaps at the boundaries.

- highlight_layer_text=*text_value*, *output_layer_number*

  An optional keyword and arguments specifying an output text layer that is a highlight layer on the generated layout. This option attaches a user-defined text string to a highlight-layer polygon. The text string is appended with the incremented cell name. The highlight_layer option does not need to be specified with this option. For example:

  ```
  highlight_layer_text=this_is_cell_number, 80.1
  ```

  If the cell name is “cell 31”, then the displayed text is as follows:

  ```
  this_is_cell_number31
  ```

  The arguments are defined as follows:

  - text_value — Specifies a literal text string. This text displays at the bottom left corner of the highlight-layer polygon.
  - output_layer_number — Specifies the layer number or layer number and datatype for the output text layer.

- text_layer=*text_value*, *attached_layer_number*, *output_layer_number*

  An optional keyword and arguments specifying an output text layer that is attached to a layer (metal or highlight) in the generated layout. For example:

  ```
  text_layer=metal2_layer_polygon,46.4,33
  ```

  The arguments are defined as follows:

  - text_value — Specifies a literal text string. This text displays at the bottom left corner of every polygon in the attached layer.
  - attached_layer_number — Specifies the layer number or layer number and datatype of the polygon layer to which the text is attached.
  - output_layer_number — Specifies the layer number or layer number and datatype of the output text layer.

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

- xorigin=*Xvalue*

  An optional keyword and argument specifying the value for the starting x-coordinate for the bottom-left corner of the generated layout. You specify the Xvalue argument as a positive floating-point number in microns. The default value is 0.

- yorigin=*Yvalue*

  An optional keyword and argument specifying the value for the starting x-coordinate for the bottom-left corner of the generated layout. You specify the Yvalue argument as a positive floating-point number in microns. The default value is 0.

- precision=*precision_value*

  An optional keyword and argument defining the database precision of the generated layout. The precision_value is the ratio of database units to user units. Preferably, the precision is specified as a positive integer. The default is 1000. For example, if the pitch size is 48 nm, and the precision is 2000, the pitch size in the generated layout is 96 nm, and if the precision is 1000 (default), the pitch size in the generated layout is 48 nm.

- flip_cells

  An optional keyword specifying to vertically flip cells referenced in the even rows of the top-cell merged layout. This is useful in the case where power rails are defined on different masks and you want to merge the cells to emulate standard cells.

- run_post_fix_only

  An optional keyword specifying to run the postfix flow (drc option) stand-alone with no layout generation. The postfix flow runs on the layouts defined by output_layout and output_folder to fix DRC violations. See "Calibre LSG DRC-Based Postfix" on page 66 or your Siemens EDA representative for more information on the postfix flow.

- resolve_contradicting_rules

  An optional keyword specifying to resolve contradicting rules encountered during the run by adding an extra rule. The extra rule is written to the contradicting_rules_resolution_log file.

- no_merge

  An optional keyword specifying not to merge the output layouts in the top cell. With this option, each generated layout clip is output to a separate layout file.

- systematic=*spg_options_file*

  An optional keyword and argument enabling the systematic pattern generator (SPG) mode. This mode can be used to run the SPG flow for creating layout clips for all pattern combinations. The spg_options_file argument contains SPG options in the format option=value. See "Systematic Pattern Generator (SPG)" on page 76 and your Siemens EDA representative for more information on this flow.

- ungrid_values=*ungrid_values_file*

  An optional keyword and argument enabling ungrid (off-pitch) patterning behavior as defined in the ungrid_values_file. See "Ungrid Values File Format" on page 49.

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

- margin_type={COMPACT | FIXED | MAXIMUM}

  An optional keyword and argument defining the type of margin space to create between the generated layout clips.

  Note

  This option does not show any difference between the three types of margin spacing except when used with the ungrid_values option.

  The margin spacing is based on the starting coordinate of the row and the calculated clip height and clip width for each margin_type as follows:

  - COMPACT (default behavior)
    - Each row starts from x-coordinate = 0 and y-coordinate = maximum clip height of the previous row added to the vertical margin.
    - Each clip in the same row starts from x-coordinate = previous clip width added to the horizontal margin.
  - FIXED
    - Each row starts from x-coordinate = 0 and y-coordinate = clip height of the corresponding clip in the previous row added to the vertical margin.
    - Each clip in the same row starts from x-coordinate = previous clip width added to the horizontal margin.
  - MAXIMUM
    - Each row starts from x-coordinate = 0 and y-coordinate = maximum clip height calculated from the specified ungrid_values added to the vertical margin.
    - Each clip in the same row starts from x-coordinate = maximum clip width calculated from the specified ungrid_values added to the horizontal margin.

- ungrid_variations_count=*number*

  An optional keyword and argument specifying the number of ungrid (off-pitch) variations for each clip within the same row. For example, if you define pattern_count=5 and ungrid_variations_count=8 with any of the ungrid_values options, this generates five different patterns, with eight ungrid variations for each pattern. The patterns are generated in five rows with each row containing eight columns. Specify this number as a positive integer.

  You can also control the number of rows and columns using the pattern_row_count and pattern_col_count options instead of pattern_count.

- min_density=*float_number_percentage*

  An optional keyword and argument specifying the minimum density required for each clip to be generated. This option is only used in SPG flow (systematic option) and cannot be specified with ungrid_values. If this option is used in the postfix flow (drc option), a

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

warning is generated. The min_density value is a floating-point number (zero to 100) expressed as a percentage. The default is 0.

- max_density=*float_number_percentage*

  An optional keyword and argument only used in SPG mode specifying the maximum density required for each clip to be generated. This option is only used in SPG flow (systematic option) and cannot be specified with ungrid_values. If this option is used in the postfix flow (drc option), a warning is generated. The max_density is a floating-point number (zero to 100) expressed as a percentage. The default is 100.

- share_same_horizontal_grid

  An optional keyword specifying to share the same clip height based on the ungrid_values per row. The behavior is as follows:

  - The row ungrid_values are fixed per each clip in the same row of the layout.
  - The row ungrid_values are chosen randomly for the first clip in each row of the layout. All remaining clips share the same ungrid_values.
  - The column ungrid_values can be different for each clip in the same row of the layout.
  - The generated layouts each share different horizontal and vertical ungrid_values when the no_merge option or SPG DisableIncrementalOutputMerge is specified.

- share_same_vertical_grid

  An optional keyword specifying to share the same clip width based on the ungrid_values per column. The behavior is as follows:

  - The column ungrid_values are fixed per each clip in the same column of the layout.
  - The column ungrid_values are chosen randomly for the first clip in each column of the layout. All remaining clips share the same ungrid_values.
  - The row ungrid_values can be different for each clip in the same column of the layout.

- final_clip_width=*clip_width*

  An optional keyword and argument defining the width of the final output clip. Use this option when the generated patterns have different widths according to the ungrid_values. This option clips the patterns with a constant mask to make them the same size. Specify the clip_width as a positive floating point number in microns.

- final_clip_height=*clip_height*

  An optional keyword and argument defining the height of the final output clip. Use this option when the generated patterns have different heights according to the ungrid_values. This option clips the patterns with a constant mask to make them the same size. Specify the clip_height as a positive floating point number in microns.

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

- highlight_left_margin=*left_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the left edge. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the left_margin_value as a floating point number in microns.

  The general behavior of the highlight margin options is as follows:

  - The enabled highlight margin option controls the highlight layer to be larger or smaller than the generated clip.
  - The start of the layout from the defined x- and y-coordinate is from the bottom left of the highlight layer of the first clip.
  - The highlight margin is calculated between the highlight layers of the clips even if the highlight layer is smaller than the original clip.

- highlight_right_margin=*right_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the right edge. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the right_margin_value as a floating point number in microns.

  The highlight_right_margin option follows the general behavior of the highlight margin options.

- highlight_top_margin=*top_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the top edge. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the top_margin_value as a floating point number in microns.

  The highlight_top_margin option follows the general behavior of the highlight margin options.

- highlight_bottom_margin=*bottom_margin_value*

  Am optional keyword and argument defining the margin value between clip layers and the highlight layer from the bottom edge. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the bottom_margin_value as a floating point number in microns.

  The highlight_bottom_margin option follows the general behavior of the highlight margin options.

- highlight_horizontal_margin=*left_right_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the left side and right sides. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the left_right_margin_value as a floating point number in microns.

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

The highlight_horizontal_margin option follows the general behavior of the highlight margin options.

- highlight_vertical_margin=*top_bottom_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the top and bottom edges. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the top_bottom_margin_value as a floating point number in microns.

  The highlight_vertical_margin option follows the general behavior of the highlight margin options.

- highlight_all_margin=*all_margin_value*

  An optional keyword and argument defining the margin value between clip layers and the highlight layer from the left, right, bottom, and top edges. Use this option to define positive or negative margins between the clip and the highlight layer specified by the highlight_layer option. Specify the all_margin_value as a floating point number in microns.

  The highlight_all_margin option follows the general behavior of the highlight margin options.

Examples

Example 1

Here is an example of running Calibre LSG from a Calibre command line with an LSG options file:

```
calibre -lsg -turbo 8 lsg_options_file
```

Here is an example of the arguments specified in an LSG options file:

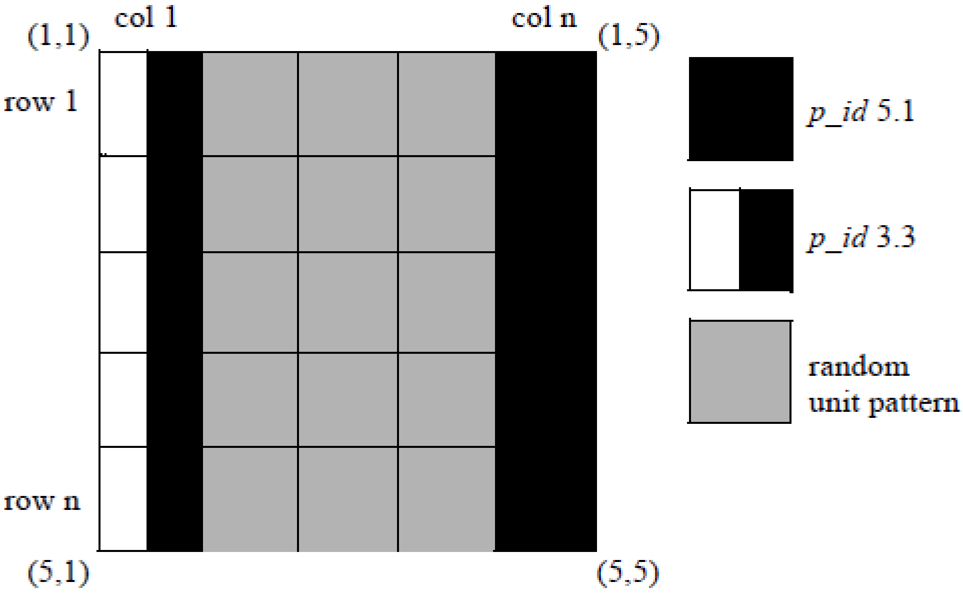
```
pattern_count=100
pitch_size=0.1
pattern_height=10
pattern_width=15
output_folder=LSG_OUT
priority_file=./inputs/weights_M1
output_file=lsg_out
```

Example 2

Here is an example of a 5 x 5 layout clip using the clip_template keyword “n” to define the last column with unit pattern *p_id* 5.1:

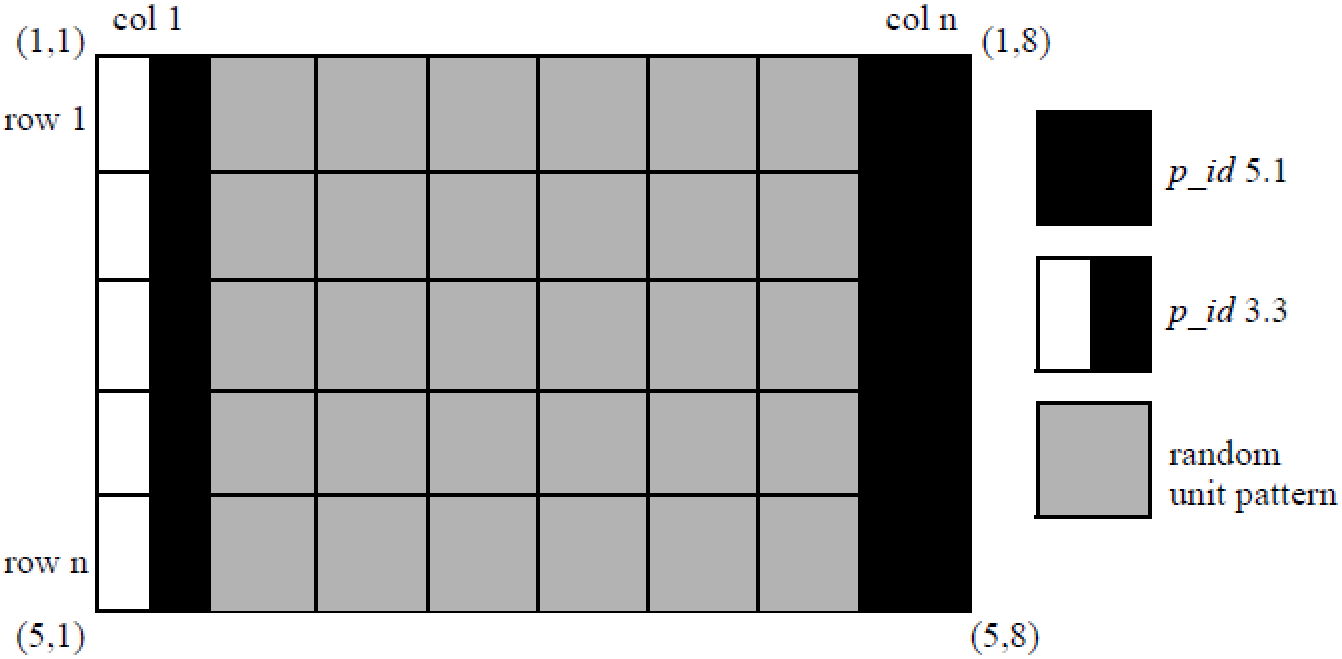
```
VARIABLE NUMBER_OF_DEFINED_COLUMNS 2
VARIABLE DEFINED_COLUMNS_1 "5.1" "n"
VARIABLE DEFINED_COLUMNS_2 "3.3" "1"
```

Figure 2-4. 5 x 5 Layout Clip with Last Column Definition “n”



If the layout clip size changes to 5 x 8, the clip template keeps the same pattern characteristics:

Figure 2-5. M-8. 5 x 8 Layout Clip with Last Column Definition “n”



Calibre LSG File Formats

Specific formatting of unit pattern priorities, definitions, and rules is required for Calibre LSG input files. This formatting is done automatically when you use the Calibre LSG GUI to create these files (recommended method). The basic format and syntax of these input files is provided as a reference for file editing or when manual file creation is the chosen option.

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

Priority File Format

Input to: Calibre LSG layout clip generation

An optional LSG priority file is used to change the weights given to each unit pattern and rotation to increase or decrease its percentage within the generated layout clips.

Note

You can use the Calibre LSG GUI to facilitate the creation of a unit pattern priority (weights) file. See “Calibre LSG GUI Reference” on page 109 for details on this interface.

Format

A priority file must conform to the following formatting and syntax rules:

- Unit pattern ID number with weight and rotation priorities are included in each line with the following syntax:

  ‘[’*patternID,patternWeight*‘]’ ‘[’*noRotationWeight,90RotationWeight, 180RotationWeight,270RotationWeight*‘]’

  where the weight values are integers greater than or equal to zero.

- A priority file with all weights specified as zero is not allowed.

An example of the priority file format is shown.

```
[1,320]  [80,0,0,0]
[2,0]    [0,0,0,0]
[3,100]  [150,0,150,20]
[4,0]    [0,0,0,0]
[5,20]   [25,25,0,0]
[6,0]    [0,0,0,0]
[7,0]    [0,0,0,0]
[8,100]  [150,0,150,20]
[9,0]    [0,0,0,0]
[10,20]  [25,25,0,0]
```

Parameters

All weights are relative to the other unit patterns specified. A greater weight means that a pattern or rotation has a higher priority of being selected.

The definitions for the parameters used for the unit pattern priority and rotation priorities are as follows:

- ***patternID***

  A required argument that is the ID number for the specified unit pattern.

- ***patternWeight***

  A required argument that specifies a weight assignment used for unit pattern selection priority.

- ***noRotationWeight***

  A required argument that specifies a weight assignment for a 0 degree or no rotation of the specified unit pattern. A weight assignment of zero can be specified.

- ***90RotationWeight***

  A required argument that specifies a weight assignment for a 90 degree rotation of the specified unit pattern. A weight assignment of zero can be specified.

- ***180RotationWeight***

  A required argument that specifies a weight assignment for a 180 degree rotation of the specified unit pattern. A weight assignment of zero can be specified.

- ***270RotationWeight***

  A required argument that specifies a weight assignment for a 270 degree rotation of the specified unit pattern. A weight assignment of zero can be specified.

Examples

In this example, pattern ID number 3 has a unit pattern priority of 80, and pattern ID number 2 has a unit pattern priority of 20, so pattern ID number 3 has four times the priority of pattern ID number 2.

For rotation priority, when pattern number 3 is selected, a rotation of 270 degrees has five times the priority of a rotation of 0 degrees (no rotation).

```
[2,20] [10,50,100,75]
[3,80] [20,0,0,100]
```

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

Unit Patterns Definition File Format

Input to: Calibre LSG layout clip generation

An optional Calibre LSG unit patterns definition file can be used to define the default unit patterns or custom unit patterns that are generated in the layout clips.

Note

You can use the Calibre LSG GUI to facilitate the creation of unit pattern definitions. See "Calibre LSG GUI Reference" on page 109 for details on this interface.

Format

The unit patterns definition file can be specified in a standard format be setting the following argument option in your LSG options file (see "calibre -lsg" on page 26 for a complete list of argument options):

```
custom_unit_patterns=./unit_def.conf
```

The unit patterns definition file must conform to the following formatting and syntax rules:

Standard Format

- The first line contains the number of unit patterns.
- Each line defines a "new" unit pattern with information as follows:
    - Unit pattern ID.
    - Number of points (vertex locations) of the unit pattern.
    - Coordinates of the points (*x*,*y*) to define the unit pattern for the possible rotations with respect to pitch size (P), where the bottom left corner of the unit pattern is at (0,0) and the top right corner is (P,P) (this is the default).
    - Default unit patterns can be defined in the same file with custom unit pattern definitions.
- Unit pattern ID with the number of vertex points and rotation points are included in each line with the following syntax:

    ***patternID*;*numberPoints*;**(*noRotation*);(*90Rotation*); (*180Rotation*);(*270Rotation*);

    where each (*nRotation*) is defined as four sets of coordinate points with each coordinate point enclosed in parentheses and separated by a colon (:) as follows:

    ‘(’x_{n0},y_{n0}‘)’:‘(’x_{n1},y_{n1}‘)’:‘(’x_{n2},y_{n2}‘)’:‘(’x_{n3},y_{n3}‘)’; …

- Optionally, a "space" unit pattern (with no points defined) can be specified in the unit patterns definition file on a separate line as follows:

    patternID;0;;;;

- An example of the unit patterns definition file using standard format is shown (specify on one line).

  ```
  2;4;(0,0):(0,P/2):(P/2,P/2):(P/2,0);(0,P/2):(0,P):(P/2,P):(P/2,P/
  2);
  (P/2,P/2):(P/2,P):(P,P):(P,P/2);(P/2,0):(P/2,P/2):(P,P/2):(P,0);
  ```

Parameters

The unit patterns definition file parameters in the standard format are defined as follows:

- ***patternID***

  A required argument that specifies the ID number for the unit pattern.

- ***numberPoints***

  A required argument that specifies the number of vertex points for the unit pattern definition.

- x_n,y_n

  An optional argument (required for patterns with non-zero vertex points) that defines the (*x*,*y*) points of the unit pattern's rotations. Possible rotations are 0 (no rotation), 90, 180, and 270 degrees. The points are relative to the pitch size (P). By default, the bottom left corner of the unit pattern is (0,0) and the top right corner is (P,P). Custom unit pattern definitions can be created that differ from these defaults.

Examples

In this standard format example, unit pattern ID number 2 has four vertex points with the following points of rotation defined (specify in one line):

```
2;4;(0,0):(0,P/2):(P/2,P/2):(P/2,0);(0,P/2):(0,P):(P/2,P):(P/2,P/2); \
(P/2,P/2):(P/2,P):(P,P):(P,P/2);(P/2,0):(P/2,P/2):(P,P/2):(P,0);
```

The points of rotation define the following unit patterns:

Rotation 0 (no rotation), unit pattern 2.1:

```
(0,0):(0,P/2):(P/2,P/2):(P/2,0)
```

Rotation 1 (90 degrees), unit pattern 2.2:

```
(0,P/2):(0,P):(P/2,P):(P/2,P/2)
```

Rotation 2 (180 degrees), unit pattern 2.3:

```
(P/2,P/2):(P/2,P):(P,P):(P,P/2)
```

Rotation 3 (270 degrees), unit pattern 2.4:

```
(P/2,0):(P/2,P/2):(P,P/2):(P,0)
```

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

Rule File Format

Input to: Calibre LSG layout clip generation

An optional Calibre LSG rule file that can be used to define neighboring unit pattern placement constraints to create layout clips that are DRC realistic.

Note

You can use the Calibre LSG GUI to facilitate the creation of an LSG rule file. See "Calibre LSG GUI Reference" on page 109 for details on this interface.

Format

You specify a rule file using the following argument in your LSG options file:

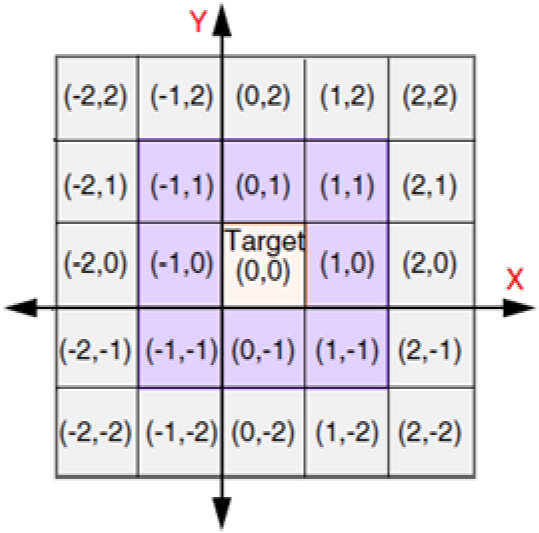
```
rules=rules_file
```

Refer to "calibre -lsg" on page 26 for Calibre LSG command usage and file argument descriptions.

The LSG rule file must conform to the following formatting and syntax rules:

- A "target" unit pattern position is defined with a lower-left origin point along with its adjacent neighbors in an x- and y-axis grid. For example, the target unit pattern in Figure 2-6 has position *x*,*y* at 0,0.

Figure 2-6. Neighbor Unit Pattern Positions



(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,0)	(-1,0)	Target (0,0)	(1,0)	(2,0)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)

- The neighboring unit patterns positions are referenced to the lower-left origin point of the target unit pattern and are constrained to avoid patterns that result in DRC violations in the output layout clip.

- Target unit pattern x- and y-origin point, neighbor unit pattern ID with weights and constraints are included in each line with the following syntax:

  ‘[’‘{’*x,y*‘}’,‘{’*target_patternID*,…‘}’‘]’;‘[’‘{’*patternID,patternWeight*‘}’;…‘]’

- If contradicting rules are defined, an error message is displayed with the contradicting rules and the pattern generated.

  An example of the rule file format is shown.

```
## Area rules
##  At least two connected unit patterns
[{-1,0},{3.2}];[{-2,0},{0.0,1.1,3.1}]; \
[{-1,1},{0.0,1.1,3.2}];[{1.1,0};]
[{0,1},{3.4}];[{1,1},{0.0,1.1}]; \
[{-1,1},{0.0,1.1,3.1}];[{1.1,0};{3.4,0}]
[{0,1},{3.1}];[{0,2},{0.0,1.1,3.2}]; \
[{-1,1},{0.0,1.1,3.1}];[{1.1,0};{3.4,0}]
```

Parameters

All unit patterns listed are neighbors to the target unit pattern position (*x*,*y*). All weights are relative to the other neighbor unit patterns specified (if more than one).

The definitions for the syntax parameters used for the rule file are as follows:

- ***x,y***

  A required argument that specifies the lower-left origin point for the target unit pattern.

- ***target_patternID***

  A required argument that specifies the pattern ID for a unit pattern in the target position. You can specify multiple unit pattern ID numbers in a comma separated list.

- ***patternID,patternWeight***

  A required argument set that specifies the pattern ID for a neighbor unit pattern with respect to the target unit pattern followed by a weight assignment for the neighbor unit pattern for selection priority. If the weight is zero, the pattern ID for that neighbor is not allowed in the layout clip.

Examples

In this example, if the target unit pattern is in position *x*= -1 and *y*= 0 and is unit pattern ID 1.1, then unit pattern ID 3.4 is disabled.

```
[{-1,0},{1.1}];[{3.4,0};]
```

Ungrid Values File Format

Input to: Calibre LSG layout clip generation

An optional LSG rules file used to specify a set of off-grid (off-pitch size) values representing different pitch heights and widths for rows and columns in the generated layout clips.

Format

You specify a rule file with ungrid values using the following keyword and argument in your LSG options file:

```
ungrid_values=ungrid_values_file
```

Refer to "calibre -lsg" on page 26 for Calibre LSG command usage and file argument descriptions.

The LSG ungrid values file must conform to the following formatting and syntax rules:

- The ungrid values file contains a list of semicolon (;) separated rule statements with one rule per line. If the ungrid value is specified with a priority weight (optional), the weight and ungrid value pairs are separated by a pipe (|). For example:

  ```
  cols : n,n-1 : 0.04|20,0.07|30;
  ```

- Each rule consists of three parts separated by a colon (:).
  - **Rule Part 1** — Specifies the rules for rows "rows", or columns "cols", or both "all".
  - **Rule Part 2** — Specifies the row or column number specifications to apply the ungrid value(s). Multiple specifications are separated by commas.
    - Numbers are between [1:*n*] (where *n* (or *N*) is equal to the total number of rows or columns) and can be replaced by a wildcard (*), which applies the rule to all rows/all columns.
    - Numbers specified by *n*-1 represent the next to the last row or column. In general, you can specify *n-x* (where *x* is any number between [1: *n*-1]).
  - **Rule Part 3** — Specifies the ungrid values and priority weights (optional) with each weight and ungrid value pair separated by comma (,). For example:

    priorityWeight,ungridValue

    - The ungrid value type is defined as follows:
      - **Discrete** — A value representing a certain pitch width or height.
      - **Continuous** — A value ending with (%) that represents a continuous range of pitch sizes beginning from the clip width/height and ending with the specified percentage added to the start value. For example, if the specified start pitch size is 0.016 um and the continuous value is 50%, this results in a valid range

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

from 0.016 to 0.024 um.

- **Range** — A set of values that specify the start, end, and step value for a range. For example, given the following values:

  Start_Value - End_Value - Step

  ```
  (0.03 - 0.04 - 0.002)
  ```

  Then the allowed values in the range are as follows:

  ```
  {0.03 - 0.032 - 0.034 - 0.036 - 0.038 - 0.04}
  ```

  - The priority weight (optional) of the given range is equally spread across the values.
  - The range start value, end value, and step must be positive floating-point numbers.
  - The end value must be larger than or equal to the start value.
  - If the specified range boundaries are outside of the scope of the given step, the end boundary is ignored.
  - The default step size, if not given, is 1 dbu (1/precision in microns).

- **Constraints** — The following constraints apply to ungrid values specifications:
  - The ungrid rules can only be specified in the ungrid values file.
  - If some discrete ungrid values have priority weight specifications and others do not, the ungrid values without weight specifications are assigned a default weight of zero, which means these ungrid values are ignored.
  - The row and column numbers can be repeated in more than one set of ungrid rules with consideration to the preceding constraint for the priority weight specifications.
  - The comments in the ungrid values file must begin with (#) and may be specified in a separate line or in the middle of a line to comment the remaining line. For example:

    ```
    1 rows : 1,n : 0.05,0.04;#comment
    2 cols : 1 : 0.04|20,0.05|30;
    3#cols : 2 : 0.06,0.07;
    ```

Parameters

- ***ungridValue***

  A required argument specifying an off-grid (off-pitch size) value to vary the row and column heights and widths in the generated layout clip. Specify the value as a positive floating-point number.

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

- *priorityWeight*

  An optional argument assigning a priority weight for an ungrid value. Paired with an ungrid value, this option prioritizes the use of that ungrid value in the generated layout clips. Specify the priority weight as a positive integer number.

- rows

  An optional argument specifying to apply the ungrid rule for rows. Specify alone or with other ungrid row and column specifications with non-conflicting values. One of “rows”, “cols”, or “all” must be specified at the beginning of each line.

- cols

  An optional argument specifying to apply the ungrid rule for columns. Specify alone or with other ungrid row and column specifications with non-conflicting values. One of “rows”, “cols”, or “all” must be specified at the beginning of each line.

- all

  An optional argument specifying to apply the ungrid rule for both rows and columns. Specify alone or with other ungrid row and column specifications with non-conflicting values. One of “rows”, “cols”, or “all” must be specified at the beginning of each line.

Examples

This example shows different ungrid rule specifications.

```
rows : 1,n 0.05,0.04;
cols : 1 : 0.04|20,0.05|30;

rows : * : 0.02|30,0.04|200;
cols : n,n-1 : 0.03|20 , 0.05|30;

all : 1,n : 0.05|60,50%|40;

all : * : (0.034 - 0.04 - 0.002) , (0.07 - 0.075);
```

Dimension-Based Rules File Format

Input to: Calibre LSG layout clip generation

An optional Calibre LSG rule file format used for generating layout clips based on off-pitch (ungrid) dimensions for rows and columns.

Format

You specify a rule file with dimension-based (and standard) LSG rules using the following argument in your LSG options file:

```
rules=rules_file
```

Refer to “calibre -lsg” on page 26 for Calibre LSG command usage and file argument descriptions.

A rules file with dimension-based rules is similar to the standard LSG rule file format but includes added specifications for two lists of x- and y-dimensions using the following formatting and syntax rules:

- A target unit pattern position is defined with a lower-left x- y-origin point along with its adjacent neighbors in an x- and y-axis grid. For example, see the target unit pattern and neighboring unit pattern positions in “Rule File Format” on page 47.
- A target unit pattern has two lists: a list of x-dimensions (widths) and a list of y-dimensions (heights) followed by a neighbor unit pattern ID with weights and constraints in each line with the following syntax (specify in a single line):

  ‘[’‘{’*x,y*‘}’,‘{’*target_patternID*:*x-dimensions*:*y-dimensions*,
  ,…**‘}’‘]’**;**‘[’‘{’*patternID*,*patternWeight*‘}’**;…**‘]’**

- A wildcard character (*) expresses the allowance of any dimension in the width or in the height lists.
- A colon (:) separates the width list and the height list from the transformed unit pattern, and a pipe (|) separates the items in each list.
- An ungrid value specified in the x- and y-dimension lists is defined by type as discrete (floating point number), continuous (integer ending with (%)), and range (Start_Value-End_Value-Step). See “Ungrid Values File Format” on page 49 for more information.

An example of a dimension-based rule is shown with the width and height dimensions highlighted as follows:

```
[{-1,0},{5.1 : 20%|0.016|0.01:0.016|0.02-0.03-0.001),\
3.1 : 50%:*,5.1 : 0.01:0.01,1.1}];
[{5.1,0};]
```

Parameters

- ***x*,*y*, *target_patternID*,** and ***patternID*,*patternWeight***

  See “Parameters” in “Rule File Format” on page 47 for descriptions.

- *x-dimensions:y-dimensions*

  An optional argument set that specifies a list of x-dimensions (widths) and a list of y-dimensions (heights) followed by a neighbor *patternID,patternWeight* pair with weights and constraints.

Examples

These examples compare two solutions (the first using standard LSG rules format, and the second using dimension-based rules format). Given the following values, the constraint is that unit pattern ID 3.1 is not allowed to appear in columns with widths 0.03 um or 0.056 um.

Unit patterns:

```
1.1, 3.1, 5.1
```

Ungrid values:

```
cols : 1,3,6 : 0.03,0.056,0.07;
```

Example 1

In this example, standard LSG rules (non-dimensional) are used to forbid the appearance of pattern ID 3.1 in columns with widths 0.03 um or 0.056 um.

These standard rules are added to the LSG rule file:

```
[{-1,0},{0.0}];[{3.1,0};]
[{-3,0},{0.0}];[{3.1,0};]
[{-6,0},{0.0}];[{3.1,0};]
```

This results in columns one and six not allowing unit pattern ID 3.1, even though the widths can accommodate the appearance of this pattern.

Example 2

In this example, dimension-based rules replace the previous non-dimensional rules in Example 1 and are used to forbid the appearance of pattern ID 3.1 in columns with widths 0.03 um or 0.056 um.

The following dimension-based rules are added to the LSG rule file (width and height dimensions highlighted):

```
[{0,1},{0.0:0.03|0.056:*,1.1:0.03|0.056:*,5.1:0.03|0.056:*}];[{3.1,0};]
```

This results in columns one and six allowing the appearance of unit pattern ID 3.1, with only column three not allowing the appearance of pattern ID 3.1.

Calibre LSG Via Command and File Formats

Certain command specifications and file formats are used for the Calibre LSG Via flow.

See "Calibre LSG Via Flow" on page 84 for information on running this flow

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

DFM LSG Via

SVRF Layer operation

Generates randomly placed vias on the input layer per the specifications. Used only by the Calibre LSG via flow.

Usage

DFM LSG VIA *layer1* [INSIDE OF *layer2*] **FILENAME *filename***

Description

Outputs a via layer on the input metal layer or at the intersections of two input metal layers. The vias are randomly placed while following the spacing and enclosure rules defined inside a via specifications file. See "Calibre LSG Via Flow Specifications File" on page 86.

Arguments

- ***layer1***

  A required argument specifying an original or derived polygon layer that represents the primary layer for vias.

- INSIDE OF *layer2*

  An optional argument set that restricts the output.

  INSIDE OF — An optional argument that constrains the output polygons (vias) on layer1 from expanding beyond the polygon boundaries of layer2. See DFM Size "INSIDE OF Behavior" in the *Standard Verification Rule Format (SVRF) Manual* for a similar argument description.

  layer2 — An optional argument specifying an original or derived polygon layer that represents a secondary layer that vias must be placed entirely inside (per INSIDE OF).

- **FILENAME "*filename*"**

  A required argument specifying the path and filename of a via specifications file (*.json* file) enclosed in quotation marks (" "). See "LSG Via Specifications File Format" on page 57.

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

Examples

This is an example of an SVRF rule file used for the Calibre LSG via flow:

```
// Input Section ****************************************************
LAYOUT SYSTEM GDSII
LAYOUT PATH "./input.gds"
LAYOUT PRIMARY "*"
PRECISION 1000

// Output Section ****************************************************
DRC MAXIMUM RESULTS ALL
DRC RESULTS DATABASE "./output.gds" GDSII
DRC SUMMARY REPORT "./drc_report.txt"

// Layer Operation Section *******************************************
LAYER metal1  16
LAYER metal2  32

vias = DFM LSG VIA metal1 INSIDE OF metal2 FILENAME "via_specs.json"
vias { COPY vias }
```

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

LSG Via Specifications File Format

Input to: Calibre LSG Via flow

A required input file to the Calibre LSG via flow that uses JavaScript Object Notation (JSON) format for specifying the via parameters and values.

Format

The path and filename of the via specifications file (*.json* extension) is included inside the via SVRF rules file for the Calibre LSG via flow. The JSON format is text-based and supports hierarchical nested data structures that coincide with the via specifications for dimension, enclosure, and spacing constraints.

Here are some basic JSON conventions:

Note

The following list contains only selected observations of JSON format conventions. Refer to https://www.json.org/ for comprehensive syntax and usage information.

- Strings are enclosed in quotation marks "*string*". Character escape sequences apply to contents within "*string*" including backslash (\) escape.
- The first string in the JSON via specifications file is used for checking forward and backward version compatibility.

```
"Version" : major.minor
```

- Floating-point numbers are supported.
- Comments can be specified as strings:

```
"comment" : "my_Via_comment"
```

- Whitespace is accepted and ignored between a pair of tokens.
- Nesting of object and array structures is supported where structures are defined as follows:
    - **Objects** — Defined as name and value pairs that are colon (:) separated and enclosed in braces ({ }) where pair is defined as { "*string*" : *value* } and *value* can be the following: *string*, *number*, *object*, *array*, "true", "false", or "null".
    - **Arrays** — Defined as ordered values that are comma (,) separated and enclosed in brackets ([ ]). Arrays include [*elements*], [*value*], and [*value* , *elements*], where *value* includes the same list as specified under Objects.

Parameters

- The structure of the via specification file with the parameters highlighted is shown:

```
o  Tag: Version

        Mandatory

        Location: Top level

        Value: (Float) x.y : where x represents major version and
        y represents minor version.

        Details:

                Backward compatibility is assured.

                Forward compatibility is assured as long as the
                difference is in the minor version only.

                Newer version with a different major number
                greater than the current major is not compatible.

o  Tag: ViaPerPolygon

        Optional: Default Value: Maximum Integer number

        Location: Top level

        Value: (Integer) Maximum number of vias per polygon.

o  Tag: Seed

        Optional: Default Value: Random

        Location: Top level

        Value: (Integer) Number that controls the randomization of
        the generated vias.

        Details:

                Fixing the same number generates the same vias
                configuration as long as all other inputs are
                the same from run to run.

o  Tag: ViaSpecs

                Mandatory

                Location: Top level

                Value: (Array) List of via specifications.

         o  Tag: Name

                Optional
```

Value: (String) name of the via that is used later as an ID for writing the layout constraints.

- Tag: **Rotation**

  Optional: Default Value true

  Location: ViaSpecs array

  Value: (Boolean) Determines if the 90 degree rotation of the via is needed.

- Tag: **Dimension**

  Mandatory

  Location: ViaSpecs array

  Value: (Object) Specifies the dimensions of the via.

  - Tag: **Height**

    Conditional Optional: If the Width is specified (square via), default value is equal to Width.

    Location: Dimension

    Value: (positive Float) The height (in the y-direction) of the via in user units.

  - Tag: **Width**

    Conditional Optional: If the height is specified (square via), default value is equal to height.

    Location: Dimension

    Value: (positive Float) The width (in the x-direction) of the via in user units.

- Tag: **Enclosure**

  Optional

  Location: ViaSpecs array

  Value: (Array) List of Enclosure ranges and values.

- Tag: **Spacing**

  Optional

  Location: ViaSpecs array

  Value: (Array) List of Spacing ranges and values.

- Tag: **WidthMin**

  Optional: Default Minimum value (zero)

  Location: Enclosure/Spacing array

  Value: (Float) The minimum metal width value for this range(inclusive) in user units.

- Tag: **WidthMax**

  Optional: Default Maximum value

  Location: Enclosure/Spacing array

  Value: (Float) The maximum metal width value for this range(exclusive) in user units.

- Tag: **LongSideEnc**

  Conditional Optional: Default value equal to ShortSideEnc (square via should have either LongSideEnc or ShortSideEnc, not both).

  Location: Enclosure array

  Value: (Float) The minimum enclosure value for this metal width range in user units associated with the long side of the bar via, and both sides of the square via.

- Tag: **ShortSideEnc**

  Conditional Optional: Default value equal LongSideEnc (square Via should have either LongSideEnc or ShortSideEnc, not both).

  Location: Enclosure array

  Value: (Float) The minimum enclosure value for this metal width range in user units associated with short side of the bar via and both sides of the square via.

- Tag: **ConditionalEnc**

  Conditional Optional: Should not be mixed with either LongSideEnc or ShortSideEnc.

  Location: Enclosure array

  Value: (Array) List of values for valid enclosure configuration. Each configuration is a list of four values that represent the minimum enclosure value for this metal width range in user units associated with each side of the via. All possible mapping between these four values and the sides of the via is considered given they follow either a clockwise or counter-clockwise order.

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

- Tag: **LongSideSpace**

  Conditional Optional: Default value equal ShortSideSpace (square via should have either LongSideSpace or ShortSideSpace, not both).

  Location: Spacing array

  Value: (Float) The minimum spacing value for this metal width range in user units associated with long side of the bar via, and both sides of the square via.

- Tag: **ShortSideSpace**

  Conditional Optional: Default value equal LongSideSpace (square via should have either LongSideSpace or ShortSideSpace, not both).

  Location: Spacing array

  Value: (Float) The minimum spacing value for this metal width range in user units associated with long side of the bar via and both sides of the square via.

- Tag: **LineEnd**

  Optional

  Location: Top level

  Value: (Object) Specify the definition of a Line End Edge and weight of via placement at these locations.

  - Tag: **MaxWidth**

    Mandatory: The maximum width of an edge to be considered as a Line End.

    Location: LineEnd

    Value: (Positive Float) The width of the edge in user units.

  - Tag: **SideMinLength**

    Mandatory: The minimum length of the two right-angled sides connected to the Line End edge.

    Location: LineEnd

    Value: (Positive Float) The length of the edge in user units.

  - Tag: **Percent**

    Mandatory: The percentage of placed Vias at Line End to the total existing Line Ends.

Location: LineEndValue: (Integer [0-100]) The ratio of vias at Line End in a percentage format.

- Tag: **LayoutConstraints**

  Optional

  Location: Top level

  Value: (Array) list of via to via constraints that apply to the entire layout.

  - Tag: **via1**

    Mandatory

    Value: (String) name of first via to which the constraint is applied.

  - Tag: **via2**

    Mandatory

    Value: (String) name of second via to which the constraint is applied.

  - Tag: **MinHorizontalSpace**

    Optional

    Value: Value of the min horizontal spacing between two vias applied across the whole layout.

    - Tag: **Opposite**

      Mandatory

      Value: (Positive Float) min opposite spacing value.

    - Tag: **Extension**

      Optional

      Value: (Positive Float) value of extension

  - Tag: **MinVerticalSpace**

    Optional

    Value: Value of the min vertical spacing between two vias applied across the entire layout.

    - Tag: **Opposite**

      Mandatory

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

Value: (Positive Float) min opposite spacing value.

- o Tag: **Extension**

  Optional

  Value: (Positive Float) value of extension.

o Tag: **MinCornerToCorner**

Optional

Value: (Positive Float) value of min corner to corner spacing between two vias applied across the entire layout.

o Tag: **MinCenterToCenter**

Optional

Value: (Positive Float) value of min center to center spacing between two vias applied across the entire layout.

Examples

See "Calibre LSG Via Flow Specifications File" on page 86 for an example via specifications file.

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

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

Chapter 3
Calibre LSG Flow Options

Information for different Calibre LSG and related flow options is provided. These flows can be used in conjunction with Calibre LSG for further analysis of layout patterns and design constraints.

Flow options for Calibre LSG enable you to perform DRC fixes, systematically test pattern combinations, explore effects of via insertion, and create arrays of randomly-placed standard cells.

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

Calibre LSG DRC-Based Postfix

The Calibre LSG postfix flow (postfix flow) performs DRC fixes on the generated layout clips based on your DRC rule specifications.

The output of a Calibre LSG run is not completely DRC-clean because certain DRC rules cannot translate into LSG rules. The postfix flow provides a solution by using your DRC rules and specifications to fix these types of violations and generate layout clips that are more DRC-clean.

The behavior of the postfix flow determines the type of DRC rules that should be included in the postfix rules and those that should be included in the LSG rules.

- The postfix flow fixes (moves edges) related to a line end rules (line end to line end, line end to edge, line end to vertex, line end to shielded line end, and so on). This means that a line end to any minimum spacing rule (min width, min area, min notch, min side to side spacing, and so on) can be included in the postfix rule file without being included in the LSG rule file.
- The postfix flow has an option to drop polygons that violate certain DRC rules that are not line end or spacing rules. This option should only be used for DRC rules that generate few violations. These DRC rules should also be included in the LSG rule file to prevent most of the violations at the beginning of the flow; otherwise, a low polygon density can result in the output layout.

Note

For any DRC rule that is not a line end or spacing rule and that could generate a large number of DRC violations, this DRC rule should be included in the LSG rule file from the beginning of the Calibre LSG flow and applied during the placement of the unit patterns in the clips during the layout generation step. This type of DRC rule should not be included in the postfix rules.

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

Calibre LSG Postfix Flow Inputs

Certain variables, rules, and specifications are required for running the postfix flow.

The postfix flow requires the following input files:

- **Postfix Variables File** — Specifies the SVRF variables that define the area, edge length, line end, and minimum width constraints.
- **Environment Variables File** — Specifies the environment settings for the DRC rule file.
- **DRC Rule File** — Specifies the DRC rule file for the generated output clips.
- **Checks File** — Specifies the set of DRC checks and actions for fixing the violations.

See "Postfix File Examples" on page 72.

Postfix Variables

The postfix variables define the constraints and values to apply for the postfix flow.

The variables are specified in the postfix variables file using SVRF variables format as follows:

```
VARIABLE VARIABLE_NAME Value
```

Values are positive floating-point numbers in microns. For example:

```
VARIABLE SMALLEST_LINE_END 0.016
```

Table 3-1. Postfix Variables

Variable Name	Description
LARGEST_LINE_END	Defines the maximum length of a line end.
LINE_END_EXTENSION	Defines the line-end extension for minimum space check.
MIN_ENCLOSURE	Defines the minimum enclosure area.
MIN_LINE_END_SPACE	Defines the minimum space between two line ends.

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

Table 3-1. Postfix Variables (cont.)

Variable Name	Description
MIN_LINE_END_TO_EDGE_SPACE	Defines the minimum space between a line end and an edge.
MIN_WIDTH	Defines the minimum width for polygons. Polygons with a width less than this value are dropped.
SHORT_EDGE_LENGTH	Defines the short-edge length.
SMALLEST_LINE_END	Defines the minimum length of a line end.
SMALLEST_NON_RECTANGULAR_AREA	Defines the minimum area of a non-rectangular shape.
SMALLEST_RECTANGULAR_AREA	Defines the minimum area of a rectangular shape.

Table 3-2. Postfix Variables Multi-Patterning

Variable Name	Description
LARGEST_LINE_END_MULTI	Defines the maximum length of a line end.
MIN_LINE_END_SPACE_MULTI	Defines the minimum space between two line ends.
MIN_LINE_END_TO_EDGE_SPACE_MULTI	Defines the minimum space between a line end and an edge.
SMALLEST_LINE_END_MULTI	Defines the minimum length of a line end.
SMALLEST_NON_RECTANGULAR_AREA_MULTI	Defines the minimum area of a non-rectangular shape.
SMALLEST_RECTANGULAR_AREA_MULTI	Defines the minimum area of a rectangular shape.
SHORT_EDGE_LENGTH_MULTI	Defines the short-edge length.

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

Table 3-2. Postfix Variables Multi-Patterning (cont.)

Variable Name	Description
SHORTEST_SIDE_CONNECTED_TO_LINE_END	Defines line ends where a convex edge is connected to other edges with the minimum length defined by this variable. If a convex edge is connected to edges shorter than this value, then the edges are not defined as line ends.
LINE_END_EXTENSION_MULTI	Defines the line-end extension for minimum space check.

In the case of multi-patterning, the postfix variable can be defined for each layer by adding a layer ID (_*layerID*) at the end of the variable name. If the variable name is specified without a layer ID, it applies to all layers without a layer ID (the default). Layer IDs are defined using the layers_numbers argument option (See “calibre -lsg” on page 26.)

For example, given a generated layout with three layers with IDs 1, 2, and 3, you can define the variable SMALLEST_LINE_END for each layer as follows:

```
VARIABLE SMALLEST_LINE_END_1 0.016
VARIABLE SMALLEST_LINE_END_2 0.016
VARIABLE SMALLEST_LINE_END_3 0.032
```

If no layer ID is specified, the variable value applies to all layers.

```
VARIABLE SMALLEST_LINE_END 0.016
```

To apply the postfix flow between more than one layer (for example, layer IDs 1 and 2) you can define the variable SMALLEST_LINE_END_MULTI as the default value, for example:

```
VARIABLE SMALLEST_LINE_END_MULTI 0.032
```

If a specific value is needed to apply the postfix flow between layer IDs 1 and 2 only, you can specify either of these variable statements:

```
SMALLEST_LINE_END_MULTI_1_2 0.016

SMALLEST_LINE_END_MULTI_2_1 0.16
```

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

Postfix Options

The postfix flow is invoked from a Linux command line within Calibre LSG.

Arguments are specified in the LSG options file and the postfix variables file.

Usage

drc=***postfix_vars_file***

Description

The postfix flow performs DRC fixes for generating DRC-cleaner layout clips. The flow uses the **drc** argument specified with a postfix variables file and other required arguments to run under Calibre LSG. See "Running Calibre LSG With PostFix" on page 74.

The options in the postfix variables file are specified with SVRF variable statements as follows (See "Postfix Variables" on page 67 for more information.):

```
VARIABLE VARIABLE_NAME Value
```

The specifications for the postfix argument options follow the format used in the LSG options file.

```
option=value
```

Note

The postfix flow uses the same argument options as Calibre LSG plus additional arguments that are specific to the postfix flow. This section contains only the postfix argument options. The arguments in common with LSG are referenced in "calibre -lsg" on page 26.

Postfix Usage Notes

- The postfix flow expects the LSG layer(s) to be defined in the LSG options file with layers_numbers or layers arguments (layer number/layer name).
- The postfix flow respects layers defined by the clip_template and highlight_layer LSG options.
- The postfix flow fixes the layouts in the output folder as defined by pattern_count when the no_merge option is defined in the LSG options file. In this case, if pattern_count is not defined, the postfix flow counts the files in the output folder and fixes them.
- The postfix flow runs separately (standalone) when run_post_fix_only is defined in the LSG options file. In this case, no layouts are generated, and the postfix flow runs on the layout defined by output_layout and output_folder.
- The standalone postfix flow (run_post_fix_only option) is enabled for the SPG flow (systematic option) in the LSG options file, only for merged outputs. It is not enabled when the no_merge (LSG options file) or DisableIncrementalOutputMerge (SPG options file) argument options are specified.

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

Arguments

- **drc=*postfix_vars_file***

  A required keyword and argument that enables the postfix flow in the LSG options file. This flow fixes specified DRC violations creating layout clips that are nearly DRC-clean. The postfix_vars_file contains variable specifications for the DRC constraints. See "Postfix File Examples" on page 72.

- **src_file=*source.env***

  A required keyword and argument for the postfix flow that specifies the path and file containing the environment variable settings to export for your DRC rule file environment. See "Postfix File Examples" on page 72. An empty file is allowed if there are no environment variable specifications.

- **rule_deck=*rules.svrf***

  A required keyword and argument for the postfix flow that specifies the path and file to your DRC rule file containing the SVRF statements and selected checks for the postfix flow. See "Postfix File Examples" on page 72.

- **check_file=*checks_file***

  A required keyword and argument for the postfix flow that specifies the path and file to a list of selected DRC checks and fixing actions. The checks are executed in the order that they appear in the file. See "Postfix File Examples" on page 72.

Examples

Here is an example of an LSG options file with the postfix flow arguments highlighted:

```
pitch_height=0.08
pitch_width=0.04
pattern_height=50
pattern_width=50
output_folder=output
priority_file=./inputs/PriorityFile
custom_unit_patterns=./inputs/unit_def_.conf
output_file=lsg_out
layer=26
rules=./inputs/LSG_RuleFile
ungrid_values=./inputs/Ungrid_values
ungrid_variations_count=1
margin_horizontal=1
margin_vertical=0.5
pattern_col_count=5
pattern_row_count=5
margin_type=COMPACT
src_file=./inputs/rules.env
rule_deck=./inputs/rules_drc.svrf
check_file=./inputs/checks_file
drc=./inputs/postfix_variables_file
```

Postfix File Examples

Reference information and example file specifications are provided for the postfix flow.

The following specifications are examples of the information required for running the postfix flow. See “Postfix Options” on page 70 for the argument options used to specify these files in your LSG options file:

Post Fix Variables File

Specified in the LSG options file:

```
drc=postfix_variables_file
```

This file includes SVRF variable statements that define the values (in microns) used by the postfix flow as constraints for area, length, line end, width, and shortest-side connected to a line end. Specify the SVRF variable statements as shown in the following example:

```
VARIABLE SMALLEST_RECTANGULAR_AREA 0.0062
VARIABLE SMALLEST_NON_RECTANGULAR_AREA 0.0062
VARIABLE SHORT_EDGE_LENGTH 0.17
VARIABLE LARGEST_LINE_END 0.34
VARIABLE MIN_WIDTH 0.044
VARIABLE SHORTEST_SIDE_CONNECTED_TO_LINE_END 0.18
```

See “Postfix Variables” on page 67 for postfix variable tables and definitions.

Environment Variables File

This file contains environment variable statements used by your SVRF rule file for running DRC such as technology settings, runsets, rule file paths, and other environment information.

Specified in the LSG options file:

```
src_file=rules.env
```

For example:

C (csh) shell

```
setenv M1 Yes
setenv M2 No
…
```

Bourne or Korn (ksh) shell:

```
export Calibre_RUNSET=./inputs
export MY_PROCESSNAME=1234
…
```

DRC Rule File

Specified in the LSG options file:

```
rule_deck=rules_drc.svrf
```

The SVRF rule file includes elements such as the layer assignments and definitions, process, rule checks, comments, specification statements, inputs and outputs, and other information specific for your design process. See “SVRF Rule File Format” in the *Standard Verification Rule Format (SVRF) Manual* and your Siemens EDA representative for assistance with creating this file.

Checks File

Specified in the LSG options file:

```
check_file=checks_file
```

This file specifies a list of selected DRC check names (as specified in the DRC rule file) for the postfix flow to modify the layout to fix DRC violations.

The postfix flow fixes four types of check violations:

- Fixes hole violations by splitting them.
- Fixes notch violations by splitting them.
- Fixes spacing violations that include a line end by moving the line end within the minimum allowed space per pitch size (line end, line side, inner vertex, shielded line end).
- Fixes other types of violations (other than spacing) by dropping the violating polygons.

Checks are fixed sequentially in the same order as listed in the checks file. Due to the order of the checks, fixing one check may affect the results of another check. For the best outcome, it is recommended to specify the selected checks in the following order:

- *check_name*; HOLE
- *check_name*; NOTCH; *spacing_value_in_microns*
- *check_name*; fix; *spacing_value_in_microns*
- *check_name*; drop

In this example, the selected DRC check names, the type of postfix flow action, and the minimum spacing value are specified as follows:

The checks are first selected from the DRC rule file.

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

```
min_metal_width {…
=0.09 …
}
min_LE_metal_spacing {…
=0.08 …
}
min_LE_LE_spacing {…
=0.08  …
}
min_LE_INNER_VERTEX_spacing {…
0.06 …
}
```

Then each selected check is specified in the checks file along with the fixing action to be taken (in this case, “fix” and “drop” only).

```
min_LE_metal_spacing; fix; 0.08
min_LE_LE_spacing; fix; 0.08
min_LE_INNER_VERTEX_spacing; fix; 0.06
min_metal_width; drop;
```

Running Calibre LSG With PostFix

The postfix flow runs within Calibre LSG from a Linux command line with specified postfix options.

Prerequisites

- See “Calibre LSG Requirements” on page 13.
- See “Postfix Options” on page 70 for argument information.
- See “Postfix File Examples” on page 72 for required input file information.

Procedure

1. Specify the drc keyword along with the required arguments for running the postfix flow in your LSG options file.

   ```
   drc=postfix_vars_file
   ```

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

For example, the following highlighted arguments are required for running the postfix flow:

```
pitch_height=0.08
pitch_width=0.04
pattern_height=50
pattern_width=50
output_folder=.
priority_file=./inputs/PriorityFile
output_file=lsg_out
layer=20
rules=./inputs/RuleFile
custom_unit_patterns=./inputs/unit_def_.conf
ungrid_values=./inputs/Ungrid
ungrid_variations_count=1
margin_horizontal=1
margin_vertical=0.5
pattern_col_count=5
pattern_row_count=5
margin_type=COMPACT
src_file=./inputs/drc.env
rule_deck=./inputs/rules_drc.svrf
check_file=./inputs/checks_file
drc=./inputs/postfix_variables_file
```

2. Run Calibre LSG from a command line shell with the following arguments:

   ```
   calibre -lsg -turbo number_of_processors lsg_options_file
   ```

   This runs Calibre LSG with the postfix arguments specified in the LSG options file.

3. (Optional) Run Calibre® nmDRC-H™ on the merged layout database to generate a summary report and results database for the final check results from your postfix flow. For example:

   ```
   calibre -drc -hier -turbo number_of_processors drc.rules
   ```

   See the *Calibre Verification User’s Manual* for more information on running Calibre nmDRC-H.

Results

The postfix flow generates DRC-fixed (cleaner) layout clips in a merged output layout.

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

Systematic Pattern Generator (SPG)

The Systematic Pattern Generator flow (SPG flow) enables the creation of small core patterns for testing the coverage of all possible unit pattern combinations in the generated layout clips.

The SPG flow works as follows:

- Creates all unit pattern combinations of *N* x *N* or *N* x *M* cells.
- Runs from a Calibre LSG command line with LSG and SPG argument options.
- Works recursively, filling each cell and filtering the unit patterns in the list by placing the unit cells starting with unit cell 1.1, then 2.*x*, … to 5.1 cells in the pattern grid.
- Creates pattern combinations with no DRC rules applied or based on user-defined DRC rules, such as minimum space and minimum width.
- Runs systematically until all unit pattern combinations are covered in the layout clips.
- (Optional) runs incrementally for larger combinations of patterns.

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

SPG Flow Options

The SPG flow is invoked from a Linux command line within Calibre LSG.

Arguments are specified in the LSG options file with additional arguments in the SPG options file.

Usage

systematic=*spg_options_file*

Description

The SPG flow creates systematic combinations of unit patterns based on argument options specified inside the Calibre LSG options file and the SPG options file. The generated layout clips from the SPG flow contain all unit pattern combinations. Optionally, the SPG flow has a feature to incrementally output batches of layout clips (for larger numbers of patterns) while the process is running.

The SPG argument options file follows the same format as the LSG options file.

```
option=value
```

The options in the SPG options file can also be specified as predefined environmental variables.

```
IncrementalClipOutput=$incClip
```

Note

The SPG flow uses the same argument options as Calibre LSG plus additional arguments that are specific to the SPG flow. This section contains only the SPG argument options. The arguments in common with LSG are referenced in "calibre -lsg" on page 26. The pattern_count, pattern_row_count, and pattern_column_count options are not used for the SPG flow.

Arguments

- **systematic=*spg_options_file***

  A required keyword and argument for enabling the SPG flow in the LSG options file. This flow can be used for creating all unit pattern combinations of a clip. The SPG options file (spg_options_file argument) contains the options specific to the SPG flow.

- IncrementalClipOutput=*number*

  An optional keyword and argument for the SPG options file defining the number of clips per batch for the incremental output. The default value is 1000 (1000 clips per file).

- DisableIncrementalOutputMerge

  An optional keyword for the SPG options file disabling the merging step of the output files. This generates each batch of clips in a separate layout file. This keyword is a boolean (if specified, the condition is true). The default is false (not specified).

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

- UseEmptyAndFullOnly

  An optional keyword for the SPG options file specifying to use empty cells (unit patterns 1.1) and filled cells (unit patterns 5.1) only. This keyword is a boolean (if specified, the condition is true). The default is false (not specified).

- DisableRules

  An optional keyword for the SPG options file specifying to run the SPG flow without the DRC rules. This keyword is a boolean (if specified, the condition is true). The default is false (not specified).

- MinWidthInPixels=*integer*

  An optional keyword and argument for the SPG options file specifying a minimum width in terms of the number of cells. This option can be used to specify how many cells must be filled to meet the DRC minimum polygon width. The value is a positive integer in pixels.

- MinSpaceInPixels=*integer*

  An optional keyword and argument for the SPG options file specifying a minimum space in terms of the number of cells. This option can be used to specify how many cells must be empty to meet the DRC minimum space. The value is a positive integer in pixels.

- min_density=*number_percentage*

  An optional keyword and argument for the SPG options file defining the percentage of minimum density (fill) required for the pattern to be generated. Specify the number as a float percentage (0 to 100). The default is 0. Patterns less than or equal to the min_density (within a tolerance of 0.0001) are excluded from the output. You can specify min_density with max_density but not with ungrid_values or post-fix (drc option).

- max_density=*number_percentage*

  An optional keyword and argument for the SPG options file defining the percentage of maximum density (fill) required for the pattern to be generated. Specify the number as a float percentage (0 to 100). The default is 100. Patterns greater than or equal to max_density (within a tolerance of 0.0001) are excluded from the output. You can specify max_density with min_density but not with the ungrid_values or postfix (drc) options.

- StartRange=*start_range_value*

  An optional keyword and argument that specifies a start range value used with the pattern density options (min_density/max_density). If specified, the density filter is applied first, and then the range filter is applied to the generated layout clip within the density range. The value must be > = 0.

- EndRange=*end_range_value*

  An optional keyword and argument that specifies an end range value with the pattern density options (min_density/max_density). If specified, the density filter is applied first, and then the range filter is applied to the generated layout clip within the density range. The value must be > 0.

Examples

Here is an example of running Calibre LSG with the LSG options file using a command-line shell:

```
calibre -lsg -turbo 8 lsg_options_file
```

Here is an example of the LSG options file with the SPG flow argument highlighted:

```
pitch_width=0.032
pitch_height=0.032
highlight_layer=1000
pattern_height=2
pattern_width=3
output_folder=output
layer=200
priority_file=./inputs/weights_file
systematic=spg_options_file
no_merge
output_format=OASIS
```

Here is an example of arguments specified in an SPG options file:

```
UseEmptyAndFullOnly
DisableRules
```

SPG DRC Rules

The SPG flow provides you with the ability to define minimum width and space constraints for applying basic DRC rules to the generated layout clips.

To apply minimum width and space constraints in the SPG flow, specify the following options in your SPG options file:

```
MinWidthInPixels=integer
MinSpaceInPixels=integer
```

Note

The minimum width and space DRC rules for the SPG flow are not applied to the border cells in the generated patterns.

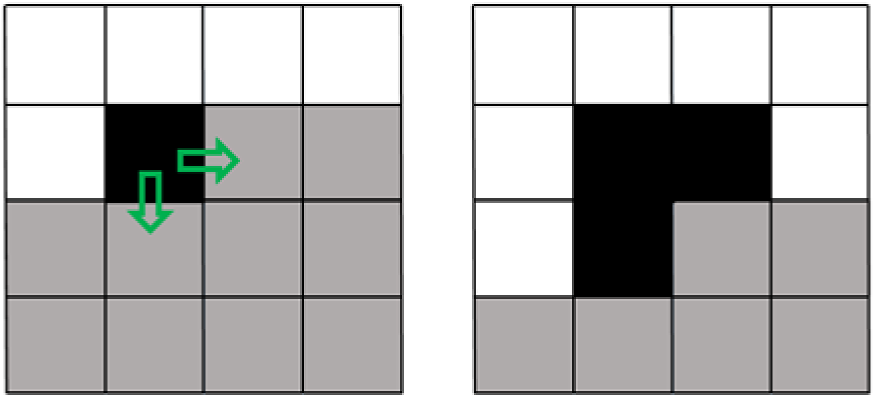
See "SPG Flow Options" on page 77 and "Running Calibre LSG With SPG" on page 81 for more information on the SPG options and flow.

The following is an example of how the SPG flow applies a minimum width constraint for a 4 x 4 generated layout clip with MinWidthInPixels=2. During pattern generation, the flow sets prefill flags designated as "FILL" (pattern 5.1), "EMPTY" (pattern 1.1) or "NA" (unassigned).

The prefill flags determine the pattern for the next cells to fill in the layout clip in order to meet the constraint. The flow performs the following steps:

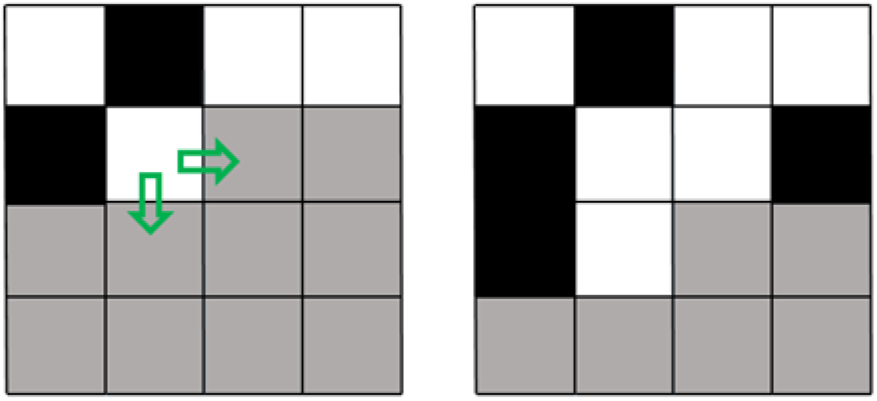
1. Checks the fill status of the current cell (target) for pattern 5.1 (filled) and minimum width > 1 pixel.
    a. Checks the cell to the left of the target for pattern 1.1 (empty), if true, it marks the next cell to the right of the target to be filled with pattern 5.1 (fill) and sets the prefill flag to FILL.
    b. Checks the cell above the target for pattern 1.1 (empty), if true, it marks the next cell below the target to be filled with pattern 5.1 (fill) and sets the prefill flag to FILL.
2. Continues with the normal flow, checking the prefill flag (either FILL, EMPTY, or NA) for the next cell to fill in the layout clip. If the prefill flag for the next cell is set to FILL, only pattern 5.1 (fill) is used.

Figure 3-1. Apply Minimum Width Constraint



For the minimum space constraint, the same steps are performed, but for this case, the flow checks the target for pattern 1.1 (empty) instead of 5.1 (fill). The next cell to the left and the next cell above the target are checked for pattern 5.1 (fill) instead of pattern 1.1 (empty). If the prefill flag for the next cell is set to EMPTY, only pattern 1.1 (empty) is used.

Figure 3-2. Apply Minimum Space Constraint



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

Note

You can also define your own DRC rules with the format used for Calibre LSG. See "Calibre LSG Rules" on page 22 and "Calibre LSG GUI Reference" on page 109.

Running Calibre LSG With SPG

The SPG flow runs within Calibre LSG from a Linux command line with a specified SPG options file.

Prerequisites

- See "Calibre LSG Requirements" on page 13.
- See "SPG Flow Options" on page 77 for argument information.

Procedure

1. Specify the keyword and argument for running the SPG flow in your LSG options file.

   ```
   systematic=spg_options_file
   ```

   For example:

   ```
   pitch_width=0.032
   pitch_height=0.032
   pattern_height=2
   pattern_width=3
   layer=200
   priority_file=./inputs/weights_file
   output_folder=output
   systematic=spg_options_file
   no_merge
   output_format=OASIS
   ```

   These rules generate layout clips of 2 x 3 patterns (six unit pattern cells per clip) as specified by the pattern_height and pattern_width arguments. The dimensions are the number of unit patterns on each edge. Each layout clip outputs to a separate file per the no_merge option.

2. Specify argument options in the SPG options file. For example:

   ```
   UseEmptyAndFullOnly
   DisableRules
   ```

   These rules generate patterns with only empty (unit pattern 1.1) and fill (unit pattern 5.1) cells. All possible pattern combinations are generated using the these two unit patterns [1.1,5.1]. No DRC rules are applied.

3. Run Calibre LSG from a command line shell:

   ```
   calibre -lsg -turbo number_of_processors lsg_options_file
   ```

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

This generates all possible pattern combinations recursively for the list of unit patterns [1.1,5.1] generating 2^6 (a total of 64 different combinations) of the layout clips.

Results

The SPG flow generates 64 individual layout clips of the unique pattern combinations. If the no_merge option is *not* specified, the flow merges the clips into a single layout database. See "SPG Generated Output" on page 82.

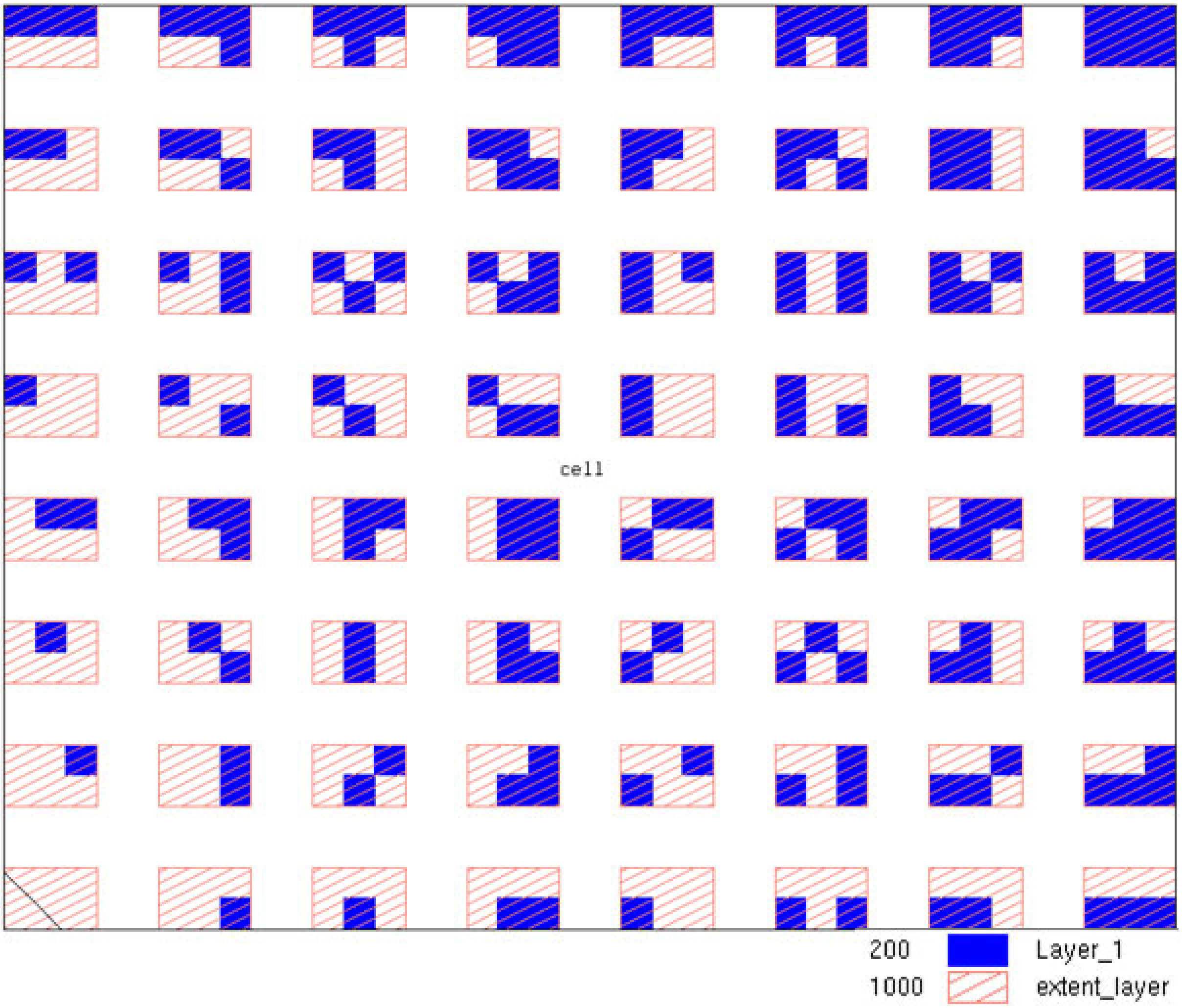
SPG Generated Output

The SPG flow generates all possible combinations for small layout clips. For larger layout clips with more pattern combination, there is an option for generating incremental output.

The following figure shows an SPG flow generated layout with 64 (2^6) combinations of a 2 x 3 unit cell pattern. In this case, the output layout clips are merged (the default behavior). See "Running Calibre LSG With SPG" on page 81 for information on how to run the flow.

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

Figure 3-3. SPG Flow Generated Output



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

Calibre LSG Via Flow

The Calibre LSG via flow enables the random generation and placement of vias in your design layout according to a set of input rules.

Using the LSG via flow facilitates CAD engineers and designers with early pattern analysis for LITHO simulation on designs with metal and via layers for finding potential sources of LITHO hotspots.

Running the Calibre LSG via flow requires certain DRC rules such as minimum enclosure and spacing in addition to specifications for the types and dimensions of the generated vias. The flow inserts and places the vias randomly based on the input layout (from Calibre LSG or an external design) following the rule checks and via specifications. The via placement occurs at areas of overlap and intersections between two layers according to the defined rules.

After the via insertion and placement finishes, the flow outputs the updated via database. As post-processing steps, you can merge the original layout database with the generated via layer in a flat (single-level) GDSII or OASIS format using Calibre® DESIGNrev™ or Calibre WORKbench (or other layout viewer). After the database merge, you can run Calibre® nmDRC-H™ or Calibre nmDRC to validate the correctness of the via-enhanced database and output a summary report of the check results to complete the post-processing on the Calibre LSG via flow.

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

Calibre LSG Via Flow Inputs

Certain via rules and specifications are required for running the Calibre LSG via flow.

The Calibre LSG via flow requires an input layout database in OASIS or GDSII format (from Calibre LSG or an external design) along with files for the via spacing and enclosure rules. There are two types of input files used by the flow:

- *VIA_Layout_Specs_SVRF_file.svrf* — The via SVRF rule file with parameters for the input layout database with the existing metal layers and the output layout database layer that contains the input metal layers and the generated vias. The path to the via specifications file is included in the layer operations section of the SVRF rule file.
- *VIA_SPECS_JSON_FILE.json* — The via specifications file in JSON format that contains the specifications for the via spacing and enclosure along with other placement constraints. This file also specifies the types of vias to generate (square, bar, or large).

Examples of these files are provided in the following sections.

Calibre LSG Via SVRF Rule File

The Calibre LSG via flow uses a Standard Verification Rule Format (SVRF) rule file. This rule file controls the processing of the Calibre LSG via flow.

The following example shows a via SVRF rule file with statements for the input layout, output database, DRC summary report, and via layer operations. The DFM LSG VIA statement generates the vias and is followed by the path to the via specifications file (*via_specs.json*).

See “DFM LSG Via” on page 55 and the *Standard Verification Rule Format (SVRF) Manual* or your Siemens EDA representative for more information on the contents of this file.

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

Example 3-1. LSG Via Flow SVRF Rule File

```
// Input Section ****************************************************
LAYOUT SYSTEM GDSII
LAYOUT PATH "./inputs/input.gds"
LAYOUT PRIMARY "*"
PRECISION 1000

// Output Section ***************************************************
DRC MAXIMUM RESULTS ALL
DRC RESULTS DATABASE "./output.gds" GDSII
DRC SUMMARY REPORT "./drc_report.txt"

// Layer Operation Section ******************************************
LAYER metal1     16

vias = DFM LSG VIA metal1 FILENAME "./inputs/via_specs.json"
vias { COPY vias }
```

Calibre LSG Via Flow Specifications File

The via specifications file contains the parameters that define the dimensions, enclosures, and spacing values for the vias in the Calibre LSG via flow.

You can define the following information in the via specifications file:

- Via types (square, bar, large) and sizes.
- Metal width and via enclosure values.
- Minimum spacing values for rules between same and different via types:
    - Minimum horizontal and vertical spacing.
        - Opposite (via-edge to via-edge) space.
        - Opposite-extended (via-edge to metal-enclosure edge) space.
    - Minimum corner-to-corner space.
    - Minimum center-to-center space.
- Maximum number of vias per polygon. For example:
    - If the ViaPerPolygon option is specified, the flow generates the defined maximum number (or less) of vias for each polygon based on the allowed locations in each polygon on the input layer.
    - If the ViaPerPolygon option is *not* specified, the flow generates the maximum number of vias that fit within the input layer without violating the defined DRC rules.
- Percentage of vias generated at line ends (with minimum enclosure) specified by the LineEnd option.

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

Each via type requires specifications for dimension, metal enclosure, and spacing as follows:

- **Dimension** — Specifies the height and width parameters (width is conditional for bar via type). The values are positive floating-point numbers.
- **Enclosure** — Specifies a range of minimum enclosure values for the via (within the metal) based on the enclosing metal width. Array and conditional specifications are supported. The values are zero or positive floating-point numbers ordered (*value1*, *value2*, *value3*) starting from the lowest value of the range. The basic form is shown in the following table:

Table 3-3. Via Enclosure Specifications

Minimum Metal Width (> =)	**Maximum Metal Width (<)**	**Minimum Enclosure (with via long side)**	**Minimum Enclosure (with via short side)**
WidthMin1	*WidthMax1*	*LongSideEnc1*	*ShortSideEnc1*
WidthMin2	*WidthMax2*	*LongSideEnc2*	*ShortSideEnc2*
WidthMin3	*WidthMax3*	*LongSideEnc3*	*ShortSideEnc3*

- **Spacing** — Specifies a range of minimum spacing values between vias based on the enclosing metal width. Array and conditional specifications are supported. The values are zero or positive floating-point numbers ordered (*value1*, *value2*, *value3*) starting from the lowest value for the range. The basic form is shown in the following table:

Table 3-4. Between Via Spacing Specifications

Minimum Metal Width (> =)	**Maximum Metal Width (<)**	**Minimum Space (with via long side)**	**Minimum Space (with via short side)**
WidthMin1	*WidthMax1*	*LongSideSpace1*	*ShortSideSpace1*
WidthMin2	*WidthMax2*	*LongSideSpace2*	*ShortSideSpace2*
WidthMin3	*WidthMax3*	*LongSideSpace3*	*ShortSideSpace3*

These specifications must be repeated for each via type along with the other applicable constraints.

The JSON format is used for handling all the via specifications needed for the Calibre LSG via flow. The following example shows a via specifications file in JSON format. This file contains the parameters and values for width, enclosure, and spacing constraints along with via definitions. See "LSG Via Specifications File Format" on page 57 and your Siemens EDA representative for more information on the contents of this file.

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

Example 3-2. Via Specifications File (JSON)

```
{
    "Version": 1.0,
    "ViaSpecs": [
        {
            "comment": "Via(BAR) 0",
            "Rotation": true,
            "Dimension": {
                "Height": 0.06,
                "Width": 0.10
            },
            "Enclosure": [
                {
                    "WidthMin": 0.000,
                    "WidthMax": 0.075,
                    "LongSideEnc": 0.024,
                    "ShortSideEnc": 0.028
                },
                {
                    "WidthMin": 0.075,
                    "LongSideEnc": 0.12,
                    "ShortSideEnc": 0.22
                }
            ],
            "Spacing": [
                {
                    "WidthMin": 0.00,
                    "LongSideSpace": 0.022,
                    "ShortSideSpace": 0.05
                }
            ]
        },
                {
            "comment": "Via(BAR) 1",
            "Rotation": true,
            "Dimension": {
                "Height": 0.07,
                "Width": 0.12
                },
            "Enclosure": [
                {
                    "WidthMin": 0.000,
                    "WidthMax": 0.075,
                    "LongSideEnc": 0.12,
                    "ShortSideEnc": 0.22
                },
                {
                    "WidthMin": 0.075,
                    "WidthMax": 0.084,
                    "LongSideEnc": 0.041,
                    "ShortSideEnc": 0.05
                },
                {
                    "WidthMin": 0.084,
                    "LongSideEnc": 0.022,
                    "ShortSideEnc": 0.023
```

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

```
        }
    ],
    "Spacing": [
        {
            "WidthMin": 0.00,
            "LongSideSpace": 0.07,
            "ShortSideSpace": 0.11
        }
    ]
},
{
    "comment": "Via(BAR) 2",
    "Rotation": true,
    "Dimension": {
        "Height": 0.08,
        "Width": 0.14
    },
    "Enclosure": [
        {
            "WidthMin": 0.000,
            "WidthMax": 0.09,
            "LongSideEnc": 0.12,
            "ShortSideEnc": 0.22
        },
        {
            "WidthMin": 0.09,
            "LongSideEnc": 0.03,
            "ShortSideEnc": 0.05
        }
    ],
    "Spacing": [
        {
            "WidthMin": 0.00,
            "LongSideSpace": 0.08,
            "ShortSideSpace": 0.12
        }
    ]
},
{
    "comment": "Via 3",
    "Dimension": {
        "Height": 0.05
    },
    "Enclosure": [
        {
            "WidthMin": 0.000,
            "WidthMax": 0.09,
            "LongSideEnc": 0.03
        },
        {
            "WidthMin": 0.09,
            "LongSideEnc": 0.03
        }
    ],
    "Spacing": [
        {
            "WidthMin": 0.00,
            "LongSideSpace": 0.06
```

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

```
                }
            ]
        },
        {
            "comment": "Via 4",
            "Dimension": {
                "Height": 0.08
            },
            "Enclosure": [
                {
                    "WidthMin": 0.000,
                    "WidthMax": 0.09,
                    "LongSideEnc": 0.04
                },
                {
                    "WidthMin": 0.09,
                    "LongSideEnc": 0.025
               }
            ],
            "Spacing": [
                {
                    "WidthMin": 0.00,
                    "LongSideSpace": 0.04
                }
            ]
        }
    ]
}
```

Running the Calibre LSG Via Flow

The Calibre LSG via flow runs within Calibre nmDRC-H from a Linux command line with a via SVRF rule file.

Prerequisites

- See “Calibre LSG Requirements” on page 13.
- Required input files as specified in “Calibre LSG Via Flow Inputs” on page 85.

Procedure

1. Run Calibre nmDRC-H with the following arguments from a command line shell:

    ```
    calibre -drc -hier -turbo number_of_processors via.svrf
    ```

    This outputs the via database containing the generated and randomly-placed vias.

    See “Calibre Command Line” in the *Calibre Administrator’s Guide* for argument information and the *Calibre Verification User’s Manual* for more information on running Calibre nmDRC-H.

2. (Post-processing) Merge the via database with the input layout database. For example, specify Calibre DESIGNrev (or other layout viewer) with arguments from the command line:

   ```
   calibredrv -a layout filemerge -in inputs/input.gds -in via_out.gds
   -exclude_layer layer_number_orig_via -out merged_out.gds
   ```

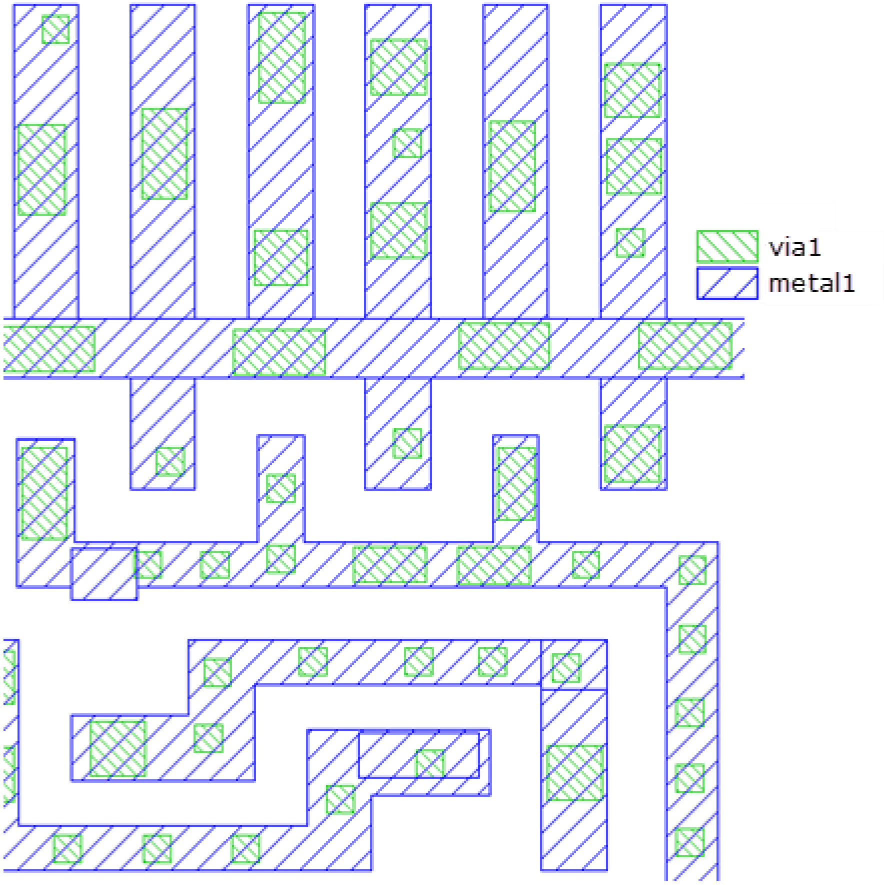
3. (Post-processing) Run Calibre nmDRC-H on the merged layout database. For example:

   ```
   calibre -drc -hier -turbo number_of_processors drc.rules
   ```

4. Load and view the output database in Calibre WORKbench (or other layout viewer). For example:

   ```
   calibrewb merged_out.gds
   ```

Figure 3-4. Calibre LSG Via Flow Merged Output



Calibre LSG Via Flow Outputs

The Calibre LSG via flow generates an output layout database with randomly placed vias that are validated using Calibre nmDRC-H.

At the completion of the flow, the following files are output:

- A results layout database that consists of the original input metal and the newly generated and placed vias. See "Running the Calibre LSG Via Flow" on page 90 for an example output layout.
- A DRC summary report with the results from the Calibre nmDRC-H run on the merged output layout database.
- A Calibre nmDRC-H transcript with rule compilation and run information. The total results for the checks are also reported at the end of the transcript.

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

Standard Cells Random Generator

The Calibre LSG Standard Cells Random Generator (Standard Cells Random Generator) is a utility that addresses the complexity of multi-height and color-aware standard cells generation and placement.

This utility facilitates quality assurance of standard cell libraries that can contain hundreds or thousands of cells. Library developers can use the Standard Cells Random Generator to ensure all combinations of neighboring-abutted cells are legal and both DRC and lithography clean for early library analysis. The arrays of randomly-placed standard cells are flipped on alternating rows to share VDD or VSS power rails.

Based on the specified command arguments, the Standard Cells Random Generator supports a colorless mode for multi-height standard cells or a color-aware mode for single-height standard cells.

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

createRandomArray

You specify createRandomArray from a command line to run the Standard Cells Random Generator using argument options for colorless or color-aware modes.

Usage

```
createRandomArray
    -inputLayoutFile input_gds_filename
    -layerNumber layer_number
    {{-arrayWidth array_width -arrayHeight array_height} |
        {-repetitions number_cell_repetitions
         [-rowSize row_cell_number -columnSize col_cell_number]}}
    -arraysNumber arrays_number
    -outputLayoutFile output_gds_filename
    {{-cellsFilePath cells_filepath
        [-vddMultiHeightCellsFilePath vdd_multiHeight_cells_filepath]
        [-vssMultiHeightCellsFilePath vss_MultiHeight_cells_filepath]} |
    {{-e1e2FilePath e1e2_filepath [-e2e1FilePath e2e1_filepath
        -e1e1FilePath e1e1_filepath -e2e2FilePath e2e2_filepath]} |
    {-e2e1FilePath e2e1_filepath [-e1e2FilePath e1e2_filepath
        -e1e1FilePath e1e1_filepath -e2e2FilePath e2e2_filepath]} |
    {-e1e1FilePath e1e1_filepath -e2e2FilePath e2e2_filepath
        [-e1e2FilePath e1e2_filepath -e2e1FilePath e2e1_filepath]}}}
    -fillerFilePath filler.txt]
    [-cellUnderTestName cell_name]
    [-outputType [gds | oasis]]
```

Description

In the colorless mode, multi-height standard cell placement is supported. The createRandomArray command takes an input layout with certain standard cells and uses the Standard Cells Random Generator to randomly generate and place the multi-height standard cells into arrays. The placement-abutted standard cell arrays are DRC compliant. Single-height filler cells are used to fill empty spaces between multi-height cells for placement continuity and to create square or rectangular shaped arrays. The output from this utility is a GDS or OASIS layout file for each placed standard cell array that can be used to develop a placement solution to supplement the standard cell library. See "Specifications for Creating Multi-Height Standard Cell Arrays" on page 98.

In the color-aware mode, the createRandomArray command takes an input layout with certain standard cells and color definitions and uses the Standard Cells Random Generator to randomly generate and place the standard cells into arrays. The placement-abutted standard cell arrays are DRC and color conflict compliant. Filler cells with different coloring combinations can be placed in empty spaces to provide color-aware placement context and to create square or rectangular shaped arrays. The output from this utility is a GDS or OASIS layout file for each placed standard cell array that can be used to develop a pre-color solution to supplement the standard cell library. See “Specifications for Creating Color-Aware Standard Cell Arrays” on page 103.

Arguments

- **-inputLayoutFile *input_gds_filename***

  A required argument that specifies the name of the GDS input layout file containing the decomposed (colored) standard cells.

- **-layerNumber *layer_number***

  A required argument that specifies the layer number containing the polygon shapes in the standard cells.

- **-arrayWidth *array_width***

  A required argument that specifies the width of the array in data base units (dbu). This argument must be used in conjunction with the **-arrayHeight** argument.

  You specify either the argument set (**-arrayWidth**, **-arrayHeight**) or the **-repetitions** argument, but not both.

- **-arrayHeight *array_height***

  A required argument that specifies the height of the array in data base units (dbu). This argument must be used in conjunction with the **-arrayWidth** argument.

  You specify either the argument set (**-arrayWidth**, **-arrayHeight**) or the **-repetitions** argument, but not both.

- **-repetitions *number_cell_repetitions***

  A required argument that specifies the number of repetitions of each standard cell to be placed. Specify the number as a positive integer. You specify either the **-repetitions** argument or the argument set (**-arrayWidth**, **-arrayHeight**), but not both.

- -rowSize *row_cell_number*

  An optional argument that specifies the number of standard cells per row. Specify the number as a positive integer. This argument must be specified in conjunction with -columnSize.

  If the -rowSize and -columnSize arguments are not specified, you can specify only the number of repetitions of standard cells to be placed with the **-repetitions** argument, which creates an approximate square array.

- -columnSize *col_cell_number*

  An optional argument and number that specifies the number of standard cells to be placed per column. Specify the number as a positive integer. This argument must be specified in conjunction with -rowSize.

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

If the -columnSize and -rowSize arguments are not specified, you can specify only the number of repetitions of standard cells to be placed with the **-repetitions** argument, which creates an approximate square array.

- **-arraysNumber *arrays_number***

  A required argument that specifies the number of arrays to be generated for each input standard cell. Specify the number as a positive integer.

- **-outputLayoutFile *output_gds_filename***

  A required argument that specifies the name of the GDS output layout file containing the random placed arrays.

- **-cellsFilePath *cells_filepath***

  A required argument (used only in colorless mode) that specifies the path to a filename containing a list of colorless cell names. These cells are of single height.

- -vddMultiHeightCellsFilePath *vdd_multiHeight_cells_filepath*

  An optional argument (used only in colorless mode) that specifies the path to a filename containing a list of colorless cell names. These cells are of multiple heights with the bottom power rail defined as VDD.

  The -fillerFilePath argument is required for multiple-height cell specifications.

- -vddMultiHeightCellsFilePath *vss_multiHeight_cells_filepath*

  An optional argument (used only in colorless mode) that specifies the path to a filename containing a list of colorless cell names. These cells are of multiple heights with the bottom power rail defined as VSS.

  The -fillerFilePath argument is required for multiple-height cell specifications.

- **-e1e2FilePath *e1e2_filepath***

  A required argument (used only in color-aware mode) that specifies the path and name of the input text file containing the standard cell names of each color type. For example, *e1e2.txt* contains all standard cell names that have left boundary polygons defined as e1 and right boundary polygons defined as e2.

  This argument is required if **-e2e1FilePath** or the argument set (**-e1e1FilePath**, **-e2e2FilePath**) is not specified.

- **-e2e1FilePath *e2e1_filepath***

  A required argument (used only in color-aware mode) that specifies the path and name of the input text file containing the standard cell names of each color type. For example, *e2e1.txt* contains all standard cell names that have left boundary polygons defined as e2 and right boundary polygons defined as e1.

  This argument is required if **-e1e2FilePath** or the argument set (**-e1e1FilePath**, **-e2e2FilePath**) is not specified.

- **-e1e1FilePath *e1e1_filepath***

  A required argument (used only in color-aware mode) that specifies the path and name of the input text file containing the standard cell names of each color type. For example, *e1e1.txt* contains all standard cell names that have boundary polygons at the right and the left defined as e1.

  This argument is required to be specified as the argument set (**-e1e1FilePath**, **-e2e2FilePath**) or with any combination of **-e1e2FilePath** and **-e2e1FilePath**.

- **-e2e2FilePath *e2e2_filepath***

  A required argument (used only in color-aware mode) that specifies the path and name of the input text file containing the standard cell names of each color type. For example, *e2e2.txt* contains all standard cell names that have boundary polygons at the right and the left defined as e2.

  This argument is required to be specified as the argument set (**-e1e1FilePath**, **-e2e2FilePath**) or with any combination of **-e1e2FilePath** and **-e2e1FilePath**.

- **-FillerFilePath *filler.txt***

  A required argument for multiple-height cells (colorless mode only) that specifies the path and the name of an input text file with a list of filler cells. The list should contain the smallest filler cell in the technology (1x size). These cells fill empty spaces in the arrays and are used to create square or rectangular output arrays.

  This argument is optional for single-height cells and the color-aware mode. For the color-aware mode, the filler cells have different coloring combinations. The names of the filler cells must end with any of the four coloring combinations (e1e1, e1e2, e2e1, and e2e2), otherwise an error message is generated. See "Example 1" on page 97.

- -cellUnderTestName *cell_name*

  An optional argument that specifies the name of the standard cell or a list of standard cell names used as input for creating the random placed arrays. If this argument is not specified, all standard cells in the input GDS file are used.

- -outputType [gds | *oas*]

  An optional argument that specifies the format of the layout database. The default is GDSII.

Examples

Example 1

Here is an example of filler cell file specifications with two coloring combinations (e1e1, e2e2). The filler cells can be used to create square or rectangular shaped arrays:

```
FILL128_DX_L_8P75TR_C68L20_e1e1
FILL8_DX_L_8P75TR_C68L20_e2e2
FILL64_DX_L_8P75TR_C68L20_e1e1
FILLSGCAP16_DX_L_8P75TR_C68L20_e2e2
```

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

Example 2

Here is an example using command line arguments to run standard cells placement in the color-aware mode:

```
createRandomArray \
 -inputLayoutFile input.gds \
 -layerNumber 108 \
 -arrayWidth 3000 \
 -arrayHeight 3000 \
 -e1e2FilePath e1e2.txt \
 -e1e1FilePath e1e1.txt \
 -e2e1FilePath e2e1.txt \
 -e2e2FilePath e2e2.txt \
 -outputLayoutFile output.gds \
 -arraysNumber 2 \
 -FillerFilePath inputs/filler.txt
```

Example 3

Here is an example using command line arguments to run multi-height standard cells placement in the colorless mode:

```
createRandomArray \
 -inputLayoutFile input.gds \
 -layerNumber 100 \
 -arrayWidth 6000 \
 -arrayHeight 4500 \
 -arraysNumber 3 \
 -outputLayoutFile output.gds \
 -cellsFilePath inputs/colorless.txt \
 -vddMultiHeightCellsFilePath inputs/vdd.txt \
 -vssMultiHeightCellsFilePath inputs/vss.txt \
 -FillerFilePath inputs/filler.txt
```

Specifications for Creating Multi-Height Standard Cell Arrays

Certain specifications and constraints apply for generating and placing multi-height standard cell arrays with the Standard Cells Random Generator.

- Placements of standard cell arrays with 1x, 2x, 3x, 4x, … height cells are only supported in colorless mode.
- The multi-height cells can be one of two types:
    - Cells that start from a VDD bottom power rail.
    - Cells that start from a VSS bottom power rail.
- The first row is placed randomly with subsequent rows collecting placement information from the previous rows.

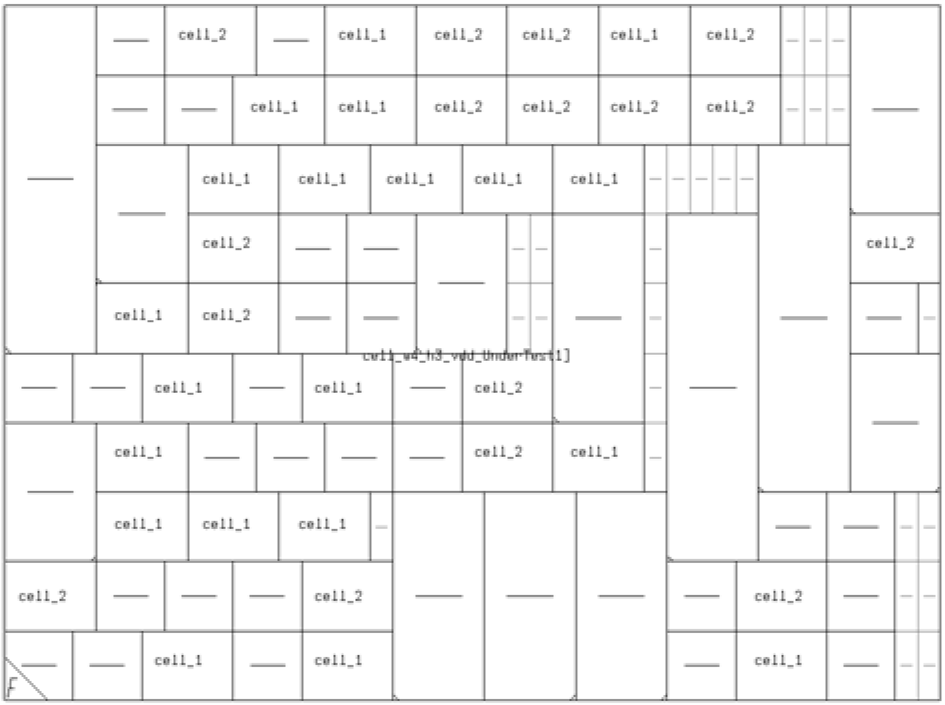
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

- Areas between multi-height cells and cells of different widths are filled randomly, with any remaining unfilled areas filled with smallest filler cells.
- Power rail continuity and cell flipping is supported as in the color-aware mode. See "Specifications for Creating Color-Aware Standard Cell Arrays" on page 103.
- Certain command arguments must be specified when running multi-height placement using the Standard Cells Random Generator. See "createRandomArray" on page 94.

Considerations for multi-height standard cells placement:

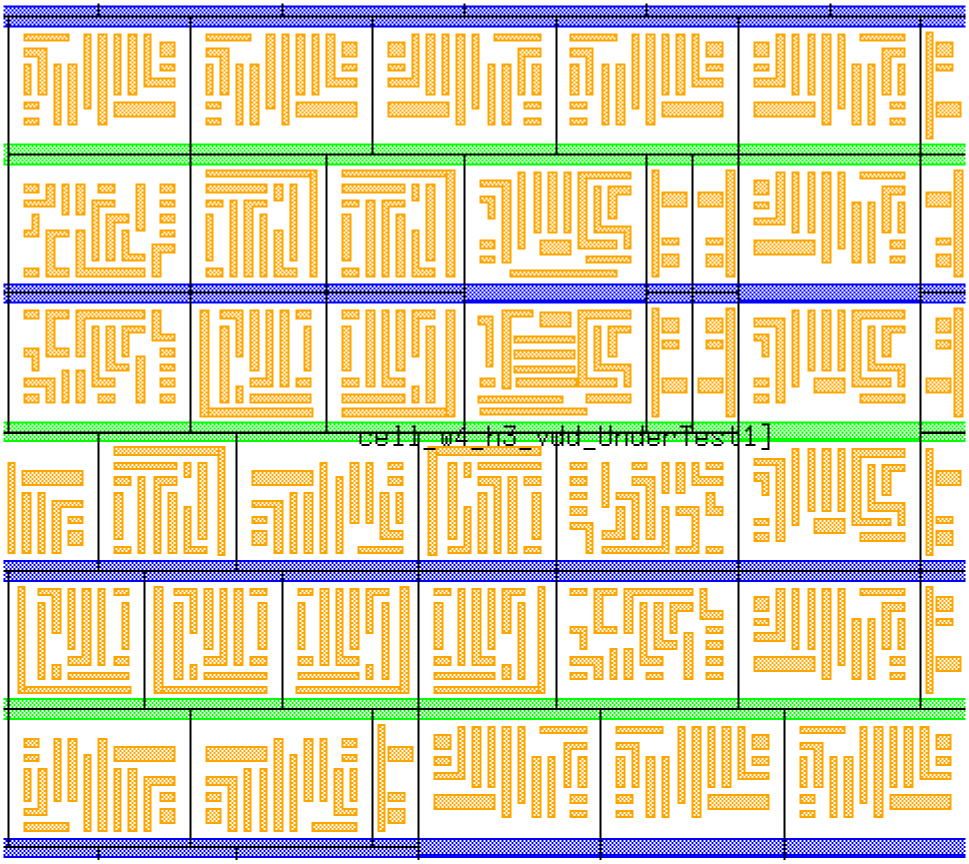
- The middle row should contain the "cell_under_test" and not be covered with a multi-height cell from the previous row.
- The last placement row (top row) should be of uniform height and not contain cells that extend outside the placement boundary.

Figure 3-5. Multi-Height Standard Cells



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

Figure 3-6. Multi-Height Colorless Standard Cell Arrays



Overview of Color-Aware Standard Cell Designs

Multi-patterning and color-aware standard cell design and placement utilities are needed as part of a solution for resolving mask resolution issues in digital designs.

Local standard cell metal layers (M1 and contact layers) are some of the most critical and complex layers in advanced node designs. This is due to the smaller feature density and minimum distances between the mask shapes. To address this challenge, certain lithography techniques such as triple-patterning lithography (TPL) and beyond are used to decompose (split) the original layout mask into separately-processed and colored masks. The assignment of shapes to each mask layer is based on the minimum distance constraints between the colored shapes and the defined setup criteria. Other considerations include minimizing stitching candidates (shapes that split a mask into overlapping shapes) and verifying any color conflicts. The Calibre multi-patterning flow and related SVRF commands can be used to perform this flow and its related tasks. Refer to the *Calibre Multi-Patterning User's and Reference Manual* for more information.

The Standard Cells Random Generator assists with standard cell library verification by generating random arrays of standard cells that can be used to test different placement contexts. It creates pre-coloring solutions for standard cells for the M1 layer, resolving layout

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

decomposition and placement simultaneously. Library developers using this utility can avoid TPL color conflicts and verify potential solutions for standard cell placement early in the design process, thus reducing cost and cycle time.

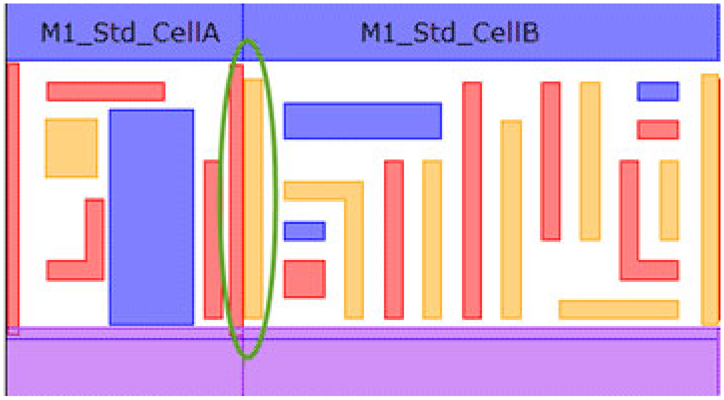
Standard Cell Color Type Definitions

Each standard cell can be defined by a color type that is based on the mask color assignment in multi-pattern lithography. Color-aware placement can randomly place standard cells with compatible color types into arrays that can be used for creating a set of pre-coloring solutions to supplement the standard cell library.

During standard cell array placement, color type awareness eliminates color conflicts between the standard cells. A standard cell with a boundary polygon (shape inside the left or right edge of the cell) that has the same color as a boundary polygon of a previously placed standard cell cannot be abutted horizontally. Power rails separate the standard cells vertically and provide immunity to color type conflicts.

Each standard cell has a specific color type which is defined by an input color type list (text file) that contains the standard cell name(s) for that color type. There are four color type lists that can be supplied as inputs (at least one color type list is required). Color types are designated as e1e1, e1e2, e2e1, and e2e2, based on the boundary polygon color combinations. The Standard Cells Random Generator considers these color type assignments when randomly placing the standard cells into *N* number of arrays. The rule requires that a standard cell must be placed next to a neighbor standard cell with a compatible color type in order to be abutted to as shown Figure 3-7.

Figure 3-7. Abutted Standard Cells with Compatible Color Types



Standard cell color types are defined by the color of the boundary polygons inside the left and right cell edges of the cell. In the standard cell examples shown, the decomposed (colored) boundary polygons are defined as red (e1) and yellow (e2).

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

Figure 3-8. Color Type e1e1 — Left e1 and Right e1

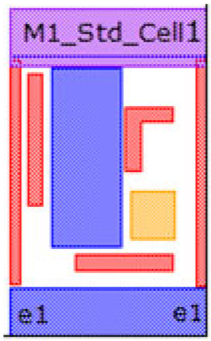


Figure 3-9. Color Type e1e2 — Left e1 and Right e2

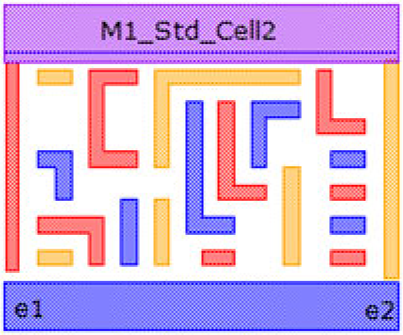


Figure 3-10. Color Type e2e1 — Left e2 and Right e1

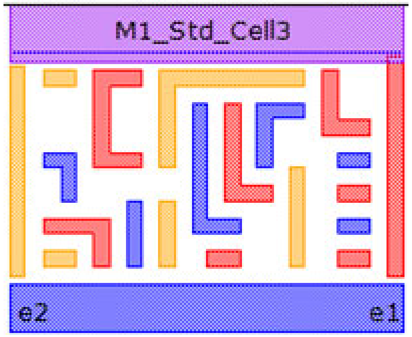
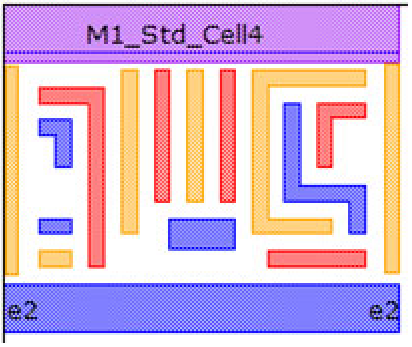


Figure 3-11. Color Type e2e2 — Left e2 and Right e2



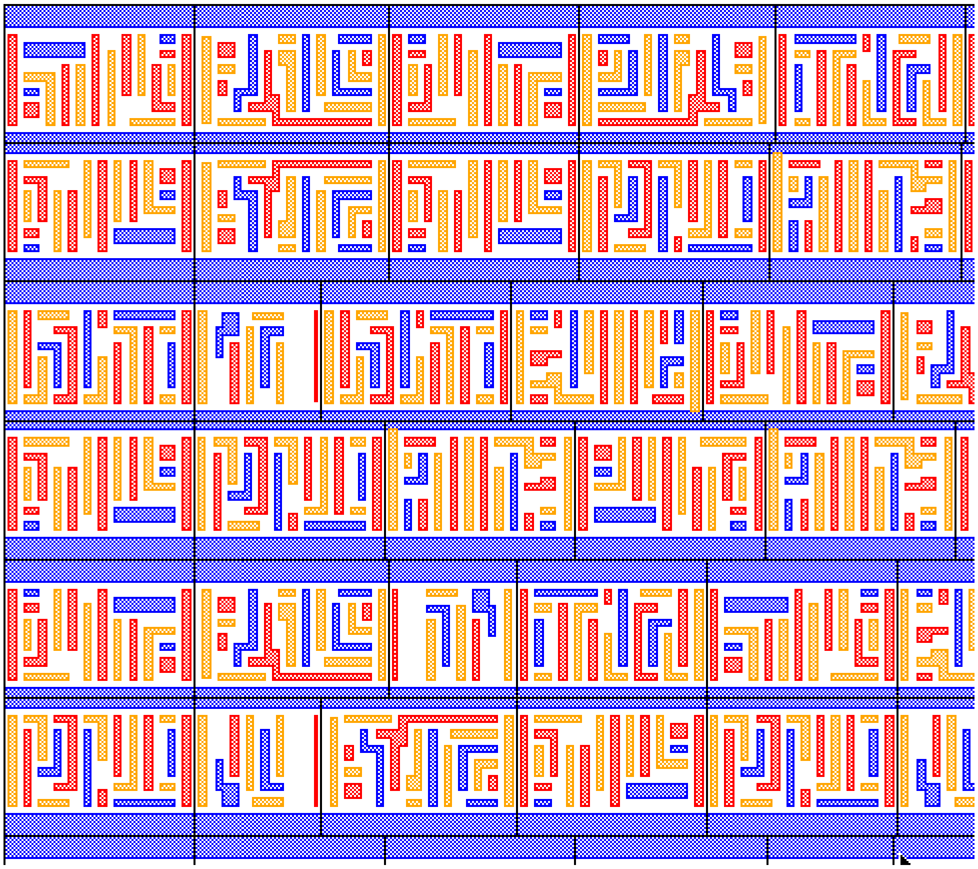
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Specifications for Creating Color-Aware Standard Cell Arrays

Certain specifications and constraints apply for generating and placing color-aware standard cell arrays with the Standard Cells Random Generator.

- Pre-coloring of mask features applies only to the M1 layer. All other layers are colorless.
- At least one color type list (text) containing the standard cell names to verify is required as an input. For a list of prerequisites for using this utility see "Generating Color-Aware Standard Cell Arrays" on page 104.
- Color awareness is maintained during placement with compatible color type constraints.
- DRC violations or color conflicts are not allowed during placement.
- Standard cells in the same row can be mirrored on the y-axis, but not on the x-axis.
- Standard cells in every other row of a generated array have a flipped orientation on the y-axis in order to share power rails.
- Standard cells with heights greater than single height are not supported in the color-aware mode. See "Specifications for Creating Multi-Height Standard Cell Arrays" on page 98.
- Certain command arguments must be specified when running color-aware placement using the Standard Cells Random Generator. See "createRandomArray" on page 94.

Figure 3-12. Color-Aware Standard Cell Arrays



Generating Color-Aware Standard Cell Arrays

You can use the Standard Cells Random Generator to create randomly-placed standard cells in arrays that can be verified and used for pre-coloring library solutions.

Prerequisites

- Access to the Calibre software tree executable from a Linux command line.
- A GDS input layout file containing the decomposed (colored) standard cells that you want to verify.
- Required input values along with a valid path and name for the generated output GDS file. See the "createRandomArray" command reference section for argument information.
- Prepared input lists of standard cell(s) with color type definitions (e1e2, e2e1, e1e1, and e2e2). You may specify any combination of color type lists with the following constraints:
    - At least one color type list must be specified.

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

- Conflicting color type lists such as (e1e1 only or e2e2 only) are not allowed.

Some combinations of color type definition lists may result in unbalanced output arrays in terms of the number of repetitions of each standard cell placed. For example, defining only e1e2 and e1e1 color type lists results in a larger number of e1e2 placements compared to e1e1 placements.

Procedure

1. Run the Standard Cells Random Generator from the command line using your arguments with the following syntax:

```
createRandomArray -inputLayoutFile input.gds -layerNumber 100 \
 -arrayWidth 2320 -arrayHeight 2880 \
 -e1e2FilePath e1e2.txt -e1e1FilePath e1e1.txt \
 -e2e1FilePath e2e1.txt -e2e2FilePath e2e2.txt \
 -outputLayoutFile output.gds -arraysNumber 5
```

2. Review the placed standard cell arrays in the output GDS file using your layout browser of choice.

3. You can re-configure the generated standard cell arrays as needed by editing or changing the combinations of the color type lists and the command arguments used for running the Standard Cells Random Generator.

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

Chapter 4
Calibre LSG GUI

The Calibre LSG GUI enables you to interactively create, edit, and save the rules and input files required for running LSG. Using the Calibre LSG GUI to interact with these input rules and files greatly reduces the time and effort needed to configure your run and facilitates rule debugging. Once the input requirements are met, the Calibre LSG GUI can be used to test your setup and perform a run.

See your Siemens EDA representative for more information on the available Calibre LSG GUI options.

DRC Rules to Calibre LSG GUI Rules

The Calibre LSG GUI is used for translating basic DRC rules into graphical rules that control the placement of the unit cells into unit patterns. These graphical rules are then used to create an LSG rule file.

After the unit cell patterns and their priority (weight) or percent usage in the layout clips is determined, basic DRC rules are applied to different unit cell placement scenarios that result in the violation of these rules. For example, if a minimum notch rule specifies that the metal spacing must be greater than or equal to 0.1 microns, then you can use the Calibre LSG GUI to draw unit cell placement scenarios that result in the violation of this rule. Once these unit cell placement scenarios are identified, LSG rules are created to prevent violating patterns from appearing in the output layout clips.

In order to draw these unit cell placement scenarios, you must first define a target unit cell position in the placement grid. The target position is determined by the latest (most recently) placed unit cell pattern that caused a violation. Because unit cell pattern placement is from the top-left corner to the bottom-right corner of the pattern, you can use this placement sequence to determine the latest-placed violating unit cell (the target position) in each placement scenario. In each of these placement scenarios, the target position is identified in the Calibre LSG GUI, and the other unit patterns are defined with respect to the target position. You use this placement information to create LSG rules that result in layout clips without DRC rule violations.

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

Calibre LSG GUI Reference

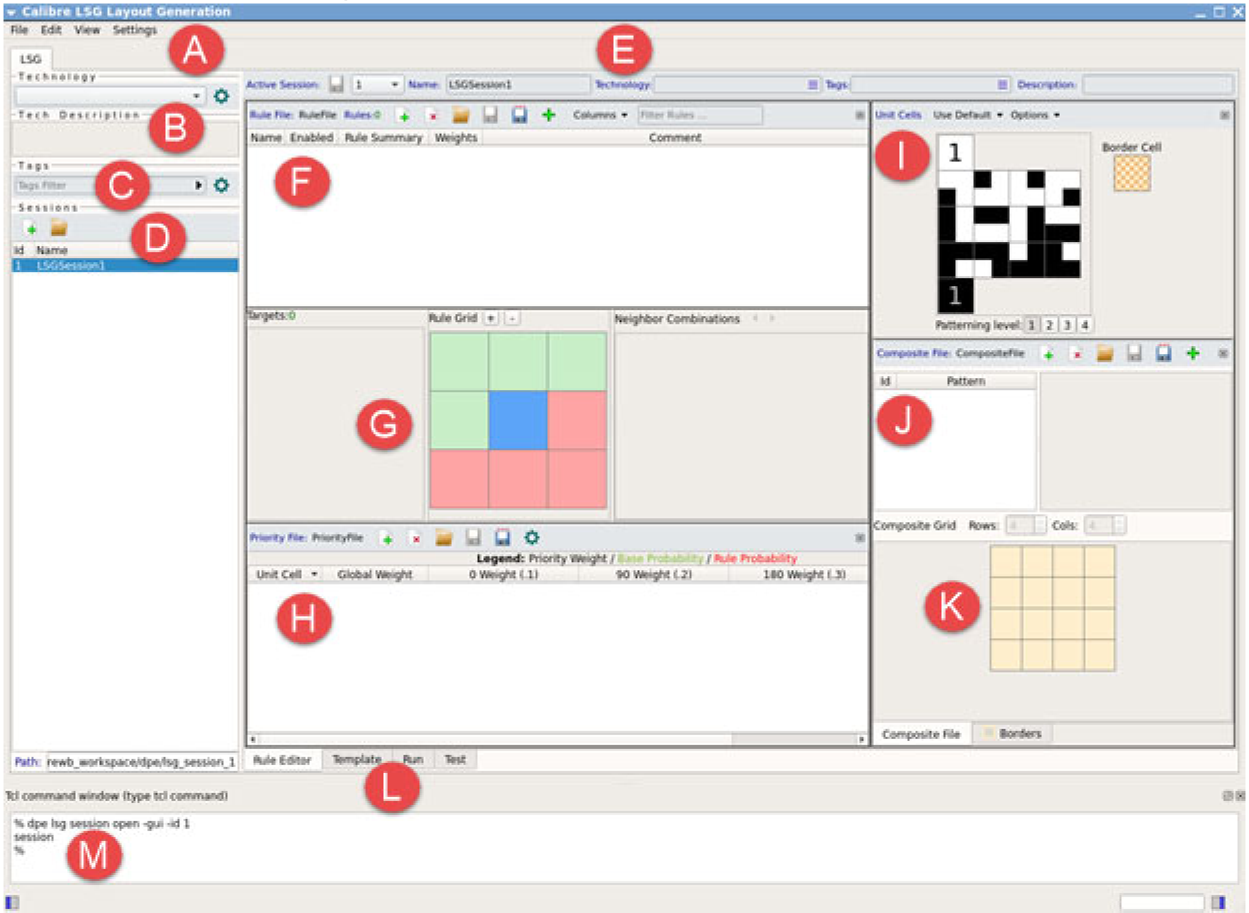
To access: From a Calibre WORKbench menu bar, **Litho** > **LSG Layout Generation**, or from a Calibre WORKbench Tcl shell call—see "Invoking Calibre LSG from a Tcl Shell Call" on page 16.

The main window of the Calibre LSG GUI is located in the **LSG** tab of the Calibre LSG Layout Generation window. The Calibre LSG GUI provides the graphical user interface for setting up and testing the rules, weights, and pattern definitions used for testing and generating the Calibre LSG layout clips.

Description

You can start an interactive session in the Calibre LSG GUI to create, edit, and save the required input and output files for Calibre LSG. The primary work modes are tabbed at the bottom of the Calibre LSG Layout Generation window and displayed in conjunction with the four global menus at the top of the window. Each of the Calibre LSG GUI panes in the main window has a specific location and purpose that is shown in Figure 4-1 on page 109 and defined by the objects in the following table.

Figure 4-1. Calibre LSG GUI Main Window



Sessions: 1 LSGSession1

Path: rewb_workspace/dpe/lsg_session_1

Active Session: 1   Name: LSGSession1   Technology: Tags: Description:

Rule File: RuleFile   Rules:0

Targets:0

Priority File: PriorityFile

Composite File: CompositeFile

```
% dpe lsg session open -gui -id 1
session
%
```

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

Objects

Table 4-1. Calibre LSG Layout Generation Window Contents

Object	Description
A	Global menus: **File** — Close the Calibre LSG Layout Generation main window. **Edit** — Cut, copy, paste, or delete rules. **View** — Toggle the view of the Tcl command window. **Settings** — Display or update the default Calibre LSG Layout Generation database and work directories.
B	**Technology** — Manage technology information.
C	**Tags** — Manage tag information.
D	**Sessions** — Start a new session or import a saved session. Shows session Id Name, and Path.
E	**Active Session —** Write and display information and settings for the active session.
F	**Rules** — Create, import, and save rule file settings.
G	**Rule Grid** — Interactively place unit cells or and combinations of unit cells in a grid relative to a target cell.
H	**Priority Weight** — Create, import, and save weight file settings for the selected unit cells.
I	**Unit Cells Options** — Click and drag selected unit cells and their rotations to the Rule Grid (up to four mask levels).
J	**Composite File** — Create, import, and save a composite patterns file.
K	**Composite Grid** — Interactively create a composite pattern file with user-defined unit cells in a grid.
L	**Tabs** — Select tabs for the primary work modes of the Calibre LSG GUI. See the specific tab pages: “Rule Editor Tab” on page 111(default window) “Template Tab” on page 115 “Run Tab” on page 117 “Test Tab” on page 120
M	**Tcl command window** — Display and enter interactive Tcl commands.

Usage Notes

For a description of the file formats and argument options used for imported files see “calibre -lsg” on page 26 and “Calibre LSG File Formats” on page 42.

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

Rule Editor Tab

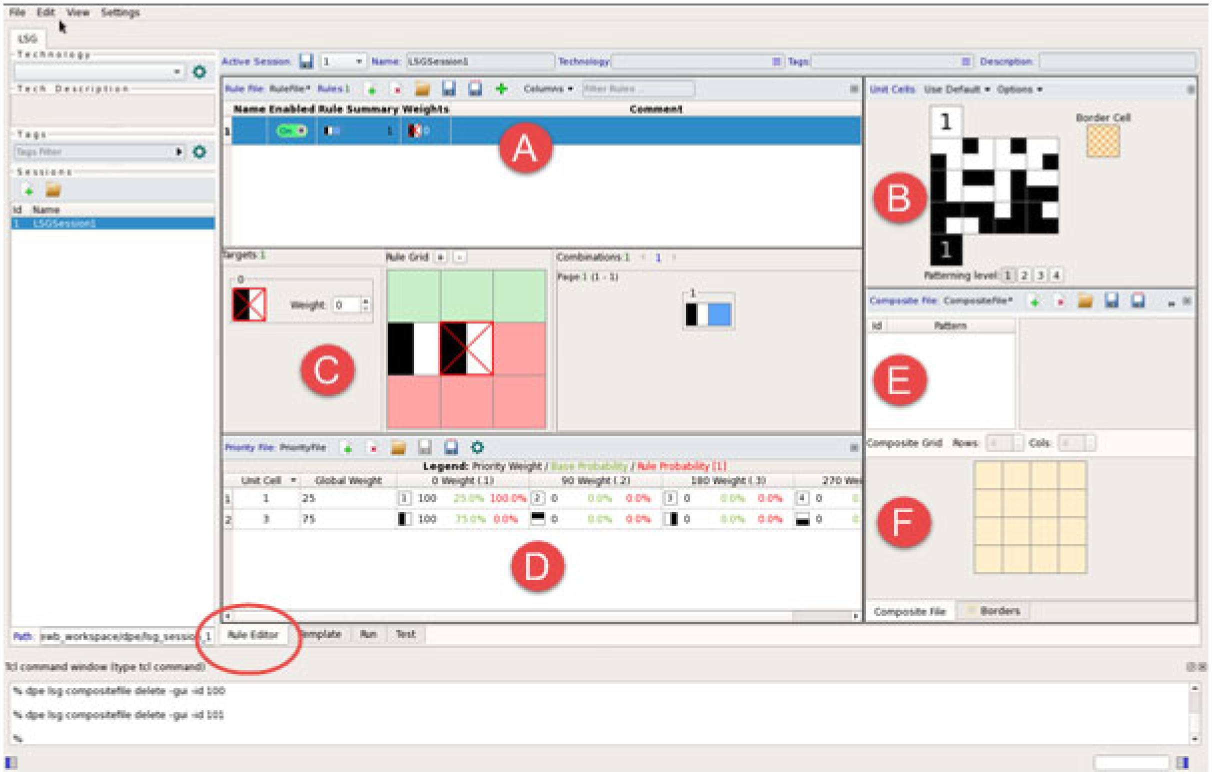
To access: From a Calibre WORKbench menu bar, **Litho**> **LSG Layout Generation**, or from a Calibre WORKbench Tcl shell call—see “Invoking Calibre LSG from a Tcl Shell Call” on page 16.

Click the **Rule Editor** tab.

Use this tab to input Calibre LSG rules, priority (weight) settings, and unit pattern information. This window mode is the same as the Calibre LSG main GUI.

Description

Figure 4-2. Rule Editor Tab



Legend: Priority Weight / Base Probability / Rule Probability [1]

	Unit Cell	Global Weight	0 Weight (1)			90 Weight (2)			180 Weight (3)			270 We
1	1	25	1 100	25.0%	100.0%	2 0	0.0%	0.0%	3 0	0.0%	0.0%	4 0
2	3	75	100	75.0%	0.0%	0	0.0%	0.0%	0	0.0%	0.0%	0

```
% dpe lsg compositefile delete -gui -id 100
% dpe lsg compositefile delete -gui -id 101
%
```

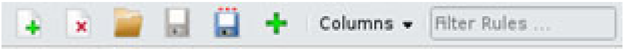
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

Objects

<table>
<tr><th>Object</th><th>Description</th></tr>
<tr><td>A</td><td>

Rule File pane — Use to perform rule file actions and display rule information.

Definitions (left to right):



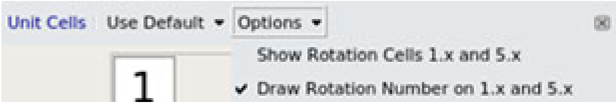
- **Create New Rule File** — Open (+) a new rule file.
- **Close Rule File** — Close (X) current rule file.
- **Import Rule File** — Import a saved rule file.
- **Save Rule File** — Save the current rule file.
- **Save as Rule File** — Save the current rule file in a specified location.
- **Add New Rule** — Add (+) a new rule to the rule file.
- **Columns** — Use the dropdown list to choose the columns to display.
- **Filter Rules** — Filter rule information by name or comment.

</td></tr>
<tr><td>B</td><td>

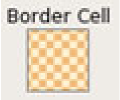
Unit Cells pane — Use to define and drag-select unit cells for placement in the Rule Grid and Composite Grid.

Definitions (top to bottom):

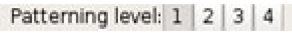
- **Use Default** — Select default or custom unit pattern options.
- **Options** — Select unit cell rotation options.


Unit Cells
Use Default
Options
Show Rotation Cells 1.x and 5.x
✔ Draw Rotation Number on 1.x and 5.x
1

- **Border Cell** — Drag-select Border Cell to Rule Grid location.



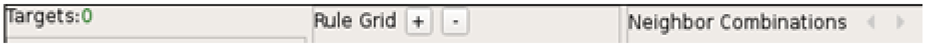
- **Patterning Level** — Display unit cell combinations for each mask layer (up to four levels).



</td></tr>
<tr><td>C</td><td>

Rule Grid pane — Use to perform unit cell placement on a grid and display relative placement information.

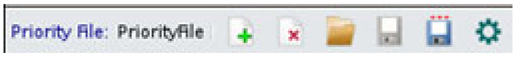
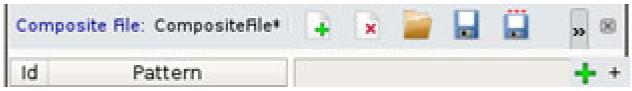
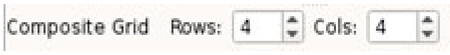
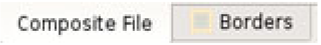
Definitions (left to right in pane):



- **Targets** — Display target unit cell information.
- **Rule Grid** — Choose (+) or (-) to change grid size.
- **Neighbor Combinations/Combinations** — Display neighbor unit cell information.

</td></tr>
</table>

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

Object	Description
D	**Priority File pane** — Use to create, import, and save priority (weights) for unit cells and rotations to a file. Definitions (left to right):  • **Create New Priority File** — Open (+) a new priority file. • **Close Rule File** — Close (X) current priority file. • **Import Priority File** — Import a saved priority file. • **Save Priority File** — Save the current priority file. • **Save as Priority File** — Save the current priority file in a specified location. • **Add/Remove cells in Priority file** — Add and remove unit cells in the priority file.
E	**Composite File pane** — Use to create, import, and save composite patterns to a file. Definitions (left to right):  • **Create New Composite File** — Open (+) a new composite file. • **Close Composite File** — Close (X) current composite file. • **Import Composite File** — Import a saved composite file. • **Save Composite File** — Save the current composite file. • **Save as Composite File** — Save the current composite file in specified location. • **Add New Pattern** — Add (+) a new composite Id and pattern.
F	**Composite Grid pane** — Use to perform unit cell placement on a grid and create and display composite patterns. Definitions (left to right):  • **Rows:** — Specify the number of rows in the composite pattern. • **Cols:** — Specify the number of columns in the composite pattern. Definitions (left to right):  • **Composite File tab** —Create and display composite patterns (default pane). • **Borders tab** — Create and display composite pattern borders.

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

Usage Notes

For a description of the argument options and the format of imported files, see "calibre -lsg" on page 26 and "Calibre LSG File Formats" on page 42.

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

Template Tab

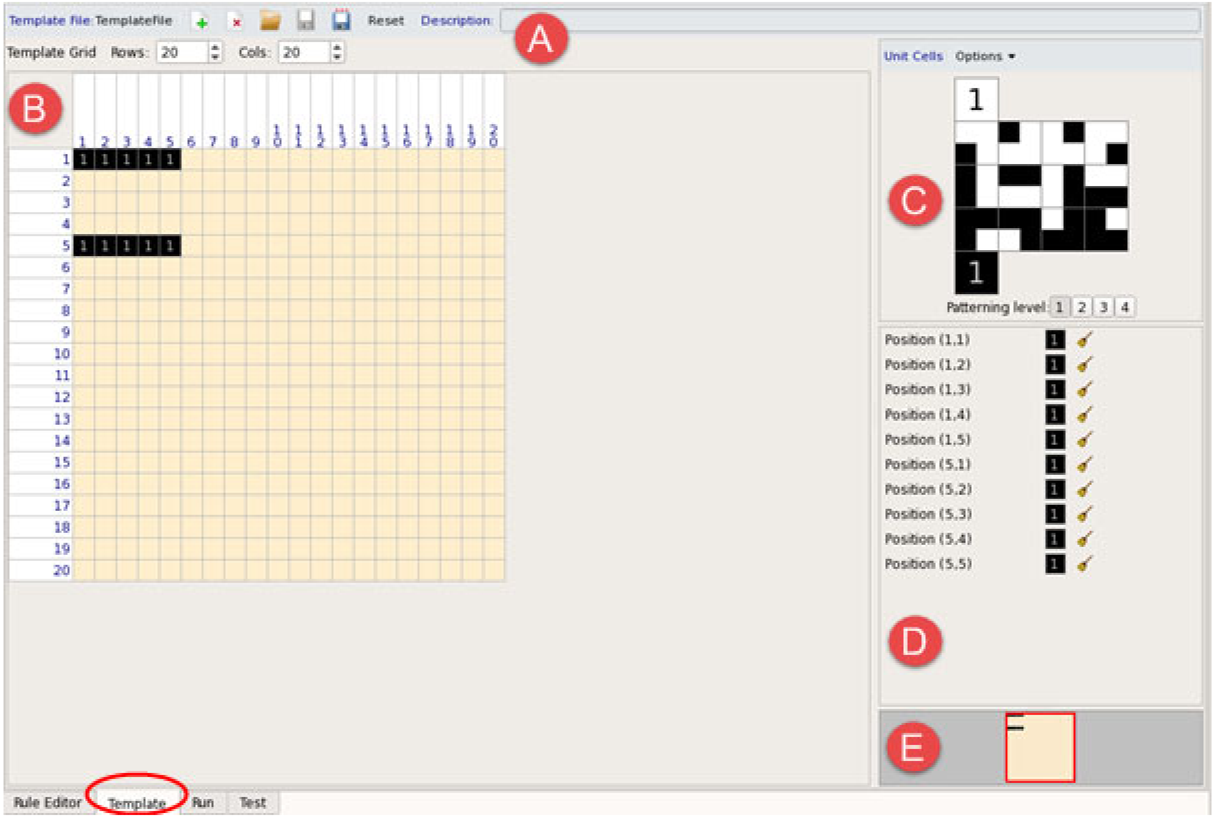
To access: From a Calibre WORKbench menu bar, **Litho** > **Calibre LSG Layout Generation**, or from a Calibre WORKbench Tcl shell call—see “Invoking Calibre LSG from a Tcl Shell Call” on page 16.

Click the **Template** tab.

Use this tab to predefine Calibre LSG unit pattern positions in a layout clip template file.

Description

Figure 4-3. Template Tab



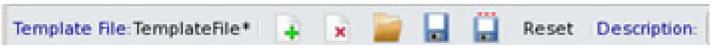
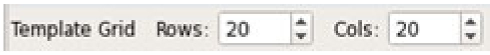
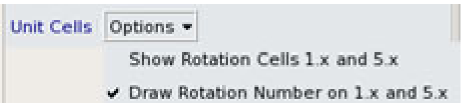
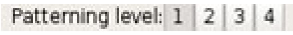
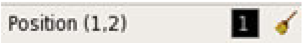
Template file: Templatefile

Template Grid Rows: 20 Cols: 20

Position (1,1) 1
Position (1,2) 1
Position (1,3) 1
Position (1,4) 1
Position (1,5) 1
Position (5,1) 1
Position (5,2) 1
Position (5,3) 1
Position (5,4) 1
Position (5,5) 1

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

Objects

Object	Description
A	**Template File pane** — Use to perform template file and pattern actions. Definitions (left to right):  • **Create New Template File** — Open (+) a new template file. • **Close Template File** — Close (X) current template file. • **Import Template File** — Import a saved template file. • **Save Template File** — Save the current template file. • **Save as Template File** — Save the current template file in a specified location. • **Reset Template** — Reset the template pattern in the Template Grid. • **Description** — Add a description for the template file.
B	**Template Grid pane** — Use to perform unit cell placement on a grid and create and display a template. Definitions (left to right):  • **Rows:** — Specify the number of rows in the template. • **Cols:** — Specify the number of columns in the template.
C	**Unit Cells pane** — Use to define and drag-select unit cells for placement in the Template Grid. Definitions (top to bottom): • **Options** — Select unit cell rotation options from the dropdown list.  Unit Cells Options Show Rotation Cells 1.x and 5.x ✔ Draw Rotation Number on 1.x and 5.x • **Patterning Level** — Display unit cell combinations for each mask layer (up to four levels). 
D	**Position pane** — Display position, unit pattern definition, and delete (clear) icon. 
E	**Reference pane** — Show the active template information.

Usage Notes

For a description of the file formats and argument options used for imported files see "calibre -lsg" on page 26 and "Calibre LSG File Formats" on page 42.

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

Run Tab

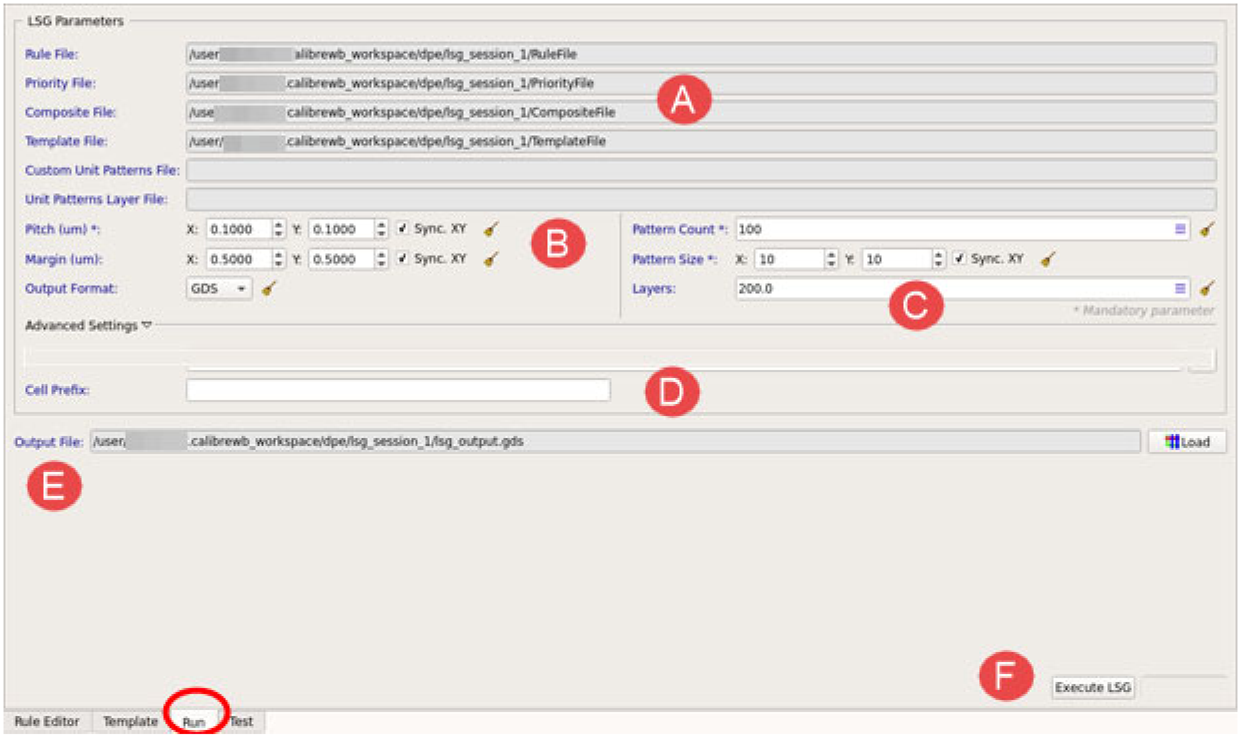
To access: From a Calibre WORKbench menu bar, **Litho** > **Calibre LSG Layout Generation**, or from a Calibre WORKbench Tcl shell call—see “Invoking Calibre LSG from a Tcl Shell Call” on page 16.

Click the **Run** tab.

Use this tab to fill in the required Calibre LSG files and layout clip information to execute a run and load the output in Calibre WORKbench.

Description

Figure 4-4. Run Tab



LSG Parameters

Rule File: /user[redacted]alibrewb_workspace/dpe/lsg_session_1/RuleFile

Priority File: /user[redacted].calibrewb_workspace/dpe/lsg_session_1/PriorityFile

Composite File: /use[redacted]calibrewb_workspace/dpe/lsg_session_1/CompositeFile

Template File: /user/[redacted].calibrewb_workspace/dpe/lsg_session_1/TemplateFile

Custom Unit Patterns File:

Unit Patterns Layer File:

Pitch (um) *: X: 0.1000 Y: 0.1000 ☑ Sync. XY

Margin (um): X: 0.5000 Y: 0.5000 ☑ Sync. XY

Output Format: GDS

Pattern Count *: 100

Pattern Size *: X: 10 Y: 10 ☑ Sync. XY

Layers: 200.0

* Mandatory parameter

Cell Prefix:

Output File: /user[redacted].calibrewb_workspace/dpe/lsg_session_1/lsg_output.gds

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

Objects

Object	Description
A	**LSG Parameters pane** — Use to display the location of the input files and parameters for your run. Files: • **Rule File** — Specify the LSG rule file. • **Priority File** — Specify the unit cell priority (weight) file. File must contain at least one non-zero weight. • **Composite File** — Specify the unit cell composite patterns file. • **Template File** — Specify the clip template file. • **Custom Unit Patterns File** — Specify a file containing custom unit pattern rules. • **Unit Patterns Layer File** — Specify the layers used for the unit patterns.
B	Parameters: **Note:** The wildcard (*) denotes a mandatory parameter. • **Pitch (um) *** — Specify the edge lengths in microns of a unit pattern (the pitch size). These values must be positive floating-point numbers 0.002 microns or greater. **Sync. XY** is checked by default. • **Margin (um)** — Specify the horizontal and vertical space in microns between unit cell patterns. These values must be positive floating-point numbers if **Sync. XY** is checked (default). If margin values are not specified, the default margin space is 2 * pitch size. • **Output Format** — Specify the output format of the layout clip (OASIS[1] or GDS). The default is OASIS.
C	Parameters: **Note:** The wildcard (*) denotes a mandatory parameter. • **Pattern Count *** — Specify the number of the layout clips that are output. This value must be a positive integer. • **Pattern Size *** — Specify the size of one layout clip in terms of the number of unit patterns used to form the clip side. This value must be a positive integer. **Sync. XY** is checked by default. • **Layers** — Specify the layout layer number for the layout clips. Specify this number as a positive integer. The default is 200.
D	Advanced Settings: **Cell Prefix** — Specify a cell name prefix for each output layout clip.
E	**Output File** — Specify the name of the output database file for the layout clips. A file extension of *.gds* or *.oas* is added depending on the database format. The default is *.oas*.

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

Object	Description
F	**Execute LSG** — Perform the Calibre LSG run. When the run completes, you can use the **Load** button to load the layout database for viewing in Calibre WORKbench.

[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.

Usage Notes

For a description of the file formats and argument options used for imported files see "calibre -lsg" on page 26 and "Calibre LSG File Formats" on page 42.

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

Test Tab

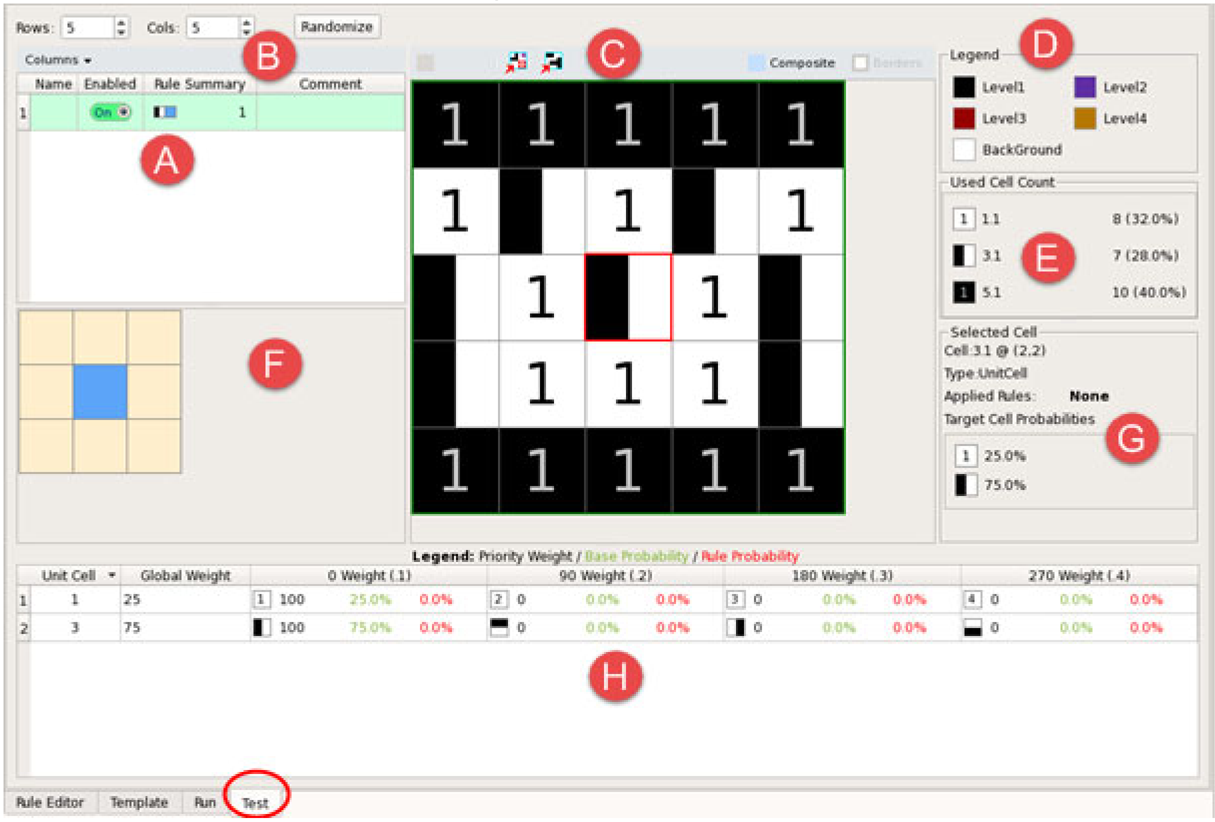
To access: From a Calibre WORKbench menu bar, **Litho** > **Calibre LSG Layout Generation**, or from a Calibre WORKbench Tcl shell call—see “Invoking Calibre LSG from a Tcl Shell Call” on page 16.

Click the **Test** tab.

Use this tab to optionally test your Calibre LSG set up before executing a run.

Description

Figure 4-5. Test Tab



	Name	Enabled	Rule Summary	Comment
1		On	1	

Legend

- Level1
- Level2
- Level3
- Level4
- BackGround

Used Cell Count

1	1.1	8 (32.0%)
◧	3.1	7 (28.0%)
1	5.1	10 (40.0%)

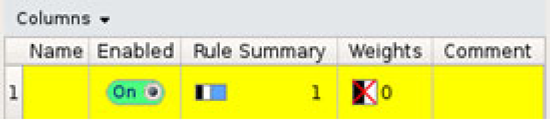
Selected Cell
Cell:3.1 @ (2,2)
Type:UnitCell
Applied Rules: **None**
Target Cell Probabilities

1	25.0%
◧	75.0%

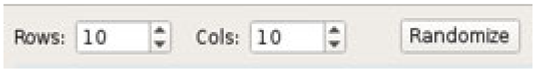
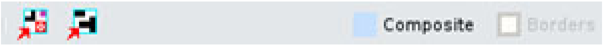
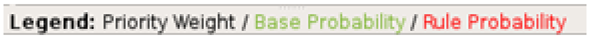
Legend: Priority Weight / Base Probability / Rule Probability

	Unit Cell ▾	Global Weight	0 Weight (.1)			90 Weight (.2)			180 Weight (.3)			270 Weight (.4)		
1	1	25	1 100	25.0%	0.0%	2 0	0.0%	0.0%	3 0	0.0%	0.0%	4 0	0.0%	0.0%
2	3	75	◧ 100	75.0%	0.0%	⬒ 0	0.0%	0.0%	◨ 0	0.0%	0.0%	⬓ 0	0.0%	0.0%

Objects

Object	Description
A	**Rule Table pane** — Select and display rule column information.  Columns ▾ Name · Enabled · Rule Summary · Weights · Comment 1 · On · 1 · 0

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

Object	Description
B	**LSG Test pane** — Enter row and column count and click **Randomize** to run an LSG setup test.  Rows: 10 Cols: 10
C	**Test pattern grid pane** — Create rules and composite patterns interactively by selecting cells in the test pattern grid. Definitions (left to right):  • Create Rule using selected cell as target. • Create Composite pattern using selected cell as the (upper-left) corner of the pattern. • Show Composite Pattern Mask. • Show Border Mask.
D	**Legend pane** — Show mask layer legend colors.
E	**Used Cell Count pane** — Show the number of each unit cell definition used in the pattern.
F	**Rule reference pane** — Display target and neighbor cell information for the selected rule in the Rule Table.
G	**Selected Cell pane** — Show unit cell type, applied rules, and target cell probability information for the selected cell in the test pattern grid.
H	**Legend: pane** — Display priority (weight) information and probability information for each unit cell and its rotation and rules in the test pattern.  Legend: Priority Weight / Base Probability / Rule Probability Definitions (left to right): • **Priority Weight** — Display the priority (weight) for a unit cell rotation. • **Base Probability** — Display the probability (percentage) of unit cells with that weight and rotation. • **Rule Probability** — Display the probability (percentage) that a rule applies to unit cells with the specified weight and rotation.

Usage Notes

For a description of the file formats and argument options used for imported files see “calibre -lsg” on page 26 and “Calibre LSG File Formats” on page 42.

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

Using the Calibre LSG GUI for Rule Writing and Editing

You can start a new session for creating and editing rules or import an existing session in the Calibre LSG GUI.

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.

Procedure

1. Invoke Calibre WORKbench and open the Calibre LSG Layout Generation window using one of these methods:

   a. "Invoking Calibre LSG from the GUI" on page 16.

   b. "Invoking Calibre LSG from a Tcl Shell Call" on page 16.

2. Choose one of the following actions:

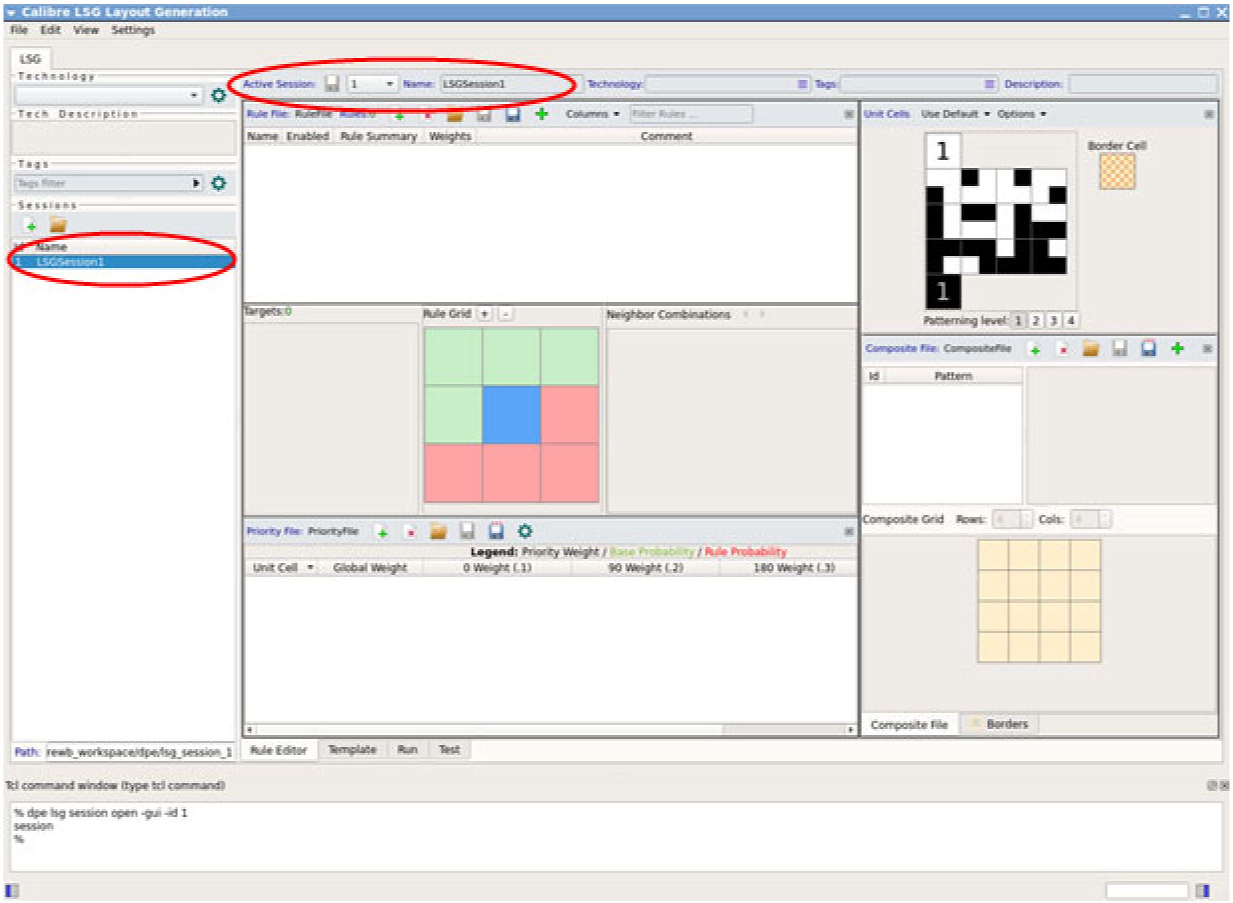
   a. To create a new session, click the green plus sign (+) icon in the Sessions pane of the **Rule Editor** tab.

   a. To import an existing session, click the folder icon in the Sessions pane of the **Rule Editor** tab.

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

Results

The **Active Session** Id number and name are displayed in the Sessions pane and in the top menu bar of the Calibre LSG Layout Generation window. The Calibre LSG GUI fields are ready for rule writing and editing tasks.


Calibre LSG Layout Generation
File Edit View Settings
LSG
Technology
Tech Description
Tags
Sessions
Rule File: RuleFile
Columns
Name Enabled Rule Summary Weights Comment
Targets: 0
Rule Grid
Neighbor Combinations
Priority File: PriorityFile
Legend: Priority Weight / Base Probability / Rule Probability
Unit Cell Global Weight 0 Weight (.1) 90 Weight (.2) 180 Weight (.3)
Unit Cells Use Default Options
Border Cell
Patterning level: 1 2 3 4
Composite File: CompositeFile
Id Pattern
Composite Grid Rows: Cols:
Composite File Borders
Rule Editor Template Run Test

Active Session: 1 Name: LSGSession1

Sessions:

Id	Name
1	LSGSession1

Path: rewb_workspace/dpe/lsg_session_1

Tcl command window (type tcl command)

```
% dpe lsg session open -gui -id 1
session
%
```

Defining Priorities and Weights

You can use the Calibre LSG GUI to select the unit cell patterns and specify their priority (weight), percent usage and rotation in the generated layout clips.

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.
- Open a session in the Calibre LSG GUI **Rule Editor** tab as described in "Using the Calibre LSG GUI for Rule Writing and Editing" on page 122.

Procedure

1. Click the green plus sign (+) icon in the **Priority File** menu bar to create a new priority (weight) file.

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

2. Click the gear icon in the **Priority File** menu bar to open the Priority File Configuration dialog box.

    a. Select the mask level tab (**Level 1** is the default).

    b. Toggle your selected Unit Cell and Cell Rotations in the Use column to **On**. For example, toggle Unit Cell 1 and Unit Cell 3 to **On**.

3. Enter the global weights in the Priority File pane Global Weight column for each of the selected unit cell patterns. For example, in the Global Weight column, enter “25” for Unit Cell 1 and “75” for Unit Cell 3 for a 1:3 usage ratio.

    The global weights are specified as positive integers with each value representing the relative weight between unit cell pattern usage in the output layout clip. Unspecified weights default to zero (no priority).

    Note

    A priority file with all weights specified as zero is not allowed.

4. Enter the rotation weights in the Priority File pane columns (0 Weight, 90 Weight, 180 Weight, and 270 Weight) for each of the selected unit cell patterns. For example, in the 0 Weight column, enter a weight of 100 for Unit Cell 1 and a weight of 100 for Unit Cell 3. This means that Unit Cell 1 and Unit Cell 3 are equally weighted and have no rotation.

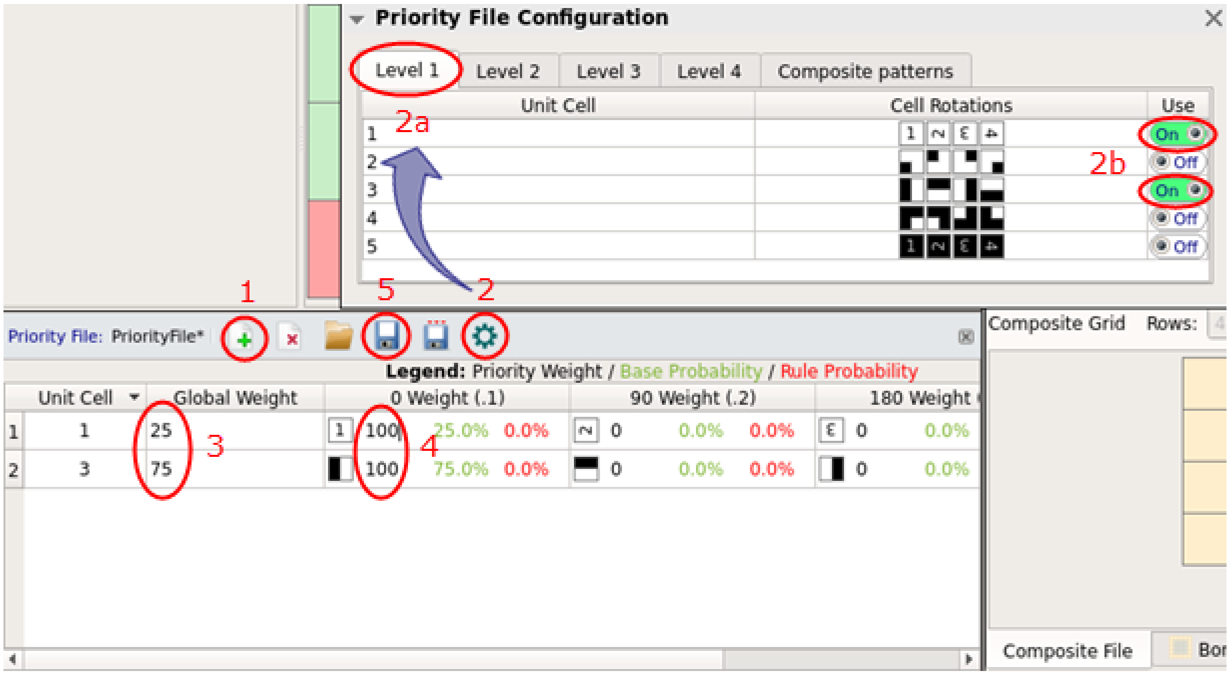
    The rotation weights are specified as positive integers with each value representing the relative weight between unit cell pattern rotations in the output layout clip. Unspecified weights default to zero (no priority).

5. Click the “Save Priority File” icon in the Priority File menu bar to save your entries to a priority (weight) file.

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

Results

The priority weights for the unit cell patterns 1.1 and 3.1 with usage ratio of 1:3 and 0 degree rotation are saved to a priority (weight) file.



	Unit Cell	Cell Rotations	Use
1			On
2			Off
3			On
4			Off
5			Off

Legend: Priority Weight / Base Probability / Rule Probability

	Unit Cell	Global Weight	0 Weight (.1)			90 Weight (.2)			180 Weight	
1	1	25	100	25.0%	0.0%	0	0.0%	0.0%	0	0.0%
2	3	75	100	75.0%	0.0%	0	0.0%	0.0%	0	0.0%

Creating Calibre LSG Rules

You can use the Calibre LSG GUI to create rules for the placement of unit cell patterns in the generated layout clips.

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.
- Select the unit cell patterns and specify their priority (weight), percent usage, and rotation as described in "Defining Priorities and Weights" on page 123.

Procedure

1. Open a session in the Calibre LSG GUI **Rule Editor** tab.
2. Click the green plus sign (+) icon (left side) of the Rule File menu bar to create a new rule file.
3. Click the green plus sign (+) icon (right side) of the Rule File menu bar to add a new rule.

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

4. For example, to create a rule that prevents unit cell pattern 3.1 from being placed beside another instance of unit cell pattern 3.1 (minimum space rule between same-mask polygons), perform the following steps:

    a. Click and drag unit cell pattern 3.1 from the Unit Cells pane to the Rule Grid pane, and place it to the left (-1, 0) of the center target position (0,0).

    b. Click and drag another instance of unit cell pattern 3.1 from the Unit Cells pane to the Rule Grid pane, and place it in the center target position (0,0).

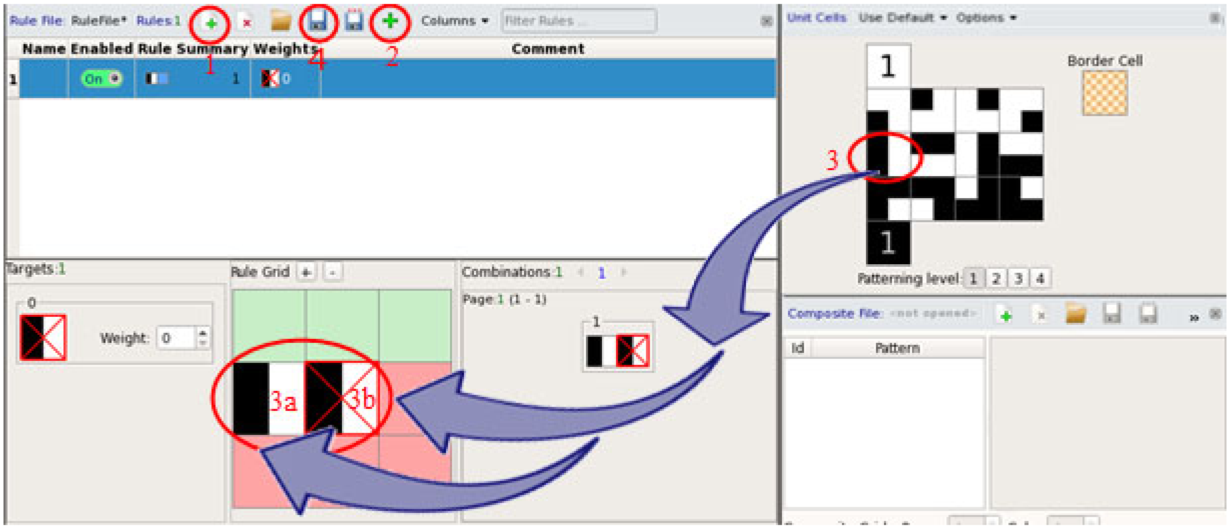
    This prevents unit cell pattern 3.1 from being placed beside any instance of unit cell pattern 3.1 and creates the following rule:

    ```
    [{-1,0},{3.1}];[{3.1,0};]
    ```

5. Click the “Save Rule File” icon to save the rule to an LSG rule file.

Results

The rule that prevents the placement of unit cell pattern 3.1 from being placed beside another unit cell pattern 3.1 is added to the LSG rule file.



Weighting Specific Patterns

You can use the Calibre LSG GUI to add extra weight for prioritizing specific patterns in the generated layout clips. Adding weight to certain patterns can replicate patterns similar to vertical and horizontal routing.

Prerequisites

- Refer to “Calibre LSG Requirements” on page 13.

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

- Open a session in the Calibre LSG GUI **Rule Editor** tab as described in "Using the Calibre LSG GUI for Rule Writing and Editing" on page 122.
- Select the unit cell patterns and specify their priority (weight), percent usage and rotation as described in "Defining Priorities and Weights" on page 123.

Procedure

1. Click the green plus sign (+) icon (left side) of the **Rule File** menu bar to create a new rule file.
2. Click the green plus sign (+) icon (right side) of the **Rule File** menu bar to add a new rule.
3. For example, to create a rule that adds extra weight to a pattern with unit cell pattern 3.1 placed on top of another unit cell pattern 3.1 (similar to a vertical routing pattern), perform the following steps:

   a. Click and drag unit cell pattern 3.1 from the Unit Cells pane to the Rule Grid pane, and place it above (0,1) the center target position (0,0).

   b. Click and drag another instance of unit cell pattern 3.1 from the Unit Cells pane to the Rule Grid pane, and place it in the center target position (0,0).

   c. Select a weight of "10" for this specific pattern from the **Weight** dropdown list.

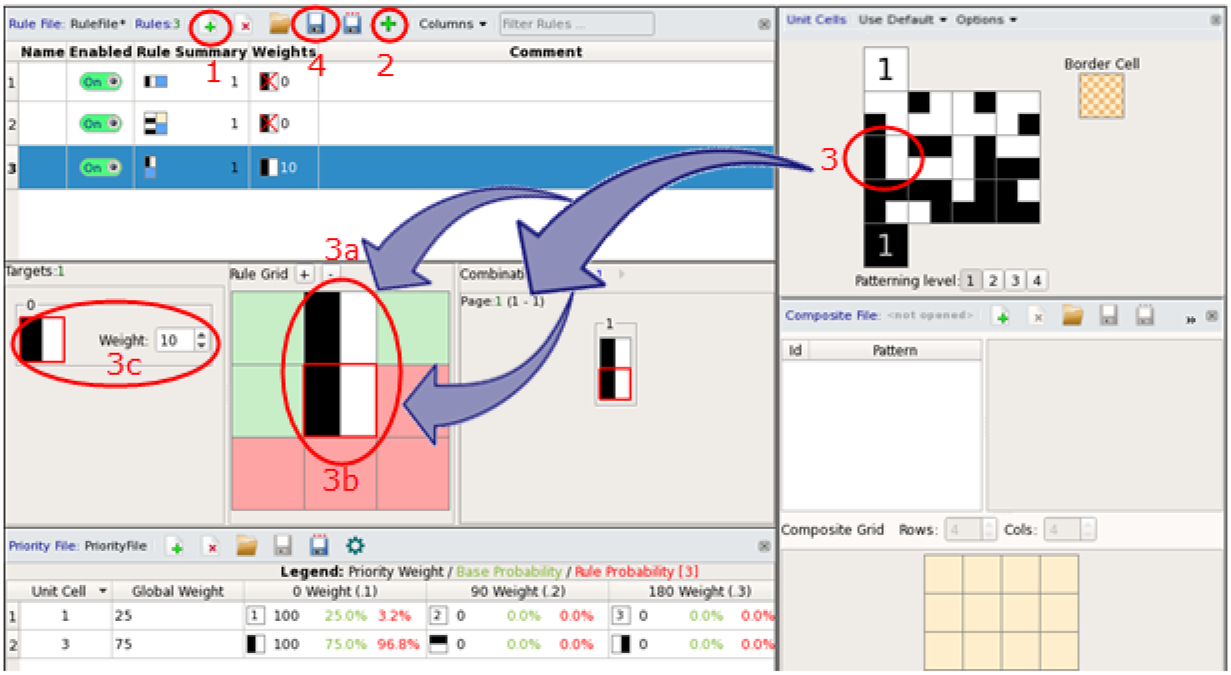
   This adds a weight of 10 for the specific pattern and creates the following rule:

   ```
   [{0,1},{3.1}];[{3.1,10};]
   ```

4. Click the "Save Rule File" icon to save the rule to an LSG rule file.

Results

The rule that adds an extra weight of 10 to a specific pattern combination of unit cell 3.1 similar to vertical routing is added to the LSG rule file.



Rule File: Rulefile* Rules:3

	Name	Enabled	Rule Summary		Weights	Comment
1		On		1	0	
2		On		1	0	
3		On		1	10	

Targets: 1

Weight: 10

Page: 1 (1 - 1)

Priority File: PriorityFile

Legend: Priority Weight / Base Probability / Rule Probability [3]

	Unit Cell	Global Weight	0 Weight (.1)			90 Weight (.2)			180 Weight (.3)		
1	1	25	100	25.0%	3.2%	0	0.0%	0.0%	0	0.0%	0.0%
2	3	75	100	75.0%	96.8%	0	0.0%	0.0%	0	0.0%	0.0%

Creating a Unit Pattern Template

You can use the Calibre LSG GUI to predefine specific unit cell pattern positions and create a template file for the generated layout clips.

Prerequisites

- Refer to “Calibre LSG Requirements” on page 13.
- Open a session in the Calibre LSG GUI **Rule Editor** tab as described in “Creating Calibre LSG Rules” on page 125.
- Select the unit cell patterns and specify their priority (weight), percent usage, and rotation as described in “Defining Priorities and Weights” on page 123.

Procedure

1. Select the **Template** tab in the Calibre LSG Layout Generation window.
2. Click the green plus sign (+) icon in the **Template File** menu bar to create a new template file.
3. Choose the number of rows and columns for the template pattern from the dropdown lists in the Template Grid pane.

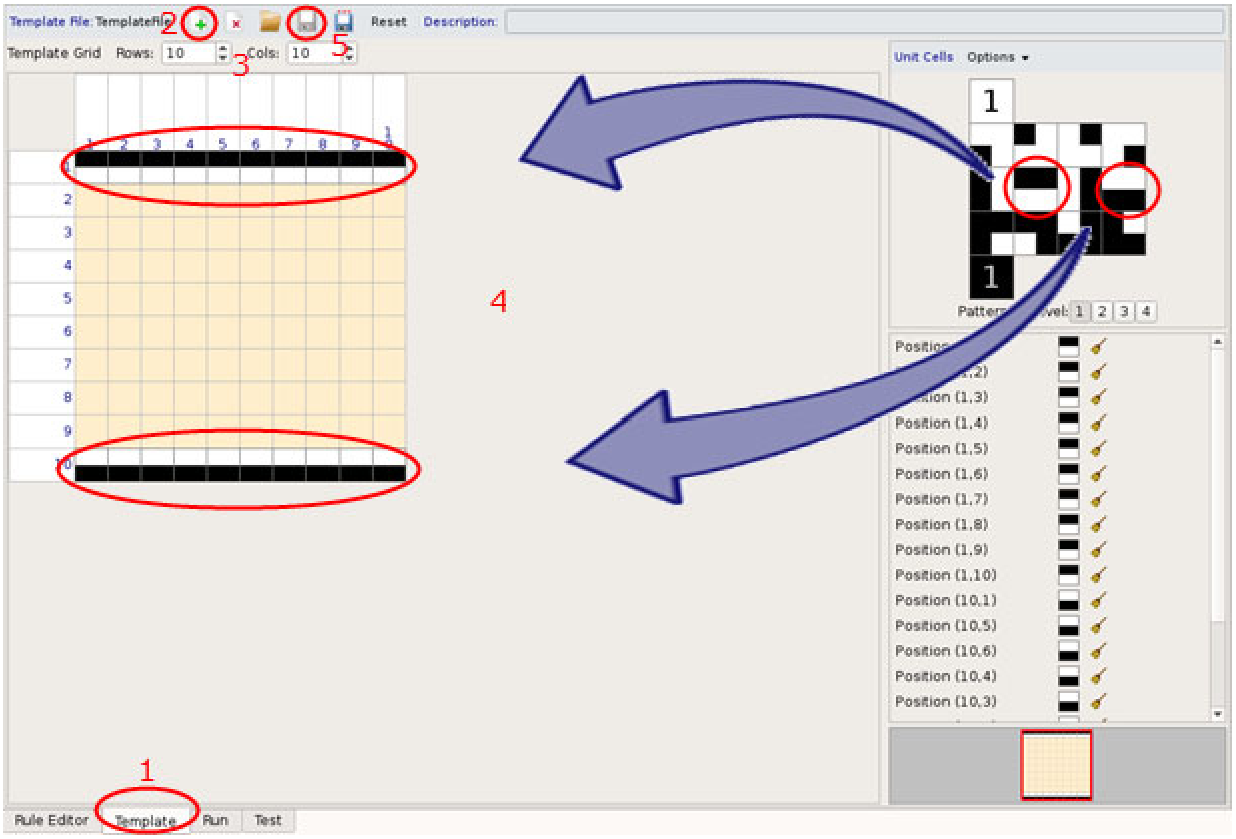
Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.

4. Click and drag unit cell patterns from the Unit Cells pane to the Template Grid pane to create the template pattern.

5. Click the “Save Template File” icon to save the rule to an LSG template file.

Results

The template pattern with unit cell patterns 3.2 in the top row and unit cell patterns 3.4 in the bottom row is saved to the LSG template file.



Testing the Calibre LSG Setup

You can use the Calibre LSG GUI to test the rules and weights in random unit cell pattern combinations before you execute the run.

Prerequisites

- Refer to “Calibre LSG Requirements” on page 13.
- Open a session in the Calibre LSG GUI **Rule Editor** tab as described in “Creating Calibre LSG Rules” on page 125.
- Select the unit cell patterns and specify their priority (weight), percent usage, and rotation as described in “Defining Priorities and Weights” on page 123.

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

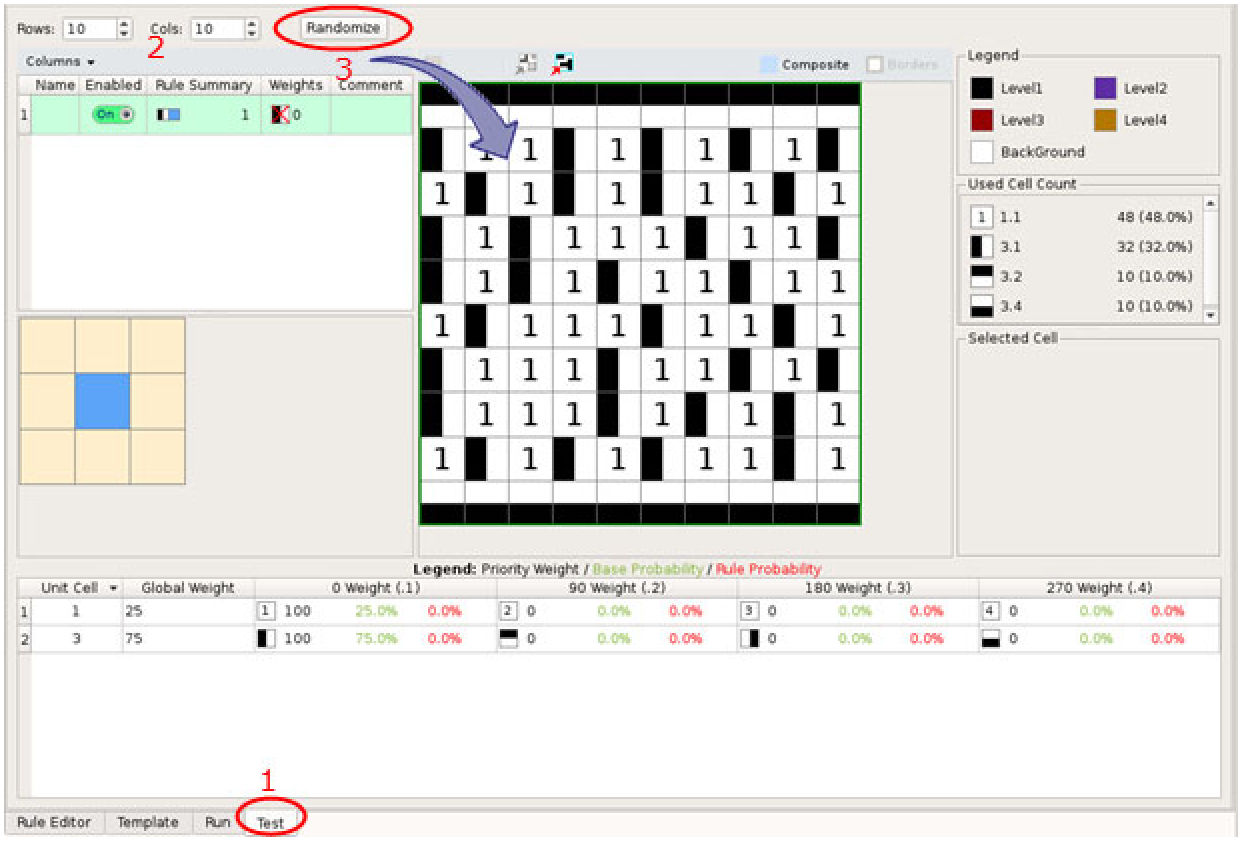
- Optionally, predefine specific unit cell pattern positions as described in “Creating a Unit Pattern Template” on page 128.

Procedure

1. Select the **Test** tab in the Calibre LSG GUI.
2. Choose the number of rows and columns for the test pattern from the dropdown lists in the Template Grid pane.
3. Click **Randomize** to run the LSG test.

Results

The Calibre LSG GUI test utility generates random placement combinations of the selected unit cell patterns using rules, weights, and template information.



Rows: 10 Cols: 10

	Name	Enabled	Rule Summary	Weights	Comment
1		On	1	0	

Legend

■	Level1	■	Level2
■	Level3	■	Level4
□	BackGround		

Used Cell Count

1	1.1	48 (48.0%)
▌	3.1	32 (32.0%)
▀	3.2	10 (10.0%)
▄	3.4	10 (10.0%)

Legend: Priority Weight / Base Probability / Rule Probability

	Unit Cell	Global Weight	0 Weight (.1)			90 Weight (.2)			180 Weight (.3)			270 Weight (.4)		
1	1	25	1 100	25.0%	0.0%	2 0	0.0%	0.0%	3 0	0.0%	0.0%	4 0	0.0%	0.0%
2	3	75	▌ 100	75.0%	0.0%	▀ 0	0.0%	0.0%	▐ 0	0.0%	0.0%	▄ 0	0.0%	0.0%

Running Calibre LSG from the GUI

You can use the Calibre LSG GUI to specify the run input information, perform the run, and load the output layout clips to Calibre WORKbench.

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

Prerequisites

- Refer to "Calibre LSG Requirements" on page 13.
- Open a session in the Calibre LSG GUI **Rule Editor** tab as described in "Creating Calibre LSG Rules" on page 125.
- Select the unit cell patterns and specify their priority (weight), percent usage, and rotation as described in "Defining Priorities and Weights" on page 123.
- Optionally, predefine specific unit cell pattern positions as described in "Creating a Unit Pattern Template" on page 128.

Procedure

1. Select the **Run** tab in the Calibre LSG GUI.
2. Review the paths for the input files for the run. The files are located by default under the following path from your home directory:

   ```
   /user/<name>/.calibrewb_workspace/dpe/lsg_session_<number>
   ```

   Note

   The Priority File path must specify a priority file with non-zero entries, or an error is issued by the tool.

3. Enter the Pitch (um) values for your design in positive floating-point numbers. These values must be a positive floating-point numbers 0.002 microns or greater. Leave **Sync. XY** checked if the pitch values are the same in the x- and y-dimensions.
4. Optionally, enter the Margin (um) values for your layout clips. If **Sync. XY** is checked (default), these values must be positive floating-point numbers with the same x- and y-dimensions. If margin values are unspecified, the default margin space is 2 * pitch size
5. Choose either **OASIS** or **GDS** from the Output Format dropdown list. The default format is OASIS.
6. Enter the Pattern Count information for your layout clips.

   a. Click the three bars at the right of the field and choose either **Pattern Count** or **Pattern Grid** in the dialog box.

   b. Enter the count or the row and column numbers manually (positive integers) or select the numbers from the dropdown list.

   c. Click **Save** to close the dialog box.
7. Enter the Pattern Size information for your layout clips (positive integers). Leave **Sync. XY** checked if the values are the same in the x- and y-dimensions.
8. Specify the Layers information for your layout clips.

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

   a. Click the three bars at the right of the Layers field and enter the layer information for your layout clips in the Set Layer(s) dialog box. The default layer number is 200.0.

9. (Optional) Click the Advanced Settings dropdown for additional run options.
10. Click **Execute LSG** to open the LSG Execution dialog box and specify one of the following output file options:

    a. Check the **Specify Output Layout File** check box and fill in the name of the layout file.

    b. Leave the **Specify Output Layout File** check box unchecked (default) and use the -replace option to use the default layout filename *lsg_output*.

    The output file is located by default under the following path from your home directory:

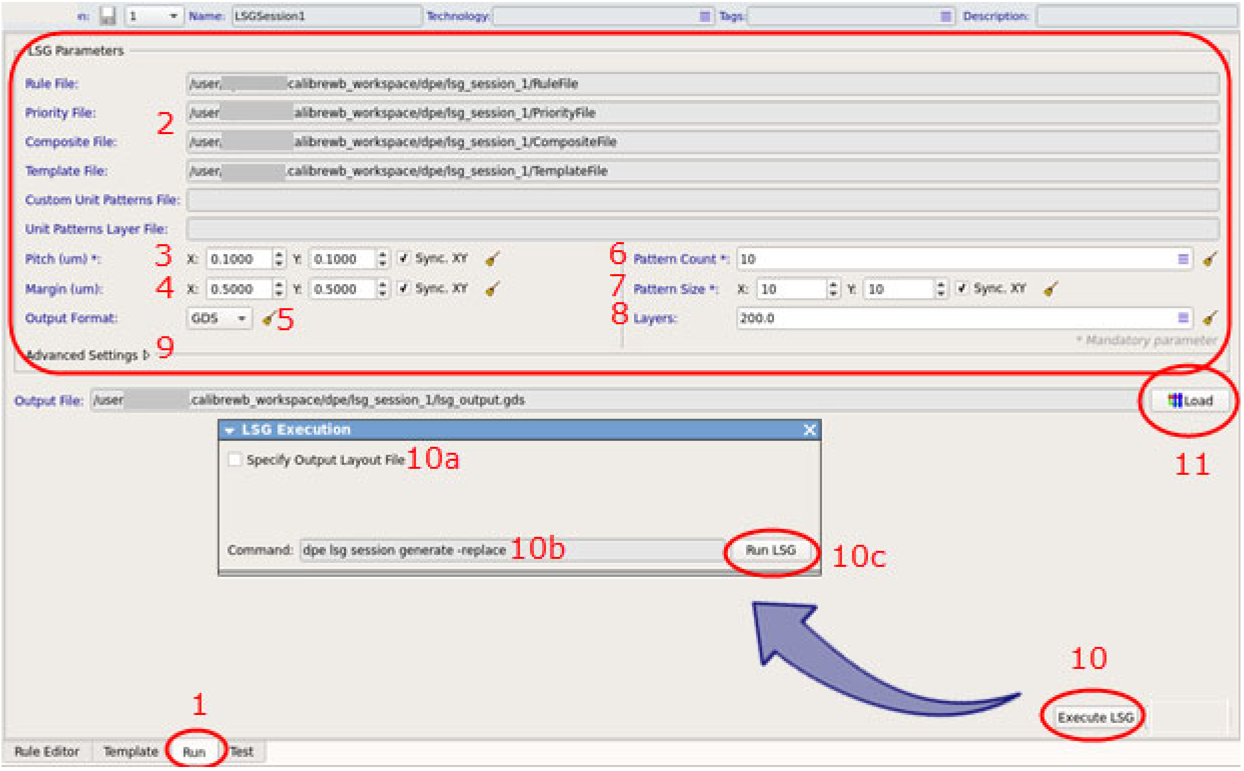
    ```
    /user/<name>/.calibrewb_workspace/dpe/lsg_session_<number>
    ```

    c. Click **Run LSG** to start the run.

11. Click the **Load** button to load the output layout clips in the Calibre WORKbench GUI window.

Results

The Calibre LSG GUI is used to execute the run and generate the layout clips using the required and optional input files and parameter information.



Name: LSGSession1

LSG Parameters

Rule File: /user/[redacted]calibrewb_workspace/dpe/lsg_session_1/RuleFile

Priority File: /user[redacted]alibrewb_workspace/dpe/lsg_session_1/PriorityFile

Composite File: /user/[redacted]alibrewb_workspace/dpe/lsg_session_1/CompositeFile

Template File: /user/[redacted]calibrewb_workspace/dpe/lsg_session_1/TemplateFile

Custom Unit Patterns File:

Unit Patterns Layer File:

Pitch (um) *: X: 0.1000 Y: 0.1000 ☑ Sync. XY

Margin (um): X: 0.5000 Y: 0.5000 ☑ Sync. XY

Output Format: GDS

Pattern Count *: 10

Pattern Size *: X: 10 Y: 10 ☑ Sync. XY

Layers: 200.0

* Mandatory parameter

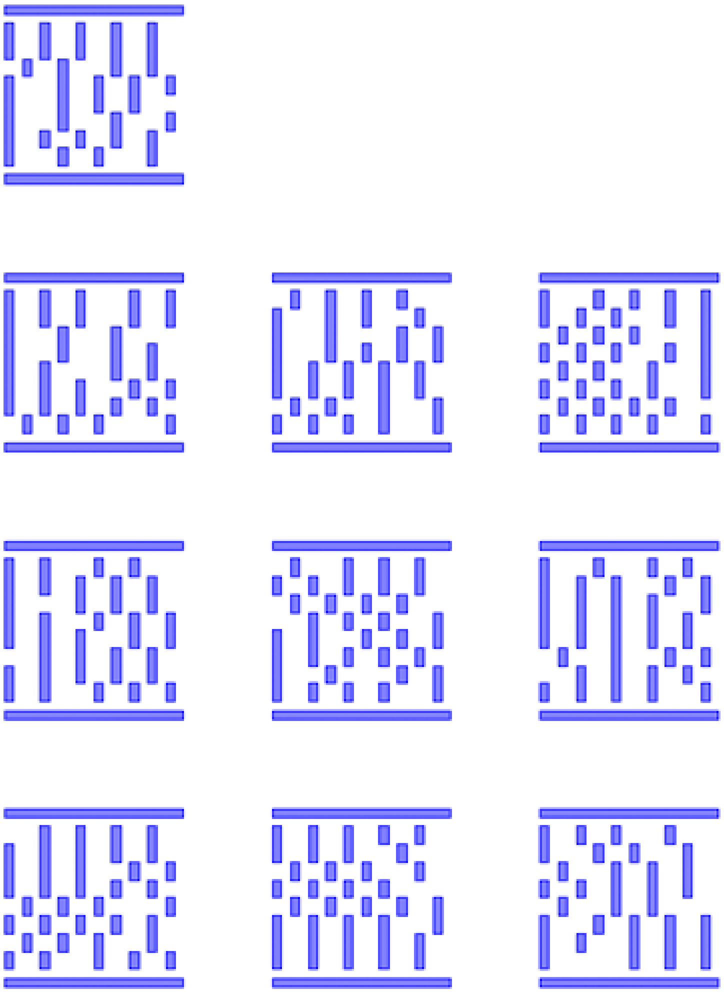
Output File: /user[redacted]calibrewb_workspace/dpe/lsg_session_1/lsg_output.gds

LSG Execution

□ Specify Output Layout File

Command: dpe lsg session generate -replace

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



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

Index

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.

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