
KLayout Photonic PCells Documentation

Release 0.1.2

Sebastian Goeldi

Jul 23, 2019

CONTENTS:

1	Introduction	1
2	First Steps	3
2.1	Prerequisites	3
2.2	Installation	3
3	Ports	5
4	Technology Import	7
4.1	Import Techfile & Creation of LayerProperties	7
4.2	Import of example Vias	9
4.3	Layermap	9
5	Code Documentation	11
5.1	drc Module	11
5.1.1	Module contents	11
5.1.2	Submodules	11
5.1.3	Multiprocessing	13
5.2	photronics Module	15
5.2.1	Module contents	15
5.2.2	Submodules	15
6	Example: Create Sample Library	17
6.1	Code Explanation	17
7	Tips & Tricks	25
7.1	Variable Names in KLayout Python	25
8	C++ Source Code	27
8.1	DrcSl Source	27
8.2	CleanerMaster Source	41
8.3	CleanerMain Source	55
8.4	CleanerSlave Source	56
	Index	59

INTRODUCTION

The KLayoutPhotonicPCells¹ module is an extension for KLayout PCells to facilitate photonic PCells. Photonics often works with the concept of ports. Ports are defined by a coordinate and a direction. In the case of this module ports will be stored in PCell parameters in the background. They are serialized KLayout Trans¹ objects. For an introduction on how to build your own PCell Library, have a look at how to create *Example Library*.

When building PCell Libraries it is recommended to build it with three packages as shown in Fig. 1.1

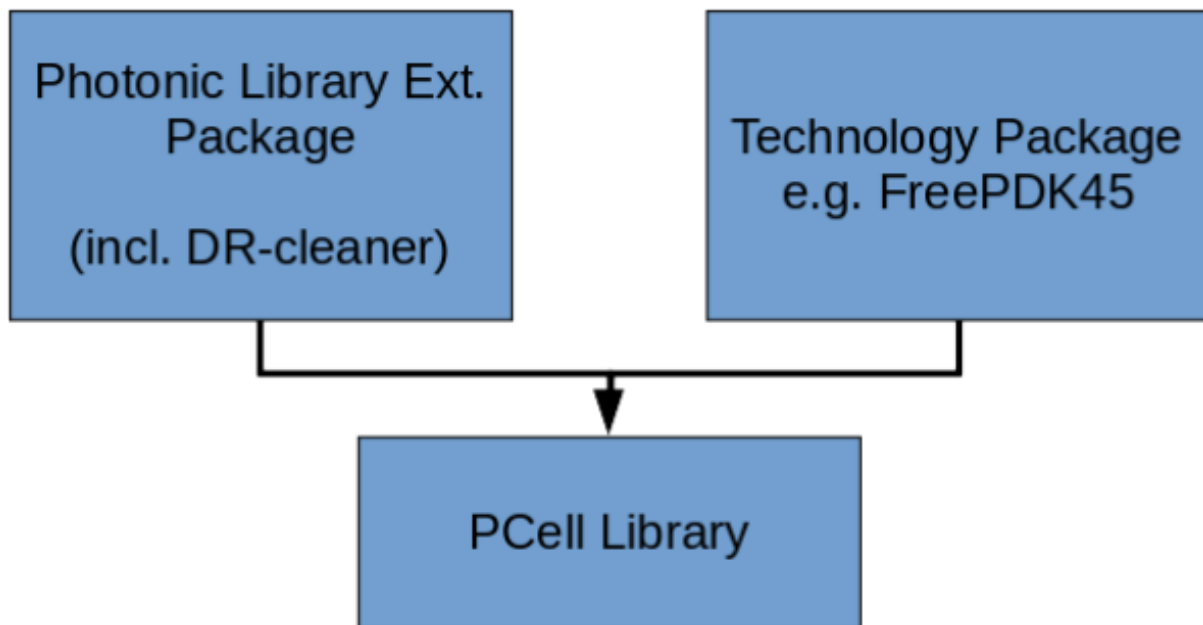


Fig. 1.1: The recommend structure for working with the photonic PCell extension: * Photonic Library Extension: New functionalities for KLayout PCells

- Ports, DR-Cleaning, DataPrep
- Technology: Contains manufacturer specific data
 - Design rules
 - Layermapping from abstract to manufacturer layers
- PCell-Library:
 - Definitions of PCells
 - Library specific modules if required

¹ https://www.klayout.de/doc/code/class_ICplxTrans.html

FIRST STEPS

2.1 Prerequisites

To use the library extension, make sure you have installed Cython. Part of the cleaning process relies on a C++ module that needs to be compiled first. To compile it we use python's [Setuptools](https://setuptools.readthedocs.io/en/latest/index.html)² and [Cython](https://cython.readthedocs.io/en/latest/index.html)³. Make sure you have these packages before starting. It is sufficient to install Cython, as setuptools is either built-in of python or installed along Cython.

2.2 Installation

This installation procedure is solely written for Linux. For this installation Cython is required. So get Cython either from the package manager of your distribution or through pip. The package is tested on Python 3.5+. No special python3 modules are used, therefore it should work with python 2.7, too. The Python version used should be the same KLayout uses. By default, this is the system interpreter for Python3. If you installed the package manually, move the unpackaged package into ~/.klayout/salt or into the KLayout folder if you used a custom directory. This tutorial assumes default pathes. After unpacking and moving you should have a ~/.klayout/salt/KLayoutPhotonicPCells/core folder. If you installed the FreePDK45_Cells & FreePDK45_tech, then you should have the folders ~/.klayout/salt/KLayoutPhotonicPCells/FreePDK45_ExampleCells and ~/.klayout/salt/KLayoutPhotonicPCells/FreePDK45_tech, too. The library extension package needs manual setup before being usable.

Use a console and execute the following commands. If you are familiar with setuptools you can skip these instructions. For further information consult the drc documentation.

```
cd ~/.klayout/salt/KLayoutPhotonicPCells/core/python/kppc/drc/  
sh compile.sh
```

² <https://setuptools.readthedocs.io/en/latest/index.html>

³ <https://cython.readthedocs.io/en/latest/index.html>

```
~/klayout/salt/zcmos/pcell/lib_ext/python/drc$ ./setup
rm: cannot remove 'slcleaner.cpy*': No such file or directory
running build_ext
cythoning slcleaner.pyx to slcleaner.cpp
building 'slcleaner' extension
creating build
creating build/temp.linux-x86_64-3.6
x86_64-linux-gnu-gcc -pthread -DDEBUG -g -fwrapv -O2 -Wall -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fPIC -I/usr/include/python3.6m -c slcleaner.cpp -o build/temp.linux-x86_64-3.6/slcleaner.o
In file included from slcleaner.cpp:584:
DrcSl.cpp: In member function 'int drclean::DrcSl::set_data(std::vector<drclean::edgecoord*>*)':
DrcSl.cpp:60:5: warning: no return statement in function returning non-void [-Wreturn-type]
    }
    ^
DrcSl.cpp: In member function 'void drclean::DrcSl::sortlist()':
DrcSl.cpp:134:21: warning: unused variable 'last_type' [-Wunused-variable]
    int last_type = 1;
    ~~~~~
DrcSl.cpp:135:21: warning: unused variable 'next_type' [-Wunused-variable]
    int next_type;
    ~~~~~
DrcSl.cpp:136:21: warning: variable 'type' set but not used [-Wunused-but-set-variable]
    int type;
    ~~~~~
DrcSl.cpp: In member function 'std::vector<int> drclean::DrcSl::get_types(int)':
DrcSl.cpp:189:13: warning: unused variable 'offset_d2' [-Wunused-variable]
    int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;
    ~~~~~
DrcSl.cpp: In member function 'int drclean::DrcSl::clean_space()':
DrcSl.cpp:295:14: warning: variable 'changed' set but not used [-Wunused-but-set-variable]
    bool changed = false;
    ~~~~~
DrcSl.cpp: In member function 'int drclean::DrcSl::clean_width()':
DrcSl.cpp:341:14: warning: variable 'changed' set but not used [-Wunused-but-set-variable]
    bool changed = false;
    ~~~~~
x86_64-linux-gnu-g++ -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-z,relro -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 build/temp.linux-x86_64-3.6/slcleaner.o -o ./slcleaner.cpython-36m-x86_64-linux-gnu.so
```

Fig. 2.1: Change directory to the drc folder and execute the setup script.

PORTS

Ports are a concept used in photonics. They are very similar to pins in electronics, as they both describe connections between cells. The big difference between ports and pins is ports have additional properties that are important for photonics. When connecting photonic devices it is necessary that the device connections are aligned. For example, if two waveguides are connected, the connected endings have to point on the opposite direction and the connections have to be the same size.

This module implements the concept of ports into KLayout PCells. Currently ports track location, orientation and length. If two ports have a mismatch in width, they cannot be connected. New ports can be created in PCells with the `kppc.photonics.PortCreation` when overriding the `kppc.photonics.PhotDevice.create_param_inst()` method in the PCell Library. If any instantiated child cells in a PCell have any open ports (not connected to another port of another child cell), they are passed upwards to the cell itself and are announced as ports of this cell. This hierarchical design allows to create arbitrary Devices independent of the order when assembling them.

Note: Make sure ports are drawn correctly. If texts in ports aren't oriented along the width of the port, set the boolean *Transform text with cell instance* in *File* → *Setup* → *Display* → *Cells* to true and make sure the text font is not set to the default font.

TECHNOLOGY IMPORT

To use KLayout and the Photonics-extension efficiently, it is recommended to create a KLayout technology. This chapter explains how to import a technology.

To use a new technology either create a new technology from the technology manager *Tools → Manage Technologies* or create a new package *Tools → Manage Packages* for the technology.

4.1 Import Techfile & Creation of LayerProperties

KLayout provides an import script for Cadence techfiles. This import creates the Layer Properties automatically for the defined layers.

The script can be found in *File → Import Cadence Techfile*

After importing, the properties can be saved via *File → Save Layer Properties*. Recommended location for the file is in the technology folder in `~/.klayout/tech/<technology-name>/<file>` or if using a package `~/.klayout/salt/<technology-package>/tech/<filename>`

Note: Suggested filename for easy use with the sample cells: FreePDK45.tf / FreePDK45.lyp

In order to use the additional abstract layers in the sample cells paste the following xml snippets into the `<>.lyp` file:

```
<properties>
  <frame-color>#01ff6b</frame-color>
  <fill-color>#01ff6b</fill-color>
  <frame-brightness>0</frame-brightness>
  <fill-brightness>0</fill-brightness>
  <dither-pattern>I3</dither-pattern>
  <line-style>I6</line-style>
  <valid>true</valid>
  <visible>true</visible>
  <transparent>false</transparent>
  <width>1</width>
  <marked>false</marked>
  <xfill>false</xfill>
  <animation>0</animation>
  <name>phot_silicon.drawing</name>
  <source>400/0@1</source>
</properties>
<properties>
  <frame-color>#808080</frame-color>
  <fill-color>#808080</fill-color>
  <frame-brightness>0</frame-brightness>
  <fill-brightness>0</fill-brightness>
  <dither-pattern>I2</dither-pattern>
  <line-style>I0</line-style>
```

(continues on next page)

(continued from previous page)

```

    <valid>true</valid>
    <visible>true</visible>
    <transparent>false</transparent>
    <width>1</width>
    <marked>false</marked>
    <xfill>false</xfill>
    <animation>0</animation>
    <name>phot_poly.drawing</name>
    <source>410/0@1</source>
  </properties>
  <properties>
    <frame-color>#ff0000</frame-color>
    <fill-color>#ff0000</fill-color>
    <frame-brightness>0</frame-brightness>
    <fill-brightness>0</fill-brightness>
    <dither-pattern>I9</dither-pattern>
    <line-style/>
    <valid>true</valid>
    <visible>true</visible>
    <transparent>false</transparent>
    <width>1</width>
    <marked>false</marked>
    <xfill>false</xfill>
    <animation>0</animation>
    <name>phot_pwell.drawing</name>
    <source>420/0@1</source>
  </properties>
  <properties>
    <frame-color>#0000ff</frame-color>
    <fill-color>#0000ff</fill-color>
    <frame-brightness>0</frame-brightness>
    <fill-brightness>0</fill-brightness>
    <dither-pattern>I5</dither-pattern>
    <line-style/>
    <valid>true</valid>
    <visible>true</visible>
    <transparent>false</transparent>
    <width>1</width>
    <marked>false</marked>
    <xfill>false</xfill>
    <animation>0</animation>
    <name>phot_nwell.drawing</name>
    <source>430/0@1</source>
  </properties>
  <properties>
    <frame-color>#ff0000</frame-color>
    <fill-color>#ff0000</fill-color>
    <frame-brightness>0</frame-brightness>
    <fill-brightness>0</fill-brightness>
    <dither-pattern>I11</dither-pattern>
    <line-style/>
    <valid>true</valid>
    <visible>true</visible>
    <transparent>false</transparent>
    <width>1</width>
    <marked>false</marked>
    <xfill>false</xfill>
    <animation>0</animation>
    <name>phot_pimplant.drawing</name>
    <source>440/0@1</source>
  </properties>

```

(continues on next page)

(continued from previous page)

```

<properties>
  <frame-color>#0000ff</frame-color>
  <fill-color>#0000ff</fill-color>
  <frame-brightness>0</frame-brightness>
  <fill-brightness>0</fill-brightness>
  <dither-pattern>I7</dither-pattern>
  <line-style/>
  <valid>true</valid>
  <visible>true</visible>
  <transparent>false</transparent>
  <width>1</width>
  <marked>false</marked>
  <xfill>false</xfill>
  <animation>0</animation>
  <name>phot_nimplant.drawing</name>
  <source>450/0@1</source>
</properties>

```

Put this block between the last properties block but before the end of the name block.

4.2 Import of example Vias

Importing a .LEF will create the layerproperties. The layerproperties are the layer-purpose-pairs of KLayout. When using the lef import script built into KLayout, it will automatically load example vias into a new layout. Unfortunately, the layers are not the correct layers from the technology files. The layers can be edited by selecting a layer in the layers sub-window and then editing the layer via *Edit* → *Layer* → *Edit Layer Specification*. Recommended place is in the `~/.klayout/tech/libraries` or if using a package: `~/.klayout/salt/<package-name>/tech/libraries`. These will automatically be loaded and are available as static cells for insert or in PCells.

4.3 Layermap

The .layermap file is usually supplied by the foundry. This file can be used in the `pcell_lib_ext` to use layernames instead of layer numbers in the PCell Library. It contains layernumber | layername | layerdatatype on each line for each layer. They have to be separated by white spaces. Afterwards, they can be used by the `self.add_layer(str varname, str layername)` function during the `__init__` of a new class of a PCell. Later the layer is accessible as `self.varname`.

Recommended place is again in the tech folder.

CODE DOCUMENTATION

5.1 drc Module

5.1.1 Module contents

5.1.2 Submodules

kppc.drc.slcleaner module

An interface to the DrcSl.cpp Class.

class kppc.drc.slcleaner.PyDrcSl

def add_data(x1, x2, y1, y2)

Insert data into the scanline cleaner. The data is an edge that will be manhattanized and cleaned.

Note: Edges should be added in such a way that the outwards face is left in the direction of p1 to p2. Klayout already does this nicely.

Parameters

- **x1** ([int](https://docs.python.org/3/library/functions.html#int)⁴) – x position of p1 of the edge
- **x2** ([int](https://docs.python.org/3/library/functions.html#int)⁵) – y position of p1 of the edge
- **y1** ([int](https://docs.python.org/3/library/functions.html#int)⁶) – x position of p2 of the edge
- **y2** ([int](https://docs.python.org/3/library/functions.html#int)⁷) – y position of p2 of the edge

clean(x = 10)

Clean data in the vector for space and width violations.

Parameters **x** – number of max tries

clean_space()

Clean the current data for space violations.

clean_width()

Clean the current data for width violations.

⁴ <https://docs.python.org/3/library/functions.html#int>

⁵ <https://docs.python.org/3/library/functions.html#int>

⁶ <https://docs.python.org/3/library/functions.html#int>

⁷ <https://docs.python.org/3/library/functions.html#int>

init_list(x1: int, x2: int, y1: int, y2: int, viospace: int, viowidth: int)

(Re-)Initialize the Cleaner. x1,2 and y1,2 define the bounding box of the cleaner.

Warning: If a corner or a complete edge is outside the bounding box and is added through the `add_data` function, a Segmentation Fault will most likely occur and the module (including Klayout) crashes. Alternatively, it will just be confined to the bounding box and the rest will be cut off.

Parameters

- **x1** – left bound of box
- **x2** – right bound of box
- **y1** – bottom bound of box
- **y2** (: integers) – top bound of box
- **viospace** (minimum space violation in database units) – minimum space violation in database units
- **viowidth** (minimum width violation in database units) – minimum width violation in database units

get_row(ind: int)

Get the edge data back to python from the C++ object.

Parameters **ind** – index of the row to retrieve data from

Returns numpy array of the edges

get_row_types(ind: int)

Get the type of edges in that row.

Parameters **ind** – index of the row

Returns numpy array of types of edges (0 for upwards facing edge, 1 for downwards)

polygons()

Returns list of crude polygons. The format is list of polygons, where a polygon is a list of tuples of (x,y)

Returns polygons in the form [(x1,y1),(x2,y2),...],...]

Return_type list

printvector(beg = -1, end = -1)

Print the data of rows/columns depending on current orientation

Parameters

- **beg** – beginning of the rows/columns that should be printed
- **end** – ending of the rows/columns that should be printed

s()

This property can be used to get the array size of the cleaner.

Returns Size of the array of vectors.

Return type `int`⁸

sort()

Sort the data in ascending order. This will also delete invalid edges, i.e. touching / overlapping polygons will be merged.

⁸ <https://docs.python.org/3/library/functions.html#int>

switch_dimensions()

Switch the orientation of the data. From row oriented to column oriented and vice-versa.

This wrapper is used to expose the design rule cleaner class to the python PCells of KLayout. The algorithm is pasted below. The algorithm uses a [Scanline Rendering Algorithm](#)⁹ to first convert the polygons from KLayout to manhattanized edges and then add them into an array representation of the polygon edges.

Source Code: [DrcSl Source](#)

5.1.3 Multiprocessing

With version 0.1.0 multiprocessing was introduced. Multiprocessing allows to use all threads of the machine to process the DRC cleaning on all threads of the CPU in parallel. This can give a considerable speed boost if multiple layers are involved and the hardware supports it.

kppc.drc.cleaner.master module

Wrapper Class for CleanerMaster.cpp

This Class creates a managed shared memory space. Polygon data for cleaning are streamed into this memory space. A slave process (cleanermain, which is a little loop for CleanerSlave.cpp).

Python Class

class PyCleanerMaster

add_edge(self, x1 : int, x2 : int, y1 : int, y2 : int)

Add an edge to the cleaner.

Parameters

- **x1** – first x coordinate
- **x2** – second x coordinate
- **y1** – first y coordinate
- **y2** – second y coordinate

done(self)

Indicates whether there is data still in the buffer from the last read or not.

Returns false if the buffer is empty and the data has been read by the slave.

Return type `bool`¹⁰

get_layer(self)

Read the next processed layer in the memory space and returns it in per line style (x coordinates per line (y coordinate)). This is considerably slower than returning the polygons.

polygons(self)

Reads the next processed layer in the memory and assembles the line style to polygons.

set_box(self, layer : int, datatype : int, violation_width : int, violation_space : int, x1 : int, x2 : int, y1 : int, y2 : int)

Allocate enough space in the shared memory to stream the cell and its polygons in.

Parameters

- **layer** – layer number

⁹ https://en.wikipedia.org/wiki/Scanline_rendering

¹⁰ <https://docs.python.org/3/library/functions.html#bool>

- **datatype** – datatype number
- **viospace** (minimum space violation in database units) – minimum space violation in database units
- **viowidth** (minimum width violation in database units) – minimum width violation in database units
- **x1** – left bound of box
- **x2** – right bound of box
- **y1** – bottom bound of box
- **y2** – top bound of box

C++ Class

class CleanerMaster

union [anonymous]

CleanerMaster(int *nlayers*)

Creates the shared memory space and resizes the vectors for *nlayers*

void **set_box**(int *layer*, int *datatype*, int *violation_width*, int *violation_space*, int *x1*, int *x2*, int *y1*, int *y2*)

Allocate enough space in the shared memory to stream the cell and its polygons in.

void **add_edge**(int *x1*, int *x2*, int *y1*, int *y2*)

Add an edge to the cleaner.

bool **done**()

Indicates whether there is data still in the buffer from the last read or not.

std::vector<std::vector<int>> CleanerMaster::**get_layer**()

Read the next processed layer in the memory space and returns it in per line style (x coordinates per line (y coordinate)).

std::vector<std::vector<std::pair<int, int>>> CleanerMaster::**get_polygons**()

Reads the next processed layer in the memory and assembles the line style to polygons.

C++ Source Code: [CleanerMaster Source](#)

CleanerMain

C++ documentation of the cleanermain. This program is a simple program with a loop that processes any layers added to the shared memory. If the process receives *SIGUSER1*, it joins the threads and terminates afterwards.

Source: [CleanerMain Source](#)

CleanerSlave

C++ documentation of the Class CleanerSlave

class CleanerSlave

void **CleanerSlave()**

Constructor of the Class The constructor opens the shared memory and initializes the allocators for the shared memory. Initializes a boost thread_pool with as many threads as the CPU supports (one per core).

void **clean()**

Checks if the shared memory has a cell layer added. If there is a layer to process, move the data to shared memory and schedule it for processing by the thread_pool.

void **join_threads()**

Wait for the thread_pool to finish all jobs and return

Source Code: [CleanerSlave Source](#)

5.2 photonics Module

This package is a library extension for KLayout to provide functionalities for photonic structures.

Warning: KLayout does not check if a loaded module has changed during runtime and thus does not reread/recompile it. This means you either must manually reload the library if you want to do it during runtime. Generally, it is easier and safer to close and reopen KLayout.

If this extension is modified (or any file in a /python directory), don't forget to either reload the module or reopen KLayout.

Note: To reload a module during runtime use the following commands in the KLayout python console (not guaranteed to work in all cases):

```
>>> from importlib import reload
>>> import <module>
>>> reload(<module>)
```

5.2.1 Module contents

5.2.2 Submodules

photonics.dataprep module

photonics.layermaps module

EXAMPLE: CREATE SAMPLE LIBRARY

In this chapter we will create an example library consisting of an MMI built with a box and linear tapers. This example builds on FreePDK45 technology. Therefore, please get the *KLayoutPhotonicPCells/FreePDK45_tech* package from the KLayout package manager. The finished file of this example can be found here [MMI_Example](#). This file can be copied into the KLayout pymacros folder (`~/.klayout/pymacros/`) and executed.

6.1 Code Explanation

All photonic libraries are derived from PhotDevice.

As an example, we will use a modified FreePDK45_Cell. We will create a 2x2 MMI. To create a new PCell Library open the MacroDevelopment of Klayout in the menu `Macros->MacroDevelopment`.

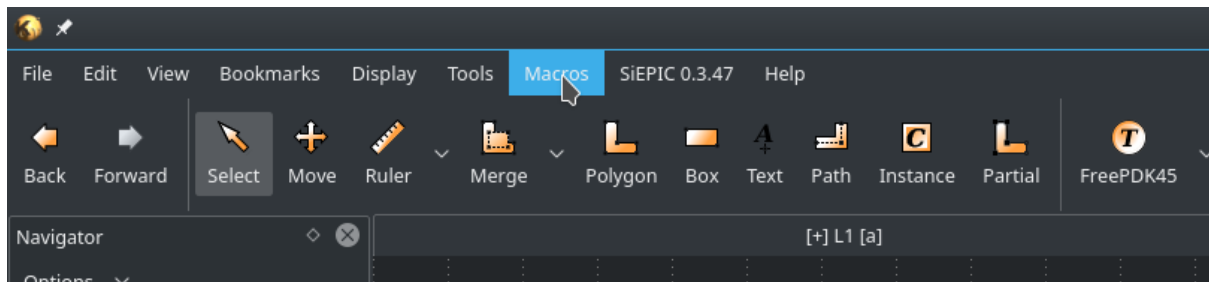


Fig. 6.1: Open the IDE through *Macros*→*MacroDevelopment*

This will open the KLayout Ruby/Python/DRC IDE. In the left sidebar choose Python as a language. In the menu choose new (second to the left, the plus sign) to create a new script/library.

From the opening context choose *PCell template (Python)*. This will create a new `.lym` file for a PCell-Library. The generated sample code is irrelevant for us as we will not use KLayout syntax, but the extension. The reason for choosing the PCell Sample instead of an empty template is, that it will be flagged as a PCell library in the background.

As a next step delete all example code. The new cell will be created from scratch. Reason for using the sample PCell is that KLayout uses some flags to define it as a PCell library.

First let's import modules we will need.

```

1 import pya
2 import math
3 from kppc.photonics import PhotDevice, PortCreation
4 import kppc.photonics.layermaps as lm
5 import numpy as np
6 import os

```

After the imports we will create a helper class. The class `kppc.photonics.PhotDevice` is technology-independent and thus needs to be supplied with information about layers, i.e. how to map layers during

dataprep and finally about the constraints for the DR-Cleaning. So let's define a helper class that all of our FreePDK45-PCells will use.

```

9 class FreePDK45Example(PhotDevice):
10     """Class that provides technology specific data. Currently the backend needs 3 things to be
11     ↳ supplied by the technology of the PCells.
12     As these are independent of specific PCells and parameters this should not give any difficulty
13     ↳ with the requirement of Klayout to have the Classes stateless.
14
15     The layermap was created from a forum suggestion
16     ('Post <https://community.cadence.com/cadence\_technology\_forums/f/custom-ic-design/37021/layer-map-file-for-gds-transfer-to-virtuoso>' ) and then some layers were added by hand.
17
18     layermap:      A dictionary containing layers with available purposes, which provide a
19     ↳ layer/purpose. This is loaded from a .layermap file.
20     Example of this FreePDK45:
21     ↳ {'active': {'blockage': (1, 1), 'drawing': (1, 0)}, 'pwell': {'blockage
22     ↳ ': (2, 1), 'drawing': (2, 0)},...}
23
24     :ivar dataprep_config: Filepath to a text file containing rules for dataprep. This file
25     ↳ contains rules for the dataprep.
26     Copied from the example dataprep.txt:
27     File Format:
28     File defining operations for dataprep
29     Format:
30     <operation> <source layers> <destination layers> <sizing amount in microns>
31     Operations supported: add,sub
32     ↳ * add: Create a region from all shapes of the source layers and combine
33     ↳ this region with each destion layer region separately
34     ↳ * sub: Same as add but don't build combination but cross-section instead
35     Sizing amount uses the klayout sizing operation to size the regions of the
36     ↳ source layers
37     During dataprep the regions are merged, meaning overlapping polygons will
38     ↳ become one Polygon
39     source/destination layers are separated by commas if there are multiple
40     ↳ Each argument is separated by white spaces. How many should not matter as
41     ↳ they will be parsed by a python str.split() operation which should be able to handle any white
42     ↳ space amount.
43     If the first word of a line is not a supported operation the line will be
44     ↳ ignored
45     The lines will be executed in order meaning and add sub operations on
46     ↳ layers will be different than a first sub and then add
47     :Examples:
48     ↳ active.blockage,poly.blockage,metal1.blockage,metal2.blockage,metal3.
49     ↳ blockage,metal4.blockage,metal5.blockage,nwell.drawing,nimplant.drawing 2.0
50
51     :ivar clean_rules:      list containing the layer/purpose numbers and the minWidth/
52     ↳ minSpacing rules for the layer/purpose pair in microns
53     :Examples:
54     ↳ [[(1, 0), 0.097, 0.077],[(2, 0), 0.23, 0.189],[(3, 0), 0.169, 0.196],
55     ↳ [(5, 0), 0.044, 0.052], ...]
56     """
57
58     # Define the metals & via names. They will be used in some PCells (Electrodes and ViaStack)
59     metal_names = ['metal' + str(i) for i in range(1, 11)]
60     via_names = ['via' + str(i) for i in range(1, 10)]
61
62     def __init__(self):
63         PhotDevice.__init__(self)
64
65         techpath = pya.Technology.technology_by_name('FreePDK45').base_path()

```

(continues on next page)

(continued from previous page)

```

52     filename = techpath + '/FreePDK45.tf'
53
54     # Check if techfile is correctly imported and located
55
56     isfile = os.path.isfile(filename)
57     if not isfile:
58         import sys
59         msg = pya.QMessageBox(pya.Application.instance().main_window())
60         msg.text = 'Please import the techfile of the technology to {} before using the module_
↪and reopen KLayout'.format(filename)
61         msg.windowTitle = 'ImportError'
62         msg.exec_()
63
64     tech = con.load_from_tech(filename)
65
66     # Get the layermap file and load it.
67     # CAREFUL: Will be used for dataprep and others
68     self.layermap = lm.load(techpath + '/FreePDK45.layermap')
69
70     # This variable will be imported by the dataprep algorithm
71     # CAREFUL: Will be imported for dataprep
72     self.dataprep_config = techpath + '/dataprep.txt'
73
74     # Rules for the cleaner in the form [[(layer1,purpose1),violation_width1,violation_space1],
↪[(layer2,purpose2),violation_width2,violation_space2],...]
75     ### CAREFUL: This variable will be imported for the cleaning.
76     self.clean_rules = [(1, 0), 0.111, 0.085], [(2, 0), 0.23, 0.188], [(3, 0), 0.14, 0.199],
↪[(5, 0), 0.044, 0.049],
77     [(4, 0), 0.046, 0.052], [(9, 0), 0.044, 0.062], [(11, 0), 0.076, 0.077],
↪[(13, 0), 0.073, 0.089],
78     [(15, 0), 0.067, 0.063], [(17, 0), 0.143, 0.137], [(19, 0), 0.158, 0.
↪14], [(21, 0), 0.145, 0.123],
79     [(23, 0), 0.514, 0.535], [(25, 0), 0.369, 0.311], [(27, 0), 0.908, 0.
↪843], [(29, 0), 0.347, 0.771],
80     [(1, 1), 1.247, 1.254], [(2, 1), 0.976, 0.905], [(3, 1), 1.165, 1.304],
↪[(5, 1), 1.073, 0.958],
81     [(4, 1), 1.058, 0.885], [(9, 1), 0.892, 0.825], [(11, 1), 1.003, 0.682],
↪[(13, 1), 0.983, 0.73],
82     [(15, 1), 1.086, 0.993], [(17, 1), 1.12, 0.812], [(19, 1), 0.941, 0.
↪765], [(21, 1), 0.942, 0.889],
83     [(23, 1), 1.044, 0.933], [(25, 1), 1.096, 1.039], [(27, 1), 0.798, 0.
↪937], [(29, 1), 1.001, 1.286]]

```

This is our basic class. Now let's create two basic PCells. First a linear taper and second a box. A box combined with 4 tapers will build a 2x2 MMI. To connect them we will use ports. The liner taper will have two ports, one on each side. The box will have four ports and each port of the box is the same size as the big part of the taper.

```

86 class ExMMIBody(FreePDK45Example):
87     """MMI Body. Since this should be a 2x2 MMI it will have 4 ports
88     """
89
90     def __init__(self):
91         FreePDK45Example.__init__(self)
92         self.add_layer('lay', "active.drawing")
93         # Important: If it should be a floating point parameter, use x.0 instead of x for default_
↪values that fall on integers, or it will be interpreted as integer
94         params = dict(length=15.,
95                       width=5.,
96                       port_offset=1.5,
97                       port_width =1.0

```

(continues on next page)

(continued from previous page)

```

98         )
99         # Register the parameters
100         self.add_params(params)
101
102     def create_param_inst(self):
103         # Create Ports here
104         ports = [PortCreation(-self.length/2, self.port_offset, 180, self.port_width),
105                 PortCreation(-self.length/2, -self.port_offset, 180, self.port_width),
106                 PortCreation(self.length/2, -self.port_offset, 0, self.port_width),
107                 PortCreation(self.length/2, self.port_offset, 0, self.port_width)]
108         return ports
109
110     def shapes(self):
111         #Create the Rectangle
112         self.create_polygon([[-self.length/2,-self.width/2],[self.length/2,-self.width/2],[self.
113 ↪length/2,self.width/2],[-self.length/2,self.width/2]],self.lay)
114
115 class ExLinTaper(FreePDK45Example):
116
117     def __init__(self):
118         FreePDK45Example.__init__(self)
119         self.add_layer('lay',"active.drawing")
120         params = dict(width_0 = .5,
121                       width_1 = 1.0,
122                       length = 2.0,
123                       )
124         self.add_params(params)
125
126     def create_param_inst(self):
127         # Create left and right port
128         port_0 = PortCreation(-self.length/2,0,180,self.width_0)
129         port_1 = PortCreation(self.length/2,0,0,self.width_1)
130         return port_0,port_1
131
132     def shapes(self):
133         # Create taper polygon
134         self.create_polygon([[-self.length/2,-self.width_0/2],
135                             [-self.length/2,self.width_0/2],
136                             [self.length/2,self.width_1/2],
137                             [self.length/2,-self.width_1/2],],
138                             self.lay)

```

Note: If we only declare one PortCreation in self.create_param_inst(self), we have to return it as: return [port]

Now let's declare the MMI. In it we will create 4 instances of tapers and one box and then connect the tapers to the box.

```

139 class Ex2x2MMI(FreePDK45Example):
140     """The MMI-cell class.
141     This class instantiates a body with 4 tapers and attaches the tapers to the the body.
142     """
143
144     def __init__(self):
145         FreePDK45Example.__init__(self)
146         self.add_layer('lay','active.drawing')
147         params = dict(wg_width=.5,
148                       length=15.0,
149                       taper_width=1.0,

```

(continues on next page)

(continued from previous page)

```

150         taper_length=2.0,
151         width=4.0,
152         taper_offset=1.0,
153     )
154     self.add_params(params)
155
156     def create_param_inst(self):
157         # Library we load the sub-cells from
158         lib = "FreePDK45_Photonic_FirstExample"
159         bodyname = "MMIBody"
160         tapername = "LinearTaper"
161
162         # Parameters used for the 4-port body
163         body_params = dict(lib = lib,
164                             cellname = bodyname,
165                             width=self.width,
166                             length=self.length,
167                             port_offset=self.taper_offset,
168                             port_width =self.taper_width,
169                             )
170         # Parameters for tapers
171         taper_params = dict(lib = lib,
172                             cellname = tapername,
173                             width_0=self.wg_width,
174                             width_1=self.taper_width,
175                             length=self.taper_length,
176                             )
177         # Create constructors for tapers and body
178         tapers = self.add_pcell_variant(taper_params,number=4)
179         body = self.add_pcell_variant(body_params)
180
181         # Connect the ports
182         for i in range(4):
183             self.connect_port_to_port(body.port(i),tapers[i].port(1))
184
185         # Return constructors
186         return tapers,body

```

Finally create the Library so that we can call it in KLayout:

```

188 class FreePDK45_ExampleLib(pya.Library):
189     def __init__(self):
190         # Set the description
191         self.description = "FirstExample"
192         self.technology = "FreePDK45"
193         # Create the PCell declarations
194         self.layout().register_pcell("2x2MMI",Ex2x2MMI())
195         self.layout().register_pcell("MMIBody",ExMMIBody())
196         self.layout().register_pcell("LinearTaper",ExLinTaper())
197
198         self.register("FreePDK45_Photonic_FirstExample")

```

And finally make KLayout compile the PCell-Library and add it to the PCell-Libraries:

```

200 # Instantiate and register the library
201 FreePDK45_ExampleLib()

```

Click Run script from the current tab (Green Arrow with a vertical line at the end).

Now you can create Instances of this parametric cell in the main window of Klayout. Click on Instance and choose the FreePDK Sample Cells [Technology FreePDK45] library from the drop-down menu. On the left of the library drop down you can choose one of the three cells. And in the tab you can adjust

parameters.

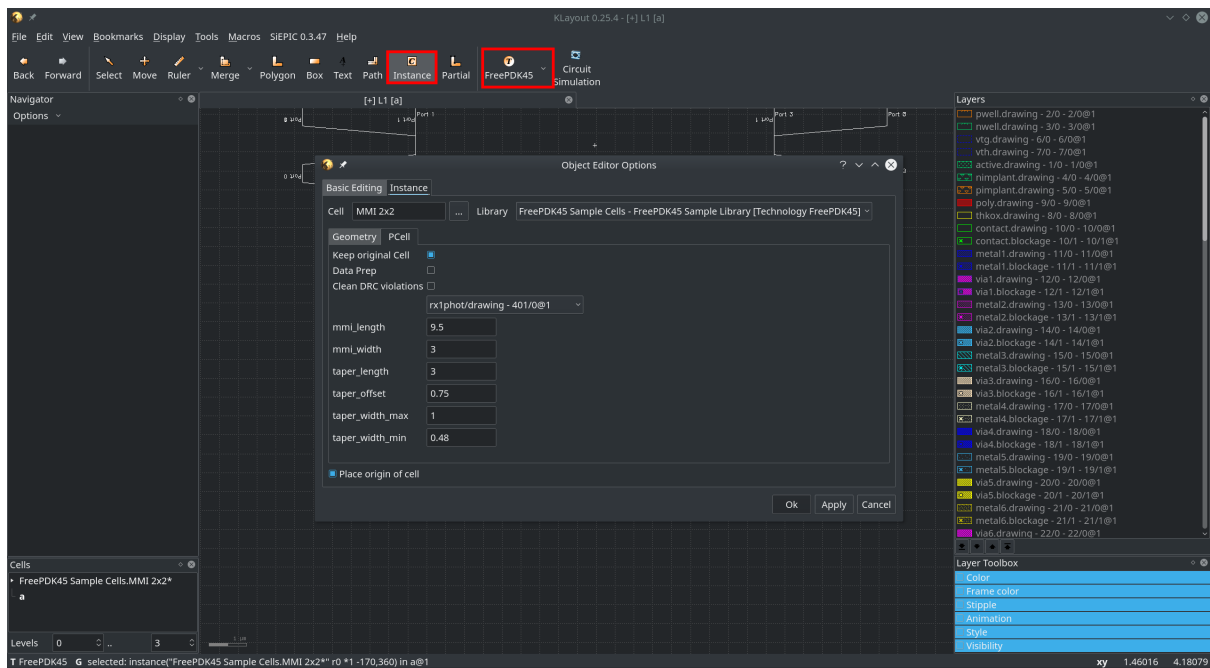


Fig. 6.4: In the main window click on Instance to create instances of the new Cell

If you click **Ok** or **Apply** you can place the new Cell with adjusted parameters. The first boolean determines whether the cell should contain only dataprep & design rule cleaned shapes or all shapes. The second tells the cell to perform dataprep and the last to make it DR-clean. The rest of the parameters are PCell specific and should be the ones defined in the `__init__(self)` function of the cell definition.

TIPS & TRICKS

7.1 Variable Names in KLayout Python

When using global variables in pymacros (scripts like cell libraries) be careful. Namespace is shared between macros. This means when for example defining the names of metal layers in two cells, one can overwrite the other one. Therefore the use of global variables is not advised and the use of a wrapper class is recommended instead. It can be defined in the same wrapper class used for defining layernames and cleaning information, for example.

C++ SOURCE CODE

8.1 DrcSI Source

```
// This file is part of KLayoutPhotonicPCells, an extension for Photonic Layouts in KLayout.
// Copyright (c) 2018, Sebastian Goeldi
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Affero General Public License as
// published by the Free Software Foundation, either version 3 of the
// License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU Affero General Public License for more details.
//
// You should have received a copy of the GNU Affero General Public License
// along with this program. If not, see <https://www.gnu.org/licenses/>.

#include "DrcSl.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <stdexcept>
#include <cmath>

#include <boost/interprocess/managed_shared_memory.hpp>
#include <boost/interprocess/containers/vector.hpp>
#include <boost/interprocess/allocators/allocator.hpp>

namespace drclean
{
// Function to compare two edgecoord structs. This is necessary for std::sort. If they are on_
↳ the same coordinate sort for type in descending order
bool compare_edgecoord(edgecoord e1, edgecoord e2)
{
    if (e1.pos==e2.pos)
        return (e2.type<e1.type);
    else
        return (e1.pos<e2.pos);
}

// Constructor. Initialize the pointers as nullptrs
```

(continues on next page)

(continued from previous page)

```

DrcSl::DrcSl()
{
    this->lver = nullptr;
    this->lhor = nullptr;
}

// Destructur: Delete the allocated vectors.
DrcSl::~DrcSl()
{
    if(this->lhor != nullptr)
        delete[] this->lhor;
    if(this->lver != nullptr)
        delete[] this->lver;
}

// Add a complete data-set. Currently not used and not exposed in the Python interface.
int DrcSl::set_data(std::vector<edgecoord> *horlist)
{
    this->l = horlist;
    return 0;
}

// Initialize the dimensions of the vector arrays and set pointers accordingly and dimension_
// units.
void DrcSl::initialize_list(int hor1, int hor2, int ver1, int ver2, int violation_space, int_
// violation_width)
{
    if(this->lhor)
    {
        delete[] this->lhor;
        this->lhor = nullptr;
    }
    if(this->lver)
    {
        delete[] this->lver;
        this->lver = nullptr;
    }
    this->lhor = new std::vector<edgecoord>[ver2-ver1+5];
    this->lver = new std::vector<edgecoord>[hor2-hor1+5];
    this->l = this->lhor;
    this->sver = hor2-hor1+5;
    this->shor = ver2-ver1+5;
    this->hor1 = hor1-2;
    this->hor2 = hor2+2;
    this->ver1 = ver1-2;
    this->ver2 = ver2+2;
    this->violation_space = violation_space;
    this->violation_width = violation_width;
    this->orientation = hor;
}

// Print the complete data set or from index beg -> end if they are set.
// -1, -1 will result in printing the whole vector.
void DrcSl::printvector(int beg, int last)
{
    if (last == -1 && beg == -1)
    {
        last = this->orientation ? this->ver2 : this->hor2;
        beg = this->orientation ? this->ver1 : this->hor1;
    }
    std::vector<edgecoord>::iterator it;

```

(continues on next page)

(continued from previous page)

```

std::cout << "size, y: " << this->sver << std::endl << "size, x: " << this->shor << std::endl;
int offset = this->orientation ? -this->hor1 : -this->ver1;
int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;
std::cout << "beg/end " << beg+offset << '/' << last+offset-1 << std::endl;

for (int i = 0; i < this->s(); i++)
{
    std::cout << "row: " << i+offset << ": [";
    for(it = this->l[i].begin(); it != this->l[i].end(); it++)
    {
        std::cout << "(" << it->pos -offset_d2<< ", " << it->type << ")";
    }
    std::cout << "]" << std::endl;
}
}

// Get the current array size of the vectors
int DrcSl::s()
{
    return this->orientation ? this->sver : this->shor;
}

// Sort all data with compare_edge_coord and remove overlapping edges, i.e. merge overlapping
// polygons in the data
void DrcSl::sortlist()
{
    // std::cout << this->s() << std::endl;
    for (int i = 0; i < this->s(); i++)
    {
        if (!this->l[i].empty())
        {
            std::sort(this->l[i].begin(), this->l[i].end(), compare_edgecoord);
            std::vector<edgecoord>::iterator it;
            it = this->l[i].begin();
            int c = 0;
            for(; it != this->l[i].end(); it++)
            {
                if (it->type == 0)
                {
                    {
                        c++;
                        if (c>1 || c<0)
                        {
                            it->rem = true;
                        }
                    }
                }
                else
                {
                    if (c>1 || c<0)
                    {
                        {
                            it->rem = true;
                        }
                        c--;
                    }
                }
            }
            this->l[i].erase(std::remove_if(this->l[i].begin(), this->l[i].end(), [](auto o)
            {
                return o.rem;
            }), this->l[i].end());
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
}

// Get data from a row (or column).
// If used after the standard sorting or cleaning function, i.e. sortlist() and cleaning(),
// the vectors should always be arranged row-oriented, meaning the same format as when added to
// the cleaner.
std::vector<int> DrcSl::get_vect(int ind)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;

    std::vector<int> res = std::vector<int>(this->l[ind+offset].size());
    std::vector<edgecoord>::iterator it;
    int i;
    for(it = this->l[ind+offset].begin(), i=0; it != this->l[ind+offset].end(); it++, i++)
    {
        if (it->type)
            res[i]=(it->pos-1-offset_d2);
        else
            res[i]=(it->pos+1-offset_d2);
    }

    return res;
}

// Function to print the types in a vector. Probably only useful for debugging purposes
std::vector<int> DrcSl::get_types(int ind)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;

    offset ++;

    std::vector<int> res = std::vector<int>(this->l[ind+offset].size());
    std::vector<edgecoord>::iterator it;
    int i;
    for(it = this->l[ind+offset].begin(), i=0; it != this->l[ind+offset].end(); it++, i++)
    {
        res[i] = it->type;
    }

    return res;
}

// Add data to the data structure. We manhattanize the edge from the input and mark left facing
// edges with -1 and
// right facing edges with +1. The get_vect() function reverses this effect.
// This should have no influence on any possible data except that it merges touching polygons.
void DrcSl::add_data(int px1, int px2, int py1, int py2)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;

    if (py2 > py1)
    {
        edgecoord p = edgecoord(px1+offset_d2-1,0);

        double dx = (double)(px2-px1)/(py2-py1);
        double x = p.pos;
        if (p.pos < 0 || p.pos > (this->orientation ? this->shor : this->sver))

```

(continues on next page)

(continued from previous page)

```

    {
        std::cout << "Error ROW (y) index out of bound " << p.pos << '/' << (this->orientation ?
↪ this->shor: this->sver) << std::endl;
        throw 1;
    }
    if (offset+py1 < 0 || py2+offset > (this->orientation ? this->sver : this->shor))
    {
        std::cout << "Error COLUMN (x) index out of bound" << py2+offset << "/" << this->s() <<
↪ std::endl;
        throw 2;
    }

    if (dx > 0)
    {
        for(int i = offset+py1; i < py2+offset; i++)
        {
            this->l[i].push_back(p);
            x+=dx;
            p.pos = int(x);
        }
    }
    else
    {
        for(int i = offset+py1; i < py2+offset-1; i++)
        {
            x+=dx;
            p.pos = int(x);
            this->l[i].push_back(p);
        }
        p.pos = px2+offset_d2-1;
        this->l[py2+offset-1].push_back(p);
    }

}
else if (py1 > py2)
{
    edgecoord p = edgecoord(px2+offset_d2+1,1);

    double dx = (double)(px1-px2)/(py1-py2);
    double x = p.pos;
    if (p.pos < 0 || p.pos > (this->orientation ? this->shor : this->sver))
    {
        std::cout << "Error ROW (y) index out of bound " << p.pos << '/' << (this->orientation ?
↪ this->shor: this->sver) << std::endl;
        throw 1;
    }
    if (offset+py1 < 0 || py2+offset > this->s())
    {
        std::cout << "Error COLUMN (x) index out of bound" << std::endl;
        throw 2;
    }

    if (dx < 0)
    {
        for(int i = offset+py2; i < py1+offset; i++)
        {
            this->l[i].push_back(p);
            x+=dx;
            p.pos = std::ceil(x);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    else
    {
        for(int i = offset+py2; i < py1+offset-1; i++)
        {
            x+=dx;
            p.pos = std::ceil(x);
            this->l[i].push_back(p);
        }
        p.pos = px1+offset_d2+1;
        this->l[py1+offset-1].push_back(p);
    }
}

// Clean data for space violations in the current orientation (row-oriented for violations,
// within the row and accordingly if column-oriented).
int DrcSl::clean_space()
{
    //Cleans space violations.
    //Returns number of space violations that were cleaned.
    std::vector<edgecoord> *il = this->l;

    //Counters to keep track of how many checks were done and how many space violations have been
    //cleaned.
    int spacevios = 0;
    int counts = 0;

    std::vector<edgecoord>::iterator it;

    for (int i = 0; i<this->s(); i++)
    {
        if (!il->empty())
        {
            bool er = false;
            it = il->begin();
            if (it == il->end())
                continue;
            it++;
            while(it+1 != il->end())
            {
                counts++;
                if ((it+1)->pos - it->pos < violation_space -1)
                {
                    er = true;
                    spacevios++;
                    it->rem = true;
                    (it+1)->rem = true;
                }
                it+=2;
            }
            if (er)
                il->erase(std::remove_if(il->begin(),il->end(),[](auto o)
                {
                    return o.rem;
                }),il->end());
            il++;
        }
    }

    // If progress output is desired uncomment the following lines
    // std::cout << "number of checks: " << counts << std::endl;
    // std::cout << "violations, space: " << spacevios << std::endl;

```

(continues on next page)

(continued from previous page)

```

    return spacevios;
}

// Clean data for width violation
int DrcSl::clean_width()
{
    std::vector<edgecoord> *il = this->l;

    int widthvios = 0;
    int counts = 0;

    std::vector<edgecoord>::iterator it;

    for (int i = 0; i<this->s(); i++)
    {
        if (!il->empty())
        {
            bool er=false;
            it = il->begin();
            while(it != il->end())
            {
                counts++;
                if ((it+1)->pos - it->pos < violation_width +1)
                {
                    er = true;
                    it->rem = true;
                    (it+1)->rem = true;
                    widthvios++;
                }
                it+=2;
            }
            if (er)
                il->erase(std::remove_if(il->begin(),il->end(),[](auto o)
                {
                    return o.rem;
                })),il->end());
        }
        il++;
    }

    // If progress output is desired uncomment the following lines
    // std::cout << "number of checks: " << counts << std::endl;
    // std::cout << "violations, width: " << widthvios << std::endl;
    return widthvios;
}

// Calculate difference between two rows or two columns. This is necessary when switching from
↪row-oriented to
// column-oriented data and vice-versa.
//
// In theory this can also be used to check for minimum edge-lengths. But for us all of these
↪requirements have been
// waived, so we don't have to check for those.
std::vector<int> DrcSl::listdif(std::vector<edgecoord> &l1, std::vector<edgecoord> &l2)
{
    /*
    ** Calculates differences between rows (or columns, depending on orientation) between two
    ↪vectors (rows/columns)

```

(continues on next page)

(continued from previous page)

```

    ** The difference between the two vectors indicate that there is a polygon border for the
↪ other orientation of the scanlines
    ** This border corresponds to edges and thus has to appear in the opposite orientation
    */

    /*
    ** Example:
    **
    ** l1 is the row/column that we compare to. Any coordinates that appear in l1, but not in l2,
↪ will be returned as ranges.
    ** example:
    ** l1 = ([1,5],[7,10],[18,20])
    ** l2 = ([4,11],[15,16])
    ** out = ([1,3],[18,20])
    */

    std::vector<int> out;
    std::vector<edgecoord>::iterator it1 = l1.begin();
    std::vector<edgecoord>::iterator it2 = l2.begin();
    int l21;
    int l22;
    for (it1 = l1.begin(); it1 != l1.end(); it1++)
    {
        int b = it1->pos;
        it1++;
        int e = it1->pos;
        int ee = e;
        bool add = true;
        while(it2 != l2.end())
        {
            l21 = it2->pos;
            l22 = (it2+1)->pos;
            if(l22 < b)
            {
                it2+=2;
            }
            else if (l22 >= e)
            {
                if (e < b || l21 <= b)
                    add = false;
                if (e > l21 -1)
                    e = l21 -1;
                break;
            }
            else if (l22 < e && l21 > b)
            {
                out.push_back(b);
                out.push_back(l21 -1);
                b = l22 + 1;
                e = ee;
                it2 += 2;
            }
            else if (l22 >= b && b >= l21)
                b = l22 + 1;
            else if (l21 <= e && e <= l22)
            {
                e = l21 + 1;
                break;
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

        if (add)
        {
            out.push_back(b);
            out.push_back(e);
        }
    }
    return out;
}

// Switch dimensions. When calculating listdiffs between two rows, we can calculate the edges in
// row direction when
// row-oriented or in column direction when column-oriented. These edges then give us column-
// orientation data and vice-versa.
void DrcSl::switch_dimensions()
{
    /*
    ** Switch row to column orientation of the scanlines.
    ** Example:
    **
    ** 5:      []
    ** 6:      []
    ** 7:      [(4,0),(10,1)]
    ** 8:      [(3,0),(7,1),(8,0),(11,1)]
    ** 9:      [(3,0),(8,1),(8,0),(11,1)]
    ** 10:     [(4,0),(8,0),(8,1),(12,1)]
    ** 11:     [(4,0),(7,1)]
    ** 12:     []
    ** 13:     []
    **
    ** Will be converted to:
    **
    ** 3:      []
    ** 4:      [(7,0),(10,1)]
    ** 5:      [(6,0),(12,1)]
    ** 6:      [(6,0),(12,1)]
    ** 7:      [(6,0),(8,1),(8,0),(11,1)]
    ** 8:      [(6,0),(8,1)]
    ** 9:      [(6,0),(11,1)]
    ** 10:     [(7,0),(11,1)]
    ** 11:     [(9,0),(11,1)]
    ** 12:     []
    **
    */

    // If progress output is desired uncomment the following lines
    // std::cout << "Switching dimensions" << std::endl;
    if(this->lhor == nullptr)
        this->lhor = new std::vector<edgecoord>[this->ver2-this->ver1];
    if(this->lver == nullptr)
        this->lver = new std::vector<edgecoord>[this->hor2-this->hor1];

    std::vector<edgecoord> *l_new;

    if (this->orientation)
    {
        for(int i = 0; i<this->shor; i++)
        {
            this->lhor[i].clear();
        }
        l_new = this->lhor;
    }

```

(continues on next page)

(continued from previous page)

```

}
else
{
    for(int i = 0; i<this->sver; i++)
    {
        this->lver[i].clear();
    }
    l_new = this->lver;
}
std::vector<edgecoord> row_last;
std::vector<edgecoord> row;
std::vector<edgecoord> row_next;
std::vector<edgecoord> *it = this->l;
std::vector<int>::iterator dit;
std::vector<int> dif1;
std::vector<int> dif2;
std::vector<edgecoord>::iterator rit;
row_last = *it;
for(rit = row_last.begin(); rit != row_last.end(); rit++)
{
    rit->pos++;
    rit++;
    rit->pos--;
}
it ++;
row = *it;
for(rit = row.begin(); rit != row.end(); rit++)
{
    rit->pos++;
    rit++;
    rit->pos--;
}
it++;
int row_number = 2;
for (int n = 2; n < this->s(); n++)
{
    row_next = *it;
    for(rit = row_next.begin(); rit != row_next.end(); rit++)
    {
        rit->pos++;
        rit++;
        rit->pos--;
    }

    dif1 = listdif(row_last,row);
    dif2 = listdif(row_next,row);

    int b;
    int e;
    dit = dif1.begin();
    while(dit != dif1.end())
    {
        b = *dit;
        dit++;
        e = *dit;
        dit++;
        for (; b!=e+1; b++)
        {
            edgecoord p = edgecoord(row_number-1,1);
            l_new[b].push_back(p);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
    dit = dif2.begin();
    while(dit != dif2.end())
    {
        b = *dit;
        dit++;
        e = *dit;
        dit++;
        for (; b!=e+1; b++)
        {
            edgecoord p = edgecoord(row_number-1,0);
            l_new[b].push_back(p);
        }
    }
    row_last = row;
    row = row_next;
    row_number++;
    it++;
}
this->l = l_new;
this->orientation = this->orientation ? hor : ver;
}

// Function that first cleans space violations then width violations and then space violations.
// ↪again.
// This does not necessarily clean all violations. For example if a fixing of a width violation.
// ↪creates a space violation
// and vice-versa, the algorithm will not fix the violation. For performance reasons
// it is still the user's task to perform DRC and ensure the design is clean. For standard.
// ↪photonic structures it is
// unlikely that such a case occurs.
void DrcSI::clean(int maxtries)
{
    for(int i = 0; i < maxtries; i++)
    {
        if(clean_space())
        {
            switch_dimensions();
        }
        else
        {
            if(clean_space())
            {
                switch_dimensions();
                continue;
            }
            else
            {
                // If progress output is desired uncomment the following lines
                // std::cout<< "Finished after " << i+1 << " tries" << std::endl;
                break;
            }
        }
    }
    for(int i = 0; i < maxtries; i++)
    {
        if(clean_width())
        {
            // If progress output is desired uncomment the following lines
            // std::cout<< "Try: " << i << "/" << maxtries << std::endl;

```

(continues on next page)

(continued from previous page)

```

        switch_dimensions();
    }
    else
    {
        if(clean_width())
        {
            switch_dimensions();
            continue;
        }
        else
        {
//            If progress output is desired uncomment the following lines
//            std::cout<< "Finished after " << i+1 << " tries" << std::endl;
            break;
        }
    }
}
for(int i = 0; i < maxtries; i++)
{
    if(clean_space())
    {
//        If progress output is desired uncomment the following lines
//        std::cout<< "Try: " << i << "/" << maxtries << std::endl;
        switch_dimensions();
    }
    else
    {
        if(clean_space())
        {
            switch_dimensions();
        }
        else
        {
            if (this->orientation)
            {
//                If progress output is desired uncomment the following lines
//                std::cout<< "Finished after " << i+1 << " tries" << std::endl;
                switch_dimensions();
                break;
            }
        }
    }
}
//    If progress output is desired uncomment the following lines
//    std::cout<< "Done cleaning" << std::endl;
}

std::vector<std::vector<int>> DrcSl::get_lines()
{
    std::vector<std::vector<int>> lines (this->s());

    int offset_d2 = this->orientation ? -this->ver1 : -this-> hor1;

    for(int i = 0; i< this->s(); i++)
    {
        for(auto iter: this->l[i])
        {
            lines[i].push_back(iter.type ? iter.pos-1-offset_d2 : iter.pos+1-offset_d2);
        }
    }
    return lines;
}

```

(continues on next page)

(continued from previous page)

```

}

std::vector<std::vector<pi>> DrcSl::get_polygons()
{
    splits.clear();
    polygons.clear();
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d1 = (this->orientation ? -this->ver1 : -this->hor1) - 1;
    int offset_d2 = (this->orientation ? -this->ver1 : -this->hor1) + 1;
    int begin = 0;

    for(int i = 1; i < this->s(); i++)
    {
        int y = i - offset;
        spv::iterator spit = splits.begin() + begin;
        spv::iterator spit_last = splits.begin() + begin;

        ev::iterator append_first = this->l[i].begin();
        ev::iterator append_last = this->l[i].begin();

        bool advance = true;

        for(ev::iterator ei = this->l[i].begin(); ei != this->l[i].end(); ei+=2)
        {
            int x1 = ei->pos - offset_d1;
            int x2 = (ei+1)->pos - offset_d2;

            if(advance)
            {
                spit = std::find_if(splits.begin(), splits.end(), [x1,x2,y,offset_d2,offset_d1]_
↳(SplitPolygon & sp)
                {
                    return ((sp.line == y) && !((sp.rx < x1) || (sp.lx > x2)));
                });
                advance = false;
            }

            if(spit != splits.end())
            {
                int ex1 = spit->lx;
                int ex2 = spit->rx;

                if(((x1 > ex2) || (x2 < ex1)) && spit->line == y)
                {
                    int l = append_last - append_first;
                    if(l == 2)
                    {
                        spit->append(append_first->pos - offset_d1, (append_first+1)->pos -offset_
↳d2, y);
                    }
                    if(l > 2)
                    {
                        for(ev::iterator eit = append_first; eit != append_last; eit +=2)
                        {
                            SplitPolygon sp = SplitPolygon();
                            sp.init(eit->pos - offset_d2, (eit+1)->pos - offset_d2, y);
                            splits.push_back(sp);
                        }
                    }
                    spit = std::find_if(splits.begin(), splits.end(), [x1,x2,y,offset_d2,offset_d1]_
↳(SplitPolygon & sp)

```

(continues on next page)

(continued from previous page)

```

        {
            return ((sp.line ==y) && !((sp.rx < x1) || (sp.lx > x2)));
        });
        if(spit == splits.end())
        {
            SplitPolygon sp = SplitPolygon();
            sp.init(x1,x2,y);
            splits.push_back(sp);
            spit_last = splits.begin();
            append_first = ei + 2;
            append_last = ei + 2;
            advance = true;
        }
        else
        {
            append_first = ei;
            append_last = ei+2;
        }
    }
    else
    {
        append_last += 2;
    }
}
else
{
    spit_last = spit;
    spit = splits.begin();
    SplitPolygon sp = SplitPolygon();
    sp.init(x1,x2,y);
    splits.push_back(sp);
    append_first = ei + 2;
    append_last = ei + 2;
    advance = true;
}
}

int l = append_last - append_first;
if(l == 2)
{
    spit->append(append_first->pos - offset_d1, (append_first+1)->pos -offset_d2, y);
}
else if(l > 2)
{
    for(ev::iterator eit = append_first; eit != append_last; eit +=2)
    {
        SplitPolygon sp = SplitPolygon();
        sp.init(eit->pos - offset_d2,(eit+1)->pos - offset_d2,y);
        splits.push_back(sp);
    }
}
spit = splits.begin() + begin;
}
for(auto sp = splits.rbegin(); sp!=splits.rend(); sp++)
{
    for(auto mp = sp->merge_to->rbegin(); mp != sp->merge_to->rend(); mp++)
    {
        sp->right_insert(*((*mp)->right));
    }
    sp->right_merge();
}

```

(continues on next page)

(continued from previous page)

```

        if(!sp->merged)
        {
            polygons.push_back(*(sp->right));
        }
    }
    return polygons;
}

} //end namespace drclean

```

8.2 CleanerMaster Source

```

// This file is part of KLayoutPhotonicPCells, an extension for Photonic Layouts in KLayout.
// Copyright (c) 2018, Sebastian Goeldi
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Affero General Public License as
// published by the Free Software Foundation, either version 3 of the
// License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU Affero General Public License for more details.
//
// You should have received a copy of the GNU Affero General Public License
// along with this program. If not, see <https://www.gnu.org/licenses/>.

#include "DrcSl.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <stdexcept>
#include <cmath>

#include <boost/interprocess/managed_shared_memory.hpp>
#include <boost/interprocess/containers/vector.hpp>
#include <boost/interprocess/allocators/allocator.hpp>

namespace drclean
{
    // Function to compare two edgecoord structs. This is necessary for std::sort. If they are on_
    ↪ the same coordinate sort for type in descending order
    bool compare_edgecoord(edgecoord e1, edgecoord e2)
    {
        if (e1.pos==e2.pos)
            return (e2.type<e1.type);
        else
            return (e1.pos<e2.pos);
    }

    // Constructor. Initialize the pointers as nullptrs
    DrcSl::DrcSl()

```

(continues on next page)

(continued from previous page)

```

{
    this->lver = nullptr;
    this->lhor = nullptr;
}

//    Destructor: Delete the allocated vectors.
DrcSl::~DrcSl()
{
    if(this->lhor != nullptr)
        delete[] this->lhor;
    if(this->lver != nullptr)
        delete[] this->lver;
}

//    Add a complete data-set. Currently not used and not exposed in the Python interface.
int DrcSl::set_data(std::vector<edgecoord> *horlist)
{
    this->l = horlist;
    return 0;
}

//    Initialize the dimensions of the vector arrays and set pointers accordingly and dimension_
↪units.
void DrcSl::initialize_list(int hor1,int hor2, int ver1, int ver2, int violation_space, int_
↪violation_width)
{
    if(this->lhor)
    {
        delete[] this->lhor;
        this->lhor = nullptr;
    }
    if(this->lver)
    {
        delete[] this->lver;
        this->lver = nullptr;
    }
    this->lhor = new std::vector<edgecoord>[ver2-ver1+5];
    this->lver = new std::vector<edgecoord>[hor2-hor1+5];
    this->l = this->lhor;
    this->sver = hor2-hor1+5;
    this->shor = ver2-ver1+5;
    this->hor1 = hor1-2;
    this->hor2 = hor2+2;
    this->ver1 = ver1-2;
    this->ver2 = ver2+2;
    this->violation_space = violation_space;
    this->violation_width = violation_width;
    this->orientation = hor;
}

//    Print the complete data set or from index beg -> end if they are set.
//    -1, -1 will result in printing the whole vector.
void DrcSl::printvector(int beg, int last)
{
    if (last == -1 && beg == -1)
    {
        last = this->orientation ? this->ver2 : this->hor2;
        beg = this->orientation ? this->ver1 : this->hor1;
    }
    std::vector<edgecoord>::iterator it;

```

(continues on next page)

(continued from previous page)

```

std::cout << "size, y: " << this->sver << std::endl << "size, x: " << this->shor << std::endl;
int offset = this->orientation ? -this->hor1 : -this->ver1;
int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;
std::cout << "beg/end " << beg+offset << '/' << last+offset-1 << std::endl;

for (int i = 0; i < this->s(); i++)
{
    std::cout << "row: " << i-offset << ": [";
    for(it = this->l[i].begin(); it != this->l[i].end(); it++)
    {
        std::cout << "(" << it->pos -offset_d2<< ", " << it->type << ")";
    }
    std::cout << "]" << std::endl;
}
}

// Get the current array size of the vectors
int DrcSl::s()
{
    return this->orientation ? this->sver : this->shor;
}

// Sort all data with compare_edge_coord and remove overlapping edges, i.e. merge overlapping_
// polygons in the data
void DrcSl::sortlist()
{
    // std::cout << this->s() << std::endl;
    for (int i = 0; i < this->s(); i++)
    {
        if (!this->l[i].empty())
        {
            std::sort(this->l[i].begin(), this->l[i].end(), compare_edgecoord);
            std::vector<edgecoord>::iterator it;
            it = this->l[i].begin();
            int c = 0;
            for(; it != this->l[i].end(); it++)
            {
                if (it->type == 0)
                {
                    c++;
                    if (c>1 || c<0)
                    {
                        it->rem = true;
                    }
                }
                else
                {
                    if (c>1 || c<0)
                    {
                        it->rem = true;
                    }
                    c--;
                }
            }
            this->l[i].erase(std::remove_if(this->l[i].begin(), this->l[i].end(), [](auto o)
            {
                return o.rem;
            }), this->l[i].end());
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
}

//    Get data from a row (or column).
//    If used after the standard sorting or cleaning function, i.e. sortlist() and cleaning(),
//    the vectors should always be arranged row-oriented, meaning the same format as when added to
↳the cleaner.
std::vector<int> DrcSl::get_vect(int ind)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;

    std::vector<int> res = std::vector<int>(this->l[ind+offset].size());
    std::vector<edgecoord>::iterator it;
    int i;
    for(it = this->l[ind+offset].begin(), i=0; it !=this->l[ind+offset].end(); it++, i++)
    {
        if (it->type)
            res[i]=(it->pos-1-offset_d2);
        else
            res[i]=(it->pos+1-offset_d2);
    }

    return res;
}

//    Function to print the types in a vector. Probably only useful for debugging purposes
std::vector<int> DrcSl::get_types(int ind)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;

    offset ++;

    std::vector<int> res = std::vector<int>(this->l[ind+offset].size());
    std::vector<edgecoord>::iterator it;
    int i;
    for(it = this->l[ind+offset].begin(), i=0; it !=this->l[ind+offset].end(); it++, i++)
    {
        res[i] = it->type;
    }

    return res;
}

//    Add data to the data structure. We manhattanize the edge from the input and mark left facing
↳edges with -1 and
//    right facing edges with +1. The get_vect() function reverses this effect.
//    This should have no influence on any possible data except that it merges touching polygons.
void DrcSl::add_data(int px1, int px2, int py1, int py2)
{
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d2 = this->orientation ? -this->ver1 : -this->hor1;

    if (py2 > py1)
    {
        edgecoord p = edgecoord(px1+offset_d2-1, 0);

        double dx = (double)(px2-px1)/(py2-py1);
        double x = p.pos;
        if (p.pos < 0 || p.pos > (this->orientation ? this->shor : this->sver))
        {

```

(continues on next page)

(continued from previous page)

```

        std::cout << "Error ROW (y) index out of bound " << p.pos << '/' << (this->orientation ?
↪ this->shor: this->sver) << std::endl;
        throw 1;
    }
    if (offset+py1 < 0 || py2+offset > (this->orientation ? this->sver : this->shor))
    {
        std::cout << "Error COLUMN (x) index out of bound" << py2+offset << "/" << this->s() <<
↪ std::endl;
        throw 2;
    }

    if (dx > 0)
    {
        for(int i = offset+py1; i < py2+offset; i++)
        {
            this->l[i].push_back(p);
            x+=dx;
            p.pos = int(x);
        }
    }
    else
    {
        for(int i = offset+py1; i < py2+offset-1; i++)
        {
            x+=dx;
            p.pos = int(x);
            this->l[i].push_back(p);
        }
        p.pos = px2+offset_d2-1;
        this->l[py2+offset-1].push_back(p);
    }

}
else if (py1 > py2)
{
    edgecoord p = edgecoord(px2+offset_d2+1,1);

    double dx = (double)(px1-px2)/(py1-py2);
    double x = p.pos;
    if (p.pos < 0 || p.pos > (this->orientation ? this->shor : this->sver))
    {
        std::cout << "Error ROW (y) index out of bound " << p.pos << '/' << (this->orientation ?
↪ this->shor: this->sver) << std::endl;
        throw 1;
    }
    if (offset+py1 < 0 || py2+offset > this->s())
    {
        std::cout << "Error COLUMN (x) index out of bound" << std::endl;
        throw 2;
    }

    if (dx < 0)
    {
        for(int i = offset+py2; i < py1+offset; i++)
        {
            this->l[i].push_back(p);
            x+=dx;
            p.pos = std::ceil(x);
        }
    }
    else

```

(continues on next page)

(continued from previous page)

```

    {
        for(int i = offset+py2; i < py1+offset-1; i++)
        {
            x+=dx;
            p.pos = std::ceil(x);
            this->l[i].push_back(p);
        }
        p.pos = px1+offset_d2+1;
        this->l[py1+offset-1].push_back(p);
    }
}

// Clean data for space violations in the current orientation (row-oriented for violations,
// within the row and accordingly if column-oriented).
int DrcSl::clean_space()
{
    //Cleans space violations.
    //Returns number of space violations that were cleaned.
    std::vector<edgecoord> *il = this->l;

    //Counters to keep track of how many checks were done and how many space violations have been
    //cleaned.
    int spacevios = 0;
    int counts = 0;

    std::vector<edgecoord>::iterator it;

    for (int i = 0; i<this->s(); i++)
    {
        if (!il->empty())
        {
            bool er = false;
            it = il->begin();
            if (it == il->end())
                continue;
            it++;
            while(it+1 != il->end())
            {
                counts++;
                if ((it+1)->pos - it->pos < violation_space -1)
                {
                    er = true;
                    spacevios++;
                    it->rem = true;
                    (it+1)->rem = true;
                }
                it+=2;
            }
            if (er)
                il->erase(std::remove_if(il->begin(),il->end(),[](auto o)
                {
                    return o.rem;
                })),il->end());
            il++;
        }
    }

    // If progress output is desired uncomment the following lines
    // std::cout << "number of checks: " << counts << std::endl;
    // std::cout << "violations, space: " << spacevios << std::endl;
    return spacevios;
}

```

(continues on next page)

(continued from previous page)

```

}

// Clean data for width violation
int DrcSl::clean_width()
{
    std::vector<edgecoord> *il = this->l;

    int widthvios = 0;
    int counts = 0;

    std::vector<edgecoord>::iterator it;

    for (int i = 0; i<this->s(); i++)
    {
        if (!il->empty())
        {
            bool er=false;
            it = il->begin();
            while(it != il->end())
            {
                counts++;
                if ((it+1)->pos - it->pos < violation_width +1)
                {
                    er = true;
                    it->rem = true;
                    (it+1)->rem = true;
                    widthvios++;
                }
                it+=2;
            }
            if (er)
            {
                il->erase(std::remove_if(il->begin(),il->end(),[](auto o)
                {
                    return o.rem;
                }),il->end());
            }
            il++;
        }
    }

    // If progress output is desired uncomment the following lines
    // std::cout << "number of checks: " << counts << std::endl;
    // std::cout << "violations, width: " << widthvios << std::endl;
    return widthvios;
}

// Calculate difference between two rows or two columns. This is necessary when switching from
↳row-oriented to
// column-oriented data and vice-versa.
//
// In theory this can also be used to check for minimum edge-lengths. But for us all of these
↳requirements have been
// waived, so we don't have to check for those.
std::vector<int> DrcSl::listdif(std::vector<edgecoord> &l1, std::vector<edgecoord> &l2)
{
    /*
    ** Calculates differences between rows (or columns, depending on orientation) between two
    ↳vectors (rows/columns)
    ** The difference between the two vectors indicate that there is a polygon border for the
    ↳other orientation of the scanlines

```

(continues on next page)

(continued from previous page)

```

** This border corresponds to edges and thus has to appear in the opposite orientation
*/

/*
** Example:
**
** l1 is the row/column that we compare to. Any coordinates that appear in l1, but not in l2,
↪ will be returned as ranges.
** example:
** l1 = ([1,5],[7,10],[18,20])
** l2 = ([4,11],[15,16])
** out = ([1,3],[18,20])
**/

std::vector<int> out;
std::vector<edgecoord>::iterator it1 = l1.begin();
std::vector<edgecoord>::iterator it2 = l2.begin();
int l21;
int l22;
for (it1 = l1.begin(); it1 != l1.end(); it1++)
{
    int b = it1->pos;
    it1++;
    int e = it1->pos;
    int ee = e;
    bool add = true;
    while(it2 != l2.end())
    {
        l21 = it2->pos;
        l22 = (it2+1)->pos;
        if(l22 < b)
        {
            it2+=2;
        }
        else if (l22 >= e)
        {
            if (e < b || l21 <= b)
                add = false;
            if (e > l21 -1)
                e = l21 -1;
            break;
        }
        else if (l22 < e && l21 > b)
        {
            out.push_back(b);
            out.push_back(l21 -1);
            b = l22 + 1;
            e = ee;
            it2 += 2;
        }
        else if (l22 >= b && b >= l21)
            b = l22 + 1;
        else if (l21 <= e && e <= l22)
        {
            e = l21 + 1;
            break;
        }
    }
    if (add)
    {

```

(continues on next page)

(continued from previous page)

```

        out.push_back(b);
        out.push_back(e);
    }
}
return out;
}

// Switch dimensions. When calculating listdiffs between two rows, we can calculate the edges in_
↳row direction when
// row-oriented or in column direction when column-oriented. These edges then give us column-
↳orientation data and vice-versa.
void DrcSl::switch_dimensions()
{
    /*
    ** Switch row to column orientation of the scanlines.
    ** Example:
    **
    ** 5:      []
    ** 6:      []
    ** 7:      [(4,0),(10,1)]
    ** 8:      [(3,0),(7,1),(8,0),(11,1)]
    ** 9:      [(3,0),(8,1),(8,0),(11,1)]
    ** 10:     [(4,0),(8,0),(8,1),(12,1)]
    ** 11:     [(4,0),(7,1)]
    ** 12:     []
    ** 13:     []
    **
    ** Will be converted to:
    **
    ** 3:      []
    ** 4:      [(7,0),(10,1)]
    ** 5:      [(6,0),(12,1)]
    ** 6:      [(6,0),(12,1)]
    ** 7:      [(6,0),(8,1),(8,0),(11,1)]
    ** 8:      [(6,0),(8,1)]
    ** 9:      [(6,0),(11,1)]
    ** 10:     [(7,0),(11,1)]
    ** 11:     [(9,0),(11,1)]
    ** 12:     []
    **
    */

    // If progress output is desired uncomment the following lines
    // std::cout << "Switching dimensions" << std::endl;
    if(this->lhor == nullptr)
        this->lhor = new std::vector<edgecoord>[this->ver2-this->ver1];
    if(this->lver == nullptr)
        this->lver = new std::vector<edgecoord>[this->hor2-this->hor1];

    std::vector<edgecoord> *l_new;

    if (this->orientation)
    {
        for(int i = 0; i<this->shor; i++)
        {
            this->lhor[i].clear();
        }
        l_new = this->lhor;
    }
    else

```

(continues on next page)

(continued from previous page)

```

{
    for(int i = 0; i<this->sver; i++)
    {
        this->lver[i].clear();
    }
    l_new = this->lver;
}
std::vector<edgecoord> row_last;
std::vector<edgecoord> row;
std::vector<edgecoord> row_next;
std::vector<edgecoord> *it = this->l;
std::vector<int>::iterator dit;
std::vector<int> dif1;
std::vector<int> dif2;
std::vector<edgecoord>::iterator rit;
row_last = *it;
for(rit = row_last.begin(); rit != row_last.end(); rit++)
{
    rit->pos++;
    rit++;
    rit->pos--;
}
it++;
row = *it;
for(rit = row.begin(); rit != row.end(); rit++)
{
    rit->pos++;
    rit++;
    rit->pos--;
}
it++;
int row_number = 2;
for (int n = 2; n < this->s(); n++)
{
    row_next = *it;
    for(rit = row_next.begin(); rit != row_next.end(); rit++)
    {
        rit->pos++;
        rit++;
        rit->pos--;
    }

    dif1 = listdif(row_last,row);
    dif2 = listdif(row_next,row);

    int b;
    int e;
    dit = dif1.begin();
    while(dit != dif1.end())
    {
        b = *dit;
        dit++;
        e = *dit;
        dit++;
        for (; b!=e+1; b++)
        {
            edgecoord p = edgecoord(row_number-1,1);
            l_new[b].push_back(p);
        }
    }
    dit = dif2.begin();
}

```

(continues on next page)

(continued from previous page)

```

while(dit != dif2.end())
{
    b = *dit;
    dit++;
    e = *dit;
    dit++;
    for (; b!=e+1; b++)
    {
        edgecoord p = edgecoord(row_number-1,0);
        l_new[b].push_back(p);
    }
}
row_last = row;
row = row_next;
row_number++;
it++;
}
this->l = l_new;
this->orientation = this->orientation ? hor : ver;
}

//    Function that first cleans space violations then width violations and then space violations.
//    ↪again.
//    This does not necessarily clean all violations. For example if a fixing of a width violation.
//    ↪creates a space violation
//    and vice-versa, the algorithm will not fix the violation. For performance reasons
//    it is still the user's task to perform DRC and ensure the design is clean. For standard.
//    ↪photonic structures it is
//    unlikely that such a case occurs.
void DrcSl::clean(int maxtries)
{
    for(int i = 0; i < maxtries; i++)
    {
        if(clean_space())
        {
            switch_dimensions();
        }
        else
        {
            if(clean_space())
            {
                switch_dimensions();
                continue;
            }
            else
            {
                //                If progress output is desired uncomment the following lines
                //                std::cout<< "Finished after " << i+1 << " tries" << std::endl;
                break;
            }
        }
    }
    for(int i = 0; i < maxtries; i++)
    {
        if(clean_width())
        {
            //                If progress output is desired uncomment the following lines
            //                std::cout<< "Try: " << i << "/" << maxtries << std::endl;
            switch_dimensions();
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

        else
        {
            if(clean_width())
            {
                switch_dimensions();
                continue;
            }
            else
            {
                //          If progress output is desired uncomment the following lines
                //          std::cout<< "Finished after " << i+1 << " tries" << std::endl;
                break;
            }
        }
    }
    for(int i = 0; i < maxtries; i++)
    {
        if(clean_space())
        {
            //          If progress output is desired uncomment the following lines
            //          std::cout<< "Try: " << i << "/" << maxtries << std::endl;
            switch_dimensions();
        }
        else
        {
            if(clean_space())
            {
                switch_dimensions();
            }
            else
            {
                if (this->orientation)
                {
                    //          If progress output is desired uncomment the following lines
                    //          std::cout<< "Finished after " << i+1 << " tries" << std::endl;
                    switch_dimensions();
                    break;
                }
            }
        }
    }
}

//          If progress output is desired uncomment the following lines
//          std::cout<< "Done cleaning" << std::endl;
}

std::vector<std::vector<int>> DrcSl::get_lines()
{
    std::vector<std::vector<int>> lines (this->s());

    int offset_d2 = this->orientation ? -this->ver1 : -this-> hor1;

    for(int i = 0; i< this->s(); i++)
    {
        for(auto iter: this->l[i])
        {
            lines[i].push_back(iter.type ? iter.pos-1-offset_d2 : iter.pos+1-offset_d2);
        }
    }
    return lines;
}

```

(continues on next page)

(continued from previous page)

```

std::vector<std::vector<pi>> DrcSl::get_polygons()
{
    splits.clear();
    polygons.clear();
    int offset = this->orientation ? -this->hor1 : -this->ver1;
    int offset_d1 = (this->orientation ? -this->ver1 : -this->hor1) - 1;
    int offset_d2 = (this->orientation ? -this->ver1 : -this->hor1) + 1;
    int begin = 0;

    for(int i = 1; i < this->s(); i++)
    {
        int y = i - offset;
        spv::iterator spit = splits.begin() + begin;
        spv::iterator spit_last = splits.begin() + begin;

        ev::iterator append_first = this->l[i].begin();
        ev::iterator append_last = this->l[i].begin();

        bool advance = true;

        for(ev::iterator ei = this->l[i].begin(); ei != this->l[i].end(); ei+=2)
        {
            int x1 = ei->pos - offset_d1;
            int x2 = (ei+1)->pos - offset_d2;

            if(advance)
            {
                spit = std::find_if(splits.begin(), splits.end(), [x1,x2,y,offset_d2,offset_d1]_
↳(SplitPolygon & sp)
                {
                    return ((sp.line == y) && !((sp.rx < x1) || (sp.lx > x2)));
                });
                advance = false;
            }

            if(spit != splits.end())
            {
                int ex1 = spit->lx;
                int ex2 = spit->rx;

                if(((x1 > ex2) || (x2 < ex1)) && spit->line == y)
                {
                    int l = append_last - append_first;
                    if(l == 2)
                    {
                        spit->append(append_first->pos - offset_d1, (append_first+1)->pos -offset_
↳d2, y);
                    }
                    if(l > 2)
                    {
                        for(ev::iterator eit = append_first; eit != append_last; eit +=2)
                        {
                            SplitPolygon sp = SplitPolygon();
                            sp.init(eit->pos - offset_d2, (eit+1)->pos - offset_d2,y);
                            splits.push_back(sp);
                        }
                    }
                    spit = std::find_if(splits.begin(), splits.end(), [x1,x2,y,offset_d2,offset_d1]_
↳(SplitPolygon & sp)
                    {

```

(continues on next page)

(continued from previous page)

```

        return ((sp.line ==y) && !((sp.rx < x1) || (sp.lx > x2)));
    });
    if(spit == splits.end())
    {
        SplitPolygon sp = SplitPolygon();
        sp.init(x1,x2,y);
        splits.push_back(sp);
        spit_last = splits.begin();
        append_first = ei + 2;
        append_last = ei + 2;
        advance = true;
    }
    else
    {
        append_first = ei;
        append_last = ei+2;
    }
}
else
{
    append_last += 2;
}
}
else
{
    spit_last = spit;
    spit = splits.begin();
    SplitPolygon sp = SplitPolygon();
    sp.init(x1,x2,y);
    splits.push_back(sp);
    append_first = ei + 2;
    append_last = ei + 2;
    advance = true;
}
}

int l = append_last - append_first;
if(l == 2)
{
    spit->append(append_first->pos - offset_d1, (append_first+1)->pos -offset_d2, y);
}
else if(l > 2)
{
    for(ev::iterator eit = append_first; eit != append_last; eit +=2)
    {
        SplitPolygon sp = SplitPolygon();
        sp.init(eit->pos - offset_d2, (eit+1)->pos - offset_d2,y);
        splits.push_back(sp);
    }
}
spit = splits.begin() + begin;
}
for(auto sp = splits.rbegin(); sp!=splits.rend(); sp++)
{
    for(auto mp = sp->merge_to->rbegin(); mp != sp->merge_to->rend(); mp++)
    {
        sp->right_insert(*((*mp)->right));
    }
    sp->right_merge();
    if(!sp->merged)

```

(continues on next page)

(continued from previous page)

```

        {
            polygons.push_back(*(sp->right));
        }
    }
    return polygons;
}

} //end namespace drclean

```

8.3 CleanerMain Source

```

// This file is part of KLayoutPhotonicPCells, an extension for Photonic Layouts in KLayout.
// Copyright (c) 2018, Sebastian Goeldi
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Affero General Public License as
// published by the Free Software Foundation, either version 3 of the
// License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU Affero General Public License for more details.
//
// You should have received a copy of the GNU Affero General Public License
// along with this program. If not, see <https://www.gnu.org/licenses/>.

#include "CleanerSlave.h"

int main(int argc, char* argv[])
{
    std::cout<< "Initializing" << std::endl;
    drclean::CleanerSlave cs = drclean::CleanerSlave();

    std::cout<< "Initialized" << std::endl;

    if (!cs.initialized)
    {
        return -1;
    }

    SignalHandler signalHandler;
    signalHandler.setSignalToHandle(SIGUSR1);

    while(!signalHandler.isSignalSet())
    {
        cs.clean();
    }

    cs.join_threads();

    return 0;
}

```

8.4 CleanerSlave Source

```
// This file is part of KLayoutPhotonicPCells, an extension for Photonic Layouts in KLayout.
// Copyright (c) 2018, Sebastian Goeldi
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Affero General Public License as
// published by the Free Software Foundation, either version 3 of the
// License, or (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU Affero General Public License for more details.
//
// You should have received a copy of the GNU Affero General Public License
// along with this program. If not, see <https://www.gnu.org/licenses/>.

#include "CleanerSlave.h"

namespace drclean
{
CleanerSlave::CleanerSlave()
{
    segment = new bi::managed_shared_memory(bi::open_only, "DRCleanEngine");

    std::cout<< "Initializing" << std::endl;

    alloc_inst = new ShmemAllocatorInt(segment->get_segment_manager());
    alloc_vec = new ShmemAllocatorIVec(segment->get_segment_manager());
    alloc_pvec = new ShmemAllocatorPVec(segment->get_segment_manager());
    alloc_poly = new ShmemAllocatorPair(segment->get_segment_manager());

    input = segment->find<ShIVector>("input").first;
    outList = segment->find<ShIVector>("outList").first;
    // imList = segment->find<ShIVector>("imList").first;

    mux_inp = new bi::named_mutex(bi::open_only, "mux_inp");
    mux_out = new bi::named_mutex(bi::open_only, "mux_out");

    pool = new boost::asio::thread_pool(boost::thread::hardware_concurrency());

    if (input)
    {
        initialized = true;
    }
}

CleanerSlave::~CleanerSlave()
{
    join_threads();
    delete alloc_inst;
    delete pool;
}

void CleanerSlave::clean()
{
    std::vector<int> *inp = new std::vector<int>();
    mux_inp->lock();
    if(!input->empty())
```

(continues on next page)

(continued from previous page)

```

{
    bi::vector<int, ShmemAllocatorInt>::iterator it;
    for(it = input->begin(); it != input->end(); it++)
    {
        inp->push_back(*it);
    }
    input->clear();
    mux_inp->unlock();
}
else
{
    mux_inp->unlock();
    delete inp;

    std::this_thread::sleep_for(std::chrono::milliseconds(30));
    return;
}

boost::asio::post(*pool, boost::bind(&CleanerSlave::threaded_DrcSl, this, inp));
// threaded_DrcSl(inp); //For single thread calculation
}

void CleanerSlave::threaded_DrcSl(std::vector<int> *inp)
{
    int layer;
    int datatype;

    DrcSl sl;
    std::vector<int>::iterator iter = inp->begin();

    if(iter!=inp->end())
    {
        layer = *(iter++);
        datatype = *(iter++);
        sl.initialize_list(*(iter),*(iter+1),*(iter+2),*(iter+3),*(iter+4),*(iter+5));
        // The first six datapoints are size (x1,x2,y1,y2) and layer, datatype information.
        int count = 6;
        iter+=6;
        while(iter!=inp->end())
        {
            count +=4;
            sl.add_data(*(iter),*(iter+1),*(iter+2),*(iter+3));
            iter+=4;
        }
    }
    else
    {
        delete inp;
        return;
    }

    delete inp;
    sl.sortlist();
    sl.clean();
    std::string layername = std::to_string(layer) + "/" + std::to_string(datatype);

    std::vector<std::vector<pi>> polys = sl.get_polygons();

    ShPVVector* polygons = segment->construct<ShPVVector>(layername.data())(*alloc_pvec);

    for(auto p: polys)

```

(continues on next page)

(continued from previous page)

```
{
    ShPVector* poly = segment->construct<ShPVector>(bi::anonymous_instance) (*alloc_poly);
    for(auto pit: p)
    {
        poly->push_back(pit);
    }
    polygons->push_back(boost::move(*poly));
}
mux_out->lock();
outList->push_back(layer);
outList->push_back(datatype);
mux_out->unlock();
}

void CleanerSlave::join_threads()
{
    pool->join();
}

};
```

Glossary:

- `genindex`
- `modindex`
- `search`

A

`add_edge()` (*PyCleanerMaster* method), 13

C

`clean()` (*kppc.drc.slcleaner.PyDrcSl* method), 11

`clean_space()` (*kppc.drc.slcleaner.PyDrcSl* method), 11

`clean_width()` (*kppc.drc.slcleaner.PyDrcSl* method), 11

`CleanerMaster` (C++ class), 14

`CleanerMaster::[anonymous]` (C++ union), 14

`CleanerMaster::[anonymous]::add_edge` (C++ function), 14

`CleanerMaster::[anonymous]::CleanerMaster` (C++ function), 14

`CleanerMaster::[anonymous]::CleanerMaster::get_layer` (C++ function), 14

`CleanerMaster::[anonymous]::CleanerMaster::get_polygons` (C++ function), 14

`CleanerMaster::[anonymous]::done` (C++ function), 14

`CleanerMaster::[anonymous]::set_box` (C++ function), 14

`CleanerSlave` (C++ class), 14

`CleanerSlave::clean` (C++ member), 15

`CleanerSlave::CleanerSlave` (C++ member), 14

`CleanerSlave::join_threads` (C++ member), 15

D

`done()` (*PyCleanerMaster* method), 13

G

`get_layer()` (*PyCleanerMaster* method), 13

`get_row()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

`get_row_types()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

I

`init_list()` (*kppc.drc.slcleaner.PyDrcSl* method), 11

K

`kppc.drc.slcleaner.PyDrcSl` (built-in class), 11

P

`polygons()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

`polygons()` (*PyCleanerMaster* method), 13

`printvector()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

`PyCleanerMaster` (built-in class), 13

S

`s()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

`set_box()` (*PyCleanerMaster* method), 13

`sort()` (*kppc.drc.slcleaner.PyDrcSl* method), 12

`switch_dimensions()` (*kppc.drc.slcleaner.PyDrcSl* method), 12