KLayout Photonic PCells Documentation

Release 0.1

Sebastian Goeldi

Nov 28, 2018

CONTENTS:

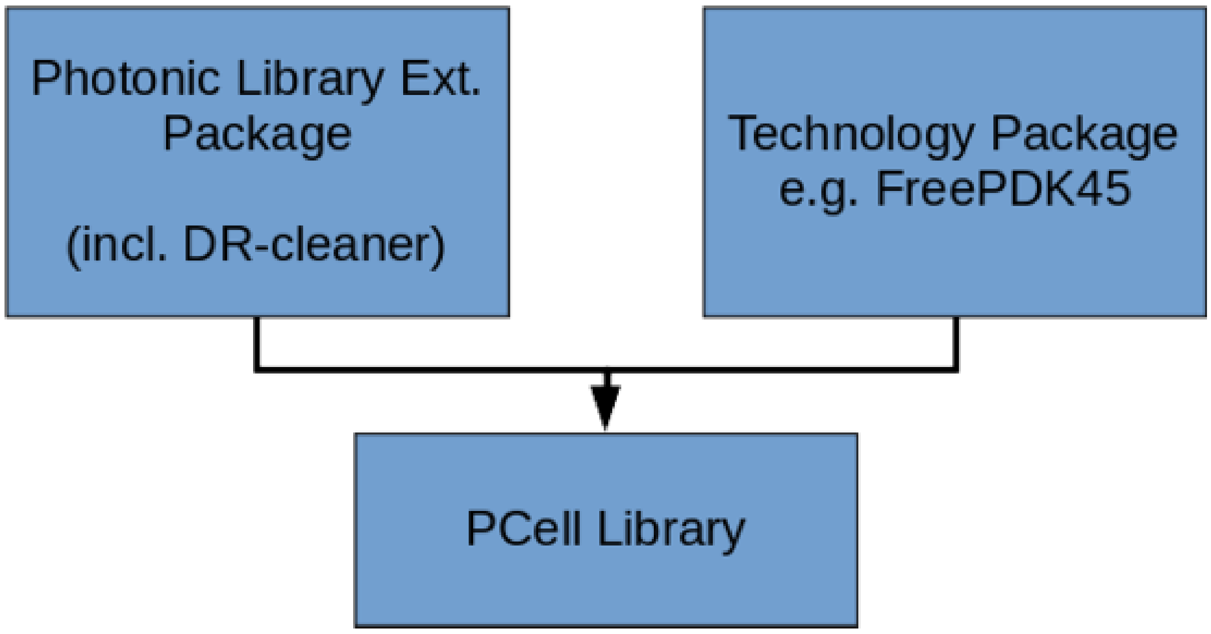
CHAPTER

ONE

INTRODUCTION

The *pcell_lib_ext* 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 Trans[1] 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



Photonic Library Ext. Package
(incl. DR-cleaner)
Technology Package e.g. FreePDK45
PCell Library

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

CHAPTER

TWO

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 pythons Setuptools[2] and Cython[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.

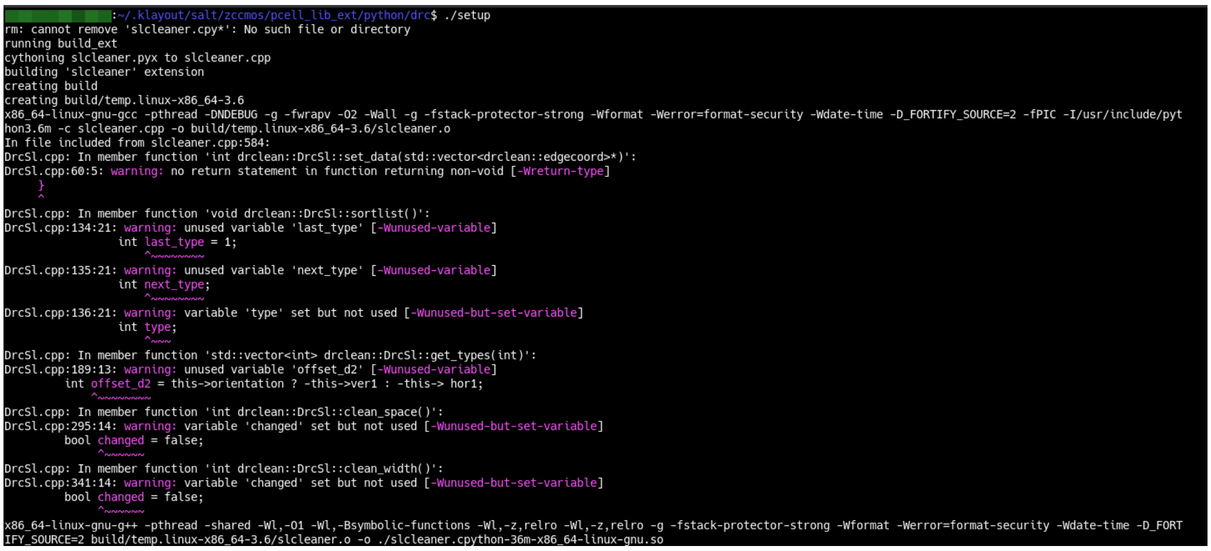
2.2 Installation

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/zccmos/pcell_ext_lib` folder. If you installed the FreePDK45_Cells & FreePDK45_tech, then you should have the folders `~/.klayout/salt/zccmos/FreePDK45_Cells` and `~/.klayout/salt/zccmos/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/zccmos/pcell_ext_lib/python/drc/
./setup
```

[2] https://setuptools.readthedocs.io/en/latest/index.html
[3] https://cython.readthedocs.io/en/latest/index.html



```
[redacted]:~/.klayout/salt/zccmos/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 -DNDEBUG -g -fwrapv -O2 -Wall -g -fstack-protector-strong -Wformat -Werror=format-security -Wdate-time -D_FORTIFY_SOURCE=2 -fPIC -I/usr/include/pyt
hon3.6m -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_FORT
IFY_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.

CHAPTER

THREE

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 `photonics.PortCreation` when overriding the `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 alond 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.

CHAPTER

FOUR

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 creats 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:

```xml
<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>
    <valid>true</valid>
```

(continues on next page)

(continued from previous page)

```
    <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>
<properties>
```

(continues on next page)

(continued from previous page)

```
    <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 befor 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 layername | layernumber | layer-datatype on each line for each layer. They have to be separated by white spaces. Afterwards, they can by 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.

CHAPTER

FIVE

CODE DOCUMENTATION OF KLAYOUTPHTONICPCELLS

5.1 drc package

5.1.1 Module contents

This module uses the C++ submodule *slcleaner*. It has to be compiled after installing the extension.

To compile the module execute the setup script `python/drc/compile.sh`. Or alternatively execute the `python/drc/slcleaner_source/setup.py` with the python3 executable and copy/move the resulting `slcleaner.[...].so` library file ino the `python/drc/` folder.

For further information consult the Cython Documentation[4].

To execute the script open a console and execute the following commands:

```
cd ~/.klayout/salt/zccmos/pcell_ext_lib/python/drc
./compile.sh
```

The bash script executes the following commands:

```
#!/bin/bash

#Script that compiles the C++ scanline algorithm with cython to a python module and copies it into␣
↪the current folder
#If there is an __init__.py in the folder the setup script will create subfolders, so avoid that
cd "$(dirname "$0")"
cd slcleaner_source
python3 setup.py build_ext -b ./
/usr/bin/python3 setup.py build_ext -b ./
cp slcleaner.cpython* ../
```

`drc.`**`clean`**(*cell*, *cleanrules*)

Clean a cell for width and space violations. This function will clear the output layers of any shapes and insert a cleaned region.

Parameters

- **cell** (<sphinx.ext.autodoc.importer._MockObject object at 0x7efd290cd550>) – pointer to the cell that needs to be cleaned
- **cleanrules** (list[5]) – list with the layerpurposepairs, violationwidths and violationspaces in the form [[[layer, purpose], violationwidth, violationspace], [[layer2, purpose2], violationwidth2, violationspace2], . . .]

5.1.2 Submodules

drc.slcleaner module

An interface to the DrcSl.cpp Class.

[4] http://cython.org/
[5] https://docs.python.org/3/library/stdtypes.html#list

`class 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`[6]) – x position of p1 of the edge
- **x2** (`int`[7]) – y position of p1 of the edge
- **y1** (`int`[8]) – x position of p2 of the edge
- **y2** (`int`[9]) – y position of p2 of the edge

`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`) – bottom bound of box
- **y2** (`top bound of box`) – 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

`sort`()

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

`clean`(*x = 10*)

Clean data in the vector for space and width violations.

Parameters **x** – number of max tries

`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

[6] https://docs.python.org/3/library/functions.html#int
[7] https://docs.python.org/3/library/functions.html#int
[8] https://docs.python.org/3/library/functions.html#int
[9] https://docs.python.org/3/library/functions.html#int

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

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)

clean_space()
Clean the current data for space violations.

clean_width()
Clean the current data for width violations.

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

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

Returns Size of the array of vectors.

Return type int[10]

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[11] to first convert the polygons from KLayout to manhattanized edges and then add them into an array representation of the polygon edges.

C++ Code:

```
//  This file is part of KLayout-photonics, 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>

namespace drclean{
```

(continues on next page)

10 https://docs.python.org/3/library/functions.html#int
11 https://en.wikipedia.org/wiki/Scanline_rendering

(continued from previous page)

```
//    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()
    {
        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;
    }

//    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;
    }
```

(continues on next page)

(continued from previous page)

```
//    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;

        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;
                int last_type = 1;
                int next_type;
                int type;
                it = this->l[i].begin();
                type = it->type;
                int c = 0;
                for(;it != this->l[i].end();it++)
                {
                    if (it->type == 0)
                    {
                        c++;
                        if (c>1 || c<0)
                        {
                            it->rem = true;
```

(continues on next page)

(continued from previous page)

```
                        }
                    }
                    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());

            }
        }
    }

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

        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
```

(continues on next page)

(continued from previous page)

```
//    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))
            {
                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+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())
            {
```

(continues on next page)

(continued from previous page)

```
                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
            {
            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.
        bool changed = false;
        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++;
                        changed = true;
                        it->rem = true;
```

(continues on next page)

(continued from previous page)

```
                        (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;

    }

//    Clean data for width violation
    int DrcSl::clean_width()
    {
        bool changed = false;
        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;
                        changed = 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;

    }
```

(continues on next page)

(continued from previous page)

```
//    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
        **  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);
```

(continues on next page)

(continued from previous page)

```
                    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)
            {
                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)
```

(continues on next page)

(continued from previous page)

```
            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
        {
            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;
```

(continues on next page)

(continued from previous page)

```
            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();
            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
```

(continues on next page)

(continued from previous page)

```
                {
//                      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();
            }
            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;
    }
}//end namespace drclean
```

5.2 photonics package

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

Photonic PCell-Extension Module

Warning: Before using this module for the first time, make sure the *drc.slcleaner* submodule is compiled and importable, as this module relies on the drc package for DR-Cleaning. See `drc` for further details.

A Module which provides extensions for standard KLyaout-PCells. This extension mainly provides functionalities for photonics. One main feature of photonics are so-called ports. These define a position and a direction on a Cell. They indicate where multiple Cells/Devices should interact with each other. For example, one can connect a waveguide with a linear taper. This module provides the classes and functions for this functionality. Additionally, this module provides a lot of convenience functions for interactions with the KLayout-API.

The main functionality for this module is in the class `PhotDevice`.

Warning: When using this module to extend a PCell-Library any PCell class has to assign valid values to the parameters `layermap` , `dataprep_config` , `clean_rules` . These are accessed by `PhotDevice`. If they aren't declared, a runtime error will occur.

class photonics.**InstanceHolder**(*cell_name*, *lib*, *pcell_decl*, *params=None*, *params_mod=None*, *id=0*)

Bases: `object`[12]

Class to keep track and hold the information of a pcell instance. The information will be processed to a PCell in `produce_impl()`

move(*x=0*, *y=0*, *rot=0*, *mirrx=False*, *mag=1*)

Moves an instance. Units of microns relative to origin.

Parameters

- **x** (`float`[13]) – x position where to move
- **y** (`float`[14]) – y position where to move

[12] https://docs.python.org/3/library/functions.html#object
[13] https://docs.python.org/3/library/functions.html#float
[14] https://docs.python.org/3/library/functions.html#float

- **rot** (int[15]) – Rotation of the object in degrees
- **mirrx** (bool[16]) – Mirror at x-axis if True
- **mag** (float[17]) – Magnification of the Cell. This feature is not tested well.

port(*port*)

Returns a reference to itself and the port number. No checks are made whether this port is valid or not! Available ports can be seen if such an object is instantiated.

Parameters **port** (int[18]) – index of the port

Returns self, port

port_to_port(*port*, *inst_holder*)

Attach one of this instance's ports to another instance's port.

Parameters

- **port** (int[19]) – port of this instance
- **inst_holder** (tuple[20]) – Tuple of the the reference to the other instance and the port to connect to. This is a tuple returned from <InstanceHolder object>.port(<portnumber>).

class photonics.**PhotDevice**

Bases: `sphinx.ext.autodoc.importer._MockObject`

Wrapper for calls to the Klayout API.

Variables

- **layermap** (dict[21]) – The layermap dictionary. This value has to be written by a child class. If undefined this class won't work and crash.
- **dataprep_config** (str[22]) – String with the path to the file containing the dataprep instructions. If left empty, dataprep will do nothing.
- **clean_rules** (list[23]) – String with the path to the file containing the DR-Cleaning rules. If left empty, DR-Cleaning will do nothing. If the cells are built similar to the FreePDK45-SampleCells example, DR-Cleaning will not work without dataprep, or will be without any effect.
- **keep** (bool[24]) – Parameter created during `__init__()` via pya.DeclarationHelper. If set to True in the PCell, all child-cells will be preserved at the end. If set to False only the Dataprep Sub-Cell will be preserved.
- **dataprep** (bool[25]) – If this flag is set, `photonics.dataprep.dataprep()` will be performed on the cell. The variable `dataprep_config` holds the path to the instructions for dataprep.
- **clean** (bool[26]) – If this flag is set, `drc.clean()` will be performed on the cell. Rules for the DR-Cleaning are pulled from `clean_rules`.

[15] https://docs.python.org/3/library/functions.html#int
[16] https://docs.python.org/3/library/functions.html#bool
[17] https://docs.python.org/3/library/functions.html#float
[18] https://docs.python.org/3/library/functions.html#int
[19] https://docs.python.org/3/library/functions.html#int
[20] https://docs.python.org/3/library/stdtypes.html#tuple
[21] https://docs.python.org/3/library/stdtypes.html#dict
[22] https://docs.python.org/3/library/stdtypes.html#str
[23] https://docs.python.org/3/library/stdtypes.html#list
[24] https://docs.python.org/3/library/functions.html#bool
[25] https://docs.python.org/3/library/functions.html#bool
[26] https://docs.python.org/3/library/functions.html#bool

- **top** (bool[27]) – Hidden parameter that indicates whether this cell is a top_cell. Default is yes. When an instance is added through `add_pcell_variant()` these cells will not be set to top_cells as they are instantiated from another cell.
- **only_top_ports** – GUI parameter. If set to true, only ports of the top most hierarchy level (top_cell) will be annotated by text.

add_layer(*var_name, name=", layer=0, datatype=0, ld=(), field_name=", hidden=False*)
Add a layer to the layer list of the pcell by name.

Parameters

- **var_name** (str[28]) – name of the variable
- **name** (str[29]) – name in the pcell window
- **layer** (int[30]) – layernumber
- **datatype** (int[31]) – layerdatatype
- **field_name** (str[32]) –
- **hidden** – hide in the GUI

Examples self.add_layer('lpp','rx1phot.drawing')

add_params(*params*)
Create the PCell conform dictionary from a parameter list

Parameters **params** (dict[33]) – Dictionary of parameters

add_pcell_variant(*params, number=1*)
Add variants of PCells. Creates a list of InstanceHolders and modifies their parameters accordingly.

Parameters

- **params** (dict[34]) – parameter list from which to create pcells
- **number** (int[35]) – Number of instances to create

Returns list of `photonics.InstanceHolder`

add_pcells(*instance_list*)
Creates list of instances of PCells. These are the effective Klayout cell instances.

Parameters **instance_list** (list[36]) – list of `photonics.InstanceHolder`

Returns list of instantiated pya.CellInstArray

calculate_ports(*instances*)
Calculates port locations in the cell layout. This is to propagate the port locations upwards

Parameters **instances** (list[37]) – list containing `photonics.InstanceHolder`

clear_ports()
Clears self.portlist and by that delete all ports. This is used when updating the Ports

[27] https://docs.python.org/3/library/functions.html#bool
[28] https://docs.python.org/3/library/stdtypes.html#str
[29] https://docs.python.org/3/library/stdtypes.html#str
[30] https://docs.python.org/3/library/functions.html#int
[31] https://docs.python.org/3/library/functions.html#int
[32] https://docs.python.org/3/library/stdtypes.html#str
[33] https://docs.python.org/3/library/stdtypes.html#dict
[34] https://docs.python.org/3/library/stdtypes.html#dict
[35] https://docs.python.org/3/library/functions.html#int
[36] https://docs.python.org/3/library/stdtypes.html#list
[37] https://docs.python.org/3/library/stdtypes.html#list

coerce_parameters_impl()

Method called by Klayout to update a PCell. For photonic PCells the ports are updated/calculated in the parameter of the PCell. And desired movement transformations are performed.

Because the calculated ports of our own PCell are used by parent cells and are needed before *~produce_impl*, we must calculate them twice. First to calculate where our own ports are and then again to instantiate the child cells. This is unfortunate but not a big problem, since dataprep and DR-cleaning take the majority of computation time.

Returns

connect_port(*pos1*, *portlist1*, *port1*, *pos2*, *portlist2*, *port2*)

Connect ports of two instances. The second instance will be transformed to attach to the first instance.

Parameters

- **pos1** (int[38]) – index of instance1
- **portlist1** (str[39]) – portlist of instance1
- **port1** (int[40]) – port number of instance1
- **pos2** (int[41]) – index of instance2
- **portlist2** (str[42]) – portlist of instance2
- **port2** (int[43]) – port number of instance2

Return type None

connect_port_to_port(*port1*, *port2*)

Connect Ports from two InstanceHolder instances.

Connect two *InstanceHolders* together. Attach <InstanceHolder instance1>.port(<port1>) to <InstanceHolder instance2>.port(<port2>). This will apply a transformation to Instance2. There can only be either a transformation through connect_port_to_port or through InstanceHolder.move

Parameters

- **port1** (tuple[44]) – <InstanceHolder instance1>.port(<port1>)
- **port2** (tuple[45]) – <InstanceHolder instance2>.port(<port2>)

create_param_inst()

To be overwritten by the effective PCell

Returns Iterable with the declarations of the child PCells.

create_path(*points*, *width*, *layer*)

Creates a pya.Path object and inserts it into the Library-PCell.

Parameters

- **points** (list[46]) – The points describing the path [[x1,y1],[x2,y2],. . .] in microns
- **width** (float[47]) – Path width

[38] https://docs.python.org/3/library/functions.html#int
[39] https://docs.python.org/3/library/stdtypes.html#str
[40] https://docs.python.org/3/library/functions.html#int
[41] https://docs.python.org/3/library/functions.html#int
[42] https://docs.python.org/3/library/stdtypes.html#str
[43] https://docs.python.org/3/library/functions.html#int
[44] https://docs.python.org/3/library/stdtypes.html#tuple
[45] https://docs.python.org/3/library/stdtypes.html#tuple
[46] https://docs.python.org/3/library/stdtypes.html#list
[47] https://docs.python.org/3/library/functions.html#float

- **layer** (<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474be0>) – layer on which the path should be made

create_polygon(*points*, *layer*)
Creates a Polygon and adjusts from microns to database units. Format: [[x1,y1],[x2,y2],...] in microns

Parameters

- **points** (list[48]) – Points defining the corners of the polygon.
- **layer** (int[49]) – layer_index of the target layer

Returns reference to polygon object

create_port(*x*, *y*, *rot=0*, *length=0*)
Creates a Port at the specified coordinates.

This function will be used when a port is created through the PortCreation tuple.

Parameters

- **x** (float[50]) – x Coordinate in microns
- **y** (float[51]) – y Coordinate in microns
- **rot** (int[52]) – Rotation in degrees
- **length** (int[53]) – length of the port in microns

decl(*libname*, *cellname*)
Get pya.PCellDeclaration of a cell in a library

Parameters

- **libname** (str[54]) – Name of the library
- **cellname** (str[55]) – Name of the cell

Returns pya.PCellDeclaration reference of PCell

flip_shape_xaxis(*shape*)
Flip a polygon (or any shape) at the x-axis

Parameters **shape** (<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474ac8>) – pya.Shape object (e.g. through photonicpcell.create_polygon obtained)

flip_shape_yaxis(*shape*)
Flip a polygon (or any shape) at the y-axis

Parameters **shape** (<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474eb8>) – pya.Shape object (e.g. through photonicpcell.create_polygon obtained)

get_layer(*name*, *purpose=”*)
Creates LayerInfo object

Creates a pya.LayerInfo object to find layer indexes in the current layout.

Parameters

- **name** (str[56]) – name of the layer

[48] https://docs.python.org/3/library/stdtypes.html#list
[49] https://docs.python.org/3/library/functions.html#int
[50] https://docs.python.org/3/library/functions.html#float
[51] https://docs.python.org/3/library/functions.html#float
[52] https://docs.python.org/3/library/functions.html#int
[53] https://docs.python.org/3/library/functions.html#int
[54] https://docs.python.org/3/library/stdtypes.html#str
[55] https://docs.python.org/3/library/stdtypes.html#str
[56] https://docs.python.org/3/library/stdtypes.html#str

- **purpose** (`str`[57]) – if not empty then layer and purpose are separate

Returns pya.LayerInfo about the layer

`get_transformations()`
Convert transformation strings back to pya.ICplxTrans objects

Returns list of pya.ICplxTrans objects

`insert_shape`(*shape*, *layer*)
Any other Klayout shape can be added to the PCell through this function.

Parameters

- **shape** (`<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6415588>`) – pya.Shape object
- **layer** (`<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474eb8>`) – layer where to write to

Returns reference to shape

`move_instance`(*ind*, *trans*, *mirror=False*)
Moves an InstanceHolder object

Parameters

- **ind** (`int`[58]) – id of the InstanceHolder
- **trans** (`<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474eb8>`) – list of transformations
- **mirror** (`bool`[59]) – bool whether to mirror the object

`produce_impl()`
Create the effective Klayout shapes. For this all the InstanceHolders are cycled through and all the child instances are created. Furthermore, if desired, dataprep is performed, which copies and sizes the shapes as desired. Dataprep will only create shapes on the topmost cell. Finally, if desired DR-cleaning is performed and in the process the shapes will be manhattanized.

`set_transformation`(*ind*, *trans*)
Transforms child cells to the intended position, defined either by connected ports or by manual positioning.

Parameters

- **ind** (`int`[60]) – index of the child cell
- **trans** (`<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474eb8>`) – Transformation object with which to transform the child cell

`shapes()`
To be overwritten by effective PCell if shapes should be desired.

Return type None[61]

`update_parameter_list`(*params*, *decl*)
Coerces parameter list. This is necessary to calculate port locations and update parameters in general.

Parameters

- **params** (`dict`[62]) – dict of parameters

[57] https://docs.python.org/3/library/stdtypes.html#str
[58] https://docs.python.org/3/library/functions.html#int
[59] https://docs.python.org/3/library/functions.html#bool
[60] https://docs.python.org/3/library/functions.html#int
[61] https://docs.python.org/3/library/constants.html#None
[62] https://docs.python.org/3/library/stdtypes.html#dict

- **decl** (<sphinx.ext.autodoc.importer._MockObject object at 0x7efcd6474470>) – pya.PCellDeclaration reference

Returns list of updated parameters

class photonics.**PortCreation**(*x*, *y*, *rot*, *length*)

Bases: tuple[63]

Custom namedtuple

This will hold informations for creating ports.

Parameters

- **x** (int[64]) – x Coordinate [microns]
- **y** (int[65]) – y Coordinate [microns]
- **rot** – Rotation in degrees
- **length** (float[66]) – Port length [microns]

length

Alias for field number 3

rot

Alias for field number 2

x

Alias for field number 0

y

Alias for field number 1

photonics.**isnamedtupleinstance**(*x*)

Test if something is a named tuple This allows to test if *x* is a port (PortCreation object) or just a list of instance descriptions

5.2.2 Submodules

photonics.dataprep module

photonics.dataprep.**add**(*layout*, *cell*, *slayers*, *dlayers*, *ex_amount*, *layers*, *out_cell=None*)

Combines all slayers' shapes into a region and merges this region with each of dlayers' regions.

Parameters

- **layout** – the layout on which the cells are located
- **cell** – the cell from which to copy the layers (source shapes)
- **slayers** – the layers to copy
- **dlayers** – the layers where to copy to
- **ex_amount** – the amount added around the source shapes
- **layers** – the layermapping
- **out_cell** – the cell where to put the shapes. If not specified, the input cell will be used.

photonics.dataprep.**dataprep**(*in_cell*, *layout*, *out_cell=None*, *config=None*, *layers_org=None*)

Dataprep that creates excludes layers etc. with boolean operation on input layers that will be added/substracted to outputlayers.

Parameters

[63] https://docs.python.org/3/library/stdtypes.html#tuple
[64] https://docs.python.org/3/library/functions.html#int
[65] https://docs.python.org/3/library/functions.html#int
[66] https://docs.python.org/3/library/functions.html#float

- **in_cell** – the cell from which to take shapes
- **layout** – the layout on which we perform the operations (most likely self.layout)
- **out_cell** – the output cell. if not specified take the input cell
- **config** – the config file. This file specifies the boolean operations (self.dataprepconfig)
- **layers_org** – the original layermap we use (most likely self.layermap)

photonics.dataprep.**file_len**(*fname*)

Returns the number of lines in the file fname

photonics.dataprep.**sub**(*layout*, *cell*, *slayers*, *dlayers*, *ex_amount*, *layers*, *out_cell=None*)

Analogous to add()

Instead of perfoming a combination with the destination layers, this function will substract the input region.

photonics.layermaps module

photonics.layermaps.**load**(*filename*)

Simple routine to read a .layermap file into a dictionary

Parameters **filename** (str[67]) – Filename with path

Returns Dictionary of dictionaries in the form of {layer: {purpose1:(layer_number,purpose_number), purpose2:(layer_number1,purpose_number2)},layer2: {. . . } }

Return type dict[68]

Examples

```
>>> import PhotPCells.layermaps as lm
>>> lm.load(os.path.expanduser('~/.klayout/salt/zccmos/FreePDK45_tech/tech/
↪FreePDK45.layermap'))
{'pwell': {'blockage': ('109', '1'), 'drawing': ('109', '0')}, ... }
```

[67] https://docs.python.org/3/library/stdtypes.html#str
[68] https://docs.python.org/3/library/stdtypes.html#dict

CHAPTER

SIX

EXAMPLE: CREATE SAMPLE LIBRARY

In this chapter we will create an example library consisting of an MMI built with a box and linear tapers. The finished file can be found in `MMI_Example.lym`. This file can be copied into the KLayout pymacros folder (`~/.klayout/pymacros/`) and executed. The chapter is an in-depth explanation of the example.

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.

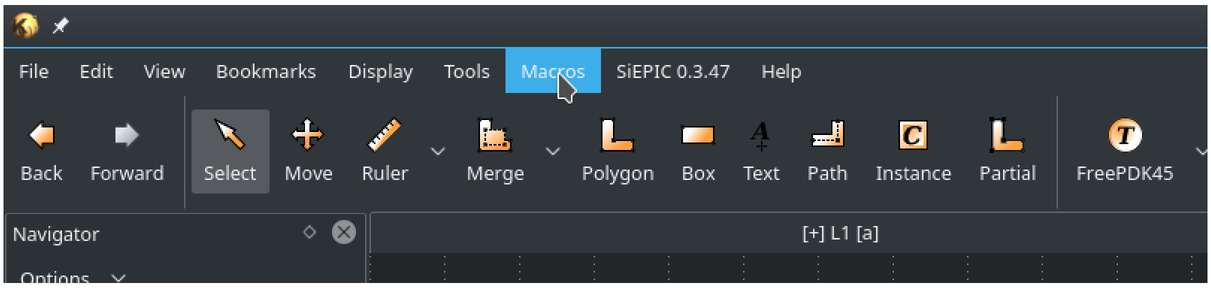


Fig. 6.1: Open the IDE through *Macros→MacroDevelopment*

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.

```python
import pya
import math
from photonics import PhotDevice, PortCreation
import photonics.layermaps as lm
import numpy as np
import os
```

After the imports we will create a helper class. The class 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.

```python
class FreePDK45Example(PhotDevice):
    """Class that provides technology specific data. Currently the backend needs 3 things to be supplied by the technology of the PCells.
```

(continues on next page)

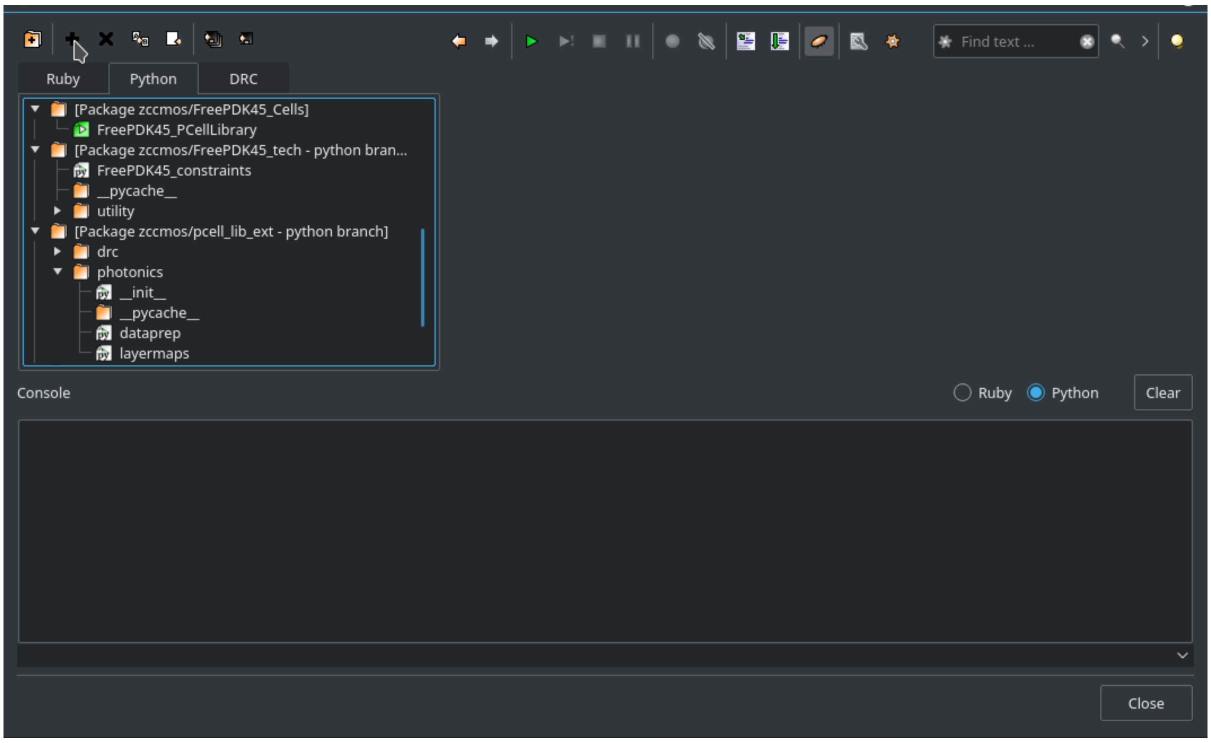
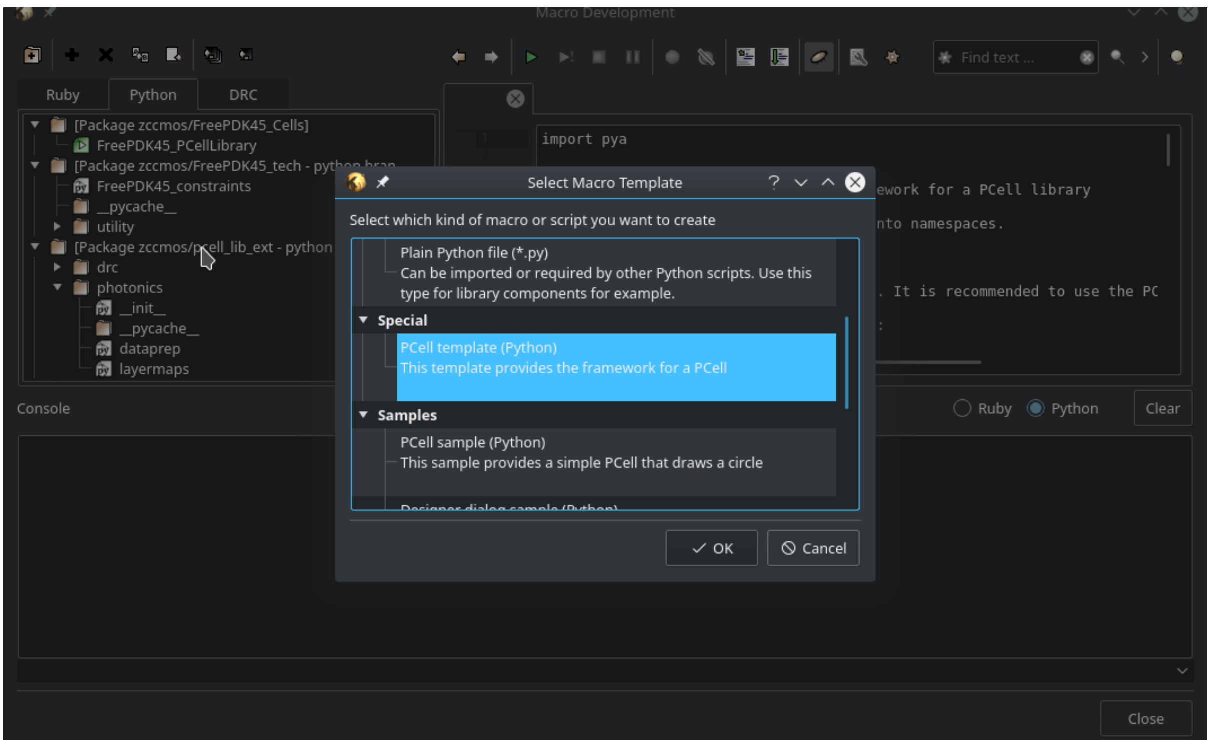


Fig. 6.2: Add a new PCell template from the Context



Select Macro Template

Select which kind of macro or script you want to create

Plain Python file (*.py)
Can be imported or required by other Python scripts. Use this type for library components for example.

Special

PCell template (Python)
This template provides the framework for a PCell

Samples

PCell sample (Python)
This sample provides a simple PCell that draws a circle

✓ OK
Cancel

Fig. 6.3: Choose PCell template (Python)

(continued from previous page)

```
    As these are independent of specific PCells and parameters this should not give any difficulty␣
↪with the requirement of Klayout to have the Classes stateless.

    The layermap was created from a forum suggestion
        (`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.

        layermap:       A dictionary containing layers with available purposes, which provide a␣
↪layer/purpose. This is loaded from a .layermap file.
                        Example of this FreePDK45:
                            {'active': {'blockage': (1, 1), 'drawing': (1, 0)}, 'pwell': {'blockage
↪': (2, 1), 'drawing': (2, 0)},...}

        :ivar dataprep_config: Filepath to a text file containing rules for dataprep. This file␣
↪contains rules for the dataprep.
                        Copied from the example dataprep.txt:
                        File Format:
                        File defining operations for dataprep
                        Format:
                        <operation> <source layers> <destination layers> <sizing amount in microns>
                        Operations supported: add,sub
                            * add: Create a region from all shapes of the source layers and combine␣
↪this region with each destion layer region separately
                            * sub: Same as add but don't build combination but cross-section instead
                        Sizing amount uses the klayout sizing operation to size the regions of the␣
↪source layers
                        During dataprep the regions are merged, meaning overlapping polygons will␣
↪become one Polygon
                        source/destination layers are separated by commas if there are multiple
                        Each argument is separated by white spaces. How many should not matter as␣
↪they will be parsed by a python str.split() operation which should be able to handle any white␣
↪space amount.
                        If the first word of a line is not a supported operation the line will be␣
↪ignored
                        The lines will be executed in order meaning and add sub operations on␣
↪layers will be different than a first sub and then add
                        :Examples:
                            active.blockage,poly.blockage,metal1.blockage,metal2.blockage,metal3.
↪blockage,metal4.blockage,metal5.blockage,nwell.drawing,nimplant.drawing 2.0

        :ivar clean_rules:     list containing the layer/purpose numbers and the minWidth/
↪minSpacing rules for the layer/purpose pair in microns
                        :Examples:
                            [[(1, 0), 0.097, 0.077],[(2, 0), 0.23, 0.189],[(3, 0), 0.169, 0.196],
↪[(5, 0), 0.044, 0.052], ...]
        """

    # Define the metals & via names. They will be used in some PCells (Electrodes and ViaStack)
    metal_names = ['metal' + str(i) for i in range(1, 11)]
    via_names = ['via' + str(i) for i in range(1, 10)]

    def __init__(self):
        PhotDevice.__init__(self)

        techpath = pya.Technology.technology_by_name('FreePDK45').base_path()

        filename = techpath + '/FreePDK45.tf'

        # Check if techfile is correctly imported and located

        isfile = os.path.isfile(filename)
```

(continues on next page)

(continued from previous page)

```
        if not isfile:
            import sys
            msg = pya.QMessageBox(pya.Application.instance().main_window())
            msg.text = 'Please import the techfile of the technology to {} before using the module and reopen KLayout'.format(filename)
            msg.windowTitle = 'ImportError'
            msg.exec_()

        tech = con.load_from_tech(filename)

        # Get the layermap file and load it.
        # CAREFUL: Will be used for dataprep and others
        self.layermap = lm.load(techpath + '/FreePDK45.layermap')

        # This variable will be imported by the dataprep algorithm
        # CAREFUL: Will be imported for dataprep
        self.dataprep_config = techpath + '/dataprep.txt'

        # Rules for the cleaner in the form [[(layer1,purpose1),violation_width1,violation_space1],[(layer2,purpose2),violation_width2,violation_space2],...]
        ### CAREFUL: This variable will be imported for the cleaning.
        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],
                            [(4, 0), 0.046, 0.052], [(9, 0), 0.044, 0.062], [(11, 0), 0.076, 0.077], [(13, 0), 0.073, 0.089],
                            [(15, 0), 0.067, 0.063], [(17, 0), 0.143, 0.137], [(19, 0), 0.158, 0.14], [(21, 0), 0.145, 0.123],
                            [(23, 0), 0.514, 0.535], [(25, 0), 0.369, 0.311], [(27, 0), 0.908, 0.843], [(29, 0), 0.347, 0.771],
                            [(1, 1), 1.247, 1.254], [(2, 1), 0.976, 0.905], [(3, 1), 1.165, 1.304], [(5, 1), 1.073, 0.958],
                            [(4, 1), 1.058, 0.885], [(9, 1), 0.892, 0.825], [(11, 1), 1.003, 0.682], [(13, 1), 0.983, 0.73],
                            [(15, 1), 1.086, 0.993], [(17, 1), 1.12, 0.812], [(19, 1), 0.941, 0.765], [(21, 1), 0.942, 0.889],
                            [(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.

```
class ExMMIBody(FreePDK45Example):
    """MMI Body. Since this should be a 2x2 MMI it will have 4 ports
    """

    def __init__(self):
        FreePDK45Example.__init__(self)
        self.add_layer('lay',"active.drawing")
        # 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
        params = dict(length=15.,
                      width=5.,
                      port_offset=1.5,
                      port_width =1.0
                      )
        # Register the parameters
        self.add_params(params)

    def create_param_inst(self):
```

(continues on next page)

(continued from previous page)

```
        # Create Ports here
        ports = [PortCreation(-self.length/2, self.port_offset, 180, self.port_width),
                PortCreation(-self.length/2, -self.port_offset, 180, self.port_width),
                PortCreation(self.length/2, -self.port_offset, 0, self.port_width),
                PortCreation(self.length/2, self.port_offset, 0, self.port_width)]
        return ports

    def shapes(self):
        #Create the Rectangle
        self.create_polygon([[-self.length/2,-self.width/2],[self.length/2,-self.width/2],[self.
↪length/2,self.width/2],[-self.length/2,self.width/2]],self.lay)

class ExLinTaper(FreePDK45Example):

    def __init__(self):
        FreePDK45Example.__init__(self)
        self.add_layer('lay',"active.drawing")
        params = dict(width_0 = .5,
                      width_1 = 1.0,
                      length = 2.0,
                      )
        self.add_params(params)

    def create_param_inst(self):
        # Create left and right port
        port_0 = PortCreation(-self.length/2,0,180,self.width_0)
        port_1 = PortCreation(self.length/2,0,0,self.width_1)
        return port_0,port_1

    def shapes(self):
        # Create taper polygon
        self.create_polygon([[-self.length/2,-self.width_0/2],
                             [-self.length/2,self.width_0/2],
                             [self.length/2,self.width_1/2],
                             [self.length/2,-self.width_1/2],],
                             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.

```
class Ex2x2MMI(FreePDK45Example):
    """The MMI-cell class.
    This class instantiates a body with 4 tapers and attaches the tapers to the the body.
    """

    def __init__(self):
        FreePDK45Example.__init__(self)
        self.add_layer('lay','active.drawing')
        params = dict(wg_width=.5,
                      length=15.0,
                      taper_width=1.0,
                      taper_length=2.0,
                      width=4.0,
                      taper_offset=1.0,
                      )
        self.add_params(params)
```

(continues on next page)

(continued from previous page)

```python

    def create_param_inst(self):
        # Library we load the sub-cells from
        lib = "FreePDK45_Photonic_FirstExample"
        bodyname = "MMIBody"
        tapername = "LinearTaper"

        # Parameters used for the 4-port body
        body_params = dict(lib = lib,
                           cellname = bodyname,
                           width=self.width,
                           length=self.length,
                           port_offset=self.taper_offset,
                           port_width =self.taper_width,
                           )
        # Parameters for tapers
        taper_params = dict(lib = lib,
                           cellname = tapername,
                           width_0=self.wg_width,
                           width_1=self.taper_width,
                           length=self.taper_length,
                           )
        # Create constructors for tapers and body
        tapers = self.add_pcell_variant(taper_params,number=4)
        body = self.add_pcell_variant(body_params)

        # Connect the ports
        for i in range(4):
            self.connect_port_to_port(body.port(i),tapers[i].port(1))

        # Return constructors
        return tapers,body
```

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

```python
class FreePDK45_ExampleLib(pya.Library):
    def __init__(self):
        # Set the description
        self.description = "FirstExample"
        self.technology = "FreePDK45"
        # Create the PCell declarations
        self.layout().register_pcell("2x2MMI",Ex2x2MMI())
        self.layout().register_pcell("MMIBody",ExMMIBody())
        self.layout().register_pcell("LinearTaper",ExLinTaper())

        self.register("FreePDK45_Photonic_FirstExample")
```

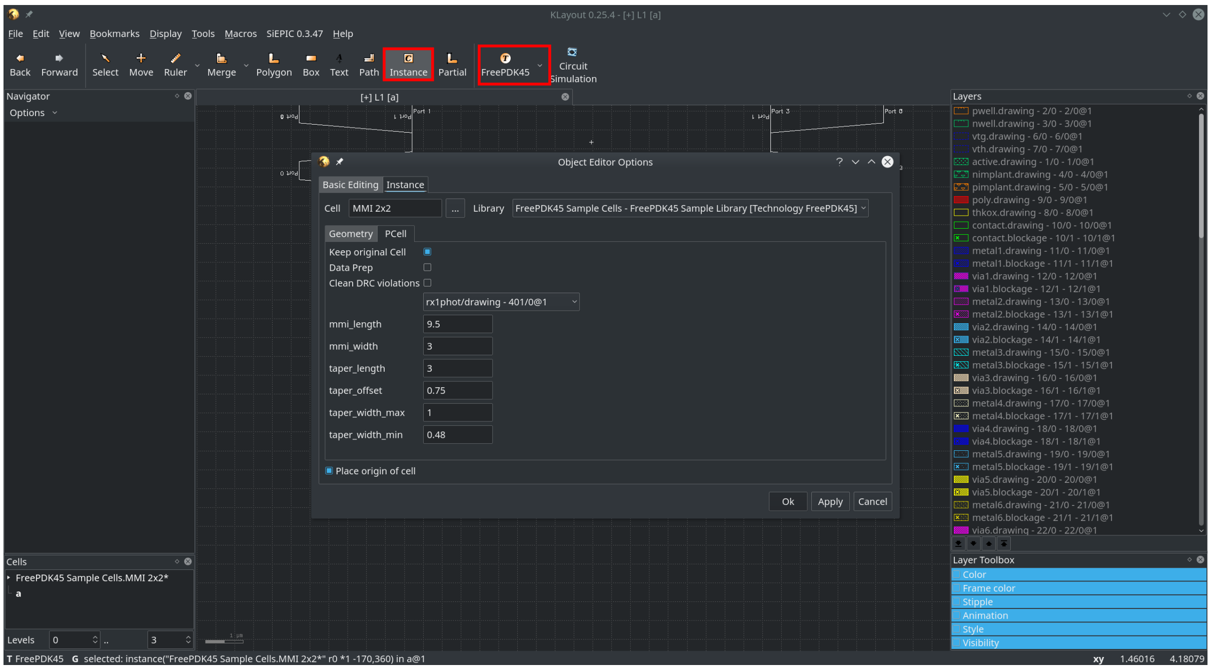
And finally make KLayout compile the PCell-Library and add it to the PCell-Libraries:

```python
# Instantiate and register the library
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.

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


KLayout 0.25.4 - [+] L1 [a]
File Edit View Bookmarks Display Tools Macros SiEPIC 0.3.47 Help
Back Forward Select Move Ruler Merge Polygon Box Text Path Instance Partial FreePDK45 Circuit Simulation
Navigator
Options
[+] L1 [a]
Port 1
Port 3
Port 0
Object Editor Options
Basic Editing
Instance
Geometry
PCell
Ok
Apply
Cancel
Layers
pwell.drawing - 2/0 - 2/0@1
nwell.drawing - 3/0 - 3/0@1
vtg.drawing - 6/0 - 6/0@1
vth.drawing - 7/0 - 7/0@1
active.drawing - 1/0 - 1/0@1
nimplant.drawing - 4/0 - 4/0@1
pimplant.drawing - 5/0 - 5/0@1
poly.drawing - 9/0 - 9/0@1
thkox.drawing - 8/0 - 8/0@1
contact.drawing - 10/0 - 10/0@1
contact.blockage - 10/1 - 10/1@1
metal1.drawing - 11/0 - 11/0@1
metal1.blockage - 11/1 - 11/1@1
via1.drawing - 12/0 - 12/0@1
via1.blockage - 12/1 - 12/1@1
metal2.drawing - 13/0 - 13/0@1
metal2.blockage - 13/1 - 13/1@1
via2.drawing - 14/0 - 14/0@1
via2.blockage - 14/1 - 14/1@1
metal3.drawing - 15/0 - 15/0@1
metal3.blockage - 15/1 - 15/1@1
via3.drawing - 16/0 - 16/0@1
via3.blockage - 16/1 - 16/1@1
metal4.drawing - 17/0 - 17/0@1
metal4.blockage - 17/1 - 17/1@1
via4.drawing - 18/0 - 18/0@1
via4.blockage - 18/1 - 18/1@1
metal5.drawing - 19/0 - 19/0@1
metal5.blockage - 19/1 - 19/1@1
via5.drawing - 20/0 - 20/0@1
via5.blockage - 20/1 - 20/1@1
metal6.drawing - 21/0 - 21/0@1
metal6.blockage - 21/1 - 21/1@1
via6.drawing - 22/0 - 22/0@1
Layer Toolbox
Color
Frame color
Stipple
Animation
Style
Visibility
Cells
FreePDK45 Sample Cells.MMI 2x2*
a
Levels

Cell: MMI 2x2
Library: FreePDK45 Sample Cells - FreePDK45 Sample Library [Technology FreePDK45]

Keep original Cell ☑
Data Prep ☐
Clean DRC violations ☐
rx1phot/drawing - 401/0@1

mmi_length	9.5
mmi_width	3
taper_length	3
taper_offset	0.75
taper_width_max	1
taper_width_min	0.48

☑ Place origin of cell

Levels 0 .. 3

T FreePDK45 G selected: instance("FreePDK45 Sample Cells.MMI 2x2*" r0 *1 -170,360) in a@1

xy 1.46016 4.18079

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

PCell specific and should be the ones defined in the `__init__(self)` function of the cell definition.

CHAPTER

SEVEN

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.

Glossary:

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX