KLayout

Designing a PCell from scratch

Matthias Köfferlein, www.klayout.org

A bit of Theory

PCell Basics

A PCell is provided by

- Implementing a class derived from “klayout.db.PCellDeclarationHelper”
- Implementing the mandatory interface methods
- Creating a Library
- Registering the PCell class inside that library
- Instantiating and registering that Library

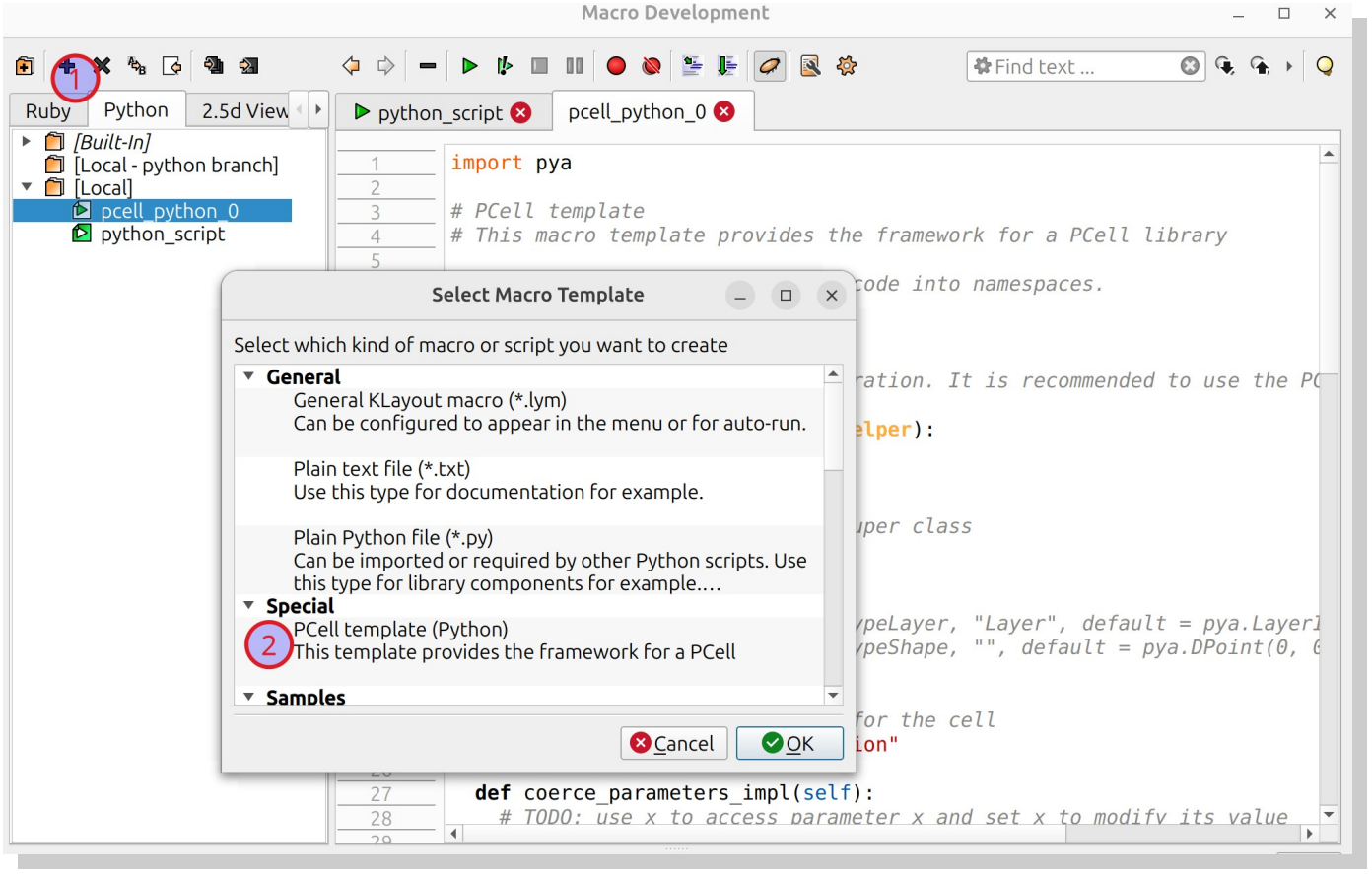
After this you can start using the PCell

Resources

- Basic explanation (Ruby)
  https://www.klayout.de/doc-qt5/programming/ruby_pcells.html
- PCellDeclarationHelper (Ruby)
  https://www.klayout.de/doc-qt5/code/class_PCellDeclarationHelper.html

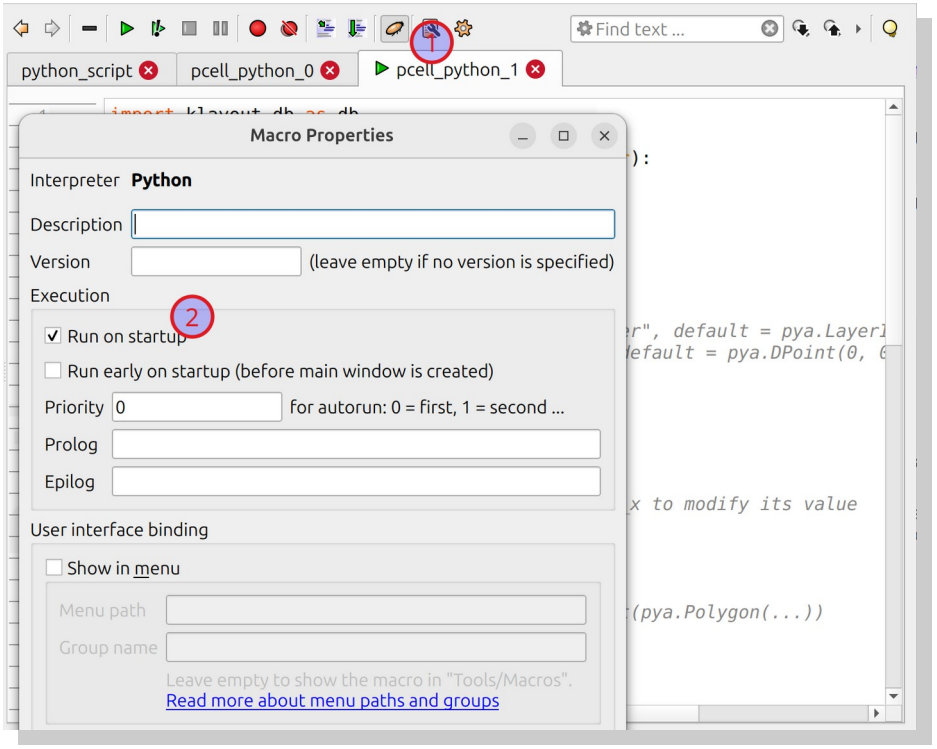
  BUG: Python version documentation is not visible because of name clash
- Your favorite AI

Start from the template



Auto-run on startup

- "Run on startup" means KLayout will execute the script automatically when started



Brief anatomy of the PCell interface

- Constructor
  - Declare parameters
- display_text_impl
  - Delivers the text that is shown for the PCell
- coerce_parameters_impl
  - Called on parameter changes, enforce boundary conditions
- produce_impl
  - Generate the layouts for the PCell
  - Can instantiate other cells if required

Things to know

- The PCell declaration object is stateless
  - You cannot store information about the PCells there
  - Use hidden parameters
- The PCell code is executed on the context of the library
  - You don’t have access to anything from the client layout
- Don’t assume a certain sequence of method calls

Practical Implementation

Notes

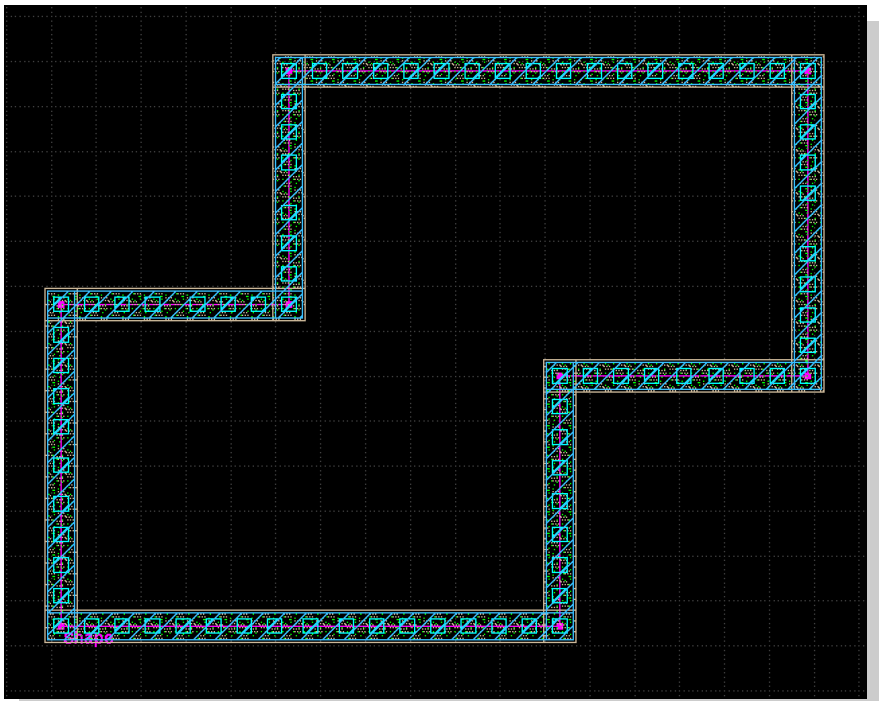
- This is an interactive lecture. You can follow it yourself or use the material from here:

  https://github.com/klayoutmatthias/heichips2026_pypcell

Our Example

A freestyle guardring

PDK: rectangular only

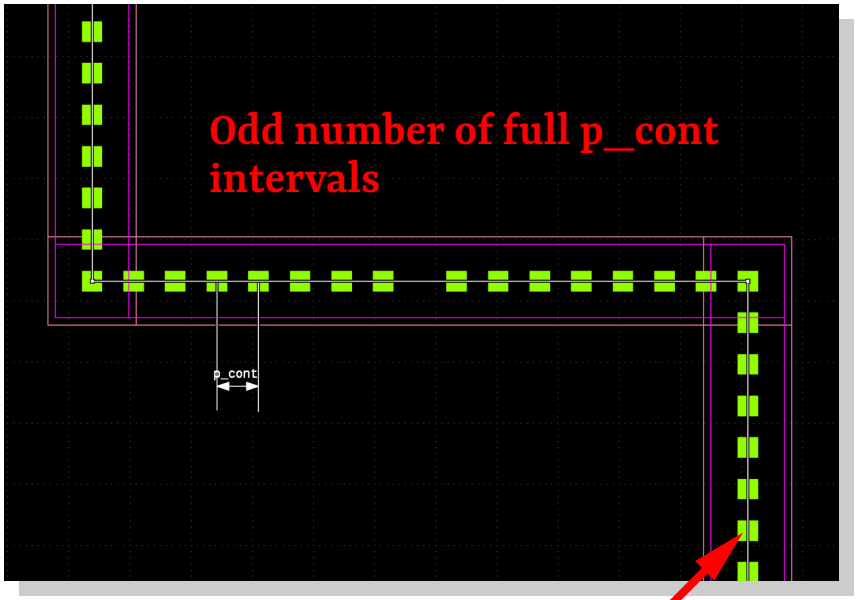


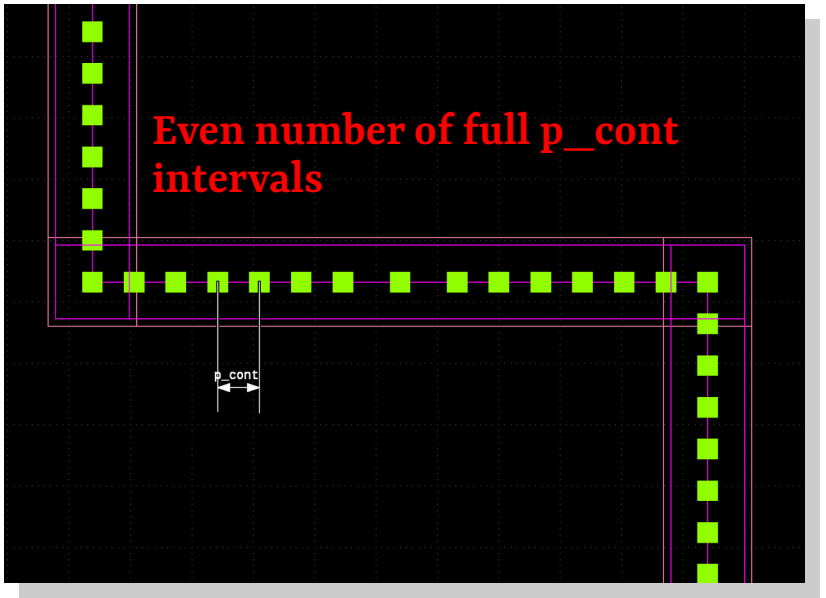
New: polygon shape

Parameters

- Bottom Layer(s)
  - Active (Diffusion) drawing layer + width (`li_active, w_active`)
  - Implant (optional) drawing layer + width (`li_impl, w_impl`)
  - Well (optional) drawing layer + width (`li_well, w_well`)
- Contacts
  - Drawing layer (`li_cont`)
  - Dimension (`w_cont`)
  - Pitch (`p_cont`)
- First metal
  - Drawing layer (`li_metal`)
  - Width (`w_metal`)
- Geometry of the freestyle guardring (`shape`)

Contact distribution





Shape defines center line of contact chain + metal + diffusion

Actual pitch < 2*p_cont !

Conclusion

- This simple example demonstrates
  - Step-by-step implementation
  - Required boiler plate code
  - Use of the Python API to implement PCells
  - Domain specific implementation decisions
  - Use of “guiding shapes”
  - Debugging

Thank you for Participating