KLayout-PEX Documentation

Martin Köhler

2025-01-22

Table of contents

1 Introduction

KLayout is an open source VLSI layout viewer and editor.

KLayout-PEX (short `KPEX`) is a parasitic extraction (PEX) tool, well integrated with KLayout by using its API.

There are multiple PEX engines supported, currently:

- FasterCap integration (field solver engine)
- MAGIC integration (wrapper calling `magic`)
- Analytical 2.5D engine (parasitic concepts and formulas of MAGIC, implemented using KLayout methods)

Tip

KPEX *tool* source code itself is made publicly available on GitHub (follow this link) and shared under the GPL-3.0 license.
KPEX *documentation* source code is made publicly available on GitHub (follow this link) and shared under the Apache-2.0 license.
Please feel free to create issues and/or submit pull requests on GitHub to fix errors and omissions!
The production of the tool and this document would be impossible without these (and many more) great open-source software products: `KLayout`, `FasterCap`, `MAGIC`, `protobuf`, `Quarto`, `Python`, `ngspice`, `Numpy`, `Scipy`, `Matplotlib`, `Git`, `Docker`, `Ubuntu`, `Linux`…

Caution

Currently, `KPEX` is developed as a Python prototype, using the KLayout Python API. This allows for a faster development cycle during the current prototyping phase. Eventually, critical parts will be re-implemented (in C++, and parallelized), to improve performance. As we're already using the KLayout API (which is pretty similar between Python, Ruby and C++), this will be relatively straight-forward.

1.1 Installation

Generally, `KPEX` is deployed using PyPi (Python Package Index), install via:

```
pip3 install --upgrade klayout-pex

kpex --version    # check the installed version
kpex --help       # this will help with command line arguments
```

As for the dependencies, there are multiple options available.

1.1.1 Option 1: Using `IIC-OSIC-TOOLS` Docker Image

We provide a comprehensive, low entry barrier Docker image that comes pre-installed with most relevant open source ASIC tools, as well as the open PDKs. This is a pre-compiled Docker image which allows to do circuit design on a virtual machine on virtually any type of computing equipment (personal PC, Raspberry Pi, cloud server) on various operating systems (Windows, macOS, Linux).

For further information please look at the Docker Hub page and for detailed instructions at the IIC-OSIC-TOOLS GitHub page.

⚠ Linux

In this document, we assume that users have a basic knowledge of Linux and how to operate it using the terminal (shell). If you are not yet familiar with Linux (which is basically a must when doing integrated circuit design as many tools are only available on Linux), then please check out a Linux introductory course or tutorial online, there are many resources available.
A summary of important Linux shell commands is provided in IIC-JKU Linux Cheat-sheet.

1.1.2 Option 2: Standalone Installation

- KLayout layout tool:
  - is mandatory for all engines (besides the MAGIC-wrapper)
  - get the latest pre-built package version
  - or follow the build instructions
- FasterCap engine:
  - optional, required to run the FasterCap engine
  - either compile your own version from the GitHub repository
  - or use precompiled versions available at https://github.com/martinjankoehler/FasterCap/releases
- MAGIC-wrapper engine:
  - optional, required to run the MAGIC-wrapper engine
  - Follow the installation instructions at the GitHub repository
- Skywater sky130A PDK:
  - optional, for now, KPEX technology specific files are deployed within the `klayout-pex` Python package
  - `pip3 install --upgrade volare` (install PDK package manager)
  - `volare ls-remote` (retrieve available PDK releases)
    - for example `PRE-RELEASE 0c1df35fd535299ea1ef74d1e9e15dedaeb34c32 (2024.12.11)`)
  - `volare enable 0c1df35fd535299ea1ef74d1e9e15dedaeb34c32` (install a PDK version)
  - PDK files now have been installed under `$HOME/.volare/sky130A`
- IHP SG13G2 PDK:
  - optional, for now, KPEX technology specific files are deployed within the `klayout-pex` Python package
  - `git clone https://github.com/IHP-GmbH/IHP-Open-PDK` (install PDK package manager)

1.1.3 Useful tools: `meshlab`

For previewing generated 3D geometries, representing the input to `FasterCap`, we recommend installing MeshLab.

`MeshLab` can open the `STL`-files generated in the `output` directory under `output/<design>/Geometries/*.stl`

2 First Steps

- The command line tool `kpex` is used to trigger the parasitic extraction flow from the terminal.
- Get help calling `kpex --help`.

2.1 Examples

Example layouts are included in the `testdata/designs` subdirectory of the KLayout-PEX source code:

```
git clone https://github.com/martinjankoehler/klayout-pex.git

# for sky130A
find testdata/designs/sky130A -name "*.gds.gz"

# for IHP SG13G2
find testdata/designs/ihp_sg13g2 -name "*.gds.gz"
```

2.1.1 Running FasterCap

Preconditions:

- `klayout-pex` was installed, see Section 1.1
- `FasterCap` was installed, see Section 1.1

> **Note**
>
> Normally, devices with SPICE simulation models (e.g. like MOM-capacitors[1] in the sky130A PDK) are ignored ("blackboxed") during parasitic extraction.
> `kpex` has an option `--blacklist n` to allow extraction of those devices (whiteboxing), which can be useful during development (during the prototype phase, whiteboxing is actually the default setting, so please use `--blacklist y` to explicitly configure blackboxing).

[1] Metal-Oxide-Metal capacitors

Let's try the following:

```
kpex --pdk sky130A --blackbox n --gds testdata/designs/sky130A/cap_vpp_04p4x04p6_l1m1m2_no
```

Note

This will report an error that we have not activated one or more engines, and list the available engines:

Argument	Description
`--fastercap y`	Run kpex/FasterCap engine
`--2.5D y`	Run kpex/2.5D engine
`--magic y`	Run MAGIC engine

Now, to run the FasterCap engine (might take a couple of minutes):

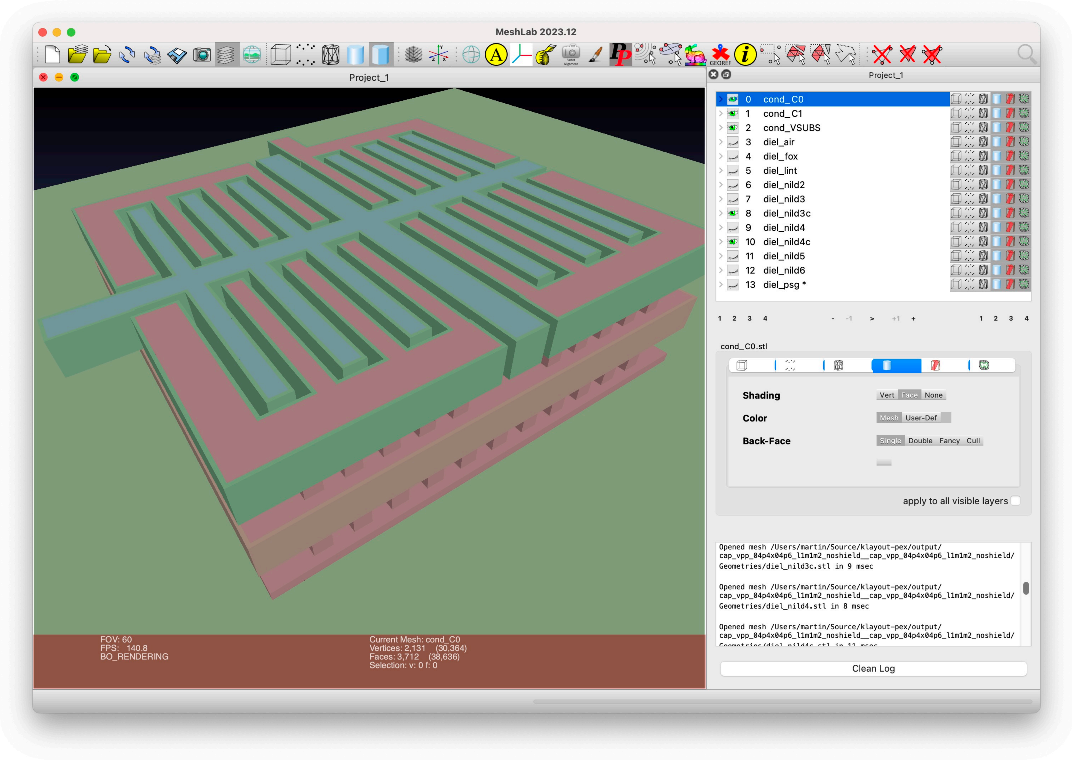
```
kpex --pdk sky130A --blackbox n --fastercap y --gds testdata/designs/sky130A/cap_vpp_04p4x
```

The default output directory is `output`, in there we see a directory `Geometries`, containing STL-files. Those STL files provide a preview of the FasterCap input files. Use MeshLab (see Section 1.1.3) to open and preview those files:

```
ls -d output/cap_vpp_04p4x04p6_l1m1m2_noshield__cap_vpp_04p4x04p6_l1m1m2_noshield/Geometri
```

Tip

- Open the STL files in MeshLab
- Use the eye buttons to hide and show each file/mesh
- Use the align tool ("A" in the toolbar) to assign different colors
- Start by showing only on the conductors (files named `cond_*.stl`)
- Then try showing different dielectrics (files named `diel_*.stl`), to see how they surround the conductors.



In the log file, we see the output of FasterCap including the Maxwell capacitance matrix:

```
Capacitance matrix is:
Dimension 3 x 3
g1_VSUBS  5.2959e-09 -4.46971e-10 -1.67304e-09
g2_C1  -5.56106e-10 1.5383e-08 -1.47213e-08
g3_C0  -1.69838e-09 -1.48846e-08 1.64502e-08
```

KPEX interprets this matrix and logs a CSV netlist, which can be pasted into a spreadsheet application:

```
Device;Net1;Net2;Capacitance [fF]
Cext_0_1;VSUBS;C1;0.5
Cext_0_2;VSUBS;C0;1.69
Cext_1_2;C1;C0;14.8
Cext_1_1;C1;VSUBS;0.08
```

In addition, a SPICE netlist is generated.

3 Supporting new PDKs

For every supported PDK, a `KPEX` technology definition is required, as well as customized PEX-“LVS” scripts.

3.1 Technology Definition Files

The `KPEX` technology definition format uses Google Protocol Buffers, so there is:

- formal schema files, defining the structure and data types involved
  - `protos/tech.proto`: main schema / entry point, includes the others
  - `protos/process_stack.proto`: describes details of the process stack, such as dielectrics and heights of layers
  - `protos/process_parasitics.proto`: parasitic tables, used to parametrize the 2.5D engine
- multiple concrete instantiations, that adhere to this schema (called *messages* in the `protobuf` lingo)
  - in the form of JSON files
  - Skywater 130A: `klayout_pex_protobuf/sky130A_tech.pb.json`
  - IHP SG13G2: `klayout_pex_protobuf/ihp_sg13g2_tech.pb.json`

> **Note**
>
> The built-in JSON tech files are programmatically generated during the build process[2]. Therefore they not part of the repository source code, but of course part of the deployed Python wheels. To review those, look into your Python `site-packages`[3]`/klayout_pex_protobuf`.

3.2 Customized PEX-“LVS” scripts

`KLayout` has built-in support for Layout-Versus-Schematic (LVS) scripts, based on its Ruby API. Customized “LVS” scripts are (“ab”)used in `KPEX`, not with the intent of comparing Layout-Versus-Schematic, but rather to extract the connectivity/net information for all polygons across multiple layers.

These customized “LVS” scripts are stored in:

- Skywater sky130A: `pdk/sky130A/libs.tech/kpex/sky130.lvs`
- IHP SG13G2: `pdk/ihp_sg13gs/libs.tech/kpex/sg13g2.lvs`

What’s specific about this customization:

[3] C++ generator scripts the built-in tech files are located in `cxx/gen_tech_pb/pdk/*.cpp`.
[3] To find the `site-packages` directory for the `klayout-pex` package, call `pip3 show klayout-pex`.

- Layers names must be assigned, using KLayout's `(name(layer, name))` function
- MOM[4] capacitors, MIM[5] capacitors and resistors should be extracted to separate layers, to enable blackboxing / whiteboxing.

The layer names in the script must correspond with the names configured in the tech JSON file.

[4] Metal-Oxide-Metal capacitors
[5] Metal-Insulator-Metal capacitors