KLayout-PEX Documentation

Martin Köhler

2026-11-16

Table of contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Acknowledgements . . . 1
1.3 About KLayout-PEX . . . 1
1.4 Status . . . 2
1.5 Installation . . . 3
1.5.1 Option 1: Using `IIC-OSIC-TOOLS` Docker Image . . . 3
1.5.2 Option 2: Standalone Installation . . . 4
1.5.3 Checking Tool Versions . . . 4
1.5.4 Useful tools: `meshlab` . . . 5

2 First Steps **6**
2.1 Example Layouts . . . 6
2.2 Choosing Device Models . . . 6
2.3 Running `KPEX/2.5D` . . . 6
2.4 Generating PEX25D . . . 7
2.5 Running the `KPEX/FasterCap` engine . . . 7
2.5.1 Extracting Full Capacitor Geometry . . . 8
2.6 Running the `KPEX/MAGIC` engine . . . 9

3 Choosing and describing PDKs **11**
3.1 Choosing the process variant . . . 11
3.2 Forming nets and identifying layers . . . 11
3.3 Keeping technology data together . . . 13
3.4 Describing the process stack . . . 13
3.4.1 Planar passivation over `gf180mcuD` . . . 16
3.5 Interpreting parasitic coefficients . . . 17
3.5.1 Direction and physical consistency . . . 17
3.6 Mapping extracted devices to circuit models . . . 18
3.6.1 Independent capacitance references . . . 18
3.7 Updating technology data . . . 19
3.8 Inspecting the generated definitions . . . 20
3.9 Selecting Technology Corners . . . 20
3.9.1 Selecting Declared Scenarios . . . 20
3.9.2 Keeping the Sources Independent . . . 22
3.9.3 Inspecting Catalog Data . . . 23
3.9.4 Tracing Sources and Assumptions . . . 23
3.9.5 Maintaining Data History . . . 24
3.9.6 Reading Upstream Corner Names . . . 25

3.9.7 Using Low/High Capacitance Components . . . 26
3.9.8 Varying GatPoly Sheet Resistance . . . 29
3.9.9 Varying CMOS5L Resistance at Fixed Geometry . . . 31
3.9.10 Varying `sky130A` Resistance . . . 32
3.9.11 Varying `gf180mcuD` Resistance . . . 35
3.9.12 Matching Physical Geometry to Analytical Tables . . . 36
3.9.13 Selecting Device-Capacitance References . . . 44
3.9.14 Recording What Was Used . . . 47
3.9.15 Limits . . . 47

4 `KPEX/FasterCap` Engine **48**
4.1 3D Input Geometries . . . 49
4.2 Example: MOM Capacitor . . . 50
4.3 Output Maxwell Capacitance Matrix . . . 50

5 `KPEX/MAGIC` Engine **54**
5.1 MAGIC database units . . . 54
5.2 Types of Parasitic Capacitances . . . 54
5.3 Substrate Capacitance . . . 54
5.4 Sidewall Capacitance . . . 55
5.5 Overlap Capacitance . . . 55
5.6 Fringe Capacitance . . . 56
5.7 Shielding Effects . . . 58
5.8 Parasitic Resistance . . . 58
5.8.1 Wire resistance . . . 61
5.8.2 Via resistance . . . 61

6 `KPEX/2.5D` Engine **63**
6.1 Choosing the Level of Detail . . . 64
6.2 Capacitance of Sloped and Holed Polygons . . . 64
6.2.1 Looking Along One Edge . . . 64
6.2.2 Integrating a Changing Sidewall Gap . . . 65
6.2.3 Integrating Fringe Across a Target . . . 66
6.2.4 Subtracting Shields and Keeping Holes . . . 67
6.2.5 Preserving the Profile When Geometry Is Cut . . . 68
6.2.6 Inspecting Sloped Wires . . . 68
6.2.7 Halo, Grid and Model Limits . . . 70
6.3 Excluding Parts of the Layout . . . 70
6.3.1 What the PDKs Mark . . . 71
6.3.2 Why the Geometry Stays . . . 72
6.3.3 What Is Left Out . . . 73
6.3.4 The Variants . . . 74
6.3.5 Examples . . . 79
6.3.6 Markers of a Layer, of a Pair and of the User . . . 82
6.3.7 Cutting Out a Cell . . . 82
6.3.8 Leaving Out a Net . . . 85
6.3.9 Field Solvers . . . 85
6.3.10 Use Cases . . . 86
6.3.11 Limits . . . 92
6.4 Connecting the Extracted Netlist . . . 93
6.4.1 Choosing Which Labels Become Ports . . . 93
6.4.2 Keeping Positional Calls Correct . . . 93
6.4.3 Following Schematic Device Names . . . 94
6.4.4 Keeping Geometry Units Separate from Electrical Values . . . 95

6.4.5 Including Both Files in a VACASK Testbench . . . 95

7 Choosing Capacitance Coupling 97
7.1 Choosing the Extraction Mode . . . 97
7.2 Eliminating Floating Conductors First . . . 98
7.2.1 Reproducing the Floating Example . . . 99
7.2.2 Extending the Elimination to a Network . . . 99
7.3 Replacing Coupling by Grounded Loads . . . 100
7.4 Grounding an Existing Netlist . . . 100
7.4.1 Distributed Wires and Intentional Capacitance . . . 100
7.4.2 Selecting Parasitics and the Reference . . . 102
7.4.3 Identifying Disconnected Pieces of One Conductor . . . 102
7.5 Keeping the Approximation Separate from Other Checks . . . 102
7.6 Limits . . . 103

8 Checking Extracted Netlists 103
8.1 Preserving Layout Connectivity . . . 103
8.1.1 Choosing Whether a Structural Problem Stops the Run . . . 104
8.2 Comparing with the Schematic . . . 104
8.2.1 Comparing Device Quantities . . . 104
8.2.2 Comparing Pins and Ports . . . 105
8.3 Checking the Response after Reduction . . . 105
8.3.1 From Branches to Terminal Admittance . . . 105
8.3.2 Comparing the Two Matrices . . . 106
8.3.3 Choosing Frequencies . . . 106
8.3.4 Example: Losing the Series Resistance of a Capacitor . . . 107
8.3.5 Limits of the Admittance Check . . . 108
8.4 Comparing Device Capacitance with a Model . . . 109
8.4.1 Selecting the Geometry . . . 109
8.4.2 Using the Selected Technology Reference . . . 109
8.4.3 Example: MOM Plates below a Separate Route . . . 110
8.4.4 Supplying References and Acceptance Bounds . . . 111
8.5 Acting on a Check Result . . . 111

9 Comparison KPEX/2.5D with MAGIC 113
9.1 Test Pattern single_plate_100um_x_100um_li1_over_substrate . . . 113
9.2 Test Pattern sidewall_20um_length_distance_200nm_li1 . . . 115
9.3 Test Pattern sideoverlap_simple_plates_li1_m1 . . . 117
9.4 Test Pattern overlap_shield_same_net_100um_x_100um_li1_m1_m2 . . . 121
9.5 Test Pattern overlap_shield_same_net_half_100um_x_100um_li1_m1_m2 . . . 123
9.6 Test Pattern plate_20um_x_20um_li1_over_ntap . . . 125
9.7 Test Pattern plate_20um_x_20um_li1_over_ptap . . . 127
9.8 Test Pattern r_single_wire_li1 . . . 128
9.9 Test Pattern r_wire_voltage_divider_li1 . . . 129
9.10 Test Pattern r_contact_1x1_minsize_mcon . . . 130
9.11 Test Pattern r_via_stack_1x1_minsize_poly_to_met5 . . . 131
9.12 Test Pattern nfet_li1_redux . . . 133
9.13 Test Pattern nfet_li1 . . . 135

10 Netlist reduction: Feasibility study 141
10.1 TICER algorithm: TIme-Constant Equilibration Reduction . . . 141
10.1.1 Comments on page 2, equation (2.3) . . . 142
10.1.2 Quick Nodes . . . 143
10.1.3 Slow Nodes . . . 144

10.2 TICER 2007 paper: Pseudocode . . . 145
10.2.1 Pseudocode for slow node elimination . . . 146
10.3 Example from TICER 2007 paper . . . 147
10.3.1 Modified Nodal Analysis in the Laplace domain . . . 148
10.3.2 Analysis of the original circuit . . . 149
10.3.3 Analysis of the reduced circuit . . . 149
10.3.4 Comparison of original and reduced circuits . . . 150

11 Appendix **154**
11.1 PEX25D File Format: Represent Interconnect Geometry and Materials . . . 154
11.1.1 Why use PEX25D? . . . 154
11.1.2 From layout to solver input . . . 155
11.1.3 Artifact filenames and encodings . . . 155
11.1.4 A complete example . . . 157
11.1.5 PEX25D textual format reference . . . 158
11.1.6 Command-line tools . . . 169
11.1.7 Protobuf API: Python Preparation and C++ Consumers . . . 173
11.2 Plugin API . . . 175
11.2.1 PEX25D Importers and Exporters . . . 175
11.2.2 Using Plugins . . . 176
11.2.3 Writing a Plugin . . . 176
11.2.4 PEX25D Exporters . . . 177
References . . . 183

1 Introduction

1.1 Motivation

In Electronic Design and Automation (EDA) for Integrated Circuits (ICs), a schematic presents an abstraction in comparison to the layout that will eventually be taped-out and fabricated by the semiconductor foundry.

While in the schematic, a connection between device terminals is seen as an equipotential, the stacked geometries in a specific layout introduce parasitic effects, which can be thought of additional resistors, capacitors (and inductors), not modeled by and missing in the original schematic.

To be able to simulate these effects, a parasitic extraction tool (PEX) is used, to extract a netlist from the layout, which represents the original schematic (created from the layout active and passive elements) augmented with the additional parasitic devices.

1.2 Acknowledgements

This project is funded by the JKU/SAL IWS Lab, a collaboration of Johannes Kepler University and Silicon Austria Labs.

This project had been further funded by the German project FMD-QNC (16ME0831).

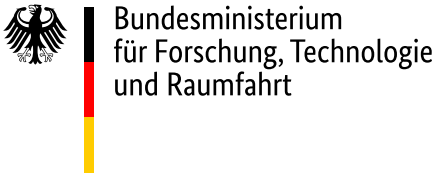
This project is further funded by the European project ODE4EC-AMS, which has received funding from the Chips Joint Undertaking (Chips JU) under Grant Agreement No 101252962. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Chips Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.



(a) Johannes Kepler University: Institute for Integrated Circuits and Quantum Computing



(a) Silicon Austria Labs



(a) Bundesministerium für Forschung, Technologie und Raumfahrt



(a) Co-funded by the European Union



(a) Chips Joint Undertaking

1.3 About KLayout-PEX

KLayout is an open source VLSI layout viewer and editor.

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

There are multiple PEX engines supported, currently:

- `FasterCap` integration (field solver engine)
- `FastCap2` integration (field solver engine, selected with `--fastcap`)
- 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`, `FastCap2`, `MAGIC`, `Palace`, `Elmer`, `OpenRCX`, `KQCircuits`, `protobuf`, `Quarto`, `Python`, `ngspice`, `Numpy`, `Scipy`, `Matplotlib`, `Git`, `Docker`, `Ubuntu`, `Linux`…

1.4 Status

> **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 Rust and parallelized to improve performance.

> **Warning**
>
> KPEX is experimental. Extraction and consistency checks do not establish calibrated accuracy for every device or layout.

Table 1: Engine capabilities

Engine	PEX mode	Status	What it produces
KPEX/MAGIC	CC, R, RC	Available	Extraction through an installed magic executable
KPEX/FasterCap	CC	Available, validation ongoing	Capacitance matrix from a field solver
KPEX/FastCap2	CC	Available, validation ongoing	Capacitance matrix from a field solver, selected with --fastcap
KPEX/2.5D	CC	Available, validation ongoing	Coupling capacitances between equipotential nets
KPEX/2.5D	R	Available, validation ongoing	Distributed resistance network
KPEX/2.5D	RC	Available, validation ongoing	Distributed resistances with coupling capacitances
KPEX/FastHenry2	R, L	Planned	No extraction command in the current release

RC includes coupling capacitances. There is no separate RCC value for --mode.

1.5 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.

1.5.1 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 Cheatsheet.

1.5.2 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/iic-jku/FasterCap/releases
- MAGIC-wrapper engine:
  - optional, required to run the MAGIC-wrapper engine
  - use `MAGIC 8.3.693` or newer for the corrected sidewall counting (see Section 5.4)
  - 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 ciel` (install PDK package manager)
  - `ciel ls-remote --pdk sky130A` (retrieve available PDK releases
    - for example `PRE-RELEASE 0c1df35fd535299ea1ef74d1e9e15dedaeb34c32 (2024.12.11)`)
  - `ciel enable --pdk sky130A 0c1df35fd535299ea1ef74d1e9e15dedaeb34c32` (install a PDK version)
  - PDK files now have been installed under `$HOME/.volare/sky130A`
- IHP SG13G2 PDK:
  - optional, for now, KPEX technology specific files are deployed within the `klayout-pex` Python package
  - `pip3 install --upgrade ciel` (install PDK package manager)
  - `ciel ls-remote --pdk ihp-sg13g2` (retrieve available PDK releases
    - for example `PRE-RELEASE cb7daaa8901016cf7c5d272dfa322c41f024931f (2025.07.18)`)
  - `ciel enable --pdk ihp-sg13g2 cb7daaa8901016cf7c5d272dfa322c41f024931f` (install a PDK version)
  - PDK files now have been installed under `$HOME/.volare/ihp-sg13g2`

1.5.3 Checking Tool Versions

The Python package requires `klayout>=0.30.12`. `KLayout 0.30.13` is recommended for terminal-lookup performance, neighborhood-visitor memory fixes and resistance-extraction crash fixes. The recommendation is a warning, so an environment still shipping `0.30.12` can run.

The CLI checks the Python module and the executable separately when both are used. Updating one does not update the other. An LVS-database-only extraction can use the module without the executable. The runtime table lists the version found, the required range and the recommended range.

Table 2: Runtime version-check policies

`--tool-version-checks`	Behavior
`on`	Default: unmet requirements stop the run, recommendations warn
`warn`	Report unmet requirements as warnings and continue
`off`	Skip version checks

An unrecognized tool version is reported as unchecked. `warn` and `off` do not repair the numerical or performance defects of an older executable.

For `--magic`, `KPEX 0.9.0` requires `MAGIC 8.3.693` or newer, which removes an extra sidewall halving. Pure `--2.5D` extraction does not require a `MAGIC` executable. The remaining multi-polygon device-terminal limitation is separate from the recommendation to upgrade `KLayout` and is not resolved by installing `0.30.13`.

1.5.4 Useful tools: `meshlab`

For previewing generated 3D geometries, representing the input to `FasterCap`, we recommend installing MeshLab. The generated `STL`-files are located at `output/<design>/Geometries/*.stl`.

2 First Steps

- The command line tool `kpex` is used to trigger the parasitic extraction flow from the terminal.
- Get extraction help with `kpex extract --help`
- Inspect or reduce netlists with the separate `netlist` command
- Inspect interchange files with the separate `pex25d` command

2.1 Example Layouts

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

```
git clone https://github.com/iic-jku/klayout-pex.git
cd klayout-pex

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

# for IHP SG13G2
find testdata/designs/ihp-sg13g2 -name "*.gds.gz"
```

2.2 Choosing Device Models

Blackboxing retains a recognized device's compact SPICE (Nagel 1975) model. Whiteboxing replaces the mapped metal-capacitor models with capacitance extracted from their physical geometry; other recognized device models remain. Whiteboxing is the current default, so choose explicitly for metal capacitors, such as the MOM-capacitors[1] in `sky130A`:

- `--2.5D --blackbox y`: retain the capacitor models and extract their external coupling
- `--fastercap --blackbox n` or `--fastcap --blackbox n`: replace the models with full-geometry field-solver capacitance

`FasterCap` and `FastCap2` reject `--blackbox y` before geometry generation when mapped metal-capacitor models would remain in the extracted netlist. The field matrix includes the intrinsic plate capacitance. Adding it beside the retained model would count that capacitance twice.

An exclusion marker alone does not identify all plate and enclosure geometry belonging to a device's intrinsic capacitance. Automatic device ownership is not implemented, and the explicit accounting policies in Section 11.1.5.7 do not bypass this guard.

A field-solver run with `--blackbox y` remains supported when no mapped metal-capacitor models are retained. This includes models removed by an explicit `--cut`.

2.3 Running `KPEX/2.5D`

From the `KPEX` source directory:

```
kpex extract --pdk sky130A --2.5D --mode RC --blackbox y \
  --gds testdata/designs/sky130A/test_patterns/nfet_li1.gds.gz \
  --out-netlist nfet_li1-rc.spice
```

[1] Metal-Oxide-Metal capacitors

`CC` extracts capacitances with one equipotential node per net. `R` resolves the wire resistance, and `RC` adds the coupling capacitances to that distributed network. The example retains the device models with `--blackbox y` (Section 2.2).

Use `--schematic` to follow an existing testbench interface (Section 6.4). `RC` enables consistency failures by default; Section 8 explains the diagnostics and the additional checks that can be requested.

Grounded capacitance is an explicit approximation: `--mode C` uses ideal conductors, while `--mode RC --coupling grounded` retains distributed resistance. The development-version options, floating-conductor treatment and standalone postprocessing are described in Section 7.

Each run directory contains `kpex_result.json`, which records the output paths, warnings, errors and final status. A written netlist can coexist with failed checks, so inspect the exit status and diagnostics before simulation.

2.4 Generating PEX25D

`kpex pex25d` generates a PEX25D geometry and material description without running a solver. It is separate from the analytical KPEX/2.5D engine. See Section 11.1 for the format specification, benefits, and standalone tools.

From the KLayout-PEX source directory:

```
kpex pex25d --pdk sky130A --blackbox y \
  --gds testdata/designs/sky130A/nmos_diode2/nmos_diode2.gds.gz \
  --out-file nmos_diode2.pex25d \
  --out-scene nmos_diode2.pex25d.scene.pb
```

The `.pex25d` file contains the text representation. The `.pex25d.scene.pb` file contains the resolved protobuf scene, including absolute layer heights and terminal intersections. Generation runs the reference validator by default; diagnostics determine the exit code even when output files have been written.

Use the standalone tool to inspect or validate an artifact:

```
pex25d show nmos_diode2.pex25d
pex25d validate nmos_diode2.pex25d --strict
```

For a complete, hand-written file that can be validated and exported without layout generation, use the downloadable example in Section 11.1.4. Terminal semantics and the current state of KLayout-generated terminals are explained in Section 11.1.5.9. For explicit capacitance result exclusions that retain physical geometry and shielding, use the downloadable accounting variant in Section 11.1.5.7.2.

2.5 Running the `KPEX/FasterCap` engine

Preconditions:

- `klayout-pex` was installed, see Section 1.5
- `FasterCap` was installed, see Section 1.5

2.5.1 Extracting Full Capacitor Geometry

This example selects `--blackbox n` to extract the capacitor's full geometry, following Section 2.2. First, try the command without selecting an engine:

```
kpex extract --pdk sky130A --blackbox n --gds \
  testdata/designs/sky130A/*/cap_vpp_04p4x04p6_l1m1m2_noshield.gds.gz
```

> **Note**
>
> This will report an error that we have not activated one or more engines, and list the available engines:
>
> | Argument | Description |
> |---|---|
> | `--fastercap` | Run kpex/FasterCap engine |
> | `--2.5D` | Run kpex/2.5D engine |
> | `--magic` | Run MAGIC engine |

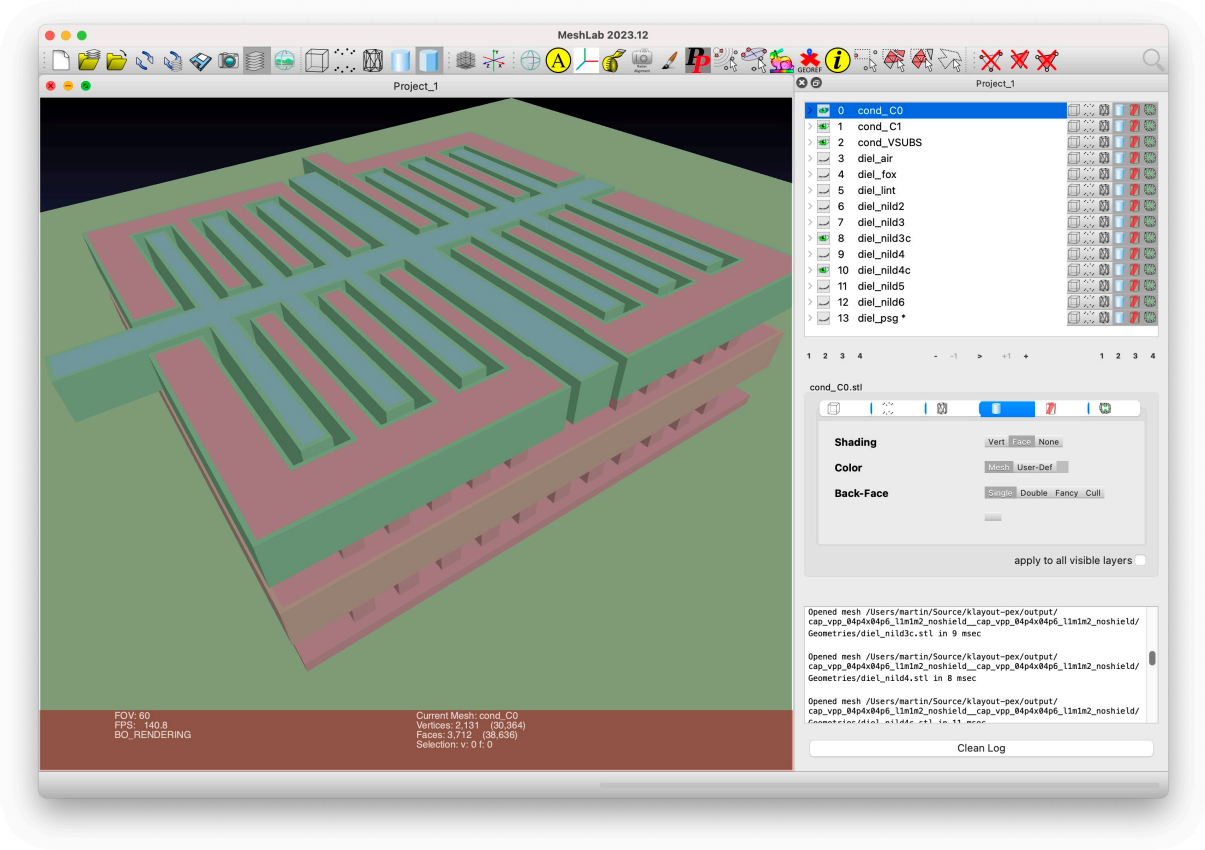
Now, to run the FasterCap engine (might take a couple of minutes):

```
kpex extract --pdk sky130A --blackbox n --fastercap --gds \
  testdata/designs/sky130A/*/cap_vpp_04p4x04p6_l1m1m2_noshield.gds.gz
```

Within the output directory (defaults to `output`), `KPEX` creates a subdirectory `Geometries`, containing STL-files that provide a preview of the `FasterCap` input geometries. Use `MeshLab` (see Section 1.5.4) to open and preview those files:

```
ls -d output/cap_vpp_04*/Geometries/*.stl
```

> **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 prints 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.

2.6 Running the `KPEX/MAGIC` engine

Preconditions:

- `klayout-pex` was installed, see Section 1.5
- `magic` was installed, see Section 1.5

The `MAGIC` section of `kpex extract --help` describes the arguments and their defaults. Important arguments:

- `--magicrc`: specify location of the `magicrc` file
- `--gds`: path to the GDS input layout
- `--magic`: enable magic engine

```
kpex extract --pdk sky130A --magic --gds \
  testdata/designs/sky130A/*/cap_vpp_04p4x04p6_l1m1m2_noshield.gds.gz
```

3 Choosing and describing PDKs

A PDK[2] supplies more than a list of GDS layer numbers. `KPEX` needs to know which shapes conduct, how they connect, what surrounds them in three dimensions, and which part of a device is already represented by its circuit model. The technology definition keeps these choices together, while the extraction algorithms remain independent of the PDK.

3.1 Choosing the process variant

`--pdk` selects a complete combination of extraction deck, layer stack, parasitic tables and device mappings. Choose the variant used by the layout and its simulation models. Changing only a top-metal name cannot turn one stack into another.

Table 3: Built-in process variants

`--pdk` value	Stack selected by `KPEX`	Extraction deck
`sky130A`	`li1`, five routing metals and the configured MIM plates	`sky130.lvs`
`gf180mcuD`	Five routing metals, thick `Metal5` and MIM option B	`gf180mcu.lvs`
`ihp-sg13g2`	Five thin metals, `TopMetal1`, `TopMetal2` and MIM plates	`sg13g2.lvs`
`ihp-sg13cmos5l`	Four thin metals and `TopMetal1`	`sg13cmos5l.lvs`

The older spelling `ihp_sg13g2` remains an alias for `ihp-sg13g2`. Use the hyphenated spelling in new commands and scripts. The other three spellings in Table 3 are the accepted identifiers, including their capitalization.

This is a list of extraction configurations, not a claim that every device of each PDK has a simulation mapping. An extracted device without a supported mapping produces an error instead of a guessed model call.

3.2 Forming nets and identifying layers

The PEX extraction decks use `KLayout`'s LVS facilities to recognize devices and establish connectivity. This step produces the flat netlist and the named, net-associated polygons in the `LVSDB`. Comparing that result with a schematic is a separate operation.

[2] Process Design Kit

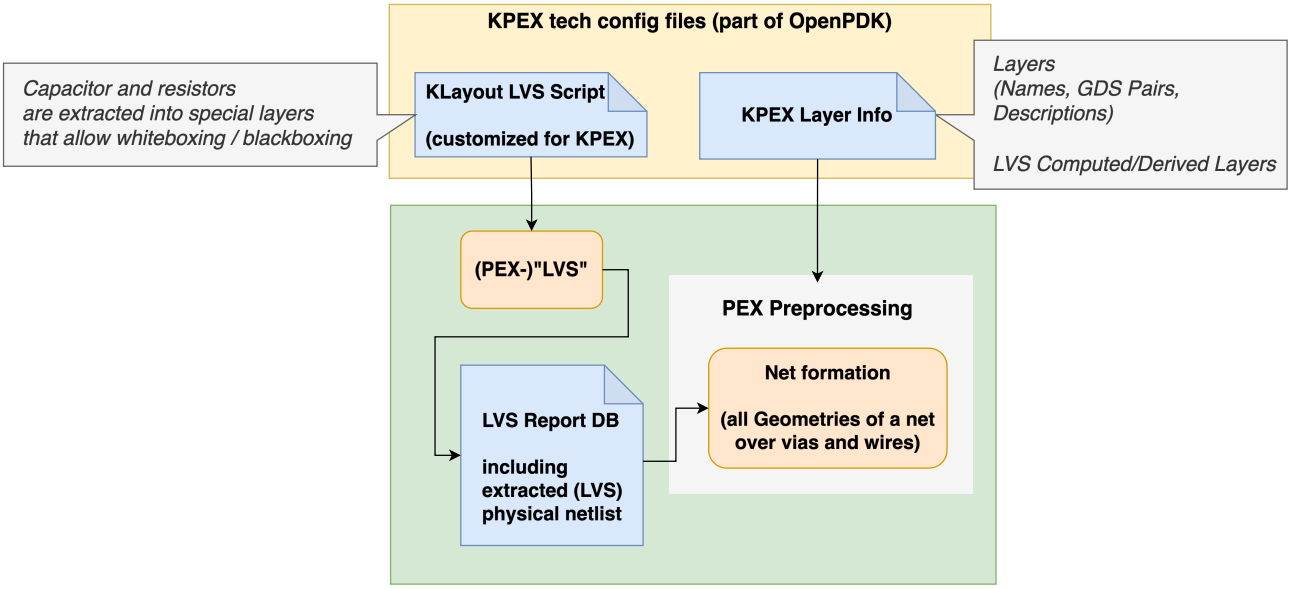


Figure 6: KPEX net formation

The deck and technology definition must agree on the meaning of each computed layer. A name alone is insufficient: a piece of drawn metal may have a different role inside a capacitor than outside it.

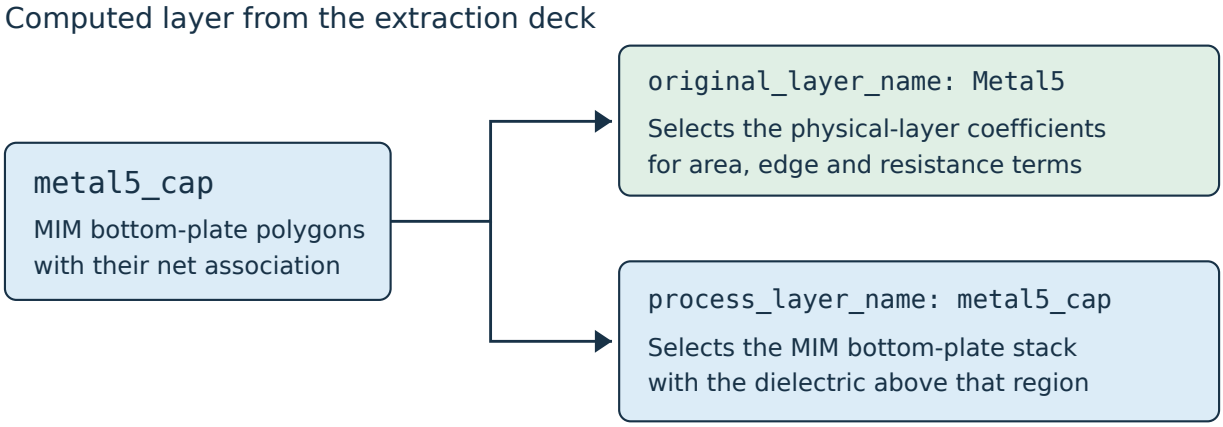


Figure 7: A computed MIM plate selects a coefficient layer and a process-stack layer independently. The names shown are from the `ihp-sg13g2` technology.

There are three identities to keep distinct:

- The **computed layer** identifies polygons produced by the extraction deck, including their net and device role
- The **original layer** selects the physical layer's parasitic coefficients
- The **process layer** selects its position and material environment in the three-dimensional stack

For example, the `ihp-sg13g2` deck splits drawn `Metal5` into ordinary routing, MIM[3] bottom-plate and device-related regions. All use coefficients keyed by `Metal5`, but their process layers distinguish metal under the MIM dielectric from ordinary metal.

[3] Metal-Insulator-Metal capacitors

Table 4: Layer identities in the `ihp-sg13g2` technology

Computed layer	Original layer	Process layer	Meaning
`metal5_n_cap`	`Metal5`	`metal5_n_cap`	Ordinary metal
`metal5_cap`	`Metal5`	`metal5_cap`	MIM bottom plate
`metal5_rfcap`	`Metal5`	`metal5_cap`	RF MIM bottom plate
`metal5_rfcap_feed`	`Metal5`	`metal5_n_cap`	RF MIM feed metal
`metal5_mom`	`Metal5`	`metal5_n_cap`	MOM finger metal

The computed-layer kind also specifies whether a region is ordinary conductor, a modeled capacitor plate, a resistor region, or device-internal geometry. This is why removing a capacitor from the parasitic sum does not necessarily remove its plates from shielding or from coupling to other nets. See Section 6 and the section on excluding parts of the layout for the extraction choices.

When adding a deck, assign stable names with `name(layer, name)`. Keep the named device regions and the technology's layer identities consistent. A pin label, a device plate and a physical routing layer are different contracts even when they originate from the same drawn GDS layer.

3.3 Keeping technology data together

The technology definition uses a Protocol Buffers schema and is distributed as JSON. Its parts answer different questions:

Table 5: Responsibilities of the technology definition

Technology data	Question answered	Used for
Drawn and computed layers	Which shapes, labels and device regions have this meaning?	Connectivity, port discovery and exclusions
Process stack	Where is each conductor, and which dielectric surrounds it?	3D geometry
Parasitic tables	What sheet resistance and fitted capacitance coefficients apply?	`KPEX/2.5D`
Device mappings	How is an extracted device written for the circuit model?	Netlist export and schematic comparison
Substrate definition	Which net or label represents the substrate?	Substrate terminal handling
Model capacitance references	What nominal device coupling can be compared independently?	Device-capacitance checks

The main schema is `tech.proto`. It incorporates `process_stack.proto`, `process_parasitics.proto`, `device_models.proto` and the substrate definition. These schemas define the data contract; the generated JSON contains the values for a particular variant.

Changing a dielectric height does **not** automatically refit the `KPEX/2.5D` coefficients. The stack and tables are separate inputs. This matters when comparing a three-dimensional solve with the table-based engine: both must describe the same intended physical process.

3.4 Describing the process stack

For 3D solvers, like the `KPEX/FasterCap` engine, 3D information about the dimensions (e.g. z-offset and thickness) of metal layers is required, as well as the dielectrics in-between.

The `sky130A` stack illustrates why a list of metal heights alone is not enough: several different dielectric profiles surround the same conductor.

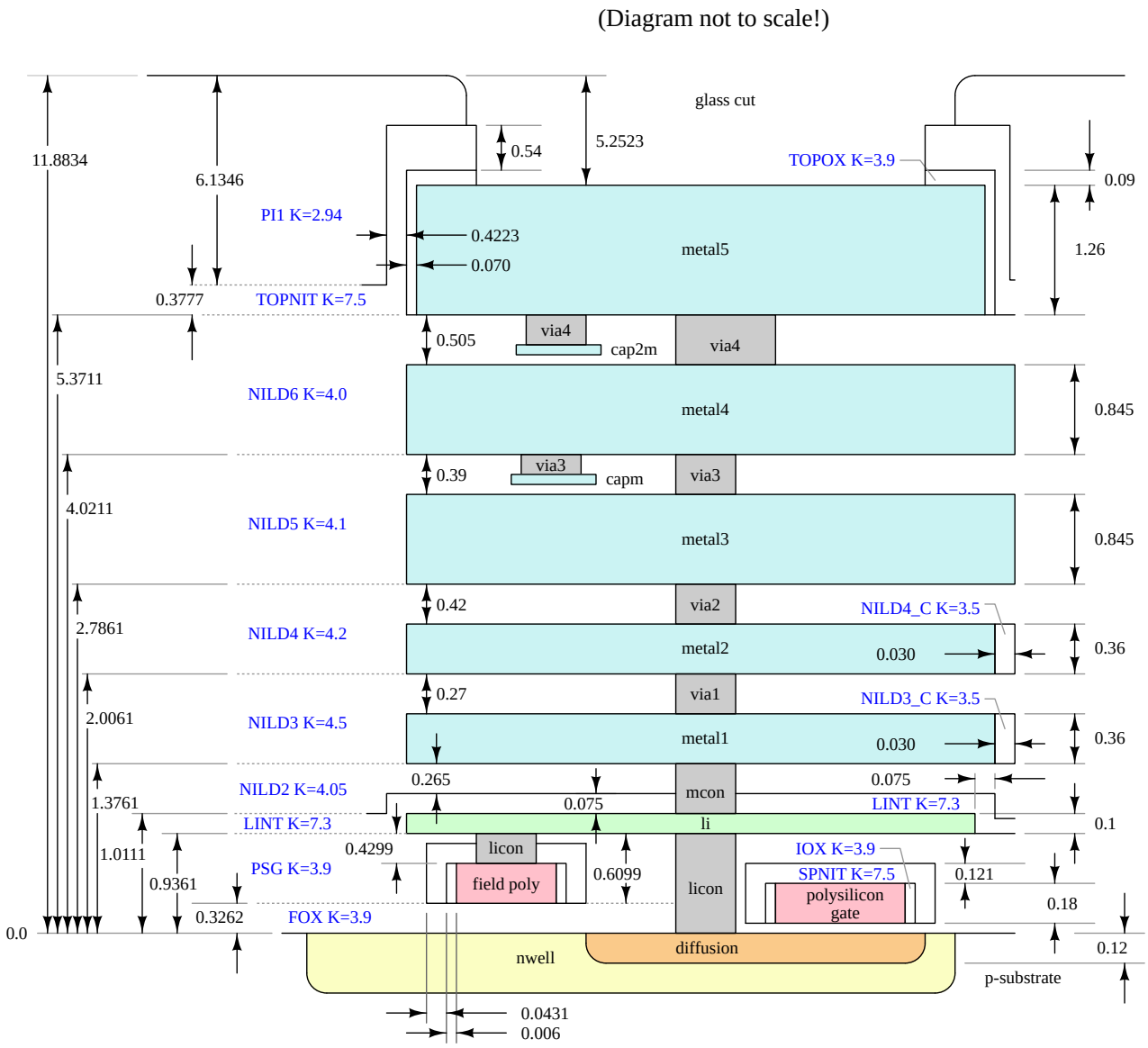


Figure 8: `sky130A` process stack (Source: (Authors 2020))

The stack distinguishes conductors from the material around them:

- Metals such as `poly`, `metal1` and `metal5` have a thickness and a bottom height relative to the substrate
- Contacts and vias, such as `licon` and `via1`, connect conductors on adjacent levels
- Sidewall dielectrics, such as `IOX` and `SPNIT`, use conformal profiles with lateral thickness and no thickness in the field. A profile may also cover the metal top
- Conformal dielectrics, such as `LINT` and `TOPNIT`, specify thickness above metal, beside it and in the field
- Simple dielectrics, such as `PSG` and `NILD2`, fill the remaining region between levels

Physical corners can change these profiles and manufactured widths. Section 3.9.12.1 describes the available source recipes, and Section 3.9.12.6 distinguishes incremental coating offsets from cumulative field extents.

The relative permittivity k belongs to each dielectric. A stack may therefore contain several dielectric boundaries around one piece of metal. The schematic labels above and the technology's physical-layer names need not be identical; their correspondence is explicit in the technology definition.

The schema excerpt below shows how these relationships are expressed:

- `ProcessStackInfo.layers` orders the layers of the stack
- `LayerInfo.layer_type` identifies the layer kind with a `LayerType` enum value
- Its `parameters` payload must match that kind
- `MetalLayer.z` locates the metal bottom relative to the substrate
- `MetalLayer.contact_above` describes the contact to the next conductor
- `MetalLayer.line_width_bias` changes total manufactured width for physical export
- `MetalLayer.physical_geometry_unavailable` marks a profile whose physical law is unqualified
- A dielectric `reference` identifies the layer or profile it wraps
- Wrapping can be nested: `SPNIT` wraps `IOX`, which wraps `poly`

```
message ProcessStackInfo {

    enum LayerType { …
        LAYER_TYPE_SIMPLE_DIELECTRIC = 50;
        LAYER_TYPE_CONFORMAL_DIELECTRIC = 60;
        LAYER_TYPE_METAL = 80;
    }

    message Contact { // Contact/Via
        string name = 1;
        string layer_below = 9;
        string metal_above = 10;
        double thickness = 20;

        double width = 30;
        double spacing = 31;
        double border = 32;
    }
    …
    message SimpleDielectricLayer { // Simple dielectric
        double dielectric_k = 10;
        string reference = 30;
    }

    message ConformalDielectricLayer { // Conformal dielectric
        double dielectric_k = 10;
        double thickness_over_metal = 20;
        double thickness_where_no_metal = 21;
        double thickness_sidewall = 22;
        string reference = 30;
    }

    message MetalLayer { // Metal
        // Bottom height above substrate and thickness, in µm
        double z = 1;
        double thickness = 2;
        double line_width_bias = 20;
        bool physical_geometry_unavailable = 30;
        Contact contact_above = 40;
    }

```

```
42      message LayerInfo {
43          string name = 1;
44          LayerType layer_type = 2;
45          oneof parameters { …
46              SimpleDielectricLayer simple_dielectric_layer = 50;
47              ConformalDielectricLayer conformal_dielectric_layer = 60;
48              MetalLayer metal_layer = 70;
49          }
50      }
51
52      repeated LayerInfo layers = 100;
53  }
```

3.4.1 Planar passivation over `gf180mcuD`

The top surface is an instructive example of why the reference of a dielectric matters. For the selected `1P5M/11KA/MIM-B` stack, the maintained cross-section drawing gives the oxide surface relative to the **bottom** of `Metal5`.

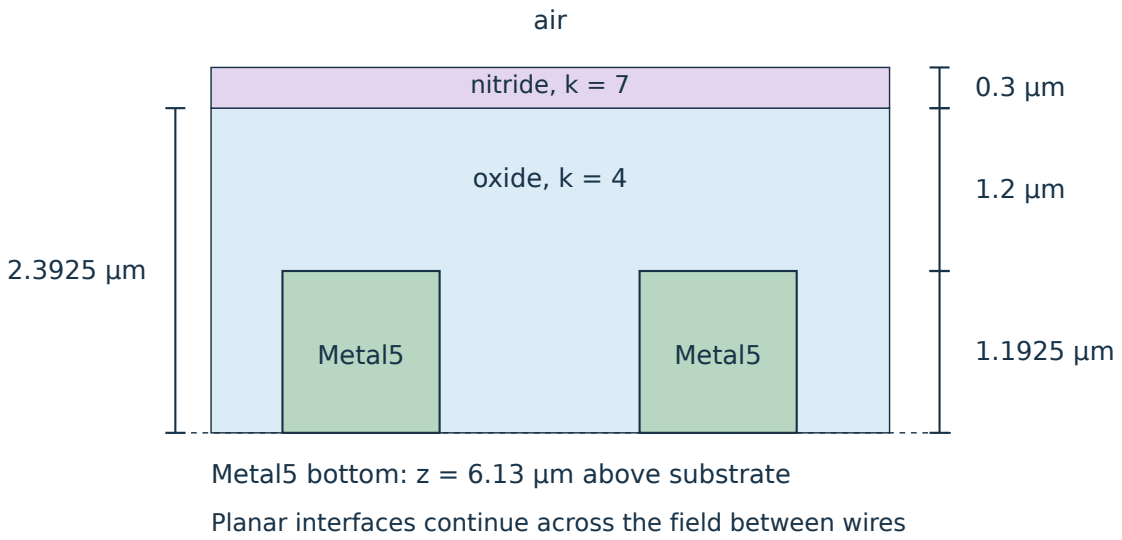


Figure 9: Planar top passivation of `gf180mcuD`, shown relative to the bottom of `Metal5`. Vertical distances follow the selected cross-section; horizontal widths are illustrative.

`Metal5` is 1.1925 μm thick. The planar oxide surface is 2.3925 μm above its bottom, leaving 1.2 μm of oxide over the metal. A 0.3 μm nitride layer sits above that surface. Its relative permittivity is 7, while the oxide uses 4, as recorded in the maintained dielectric table.

The technology expresses these planar surfaces using two wrapped dielectric profiles:

- `pass` wraps `Metal5`, with 1.2 μm above metal and 2.3925 μm in the field
- `sin` wraps `pass`, with 0.3 μm above the oxide and 2.6925 μm in the field
- Both use zero sidewall growth, preserving flat interfaces over the gap

The field height of `sin` includes the oxide height because it is measured from the root metal's bottom. Treating it as only 0.3 μm would place nitride near the bottom of the gap instead of above the oxide.

The general dielectric table lists 2.3 μm for the `11KA` oxide. `KPEX` uses the detailed cross-section's 2.3925 μm consistently with the metal thickness from the same drawing. This is an explicit source

choice, not a fitted capacitance correction. The maintained community PDK repository is the source of these dimensions.

3.5 Interpreting parasitic coefficients

The capacitance tables are a compact model of geometry. Each family has different dimensions because it multiplies a different geometric quantity. Mixing their units can yield plausible-looking values with the wrong scale.

Table 6: Capacitance coefficient units

Table family	Stored quantity	Unit	Geometric factor
Substrate area	Area density	aF/μm^2	Exposed area
Substrate perimeter	Edge density	aF/μm	Edge length and fringe fraction
Interlayer overlap	Area density	aF/μm^2	Unshielded overlap area
Same-layer sidewall	Numerator b, offset o	aF, μm	Length divided by effective gap
Directed side overlap	Edge density	aF/μm	Edge length and fringe fraction

For an overlap area A and area coefficient c_A, the model contribution is $C = c_A A$. The coefficient is per **square** micrometre. A perimeter or fringe coefficient instead multiplies a length, and the fringe calculation supplies the fraction left after distance limits and shielding.

The sidewall coefficient has a different form. For two facing parallel edges of length L, gap s and fitted offset o, the table convention is

$$C_{\text{pair}} = b\,\frac{L}{s_{\text{eff}}}, \qquad s_{\text{eff}} = \max(s + o, s/2).$$

The ratio of lengths is dimensionless, so b is in attofarads. It is **not** an area density or a capacitance per micrometre. A negative fitted offset is allowed; the lower bound on effective separation prevents a zero or negative denominator for positive gaps. For a varying gap, integrate this density along the edge instead of substituting an average gap.

This equation defines the full-pair coefficient convention. In `KPEX 0.9.0`, each of the two directed visits contributes half, so their sum follows this law. Earlier versions applied an extra halving. The coefficient must not be doubled to compensate for that older behavior: contribution counting and physical table values are separate parts of the model.

The stored capacitance units are attofarads; extraction reports commonly use femtofarads, with $1000\,\text{aF} = 1\,\text{fF}$. Resistance tables use milliohms per square for sheets and milliohms per contact or via cut.

3.5.1 Direction and physical consistency

A side-overlap row from an edge on `Metal1` toward a conductor on `Metal2` describes a different geometric contribution from a row in the reverse direction. These coefficients can differ without violating reciprocal capacitance: the final coupling of a node pair includes the applicable contributions, not a choice of just one directional row.

Physical comparisons need the whole environment:

- Equal conductor spacing and equal dielectric material give the same parallel-plate density
- Equal gaps alone do not imply equal sidewall fits when height above substrate or nearby conductors differs
- Variants with identical complete conductor and dielectric surroundings should not acquire different values solely from renaming layers
- A changed fit can alter both b and o, so compare its predicted capacitance over the relevant gaps

Shared lower-layer dimensions do not make two complete stacks identical. Different upper metals or passivation can change the fitted capacitance of those lower conductors.

The tables are based in part on the extraction sections of `MAGIC` technology files. Importing those values does not import `MAGIC`'s layer classification, device recognition, shielding or grid rounding. Agreement of one test pattern therefore does not establish agreement for every layout.

3.6 Mapping extracted devices to circuit models

An LVS device class is not necessarily the model name or primitive type expected by a simulator. The mapping declares:

- The instance prefix, such as `M` or `X`
- The target model or subcircuit name
- The order of its terminals
- Which parameters are exported, their names and their conversion factors
- Fixed parameters, such as an RF-mode selector

For example, the `ihp-sg13*` LVS class `rfnmos` maps to an `X` call of `sg13_lv_nmos` with `rfmode=1`. Its terminal order is `D`, `G`, `S`, `B`. The mapping supplies this information explicitly instead of assuming that every four-terminal device is a native MOS model.

Lengths and areas also need an explicit contract. An `ihp-sg13*` LVS length stored in micrometres is exported to the model in metres using `lvs_parameter_factor=1e-6`. An area needs the square of that length factor. A mapping may give `schematic_parameter_factors` for different SPICE prefixes: a native `M` device read with geometry scaling and an `X` wrapper parameter do not have identical unit semantics. These factors describe the PDK model interface; they are separate from the netlist's `.option scale` setting.

For symmetric area/perimeter models, the mapping can recover equivalent rectangle sides from area A and perimeter P. The sides solve $wl = A$ and $2(w + l) = P$. This preserves those two geometric quantities without claiming to recover the original orientation. It is unsuitable for a transistor or varactor whose model depends differently on width and length.

3.6.1 Independent capacitance references

A `KIND_METAL_CAPACITOR` mapping can also contain `capacitance_references`. These describe the nominal mutual capacitance of a particular unordered terminal pair. They are used by the device-capacitance check, not to tune the extraction result.

Each reference states:

- Two terminal names whose local plates can be compared
- A capacitance expression in farads, using mapped model parameters
- Optional derived parameters and a validity condition
- A source and its nominal assumptions

The instance multiplier belongs in the reference exactly once. A validity condition restricts the expression to the supported geometry; it is not permission to extrapolate outside the model's domain. The check evaluates the recorded expression without running a simulator or searching arbitrary SPICE libraries.

The bundled references currently cover `cap_cmomf` and `cap_cmomi` in `ihp-sg13*`. The `cap_cmomf` reference uses nominal area density and the selected metal count. The `cap_cmomi` reference additionally reflects its discrete finger array and feed option. Their provenance is pinned to the device model sources.

These references do not include process-corner or mismatch variation. Some densities are estimates or derived from a sibling stack, as stated in each reference's `description`. A ratio between extracted and nominal capacitance helps find a discrepancy; it does not by itself identify which model is physically correct. Unlisted device pairs have no such reference check.

3.7 Updating technology data

The built-in definitions are produced by Python generators[4]. The generated JSON and Python schema modules are packaged in the distribution, but are not committed source files in the code repository. Editing an installed JSON file is temporary: a rebuild or package upgrade replaces it.

For a source change:

1. Change the relevant PDK generator and maintain its source records, assumptions and data-revision history as in Section 3.9.5
2. Run `./gen_tech_pb.sh` from the `klayout-pex` checkout to regenerate the schema modules and all built-in JSON definitions
3. Compare the resulting tables by physical layer and coefficient family
4. Check the affected geometries and sibling variants before interpreting changed extraction values

The generators contain deliberate projections, not only verbatim copies of upstream tables. Diffusion rows may be shared between extracted implant or tap regions. MIM plates retain their model-derived plate coupling while surrounding parasitics come from other rows. Resistance overrides have their own sources. A table refresh must preserve these distinctions and show changed values for review.

The nominal `sky130A` and `gf180mcuD` tables in `KPEX 0.9.0` follow the maintained `open-pdks` extraction sources. The `ihp-sg13g2` and `ihp-sg13cmos5l` tables use the corrected extraction fits at `14b1ab8`. These refreshes preserve the diffusion and MIM derivations and resistance overrides. They do not change the independent device-model references. Automatic calibration and selectable capacitance corners are not available in `0.9.0`. The development version adds complete low/high capacitance components for all four PDKs; Section 3.9.7 explains their selection and the remaining qualification limits. Both IHP variants also provide isolated `GatPoly` sheet-resistance sensitivities, described in Section 3.9.8. The `gf180mcuD` resistance sensitivities use explicit plate-source assumptions and distinguish unspecified contact minima from an ideal-contact extraction assumption (Section 3.9.11).

The `sky130A` development catalog also provides four fixed-layout resistance sensitivities with explicit MIM sheet bounds. Their source-dependent overrides, engineering assumptions and via-stack example are described in Section 3.9.10.

[4] The Python generators in `scripts/gen_tech_pb` define the built-in technologies. `./gen_tech_pb.sh` regenerates their JSON files and the Python modules from the protobuf schemas.

3.8 Inspecting the generated definitions

The previews below show the complete resolved nominal `Technology` messages for the development version following `0.9.0`. They contain the selected coefficients, physical stack and device interfaces used by extraction. The catalog examples in Section 3.9.3 show how these components are selected and how their provenance is recorded.

To inspect the data actually installed, locate `klayout_pex_protobuf` beside the `klayout_pex` package in `site-packages`[5]. The resolved nominal snapshots use the PDK identifiers from Table 3, followed by `_tech.pb.json`. The catalogs and their component datasets are under `tech/<PDK>/`.

Each process-stack layer declares its kind in `layer_type` and supplies exactly one matching `parameters` payload. For example, `LAYER_TYPE_METAL` requires `metal_layer`. Unspecified or unknown kinds, missing payloads and mismatches are rejected when the technology is loaded.

3.9 Selecting Technology Corners

An extraction corner selects the data used to calculate parasitics. It does not automatically select transistor models, temperature or supply voltage for the subsequent simulation. Those choices can be independent in an open PDK.

> **Note**
>
> Corner selection belongs to the development version following `0.9.0`. Each supported technology supplies complete `nominal`, `low` and `high` capacitance components. A custom scenario can select those components explicitly.
> The development catalogs also provide:
>
> - Isolated IHP `GatPoly` sheet-resistance sensitivities (Section 3.9.8)
> - Complete `ihp-sg13cmos5l` fixed-layout resistance sensitivities (Section 3.9.9)
> - `sky130A` resistance sensitivities with explicit MIM sheet bounds (Section 3.9.10)
> - `gf180mcuD` resistance sensitivities with explicit plate and ideal-contact assumptions (Section 3.9.11)
> - Independent MIM device-reference components (Section 3.9.13)
> - Source-related physical recipes (Section 3.9.12.1)
>
> These selections retain their documented assumptions. An upstream style name does not automatically declare a matching physical stack or a correlated process corner.

3.9.1 Selecting Declared Scenarios

Both `kpex extract` and `kpex pex25d` accept `--corner`. A scenario names complete resistance, capacitance, physical-stack and optional device-reference datasets. With no explicit option, the catalog's declared default is selected.

```
kpex extract --pdk sky130A --2.5D --mode RC --corner nominal \
  --gds r_wire_voltage_divider_li1.gds.gz \
  --out-netlist divider.spice --out-result divider-result.json
```

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_wire_voltage_divider_li1.gds.gz

[5] To find the `site-packages` directory for the `klayout-pex` package, call `pip3 show klayout-pex`.

This example uses the bundled nominal data and a distributed wire pattern. Omitting `--corner nominal` produces the same technology selection while `nominal` is the catalog default.

A separately supplied catalog can be selected explicitly:

```
kpex extract --pdk sky130A --2.5D --mode RC \
  --tech-catalog path/to/catalog.pb.json --corner nominal \
  --gds r_wire_voltage_divider_li1.gds.gz \
  --out-netlist divider.spice
```

The catalog must identify the selected `--pdk`. A catalog for one process cannot silently change the layout/LVS interface of another. Scenario identifiers are case-sensitive: `nominal` and `Nominal` are different names.

Unknown selections fail before the layout or LVS database is read. The diagnostic lists the valid scenarios, so a typo cannot silently fall back to nominal extraction.

3.9.2 Keeping the Sources Independent

The four datasets answer different physical questions:

Table 7: Component datasets selected by a scenario

Dataset	What it determines
Resistance	Sheet resistance and per-cut contact/via resistance
Capacitance	Area, sidewall and fringe coefficients, including fitted offsets and halo
Physical stack	Layer positions, film thicknesses and dielectric constants for geometry generation
Device references	Independent capacitance estimates used by device checks

Layout layers, computed-layer identities, device interfaces, substrate connectivity and exclusion definitions remain shared. Selection resolves a fresh complete technology before extraction caches are constructed. Running one corner and then another does not accumulate table rows or modify a previous selection.

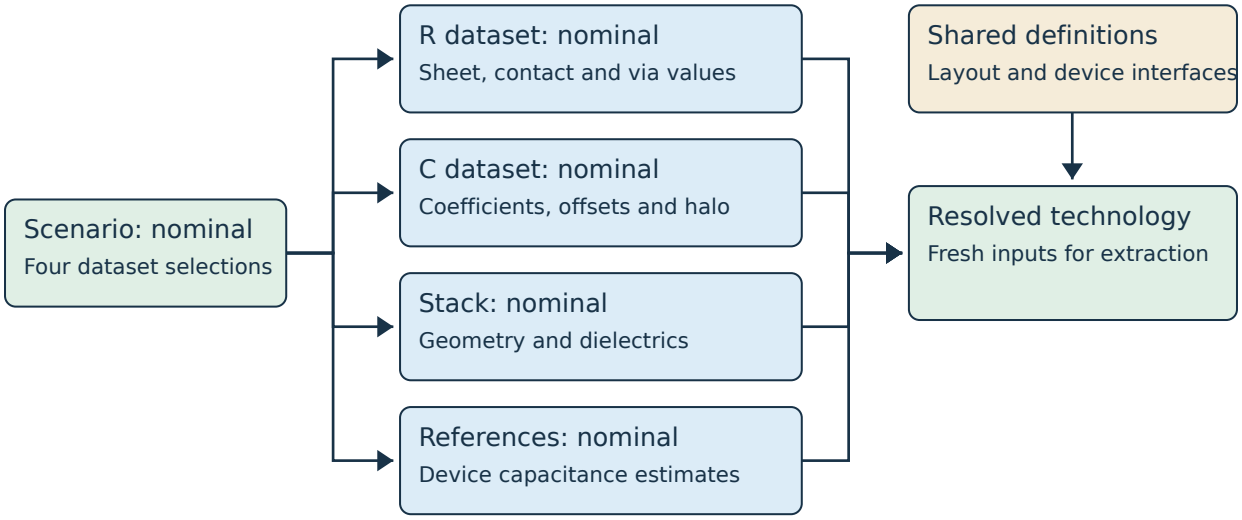


Figure 10: One explicit scenario selects separate complete datasets. Shared layout and device interfaces remain unchanged. The nominal labels are those of the first bundled catalogs.

A missing table entry is not the same as an explicit zero. Each selected resistance and capacitance table must cover the same row keys as the default scenario. Complete components can be shared by several scenarios without inheriting unspecified numerical entries from nominal data.

3.9.3 Inspecting Catalog Data

The following examples contain the generated `pb.json` files for the current corner implementation. Each complete example includes its catalog, shared technology, all four nominal datasets, and both additional capacitance datasets (`low` and `high`). The bundles also contain the available resistance, MIM device-reference and physical-stack components described below. Unpack a bundle before passing its catalog to `--tech-catalog`; component paths are relative to that catalog.

Table 8: Generated catalogs and their complete datasets

Technology	Catalog	Complete example
`sky130A`	`catalog.pb.json`	`sky130A.zip`
`gf180mcuD`	`catalog.pb.json`	`gf180mcuD.zip`
`ihp-sg13g2`	`catalog.pb.json`	`ihp-sg13g2.zip`
`ihp-sg13cmos5l`	`catalog.pb.json`	`ihp-sg13cmos5l.zip`

For example, the `sky130A` catalog selects separate resistance, capacitance, physical-stack and device-reference datasets. The shared technology supplies the layer and device interfaces.

The catalog and datasets each have a `description`. A `TechnologyFile` reference identifies the file by `path` and `sha256`; the description and provenance belong to the referenced content.

3.9.4 Tracing Sources and Assumptions

Each catalog and dataset has structured `provenance`:

Table 9: Evidence accompanying technology data

Field	Meaning
`data_revision`	Authored revision of this catalog or dataset
`sources`	Source identifiers, descriptions, URIs, revisions, hashes and preprocessing options
`assumptions`	Explicit projections or approximations, their reasons and supporting source identifiers
`history`	Dated changes with their revision, summary, reason, source identifiers and review links

A source must identify a revision or an exact SHA256 digest. Extraction records these references without downloading them. A source revision identifies the input document, while `data_revision` identifies the KPEX interpretation of that input. The two can change independently.

For example, the `ihp-sg13g2` capacitance dataset identifies the corrected extraction file at `14b1ab8`. Its resistance dataset separately records the nominal `GatPoly` and MIM assumptions. The device-reference dataset cites its independent MOM models. A common process name does not make those sources interchangeable.

An assumption states both what was assumed and why. Diffusion projection and the thin-plate approximation for surrounding MIM coupling therefore remain visible alongside the imported coefficients. Recording an assumption does not establish a measured error bound or qualify a physical corner.

3.9.5 Maintaining Data History

Maintain history when reviewing a data change:

1. Add new source records for changed upstream inputs, with new identifiers and pinned revisions or hashes
2. Describe changed assumptions and their physical reasons
3. Append a history entry with the change date, summary, reason and supporting source or review references
4. Set `data_revision` to that final entry's revision, then regenerate the files and their catalog hashes
5. Compare the numerical tables and check the affected geometries and sibling variants

Keep previous history entries and the source identities they reference unchanged. Entries are ordered from oldest to newest. Multiple revisions on one date are allowed. A build does not invent a new revision or timestamp, and revision labels are not interpreted as corner names or version ranges.

The catalog's history describes its available components and declared scenarios. Each component retains its own history, so updating capacitance does not invent a resistance or physical-stack revision.

`schema_version` describes the catalog format. `data_revision` describes the data and its interpretation. File hashes identify exact serialized bytes, while the resolved payload hashes identify numerical inputs. A clarification of an assumption changes file identity without changing coefficient identity.

The technology format is under development before `1.0.0`. After a schema change, regenerate the examples with the matching KPEX version. Assigned field numbers and JSON names will become compatibility contracts when the format is stabilized.

3.9.6 Reading Upstream Corner Names

The maintained `MAGIC` extraction decks for the supported technologies declare these R/C combinations:

Table 10: Explicit upstream extraction combinations, distinct from bundled availability

Style suffix	Resistance	Capacitance
Nominal style	Nominal	Nominal
`hrhc`	High	High
`hrlc`	High	Low
`lrhc`	Low	High
`lrlc`	Low	Low

The sources are the maintained OpenPDKs `sky130` deck, OpenPDKs `gf180mcu` deck, and the corrected `ihp-sg13g2` and `ihp-sg13cmos5l` decks.

These names do not define all nine products of low, nominal and high R/C. Nor do transistor labels such as `tt`, `ff` and `ss` imply an extraction corner. For example, the `sky130` simulation library separates MOS corner selection from its R/C model sections.

KPEX also contains derived MIM and diffusion rows and resistance values from other process sources. An imported corner must resolve those rows explicitly. The presence of a high-C block in an extraction deck is not evidence for an otherwise undocumented MIM sheet-resistance limit.

Separate component files do not imply statistical independence of R and C. In `sky130A`, some resistance-model overrides differ between `hrhc` and `hrlc`, even though both names start with high R. Such sources require scenario-specific R datasets rather than one generic high-R component.

3.9.7 Using Low/High Capacitance Components

Each supported PDK has complete `nominal`, `low` and `high` capacitance datasets. The additional components cover substrate area and perimeter, overlap, fitted sidewall and directed fringe terms. They retain the diffusion projections and MIM construction used by the nominal technology, with the selected capacitance data.

Table 11: Capacitance components and selection examples

Technology	Capacitance datasets	Selection examples
`sky130A`	`nominal`, `low`, `high`	`nominal`; R/C sensitivities in Section 3.9.10
`gf180mcuD`	`nominal`, `low`, `high`	`nominal`; R-only sensitivities in Section 3.9.11
`ihp-sg13g2`	`nominal`, `low`, `high`	`nominal`; `GatPoly` sensitivities in Section 3.9.8
`ihp-sg13cmos5l`	`nominal`, `low`, `high`	`nominal`; sheet and full resistance sensitivities in Section 3.9.9

Each catalog also declares `capiche-low-c` and `capiche-high-c`, selecting the source-related physical recipes described in Section 3.9.12.1.

The `low` tables agree between the reviewed `hrlc` and `lrlc` styles, and the `high` tables agree between `hrhc` and `lrhc`, for all four PDKs. This permits sharing each capacitance component across those resistance choices; it does not prove that resistance can be factored in the same way.

These are complete tables, not a global scale factor applied to nominal capacitance. A sidewall fit changes its numerator and spacing offset together, so a larger numerator alone does not identify the larger predicted capacitance. Compare the resulting law over the relevant geometry, as described in Section 3.5.

The component provenance records the source choices that complete the tables:

- `gf180mcuD`: the missing low-C `Metal2`-to-well fringe row uses the same-corner substrate perimeter coefficient, with the established well-plane projection
- `ihp-sg13g2`: the MIM area law varies with the selected data, while its literal edge coefficient remains unchanged, with the separate `cap_cpara` factor excluded from that edge law
- Every PDK retains its explicit diffusion and surrounding-MIM coupling assumptions, without claiming matching physical-stack or device-reference data

3.9.7.1 Holding Resistance Nominal

A sensitivity study can deliberately hold resistance nominal and vary only capacitance. This is a user-defined comparison, not a claim that the foundry specifies that R/C combination as a correlated process corner.

The complete `sky130A` sensitivity example contains the same component bytes as the bundled catalog, plus two declared scenarios, `nominal-r-low-c` and `nominal-r-high-c`. Unpacking it creates the `sky130A-sensitivity` directory. Its default remains `nominal`, and the custom scenario definitions are recorded in a new catalog-history entry.

The low-C scenario selects:

```
{
  "id": "nominal-r-low-c",
  "resistance": "nominal",
  "capacitance": "low",
  "process_stack": "nominal",
  "physical_stack_matches": false,
  "device_references": ""
}
```

The high-C scenario selects `"capacitance": "high"` under its distinct scenario identifier. Both use the nominal stack only for compatible layer identities and conductor ordering. The `physical_stack_matches: false` and empty `device_references` declarations prevent that structural use from claiming matched solver geometry or independent corner-specific device references.

3.9.7.2 Comparing Extracted Couplings

The following commands use the same `sky130A` layout of overlapping `li1` and `met1` plates for all three selections. The `CC` mode retains mutual and substrate capacitances; the nominal resistance selection is present in the catalog but does not add resistors in this mode.

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/overlap_plates_100um_x_100um_li1_m1.gds.gz

```
unzip sky130A-sensitivity.zip
for corner in nominal nominal-r-low-c nominal-r-high-c; do
  kpex extract --pdk sky130A --2.5D --mode CC \
    --tech-catalog sky130A-sensitivity/catalog.pb.json \
    --corner "$corner" \
    --gds overlap_plates_100um_x_100um_li1_m1.gds.gz \
    --out-dir "output/$corner" \
    --out-netlist "output/$corner.spice" \
    --out-result "output/$corner.json"
done
```

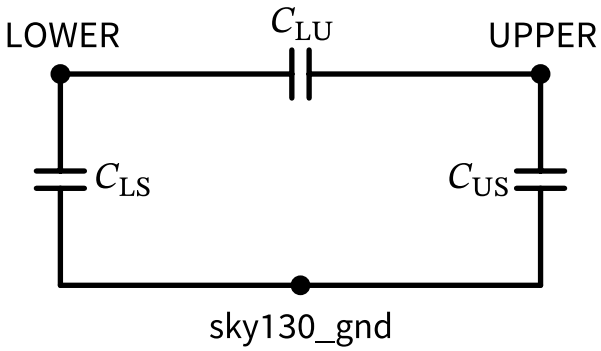


Figure 11: The three-terminal capacitance network has the same connections in every selection. Only the coefficient datasets change.

The three extracted couplings in Figure 11 are:

Table 12: Extracted capacitances of the same plate layout

Selected scenario	C_{LU} [fF]	C_{LS} [fF]	C_{US} [fF]
`nominal-r-low-c`	203.108	305.838	162.210
`nominal`	301.336	382.940	205.944
`nominal-r-high-c`	523.375	515.064	277.637

C_{LU} connects `LOWER` ↔ `UPPER`, C_{LS} connects `LOWER` ↔ `sky130_gnd`, and C_{US} connects `UPPER` ↔ `sky130_gnd`. These are individual netlist capacitors, not diagonal entries of a capacitance matrix. Each result JSON records the selected component identities and provenance.

The different ratios between the columns show why one multiplier cannot reproduce these tables. To study a distributed RC layout, use `--mode RC` with the same explicitly declared scenarios; resistance then remains nominal while capacitance varies.

3.9.8 Varying GatPoly Sheet Resistance

Both `ihp-sg13g2` and `ihp-sg13cmos5l` provide `gatpoly-low-sheet` and `gatpoly-high-sheet`. They vary only the sheet resistance of extracted `GatPoly` interconnect. Every other layer, contact and via resistance, and all capacitance coefficients, remain explicitly nominal.

The values come directly from each process's maintained interconnect tables: 6.2, 7.0 and 7.8 Ω/□. The sources and hashes are recorded separately in the `ihp-sg13g2` low-sheet dataset and the `ihp-sg13cmos5l` low-sheet dataset.

These are fixed-layout sensitivity selections. They do not apply the thickness or linewidth variation of a full process corner, select a transistor corner, or change temperature. Their nominal stack provides layer identities and ordering; `physical_stack_matches: false` prevents treating it as matched field-solver geometry. These sheet-only scenarios select no device-reference component. An automatic device-capacitance comparison requires a separately justified explicit reference.

3.9.8.1 Comparing Strip Resistance

The example LVS database contains one 10 µm × 0.5 µm `GatPoly` strip. Ports `A` and `B` span opposite ends, with no contacts and no recognized resistor device. The same conductor and pin definitions apply to both IHP variants.

```
for corner in nominal gatpoly-low-sheet gatpoly-high-sheet; do
  kpex extract --pdk ihp-sg13g2 --2.5D --mode R \
    --corner "$corner" --lvsdb ihp-gatpoly-strip.lvsdb \
    --out-dir "output/$corner" \
    --out-netlist "output/$corner.spice"
done
```

For this rectangle, the independent reference is

$$R_{AB} = R_{\square}\frac{L}{W} = 20R_{\square}.$$

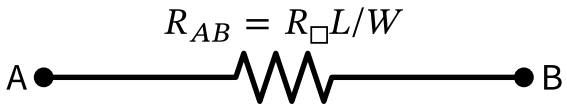


Figure 12: Two-terminal equivalent of the fixed strip. Its geometry and ports remain the same for all three sheet-resistance selections.

Table 13: Terminal resistance of the same fixed strip

Selected scenario	Sheet resistance [Ω/□]	Extracted resistance `A` ↔ `B` [Ω]
`gatpoly-low-sheet`	6.2	124
`nominal`	7.0	140
`gatpoly-high-sheet`	7.8	156

Repeating the commands with `--pdk ihp-sg13cmos5l` gives the same values. Circuit simulations in `ngspice`, `Xyce` and `VACASK` agree with the sheet law, including strips at the published 0.13 µm interconnect width and the next layout-grid step.

3.9.8.2 Keeping Compact Resistors Separate

A recognized `rsil`, `rppd` or `rhigh` resistor keeps its compact model, terminals and dimensions. Its modeled body is excluded from extracted interconnect. The `rsil` model has its own sheet parameters, effective-width correction and contact terms; applying those again to every drawn `GatPoly` polygon would mix two different resistance models.

For this reason, the selectable wire limits are 6.2 and 7.8 Ω/□ from the interconnect tables, rather than the compact model's 6.02 and 7.98 Ω/□. No compact-model width shift is applied to the strip. Selecting one of these sensitivities does not select a compact-resistor simulation corner.

3.9.9 Varying CMOS5L Resistance at Fixed Geometry

The `ihp-sg13cmos5l` catalog provides `fixed-layout-low-r` and `fixed-layout-high-r`. They select complete resistance tables for the existing extracted interconnect layers, contacts and vias, while retaining nominal capacitance and the drawn geometry. Select a scenario with the usual extraction options:

```
kpex extract --pdk ihp-sg13cmos5l --2.5D --mode RC \
  --corner fixed-layout-low-r --gds design.gds \
  --out-netlist design-low-r.spice --out-result design-low-r.json
```

Use `fixed-layout-high-r` for the corresponding high-resistance extraction, or `nominal` for the unchanged default. The narrower `gatpoly-low-sheet` and `gatpoly-high-sheet` scenarios vary only `GatPoly`; all their other resistance rows remain nominal.

3.9.9.1 Applying Sheet and Contact Laws

A rectangular wire follows $R = R_{\square} L/W$. Each layer has its own sheet resistance; the low/high scenarios therefore do not multiply the entire network by one common factor.

Table 14: `ihp-sg13cmos5l` sheet resistance in $\Omega/\square$

Conductor	Low	Nominal	High
`GatPoly`	6.2	7.0	7.8
`Metal1`	0.085	0.110	0.135
`Metal2`, `Metal3`, `Metal4`	0.073	0.088	0.103
`TopMetal1`	0.015	0.018	0.021

A contact or via contributes its per-cut resistance. With equipotential terminals on both sides, N identical cuts in parallel give $R = R_{\text{cut}}/N$. A real contact array also has access and spreading resistance in its conductors, which the resistance extraction retains.

Table 15: `ihp-sg13cmos5l` contact and via resistance in Ω/cut

Contact or via	Low	Nominal	High
`Cont` on `nSD` or `pSD`	8	17	22
`Cont` on `GatPoly`	8	15	20
`Via1`, `Via2`, `Via3`	5	9	20
`TopVia1`	1	2.2	4

These rows come from the variant's maintained minimum, typical and maximum interconnect tables. In particular, those sources place `TopVia1` between `Metal4` and `TopMetal1` and specify its 0.42 μm square cut directly. This resolves the row left unspecified in the variant's process-specification contact table.

3.9.9.2 Interpreting Resistance Sensitivity

The selection changes the interconnect material laws applied to the existing polygons. Compact-device models retain their own internal resistances and geometry rules. Diffusion regions remain ideal device-terminal endpoints under the existing extraction boundary.

The source interconnect files also vary physical parameters, including `GatPoly` thickness. The fixed-layout resistance selections do not apply those geometry changes, temperature variation or compact-model width corrections. They use nominal analytical capacitance and declare no matching field-solver stack or automatic device-capacitance references, as explained in Section 3.9.12.

The low-R dataset and high-R dataset record the exact source revisions, hashes and assumptions. Their source-defined resistance bounds are suitable for sensitivity studies; combining them with nonnominal capacitance requires an explicit scenario and a justified interpretation of that combination.

3.9.10 Varying `sky130A` Resistance

The `sky130A` catalog provides four fixed-layout resistance sensitivities. Each selects a complete resistance table and the corresponding `low` or `high` capacitance table. `nominal` remains the default.

Table 16: Complete resistance selections and their MIM-bound policy

Selected scenario	C dataset	MIM sheet resistance [Ω/□]
`nominal`	`nominal`	5.8
`hrhc-mim-bound`	`high`	6.8
`hrlc-mim-bound`	`low`	6.8
`lrhc-mim-bound`	`high`	4.8
`lrlc-mim-bound`	`low`	4.8

The four resistance datasets retain differences that one generic high-R and one generic low-R table would lose. The source parameters `rnw`, `rcvia3` and `rcvia4` map to the existing `nwell`, `via3` and `via4` resistance rows. The two via values also apply to their MIM contact aliases, `via3_cap` and `via4_cap`.

Table 17: Resistance overrides retained from each maintained R/C model

R/C source style	`nwell` [Ω/□]	`via3` [Ω/cut]	`via4` [Ω/cut]
`nominal`	1700	3.41	0.38
`hrhc`	2022	6.623	0.7377
`hrlc`	2160	8	0.891
`lrhc`	1240	0.5	0.012
`lrlc`	1378	1.373	0.1224

Every dataset explicitly supplies all ten layer, four contact and five via rows. Missing values are errors; they do not inherit nominal resistance. Other sheet and cut values follow the reviewed extraction tables, with their units and source hashes recorded in the dataset provenance.

3.9.10.1 Understanding MIM Bounds

The published MIM specification gives `RSCAPM` as 5.8 Ω/□, bounded by 4.8 and 6.8 Ω/□, for both MIM top electrodes. These are the `capm` and `capm2` sheet-resistance rows.

The `-mim-bound` selections pair the upper limit with high-R styles and the lower limit with low-R styles. This is an explicit engineering sensitivity policy. The specification does not establish that these limits occur together with the selected R/C model style. The component provenance records that assumption separately from the imported model parameters.

Keep the drawn polygons and contact count fixed when interpreting the result. For a rectangular sheet and N identical contacts in parallel, the material laws are

$$R_{\text{sheet}} = R_{\square}\frac{L}{W}, \qquad R_{\text{contacts}} = \frac{R_{\text{cut}}}{N}.$$

No compact-model linewidth shift is applied to the extracted sheet, and the actual cuts determine N. The device models for `capm` and `capm2` have their own effective-width and effective-length corrections and estimated contact counts. Applying those corrections again to the extraction geometry would change the meaning of this fixed-layout comparison.

These selections support analytical extraction with `--2.5D`. They do not specify a combined geometry/material worst case, a transistor corner, temperature, or a probability of the selected values occurring together. Their nominal stack supplies compatible layer identities and ordering; `physical_stack_matches: false` prevents exporting it as matched corner geometry. Automatic corner-specific device-capacitance checks are unavailable unless an independent reference is supplied explicitly.

3.9.10.2 Comparing Via Stacks

The example LVS database contains one cut at each connection from `poly` to `met5`, with ports at the cut centres and no intervening wire length. Download the `sky130A` catalog bundle and keep the layout unchanged for all five selections.

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_via_stack_1x1_minsize_poly_to_met5.gds.gz

```
unzip sky130A.zip
for corner in nominal hrhc-mim-bound hrlc-mim-bound \
    lrhc-mim-bound lrlc-mim-bound; do
  kpex extract --pdk sky130A --2.5D --mode R \
    --tech-catalog sky130A/catalog.pb.json --corner "$corner" \
    --lvsdb sky130-via-stack.lvsdb.gz \
    --out-dir "output/$corner" \
    --out-netlist "output/$corner.spice" \
    --out-result "output/$corner.json"
done
```

With `hrhc-mim-bound`, the written netlist contains these six physical-cut resistors:

```
Rext_2 li1 met1 23
Rext_3 li1 poly 252
Rext_4 met1 met2 15
Rext_5 met2 met3 8
Rext_6 met3 met4 6.623
Rext_7 met4 met5 0.7377
```

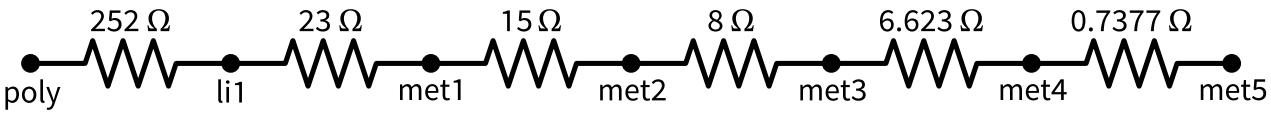


Figure 13: Physical cut resistances for `hrhc-mim-bound`, with the seven exposed netlist ports.

The full netlist also includes a dangling 0.001 Ω tie from `li1` to the original LVS net name, omitted from the excerpt and Figure 13. With the intermediate ports unloaded, this branch draws no current and the six physical resistances add in series:

Table 18: Terminal resistance of the same extracted via stack

Selected scenario and written netlist	Resistance `poly` ↔ `met5` [Ω]
`nominal`	173.0000
`hrhc-mim-bound`	305.3607
`hrlc-mim-bound`	306.8910
`lrhc-mim-bound`	56.6120
`lrlc-mim-bound`	57.5954

The distinct high-R and low-R results come from `via3` and `via4`, which depend on both R and C choices. This via stack demonstrates the source-dependent cut values; it does not establish correlation for MIM sheet bounds.

3.9.11 Varying `gf180mcuD` Resistance

The `gf180mcuD` catalog provides three resistance sensitivities with explicit source assumptions for the existing layout. They retain nominal capacitance and geometry. The default `nominal` scenario remains unchanged.

- `fixed-layout-reference-r` uses a `FuseTop` sheet resistance of 180 mΩ/□
- `fixed-layout-high-r` uses the high sheet and contact values, including 207 mΩ/□ for `FuseTop`
- `fixed-layout-low-r-ideal-contacts` uses low sheet values, including 153 mΩ/□ for `FuseTop`, with an explicitly assumed ideal contact/via connection

The low and high plate values bracket the 180 mΩ/□ **reference**, not the default nominal value of 500 mΩ/□. The scenario names keep that distinction visible when comparing results.

3.9.11.1 Separating Plate Sources

The maintained interconnect sheet-resistance table specifies $0.18 \pm 0.027\,\Omega/\square$ for the MIM plate.

The named stack is `TiN600AMIMAL2000ATiN350A`. Its 0.295 μm thickness agrees with the conductor table.

The maintainer's interpretation supports this material-stack value and assumes that the density options share the plate construction. Applying it to the selected five-metal, 2 fF/µm² `MIM-B` option is an engineering assumption: the option's electrical table does not specify plate resistance.

`KPEX` retains 500 mΩ/□ in `nominal` because that was its established estimate from the 1/1.5 fF specifications. Selecting the new reference is deliberate; installing a newer version does not change the default estimate.

3.9.11.2 Applying Sheet and Contact Values

The conductor values below use $R = R_{\square} L/W$ at unchanged drawn dimensions. The electrical sheet-resistance specification supplies the conductor limits other than the `FuseTop` plate values discussed above. `Metal5` uses the selected thick `11kA` top-metal row, not the ordinary fifth-metal row.

Table 19: Fixed-layout `gf180mcuD` sheet sensitivity in Ω/□

Conductor	Low	Reference	High
`Poly2`	1.0	7.3	15.0
`Metal1`, `Metal2`, `Metal3`, `Metal4`	0.076	0.090	0.104
`Metal5`	0.031	0.040	0.049
`Nwell`	800	1000	1200
`FuseTop`	0.153	0.180	0.207

Each cut contributes its per-cut resistance. With equipotential landing pads, N identical cuts in parallel give $R = R_{\text{cut}}/N$. The extraction also retains conductor access and spreading resistance for real layouts.

Table 20: Fixed-layout `gf180mcuD` contact sensitivity in Ω/cut

Contact or via	Ideal-contact assumption	Reference	High
`Contact` on `Nplus` or `Nwell`	0	6.3	15
`Contact` on `Pplus`	0	5.2	15
`Contact` on `Poly2`	0	5.9	15
`Via1`, `Via2`, `Via3`, `Via4`	0	4.5	15

The primary contact specification writes `--` for all minimum values. That means **unspecified**, not zero. The separately named ideal-contact scenario follows the explicit zeros in the maintained `MAGIC` low-R extraction block. It is not a measured physical lower limit. The reference and high datasets record only the primary nonzero values.

The `Poly2` reference contact remains 5.9 Ω/cut. The contact from `Nwell` uses the established `Nplus` contact mapping, and the MIM top landing uses the `Via4` per-cut law. With `--blackbox y`, recognized compact devices retain their own models and terminal boundaries (Section 2.3).

3.9.11.3 Selecting the Sensitivity

```
kpex extract --pdk gf180mcuD --2.5D --mode RC --blackbox y \
  --corner fixed-layout-reference-r --gds design.gds \
  --out-netlist design-reference-r.spice
```

Select `fixed-layout-high-r` or `fixed-layout-low-r-ideal-contacts` explicitly for the other comparisons. Do not interpret the default `nominal` result as the midpoint of this plate-resistance range.

The ideal-contact low-R dataset, reference dataset, and high-R dataset record the exact source revisions, hashes and assumptions.

3.9.11.4 Limits

These are fixed-layout sensitivities. They do not establish a foundry-qualified 2 fF plate law, correlated R/C corners, simultaneous manufacturing extrema, width or temperature corrections, or statistical coverage. They select no automatic device-capacitance reference and declare no matching field-solver stack. Combining them with nonnominal capacitance needs a separately declared scenario and a justified interpretation.

An ideal contact merges its electrical nodes. Separately named SPICE ports on the same node retain the existing 1 mΩ label tie. That export tie is not a physical contact value and does not divide by the number of cuts.

3.9.12 Matching Physical Geometry to Analytical Tables

Analytical extraction needs conductor ordering and connectivity from the stack, while its numerical capacitance comes from tables. A table-only corner can therefore use a nominal stack explicitly for structural purposes.

A field solver instead calculates capacitance from physical geometry and materials. Exporting nominal geometry under a nonnominal label would not reproduce a table corner. A scenario consequently declares whether its selected physical stack matches the scenario.

Table 21: Capabilities checked before layout loading

Requested operation	Required evidence
`--2.5D` extraction	Complete R/C datasets and compatible conductor identities
`kpex pex25d`	A declared matching physical stack
`--fastcap` or `--fastercap`	A declared matching physical stack
Automatic `--check-device-caps`	A device-reference dataset for the scenario

A missing capability produces a diagnostic before extraction. Device checks can instead use an explicitly supplied reference JSON file. Its physical assumptions remain the user's choice. A nominal reference is never inherited automatically by a scenario that declares none.

The matching-stack declaration records a dataset contract, not foundry certification. It is checked before layout loading; availability of individual physical profiles is checked against the extracted shapes afterward. A numerical recipe may vary permittivity, thickness or manufactured width. Those choices require their own sources and must not be inferred from the spelling of a corner name.

3.9.12.1 Selecting Source Recipes

The development catalogs provide `capiche-low-c` and `capiche-high-c` for `sky130A`, `gf180mcuD`, `ihp-sg13g2` and `ihp-sg13cmos5l`. These select the physical interconnect recipes from the pinned Capiche sources, together with the corresponding empirical C tables.

Table 22: Source-related physical selections, available in each of the four catalogs

Selected scenario	R dataset	C dataset	Physical component
`capiche-low-c`	`nominal`	`low`	`capiche-low`
`capiche-high-c`	`nominal`	`high`	`capiche-high`

The source mapping relates the physical recipes to the extraction styles. The catalog records the separate source revisions for the physical recipe and empirical coefficients. This relationship does not make a finite 3D solve numerically identical to a fitted table, establish a correlated resistance corner, or constitute silicon calibration.

The R dataset deliberately stays `nominal`. These selections declare matching physical geometry within the supported profile scope, but no automatic device-capacitance reference. Unsupported MIM profiles are checked against the actual layout, as described in Section 3.9.12.5.

Table 23: Physical components with their source records and explicit assumptions

Technology	Low physical component	High physical component
`sky130A`	`capiche-low`	`capiche-high`
`gf180mcuD`	`capiche-low`	`capiche-high`
`ihp-sg13g2`	`capiche-low`	`capiche-high`
`ihp-sg13cmos5l`	`capiche-low`	`capiche-high`

A source material named `air` is not necessarily vacuum. `kpex pex25d` preserves its declared permittivity. `kpex extract --fastercap` and `--fastcap` use `--k-void` instead, defaulting to 3.9. To retain the source background, select the value in Table 24.

Table 24: Source background permittivity, distinct from the extraction command's default

Technology	`capiche-low-c` background k	`capiche-high-c` background k
`sky130A`	2.94	3.5
`gf180mcuD`	3.0	3.0
`ihp-sg13g2`	3.0	3.0
`ihp-sg13cmos5l`	3.0	3.0

Both extraction options use the shared `PEX25DFile`, `PEX25DScene` and scene exporter. Their electrostatic scope retains extracted diffusion and tap solids; `kpex pex25d` defaults to interconnect with device-terminal anchors. Background and domain settings can differ too. Compare exporters using the same prepared scene and solver settings.

The source substrate is an ideal 2D ground surface at $z = 0$. The 3D recipes retain a finite ground thickness as numerical truncation. `sky130A` also retains a 0.1 µm clearance below that surface, so a full ground plane does not touch device contacts beginning at $z = 0$. Its exported ground top is therefore $z = -0.1$ µm. The `gf180mcuD` and `ihp-sg13*` recipes retain their existing zero clearance. The component provenance records this 3D boundary policy separately from the source geometry.

3.9.12.2 Applying Manufactured Width

`MetalLayer.line_width_bias` is the signed **total** change from drawn to manufactured width, in micrometres. If the drawn width is W and the bias is b,

$$W_{\text{physical}} = W + b, \qquad \Delta_{\text{edge}} = \frac{b}{2}.$$

Positive b grows metal; negative b erodes it. Every contour edge follows the same law, including line ends and holes. Growing metal reduces a hole; eroding metal enlarges it. Contact cuts retain their drawn shapes and positions.

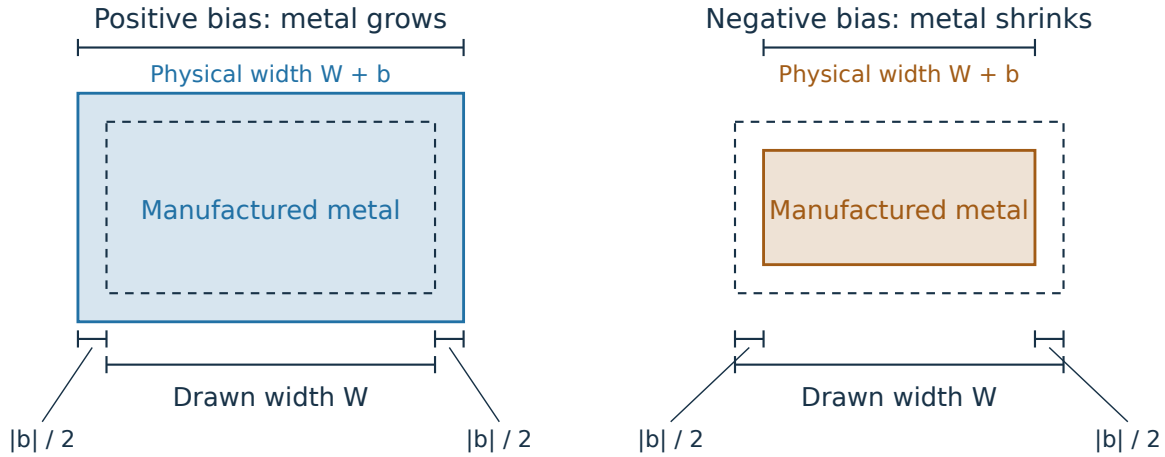


Figure 14: Signed full-width bias moves both boundaries. The drawing is schematic; the displacement is exaggerated for clarity.

The physical grid must represent the half-bias exactly. For example, the `sky130A met1` change of −0.0206 µm moves each boundary inward by 0.0103 µm: 103 units on a 0.0001 µm grid. A half-bias between grid points is rejected; `kpex pex25d --grid` can select a finer grid that also divides the layout DBU. Non-Manhattan polygon vertices are snapped to that grid, so their geometric displacement has a grid-sized discretization error.

The geometry must retain its electrical connections. Export stops if a bias:

- Removes a conductor or separates an originally connected polygon
- Breaks a contact, terminal or connection between touching conductor profiles
- Joins different conductors, including a new boundary touch
- Exceeds the native coordinate range

Touching profiles can require a joint physical recipe. Independently specified offsets that would open an internal seam are rejected, even when each profile remains nonempty. This avoids silently converting one conductor into disconnected pieces. Zero bias preserves drawn geometry.

These offsets apply to physical export. They do not resize polygons for analytical `--2.5D` extraction or replace its selected R/C coefficient tables.

3.9.12.3 Exporting the Same Layout

The example has two drawn 100 µm square plates: `li1` from $(0, 0)$ to $(100, 100)$ µm, and `met1` from $(50, 50)$ to $(150, 150)$ µm. Their drawn overlap is 50 µm square. Download the LVS database and the `sky130A` catalog bundle.

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/overlap_plates_100um_x_100um_li1_m1.gds.gz

```
mkdir -p catalogs
unzip sky130A.zip -d catalogs
for corner in capiche-low-c capiche-high-c; do
  kpex pex25d --pdk sky130A \
    --tech-catalog catalogs/sky130A/catalog.pb.json \
    --corner "$corner" --lvsdb overlap.lvsdb.gz \
    --grid 0.0001 --out-dir "output/$corner" \
    --out-file "output/$corner.pex25d" \
    --out-scene "output/$corner.pex25d.scene.pb"
done
```

The generated low-C file contains these layer extents:

```
METAL li1 Z_OFFSETS 1.177 1.257
METAL met1 Z_OFFSETS 1.7295 2.0595
```

The high-C file instead contains:

```
METAL li1 Z_OFFSETS 0.6917 0.8067
METAL met1 Z_OFFSETS 1.0137 1.4037
```

Table 25: Dimensions read from the two generated physical files

Selected scenario	`li1` width [µm]	`met1` width [µm]	Vertical gap [µm]
`capiche-low-c`	99.9770	99.9794	0.4725
`capiche-high-c`	100.0230	100.0206	0.2070

The gap is the `met1` bottom minus the `li1` top. Both the XY boundaries and the stack heights change; one global capacitance multiplier cannot describe this geometry. The resolved low-C scene and high-C scene are ready for a `PEX25D` exporter.

3.9.12.4 Comparing Finite Plates

A representative numerical check uses two aligned 10 µm square plates on the first and second routing-metal layers. It applies the selected manufactured widths, stack heights and dielectric constants, with an 8 µm field margin. The mutual capacitance is the average magnitude of the two off-diagonal matrix entries; substrate coupling is a different observable.

The independent area-only estimate is

$$C_{\text{area}} = \epsilon_0 \epsilon_r \frac{W^2_{\text{physical}}}{g},$$

where g is the selected dielectric gap. Finite plates also have edge coupling, which this area law omits. Comparing the two exposes mistakes in units, gap, width or permittivity without treating the area law as an exact finite-geometry solution.

From a source checkout containing these recipes, the recorded cases can be reproduced with:

```
poetry run pytest tests/physical_corner_data_test.py::\
test_physical_corner_parallel_plates_converge_and_preserve_expected_order \
  -q -s
```

The recorded results and solver logs use `KLayout` module `0.30.13` and `FasterCap 6.2.1`. Each case was solved at tolerances `0.02` and `0.005`; the 3D column gives the latter result.

Table 26: Representative finite-plate checks of the physical implementation

Technology	Recipe	Area law [fF]	3D mutual [fF]	Refinement [%]
`gf180mcuD`	`low`	4.925835	5.465495	0.005
`gf180mcuD`	`high`	7.363150	8.017195	0.313
`ihp-sg13g2`	`low`	6.050362	6.574805	0.048
`ihp-sg13g2`	`high`	7.394886	8.052925	-0.027
`ihp-sg13cmos5l`	`low`	6.050362	6.586825	0.178
`ihp-sg13cmos5l`	`high`	7.394886	8.064470	0.160
`sky130A`	`low`	9.227874	9.922760	0.122
`sky130A`	`high`	34.754573	35.701250	0.004

The finite-plate results exceed the area estimates by 2.724–10.956%. Tightening the solver tolerance changes the mutual capacitance by at most 0.313% in magnitude. The `high` recipe gives a larger capacitance than `low` for every tested technology.

This validates representative physical dimensions, material assignment and numerical behavior. It does not establish agreement with the empirical analytical tables, qualify arbitrary layouts or provide silicon calibration. A fitted table also contains its own geometry conventions and finite-length assumptions.

3.9.12.5 Checking Available Profiles

A physical recipe may describe interconnect without supplying a qualified MIM corner law. A nominal MIM plate height or thickness retained for layer ordering is insufficient evidence for a nonnominal 3D model. `MetalLayer.physical_geometry_unavailable` makes that distinction explicit.

The source recipes mark these conductor profiles unavailable:

Table 27: Layout-dependent restrictions in both source recipes

Technology	Unavailable physical conductor profiles
`sky130A`	`met3_cap`, `capm`, `met4_cap`, `capm2`
`gf180mcuD`	`metal4_cap`, `FuseTop`
`ihp-sg13g2`	`metal5_cap`, `cmim_top`
`ihp-sg13cmos5l`	No profiles flagged unavailable in this recipe

A layout containing an unavailable profile **or a contact touching it** is rejected before physical output is generated. If the layout contains neither, the profile, associated contacts and dependent conformal coatings are omitted from the exported physical model. A missing layout-layer mapping also fails, because absence could not be verified.

For example, download the MIM LVS database:

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/cap_mim_m3_20um_x_20um_over_m2_wire.gds.gz

```
kpex pex25d --pdk sky130A \
  --tech-catalog catalogs/sky130A/catalog.pb.json \
  --corner capiche-low-c --lvsdb mim.lvsdb.gz --blackbox n \
  --out-dir output/mim --out-file output/mim.pex25d
```

The run rejects the unsupported plate:

```
Could not generate PEX25D: Selected physical recipe has no
qualified geometry for 'capm', present on net 'B'
```

No `output/mim.pex25d` is produced. Selecting `nominal` is a separate comparison using the existing nominal geometry, not a substitute for the missing MIM corner. The missing evidence is a source-backed combination of MIM plate geometry, surrounding dielectric geometry and relevant dimensional changes for that corner.

3.9.12.6 Reading Coating Thicknesses

A conformal coating can wrap a metal or another coating. Its top and side thicknesses are **incremental** from the wrapped profile. Its field thickness is the **total height from the bottom of the root metal**, including any inner field coating. The latter is not added again along the chain.

The `ihp-sg13cmos5l` recipe illustrates the distinction. `pass1` wraps `TopMetal1`; `pass2` wraps `pass1`. Both reference the bottom of `TopMetal1` for their field extents.

Table 28: Coating geometry shared by the low-C and high-C `ihp-sg13cmos5l` recipes

Coating	Wraps	Top increment [µm]	Side increment [µm]	Field extent [µm]
`pass1`	`TopMetal1`	1.5	0.3	1.5
`pass2`	`pass1`	0.4	0.3	1.9

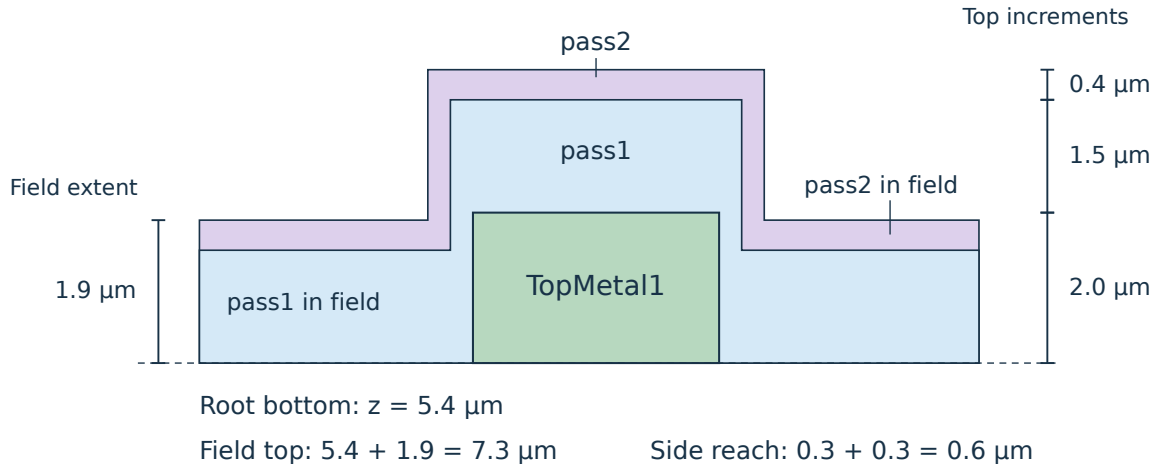


Figure 15: Incremental top and side offsets versus cumulative field height in the `ihp-sg13cmos5l` recipe. Vertical dimensions follow the source; the wire width is illustrative.

`TopMetal1` starts at $z = 5.4$ µm and is 2.0 µm thick. Over metal, the outer top is therefore $5.4 + 2.0 + 1.5 + 0.4 = 9.3$ µm. Beside metal, the outer reach is $0.3 + 0.3 = 0.6$ µm. In the field, the outer height is $5.4 + 1.9 = 7.3$ µm.

Material precedence leaves `pass1` in the lower 1.5 µm of the field and `pass2` in the next 0.4 µm. Setting the `pass2` field value to only 0.4 µm would hide it entirely beneath `pass1`. This error can be present even when no `TopMetal1` shapes occur in the layout.

The low/high choices vary the relative permittivities of these coatings: `pass1` uses 3.69/4.51 and `pass2` uses 5.94/7.26. Their geometry above is unchanged. These values come from the low and high source recipes. Section 11.1.5.5 describes the corresponding `WRAPS` and material-precedence rules.

3.9.13 Selecting Device-Capacitance References

A capacitance component describes the extracted interconnect. A device-reference component supplies independent model expressions for `--check-device-caps` (Section 8.4). Selecting one does not select the other. The catalog must name the intended reference explicitly.

The additional reference components cover these MIM models:

Table 29: Explicit source-backed MIM reference components

Technology	Supported device	Reference component IDs
`sky130A`	Both mapped MIM plate options	`hrhc`, `hrlc`, `lrhc`, `lrlc`
`gf180mcuD`	`cap_mim_2f0fF`	`low`, `high`
`ihp-sg13g2`	`cap_cmim`	`low`, `high`
`ihp-sg13cmos5l`	No additional MIM reference component	No additional component

For `sky130A`, the two mapped devices are:

- `sky130_fd_pr__model__cap_mim`
- `sky130_fd_pr__model__cap_mim_m4`

The references describe low-frequency mutual capacitance at zero bias and zero mismatch. The model reference temperatures are 27 °C for `sky130A` and `ihp-sg13g2`, and 25 °C for `gf180mcuD`. Series resistance, leakage and substrate shunts are separate observables.

The source equations are checked against pinned compact-model simulations. For a two-terminal device, the mutual capacitance follows the off-diagonal admittance:

$$C_{AB} = -\frac{\operatorname{Im} Y_{AB}}{\omega}.$$

The comparison uses frequencies low enough that series resistance does not materially change this observable. It does not interpret leakage conductance as capacitance or compare the sum of all capacitances attached to a net.

3.9.13.1 Preserving the Model's Corner Law

Different models vary different terms:

- `sky130A`: the R/C style changes the effective dimensions as well as area and edge coefficients. Two styles with the same C suffix can therefore require different reference expressions
- `gf180mcuD`: the supported 2 fF MIM option scales both area and perimeter terms by its own MIM corner parameter
- `ihp-sg13g2`: `cap_cmim` varies its area density while retaining the literal edge density of 40 aF/μm. Its multiplier applies once to the complete device capacitance

For the IHP reference, with mapped dimensions expressed in micrometres,

$$C_{\text{reference}} = m\left[C_A W L + 2C_P(W + L)\right].$$

The `low` and `high` components use $C_A = 1.35$ and $1.65\,\text{fF}/\mu\text{m}^2$, respectively, while $C_P = 0.04\,\text{fF}/\mu\text{m}$ stays fixed. A global multiplier applied to nominal capacitance would incorrectly change that edge term too.

3.9.13.2 Comparing MIM Plates at Two Capacitance Selections

The complete MIM comparison catalog declares `mim-low-c` and `mim-high-c`. They pair the corresponding C tables and device-reference component while holding resistance nominal. This is an explicitly authored sensitivity comparison, without a claim of correlated resistance or matching field-solver geometry.

The low selection contains:

```
{
  "id": "mim-low-c",
  "resistance": "nominal",
  "capacitance": "low",
  "process_stack": "nominal",
  "device_references": "low",
  "physical_stack_matches": false
}
```

The high selection names `high` for both capacitance and device references. Neither selection inherits the nominal MOM references.

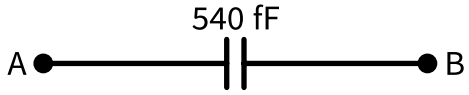
GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/cap_cmim_20um_x_20um_over_m4_wire.gds.gz

The layout contains a 20 µm × 20 µm `cap_cmim` with terminals `A` ↔ `B`, above a separate `Metal4` route. Set `KPEX_DIR` to the source checkout, then run:

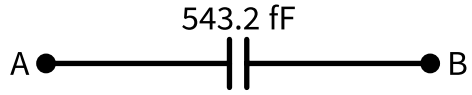
```
unzip ihp-mim-reference.zip
pattern=cap_cmim_20um_x_20um_over_m4_wire
layouts="$KPEX_DIR/testdata/designs/ihp-sg13g2/test_patterns"
lvsdbs="$KPEX_DIR/testdata/lvsdb/ihp-sg13g2/test_patterns"
for corner in mim-low-c mim-high-c; do
  kpex extract --pdk ihp-sg13g2 --2.5D --mode CC \
    --blackbox y --halo 8 --scale y \
    --gds "$layouts/$pattern.gds.gz" \
    --lvsdb "$lvsdbs/$pattern.lvsdb.gz" \
    --tech-catalog ihp-mim-reference/catalog.pb.json \
    --corner "$corner" --check-device-caps \
    --out-dir "output/$corner" \
    --out-result "output/$corner.json"
done
```

Table 30: Recorded device-local comparisons of the same MIM layout

Selected scenario	Marker coupling [fF]	Model reference [fF]	Ratio
`mim-low-c`	540.0	543.2	0.994109
`mim-high-c`	660.0	663.2	0.995175



(a) Observed marker coupling at `mim-low-c`



(b) Independent model reference at `mim-low-c`

Figure 16: The two values describe the same terminal pair under the same selected C corner, but come from different calculations. The comparison does not add a capacitor to the exported netlist.

In this LVS definition, both device-owned plates stop at the MIM rectangle. The lower electrode's 2 µm overhang is ordinary `Metal5`, outside that ownership region. The marker observer therefore excludes fringe coupling to the overhang and reports area coupling only, while the model includes its 3.2 fF perimeter contribution.

The separate route's coupling is also outside this device-local comparison. Both results are `informational`: no acceptance bound was supplied. Their ratios compare different observation domains and must not be interpreted as qualification of the complete device capacitance.

3.9.13.3 Limits of Device-Corner References

The components cover only the listed MIM models. MOM devices, RF MIM models, other MIM density options and unlisted capacitor types remain unchecked. A layout with no supported references fails a requested automatic check with an explanation.

The existing `nominal` reference component is unchanged. In particular, the IHP nominal MOM engineering estimates remain available under their original assumptions; the new MIM components do not supply nonnominal MOM laws or automatically add nominal MIM checks.

The selected expressions reject nonpositive effective dimensions and invalid multiplicities. They do not replace PDK layout rules, qualify arbitrary bias or temperature, or define statistical correlations with transistor corners. A custom explicit reference file remains available when another independently justified comparison is needed.

3.9.14 Recording What Was Used

`kpex_result.json`, or the path selected by `--out-result`, records `technology_selection`:

- Technology and scenario identifiers
- Catalog path and SHA256 of its exact bytes
- Catalog description, structured provenance and data revision
- Selected component identifiers, content hashes, source records, assumptions and revision histories
- Hash of the complete selected input technology
- Whether a matching physical stack and device references were declared

Sources, source references and revision history are validated before using the selected data. A missing source identity, unknown source identifier, invalid calendar date or inconsistent current revision is an error.

Each component file is checked against the catalog's byte hash. Changing or reformatting a file while retaining an old byte hash is detected. A byte-identical copy retains the same hash. The resolved content identities are separate: irrelevant JSON object-key order does not change the identity, while physically significant stack and terminal order is retained.

`technology_selection.resolved_technology_sha256` identifies the selected input before runtime overrides. An extraction with `--halo 3` records its effective halo separately as `halo_um`, and preserves the override in `arguments`. Keep the complete result with the netlist rather than using a scenario label alone as a reproducibility key.

`PEX25D` metadata records the selected `process_corner`. Physical-stack identity remains distinct from the capacitance-table identity, which matters when comparing geometry generation with fitted or imported coefficients.

3.9.15 Limits

- Explicit `--corner` or `--tech-catalog` cannot be combined with `--magic`, whose installed technology and extraction style are selected independently
- A different conductor topology or layout/LVS interface needs a separate technology catalog
- No voltage, temperature, statistical variation or transistor-model selection is inferred
- An analytical corner without a matching physical recipe cannot be exported as corner-matched solver geometry
- Only the scenarios declared by the supplied catalog can be selected

4 KPEX/FasterCap Engine

FasterCap is a 3D and 2D parallel capacitance field solver, inspired by FastCap2. The maintained fork is iic-jku/FasterCap.

Starting from an input layout (e.g. GDS file) and a process stack-up (part of the Section 3.3), KPEX creates input geometries for FasterCap. After running FasterCap, the Maxwell capacitance matrix is parsed and interpreted to obtain the parasitic capacitances.

See Section 2.5 to get started with a first extraction example.

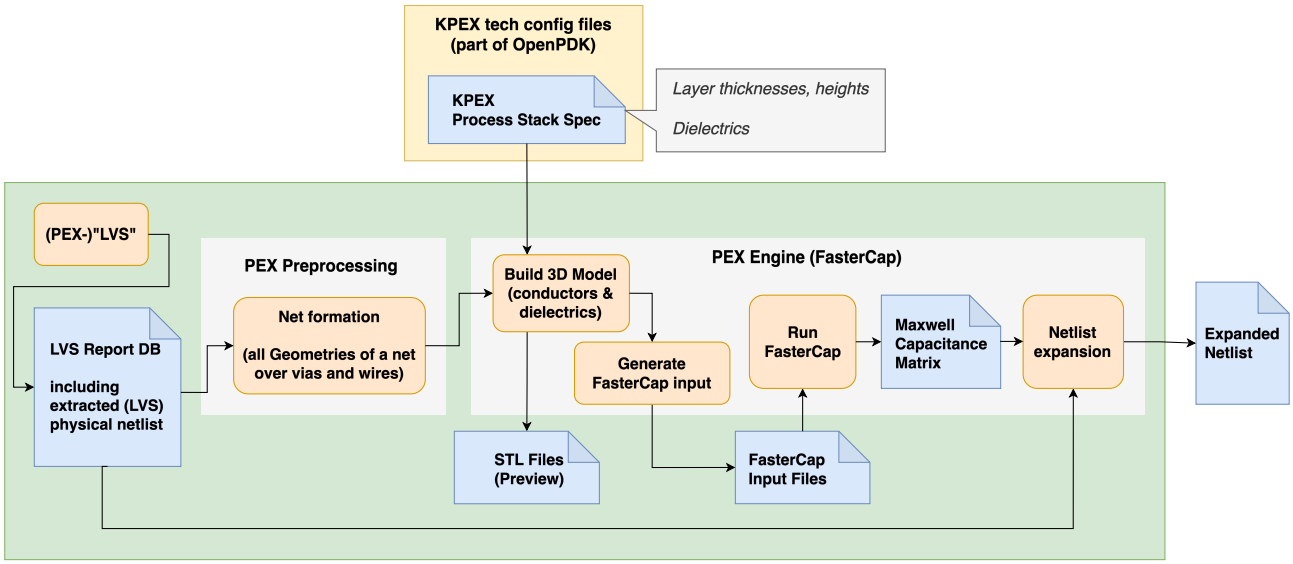


Figure 17: KPEX/FasterCap Engine

4.1 3D Input Geometries

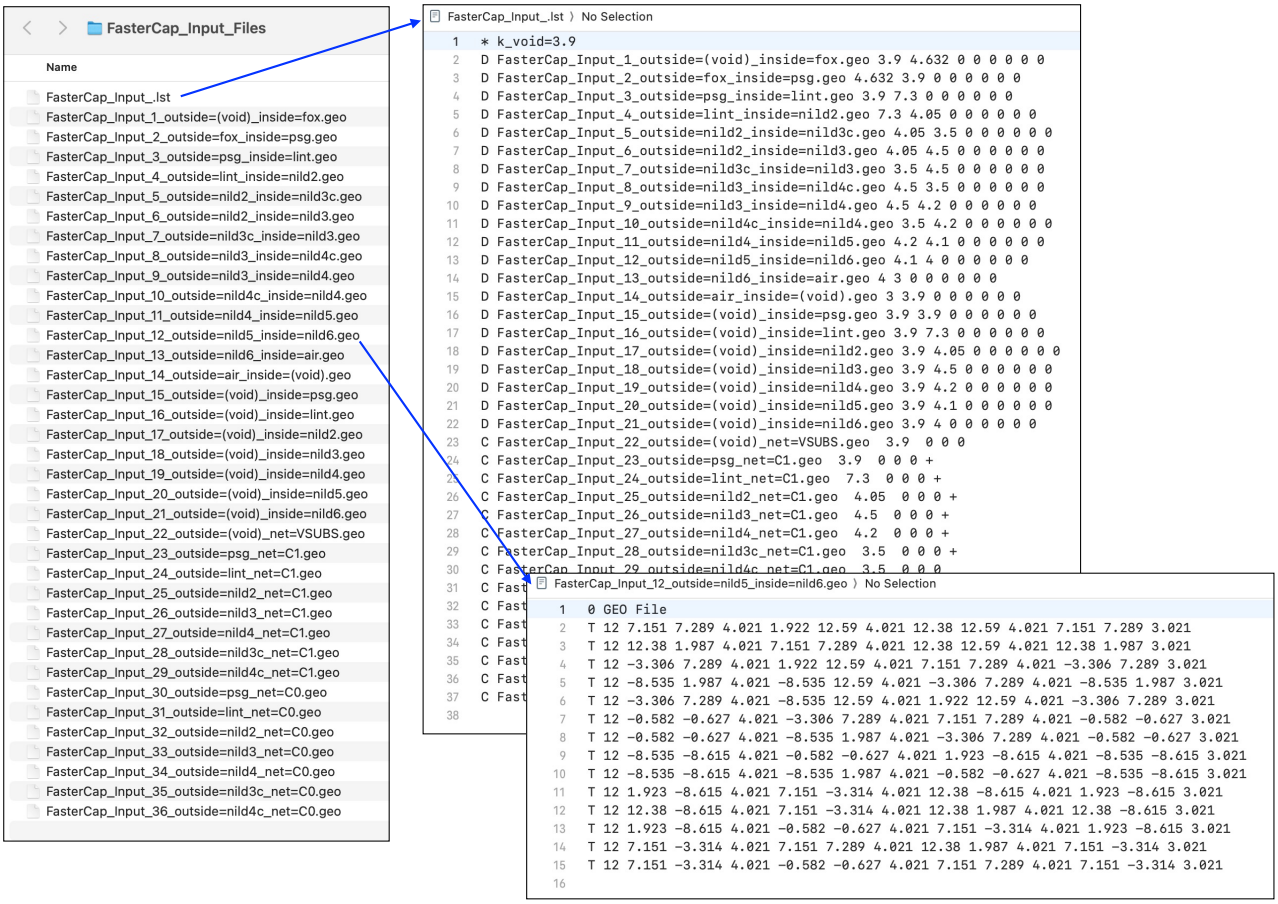


Figure 18: `FasterCap` 3D Input: File System Overview

The `FasterCap` input files and their format are documented in the PDF manual (Di Lorenzo 2019).

`KPEX` generates 3D input geometries:

- `*.lst` file: Main input file
  - defines dielectric instances
  - defines conductor instances
  - each instance refers to a `*.geo` file
- `*.geo` files: Defines single geometry
  - defines shapes (e.g. triangles)
  - Each shape has a reference point to define inside/outsides

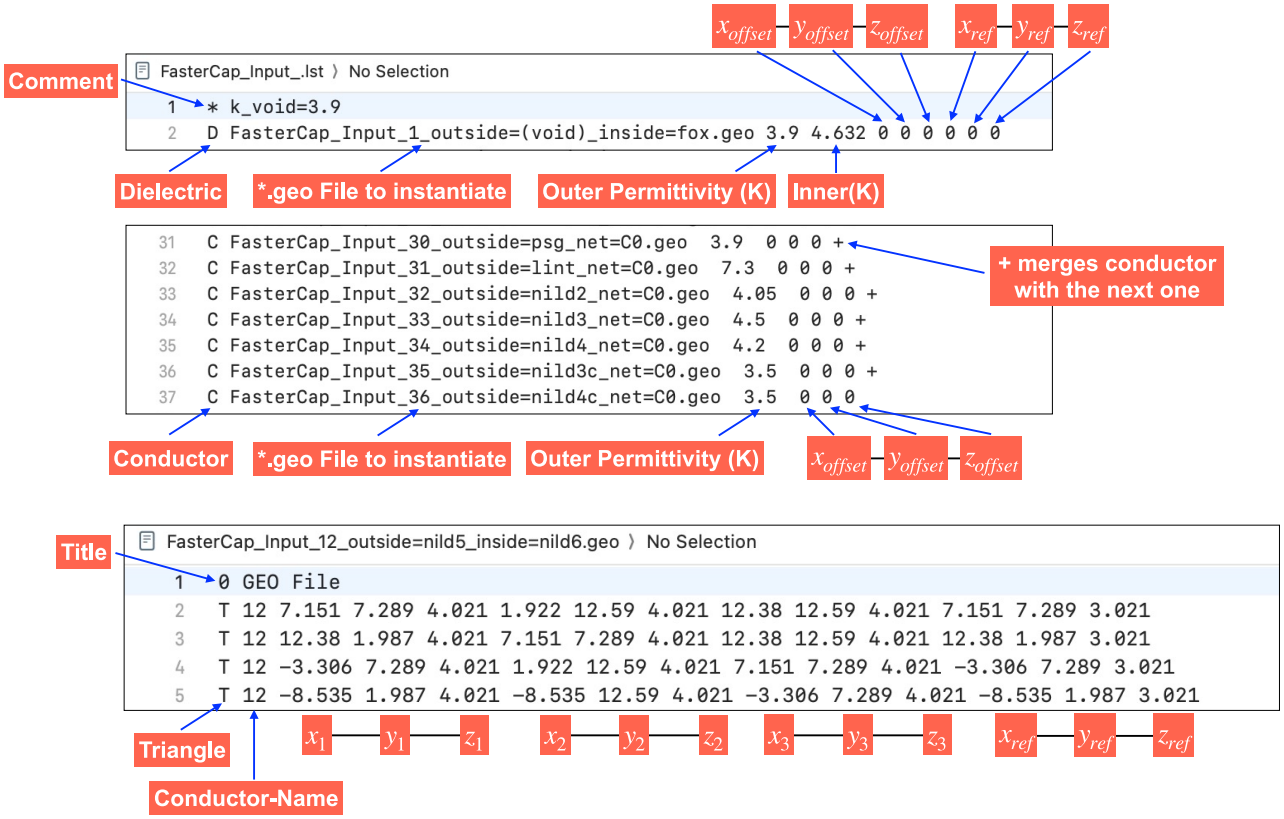


Figure 19: `FasterCap` 3D Input: File Format

4.2 Example: MOM Capacitor

Figure 20 depicts the MOM capacitor from Section 2.5.

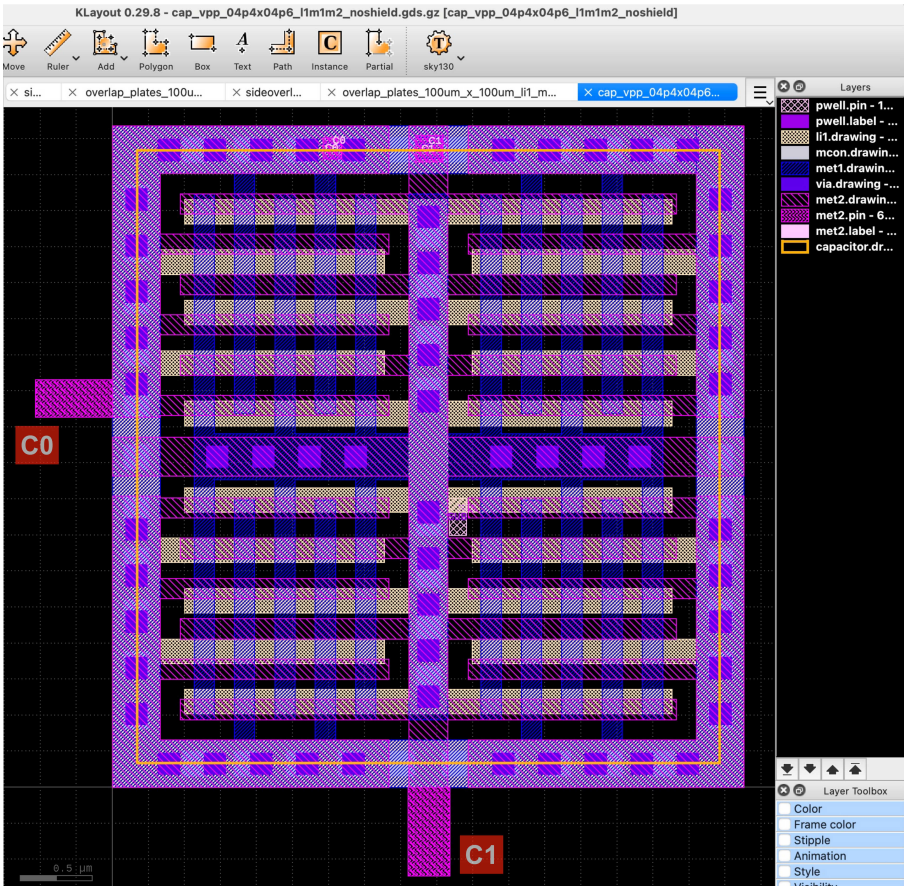
The corresponding schematic representation of Figure 21 contains 3 conductors (N_1, N_2 and N_3), and coupling capacitances:

- Capacitances between conductors: C_{ij} where $i \neq j$
  - C_{23} is the capacitance "intended" by the MOM designer
- Capacitances between conductors and ground: C_{ii}

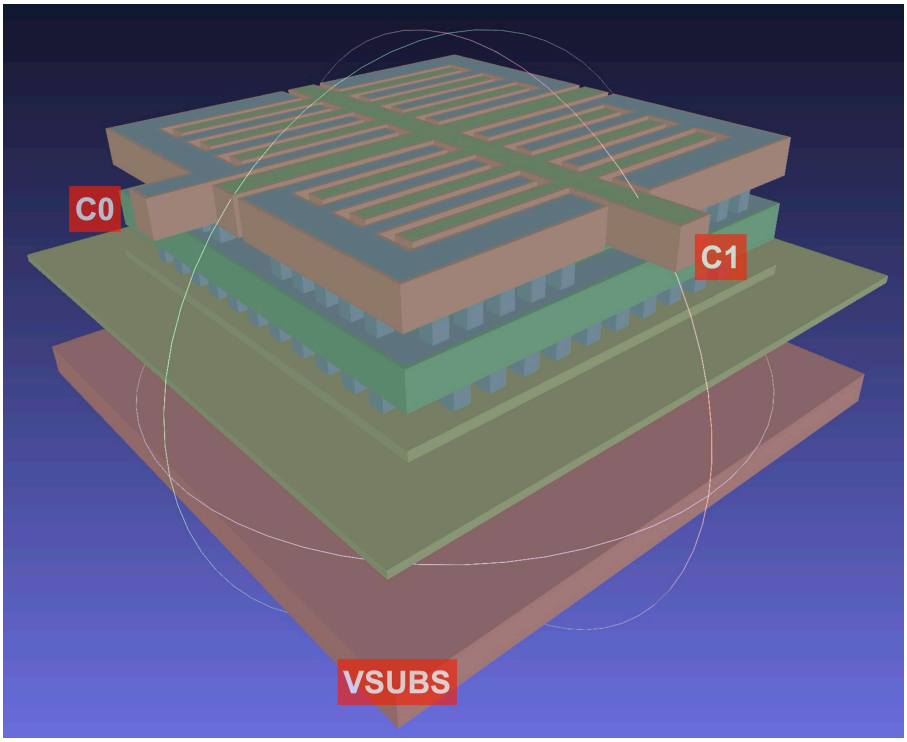
4.3 Output Maxwell Capacitance Matrix

A Maxwell capacitance matrix (Maxwell 1873) provides the relation between voltages on a set of conductors and the charges on these conductors, as described by the `FasterCap` author in the white paper (Di Lorenzo 2023).

`FasterCap` log output prints the Maxwell capacitance matrix (one for each iteration/refinement).



(a) MOM Capacitor: GDS Layout



(b) MOM Capacitor: `MeshLab` 3D Preview

Figure 20: MOM Capacitor

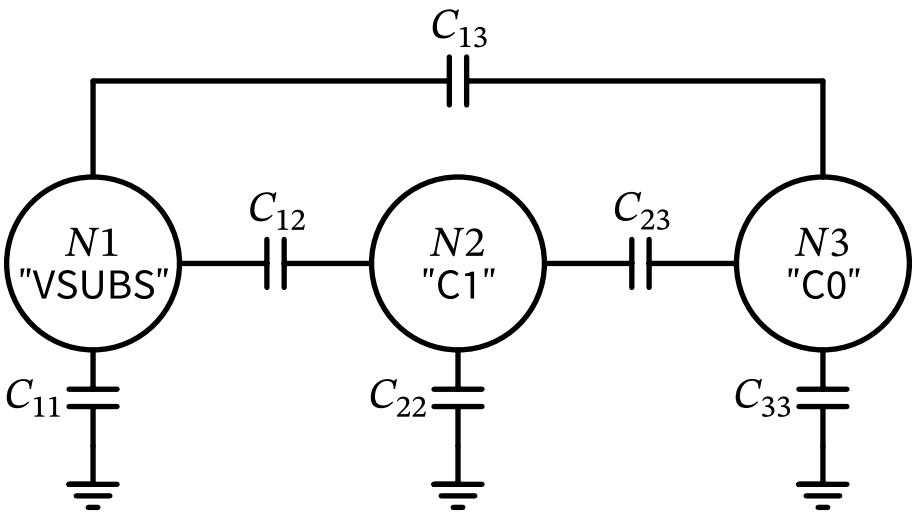


Figure 21: Schematic representation of the MOM capacitor.

```
Capacitance matrix is:
Dimension 3 x 3
g1_VSUBS  2.08888e-09 -1.46568e-10 -7.37007e-10
g2_C1  -1.64457e-10 1.47687e-08 -1.44368e-08
g3_C0  -7.68811e-10 -1.4528e-08 1.54806e-08

Weighted Frobenius norm of the difference between capacitance (auto option): 0.005031

Solve statistics:
Number of input panels: 18819 of which 5856 conductors and 12963 dielectric
Number of input panels to solver engine: 18819
Number of panels after refinement: 47371
Number of potential estimates: 6249961
Number of links: 19880193 (uncompressed 2244011641, compression ratio is 99.1%)
Max recursion level: 35
Max Mesh relative refinement value: 0.00126534
Time for reading input file: 0.014353s
Time for building super hierarchy: 0.001892s
Time for discretization: 0.562416s
Time for building potential matrix: 0.673007s
Time for precond calculation: 0.251937s
Time for gmres solving: 20.033398s
```

Figure 22: `FasterCap` Log Output: Maxwell Capacitance Matrix

- Matrix Properties:
  - Scaling: units have to be divided by 10^{-6}
  - rows and columns are the same (list of net names)
  - Row Cells:
    - off diagonals cells contains the coupling between row/col nets (times -1)
    - diagonal cells contains the sum of the absolute values of all other cells in the row
  - Matrix Symmetry:
    - in theory (ideal world), the matrix would be symmetric
    - in practice it's not

    - therefore `FastCap2` did average the off-diagonals
    - `FasterCap` does not average, so it's done as part of `KPEX`

$$C_{3\times3} = \begin{matrix} \\ \text{VSUBS} \\ \text{C0} \\ \text{C1} \end{matrix} \begin{matrix} \begin{matrix} \text{VSUBS} & \text{C0} & \text{C1} \end{matrix} \\ \begin{bmatrix} c_{11}+c_{12}+c_{13} & -c_{12} & -c_{13} \\ -c_{21} & c_{22}+c_{22}+c_{23} & -c_{23} \\ -c_{31} & -c_{32} & c_{31}+c_{32}+c_{33} \end{bmatrix} \end{matrix}$$

$$C_{avg} = \begin{bmatrix} c_{11} - |\text{avg}(c_{12}, c_{21})| - |\text{avg}(c_{13}, c_{31})| & |\text{avg}(c_{12}, c_{21})| & |\text{avg}(c_{13}, c_{31})| \\ |\text{avg}(c_{12}, c_{21})| & c_{22} - |\text{avg}(c_{12}, c_{21})| - |\text{avg}(c_{23}, c_{32})| & |\text{avg}(c_{23}, c_{32})| \\ |\text{avg}(c_{13}, c_{31})| & |\text{avg}(c_{23}, c_{32})| & c_{33} - |\text{avg}(c_{13}, c_{31})| - |\text{avg}(c_{23}, c_{32})| \end{bmatrix}$$

$$C_{femtofarad} = \begin{bmatrix} 2.09 - \text{avg}(0.147, 0.164) - \text{avg}(0.74, 0.77) & \text{avg}(0.147, 0.164) & \text{avg}(0.74, 0.77) \\ \text{avg}(0.147, 0.164) & 14.77 - \text{avg}(0.147, 0.164) - \text{avg}(14.44, 14.52) & \text{avg}(14.44, 14.52) \\ \text{avg}(0.74, 0.77) & \text{avg}(14.44\ 14.52) & 15.48 - \text{avg}(0.74, 0.77) - \text{avg}(14.44\ 14.52) \end{bmatrix}$$

$$= \begin{bmatrix} 1.18 & 0.16 & 0.75 \\ 0.16 & 0.13 & 14.48 \\ 0.75 & 14.48 & 0.25 \end{bmatrix}$$

Result CSV

Device	Net1	Net2	Capacitance [F]	Capacitance [fF]
Cext_0_1	VSUBS	C1	1.555125e-16	0.16f
Cext_0_2	VSUBS	C0	7.52909e-16	0.75f
Cext_1_2	C1	C0	1.44824e-14	14.48f
Cext_1_1	C1	VSUBS	1.307875e-16	0.13f
Cext_2_2	C0	VSUBS	2.45291e-16	0.25f

Figure 23: `FasterCap` Maxwell Capacitance Matrix: Interpretation

5 KPEX/MAGIC Engine

This engine is merely just a wrapper around `magic`, which prepares a TCL script that opens the layout file and starts MAGIC's PEX flow. See Section 2.6 to get started with a first extraction example.

The following chapter will illustrate concepts of the parasitic extraction done in `MAGIC`, also motivated by the fact that the engine in Section 6 will be based on those. Major parts of this illustration, the figures and concepts are based on work done by `MAGIC` maintainer Tim Edwards, especially his talk from FSiC conference 2022 (Edwards 2022) and a (conceptual follow-up) ChipsAlliance meeting on April 4, 2023, see (Edwards 2023b) and (Edwards 2023a). In addition, code review and debugging of the MAGIC codebase was performed.

5.1 MAGIC database units

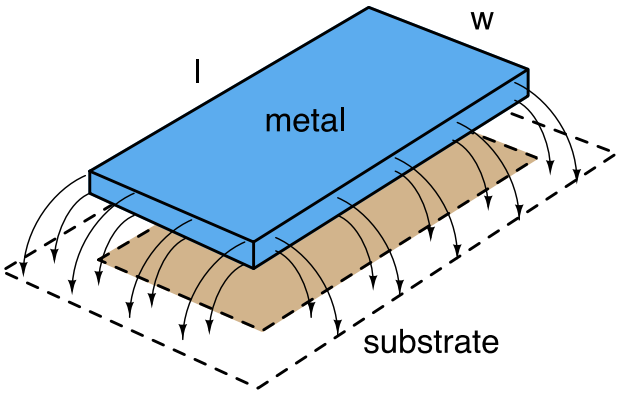
- To convert between MAGIC database units and μm, a scaling factor α is used, so that $L_{\mu m} = \frac{L_{dbu}}{\alpha}$
- E.g. for `sky130A`, $\alpha = 200.0$

5.2 Types of Parasitic Capacitances

MAGIC models multiple types of capacitances:

- *Substrate Overlap:* Overlap area of a metal with the substrate
- *Substrate Fringing:* Sidewall of a metal fringes out to substrate
- *Sidewall Capacitance:* Coupling between adjacent sidewalls on the same layer
- *Overlap Capacitance:* Overlap on different metal layers
- *Fringe Capacitance ("Side Overlap"):* Sidewall of a metal fringes out other metal layers

5.3 Substrate Capacitance



- Overlapping area:

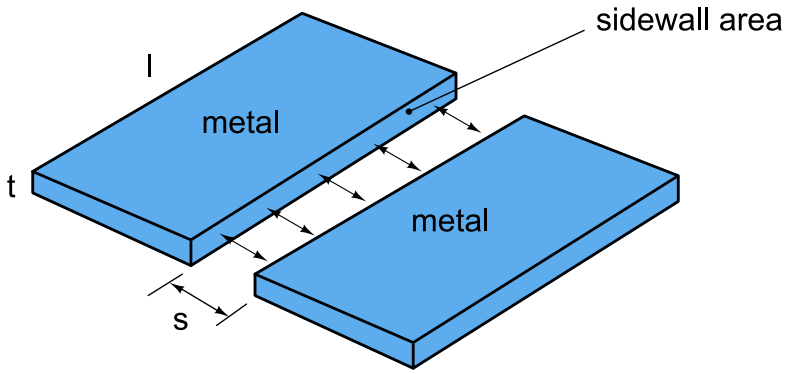
$$C_{area} = \frac{\epsilon_{si} * K}{d} * \text{area} \quad \left[\frac{F}{\mu m^2} * \mu m^2\right]$$

- Fringe ("Perimeter"):

$$C_{fringe\,to\,substrate} = \text{perimeter} * C_{perim} = (2l + 2w) * C_{perim}$$

- Coefficients like C_{perim} are part of the tech files (Parasitic Tables)

5.4 Sidewall Capacitance



$$C_{sidewall} = \frac{\epsilon_{si} * K}{s} * \text{sidewall area} \quad \left[\frac{F}{\mu m^2} * \mu m^2\right]$$
$$= \frac{\epsilon_{si} * K}{s} * t * l \quad \left[\frac{F}{\mu m^2} * \mu m * \mu m\right]$$
$$C_{sidewall} = \frac{C_{sidewall\,coeff}}{s} * l \quad \left[\frac{F}{\mu m} * \mu m\right]$$

The coefficient describes the coupling between the two facing sidewalls, but `MAGIC` walks every edge, so each of the two edges contributes half of it:

$$C_{sidewall\,per\,edge} = \frac{1}{2} * \frac{C_{sidewall\,coeff}}{s} * l$$

- Coefficients are part of the tech files (Parasitic Tables)
- Layer thickness t is normally multiplied into the coefficient
- Foundry tables give constant coefficient referenced to $s = 1$

> ⚠ Count the sidewall pair once
>
> The sidewall coefficient describes the full pair. Half belongs to each directed edge visit. A second halving of the already weighted coefficient makes the pair too small by a factor of two.
> `MAGIC 8.3.693` removes that extra halving during final technology processing. The earlier interpretation that revisions before `8.3.679` simply counted the pair twice was incorrect. The upstream correction explains the two sites that applied the factor.
> `KPEX 0.9.0` restores the full pair value for both constant and sloped gaps. Updating the physical coefficient tables is a separate change: multiplying all extraction results by two would also alter overlap and fringe contributions, which are not affected by this counting correction.

5.5 Overlap Capacitance

- Overlapping area:

$$C_{area} = \frac{\epsilon_{si} * K}{d} * \text{area} \quad \left[\frac{F}{\mu m^2} * \mu m^2\right]$$

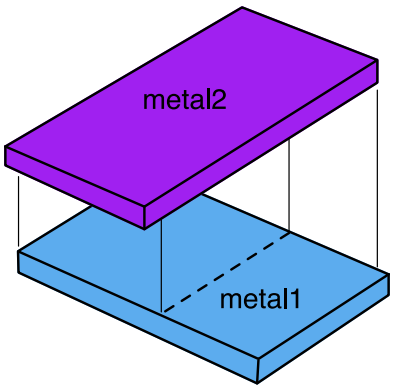


Figure 24: Overlap Capacitance

5.6 Fringe Capacitance

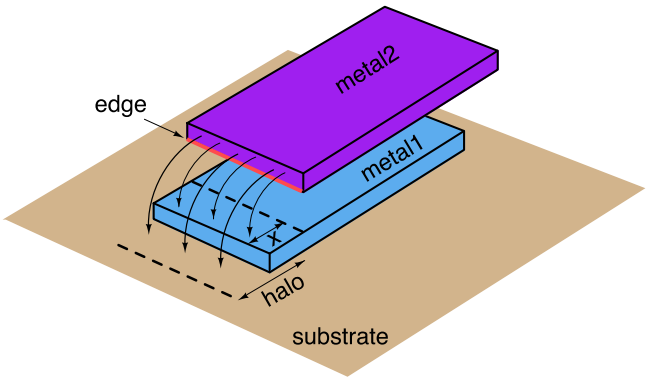


Figure 25: Fringe Capacitance: Overlapping (1)

- Causing sidewall (its bottom edge depicted red)
- Assume: Field is bounded by fringe halo (e.g. 8 μm away from edge)
- Fractions of fringe goes to metal1 or substrate

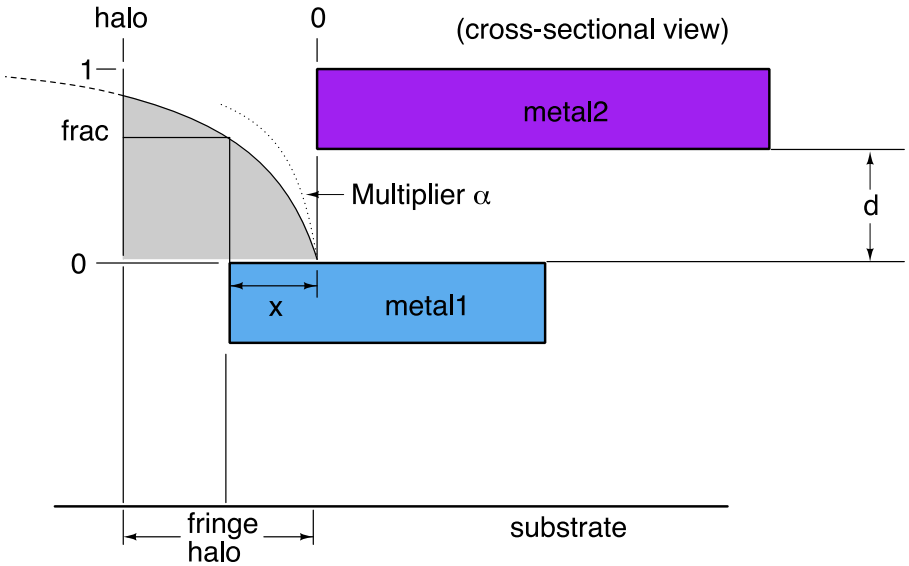


Figure 26: Fringe Capacitance: Overlapping (2)

- Multiplier α (comes from tech files: overlap table)
  - determines how quickly fringe capacitance drops with increasing distance

  - α is related to distance d between layers
  - α is proportional to $C_{overlap_{metal1 \leftrightarrow metal2}} = \frac{\epsilon_{si} * K}{d} * \text{area}$ (for fixed value of area $1\,\mu m^2$)

- Fringe Fractions:
  - $\text{frac}_{metal1} = \frac{2}{\pi} * \text{atan}(\alpha_{metal2 \rightarrow metal1} * x)$
  - $\text{frac}_{sub} = \frac{2}{\pi} * \text{atan}(\alpha_{metal2 \rightarrow sub} * (\text{halo} - x))$
  - $\frac{2}{\pi}$ is multiplied because of scaling to interval [0.0, 1.0], as $\text{atan}(\infty) = \frac{\pi}{2}$

- Overlap capacitance:
  - $C_{overlap} = \frac{\epsilon_{si} * K}{d} * \text{area} \quad (\text{with area} = 1 \mu m^2)$

- Coupling capacitance $metal1 \leftrightarrow metal2$:
  - $\alpha_{metal1 \leftrightarrow metal2} = \alpha_{scalefac} * C_{overlap_{metal1 \leftrightarrow metal2}}$
  - $\text{frac}_{metal1} = \frac{2}{\pi} * \text{atan}(\alpha_{metal1 \leftrightarrow metal2} * x)$
  - effective length = edge length $*$ frac_{metal1}
  - $C_{fringe_{metal2 \rightarrow metal1}}$ = effective length $* C_{sideoverlap_{metal2 \rightarrow metal1}}$

- Coupling capacitance $metal1 \leftrightarrow sub$:
  - $\alpha_{metal1 \leftrightarrow sub} = \alpha_{scalefac} * C_{overlap_{metal1 \leftrightarrow sub}}$
  - $\text{frac}_{sub} = \frac{2}{\pi} * \text{atan}(\alpha_{metal1 \leftrightarrow sub} * (\text{halo} - x))$
  - effective length = edge length $*$ frac_{sub}
  - $C_{fringe_{metal2 \rightarrow sub}}$ = effective length $* C_{sideoverlap_{metal2 \rightarrow sub}}$

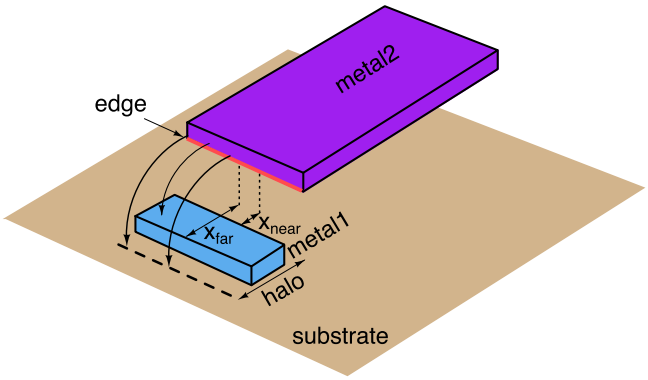


Figure 27: Fringe Capacitance: Non Overlapping (1)

- Partial side overlap
  - In case there is only a partial side overlap, the non-existing near fraction is subtracted from the far fraction
  - $metal1$ wire is offset, starts at x_{near}
  - $metal1$ ends at x_{far}

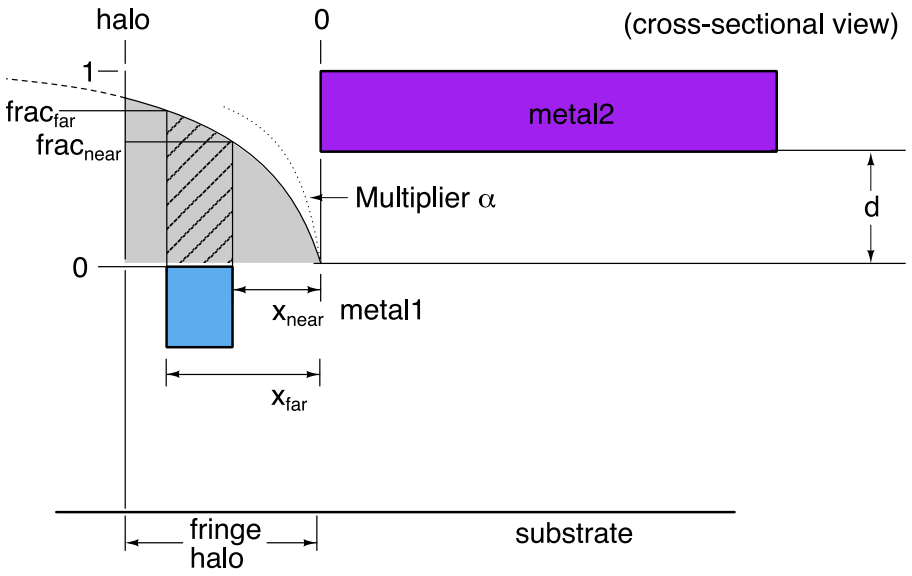


Figure 28: Fringe Capacitance: Non Overlapping (2)

- $\text{frac}_{near} = \frac{2}{\pi} * \text{atan}(\alpha * x_{near})$
- $\text{frac}_{far} = \frac{2}{\pi} * \text{atan}(\alpha * x_{far})$
- $\text{frac} = \text{frac}_{far} - \text{frac}_{near}$

5.7 Shielding Effects

Table 31: Shielding effects

Type	Shielding To Substrate	Between layers	On same layer
Overlap shielding	✓	✓	✗
Sidewall shielding	✗	✗	✓
Lateral fringe shielding	✗	✓	✓
Vertical fringe shielding	✓	✓	✗

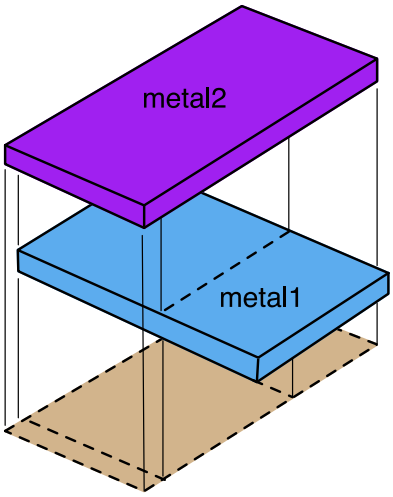
Note, given an analyzed sidewall (edge in 2D), lateral fringe shielding

- is caused by opposing shapes on the same layer
  - even by the same polygon
  - by other polygons (same net)
  - by other polygons (different net) — this is what we also look at when analyzing sidewall coupling
- will shield the fringing to the lower layers, i.e. in Figure 30 the coupling between the two shapes is the same in the above and below cases

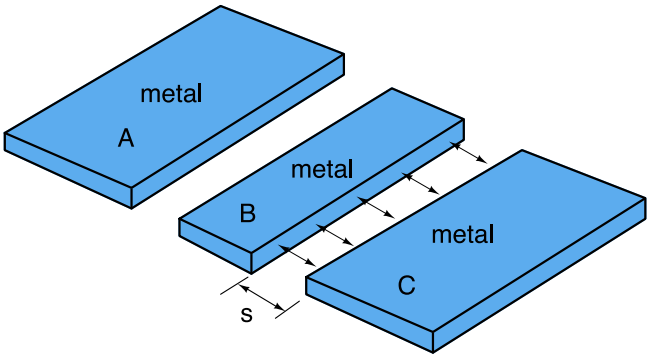
5.8 Parasitic Resistance

Magic constructs a graph of resistors between nodes

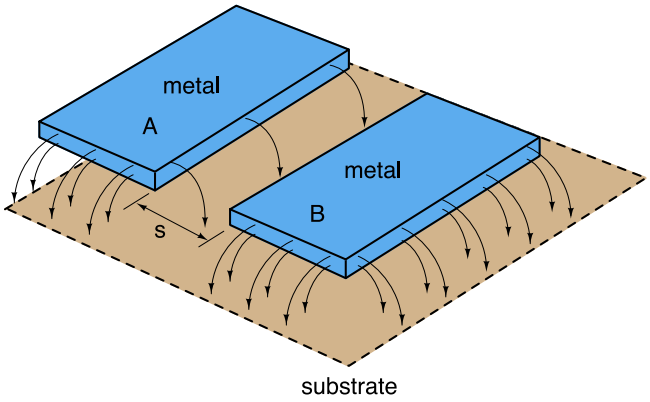
- device terminals
- pins
- junctions (vias or branching on the same metal layer)



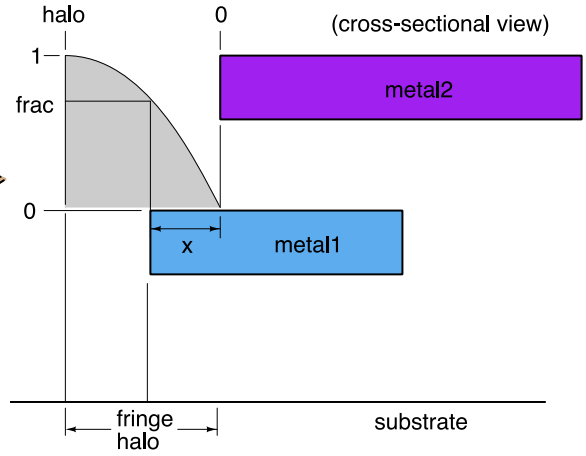
(a) Overlap shielding



(b) Sidewall shielding



(c) Lateral fringe shielding (1/2)



(d) Vertical fringe shielding

Figure 29: Shielding Effects

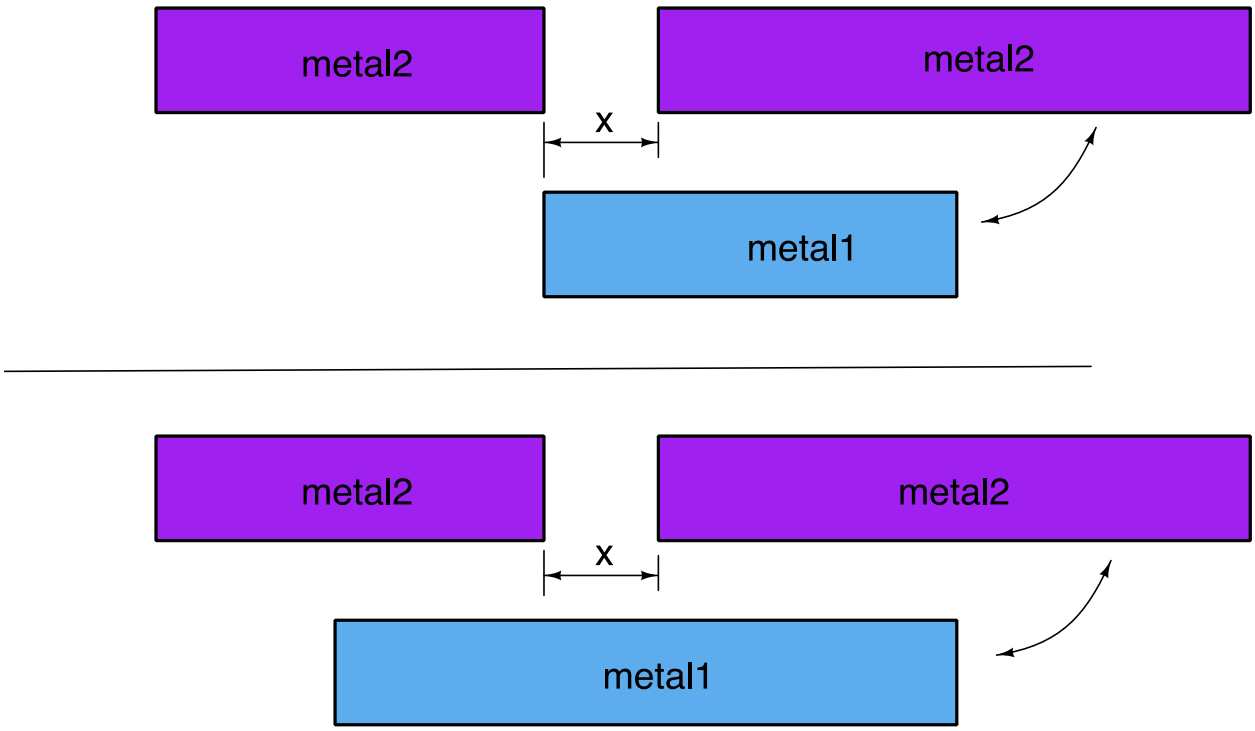
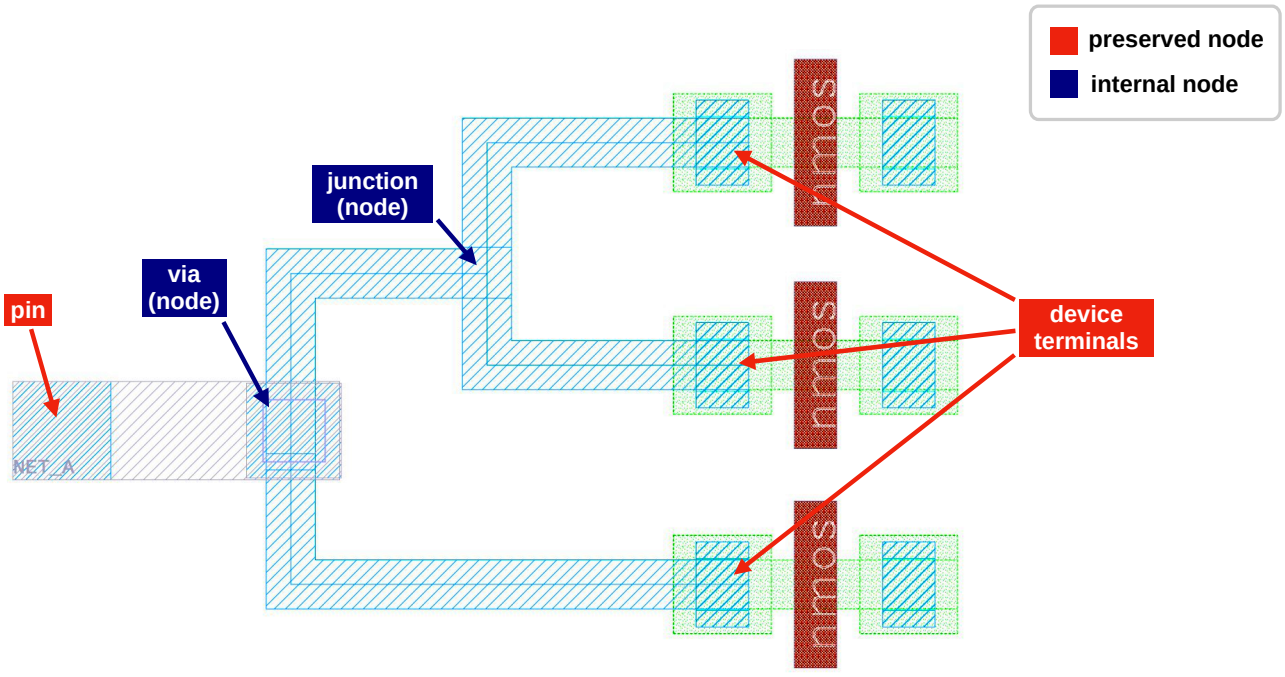
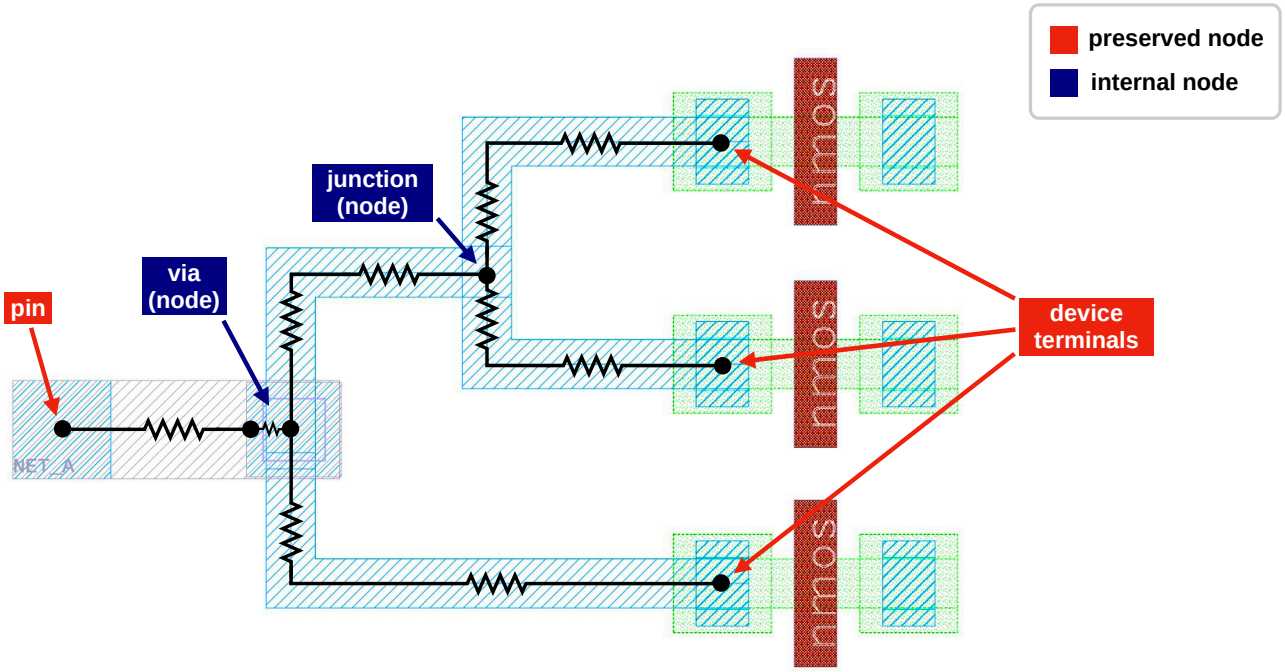


Figure 30: Lateral fringe shielding (2/2)





5.8.1 Wire resistance

- Given a wire with length l and height h, the basic formula is

$$R_{wire} = \frac{l}{h} * R_{coeff} \qquad \left[\frac{\mu m}{\mu m} * m\Omega\right]$$

- Coefficient R_{coeff} is part of the tech files (Parasitic Tables)
  - defined for every metal layer
  - in $m\Omega$ for $1\,\mu m^2$
  - Coefficient already includes the thickness aspect of the layer, so the formula works in 2D

5.8.2 Via resistance

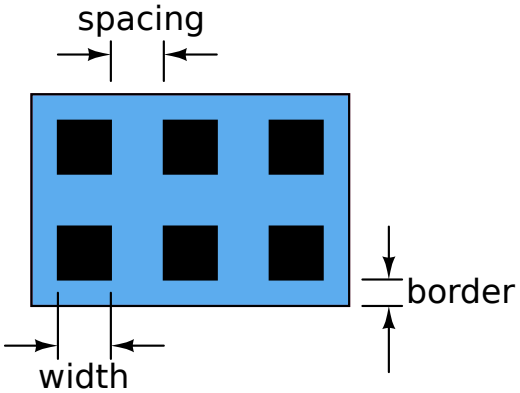


Figure 31: Via dimensions)

- Given
  - drawn via in the layout, with width w and height h
  - design rules defined for each via layer:

    - via width *viawidth*
    - *spacing* between vias (in case of a via array)
    - *border* on each side of the via

- MAGIC interpretation of the via drawing
  - determine number of vias in x and y direction, i.e. n_x and n_y
  - if x and y dimensions are below the minimum size, MAGIC counts 1 via in each direction
  - if the dimensions are larger, we calculate how much vias fit

$$R_{via} = \frac{R_{coeff}}{n_x * n_y} \quad \left[\frac{m\Omega}{\text{via count}}\right]$$

$$n_x = 1 + \left\lfloor \frac{w - (viawidth + 2 * border)}{viawidth + spacing} \right\rfloor$$

$$n_y = 1 + \left\lfloor \frac{h - (viawidth + 2 * border)}{viawidth + spacing} \right\rfloor$$

- Coefficient R_{coeff} is part of the tech files (Parasitic Tables)
  - defined for every via layer
  - in $m\Omega$ per via
  - Coefficient already includes the thickness aspect of the layer, so the formula works in 2D

6 KPEX/2.5D Engine

This analytical extraction engine is separate from the **PEX25D interchange format** described in Section 11.1. `kpex extract --2.5D` selects the engine; `kpex pex25d` generates a geometry and material description without running an engine.

Field solvers are precise, yet slow, they are useful to obtain a golden reference.

For most use cases, a faster engine is desirable. `KPEX/MAGIC` is such an engine, but the `MAGIC` code is tightly coupled with the database, layer/via design choices, and user interface of `MAGIC`. For example, it runs single-threaded.

Therefore, the `KPEX` 2.5D Engine intends to implement the concepts and formulas of `MAGIC` (see Section 5.2), but in a way that is best suited to the `KLayout` API, and is modular and open for parallel execution.

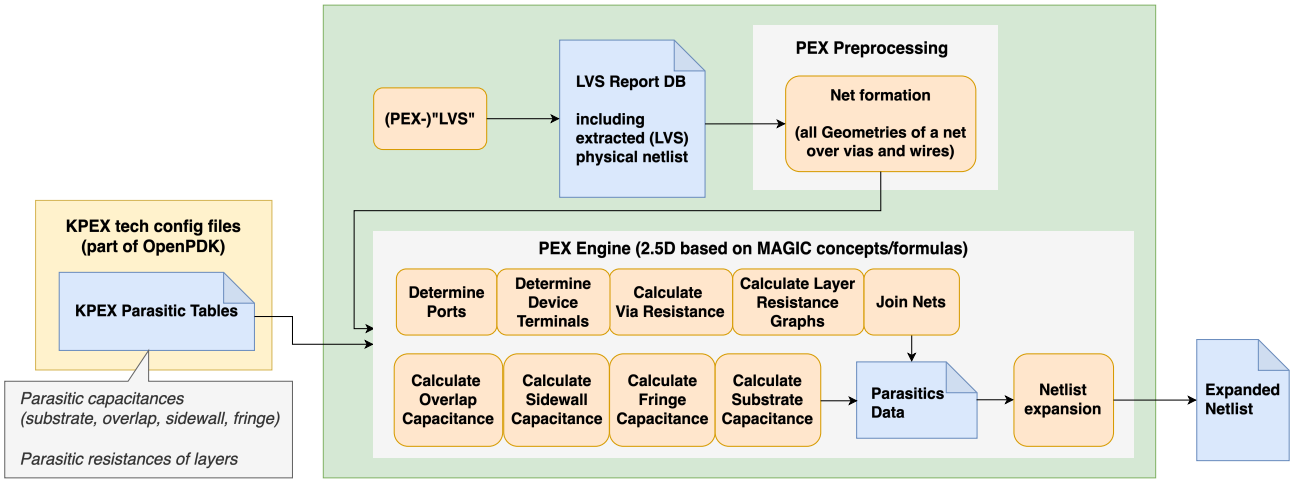


Figure 32: `KPEX` 2.5D Engine

In Figure 32, we see that as in the `KPEX/FasterCap` engine (see Section 4), a "LVS" script is used to extract the connectivity information and create an LVS report database.

As mentioned in Section 1.4, `KPEX` uses the KLayout API. During the course of this project, in cooperation with Matthias Köfferlein, the `KLayout` API was extended, in order to simplify parasitic extraction.

The engine uses those new API classes to extract parasitic capacitances:

- `klayout.db.PolygonNeighborhoodVisitor` (since `0.30.1`): used during the extraction of overlap capacitances (see Section 5.5)
- `klayout.db.EdgeNeighborhoodVisitor` (since `0.30.1`): used during the extraction of sidewall capacitances (see Section 5.4) and fringe capacitances (see Section 5.6)
- Geometry classes with properties in `klayout.db` (since `0.30.1`): store net information directly on polygons, edges, edge pairs and boxes

The engine uses those new API classes to extract parasitic resistances:

- `klayout.pex` module (since KLayout `0.30.2`)
  - `klayout.pex.RNetExtractor`: used to extract resistance networks
  - `klayout.pex.RExtractorTech`: minimal process stack description for conductor and via layers

6.1 Choosing the Level of Detail

`--mode CC` keeps each connected net equipotential. `--mode R` breaks its conductors into a resistance network. `--mode RC` places extracted capacitances on that distributed network as well, so the position of a load along a wire affects the simulated response.

Grounded capacitance is an explicit approximation: `--mode C` uses ideal conductors, while `--mode RC --coupling grounded` retains distributed resistance. The development-version options, floating-conductor treatment and standalone postprocessing are described in Section 7.

The capacitance model has separate area-overlap, facing-sidewall and fringe terms. Polygon integration preserves their geometric meaning for sloped boundaries (Section 6.2). It does not turn the analytical model into a field solver or remove the need to check its coefficients against the intended technology.

The halo limits which neighboring geometry participates in sidewall and fringe extraction. Keep the same halo and fringe normalization when comparing runs. Recent neighborhood batching and rectangular-shield interval subtraction reduce repeated geometric work while retaining the polygon path for general shapes. They do not justify shrinking the halo to obtain a faster result.

6.2 Capacitance of Sloped and Holed Polygons

A wire does not have to end in a rectangle. A taper, a diagonal segment, or a shield with a hole changes the distance to the neighboring conductor along the edge. `KPEX/2.5D` integrates that changing geometry using the same overlap, sidewall, and fringe models introduced in Section 5.2.

This extends where those models can be applied. It does not turn the analytical extraction into a three-dimensional field solution: the dielectric stack and fitted parasitic coefficients still determine the model's accuracy.

6.2.1 Looking Along One Edge

For each source edge, imagine rotating the layout until that edge lies on the horizontal axis. Let x run along the edge and y point outwards from its conductor. The local coordinate system is attached to the edge, so a diagonal wire needs no special capacitance formula.

The neighborhood is divided into intervals where the facing boundaries remain straight. Within one interval:

- A sidewall gap is a linear function $d(x)$
- A fringe target has a near boundary $n(x)$ and a far boundary $f(x)$
- A shield can shorten either boundary or remove a part of the target

A bounding box loses this information. It fills the empty space beside a triangle and across a hole. Using the minimum distance everywhere also assigns the closest part's stronger coupling to the whole edge.

Overlap capacitance needs no distance integration in this plane. It is proportional to the actual unshielded intersection area, including the subtraction of holes:

$$C_{\text{overlap}} = \frac{k_{\text{ov}}}{1000} A_{\text{visible}}.$$

Here k_{ov} is in aF/μm^2, area is in μm^2, and the result is in fF.

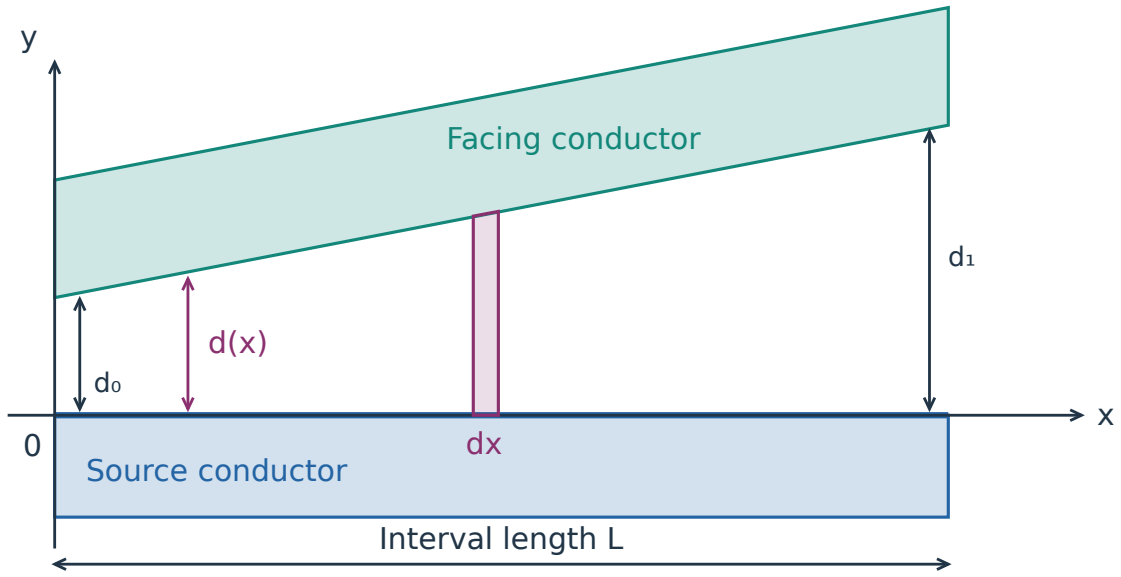


Figure 33: An edge-local view of a changing gap. The coordinate system follows the source edge even when the original layout is rotated.

6.2.2 Integrating a Changing Sidewall Gap

Take an edge interval of length L with endpoint gaps d_0 and d_1:

$$d(x) = d_0 + \frac{d_1 - d_0}{L}\, x, \qquad 0 \le x \le L.$$

The technology fit adds an offset o to the physical gap. For a positive physical gap, a negative offset must not make the fitted separation zero or negative. KPEX/2.5D therefore uses

$$s(d) = \max\left(d + o, \frac{d}{2}\right).$$

The geometry factor is the integral of the reciprocal separation:

$$G_{\mathrm{sw}} = \int_0^L \frac{dx}{s(d(x))}.$$

For a full-pair sidewall coefficient k_{sw} in aF, one directed edge contributes

$$C_{\mathrm{sw,edge}} = \frac{k_{\mathrm{sw}} G_{\mathrm{sw}}}{2 \cdot 1000} \quad [\mathrm{fF}].$$

The two directed visits together count the pair once. The geometry integral introduces no additional half, and the conversion from aF to fF divides by 1000. This counting convention is separate from the shape of the gap (Section 5.4).

If the whole interval uses $s = d + o$, the integral has a closed form:

$$G_{\mathrm{sw}} = \begin{cases} \dfrac{L}{d_0 + o}, & d_0 = d_1, \\ \dfrac{L}{d_1 - d_0} \ln\left(\dfrac{d_1 + o}{d_0 + o}\right), & d_0 \neq d_1. \end{cases}$$

The logarithm is the result of integrating $1/(a+bx)$. It is not a correction fitted specifically to diagonal wires. Reversing the edge exchanges d_0 and d_1 and changes both signs, leaving the result unchanged.

For a negative o, the two separation branches meet at $d = -2o$. If a taper crosses that gap, split its integral there. On the clamped part, integrate $2/d$ instead of $1/(d + o)$. The two branches agree at the split, so no capacitance jump is introduced.

6.2.2.1 Why the Mean Gap Is Insufficient

Consider $L = 4\,\mu$m, a gap widening from 0.2 to $0.6\,\mu$m, and $o = 0.1\,\mu$m. These values describe a geometry exercise, independently of any PDK coefficient:

$$G_{\mathrm{sw}} = \frac{4}{0.6 - 0.2} \ln\left(\frac{0.6 + 0.1}{0.2 + 0.1}\right) = 10 \ln(7/3).$$

Replacing the taper by its mean gap gives $4/(0.4 + 0.1) = 8$. The integral is larger because reducing a small gap strengthens coupling more than increasing an already large gap weakens it.

The first half of the edge carries the fraction

$$\frac{C_{[0,L/2]}}{C_{[0,L]}} = \frac{\ln(5/3)}{\ln(7/3)},$$

which is greater than one half. This matters when the capacitance is assigned to different nodes of an extracted resistor network.

6.2.3 Integrating Fringe Across a Target

The fringe model uses a cumulative fraction

$$F(y) = \frac{2}{\pi} \arctan(\alpha \max(0, y)).$$

$F(y)$ describes how much of the fringe is reached by distance y. Its slope decreases with distance. A strip near the source edge therefore receives more fringe than an equally wide strip farther away.

For a strip extending from $n(x)$ to $f(x)$, the received fraction is $F(f(x)) - F(n(x))$. Integrating along the source edge gives

$$C_{\mathrm{fringe}} = \frac{k_{\mathrm{fr}}}{1000\,N} \int_0^L [F(f(x)) - F(n(x))]\,dx.$$

k_{fr} is the directional side-overlap coefficient in aF/μm. N describes halo normalization, explained below. The overlap table supplies the parameter α for this source/target layer pair, while the side-overlap table supplies k_{fr}. They play different roles: one sets the distance dependence, the other its magnitude.

In the current coefficient convention, the numerical value of α in $\mu\mathrm{m}^{-1}$ is 0.02 times the overlap coefficient in aF/$\mu\mathrm{m}^2$. This scaling belongs to the empirical fringe model; it is not an identity between capacitance and inverse distance.

The near and far boundaries of a trapezoid are linear functions. Their integrals can be evaluated analytically using

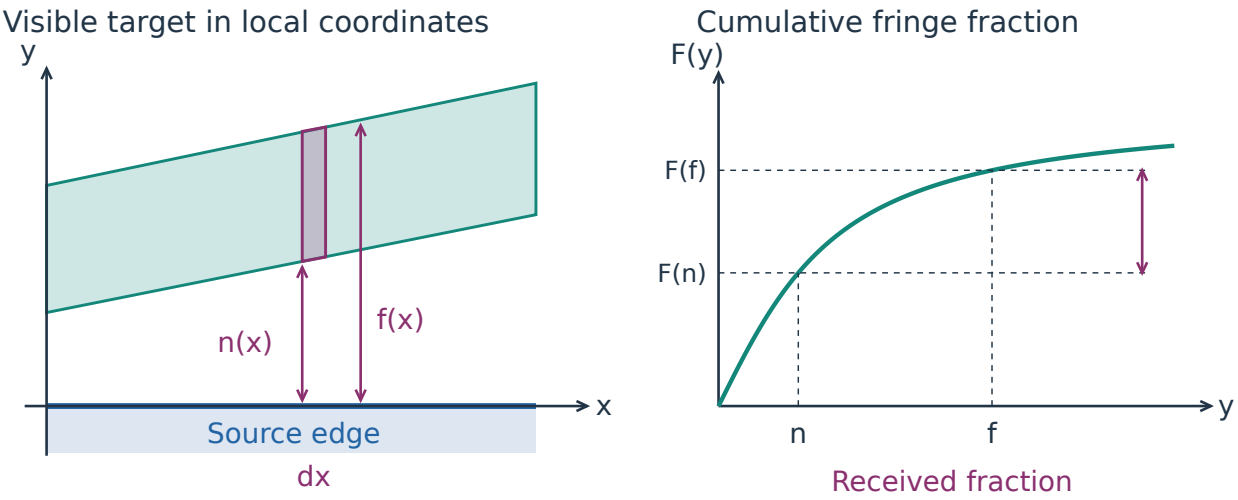


Figure 34: A fringe target receives the difference between the cumulative fractions at its far and near boundaries. Both boundaries may vary along the edge.

$$J(z) = z\arctan(z) - \frac{1}{2}\ln(1+z^2), \qquad J'(z) = \arctan(z).$$

For $q(x) = q_0 + mx > 0$, $m \neq 0$ and $\alpha > 0$:

$$\int_a^b F(q(x))\,dx = \frac{2}{\pi\alpha m}\left[J(\alpha q(b)) - J(\alpha q(a))\right].$$

A constant boundary gives $(b-a)F(q_0)$. A boundary crossing zero is split there, because the part behind the source edge contributes no outward fringe. The implementation evaluates equivalent forms that avoid subtracting nearly equal numbers when the slope or α is small.

6.2.4 Subtracting Shields and Keeping Holes

A nearer conductor on the same layer limits the lateral reach. Its sloping facing edge produces a sloping shadow, rather than a rectangle beginning at one mean distance. This shadow also applies when the nearby shape belongs to the same net or to the opposite side of the same polygon.

For a target on a different layer, conductors between the source and target provide additional shielding. The source-to-target ordering matters: a conductor beyond the target does not shield that target. Above and below the source are handled separately.

After subtracting these shields, a target may be concave or have holes. KPEX/2.5D partitions the remaining geometry into vertical trapezoids in the source edge's local coordinates:

1. Split at the distinct horizontal positions of polygon vertices
2. Find the boundary crossings within each strip
3. Pair successive crossings as filled intervals, leaving holes empty
4. Integrate each trapezoid and add its contribution

This is a partition of the visible polygon, not a replacement by rectangles. Intersections at a new strip boundary remain floating-point values for the capacitance calculation. Only the geometry drawn in the extraction report is rounded back onto the layout grid.

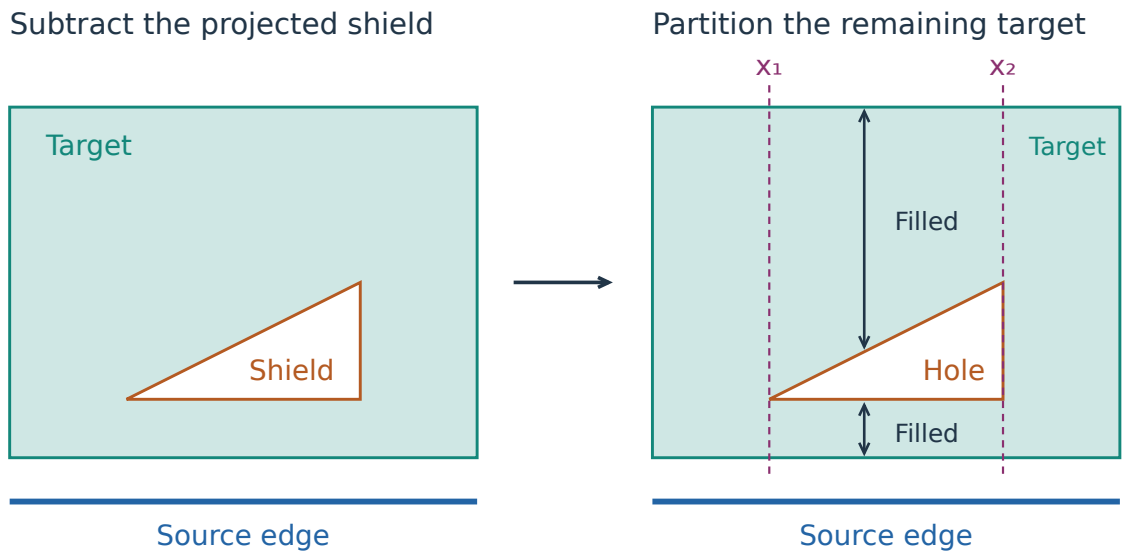


Figure 35: A triangular shield leaves a hole in the target. Vertical strips pair the filled boundary crossings, so the hole contributes no fringe. The dashed lines mark decomposition boundaries, not new conductors.

The small-contribution cutoff is applied to each resulting unshielded polygon before its internal trapezoids are emitted. A polygon that contributes more than 0.0001 fF remains even if each trapezoid individually contributes less. A value equal to the cutoff is omitted. Changing the mathematical subdivision therefore does not change whether that polygon is retained.

6.2.5 Preserving the Profile When Geometry Is Cut

The total capacitance is not enough when a marker removes part of an edge or when RC extraction divides it between nodes. The taper example above has equal lengths on either side of its midpoint but unequal capacitances.

`KPEX/2.5D` retains an integral profile for a sloped contribution. A cut asks the profile for the capacitance over the retained interval. A later cut uses the original coordinates of that interval. It does not restart the distance ramp at its new left endpoint.

This makes three operations consistent:

- Summing the parts reproduces the original contribution before cutoffs
- An exclusion marker removes the capacitance carried by its covered geometry
- RC node assignment uses the local capacitance density, rather than only the fraction of edge length

For overlap capacitance, the density within one layer pair is uniform per area, so area fractions remain appropriate.

6.2.6 Inspecting Sloped Wires

The `sky130A` pattern below has a tapered conductor `A`, a bent conductor `B`, and a rectangular conductor `C`. The `A` ↔ `B` gap varies along the taper, whereas part of the `B` ↔ `C` gap is constant.

GDS: `sidewall_non_parallel_li1.gds.gz`

From the `klayout-pex` source checkout, the committed LVS database allows the extraction to be repeated without running LVS:

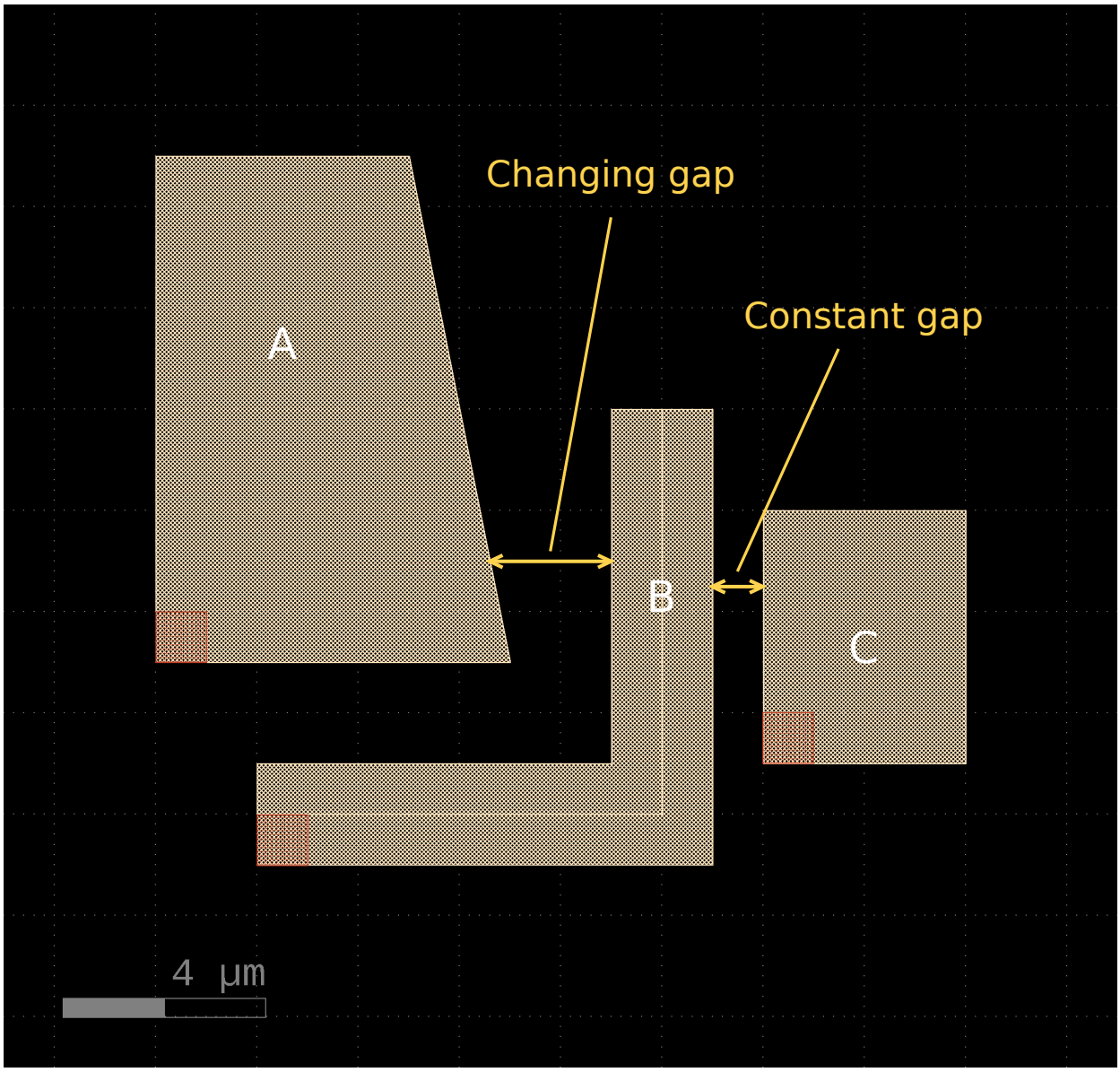


Figure 36: The `sky130A` sloped-wire example on `li1`. The taper changes the `A` ↔ `B` separation along the facing interval.

```
lvsdb_dir=testdata/lvsdb/sky130A/test_patterns
kpex extract --pdk sky130A --mode CC --blackbox n \
  --lvsdb "$lvsdb_dir/sidewall_non_parallel_li1.lvsdb.gz" \
  --2.5D --halo 8 --scale y --report-caps y \
  --out-dir output_polygon_example
```

The example was checked with `klayout-pex 0.9.0` and the `KLayout` Python module `0.30.13`. `--report-caps y` adds each contribution and its geometry to the extraction report. Inspect the `A` ↔ `B` sidewall entries to see how the facing intervals follow the taper. The exported netlist combines the contributions belonging to the same pair of nets.

6.2.7 Halo, Grid and Model Limits

The halo is the finite lateral neighborhood searched around an edge. `--halo` overrides the technology's `side_halo`, in μm. It changes which geometry is considered, so it is a modeling choice as well as a runtime limit.

The default `--scale y` normalizes the fringe fraction to that halo. For a positive overlap coefficient and halo h, $N = F(h)$, and a target occupying the full interval from 0 to h receives the full coefficient for that edge length. A zero overlap coefficient gives $\alpha = 0$ and zero fringe contribution before normalization. With `--scale n`, $N = 1$, so the tail beyond the halo remains omitted. A larger halo can change both the visible targets and, when scaling is enabled, the normalization of nearer ones.

The `--scale` flag here concerns fringe fractions. It is independent of SPICE `.option scale`, which controls the interpretation of selected device geometry parameters.

These limits matter when comparing results:

- An overlap coefficient of exactly zero gives zero fringe, including with halo scaling enabled
- Very small positive overlap coefficients approach a uniform distribution across a normalized halo
- Rotating a layout onto an integer grid can move vertices and neighborhood boundaries slightly
- Intervals no longer than one database unit are discarded to suppress degenerate edge neighborhoods
- The analytical model does not resolve full three-dimensional corner fields or replace coefficient calibration

For a rotation comparison, keep the same physical dimensions, coefficient tables, halo, and scaling option. A finer database unit reduces geometric rounding. Quarter-turn rotations preserve integer-grid coordinates exactly; arbitrary angles usually do not.

6.3 Excluding Parts of the Layout

Some structures are better served by a model than by a 2.5D extraction: an inductor by an EM simulation, which also has its inductance, a hard block by a netlist of its own, or the power grid of a block by the extraction of the top level. FEM co-simulation for example matches measured mm-wave amplifiers better than an RC extraction (Sagouo Minko and Stander 2019). The rest of the layout should still be extracted, and connect to the model.

This raises four questions, which this section answers:

- How does a layout say what the extraction should leave out, and for which layers?
- What happens to the shapes there: can they just be removed, with dielectric in their place?
- What about the capacitances between what is left out and what is not?
- How does the rest of the layout connect to the model?

There are three ways to select what is left out, which follow the same rule (Section 6.3.3):

Table 32: Ways to select what is left out

Option	Selects	Connection to the model
`--exclusion-markers {ignore,tie,ports}`	what is within the marker layers of a PDK, or of the user (Section 6.3.4)	as the value says
`--cut CELL[=PREFIX]`	the shapes and the devices of the instances of a cell, not the routing over them (Section 6.3.7)	ports
`--exclude-net NET`	a net (Section 6.3.8)	the net itself

By default, nothing is left out, so that a netlist doesn't change without one of these. Markers and cells are read from the GDS, so that these options need `--gds`, also together with `--lvsdb`, as an LVS database only has the shapes of nets.

The options work with `--2.5D`, `--fastercap` and `--fastcap` (Section 6.3.9), not with `--magic`.

6.3.1 What the PDKs Mark

Table 33: Markers of the supported PDKs

PDK	Marker	Applies to	Drawn by
`ihp-sg13*`	`NoRCX.drawing 148/0`	all layers of the stack	the pcells `inductor2` and `inductor3` (parameter `blockqrc`, on by default), the same polygon as `IND`
`ihp-sg13*`	`<layer>.noqrc */28`	one layer each: `Activ`, `GatPoly`, `Metal1` to `Metal5`, `BackMetal1`, `TopMetal1`, `TopMetal2`	the pcell `rfcmim`, around the device by 5.6 µm, up to `TopMetal1`
`ihp-sg13*`	`NoRCX.<pair>` `148/41` to `55`, `148/123` to `127`, `148/300`, `148/301`	pairs of metals from `m2m3` up, metal to substrate (`m1sub`)	no pcell
`ihp-sg13*`	`IND 27/0`	the metal of an inductor, whose pins touch the edge of `IND`	inductors
`sky130A`	none	`inductor.drawing 82/24` only takes `met2` and `met3` out of the conductors of LVS	RF coils
`gf180mcuD`	none	`IND_MK 151/5` marks inductors: the LVS deck only defines it, the DRC keeps `COMP` and `Poly2` fill 3 µm away	

The layer rules of the IHP PDKs don't describe the markers beyond one line each. What the pair markers can mean follows from their names, as they name a pair of plates. Of the 533 GDS of the IHP reference layouts and LVS testcases, 25 have a marker, all of them `noqrc` of `rfcmim`.

By default, `KPEX` honors `NoRCX.drawing` and the pair markers. `noqrc` only takes effect on request, with `--exclusion-marker-layer` (Section 6.3.6): the extraction of the `rfcmim` already knows the device, in white-box mode it would lose the capacitor, and in a reference layout it would hit 2624 µm² of metal that isn't part of the device.

`sky130A` and `gf180mcuD` have no layer for this, in their layer tables or in the cells and layouts looked at. SkyWater's extraction rules leave parasitics to the models by device recognition (varactors, MIM capacitors, RF ESD diodes, precision poly resistors), and MAGIC by the property `device` of a cell, which zeroes the parasitics of the cell. So these PDKs take markers of the user (Section 6.3.6).

6.3.2 Why the Geometry Stays

Tools take one of three approaches, here by the public material of commercial tools:

Table 34: Approaches of tools

Approach	The shapes	What is around them	Used by
Remove	are not passed to the solver	loses its load, which the vendor warns about	commercial tool A
Keep, couple to ground	are considered	keeps its load, with no path into the device	commercial tool A, commercial tool B for nets within a cell
Keep, couple to the net	are interconnect with all cross-coupling, a model replaces the RC model	keeps its load, on the net	commercial tool A, commercial tool B for nets with a pin

Removing the shapes is the simple approach, but it changes what is outside them, as Figure 37 shows. A 2.5D extraction rests on the shielding by what is in between: neighbors reduce the coupling across a layer from 16.3 % to 1.8 % of the total, and it's only negligible above a metal density threshold (Cong et al. 1997). The same holds for metal without a function: ignoring dummy fill gives errors above 30 %, and the fill on other layers matters more than the fill on the same layer (Kurokawa et al. 2004). Full-chip extractors handle grounded fill, floating fill needs a 3D solver (Kahng et al. 2006; Yu et al. 2006).

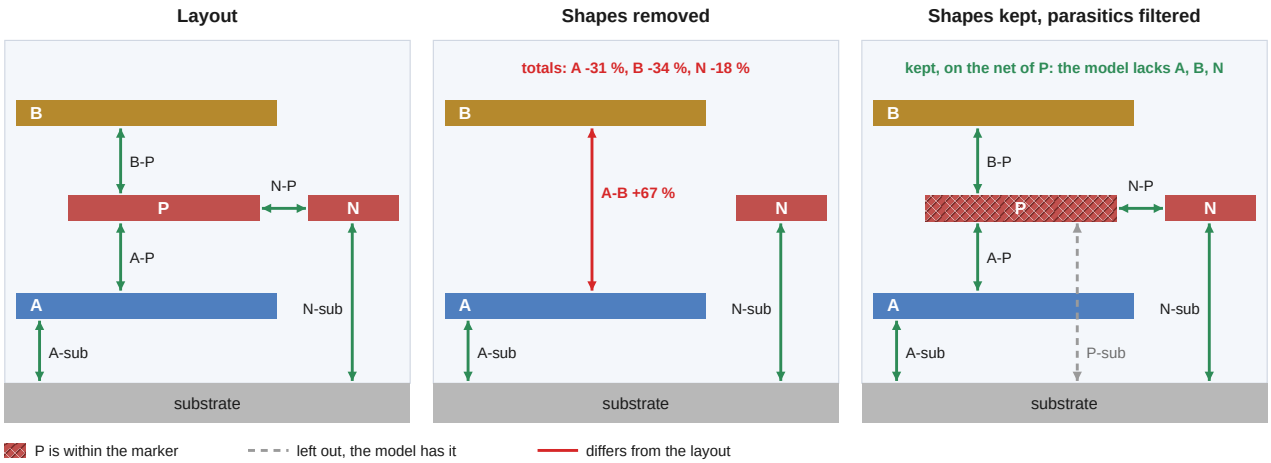


Figure 37: The plate `P` is left to a model, the wires `A`, `B` and `N` are not. Removing `P` couples `A` to `B` through the hole, and takes its load from all of them. `KPEX` keeps `P` in the field model, and leaves out only what the model has.

Table 35 has this for small layouts of `ihp-sg13g2`, in `--mode CC`, with FasterCap in brackets. FasterCap is no reference here, it only shows the same trend. "Coupling dropped" keeps the shape as a shield, but leaves out all of its capacitances, also those to the wires.

Table 35: What removing a shape does to what is around it

Layout	Value	With the shape	Shape removed	Coupling dropped
Metal3 plate of 20 µm x 20 µm between 2 µm wide Metal2 wire A and Metal4 wire B	A ↔ B	1.39 fF	+67 % (+63 %)	unchanged
	A total	11.19 fF	-31 % (-24 %)	-39 % (-41 %)
	B total	8.08 fF	-34 % (-30 %)	-60 % (-57 %)
TopMetal1 wire over a Metal5 comb of 50 % density	total	11.60 fF	-39 % (-42 %)	-56 % (-73 %)
Metal3 comb between Metal5 wire T and Metal1 wire U	T ↔ U	1.16 fF	+38 %	
Metal3 wire 0.3 µm beside a Metal3 plate	total	5.05 fF	-18 % (-7 %)	-29 % (-38 %)
Metal5 wire under the coil of an inductor2	total	10.27 fF	-10 %	
20 µm wire ending at the cut, instead of going on	total	1.797 fF	+3.5 %	

So the shapes stay: without them, the wires around them have wrong values, which neither the netlist nor the model can correct.

> **Note**
>
> With a marker over all layers, the parts of A and B within it are left out too, so the first rows of Table 35 apply to a neighbor like N, and to the parts near the edge of the marker. They apply fully where only some shapes are left out at a location: a marker of one layer (Section 6.3.6), a cell with routing over it (Section 6.3.7), or a net (Section 6.3.8).

The coil of a PDK inductor is an exception that does no harm: the deck of KPEX takes the metal within IND out of the conductors, so that it is no shield, but it is 30 µm from the edge of its polygon, beyond the halo of 8 µm, so that nothing outside the polygon is affected.

6.3.3 What Is Left Out

A conductor is *in* where it lies within what is left out on its layer, otherwise *out*. Each polygon of the merged markers, and each instance of a cut cell, stands for a model of its own, which is called a *block* below.

Table 36: Which capacitances are left out

Capacitance between	What happens	Why
in and in, within one block	left out	the model has it
in and substrate	left out	the model has it, also where the fringe ends on the substrate beyond the marker
out and out, out and substrate	unchanged	nothing else has it
out and in	kept, on the net of the in plate, at its nearest node	the model doesn't know the out plate
out and in, where nothing connects the net of the in plate to the netlist	goes to the substrate	it has no node, a model wouldn't know this metal either

Capacitance between	What happens	Why
in and in, within different blocks	kept	neither model has it
parts of one net that only a model joins, with `--exclusion-markers ports`	kept, like between two nets	they are apart in the netlist, and neither the netlist nor the model has this capacitance otherwise

A capacitance partly within a marker is split: for an overlap by its area, for a sidewall or fringe along its edge. A fringe goes by the plate next to the edge, also on a well that goes on across a marker.

The resistance network has no resistance within a block. A via is within a block by its center, on both of its conductors, as vias can stick out past their metal, and the resistance of a part of a via would be made up. Otherwise it keeps its resistance and the metal it lands on.

Devices within a marker stay in the netlist, as the PDK inductors do. In white-box mode (`--blackbox n`), a capacitor within a marker is lost, as the device is removed and its plates are filtered. A warning names these devices. The devices of a cut cell are left out (Section 6.3.7).

A marker over routing costs that wire much of its capacitance, so a warning names the nets partly within markers, with their layers.

6.3.4 The Variants

Figure 38 shows the three values of `--exclusion-markers` for a wire through a marker. A `--mode CC` netlist has one node per net, so `tie` and `ports` are the same there, and a warning says that it has no ports for a model.

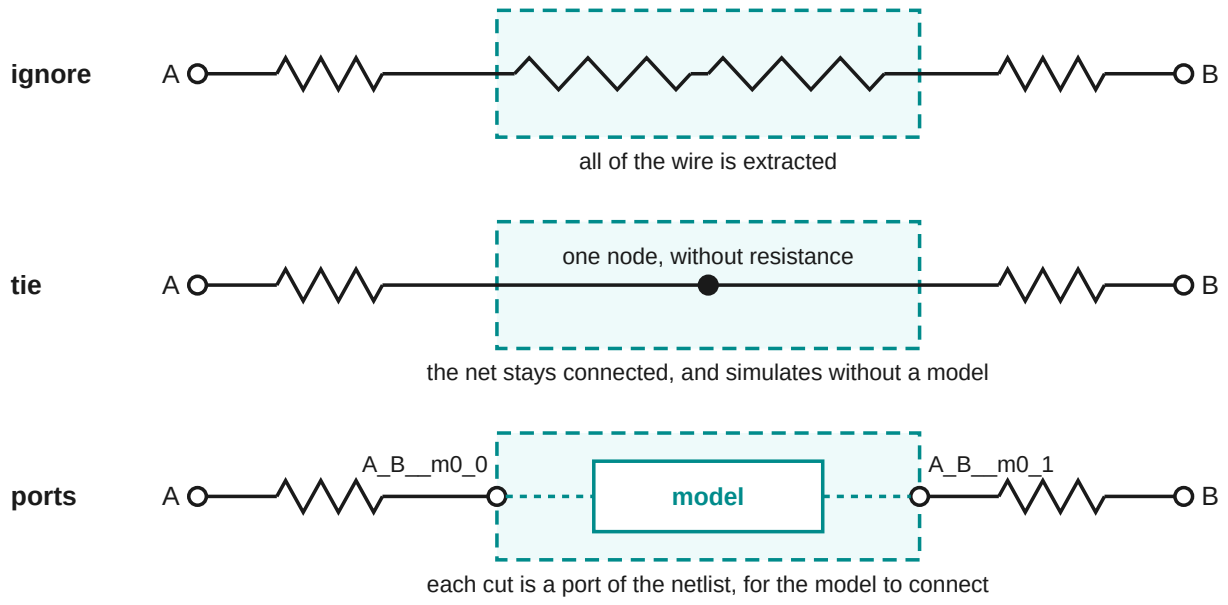


Figure 38: A wire from `A` to `B` through a marker with `--exclusion-markers ignore`, `tie` and `ports`

6.3.4.1 A Wire Through a Marker

Test pattern `exclusion_wire_through`, a `Metal3` wire of 60 µm x 2 µm with a marker over its middle third, is the example of the variants below:

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/exclusion_wire_through.gds.gz

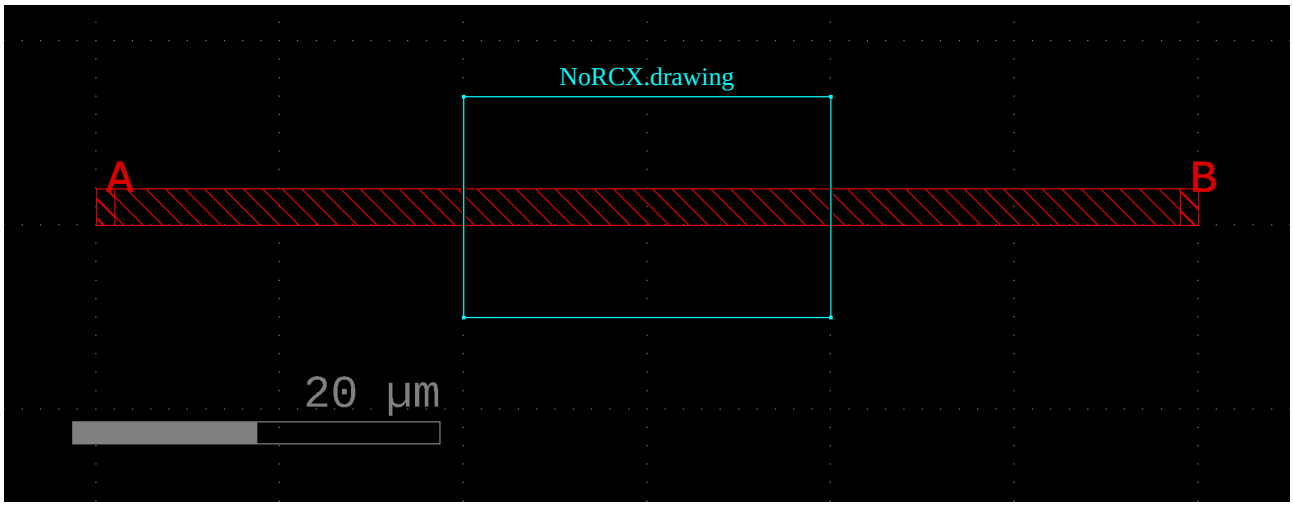


Figure 39: Test pattern `exclusion_wire_through`

Table 37: Extracted values of `exclusion_wire_through`

`--exclusion-markers`	`--mode CC`	`--mode R`
`ignore`	`A,B` ↔ `VSUBS` 5.327 fF	`A` ↔ `B` 2.596 Ω
`tie`	`A,B` ↔ `VSUBS` 3.593 fF	`A` ↔ `A,B` 0.858 Ω, `A,B` ↔ `B` 0.858 Ω
`ports`	`A,B` ↔ `VSUBS` 3.593 fF	`A` ↔ `A_B__m0_0` 0.858 Ω, `A_B__m0_1` ↔ `B` 0.858 Ω

6.3.4.2 `ignore`

What is within a marker is extracted like the rest of the layout. This is the default, as Figure 40 shows:

```
.SUBCKT exclusion_wire_through A B sub!
Cext_1 A sub! 2.66364f
Cext_2 B sub! 2.66364f
Rext_1 A A|B 0.001
Rext_2 A B 2.596
.ENDS exclusion_wire_through
```

6.3.4.3 `tie`

What is within a marker has no parasitics, and each net stays connected across it, so that the netlist simulates without a model. It's useful to see what the rest of a layout adds, or if a model is only needed for some of the analyses (Figure 41):

```
.SUBCKT exclusion_wire_through A B sub!
Cext_1 A sub! 0.951336f
Cext_2 A|B sub! 1.69065f
Cext_3 B sub! 0.951336f
Rext_1 A A|B 0.857956
```

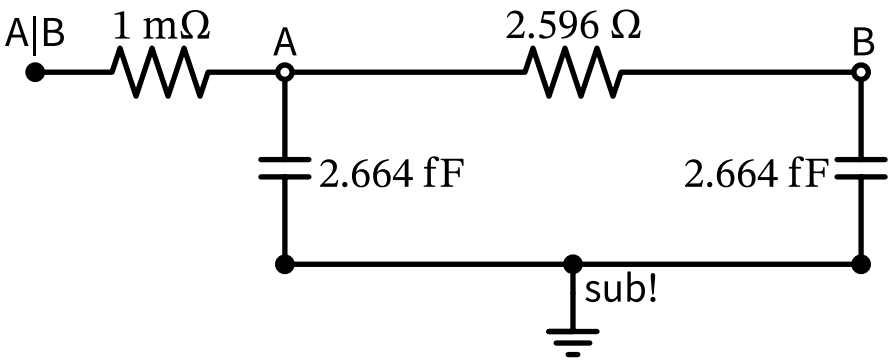


Figure 40: The netlist of `exclusion_wire_through` with `ignore`. Open dots are ports, the wire is one resistor, and `A|B`, the node of the net, is tied to the port `A`.

```
Rext_2 A|B B 0.858
.ENDS exclusion_wire_through
```

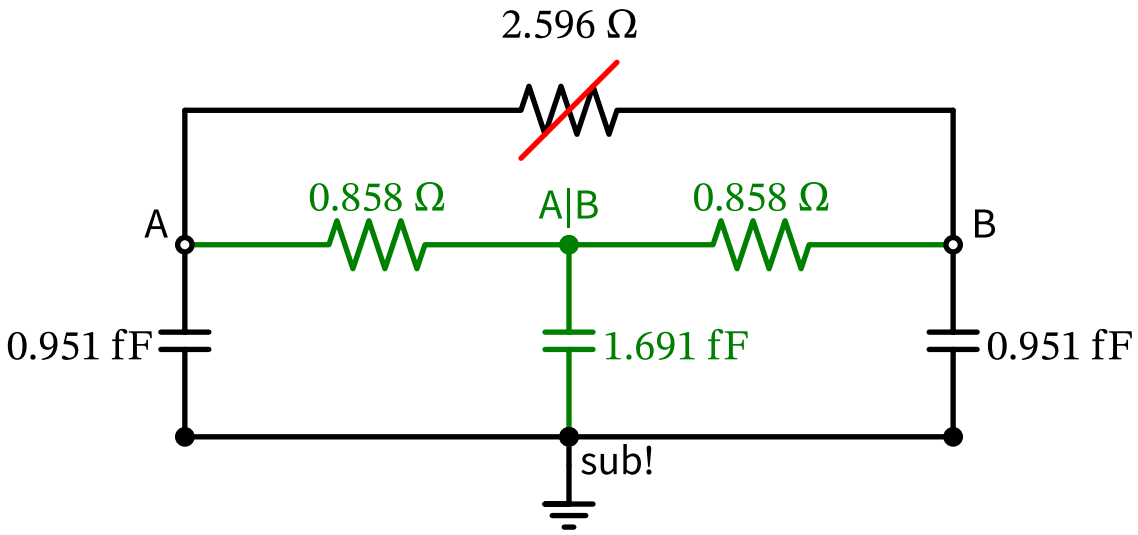


Figure 41: The netlist of `exclusion_wire_through` with `tie`. In green what replaces the resistor of `ignore`, struck through in red: a resistor outside the marker on each side, and one ideal node for all that is within the marker.

6.3.4.3.1 Metal Joined Only Outside the Marker

The metal within a marker stays apart where nothing within the marker joins it. In the test pattern `exclusion_stubs` (Figure 42), two stubs of a net end within the marker, which only the metal outside it joins. So their pins `S1` and `S2` stay two nodes, with 1.046 Ω to `A` each, as a tie would short the metal outside the marker.

6.3.4.4 `ports`

Each cut of a net at a marker is a port of the netlist, for a model to connect (Figure 43):

```
.SUBCKT exclusion_wire_through A A_B__m0_0 A_B__m0_1 B sub!
Cext_1 A sub! 0.951336f
Cext_2 A_B__m0_0 sub! 0.845326f
```

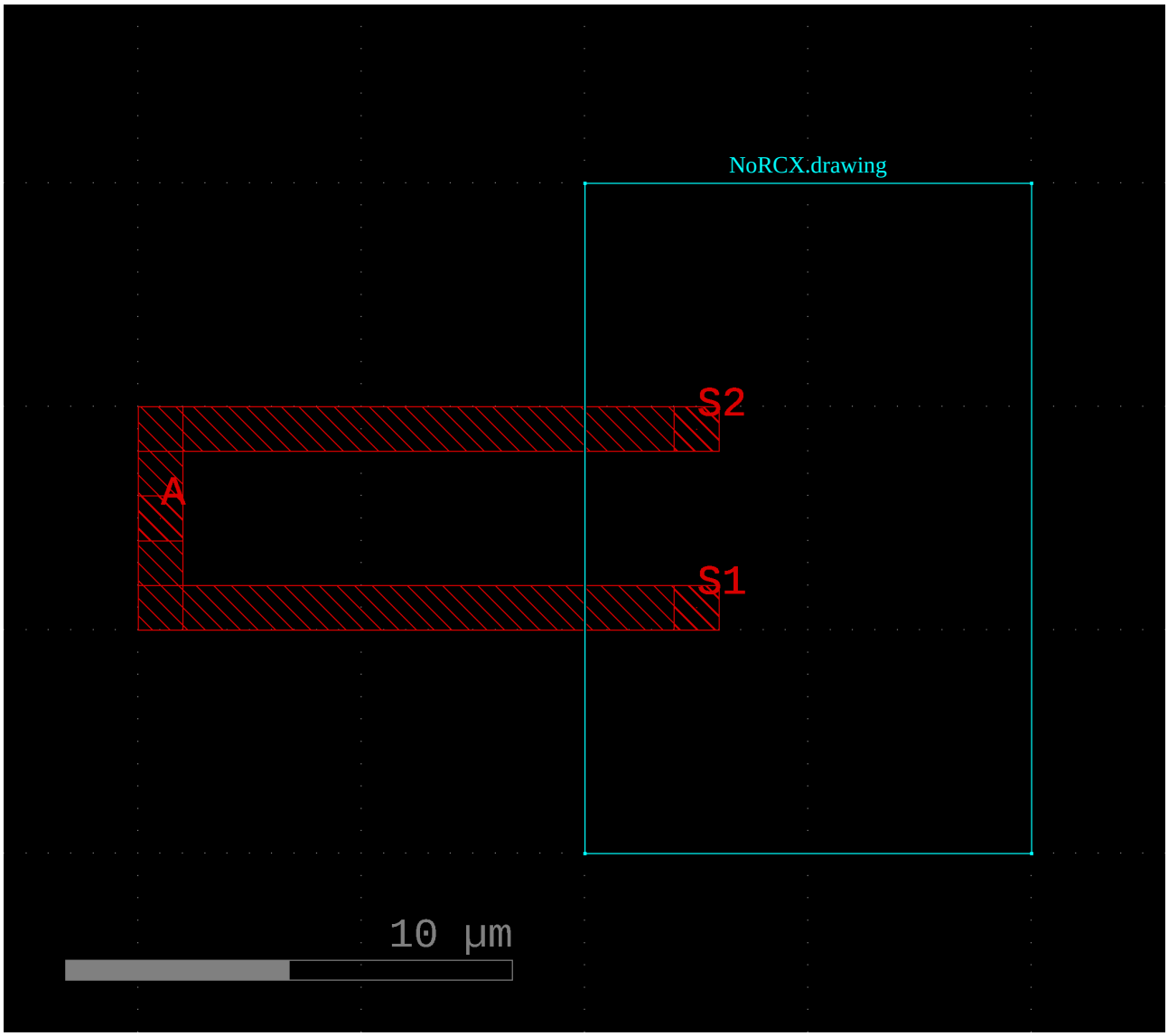


Figure 42: Test pattern `exclusion_stubs`

```
Cext_3 A_B__m0_1 sub! 0.845326f
Cext_4 B sub! 0.951336f
Rext_1 A A|B 0.001
Rext_2 A A_B__m0_0 0.857956
Rext_3 A_B__m0_1 B 0.858
.ENDS exclusion_wire_through
```

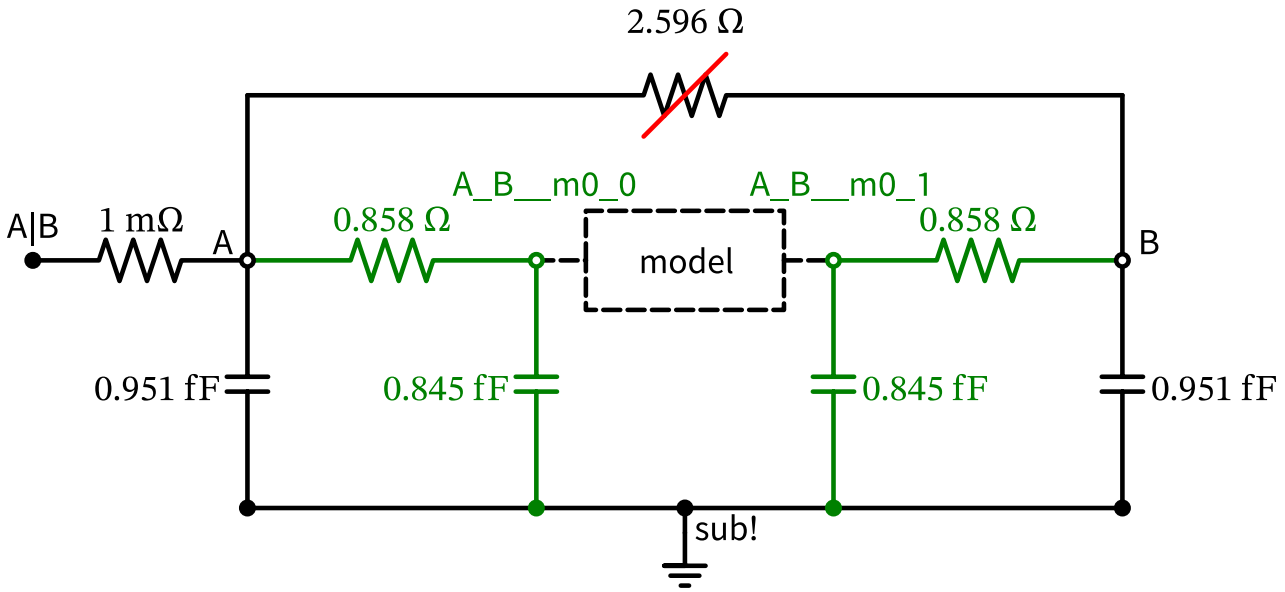


Figure 43: The netlist of `exclusion_wire_through` with `ports`. In green what replaces the resistor of `ignore`, struck through in red: a resistor outside the marker on each side, up to a port at each cut. Dashed, the model of what the marker stands for, which is no part of this netlist.

A port is named after the pin of the net within the marker that is nearest to the cut, on its layer before others, otherwise `<net>__m<marker>_<number>`. So labels within a marker name the ports. In `exclusion_stubs`, these are `S1` and `S2`. Such a pin only names a cut of the metal it is joined to within the marker, and not if the net has a pin of that label outside the marker, as a port is one node. A net entirely within a marker has a port at its pin.

`--check` accepts the pieces of a net that only these ports join. These pieces stay apart in the netlist, also where taps or wells would join them ideally otherwise, as that would short the model. And a part of a well beyond a cut keeps its port and its resistance, with the bulks of the devices on it. A bulk within the marker is a port of the model as well.

A port is one node, which is right across the current, also on a wide wire. Where the metal goes on beside a marker, along a cut, the whole cut would short that metal, so the port is a piece of 1 µm in the middle of the cut. The metal beside it then carries the current alone: test pattern `cut_strip_along_wire`, a wire of 100 µm x 1 µm with half of its width within a marker along 50 µm, has 8.78 Ω flat and 17.41 Ω with `ports`, as a port can't connect a model in parallel to that metal.

6.3.4.4.1 Capacitances Between the Parts of a Net

Metal that only a model joins is one net in LVS, but apart in the netlist with `ports`, so the capacitances between it count, unlike within a net otherwise. In test pattern `exclusion_hairpin` (Figure 44), two `Metal3` wires 2 µm apart end within the marker, where a bridge joins them. In `exclusion_hairpin_open`, the bridge has a gap, so they are two nets. With `ports`, both have the same values:

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/exclusion_hairpin.gds.gz

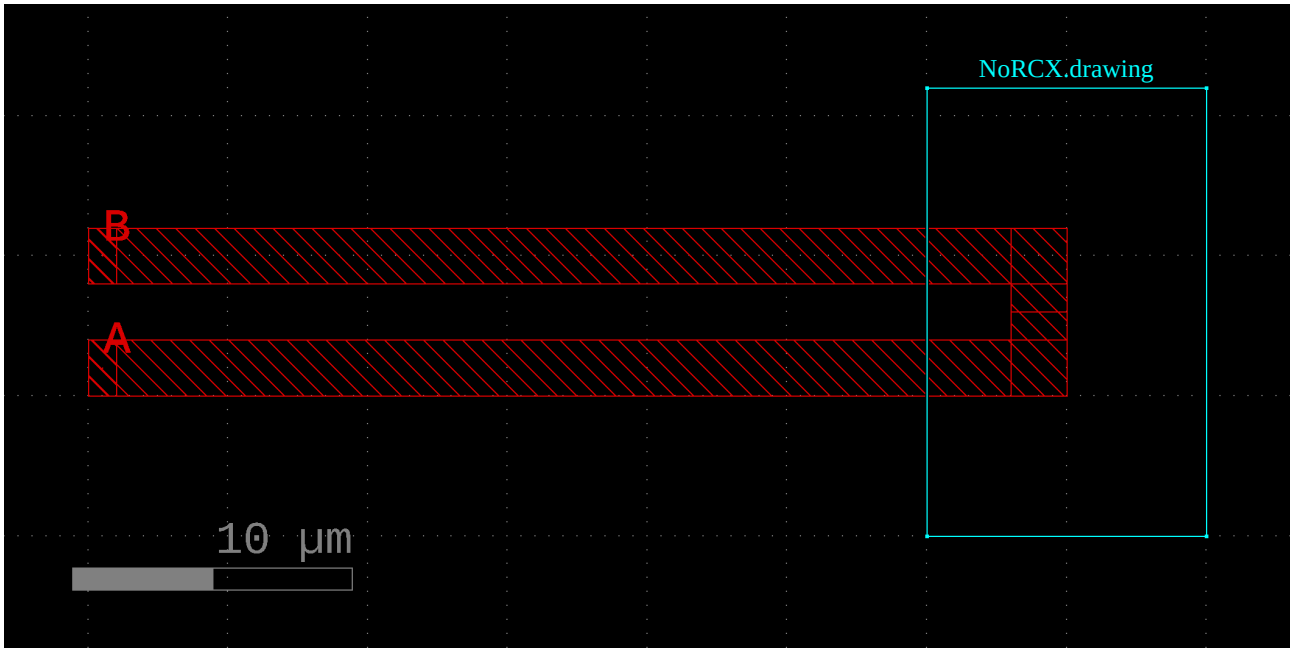


Figure 44: Test pattern `exclusion_hairpin`

Table 38: Capacitance between the wires of `exclusion_hairpin`

Capacitance	`exclusion_hairpin`, `ignore` or `tie`	`exclusion_hairpin`, `ports`	`exclusion_hairpin_open`, `ports`
from the pin of `A` to the pin of `B`		0.147 fF	0.147 fF
between the cuts		0.142 fF	0.142 fF

With `tie`, the net stays connected, so these wires are a net like without a marker, which has no capacitances within it. In `--mode CC`, a net is one node, so there are no ports and no such capacitances.

6.3.4.4.2 Checking a Model

To check that a model fits, a test extracts the part of the wire within the marker on its own, as a stand-in for a model, connects it between the ports, and compares the port admittance in `ngspice` with the one of the flat extraction. They differ by 2.3 %, by the fringe of that part at the cuts, where the flat wire has no edges.

6.3.5 Examples

6.3.5.1 A Neighbor Outside a Marker

Test pattern `exclusion_plate_neighbor` (Figure 45), a `Metal3` plate `P` just within a marker, with a `Metal3` wire `N` 0.3 µm beside it. `N` keeps all of its values, as it is out, and `P` keeps only its capacitance to `N`.

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/exclusion_plate_neighbor.gds.gz

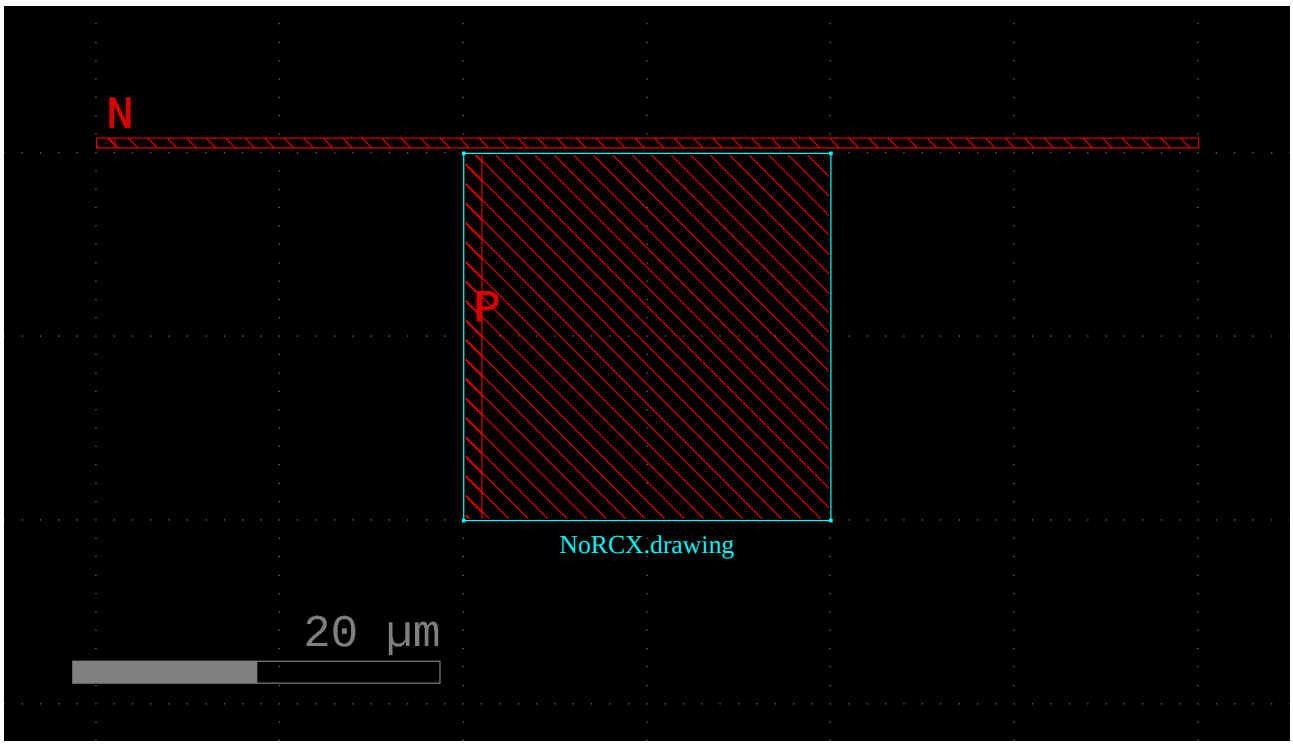


Figure 45: Test pattern `exclusion_plate_neighbor`

Table 39: Capacitances of `exclusion_plate_neighbor`

Nets	`ignore` [fF]	`tie`, `ports` [fF]	Plates
`N` ↔ `P`	1.478	1.478	out and in
`N` ↔ `VSUBS`	3.568	3.568	out and substrate
`P` ↔ `VSUBS`	6.720		in and substrate

Without a label on the plate, in `exclusion_floating_plate_neighbor`, nothing connects it to the netlist, so that the capacitance to it goes to the substrate: `N` ↔ `VSUBS` is 5.045 fF.

6.3.5.2 Two Markers

Two such plates `P` and `Q`, 0.5 µm apart. With a marker each (`exclusion_plates_2_markers`), they keep their 0.828 fF to each other, as neither model has it. With one marker over both (`exclusion_plates_1_marker`), the netlist has no capacitances at all.

6.3.5.3 Vias

Test pattern `exclusion_via_stack`, a wire from `A` to `B` on `Metal2`, with a bridge on `Metal3` over a gap in the middle, with a marker around the bridge:

Table 40: Resistances of `exclusion_via_stack`

`--exclusion-markers`	Resistors
`ignore`	0.568 Ω, via 2.031 Ω, 0.176 Ω, via 2.031 Ω, 0.532 Ω
`tie`	`A` ↔ `A,B` 0.418 Ω, `A,B` ↔ `B` 0.418 Ω
`ports`	`A` ↔ `A_B__m0_0` 0.418 Ω, `A_B__m0_1` ↔ `B` 0.418 Ω

In `exclusion_via_across`, the center of the via is outside the marker, so it keeps its 2.031 Ω, and only the `Metal3` within the marker has no resistance.

6.3.5.4 Devices

Test pattern `exclusion_nmos_diode2` (Figure 46), with a marker over a MOS transistor. The metal of its nets leaves the marker on `Metal1` and `Metal2`.

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/exclusion_nmos_diode2.gds.gz

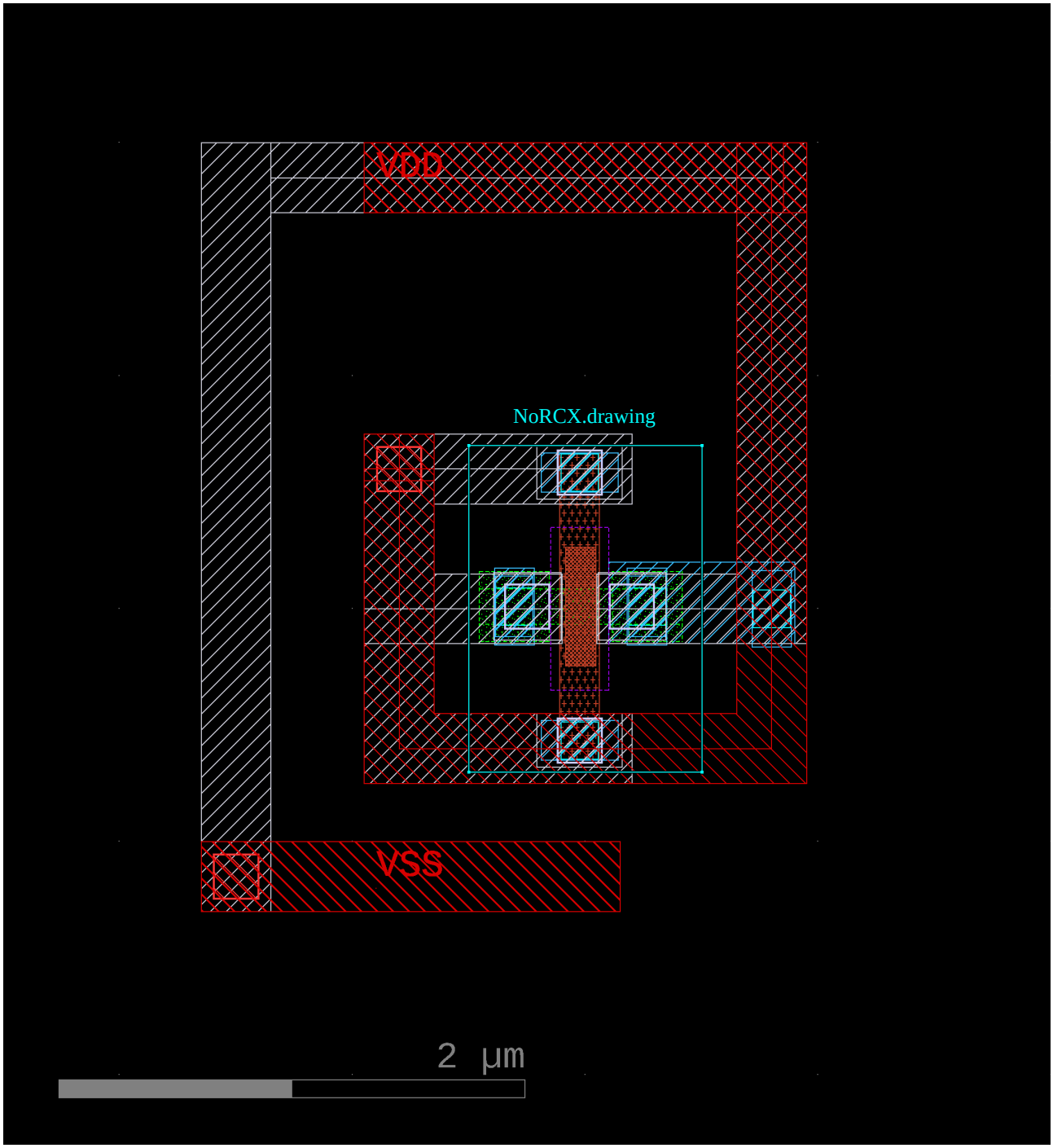


Figure 46: Test pattern `exclusion_nmos_diode2`

With `tie`, its terminals are on the nodes of the metal within the marker. With `ports`, each terminal is a port of its own, as the model connects them:

```
X$1 VSS__m0_2 VDD__m0_4 VDD__m0_5 sub! sg13_lv_nmos w=1.5e-07 l=1.7e-07
```

The bulk stays on the substrate, which is no conductor of the resistance network.

6.3.6 Markers of a Layer, of a Pair and of the User

`--exclusion-marker-layer` selects the markers, instead of those on by default. It takes the name of a marker of the PDK, or any GDS layer, for all layers of the stack (`200/0`) or for some (`200/0:met2,met3`). So a PDK without markers can have them too, such as `sky130A` and `gf180mcuD`, and `--exclusion-markers ignore` stays the default.

A marker of a layer leaves out only what is on that layer. In test pattern `exclusion_plate_noqrc`, a `Metal3` plate `P` is between a `Metal2` wire `A` and a `Metal4` wire `B`, with `Metal3.noqrc` over the plate. So `P` loses its capacitance to the substrate, and the wires keep all of theirs, with the plate as a shield between them. This is the case that Table 35 is about.

A pair marker only leaves out the capacitances between its two layers, so it has no effect on the resistances. The same plate with `m2m3` (`exclusion_plate_pair_m2m3`), and with `m3sub` (`exclusion_plate_pair_m3sub`):

Table 41: Markers of a layer and of a pair on `exclusion_plate_between_wires`

Nets	`ignore` [fF]	`Metal3.noqrc` [fF]	`NoRCX.m2m3` [fF]	`NoRCX.m3sub` [fF]
`A` ↔ `B`	1.387	1.387	1.387	1.387
`A` ↔ `P`	4.363	4.363	0.198	4.363
`B` ↔ `P`	4.831	4.831	4.831	4.831
`A` ↔ `VSUBS`	5.437	5.437	5.437	5.437
`B` ↔ `VSUBS`	1.862	1.862	1.862	1.862
`P` ↔ `VSUBS`	6.701		6.701	

The 0.198 fF that stay with `m2m3` are the fringe to the part of `A` beyond the marker.

6.3.7 Cutting Out a Cell

`--cut CELL[=PREFIX]` leaves the instances of a cell to a model. `CELL` is a name or a glob pattern, and a cell within a cut cell is part of it. Each `--cut` needs instances of its own: the first one that matches a cell takes them.

What is left out:

- The shapes of the cell, on the layers of the stack, so routing over it stays extracted, unlike with a marker over all layers:
  - metal and wells by their drawn layers, with the fill that LVS joins to them
  - what LVS computes from the diffusion (e.g. source and drain) by the diffusion of the cell
  - not its boundary or the overhang of its wells and implants, which take nothing of the layout under them
- The devices of the cell, also for `--check`. As the netlist of LVS is flat, they go by what the cell draws at their terminals, on the layers that LVS makes a device of that class from, which the tech info has (the diffusion and the gate of a MOS, the marker of a resistor, a diode, a BJT or a capacitor):

  - on each layer, the cell has to draw a polygon at all terminals, and more of them than the rest of the layout, as a diffusion can go on beyond a cell, and a tap can overlap the one of a neighbor
  - so a device stays if it's under the boundary or the fill of a cell, beside it on the same diffusion, or has only wires of the cell on its terminals
  - and fill, a boundary or a wire of the top cell leaves a device to the cell
- Diffusion that no layer draws, which LVS fills up for a device of the cell (e.g. the drain of an LDMOS, across the gap to its gate), but not what the rest of the layout draws

`--cut-keep-net NET` takes the shapes of a net out of the cell, also its wells, so that they are extracted like the rest of the layout, e.g. a guard ring.

Where a net enters the cell is a port of the netlist, as with `--exclusion-markers ports`, named:

- `<prefix>__<label>`, by the labels of the cell on the metal it is joined to within the cell, nearest first. Several cuts of that metal share the port, as they are one pin of the model
- `<prefix>__<terminal>`, for a terminal of a device of the cell on the metal around it, numbered if the block has several such nodes on a net (`X__S_0`, `X__S_1`)
- `<prefix>__<net>_<number>` otherwise

The prefix:

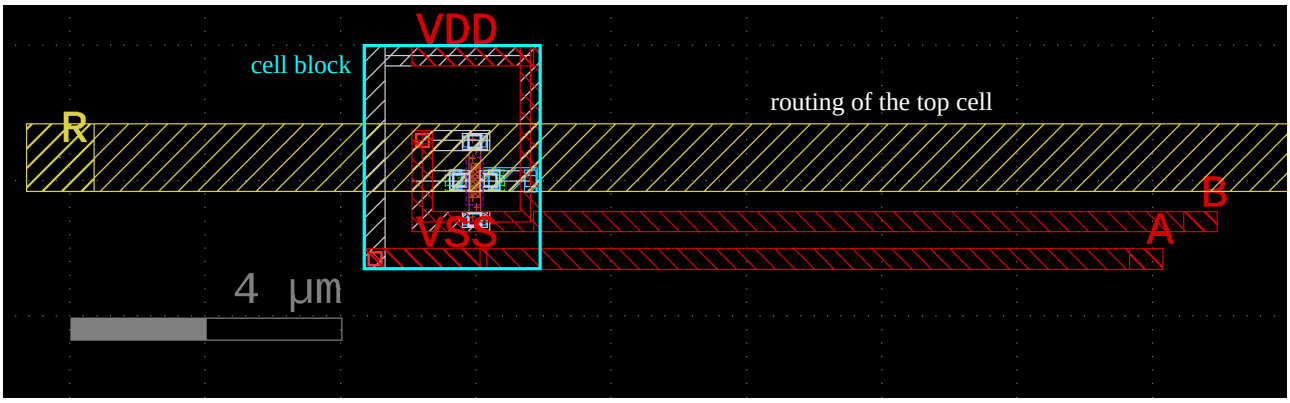
- is `PREFIX` for a single instance, with a number for several (`x_0`, `x_1`)
- is by default the name of the cell with a number per cell (`coil_0`)
- has letters, digits and `_` only, as it names ports
- has to differ for each instance, so `--cut a=S_0 --cut b=S` is an error if `b` has several instances

Test pattern `cut_block_with_mos` has the MOS of the design `nmos_diode2` in the cell `block`, whose pins `VDD` and `VSS` on `Metal3` the top cell wires on to `A` and `B`, and a `Metal5` wire `R` over the cell (Figure 47). With `--cut block=X`:

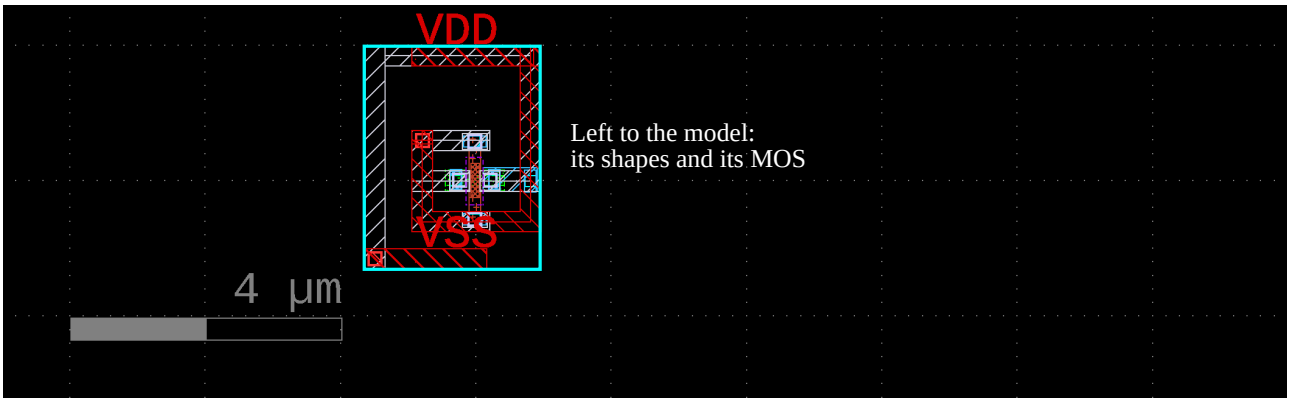
GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/cut_block_with_mos.gds.gz

```
.SUBCKT cut_block_with_mos A X__VSS B X__VDD sub! R
Cext_1 A B 0.397445f
Cext_2 A R 0.016564f
Cext_3 A sub! 0.197343f
Cext_4 A X__VDD 0.07352f
Cext_5 B R 0.0584666f
Cext_6 B sub! 0.221065f
Cext_7 R sub! 1.40824f
Cext_8 R X__VDD 0.194269f
Cext_9 R X__VSS 0.0809391f
Cext_10 sub! X__VDD 0.178802f
Cext_11 sub! X__VSS 0.178581f
Cext_12 X__VDD X__VSS 0.4502f
Rext_1 A X__VSS 2.86
Rext_2 B X__VDD 2.86
.ENDS cut_block_with_mos
```

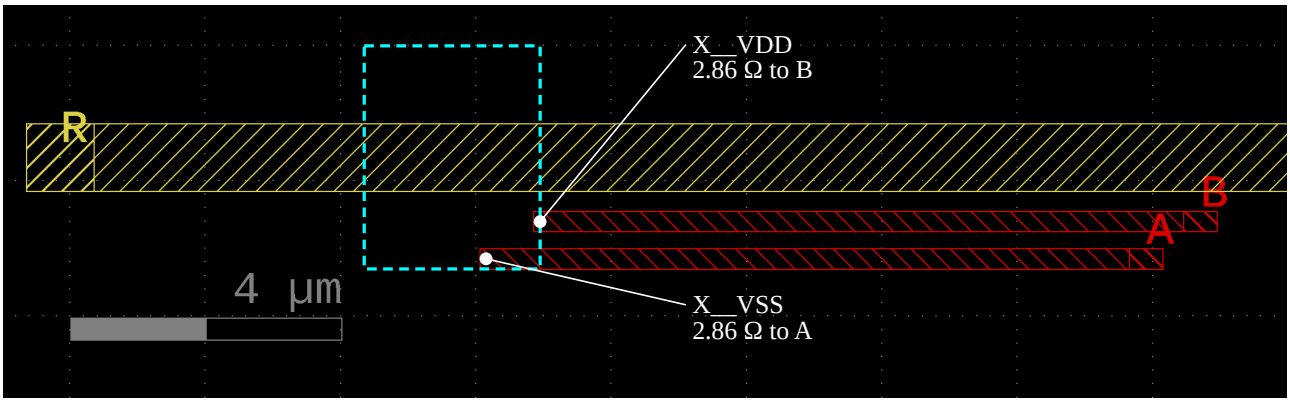
There is no MOS, the two nets end on the ports for the model, the wire `R` keeps its capacitances, also to the metal of the cell, and the 2.86 Ω are the ones of the wires of the top cell.



(a) The layout, with the cell `block` framed



(b) What the model stands for: the shapes and the devices of the cell



(c) What is extracted: the rest of the layout, with a port where a net enters the cell

Figure 47: Test pattern `cut_block_with_mos`, as drawn, and split by `--cut block=X`

6.3.8 Leaving Out a Net

`--exclude-net NET`, a name or a glob pattern, is for a net that something else models or drives, such as the power grid of a block:

- it has no resistance, so the netlist has one node for it, which also all of its device terminals are on
- it has no capacitance to the substrate
- the capacitances between it and other nets stay, as they are the load of those nets, also between two excluded nets, and to a well of another net

`exclusion_plate_neighbor` with `--exclude-net P` gives the values of `tie` in Table 39, without needing a marker or a GDS. `exclusion_nmos_diode2` with `--exclude-net 'V*'` in `--mode RC` has no resistor at all, and only 0.492 fF between `VDD` and `VSS`.

The values of the nets around it are as without the option, as the net stays in the field model. Unlike with a block, this goes by the name of the net, so that a large grid doesn't slow down the extraction.

6.3.9 Field Solvers

A field solver only tells the capacitances between its conductors, not where they end on them. So with these options, a net is one conductor for its part outside the blocks, and one for each set of blocks it is within, as these can overlap. The substrate is a conductor of its own, also if its net has metal. There are no panels between the conductors of a net, as those would be coincident. The matrix goes through the rule of Table 36, and the parts of each net are summed up: with all at the same potential, the charges they induce add up to the charge of the net as one conductor.

The netlist of a field solver has one node per net, so it has no ports, as with `--mode CC`. A warning says so, with `ports` and with `--cut`. It also has no capacitances between the parts of a net that only a model joins.

FasterCap gives the same picture as Table 39 and Table 45, with its own values:

Table 42: With FasterCap, which is no reference here, only a trend

Layout	Nets	as drawn [fF]	what is left out, left out [fF]
`exclusion_plate_neighbor`, `tie`	`N` ↔ `P`	2.16	2.16
	`N` ↔ substrate	3.59	3.59
	`P` ↔ substrate	8.24	
`exclusion_wire_end`, half of the wire within the marker, `tie`	`W` ↔ substrate	4.40	2.20
`exclusion_wire_through`, a third of it, `tie`	wire ↔ substrate	6.38	4.37
`cut_coil`, `--cut coil --cut-keep-net VDD`	coil ↔ `VC`	3.13	3.05
	coil ↔ `VDD`	4.92	4.93
	`VC` ↔ `VDD`	3.22	3.20
	`VC` ↔ substrate	7.23	7.20
	`VDD` ↔ substrate	26.28	26.10
	coil ↔ substrate	22.00	5.48

Where a net is two conductors, the values around it differ by up to 2.5 %, as the mesh differs at the ends of the parts.

6.3.10 Use Cases

6.3.10.1 Inductor of the PDK, with EM Model

The pcells draw `NoRCX.drawing`, and the LVS deck of `KPEX` recognizes the inductor as a subcircuit with its terminals at its leads, so a model replaces it by its name, without any ports. What `--exclusion-markers` changes is the rest of what is within its polygon, such as a wire under the coil (`exclusion_inductor2_wire_under`, Figure 48, with `--blackbox y`):

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/exclusion_inductor2_wire_under.gds.gz

Table 43: A wire under the coil of `inductor2`

	`ignore`	`tie`	`ports`
`Y1` ↔ `Y2`	4.356 Ω	0.858 Ω	0.429 Ω to each port
`Y` to substrate	7.044 fF	1.532 fF	1.532 fF
`LA`, `LB`	unchanged	unchanged	unchanged

So a model of the coil that has that wire in its simulation should be used with `tie` or `ports`, and one without with `ignore`.

This is what #121 asks for: most of a circuit with the 2.5D engine, and EM models for the sensitive RF parts.

6.3.10.2 Coil of LC Oscillator

#213 extracts the core of an oscillator without the coil of its tank, which has an EM-fitted model, but with its leads and its guard ring, with ports where the core meets the coil. Outside `KPEX`, this took a cut of the GDS, with workarounds for the recognition texts of devices and the labels of child cells.

Test pattern `cut_coil` (Figure 49) is a small layout with these parts, not that design:

- the cell `coil`, with a `TopMetal2` loop with stubs of its leads, a center tap on `TopMetal1`, and a `TopMetal1` guard ring around them, and the labels `LA`, `LB` and `CT`
- wires of the top cell, which go on from the stubs to `OSCP`, `OSCN` and `VCT`, and a stub with the label `VDD` on the guard ring
- a `Metal5` wire `VC` of the top cell under the coil

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/ihp-sg13g2/test_patterns/cut_coil.gds.gz

With `kpex extract --pdk ihp-sg13g2 --gds cut_coil.gds.gz --2.5D --mode RC --cut coil=xind --cut-keep-net VDD`:

```
.SUBCKT cut_coil OSCN OSCP VCT xind__CT xind__LA xind__LB VC VDD sub!
Cext_1 OSCN sub! 1.04552f
Cext_2 OSCP sub! 1.04552f
Cext_3 VC VDD 2.1086f
Cext_4 VC sub! 8.43019f
Cext_5 VC xind__CT 0.89613f
Cext_6 VC xind__LA 0.398954f
Cext_7 VC xind__LB 0.398954f
Cext_8 VCT sub! 1.17194f
Cext_9 VDD sub! 30.0624f
Cext_10 VDD xind__LA 0.110313f
Cext_11 VDD xind__LB 0.110313f
```

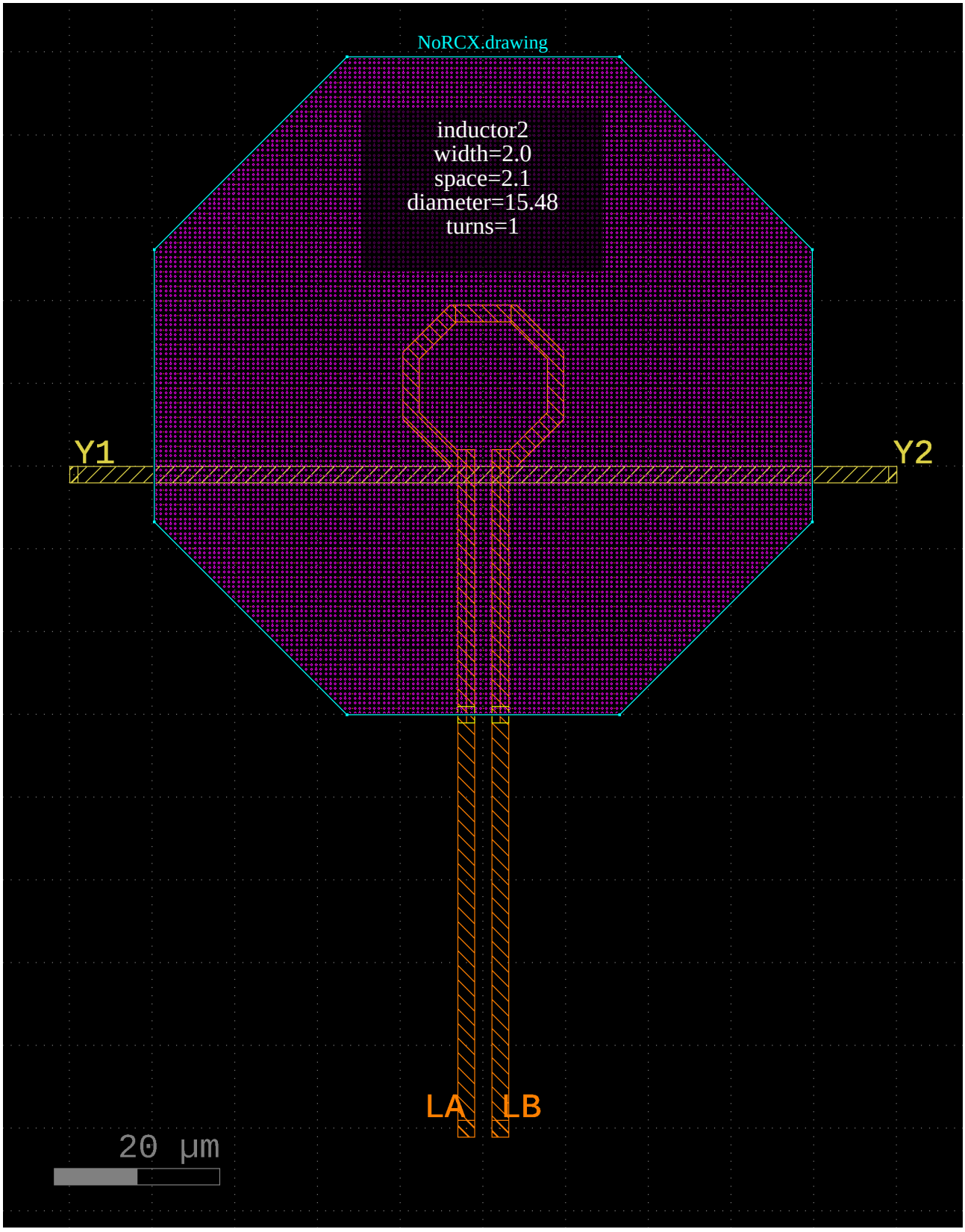


Figure 48: Test pattern `exclusion_inductor2_wire_under`, the marker is the same polygon as `IND`

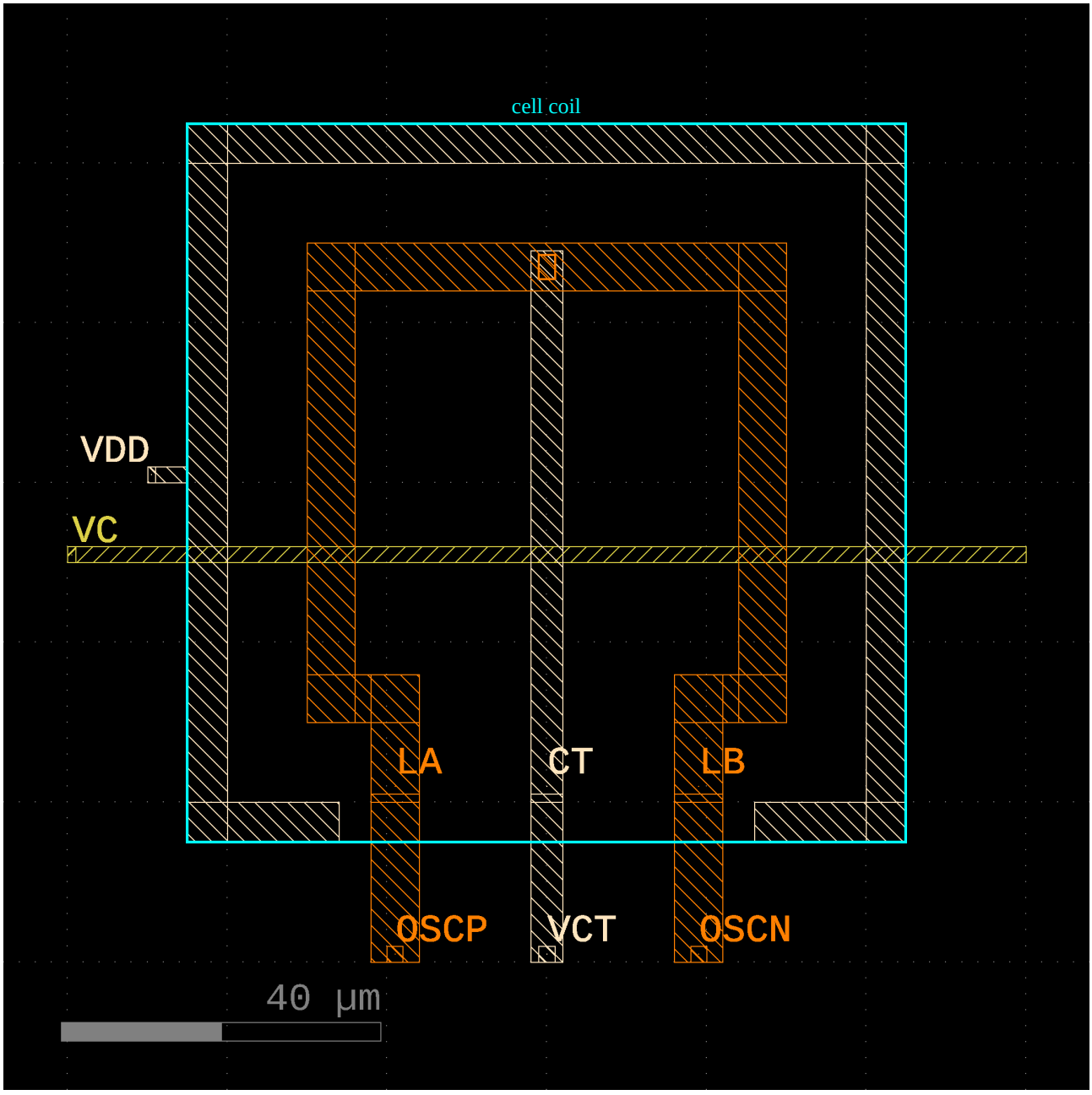


Figure 49: Test pattern `cut_coil`

```
Cext_12 sub! xind__CT 0.925084f
Cext_13 sub! xind__LA 0.703486f
Cext_14 sub! xind__LB 0.703486f
Rext_1 OSCN OSCN|OSCP|VCT 0.001
Rext_2 OSCN xind__LB 0.0348333333333
Rext_3 OSCP xind__LA 0.0348333333333
Rext_4 VCT xind__CT 0.0855
.ENDS cut_coil
```

The model goes between the ports, in the netlist that includes this one:

```
XCORE OSCN OSCP VCT xind__CT xind__LA xind__LB VC VDD 0 cut_coil
XL xind__LA xind__LB xind__CT coil_em_model
```

`--mode CC` shows what is left out, with the net of the coil being `OSCN,OSCP,VCT`:

Table 44: Capacitances of `cut_coil`

Nets	as drawn [fF]	with `--cut` [fF]
coil ↔ `VSUBS`	28.432	5.595
coil ↔ `VC`	1.694	1.694
coil ↔ `VDD`	0.221	0.221
`VC` ↔ `VDD`	2.109	2.109
`VC` ↔ `VSUBS`	8.430	8.430
`VDD` ↔ `VSUBS`	30.062	30.062

The wire `VC`, which is no shape of the cell, keeps all of its capacitances, also to the coil, which a model wouldn't know. Without `--cut-keep-net VDD`, the guard ring is left to the model too, and the net `VDD` enters the cell at the port `xind__VDD_0`.

To check that this fits, a test extracts the coil on its own (`cut_coil_model`), as a stand-in for a model, connects it between the ports, and compares the port admittance in `ngspice` with the one of the flat extraction:

- the resistances, between `OSCP`, `OSCN` and `VCT`, are the same within 0.1 %
- so are the values of `VC` and `VDD` to each other and to the substrate, and their capacitances to the coil in sum, which are at the ports instead of along the coil
- the coil to the substrate is 10 % higher, as a model on its own has no shielding by the wires around it

Compared with the cut of the GDS:

- running LVS on the whole layout keeps the recognition texts of the devices, and needs no selection of the labels of child cells
- the ports go by the metal, wherever a net enters, so that the leads need no pins, and the nets their names from the top level
- what a block has around it stays in the field model

`--exclusion-markers ports` does the same without a cell, for a marker drawn over the coil, inside the guard ring, as in test pattern `exclusion_coil_guard_ring` (Figure 50), whose labels name the ports. As a marker applies to all layers, routing through it is left out too.

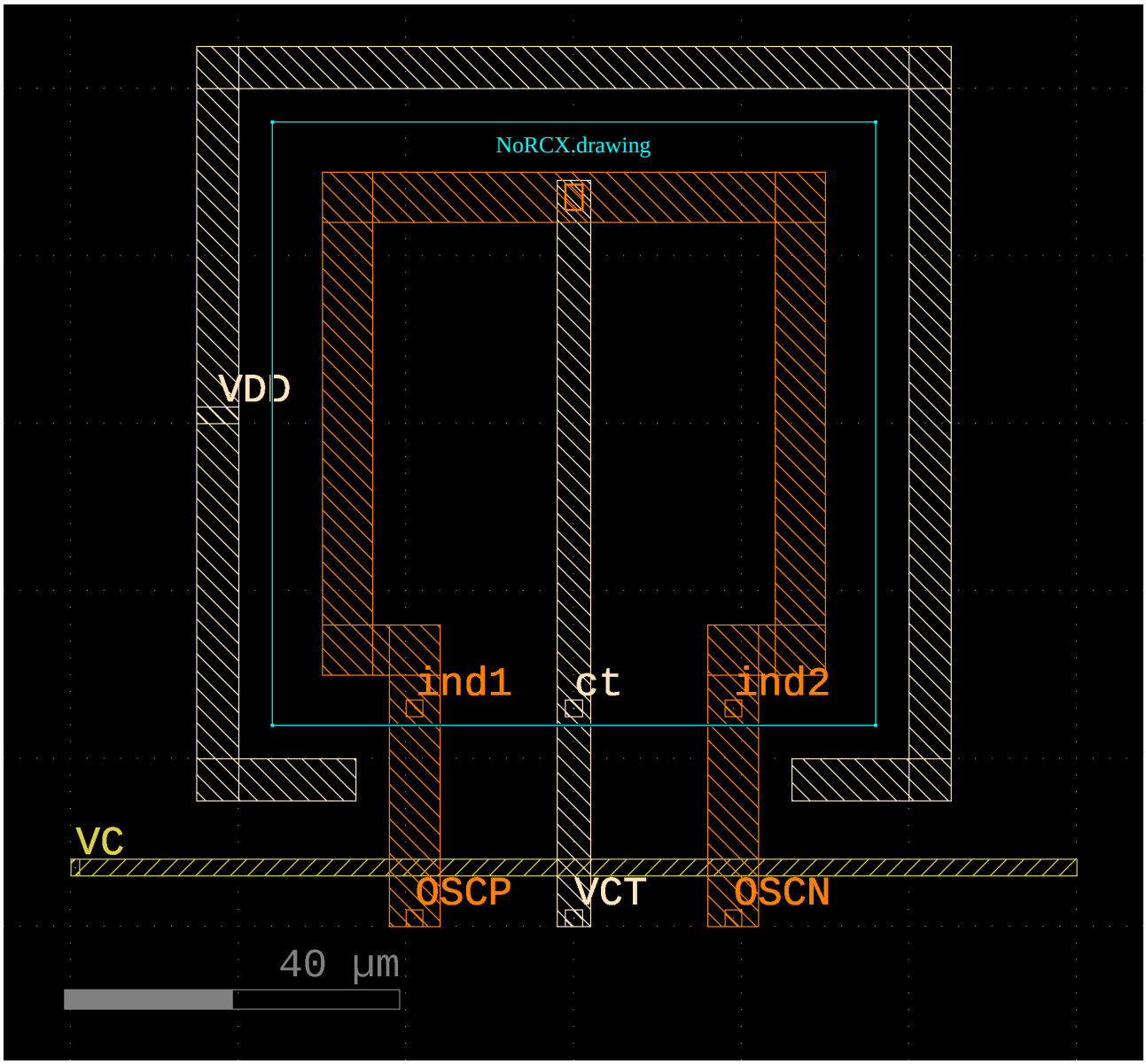


Figure 50: Test pattern `exclusion_coil_guard_ring`

Table 45: Capacitances of `exclusion_coil_guard_ring`

Nets	`ignore` [fF]	`tie`, `ports` [fF]
coil ↔ `VSUBS`	28.432	6.117
coil ↔ `VC`	1.658	1.658
coil ↔ `VDD`	0.221	0.221
`VC` ↔ `VDD`	0.056	0.056
`VC` ↔ `VSUBS`	8.430	8.430
`VDD` ↔ `VSUBS`	29.917	29.917

The second part of #213, every label a port for the wiring between hard blocks, is what the netlists of `--mode R` and `RC` do for the labels of a net in the top cell. Labels in the cells of the blocks become ports with `--cut`.

6.3.10.3 Power Grid of Block

#85 extracts a block whose power grid is part of the top level:

```
kpex extract --pdk ihp-sg13g2 --gds block.gds.gz --2.5D --mode RC \
             --exclude-net VDD --exclude-net VSS
```

So `VDD` and `VSS` are ideal nodes of the netlist, with all device terminals on them, and each signal keeps its load to them, as the grid stays in the field model. For debugging, leaving out the resistance of one net at a time shows what it adds to a simulation.

6.3.11 Limits

- A model whose simulation includes a neighbor counts their coupling twice, unless the neighbor is within the block too. The boundary of such hybrid flows seems to be little studied: the literature has the error of ports within an EM model (Rautio 2008), but no work on the double counting was found
- A model extracted or simulated on its own has no shielding by what is around it (Section 6.3.10)
- The coupling of a conductor outside to one within a block lands on the nearest node of its net, which is approximate in a large block
- KLayout's square counting, which the resistance network uses, has the current along the longest edge of each convex part of a polygon, also without these options. So a part beyond a cut that is shorter than it is wide has too little resistance, which matters on wells: two `nwell` taps of `r_ntap_pair_nwell_li1_m1`, with a marker of 2 µm between them across the `nwell` of 7 µm, have 212 Ω with `tie`, 1948 Ω by tessellation. Wires at an angle have too little too, a `Metal3` wire of 60 µm x 1 µm has 5.28 Ω at 0° and 2.65 Ω at 45°, and a corner of a marker on such a wire can give several times the value of the whole wire
- A net that goes on from one cut cell to another, where they abut, has no port between them, for which a warning names the net and the cells
- `--cut` of an instance at an angle other than a multiple of 90°, or of one with such an instance below it, is an error, as the LVS database only has the bounding boxes of the shapes of such an instance
- There is no comparison with silicon or an EM simulation yet: the reference of `tie`, `ports` and `--cut` is the flat extraction
- The exports of 2.5D scenes for other field solvers don't honor these options. MAGIC doesn't either, so they are an error together with it

6.4 Connecting the Extracted Netlist

A parasitic netlist has two jobs: describe the layout's electrical network, and expose an interface that the testbench can call correctly. A correct capacitance attached to the wrong positional pin is still a wrong simulation.

`--schematic` supplies two pieces of that interface to `KPEX/2.5D`: the order of the top-level ports, and names for devices that can be matched unambiguously. It does not replace a full LVS comparison. The checks in Section 8 answer narrower questions about missing devices, geometry totals, pins and connectivity.

6.4.1 Choosing Which Labels Become Ports

A label inside a child cell often names an internal net. Making every such label a top-level port can expose implementation details of a block to its testbench. Conversely, a layout without top-level pin labels may rely on the labels of its children.

Table 46: Choosing the port-label policy

Input and policy	Labels used for ports
`--gds`, `--port-labels auto`, top cell has a pin label	Top-level labels
`--gds`, `--port-labels auto`, no top-level pin label	Labels across the hierarchy
`--gds`, `--port-labels top`	Top-level labels, even when there are none
`--lvsdb` alone, `--port-labels auto`	Hierarchical labels retained by the database

`auto` is the default. `--port-labels top` needs `--gds`, including when an LVS database supplies the extracted nets. Substrate handling and top-level text-only ports remain in place. Selecting top-level labels does not remove the child geometry or the internal nets from extraction.

6.4.2 Keeping Positional Calls Correct

SPICE connects a subcircuit call by position. The first node in the call connects to the first port of its `.subckt` declaration, regardless of the spelling of the two names.

`KPEX/2.5D` defines the exported order as follows:

- With `--schematic`, the matching top subcircuit's ports come first, in schematic order
- Any remaining ports follow, sorted by name, comparing without case first and exact spelling second
- Without a usable schematic top, all exported ports use that name order
- Matching ignores case, while the exported spelling comes from the extracted net

This applies to `CC`, `C`, `R` and `RC`, to reduced netlists, to SPICE and `VACASK`, and to the port lists recorded in `kpex_result.json`. It changes the interface order, not which physical net each port connects to. In `CC` and `C`, several labels on one equipotential net can remain one exported port; its earliest schematic alias determines its position.

6.4.2.1 Transistor with Schematic Port Order

Download ordered.spice, whose top declaration is `.subckt nfet_li1 S G D` and whose transistor is named `Xtail`. From the `KPEX` source directory, run:

```
kpex extract --pdk sky130A --2.5D --mode CC --blackbox y \
  --gds testdata/designs/sky130A/test_patterns/nfet_li1.gds.gz \
  --schematic ordered.spice --out-netlist ordered-extracted.spice
```

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/nfet_li1.gds.gz

The declaration and device lines from that run are:

```
.SUBCKT nfet_li1 S G D sky130_gnd
Xtail D G S sky130_gnd sky130_fd_pr__nfet_01v8 l=0.15 w=0.65 as=0.169 ad=0.169
+ ps=1.82 pd=1.82
```

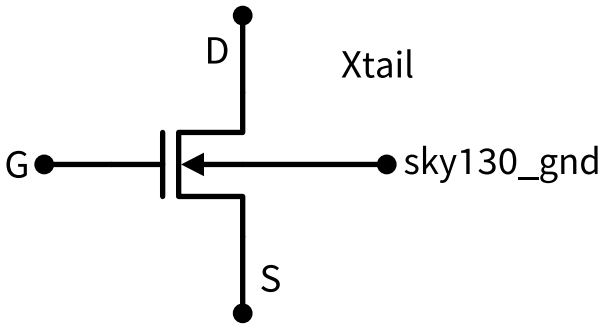


Figure 51: The named transistor and its terminal nets. The top-level port order is S, G, D, sky130_gnd; the model has its own D, G, S, bulk order. Parasitic capacitors are omitted here to make the interface visible.

The top-level interface starts with `S G D`. The substrate `sky130_gnd`, absent from the schematic's port declaration, follows them. The transistor line has its own model terminal order; it need not have the same order as the top-level interface. For this symmetric MOS device, exchanging source and drain still permits the match.

6.4.3 Following Schematic Device Names

Device naming needs a schematic `.subckt` whose name matches the layout top cell. Port ordering can additionally use the sole uncalled subcircuit when the top name differs; that fallback does not rename devices.

Matching uses the device mappings of the technology, connectivity and device geometry. A successful match retains the schematic name, as `Xtail` above. Devices inside schematic subcircuits can retain hierarchical paths such as `X1.Xtail`. Parallel layout fingers of one schematic MOS receive suffixes such as `_0`, `_1` and `_2`.

Names alone are not evidence that two devices correspond. Indistinguishable parallel schematic instances, missing model mappings or an ambiguous comparison keep layout-derived names and produce a warning. An assignment that would create duplicate output names is rejected. The model mapping still selects the proper SPICE designator, so a native schematic `M` may be emitted through an `X` wrapper when that is the PDK's model interface.

6.4.4 Keeping Geometry Units Separate from Electrical Values

The SPICE directives `.option scale=1u` and `.options scale=1u` set one geometry scale for the whole parsed netlist. They are not per-device or per-model settings. A declaration inside a subcircuit still supplies that same global setting.

For scale s, the schematic geometry evaluator applies:

Table 47: Geometry scaled while evaluating the schematic

Instance	Parameters	Effective value
Native `M`	`l`, `w`, `pd`, `ps`	Parameter value multiplied by s
Native `M`	`ad`, `as`	Parameter value multiplied by s^2
Native `R` or `C`	`l`, `w`	Parameter value multiplied by s
`X` call	Wrapper arguments	Unchanged at the wrapper interface
Any instance	Electrical values, multipliers and unrelated parameters	Unchanged

An `X` parameter named `w` is not automatically a length: its meaning belongs to the wrapper. If the wrapper is defined in the input, its native devices receive scaling after parameter evaluation. For an external model wrapper, the technology's device mapping translates its declared units into the layout units used for matching and checks.

This distinction avoids scaling twice. The technology mapping describes the model interface; `.options scale` describes the netlist's global geometry convention. Neither should be inferred from the other. Diode area factors and other model-specific parameters also cannot be treated as physical areas just because of their names.

The schematic reader accepts only `scale` on `.option` or `.options`. Keep simulator analyses, `.model`, `.include`, `.lib` and unrelated options in the testbench or model library, outside this input. It is a circuit description reader, not a complete simulation-deck reader.

The parser accepts a finite positive SPICE number, including suffixes. `1u` and `1000n` express the same scale. SPICE suffixes are case-insensitive: `M` means milli, and `MEG` means mega. Identical repeated declarations are accepted, while conflicting values, zero and negative scales are rejected.

The extraction argument `--scale y` is a different setting: it normalizes the fringe model at the halo boundary (Section 6.2). It does not rescale layout coordinates or SPICE device geometry.

6.4.5 Including Both Files in a `VACASK` Testbench

Conversion preserves the global setting in a companion options fragment. Download geometry-scale.spice, which contains a native `Mnative` and an external `Xwrapped` in parallel, both with `w=4 l=2`, under `.option scale=1u`:

```
netlist convert geometry-scale.spice --to vacask \
  --out-netlist geometry-scale.inc
```

The circuit include contains:

```
subckt pair (d g s b)
  Mnative (d g s b) nmos w=4 l=2 ad=3 as=5 pd=7 ps=11
  Xwrapped (d g s b) external w=4 l=2 ad=3 as=5 pd=7 ps=11
ends
```

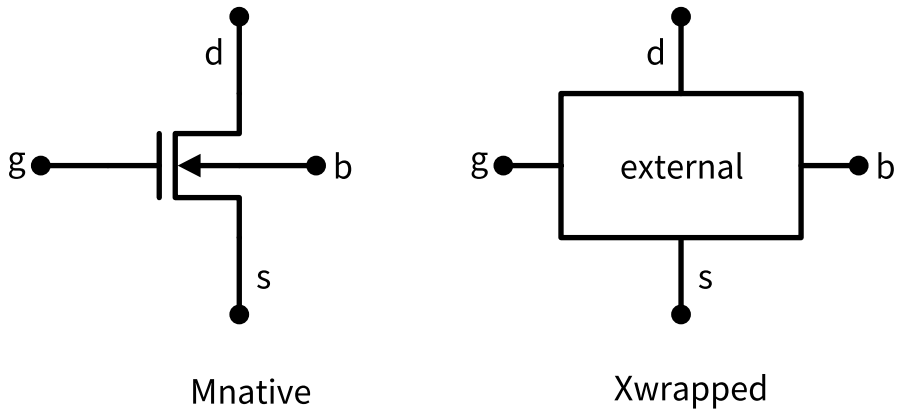


Figure 52: The native device and the external wrapper share d, g, s and b. Each native MOS receives the global scale once; the wrapper arguments retain their interface values.

The companion `geometry-scale.options.inc` contains:

```
options scale=1e-06
```

Include the circuit at circuit scope, and the options inside the testbench's existing `control` block, before analyses:

```
Geometry scale example

include "geometry-scale-models.inc"
include "geometry-scale.inc"
xpair (d g 0 0) pair
model supply vsource
vd (d 0) supply dc=1 mag=1
vg (g 0) supply dc=1

control
  include "geometry-scale.options.inc"
  analysis op1 op
  analysis ac1 ac values=[1e6]
endc
```

Save the complete testbench beside the converted files and run `vacask geometry-scale.sim`. Its operating-point and AC analyses both run with the fragment above.

The example model file defines `external` as a wrapper around the same native model. The native device therefore sees a width of 4 µm and length of 2 µm in both paths. The `X` arguments remain unchanged so that the wrapper passes them on in its expected units.

The options affect the **whole simulation**, including other included circuits and external wrappers. Combining circuits that assume different global scales requires normalizing their model interfaces first. Omitting the options file or applying a second scale to the instance values changes the geometry.

`kpex extract --netlist-format vacask --out-netlist extracted.inc` also writes a companion `extracted.options.inc`. Keep both files together, even when the options set `scale=1`. A converter does not translate model libraries: the testbench must supply compatible `VACASK` models. With PDK wrappers, pass `--pdk` to `netlist convert` so that multiplier handling follows their declared interfaces.

The examples were verified with `klayout-pex 0.9.0` and `KLayout 0.30.13`. The port-order and naming guarantees are independent of the capacitance coefficients that may change in subsequent releases.

7 Choosing Capacitance Coupling

Two neighboring conductors exchange current when their voltages change differently. Keeping that coupling is necessary for crosstalk and charge sharing. A grounded approximation instead describes the load of each conductor when the others are held at the reference potential.

> **Note**
>
> These options belong to the development version following `0.9.0`. The released `0.9.0` command line does not yet provide `--mode C`, `--coupling` or `netlist ground`.

7.1 Choosing the Extraction Mode

The analytical engine keeps coupling by default. Select grounded output explicitly:

Table 48: Extraction modes and their electrical meaning

Options after `--2.5D`	Wires	Capacitance
`--mode CC`	Ideal conductors	Coupled
`--mode C`	Ideal conductors	Grounded after floating elimination
`--mode RC`	Distributed resistance	Coupled
`--mode RC --coupling grounded`	Distributed resistance	Grounded after floating elimination
`--mode R`	Distributed resistance	None

`--mode CC --coupling grounded` is equivalent to `--mode C`. `--coupling coupled` explicitly retains the default policy for `CC` or `RC`. It cannot be combined with `C`. `R` has no capacitance policy to select.

For grounded capacitance with ideal interconnects:

```
kpex extract --pdk sky130A --2.5D --mode C --blackbox y \
  --gds layout.gds --out-netlist layout-c.spice
```

For grounded coupling while retaining the distributed resistance:

```
kpex extract --pdk sky130A --2.5D --mode RC \
  --coupling grounded --blackbox y \
  --gds layout.gds --out-netlist layout-rc.spice
```

The reference is the extracted substrate net, or `VSUBS` when no substrate net is available. It is not implicitly SPICE node `0`. Connect the exported reference to the intended testbench potential.

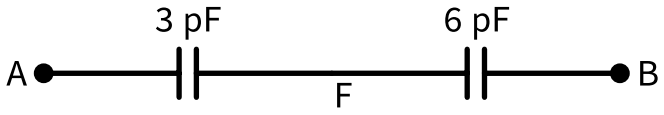
7.2 Eliminating Floating Conductors First

A floating conductor has no port or retained device terminal that can drive it. It can still transfer charge between other conductors. Connecting it to the reference would change that transfer, so floating capacitance is eliminated before any coupling is grounded.

Consider `A` ↔ `F` with 3 pF and `F` ↔ `B` with 6 pF, where `F` is floating. The two capacitors form a series connection:

$$C_{AB} = \frac{C_{AF}C_{FB}}{C_{AF} + C_{FB}} = \frac{3 \times 6}{3 + 6}\ \text{pF} = 2\ \text{pF}.$$

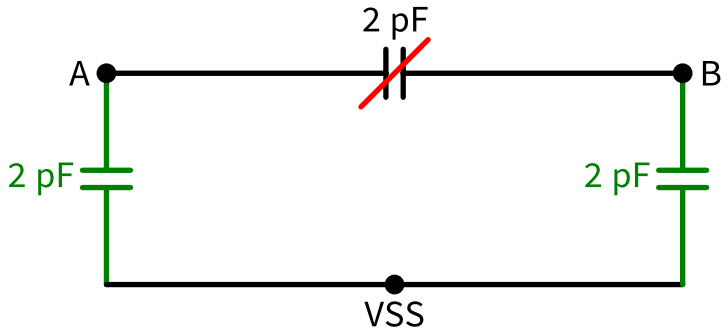
The first transformation is exact for this capacitor network. The subsequent replacement of `A` ↔ `B` by loads to the reference is an approximation.



(a) Coupled input: F has no external terminal.



(b) Exact elimination: the retained terminals see a 2 pF coupling.



(c) Grounded approximation: each endpoint receives the full 2 pF load.

Figure 53: Exact floating elimination precedes the grounded approximation. Red strikes through the removed coupling. Green identifies each newly introduced capacitor. `VSS` is an explicit reference, not a new connection to `F`.

7.2.1 Reproducing the Floating Example

Download floating-divider.spice:

```
.subckt divider A B VSS
Cext_af A F 3p
Cext_fb F B 6p
.ends divider
```

Run:

```
netlist ground floating-divider.spice --reference VSS \
  --out-netlist grounded-divider.spice
```

The written output contains:

```
.subckt divider A B VSS
* netlist ground: exact floating elimination, then full inter-net loads to VSS
Cgrnd_1 A VSS 2p
Cgrnd_2 B VSS 2p
.ends divider
```

No element ties `F` to `VSS`. The node disappears through elimination. Grounding `F` first would have produced unequal 3 pF and 6 pF loads instead of the two 2 pF loads above.

7.2.2 Extending the Elimination to a Network

For a floating node connected to several neighbors, eliminating its voltage adds a capacitance between every pair of neighbors:

$$C_{ij}^{\text{added}} = \frac{C_{iF} C_{Fj}}{\sum_k C_{Fk}}.$$

Existing parallel capacitances add to these contributions. Repeating the operation gives the Schur complement of the floating block of the capacitance matrix:

$$\mathbf{C}_{\text{retained}} = \mathbf{C}_{tt} - \mathbf{C}_{tf} \mathbf{C}_{ff}^{-1} \mathbf{C}_{ft}.$$

The expression explains the result. The calculation uses sparse node elimination. An isolated component with no observable connection contributes no terminal load. A floating component attached to only one retained node also contributes no load: its potential follows that node.

Ports created by labels and exclusion interfaces remain observable. The log names eliminated floating conductors, so an unintentionally disconnected supply can be distinguished from intended floating fill.

7.3 Replacing Coupling by Grounded Loads

After floating elimination, a capacitance C between distinct retained conductors becomes C to the reference at **each** endpoint. It is not divided in half.

For a single pair, with voltages measured relative to the reference:

$$\begin{bmatrix} i_A \\ i_B \end{bmatrix} = j\omega \begin{bmatrix} C & -C \\ -C & C \end{bmatrix} \begin{bmatrix} v_A \\ v_B \end{bmatrix} \quad \longrightarrow \quad j\omega \begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} v_A \\ v_B \end{bmatrix}.$$

This preserves the diagonal of the reduced nodal capacitance matrix after floating elimination. It removes transfer admittance between conductors. With distributed resistance, capacitance between nodes of the **same** conductor remains at its original endpoints.

This is a nodal capacitance property, not a general equality of RC port admittance. Holding an external port at the reference does not clamp every internal node behind its wire resistance.

For the ideal-conductor pair above, the approximation has a specific operating assumption:

Table 49: The load depends on how the neighboring conductor moves

Motion of `B` while `A` changes	Original current from `A` through the pair	Grounded model
`B` is held at the reference	$C\,dv_A/dt$	Same load
`B` follows `A`	Zero	Still charges its reference capacitor
`B` moves equally in the opposite direction	$2C\,dv_A/dt$	Uses $C\,dv_A/dt$

Grounded output therefore does not provide a universal worst case. Use coupled output to study crosstalk, differential switching or charge sharing. No switching multiplier or capacitance threshold is inferred.

7.4 Grounding an Existing Netlist

`netlist ground` performs the same transformation without loading a layout or PDK. It selects parasitic elements and one SPICE scope, retains the rest of the input, and writes a separate file.

7.4.1 Distributed Wires and Intentional Capacitance

Download coupled-rc.spice. `A1` belongs to conductor `A` through `Rext_a`. `B1` belongs to `B` through `Rext_b`. `F` is capacitor-only floating fill. `Cload` is an intentional capacitor that is outside the default parasitic selection.

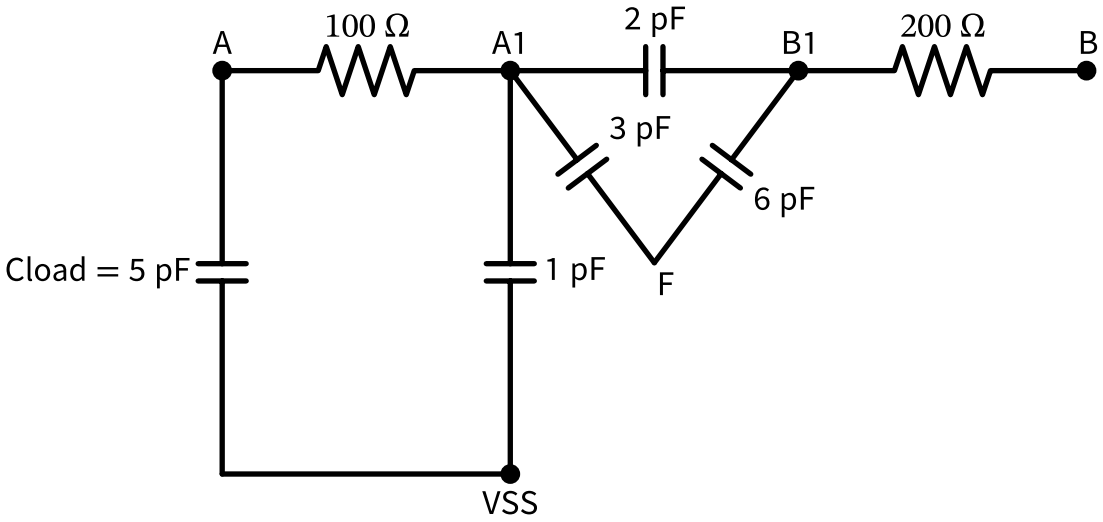
```
.subckt coupled A B VSS
Rext_a A A1 100
Rext_b B B1 200
Cext_ab A1 B1 2p
Cext_af A1 F 3p
Cext_fb F B1 6p
Cext_sub A1 VSS 1p
Cload A VSS 5p
.ends coupled
```

Run:

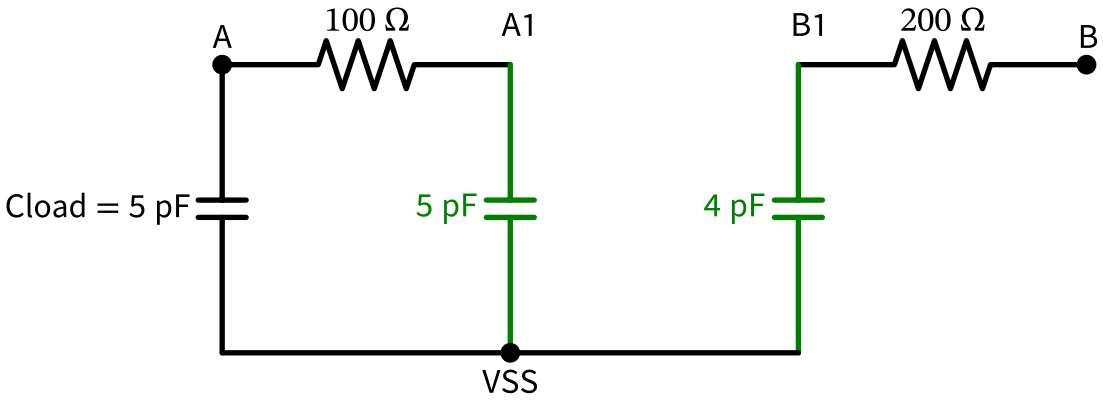
```
netlist ground coupled-rc.spice --reference VSS \
  --out-netlist grounded-rc.spice
```

Eliminating `F` adds 2 pF to the existing 2 pF coupling. The resulting 4 pF is grounded at `A1` and `B1`. `A1` already has 1 pF to `VSS`, giving 5 pF there. The written output is:

```
.subckt coupled A B VSS
Rext_a A A1 100
Rext_b B B1 200
* netlist ground: exact floating elimination, then full inter-net loads to VSS
Cgrnd_1 A1 VSS 5p
Cgrnd_2 B1 VSS 4p
Cload A VSS 5p
.ends coupled
```



(a) Coupled input: two resistive conductors, floating `F` and retained `Cload`.



(b) Grounded output: wire resistors and `Cload` retain their endpoints and values.

Figure 54: The example retains the distributed wire resistances. New grounded loads remain at `A1` and `B1`. `Cload` stays at port `A`.

The two resistor lines and `Cload` remain unchanged. The grounded loads stay behind the wire resistances. They are not moved to ports `A` and `B`. Existing capacitance to a distributed reference node also stays on that node, preserving its reference-side resistance.

7.4.2 Selecting Parasitics and the Reference

Table 50: Selecting the transformation without selecting intentional devices

Setting	Contract
`--reference NODE`	Existing node, port or global, or `0`/`gnd`, required for new loads
Default capacitor selection	`Cext_*` and `Cgrnd_*`
Default resistor selection	`Rext_*`, used to identify conductors
`--capacitor PATTERN`	Repeatable replacement for the default capacitor patterns
`--resistor PATTERN`	Repeatable replacement for the default resistor patterns
`--cell NAME`	Selects one subcircuit explicitly
`--top-level`	Selects elements outside subcircuits
`--keep NODE`	Retains the node and its entire conductor

Names and patterns are case-insensitive. Capacitor patterns must start with `C` and resistor patterns with `R`, so a bare `*` is rejected. Quote patterns so the shell does not expand them. For a different extractor's naming convention:

```
netlist ground extracted.spice --cell amplifier --reference VSS \
  --capacitor 'Cpar_*' --resistor 'Rwire_*' --keep bias_internal
```

Subcircuit ports, globals, terminals on retained device/source lines and `--keep` nodes retain their conductors. A capacitance does not make a DC connection between conductors. Selected resistors do, but their finite values and textual representation remain in the output.

The reference is local to the selected scope. SPICE ground aliases `0` and `gnd` identify the same reference conductor for this transformation. Unrelated names such as `VSS` do not become ground implicitly.

7.4.3 Identifying Disconnected Pieces of One Conductor

A netlist alone may not reveal that two disconnected wire pieces belong to one original layout net. Supply that identity explicitly in a flat JSON object:

```
{"OUT.1": "OUT", "OUT.2": "OUT"}
```

Save it as `nodes.json` and add `--net-map nodes.json`. The map groups conductor identities for the coupling decision. It does not short the nodes, rename them or add a resistor. It cannot assign different conductors to nodes already joined by selected resistors.

7.5 Keeping the Approximation Separate from Other Checks

Grounding occurs after capacitance has been assigned to distributed nodes and before optional network reduction. A reduction check therefore compares the grounded network with its further reduced version. It does not certify equivalence to the original coupled network.

The geometry reports and `--check-device-caps` continue to describe the coupled extraction. Intentional capacitor models remain in place with `--blackbox y`. When a layout contains metal capacitors that white-box extraction would remove, grounded output asks for `--blackbox y` before capacitance extraction. Once device capacitance and parasitics have been mixed in a plain netlist, postprocessing cannot recover that distinction.

7.6 Limits

- `RC` grounding rejects floating islands containing resistance because their frequency-dependent response cannot generally be replaced by fixed capacitors
- `C` uses ideal conductors and can eliminate their floating capacitance network exactly
- Exact elimination stops if a remaining floating node exceeds the 64-neighbor fill-in limit after other floating nodes have been removed
- Standalone selection requires numeric two-node resistor and capacitor primitives
- Selected resistor values must be strictly positive
- Symbolic values and ambiguous scopes produce diagnostics
- `netlist ground` refuses to overwrite its input and validates the selection before writing output
- The extraction options apply only to `--2.5D`

Keep coupled output for unsupported floating networks. Grounded output does not tie a remaining floating island to the reference as a fallback.

8 Checking Extracted Netlists

A plausible simulation result does not establish that an extracted netlist is complete. A device terminal can be disconnected from the resistance of its wire, a port can be missing, or a reduction can remove the loss that matters at the operating frequency. These failures need different comparisons.

`KPEX/2.5D` checks the following stages:

Table 51: Checks answer different questions.

Check	Compared quantities	Enabled by
Netlist structure	Connectivity, devices, ports and capacitance totals before reduction	Automatic, with failure policy controlled by `--check`
Schematic consistency	Device classes, aggregate device dimensions and top-level pins	`--schematic`
Reduction response	Terminal admittance of the written original and reduced R/C networks	`--check-frequency` or `--check-frequency-range`
Device capacitance	Marker-internal plate coupling and independently selected device reference	`--check-device-caps`

The structural checks guard the translation from layout connectivity into a netlist. The admittance check guards a subsequent approximation. The device comparison examines the physical capacitance estimate itself. Passing one does not imply that the other two pass.

8.1 Preserving Layout Connectivity

Resistance extraction splits one layout net into many nodes. These node names need not resemble the original net name, and device names may come from the schematic (Section 6.4.3). The structural comparison therefore follows the layout's net identities and device identities, rather than trying to infer connectivity from spelling.

For each original net, `KPEX` follows the connected component formed by its parasitic resistors. Capacitors do not join these components: they couple nets electrically, but do not make a DC connection.

The comparison looks for:

- Original nets missing from the expanded netlist

- Resistors that incorrectly join two original nets
- Resistor components that reach no original net or model interface port
- Device terminals attached to the wrong net's resistor network
- Device terminals attached to a different node than their extracted terminal port
- Missing device instances, or retained whiteboxed devices that would count their capacitance twice
- Missing ports and ports that touch nothing, with the exceptions required by exclusion interfaces
- Node names that collide when SPICE treats names without regard to case

For capacitance, the comparison first maps the distributed nodes back to their original nets. It then sums the parasitic capacitors for every pair of nets, and compares these sums with the extraction summary. The relative tolerance is 10^{-9}; this comparison allows rounding, not a physical modeling error. A capacitance to the substrate is omitted from this comparison if the chosen substrate connection shorts its two endpoints together.

8.1.1 Choosing Whether a Structural Problem Stops the Run

In `--mode RC`, structural problems fail the run by default. In the other modes they are warnings by default. `--check y` makes them failures explicitly, and `--check n` accepts the netlist while keeping the warnings visible.

The checks run on the full extracted network before reduction. A reduction therefore cannot hide a structural defect by removing the affected nodes. For a structural failure found after extraction, the netlist remains available for inspection.

A separate warning identifies nets with parasitic elements but no device terminal or port that can drive them. Floating fill can legitimately produce this warning. A supply rail under a wrong label can produce it too, so it deserves review. This warning alone does not fail `--check`.

8.2 Comparing with the Schematic

With `--schematic`, `KPEX` also compares the extracted devices with the schematic, using the technology's device mappings (Section 3.6). These checks supplement LVS; they do not perform a complete schematic-versus-layout topology comparison.

8.2.1 Comparing Device Quantities

Counting instances is insufficient. One schematic MOS instance can become several parallel fingers in the layout. Conversely, a layout extractor can combine devices. The comparison uses totals that survive these representations:

- Device-class presence
- Total WL when both dimensions are mapped, or total area when an area is mapped instead
- Total W for mapped MOS devices
- Total mapped multiplicity, such as emitter count

Multipliers and the number of hierarchical instances are included. Schematic dimensions are converted back to the LVS units before summation, including the SPICE geometry scaling discussed in Section 6.4.4. The relative tolerance for device totals is 10^{-6}.

A missing parameter is not silently replaced by an invented value. If the schematic leaves a relevant quantity to a model default, the corresponding total is left out and reported. A class without mapped totals is checked for presence only. Equal totals can still conceal an incorrectly wired device, which is why they do not replace LVS.

8.2.2 Comparing Pins and Ports

The top schematic subcircuit is selected by the layout cell name, without regard to case. If that name is absent, the schematic must have exactly one subcircuit that no other element calls.

The comparison reports schematic pins without extracted ports, and extracted ports without schematic pins. It accounts for interfaces that extraction adds or treats specially:

- The selected substrate net may be global in the schematic
- Exclusion-marker and cut-cell model ports are extra interfaces
- A schematic pin that reaches only mapped bulk terminals can be absent from the extracted ports
- A bulk-only pin is compared normally if an extracted port has its name

Port order is a separate interface contract, described in Section 6.4.2.

8.3 Checking the Response after Reduction

A smaller netlist should present nearly the same electrical response to its surroundings. Rather than comparing resistor counts or individual component values, the admittance check asks:

> For the same terminal voltages at this frequency, how much do the resulting terminal currents change?

The check is optional because exact elimination can be more expensive than the reduction itself. It works with `netlist reduce`, and with `kpex extract --2.5D --reduce floating` or `--reduce ticer` in `R` or `RC` mode.

8.3.1 From Branches to Terminal Admittance

At frequency f, a branch with conductance g and capacitance c has complex admittance

$$y(f) = g + j\omega c, \qquad \omega = 2\pi f.$$

A branch between nodes a and b adds $+y$ to both diagonal entries of the nodal admittance matrix, and $-y$ to the two off-diagonal entries. This gives

$$\mathbf{i} = \mathbf{Y}(f)\,\mathbf{v}, \qquad \mathbf{Y}(f) = \mathbf{G} + j\omega\mathbf{C}.$$

Partition the nodes into terminals t that the surrounding circuit can observe, and internal nodes i that it cannot drive directly:

$$\begin{bmatrix}\mathbf{i}_t \\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{Y}_{tt} & \mathbf{Y}_{ti} \\ \mathbf{Y}_{it} & \mathbf{Y}_{ii}\end{bmatrix}\begin{bmatrix}\mathbf{v}_t \\ \mathbf{v}_i\end{bmatrix}.$$

Eliminating the internal voltages gives the terminal response

$$\mathbf{Y}_{\text{term}} = \mathbf{Y}_{tt} - \mathbf{Y}_{ti}\mathbf{Y}_{ii}^{-1}\mathbf{Y}_{it}.$$

This is a Schur complement. The implementation performs sparse, complex node elimination rather than forming a dense inverse. For one internal star node, eliminating it adds a branch between each pair of its neighbors:

$$y_{ab}^{\text{added}} = \frac{y_{an}y_{nb}}{\sum_k y_{nk}}.$$

This evaluation uses the full complex admittances at the requested frequency. It is independent of the approximate low-frequency reduction described in Section 10.

The terminals include the subcircuit ports, global and ground nodes, nodes touched by non-plain-R/C lines, and nodes selected with `netlist reduce --keep`. In particular, device and subcircuit terminals remain observable. The global reference `0` is also included so that added tie resistors affect the comparison.

Each subcircuit and the top level are evaluated separately. Components with no path to an observed terminal at that frequency do not contribute. At DC, capacitors are open circuits. The check does not add artificial ground conductances to make an isolated component solvable.

8.3.2 Comparing the Two Matrices

The reported error is

$$\varepsilon(f) = \frac{\left\|\mathbf{Y}_{\text{reduced}}(f) - \mathbf{Y}_{\text{original}}(f)\right\|_F}{\left\|\mathbf{Y}_{\text{original}}(f)\right\|_F}, \qquad \|\mathbf{Y}\|_F = \sqrt{\sum_{a,b} |Y_{ab}|^2}.$$

Both real and imaginary parts contribute. The check reads the written netlists, so its result includes rounded component values and any real tie resistors introduced by reduction.

The default limit is `--check-max-relative-error 0.01`, or 1%. Equality with the limit passes. The limit must be strictly between zero and one.

There is no absolute-error floor. A tiny capacitance is still compared relatively. Two zero matrices have zero error, while a nonzero reduced response against a zero original response has infinite error.

8.3.3 Choosing Frequencies

Use a single frequency when one operating point is the question. Use a range when the circuit must behave across a band:

```
netlist reduce extracted.spice \
  --check-frequency-range 1Meg 10G \
  --check-points-per-decade 10 \
  --check-include-dc
```

`--check-frequency-range` takes **two arguments**, the start and stop frequencies. SPICE suffixes are accepted. The endpoints are included, and the samples are logarithmically spaced. The default density is ten points per decade. A partial final decade still includes the exact upper endpoint.

The range requires $0 < f_{\text{start}} \leq f_{\text{stop}}$. Equal endpoints produce one positive-frequency sample. `--check-include-dc` adds a separate sample at zero. For a DC-only check, use `--check-frequency 0` instead.

Every sample in every checked scope must pass. The report names the worst sampled error and its frequency. A passing sweep covers those samples, not every frequency between them. Increase the density near an important transition or inspect a simulator response when that distinction matters.

8.3.4 Example: Losing the Series Resistance of a Capacitor

Download the original `series-rc.spice`. The internal node `n` has a time constant $\tau = RC = 100\,\text{ps}$. Choosing a threshold of `1n` allows its elimination:

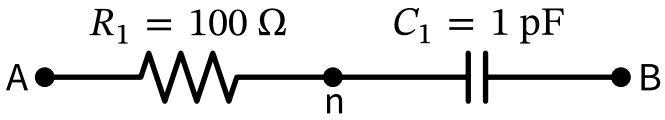
```
netlist reduce series-rc.spice --ticer-tau 1n \
  --check-frequency-range 1Meg 1G \
  --check-points-per-decade 2 --check-include-dc \
  --out-netlist series-rc-reduced.spice
```

The input contains:

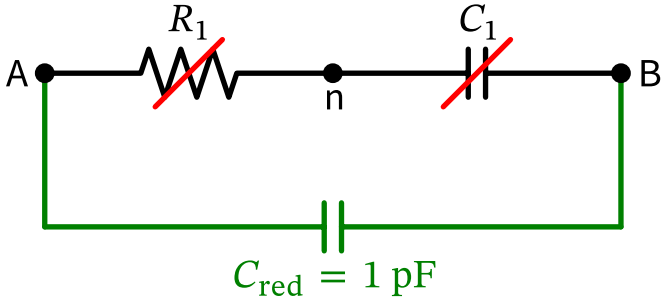
```
.SUBCKT series_rc A B
R1 A n 100
C1 n B 1p
.ENDS series_rc
```

The written reduced netlist contains:

```
.SUBCKT series_rc A B
Cred_1 A B 1p
.ENDS series_rc
```



(a) Original: the internal node stores charge and the resistor dissipates energy.



(b) Aggressive reduction: the capacitor remains, but its series loss disappears.

Figure 55: `A` ↔ `B` before and after eliminating `n`. Red strokes mark removed elements, and green marks the new element.

For this circuit the original and reduced branch admittances are

$$y_{\text{original}} = \frac{j\omega C}{1 + j\omega RC}, \qquad y_{\text{reduced}} = j\omega C.$$

Their two-terminal matrices have the same sign pattern, so the Frobenius error reduces to the scalar relative error:

$$\varepsilon(f) = \frac{|y_{\text{reduced}} - y_{\text{original}}|}{|y_{\text{original}}|} = \omega RC \quad (f > 0).$$

The run checks eight points, including DC, and exits with code `1`. Its reported worst relative Frobenius error is `0.628319` at `1G` Hz, against the `0.01` limit. At `1Meg` Hz alone it would pass, with relative error `0.000628319`.

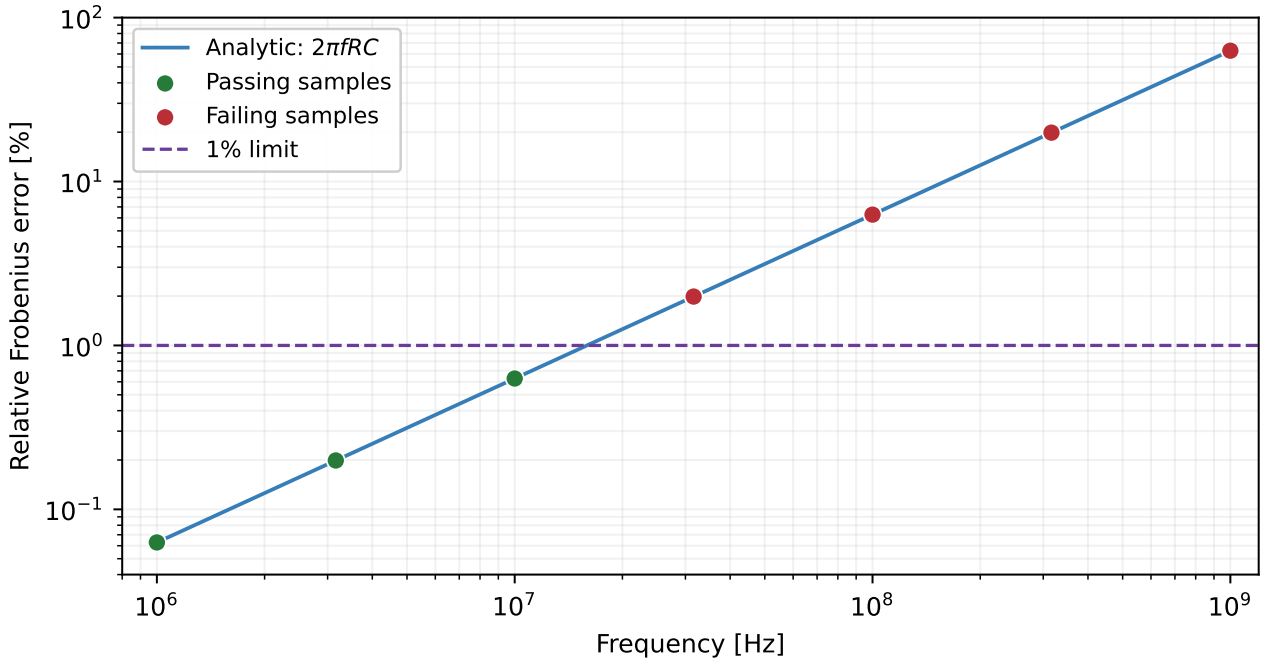


Figure 56: The sampled check follows the analytic error of the series RC example. Passing at 1 MHz does not cover 1 GHz. DC passes because both networks are open there.

A smaller threshold preserves the internal node in this example:

```
netlist reduce series-rc.spice --ticer-tau 1p \
  --check-frequency-range 1Meg 1G \
  --check-points-per-decade 2 --check-include-dc \
  --out-netlist series-rc-kept.spice
```

This run exits with code `0` and reports zero error at every sampled frequency. It keeps one resistor, one capacitor and three nodes. In a larger network, reducing the threshold usually trades fewer eliminated nodes for a closer response.

8.3.5 Limits of the Admittance Check

The matrix comparison covers the plain, numerically valued resistors and capacitors that the reduction reads. It does not simulate nonlinear device models, resolve a transistor's bias point, or establish that the extracted parasitic values match silicon.

A relative norm over the whole matrix is not a separate relative bound on every small matrix entry. Nor does a small admittance error guarantee an acceptable quality factor: a small dissipative part can matter even when the much larger capacitive response dominates the norm. Keep the full network or use a sufficiently small reduction threshold when loss is important, and inspect the relevant circuit response.

Exact elimination can create many new couplings. The check limits each evaluation to 100,000 nodes/coupling entries and 1,000,000 coupling updates. It also limits the complete request to 1,001 frequency points and 10,000,000 graph entries/coupling updates across all scopes and samples. Exceeding a limit, or encountering an unevaluable numerical result, fails the requested check rather than silently accepting an unchecked network.

8.4 Comparing Device Capacitance with a Model

In KPEX/2.5D, a blackboxed capacitor keeps its device model in the netlist. The extractor suppresses coupling between its internal plates so that the netlist does not count that capacitance twice. Ordinarily, that suppression also hides a useful diagnostic: how closely would the geometry-based extraction agree with the device's nominal capacitance?

For field solvers, retaining a metal-capacitor model needs intrinsic device ownership that is not yet implemented; Section 2.2 explains the guard and supported alternatives.

`--check-device-caps` observes those otherwise suppressed contributions. It does not add them to the exported netlist. The prerequisites are `--gds`, `--2.5D`, `--blackbox y`, and `--mode CC` or `--mode RC`.

8.4.1 Selecting the Geometry

The comparison identifies the selected device's recognition marker in the original layout, using its terminal geometry and the recognition layers declared by the technology. It then observes coupling only when both participating plate shapes:

- Belong to that same marker
- Belong to the selected pair of terminal nets
- Lie wholly inside the marker

The plate polygons are tagged, rather than clipped. Clipping at the marker boundary would manufacture new edges and therefore change sidewall and fringe contributions. A plate crossing that boundary is reported as unevaluable instead.

The comparison accumulates overlap, sidewall and fringe contributions separately:

$$C_{\text{geometry}} = C_{\text{overlap}} + C_{\text{sidewall}} + C_{\text{fringe}}, \qquad r = \frac{C_{\text{geometry}}}{C_{\text{reference}}}.$$

This is a device-local measurement. Substrate capacitance, coupling to external routing, and other capacitors connected between the same two net names are not a substitute for it. Output exclusions do not turn it into a subtraction of two whole-net capacitance totals.

Ambiguous geometry is not guessed. Missing or overlapping recognition markers, a marker containing another device's terminals, shorted selected terminals, and missing internal plate geometry are reported as errors or unevaluable comparisons.

8.4.2 Using the Selected Technology Reference

With no filename after `--check-device-caps`, the reference comes from the technology's device mapping (Section 3.6.1). The selected device-reference component supplies an expression, its valid parameter domain, a source and its assumptions. It is evaluated using the mapped device parameters. A nonnominal scenario must explicitly select a matching reference component; Section 3.9.13 describes the supported MIM combinations. The default nominal comparison remains unchanged.

KPEX does not search simulator model libraries at runtime or use an arbitrary LVS capacitance placeholder as the reference. Unsupported metal capacitors are listed as unchecked. If the layout has no available references, the requested comparison fails with an explanation.

There are **no default ratio bounds**. An evaluable automatic comparison is `informational`, not a declaration that the extraction is accurate enough for a circuit. This matters especially when the nominal model is itself an engineering estimate.

8.4.3 Example: MOM Plates below a Separate Route

GDS: https://github.com/iic-jku/klayout-pex/blob/v0.9.0/testdata/designs/ihp-sg13g2/test_patterns/cap_cmomf_w10u_l25u_m2_m4_m5_wire.gds.gz

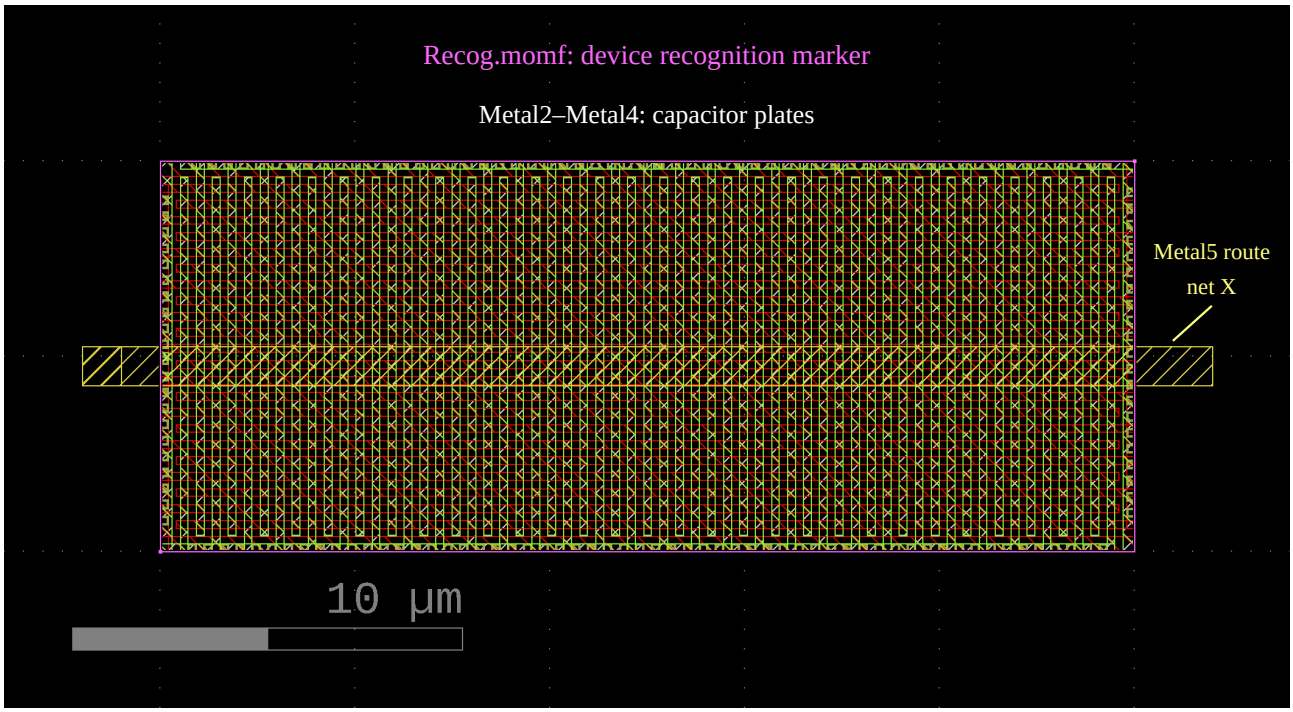


Figure 57: The MOM plates occupy the recognition marker. The separate `Metal5` route crosses above them; its coupling is external to the device comparison.

The `ihp-sg13g2` example has a `cap_cmomf` device with `Metal2`–`Metal4` plates, width 10 μm and length 25 μm. A separate `Metal5` route on net `X` crosses above it. The model reference is the nominal mutual capacitance of `mim_top` ↔ `mim_btm`, with no corner or mismatch variation.

Set `KPEX_DIR` to the source checkout. This command uses its committed LVS database as well as the GDS, so that the comparison can find the original recognition marker. Here `--scale y` normalizes the fringe response at the halo, independently of SPICE geometry scaling:

```
pattern=cap_cmomf_w10u_l25u_m2_m4_m5_wire
layouts="$KPEX_DIR/testdata/designs/ihp-sg13g2/test_patterns"
lvsdbs="$KPEX_DIR/testdata/lvsdb/ihp-sg13g2/test_patterns"
kpex extract --pdk ihp-sg13g2 --2.5D --mode CC \
  --blackbox y --halo 8 --scale y \
  --gds "$layouts/$pattern.gds.gz" \
  --lvsdb "$lvsdbs/$pattern.lvsdb.gz" \
  --check-device-caps --out-dir checks-mom \
  --out-result checks-mom.json
```

With `klayout-pex 0.9.0`, including the sidewall counting correction and refreshed nominal coefficients, the comparison reports:

```
$1 (cap_cmomf), mim_top / mim_btm:
marker coupling 236.403 fF, model reference 228.75 fF,
ratio 1.03345 (informational)
```

The device comparison is kept separately from the parasitic netlist. The netlist retains the compact-model instance and the external couplings, including those to `X` and the substrate. `checks-mom.json` records the comparison in `device_capacitance_checks`, including the three contributions in farads, the ratio, the reference source and its assumptions.

8.4.4 Supplying References and Acceptance Bounds

An explicit JSON file replaces the automatic selection. It can select one device and terminal pair, supply an independent reference, and optionally make the comparison a pass/fail check. For the preceding example, this reference file specifies an illustrative ±5% acceptance interval:

```
{
  "format_version": 1,
  "references": [
    {
      "device": "$1",
      "device_class": "cap_cmomf",
      "terminals": ["mim_top", "mim_btm"],
      "capacitance": "228.75f",
      "min_ratio": 0.95,
      "max_ratio": 1.05
    }
  ]
}
```

Repeat the example command with `--check-device-caps mom-reference.json`. These bounds pass for the recorded ratio `1.03345`, producing exit code `0`.

8.4.4.1 Rejected Comparison

To exercise the failure path, change `max_ratio` in the reference file to `1.0` and repeat the same command. The measured ratio now exceeds the upper bound, producing exit code `1`. The extracted netlist and the comparison remain available to inspect.

These bounds illustrate the mechanism. Choose a justified acceptance interval for the model, its assumptions and the intended use. Passing an interval does not establish calibration.

A numeric `capacitance` is in farads, and a string accepts SPICE suffixes. The capacitance must be finite and positive. Ratio bounds are optional, finite and nonnegative, with `min_ratio` no greater than `max_ratio`. Bounds are inclusive. Omitting both leaves the result informational. Duplicate references for the same device and unordered terminal pair are rejected.

`device` and `device_class` identify the device in the extracted LVS circuit, before schematic renaming. Use the names in `device_capacitance_checks` or the LVS database. The exported netlist may use different schematic-derived names (Section 6.4.3). Terminal names must match that LVS device class.

8.5 Acting on a Check Result

The different failure policies remain independent:

Table 52: Interpret each check according to what it measures.

Result	Meaning	Next action
Structural problem	The extracted netlist does not preserve a checked connectivity or quantity invariant	Inspect the reported terminal, port or net before simulation
Admittance error above the limit	Reduction changed the sampled terminal response too much	Reduce the time-constant threshold or retain the full network
Device ratio outside supplied bounds	Geometry and the chosen reference disagree beyond the selected interval	Review geometry, coefficients, reference assumptions and bounds
Informational device ratio	A comparison was evaluated without acceptance bounds	Decide whether the reference is appropriate for the intended use
Unevaluable requested comparison	The requested evidence could not be established	Resolve the stated geometry, reference, numerical or resource problem

`--check n` changes the structural failure policy only. It does not disable a requested device comparison or admittance check. When several checks fail, their problems are retained together in the run result.

The examples above were verified with `klayout-pex 0.9.0` and the `KLayout` Python module `0.30.13`. The device example explicitly fixes the halo and fringe scaling. Its numbers document that run, not a calibration target or a promise that later coefficient revisions preserve them.

9 Comparison KPEX/2.5D with MAGIC

The initial goal of KPEX/2.5D is to basically come up with the same results as KPEX/MAGIC.

> **Note**
>
> Notes about this comparison:
>
> - Version `Magic 8.3 rev 486` is used, but augmented with debug logging[6], which will be shown and discussed for each example.
>   - The sidewall rows were later halved under an interpretation that has since been corrected. They are historical results, not a reference for `MAGIC 8.3.693` or `KPEX 0.9.0`, which restore the full-pair convention (Section 5.4)
> - KPEX/MAGIC halo is set to `--magic-halo=100000`
>   - NOTE: the screenshots however were created with `--magic-halo=8` to give better illustrations
> - KPEX/2.5D halo is set to `--halo=100000`
> - The test patterns `overlap_shield_same_net_*` and `plate_20um_x_20um_li1_over_*` were compared with `MAGIC 8.3.681` without debug logging, so their MAGIC capacitances are given per net pair.

9.1 Test Pattern `single_plate_100um_x_100um_li1_over_substrate`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/single_plate_100um_x_100um_li1_over_substrate.gds.gz

Table 53: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Overlap	li1	li1	substrate	VSUBS	369.9	369.9	4

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Fringe (top)	li1	li1	substrate	VSUBS	4.07	4.07	5
Fringe (left)	li1	li1	substrate	VSUBS	4.07	4.07	6
Fringe (right)	li1	li1	substrate	VSUBS	4.07	4.07	7
Fringe (bottom)	li1	li1	substrate	VSUBS	4.07	4.07	8

```
Magic 8.3 revision 486 - Compiled on `date`.
----------------------------------------------------

CapDebug (extNodeAreaFunc/Area) layer li(90), net li_0_0#, area=400000000 (10000 µm^2) nreg_ca
CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer li(90), net li_0_0#, length=20000 (100 µm),
```

[6] A fork of MAGIC which includes debug logging about the different parasitic contributions is hosted here: https://github.com/martinjankoehler/magic

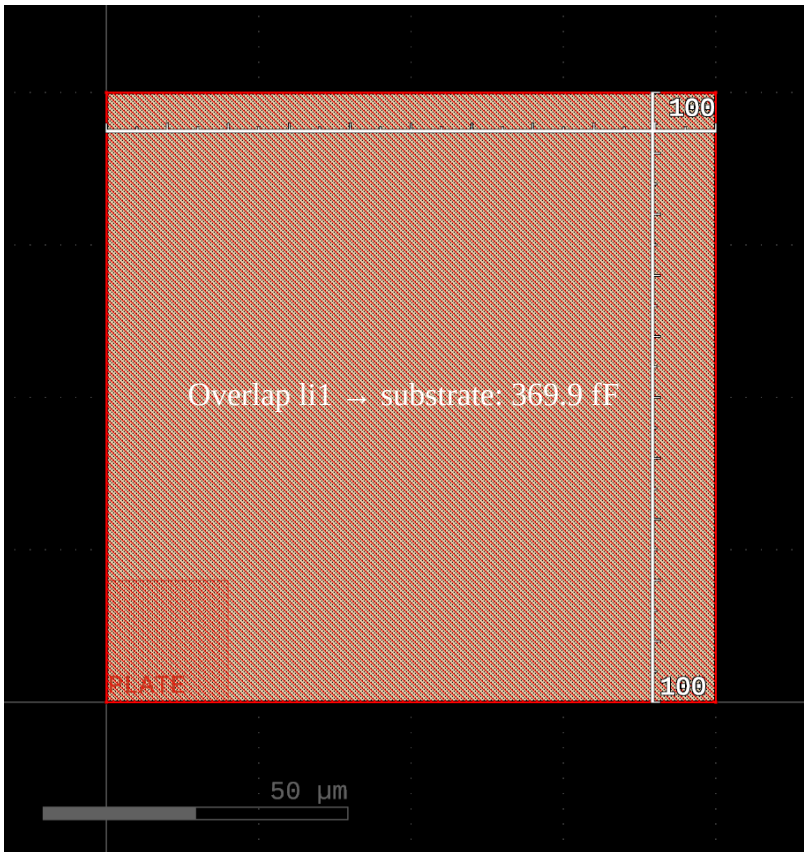


Figure 58: Overlap capacitance li1 to substrate.

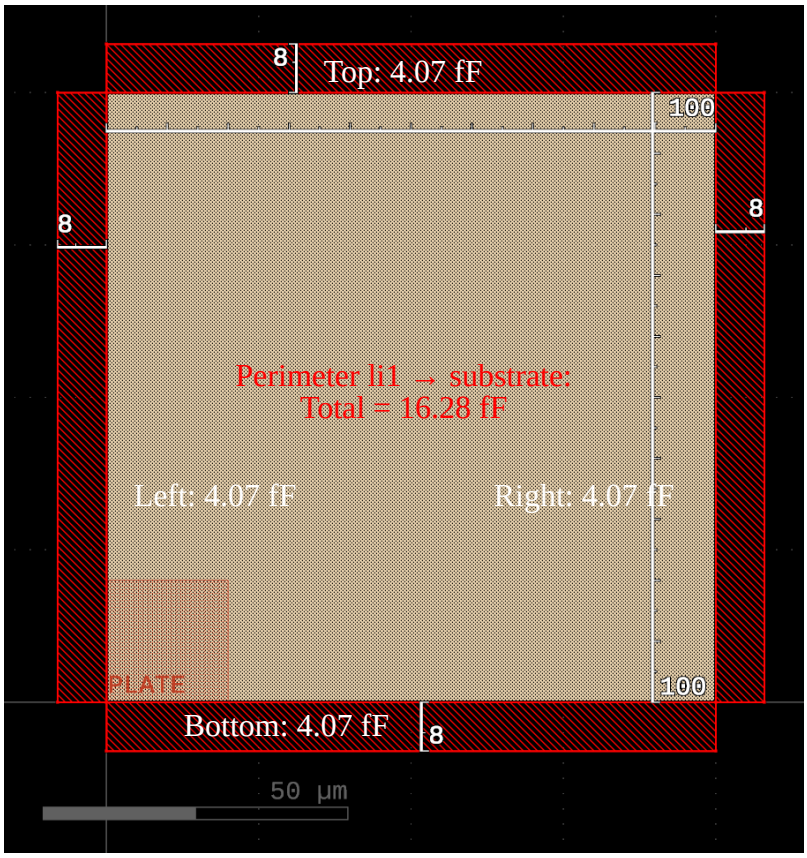


Figure 59: Perimeter (fringe) capacitance li1 to substrate.

```
CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer li(90), net li_0_0#, length=20000 (100 µm)
CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer li(90), net li_0_0#, length=20000 (100 µ
CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer li(90), net li_0_0#, length=20000 (100 µm
CapDebug (extSetResist): li_0_0# area=400000000 (10000 µm^2) perim=80000 (400 µm)
CapDebug ---
```

9.2 Test Pattern `sidewall_20um_length_distance_200nm_li1`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/sidewall_20um_length_distance_200nm_li1.gds.gz

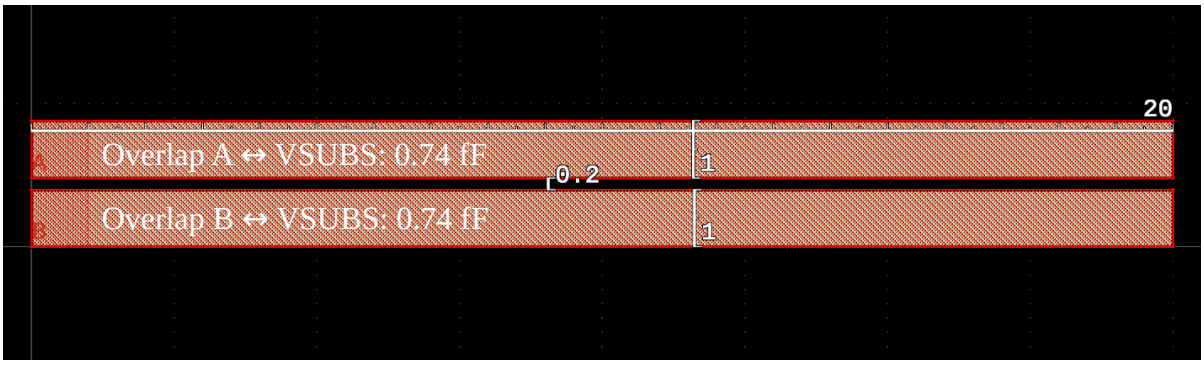


Figure 60: Overlap capacitance of nets on li1.

Table 54: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Overlap	li1	A	substrate	VSUBS	0.7398	0.74	4
Overlap	li1	B	substrate	VSUBS	0.7398	0.74	11

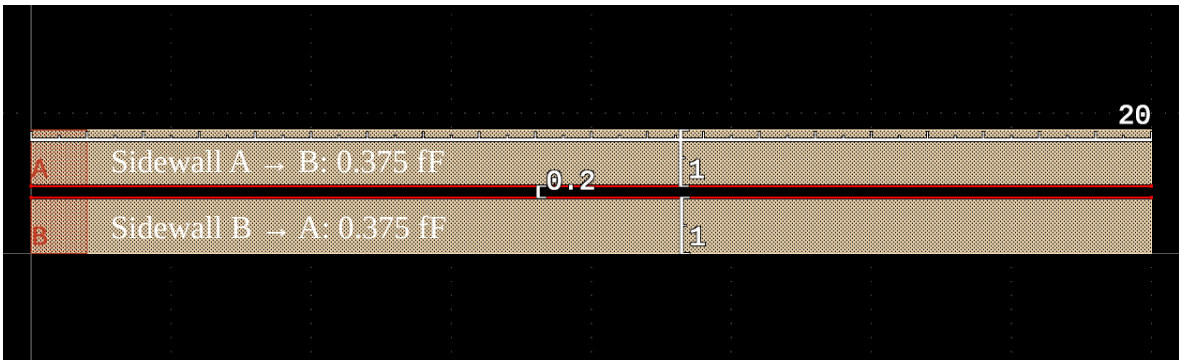


Figure 61: Sidewall capacitance between nets A and B on li1.

Table 55: Extracted Parasitic Sidewall Capacitances

Description	Layer	Net1	Net2	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Sidewall	li1	A	B	0.375	0.375	18
Sidewall	li1	B	A	0.375	0.375	21

The historical table sums to 0.75 fF. Lines 18 and 21 of the recorded `CapDebug` output instead sum to 1.5 fF. The table was manually halved under the superseded interpretation described above. These rows need a new comparison with the corrected tools and matching coefficient tables before they can serve as a current numerical reference.

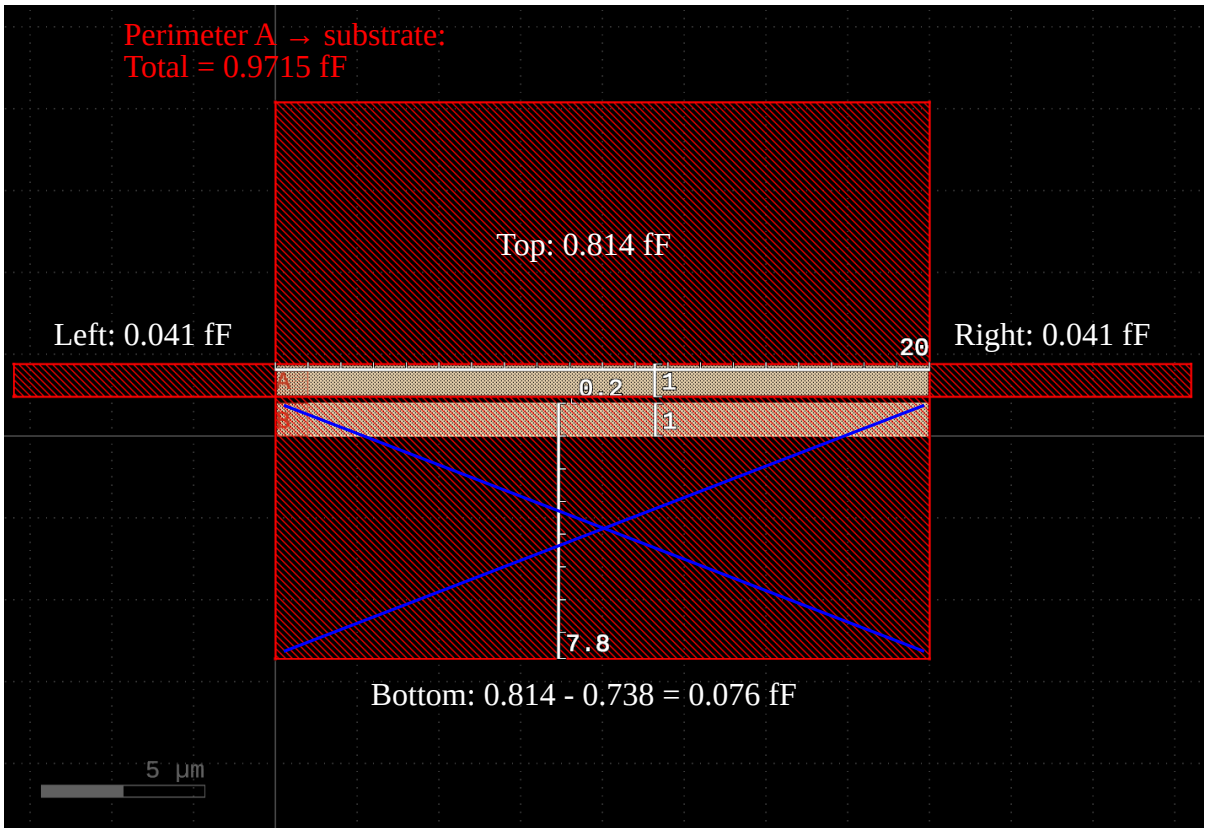


Figure 62: Fringe capacitance between net A and substrate.

Table 56: Extracted Parasitic Fringe Capacitances between nets A, B on li1 and substrate

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Fringe (top)	li1	A	substrate	VSUBS	0.814	0.814	5
Fringe (left)	li1	A	substrate	VSUBS	0.0407	0.041	6
Fringe (bottom)	li1	A	substrate	VSUBS	0.814	0.076	7
Fringe (bottom)	li1	A	substrate	VSUBS	-0.7379	—	19
Fringe (right)	li1	A	substrate	VSUBS	0.0407	0.041	8
Fringe (top)	li1	B	substrate	VSUBS	0.814	0.076	12
Fringe (top)	li1	B	substrate	VSUBS	-0.7379	—	22
Fringe (left)	li1	B	substrate	VSUBS	0.0407	0.041	13
Fringe (bottom)	li1	B	substrate	VSUBS	0.814	0.814	14
Fringe (right)	li1	B	substrate	VSUBS	0.0407	0.041	15

```
1 Magic 8.3 revision 486 - Compiled on `date`.
2 ----------------------------------------------------
3
4 CapDebug (extNodeAreaFunc/Area) layer li(90), net li_0_240#, area=800000 (20 µm^2) nreg_cap +=
```

```
CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer li(90), net li_0_240#, length=4000 (20 µm),
CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer li(90), net li_0_240#, length=200 (1 µm),
CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer li(90), net li_0_240#, length=4000 (20 µ
CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer li(90), net li_0_240#, length=200 (1 µm),
CapDebug (extSetResist): li_0_240# area=800000 (20 µm^2) perim=8400 (42 µm)
CapDebug ---
CapDebug (extNodeAreaFunc/Area) layer li(90), net li_0_0#, area=800000 (20 µm^2) nreg_cap += 0
CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer li(90), net li_0_0#, length=4000 (20 µm), n
CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer li(90), net li_0_0#, length=200 (1 µm), nr
CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer li(90), net li_0_0#, length=4000 (20 µm)
CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer li(90), net li_0_0#, length=200 (1 µm), n
CapDebug (extSetResist): li_0_0# area=800000 (20 µm^2) perim=8400 (42 µm)
CapDebug ---
CapDebug (sidewall): A-B (layer li), overlap=4000 (20 µm), sep=40 (0.2 µm), e->ec_cap=12.75 (0
CapDebug (obsolete_fringe (blocked)): A -= 0.737881 fF …    now A == 1.711319 fF
    overlapMult=0.003699 (3.699 aF/µm^2) dnear=40 (0.2 µm), snear=0.0935129 (0.000467564 µm),
CapDebug (sidewall): A-B (layer li), overlap=4000 (20 µm), sep=40 (0.2 µm), e->ec_cap=12.75 (0
CapDebug (obsolete_fringe (blocked)): B -= 0.737881 fF …    now B == 1.711319 fF
    overlapMult=0.003699 (3.699 aF/µm^2) dnear=40 (0.2 µm), snear=0.0935129 (0.000467564 µm),
exttospice finished.
```

9.3 Test Pattern `sideoverlap_simple_plates_li1_m1`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/sideoverlap_simple_plates_li1_m1.gds.gz

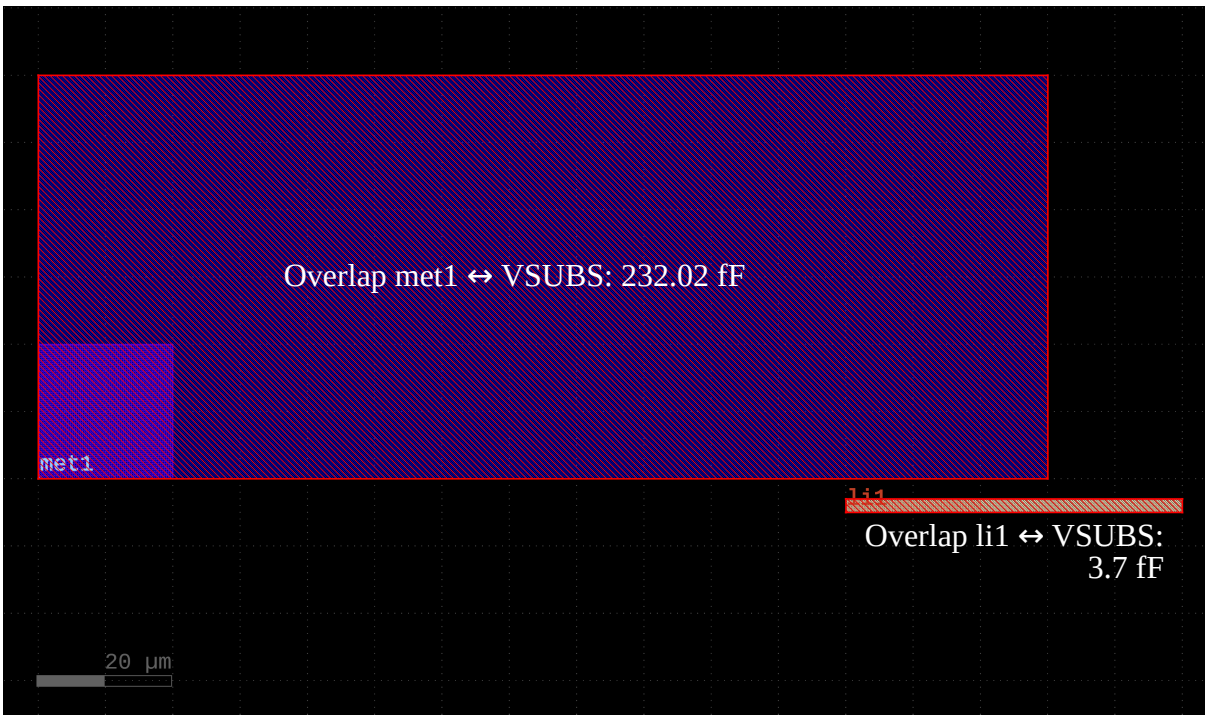


Figure 63: Overlap capacitances to substrate.

Table 57: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Overlap	li1	li1	substrate	VSUBS	3.699	3.7	7
Overlap	met1	met1	substrate	VSUBS	232.02	232.02	15

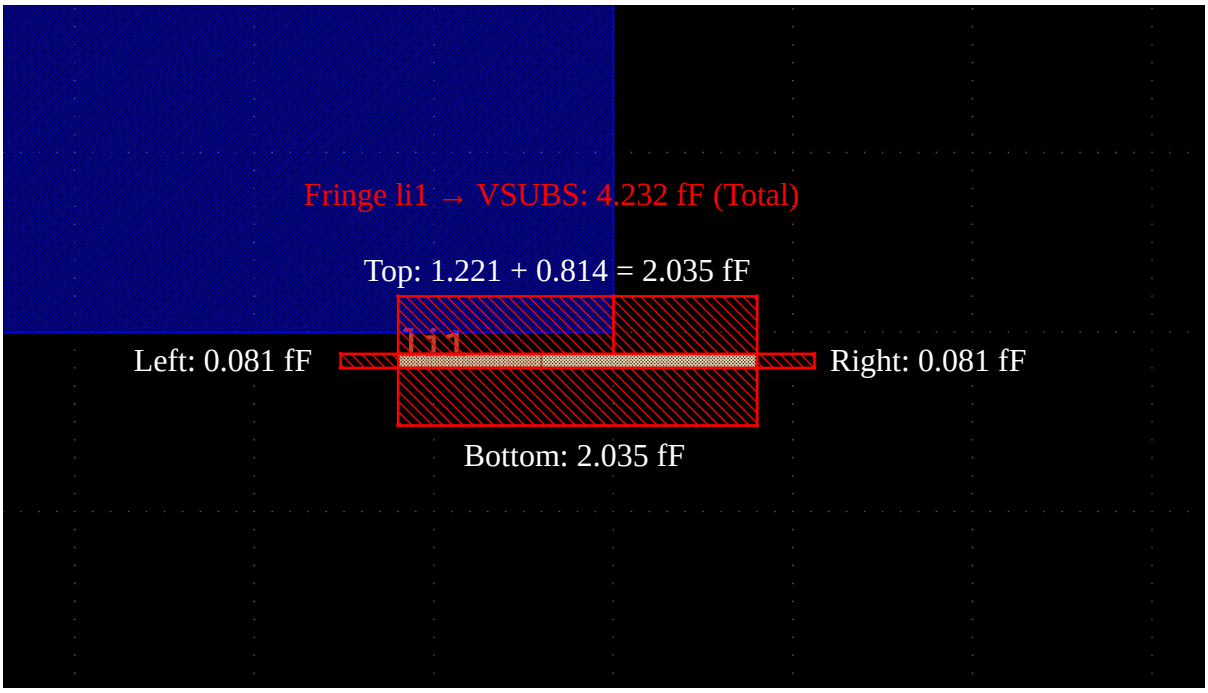


Figure 64: Fringe capacitances li1 to substrate.

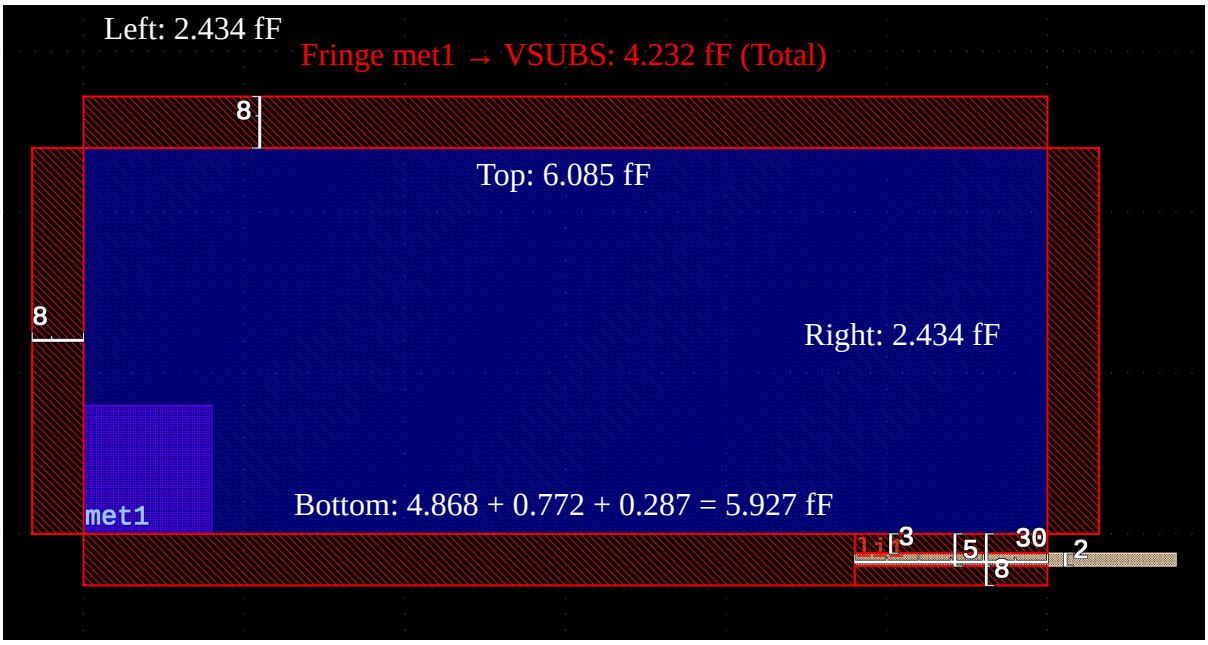


Figure 65: Fringe capacitances met1 to substrate.

Table 58: Extracted Parasitic Fringe Capacitances between metals and substrate

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Fringe (top)	li1	li1	substrate	VSUBS	2.035	2.035	9
Fringe (bottom)	li1	li1	substrate	VSUBS	2.035	2.035	11
Fringe (left)	li1	li1	substrate	VSUBS	0.081	0.081	10
Fringe (right)	li1	li1	substrate	VSUBS	0.081	0.081	12
Fringe (top)	met1	met1	substrate	VSUBS	6.0855	6.085	17
Fringe (bottom)	met1	met1	substrate	VSUBS	6.0855	5.927	19
Fringe (bottom)	met1	met1	substrate	VSUBS	-0.1579	—	19
Fringe (left)	met1	met1	substrate	VSUBS	2.4342	2.434	18
Fringe (right)	met1	met1	substrate	VSUBS	2.4342	2.434	20

Note

`MAGIC` handles shielding differently than `KPEX/2.5D`, it first assumes no shield exists, and later subtracts portions to arrive at the shielded value. That's why the bold cells differ. Line 15 - 20 accumulate contributions, whereas line 34 subtracts a contribution:

- Line 15: `CapDebug (extNodeAreaFunc/Area) layer m1(97), net m1_10000_10000#, area=360000000 (9000 µm^2) nreg_cap += 232.02 fF`
- Line 17: `CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer m1(97), net m1_10000_10000#, length=30000 (150 µm), nreg_cap += 6.0855 fF (now nreg_cap = 238.105 fF)`
- Line 18: `CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer m1(97), net m1_10000_10000#, length=12000 (60 µm), nreg_cap += 2.4342 fF (now nreg_cap = 240.54 fF)`
- Line 19: `CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer m1(97), net m1_10000_10000#, length=30000 (150 µm), nreg_cap += 6.0855 fF (now nreg_cap = 246.625 fF)`
- Line 20: `CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer m1(97), net m1_10000_10000#, length=12000 (60 µm), nreg_cap += 2.4342 fF (now nreg_cap = 249.059 fF)`
- Line 34: `nreg_cap -= 0.157966 fF (now nreg_cap = 248.901 fF)`

Comparison:

- `KPEX/2.5D`: $4.86834 + 0.77242 + 0.2867 = 5.9275$
- `MAGIC`: $6.0855 - 0.157966 = 5.9275$

Table 59: Extracted Parasitic Fringe Capacitances between metals

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Fringe (top)	li1	li1	met1	met1	0.598	0.06	26
Fringe (bottom)	met1	met1	li1	li1	0.0654	0.065	36

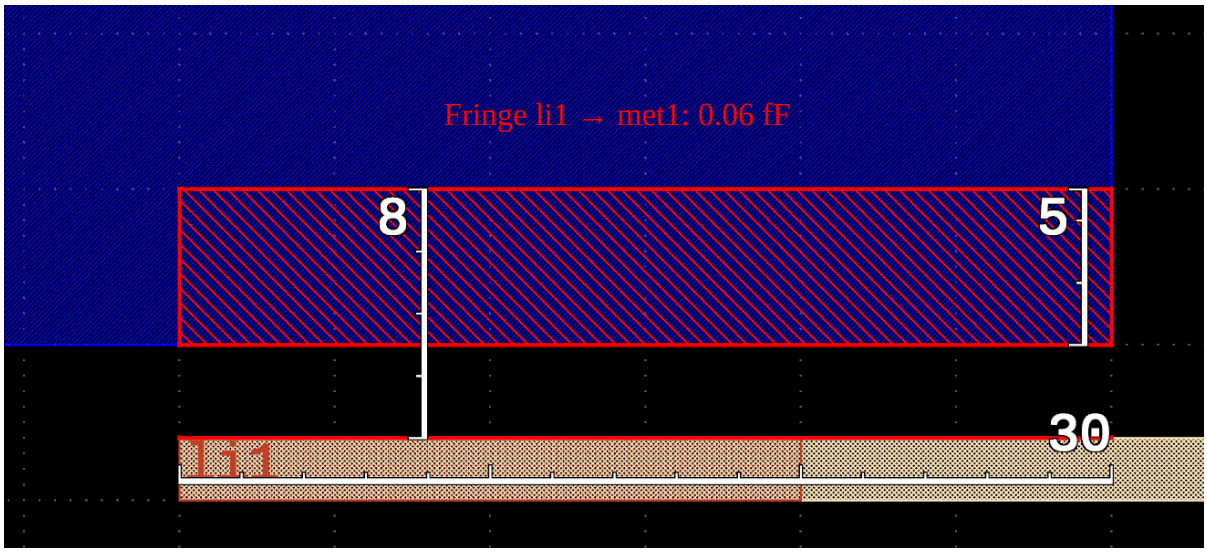


Figure 66: Fringe capacitances li1 to m1.

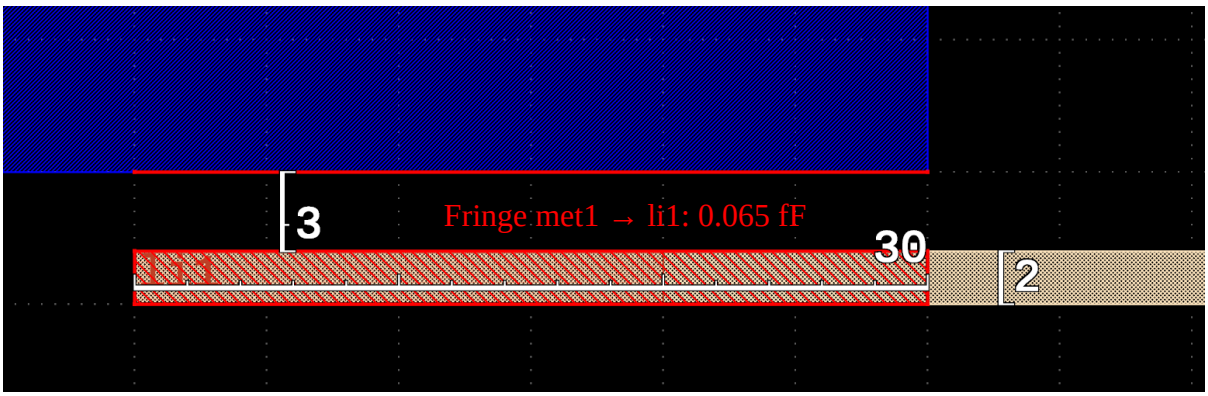


Figure 67: Fringe capacitances m1 to li1.

```
Magic 8.3 revision 486 - Compiled on `date`.

----------------------------------------------------

Extracting sideoverlap_simple_plates_li1_m1 into /Users/martin/Source/klayout-pex/output_sky13

CapDebug (extNodeAreaFunc/Area) layer li(90), net li_34000_9000, area=4000000 (100 µm^2) nreg_

CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer li(90), net li_34000_9000, length=10000 (50
CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer li(90), net li_34000_9000, length=400 (2 µ
CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer li(90), net li_34000_9000, length=10000
CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer li(90), net li_34000_9000, length=400 (2
CapDebug (extSetResist): li_34000_9000 area=4000000 (100 µm^2) perim=20800 (104 µm)
CapDebug ---
CapDebug (extNodeAreaFunc/Area) layer m1(97), net m1_10000_10000#, area=360000000 (9000 µm^2)

CapDebug (extNodeAreaFunc/Perimeter/TopSide) layer m1(97), net m1_10000_10000#, length=30000 (
CapDebug (extNodeAreaFunc/Perimeter/LeftSide) layer m1(97), net m1_10000_10000#, length=12000
CapDebug (extNodeAreaFunc/Perimeter/BottomSide) layer m1(97), net m1_10000_10000#, length=3000
CapDebug (extNodeAreaFunc/Perimeter/RightSide) layer m1(97), net m1_10000_10000#, length=12000
CapDebug (extSetResist): m1_10000_10000# area=360000000 (9000 µm^2) perim=84000 (420 µm)
CapDebug ---
CapDebug (extSideOverlapHalo): (li-m1) length=30.000000 µm, mult=0.011420, dnear=3.000000 µm (
CapDebug (extSideOverlapHalo) (li-substrate): mult=0.003699, snear=0.730477, sfar=0.893413 (sf
CapDebug (extSideOverlapHalo): efflength=1.72351 µm, cap+=0.299029 aF, cap=59.8059 aF, e->ec_c
CapDebug (sideoverlaphalo): met1-li1 += 0.059806 fF …   now met1-li1 == 0.059806 fF
CapDebug ---
CapDebug (extSideOverlapHalo): (m1-li) length=30.000000 µm, mult=0.011420, dnear=3.000000 µm (
CapDebug (extSideOverlapHalo) (m1-substrate): mult=0.002578, snear=0.634619, sfar=0.764408 (sf
CapDebug (extSideOverlapHalo): efflength=1.0996 µm, cap+=0.327131 aF, cap=65.4263 aF, e->ec_ca
CapDebug (extSideOverlapHalo/obsolete_perimcap) layer m1(97), net met1,
    efflength=3.89366 µm (778.733) = (sfrac(0.129789) - subfrac(0)) * length(30 µm)
    exts_perimCap[m1][0] = 0.20285
    nreg_cap -= 0.157966 fF (now nreg_cap = 248.901 fF)
CapDebug (obsolete_perimcap): met1 -= 0.157966 fF … now met1 == 248.901423 fF
CapDebug (sideoverlaphalo): met1-li1 += 0.065426 fF …   now met1-li1 == 0.125232 fF
```

9.4 Test Pattern `overlap_shield_same_net_100um_x_100um_li1_m1_m2`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/overlap_shield_same_net_100um_x_100um_li1_m1_m2.gds.gz

Plates of $100\,\mu m \times 100\,\mu m$: `li1` and `met1` of net `A`, joined by an `mcon` in a corner, under `met2` of net `B`. `met1` shields `li1` from `met2`, although both are of net `A`.

Table 60: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Overlap	met2	B	met1	A	1338.6
Overlap	met2	B	li1	A	—
Overlap	li1	A	substrate	VSUBS	369.9

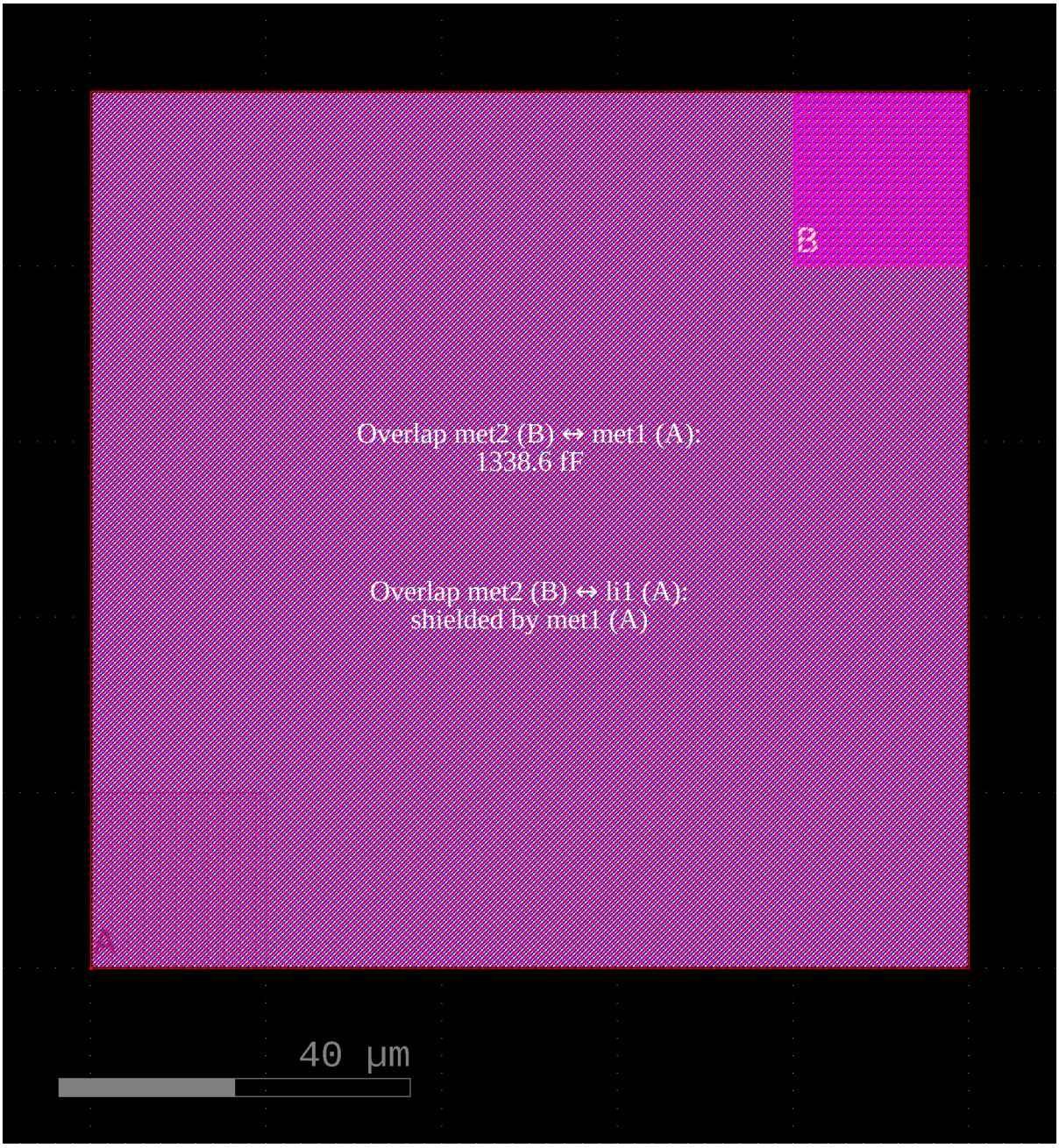


Figure 68: Overlap capacitances between the nets A and B, met1 (A) covers li1 (A).

Table 61: Extracted Parasitic Fringe Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Fringe (all sides)	li1	A	substrate	VSUBS	16.28
Fringe (all sides)	met1	A	substrate	VSUBS	16.228
Fringe (all sides)	met2	B	substrate	VSUBS	15.104

Table 62: Extracted Parasitic Capacitances per Net Pair

Net 1	Net 2	MAGIC [fF]	KPEX/2.5D [fF]
A	B	1338.6	1338.6
A	VSUBS	402.41	402.408
B	VSUBS	15.104	15.104

> **Note**
>
> Up to #299, only shapes of other nets shielded in `KPEX/2.5D`, so `A` ↔ `B` also got the overlap of `met2` and `li1` below `met1`: $1338.6 + 375.6 = 1714.2\,fF$.

9.5 Test Pattern `overlap_shield_same_net_half_100um_x_100um_li1_m1_m2`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/overlap_shield_same_net_half_100um_x_100um_li1_m1_m2.gds.gz

The same plates, but `met1` covers only the left half of `li1`: `met2` overlaps `met1` on the left half and `li1` on the right half.

Table 63: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Overlap	met2	B	met1	A	669.3
Overlap	met2	B	li1	A	187.8
Overlap	li1	A	substrate	VSUBS	369.9

Table 64: Extracted Parasitic Fringe Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Fringe (all sides)	li1	A	substrate	VSUBS	16.28
Fringe (all sides)	met1	A	substrate	VSUBS	8.214
Fringe (right)	met1	A	met2	B	4.796
Fringe (all sides)	met2	B	substrate	VSUBS	15.104

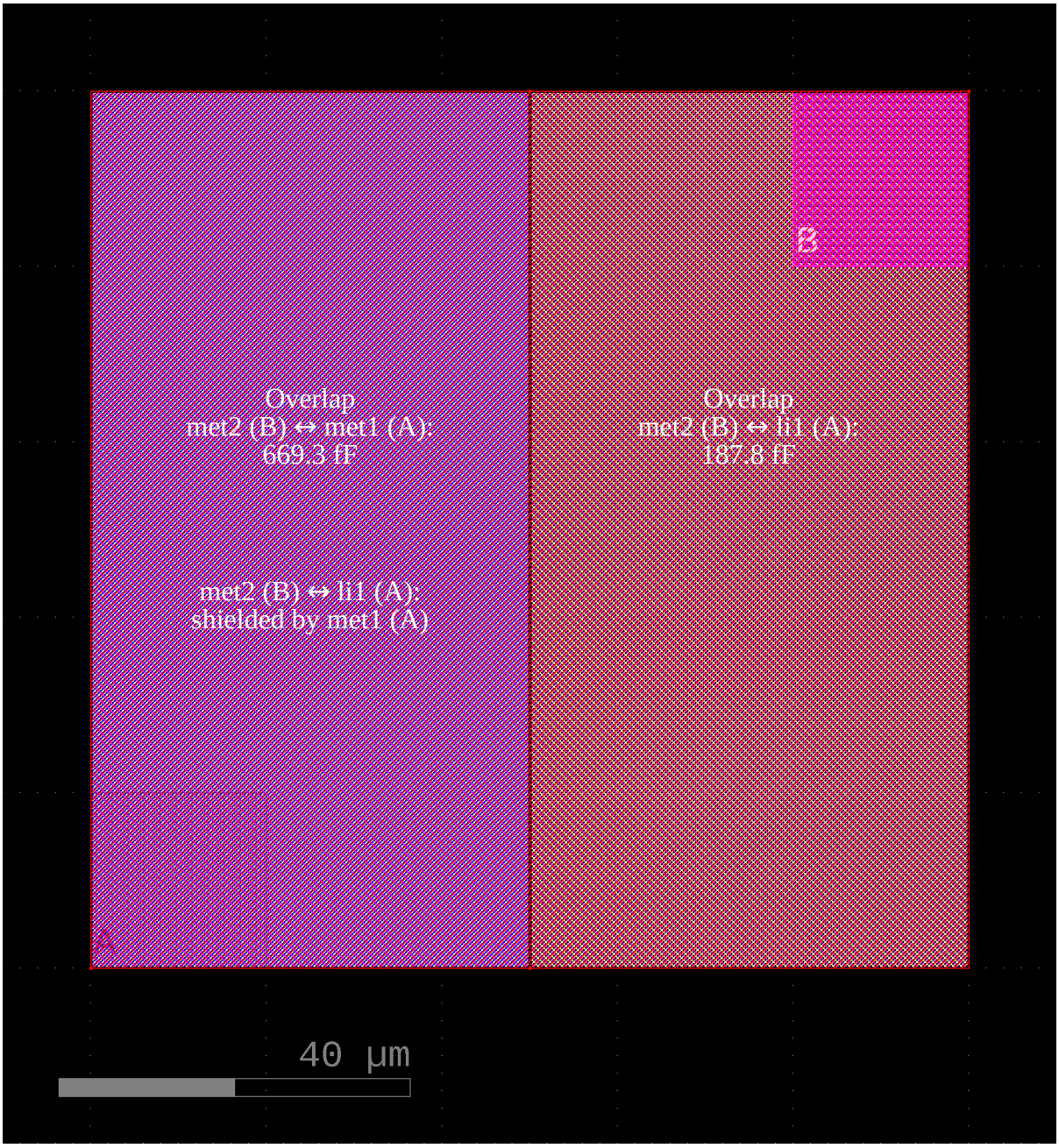


Figure 69: Overlap capacitances between the nets A and B, met1 (A) covers the left half of li1 (A).

Table 65: Extracted Parasitic Capacitances per Net Pair

Net 1	Net 2	MAGIC [fF]	KPEX/2.5D [fF]
A	B	861.9	861.896
A	VSUBS	394.39	394.394
B	VSUBS	15.104	15.104

> **Note**
>
> Up to #299, `KPEX/2.5D` also counted the overlap of `met2` and `li1` below `met1` (the left half), so `A` ↔ `B` was $861.896 + 187.8 = 1049.696\,fF$.

9.6 Test Pattern `plate_20um_x_20um_li1_over_ntap`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/plate_20um_x_20um_li1_over_ntap.gds.gz

A `li1` plate of $20\,\mu m \times 20\,\mu m$ of net `A` over the middle of an n-tap of $40\,\mu m \times 40\,\mu m$ in an nwell of net `W`. The tap is active area like the source/drain, so `li1` has the capacitance of the diffusion to it.

Table 66: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Overlap	li1	A	tap (ntap)	W	22.12

Table 67: Extracted Parasitic Fringe Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Fringe (all sides)	li1	A	tap (ntap)	W	3.338
Fringe (all sides)	li1	A	nwell	W	0.046
Fringe (all sides)	li1	A	substrate	VSUBS	0.233

Table 68: Extracted Parasitic Capacitances per Net Pair

Net 1	Net 2	MAGIC [fF]	KPEX/2.5D [fF]
A	W	25.5042	25.504
A	VSUBS	0.2326	0.232
W	VSUBS	232.32	—

> **Note**
>
> - `KPEX/2.5D` has no capacitance between the nwell and the substrate, the PDK models that junction as a diode.
> - Up to #301, `KPEX/2.5D` didn't know the taps in the capacitance extraction, so `li1` had the capacitance to the nwell below the tap: `A` ↔ `W` $17.82\,fF$.

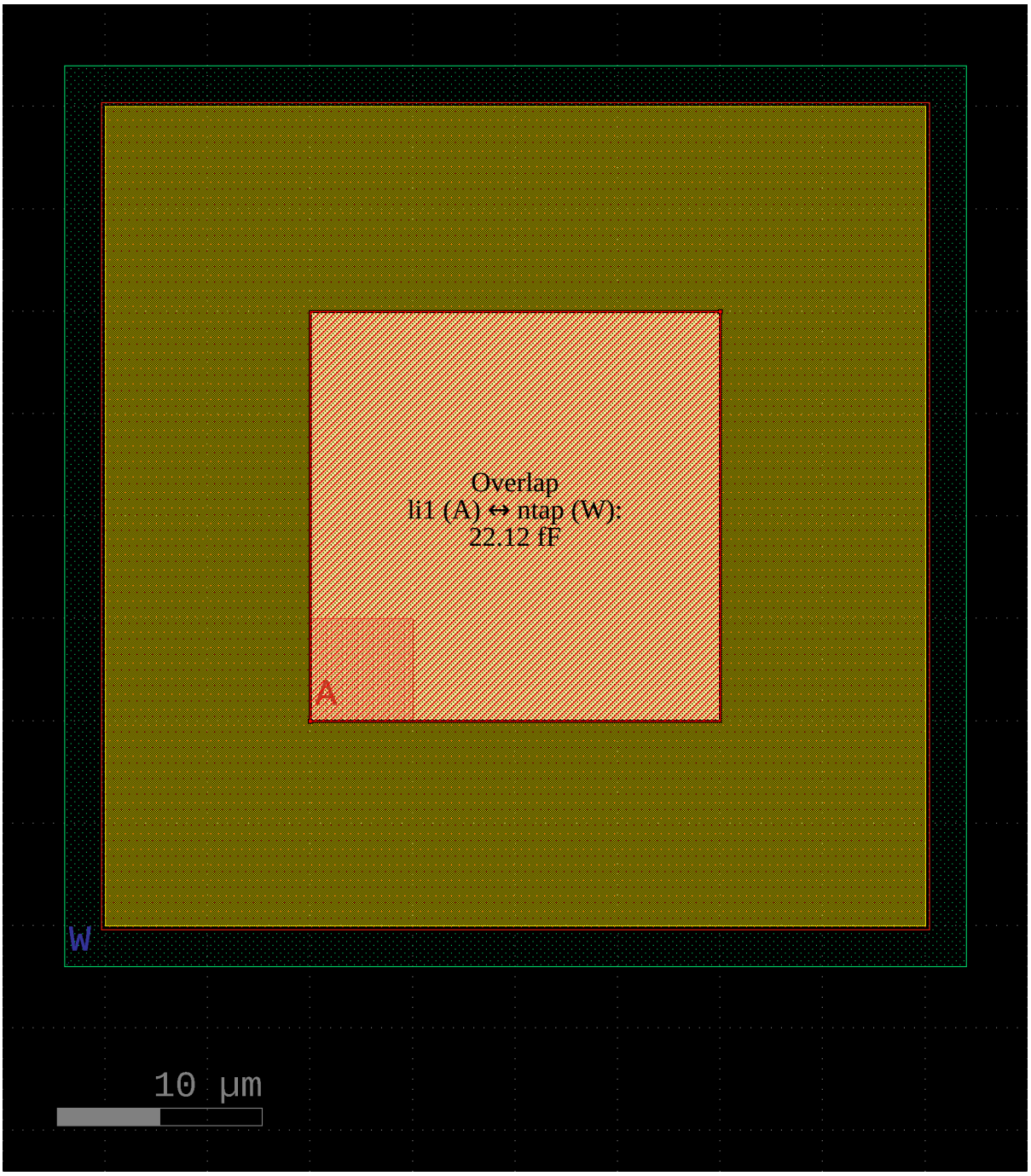


Figure 70: Overlap capacitance between li1 (A) and the n-tap of the nwell (W).

9.7 Test Pattern `plate_20um_x_20um_li1_over_ptap`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/plate_20um_x_20um_li1_over_ptap.gds.gz

The same plate over a p-tap, which a `licon` in its corner ties to the `li1` pad `VSS` (like in the standard cells, as in the sky130A LVS deck only the `licon` joins a p-tap to a net).

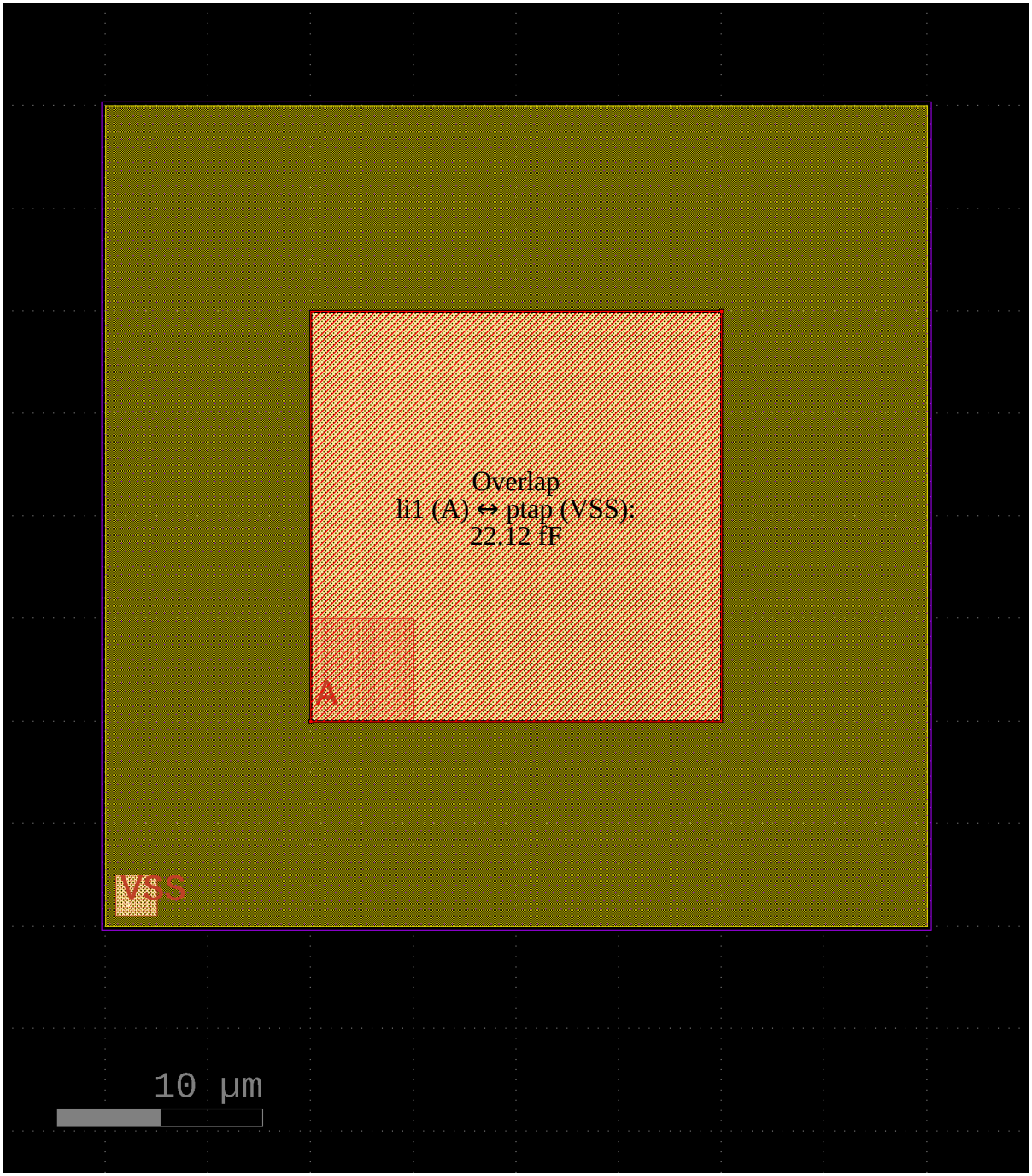


Figure 71: Overlap capacitance between li1 (A) and the p-tap (VSS).

Table 69: Extracted Parasitic Overlap Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Overlap	li1	A	tap (ptap)	VSS	22.12

Table 70: Extracted Parasitic Fringe Capacitances

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	KPEX/2.5D [fF]
Fringe (all sides)	li1	A	tap (ptap)	VSS	3.338
Fringe (all sides)	li1	A	substrate	VSUBS	0.279
Fringe (all sides)	li1	VSS	substrate	VSUBS	0.130

Table 71: Extracted Parasitic Capacitances per Net Pair

Net 1	Net 2	MAGIC [fF]	KPEX/2.5D [fF]
A	VSS	18.3132	25.458
A	VSUBS	—	0.278
VSS	VSUBS	—	0.13

Note

`MAGIC` and `KPEX/2.5D` differ here on purpose:

- `MAGIC` makes the p-tap part of its substrate node (`substrate *ppdiff,…` of the sky130A tech), whose area gets the field's 36.99 $aF/\mu m^2$ (14.80 fF), but its fringe the diffusion's: 18.31 fF to the one node `VSS`.
- `KPEX/2.5D` has the p-tap as diffusion, the same stack as the source/drain, like the n-tap: 55.3 $aF/\mu m^2$ (22.12 fF).
- In the sky130A LVS deck, the p-tap (`VSS`) and the substrate (`VSUBS`) are different nets, `MAGIC` joins them.
- Up to #301, `KPEX/2.5D` had `li1` over the field: `A` ↔ `VSUBS` 18.05 fF.

9.8 Test Pattern `r_single_wire_li1`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_single_wire_li1.gds.gz

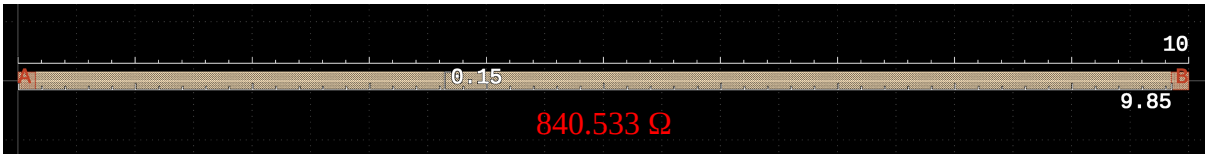


Figure 72: Resistance of a single wire on li1.

- Wire length $L = 9.85\,\mu m$
- Wire height $H = 0.15\,\mu m$
- Layer is `li1`
- Parasitic Table Coefficient $R_{coeff}(li1) = 12800\,m\Omega$

If we apply the formula illustrated in Section 5.8.1, we get

$$R_{wire} = \frac{L}{H} * R_{coeff}$$

$$R_{wire} = \frac{9.85\,\mu m}{0.15\,\mu m} * 12800\,m\Omega = 84053\dot{}\,m\Omega = 840.53\dot{}\,\Omega$$

```
Magic 8.3 revision 486 - Compiled on `date`.
----------------------------------------------------
...
Warning:  Ports "A" and "B" are electrically shorted.

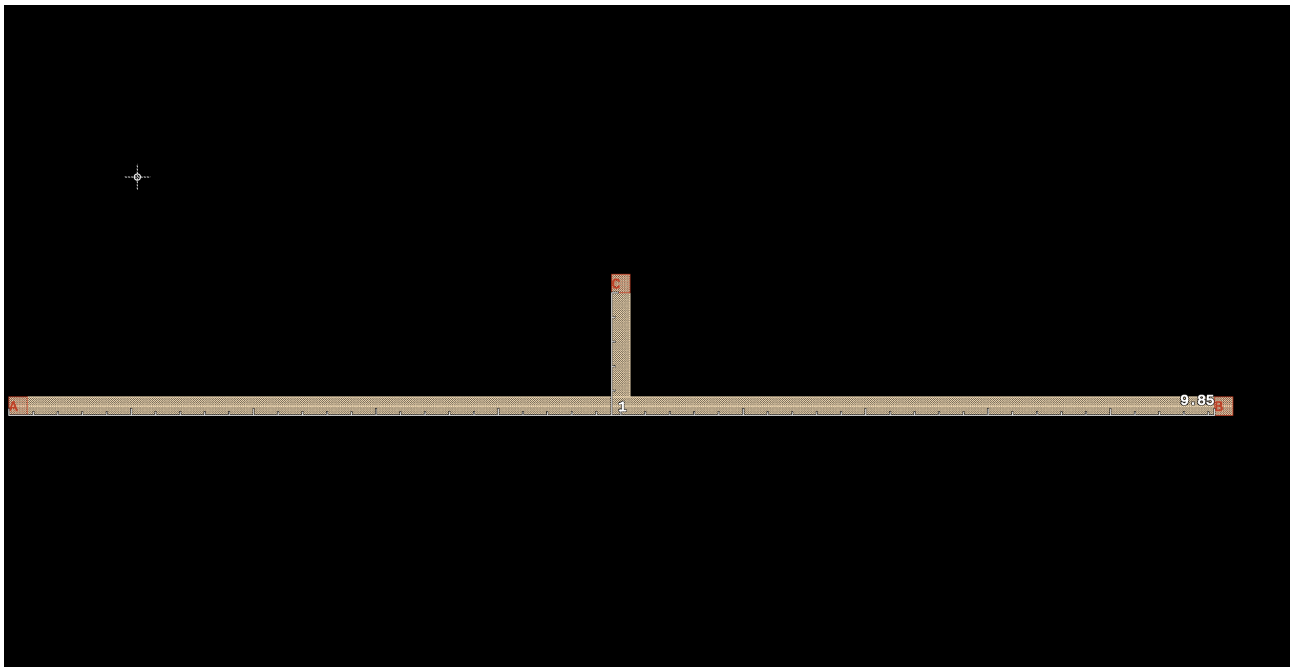
Location is (1970, -15); drivepoint (1970, -15)
Location is (0, -15); drivepoint (0, -15)

ResCalcEastWest: A (0, -0.075) <-> B (9.85, -0.075) @ li
    exts_sheetResist[li]=12800, length=9.85, height=0.15, resistor->rr_value = 840533 mΩ
ResCalcEastWest: A (0, -0.075) <-> B (9.85, -0.075) @ li
    exts_sheetResist[li]=12800, length=9.85, height=0.15, resistor->rr_value = 840533 mΩ
Total Nets: 2
```

> **Note**
>
> MAGIC recognizes that ports A and B are shorted. The purpose of these test patterns is to look at minimized problems in isolation, like the resistance of the wire here. They are not realistic examples, so it appears to be a bug in that corner case, that the resistance is reported twice.

9.9 Test Pattern r_wire_voltage_divider_li1

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_wire_voltage_divider_li1.gds.gz



With MAGIC, we get (again, R3-R8 are redundant):

```
.subckt r_wire_voltage_divider_li1 B A C
R0 A C.n0 426.668
R1 C.n0 B 413.868
R2 C.n0 C 72.5338
.ends
```

Table 72: Extracted Parasitic Resistances between pins

Description	Layer	Node1	Node2	MAGIC [Ω]	KPEX/2.5D [Ω]
Wire	li1	A	internal N1	426.668	426.667
Wire	li1	B	internal N1	413.868	413.867
Wire	li1	C	internal N1	72.5338	72.533

9.10 Test Pattern `r_contact_1x1_minsize_mcon`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_contact_1x1_minsize_mcon.gds.gz

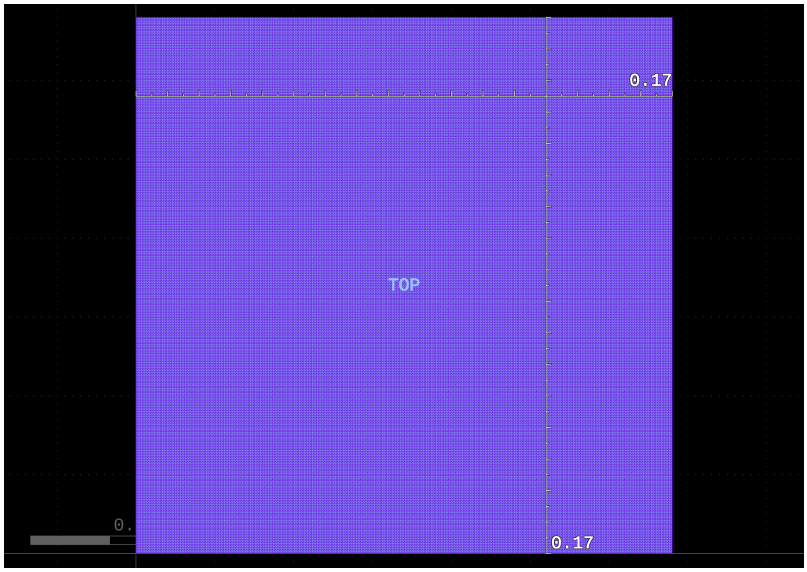


Figure 73: Resistance of a minimum sized via on mcon.

- Layers:
  - Top Layer: `met1`
  - Via: `mcon`
  - Bottom Layer: `li1`
- Design rules:
  - Via width: $viawidth = 0.17\,\mu m$
  - Via spacing: $spacing = 0.19\,\mu m$
  - Via border: $border = 0.0\,\mu m$
- Parasitic Table Coefficient $R_{coeff}(mcon) = 9300\,m\Omega$
- Drawn via:
  - $w = 0.17\,\mu m$
  - $h = 0.17\,\mu m$

If we apply the formula illustrated in Section 5.8.2, we get

$$n_x = 1 + \left\lfloor \frac{w - (viawidth + 2 * border)}{viawidth + spacing} \right\rfloor$$

$$n_x = 1 + \left\lfloor \frac{0.17\,\mu m - (0.17 + 2 * 0.0)\,\mu m}{0.17\,\mu m + 0.19\,\mu m} \right\rfloor = 1$$

$$n_y = 1 + \left\lfloor \frac{h - (viawidth + 2 * border)}{viawidth + spacing} \right\rfloor$$

$$n_y = 1 + \left\lfloor \frac{0.17\,\mu m - (0.17 + 2 * 0.0)\,\mu m}{0.17\,\mu m + 0.19\,\mu m} \right\rfloor = 1$$

$$R_{via} = \frac{R_{coeff}}{n_x * n_y} \quad \left[\frac{m\Omega}{\text{via count}}\right]$$

$$R_{via} = \frac{9300\,m\Omega}{1 * 1} = 9300\,m\Omega = 9.3\,\Omega$$

Table 73: Extracted Parasitic Resistances

Device	Description	Layer	Node1	Node2	MAGIC [Ω]	KPEX/2.5D [Ω]
R2	Wire	mcon	internal `$0.16`	pin `BOT`		0.0
R1	Via	li1	internal `$0.16`	internal `$1.23`	9.3	9.3
R3	Wire	met1	internal `$1.23`	pin `TOP`		0.0

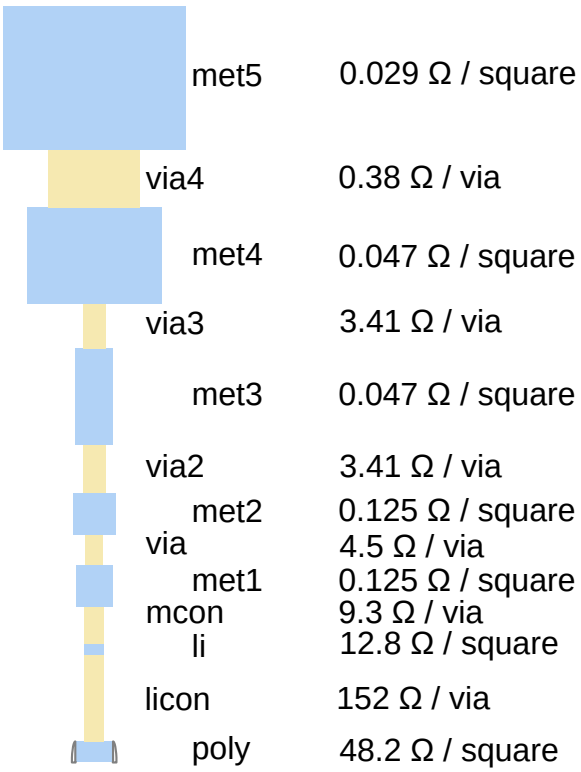
MAGIC extraction results are shown below.

```
Magic 8.3 revision 486 - Compiled on `date`.
------------------------------------------------------
...
Warning:  Ports "TOP" and "BOT" are electrically shorted.
Location is (0, 0); drivepoint (0, 0)
Location is (0, 0); drivepoint (0, 0)

ResDoContacts: (null) (0.085, 0.085) <-> (null) (0.085, 0.085) @ v0
    W = 0.17 µm, H = 0.17 µm
    exts_viaResist[v0]=9300, viawidth=0.17 µm, spacing=0.19 µm, border=0 µm
    squaresx=1, squaresy=1, resistor->rr_value = 9300 mΩ
```

9.11 Test Pattern `r_via_stack_1x1_minsize_poly_to_met5`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_via_stack_1x1_minsize_poly_to_met5.gds.gz



On each metal layer, a pin is placed directly in middle of the via.

Table 74: Extracted Parasitic Resistances between pins

Device	Description	Layer	Node1	Node2	MAGIC [Ω]	KPEX/2.5D [Ω]
R2	Wire	poly	internal `$0.17`	pin `poly`		0.0
R1	Via	licon	internal `$0.17`	internal `$1.18`		152.0
R4	Wire	li1	internal `$1.18`	pin `li1`		0.0
R3	Wire	li1	internal `$1.18`	internal `$2.18`		0.0
R8	Via	mcon	internal `$2.18`	internal `$3.25`		9.3
R10	Wire	met1	internal `$3.25`	pin `met1`		0.0
R9	Wire	met1	internal `$3.25`	internal `$4.25`		0.0
R11	Via	via	internal `$4.25`	internal `$5.26`		4.5
R13	Wire	met2	internal `$5.26`	pin `met2`		0.0
R12	Wire	met2	internal `$5.26`	internal `$6.26`		0.0
R14	Via	via2	internal `$6.26`	internal `$7.43`		3.41
R16	Wire	met3	internal `$7.43`	pin `met3`		0.0
R15	Wire	met3	internal `$7.43`	internal `$8.43`		0.0
R17	Via	via3	internal `$8.43`	internal `$9.40`		3.41
R18	Wire	met4	internal `$9.40`	pin `met4`		0.0
R6	Wire	met4	internal `$10.40`	internal `$9.40`		0.0
R5	Via	via4	internal `$10.40`	internal `$11.27`		0.38
R7	Wire	met5	internal `$11.27`	pin `met5`		0.0

KPEX/2.5D extraction results are shown below.

```
Device;Net1;Net2;Capacitance [fF];Resistance [Ω]
R2;$0.17;poly;;0.0
R1;$0.17;$1.18;;152.0
R4;$1.18;li1;;0.0
R3;$1.18;$2.18;;0.0
```

```
R8;$2.18;$3.25;;9.3
R10;$3.25;met1;;0.0
R9;$3.25;$4.25;;0.0
R11;$4.25;$5.26;;4.5
R13;$5.26;met2;;0.0
R12;$5.26;$6.26;;0.0
R14;$6.26;$7.43;;3.41
R16;$7.43;met3;;0.0
R15;$7.43;$8.43;;0.0
R17;$8.43;$9.40;;3.41
R18;$9.40;met4;;0.0
R6;$10.40;$9.40;;0.0
R5;$10.40;$11.27;;0.38
R7;$11.27;met5;;0.0
```

MAGIC extraction results are shown below.

```
.subckt r_via_stack_1x1_minsize_poly_to_met5 poly li1 met1 met2 met3 met4 met5
R0 li1.n3 poly 175.37
R1 li1.n4 li1.n3 9.3005
R2 li1.n3 li1 6.4005
R3 li1.n4 li1.n2 4.5005
R4 li1.n1 li1.n0 3.4105
R5 li1.n2 li1.n1 3.4105
R6 li1.n0 met5 0.3834
R7 met1 li1.n4 0.063
R8 li1.n2 met2 0.0545541
R9 li1.n1 met3 0.0232879
R10 li1.n0 met4 0.00687288
; removed the redundant devices (5.3.540)
.ends
```

9.12 Test Pattern `nfet_li1_redux`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/nfet_li1_redux.gds.gz

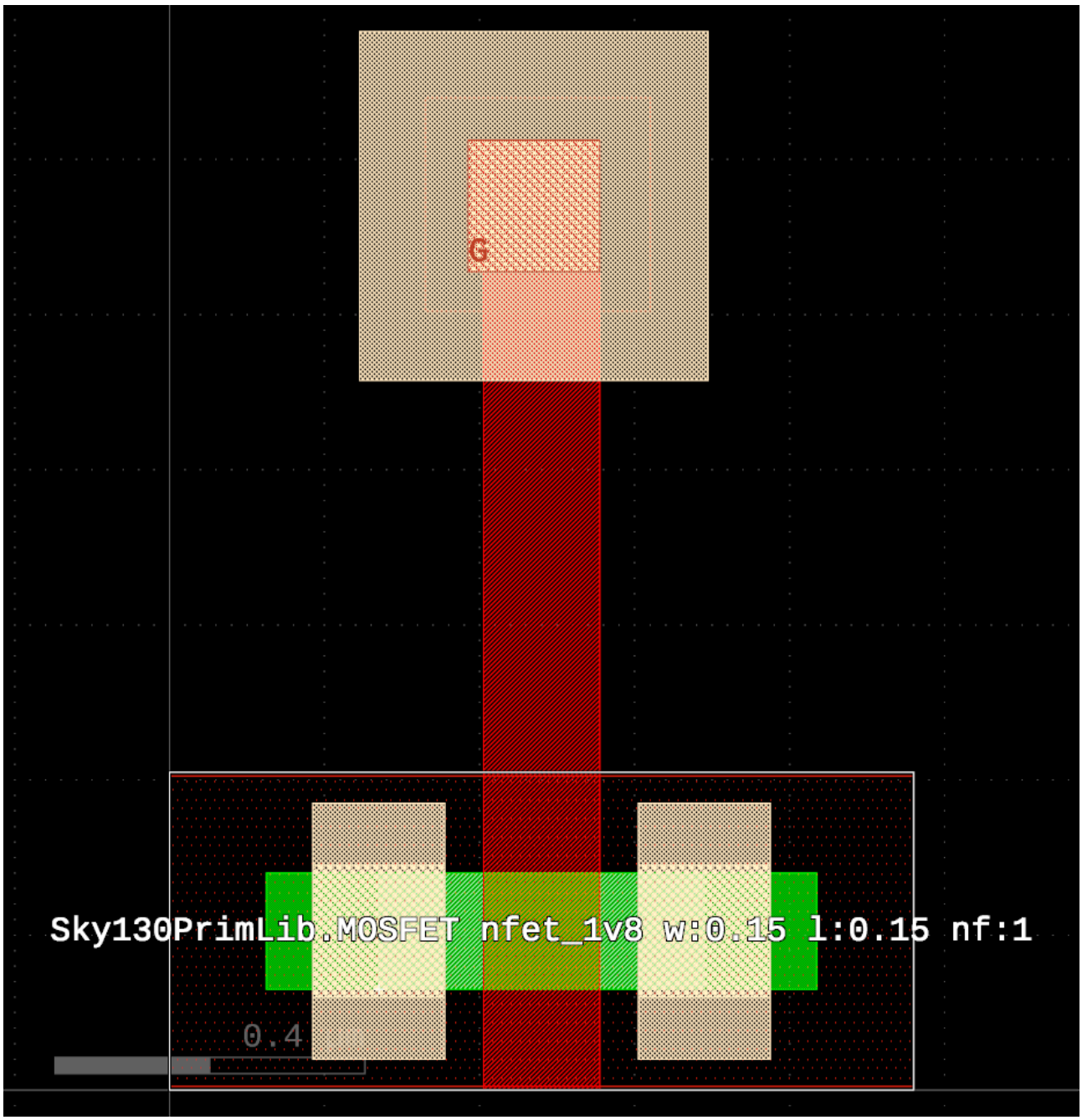


Table 75: Extracted Parasitic Resistances

Device	Description	Layer	Node1	Node2	KPEX/2.5D [Ω]
R1	Wire	li1	internal `G.$1.17`	pin `G`	2.418
R2	Via	licon	internal `G.$0.16` (poly)	internal `G.$1.17` (li1)	152.0
R3	Wire	poly	`G.P0.16` (`G` device terminal)	internal `G.$0.16`	316.321

KPEX/2.5D extraction results are shown below.

```
Device;Net1;Net2;Capacitance [fF];Resistance [Ω]
R1;G;G.$1.17;;2.418
R2;G.$0.16;G.$1.17;;152.0
R3;G.$0.16;G.P0.16;;316.321
```

MAGIC extraction results are shown below.

```
.subckt nfet_li1 VNB G S D
X0 a_111_26# G a_25_26# VSUBS sky130_fd_pr__nfet_01v8 ad=0.042 pd=0.86 as=0.042 ps=0.86 w=0.15
R0 G G.n0 451.651
.ends
```

> **Note**
>
> Notes about this comparison:
>
> - We can see that MAGIC did not extract the interconnection resistances
> - Only exception is `G` ↔ `G.n0`, but node `G.n0` is dangling and not connected

9.13 Test Pattern `nfet_li1`

GDS: https://github.com/iic-jku/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/nfet_li1.gds.gz

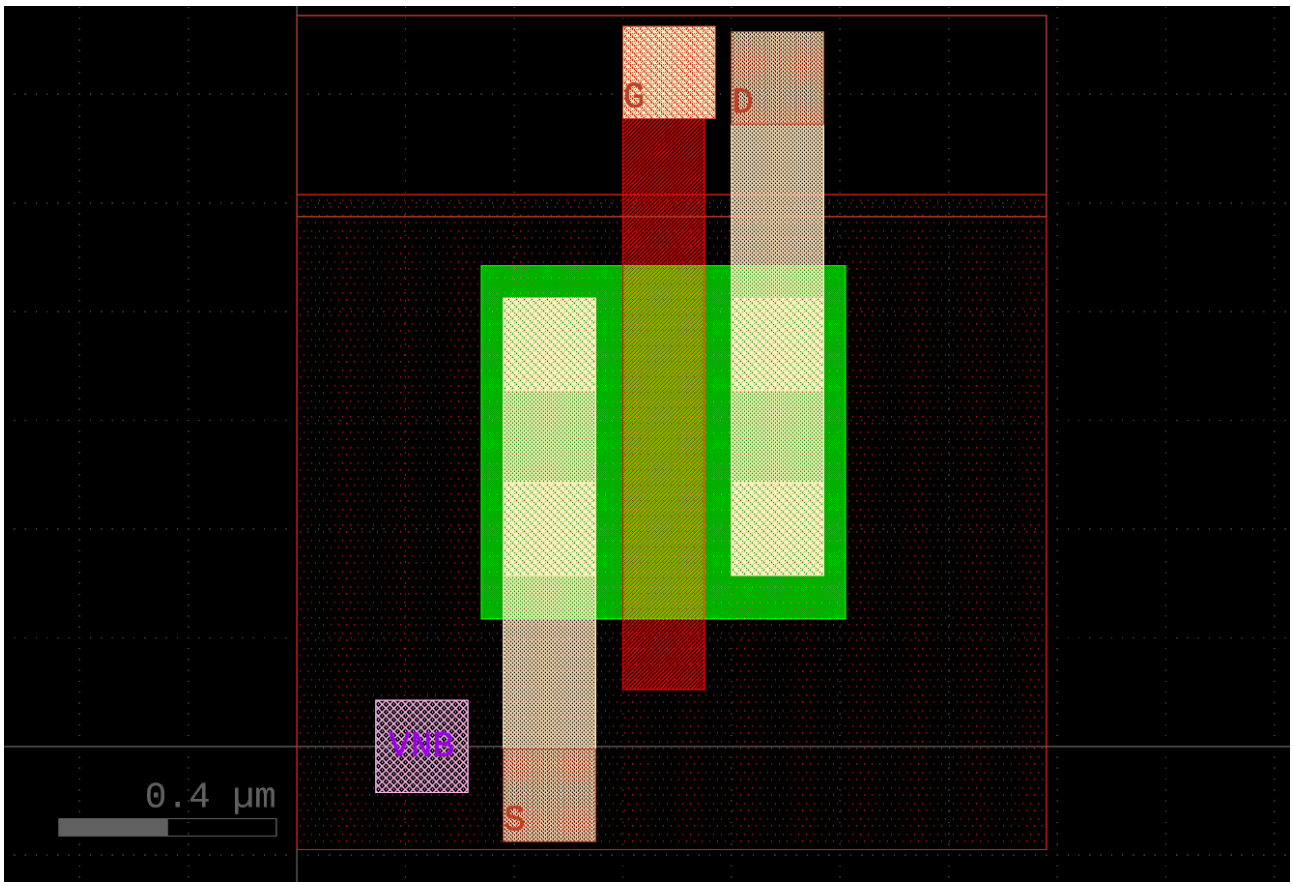


Figure 74: nfet example (2D layout view)

Figure 75 gives an 3D overview over the 3 nets.

KPEX/2.5D extraction results are shown below.

[7] The 3D model of this figure was generated by `KPEX/FasterCap` and saved as STL files. The screenshot was created by opening the STL files in MeshLab.

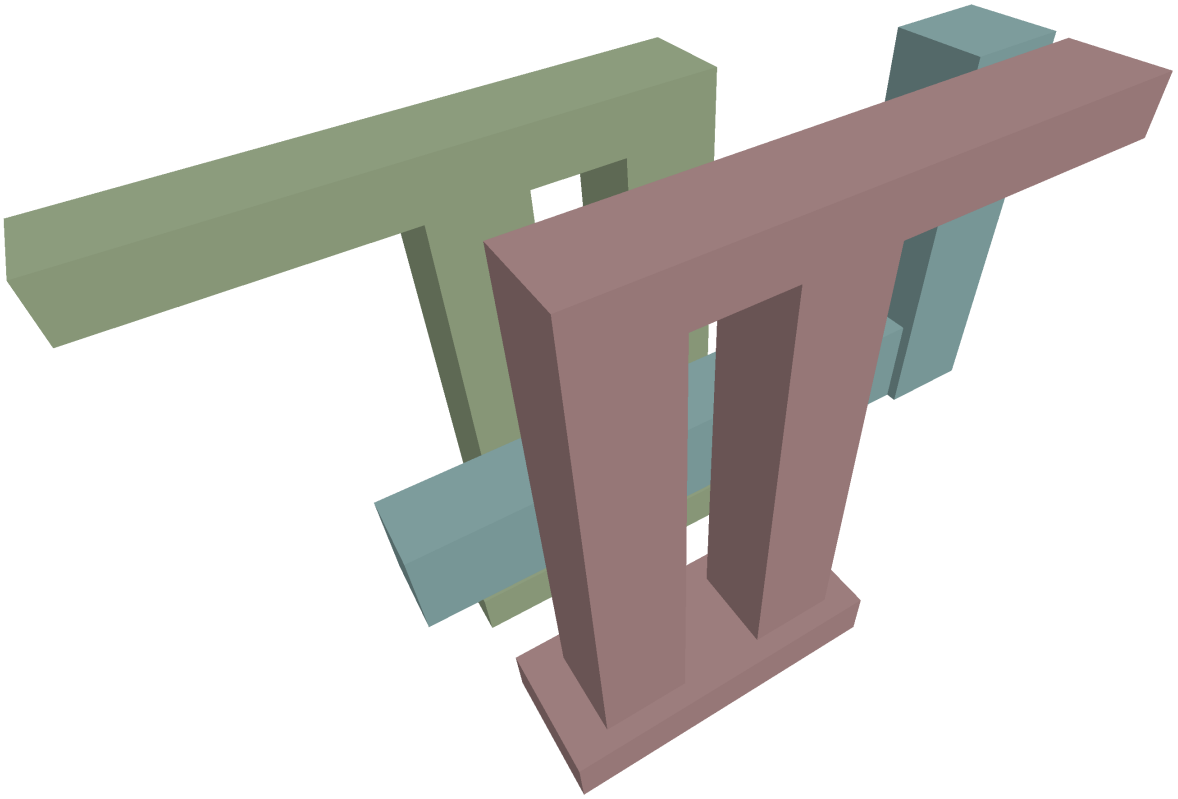


Figure 75: nfet example (3D view[7])

Table 76: Extracted Parasitic Resistances

Device	Description	Layer	Net	Node1	Node2	MAGIC [Ω]	KPEX/2.5D [Ω]
R1	Wire	li1	D	pin D	internal D.$3.18		30.494
R4	Wire	li1	D	internal D.$1.18	internal D.$3.18		25.6
R2	Via	licon	D	internal D.$0.13 (nsdm)	internal D.$1.18 (li1)		185.0
R5	Via	licon	D	internal D.$2.13 (nsdm)	internal D.$3.18 (li1)		185.0
R3	Wire	nsdm	D	internal D.$0.13	D.P0.13 (D device terminal)		0.0
R6	Wire	nsdm	D	internal D.$2.13	D.P0.13 (D device terminal)		0.0
R7	Wire	nsdm	G	pin G	internal G.$1.18		6.4
R8	Via	licon	G	internal G.$0.17 (poly)	internal G.$1.18 (li1)		152.0
R9	Wire	poly	G	internal G.$0.17	G.P0.17 (G device terminal)		234.381
R10	Wire	li1	S	pin S	internal S.$1.18		43.294
R13	Wire	nsdm	S	internal S.$1.18	S.$3.18 (S device terminal)		0.0
R11	Via	licon	S	internal S.$0.13 (nsdm)	internal S.$1.18 (li1)		185.0
R14	Via	licon	S	internal S.$2.13 (nsdm)	internal S.$3.18 (li1)		185.0
R12	Wire	nsdm	S	internal S.$0.13	S.P0.13 (S device terminal)		0.0
R15	Wire	nsdm	S	internal S.$2.13	S.P0.13 (S device terminal)		0.0

Warning

To improve readability, the illustrations in Figure 76 and Figure 77 do not depict all nodes correctly in regard of the the z-axis orientation

- nodes on metal layers are depicted residing on the top side of the layers and not inside or below the wires / vias
- currently, we treat the metal layer z-axis as an equipotential
- so in reality, R1, R4, R7, R9 would be the whole wire section

Figure 76 illustrates the parasitic resistance contributions for net D (all nets can be seen in Figure 75). Net G is illustrated in Figure 77. Net S is not worth showing, as it is similar to net D.

MAGIC extraction results are shown below.

```
.subckt nfet_li1 VNB G S D
X0 D G S VNB sky130_fd_pr__nfet_01v8 ad=0.169 pd=1.82 as=0.169 ps=1.82 w=0.65 l=0.15
R0 G G.n0 373.693
.ends
```

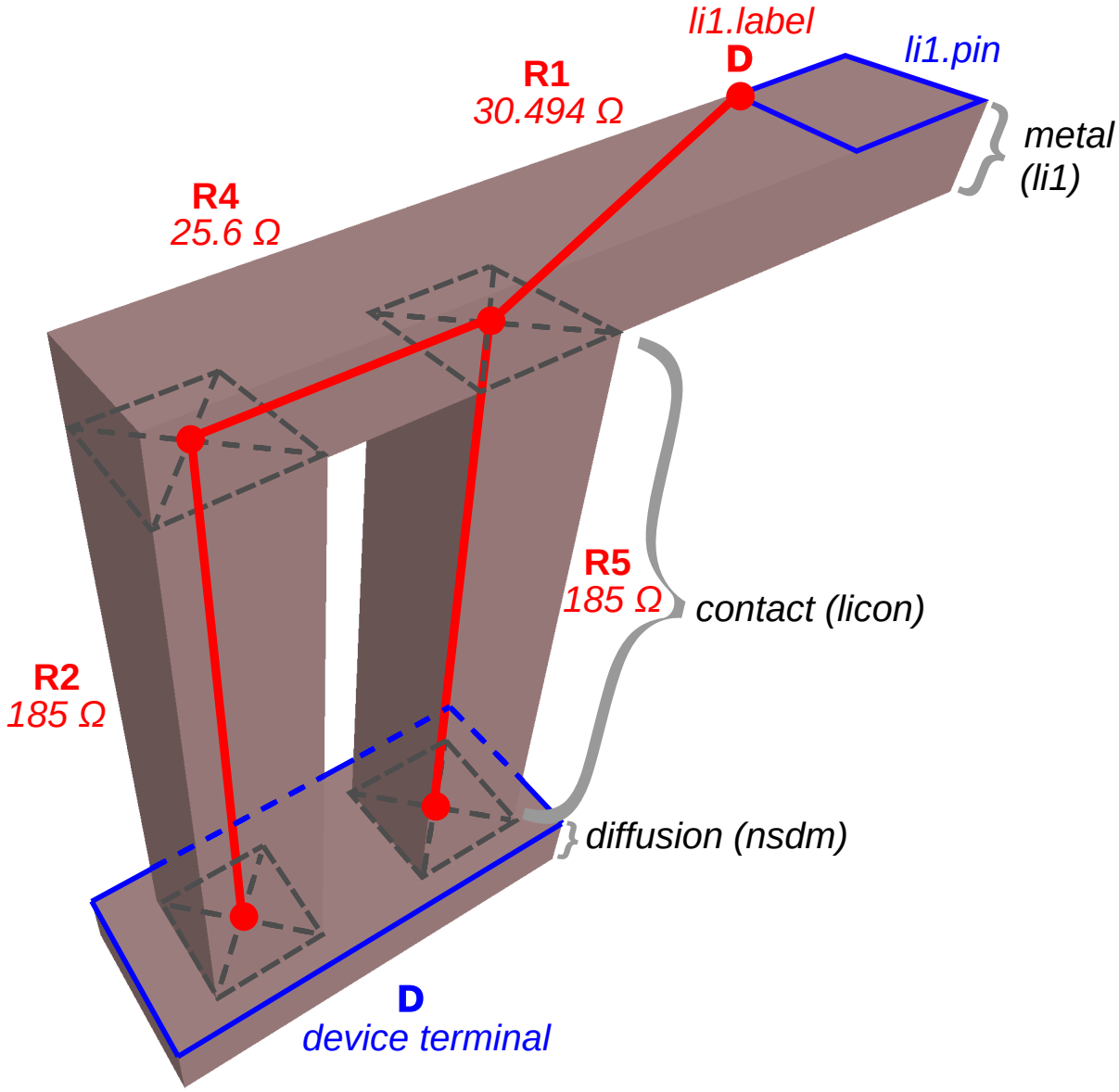


Figure 76: Resistance extraction of nfet example (only net D)"

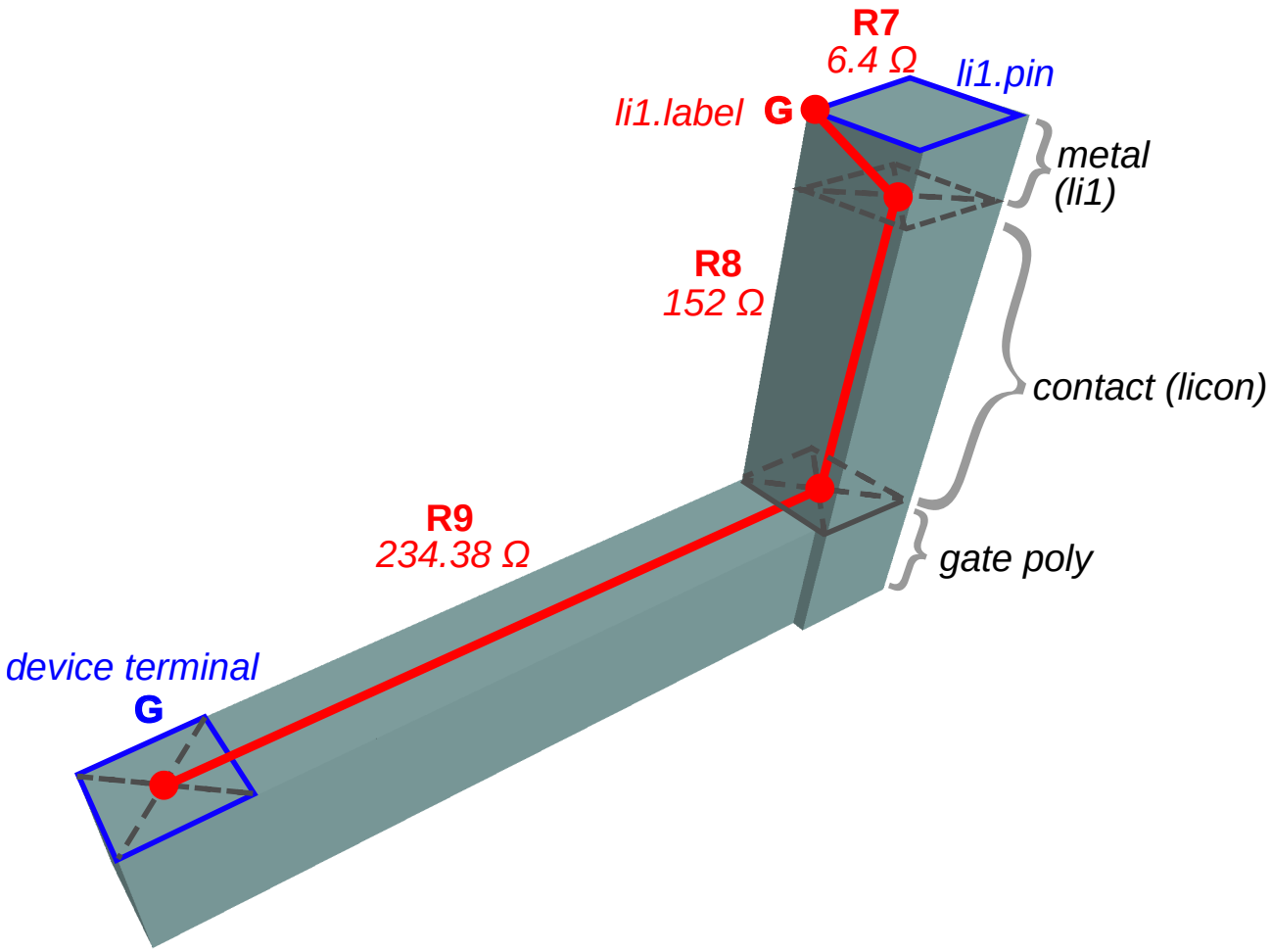


Figure 77: Resistance extraction of nfet example (only net G)”

> **Note**
>
> Notes about this comparison:
>
> - We can see that MAGIC did not extract the interconnection resistances
> - Only exception is `G` ↔ `G.n0`, but node `G.n0` is dangling and not connected

10 Netlist reduction: Feasibility study

10.1 TICER algorithm: TIme-Constant Equilibration Reduction

This section reviews the original TICER paper (Sheehan 1999), and illustrates the main ideas.

Given N-terminal star network:

- node N is the center
- terminals labeled from 0 to $N-1$ (in Figure 78, m is used instead of $N-1$)
- each branch, going from the terminal T_i to N, contains a resistor and a capacitor in parallel
- resistor values are given as conductance (easier parallel summation): g_{ij} [℧]
- capacitances values: c_{ij} [F]

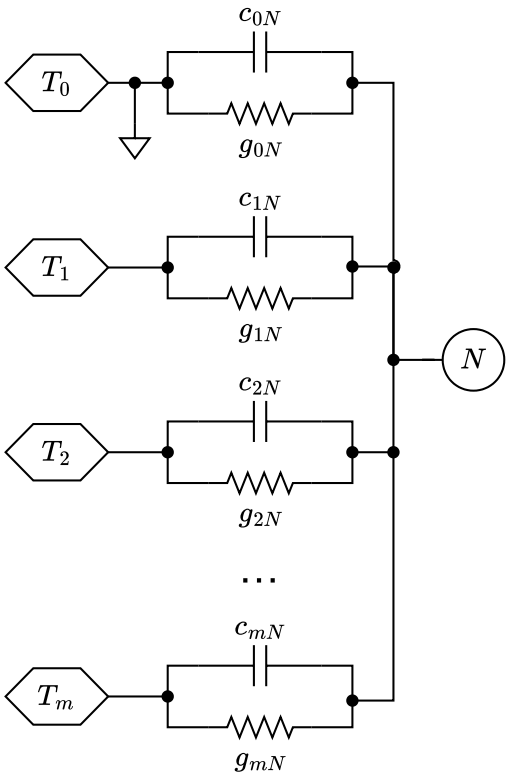


Figure 78: N-terminal star network

Sum of resistances γ_N, and capacitances χ_N:

$$\gamma_N = \sum_{k=0}^{N-1} g_{kN}, \quad \chi_N = \sum_{k=0}^{N-1} c_{kN} \tag{1.2}$$

The time constant τ_N of a given node N then is:

$$\tau_N = \frac{\chi_N}{\gamma_N} = \frac{\sum_{k=0}^{N-1} c_{kN}}{\sum_{k=0}^{N-1} g_{kN}} \tag{1.3}$$

Basic node elimination idea: Given a desired frequency window, using the time-constants, we can classify nodes into 3 groups: slow / normal / quick. Then we can eliminate slow and quick nodes, as the normal nodes will suffice to preserve the behavior for the frequency window:

RC circuit in Laplace domain:

$$(s\mathbf{C} + \mathbf{G})\mathbf{v} = \mathbf{Y}\mathbf{v} = \mathbf{J} \tag{2.1}$$

- $s \in \mathbb{C}$: Complex Frequency
- $\mathbf{C} \in \mathbb{R}^{N-1\times N-1}$: nodal capacitances
- $\mathbf{G} \in \mathbb{R}^{N-1\times N-1}$: nodal conductances
- $\mathbf{v} \in \mathbb{R}^{N-1}$: nodal voltages
- $\mathbf{J} \in \mathbb{R}^{N-1}$: current sources at the nodes

This can be written as a block system:

$$\begin{bmatrix} \tilde{\mathbf{Y}} & \mathbf{y} \\ \mathbf{y}^T & s\chi_N + \gamma_N \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{v}} \\ v_N \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{J}} \\ j_N \end{bmatrix} \tag{2.2}$$

> **Note**
>
> In equation (2.2), Sheehan uses the tilde accent to name the remainder of the matrix $\mathbf{Y}$ as $\tilde{\mathbf{Y}}$, when looking only at the last row and column (same principle applies to vector $\mathbf{v}$ and $\tilde{\mathbf{v}}$):
>
> $$\mathbf{Y} = \left[\begin{array}{cccc:c} y_{11} & y_{12} & \cdots & y_{1,N-1} & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2,N-1} & y_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hdashline y_{N1} & y_{N} & \cdots & y_{N,N-1} & y_{NN} \end{array}\right] = \left[\begin{array}{ccccc} & & & & y_{1N} \\ & \tilde{\mathbf{Y}} & & & y_{2N} \\ & & & & \vdots \\ \hdashline y_{N1} & y_{N2} & \cdots & y_{N,N-1} & y_{NN} \end{array}\right] = \left[\begin{array}{c:c} \tilde{\mathbf{Y}} & \mathbf{y} \\ \hdashline \mathbf{y}^T & y_{NN} \end{array}\right] \tag{1}$$

10.1.1 Comments on page 2, equation (2.3)

To arrive at equations (2.3), (2.4), and (2.5) from equation (2.2), the process involves solving for v_N, the voltage at the node to be eliminated (node N), and substituting it back into the system of equations.

Starting with equation (2.2):

$$\begin{bmatrix} \tilde{\mathbf{Y}} & \mathbf{y} \\ \mathbf{y}^T & s\chi_N + \gamma_N \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{v}} \\ v_N \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{J}} \\ j_N \end{bmatrix} \tag{2.2}$$

we have two block equations:

$$\tilde{\mathbf{Y}}\tilde{\mathbf{v}} + \mathbf{y}v_N = \tilde{\mathbf{J}} \qquad \text{(First block equation of 2.2)}$$

$$\mathbf{y}^T\tilde{\mathbf{v}} + (s\chi_N + \gamma_N)v_N = j_N \qquad \text{(Second block equation of 2.2)}$$

Solve the second block equation for v_N:

$$v_N = \frac{j_N - \mathbf{y}^T\tilde{\mathbf{v}}}{s\chi_N + \gamma_N}$$

Substitute the expression for v_N in the first block equation:

$$\tilde{\mathbf{Y}}\tilde{\mathbf{v}} + \mathbf{y}\left(\frac{j_N - \mathbf{y}^T\tilde{\mathbf{v}}}{s\chi_N + \gamma_N}\right) = \tilde{\mathbf{J}}$$

Simplifying, we get:

$$\left(\tilde{\mathbf{Y}} - \frac{\mathbf{y}\mathbf{y}^T}{s\chi_N + \gamma_N}\right)\tilde{\mathbf{v}} = \tilde{\mathbf{J}} - \frac{\mathbf{y}j_N}{s\chi_N + \gamma_N}$$

This leads to the modified system:

$$(\tilde{Y} - \mathbf{E})\tilde{\mathbf{v}} = \tilde{\mathbf{J}} - \mathbf{F} \tag{2.3}$$

Where:

$$\mathbf{E} = \frac{\mathbf{y}\mathbf{y}^T}{s\chi_N + \gamma_N}$$

$$\mathbf{F} = \frac{\mathbf{y}j_N}{s\chi_N + \gamma_N}$$

> **Note**
>
> There's a typo in the original paper in (2.3), the printed variable $\mathbf{v_N}$ should instead be $\tilde{\mathbf{v}}$.

So we arrive at:

$$\mathbf{E}_{ij} = \frac{(g_{iN} + sc_{iN})(g_{jN} + sc_{jN})}{s\chi_N + \gamma_N} \tag{2.4}$$

$$\mathbf{F}_i = \frac{g_{iN} + sc_{iN}}{s\chi_N + \gamma_N} j_N \tag{2.5}$$

10.1.2 Quick Nodes

Suppose:

- we eliminate a quick node N
- $s\chi_N \ll \gamma_N$
- equivalently, $|s\tau_N| \ll 1$
- we approximate $\mathbf{E}_{ij}$ from (2.4)
- we eliminate higher-order terms, containing factors like s^n for $n \geq 2$

$$\begin{aligned}\mathbf{E}_{ij} &= \frac{(g_{iN} + sc_{iN})(g_{jN} + sc_{jN})}{s\chi_N + \gamma_N} \\ &= \frac{g_{iN}g_{jN}}{s\chi_N + \gamma_N} + \frac{s\left(g_{jN}c_{iN} + g_{iN}c_{jN}\right)}{s\chi_N + \gamma_N} + \frac{s^2 c_{iN}c_{jN}}{s\chi_N + \gamma_N}\end{aligned}$$

$$\mathbf{E}_{ij} \approx \frac{g_{iN}g_{jN}}{\cancel{s\chi_N} + \gamma_N} + s\frac{g_{jN}c_{iN} + g_{iN}c_{jN}}{\cancel{s\chi_N} + \gamma_N} + \frac{\cancel{s^2 c_{iN}c_{jN}}}{\cancel{s\chi_N} + \gamma_N}$$

$$\mathbf{E}_{ij} \approx \frac{g_{iN}g_{jN}}{\gamma_N} + s\frac{g_{jN}c_{iN} + g_{iN}c_{jN}}{\gamma_N} \tag{3.2}$$

Eliminating a quick node:

Recipe for translating (3.2) into a modified circuit:

- remove all resistors and capacitors connecting other nodes to N
- insert new resistors and capacitors between former neighbors i, j of N according to these rules:

– If nodes i and j had been connected to N through conductances g_{iN} and g_{jN}, insert conductance $\frac{g_{iN}g_{jN}}{\gamma_N}$ between i and j
– If node i had a capacitor c_{iN} to N, and node j had a conductance g_{jN} to N, insert a capacitor $\frac{c_{iN}g_{jN}}{\gamma_N}$ between i and j

10.1.3 Slow Nodes

Suppose:

- we eliminate a slow node N
- $s\chi_N \gg \gamma_N$
- equivalently, $|s\tau_N| \gg 1$
- we approximate $\mathbf{E}_{ij}$ from (2.4)
- we retain terms containing s
- to preserve DC characteristics, we use $\frac{g_{iN}g_{jN}}{\gamma_N}$ in place of whatever constant terms come from the equation
- higher-order terms in γ_N and s are neglected to simplify the expression.

Starting with (2.4):

$$\mathbf{E}_{ij} = \frac{(g_{iN} + sc_{iN})(g_{jN} + sc_{jN})}{s\chi_N + \gamma_N} \tag{2.4}$$

Given the approximation:

$$\frac{1}{s\chi_N + \gamma_N} \approx \frac{1}{s\chi_N}\left(1 - \frac{\gamma_N}{s\chi_N}\right) \tag{4.2}$$

Substitute (4.2) into (2.4):

$$\begin{aligned}\mathbf{E}_{ij} &\approx \frac{(g_{iN} + sc_{iN})(g_{jN} + sc_{jN})}{s\chi_N}\left(1 - \frac{\gamma_N}{s\chi_N}\right) \\ &\approx \left(\frac{g_{iN}g_{jN}}{s\chi_N} + \frac{\cancel{s}\left(g_{jN}c_{iN} + g_{iN}c_{jN}\right)}{\cancel{s}\chi_N} + \frac{\cancel{s^2}c_{iN}c_{jN}}{\cancel{s}\chi_N}\right)\left(1 - \frac{\gamma_N}{s\chi_N}\right)\end{aligned}$$

Expand:

$$\begin{aligned}\mathbf{E}_{ij} = &\frac{g_{iN}g_{jN}}{s\chi_N} - \frac{g_{iN}g_{jN}\gamma_N}{s^2\chi_N^2} \\ &+ \frac{g_{jN}c_{iN} + g_{iN}c_{jN}}{\chi_N} - \frac{\left(g_{jN}c_{iN} + g_{iN}c_{jN}\right)\gamma_N}{s\chi_N^2} \\ &+ \frac{sc_{iN}c_{jN}}{\chi_N} - \frac{\cancel{s}c_{iN}c_{jN}\gamma_N}{\cancel{s}\chi_N^2}\end{aligned}$$

After we remove constant terms:

$$\mathbf{E}_{ij} \approx \frac{g_{iN}g_{jN}}{s\chi_N} - \frac{g_{iN}g_{jN}\gamma_N}{s^2\chi_N^2} - \frac{\left(g_{jN}c_{iN} + g_{iN}c_{jN}\right)\gamma_N}{s\chi_N^2} + s\frac{c_{iN}c_{jN}}{\chi_N}$$

Remove higher-order terms in γ_N and s:

$$\mathbf{E}_{ij} \approx \frac{g_{iN} g_{jN}}{s\chi_N} + s\frac{c_{iN} c_{jN}}{\chi_N}$$

Sheehan arrives at (4.1):

$$\mathbf{E}_{ij} \approx \frac{g_{iN} g_{jN}}{\gamma_N} + s\frac{c_{iN} c_{jN}}{\chi_N} \tag{4.1}$$

- Rules:
  - If nodes i and j had been connected to N through conductances g_{iN} and g_{jN}, insert conductance $\frac{g_{iN} g_{jN}}{\gamma_N}$ from i to j
  - If node i had a capacitor c_{iN} to N, and node j had a capacitor c_{jN} to N, insert a capacitor $\frac{c_{iN} c_{jN}}{\gamma_N}$ from i to j

10.2 TICER 2007 paper: Pseudocode

This section reviews some aspects of the 2007 TICER paper (Sheehan 2007).

User-supplied algorithm parameters:

- Set of **fixed** nodes (designated to be left alone, i.e., not to be eliminated)
- **MaxDeg** $\in \mathbb{N} \setminus \{0, 1\}$: number of passes (user-supplied parameter)
- $f^{\max}$: maximum operating frequency of interest
- $\epsilon \in \mathbb{R}_{[0,1]}$: small number enforcing the requirement that $|s\tau_N| \ll 1$

In the paper, figure 5 and 6 demonstrate the algorithm for eliminating a quick node (here is a python conversion of those).

```
def eliminate_quick_node(N: Node):
   neighbors = nodes_incident_to(node=N)
   for i in neighbors:
      g_iN = conductance[i, N]
      if g_iN == 0:
         continue
      for j in neighbors:
         if i <= j:
            continue
         g_jN = conductance[j, N]
         c_jN = capacitance[j, N]
         gamma_N = incident_conductance_sum(node=N)
         if g_jN > 0:
            add_conductance(i, j, g_iN * g_jN / gamma_N)
         if c_jN > 0:
            add_capacitance(i, j, g_iN * c_jN / gamma_N)
   remove_all_incident_conductances_and_capacitances(i, N)
```

```
def ticer(freq_max: float, epsilon: float, max_deg: int):
   for deg in range(2, max_deg):
      Q = queue.Queue(nodes_not_fixed())
      while not Q.empty():
         N = Q.get()  # pop
         if number_of_incident_resistors(N) > deg:
            continue
```

```
            tau_N = time_constant(N)
            if 2 * math.pi * freq_max * tau_N <= epsilon:
               # ensure the neighbors are in the queue,
               # because the neighbors' incident resistors and time constants
               # might have been changed, so they should be reconsidered for elimination
               missing_neighbors = [n in nodes_incident_to(node=N) if n not in Q]
               q.push(missing_neighbors)

               eliminate_quick_node(N)

   # NOTE: "leaf node" N means, number_of_incident_resistors(N) == 1
   for N in leaf_nodes():
      if N in fixed_nodes():  # N is protected
         continue

      tau_N = time_constant(N)
      if 2 * math.pi * freq_max * tau_N <= epsilon:
         eliminate_quick_node(N)
```

10.2.1 Pseudocode for slow node elimination

Sheehan mentions that elimination of slow nodes is of reduced importances, and therefore does not explicitly provide pseudocode for this case, instead refers to the original TICER paper (Sheehan 1999).

So if we fill in the gaps we arrive at:

```
def eliminate_slow_node(N: Node):
   neighbors = nodes_incident_to(node=N)
   for i in neighbors:
      g_iN = conductance[i, N]
      c_iN = capacitance[i, N]
      for j in neighbors:
         if i <= j:
            continue
         g_jN = conductance[j, N]
         c_jN = capacitance[j, N]
         gamma_N = incident_conductance_sum(node=N)
         if g_iN > 0 and g_jN > 0:
            add_conductance(i, j, g_iN * g_jN / gamma_N)
         if c_iN > 0 and c_jN > 0:
            add_capacitance(i, j, c_iN * c_jN / gamma_N)
   remove_all_incident_conductances_and_capacitances(i, N)
```

To integrate this into the algorithm, we additionally need a minimum frequency $f^{\min}$ parameter supplied by the user.

```
def ticer(freq_min: float, freq_max: float, epsilon: float, max_deg: int):
   for deg in range(2, max_deg):
      Q = queue.Queue(nodes_not_fixed())
      while not Q.empty():
         N = Q.get()  # pop
```

```
        if number_of_incident_resistors(N) > deg:
            continue
        tau_N = time_constant(N)
        if 2 * math.pi * freq_max * tau_N <= epsilon:
            # NOTE: ensure the neighbors are in the queue,
            # because the neighbors' incident resistors and time constants
            # might have been changed, so they should be reconsidered for elimination
            missing_neighbors = [n in nodes_incident_to(node=N) if n not in Q]
            q.push(missing_neighbors)

            eliminate_quick_node(N)

        if 2 * math.pi * freq_min * tau_N <= epsilon:
            # NOTE: ensure the neighbors are in the queue,
            # because the neighbors' incident resistors and time constants
            # might have been changed, so they should be reconsidered for elimination
            missing_neighbors = [n in nodes_incident_to(node=N) if n not in Q]
            q.push(missing_neighbors)

            eliminate_slow_node(N)

# NOTE: "leaf node" N means, number_of_incident_resistors(N) == 1
for N in leaf_nodes():
    if N in fixed_nodes():  # N is protected
        continue

    tau_N = time_constant(N)
    if 2 * math.pi * freq_max * tau_N <= epsilon:
        eliminate_quick_node(N)
```

10.3 Example from TICER 2007 paper

Let's reduce the example presented by Sheehan in the from the 2007 paper (Sheehan 2007), as shown in Figure 79. The goal is to eliminate the quick node v_3.

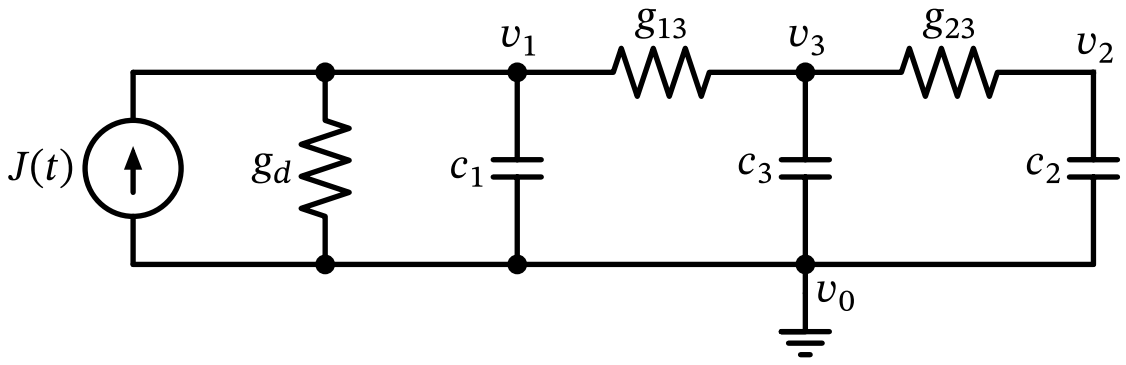


Figure 79: Passive RC circuit shown in Sheehan (2007).

- Time constant $\tau_{v_3} = \frac{c_3}{g_{13}+g_{23}}$
- Quick node condition: is_quick$(N) := 2\pi f^{\max}\tau_N < \epsilon$

Elimination of node N_2:

- Incident devices R_{13}, R_{23}, C_3 are removed
- New devices for neighbors
  - Between nodes (v_0, v_1):
    - add $c_{10} = \frac{g_{13} * c_3}{g_{13}+g_{23}}$
  - Between nodes (v_0, v_2):
    - add $c_{20} = \frac{g_{23} * c_3}{g_{13}+g_{23}}$
  - Between nodes (v_1, v_2):
    - add $g_{12} = \frac{g_{13} * g_{23}}{g_{13}+g_{23}}$

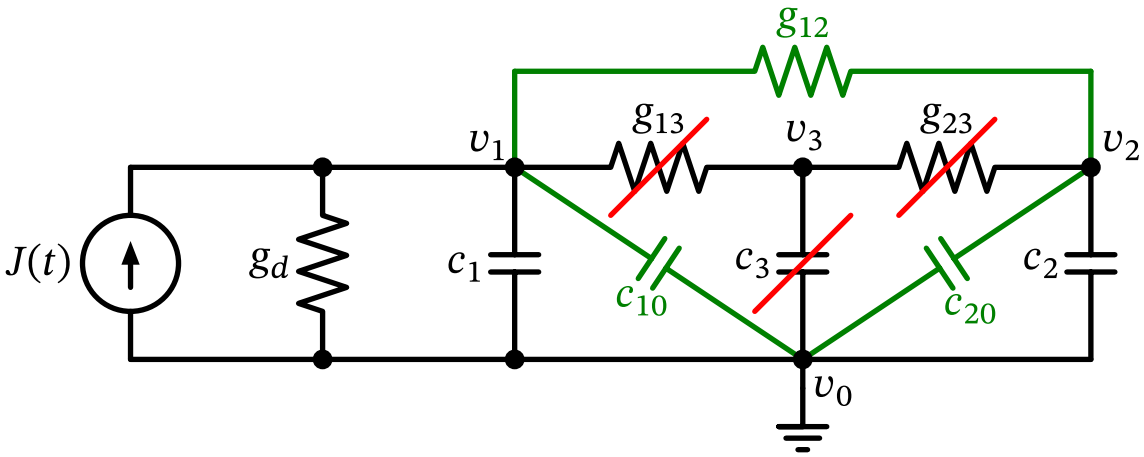


Figure 80: Passive RC circuit after the elimination of v_3.

10.3.1 Modified Nodal Analysis in the Laplace domain

RC circuit in Laplace domain:

$$(s\mathbf{C} + \mathbf{G})\mathbf{v} = \mathbf{Y}\mathbf{v} = \mathbf{J} \tag{2.1}$$

Solve for voltages:

$$\mathbf{v} = (s\mathbf{C} + \mathbf{G})^{-1}\mathbf{J} = Y^{-1}\mathbf{J}$$

Transfer function:

$$H(s) = \frac{\mathbf{v}(s)}{\mathbf{J}(s)} = Y^{-1}(s) = (s\mathbf{G} + \mathbf{C})^{-1}$$

E.g. the transfer function (for input $J_1(s)$, output $V_2(s)$) is:

$$H(s) = \frac{\text{adj}(s\mathbf{C} + \mathbf{G})_{21}}{det(s\mathbf{C} + \mathbf{G})}$$

10.3.2 Analysis of the original circuit

So for the original circuit of Figure 79, we have 3 nodes, in addition to GND (v_0).

So we have 3 nodal equations:

$$\begin{aligned} c_1\dot{v_1} + g_d v_1 + g_{13}(v_1 - v_3) &= j_1(t) \\ c_2\dot{v_2} + g_{23}(v_2 - v_3) &= 0 \\ c_3\dot{v_3} + g_{13}(v_3 - v_1) + g_{23} * (v_3 - v_2) &= 0 \end{aligned}$$

Laplace domain (MNA[8] form):

$$\left(s \begin{bmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix} + \begin{bmatrix} g_d + g_{13} & 0 & -g_{13} \\ 0 & g_{23} & -g_{23} \\ -g_{13} & -g_{23} & g_{13} + g_{23} \end{bmatrix} \right) \begin{bmatrix} V_1(s) \\ V_2(s) \\ V_3(s) \end{bmatrix} = \begin{bmatrix} J_1(s) \\ 0 \\ 0 \end{bmatrix}$$

So given the values from Sheehan's numerical example:

- $g_d = g_{13} = g_{23} = 1$
- $c_1 = c_2 = 1$
- $c_3 = 0.01$

Node v_3 is a quick node:

- Time constant: $\tau_{v_3} = \frac{c_3}{g_{13}+g_{23}} = \frac{0.001}{1+1} = 0.0005$
- Frequency: $f = \frac{1}{2\pi\tau_{v_3}} \approx 318.31\,\text{Hz}$

We have a system with

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

Transfer function:

$$H(s) = \frac{1}{0.01s^3 + 2.03s^2 + 4.02s + 1}$$

10.3.3 Analysis of the reduced circuit

The reduced circuit in Figure 80 only has 2 remaining nodes (besides GND).

So we have 2 nodal equations:

$$\begin{aligned} (c_1 + c_{10})\dot{v_1} + g_d v_1 + g_{12}(v_1 - v_2) &= j_1(t) \\ (c_2 + c_{20})\dot{v_2} + g_{12}(v_2 - v_1) &= 0 \end{aligned}$$

Laplace domain (MNA[9] form):

$$\left(s \begin{bmatrix} c_1 + c_{10} & 0 \\ 0 & c_2 + c_{20} \end{bmatrix} + \begin{bmatrix} g_d + g_{12} & -g_{12} \\ -g_{12} & g_{12} \end{bmatrix} \right) \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} J_1(s) \\ 0 \end{bmatrix}$$

New devices:

[8] Modified Nodal Analysis
[9] Modified Nodal Analysis

- $c_{10} = \frac{g_{13} * c_3}{g_{13}+g_{23}} = \frac{1*0.01}{1+1} = 0.005$
- $c_{20} = \frac{g_{23} * c_3}{g_{13}+g_{23}} = \frac{1*0.01}{1+1} = 0.005$
- $g_{12} = \frac{g_{13} * g_{23}}{g_{13}+g_{23}} = \frac{1*1}{1+1} = 0.5$

We have a system with:

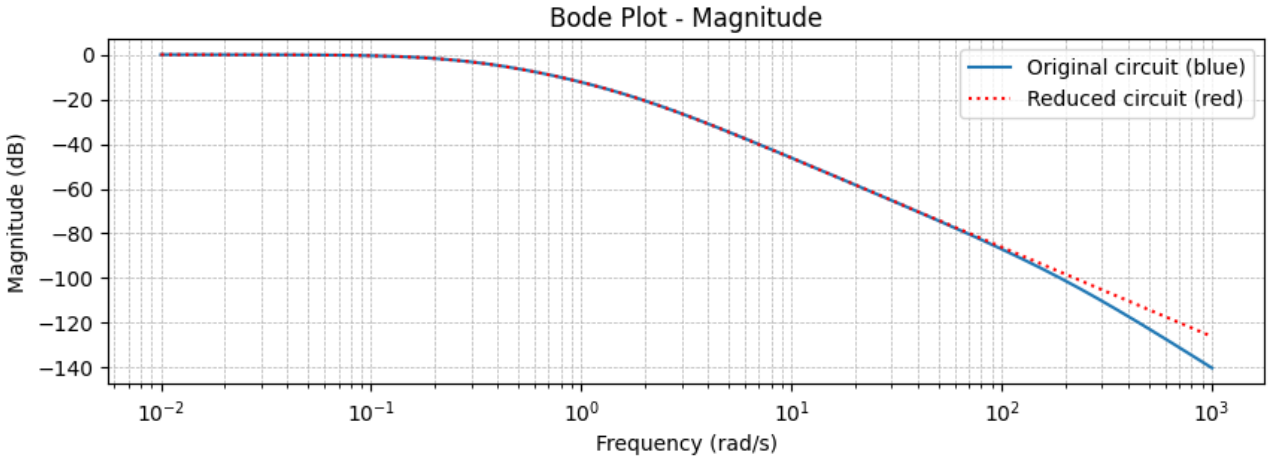
$$\mathbf{C} = \begin{bmatrix} 1.005 & 0 \\ 0 & 1.005 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.5 \end{bmatrix}$$

Transfer function:

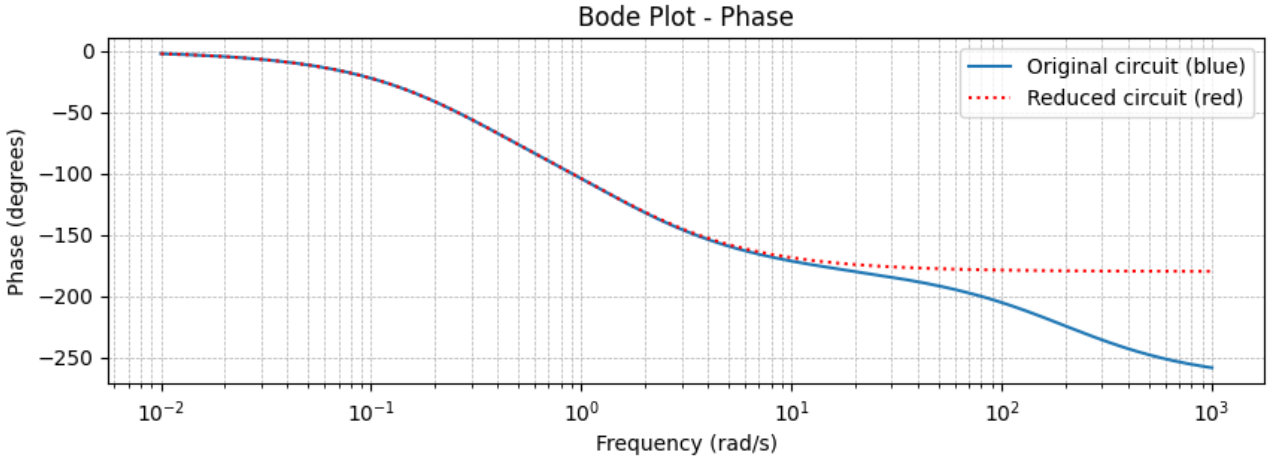
$$H(s) = \frac{0.5}{1.010025s^2 + 2.01s + 0.5}$$

10.3.4 Comparison of original and reduced circuits

In the Bode plots, Figure 81a and Figure 81b, we see that magnitude and phase responses stays the same up to a certain frequency.



(a) Bode plot: Frequency versus Magnitude



(b) Bode plot: Frequency versus Phase

Figure 81: Bode Plots

In the original circuit we have an additional pole at $s = -201$, which is expendable (Figure 82).

In Figure 83 we remove this pole, for better comparison with the reduced circuit (Figure 84), where that pole is no longer present, but the others still are.

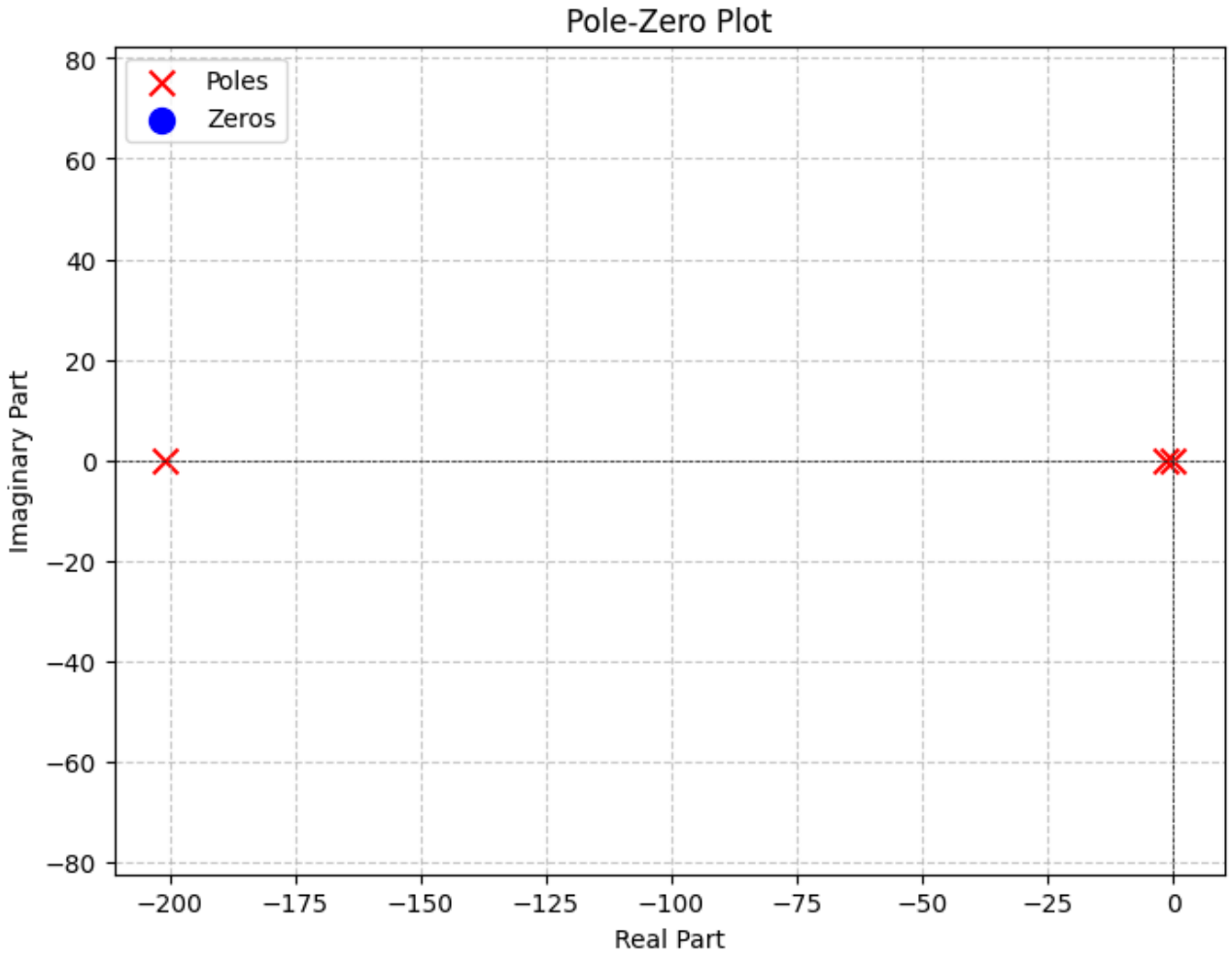


Figure 82: Pole-Zero plot: Original circuit

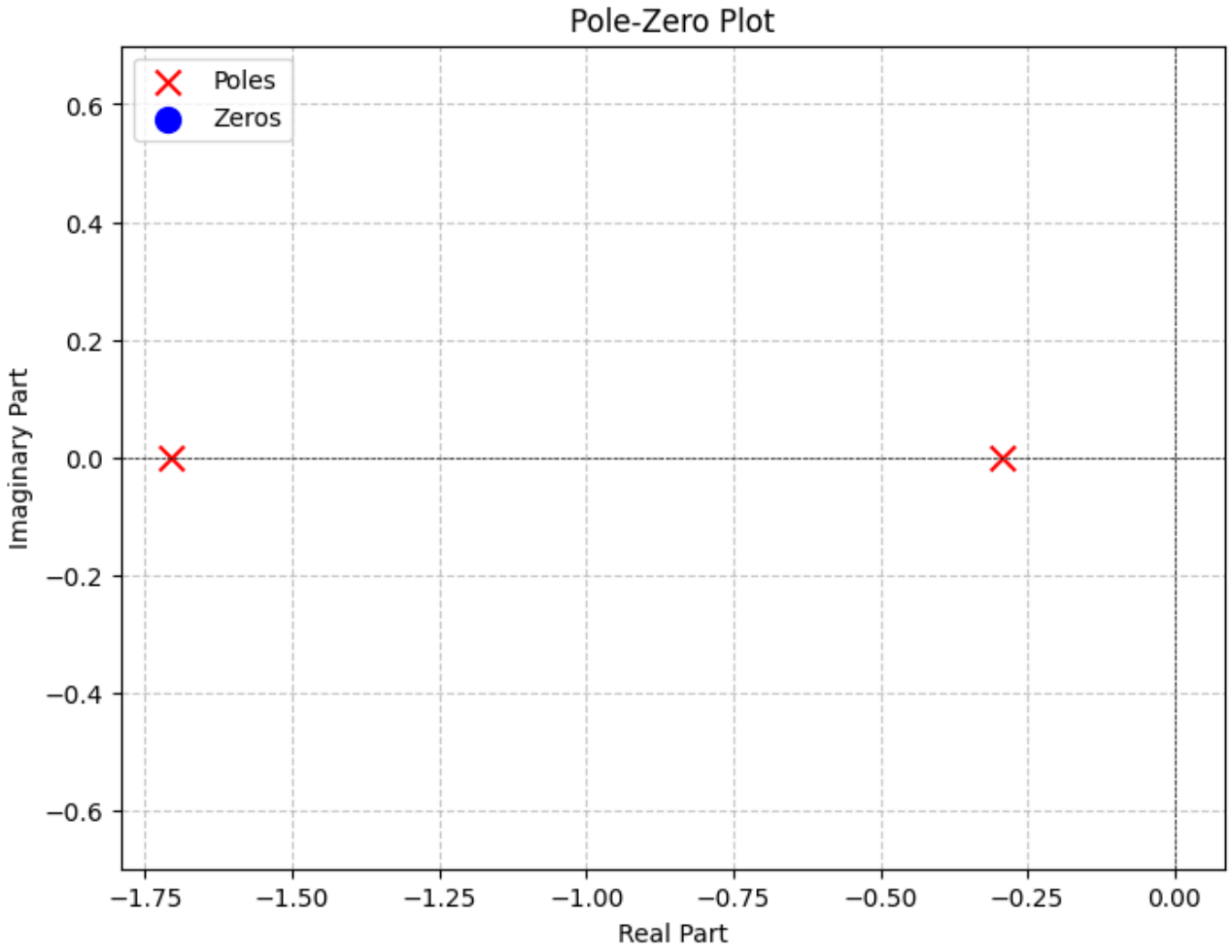


Figure 83: Pole-Zero plot: Original circuit (without the pole at $s = -201$)

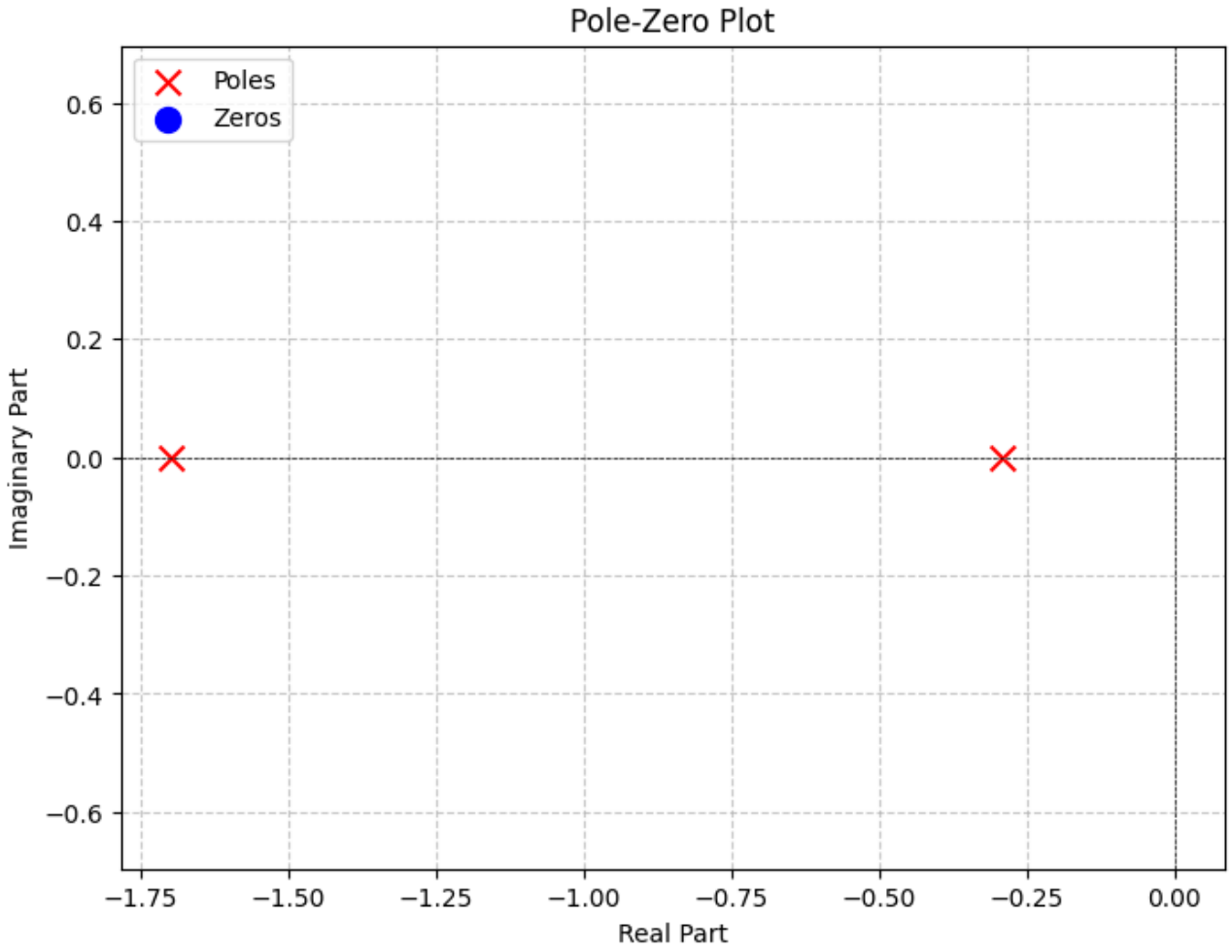


Figure 84: Pole-Zero plot: Reduced circuit

11 Appendix

11.1 PEX25D File Format: Represent Interconnect Geometry and Materials

The **PEX25D format** (Parasitic Extraction 2.5D) describes interconnect geometry and materials for exchange between tools. It combines two-dimensional (XY) conductor shapes with a vertical process stack, so a solver adapter can reconstruct the three-dimensional structure without the original layout or PDK files. The same structure can describe an electrostatic capacitance problem and, with resistance parameters and terminals, a DC resistance problem.

This appendix specifies the PEX25D format and explains its protobuf API and command-line tools. Geometry-only output uses `PEX25D 1.0-rc1`; output with explicit capacitance accounting uses `PEX25D 1.0-rc2 REQUIRES CAPABILITIES CAPACITANCE_ACCOUNTING`. The format version is independent of the KLayout-PEX package version.

> **i** The PEX25D format and the KPEX/2.5D engine
>
> The **PEX25D format** defines an interchange description. `kpex pex25d` generates files in this format; `pex25d` inspects and processes them.
> **KPEX/2.5D** is the analytical extraction engine described in Section 6, selected with `kpex extract --2.5D`. It uses fitted technology coefficients and shielding rules. Generating or resolving PEX25D files does not run that engine, and the format does not contain its fitted capacitance tables.

11.1.1 Why use PEX25D?

- **Exchange a physical problem between tools.** A layout producer and a solver adapter can agree on conductor identities, geometry, materials, and terminals through one documented interface. The adapter does not need to repeat LVS or understand the producer's PDK representation.
- **Compare different solvers on the same structure.** Export one PEX25D description through different solver adapters to compare extracted results using the same conductor geometry, material properties, and net identities. This helps distinguish solver differences from differences in input preparation. For a meaningful comparison, also account for each adapter's meshing, boundary treatment, and supported format features.
- **Inspect and reproduce solver input.** Save a readable text file for review, a compact protobuf scene for repeated runs, or a small hand-written example for a bug report. Validation can run before meshing or starting an expensive field solve.
- **Change process corners consistently.** Only the ground plane and metal profiles declare absolute heights. Via endpoints, dielectric bands, and film growth refer to those profiles, so a change to a metal height propagates through the stack. `INCLUDE` can separate a technology stack from reusable geometry.
- **Share one resolver across consumers.** The protobuf API separates what the author wrote from the resolved scene. Adapters consume the latter and share the interpretation of `CONNECTS`, `BETWEEN`, `WRAPS`, and terminal intersections.
- **Pass prepared input directly to C++ engines.** Python can prepare and resolve the structure, then forward a binary `PEX25DScene` to a C++ consumer using generated protobuf bindings. The C++ engine needs neither a PEX25D text parser nor a second implementation of stack resolution. This supports keeping preparation in Python while moving computation-intensive work to C++.
- **Preserve exact dimensions.** Coordinates are integer grid units internally. Coincident vertical faces remain equal after relative heights are computed, without accumulating floating-point discrepancies.

The PEX25D format describes the physical interconnect structure. An analytical consumer still needs its own technology data, potentially matched through `META technology` and the layer names; this does not imply that the current KPEX/2.5D engine reads PEX25D files.

The following are outside the scope of version 1.x:

- Extracted results and fitted parasitic models, including calibrated capacitance tables.
- Inductance.
- Frequency-dependent effects, such as skin effect or dispersive permittivity.
- Self-heating.
- Electromigration.
- Substrate resistance; the ground plane is an ideal 0 V reference.

11.1.2 From layout to solver input

The workflow has five distinct operations:

Operation	Tool or API	Result
Generate from layout connectivity and PDK stack	`kpex pex25d`	`PEX25DFile`, optionally also `PEX25DScene`
Translate another tool's scene description	Importer; `pex25d import`	`PEX25DFile`
Read or re-encode a description	Text reader; `pex25d convert`	The same artifact kind in another encoding
Resolve relative geometry	Resolver; `pex25d resolve`	`PEX25DScene`
Mesh and export for a solver	Exporter; `pex25d export`	Native solver input files

Importers and exporters are plugins: `klayout-pex` ships some of them, further ones are installed as separate Python packages.

The intended project architecture keeps layout and technology preparation in **Python** and moves computation-intensive portions into **C++** engines. The resolved binary scene is the interface between those stages: Python produces it, and C++ reads the same protobuf message through generated bindings. This is the direction for moving computation to C++; it does not mean all current engines have already been ported.

`PEX25DFile` is the literal, unresolved model: it retains profile references and relative descriptions. The text reader flattens `INCLUDE` files and groups records by type. Comments, whitespace, and original record order are not preserved, so conversion is not a byte-for-byte text round trip.

`PEX25DScene` carries absolute Z extents, dielectric roots and wrap depths, conductor geometry grouped by layer, and terminal intersections. When a domain was requested, it contains the resulting box. Without a domain record, the resolver retains finite geometry bounds and leaves boundary selection to the adapter. The scene still describes 2.5D geometry; it is not a triangulated solver mesh.

Resolution is one-way. A scene has no representation in the PEX25D text syntax, and `convert` cannot reconstruct the original file from it. Keep the file when you want to edit relative stack definitions later.

11.1.3 Artifact filenames and encodings

The filename identifies both the artifact kind (unresolved file or resolved scene) and its encoding. The tools infer these from the following suffixes:

Conventional filename	Protobuf message	Encoding
`name.pex25d`	–	PEX25D textual format
`name.pex25d.pb`	`PEX25DFile`	Binary protobuf
`name.pex25d.textpb`	`PEX25DFile`	Protobuf text
`name.pex25d.scene.pb`	`PEX25DScene`	Binary protobuf
`name.pex25d.scene.textpb`	`PEX25DScene`	Protobuf text

Reading the PEX25D textual format produces a `PEX25DFile` message in memory.

A trailing `.gz` enables gzip for any of these filenames. Protobuf text (`textpb`) uses protobuf field names and braces; it is a different syntax from the PEX25D textual format.

11.1.4 A complete example

Download two_nets.pex25d to try the commands below without a layout or an LVS run. This small, illustrative stack is not calibrated PDK data. Conductor `A` crosses from `met1` to `met2` through one explicit via cut and has two terminals. Conductor `B` is a nearby metal polygon.

```
# Illustrative stack; these values do not represent a calibrated PDK.
PEX25D 1.0-rc1
UNITS LENGTH um GRID 0.0001
META technology example
META source_cell two_nets
META source_dbu 0.001

GROUND_PLANE subs Z_OFFSETS -0.4 -0.1
METAL met1 Z_OFFSETS 1.0 1.4
METAL met2 Z_OFFSETS 2.0 2.4
VIA via1 CONNECTS met1 met2

DIELECTRIC_BACKGROUND air PERMITTIVITY 1.0
DIELECTRIC_SIMPLE fox WRAPS subs PERMITTIVITY 3.9 \
    BETWEEN subs met1
DIELECTRIC_CONFORMAL lint WRAPS met1 PERMITTIVITY 7.3 \
    THICKNESS_OVER_WRAPPED 0.1 \
    THICKNESS_BESIDE_WRAPPED 0.08 \
    THICKNESS_ON_FIELD 0.0
DIELECTRIC_SIMPLE ild WRAPS lint PERMITTIVITY 4.1 \
    BETWEEN met1 met2

RESISTANCE TEMPERATURE 25.0
RESISTANCE METAL met1 SHEET 0.125
RESISTANCE METAL met2 SHEET 0.125
RESISTANCE VIA via1 PER_CUT 4.5

CONDUCTOR A neta
CONDUCTOR B netb
BOX CONDUCTOR A LAYER met1 LL 0.0 0.0 UR 1.0 0.5
BOX CONDUCTOR A LAYER via1 LL 0.2 0.1 UR 0.4 0.3
BOX CONDUCTOR A LAYER met2 LL 0.0 0.0 UR 1.0 0.5
POLYGON CONDUCTOR B LAYER met1 \
    OUTER 2.0 0.0 3.0 0.0 3.0 1.0 2.0 1.0
TERMINAL input CONDUCTOR A LAYER met1 KIND PIN \
    LL 0.0 0.0 UR 0.1 0.5
TERMINAL output CONDUCTOR A LAYER met2 KIND PIN \
    LL 0.9 0.0 UR 1.0 0.5

DOMAIN_MARGIN X 4.0 Y 4.0 Z 2.0

```

The via spans **1.4–2.0 µm**: the top of `met1` to the bottom of `met2`. The `lint` film reaches 0.1 µm above `met1` and 0.08 µm beside it. `ild` fills the band from the **bottom** of `met1` at 1.0 µm to the bottom of `met2` at 2.0 µm. Where both dielectrics are present, `lint` wins because it has wrap depth 1 and `ild` has depth 2.

For capacitance, `A` is equipotential despite its two terminals. For resistance, those terminals define the two nodes between which its distributed interconnect is reduced. `B` has no terminals, so it participates in capacitance but contributes no resistance network.

11.1.5 PEX25D textual format reference

In the syntax templates below, angle brackets identify values to supply; square brackets mark optional clauses. Neither is written literally into a file.

11.1.5.1 Records, versions, and grid units

Files are UTF-8 and line-oriented. Keywords are case-sensitive. Whitespace separates tokens, `#` starts a comment outside a quoted value, and a trailing backslash continues a record on the next line. Quote metadata values or paths containing whitespace. Clauses **within** a record must follow the documented order.

```
PEX25D <major>.<minor>[-<suffix>] \
    [REQUIRES CAPABILITIES <capability>[, <capability> ...]]
UNITS LENGTH <um|nm|m> GRID <grid>
```

`PEX25D` must be the first record of the top-level file. `UNITS` must precede numeric records, including numbers brought in by `INCLUDE`. After these, records can appear in any order and may reference profiles declared later.

A reader accepts minor versions with the same supported major version. It skips an unknown top-level record with a warning; it rejects an unsupported major version. An unknown clause in a known record is an error. A clause keyword such as `WRAPS` or `THICKNESS_ON_FIELD` at the beginning of a record is also an error, since it may indicate a missing continuation backslash. An optional version suffix consists of letters, digits, dots, or hyphens and does not affect compatibility; writers should preserve it. The `REQUIRES CAPABILITIES` clause lists capability names separated by commas, with optional whitespace around each comma. Empty entries and repeated names are errors. Required capabilities are checked independently: a consumer must reject every capability it does not implement. A required capability cannot be replaced by a `META` entry.

Coordinates, Z offsets, thicknesses, and margins use the declared length unit and must be multiples of `GRID`. Decimal and scientific notation are accepted, including `.5`, `5.`, and `1e-3`. There are no per-value unit overrides or thousands separators. To check a length, divide by `GRID`, round to the nearest integer, and require the quotient to be within 10^{-6} of that integer. A value outside that tolerance is an error, not permission to snap the input geometry.

For `UNITS LENGTH um GRID 0.0001`, a coordinate of `0.3262` is stored as **3262 grid units**. Protobuf represents the grid itself as an exact rational. It stores coordinates in signed 64-bit integers, not floating-point micrometres. Derived vertical positions therefore remain exact. Lateral offsets of non-Manhattan edges can fall off-grid; the specification requires rounding those *derived offsets* to the nearest grid point.

Permittivity and resistance parameters are material values and are **not** grid-quantized. The PEX25D grid and the source layout database unit (DBU) are distinct: `GRID 0.0001` and a layout DBU of `0.001` µm differ by a factor of ten.

11.1.5.2 Metadata and includes

```
META <key> <value>
INCLUDE <path>
```

Metadata records describe provenance and do not change geometry or material interpretation. A key may occur only once across the entire include tree. Unknown metadata keys are silently accepted. Defined keys are `technology`, `process_corner`, `source_cell`, `source_layout_hash` (for example, `sha256:<digest>`), `source_dbu`, and `generator`. Prefix private keys with a vendor tag such as `x-kpex-`. If `source_dbu` is supplied, the grid must divide it exactly.

An included file contains records without its own `PEX25D` or `UNITS` header. Relative paths are resolved against the including file's directory; nested includes are allowed, cycles are errors. For input from stdin, relative includes resolve against the current working directory.

For example, keep the header in `design.pex25d`, put profiles and material values in `stack-typical.inc`, and put conductor shapes in `geometry.inc`:

```
PEX25D 1.0-rc1
UNITS LENGTH um GRID 0.0001
META technology example
META process_corner typical
INCLUDE stack-typical.inc
INCLUDE geometry.inc
```

Both included files use the top-level length unit and grid. Changing the stack include selects another physical corner; it does not supply a new calibration table to an analytical engine.

11.1.5.3 Ground plane, metals, and vias

```
GROUND_PLANE <name> [NET <reference-net>] \
    Z_OFFSETS <zlow> <zhigh>
METAL <name> Z_OFFSETS <zlow> <zhigh>
VIA <name> CONNECTS <below> <above>
```

The text format requires exactly one ground plane. Before openings are subtracted, it occupies an infinite XY slab at 0 V. Ground-plane and metal Z ranges are absolute, with `zlow < zhigh`. Metals become solids only where conductor shapes are drawn. An explicit `NET` separates the reference net from the geometric profile name, as described in Section 11.1.5.4.

A via fills the vertical gap between its ordered endpoints:

$$z_{\text{low,via}} = z_{\text{high,below}}, \qquad z_{\text{high,via}} = z_{\text{low,above}}.$$

Endpoints may name profiles with Z extents, including the ground plane for substrate or well taps. Contacts landing on different layers need distinct via profiles; contacts with equal heights but different resistances may also need distinct profiles.

Ground-plane, metal, via, dielectric, and background names share a unique **profile namespace**. Conductor shortnames have a separate namespace, and terminal names another. Shape and terminal `LAYER` references must select a metal or via, never the ground plane or a dielectric.

A metal profile without shapes is legal. Such an **anchor** can locate the bottom of a contact at an excluded device layer without introducing that layer's solid. It has no resistance requirement and does not enlarge the computational domain or overlap the ground plane merely because its Z range lies inside it.

11.1.5.4 Ground openings and electrical ownership

A contact can end exactly at the top of the ground slab without belonging to its reference net. If the exporter fills the entire slab below that contact, their shared face creates an unintended electrical connection. An opening removes ground material under the contact and leaves its height and electrical identity unchanged.

This is a physical geometry operation. Grounding an extracted capacitance netlist, described in Section 7.4, changes an electrical approximation after extraction; it does not repair the field solver's geometry.

```
GROUND_PLANE <profile> [NET <reference-net>] \
    Z_OFFSETS <zlow> <zhigh>
GROUND_CUTOUT <profile> \
    OUTER <x1> <y1> <x2> <y2> ... \
    [HOLE <x1> <y1> <x2> <y2> ...]
```

The profile name identifies the geometric slab and remains the target of `WRAPS` and `GROUND_CUTOUT`. `NET` identifies the electrical reference. For example, `GROUND_PLANE subs NET VSS` keeps the profile `subs` while connecting the retained ground material to `VSS`. Without `NET`, its electrical name is the profile name. A ground opening requires an explicit reference net, and that net cannot be the reserved name `FLOATING`.

All opening polygons are unioned and removed through the original slab's full thickness. The remaining material stays at the original Z coordinates. A polygon's `HOLE` keeps a reference-material island inside that opening. The island remains on the declared reference net, rather than becoming a floating conductor. Another opening can still remove that island. Overlapping openings are subtracted once. A coating rooted on the ground follows the retained reference geometry and its own growth rules.

11.1.5.4.1 Contact at the reference surface

Download ground_opening.pex25d. This small illustrative scene places a 1×1 µm contact at `z = 0`, the top of the reference slab. The opening extends from -1 to 2 µm in both X and Y, leaving 1 µm of lateral clearance around the contact. The finite domain bounds the solver model; the opening does not move either body vertically.

```
PEX25D 1.0-rc1
UNITS LENGTH um GRID 0.001
GROUND_PLANE subs NET VSS Z_OFFSETS -0.1 0
GROUND_CUTOUT subs OUTER -1 -1 2 -1 2 2 -1 2
METAL contact Z_OFFSETS 0 0.2
DIELECTRIC_BACKGROUND air PERMITTIVITY 1
CONDUCTOR pin signal
BOX CONDUCTOR pin LAYER contact LL 0 0 UR 1 1
DOMAIN_BOX LL -3 -3 -1 UR 3 3 1

```

Resolve and export the same scene:

```
pex25d validate --strict ground_opening.pex25d
pex25d resolve ground_opening.pex25d \
    -o ground_opening.pex25d.scene.pb
pex25d export ground_opening.pex25d.scene.pb \
    --to fastercap --out-dir ground_opening
```

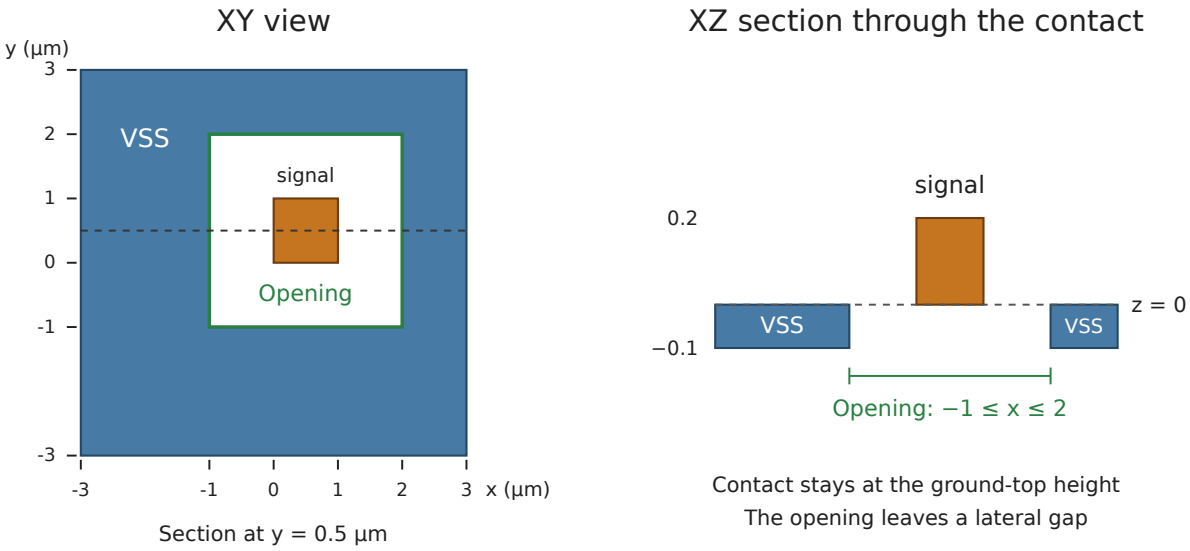


Figure 85: The opening removes reference material through its full thickness. The signal contact keeps its original position and net. Dimensions are in µm.

The exporter produces two electrical bodies: reference net `VSS` and signal net `signal`. The opening prevents a conducting path between them while preserving their capacitive coupling. A contact tied explicitly to the reference net is permitted to meet the retained reference material.

11.1.5.4.2 Floating body with the same label

The shortname of a floating conductor is a label for that independent body. It does not connect the body to a net with the same spelling. Download ground_opening_floating.pex25d, which changes the example's conductor to:

```
CONDUCTOR VSS FLOATING
BOX CONDUCTOR VSS LAYER contact LL 0 0 UR 1 1
```

The reference still belongs to net `VSS`. The floating body's potential remains independent, as Figure 86 shows.

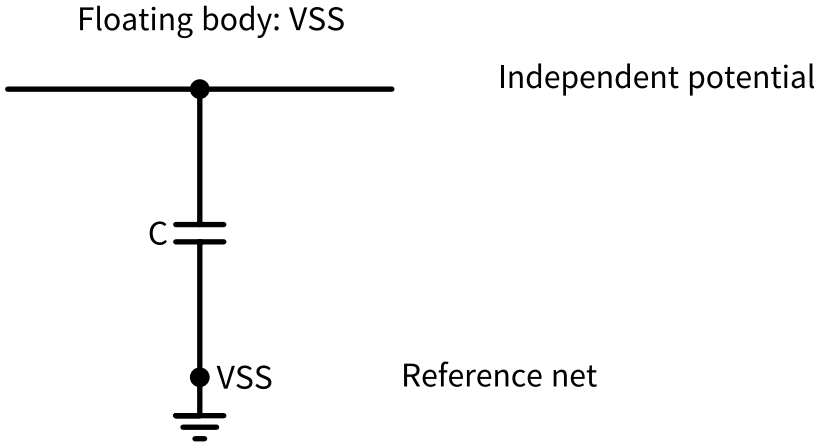


Figure 86: A floating body labeled `VSS` couples capacitively to reference net `VSS`. Equal spelling does not make an electrical connection.

```
pex25d export ground_opening_floating.pex25d \
    --to fastercap --out-dir ground_opening_floating
```

Solvers with one flat label namespace receive distinct names: `VSS` for the reference net and `FLOATING:VSS` for this floating body. A further collision gets a suffix. When labels need disambiguation, the export also writes a `*_body_names.json` mapping that distinguishes named nets from floating bodies. Read that mapping when associating solver results with the source scene.

11.1.5.4.3 Openings generated from layout

`kpex pex25d` uses omitted diffusion footprints to derive the ground openings. It expands their union by 1 µm, rounded down to the layout's database unit, and records the numerical exclusion policy in the output metadata. This clearance follows the direct field-solver preparation and is not a manufactured dimension of the process.

The generated scene preserves:

- The reference net from the extracted circuit
- The original ground and contact Z coordinates
- Each contact's electrical net

The opening remains part of the resolved scene, so a downstream exporter can preserve it without access to the original layout or PDK. No separate command-line switch is needed.

11.1.5.4.4 Limits

A different-net or floating conductor must not touch retained reference material. Both `pex25d validate --strict` and export preflight report `PEX25D-E0310` when it would, including an exact boundary touch. The ordinary `pex25d validate` command can accept an intermediate scene with such contact; successful structural validation alone does not establish that it is safe to export. Cutout geometry that the backend cannot represent, or unavailable geometry support, reports `PEX25D-E0311` instead of silently altering the opening.

An exporter that cannot represent ground openings or separate electrical identities rejects the scene before writing solver input. Update that exporter before using these scenes; omitting an opening would change the electrical problem. The ground slab is still an ideal reference and does not model substrate resistance.

11.1.5.5 Dielectrics and material precedence

```
DIELECTRIC_SIMPLE <name> WRAPS <profile> PERMITTIVITY <k> \
    BETWEEN <below> <above>
DIELECTRIC_CONFORMAL <name> WRAPS <profile> PERMITTIVITY <k> \
    THICKNESS_OVER_WRAPPED <top> \
    THICKNESS_BESIDE_WRAPPED <side> \
    THICKNESS_ON_FIELD <field>
DIELECTRIC_BACKGROUND <name> PERMITTIVITY <k>
```

`PERMITTIVITY` is relative, isotropic, and frequency-independent; vacuum has a value of 1.0. Exactly one background is required. It fills whatever no conductor, ground plane, or dielectric occupies.

`DIELECTRIC_SIMPLE` occupies a band across the whole XY plane, from the **bottom face of `below` to the bottom face of `above`**. This differs from `VIA CONNECTS`, which starts at the lower profile's *top*.

BETWEEN determines the band's geometry; WRAPS determines its precedence. A simple fill must wrap the outermost film anchored on its lower object, or the lower object itself if it has no film.

DIELECTRIC_CONFORMAL grows from the surface of the profile it wraps:

Thickness	Meaning
THICKNESS_OVER_WRAPPED	Growth above its top face; zero is flush with that face
THICKNESS_BESIDE_WRAPPED	Growth beside its side faces
THICKNESS_ON_FIELD	Total field extent measured upward from the bottom of the root conductor or ground plane; zero leaves no field film

Top and side thicknesses are incremental along a WRAPS chain. Field extents are cumulative from the root bottom and are not added along the chain; Section 3.9.12.6 illustrates the difference. If one film extends 0.006 µm beside a metal and the next extends 0.0431 µm beside that film, the outer extent is 0.0491 µm beyond the metal. A sidewall film is expressed by a conformal record with zero field thickness; there is no separate DIELECTRIC_SIDEWALL record.

Profiles describe grown solids containing the wrapped object, rather than pre-subtracted shells. Material occupancy determines the visible material:

1. Conductors and the ground plane take precedence over dielectrics.
2. Among dielectrics claiming a point, the smallest WRAPS depth wins. A film on a metal, via, or ground plane has depth 1; wrapping another dielectric adds one.
3. The background fills unclaimed space.

WRAPS references must be acyclic. Overlapping dielectrics at equal depth are ambiguous and invalid; changing record order cannot fix them. Re-anchor the fill or film to express the intended precedence. These rules make stack order in the file a matter of readability rather than material interpretation.

11.1.5.6 Conductors and XY shapes

```
CONDUCTOR <shortname> <net>
BOX CONDUCTOR <shortname> LAYER <profile> LL <x> <y> UR <x> <y>
POLYGON CONDUCTOR <shortname> LAYER <profile> \
    OUTER <x1> <y1> <x2> <y2> ... [HOLE <x1> <y1> <x2> <y2> ...]
```

Each shape is extruded over its layer's Z extent. A conductor's shapes are unioned and may be disconnected. Different conductor bodies must not overlap in volume. The meaning of a shared net depends on the analysis:

Analysis	Meaning of conductors sharing a net
Capacitance	All are constrained to one potential; results are combined per net
Resistance	Each conductor forms a separate network between its own terminals; external connections belong to the enclosing flow

The exact, case-sensitive net name FLOATING is reserved for unconnected shielding bodies such as dummy fill. Each floating conductor has its own unknown potential, is reported under its shortname,

and must not have terminals. Floating bodies are not shorted together just because they all use that reserved net name.

Boxes use lower-left and upper-right corners. Polygon rings are implicitly closed, so do not repeat the first vertex. An outer ring may be concave; outer and hole rings must be simple. Holes must lie strictly inside their outer ring and may not touch or intersect another ring. Winding direction is immaterial. Use separate polygon records for disconnected regions or an island inside a hole.

Every via cut is an explicit shape. There is no via-array record, and readers do not infer cuts from spacing or enclosure rules. This preserves dielectric gaps between cuts and makes `PER_CUT` resistance unambiguous.

11.1.5.7 Capacitance Accounting

A result exclusion leaves the conductor in the physical problem. It still shields other conductors and keeps its original electrical potential. The policy selects which mutual-capacitance contributions enter the reported network; it does not remove metal, open a via, or create another `NET`.

A file with accounting records must declare the required capability in its first record:

```
PEX25D 1.0-rc2 REQUIRES CAPABILITIES CAPACITANCE_ACCOUNTING
UNITS LENGTH um GRID 0.0001
```

The `rc2` suffix identifies the format revision used by this writer. The mandatory capability declaration carries the semantics: omitting it is an error even when the suffix is `rc2`. Unknown required capabilities are errors, including unknown numeric enum values in binary protobuf input. Geometry-only files need no capability declaration.

11.1.5.7.1 Masks and Exclusion Policies

```
CAP_REGION <name> LAYER <layer> OUTER <xy...> [HOLE <xy...>]
CAP_EXCLUDE <name> WITHIN <region> ALL_LAYER_PAIRS
CAP_EXCLUDE <name> WITHIN <region> LAYER_PAIR <layer-a> <layer-b>
CAP_EXCLUDE_SUBSTRATE <net>
```

`CAP_REGION` is a polygon mask on one conductor layer. Repeating the same region and layer adds polygons to their union. Repeating the region with another layer gives that region a mask on the other layer. Holes remain outside the mask. A missing mask on a layer selects nothing there; it does not inherit another layer's footprint.

`CAP_EXCLUDE` names a result policy independently of its region:

- `ALL_LAYER_PAIRS` excludes coupling between parts inside the same named region, across all their layers
- `LAYER_PAIR` limits that exclusion to the unordered pair of named layers; both conductor layers need masks in the region
- A physical ground profile may be one layer of `LAYER_PAIR`; only the non-ground layer needs a mask
- The ground body is never locally partitioned; `ALL_LAYER_PAIRS` also excludes member-part coupling to that reference body

`CAP_EXCLUDE_SUBSTRATE` selects the physical reference body. It does not exclude coupling to every routed conductor sharing the reference's electrical net name. Net names must name a connected net; `FLOATING` is not a connected net. The reference net itself may be listed.

Overlapping regions retain all their memberships. A policy applies when its own membership and layer conditions match; another overlapping region does not broaden those conditions. A mask touching only a conductor edge selects no area.

These are explicit result policies. They do not infer the intrinsic capacitance owned by a compact device model from a terminal mask or from two net names. In particular, they do not make field-solver extraction with retained metal-capacitor models safe. `kpex extract` rejects that combination before geometry generation, as described in Section 2.2.

11.1.5.7.2 Run the Accounting Example

Download two_nets_accounting.pex25d. It has the same physical geometry, materials, nets and terminals as Section 11.1.4. Its header declares `CAPACITANCE_ACCOUNTING`, and these records add the result policy:

```
CAP_REGION window1 LAYER met1 \
    OUTER -0.1 -0.1 2.6 -0.1 2.6 1.1 -0.1 1.1
CAP_REGION window1 LAYER met2 \
    OUTER -0.1 -0.1 2.6 -0.1 2.6 1.1 -0.1 1.1
CAP_EXCLUDE internal12 WITHIN window1 LAYER_PAIR met1 met2
CAP_EXCLUDE_SUBSTRATE neta
```

Validate and resolve the downloaded file:

```
pex25d validate two_nets_accounting.pex25d --strict --werror
pex25d resolve two_nets_accounting.pex25d \
  -o two_nets_accounting.pex25d.scene.textpb --strict
```

The example validates without diagnostics. The resolved protobuf text contains these parts; areas below come from the actual resolved polygons:

Table 77: Accounting parts from the resolved two-net example

Part ID	Layer	NET	Region membership	Area (µm²)
1:A:met1:0	met1	neta	window1	0.50
1:A:via1:0	via1	neta	None	0.04
1:A:met2:0	met2	neta	window1	0.50
1:B:met1:0	met1	netb	window1	0.60
1:B:met1:1	met1	netb	None	0.40

`A` keeps one electrical potential across its three parts. The `via1` cut has no membership because the region has no mask on that layer. `B` is split at $x = 2.6\,\mu$m, but both parts still belong to `netb`.

The `internal12` policy excludes the contribution between `A` on `met2` and the part of `B` on `met1` inside `window1`. It retains coupling to the outside part of `B`, and retains `met1` ↔ `met1` coupling. The separate substrate policy excludes `neta`'s coupling to the physical reference body. None of these records changes the geometry that shields the remaining interactions.

11.1.5.7.3 Parts and Common Electrical Potential

The file and scene both declare mandatory capabilities through the general `RequiredCapability` enum. The `CAPACITANCE_ACCOUNTING` capability adds a typed `capacitance_accounting` policy. In a resolved scene, `ConductorLayerRegion` also records each nonempty conductor part when masks are present:

Table 78: Required semantics and accounting identities

Protobuf field	Tag	Meaning
`required_capabilities`	50	Required semantics on both `PEX25DFile` and `PEX25DScene`
`capacitance_accounting`	60	Typed masks, exclusions and substrate-net selection on both messages
`accounting_part_id`	10	Nonempty, artifact-unique accounting identity
`accounting_regions`	20	Names of regions selecting that part on its layer

`accounting_part_id` is optional for geometry-only scenes and substrate-only policies. With accounting masks, every nonempty conductor region needs this opaque artifact-local identifier. It is not a new net name. A consumer must not decode electrical ownership from its spelling. The enclosing conductor's `net` and `floating` fields retain that meaning. Parts have disjoint area within each conductor and layer, and their union equals the original geometry.

When a solver reports separate charge groups, all parts of one connected net must retain its common potential. For Maxwell coefficients, aggregation follows linearity:

$$C_{IJ} = \sum_{i \in I} \sum_{j \in J} C_{ij},$$

where i, j identify accounting parts and I, J identify their electrical bodies. The consumer applies the explicit exclusion policy while preserving the reference-body convention. Keeping only one coefficient per net pair before applying the policy would lose the distinction between inside and outside contributions.

Accepting these scenes requires `SceneCapability.CAPACITANCE_ACCOUNTING`. The exporter returns that member from its `supported_capabilities()` class method; see Section 11.2.4.2. Successful protobuf parsing is not that declaration. Older binary consumers can ignore unknown fields, so they must gain the required-capability check and result-accounting support before receiving policy-bearing scenes. Regenerating bindings alone is insufficient. The guarded text header also makes older text readers reject the unfamiliar clause instead of silently discarding policy records.

11.1.5.7.4 Exact Grid and Via Ownership

The masks classify geometry on the scene's exact grid. An angled edge can cross a mask boundary between grid points even when all original vertices lie on the grid. Rounding that crossing can move an edge, lose area or leave a sliver; accounting resolution rejects it with `PEX25D-E0285` and asks for a finer `GRID`.

Authored files keep their declared grid. The resolver does not silently rewrite it. In the integrated layout-to-field-solver flow, KPEX can instead rebuild from the original layout on a finer rational grid. It derives the required integer refinement from the exact intersection denominators and retries without

moving physical coordinates. The binary scene preserves that rational grid; a non-terminating decimal grid cannot be written in the decimal text syntax.

Via ownership is decided on the **original layout grid** before that refinement. The layout producer classifies whole cuts and records their complete footprints. Refining the scene grid therefore does not move a cut across an exclusion boundary through a different rounded bounding-box centre. An authored mask must cover a via cut completely or miss it completely; a mask that cuts through a via is an error.

Resolving masks uses KLayout polygon operations, loaded only for that geometric step. Ordinary format I/O and structural policy checks remain independent of KLayout. The current polygon backend requires XY coordinates strictly between the signed 32-bit endpoints, even though protobuf stores signed 64-bit coordinates. Grid refinement that exceeds those native limits fails with a domain diagnostic.

11.1.5.7.5 Accounting Diagnostics

Table 79: Accounting diagnostics before solver execution

Code	Problem
`PEX25D-E0280`	Unknown, repeated or missing required capability
`PEX25D-E0281`	Empty policy or mask, or duplicate region, layer or exclusion identity
`PEX25D-E0282`	Unknown region, layer or connected net, or membership without a layer mask
`PEX25D-E0283`	Unknown exclusion kind or incompatible kind-specific fields
`PEX25D-E0284`	Missing, duplicate or empty accounting part, or parts without a policy
`PEX25D-E0285`	Geometry cannot be resolved exactly, a via is clipped, native bounds are exceeded, or the geometry backend is unavailable

Malformed mask rings use the polygon diagnostics as well. Their wellformedness is checked before partitioning, without waiting for `--strict`. An adapter must also reject a serialized scene whose part memberships disagree with its masks or whose parts overlap.

11.1.5.8 Resistance parameters

```
RESISTANCE TEMPERATURE <celsius>
RESISTANCE METAL <profile> SHEET <ohm_per_square> [TC1 <value>] [TC2 <value>]
RESISTANCE VIA <profile> PER_CUT <ohm> [TC1 <value>] [TC2 <value>]
```

Resistance records are optional. There may be at most one reference temperature, one sheet resistance per metal profile, and one per-cut resistance per via profile. `SHEET` is in ohms per square, and `PER_CUT` in ohms per cut, independently of the length unit. KPEX technology data stores resistance in milliohms; a producer converting that data divides by 1000.

Missing resistance means **unspecified**, not zero. A resistance consumer must reject a profile carrying geometry if it has no resistance value. Unused anchor profiles are exempt. A capacitance consumer can ignore resistance parameters.

Sheet resistance is a calibrated electrical property. Do not reconstruct it from the drawn metal thickness or invert it to infer bulk resistivity. Two geometrically equal contacts can likewise have different per-cut resistances, for example when they land on different diffusion types.

The optional temperature coefficients have units of K^{-1} and K^{-2}:

$$R(T) = R_{\text{ref}}\left[1 + TC1\,(T - T_{\text{ref}}) + TC2\,(T - T_{\text{ref}})^2\right].$$

The reader and resolver carry these values through; the consumer applies them at its operating temperature. An absent reference temperature does not imply 25 °C. The enclosing flow must supply it if needed.

11.1.5.9 Terminals and compact-device boundaries

```
TERMINAL <name> CONDUCTOR <shortname> LAYER <profile> [KIND <kind>] \
    LL <x> <y> UR <x> <y>
```

A terminal is the intersection of its rectangular region with the selected conductor's geometry on one metal or via profile. It must select a non-empty region. Several disjoint pieces may form **one electrical node**, but a terminal may not partially select a via cut: each selected cut must be covered completely.

`KIND PIN` denotes a port or probe point. `KIND DEVICE_TERMINAL` marks the boundary to a compact device model. Omitting `KIND` leaves the role unspecified. Names are unique among terminals and become network node names.

A metal terminal makes its selected region equipotential over the layer's thickness; a 3D resistance adapter may choose a drive face. A via terminal is a node at the **lower face** of the selected cuts. It does not short through the via, so the contact resistance above that node remains part of the extracted network. A conductor with fewer than two terminals contributes no resistance elements.

> **!** Terminal size changes the resistance problem
>
> Everything selected by a terminal is shorted together. A marker covering an entire wire or gate can therefore remove the resistance you intend to extract. Size the region to the pin or device boundary it represents, rather than copying a large enclosing shape.

Geometry owned by a compact model is excluded from the interconnect description to avoid counting its parasitics twice. For contacted diffusion whose contacts belong to the interconnect, the device boundary lies at the contacts' lower faces. Diffusion and its internal spreading resistance remain in the device model; an empty anchor profile locates the contacts without drawing diffusion solids. When the device model also includes the contacts, the boundary instead lies on the routing metal above them.

The lower face selected by `DEVICE_TERMINAL` is an **open boundary of the extracted region**, for both capacitance and resistance. This prevents a device contact coincident with the ground-plane surface from becoming a short to 0 V or adding device junction capacitance. Only the marked portion is opened. A substrate tap without a device terminal retains ordinary electrical contact with the ground plane. These are specification requirements for consumers; the current FasterCap adapter does not yet implement device-terminal open faces.

11.1.5.9.1 KLayout-generated terminals

The layout producer derives `PIN` records from labels inside layout pin markers. Each pin region extends one layout DBU from the label in every direction, preserving wire resistance even when the pin marker itself covers a long wire.

Device terminals use LVS terminal geometry on existing conductive profiles. Several disjoint pieces can share one node when their bounding box selects exactly those pieces; an abutting boundary

selects a strip one DBU wide inside the interconnect. Generation rejects a box that would select additional interconnect or a node that spans multiple profiles. Names use `pin:<label>` and `device:<instance>:<terminal>`, with percent encoding for syntax-sensitive characters and numeric suffixes for uniqueness.

Source and drain terminals on diffusion are generated at the lower faces of their contact cuts. For example, `sky130A` MOS terminals use `licon_nsd_con` or `licon_psd_con`; `ihp-sg13g2` uses `cont_nsd_con` or `cont_psd_con`. The boundary follows the model contract: `gf180mcuD` diffusion-resistor terminals are on `Metal1`, because those resistor models include the contacts.

The resolver clips Manhattan polygons exactly, including holes and disconnected intersections. Partial clipping of non-Manhattan polygons remains unsupported. The exporter can preserve only the device-terminal pieces supplied by LVS: the remaining multi-polygon terminal limitation in `KLayout` can omit pieces before export, including with `0.30.13`.

11.1.5.10 Computational domain

```
DOMAIN_MARGIN X <xmargin> Y <ymargin> Z <zmargin>
DOMAIN_BOX LL <x> <y> <z> UR <x> <y> <z>
```

These optional records help adapters select a finite computational region; they do not prescribe boundary conditions. `DOMAIN_MARGIN` expands the finite geometry bounds in both X and Y directions and only upward in Z. The ground-plane top forms the lower boundary. An adapter may increase the requested clearance but must not reduce it.

Finite geometry includes conductor shapes and laterally bounded conformal films, including their lateral growth. The ground plane, background, simple dielectric bands, field-covering films, and unused anchor profiles do not enlarge those bounds.

`DOMAIN_BOX` specifies the box directly and takes precedence over `DOMAIN_MARGIN` if both occur, with a warning. Finite non-ground-plane geometry must fit inside; touching a box face is allowed. The lower Z face normally coincides with the ground-plane top. Without either record, domain choice belongs to the adapter. The current FasterCap/FastCap2 exporter uses the resolved box's XY extent to bound laterally infinite materials; it does not turn the box into a prescribed six-face boundary condition.

11.1.6 Command-line tools

The `klayout-pex` package supplies `kpex` and the separate `pex25d` entry point. Check command availability with `--help`; older installed releases may not have these commands.

```
kpex pex25d --help
pex25d --help
pex25d validate --help
pex25d export --help
```

Reading, converting and summarizing artifacts do not require KLayout, a layout, LVS or a running solver. Structural accounting checks are also independent of KLayout. Resolution of accounting masks requires the KLayout geometry backend; `validate` on an unresolved file performs that resolution too. Geometry-only stack resolution retains its format-only path. Generating from a layout and exporting through the current meshing backend also require KLayout. This is a runtime distinction; the distributed `klayout-pex` package still declares KLayout as a dependency.

11.1.6.1 Generate from a layout: `kpex pex25d`

This subcommand runs layout connectivity extraction through LVS, combines it with the selected PDK's stack and resistance data, and writes artifacts in the PEX25D format before solver execution. The layout example in Section 2.4 shows both outputs. At least one output must be requested, and the two paths must differ.

Option	Use
`--out-file PATH`	Write an unresolved file; `.pex25d`, `.pex25d.pb`, or `.pex25d.textpb` selects its encoding
`--out-scene PATH`	Write a resolved scene; use `.pex25d.scene.pb` or `.pex25d.scene.textpb`
`--format FORMAT`	Select `auto`, `text`, `pb`, or `textpb`; one setting applies to both requested outputs
`--grid UM`	Grid in µm; default `0.0001`, must represent stack dimensions and divide the layout DBU
`--domain-margin UM`	Emit equal X, Y, and upper-Z margins; default is no domain record
`--validate BOOL`	Run reference validation; `y` (default) or `n`
`--comments BOOL`	Add syntax hints to text output; `y` or `n` (default)
`--with-source-refs BOOL`	Retain per-record provenance in protobuf output; `y` or `n` (default)

Layout, PDK, top-cell, blackboxing, and material selection options are listed by `kpex pex25d --help`. The artifact's kind is selected by the output option, so requesting `--format text` together with `--out-scene` is invalid. Filenames ending in `.gz` produce compressed output.

`--strict`, `--werror`, `--diagnostics`, and `--out-diagnostics` are also available, with the meanings described in Section 11.1.6.2.

Generation can leave artifacts on disk even when validation reports errors. Always inspect the exit status and diagnostics before using them as solver input. The bundled `sky130A` stack uses separate `capild3` and `capild4` profiles for the two MIM levels, with the corresponding dielectric anchors. The generation example above validates with the complete dielectric stack; no dielectric exclusion is needed. It reports `PEX25D-W0265` because `licon_ptap_con` carries geometry without a specified resistance. A capacitance consumer can ignore that resistance data, while a resistance consumer needs the missing value.

11.1.6.2 Validate and inspect

Run these commands in the directory containing the downloaded example:

```
pex25d validate two_nets.pex25d
pex25d validate two_nets.pex25d --strict
pex25d show two_nets.pex25d
pex25d show two_nets.pex25d --section layers --section terminals
```

Normal validation checks syntax, units, references, wrap anchoring and depth conflicts, resistance records, terminal selection, and required accounting semantics. Accounting masks are checked before geometric partitioning; invalid required capabilities and policies are errors in binary scenes too. `--strict` adds geometric checks for boxes and rings, hole containment, and conductor volume overlap. `show` summarizes the artifact as supplied; resolve it first when you want to inspect derived layer heights.

`--section` is repeatable and accepts `header`, `meta`, `layers`, `dielectrics`, `conductors`, `terminals`, `resistance`, `domain`, and `all`. Human-readable summaries and diagnostic wording are not stable APIs.

For automated checks of a custom PEX25D writer, save a structured report and use the exit status:

```
pex25d validate two_nets.pex25d --strict --werror \
  --diagnostics json --out-diagnostics validation.json
pex25d validate two_nets.pex25d --strict \
  --diagnostics pb --out-diagnostics validation.pb
```

JSON and binary protobuf reports carry stable `PEX25D-Ennnn` codes, severity, validation tier, and source positions where available. Use an explicit report path to keep the report separate from console logging. `--werror` makes warnings fail the run; without it, warnings alone do not fail validation.

Exit status	Meaning
0	Success
1	Validation or resolution errors, or warnings with `--werror`
2	Invalid command line, I/O failure, missing dependency, or export failure
3	An operation reports that it is not implemented
4	An importer or exporter plugin failed unexpectedly (traceback with `--log-level debug`)

The validator is a reference check, not a guarantee that every adapter implements every specification rule. For example, the current depth-tie check detects competing profiles on the same wrap root; it is not a complete geometric intersection check for films on different roots. Terminal-boundary support must also be checked at the adapter level.

11.1.6.3 Convert encodings and resolve a scene

```
pex25d convert two_nets.pex25d -o two_nets.pex25d.pb
pex25d convert two_nets.pex25d.pb -o two_nets.pex25d.textpb
pex25d convert two_nets.pex25d.pb -o two_nets-copy.pex25d --comments
pex25d convert two_nets.pex25d -o two_nets.pex25d.pb.gz

pex25d resolve two_nets.pex25d -o two_nets.pex25d.scene.pb --strict
pex25d convert two_nets.pex25d.scene.pb -o two_nets.pex25d.scene.textpb
pex25d validate two_nets.pex25d.scene.pb --strict
pex25d show two_nets.pex25d.scene.pb --section layers --section domain
```

`convert` changes encoding within one kind; it does not validate the complete physical problem. `resolve` computes the scene and accepts `--strict`, `--werror`, and the diagnostics options.

`export` resolves file input, then runs the normal reference validation on the scene before invoking the exporter. The same check applies to an already-resolved scene. Validation errors prevent export and produce CLI exit code `1`. Warnings do not prevent the exporter from writing files; `--werror` can therefore make the command fail after files were written. For stricter acceptance before any export, run `pex25d validate --strict --werror` first.

Use `--with-source-refs` on text conversion or resolution when you want the protobuf artifact to retain locations in an include tree. Diagnostics produced while reading text have source positions even without that flag. `--comments` adds syntax hints when writing PEX25D text; unlike the `kpex` option, it is a flag and takes no `y` argument.

11.1.6.4 Import from another tool

```
pex25d importers
pex25d import design.uf --from openrcx-uf -o design.pex25d
```

An importer translates another tool's scene description into an unresolved `PEX25DFile`. `pex25d import` validates the result and writes it even when it is invalid, so that it can be reviewed; the exit status tells whether it is valid. `--supporting NAME=PATH` passes further inputs the importer asks for, e.g. a process stack, and `--option NAME=VALUE` passes importer options. The built-in `openrcx-uf` importer (OpenRCX Universal Pattern Format) is a stub for now and exits with status `3`.

11.1.6.5 Export native solver input

```
pex25d exporters
pex25d export two_nets.pex25d.scene.pb \
  --to fastercap --out-dir solver/fastercap --prefix two_nets
pex25d export two_nets.pex25d \
  --to fastcap2 --out-dir solver/fastcap2 --prefix two_nets
pex25d export two_nets.pex25d --to stl --out-dir solver/stl
```

`--to` selects an exporter by its name, or by `package:name` when two installed packages provide the same name. `pex25d exporters` lists the installed ones. `klayout-pex` ships three exporters:

- `fastercap` and `fastcap2` write a `.lst` entry file and its referenced surface files; keep them together.
- `stl` writes the solids of the scene as STL files, one per dielectric and net, for viewing (e.g. in `MeshLab`).

File input is resolved automatically. Export generates meshes and input files, **without running the solver or producing a capacitance matrix**. The solver executable is needed when those files are subsequently solved. `kpex extract` is the separate integrated flow for running engines from a layout.

Exporter options are passed with `--option NAME=VALUE`, which can be repeated (since `klayout-pex` 0.6.3). VALUE is read as JSON where it parses (`10`, `1e-9`, `true`), otherwise as text. For FasterCap and FastCap2, `--stl`, `--field-margin`, `--delaunay-amax`, `--delaunay-b`, and `--geo-check` are shorthands for their options. The default lateral field margin is 8 µm when the scene has no explicit or margin-derived domain.

The built-in exporters write capacitance input. They do not extract a resistance network from `SHEET`, `PER_CUT`, or terminal records. Exporters for the FEM solvers Elmer and Palace are available as separate packages, klayout-pex-plugin-elmer and klayout-pex-plugin-palace.

11.1.6.6 Pipelines and explicit encodings

A path of `-` means stdin for input and stdout for output. Artifact output reserves stdout and redirects logging and diagnostics to stderr. Use explicit kinds and encodings in a binary pipeline:

```
pex25d resolve two_nets.pex25d -o - --out-kind scene --out-format pb \
  | pex25d show - --in-kind scene --in-format pb --section layers
```

For filenames without conventional suffixes, supply **both** `--in-kind file|scene` and `--in-format text|pb|textpb`, or the corresponding `--out-kind` and `--out-format`. These are options of the standalone tool; `kpex pex25d` uses its output options and `--format` instead. Stdin defaults to unresolved text, while `resolve` defaults to binary scene output on stdout. Gzip is inferred from filenames; use a shell gzip tool when compressing a stream.

11.1.7 Protobuf API: Python Preparation and C++ Consumers

The PEX25D schemas define the file and scene messages, exact units, geometry, dielectric and resistance records, terminals, domain, diagnostics, and source references. Generated bindings let another language read binary artifacts without implementing the text parser. An adapter should consume `PEX25DScene` and use the shared resolver for file input.

One import is enough: `from klayout_pex import pex25d` provides the verbs `read`, `write`, `resolve`, `validate`, `import_file`, `export` and `show`, the artifact and diagnostics types, every error the pipeline raises, and `pex25d.proto` for the generated messages (`pex25d.proto.PEX25DFile()`, or a module such as `pex25d.proto.dielectric` for its enums). Names listed in `pex25d.__all__` are the supported surface, while the submodules stay importable for the classes behind the verbs, e.g. `resolver.Resolver` and `validator.Validator`.

`read()` and `write()` take a path in any supported encoding and derive kind and encoding from the conventional suffixes (Section 11.1.3); `read_text()` and `write_text()` work on the bytes of the text format. `validate()` fills a diagnostics report and never raises for an invalid message; `resolve()` computes the scene and raises `ResolveError` when resolution cannot proceed. Reading, writing and structural policy validation do not import `klayout` at runtime. Resolving accounting masks needs its geometry backend, as described in Section 11.1.5.7.4. They are distributed within `klayout-pex`, whose normal installation includes `klayout>=0.30.12`, `protobuf` and the other declared dependencies.

This Python example loads and resolves the downloadable file, then requests native FasterCap input and STL solids through the exporter API:

```python
from klayout_pex import pex25d

report = pex25d.DiagnosticsReport(warnings_are_errors=True)
description = pex25d.read("two_nets.pex25d", report=report, with_source_refs=True)
scene = pex25d.resolve(description, report=report)
pex25d.validate(description, report=report, strict=True, scene=scene)
if report.exit_code:
    raise RuntimeError("PEX25D validation failed")

pex25d.write(scene, "two_nets.pex25d.scene.pb")
paths = pex25d.export(
    scene,
    target="fastercap",
    output_dir_path="solver/api",
    prefix="two_nets",
    options={"write_stl": True},
)
print(paths[0])
```

A C++ consumer can read the uncompressed binary scene written above using the bindings generated from `pex25d_scene.proto` and its imports. For example, this complete program loads a scene and reports its size; the extraction engine would then operate on that message:

```cpp
#include <fstream>
#include <iostream>
#include "kpex/pex25d/pex25d_scene.pb.h"

int main(int argc, char **argv) {
    if (argc != 2) {
        std::cerr << "Usage: read_scene FILE.pex25d.scene.pb\n";
        return 2;
    }
    std::ifstream input(argv[1], std::ios::binary);
    kpex::pex25d::PEX25DScene scene;
    if (!input || !scene.ParseFromIstream(&input)) {
        std::cerr << "Cannot read PEX25DScene\n";
        return 1;
    }
    std::cout << scene.layers_size() << " layers, "
              << scene.conductors_size() << " conductors\n";
    return 0;
}
```

Compile consumers with the generated `.pb.cc` files and the protobuf C++ runtime. A successful protobuf parse establishes that the binary message can be decoded; reference validation still belongs in the preparation flow, and an engine must check the format version and the features it supports. Keep the `.scene` kind explicit when exchanging files, since both unresolved files and resolved scenes can use binary protobuf encoding.

When consuming the scene directly, convert grid coordinates using `grid_numerator / grid_denominator` in the declared length unit. Check `scene.domain.origin` before using `scene.domain.box`: the domain message may exist only to carry geometry bounds. Preserve conductor net identities, floating-body independence, resistance units, and terminal boundary semantics when mapping the scene to a solver.

11.2 Plugin API

KPEX can be extended by plugins, which are separate Python packages. Once a plugin is installed (e.g. via pip3), KPEX finds it automatically, there is no need to change or re-install KPEX itself.

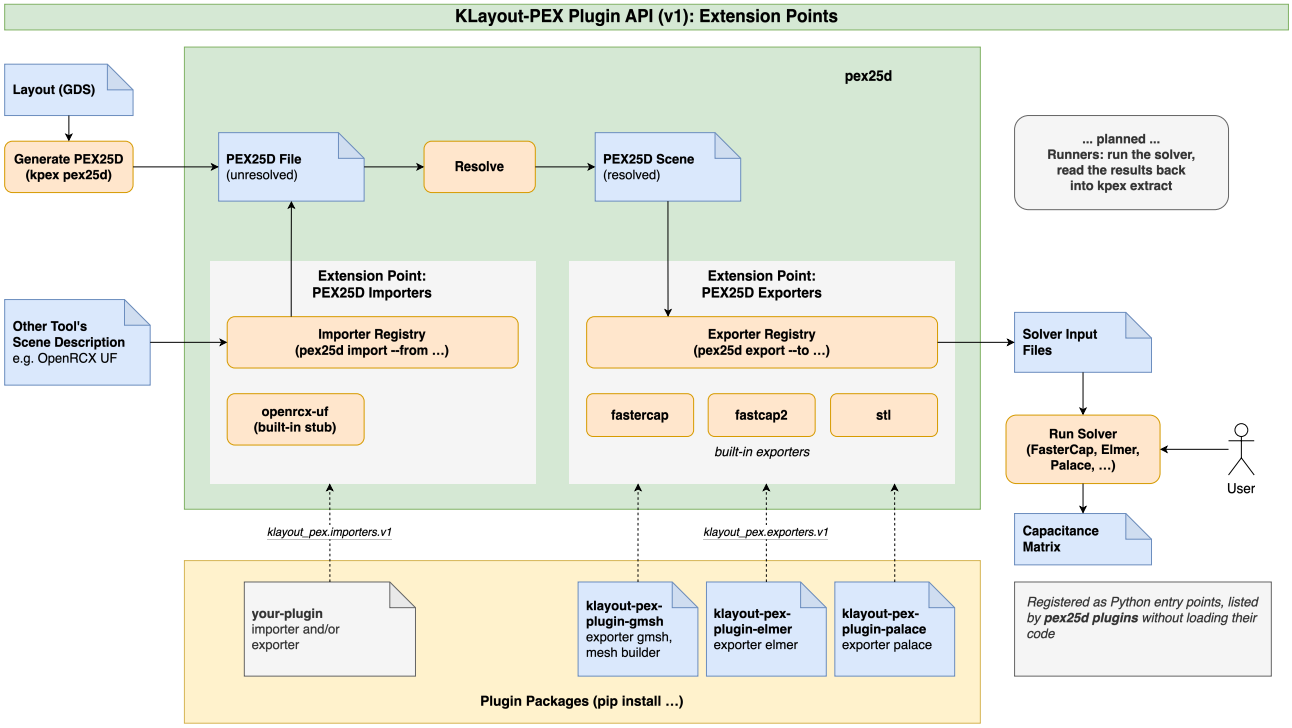


Figure 87: KPEX Plugin API: Extension Points

11.2.1 PEX25D Importers and Exporters

Currently, there are two extension points, both built around the PEX25D format (see Section 11.1):

Extension Point	PEX25D Importer	PEX25D Exporter
Description	Translate the scene description of another tool into a `PEX25DFile`	Write the input files of a solver from a resolved `PEX25DScene`
Input	Files of another tool, e.g. OpenRCX Universal Pattern Format	`PEX25DScene` protobuf
Entry Point Group	`klayout_pex.importers.v1`	`klayout_pex.exporters.v1`
Protocol	`PEX25DImporter`	`PEX25DSceneExporter`
CLI Invocation	`pex25d import --from NAME`	`pex25d export --to NAME`
Built-In Plugins	`openrcx-uf` (stub)	`fastercap`, `fastcap2`, `stl`
Maintained External Plugins	none yet	`elmer`, `palace`, `gmsh` (see Section 11.2.4.3)

Running the solver and reading its results back into `kpex extract` is not part of the plugin API yet, this is planned for a future version.

> **Caution**
>
> The plugin API `v1` is provisional: it is currently tested together with colleagues, and may still change until it is declared stable. Afterwards, breaking changes will get a new version `v2`, with

their own entry point groups.
Declare the range of `klayout-pex` versions supported by the plugin. The maintained `gmsh`, `elmer` and `palace` plugins at version `0.1.1` require `klayout-pex>=0.6.2,<1`, allowing later `0.x` releases. That package range does not make the provisional API stable; plugin maintainers still need to check compatibility when it changes.

11.2.2 Using Plugins

```
pip3 install --upgrade klayout-pex-plugin-palace

pex25d plugins     # overview: selector, package, version, description
pex25d exporters   # selectors only, one per line (for scripts)
pex25d importers
```

`pex25d plugins` only reads the package metadata, a plugin itself is loaded when it is used. A plugin that fails to load (e.g. because of a missing dependency) therefore does not affect the other ones.

Selecting a plugin:

- by its name, e.g. `--to palace`
- by `package:name`, if two packages provide the same name, e.g. `--to klayout-pex:stl`
- built-in plugins take precedence for plain names

Plugin specific options are passed with `--option NAME=VALUE` (since `klayout-pex` 0.6.3). The value is read as JSON if possible (`10`, `1e-9`, `true`), otherwise as text:

```
pex25d export cell.pex25d --to palace --out-dir deck \
  --option mesh_size_edge_um=0.2 --option order=2
```

Exit status	Meaning
1	Scene validation errors, or warnings with `--werror`
2	Unknown plugin, invalid option, or the plugin rejected its input
3	The plugin does not implement the requested feature yet
4	Unexpected plugin failure (`--log-level debug` shows the traceback)

11.2.3 Writing a Plugin

A plugin is a Python package, which registers a factory function under one of the entry point groups. The factory is called without arguments and returns an object implementing the protocol. The protocols are defined in `klayout_pex.plugin_api.v1`, deriving from them is optional.

For the plugins maintained by us, we use these conventions:

- GitHub repository and PyPI package: `klayout-pex-plugin-<name>`
- Python package: `klayout_pex_plugins.<name>`

  - `site-packages/klayout_pex_plugins` is a namespace package shared by all plugins
  - it must not contain an `__init__.py`, otherwise all other plugins become invisible

The registration in `pyproject.toml`:

```toml
[project]
name = "klayout-pex-plugin-example"
dependencies = ["klayout-pex>=0.6.3,<1"]

[project.entry-points."klayout_pex.exporters.v1"]
nets = "klayout_pex_plugins.example:create_exporter"
```

General rules for plugins:

- import heavy dependencies (e.g. a mesh generator) only when the plugin is used, not when its module is imported
- check the names and values of the options, raise `ExportError` / `ImporterError` for invalid options or unsupported input
- raise `ExporterUnavailable` / `ImporterUnavailable` when a dependency is missing, e.g. native binaries or libraries
- raise `NotImplementedError` for features that are not implemented yet
- do not modify the scene or the options passed in

11.2.4 PEX25D Exporters

An exporter writes the input files of a solver from a resolved `PEX25DScene` (see Section 11.1.2), it does not run the solver:

```python
class PEX25DSceneExporter(Protocol):
    @classmethod
    def supported_capabilities(cls) -> FrozenSet[SceneCapability]:
        return frozenset()

    def export(self,
               scene: PEX25DScene,
               *,
               output_dir_path: str,
               prefix: str = '',
               options: Optional[Mapping[str, Any]] = None) -> List[str]:
        ...
```

- `scene`: the resolved scene (`pex25d export` resolves a `PEX25DFile` first)
- `output_dir_path`: the directory for the files, the exporter creates it if necessary
- `prefix`: prefix for the file names, an empty prefix means the exporter's own default
- `options`: exporter specific options, e.g. from `--option`
- returns the paths of the files written, the main input file first

A minimal exporter, which writes the net of each conductor into a text file:

```python
import os
from typing import Any, List, Mapping, Optional

from klayout_pex.plugin_api.v1 import ExportError, PEX25DSceneExporter
```

```

class NetsExporter(PEX25DSceneExporter):
    def export(self, scene, *, output_dir_path: str, prefix: str = '',
               options: Optional[Mapping[str, Any]] = None) -> List[str]:
        if options:
            raise ExportError(f"unknown options: {', '.join(options)}")
        os.makedirs(output_dir_path, exist_ok=True)
        path = os.path.join(output_dir_path, f"{prefix}nets.txt")
        with open(path, 'w', encoding='utf-8') as file:
            for conductor in scene.conductors:
                file.write(f"{conductor.name} {conductor.net}\n")
        return [path]


def create_exporter() -> NetsExporter:
    return NetsExporter()
```

11.2.4.1 Preserve Scene Capabilities

The host checks required scene capabilities before invoking an exporter.

The `supported_capabilities()` class method returns a `FrozenSet[SceneCapability]`: an immutable set of typed enum members. Support can be queried on the class without constructing an exporter or loading its optional geometry dependencies. Derived exporters can extend `super().supported_capabilities()`.

The default method returns an empty set. Structural implementations without the method also support only scenes requiring no additional capabilities. They cannot silently consume required geometry, electrical-identity or accounting semantics.

A physical exporter must preserve both the reference geometry and the electrical ownership described in Section 11.1.5.4. The geometry and ownership contracts are:

- `GROUND_CUTOUTS` means the exporter subtracts the complete opening geometry from the reference slab and preserves the corresponding coating geometry
- `ELECTRICAL_IDENTITY` means it respects the reference net and keeps independent floating bodies distinct from named nets

Declaring a capability is a promise that the exporter implements it. A scene requiring an undeclared capability is rejected before export. Solver labels may need disambiguation; preserve the returned body-name mapping alongside the solver artifacts.

Use the shared helpers for the same geometry and ownership rules as the built-in exporters. Given a resolved `scene` and a finite `field_region` in the scene's integer coordinate grid:

```
from klayout_pex.plugin_api.v1 import (
    ground_body,
    retained_ground_region,
    solver_body_names,
)

labels = solver_body_names(scene)
reference_name = labels[ground_body(scene)]
reference_region = retained_ground_region(
```

```
    scene.ground_plane, field_region
)
```

`ground_body` and `conductor_body` return an `ElectricalBody`. Its `BodyKind` distinguishes a named net (`NET`) from an independent floating body (`FLOATING_BODY`). The identity does not depend on label spelling. `retained_ground_region` subtracts the union of openings while preserving their islands. `write_solver_body_map` writes the label mapping when disambiguation is needed.

The geometry helpers use `KLayout` regions when called. Importing the public helper names does not load that optional geometry backend, so plugin discovery can remain independent of it.

11.2.4.2 Capacitance Accounting

Table 80: Exporter support for capacitance accounting

`SceneCapability` member	Contract
`CAPACITANCE_ACCOUNTING`	Preserve physical geometry and common electrical potential, retain accounting identities through the solve, and apply the explicit result policies

An exporter whose `export()` implements the complete accounting contract overrides the class method:

```
from typing import FrozenSet
from klayout_pex.plugin_api.v1 import (
    PEX25DSceneExporter, SceneCapability,
)


class AccountingExporter(PEX25DSceneExporter):
    @classmethod
    def supported_capabilities(cls) -> FrozenSet[SceneCapability]:
        return frozenset({
            SceneCapability.CAPACITANCE_ACCOUNTING,
        })
```

An exporter declares every contract it implements. All capabilities required by a scene must be supported; declaring an unrelated capability cannot substitute for a missing one. The built-in `FasterCap` and `FastCap2` exporters declare all three. The `STL` exporter declares the two geometry and electrical-identity contracts, so it rejects accounting scenes.

Declaring the capability alone does not implement support. The exporter must distinguish physical electrical bodies from charge-accounting parts, preserve `accounting_part_id` and `accounting_regions`, and keep the physical reference body distinct from routing on the same net. Parts of a connected `NET` retain one potential; splitting their accounting identity must not introduce another independent physical terminal.

The built-in `FasterCap`/`FastCap2` path records the accounting map alongside solver input and uses it when consuming the matrix. A plugin that cannot carry that meaning through its solver and result consumer must leave that capability undeclared. The host then rejects the scene before the plugin writes output. Unknown protobuf required-capability values are rejected independently of the plugin's declaration.

See Section 11.1.5.7 for the text records, exact-grid rules and runnable example. These policies do not infer intrinsic compact-device capacitance ownership.

11.2.4.3 Example: FEM Solvers Elmer and Palace

As examples for external exporters, we provide plugins for two open source FEM field solvers, both used for capacitance extraction (electrostatics):

- klayout-pex-plugin-elmer: exporter `elmer` for Elmer FEM (`StatElecSolver`)
- klayout-pex-plugin-palace: exporter `palace` for Palace (`Electrostatic` problem)
- both use the volume mesh of klayout-pex-plugin-gmsh, which is also available as exporter `gmsh`

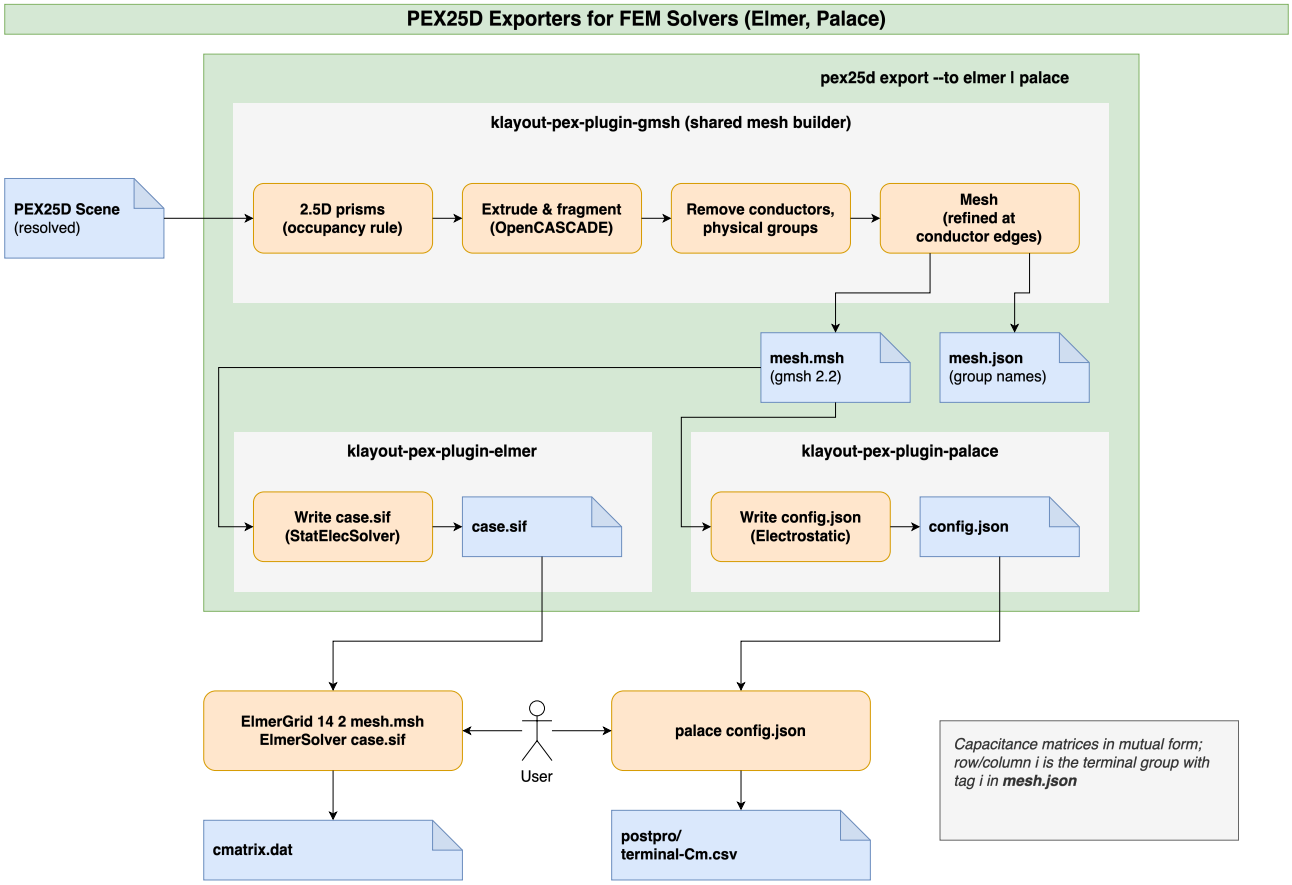


Figure 88: PEX25D Exporters for FEM Solvers

The mesh is built in these steps:

- 2.5D prisms: the domain is sliced at every z where a solid starts or ends
  - within each slab, conductors and the ground plane come first, then the dielectrics by wrap depth (see Section 11.1.5.5), and the background fills the rest
  - so the prisms fill the domain exactly once
- the prisms are extruded and fragmented using `OpenCASCADE`, so touching solids share their faces
- the conductors are removed, only their surfaces remain, as one physical group per net
- the elements are refined at the conductor edges, where the field concentrates
  - by default, down to the thickness of the thinnest conductor layer

Elmer FEM:

```
pip3 install --upgrade klayout-pex-plugin-elmer

pex25d export cell.pex25d --to elmer --out-dir elmer
cd elmer
```

```
ElmerGrid 14 2 mesh.msh -autoclean   # gmsh mesh -> Elmer mesh
ElmerSolver case.sif                 # writes cmatrix.dat
```

Palace:

```
pip3 install --upgrade klayout-pex-plugin-palace

pex25d export cell.pex25d --to palace --out-dir palace
cd palace
palace -np 4 config.json             # writes postpro/terminal-Cm.csv
```

Both solvers write the capacitance matrix in mutual form:

- off-diagonal cells contain the coupling capacitance between two terminals
- diagonal cells contain the capacitance to ground (zero, unless the outer boundary is grounded)
- row/column i belongs to the terminal group with tag i in `mesh.json` (nets, floating conductors, ground plane)

The options of the plugins (mesh sizes, element order, outer boundary condition, …) are listed in the `README` files of the repositories.

> **IIC-OSIC-TOOLS**
>
> The `IIC-OSIC-TOOLS` Docker image (see Section 1.5) contains Palace, Elmer FEM is not included yet. When running as `root`, call `palace -serial config.json`, as `mpirun` refuses to run as `root`.

For the test design `overlap_plates_100um_x_100um_m1_m2` (`ihp-sg13g2`, two plates of 100 µm × 100 µm in `Metal1` and `Metal2`, overlapping by 50 µm × 50 µm), the coupling capacitance between the plates is:

Solver	Mesh	$C_{LOWER,UPPER}$
Elmer FEM	default (94k tetrahedra)	181.3 fF
Palace, order 1	same mesh	181.3 fF
Palace, order 2	same mesh	179.4 fF
Elmer FEM	edges refined to 0.21 µm (~180k tetrahedra)	180.7 fF

A parallel plate estimate of the overlap alone gives 168 fF, the difference being the fringe capacitance.

References

Authors, SkyWater PDK. 2020. "SkyWater Open Source PDK." May. https://skywater-pdk.readthedocs.io.

Cong, Jason, Lei He, Andrew B. Kahng, David Noice, Nagesh Shirali, and Steve H.-C. Yen. 1997. "Analysis and Justification of a Simple, Practical 2 1/2-d Capacitance Extraction Methodology." *Proceedings of the 34th Annual Conference on Design Automation Conference - DAC '97*, 627–32. https://doi.org/10.1145/266021.266303.

Di Lorenzo, Enrico. 2019. "FasterCap Embedded Help (PDF Edition)." https://github.com/iic-jku/FasterCap/tree/master/doc/pdf.

Di Lorenzo, Enrico. 2023. "The Maxwell Capacitance Matrix: White Paper WP110301. Revision 03." https://www.fastfieldsolvers.com/Papers/The_Maxwell_Capacitance_Matrix_WP110301_R03.pdf.

Edwards, R. Timothy. 2022. *Whom Do You Trust? Validating Process Parameters for Open-Source Tools.* https://wiki.f-si.org/index.php?title=Whom_do_you_trust%3F:_Validating_process_parameters_for_open-source_tools.

Edwards, R. Timothy. 2023a. *Full r-c Extraction in Magic (Presentation Slides).* https://lists.chipsalliance.org/g/analog-wg/attachment/99/0/AWG_040423_Tim_Edwards_chips_alliance_slides.pdf.

Edwards, R. Timothy. 2023b. *Full r-c Extraction in Magic (Presentation Video).* https://www.youtube.com/watch?v=4d2mtiEHHeo.

Kahng, A. B., K. Samadi, and P. Sharma. 2006. "Study of Floating Fill Impact on Interconnect Capacitance." *7th International Symposium on Quality Electronic Design (ISQED'06)*, 691–96. https://doi.org/10.1109/isqed.2006.126.

Kurokawa, A., T. Kanamoto, A. Kasebe, Y. Inoue, and H. Masuda. 2004. "Efficient Capacitance Extraction Method for Interconnects with Dummy Fills." *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference (IEEE Cat. No.04CH37571)*, 485–88. https://doi.org/10.1109/cicc.2004.1358863.

Maxwell, James Clerk. 1873. *A Treatise on Electricity and Magnetism.* Vol. 1. Clarendon Press.

Nagel, Laurence W. 1975. "SPICE2: A Computer Program to Simulate Semiconductor Circuits." PhD Thesis UCB/ERL M520. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html.

Rautio, James C. 2008. "RFIC Design Methodologies Using Perfectly Calibrated Internal Ports." *2008 IEEE Compound Semiconductor Integrated Circuits Symposium*, 1–5. https://doi.org/10.1109/csics.2008.13.

Sagouo Minko, Flavien, and Tinus Stander. 2019. "A Comparison of Three-Dimensional Electromagnetic and RC Parasitic Extraction Analysis of Mm-Wave on-Chip Passives in SiGe BiCMOS Low-Noise Amplifiers." *International Journal of RF and Microwave Computer-Aided Engineering* 30 (2). https://doi.org/10.1002/mmce.22019.

Sheehan, B. N. 1999. "TICER: Realizable Reduction of Extracted RC Circuits." *1999 IEEE/ACM*

International Conference on Computer-Aided Design. Digest of Technical Papers (Cat. No.99CH37051), 200–203. https://doi.org/10.1109/ICCAD.1999.810649.

Sheehan, Bernard N. 2007. "Realizable Reduction of *RC* Networks." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26 (8): 1393–407. https://doi.org/10.1109/TCAD.2007.891374.

Yu, Wenjian, Mengsheng Zhang, and Zeyi Wang. 2006. "Efficient 3-d Extraction of Interconnect Capacitance Considering Floating Metal Fills with Boundary Element Method." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25 (1): 12–18. https://doi.org/10.1109/tcad.2005.853690.