KLayout-PEX Documentation

Martin Köhler

2025-08-19

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 . . . 2
1.5.1 Option 1: Using `IIC-OSIC-TOOLS` Docker Image . . . 2
1.5.2 Option 2: Standalone Installation . . . 3
1.5.3 Useful tools: `meshlab` . . . 4

2 First Steps **5**
2.1 Example Layouts . . . 5
2.2 Running the `KPEX/FasterCap` engine . . . 5
2.3 Running the `KPEX/MAGIC` engine . . . 7

3 Supporting new PDKs **8**
3.1 Customized PEX-"LVS" scripts . . . 8
3.2 Technology Definition Files . . . 9
3.2.1 JSON tech files for supported PDKs . . . 9
3.2.2 Process Stackup Definition . . . 9

4 `KPEX/FasterCap` Engine **13**
4.1 3D Input Geometries . . . 14
4.2 Example: MOM Capacitor . . . 15
4.3 Output Maxwell Capacitance Matrix . . . 15

5 `KPEX/MAGIC` Engine **19**
5.1 MAGIC database units . . . 19
5.2 Types of Parasitic Capacitances . . . 19
5.3 Substrate Capacitance . . . 19
5.4 Sidewall Capacitance . . . 20
5.5 Overlap Capacitance . . . 20
5.6 Fringe Capacitance . . . 21
5.7 Shielding Effects . . . 23
5.8 Parasitic Resistance . . . 24
5.8.1 Wire resistance . . . 25
5.8.2 Via resistance . . . 25

6 `KPEX/2.5D` Engine **27**

7 Comparison KPEX/2.5D with MAGIC **29**
7.1 Test Pattern `single_plate_100um_x_100um_li1_over_substrate` 29
7.2 Test Pattern `sidewall_20um_length_distance_200nm_li1` 31
7.3 Test Pattern `sideoverlap_simple_plates_li1_m1` 32
7.4 Test Pattern `r_single_wire_li1` 37
7.5 Test Pattern `r_contact_1x1_minsize_mcon` 38
7.6 Test Pattern `r_wire_voltage_divider_li1` 40

8 Netlist reduction: Feasibility study **41**
8.1 TICER algorithm: TIme-Constant Equilibration Reduction 41
8.1.1 Comments on page 2, equation (2.3) 42
8.1.2 Quick Nodes 43
8.1.3 Slow Nodes 44
8.2 TICER 2007 paper: Pseudocode 45
8.2.1 Pseudocode for slow node elimination 46
8.3 Example from TICER 2007 paper 47
8.3.1 Modified Nodal Analysis in the Laplace domain 48
8.3.2 Analysis of the original circuit 49
8.3.3 Analysis of the reduced circuit 49
8.3.4 Comparison of original and reduced circuits 50

9 Appendix **52**
References 52

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

Special thanks to the public funded German project FMD-QNC (16ME0831) https://www.elektronikforschung.de/projekte/fmd-qnc for financial support to this work.

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

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

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 C++, and parallelized), to improve performance. As we're already using the KLayout API (which is pretty similar between Python, Ruby and C++), this will be relatively straight-forward.

> **Warning**
>
> Please keep in mind that this software is early stage, and not yet intended for production use.

Engine	PEX Type	Status	Description
`KPEX/MAGIC`	`CC`, `RC`	Usable	Wrapper engine, using installed `magic` tool
`KPEX/FasterCap`	`CC`	Usable, **pending QA**	Field solver engine using `FasterCap`
`KPEX/FastHenry2`	`R`, `L`	Planned	Field solver engine using `FastHenry2`
`KPEX/2.5D`	`CC`	Usable, **pending QA**	Prototype engine implementing MAGIC concepts/formulas with `KLayout` means
`KPEX/2.5D`	`R`	Usable, **pending QA**	Prototype engine implementing resistance extraction with `KLayout` means
`KPEX/2.5D`	`RC`,`RCC`	Planned	

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/martinjankoehler/FasterCap/releases
- MAGIC-wrapper engine:
  - optional, required to run the MAGIC-wrapper engine
  - Follow the installation instructions at the GitHub repository
- Skywater sky130A PDK:
  - optional, for now, KPEX technology specific files are deployed within the `klayout-pex` Python package
  - `pip3 install --upgrade 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 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 help calling `kpex --help`.

2.1 Example Layouts

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

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

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

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

2.2 Running the `KPEX/FasterCap` engine

Preconditions:

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

> **Note**
>
> Normally, devices with SPICE (Nagel 1975) simulation models (e.g. like MOM-capacitors[1] in the sky130A PDK) are ignored ("blackboxed") during parasitic extraction.
> `kpex` has an option `--blacklist n` to allow extraction of those devices (whiteboxing), which can be useful during development (during the prototype phase, whiteboxing is actually the default setting, so please use `--blacklist y` to explicitly configure blackboxing).

Let's try the following:

```
kpex --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):

[1] Metal-Oxide-Metal capacitors

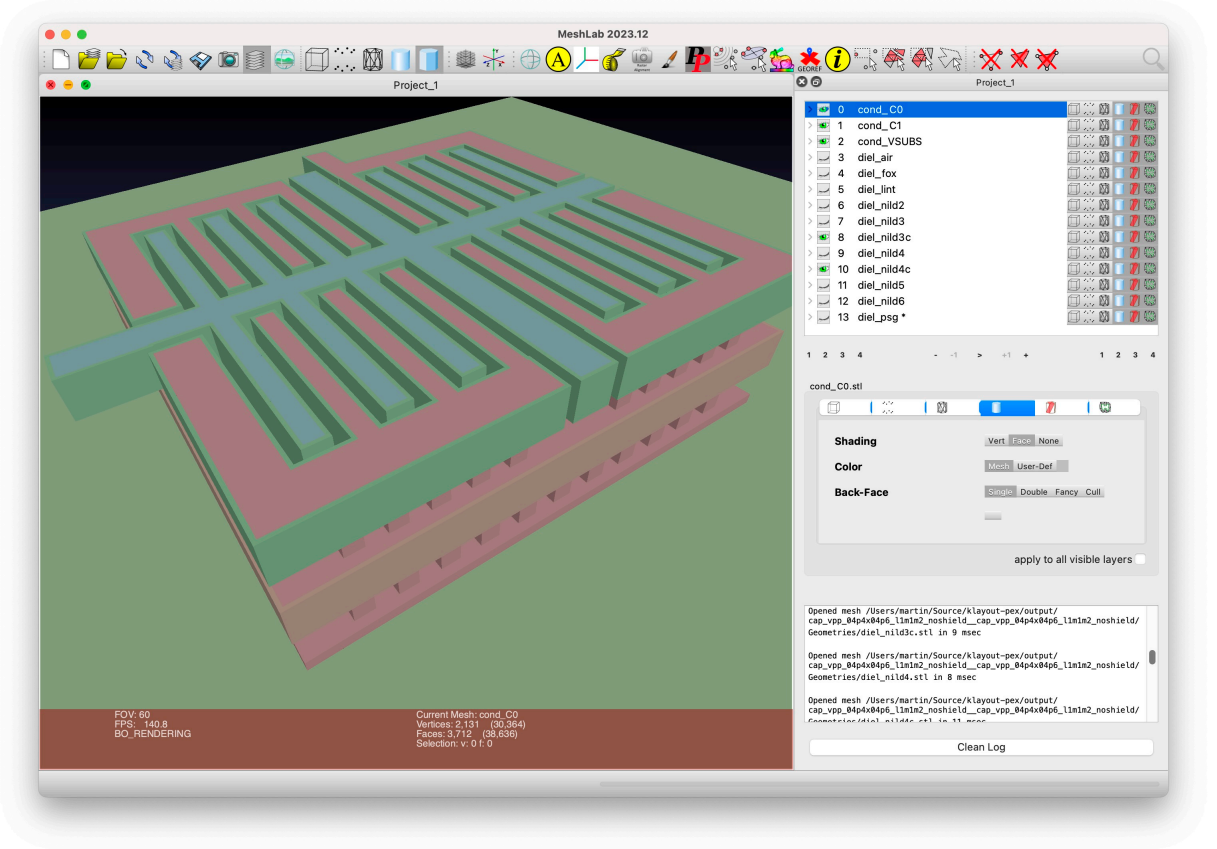
```
kpex --pdk sky130A --blackbox n --fastercap --gds \
  testdata/designs/sky130A/*/cap_vpp_04p4x04p6_l1m1m2_noshield.gds.gz
```

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 1.5.3) 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.

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

3 Supporting new PDKs

For every supported PDK[2], a `KPEX` technology definition is required, as well as customized PEX-"LVS" scripts.

3.1 Customized PEX-"LVS" scripts

`KLayout` has built-in support for Layout-Versus-Schematic (LVS) scripts, based on its Ruby API. Customized "LVS" scripts are ("ab")used in `KPEX`, not with the intent of comparing Layout-Versus-Schematic, but rather to extract the connectivity/net information for all polygons across multiple layers. The resulting net information is stored in a `KLayout LVS Database` ("LVSDB") within the run directory.

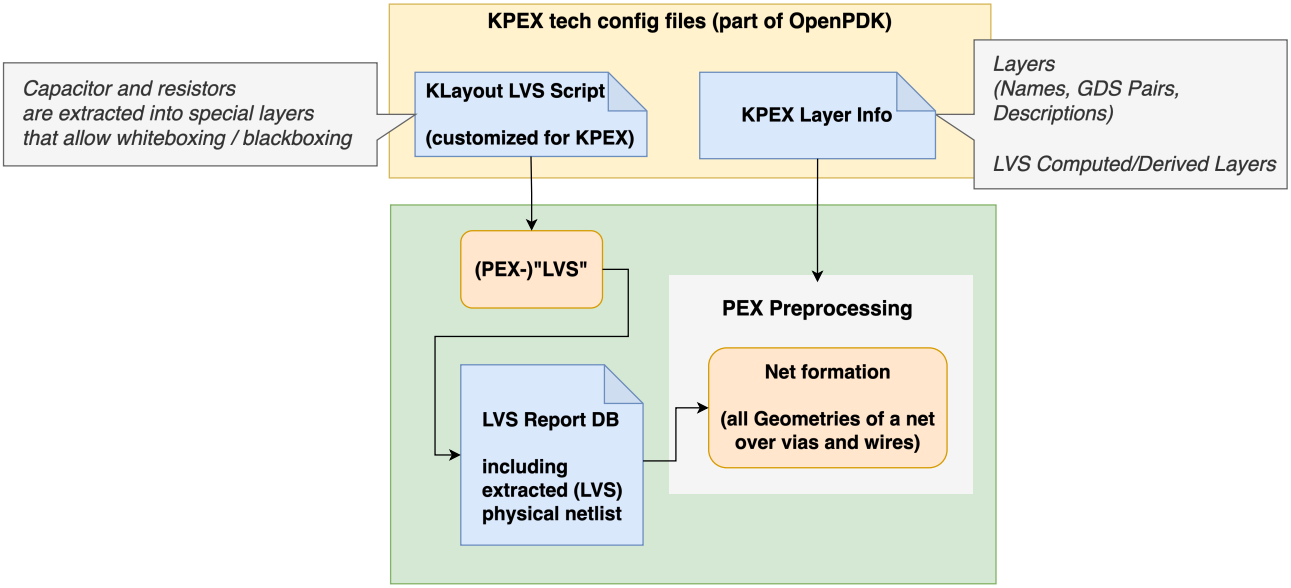


Figure 1: `KPEX` Net Formation

These customized "LVS" scripts are stored in:

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

What's specific about this customization:

- Layers names must be assigned, using KLayout's `(name(layer, name))` function
- MOM[3] capacitors, MIM[4] capacitors and resistors should be extracted to separate layers, to enable blackboxing / whiteboxing.

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

[2]Process Design Kit
[3]Metal-Oxide-Metal capacitors
[4]Metal-Insulator-Metal capacitors

3.2 Technology Definition Files

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

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

3.2.1 JSON tech files for supported PDKs

> **Note**
>
> The built-in JSON tech files are programmatically generated during the build process[5]. Therefore they not part of the repository source code, but of course part of the deployed Python wheels. To review those, look into your Python `site-packages`[6]/`klayout_pex_protobuf`.

3.2.2 Process Stackup Definition

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 Skywater `sky130A` process includes more intricate types of dielectrics (e.g., compared to IHP `sg13g2`), therefore I'll use this PDK as an example here.

We can derive the following data from Figure 2:

- Metal layers = {poly, li, metal1, metal2, metal3, metal4, metal5, capm, cap2m}
  - thickness
  - z-offset (relative to substrate)
- Contact layers = {licon}
- Via layers = {mcon, via1, via2, via3, via4}
- Dielectrics
  - dielectric constant k, the relative permittivity of the dielectric material
  - Different types:
    - Sidewall dielectrics = {IOX, SPNIT, NILD3_C, NILD4_C}
      - width of the sidewall around metals
      - height above metal
    - Conformal dielectrics = {LINT, TOPNIT, capild}
      - thickness above metal
      - thickness where no metal

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

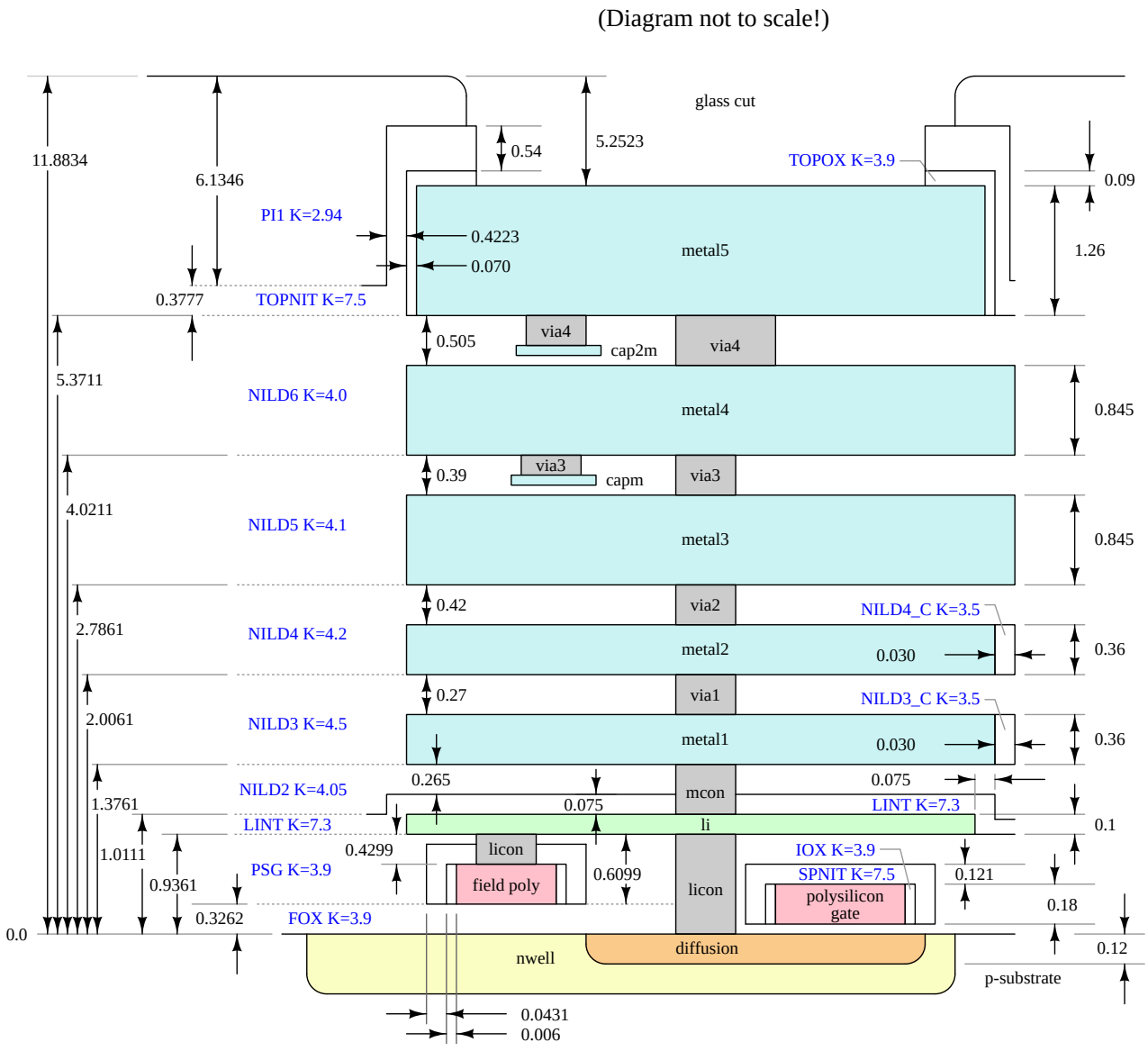


Figure 2: Skywater `sky130A` Process Stackup (Source: (Authors 2020))

        - thickness of the sidewall
    - Simple dielectrics = {PSG, NILD2, NILD3, NILD4, NILD5, NILD6}
        - embracing everything between 2 layers, including the sidewall and conformal dielectrics

An except of the `process_stack.proto` schema is shown in the code listing below, note:

- Line 3: `ProcessStackInfo.LayerType` is an enumeration type for the possible types of layers in the process stack
- Line 57: `ProcessStackInfo.layers` is a list of `LayerInfo`
  - the order of this list defines the orders of layers
- Line 40: `MetalLayer` describes a metal layer
- Line 43: `MetalLayer.contact_above` points to the via connecting to the metal layer above (omitted for the top metal layer)
- Line 30, 37: `ConformalDielectricLayer.reference` and `SidewallDielectricLayer.reference`
  - these refer to a dielectric or metal layer that they wrap around
  - `ConformalDielectricLayer` and `ConformalDielectricLayer` can be wrapped, e.g. `SPNIT` wraps `IOX`, which wraps `poly`

```
1  message ProcessStackInfo {
2
3      enum LayerType { …
4          LAYER_TYPE_SIMPLE_DIELECTRIC = 50;
5          LAYER_TYPE_CONFORMAL_DIELECTRIC = 60;
6          LAYER_TYPE_SIDEWALL_DIELECTRIC = 70;
7          LAYER_TYPE_METAL = 80;
8      }
9
10     message Contact { // Contact/Via
11         string name = 1;
12         string metal_above = 10;
13         double thickness = 20;
14
15         double width = 30;
16         double spacing = 31;
17         double border = 32;
18     }
19     …
20     message SimpleDielectricLayer { // Simple dielectric
21         double dielectric_k = 10;
22         string reference = 30;
23     }
24
25     message ConformalDielectricLayer { // Conformal dielectric
26         double dielectric_k = 10;
27         double thickness_over_metal = 20;
28         double thickness_where_no_metal = 21;
29         double thickness_sidewall = 22;
30         string reference = 30;
31     }
32
33     message SidewallDielectricLayer { // Sidewall dielectric
```

```
        double dielectric_k = 10;
        double height_above_metal = 20; // might be 0 if none
        double width_outside_sidewall = 21;
        string reference = 30;
    }

    message MetalLayer { // Metal
        double z = 1;          // z-offset in µm (of layer bottom), above substrate
        double thickness = 2;  // thickness in µm
        Contact contact_above = 40;
    }

    message LayerInfo {
        string name = 1;
        LayerType layer_type = 2;
        oneof parameters { …
            SimpleDielectricLayer simple_dielectric_layer = 12;
            ConformalDielectricLayer conformal_dielectric_layer = 13;
            SidewallDielectricLayer sidewall_dielectric_layer = 14;
            MetalLayer metal_layer = 15;
        }
    }

    repeated LayerInfo layers = 100;
}
```

4 KPEX/FasterCap Engine

FasterCap is a 3D and 2D parallel capacitance field solver, inspired by FastCap2. https://www.fastfieldsolvers.com/fastercap.htm

Starting from an input layout (e.g. GDS file) and a process stack-up (part of the Section 3.2), KPEX creates input geometries for FasterCap. After running FasterCap, the Maxwell capacitance matrix is parsed and interpreted to obtain the parasitic capacitances.

See Section 2.2 to get started with a first extraction example.

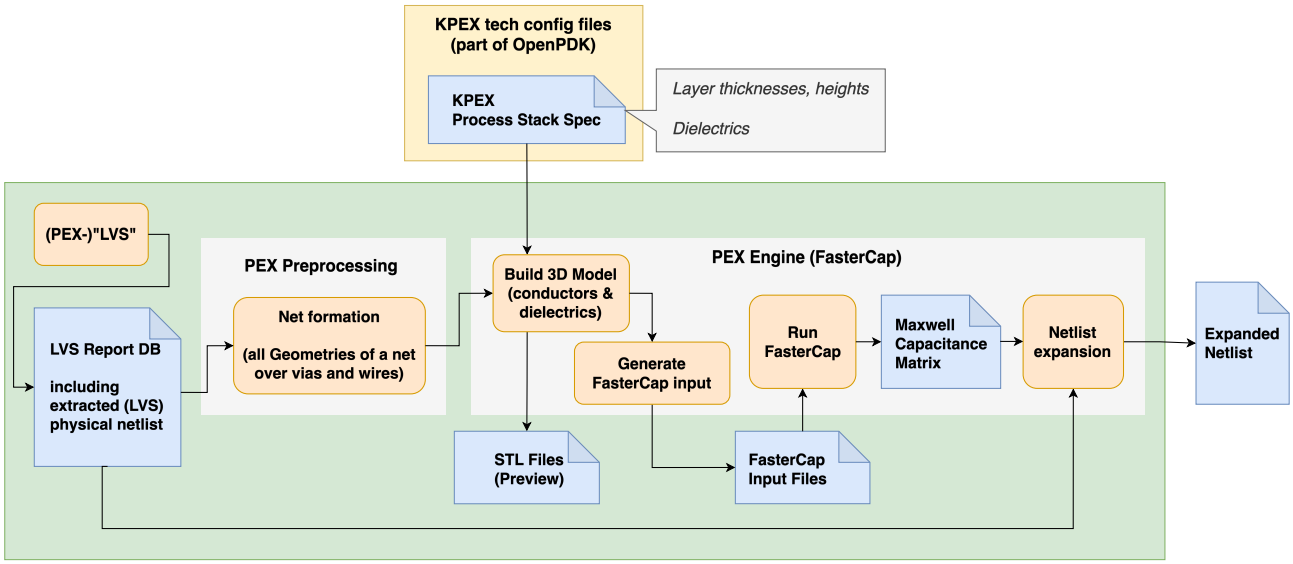


Figure 3: KPEX/FasterCap Engine

4.1 3D Input Geometries

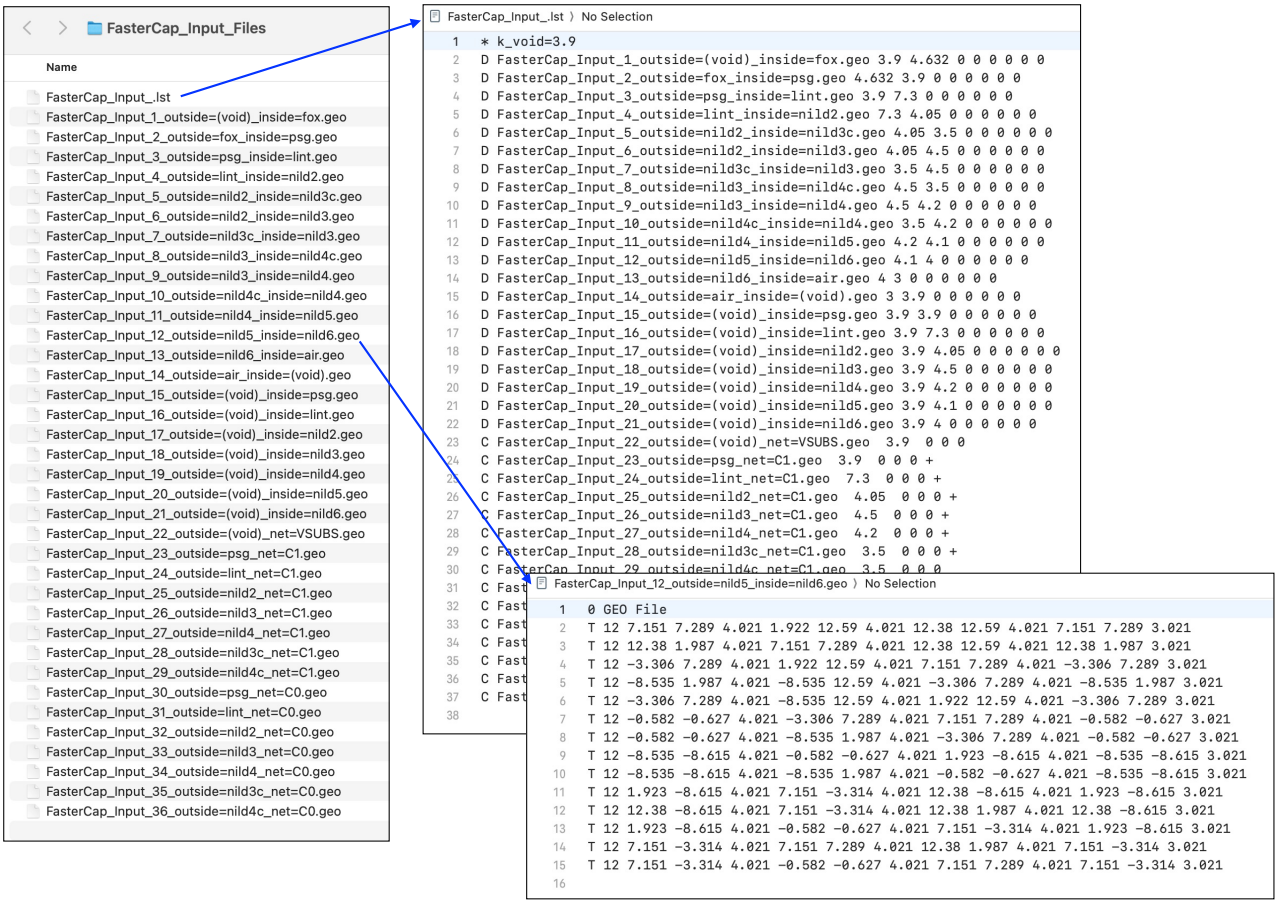


Figure 4: `FasterCap` 3D Input: File System Overview

The `FasterCap` input files and their format is documented in (Di Lorenzo 2019), a PDF version of the Windows-specific `*.chm` file is available at https://github.com/martinjankoehler/FasterCap/tree/master/doc/pdf.

`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

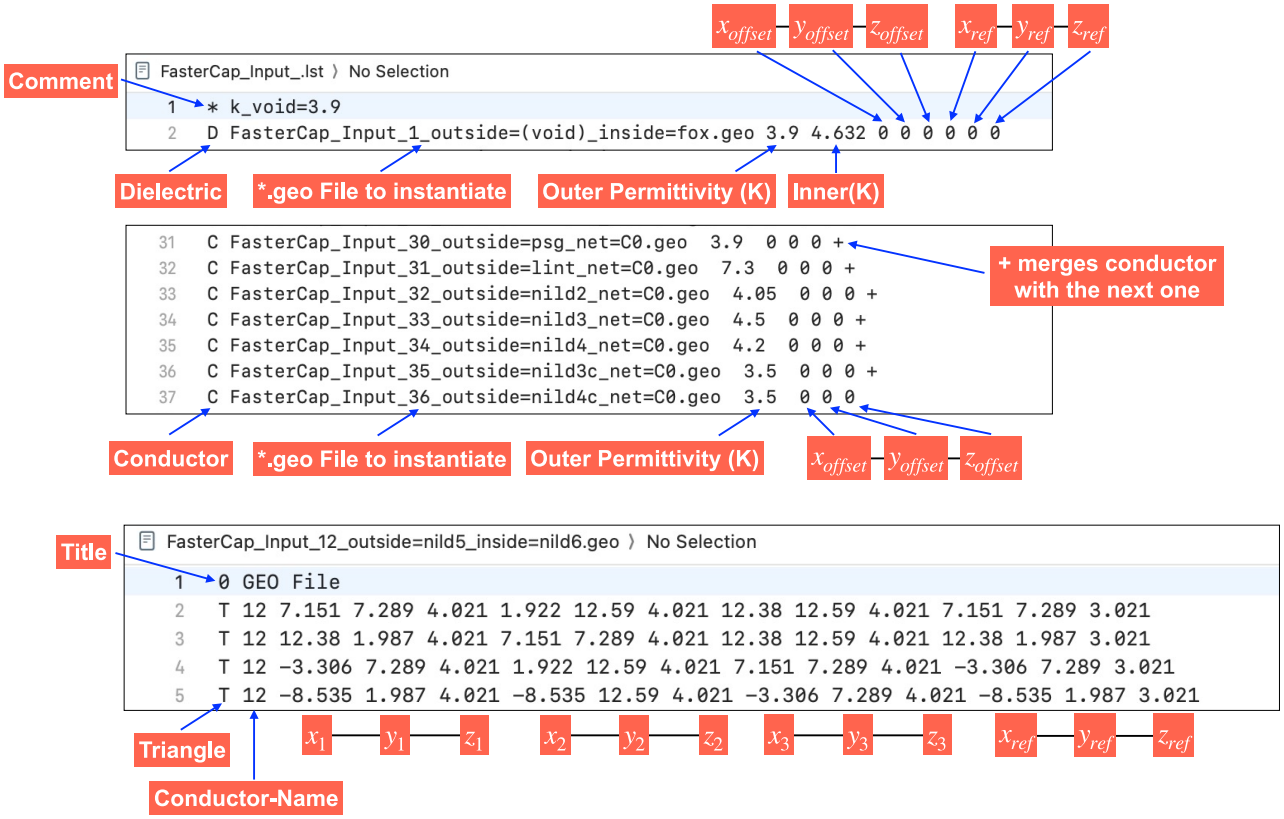


Figure 5: FasterCap 3D Input: File Format

4.2 Example: MOM Capacitor

Figure 6 depicts the MOM capacitor example of a from Section 2.2).

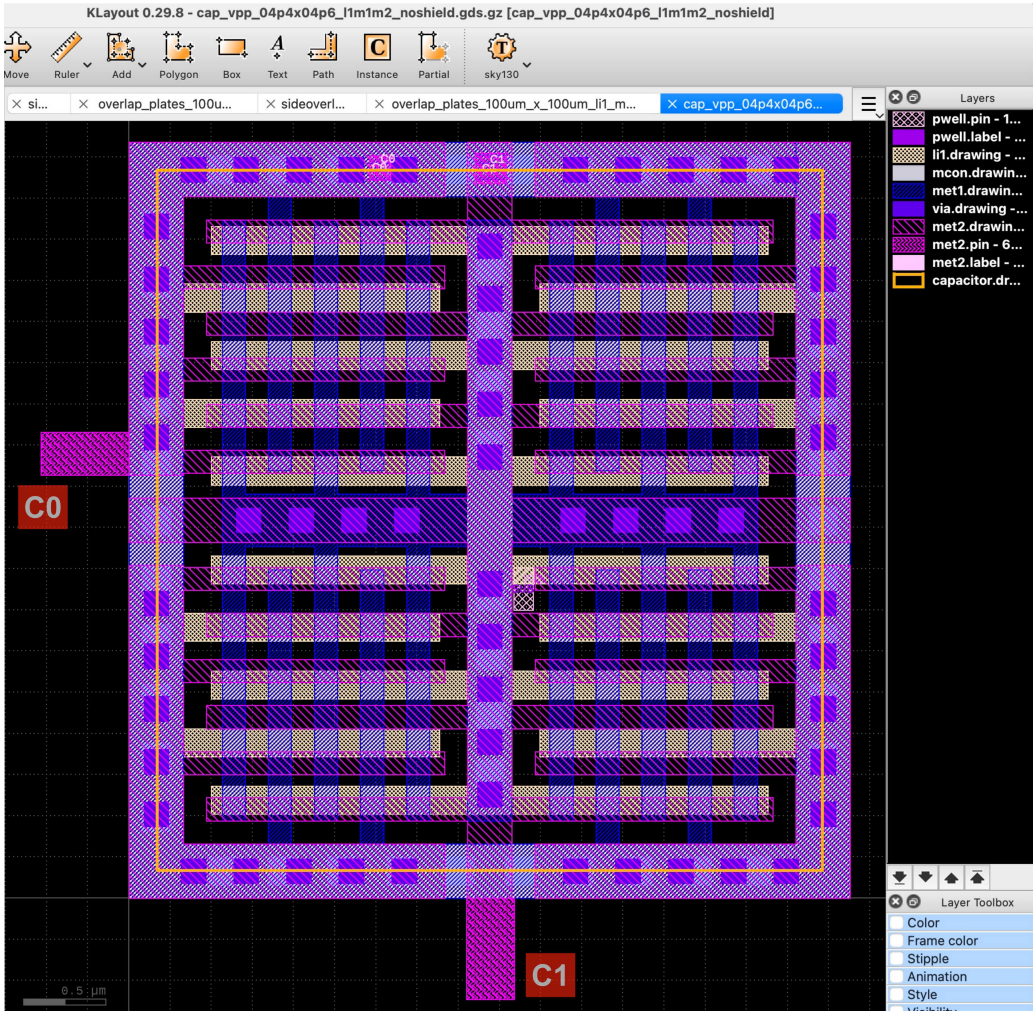
The corresponding schematic representation of Figure 7 contains 3 conductors (N_1, N_2 and N_3), and coupling capacitances:

- 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}

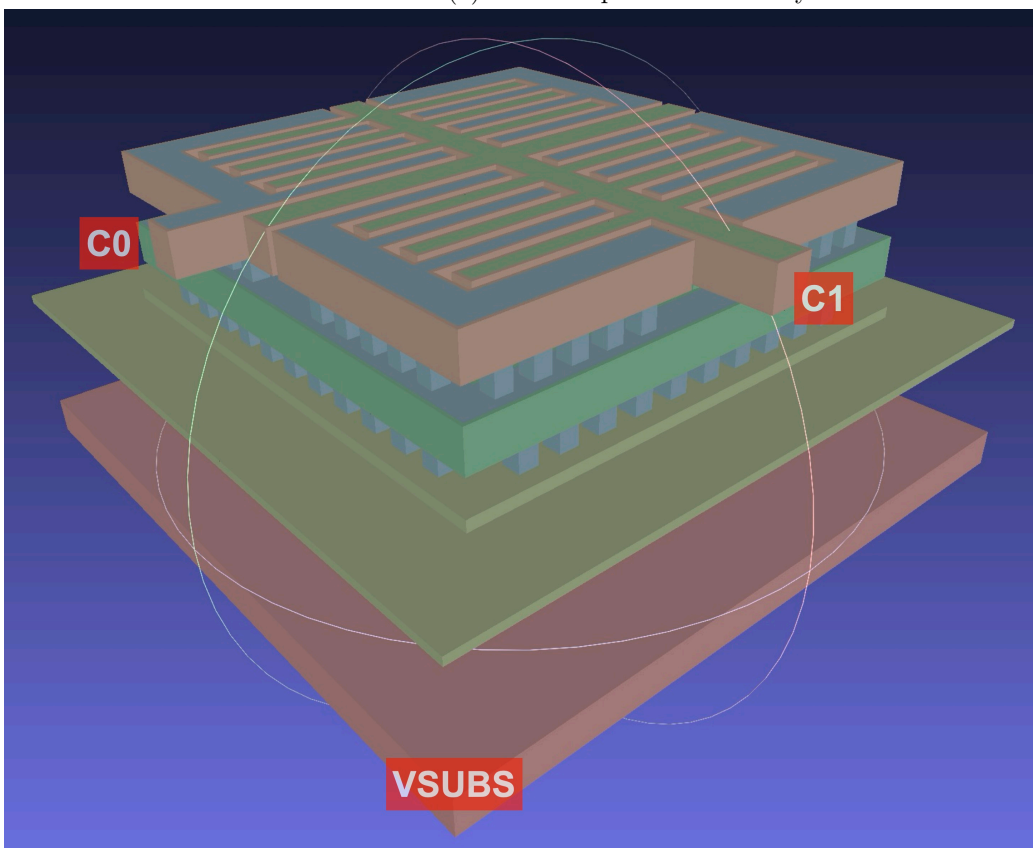
4.3 Output Maxwell Capacitance Matrix

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 6: MOM Capacitor

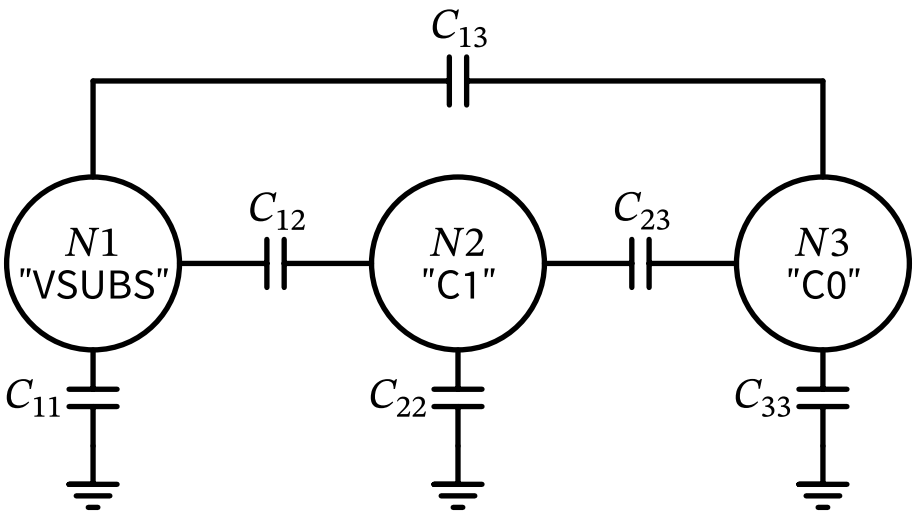


Figure 7: Schematic representation of the MOM capacitor.

```
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 8: `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`

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

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

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

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

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 9: `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 2.3 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

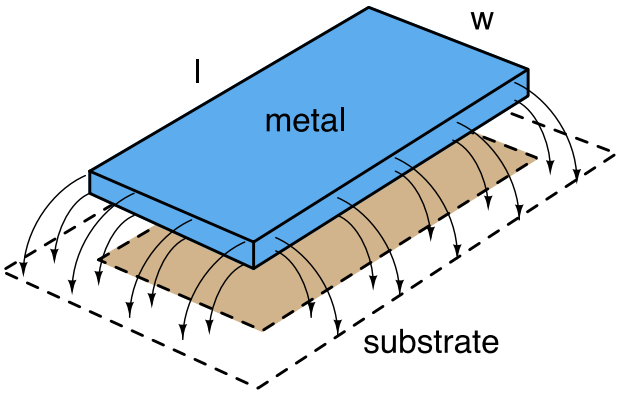
- To convert between MAGIC database units and μm, a scaling factor α is used, so that $L_{\mu m} = \frac{L_{dbu}}{\alpha}$
- E.g. for `sky130A`, $\alpha = 200.0$

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:

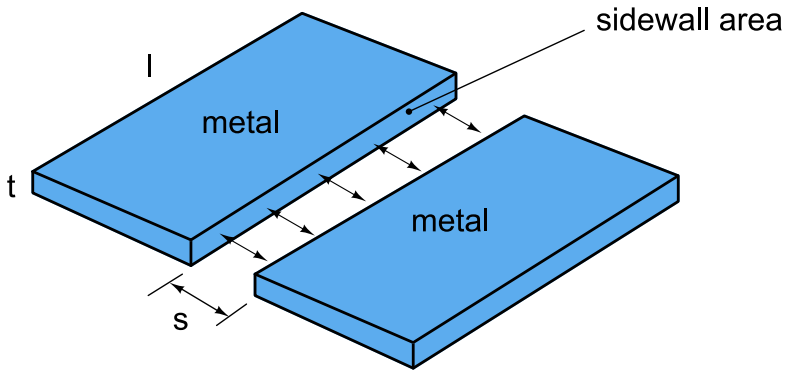
$$C_{area} = \frac{\epsilon_{si} * K}{d} * \text{area} \quad \left[\frac{F}{\mu m^2} * \mu m^2\right]$$

- 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



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

- 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 $s = 1$

5.5 Overlap Capacitance

- Overlapping area:

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

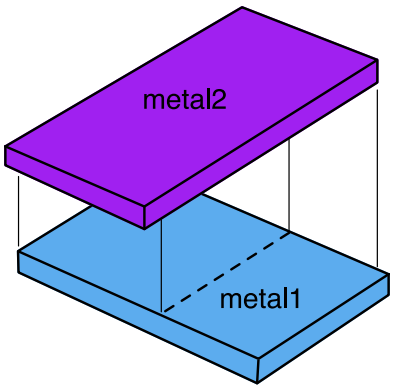


Figure 10: Overlap Capacitance

5.6 Fringe Capacitance

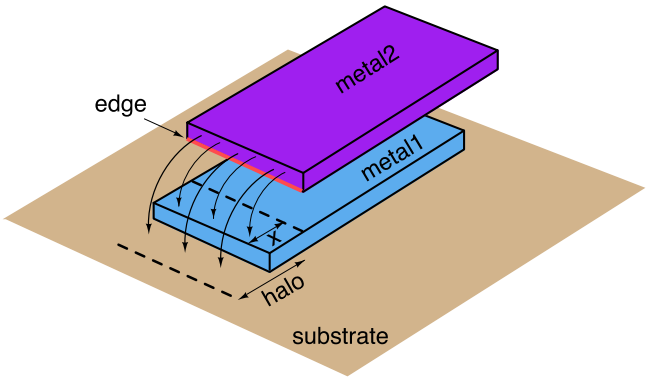


Figure 11: Fringe Capacitance: Overlapping (1)

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

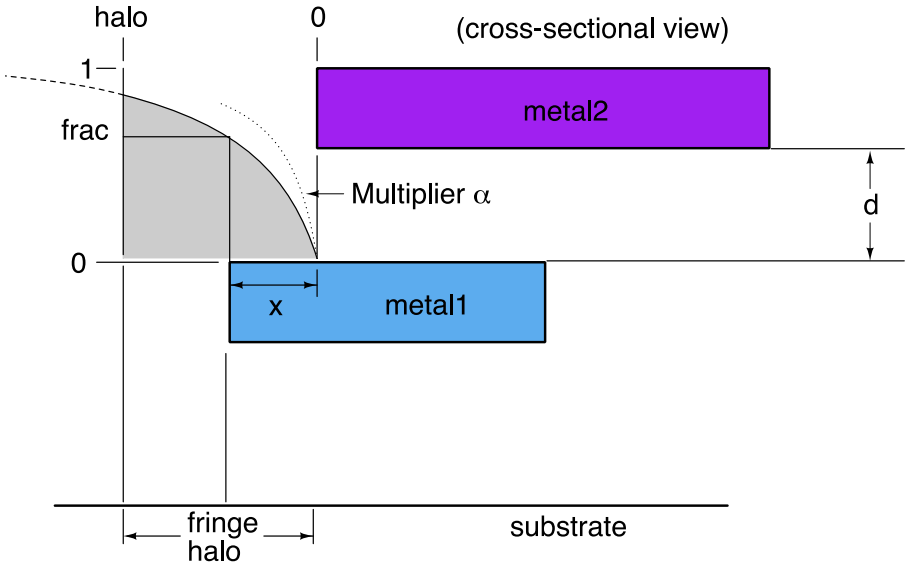


Figure 12: Fringe Capacitance: Overlapping (2)

- 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:
  - $\text{frac}_{metal1} = \frac{2}{\pi} * \text{atan}(\alpha_{metal2 \to metal1} * x)$
  - $\text{frac}_{sub} = \frac{2}{\pi} * \text{atan}(\alpha_{metal2 \to sub} * (\text{halo} - x))$
  - $\frac{2}{\pi}$ 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 $* C_{sideoverlap_{metal2 \rightarrow sub}}$

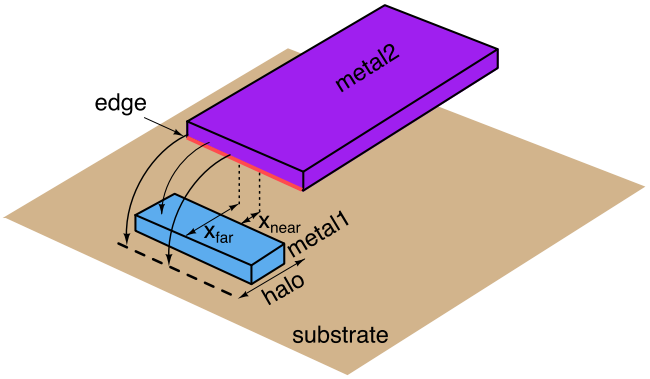


Figure 13: Fringe Capacitance: Non Overlapping (1)

- 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}

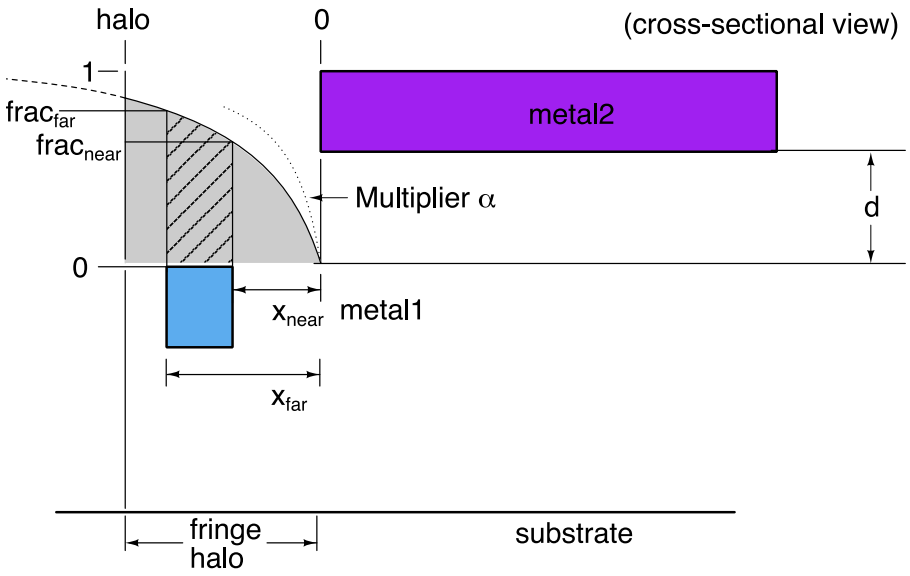


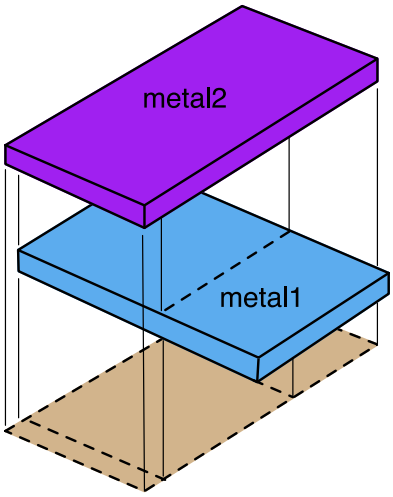
Figure 14: Fringe Capacitance: Non Overlapping (2)

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

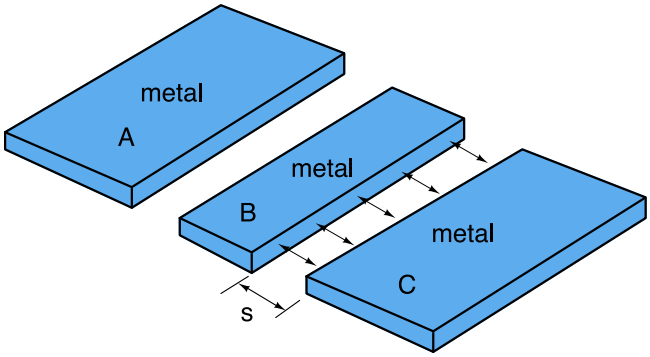
5.7 Shielding Effects

Table 3: Shielding effects

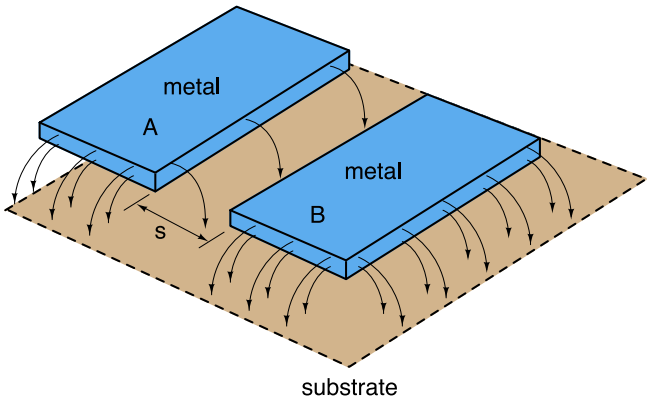
Type	Shielding To Substrate	Between layers	On same layer
Overlap shielding	✓	✓	✗
Sidewall shielding	✗	✗	✓
Lateral fringe shielding	✗	✓	✓
Vertical fringe shielding	✓	✓	✗



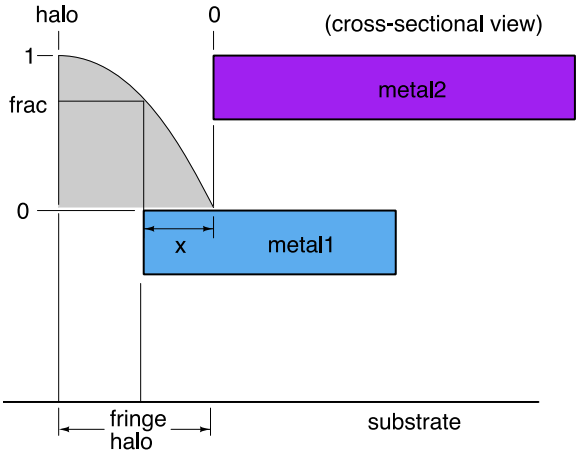
(a) Overlap shielding



(b) Sidewall shielding



(c) Lateral fringe shielding (1/2)



(d) Vertical fringe shielding

Figure 15: Shielding Effects

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 16 the coupling between the two shapes is the same in the above and below cases

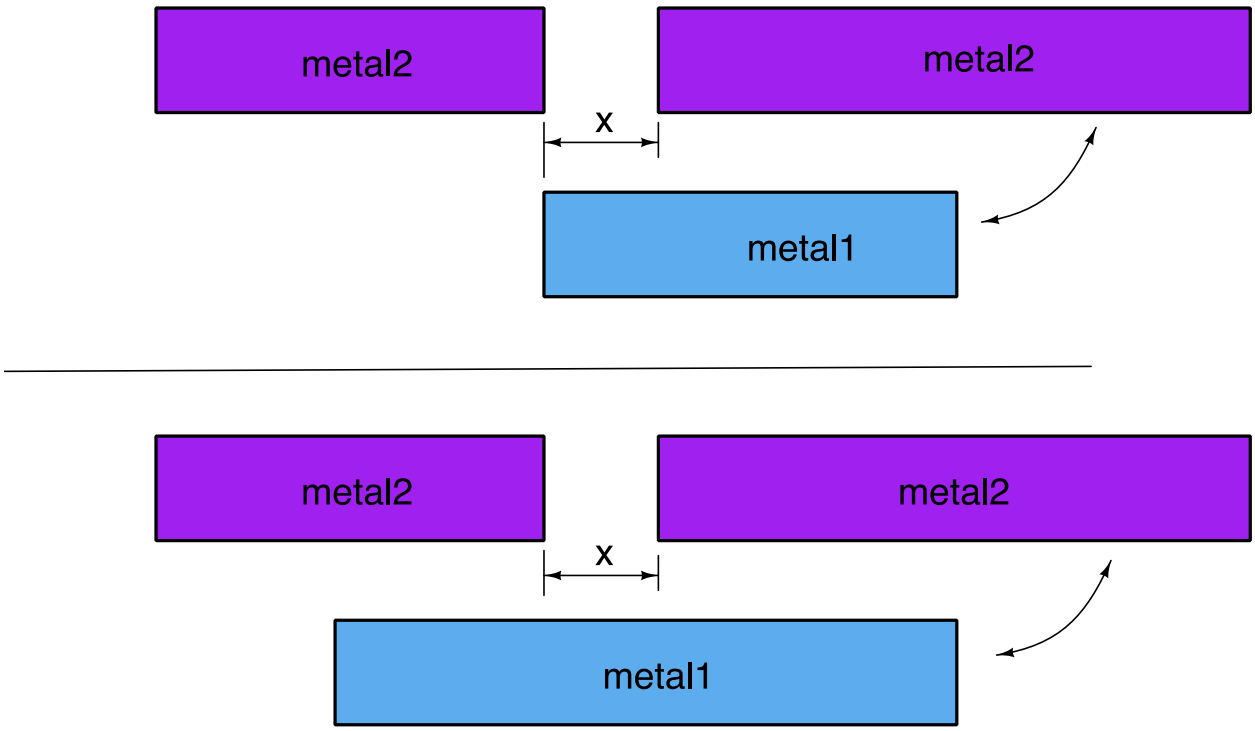
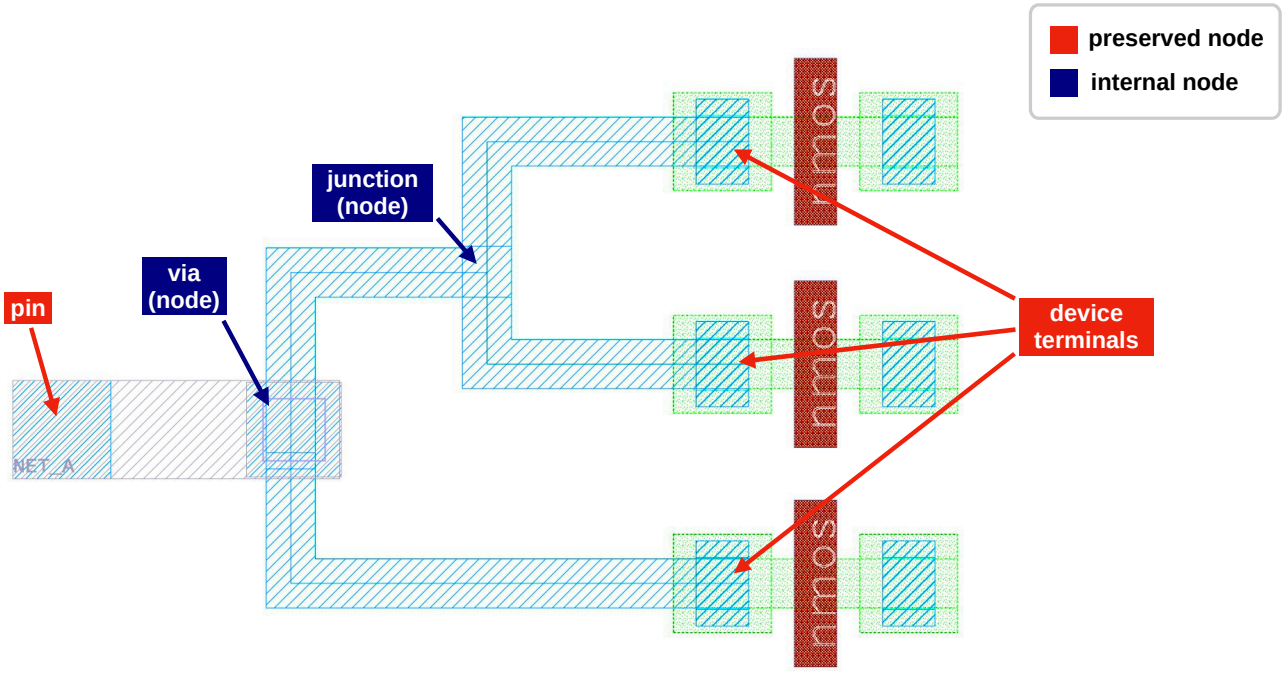


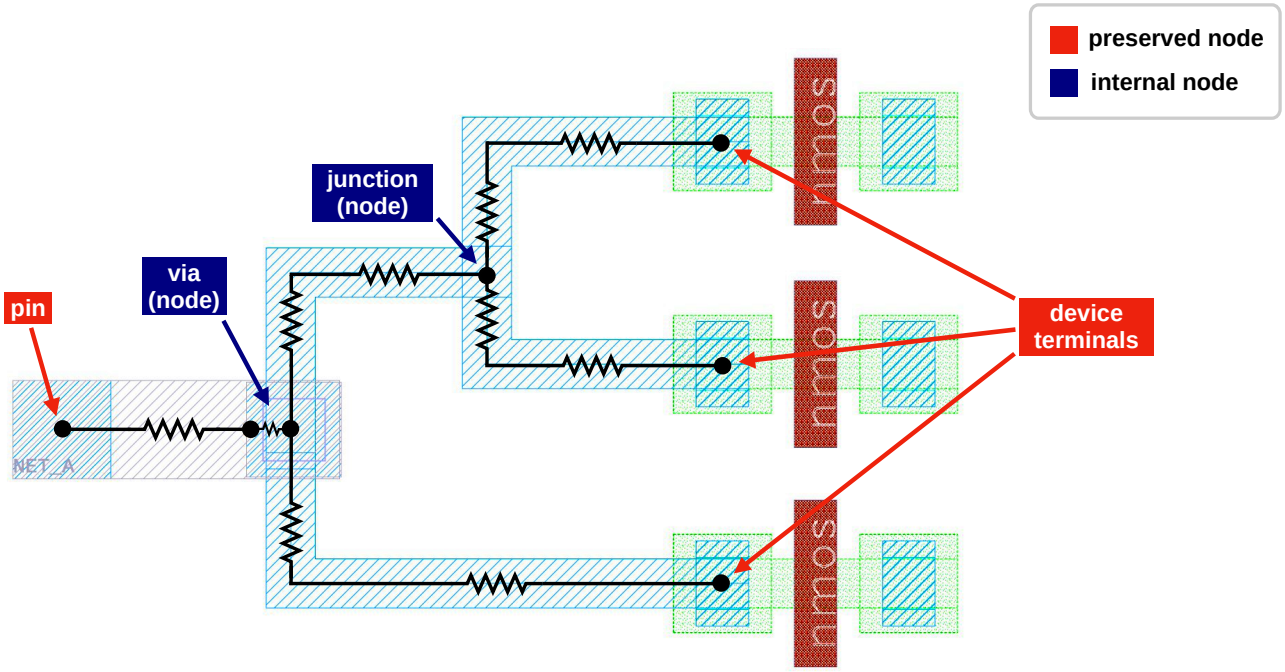
Figure 16: Lateral fringe shielding (2/2)

5.8 Parasitic Resistance

Magic constructs a graph of resistors between nodes

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





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

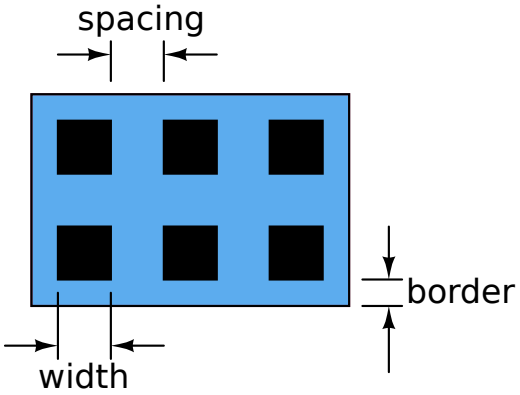


Figure 17: Via dimensions)

- 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

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.

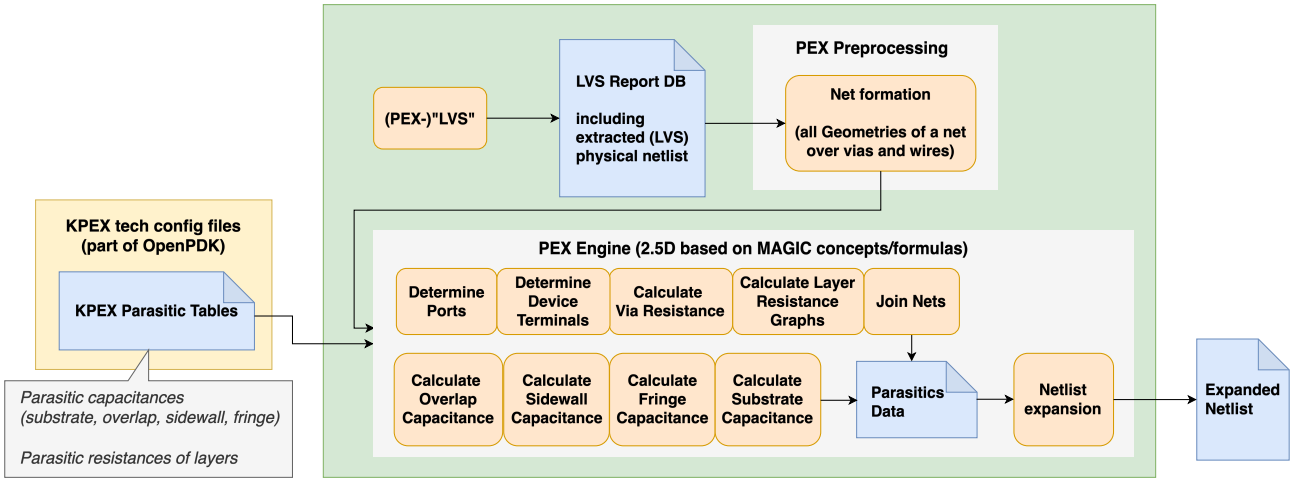


Figure 18: KPEX 2.5D Engine

In Figure 18, 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 coarse 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)
- klayout.db.PolygonWithProperties / klayout.db.EdgeWithProperties / klayout.db.EdgePairWithPr / klayout.db.BoxWithProperties (since 0.30.1): used to store the net information directly within geometry objects

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

> **Caution**
>
> **This section is under heavy construction!**

7 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[7], which will be shown and discussed for each example.
> - `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`

7.1 Test Pattern `single_plate_100um_x_100um_li1_over_substrate`

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

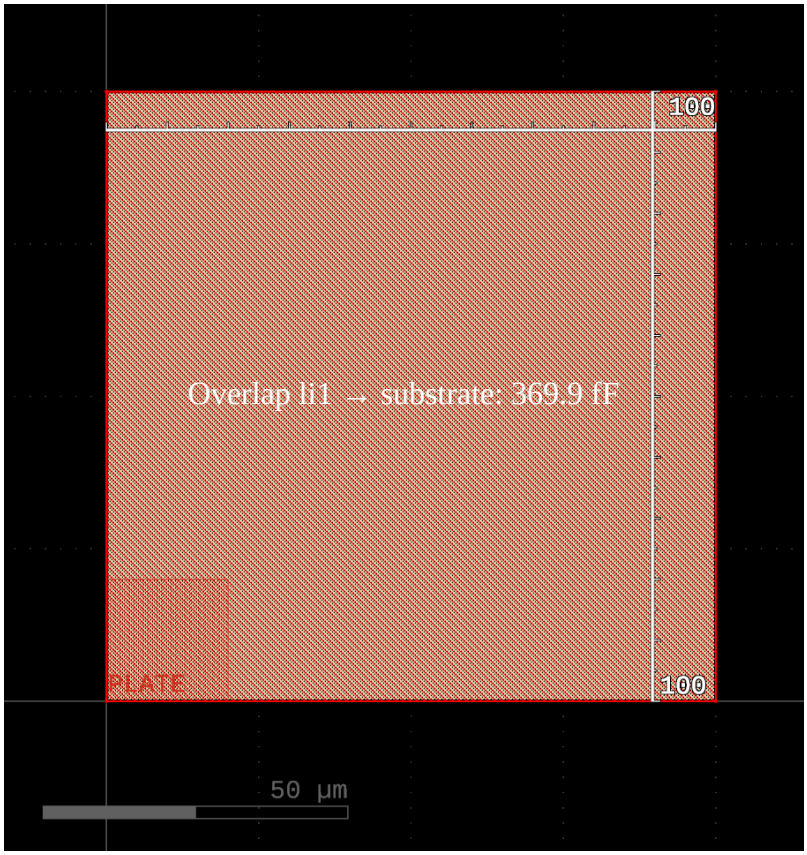


Figure 19: Overlap capacitance li1 to substrate.

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

Table 4: 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

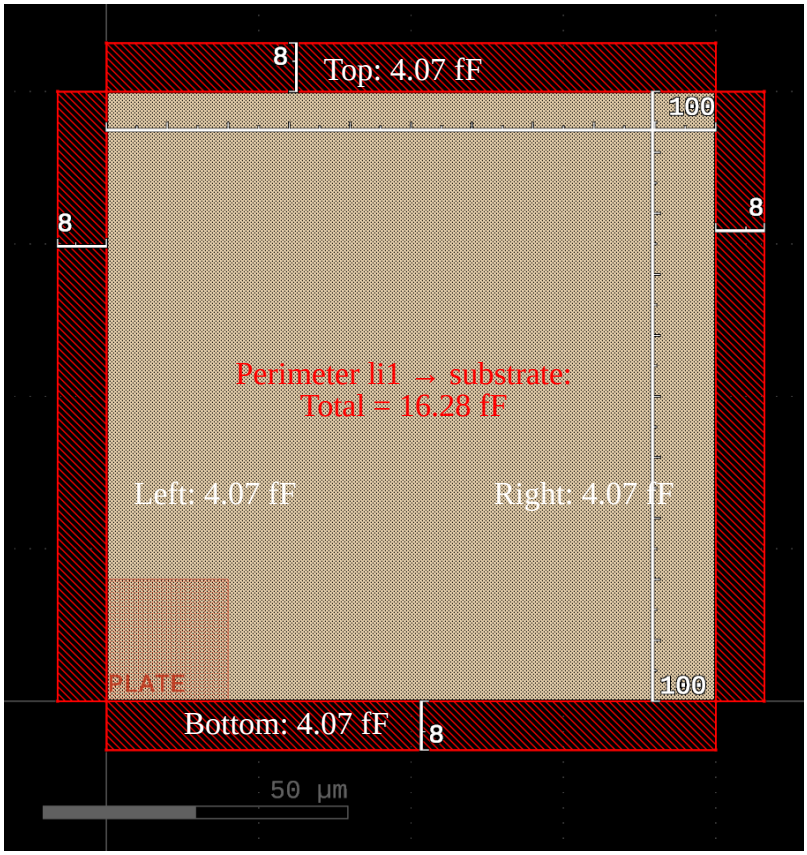


Figure 20: Perimeter (fringe) capacitance li1 to substrate.

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),
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 ---
```

7.2 Test Pattern `sidewall_20um_length_distance_200nm_li1`

GDS: https://github.com/martinjankoehler/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/sidewall_20um_length_distance_200nm_li1.gds.gz

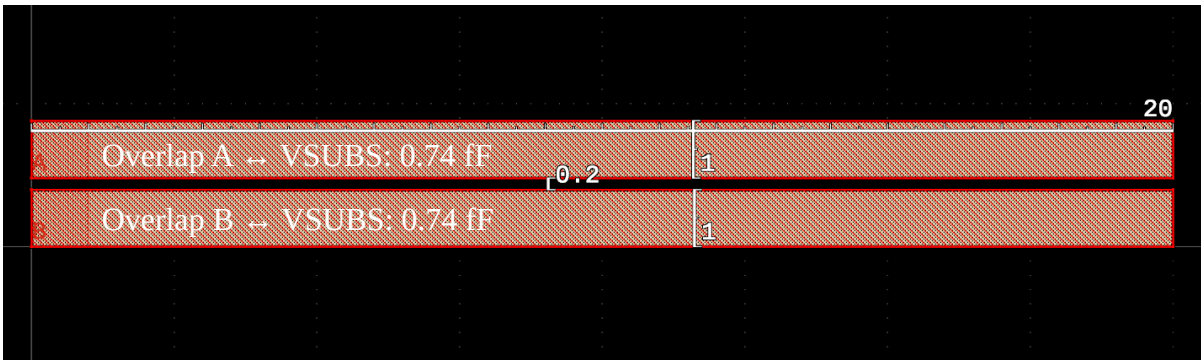


Figure 21: Overlap capacitance of nets on li1.

Table 6: 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

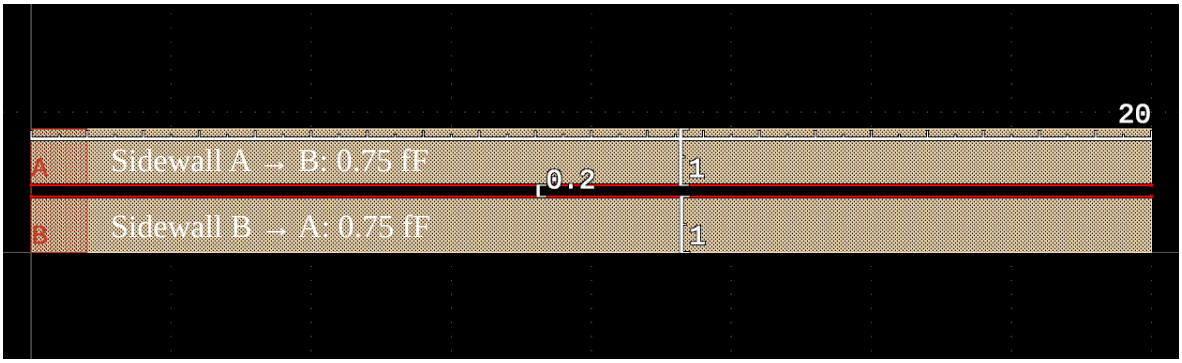


Figure 22: Sidewall capacitance between nets A and B on li1.

Table 7: Extracted Parasitic Sidewall Capacitances

Description	Layer	Net1	Net2	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
Sidewall	li1	A	B	0.75	0.75	18
Sidewall	li1	B	A	0.75	0.75	21

Table 8: 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

Description	Layer Top	Net Top	Layer Bottom	Net Bottom	MAGIC [fF]	KPEX/2.5D [fF]	MAGIC Lines
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

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

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.
```

7.3 Test Pattern `sideoverlap_simple_plates_li1_m1`

GDS: https://github.com/martinjankoehler/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/sideoverlap_simple_plates_li1_m1.gds.gz

Table 9: 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

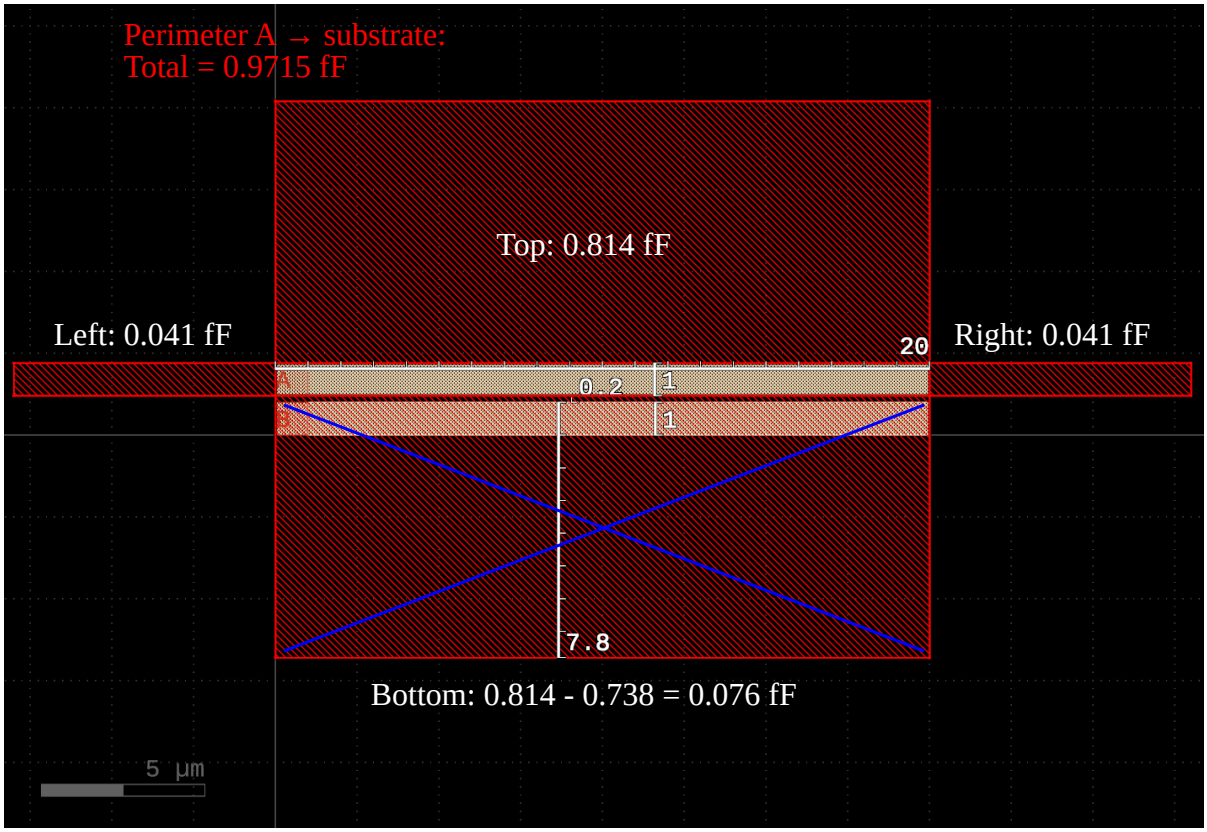


Figure 23: Fringe capacitance between net A and substrate.

Figure 24: Overlap capacitances to substrate.

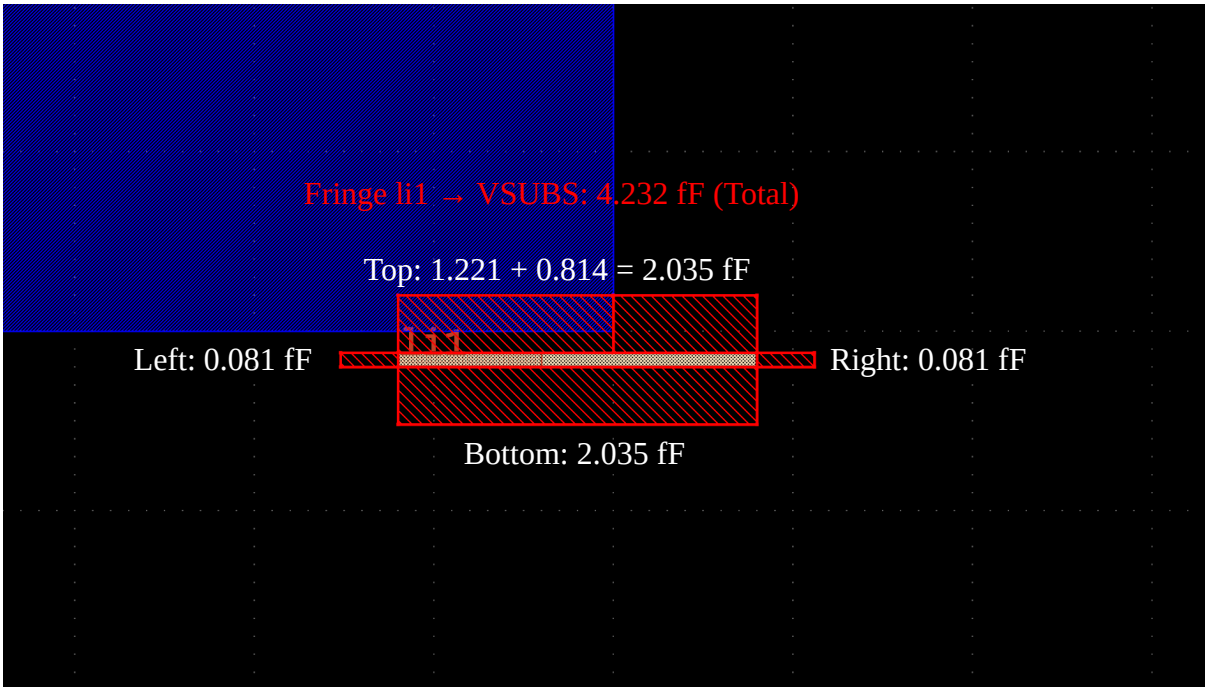


Figure 25: Fringe capacitances li1 to substrate.

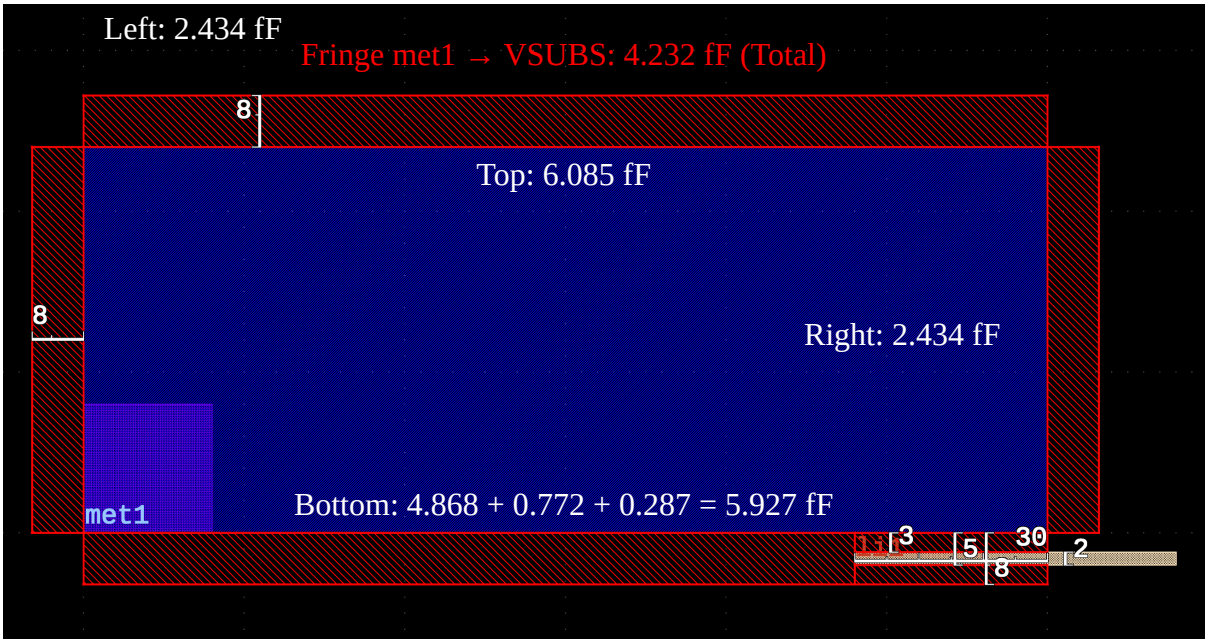


Figure 26: Fringe capacitances met1 to substrate.

Table 10: 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 11: 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

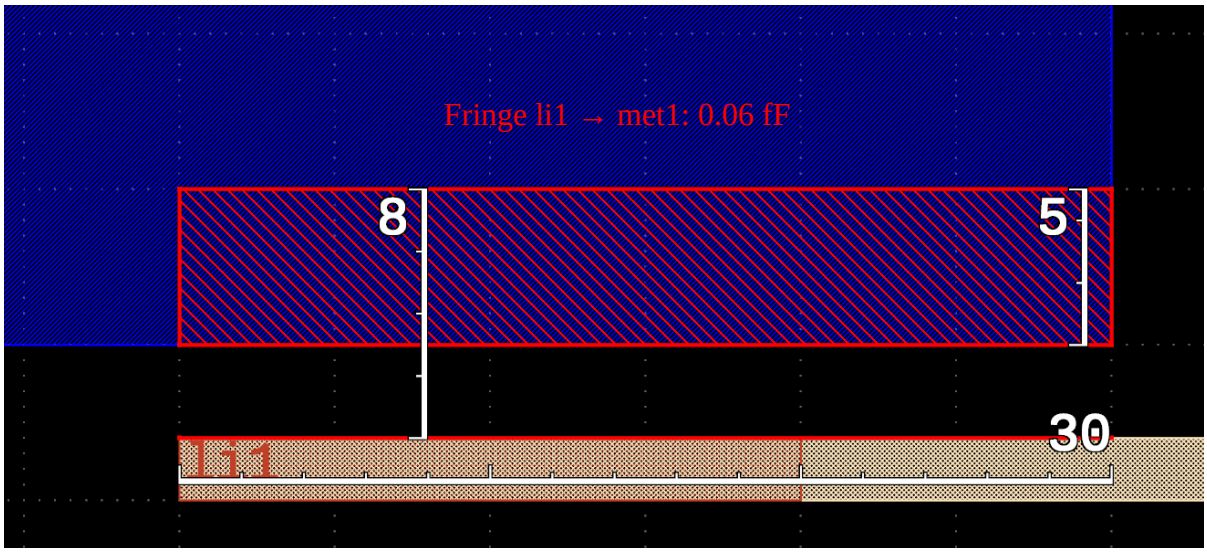


Figure 27: Fringe capacitances li1 to m1.

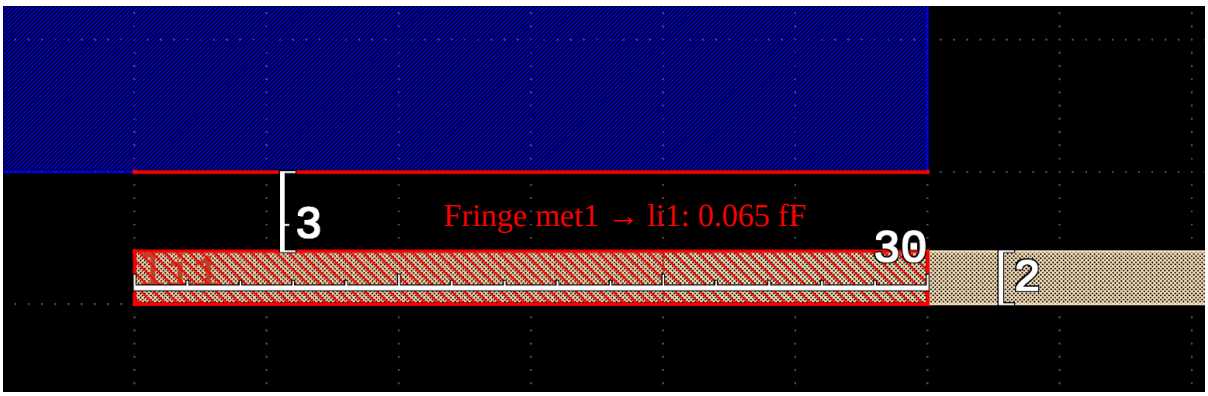


Figure 28: Fringe capacitances m1 to li1.

```
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
```

7.4 Test Pattern `r_single_wire_li1`

GDS: https://github.com/martinjankoehler/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_single_wire_li1.gds.gz

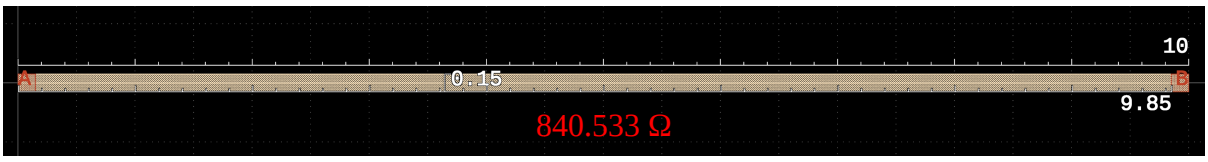


Figure 29: Resistance of a single wire on li1.

- 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

$$R_{wire} = \frac{L}{H} * R_{coeff}$$

$$R_{wire} = \frac{9.85\,\mu m}{0.15\,\mu m} * 12800\,m\Omega = 84053\dot{3}\,m\Omega = 840.53\dot{3}\,\Omega$$

```
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.

7.5 Test Pattern `r_contact_1x1_minsize_mcon`

GDS: https://github.com/martinjankoehler/klayout-pex/blob/main/testdata/designs/sky130A/test_patterns/r_contact_1x1_minsize_mcon.gds.gz

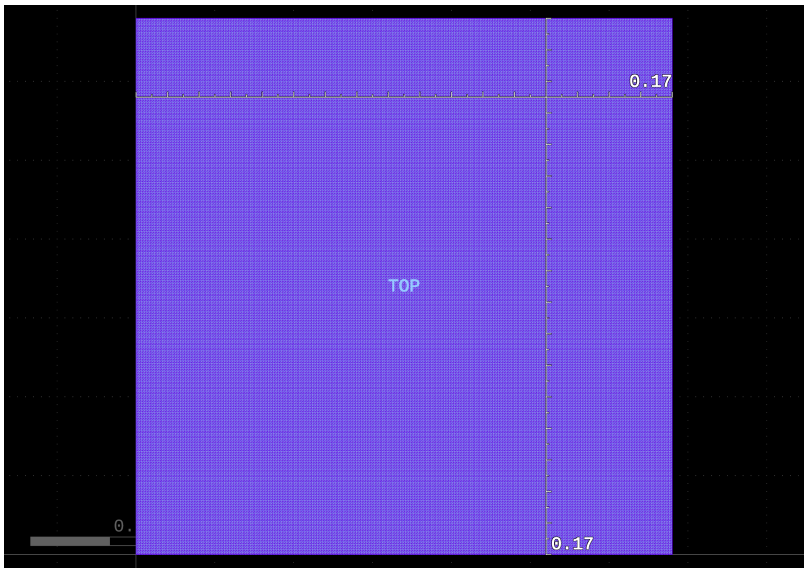


Figure 30: Resistance of a minimum sized via on mcon.

- 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

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

$$n_x = 1 + \left\lfloor \frac{0.17\,\mu m - (0.17 + 2 * 0.0)\,\mu m}{0.17\,\mu m + 0.19\,\mu m} \right\rfloor = 1$$

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

$$n_y = 1 + \left\lfloor \frac{0.17\,\mu m - (0.17 + 2 * 0.0)\,\mu m}{0.17\,\mu m + 0.19\,\mu m} \right\rfloor = 1$$

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

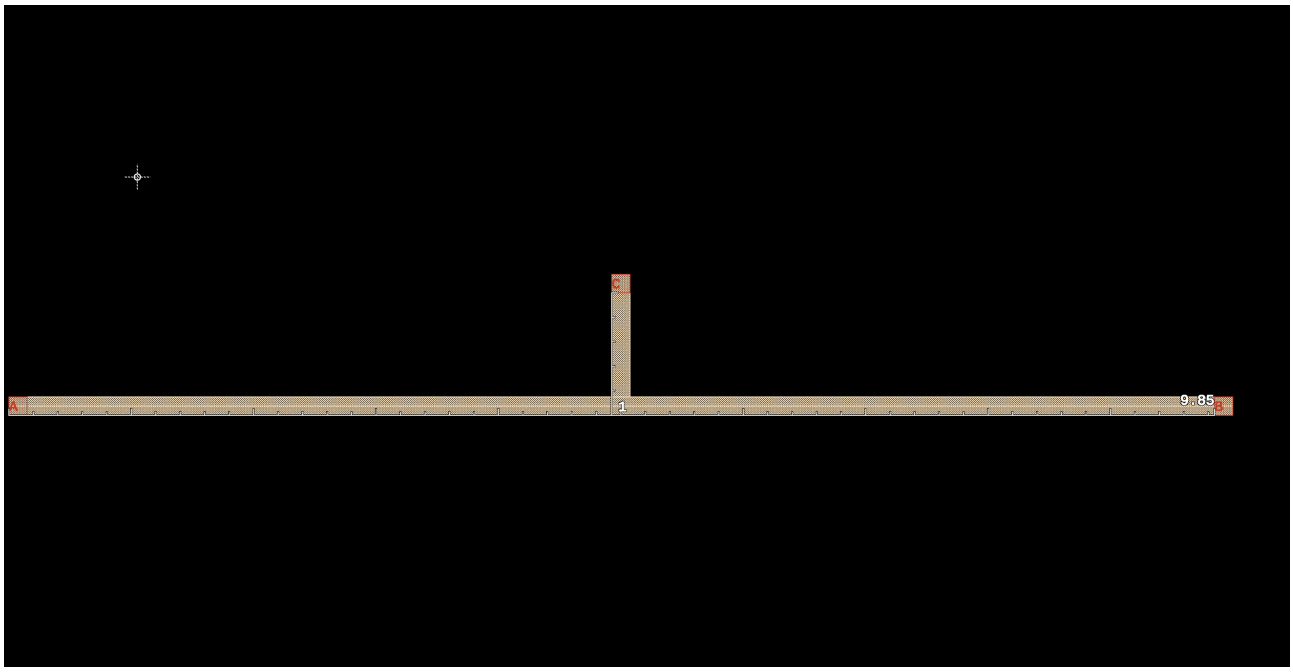
$$R_{via} = \frac{9300\,m\Omega}{1 * 1} = 9300\,m\Omega = 9.3\,\Omega$$

```
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
```

7.6 Test Pattern `r_wire_voltage_divider_li1`

GDS: https://github.com/martinjankoehler/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 12: 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

8 Netlist reduction: Feasibility study

8.1 TICER algorithm: TIme-Constant Equilibration Reduction

This section reviews the original TICER paper (B. N. 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 31, 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]

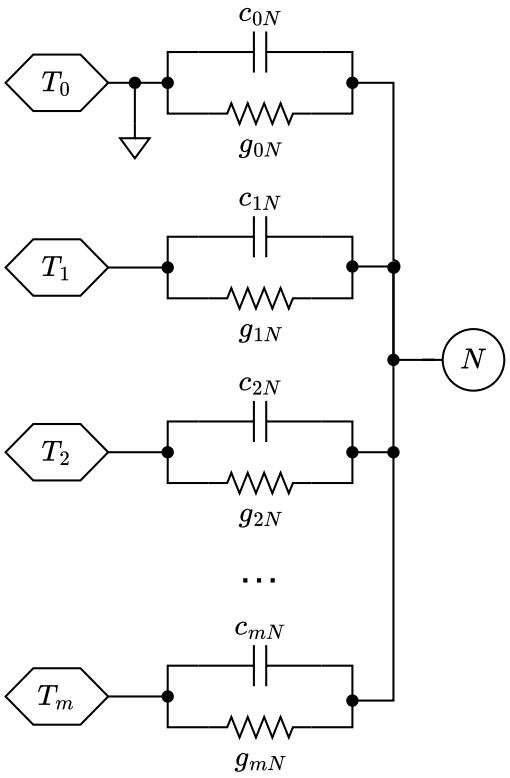


Figure 31: N-terminal star network

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 $\tilde{\mathbf{v}}$):
>
> $$\mathbf{Y} = \left[\begin{array}{cccc:c} y_{11} & y_{12} & \cdots & y_{1,N-1} & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2,N-1} & y_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hdashline y_{N1} & y_{N} & \cdots & y_{N,N-1} & y_{NN} \end{array}\right] = \left[\begin{array}{ccccc} & & & & y_{1N} \\ & \tilde{\mathbf{Y}} & & & y_{2N} \\ & & & & \vdots \\ \hdashline y_{N1} & y_{N2} & \cdots & y_{N,N-1} & y_{NN} \end{array}\right] = \left[\begin{array}{c:c} \tilde{\mathbf{Y}} & \mathbf{y} \\ \hdashline \mathbf{y}^T & y_{NN} \end{array}\right] \tag{1}$$

8.1.1 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:

$$\mathbf{E}_{ij} = \frac{(g_{iN} + sc_{iN})(g_{jN} + sc_{jN})}{s\chi_N + \gamma_N} \tag{2.4}$$

$$\mathbf{F}_i = \frac{g_{iN} + sc_{iN}}{s\chi_N + \gamma_N} j_N \tag{2.5}$$

8.1.2 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

8.1.3 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

8.2 TICER 2007 paper: Pseudocode

This section reviews some aspects of the 2007 TICER paper (Bernard N. 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)
```

8.2.1 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 (B. N. 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)
```

8.3 Example from TICER 2007 paper

Let's reduce the example presented by Sheehan in the from the 2007 paper (Bernard N. Sheehan 2007), as shown in Figure 32. The goal is to eliminate the quick node v_3.

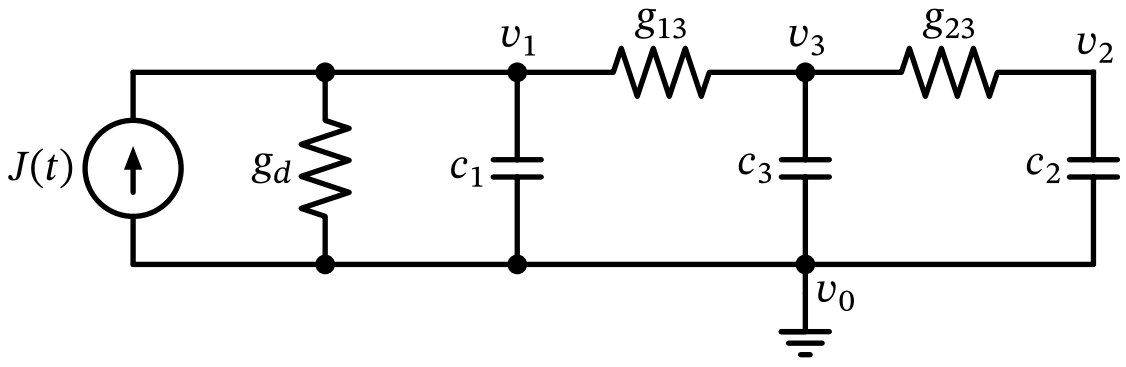


Figure 32: Passive RC circuit shown in Bernard N. Sheehan (2007).

- 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 $g_{12} = \frac{g_{13} * g_{23}}{g_{13}+g_{23}}$

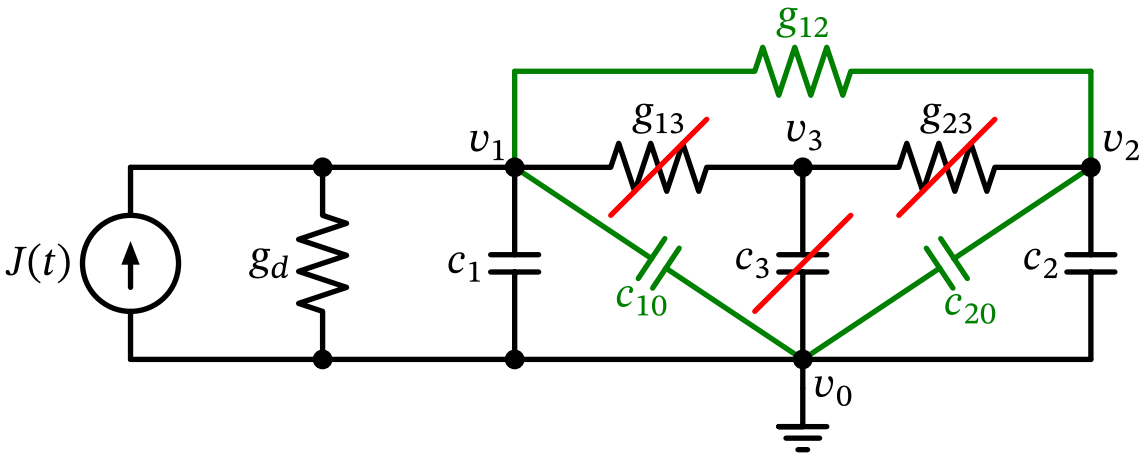


Figure 33: Passive RC circuit after the elimination of v_3.

8.3.1 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:

$$H(s) = \frac{\text{adj}(s\mathbf{C} + \mathbf{G})_{21}}{det(s\mathbf{C} + \mathbf{G})}$$

8.3.2 Analysis of the original circuit

So for the original circuit of Figure 32, 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:

$$H(s) = \frac{1}{0.01s^3 + 2.03s^2 + 4.02s + 1}$$

8.3.3 Analysis of the reduced circuit

The reduced circuit in Figure 33 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:

$$H(s) = \frac{0.5}{1.010025s^2 + 2.01s + 0.5}$$

8.3.4 Comparison of original and reduced circuits

In the Bode plots, **?@fig-bode-plot-ft-vs-mag** and **?@fig-bode-plot-ft-vs-phase**, we see that magnitude and phase responses stays the same up to a certain frequency.

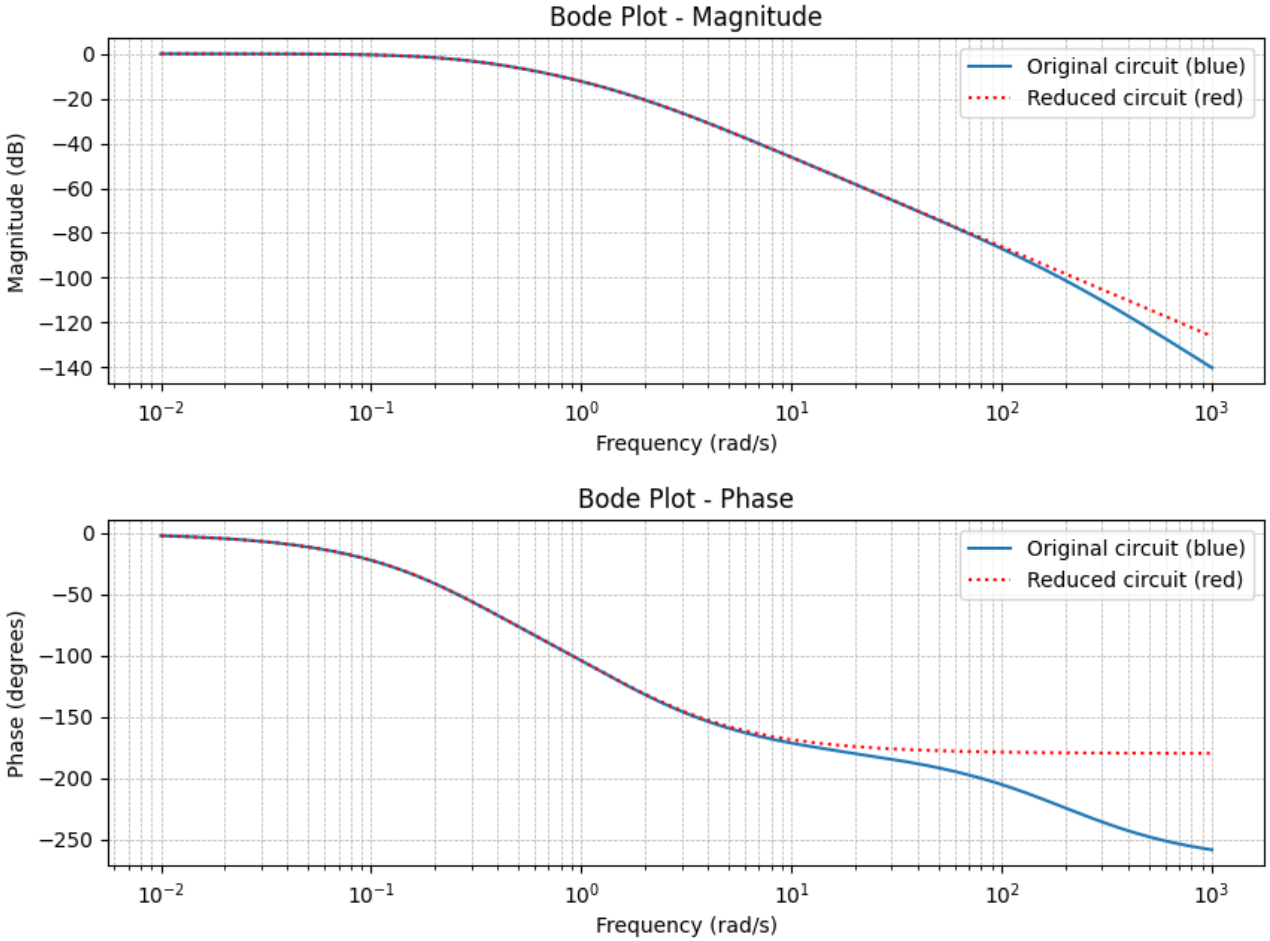
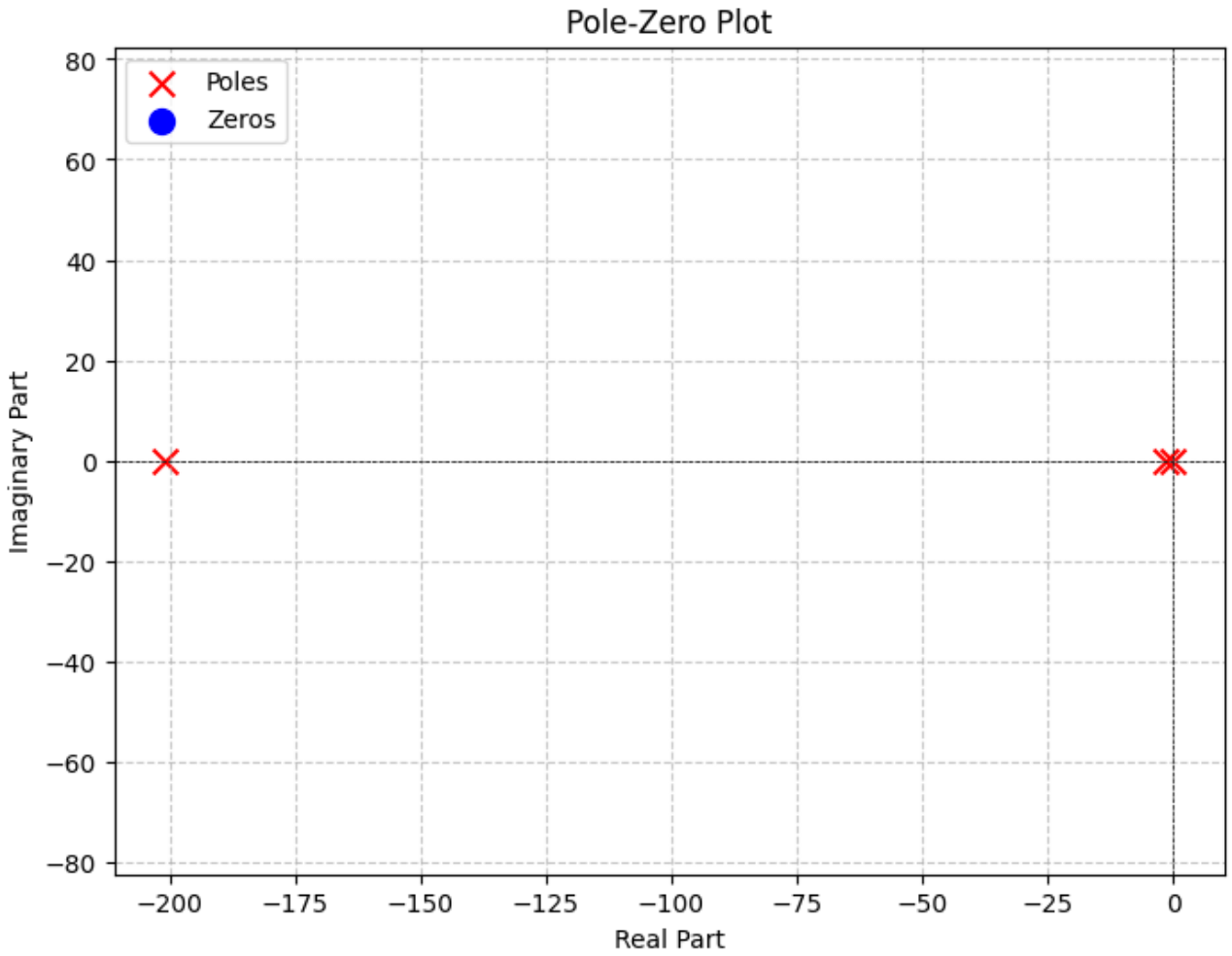
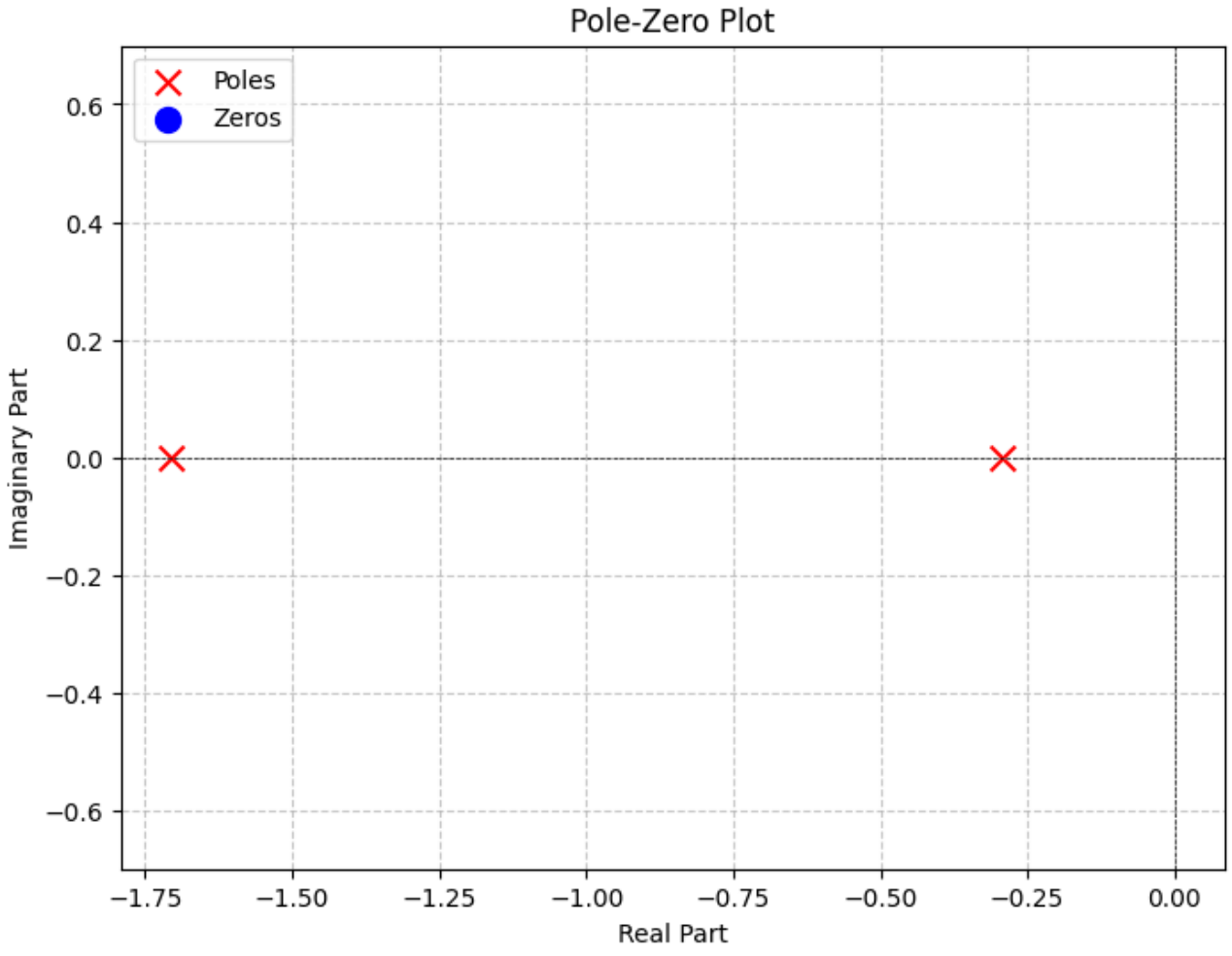


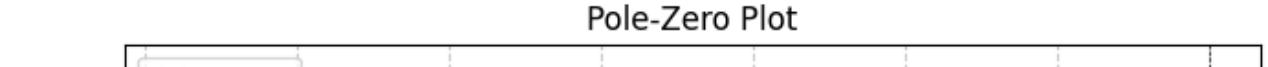
Figure 34: Bode Plots

In the original circuit we have an additional pole at $s = 201$, which is expendable (**?@fig-pole-zero-plot-original**).

In **?@fig-pole-zero-plot-original-without-201** we remove this pole, for better comparison with the reduced circuit (**?@fig-pole-zero-plot-reduced**), where that pole is no longer present, but the others still are.


Pole-Zero Plot
Poles
Zeros
Imaginary Part
Real Part
80
60
40
20
0
−20
−40
−60
−80
−200
−175
−150
−125
−100
−75
−50
−25
0


Pole-Zero Plot
Poles
Zeros
Imaginary Part
Real Part
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.75
−1.50
−1.25
−1.00
−0.75
−0.50
−0.25
0.00


Pole-Zero Plot

9 Appendix

References

Authors, SkyWater PDK. 2020. "SkyWater Open Source PDK." May 2020. https://skywater-pdk.readthedocs.io.

Di Lorenzo, Enrico. 2019. "FasterCap Embedded Help." 2019. https://www.fastfieldsolvers.com/Download/FasterCapHelp.chm.

———. 2023. "The Maxwell Capacitance Matrix: White Paper WP110301. Revision 03." 2023. 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.

———. 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.

———. 2023b. "Full r-c Extraction in Magic (Presentation Video)." https://www.youtube.com/watch?v=4d2mtiEHHeo.

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, EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html.

Sheehan, B. N. 1999. "TICER: Realizable Reduction of Extracted RC Circuits." In *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): 1393–1407. https://doi.org/10.1109/TCAD.2007.891374.