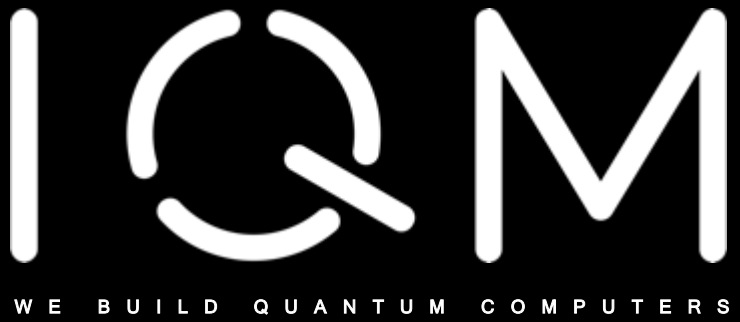


Free Silicon Conference 2023

Superconducting quantum processor design with KQCircuits®

Pavel Smirnov
Software Engineer, Design and Simulation team

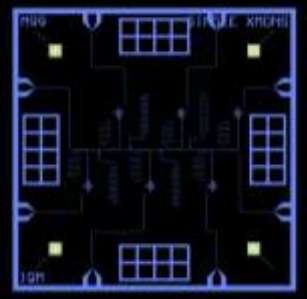
github.com/iqm-finland/KQCircuits

What is KQCircuits?

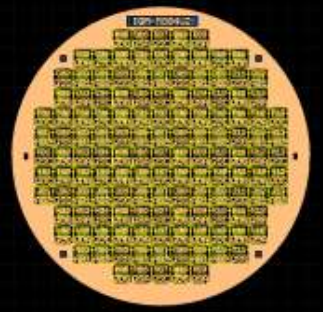
- A plug-in for KLayout (www.klayout.de), SALT package available
- An open-source framework for designing superconducting circuit geometry
  - Multi-layer 2-dimensional -> allows flip chip design
- Libraries of standard, usable Elements and Chips
- Focus on parametrized design enables large scale and reusable designs
- Integrates with finite element simulation tools
- Tools for quality control: design rules, netlist export, FEM simulations

What does KQCircuits provide?

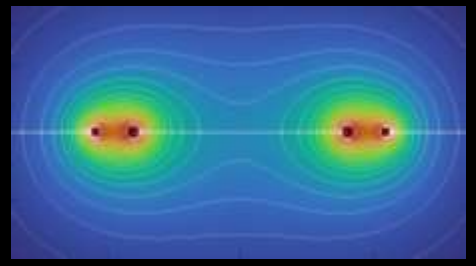
Parametrized Elements



Example Chips



Mask Layout generation

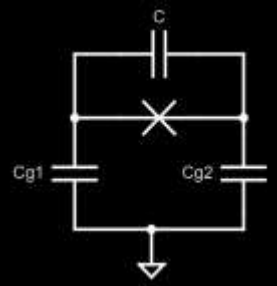


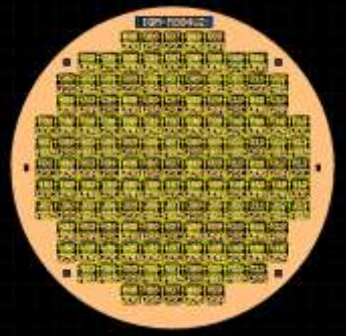
Export to Finite Element simulations
Elmer (open source)
Ansys (HFSS, Q3D)
Sonnet

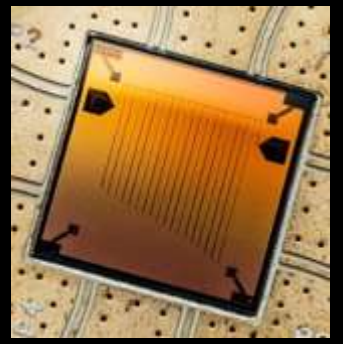
Validation tools
Netlist export
Design rule check

QPU design process

$H(\omega, g)$







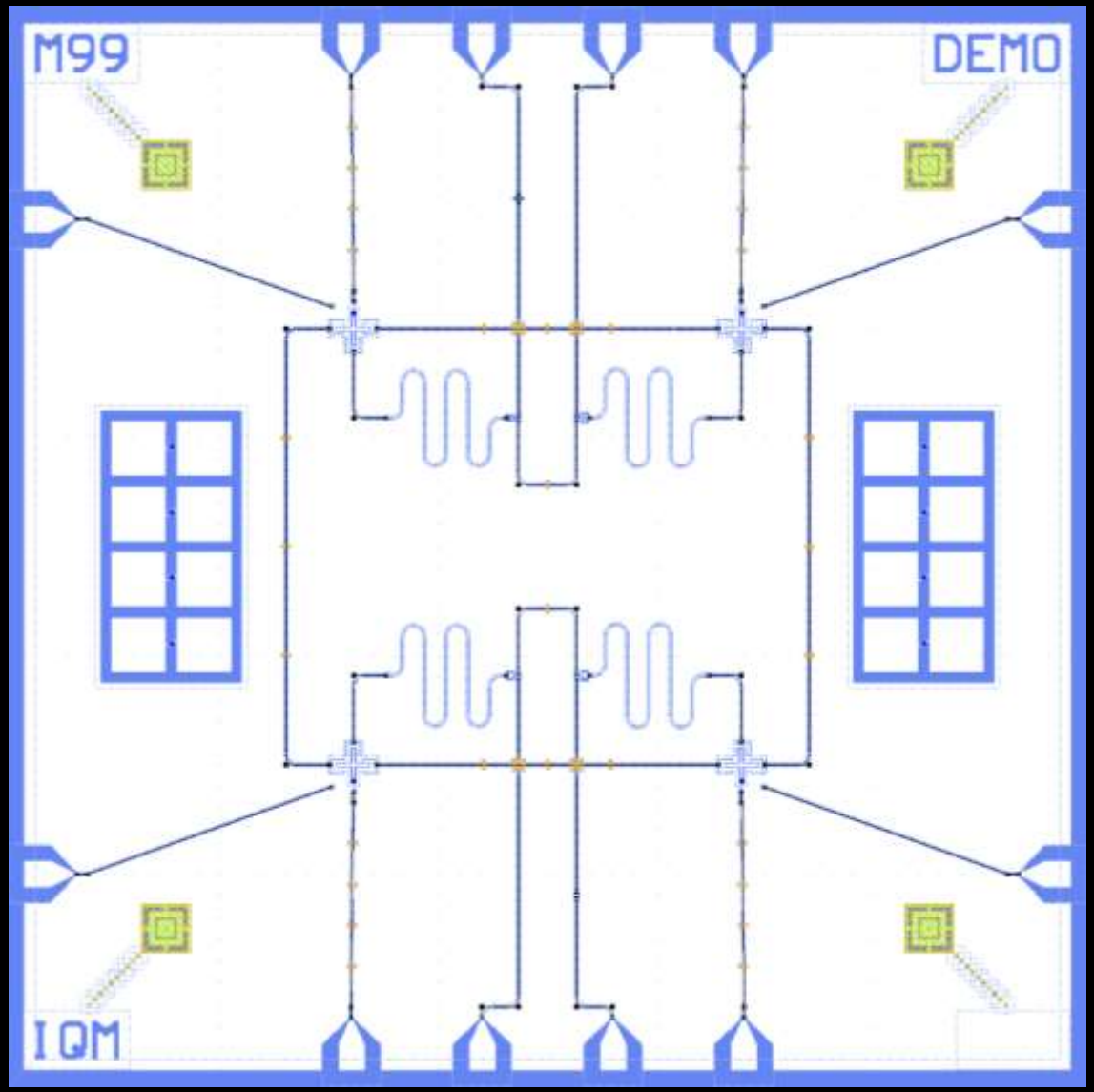
Hamiltonian ⇨ Circuit ⇨ Geometry KQC ⇨ Mask Layout KQC ⇨ Fabrication

Element library
Components FEM
(enabled by KQC)

Validation
Netlist, DRC, Full FEM
(enabled by KQC)

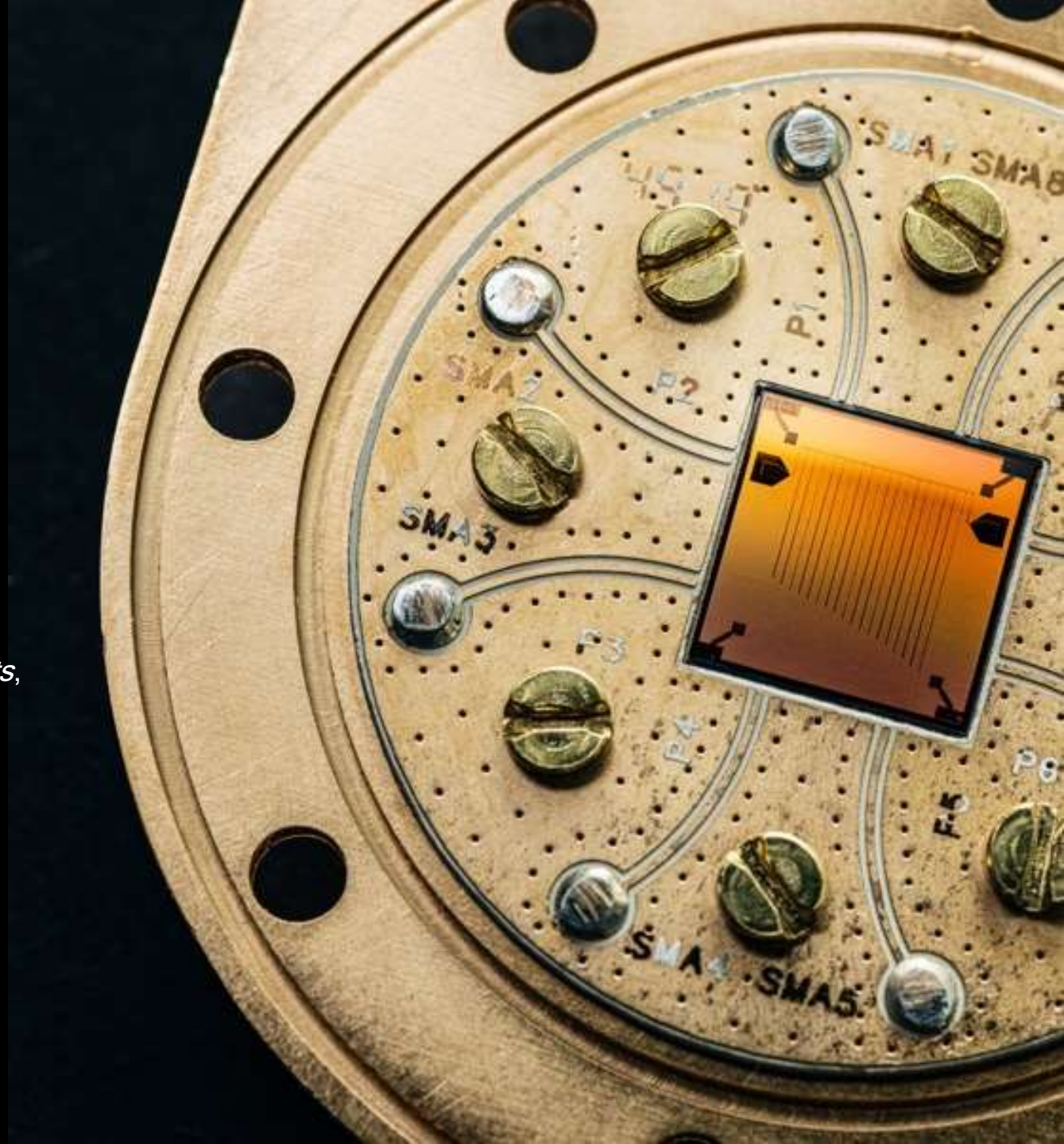
Superconducting Quantum processing unit

- Slab of Si covered by metal film
- Geometry etched to the film
- Consists of elements
  - Qubits
  - Couplers
  - Readout resonators
  - Drivelines and fluxlines
  - Probelines



IQM

Further reading on the principles of quantum chips:

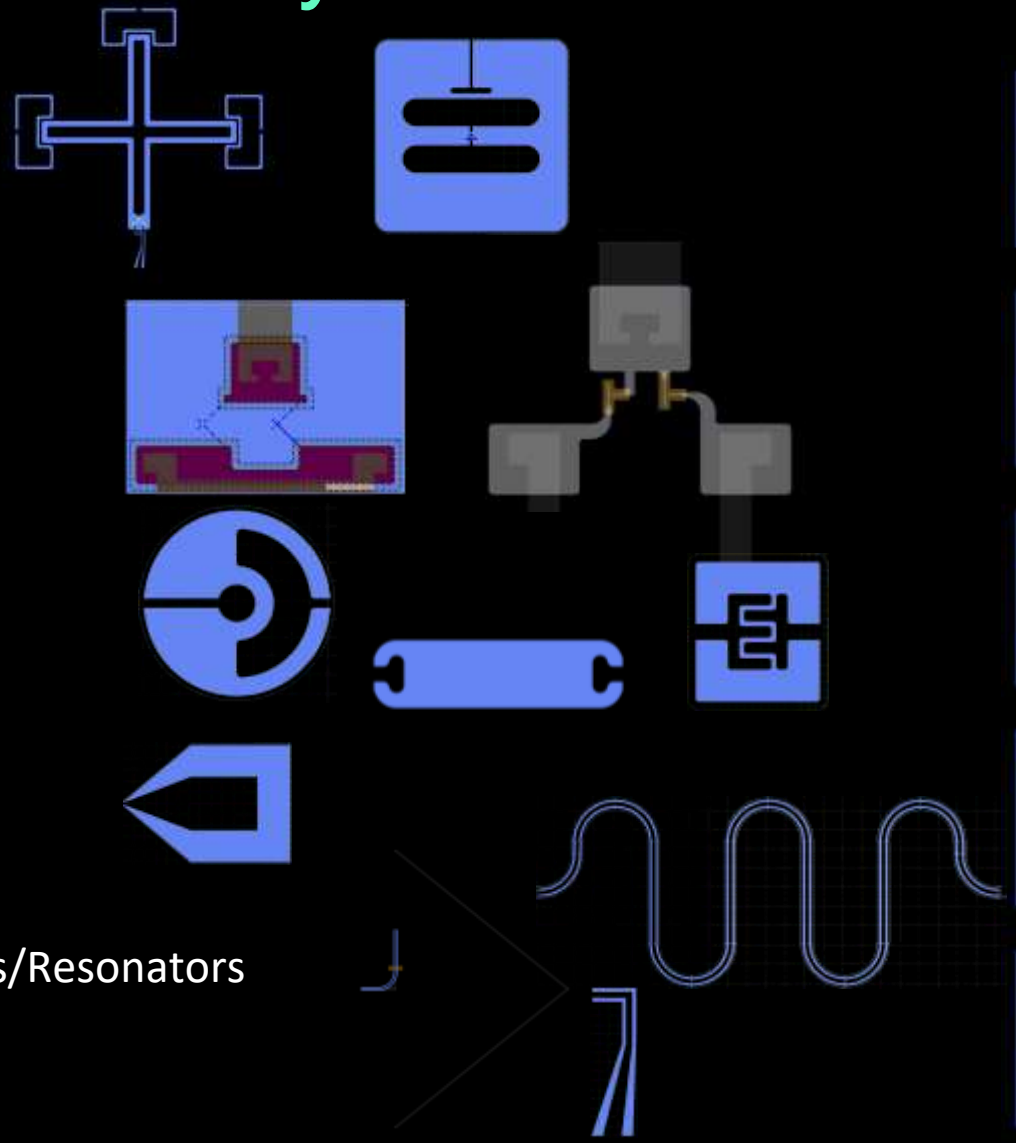


P Krantz et al, *A quantum engineer's guide to superconducting qubits*, Applied Physics Reviews **6**, 021318 (2019)

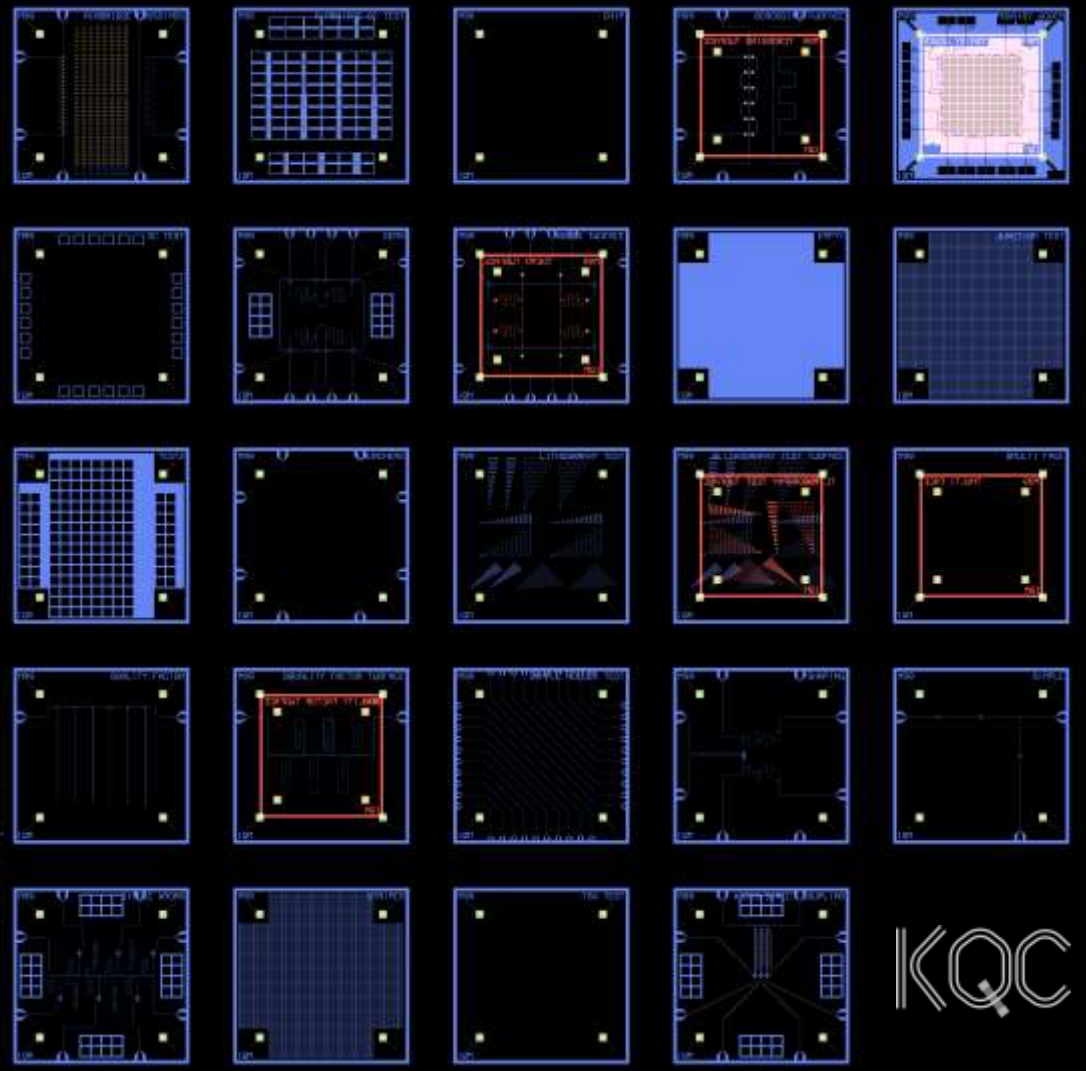
J. Koch et al, *Charge insensitive qubit design derived from the Cooper pair box,* Phys. Rev. A 76, 042319 (2007)

Element Library

- Qubits
- Junctions
- Capacitors
- Launchers
- Waveguides/Resonators



Chip Library



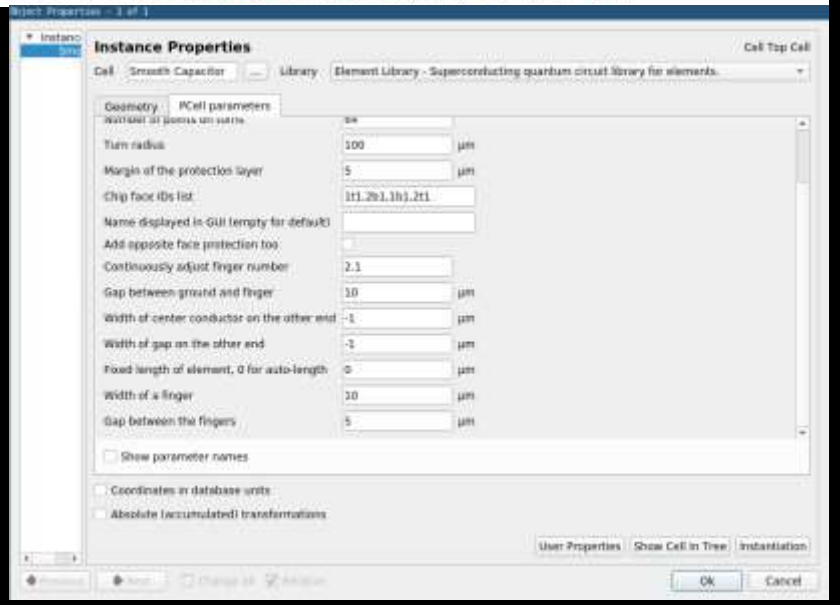
Implement your own parametric elements

```python
@add_parameters_from(FingerCapacitorSquare, "fixed_length", "a2", "b2", finger_width=10, finger_gap=5)
class SmoothCapacitor(Element):
    """The PCell declaration for a smooth finger capacitor.

    SmoothCapacitor is a finger capacitor, which has continuous geometry changes
    through the capacitance range. This leads to continuous capacitance function,
    which enables using capacitor inside numerical optimization methods.

    Capacitance range is achieved by changing single parameter called `finger_control`.
    """

    finger_control = Param(pdt.TypeDouble, "Continuously adjust finger number", 2.1,
                           docstring="Parameter for capacitor growth (related to number of fingers per side)")
    ground_gap = Param(pdt.TypeDouble, "Gap between ground and finger", 10, unit="µm")
```



Tools for composing complex chip layouts

- Composite waveguides
  - Inline elements (capacitors, splitters, …)
  - Adjustable-length segments
- Hybrid GUI + code workflow

Sketch layout in GUI

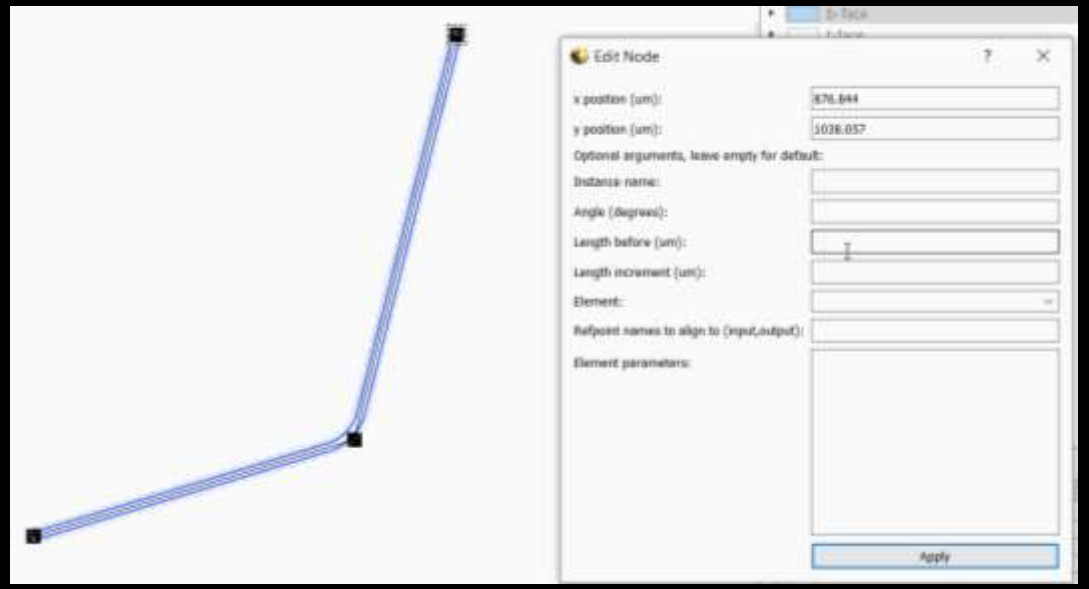
↓

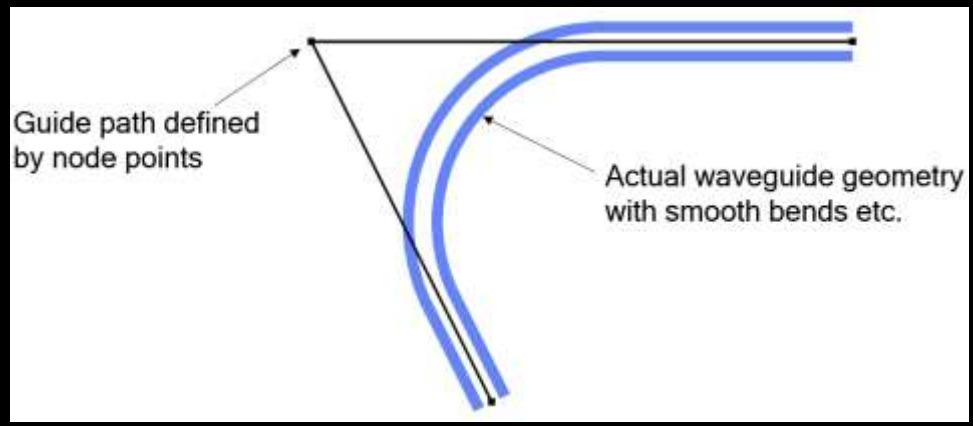
Generate & edit Python code

View / edit / route in GUI

↓

Export & verification

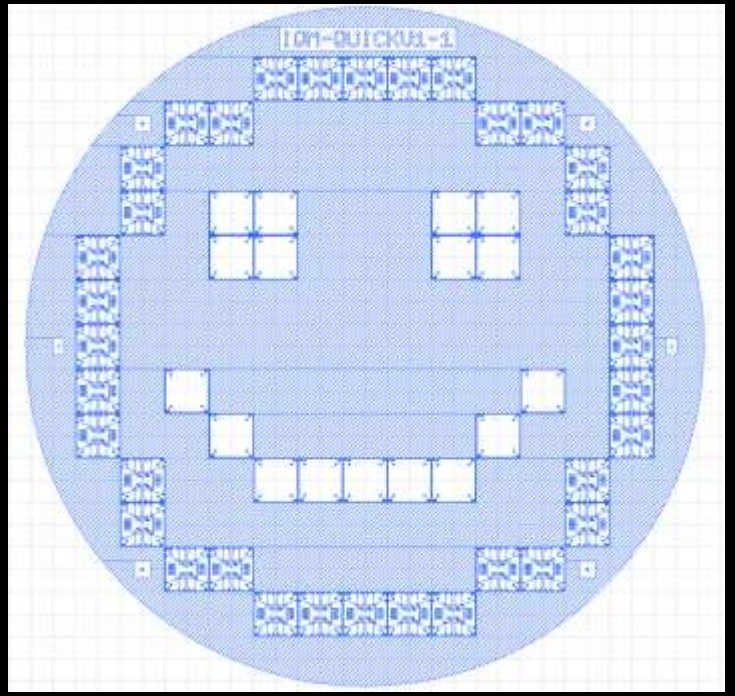




Export masks

```
qdemo.add_mask_layout([
    ["---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---"],
    ["---", "---", "---", "---", "---", "DE1", "DE1", "DE1", "DE1", "DE1", "---", "---", "---", "---", "---"],
    ["---", "---", "---", "DE1", "DE1", "---", "---", "---", "---", "---", "DE1", "DE1", "---", "---", "---"],
    ["---", "---", "DE1", "---", "---", "---", "---", "---", "---", "---", "---", "---", "DE1", "---", "---"],
    ["---", "---", "DE1", "---", "CH1", "CH1", "---", "---", "---", "CH1", "CH1", "---", "DE1", "---", "---"],
    ["---", "DE1", "---", "---", "CH1", "CH1", "---", "---", "---", "CH1", "CH1", "---", "---", "DE1", "---"],
    ["---", "DE1", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "DE1", "---"],
    ["---", "DE1", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "DE1", "---"],
    ["---", "DE1", "---", "CH1", "---", "---", "---", "---", "---", "---", "---", "CH1", "---", "DE1", "---"],
    ["---", "DE1", "---", "---", "CH1", "---", "---", "---", "---", "---", "CH1", "---", "---", "DE1", "---"],
    ["---", "---", "DE1", "---", "---", "CH1", "CH1", "CH1", "CH1", "CH1", "---", "---", "DE1", "---", "---"],
    ["---", "---", "DE1", "---", "---", "---", "---", "---", "---", "---", "---", "---", "DE1", "---", "---"],
    ["---", "---", "---", "DE1", "DE1", "---", "---", "---", "---", "---", "DE1", "DE1", "---", "---", "---"],
    ["---", "---", "---", "---", "---", "DE1", "DE1", "DE1", "DE1", "DE1", "---", "---", "---", "---", "---"],
    ["---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---", "---"],
], "ltl", layers_to_mask=layers_to_mask)
```

- Merge metallization layers
- Generate masks for wafer-scale lithography
- Adds alignment markers
- Configurable which layers are exported to a mask
- Optionally invert layer polarity
- Export mask files as well as individual chips and related files



Verification

- Netlist verification
  - Target: detect technical errors such as wrong segment length
- KLayout’s Design Rule Check (DRC)
  - Checks spacing of structures in critical layers
  - Checks crossing waveguides
  - Target: detect simple geometry errors
- Custom FEM verification
  - Estimate e.g. crosstalk effects
  - KQC export + manual simulation work

Simulations

[1] Z. K. Minev et al., 'Energy-participation quantization of Josephson circuits', npj Quantum Inf, vol. 7, no. 1, Art. no. 1, Aug. 2021
[2] P. Råback et al., 'Elmer'. CSC - IT Center for Science, Available: github.com/ElmerCSC/elmerfem

- Ansys
  - Q3D – capacitance extraction using method of moments
  - HFSS – S matrix extraction
  - HFSS Eigenmode with ideal inductors modelling junctions/SQUIDs
  - pyEPR support [1]
- Elmer (Open source FEM solver by CSC Finland) [2]
  - 3D capacitance using FEM
  - 2D cross-sections in Elmer
    - Kinetic inductance for some penetration depth
    - Energy participation ratio (EPR)
- FEM component in KQC workflow quite replaceable with other third-party simulator software as we learn about them (e.g. Sonnet, AWS Palace)

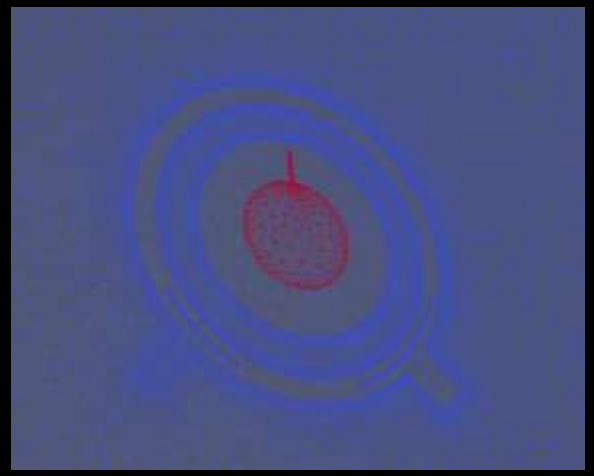
https://www.ansys.com/

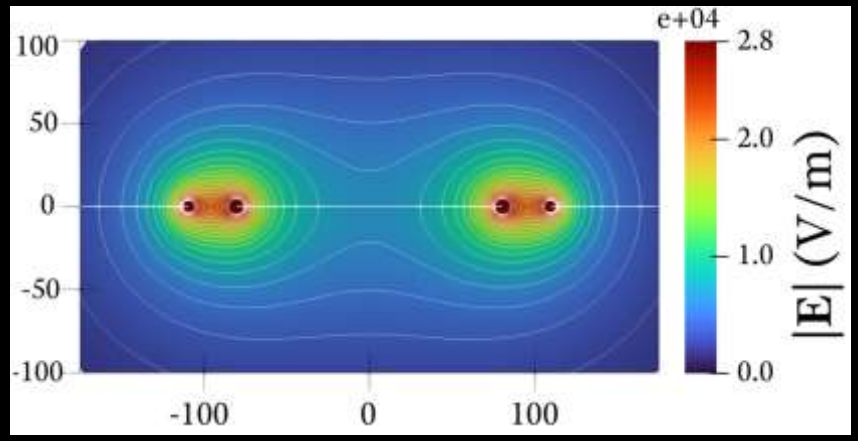
https://www.csc.fi/fi/web/elmer

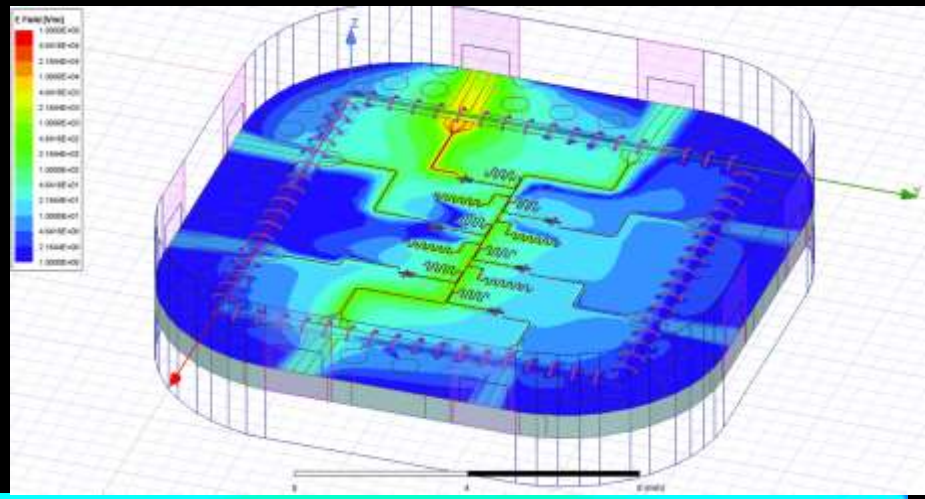
https://www.sonnetsoftware.com/

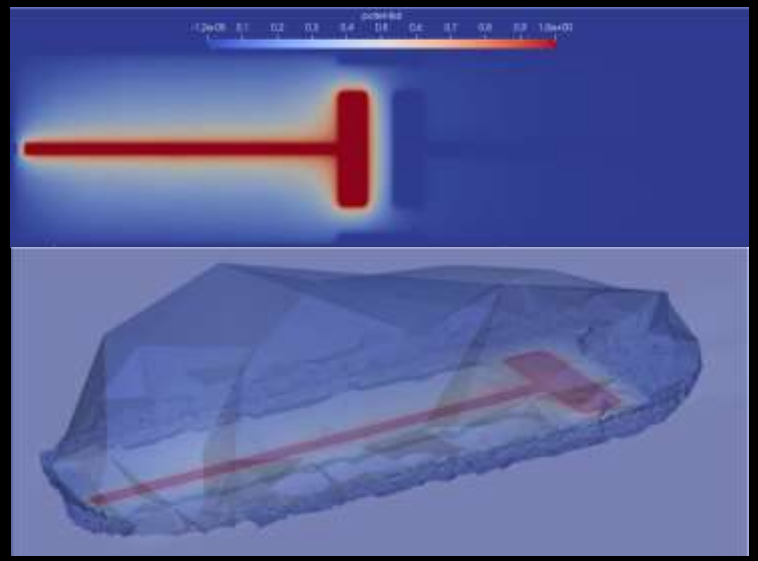
https://awslabs.github.io/palace/stable/

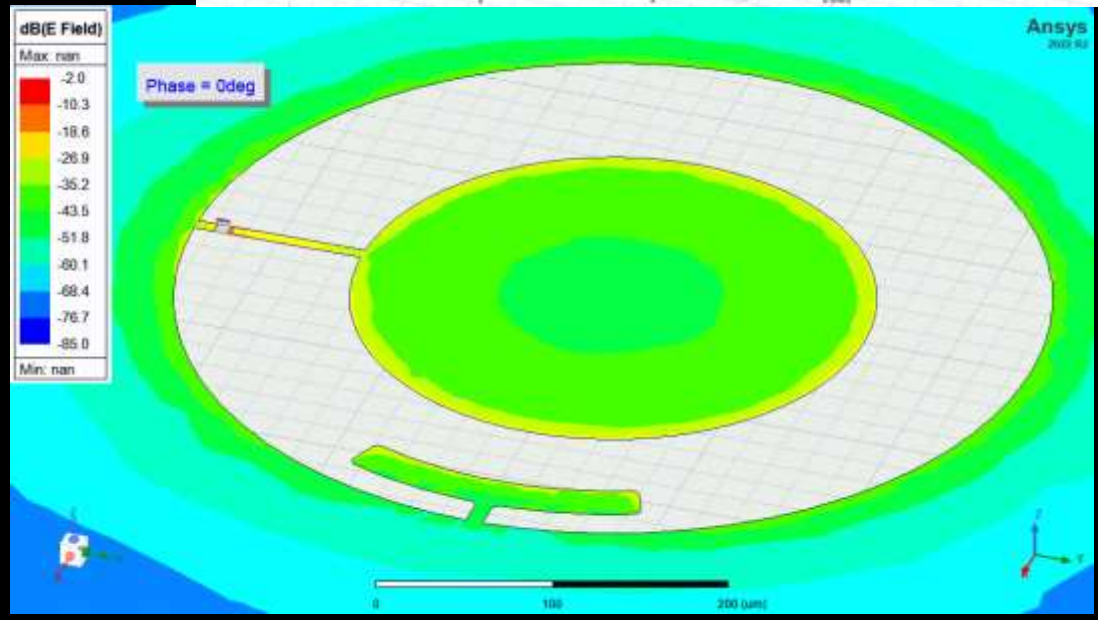
Electromagnetic simulations





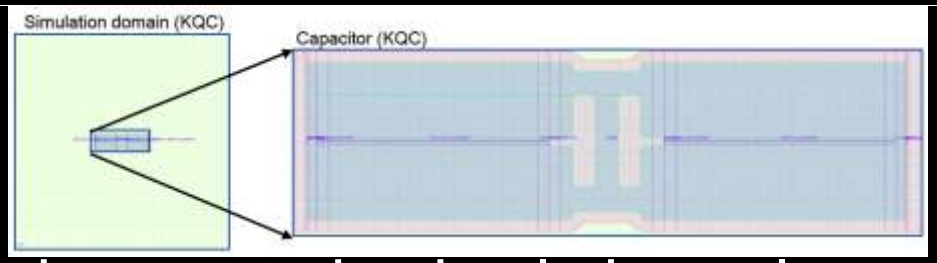


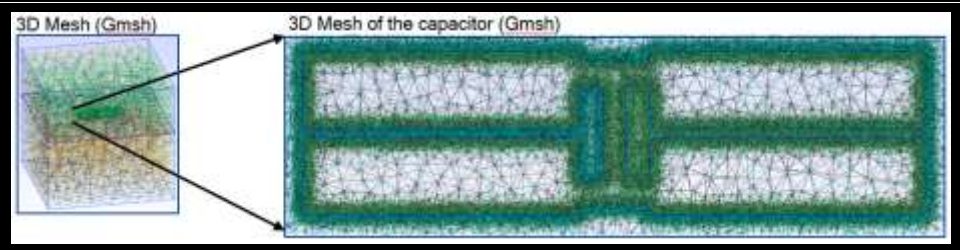




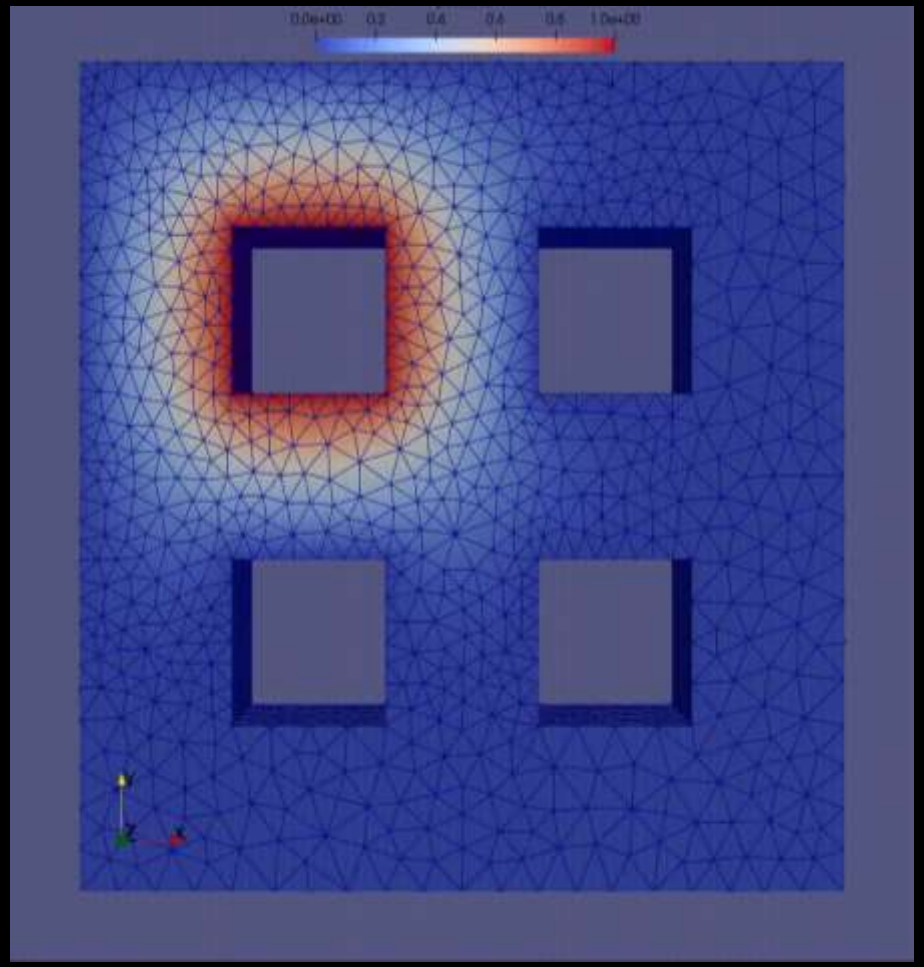
IQM

Meshing





- Divides space into a mesh, simulation value sampled at each mesh node
- Regions where simulation value changes a lot (higher gradient) should have more dense meshing
- Can be handcrafted (Gmsh), but adaptive meshing reduces trial-and-error
- Ansys has adaptive meshing, Elmer parallelised adaptive meshing coming soon



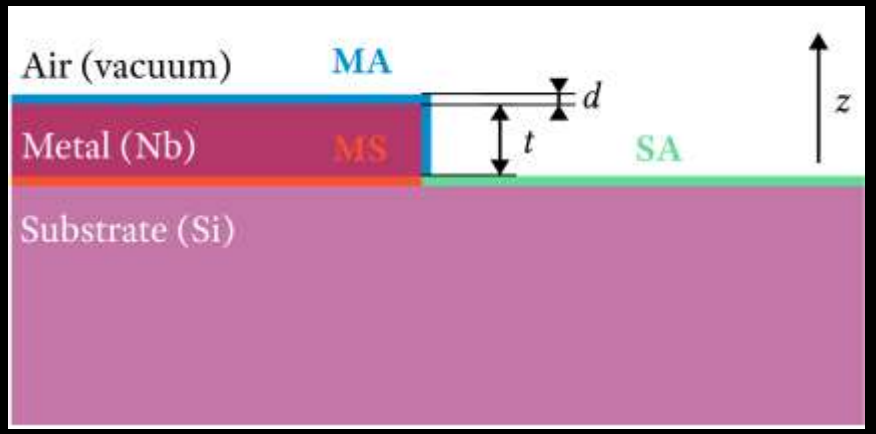
Gmsh

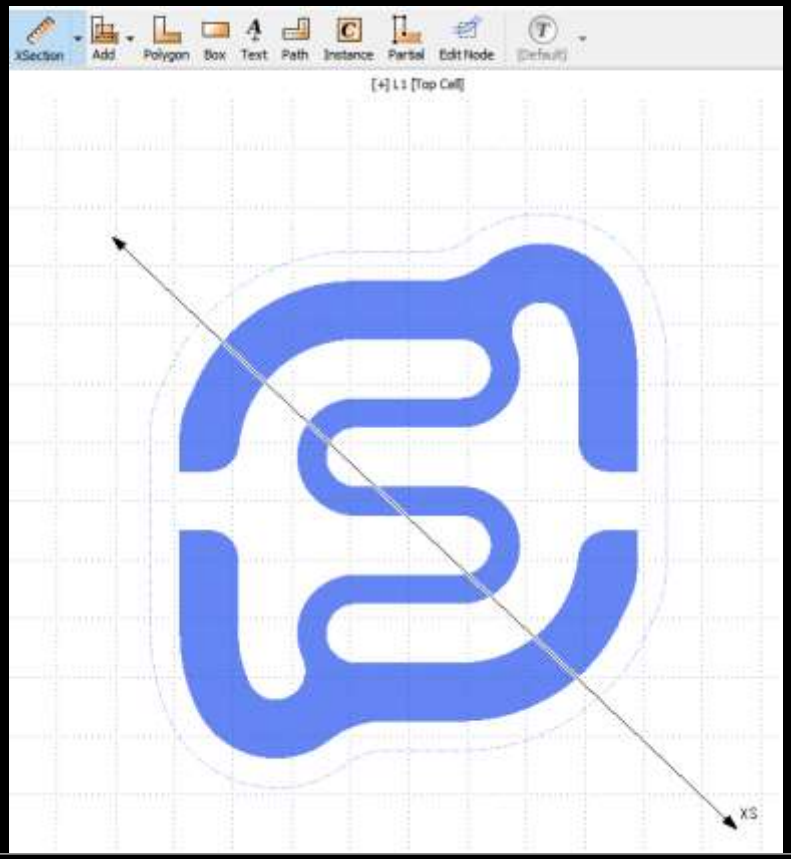
https://gmsh.info/

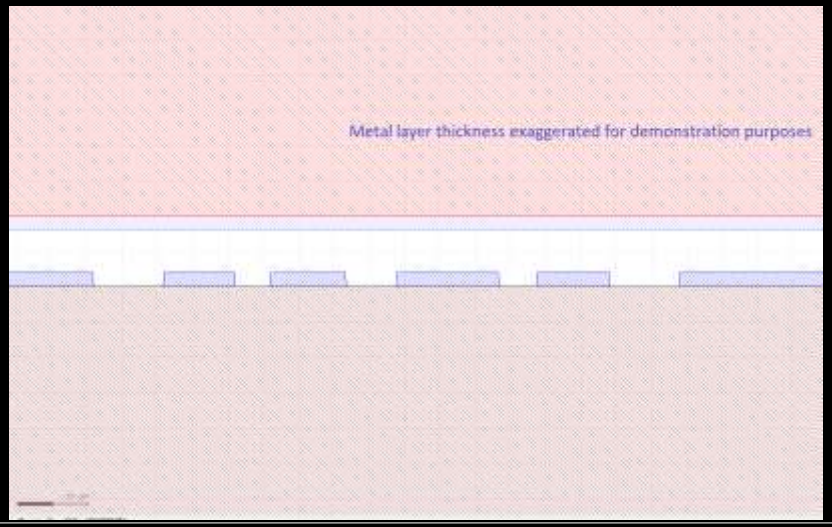
https://wiki.f-si.org/index.php?title=KLayout_XSection_tool_-_Deep_insights_or_nonsense_in_colors%3F

Using XSection to study energy participation ratio (EPR)

Cross-section of planar geometry with lossy interfaces MA, MS & SA







IQM

Thank you!