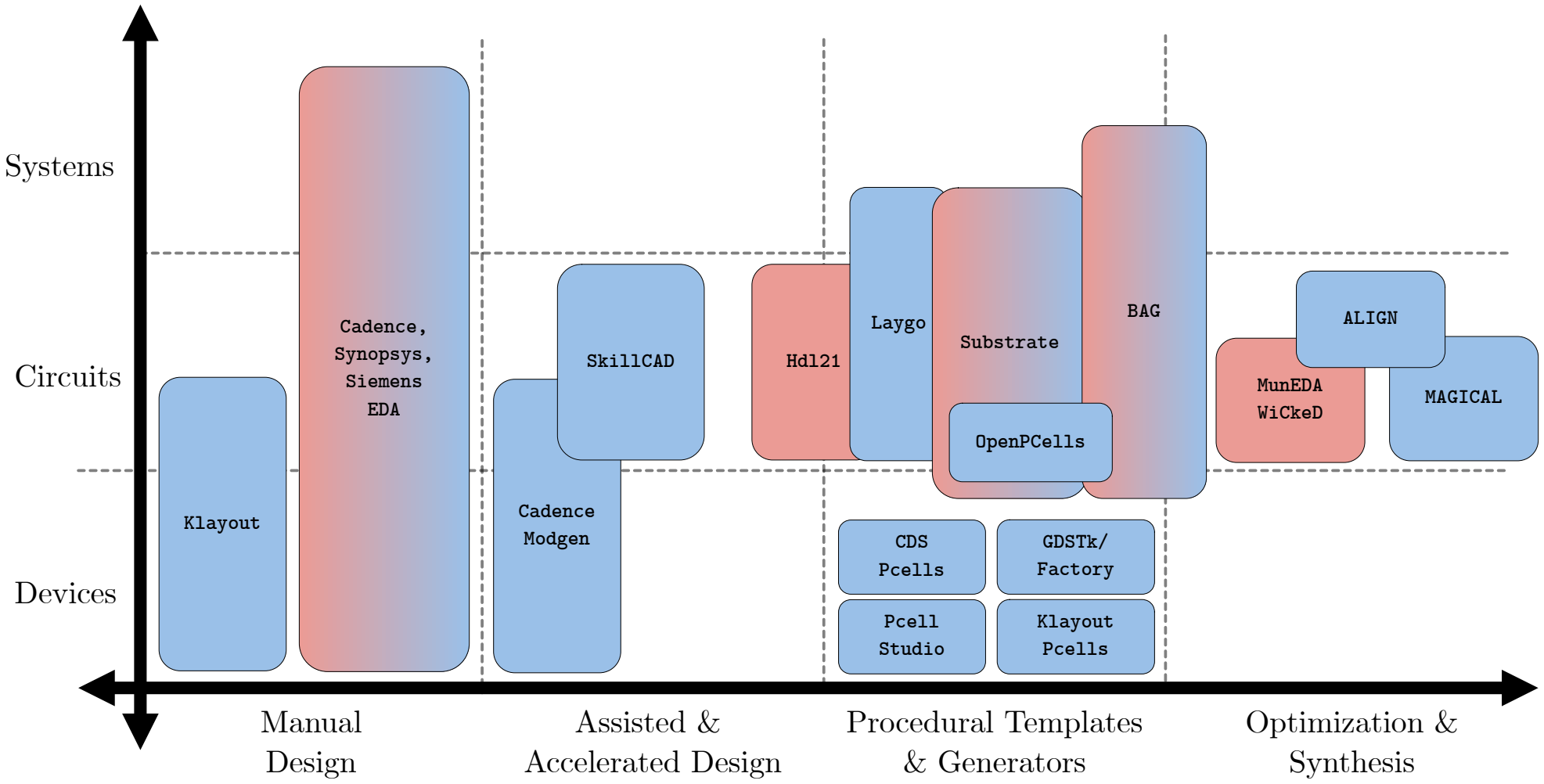
Mixed-signal parameterized cells in general-purpose programming languages

Why chip design should be software-driven

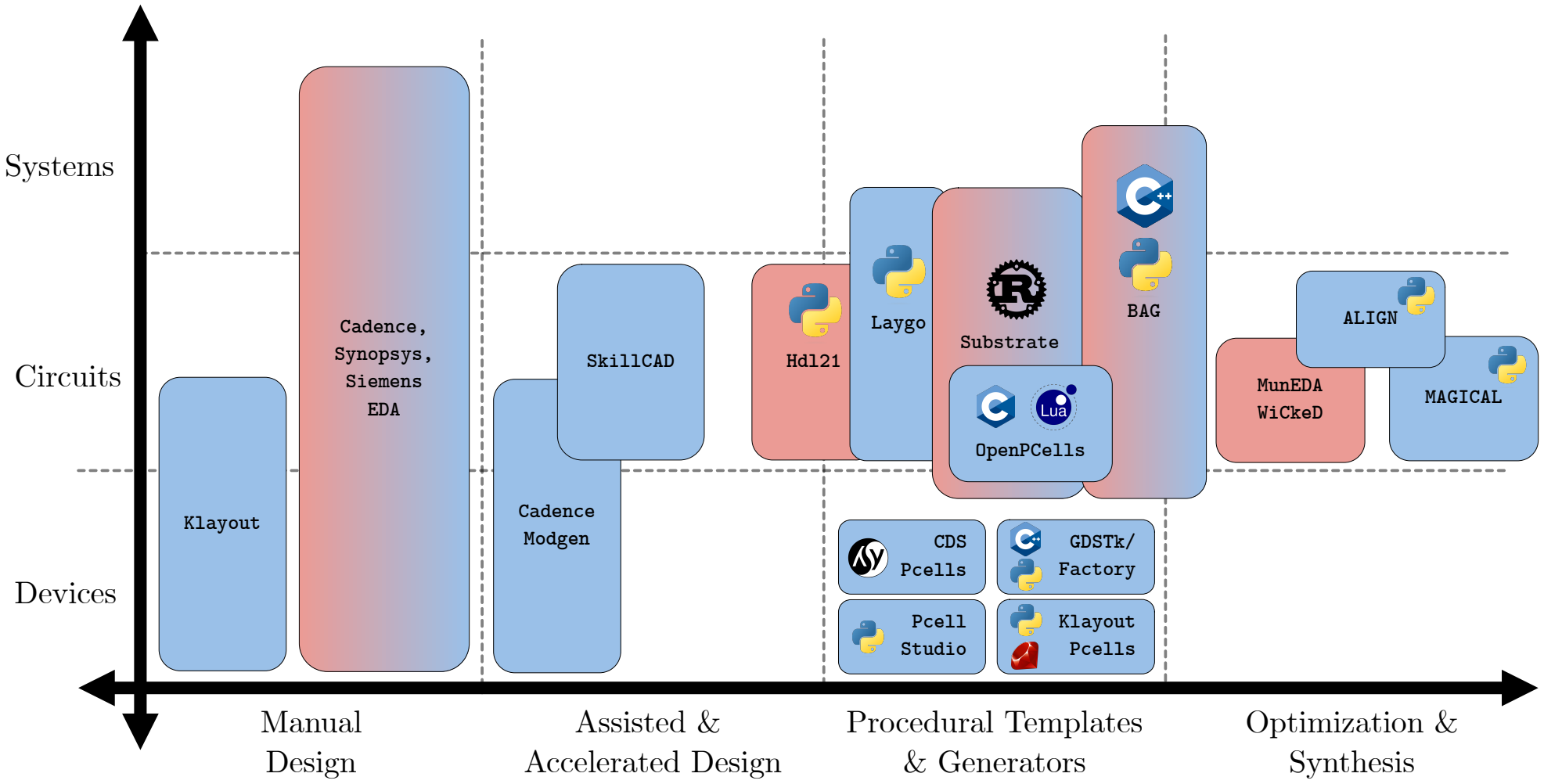
Kennedy Caisley
18 March, 2024

UNIVERSITÄT BONN
FTD

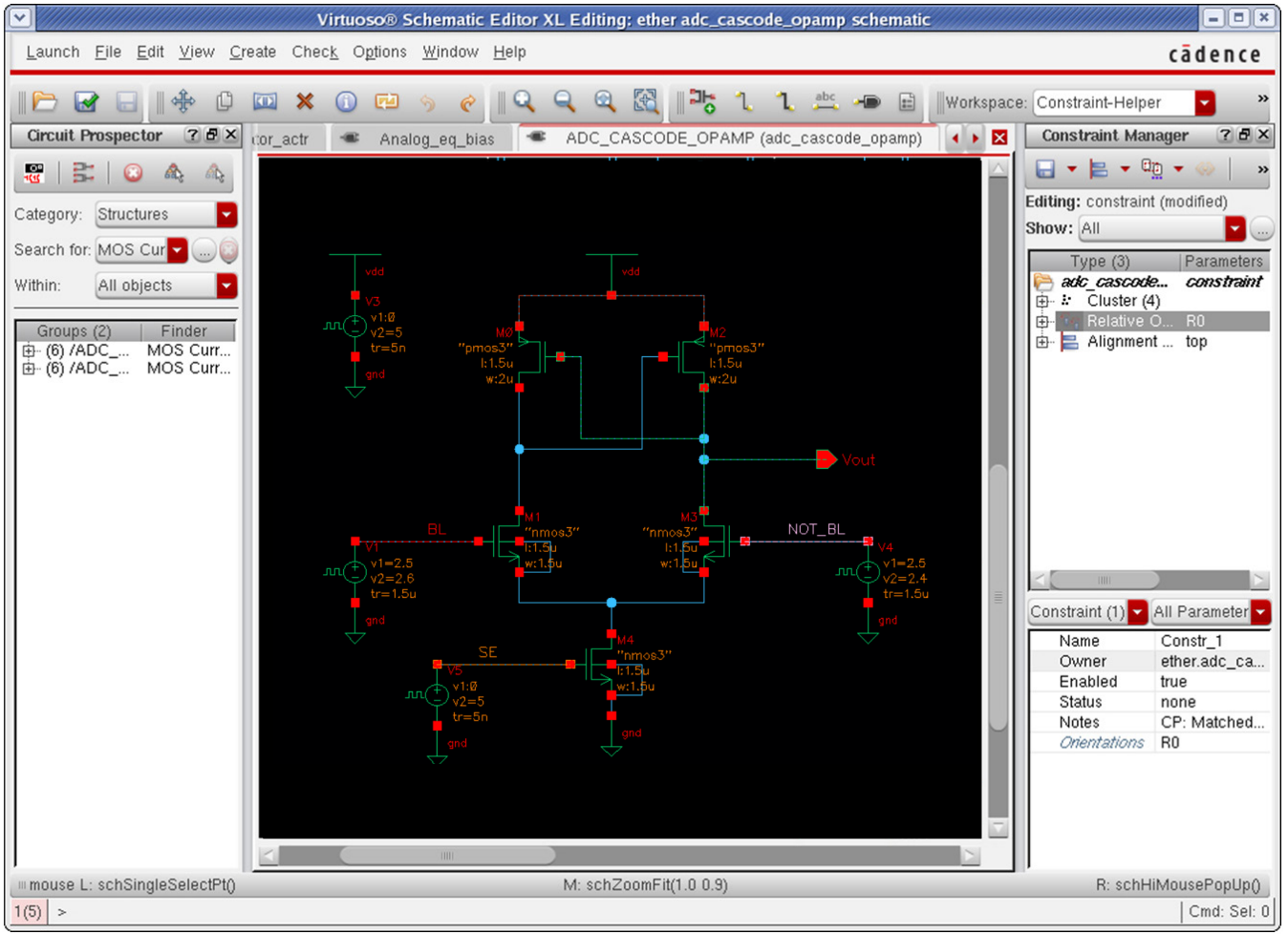
Schematic and Layout Design Tools



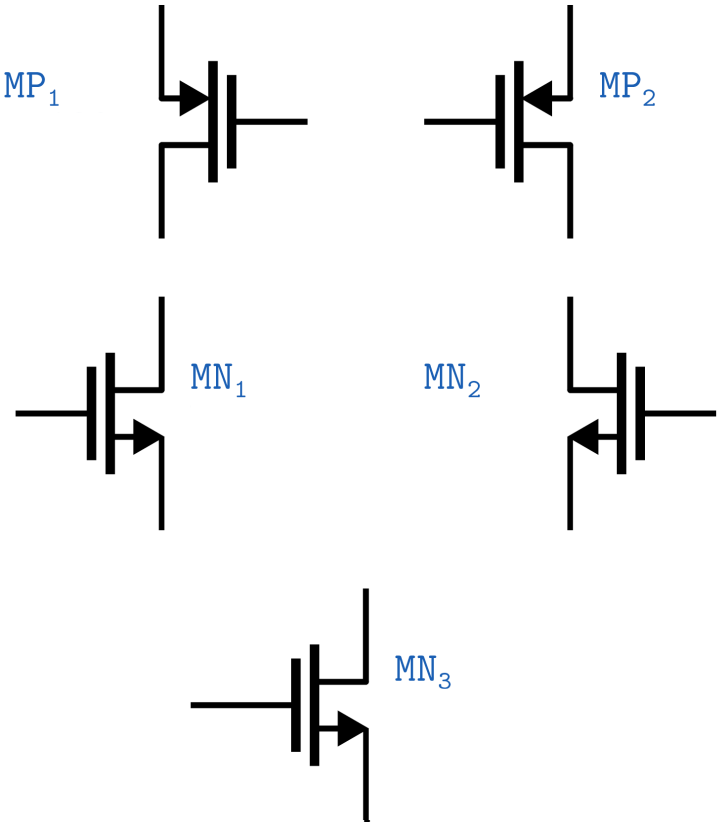
Schematic and Layout Design Tools



Let's start with schematics.

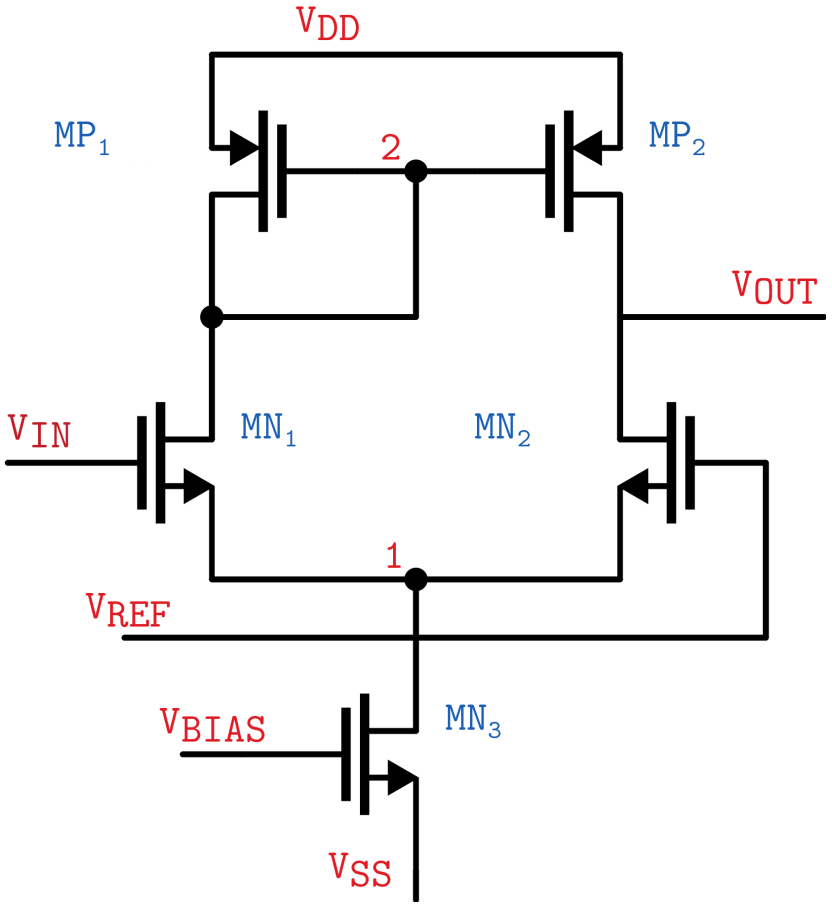


What do schematics actually represent?



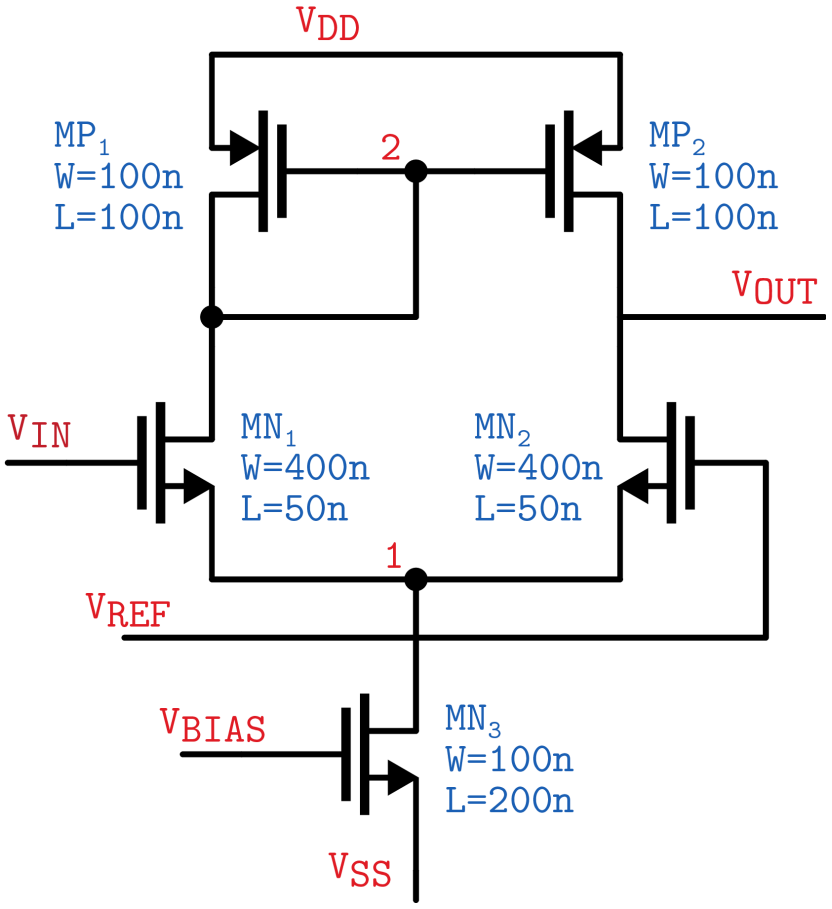

```
* Amplifier netlist *
MP1
MP2
MN1
MN2
MN3
```

What do schematics actually represent?




```
* Amplifier netlist *
MP1 2 2 vdd
MP2 2 vout vdd
MN1 vin 2 1
MN2 vref vout 1
MN3 vbias 1 vss
```

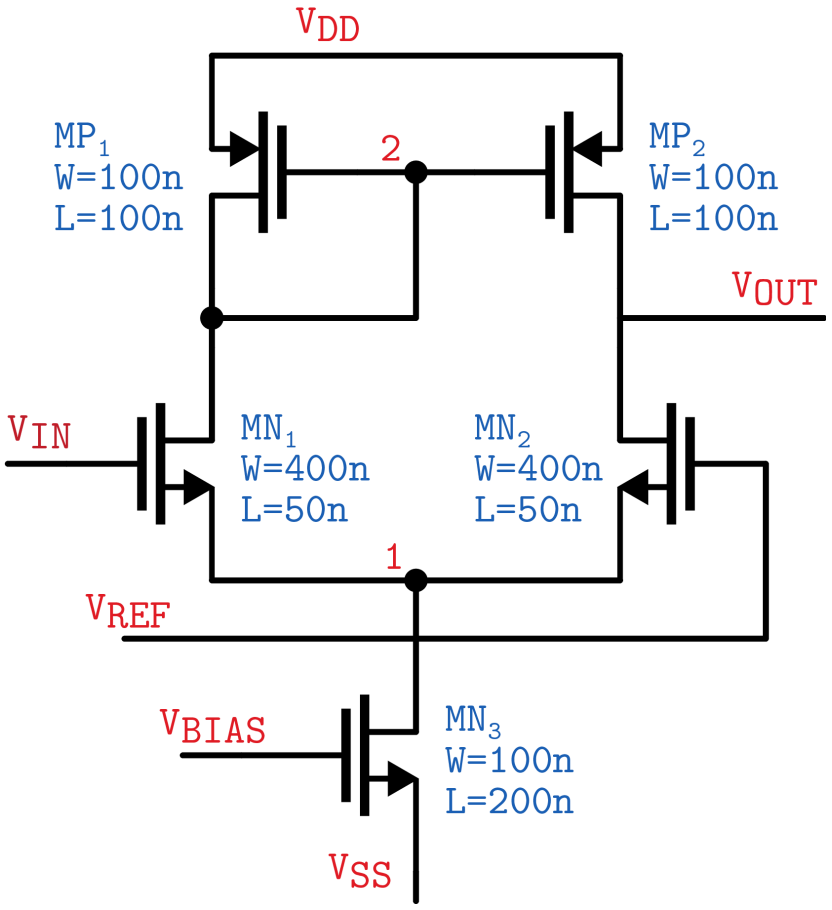
What do schematics actually represent?




```
* Amplifier netlist *
MP1 2 2 vdd     w=100n l=100n
MP2 2 vout vdd  w=100n l=100n
MN1 vin 2 1     w=400n l=50n
MN2 vref vout 1 w=400n l=50n
MN3 vbias 1 vss w=100n l=200n
```

We've built a system of DEQs w/ compact models!

Some issues & observations



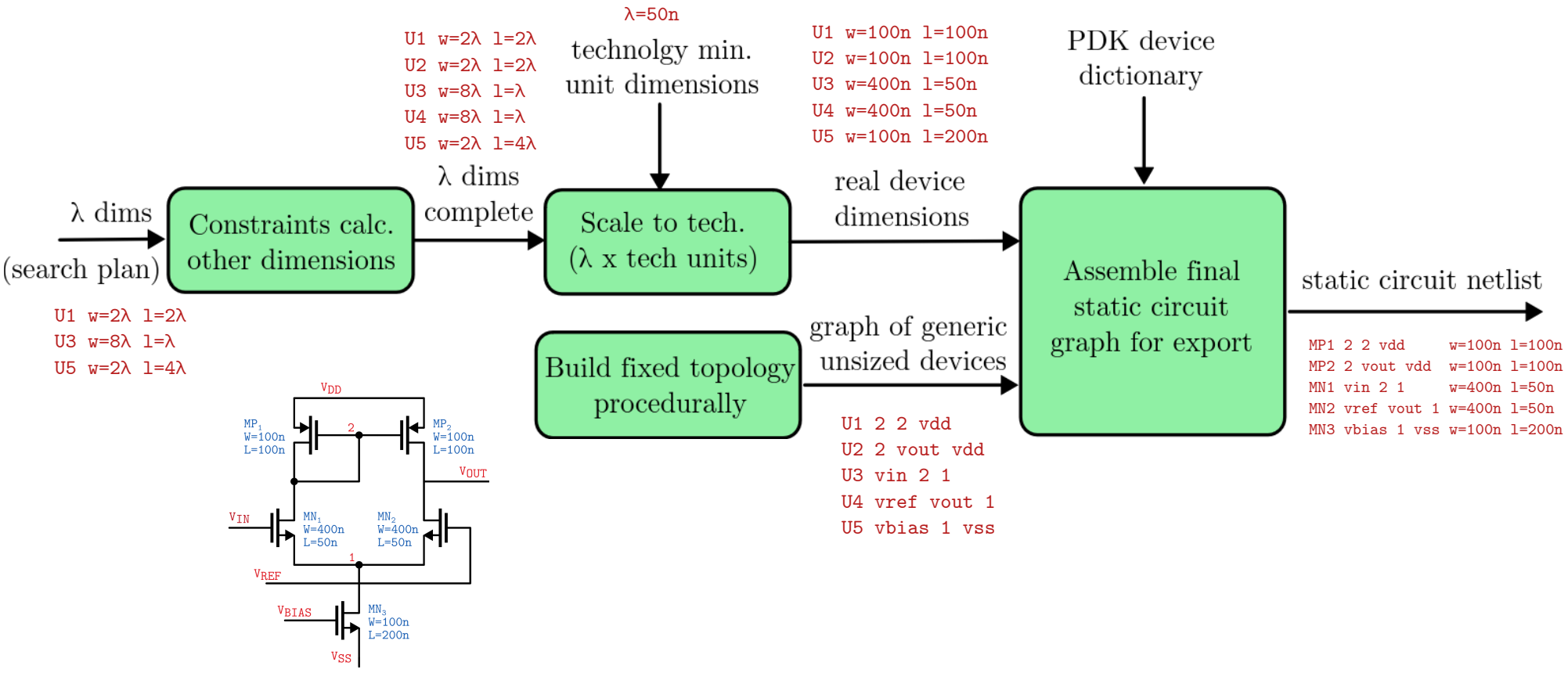
Structural info != performance. Must iterate & simulate

```
* Amplifier netlist *
MP1 2 2 vdd      w=100n l=100n
MP2 2 vout vdd   w=100n l=100n
MN1 vin 2 1      w=400n l=50n
MN2 vref vout 1  w=400n l=50n
MN3 vbias 1 vss  w=100n l=200n
```

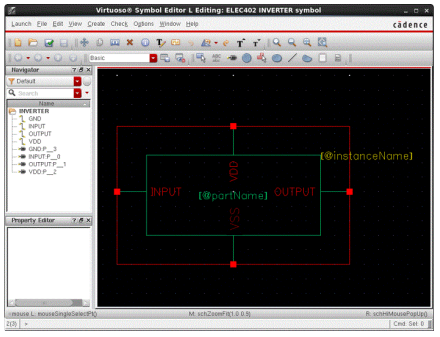
Often a couple topologies to try

But sizing choice is enormous

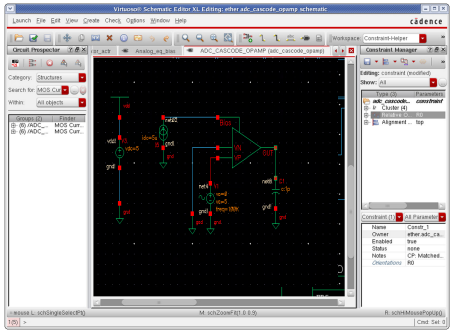
What if we express this in code?



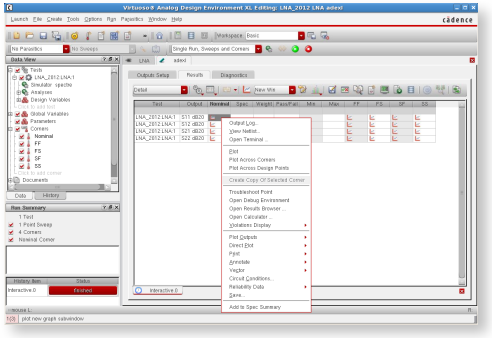
Testbenches, Simulation, & Analysis



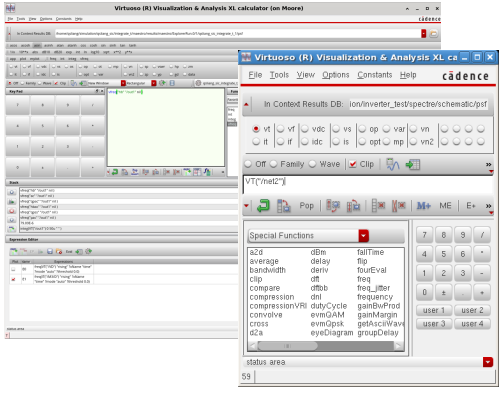
1. Create symbol



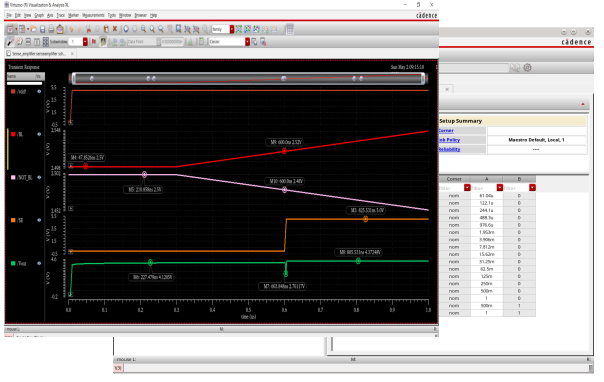
2. Instantiate DUT
3. Wrap with test sources/loads/mocks



4. Simulation runs / options
5. Sweeps and variables
6. Test points

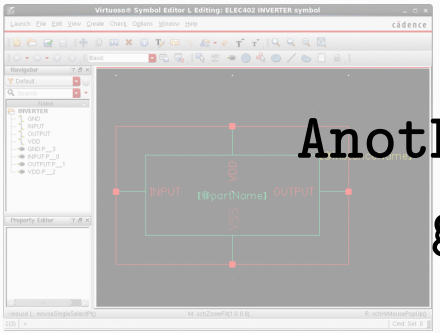


7. Post analysis of outputs



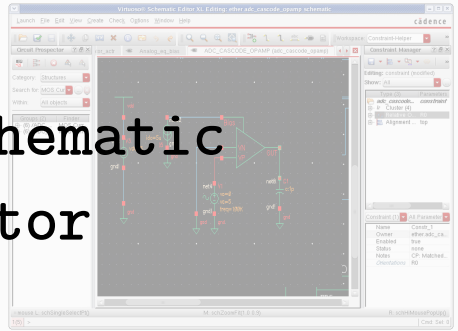
7. Visualization

Testing with code

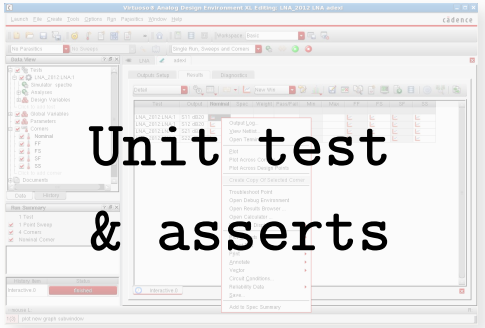


1. Create symbol

Another schematic generator

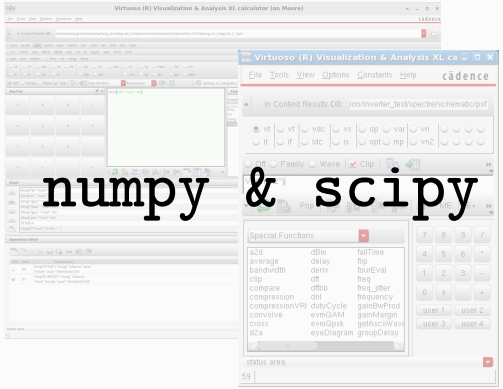


2. Instantiate DUT
3. Wrap with test sources/loads/mocks



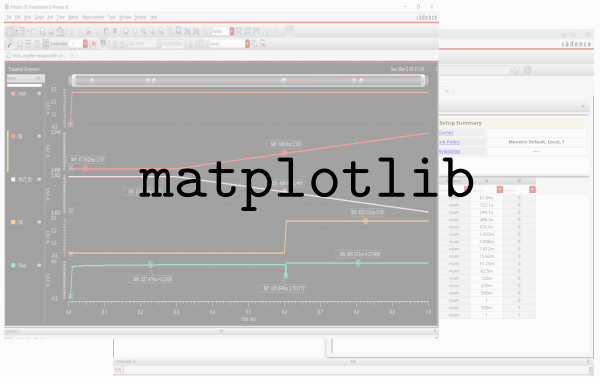
Unit test & asserts

4. Simulation runs / options
5. Sweeps and variables
6. Test points



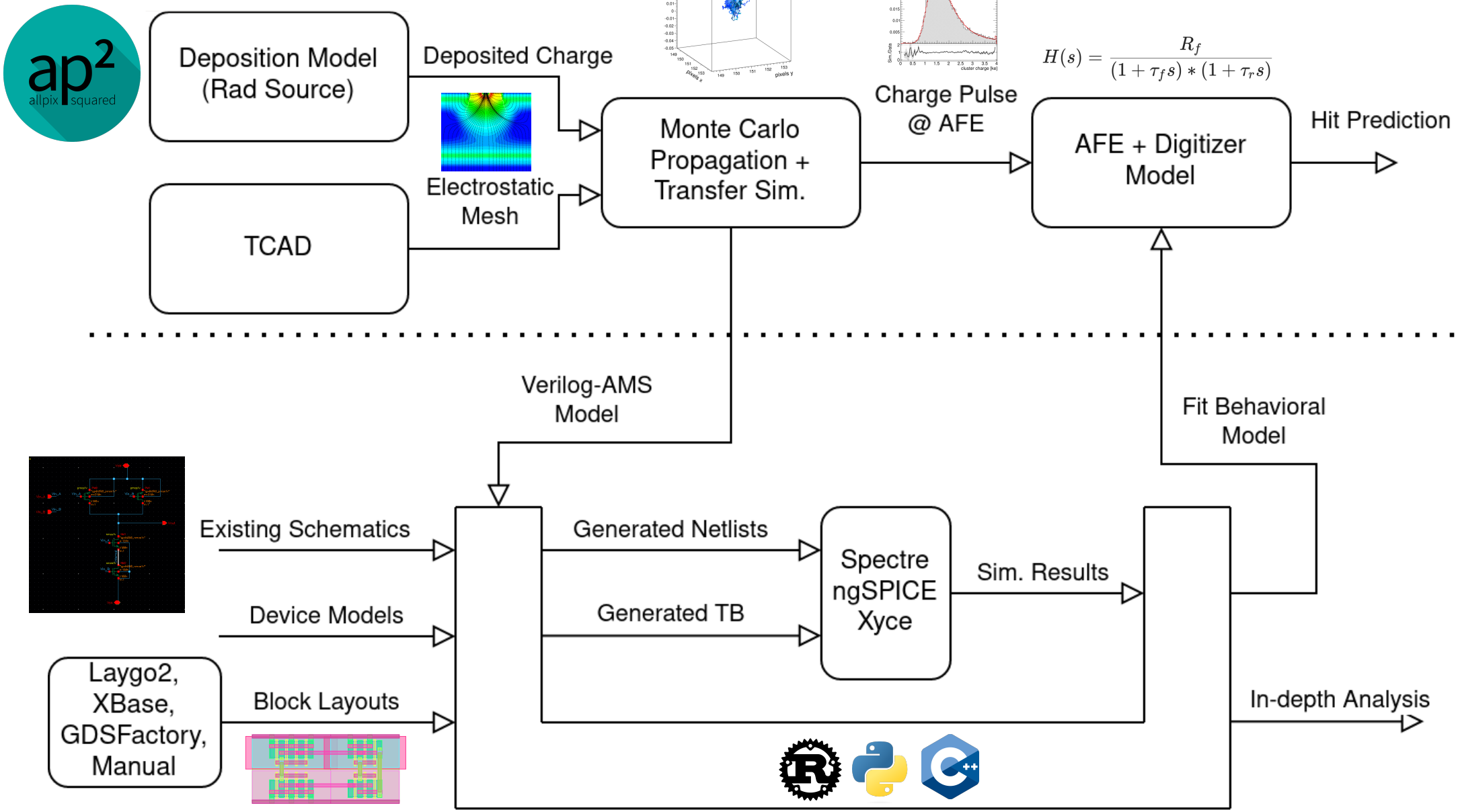
numpy & scipy

7. Post analysis of outputs

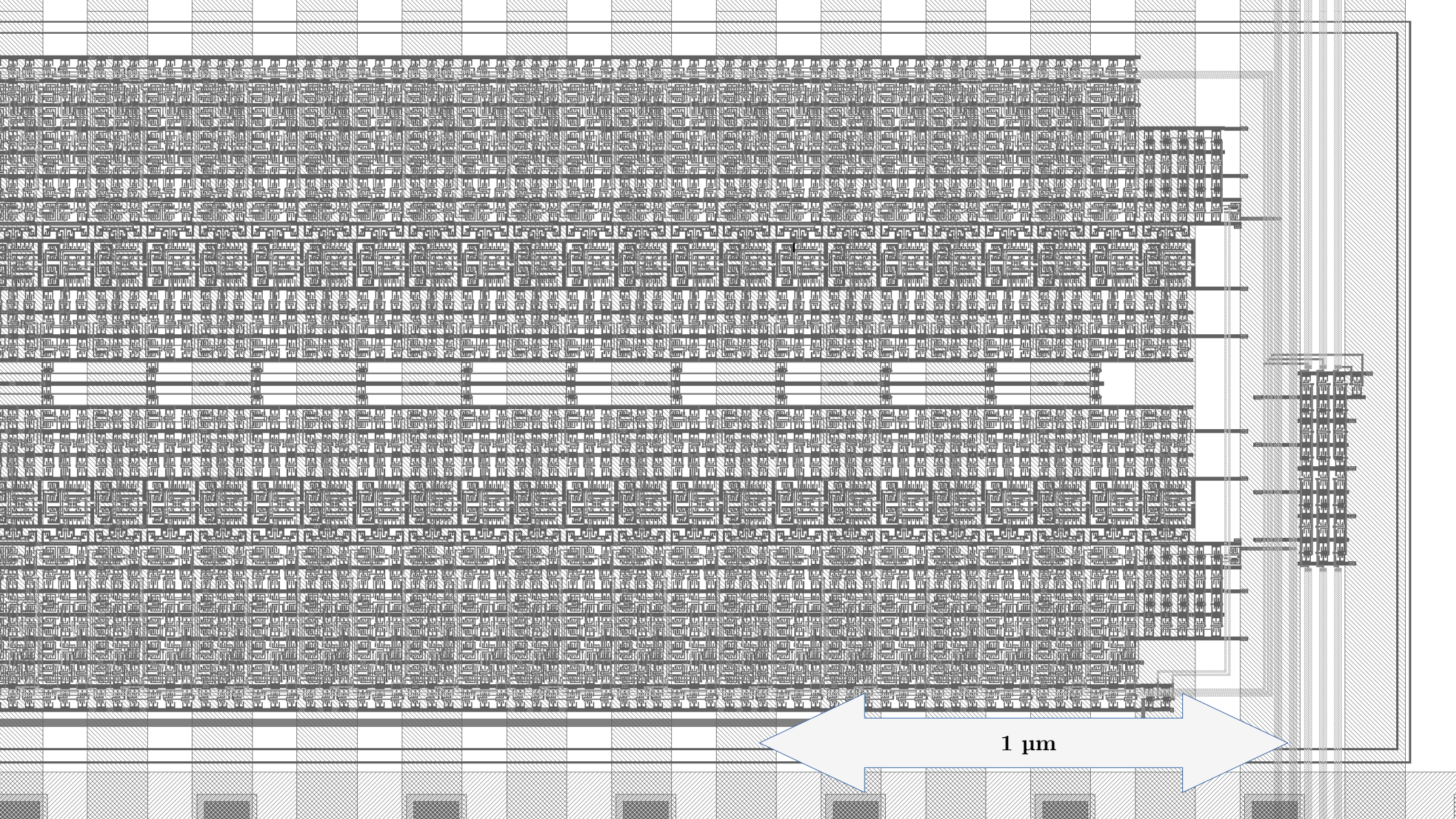


matplotlib

7. Visualization



DESY/CERN. 2020. doi.org/10.1016/j.nima.2020.163784


1 µm

180nm → 130nm → 65nm → 28nm

0.5x Reduced analog supply

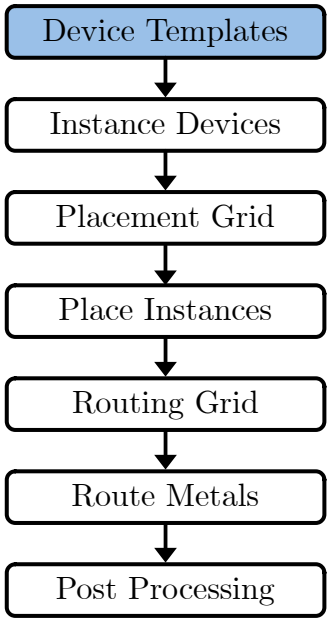
2x Faster devices

2-3x device density

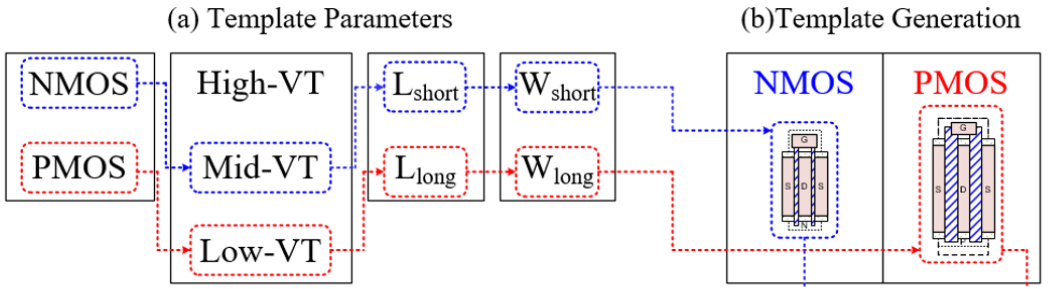
2x DRC rules, compact = better

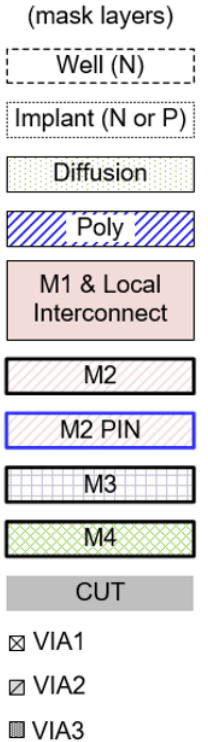
2x cost -> 8000€ / mm^2

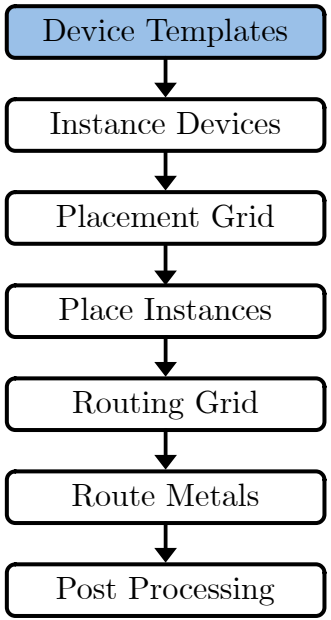
"Correct by construction"



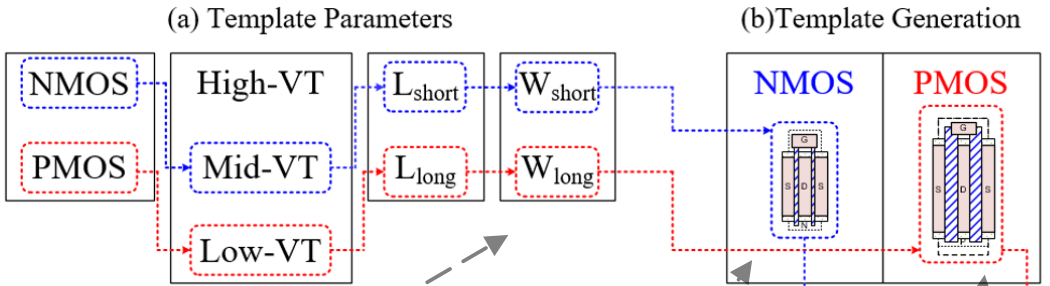
Procedural layout in code







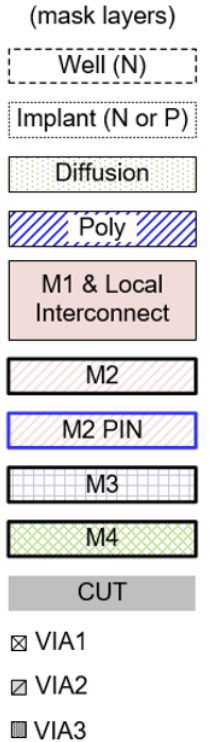
Procedural layout in code

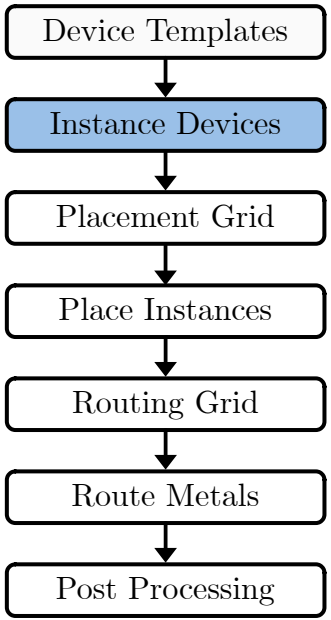


Limit unit device, as all should fit in a predefined area

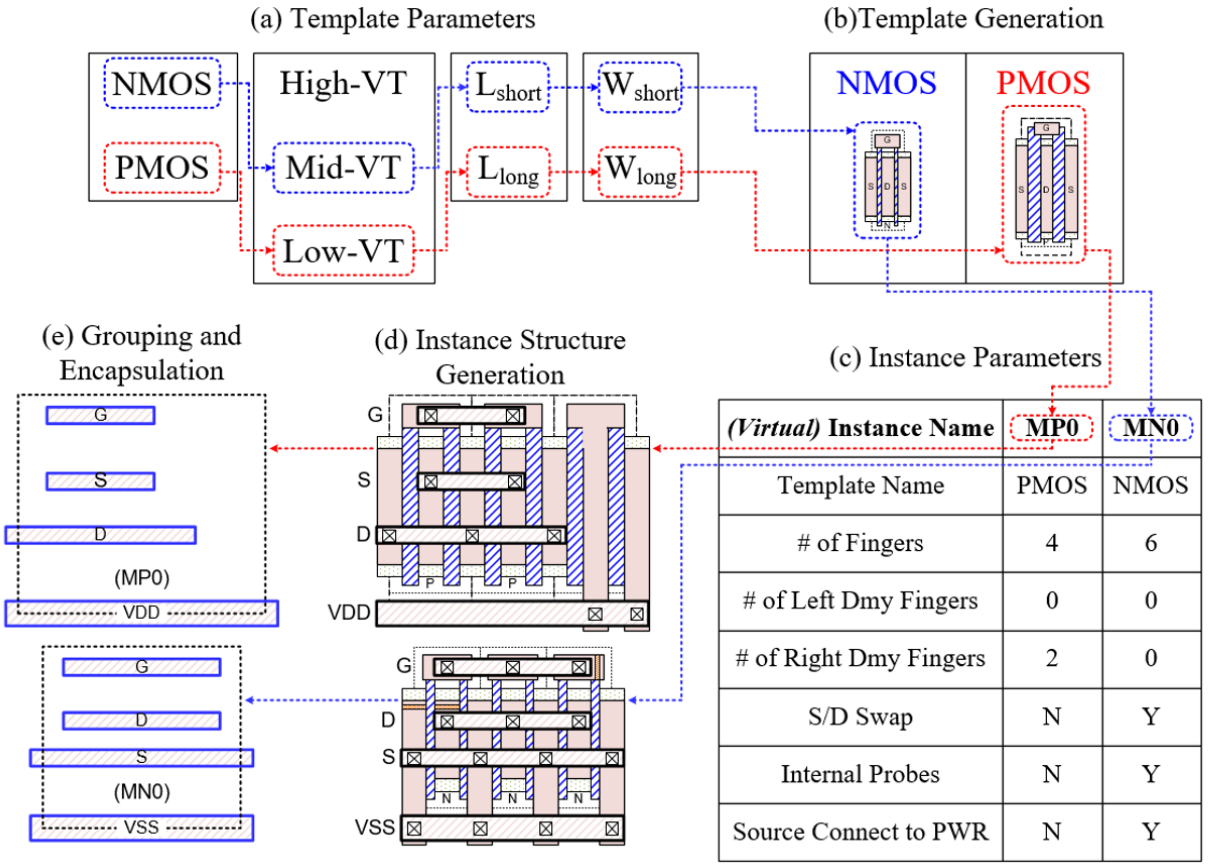
Mimic PDK PCells

Similar approach for capacitors, resistors



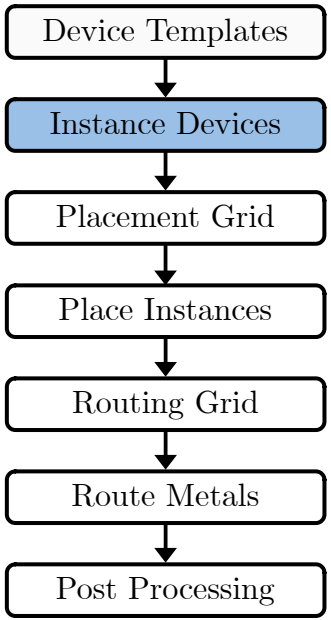


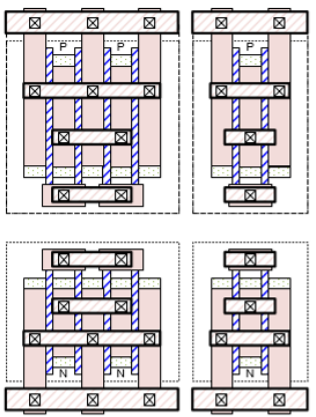
Procedural layout in code



(Virtual) Instance Name	MP0	MN0
Template Name	PMOS	NMOS
# of Fingers	4	6
# of Left Dmy Fingers	0	0
# of Right Dmy Fingers	2	0
S/D Swap	N	Y
Internal Probes	N	Y
Source Connect to PWR	N	Y

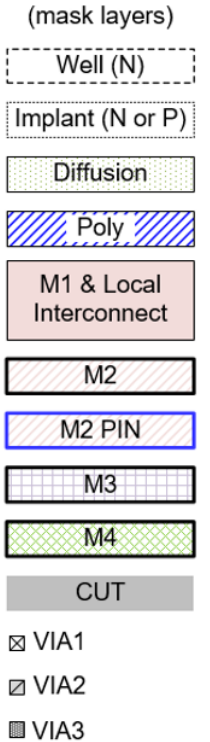
https://laygo2.github.io/





(a) Generate virtual instances and encapsulate

Reminder: We divide large devices in netlist between multiple layout cells



Device Templates

Instance Devices

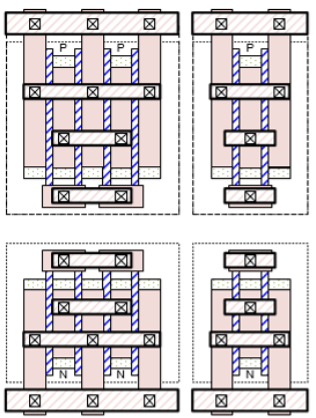
Placement Grid

Place Instances

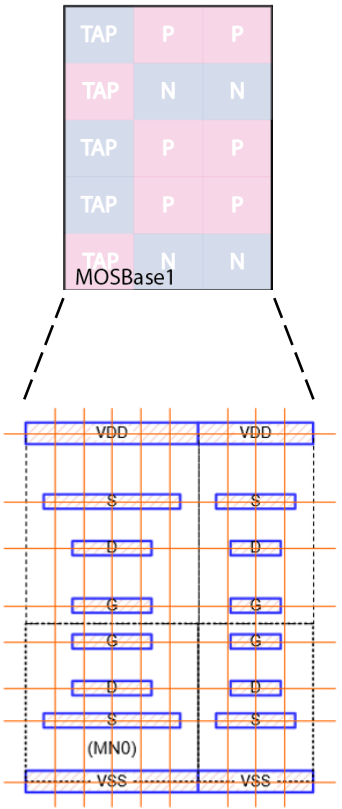
Routing Grid

Route Metals

Post Processing



(a) Generate virtual instances and encapsulate



(b) Place instances and generate M1-M2 grid

(mask layers)

Well (N)

Implant (N or P)

Diffusion

Poly

M1 & Local Interconnect

M2

M2 PIN

M3

M4

CUT

VIA1

VIA2

VIA3

https://laygo2.github.io/

Device Templates

Instance Devices

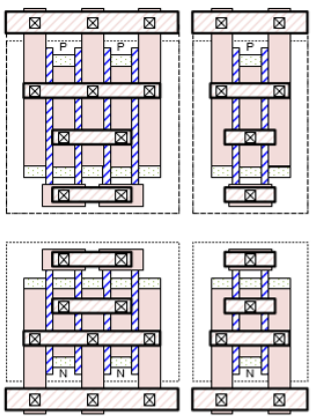
Placement Grid

Place Instances

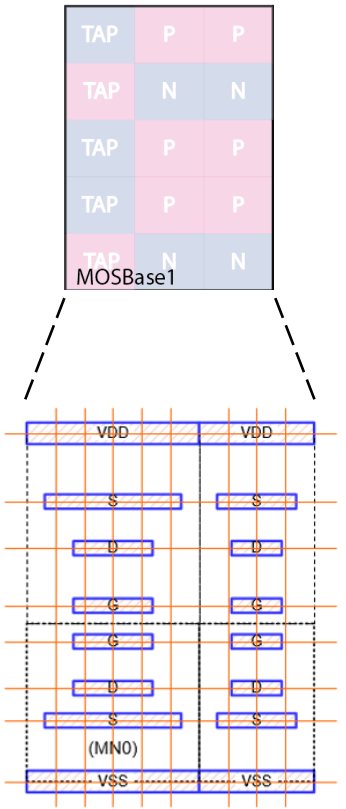
Routing Grid

Route Metals

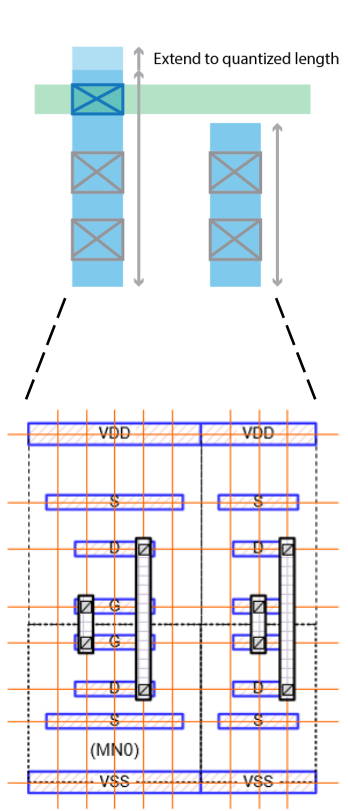
Post Processing



(a) Generate virtual instances and encapsulate



(b) Place instances and generate M1-M2 grid



(c) Place M2 wires

(mask layers)

Well (N)

Implant (N or P)

Diffusion

Poly

M1 & Local Interconnect

M2

M2 PIN

M3

M4

CUT

VIA1

VIA2

VIA3

https://laygo2.github.io/

Device Templates

Instance Devices

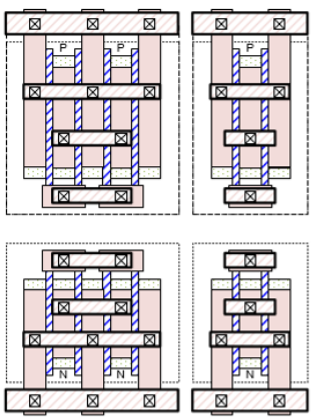
Placement Grid

Place Instances

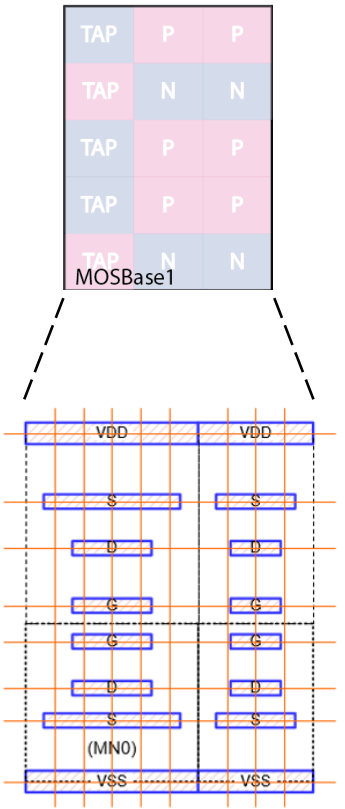
Routing Grid

Route Metals

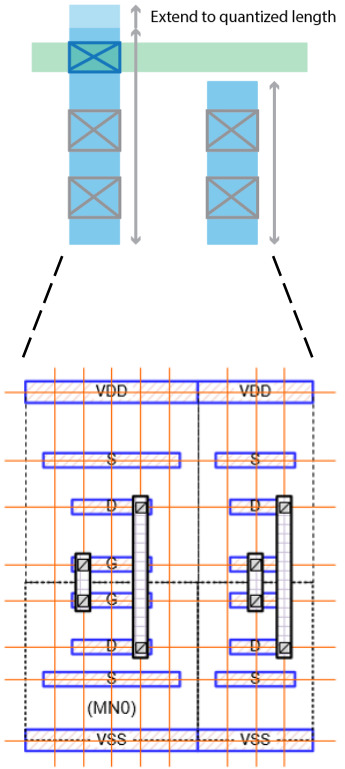
Post Processing



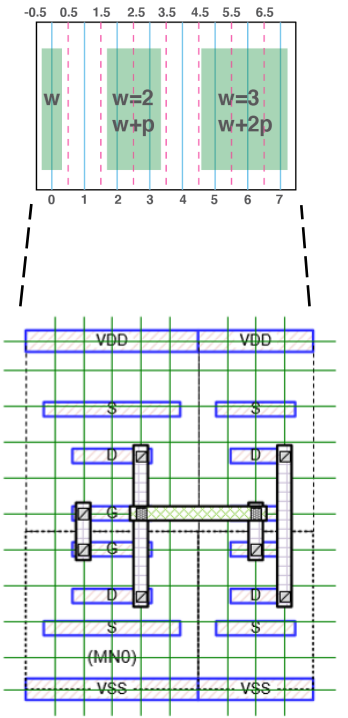
(a) Generate virtual instances and encapsulate



(b) Place instances and generate M1-M2 grid



(c) Place M2 wires



(d) Generate M2-M3 grid and place M3 wires

(mask layers)

Well (N)

Implant (N or P)

Diffusion

Poly

M1 & Local Interconnect

M2

M2 PIN

M3

M4

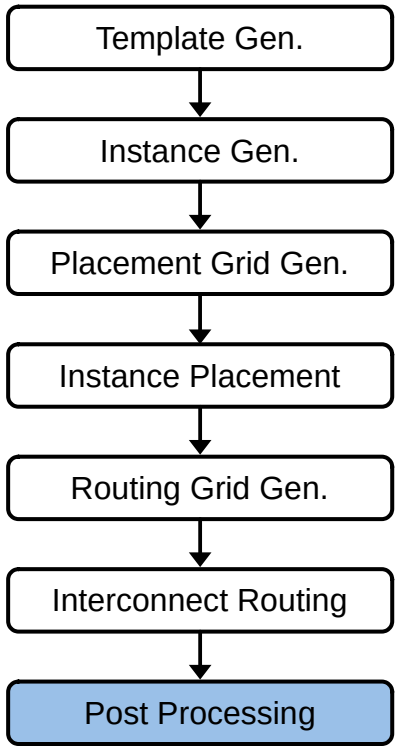
CUT

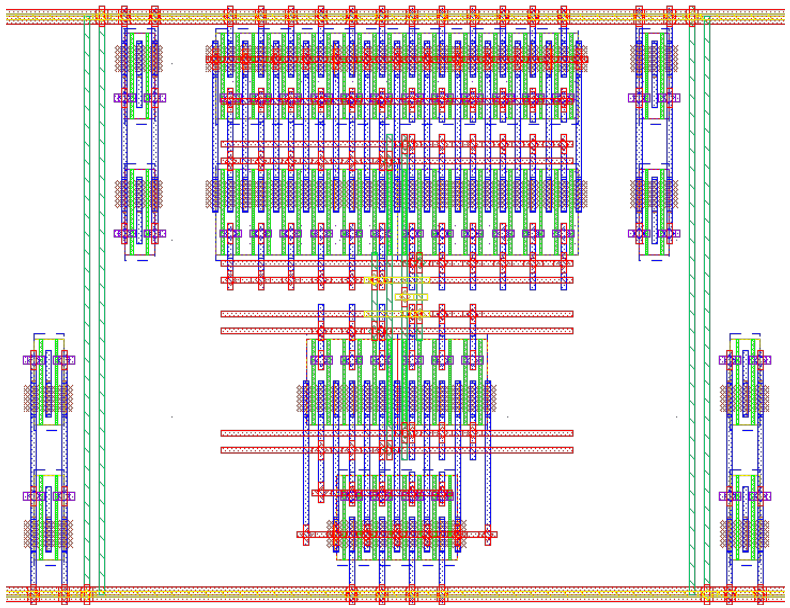
VIA1

VIA2

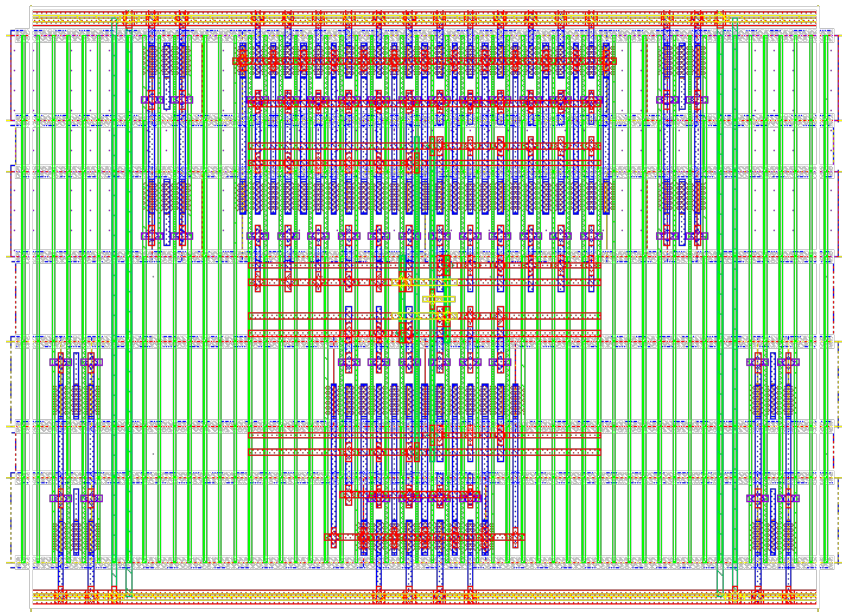
VIA3

https://laygo2.github.io/

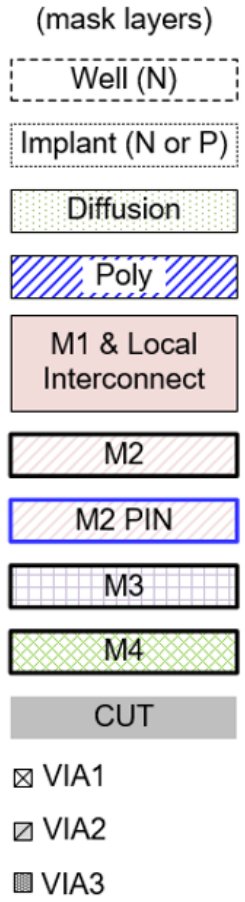




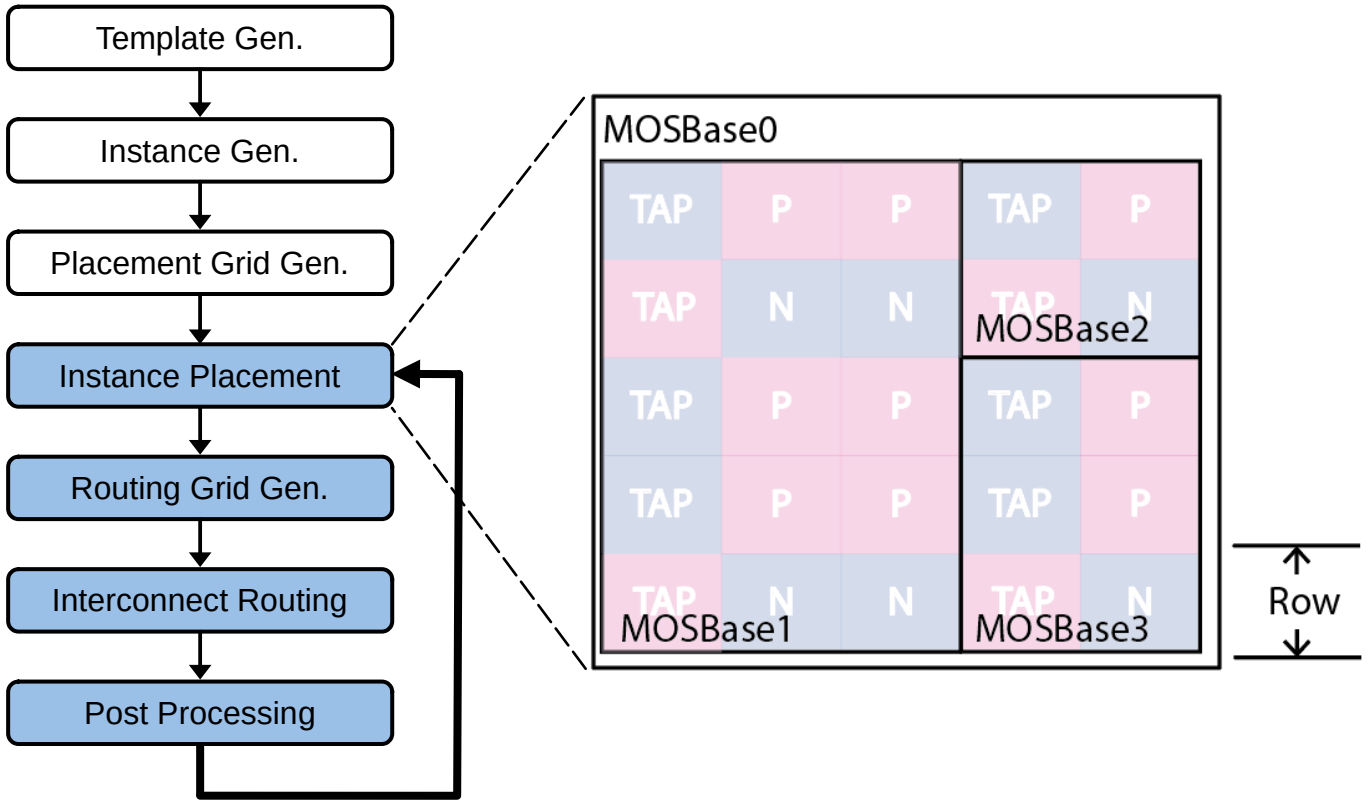
(a) Explicit device layout

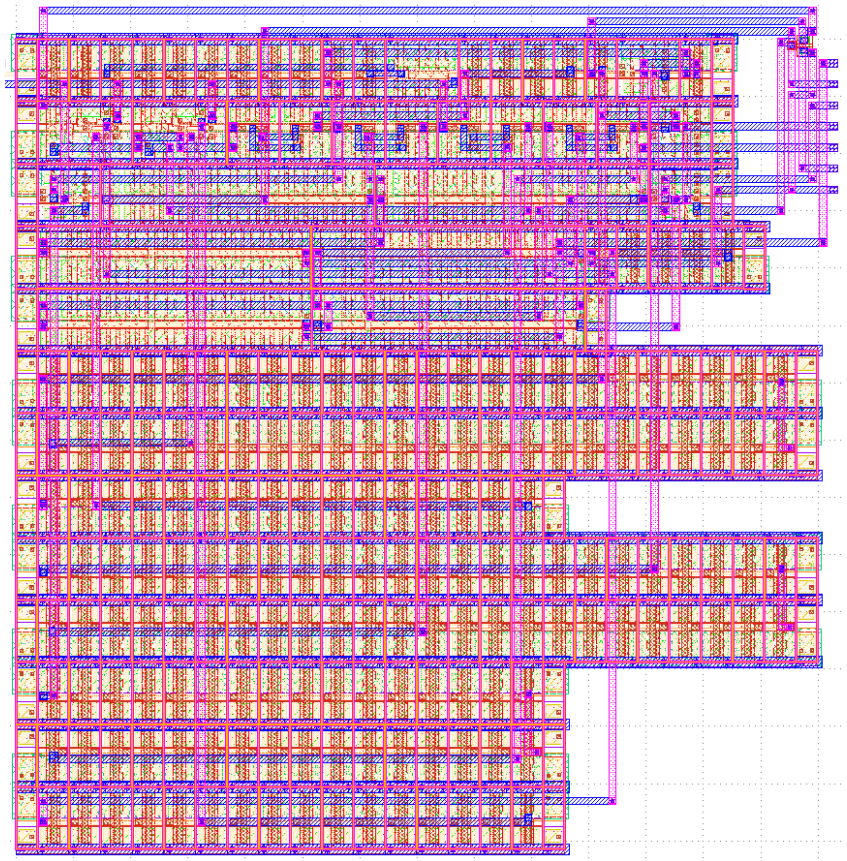


(b) After post processing: dummy devices, poly fill/cut, guard rings, etc

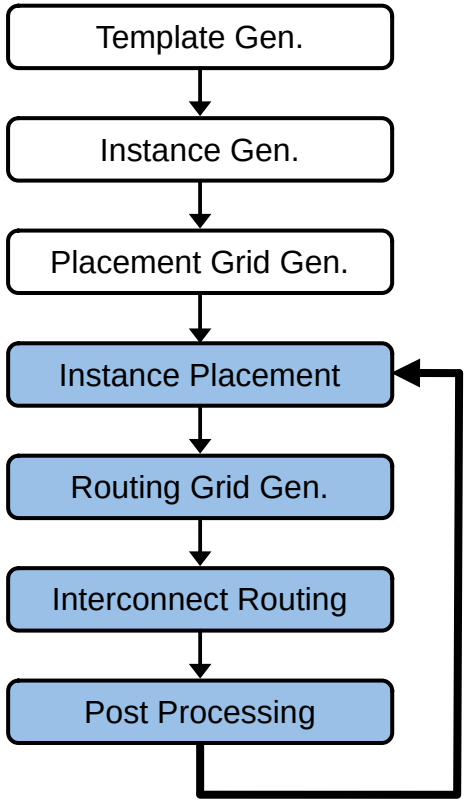


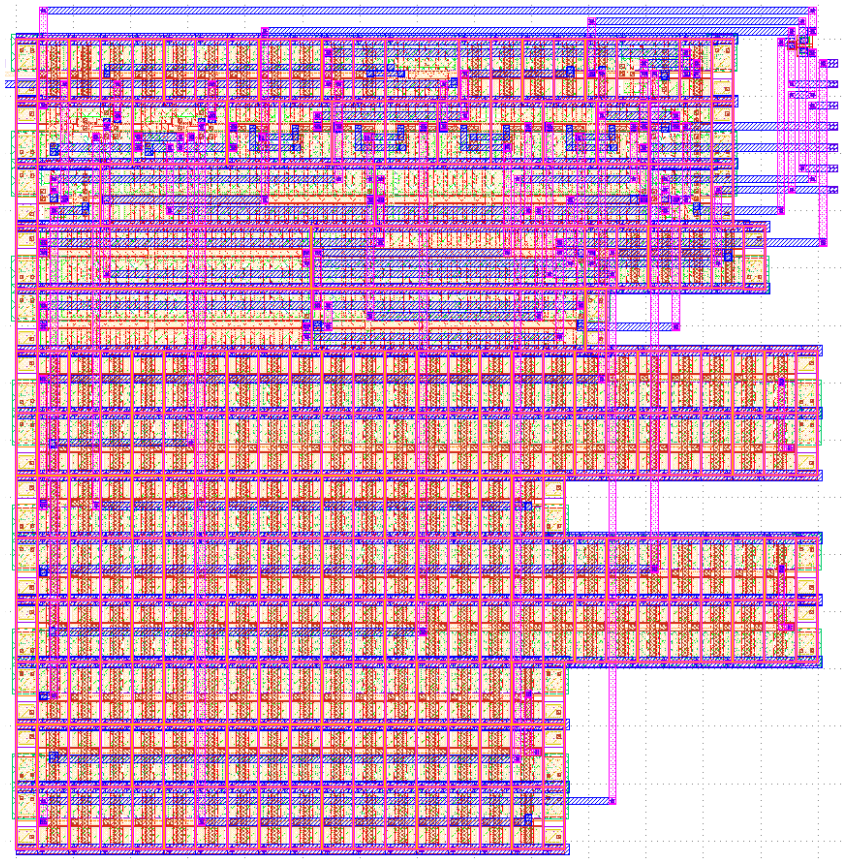
https://bag3-readthedocs.readthedocs.io/en/latest/



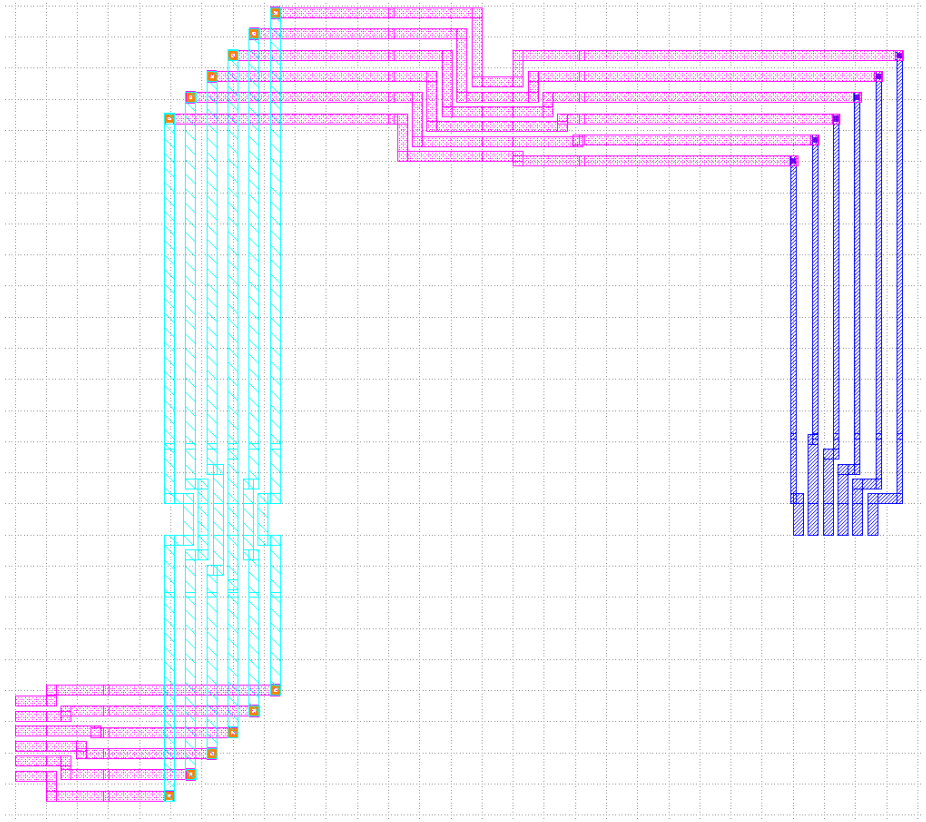


(a) Hierarchical layout w/ interconnect



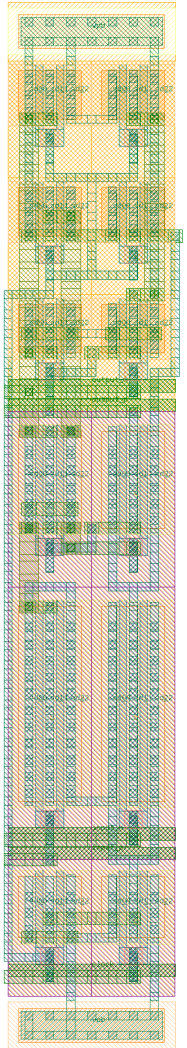


(a) Hierarchical layout w/ interconnect



(b) Jog interconnect to pins

https://api.substratelabs.io/substrate/



```
* StrongARM latch, produced from Substrate

.SUBCKT nmos_tile_w1250_l150_nf2 sd_0 sd_1 sd_2 g b
  Minst0 sd_0 g sd_1 b nshort l=0.150 mult=1 nf=1 w=1.250
  Minst1 sd_1 g sd_2 b nshort l=0.150 mult=1 nf=1 w=1.250
.ENDS nmos_tile_w1250_l150_nf2

.SUBCKT nmos_tile_w4000_l150_nf2 sd_0 sd_1 sd_2 g b
  Minst0 sd_0 g sd_1 b nshort l=0.150 mult=1 nf=1 w=4.000
  Minst1 sd_1 g sd_2 b nshort l=0.150 mult=1 nf=1 w=4.000
.ENDS nmos_tile_w4000_l150_nf2

.SUBCKT nmos_tile_w2000_l150_nf2 sd_0 sd_1 sd_2 g b
  Minst0 sd_0 g sd_1 b nshort l=0.150 mult=1 nf=1 w=2.000
  Minst1 sd_1 g sd_2 b nshort l=0.150 mult=1 nf=1 w=2.000
.ENDS nmos_tile_w2000_l150_nf2

.SUBCKT pmos_tile_w1000_l150_nf2 sd_0 sd_1 sd_2 g b
  Minst0 sd_0 g sd_1 b pshort l=0.150 mult=1 nf=1 w=1.000
  Minst1 sd_1 g sd_2 b pshort l=0.150 mult=1 nf=1 w=1.000
.ENDS pmos_tile_w1000_l150_nf2

.SUBCKT atoll_strong_arm_instance input_p input_n output_p output_n clock
vdd vss
  Xinst0 vss tail vss clock vss nmos_tile_w1250_l150_nf2
  Xinst1 vss tail vss clock vss nmos_tile_w1250_l150_nf2
  Xinst2 tail intn tail input_p vss nmos_tile_w4000_l150_nf2
  Xinst3 tail intp tail input_n vss nmos_tile_w4000_l150_nf2
  Xinst4 intn output_n intn output_p vss nmos_tile_w2000_l150_nf2
  Xinst5 intp output_p intp output_n vss nmos_tile_w2000_l150_nf2
  Xinst6 vdd output_n vdd output_p vdd pmos_tile_w1000_l150_nf2
  Xinst7 vdd output_p vdd output_n vdd pmos_tile_w1000_l150_nf2
  Xinst8 vdd output_n vdd clock vdd pmos_tile_w1000_l150_nf2
  Xinst9 vdd output_p vdd clock vdd pmos_tile_w1000_l150_nf2
  Xinst10 vdd intn vdd clock vdd pmos_tile_w1000_l150_nf2
  Xinst11 vdd intp vdd clock vdd pmos_tile_w1000_l150_nf2
.ENDS atoll_strong_arm_instance
```

https://github.com/ucb-substrate/strongarm

Schematic and Layout Design Tools