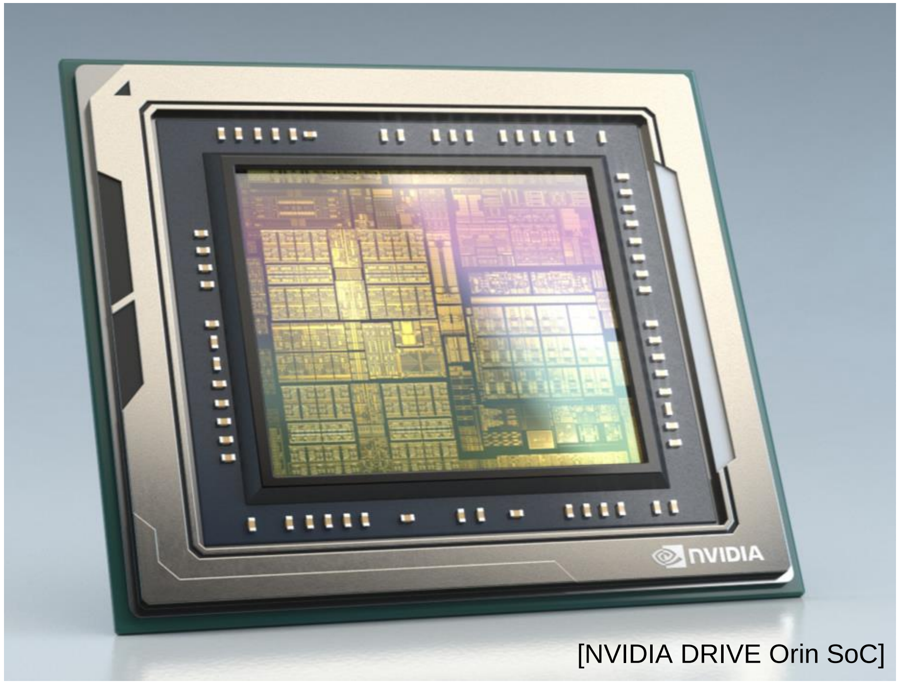
Teaching Mixed-Signal Design Using Open-Source Tools

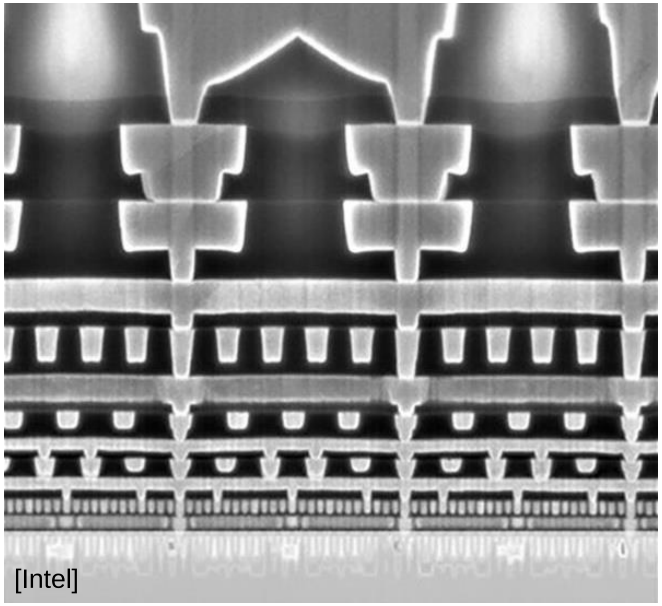
Boris Murmann

bmurmann@hawaii.edu

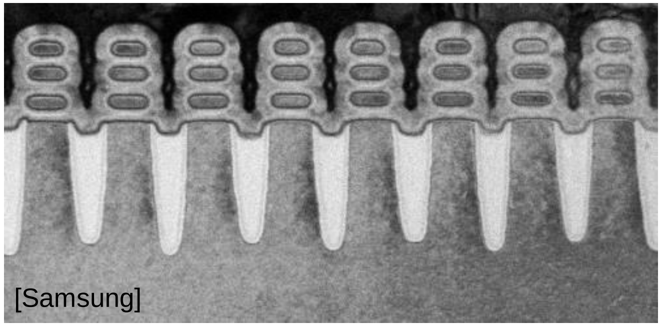
Today's Chip Technology: Truly Amazing, But not "Cool"?



[NVIDIA DRIVE Orin SoC]



[Intel]



[Samsung]

Competition for Tech Talent

- Ample alternatives that provide nearly instant gratification
- For example, can build and train an ML system within days…

Abstraction Layer Sandwich

Software Systems
Algorithms

Hardware Systems
Circuits

Devices
Materials

Traditional learning trajectory for IC designers

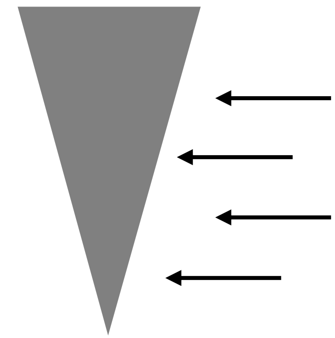
- The current education system requires far too may prerequisite courses before exposing students to analog/mixed-signal IC design
- The field was created bottom-up, but innovation (and excitement) is progressively shifting to higher levels of abstraction
- We should capitalize on this trend to re-energize chip design education

Starting From the Top

Software Systems
Algorithms

Hardware Systems
Circuits

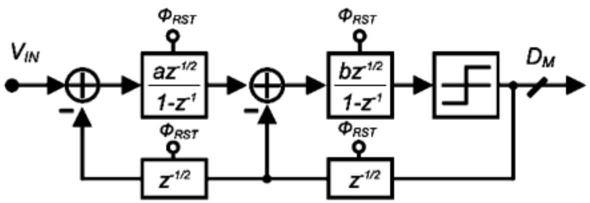
Devices
Materials

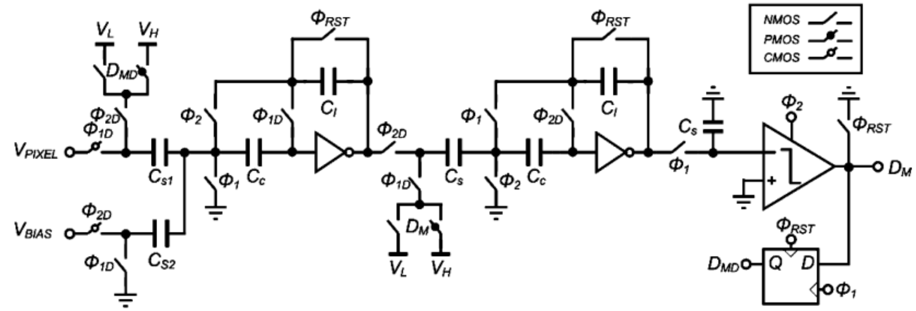


- It is not necessary to understand the entire sandwich learn the basics of chip design (including mixed-signal IC design)
- Possible approaches
  - Follow along as the instructor creates a “template” design
  - Form teams of students with complementary skill sets
    - Some may understand transistors, some excel at software, etc.

Overview of EE 628 “Tape-out Course” (University of Hawaii)

[Chae, JSSC 1/2011]





High-level model → Design and layout of complete transistor-level circuit

Design concept
Python
Scilab
…

→ **Schematic entry**
Xschem

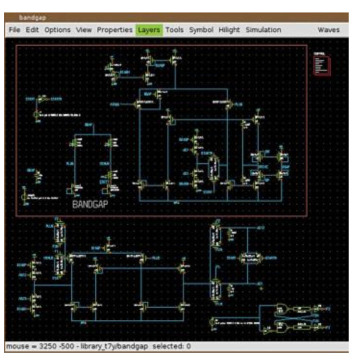
→ **Simulation**
Ngspice

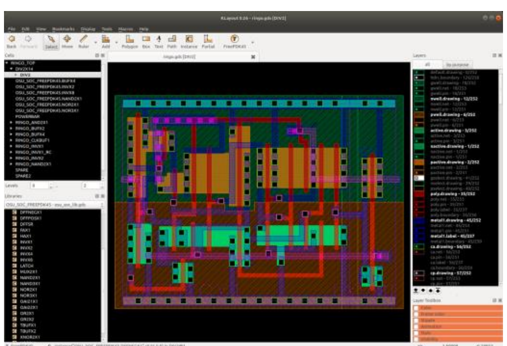
→ **Layout**
KLayout

→ **DRC LVS**
KLayout

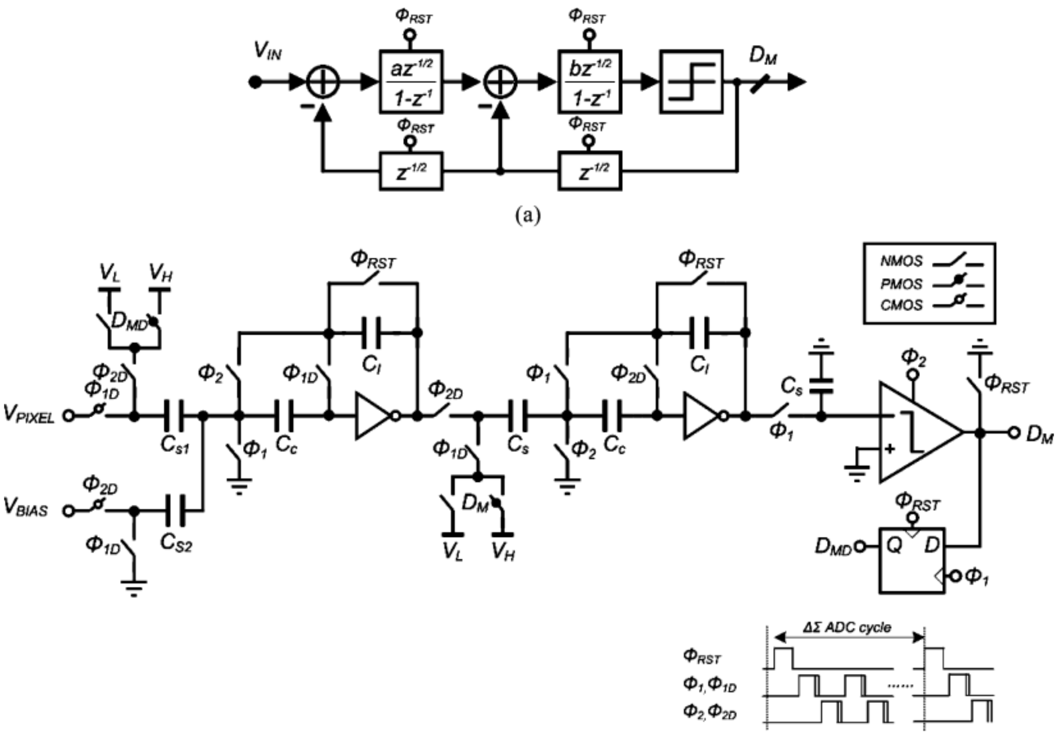
→ **Parasitic extraction**
Magic

Post- processing & Visualization
Gaw
Xschem
Python
…





Template Project: Incremental Delta-Sigma A/D Converter



(a)

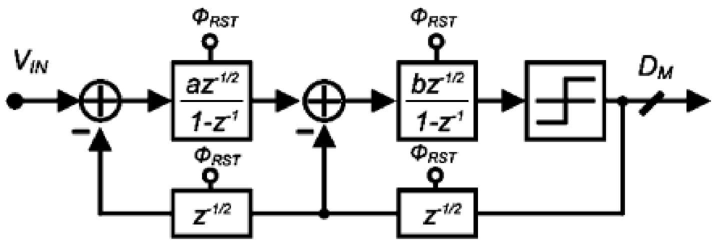
(b)

- Attractive features
  - Can study operation & nonidealities in software
  - Can do a gradual transition to real circuits & transistors
  - Given implementation has low overhead (no opamps)
  - Circuit has low pin count, easy to fit many copies in one package

Y. Chae et al., "A 2.1 M Pixels, 120 Frame/s CMOS Image Sensor With Column-Parallel ADC Architecture," in IEEE Journal of Solid-State Circuits, Jan. 2011. https://ieeexplore.ieee.org/document/5641589

Lots of Interesting Things to Learn

How does the ideal model work?

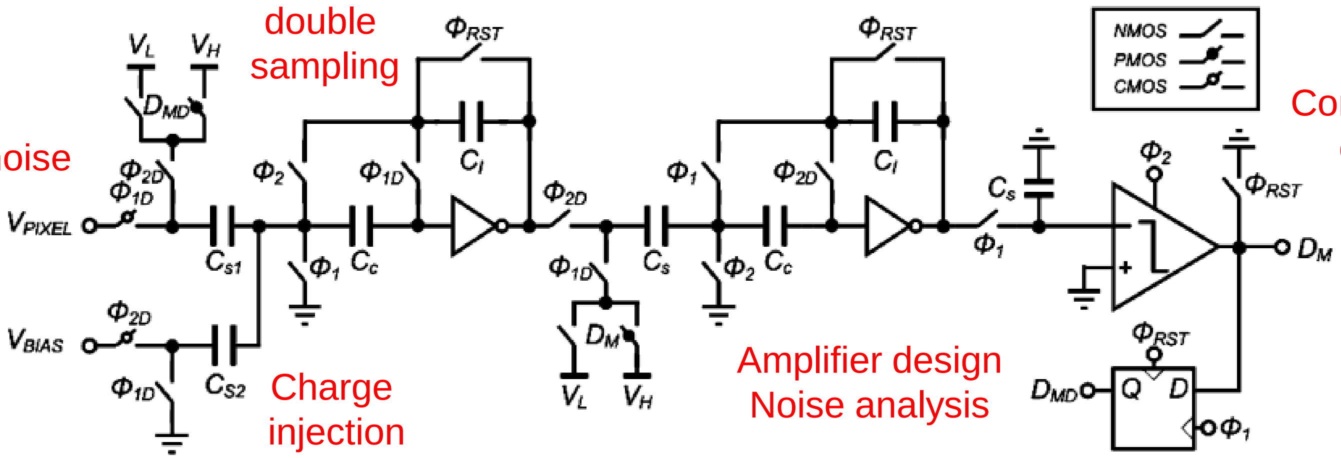


(a)

Correlated double sampling

kT/C noise

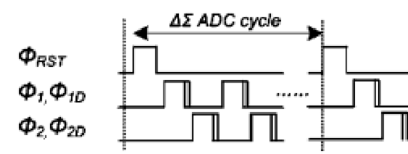
Comparator design



Charge injection

Amplifier design
Noise analysis

Clock generation



Possible to complete project without understanding all details…
But students may want to take the next course to learn more…

EE 628 Course Outline

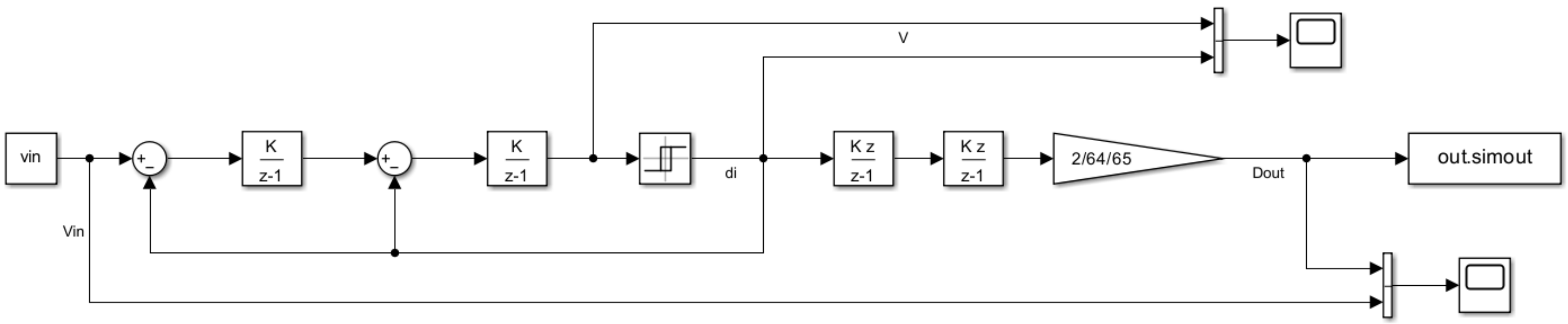
- High-level analysis and simulation of the template ADC
  - Using Scilab, Simulink, etc.
- Build and simulate the idealized spice-level circuit
  - Using ideal switches and controlled sources (no transistors)
- Build, analyze and simulate the components (with transistors)
  - Switches, integrator, comparator, clock generator
- Mid-semester team presentations
- Assemble the complete circuit
  - Insert components one by one and verify operation
- Layout, DRC, LVS
  - First using a trivial example, then for the designed blocks & chip level
- Final team presentations
- Tapeout!

Lecture Structure

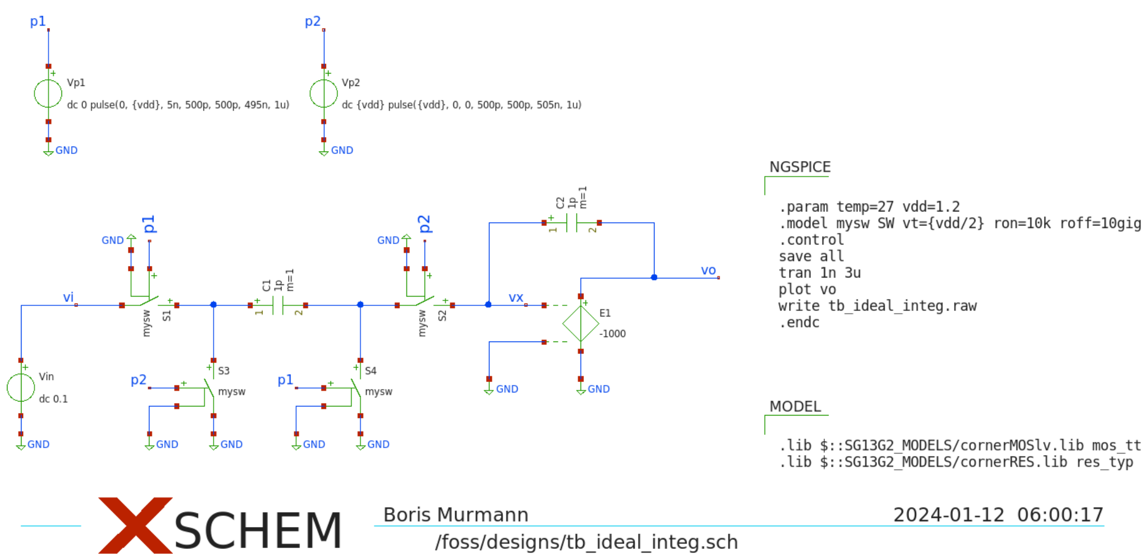
<u>Classical lecture material</u>
Circuit design
Circuit simulation
Analysis of nonidealities
Technology aspects
Layout basics
…

<u>Demo & Discussion Time</u>
Logistics
Tool demos
Troubleshooting
Student presentations
…

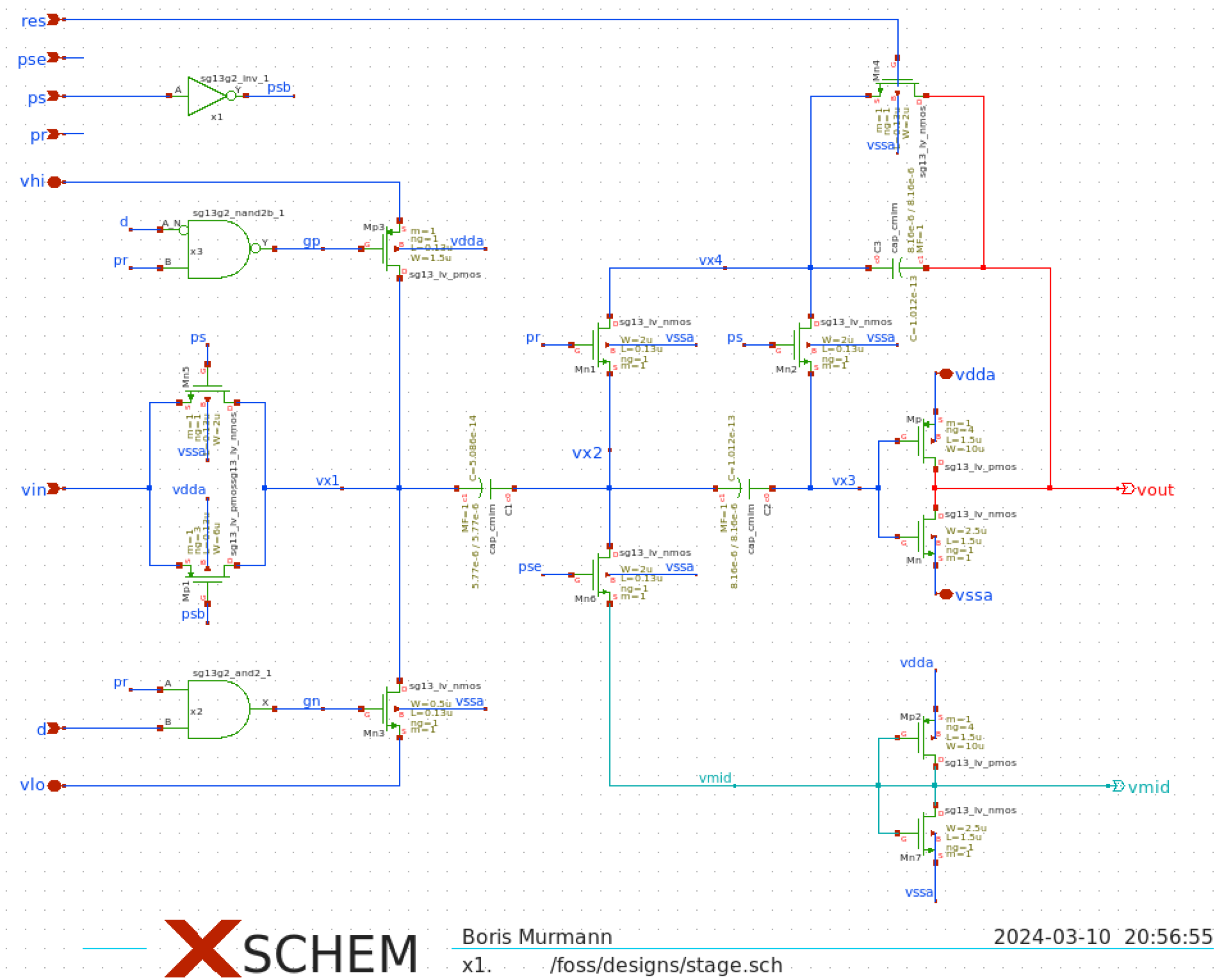
Progression



Conceptual integrator circuit

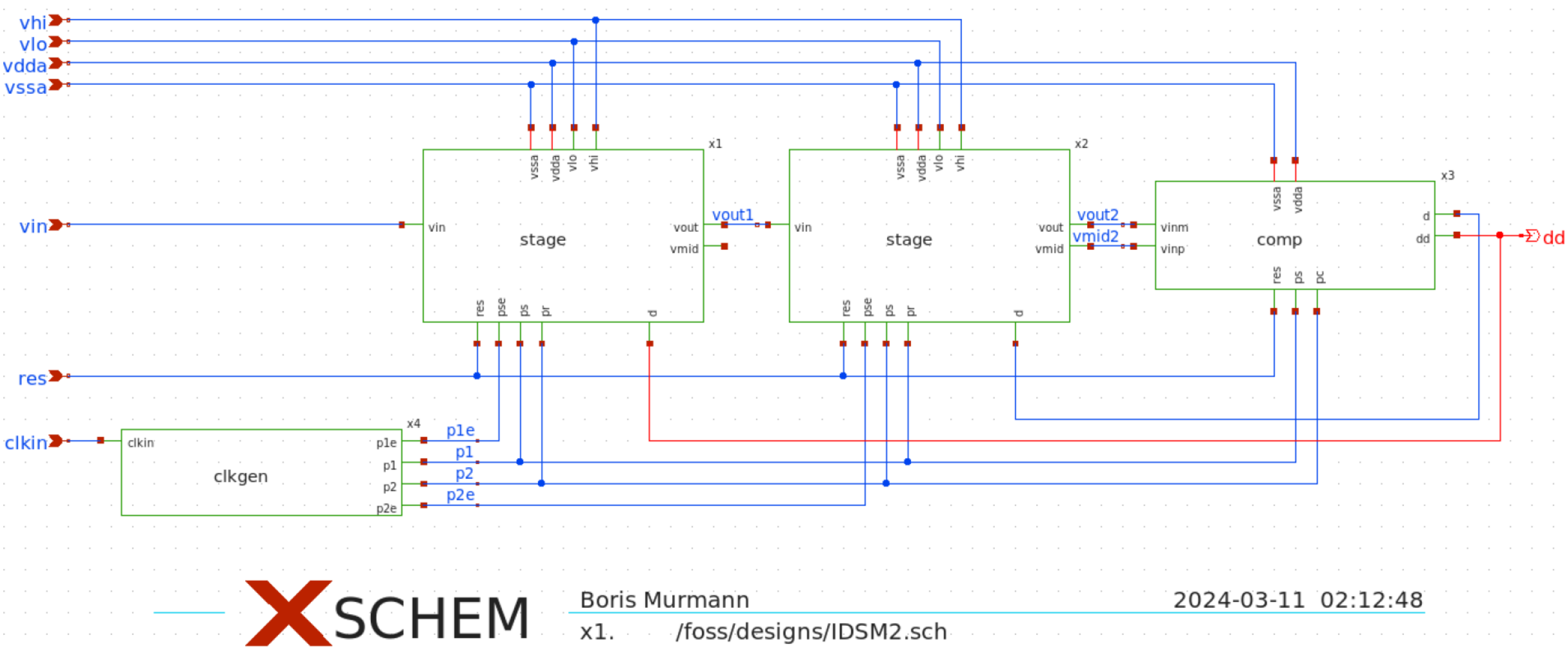


Complete Stage

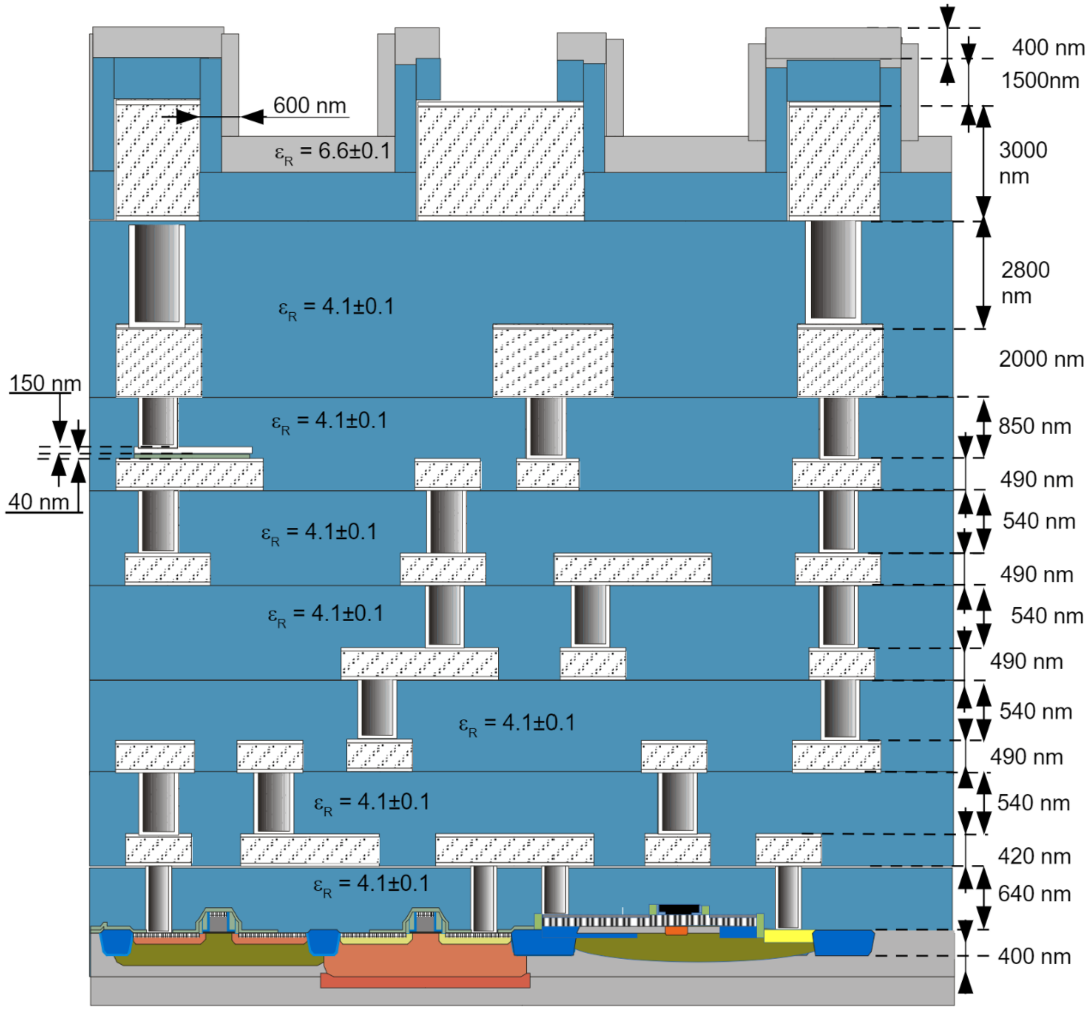


Complete Modulator

p1: Stage 1 samples, stage 2 redistributes, comparator samples, dd toggles (used by stage 1 during p2)
p2: Stage 2 samples, stage 1 redistributes, comparator decides, d toggles (used by stage 2 during p1)

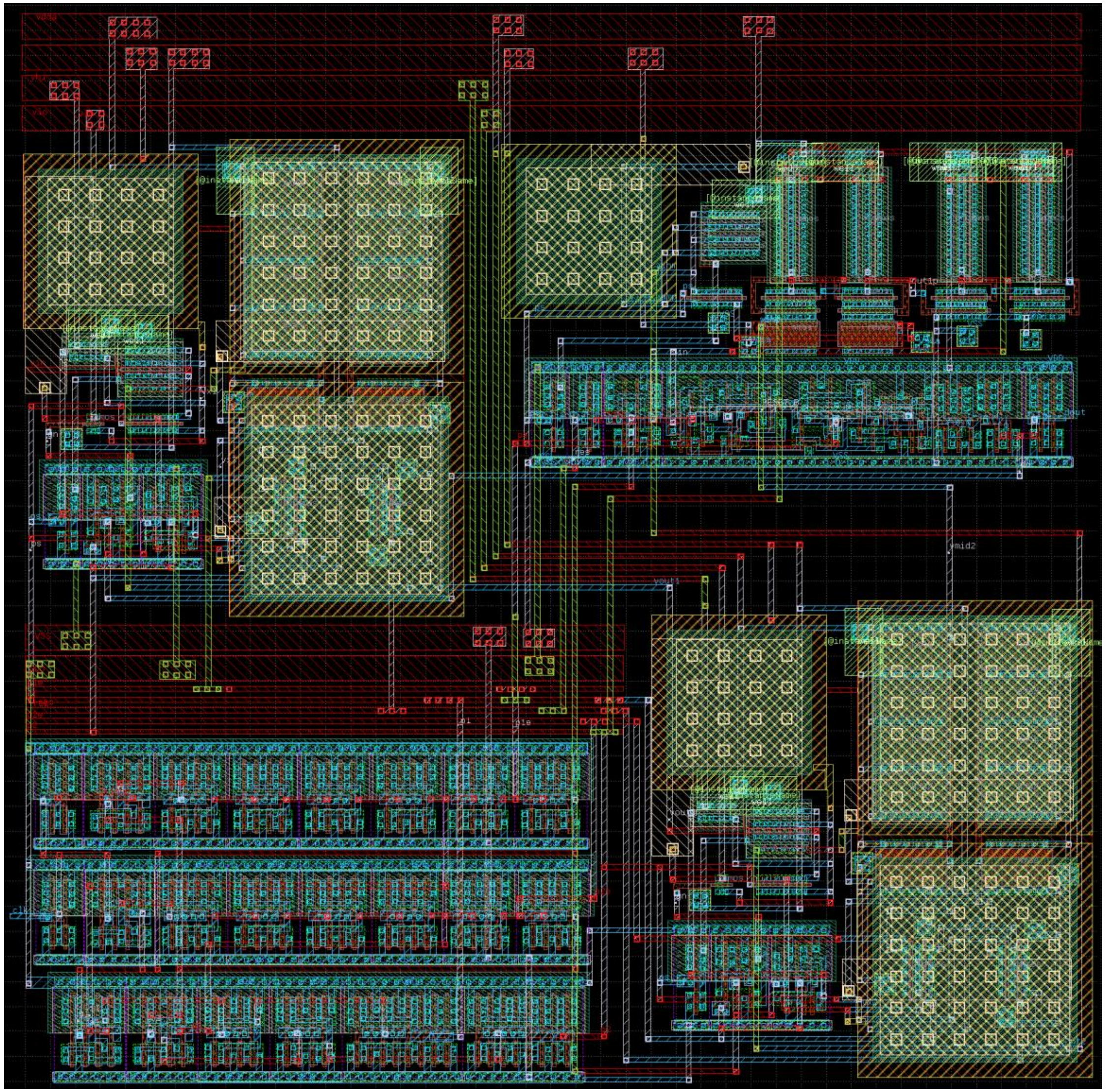


Technology: IHP SG13G2 (Open-Source PDK)

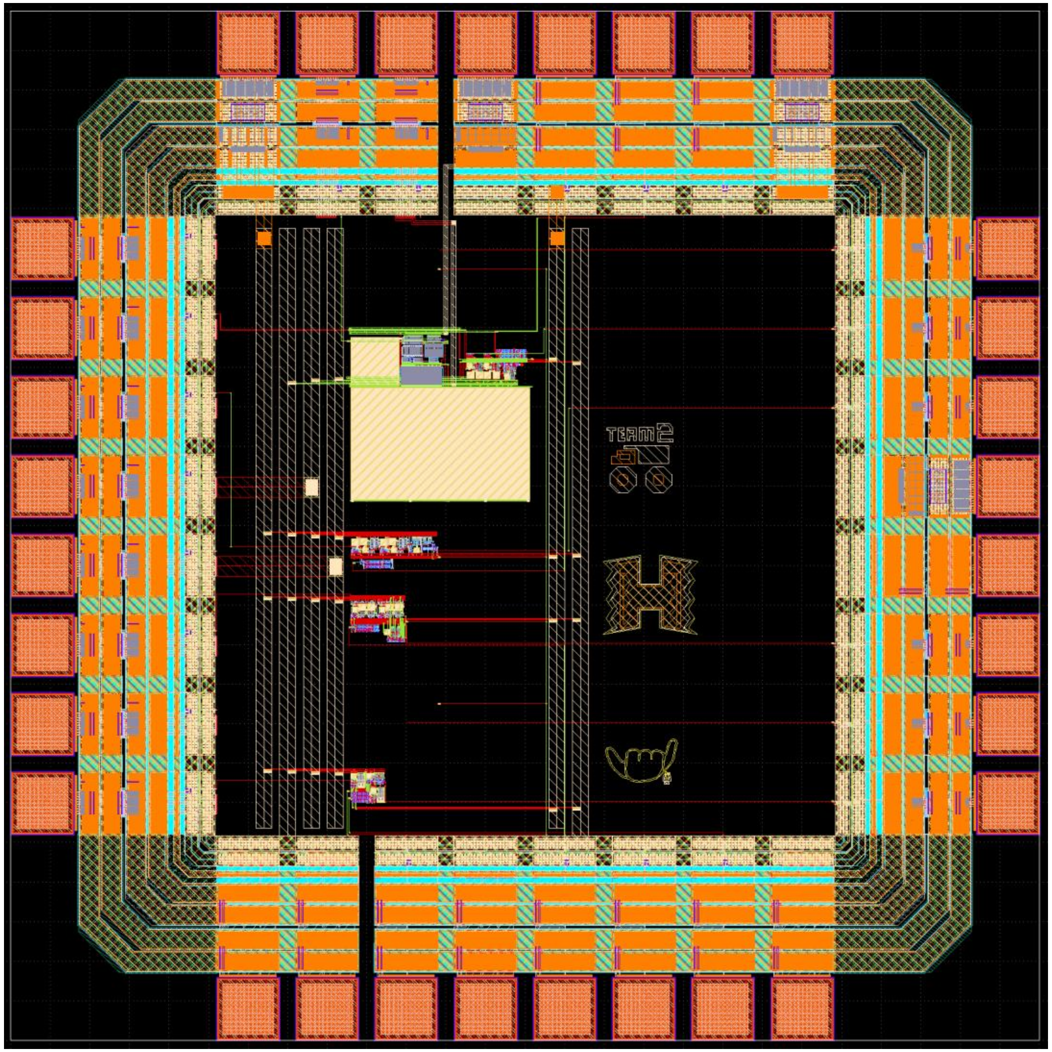


https://github.com/IHP-GmbH/IHP-Open-PDK/blob/main/ihp-sg13g2/libs.doc/doc/SG13G2_os_process_spec.pdf

Sample Layout

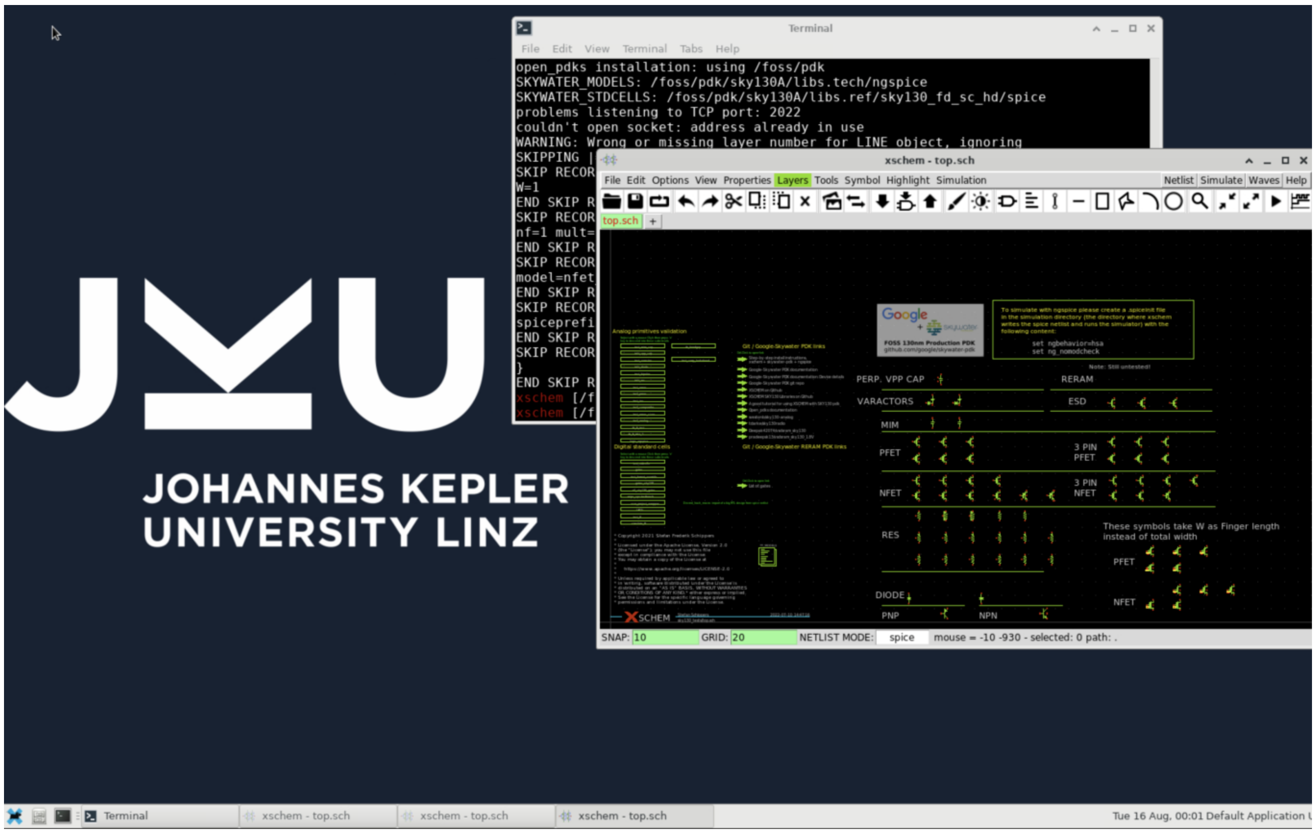


Shared Chip (6 Delta-Sigma Modulators)



Open-Source Tools via Docker Container

- https://github.com/hpretl/iic-osic-tools
- Not the highest performance option, but easy entry point for students



- covered Verilog code coverage
- cvc circuit validity checker (ERC)
- fault design-for-test (DFT) solution
- gaw3-xschem waveform plot tool for `xschem`
- gdsfactory Python library for GDS generation
- gdspy Python module for creation and manipulation of GDS files
- ghdl VHDL simulator
- gtkwave waveform plot tool for digital simulation
- iic-osic collection of useful scripts and documentation
- irsim switch-level digital simulator
- iverilog Verilog simulator
- klayout layout tool
- magic layout tool with DRC and PEX
- netgen netlist comparison (LVS)
- ngscope waveform plot tool for `ngspice`
- ngspice SPICE analog simulator
- open_pdks PDK setup scripts
- openlane digital RTL2GDS flow
- openroad collection of tools for `openlane`
- opensta static timing analyzer for digital flow
- padring padring generation tool
- vlog2verilog Verilog file conversion
- risc-v toolchain GNU compiler toolchain for RISC-V RV32I cores
- siliconcompiler modular build system for hardware
- sky130 SkyWater Technologies 130nm CMOS PDK
- verilator fast Verilog simulator
- xschem schematic editor
- xyce fast parallel SPICE simulator (incl. `xdm` netlist conversion tool)
- yosys Verilog synthesis tool (with GHDL plugin for VHDL synthesis)

The Power of Open-Source PDKs and Tools

- No license limitations
- Convenient sharing of all course materials through GitHub
- Easy to interface with Python (students typically have prior exposure)
- Students learn basic data ops and GitHub collaboration
  - Important skillset, regardless of specialization

Demo: Possibilities with Colab Mini-Tutorials

Open in Colab

SKY130 Track and Hold Circuit

Tool setup

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
CONDA_PREFIX = os.environ.get('CONDA_PREFIX', None)
if not CONDA_PREFIX:
  !python -m pip install condacolab
  import condacolab
  condacolab.install()

# install sky130a
!conda install -c litex-hub open_pdks.sky130a

# install ngspice
!conda install -c litex-hub ngspice
```

https://github.com/bmurmann/Ngspice-on-Colab/blob/main/notebooks/SKY130_Track_and_Hold.ipynb

Circuit

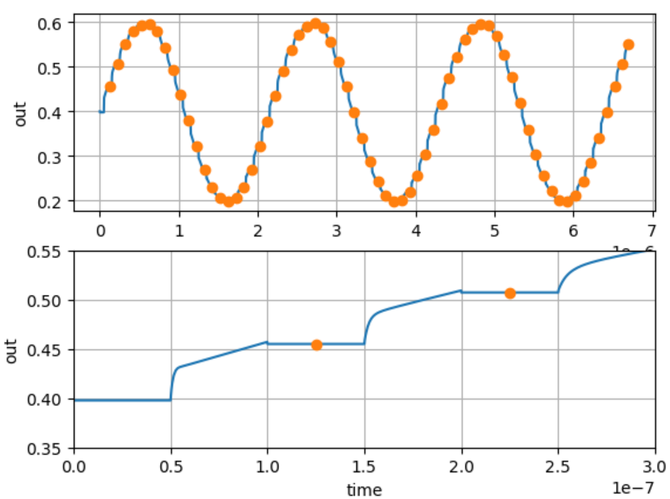
```
%%writefile netlist.spice
* Track-and-hold circuit using single NMOS
.lib "/usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice" tt
x1 in clk out 0 sky130_fd_pr__nfet_01v8_lvt w=5 l=0.15
cl out 0 100f
vin  in 0 sin (0.4 0.2 {fin})
vclk clk 0 pulse (1.2 0 0 100p 100p {per/2} {per})
.param nfft=64 fclk=10Meg per=1/fclk cycles=3 fin=fclk*cycles/nfft
```

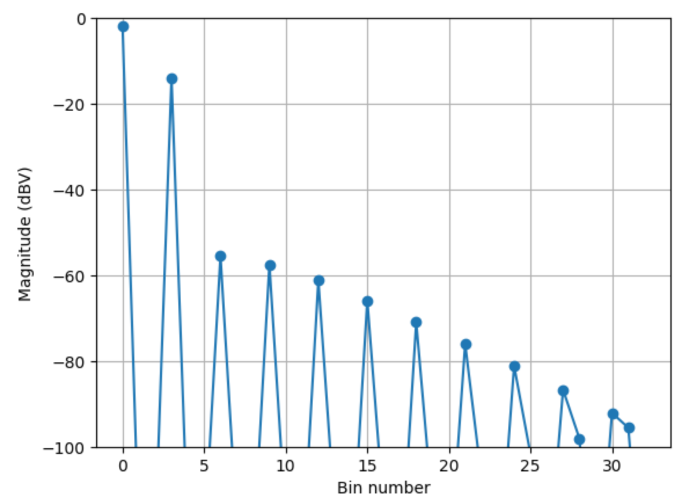
Raw simulation data

```
df1 = pd.read_csv("output1.txt", delim_whitespace=True)
df1
```

	time	out
0	0.000000e+00	0.400000
1	1.000000e-12	0.399961
2	2.000000e-12	0.399922
3	4.000000e-12	0.399845
4	8.000000e-12	0.399695
...	...	...

Postprocessing





Demo: Simulation Flow (Xschem/Ngspice/Jupyter Notebook)