



Leibniz Institute  
for high  
performance  
microelectronics

# End-to-End Bandgap Reference Design: From Schematic to Layout Essentials

Phillip Ferreira Baade–Pedersen (Design Engineer)

Dr. Ing. Christian Wittke (Scientist)

Frankfurt (Oder), IHP - 20/05/2025

**Projects:** BMBF → FMD-QNC (16ME0831)

# Agenda For today



Leibniz Institute  
for high  
performance  
microelectronics

**09:00 - 09:15 | Recap | Questions?**

*Review of Key Takeaways from Yesterday*

**09:15 – 09:30 | Module Overview**

*Overview and Introduction to Today's Module*

**09:30 - 10:45 | Expert Talk**

*Dr. Prabhat Kumar Dubey: Insights on Model Generation in the Open PDK*

**10:45 - 12:00 | Introduction | Hands On**

*Designing the Testbench for OTA and Bandgap Reference*

**12:00 - 13:00 | Lunch Brake | Catching Up**

**13:00 - 15:00 | Continuation of Previous Session's Work**

*Completing Testbenches with Mismatch Analysis and Starting Layout Design*

**15:00 – 15:30 | Coffee Break | Caching Up**

**15:30 - 17:00 | Hands-On Session**

*Guided Layout Session or Continuation of Testbench Design and Simulations*



# Key Takeaways from Yesterday

- 0 Introduction to the IHP Open PDK
- 0 Learning the basics in Xschem related to simulations
- 0 Basic Klayout maneuvering
- 0 First examples on physical verification

## Questions?

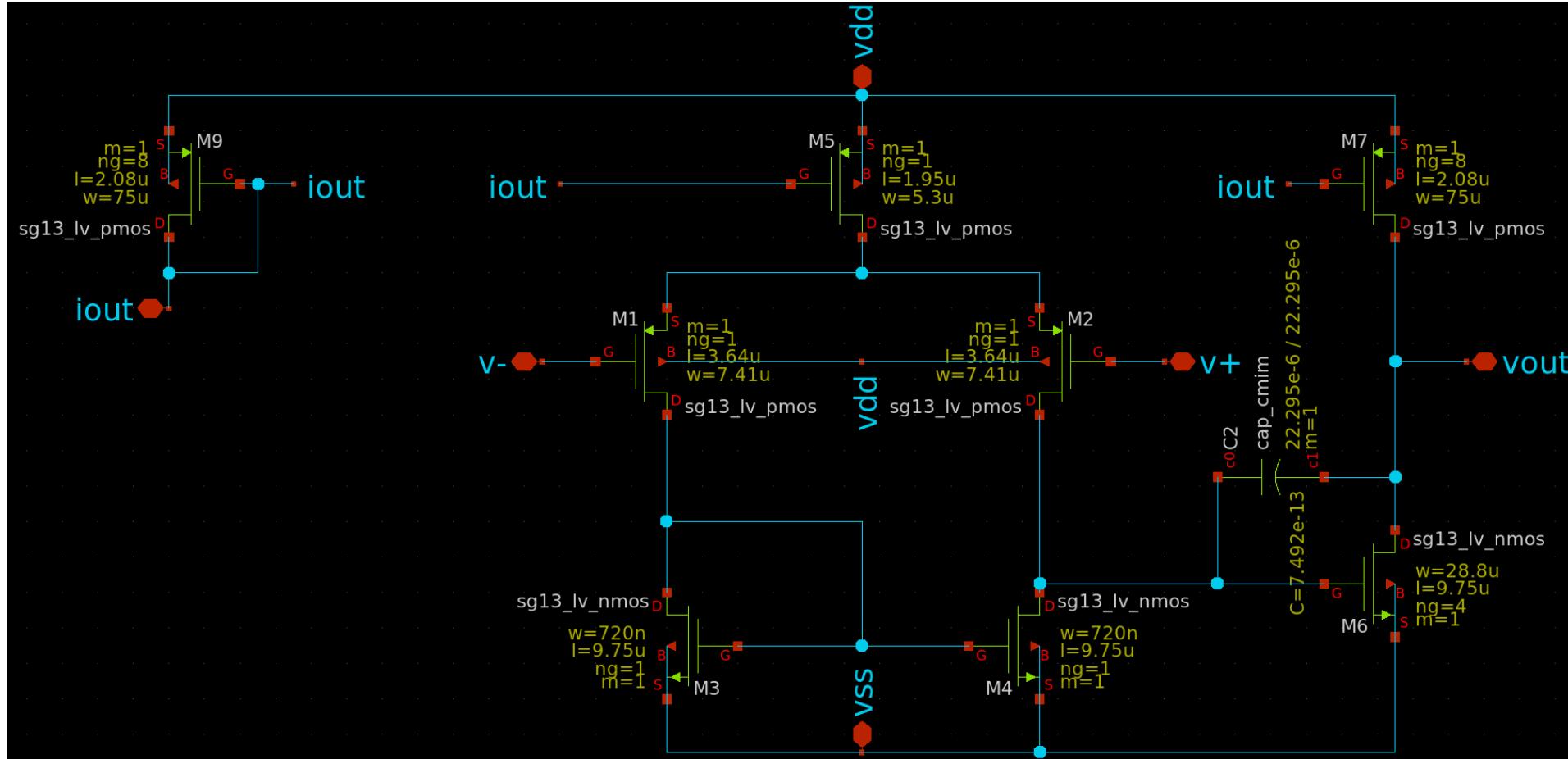
- 0 Was something from yesterday unclear?
- 0 Are we going to fast?
- 0 Do you need more time for the small exercise?

# Overview Of Todays Plan



- 0 OTA Design & Analysis:** Perform DC and AC analysis to evaluate key design metrics.
- 0 Bandgap Voltage Reference:** Conduct DC, transient, and mismatch analysis using Monte Carlo sampling.
- 0 From Design to Tape-Out:** Explore the final layout and discuss the steps required for tape-out in the open-source domain.
- 0 Layout Competition:** Compete to design the best OTA layout with the fewest DRC/LVS errors.

# OTA Design (Two Stage Miller Compensated)



# Design Methodology In OS (Gm/ID)



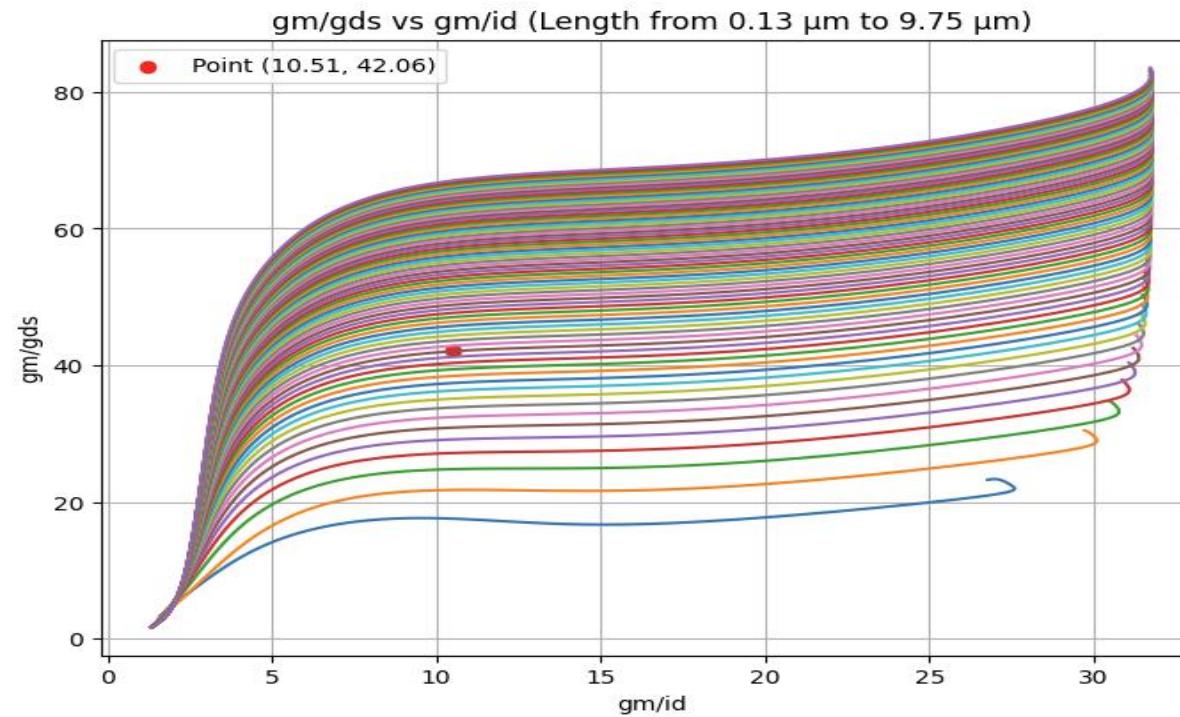
Select Length: Show All

Select gm/id (uncheck for gm/gds)

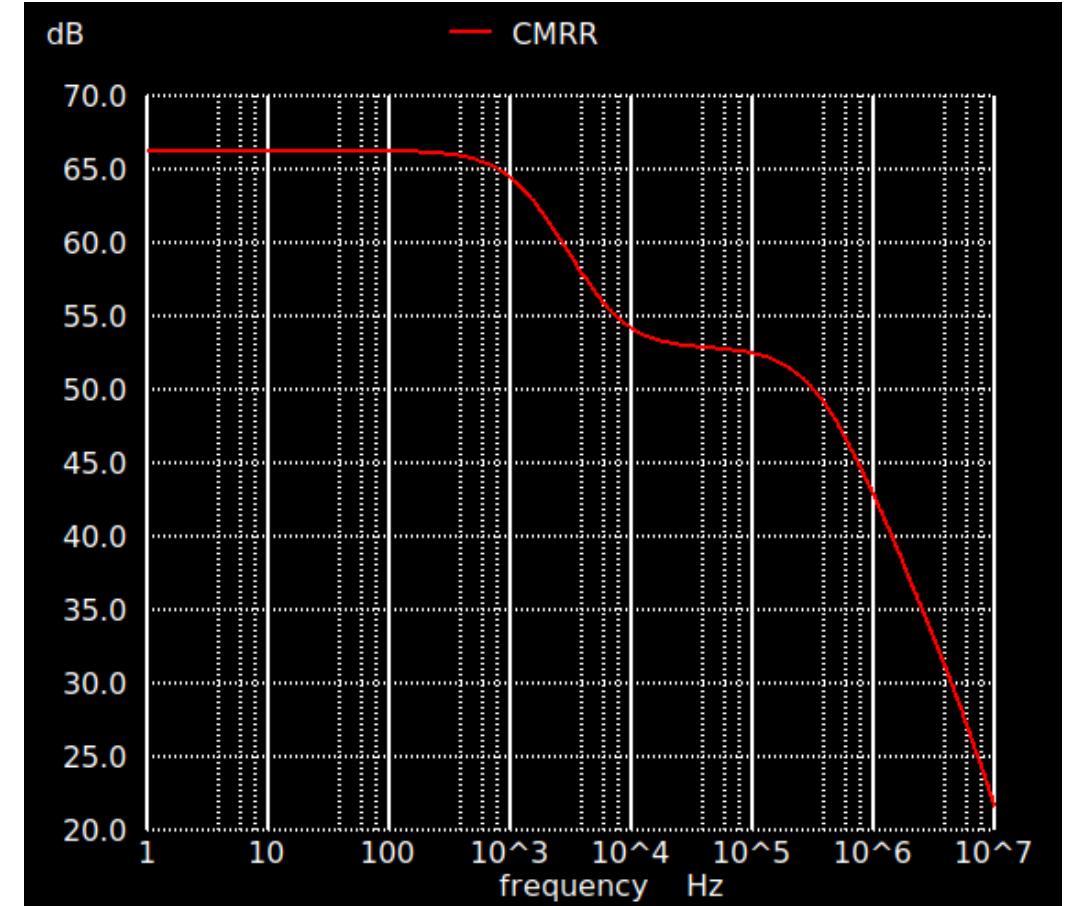
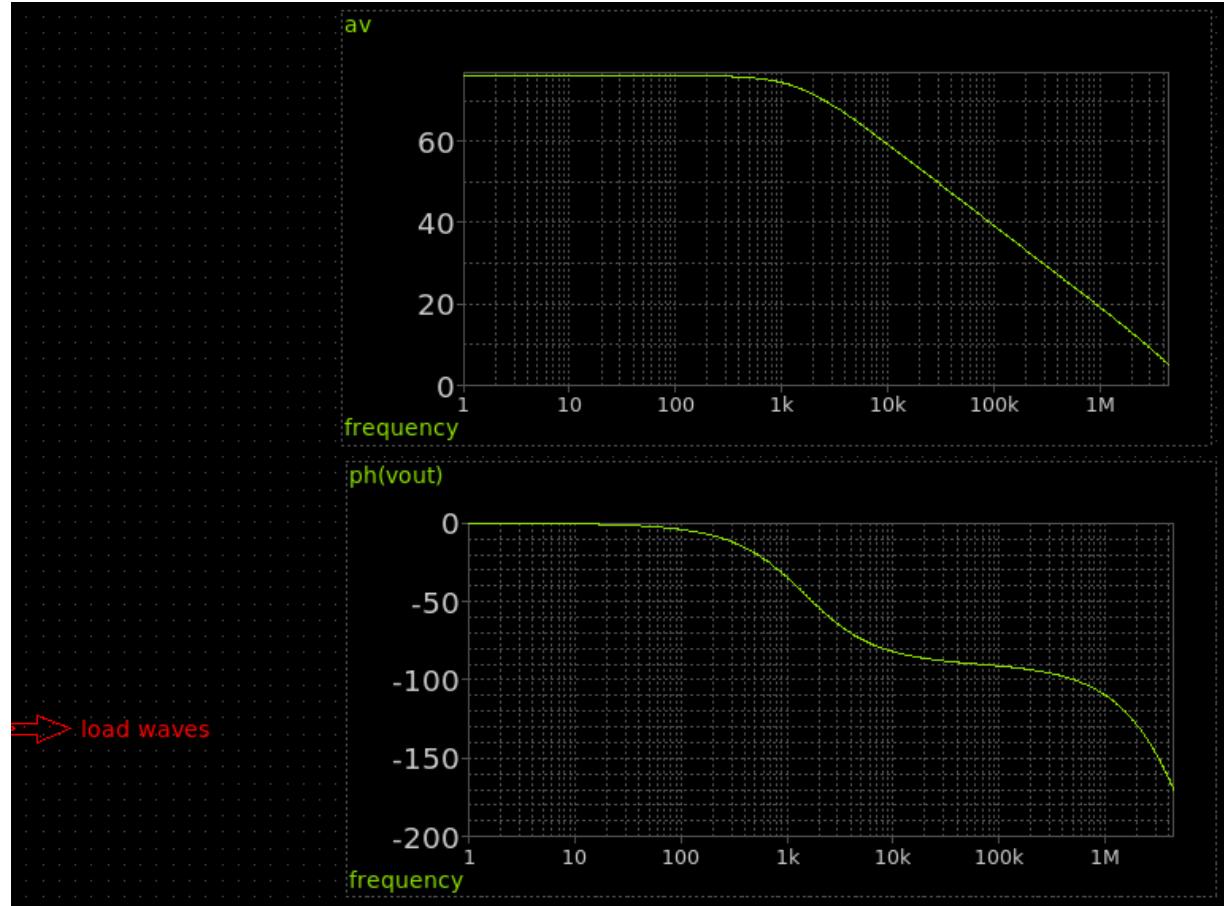
Select gm/id: 10.513728357868652 Set gm/gds: 0

Corresponding gm/gds value: 0.39

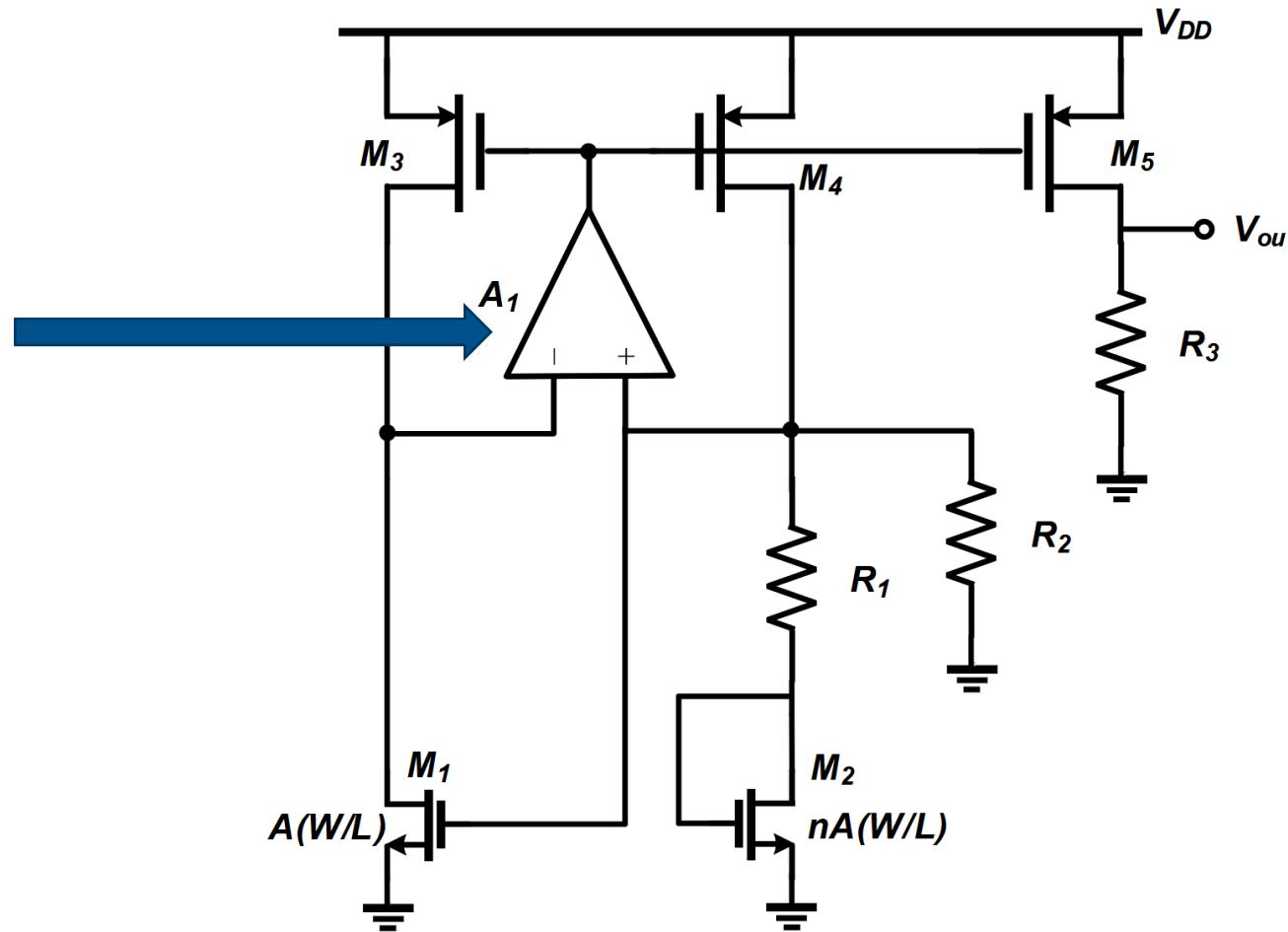
The corresponding gm/gds value for gm/id = 10.51 is: 42.06



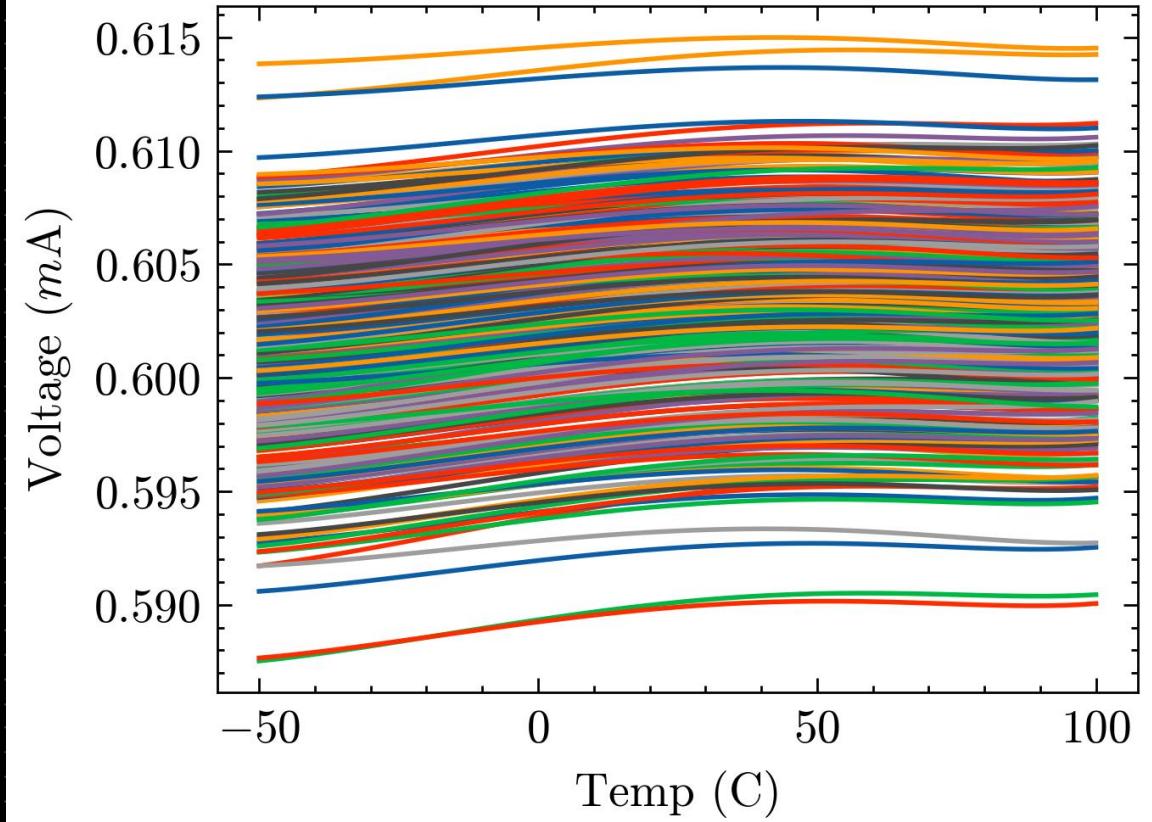
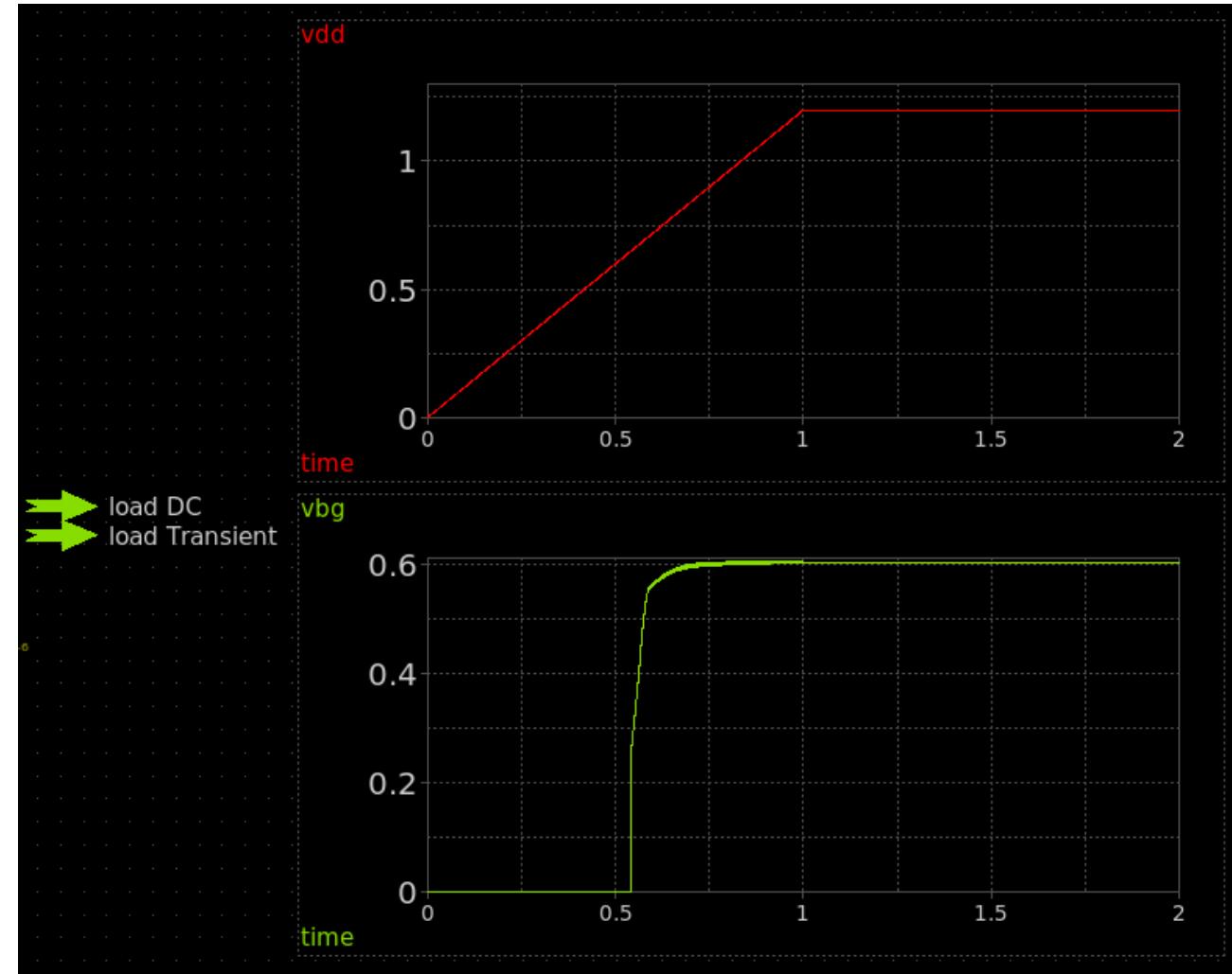
# Testbenches



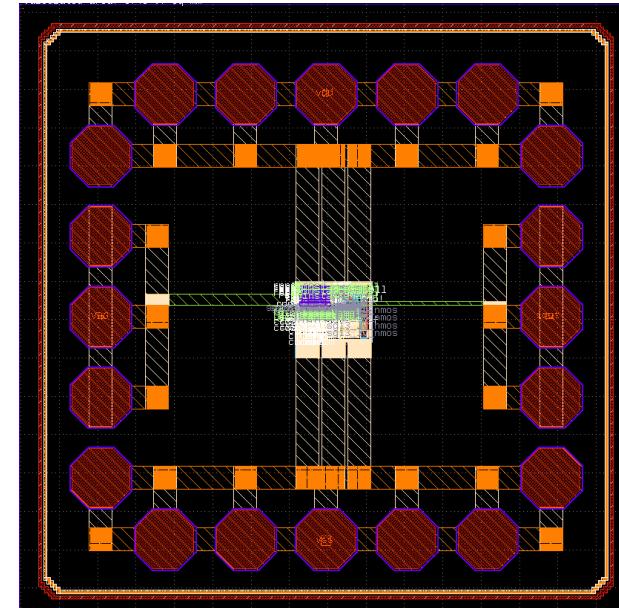
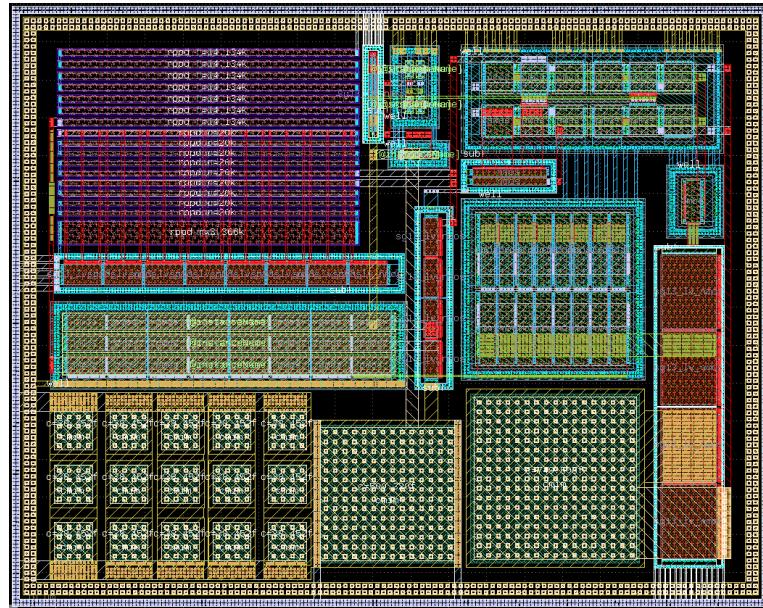
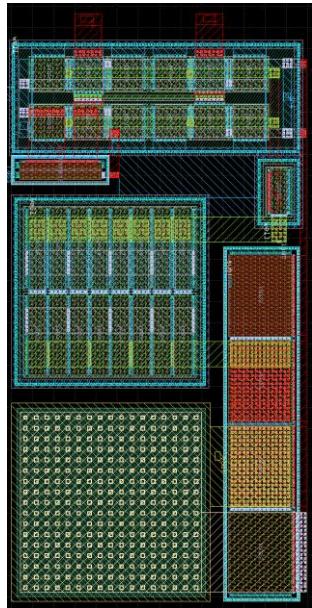
# Bandgap Reference



# Evaluating Overall Performance



# Bandgap Reference (Layout)





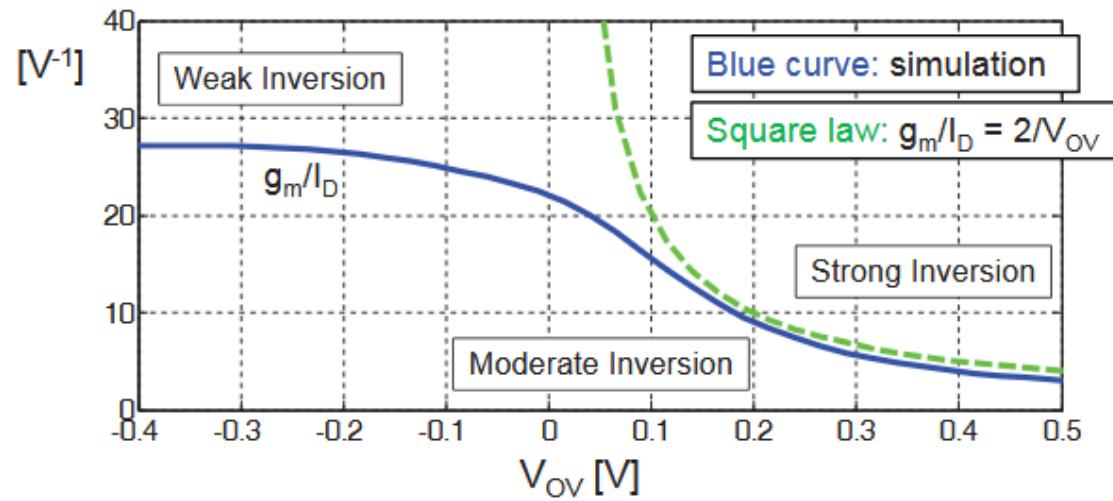
Leibniz Institute  
for high  
performance  
microelectronics

# Expert Talk

## Dr. Prabhat Kumar Dubey

Insights on Model Generation in the Open PDK

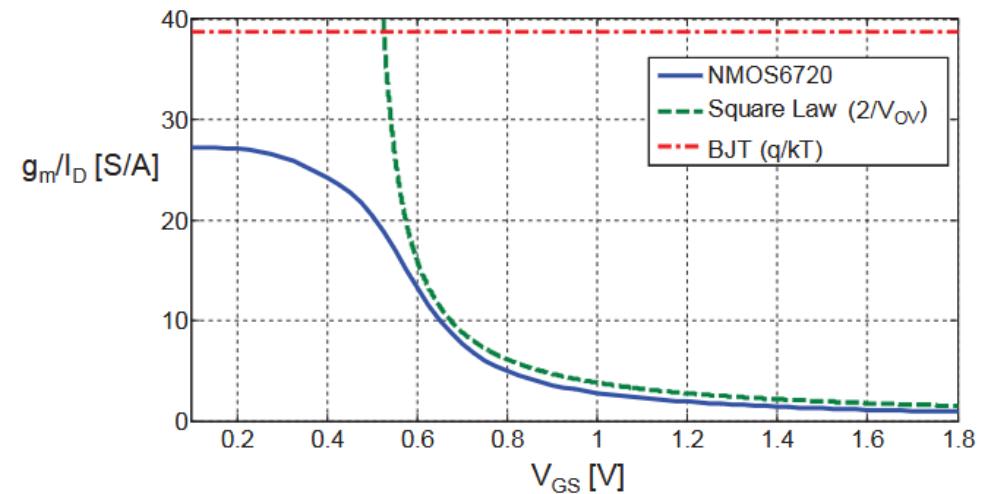
# Small Insights In The Design Phase



R. Walker

ECE/CS 5720/6720 Fall 2017 – Chapter 5

3



Ross Walker ECE/CS 5720/6720 Fall 2017 University of Utah Partly adapted from Stanford's analog circuit design sequence

# Small Insights In The Design Phase ( Live Demo)



The image shows two side-by-side screenshots of GitHub repositories. On the left is the repository `gmid` (Public), which has one branch and no tags. The commit history shows several commits from user `medwatt` over the past few days, detailing changes to `figures`, `src/mosplot`, `tests`, `.gitignore`, `README.md`, and `setup.py`. On the right is the repository `PhillipRambo` (update markdwon for `gmid`), which contains a folder structure with files like `inverter`, `scripting`, `README.md`, `foundations.md`, and two Python files: `sg13_nmos_lv.py` and `sg13_pmos_lv.py`. Two red arrows point from the commit history of the `gmid` repository to the corresponding files in the `PhillipRambo` repository, indicating a comparison or transfer of code.

Original Repository: <https://github.com/medwatt/gmid>

# Small Insights In The Design Phase ( Live Demo)



## -0 Setup Procedure:

Clone metwatts gmid repository

*<https://github.com/metwatt/gmid>*

Create virtual env

*python3 -m venv gmid\_env*

Source the environment

*source gmid\_env/bin/activate*

Inside the gmid repo run

*pip install .*

# Small Insights In The Design Phase ( Live Demo)



```
from mosplot.lookup_table_generator.simulators import NgspiceSimulator, HspiceSimulator
from mosplot.lookup_table_generator import LookupTableGenerator, TransistorSweep
# One of `include_paths` or `lib_mappings` must be specified.
# The rest are optional.

ngspice = NgspiceSimulator(
    # Provide path to simulator if simulator is not in system path.
    simulator_path="ngspice",

    # Default simulation temperature. Override if needed.
    temperature=27,

    # All parameters are saved by default. override if needed.
    parameters_to_save=["id", "vth", "vdsat", "gm", "gds", "vgs"],

    # Files to include with `.LIB`.
    lib_mappings = [
        ("/home/pedersen/IHP-Open-PDK/ihp-sg13g2/libs.tech/ngspice/models/cornerMOSlv.lib", "mos_tt") # Put your own path to the corner lib
    ],

    # If the transistor is defined inside a subcircuit in
    # the library files, you must specify the symbol used (first entry)
    # and the hierarchical name (second entry). Override if needed.
    mos_spice_symbols = ("XM1", "n.xm1.nsg13_lv_nmos"),

    # Specify the width. For devices that do not take a width,
    # you can specify other parameters such as the number of fingers.
    # The keys are exactly those recognized by the model.
    device_parameters = {
        "w": 10e-6,
    }
)

nmos_sweep = TransistorSweep(
    mos_type="nmos",
    vgs=(0, 1.2, 0.01),
    vds=(0, 1.2, 0.01),
    vbs=(0, -1.2, -0.1),
    length = [130e-9, 260e-9, 390e-9, 520e-9, 650e-9, 780e-9, 910e-9, 1040e-9, 1170e-9, 1300e-9, 1430e-9, 1560e-9, 1690e-9, 1820e-9, 1950e-9, 2080e-9, 2210e-9, 2340e-9, 2470e-9, 2600e-9, 2730e-9, 2860e-9, 2990e-9, 3120e-9, 3250e-9, 3380e-9, 3510e-9, 3640e-9, 3770e-9, 3900e-9, 4030e-9, 4160e-9, 4290e-9, 4420e-9, 4550e-9, 4680e-9, 4810e-9, 4940e-9, 5070e-9, 5200e-9, 5330e-9, 5460e-9, 5590e-9, 5720e-9, 5850e-9, 5980e-9, 6110e-9, 6240e-9, 6370e-9, 6500e-9, 6630e-9, 6760e-9, 6890e-9, 7020e-9, 7150e-9, 7280e-9, 7410e-9, 7540e-9, 7670e-9, 7800e-9, 7930e-9, 8060e-9, 8190e-9, 8320e-9, 8450e-9, 8580e-9, 8710e-9, 8840e-9, 8970e-9, 9100e-9, 9230e-9, 9360e-9, 9490e-9, 9620e-9, 9750e-9, 9880e-9]
)
```

[IHP-AnalogAcademy/modules/module\\_0\\_foundations/sg13\\_nmos\\_lv.py](#)

[IHP-AnalogAcademy/modules/module\\_0\\_foundations/sg13\\_pmos\\_lv.py](#)

# Small Insights In The Design Phase ( Live Demo)



## -0 Setup Procedure:

Create the LUTs from module\_0\_foundations/sg13\_nmos\_lv.py

*python3 sg13\_nmos\_lv.py*

Now navigate to module\_0\_foundations/scripting and run

*pip install -r requirements.txt*

From the same location launch the jupyter lab file

*jupyter lab gmid\_test.ipynb*

*For the jupyter lab file you may only get a link you can paste into chrome/firefox*

# Small Insights In The Design Phase ( Live Demo)



```
----- Sample Netlist -----  
* Lookup Table Generation *  
.lib '/home/pedersen/IHP-Open-PDK/ihp-sg13g2/libs.tech/ngspice/models/cornerMOSlv.lib' mos_tt  
VGS NG 0 DC=0  
VBS NB 0 DC=0.0  
VDS ND 0 DC=0  
XM1 ND NG 0 NB sg13_lv_nmos L=1.3e-07 w=1e-05  
.options TEMP = 27  
.options TNOM = 27  
.control  
save i(vds)  
save @n.xm1.ng13_lv_nmos[vth]  
save @n.xm1.ng13_lv_nmos[vdsat]  
save @n.xm1.ng13_lv_nmos[gm]  
save @n.xm1.ng13_lv_nmos[gds]  
dc VDS 0 1.2 0.01 VGS 0 1.2 0.01  
let i_vds = -i(vds)  
write output all  
.endc  
.end  
-----  
Total simulation jobs: 65  
Progress: 1/65 jobs completed  
Progress: 2/65 jobs completed  
Progress: 3/65 jobs completed  
Progress: 4/65 jobs completed  
Progress: 5/65 jobs completed  
Progress: 6/65 jobs completed  
Progress: 7/65 jobs completed  
Progress: 8/65 jobs completed  
Progress: 9/65 jobs completed  
Progress: 10/65 jobs completed  
Progress: 11/65 jobs completed  
Progress: 12/65 jobs completed  
Progress: 13/65 jobs completed  
Progress: 14/65 jobs completed  
Progress: 15/65 jobs completed  
Progress: 16/65 jobs completed  
Progress: 17/65 jobs completed  
Progress: 18/65 jobs completed  
Progress: 19/65 jobs completed  
Progress: 20/65 jobs completed  
Progress: 21/65 jobs completed
```

Saves the data in LUTs folder in the .npz format

# Small Insights In The Design Phase ( Live Demo)



```
[101]: import matplotlib.pyplot as plt
import numpy as np
from mosplot.plot import load_lookup_table, Mosfet, Expression
import ipywidgets as widgets
from ipywidgets import interactive
from ipywidgets import interactive_output, HBox, VBox
import matplotlib.ticker as ticker

[102]: lookup_table_nmos = load_lookup_table('../sg13_nmos_lv_LUT.npz')
lookup_table_pmos = load_lookup_table('../sg13_pmos_lv_LUT.npz')

[103]: print(lookup_table.keys())
dict_keys(['sg13_lv_nmos ', 'sg13_lv_pmos', 'description', 'simulator', 'parameter_names', 'device_parameters'])

[108]: nmos = Mosfet(lookup_table=lookup_table_nmos, mos="sg13_lv_nmos ", vbs=0.0, vds=0.6)
pmos = Mosfet(lookup_table=lookup_table_pmos, mos="sg13_lv_pmos", vbs=0.0, vds=-0.6, vgs=(-1.2, -0.15))

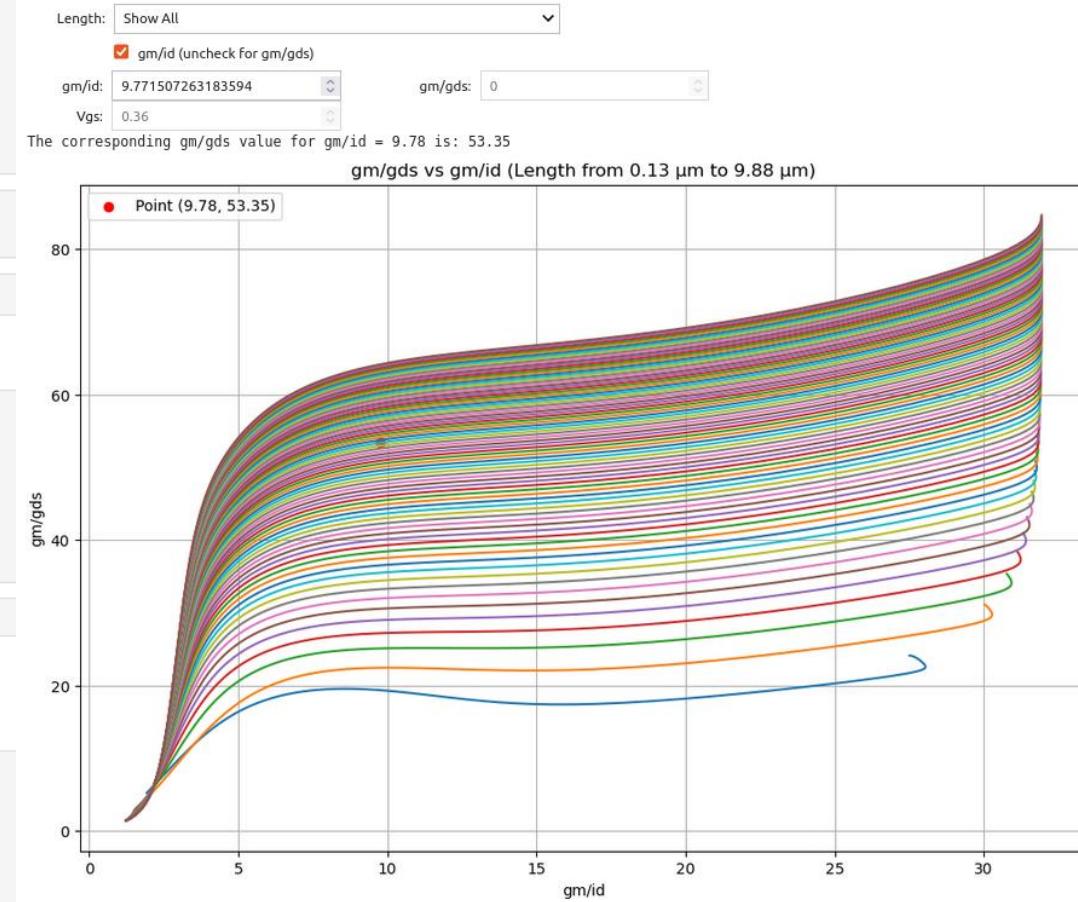
rows_0, cols_0 = np.shape(nmos.extracted_table['gm']) # just for getting the shape of the data
rows_1, cols_1 = np.shape(pmos.extracted_table['gm']) # just for getting the shape of the data
reshaped_lengths_nmos = np.tile(nmos.length[:, np.newaxis], (1, cols_0))
reshaped_lengths_pmos = np.tile(pmos.length[:, np.newaxis], (1, cols_1))

def plot_data_vs_data(x_values, y_values, z_values, length, x_axis_name, y_axis_name='y', y_multiplier=1, log=False):...
```

## NMOS GMID

```
[110]: width_values = nmos.width
id_values = nmos.extracted_table['id']
gm_values = nmos.extracted_table['gm']
gds_values = nmos.extracted_table['gds']
vgs_values = nmos.extracted_table['vgs']

plot_data_vs_data(gm_values/id_values, gm_values/gds_values, vgs_values, reshaped_lengths_nmos, 'gm/id', 'gm/gds')
```



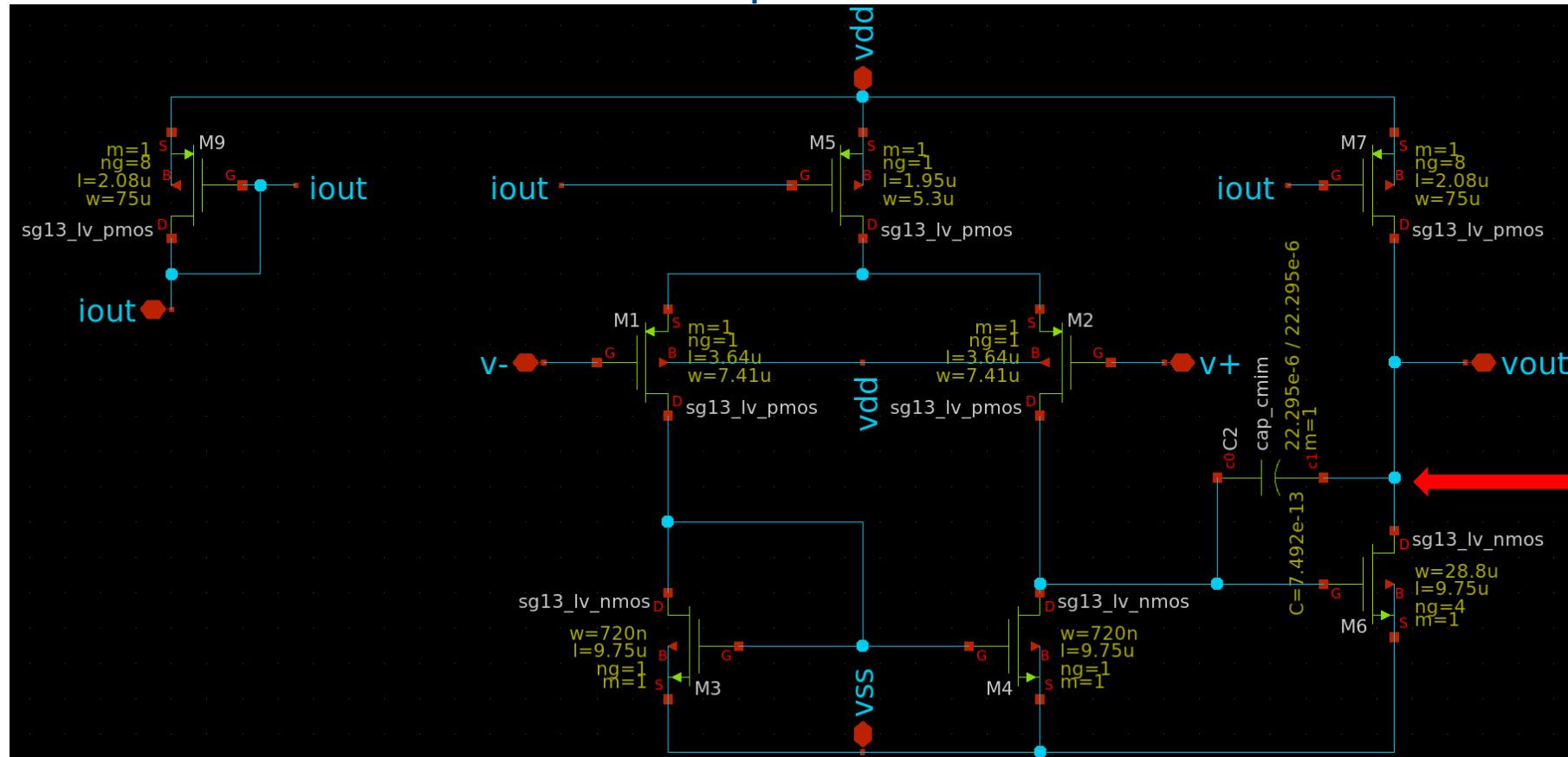
# Part 1

Miller Compensated OTA-Design

Reference: /modules/module\_1\_bandgap\_reference/part\_1 OTA

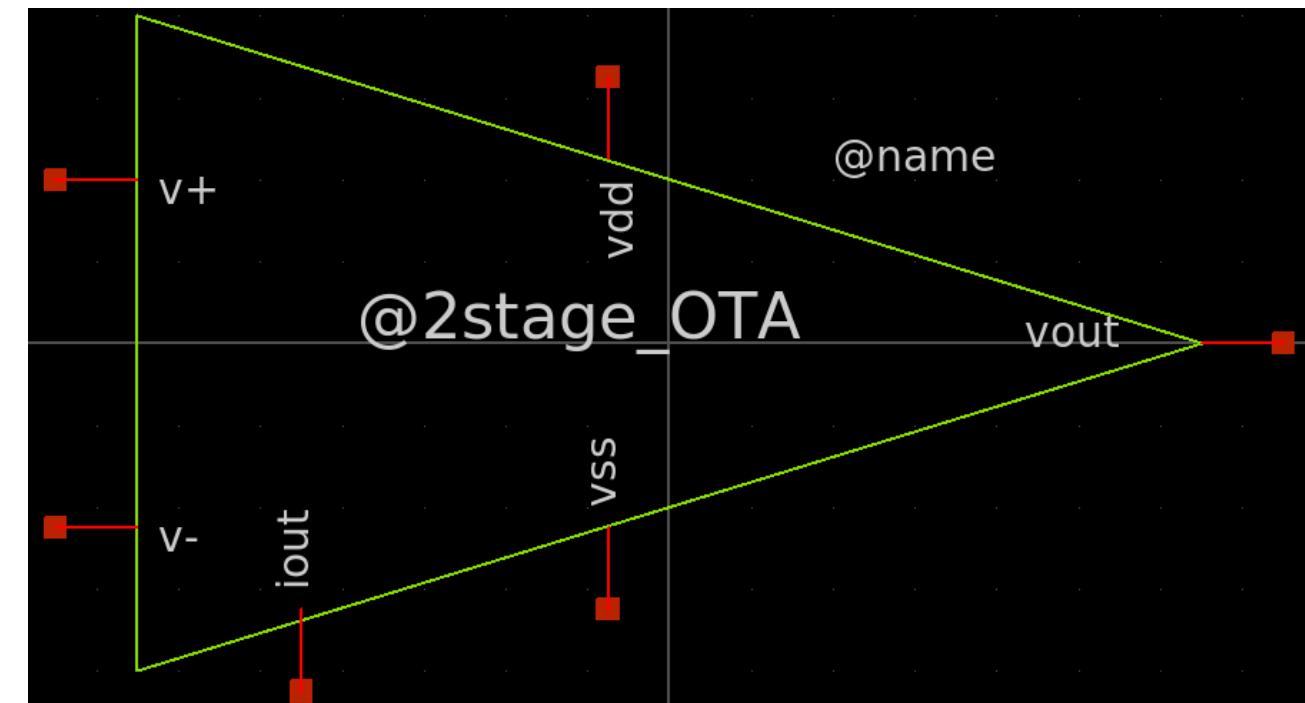
# Schematic & Symbol

- 0 Create a folder for this module and organize yourself within
- 0 Create a Schematic for the OTA seen in the picture



# Schematic & Symbol

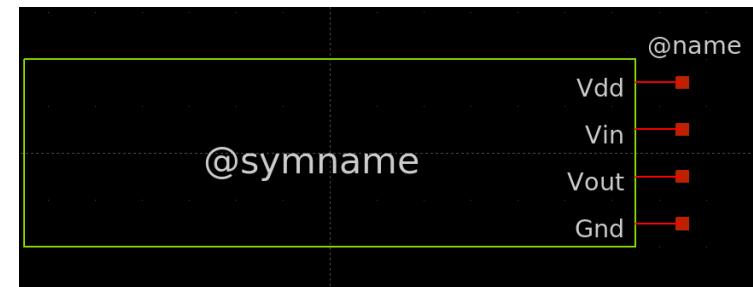
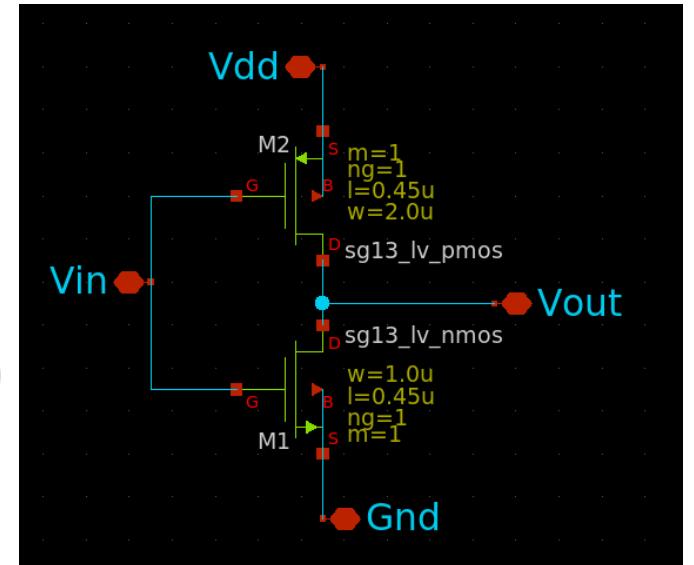
- 0 Create the OTA symbol in your preferred style
- 0 The next slide provides a reminder on how to create a symbol



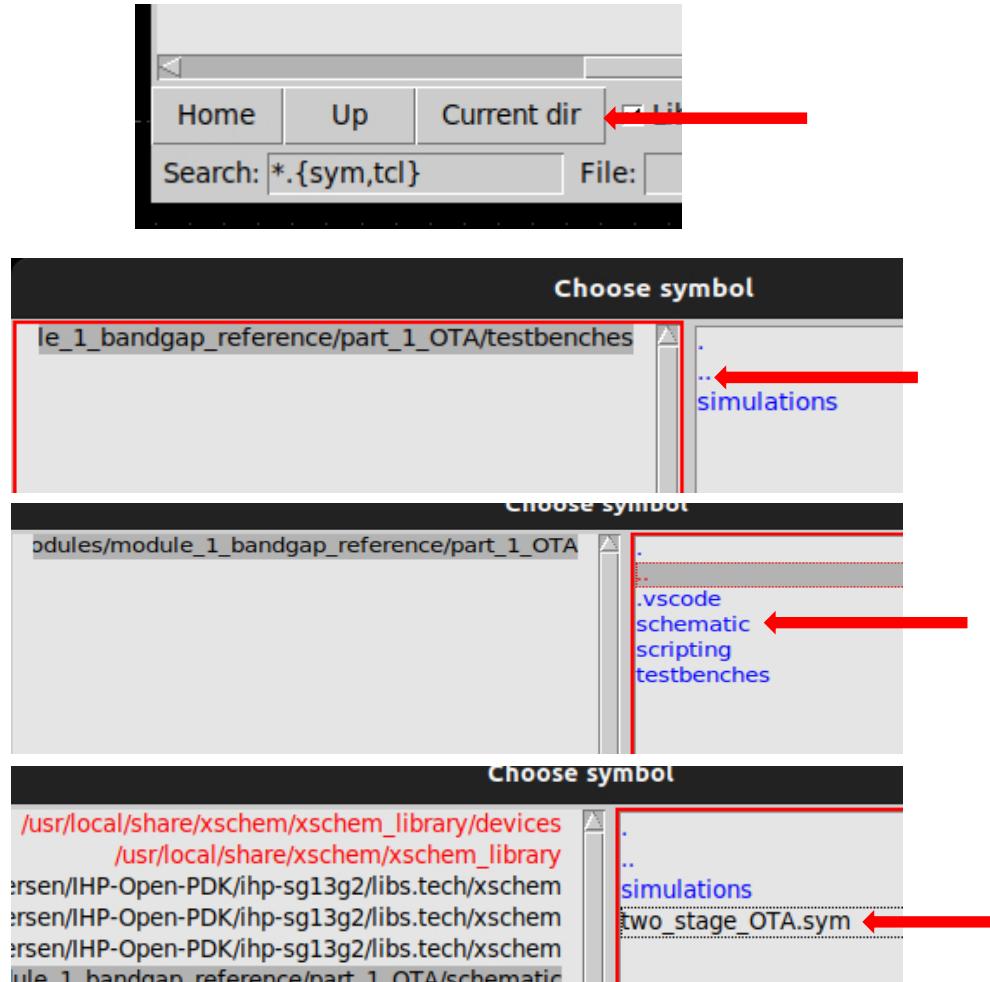
# Creating a Symbol (from yesterday)



- 0 For Creating the Symbol press **A**
- 0 This creates a symbol file in the directory of the schematic, navigate to this file and open it by writing **(xschem inverter.sym)**
- 0 Now you can drag the already existing lines to create the boundaries or use the tools seen in the toolbar to define custom shapes



# Accessing the Symbol (Two Options)



The image shows a terminal window with the following content:

```

→ testbenches git:(main) ✘ ls
ota_testbench_mc_mis.sch ota_testbench.sch simulations xschemrc

```

The terminal then displays the contents of the `xschemrc` file:

```

Open Save ~projects/IHP-AnalogAcademy/modules/module_1_bandgap_reference/part_1_OTA/testbenches xschemrc
1 # xschemrc - Custom configuration file for xschem
2 # This file sources another xschemrc file from a known location
3
4 # Source the base configuration from a known location
5 source $::env(PDK_ROOT)/$::env(PDK)/libs.tech/xschem/xschemrc
6
7 # (Optional) Add any custom overrides or extensions below
8 # set xschem_library_path /home/user/my_libs
9 # set xschem_gui_font "Monospace 10"
10
11 ##### include skywater libraries. Here I use [pwd]. This works if I start xschem from here.
12 #####only if you dont have this setup already #####
13 ##append XSCHM_LIBRARY_PATH :[file dirname [info script]]
14
15
16 ##### Add custom libraries (directories with .lib files)
17 append XSCHM_LIBRARY_PATH :$PDK_ROOT/ihp-sg13g2/libs.tech/xschem
18 append XSCHM_LIBRARY_PATH ../../schematic/
19 append XSCHM_LIBRARY_PATH ../../part_2_full_bgr/schematic/verilog/veriloga_tbs/

```

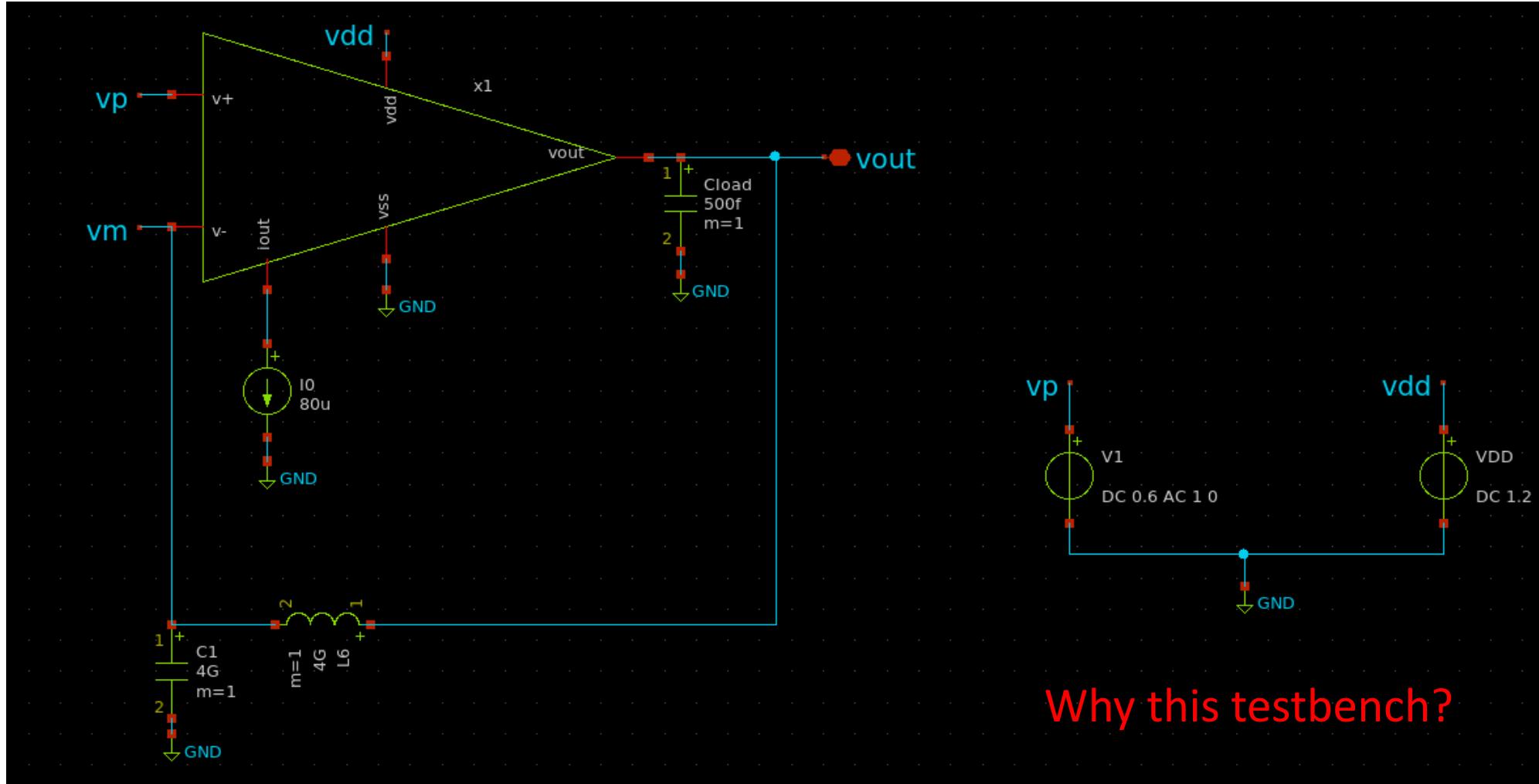
Finally, the terminal shows the search path:

```

Choose
/usr/local/share/xschem/xschem_library/devices
/usr/local/share/xschem/xschem_library
ersen/IHP-Open-PDK/ihp-sg13g2/libs.tech/xschem
ersen/IHP-Open-PDK/ihp-sg13g2/libs.tech/xschem
ersen/IHP-Open-PDK/ihp-sg13g2/libs.tech/xschem
file_1_bandgap_reference/part_1 OTA/schematic
ce/part_2_full_bgr/schematic/verilog/veriloga_tbs
.
```

A red arrow points to the `two_stage_OTA.sym` file in the list.

# OTA Testbench



# OTA Testbench: Finish The Testbench



## Model Definition

```
Path: /usr/local/share/xschem/xschem_library/devices
Symbol devices/code_shown.sym      OK      Cancel
 No change properties  Preserve unchanged props  Copy cell
name=MODEL only_toplevel=true
format="tcleval( @value )"
value=""
.lib $::SG13G2_MODELS/cornerCAP.lib cap_typ ←
.lib cornerMOSlv.lib mos_tt
"
```

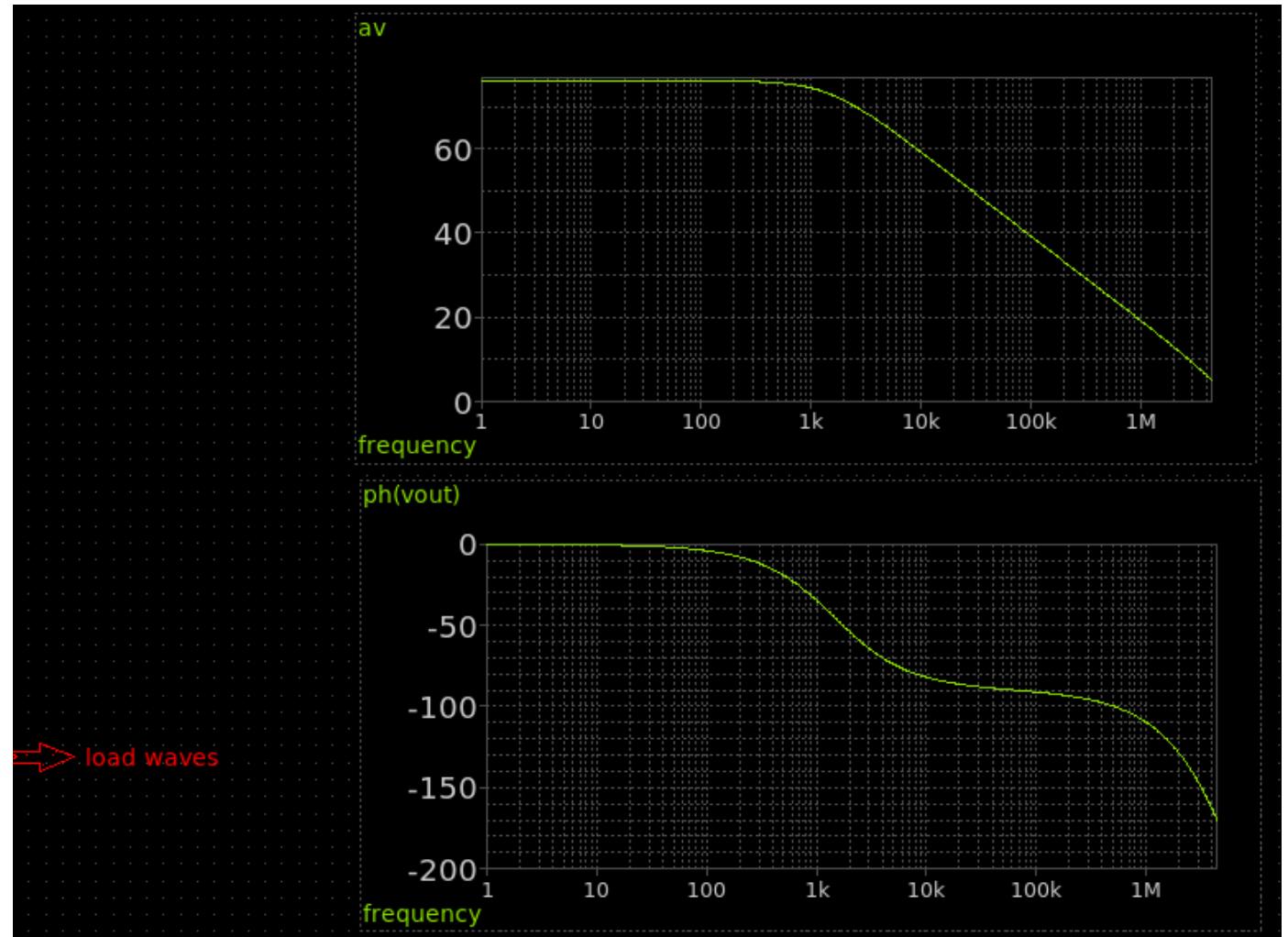
## Simulation Code

```
name=NGSPICE only_toplevel=false
value=""
.control
op
save all
write tb OTA_op.raw
.endc

.control
op
ac dec 100 1 10e6
save all
let Av = db(vout))
let phase    = 180*cph(vout)/pi
write output_file.raw
.endc
"
```

DC Sim

AC Sim



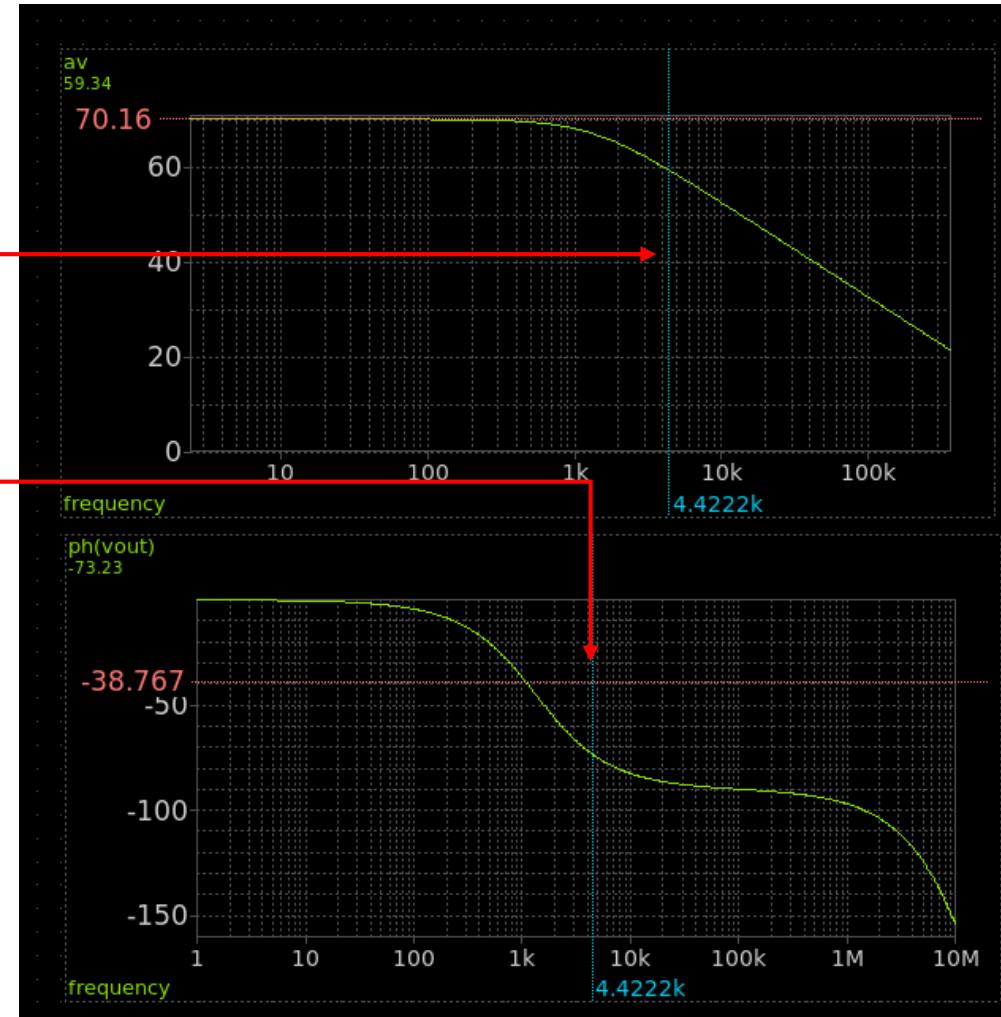
# OTA Testbench: Finish The Testbench



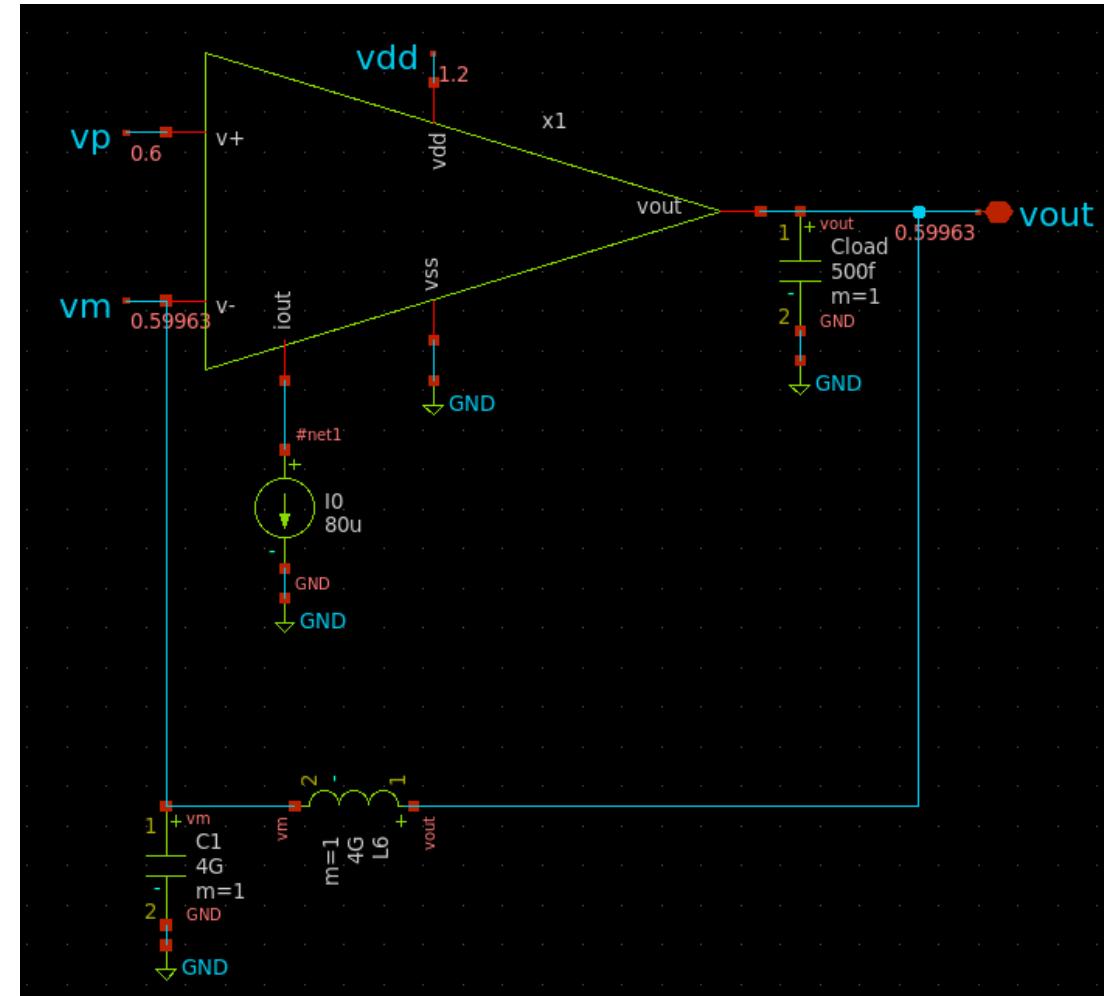
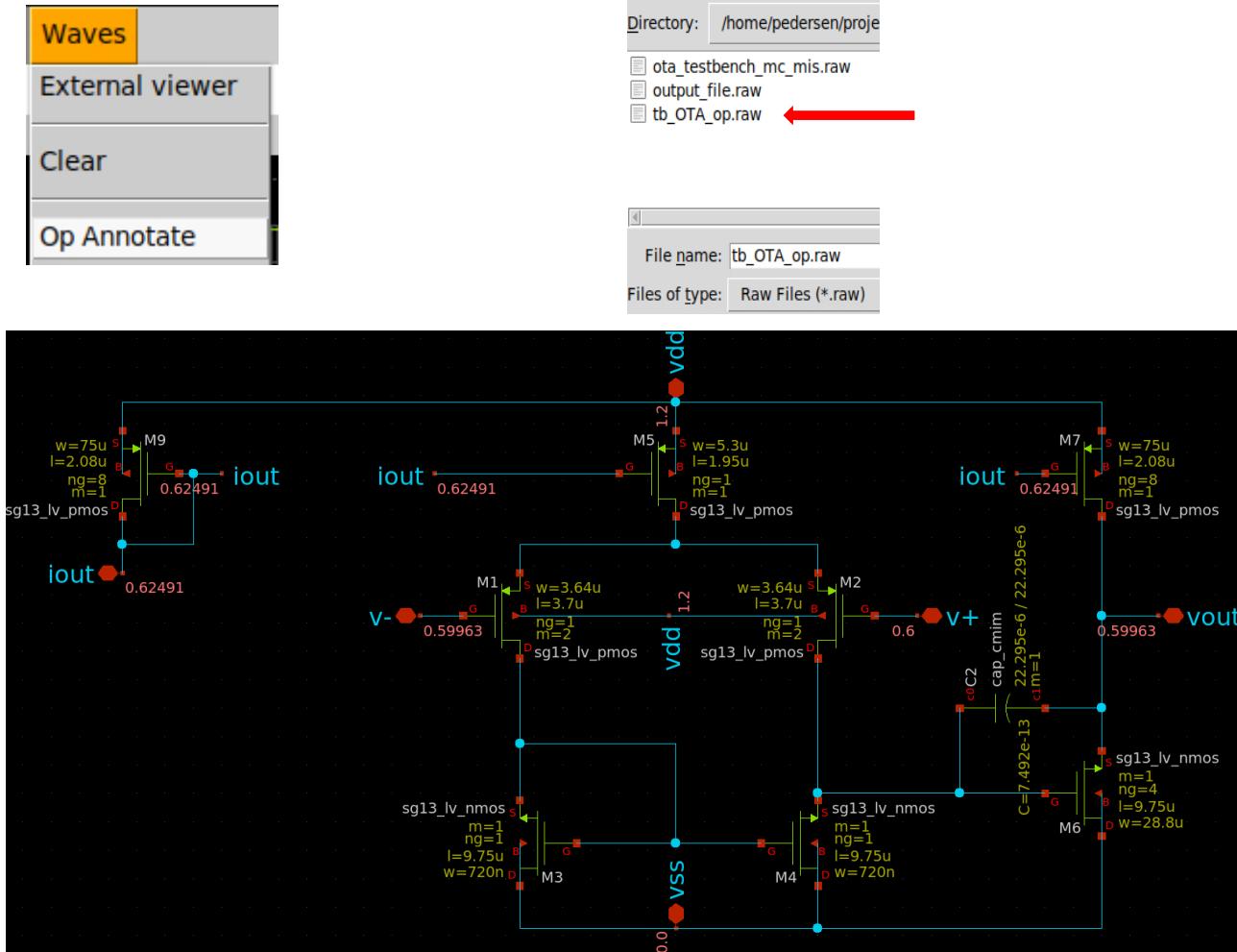
**Shortcuts for measuring in plot:**

*Vertical line: A*

*Horizontal Line: Shift + A*



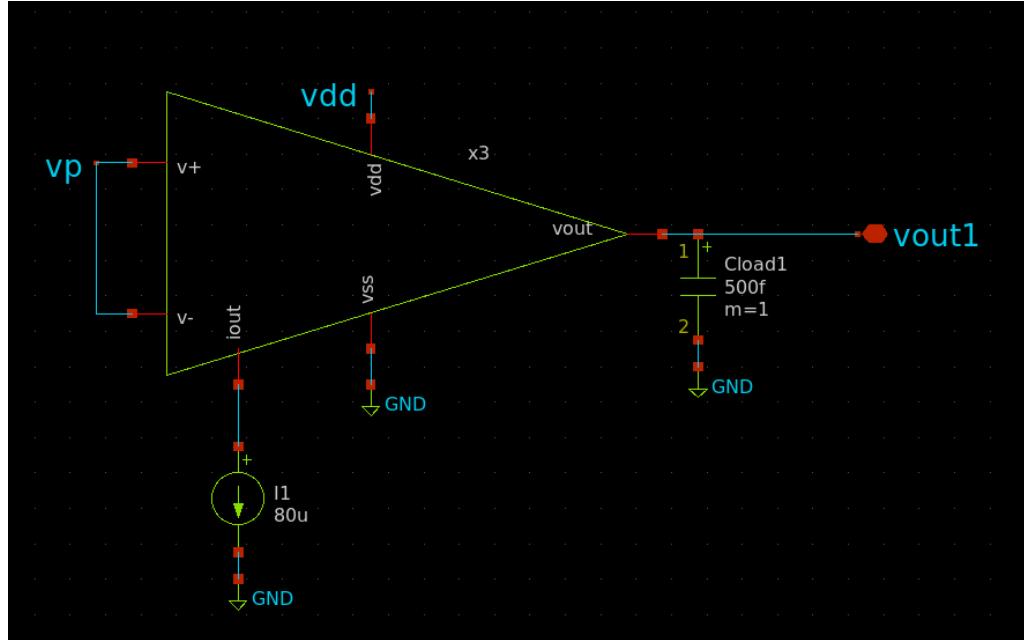
# OTA Testbench: Op Annotation



# OTA Testbench: CMRR & PSRR

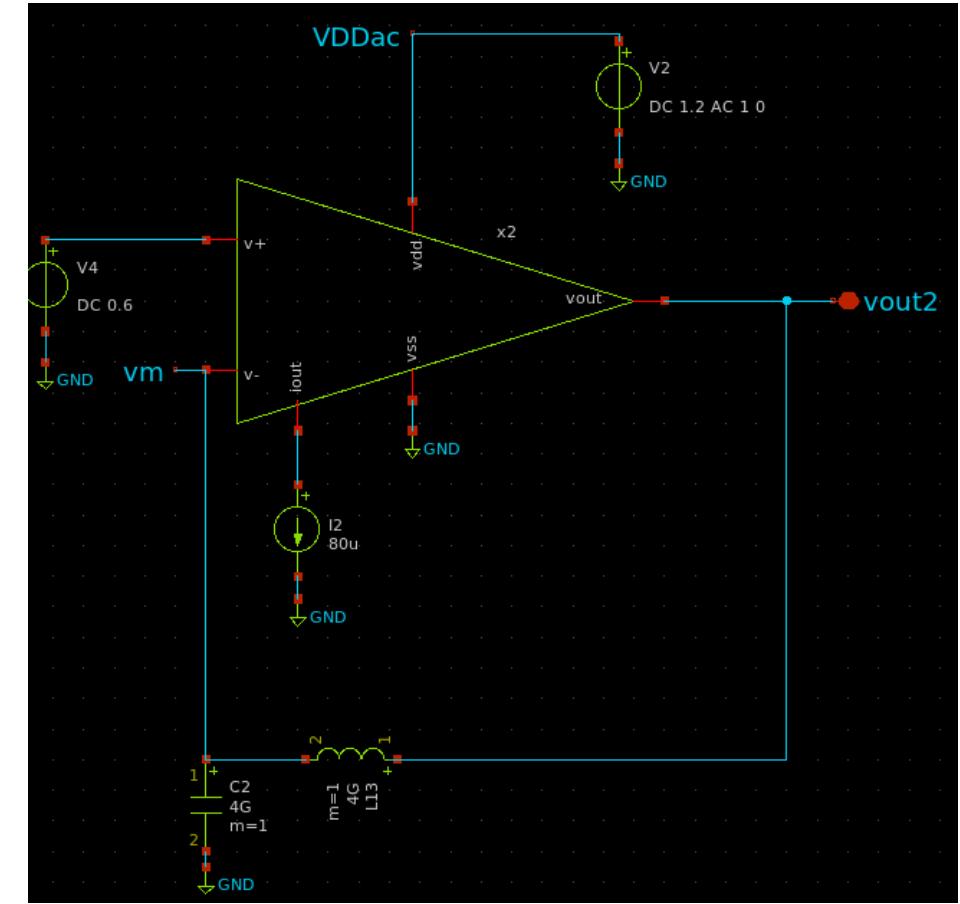


## HINTS



$$CMRR_{dB} = 20 \log_{10} \left( \frac{A_{diff}}{A_{cm}} \right)$$

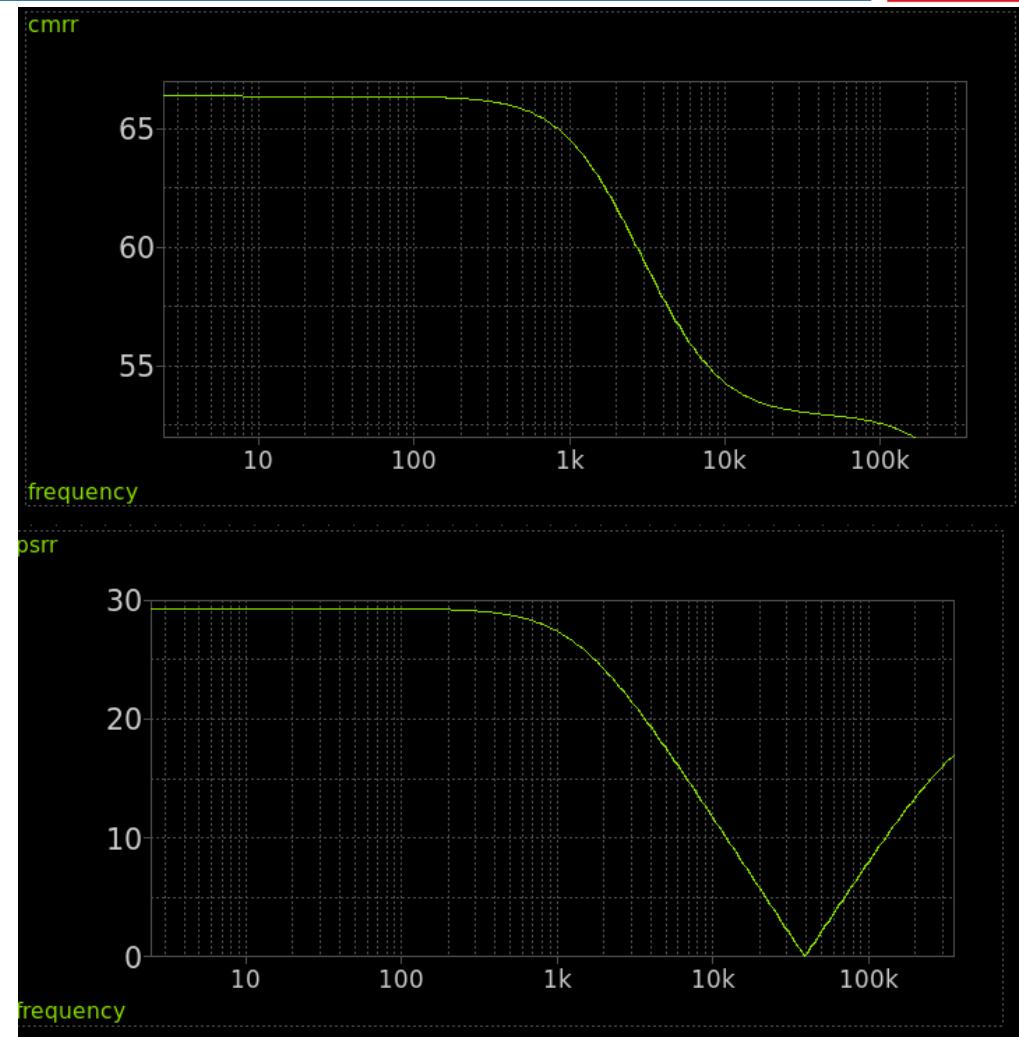
$$PSRR_{dB} = 20 \log_{10} \left( \frac{\Delta V_{out}}{\Delta V_{DD(AC)}} \right)$$



# OTA Testbench: CMRR & PSRR



```
.control  
op  
ac dec 100 1 10e6  
save all  
  
let Av = db(v(vout))  
  
let PSRR = db(v(vout2)/v(VDDac))  
  
let CMRR = db((v(vout)/v(vp))/(v(vout1)/v(vp)))  
  
let phase = 180*cph(vout)/pi  
  
write output_file.raw  
.endc
```



# Catching Up / Lunch Time !

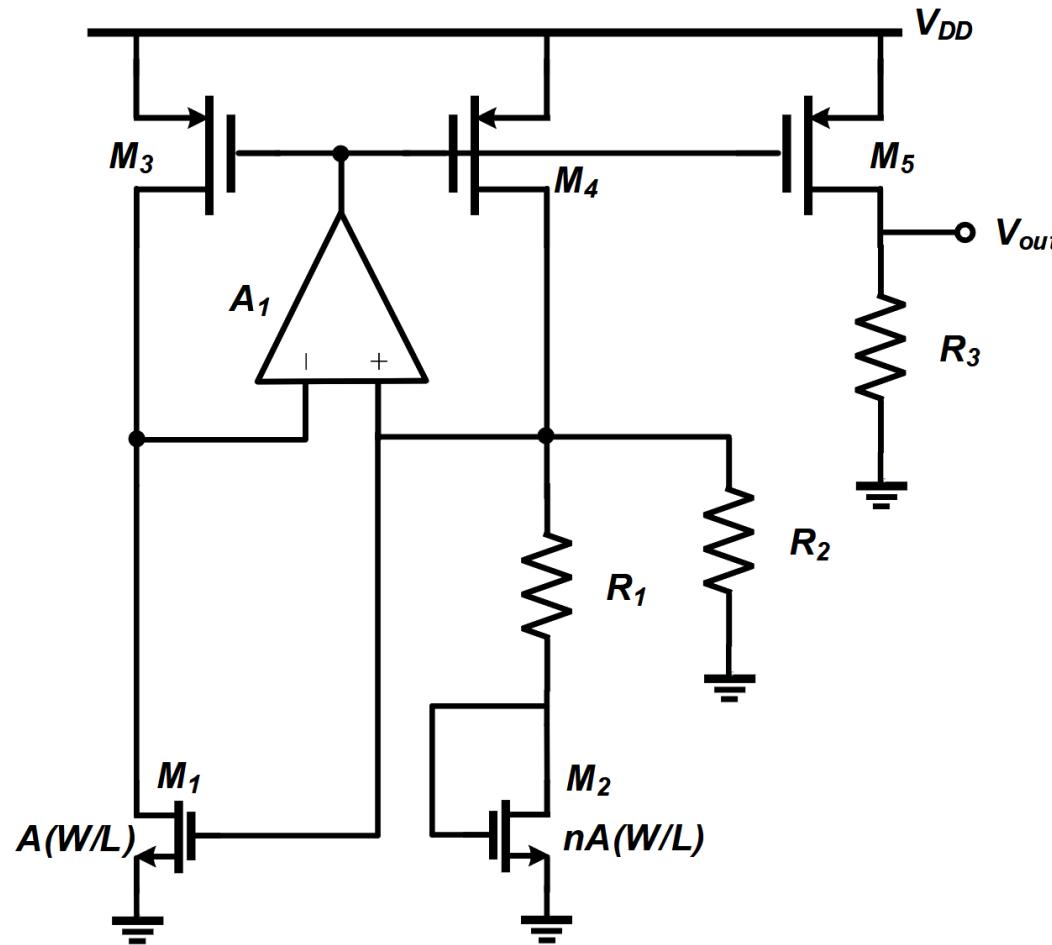
Next session:  
Bandgap Reference Design

# Part 2

Bandgap Reference Design

Reference: /modules/module\_1\_bandgap\_reference/part\_2\_full\_bgr

# Bandgap Reference

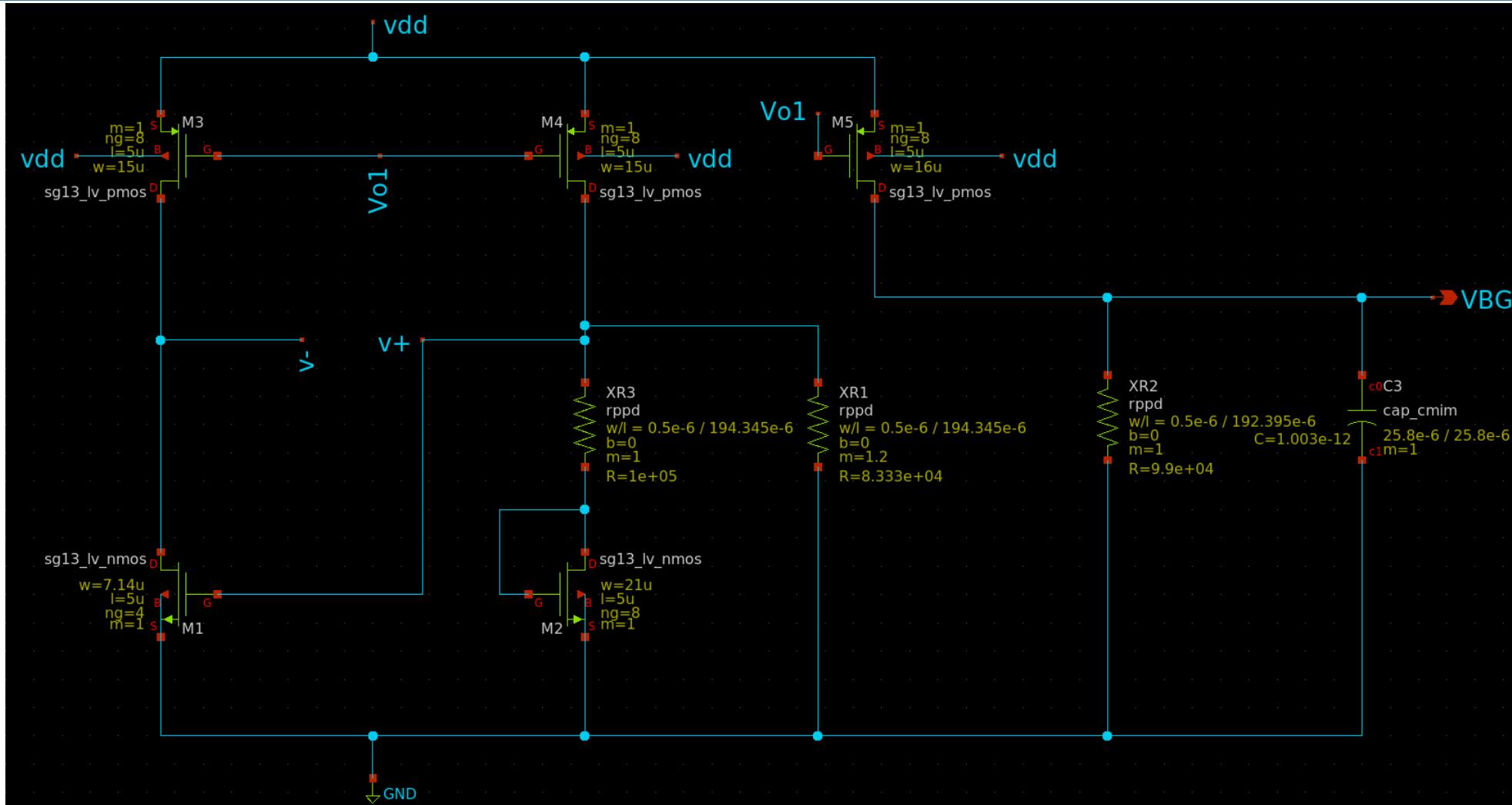


# Bandgap Reference

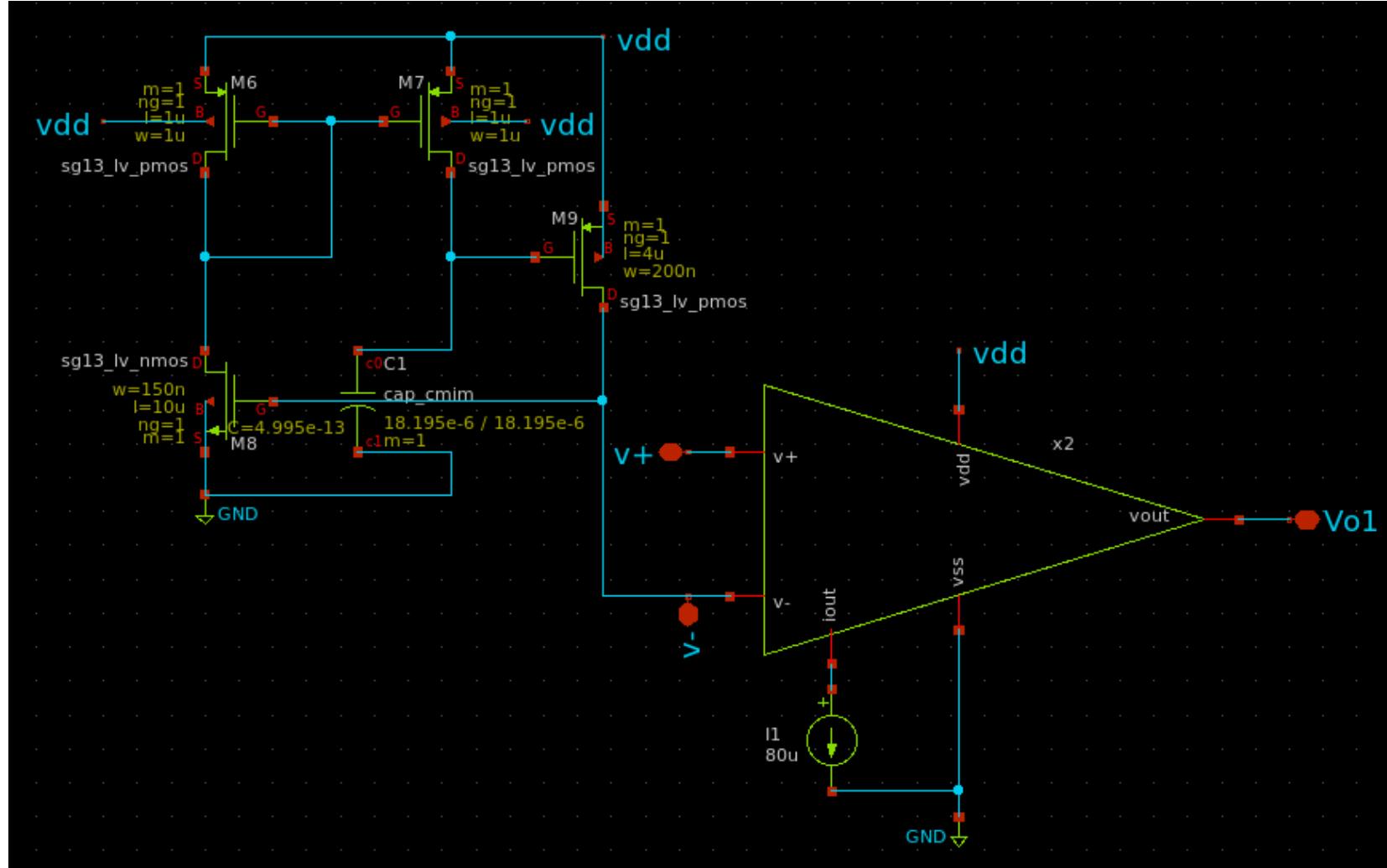


| Resistor Type | Material                        | Sheet Resistance | Temperature Coefficient (TC) | Applications                                |
|---------------|---------------------------------|------------------|------------------------------|---|
| Rsil          | Salicided n-doped polysilicon   | 7 Ω/□            | 3100 ppm/K                   | Low-resistance paths, non-precision use     |
| → Rppd        | Unsalicided p-doped polysilicon | 260 Ω/□          | 170 ppm/K                    | Precision and temperature-sensitive designs |
| Rhigh         | High-resistance structures      | Very high        | Variable                     | High-impedance paths                        |

# Bandgap Reference Core Schematic



# Start-up Circuit Schematic



# Import Of Models



```
name=MODEL only_toplevel=true  
format="tcleval( @value )" value="  
.lib $::SG13G2_MODELS/cornerCAP.lib cap_typ  
.lib $::SG13G2_MODELS/cornerRES.lib res_typ  
.lib cornerMOSlv.lib mos_tt "
```

# Simulation



## DC Simulation

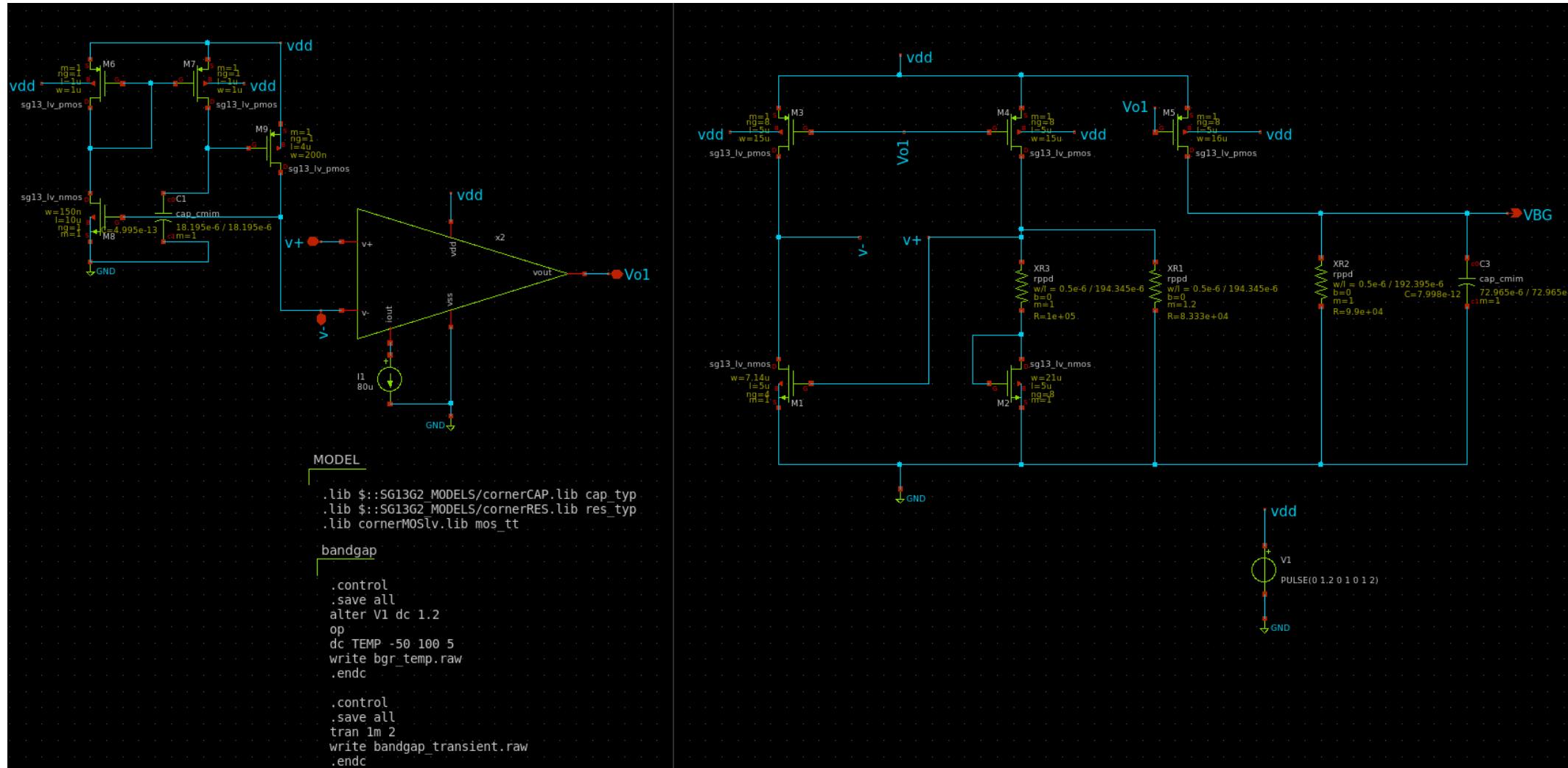
```
.control  
.save all  
alter V1 dc 1.2  
op  
dc TEMP -50 100 5  
write bgr_temp.raw .endc
```

## Transient Simulation

```
.control .save all  
tran 1m 2  
write bandgap_transient.raw .endc
```

VDD → "PULSE(0 1.2 0 1 0 1 2)"

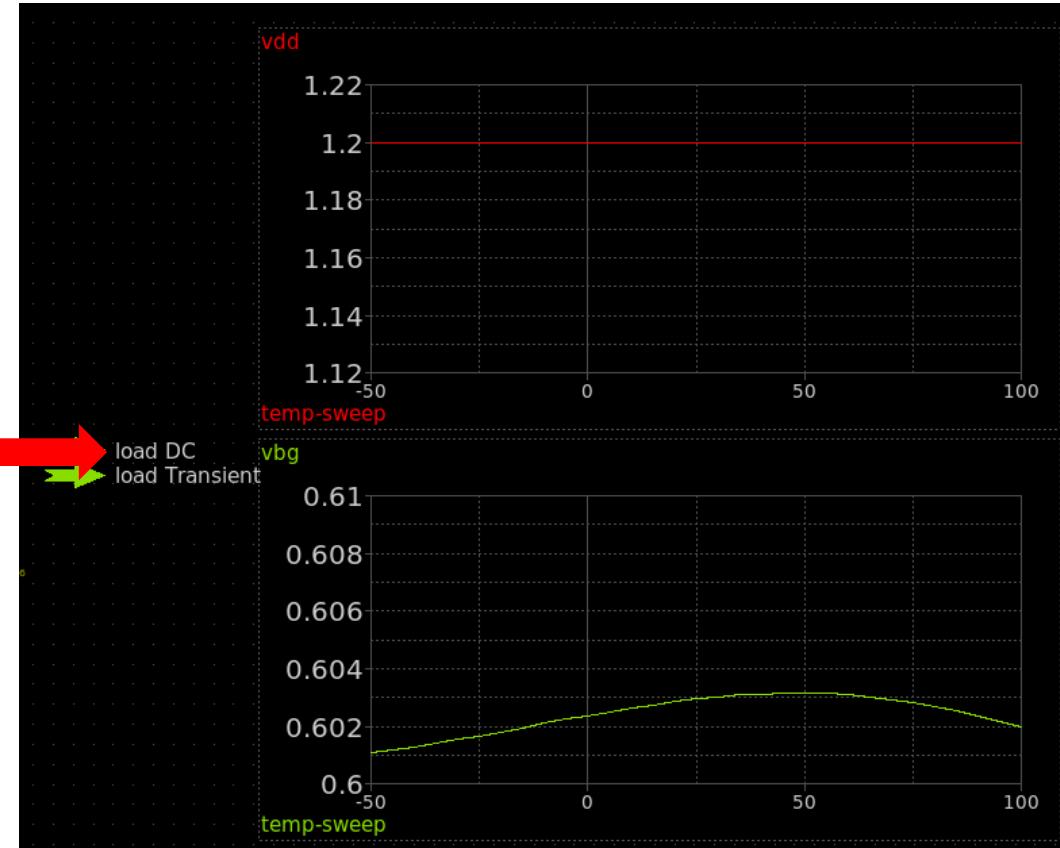
# Simulation



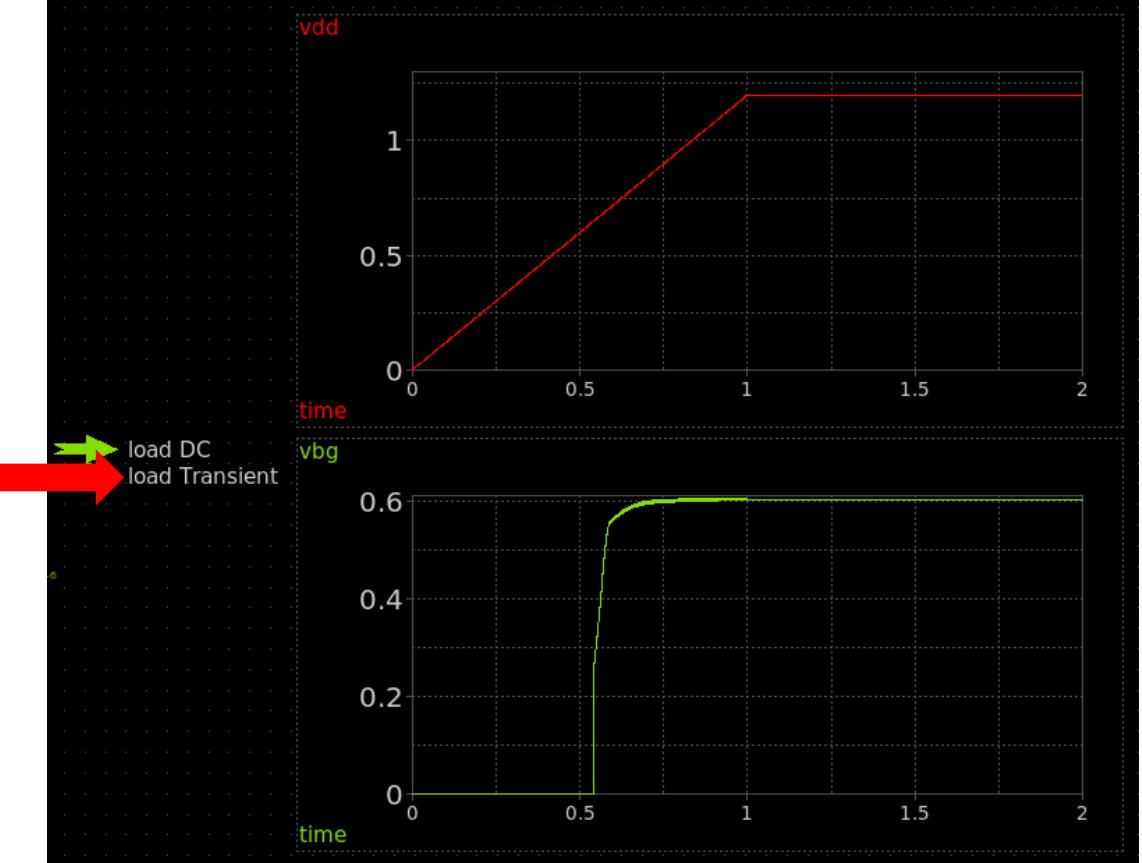
# plots



bgr\_temp.raw



bandgap\_transient.raw



## 🔍 Mismatch in IC Manufacturing

- Caused by **process variations**: geometry, doping, threshold voltage etc.
- Critical in **symmetry-based circuits** (e.g., differential pairs, bandgap references).
- Leads to **imbalanced currents**, voltage offsets, and **reduced accuracy**.

## Monte Carlo Simulations: Why They Matter

- Simulate **random device-level variations**.
- Evaluate **circuit robustness** under realistic conditions.
- **1σ (~68%), 2σ (~95%), 3σ (~99.7%)** coverage of variation.

# Mismatch Models and Code

```
name=NGSPICE only_toplevel=false
value="
.control
    let run = 1
    let mc_runs = 200
    set curplot = new
    set scratch = $curplot
    dowhile run <= mc_runs
        reset
        dc temp 100 -50 -5
        set run = $&run
        set dc = $curplot
        setplot $scratch
        let off{$run} = {$dc}.v(VBG)
        let mytemp{$run} = \"{$dc}.temp-sweep\"
        setplot $dc
        let run = run + 1
    end
    set nolegend
    plot {$scratch}.allv vs {$scratch}.mytemp1

    write bandgap_testbench_mc_mis.raw {$scratch}.allv {$scratch}.mytemp1
.endc
```

*Initializes the run counter*  
*Sets the number of Monte Carlo runs*  
*Creates a new plot and assigns to scratch*

*Starts a loop that runs until run > 200*  
*Resets the data to a clean state for each run*  
*Performs a **DC sweep** over temperature range*  
*Using the run value as a variable*  
*Saves the current plot name to variable **dc***  
*Switches to the scratch plot to store results*  
*Extracts the VBG voltage and stores it under the current run*  
*Saves the temp data for the current run*  
*Returns to the **dc** plot*  
*increments the run counter*

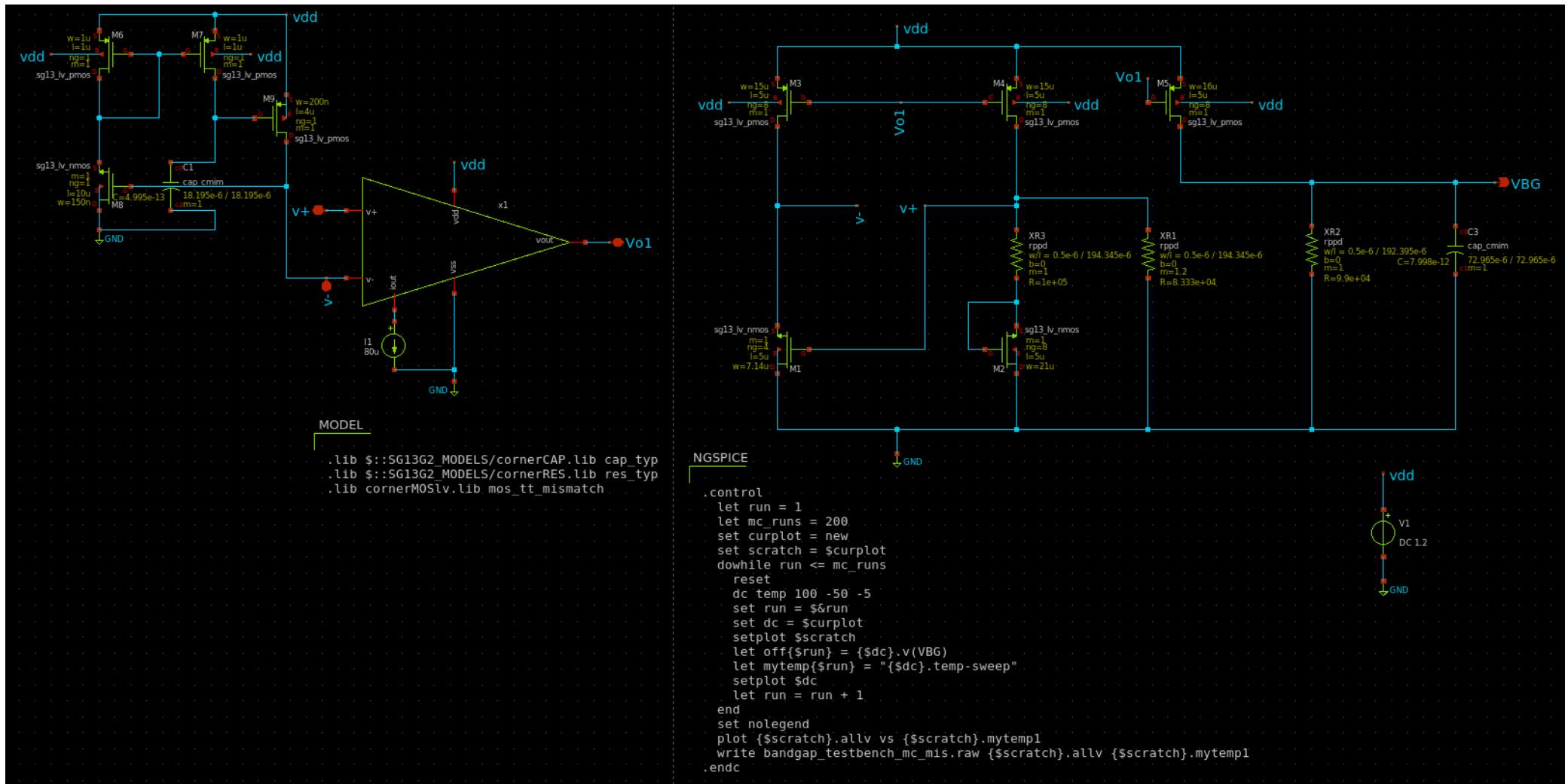
*Disables legend*  
*Plots all the waves against the temperature*



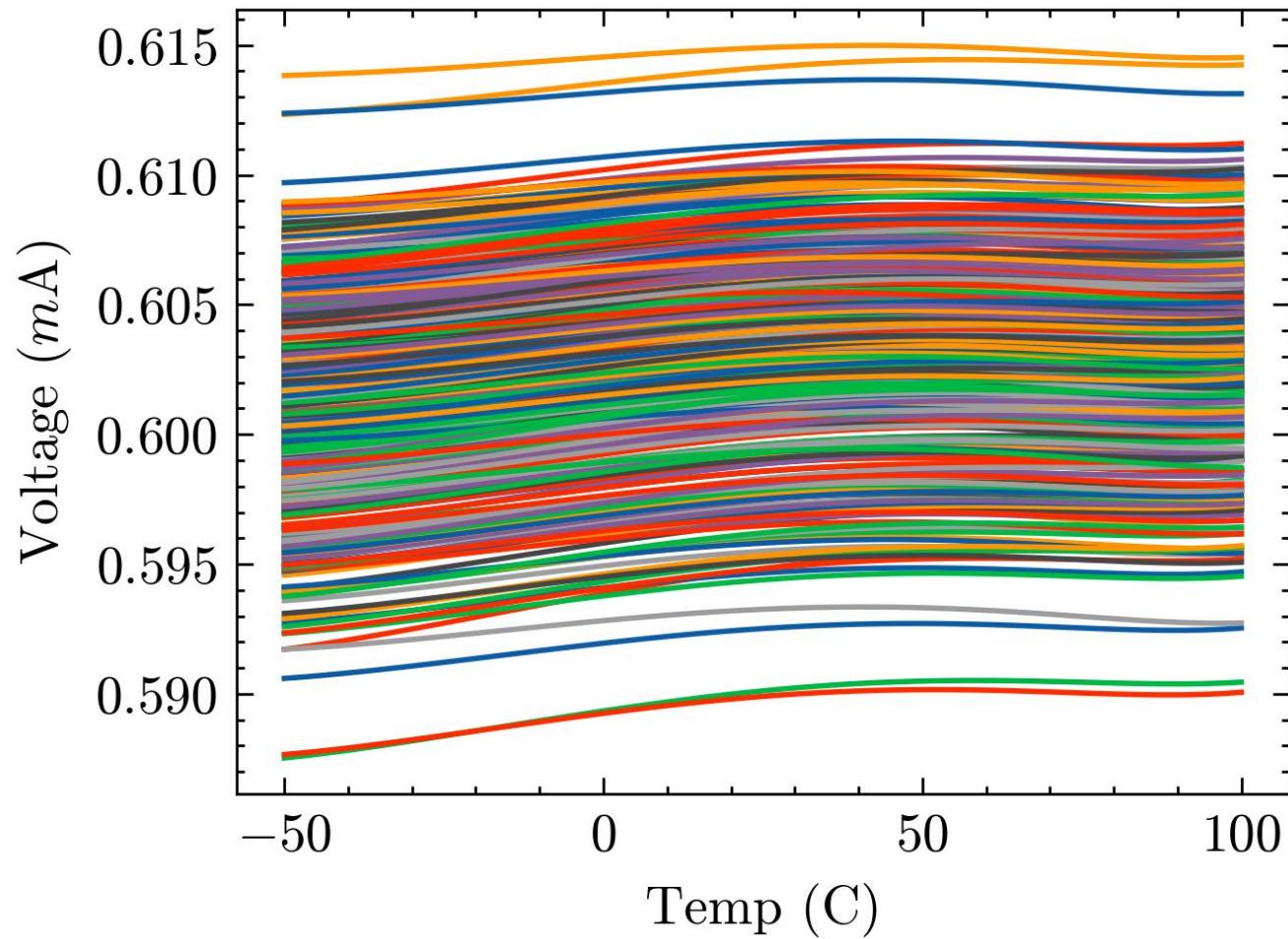
# Mismatch Models and Code

```
.lib $::SG13G2_MODELS/cornerCAP  
.lib cap_typ .lib $::SG13G2_MODELS/cornerRES  
.lib res_typ .lib cornerMOSlv  
.lib mos_tt_mismatch
```

# Testbench

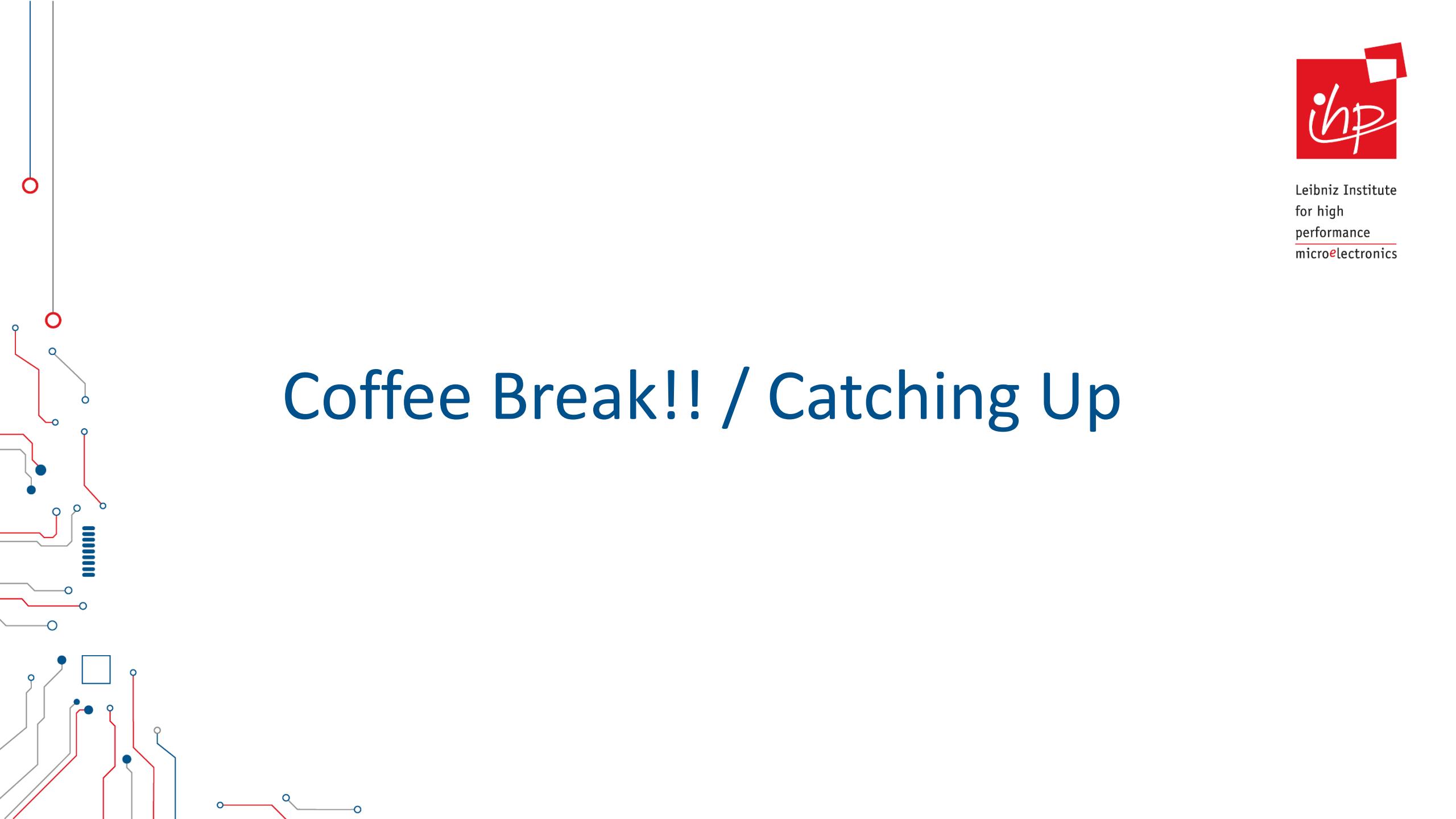


# Results





Leibniz Institute  
for high  
performance  
microelectronics

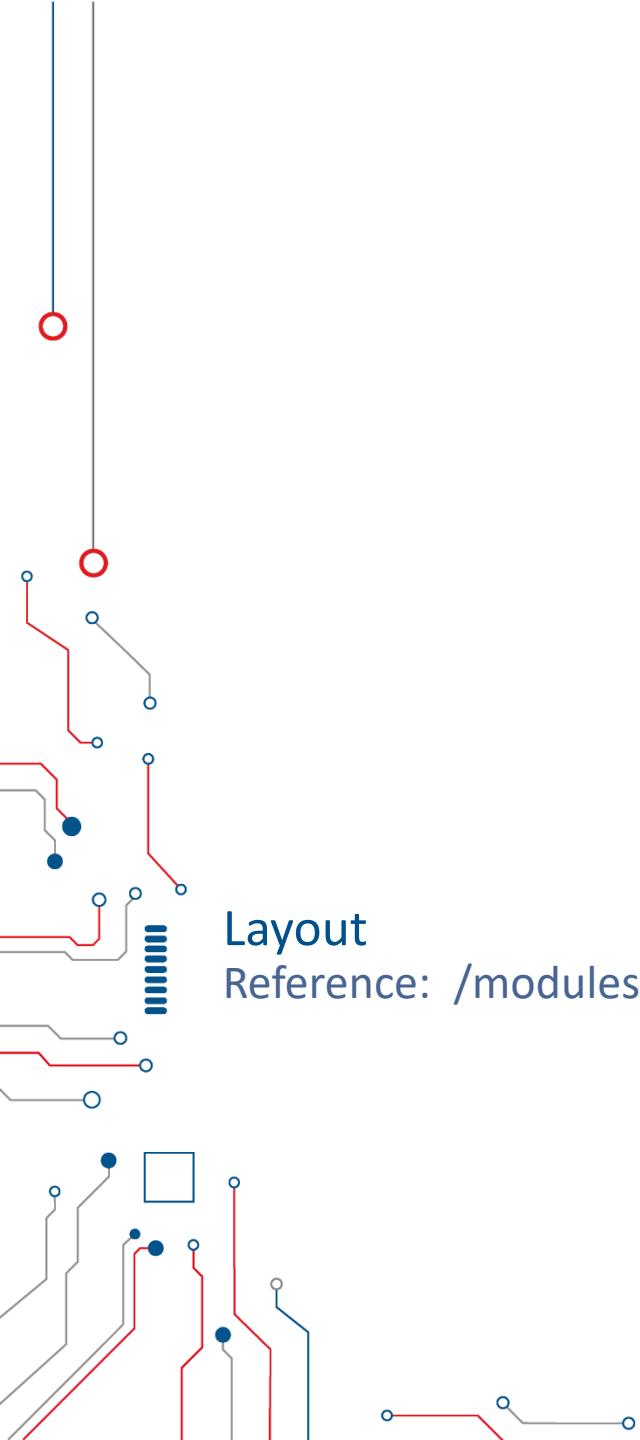
A faint background diagram of a microelectronic circuit is visible. It features a complex network of red, blue, and grey lines representing conductors and insulators. Various components are depicted, including resistors represented by blue dots, capacitors by small squares, and inductors by vertical blue bars. Some nodes are marked with open circles, while others are solid red or blue circles.

# Coffee Break!! / Catching Up

# Part 3

## Layout

Reference: [/modules/module\\_1\\_bandgap\\_reference/part\\_3\\_layout](#)



# Layout Competition!

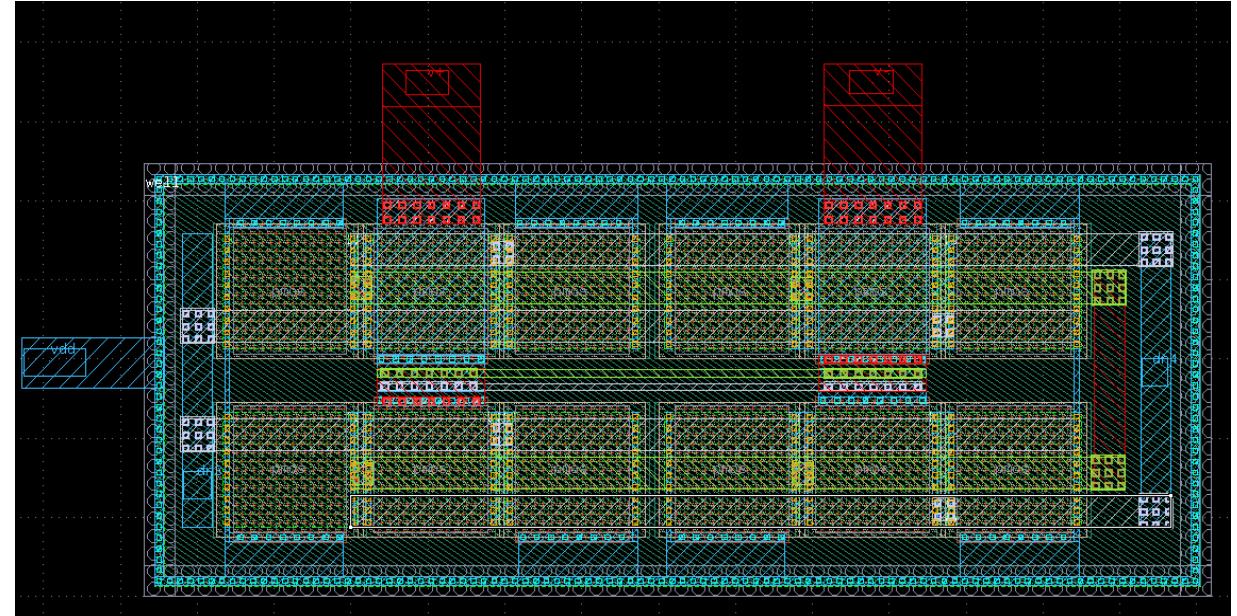
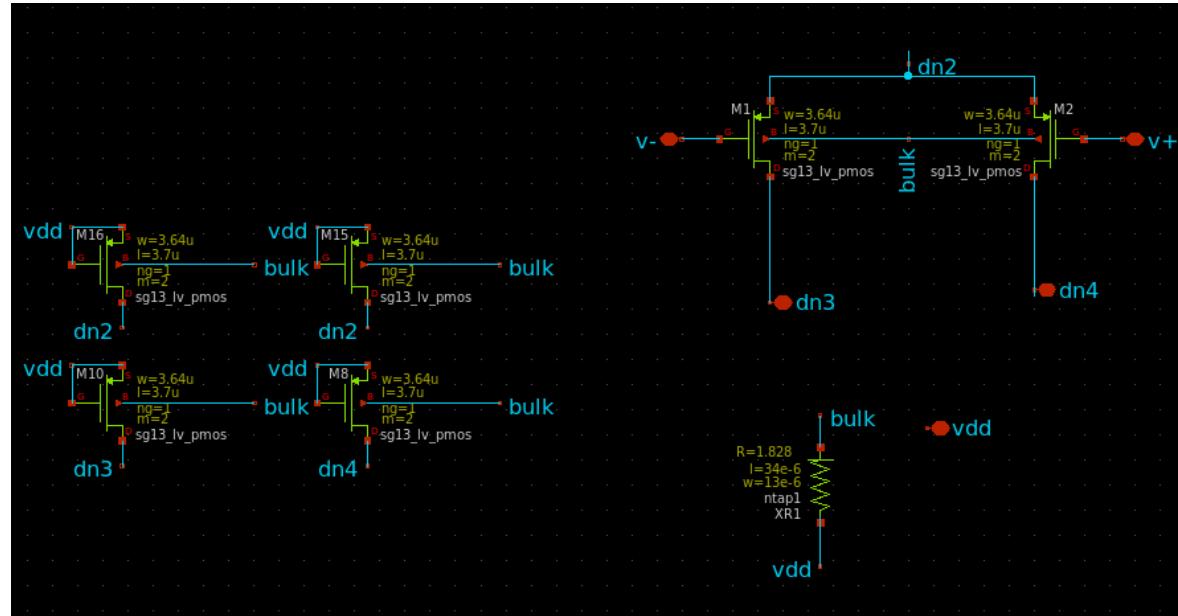


- 0 **Concept:** OTA layout, get as far as you can!
- 0 **Motivation:** Technical skills and awesome secret gift
- 0 **Evaluation:** On Friday we will look at each layout and a winner will be selected
- 0 **Metrics:** DRC/LVS errors, style and completeness
- 0 The remaining time over the next few days can be dedicated to the layout; it doesn't need to be completed today



I will be available to offer assistance!

# Back Annotation



| Objects   | Layout                               | Reference                    |
|---|--------------------------------------|------------------------------|
| ▼ <b>input_common_centroid</b> ↔ <b>INPUT_COI</b> | <b>input_common_centroid</b>         | <b>INPUT_COMMON_CENTROID</b> |
| ▶  Pins   |                                      |                              |
| ▶  Nets   |                                      |                              |
| ▼  Devices  |                                      |                              |
| ▶  ntap1 ↔ NTAP1                                  | \$13 / ntap1 [A=28.7556, P=0.18552k] | 1 / NTAP1 [A=0.442k, P=94]   |

# Back Annotation



```
3 *.PININFO v-:B v+:B vdd:B dn3:B dn4:B
4 M1 dn3 v- dn2 bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
5 M2 dn4 v+ dn2 bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
6 M8 dn4 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
7 M10 dn3 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
8 M15 dn2 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
9 M16 dn2 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
0 R1 vdd bulk ntap1 A=4.42e-10 P=9.4e-05
1 .ends
```

```
.subckt input_common_centroid v- v+ vdd dn3 dn4
*.PININFO v-:B v+:B vdd:B dn3:B dn4:B
M1 dn3 v- dn2 bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
M2 dn4 v+ dn2 bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
M8 dn4 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
M10 dn3 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
M15 dn2 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
M16 dn2 vdd vdd bulk sg13_lv_pmos w=3.64u l=3.7u ng=1 m=2
R1 vdd bulk ntap1 A=28.7556e-12 P=0.18552e-03
.ends
```

| Objects                           | Layout                | Reference             |
|-----------------------------------|-----------------------|-----------------------|
| input_common_centroid ⇄ INPUT_COM | input_common_centroid | INPUT_COMMON_CENTROID |

Detailed description of the table content:

- input\_common\_centroid ⇄ INPUT\_COM**:
  - Pins
  - Nets
  - Devices
    - ntap1 ⇄ NTAP1
    - sg13\_lv\_pmos ⇄ SG13\_LV\_PMOS
    - sg13\_lv\_pmos ⇄ SG13\_LV\_PMOS
    - sg13\_lv\_pmos ⇄ SG13\_LV\_PMOS
    - sg13\_lv\_pmos ⇄ SG13\_LV\_PMOS
    - sg13\_lv\_pmos ⇄ SG13\_LV\_PMOS
- Reference**:
  - \$13 / ntap1 [A=28.7556, P=0.18552k] 1 / NTAP1 [A=28.7556, P=0.18552k]
  - \$4 / sg13\_lv\_pmos [L=3.7, W=7.28, AS=2.1 / SG13\_LV\_PMOS [L=3.7, W=7.28]
  - \$3 / sg13\_lv\_pmos [L=3.7, W=7.28, AS=2.2 / SG13\_LV\_PMOS [L=3.7, W=7.28]
  - \$5 / sg13\_lv\_pmos [L=3.7, W=7.28, AS=2.8 / SG13\_LV\_PMOS [L=3.7, W=7.28]
  - \$6 / sg13\_lv\_pmos [L=3.7, W=7.28, AS=2.10 / SG13\_LV\_PMOS [L=3.7, W=7.28]
  - \$1 / sg13\_lv\_pmos [L=3.7, W=14.56, AS=15 / SG13\_LV\_PMOS [L=3.7, W=14.56]

# Alternatively

---



- 0 Complete unfinished tasks from today or yesterday
- 0 Explore other design metrices for the OTA or Bandgap
- 0 Ask questions/ get help if something wasn't clear today
- 0 Relax 😊

# What Will We Do Tomorrow?



- 0 **QUCS-S:** Getting familiar with QUCS-S and Xyce
- 0 **50GHz MPA:** Perform S-parameter analysis, along with DC and Harmonic Balancing
- 0 **Matching:** Performing simple matching using online tool and tuning in QUCS-S
- 0 **OpenEMS:** Expert talk on OpenEMS and how to interface with it using Python
- 0 **EM Simulation:** EM simulating components for the 50GHz MPA