Analog Circuit Design

Harald Pretl Michael Koefinger

2024-09-14

Table of contents

1 Introduction 4
1.1 IHP's SG13G2 130nm CMOS Technology 4
1.2 Schematic Entry Using Xschem 5
1.3 Circuit Simulation Using ngspice 5
1.4 Integrated IC Design Environment (IIC-OSIC-TOOLS) 5

2 First Steps 6
2.1 The Metal-Oxide-Semiconductor Field-Effect-Transistor (MOSFET) 6
2.1.1 Large-Signal MOSFET Model 9
2.1.2 Small-Signal MOSFET Model 10
2.2 Conclusion 12

3 Transistor Sizing Using gm/ID Methodology 12
3.1 MOSFET Characterization Testbench 13
3.2 NMOS Characterization 15
3.3 PMOS Characterization 22

4 First Circuit: MOSFET Diode 29
4.1 MOSFET Diode Sizing 30
4.2 MOSFET Diode Large-Signal Behaviour 31
4.3 MOSFET Diode Small-Signal Analysis 32
4.4 MOSFET Diode Stability Analysis 33
4.5 MOSFET Diode Noise Calculation 35
4.6 Conclusion 38

5 Current Mirror 38

6 Differential Pair 39
6.1 Differential Operation of the Diffpair 40
6.2 Common-Mode Operation of the Diffpair 41

7 A Basic 5-Transistor OTA **42**
7.1 Voltage Buffer with OTA 44
7.2 Large-Signal Analysis of the OTA 45
7.3 Small-Signal Analysis of the OTA 45
7.3.1 OTA Small-Signal Transfer Function 46
7.3.2 OTA Noise 49
7.4 5T-OTA Sizing 52

8 Sizing for Basic 5T-OTA **52**
8.1 5T-OTA Simulation 57
8.2 5T-OTA Simulation versus PVT 57

9 CACE Summary for ota-5t **58**
9.1 Plots 59
9.2 gain_vs_temp 59
9.3 gain_vs_vin 60
9.4 gain_vs_vdd 61
9.5 gain_vs_corner 62
9.6 bw_vs_temp 63
9.7 bw_vs_vin 64
9.8 bw_vs_vdd 65
9.9 bw_vs_corner 66
9.10 noise_vs_temp 67
9.11 noise_vs_vin 68
9.12 noise_vs_vdd 69
9.13 noise_vs_corner 70
9.14 settling_vs_temp 71
9.15 settling_vs_vin 72
9.16 settling_vs_vdd 73
9.17 settling_vs_corner 74
9.17.1 PVT Simulation Analysis 74

10 Cascode Stage **74**
10.1 Cascode Output Impedance 77
10.2 Cascode Input Impedance 79

11 Improved OTA **80**
11.1 Sizing the Improved OTA 82

12 Sizing for Basic (Improved) OTA **82**
12.1 Designing the Improved OTA 89
12.1.1 Discussion of the OTA Design 89
12.2 Simulation of Improved OTA 91
12.3 Corner Simulation of Improved OTA 93

13 CACE Summary for ota-improved **93**
13.1 Plots 95

13.2 gain_vs_temp 95
13.3 gain_vs_vin 97
13.4 gain_vs_vdd 99
13.5 gain_vs_corner 101
13.6 bw_vs_temp 103
13.7 bw_vs_vin 105
13.8 bw_vs_vdd 107
13.9 bw_vs_corner 109
13.10noise_vs_temp 111
13.11noise_vs_vin 113
13.12noise_vs_vdd 115
13.13noise_vs_corner 117
13.14settling_vs_temp 119
13.15settling_vs_vin 121
13.16settling_vs_vdd 123
13.17settling_vs_corner 125

14 A Fully-Differential OTA 126

15 Biasing the OTA 126

16 An RC-OPAMP Filter 126

17 Summary & Conclusion 126

18 Appendix: Middlebrook's Method 127

19 Appendix: Miller's Theorem 128

20 Appendix: 5T-OTA Small-Signal Output Impedance 129
20.1 Open-Loop Configuration 130
20.2 Closed-Loop Configuration 131

21 Appendix: ngspice Cheatsheet 132
21.0.1 Commands 133
21.0.2 Options 134
21.0.3 Convergence Helper 134

22 Appendix: Xschem Cheatsheet 134

23 Appendix: Circuit Designer's Etiquette 135

24 Circuit Designer's Etiquette 135
24.1 Prolog 135
24.2 Pins 136
24.3 Schematics 136
24.4 Symbols 138
24.5 Design Robustness 138

24.6 Rules for Good Mixed-Signal and RF Circuits 139
24.7 VHDL/Verilog Coding Guide 139
24.8 Further Reading 140

1 Introduction

This is the material for an intermediate-level MOSFET circuit design course, held at JKU under course number 336.009 ("KV Analoge Schaltungstechnik").

The course makes heavy use of circuit simulation, using **Xschem** for schematic entry and **ngspice** for simulation. The 130nm CMOS technology **SG13G2** from IHP Microelectronics is used.

Tools and PDK are integrated in the **IIC-OSIC-TOOLS** Docker image, which will be used during the coursework.

! Important

All course material is made publicly available on GitHub and shared under the Apache-2.0 license.

1.1 IHP's SG13G2 130nm CMOS Technology

SG13G2 is the name of a 130nm CMOS technology (strictly speaking BiCMOS) from IHP Microelectronics. It features low-voltage (thin-oxide) core MOSFET, high-voltage (thick-oxide) I/O MOSFET, various types of linear resistors, and 7 layers of Aluminium metallization (5 thin plus 2 thick metal layers). This PDK is open-source, and the complete process specification can be found at SG13G2 process specification. While we will not do layouts in this course, the layout rules can be found at SG13G2 layout rules.

For our circuit design, the most important parameters of the available devices are summarized in the following table:

Table 1: IHP SG13G2 devices

Device	Name	Specification
Low-voltage (LV) NMOS	`sg13_lv_nmos`	operating voltage (nom.) $V_{\mathrm{DD}} = 1.5\,\mathrm{V}$, $L_{\mathrm{min}} = 0.13\,\mu\mathrm{m}$, $V_{\mathrm{th}} \approx 0.5\,\mathrm{V}$; triple-well available
Low-voltage (LV) PMOS	`sg13_lv_pmos`	operating voltage (nom.) $V_{\mathrm{DD}} = 1.5\,\mathrm{V}$, $L_{\mathrm{min}} = 0.13\,\mu\mathrm{m}$, $V_{\mathrm{th}} \approx -0.47\,\mathrm{V}$
High-voltage (HV) NMOS	`sg13_hv_nmos`	operating voltage (nom.) $V_{\mathrm{DD}} = 3.3\,\mathrm{V}$, $L_{\mathrm{min}} = 0.45\,\mu\mathrm{m}$, $V_{\mathrm{th}} \approx 0.7\,\mathrm{V}$; triple-well available
High-voltage (HV) PMOS	`sg13_hv_pmos`	operating voltage (nom.) $V_{\mathrm{DD}} = 3.3\,\mathrm{V}$, $L_{\mathrm{min}} = 0.45\,\mu\mathrm{m}$, $V_{\mathrm{th}} \approx -0.65\,\mathrm{V}$
Silicided poly resistor	`rsil`	$R_{\square} = 7\,\Omega \pm 10\%$, $\mathrm{TC}_1 = 3100\,\mathrm{ppm/K}$

Device	Name	Specification
Poly resistor	`rppd`	$R_{\square} = 260\,\Omega \pm 10\%$, $\mathrm{TC}_1 = 170\,\mathrm{ppm/K}$
Poly resistor high	`rhigh`	$R_{\square} = 1360\,\Omega \pm 15\%$, $\mathrm{TC}_1 = -2300\,\mathrm{ppm/K}$
MIM capacitor	`cap_cmim`	$C' = 1.5\,\mathrm{fF}/\mu\mathrm{m}^2 \pm 10\%$, $\mathrm{VC}_1 = -26\mathrm{ppm/V}$, $\mathrm{TC}_1 = 3.6\mathrm{ppm/K}$, breakdown voltage $> 15\,\mathrm{V}$
MOM capacitor	n/a	The metal stack is well-suited for MOM capacitors due to 5 thin metal layers, but no primitive capacitor device is available at this point.

1.2 Schematic Entry Using Xschem

Xschem is an open-source schematic entry tool with emphasis on integrated circuits. For up-to-date information of the many features of Xschem and the basic operation of it please look at the available online documentation. Usage of Xschem will be learned with the first few basic examples, essentially using a single MOSFET. The usage model of Xschem is that the schematic is hierarchically drawn, and the simulation and evaluation statements are contained in the schematics. Further, Xschem offers embedded graphing, which we will mostly use.

1.3 Circuit Simulation Using ngspice

ngspice is an open-source circuit simulator with SPICE dependency (Nagel 1975). Besides the usual simulated types like `op` (operating point), `dc` (dc sweeps), `tran` (time-domain), or `ac` (small-signal frquency sweeps), ngspice offers a script-like control interface, where many different simulation controls and result evaluations can be done. For detailed information please refer to the latest online manual.

1.4 Integrated IC Design Environment (IIC-OSIC-TOOLS)

In order to make use of the various required components (tools like Xschem and ngspice, PDKs like SG13G2) easier, we will use the **IIC-OSIC-TOOLS**. 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 like installed tools, how to setup a VM, etc. please look at IIC-OSIC-TOOLS GitHub page.

⚠ Preparation

Please make sure to receive information about your personal VM access ahead of the course start.

Experienced users can install this image on their personal computer, for JKU students the IIC will host a VM on our compute cluster and provide personal login credentials.

⚠ Linux

In this course, we assume that students have a basic knowledge of Linux and how to operate it using the terminal. 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 ressources available.

2 First Steps

In this first chapter we will learn to use Xschem for schematic entry, and how to operate the ngspice SPICE simulator for circuit simulations. Further, we will make ourself familiar with the transistor and other passive components available in the IHP Microelectronics SG13G2 technology. While this is strictly speaking a BiCMOS technology offering MOSFETs as well as SiGe HBTs, we will use it as a pure CMOS technology.

2.1 The Metal-Oxide-Semiconductor Field-Effect-Transistor (MOSFET)

In this course, we will not dive into semiconductor physics and derive the device operation bottom-up starting from a fundamental level governed by quantum mechanics. Instead, we will treat the MOSFET as a macroscopic by assuming we have a 4-terminal device, and the performance of this device regarding its terminal voltages and currents we will largely derive from the simulation model.

The circuit symbol that we will use for the n-channel MOSFET is shown in Figure 1, and for the p-channel MOSFET it is shown in Figure 2. A control voltage between gate ("G") and source ("S") causes a current to flow between drain ("D") and source. The MOSFET is a 4-terminal device, so the bulk ("B") can also control the drain-source current flow. Often, the bulk is connected to source, and then the bulk terminal is not shown to declutter the schematics.

ℹ MOSFET Background

Strictly speaking is the drain-source current of a MOSFET controlled by the voltage between gate and bulk and the voltage between drain and source. Since bulk is often connected to source anyway, and many circuit designers historically were already familiar with the operation of the bipolar junction transistor, it is common to consider the gate-source voltage (besides the drain-source voltage) as the controlling voltage. This focus on gate-source implies that the source is special compared to the drain. In a typical physical MOSFET, however, the drain and source are constructed exactly the same, and which terminal is drain, and which terminal is source, is only determined by the applied voltage potentials, and can change dynamically during operation (think of a MOSFET operating as a switch… which side is the drain, which side is the source?).

Unfortunately, this focus on a "special" source has made its way into some MOSFET compact models. The model that is used in SG13G2 luckily uses the PSP model, which is formulated symmetrically with regards to drain and source, and is thus very well suited for analog and RF circuit design. For a detailed understanding of the PSP model please refer to the model documentation.

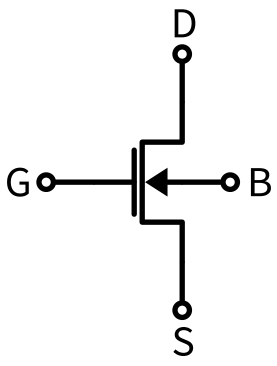


Figure 1: Circuit symbol of n-channel MOSFET.

Source: Article Notebook

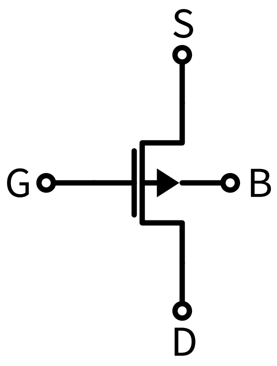


Figure 2: Circuit symbol of p-channel MOSFET.

Source: Article Notebook

For hand calculations and theoretical discussions we will use the following simplified large-signal model, shown in Figure 3. A current source I_{DS} models the current flow between drain and source, and it is controlled by the three control voltages V_{GS}, V_{DS}, and V_{SB}. Note that in this way (since $I_{\mathrm{DS}} = f(V_{\mathrm{DS}})$) also a resistive behavior between D and S can be modelled. In case that B and S are shorted then simply $V_{\mathrm{SB}} = 0$.

Source: Article Notebook

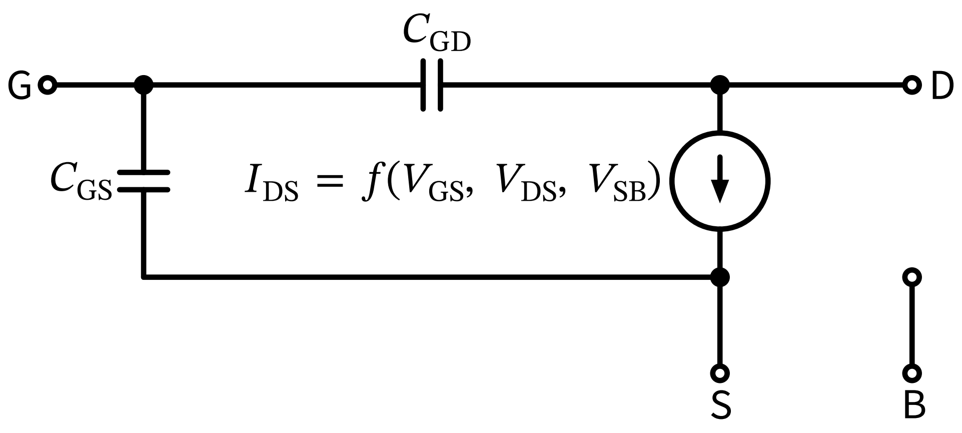


Figure 3: The MOSFET large-signal model.

Source: Article Notebook

In an ideal MOSFET no dc current is flowing into the gate, the behavior is purely capacitive. We model this by two capacitors: $C_{GG} = C_{GS} + C_{GD}$ is the total capacitance when looking into the gate of the MOSFET. C_{GS} is usually the dominant capacitance, and C_{GD} models the capacitive feedback between D and G, usually induced by a topological overlap capacitance in the physical construction of the MOSFET. This capacitance is often small compared to C_{GS}, but in situations where we have a large voltage swing at the drain this capacitance will be affected by the Miller effect (see Section 19). In hand calculations we will often set $C_{GD} = 0$.

MOSFET Bulk Terminal

The bulk connection in Figure 3 seems floating as we only consider it a control terminal, where the potential difference between source and bulk influences the behaviour of the MOSFET. However, we do not consider resistive or capacitive effects associated with this node, which is of course a gross simplification, but nevertheless one we will make in this course.

Now, as we are skipping the bottom-up approach of deriving the MOSFET large-signal behaviour from basic principles, we need to understand the behaviour of the elements of the large-signal model in Figure 3 by using a circuit simulator and observing what happens. And generally, a first step in any new IC technology should be to investigate basic MOSFET performance, by doing simple dc sweeps of V_{GS} and V_{DS} and looking at I_{DS} and other large- and small-signal parameters.

As a side note, the students who want to understand MOSFET behaviour from a physical angle should consult the MOSFET chapter from the JKU course "Design of Complex Integrated Circuits" (VL 336.048). A great introduction into MOSFET operation and fabrication is given in (Hu 2010), which is available freely online and is a recommended read. A very detailed description of the MOSFET (leaving usually no question unanswered) is provided in (Tsividis and McAndrew 2011).

Now, in order to get started, basic Xschem testbenches are prepared, and first simple dc sweeps of various voltages and currents will be done. But before that, please look at the import note below!

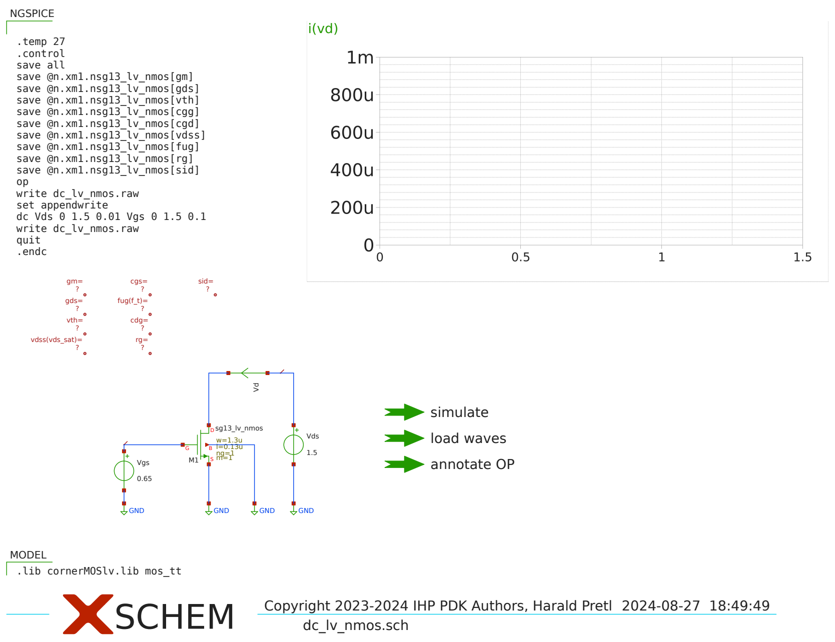
! Mathematical Notation

Throughout this material, we will largely stick to the following notation:

- A **dc quantity** is shown with an upper-case letter with upper-case subscripts, like V_{GS}.
- Double-subscripts denote **dc sources**, like V_{DD} and V_{SS}.
- An **ac (small-signal) quantity** is a lower-case letter with a lower-case subscript, like g_{m}.
- A **total quantity** (dc plus ac) is shown as a lowercase letter with upper-case subscript, like i_{DS}.
- A upper-case letter with a lower-case subscript is used to denote **RMS quantities**, like I_{ds}.

2.1.1 Large-Signal MOSFET Model

We start with an investigation into the large-signal MOSFET model shown in Figure 3 by using the simple testbench for the LV NMOS shown in Figure 4.



NGSPICE

```
.temp 27
.control
save all
save @n.xm1.nsg13_lv_nmos[gm]
save @n.xm1.nsg13_lv_nmos[gds]
save @n.xm1.nsg13_lv_nmos[vth]
save @n.xm1.nsg13_lv_nmos[cgg]
save @n.xm1.nsg13_lv_nmos[cgd]
save @n.xm1.nsg13_lv_nmos[vdss]
save @n.xm1.nsg13_lv_nmos[fug]
save @n.xm1.nsg13_lv_nmos[rg]
save @n.xm1.nsg13_lv_nmos[sid]
op
write dc_lv_nmos.raw
set appendwrite
dc Vds 0 1.5 0.01 Vgs 0 1.5 0.1
write dc_lv_nmos.raw
quit
.endc
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Copyright 2023-2024 IHP PDK Authors, Harald Pretl 2024-08-27 18:49:49
dc_lv_nmos.sch

Figure 4: Testbench for NMOS dc sweeps.

Exercise: MOSFET Investigation

Please try to execute the following steps and answer these questions:

1. Get the LV NMOS testbench (available at https://github.com/iic-jku/analog-circuit-design/blob/main/xschem/dc_lv_nmos.sch) working in your IIC-OSIC-TOOLS environment.
2. Make yourself familiar with Xschem (change the schematic in various ways, run a simulation, graph the result).
3. Make youself familiar with ngspice (run various simulations, save nets and parameters, use the embedded Xschem graphing, explore the interactive ngspice shell to look at MOSFET model parameters).
4. Explore the LV NMOS `sg13_lv_nmos`:
   1. How is I_{DS} affected by V_{GS} and V_{DS}?
   2. Change W and L of the MOSFET. What is the impact on the above parameters? Can you explain the variations?
   3. When looking at the model parameters in ngspice, you see that there is a C_{GD} and a C_{DG}. Why is this, what could be the difference? Sometimes these capacitors show a negative value, why?
5. Build testbenches in Xschem for the LV PMOS, the HV NMOS, and the HV PMOS. Explore the different results.
   1. For a given W and L, which device provides more drain current? How are the capacitances related?
   2. If you would have to size an inverter, what would be the ideal ratio of W_p/W_n? Will you exactly design this ratio, or are the reasons to deviate?
   3. There are LV and HV MOSFETs, and you investigated the difference in performance. What is the rationale when designing circuits for selection either an LV type, and when to choose an HV type?
6. Build a test bench to explore the body effect, start with LV NMOS.
   1. What happens when $V_{\mathrm{BS}} \neq 0$?

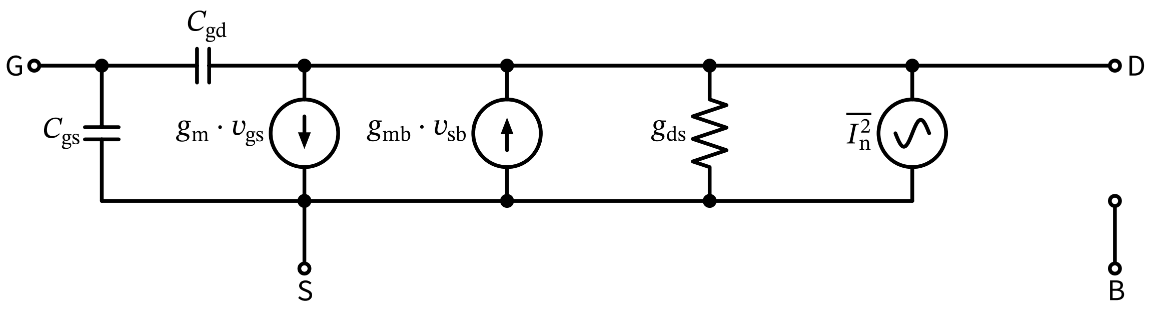
2.1.2 Small-Signal MOSFET Model

As you have seen in the previous investigations, the large-signal model of Figure 3 describes the behaviour of the MOSFET across a wide range of voltages applied at the MOSFET terminals. Unfortunately, for hand analysis dealing with a nonlinear model is close to impossible, at the very least it is quite tedious.

However, for many practical situations, we bias a MOSFET with a set of dc voltages applied to its terminal, and only apply small signal excursions during operation. If we do this, we can linearize the large-signal model in this dc operating point, and resort to a small-signal model which can be very useful for hand calculations. Many experienced designers analyze their circuits by doing these kind of hand calculations and describing the circuit analytically, which is a great way to understand fundamental performance limits and relationships between parameters.

We will use the small-signal MOSFET model shown in Figure 5 for this course. The current-source $i_{\mathrm{ds}} = g_{\mathrm{m}} v_{\mathrm{gs}}$ models the drain current as a function of v_{gs}, and the resistor g_{ds} models the dependency of the drain current by v_{ds}. The drain current dependency on the source-bulk voltage (the so-called "body effect") is introduced by the current source $i_{\mathrm{ds}} = g_{\mathrm{mb}} v_{\mathrm{sb}}$.

Source: Article Notebook



C_{gd}
G
D
C_{gs}
$g_{\mathrm{m}} \cdot v_{\mathrm{gs}}$
$g_{\mathrm{mb}} \cdot v_{\mathrm{sb}}$
g_{ds}
$\overline{I_{\mathrm{n}}^2}$
S
B

Figure 5: The MOSFET small-signal model.

Source: Article Notebook

As any electronic device the MOSFET introduces noise into the circuit. In this course we will only consider the drain-source current noise of the MOSFET, given by

$$\overline{I_{\mathrm{n}}^2} = 4kT\gamma g_{\mathrm{d0}}, \tag{1}$$

where $\overline{I_{\mathrm{n}}^2}$ is the power-spectral density of the noise in A^2/Hz; k is the Boltzmann constant; T is the absolute temperature; γ is a parameter in simplified theory changing between $\gamma = 2/3$ in saturation and $\gamma = 1$ for triode operation; g_{d0} is equal to g_{m} in saturation and g_{ds} in triode).

MOSFET Triode and Saturation Region

Sometimes we will refer to different operating modes of the MOSFET like "saturation" or "triode". Generally speaking, when the drain-source voltage is small, then the MOSFET acts as a resistor, and this mode of operation we call "triode" mode. When the drain-source voltage is increased, at some point the drain-source current saturates and is no longer a strong function of the drain-source voltage. This mode is called "saturation" mode. As you can see in the large-signal investigations, these transitions happen gradually, and it is difficult to define a precise point where one operating mode switches to the other one. In this sense we use terms like "triode" and "saturation" only in an approximative sense.

Now we need to see how the small-signal parameters seen in Figure 5 can be investigated and estimated using circuit simulation.

Exercise: MOSFET Small-Signal Parameters

Please try to execute the following steps and answer the following questions:

1. Reuse the LV NMOS testbench (available at https://github.com/iic-jku/analog-circuit-design/blob/main/xschem/dc_lv_nmos.sch).
2. Explore the LV NMOS `sg13_lv_nmos`:
    1. How are g_{m} and g_{ds} changing when you change the dc node voltages?
    2. What is the ratio of g_{m} to g_{mb}? What is the physical reason behind this ratio (you might want to revisit MOSFET device physics at this point)?
    3. Take a look at the device capacitances C_{gs} and C_{gd}. Why are they important? What is the relation to f_{T}? *Note: f_{T} is the transit frequency where the current gain of the MOSFET drops to 1, and can be approximated by* $2\pi f_{\mathrm{T}} = g_{\mathrm{m}}/C_{\mathrm{gg}}$.
    4. Look at the drain noise current according to the MOSFET model and compare with a hand calculation of the noise. In the noise equation there is the factor γ, which in triode is $\gamma = 1$ and in saturation is $\gamma = 2/3$ according to basic text books. Which value of γ are you calculating? Why might it be different?
3. Go back to your testbench for the LVS PMOS `sg13_lv_pmos`:
    1. What is the difference in g_{m}, g_{ds}, and other parameters between the NMOS and the PMOS? Why could they be different?

2.2 Conclusion

Congratulations for making it thus far! By now you should have a solid grasp of the tool handling of Xschem and ngspice, and you should be familiar with the large- and small-signal operation of both NMOS and PMOS, and the parameters describing these behaviours. If you feel you are not sufficiently fluent in these things, please go back to the beginning of Section 2.1 and revisit the relevant sections, or dive into further reading about the MOSFET operation, like in (Hu 2010).

3 Transistor Sizing Using gm/ID Methodology

When designing integrated circuits it is an important question how to select various parameters of a MOSFET, like W, L, or the bias current I_{D}. In comparison to using discrete components in PCB design, or also compared to a bipolar junction transistor (BJT), we have these degrees of freedom, which make integrated circuit design so interesting.

Often, transistor sizing in entry-level courses is based on the square-law model, where a simple analytical equation for the drain current can be derived. However, in nanometer CMOS, the MOSFET behaviour is much more complex than these simple models. Also, this highly simplified derivations introduce concepts like the threshold voltage or the overdrive voltage, which are interesting from a theoretical viewpoint, but bear little practical use.

MOSFET Square-Law Model

One of the many simplifications of the square-law model is that the mobility of the charge carriers is assumed constant (it is not). Further, the existance of a threshold voltage is assumed, but in fact this voltage is just existing given a certain definition, and depending on definition, its value changed. In addition, in nm CMOS, the threshold voltage is a function on many thing, like W and L.

An additional shortcoming of the square-law model is that it is only valid in strong inversion, i.e. for large V_{GS} where the drain current is dominated by the drift current. As soon as the gate-source voltage gets smaller, the square-law model breaks, as the drain current component based on diffusion currents gets dominant. Modern compact MOSFET models (like the PSP model used in SG13G2) use hundreds of parameters and fairly complex equations to somewhat properly describe MOSFET behaviour over a wide range of parameters like W, L, and temperature. A modern approach to MOSFET sizing is thus based on the thought to use exactly these MOSFET models, characterize them, put the resulting data into tables and charts, and thus learn about the complext MOSFET behaviour and use it for MOSFET sizing.

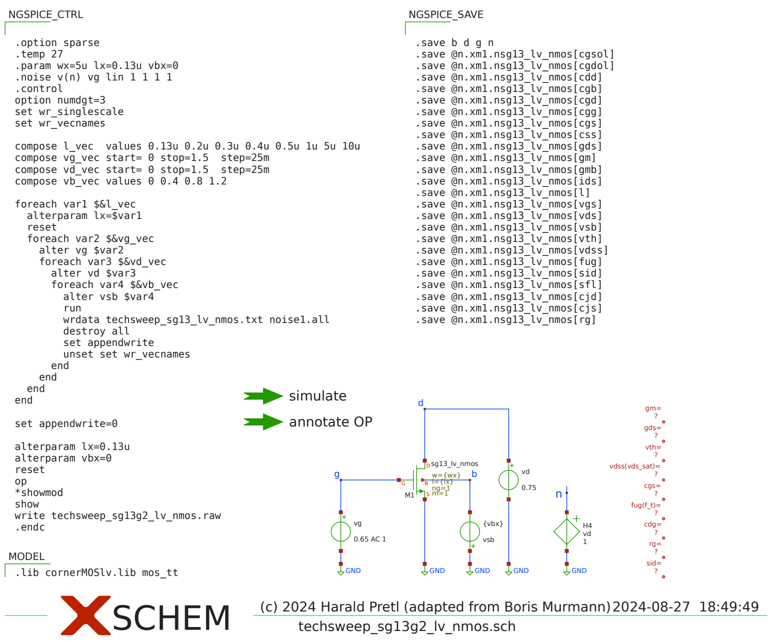
Being a well-established approach we select the $g_{\mathrm{m}}/I_{\mathrm{D}}$ methodology introduced by P. Jespers and B. Murmann in (Jespers and Murmann 2017). A brief introduction is available here as well.

The $g_{\mathrm{m}}/I_{\mathrm{D}}$ methodology has the huge advantage that is catches MOSFET behavior quite accurately over a wide range of operating conditions, and the curves look very similar for pretty much all CMOS technologies, form micrometer bulk CMOS down to nanometer FinFET devices. Of course the absolute values change, but the method applies universally.

3.1 MOSFET Characterization Testbench

In order to get the required tabulated data we use a testbench in Xschem which sweeps the terminal voltages, and records various large- and small-signal parameters, which are then stored in large tables. The testbench for the LV NMOS is shown in Figure 6, and the TB for the LV PMOS is shown in Figure 7.

We will use Jupyter notebooks to inspect the resulting data, and interpret some important graphs. This will greatly help to understand the MOSFET behaviour.



NGSPICE_CTRL

```
.option sparse
.temp 27
.param wx=5u lx=0.13u vbx=0
.noise v(n) vg lin 1 1 1 1
.control
option numdgt=3
set wr_singlescale
set wr_vecnames

compose l_vec  values 0.13u 0.2u 0.3u 0.4u 0.5u 1u 5u 10u
compose vg_vec start= 0 stop=1.5  step=25m
compose vd_vec start= 0 stop=1.5  step=25m
compose vb_vec values 0 0.4 0.8 1.2

foreach var1 $&l_vec
  alterparam lx=$var1
  reset
  foreach var2 $&vg_vec
    alter vg $var2
    foreach var3 $&vd_vec
      alter vd $var3
      foreach var4 $&vb_vec
        alter vsb $var4
        run
        wrdata techsweep_sg13_lv_nmos.txt noise1.all
        destroy all
        set appendwrite
        unset set wr_vecnames
      end
    end
  end
end

set appendwrite=0

alterparam lx=0.13u
alterparam vbx=0
reset
op
*showmod
show
write techsweep_sg13g2_lv_nmos.raw
.endc
```

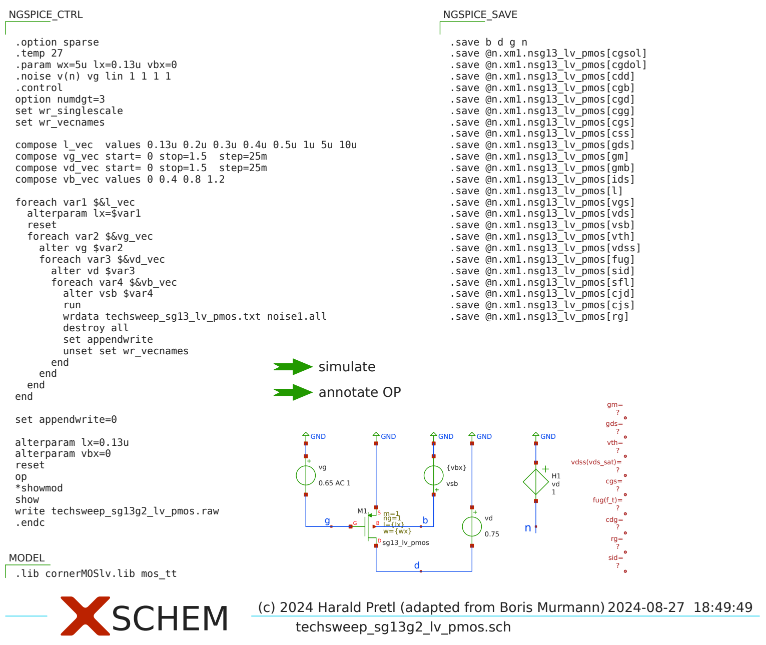
NGSPICE_SAVE

```
.save b d g n
.save @n.xm1.nsg13_lv_nmos[cgsol]
.save @n.xm1.nsg13_lv_nmos[cgdol]
.save @n.xm1.nsg13_lv_nmos[cdd]
.save @n.xm1.nsg13_lv_nmos[cgb]
.save @n.xm1.nsg13_lv_nmos[cgd]
.save @n.xm1.nsg13_lv_nmos[cgg]
.save @n.xm1.nsg13_lv_nmos[cgs]
.save @n.xm1.nsg13_lv_nmos[css]
.save @n.xm1.nsg13_lv_nmos[gds]
.save @n.xm1.nsg13_lv_nmos[gm]
.save @n.xm1.nsg13_lv_nmos[gmb]
.save @n.xm1.nsg13_lv_nmos[ids]
.save @n.xm1.nsg13_lv_nmos[l]
.save @n.xm1.nsg13_lv_nmos[vgs]
.save @n.xm1.nsg13_lv_nmos[vds]
.save @n.xm1.nsg13_lv_nmos[vsb]
.save @n.xm1.nsg13_lv_nmos[vth]
.save @n.xm1.nsg13_lv_nmos[vdss]
.save @n.xm1.nsg13_lv_nmos[fug]
.save @n.xm1.nsg13_lv_nmos[sid]
.save @n.xm1.nsg13_lv_nmos[sfl]
.save @n.xm1.nsg13_lv_nmos[cjd]
.save @n.xm1.nsg13_lv_nmos[cjs]
.save @n.xm1.nsg13_lv_nmos[rg]
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Figure 6: Testbench for LV NMOS $g_\mathrm{m}/I_\mathrm{D}$ characterization.



NGSPICE_CTRL

```
.option sparse
.temp 27
.param wx=5u lx=0.13u vbx=0
.noise v(n) vg lin 1 1 1 1
.control
option numdgt=3
set wr_singlescale
set wr_vecnames

compose l_vec  values 0.13u 0.2u 0.3u 0.4u 0.5u 1u 5u 10u
compose vg_vec start= 0 stop=1.5  step=25m
compose vd_vec start= 0 stop=1.5  step=25m
compose vb_vec values 0 0.4 0.8 1.2

foreach var1 $&l_vec
  alterparam lx=$var1
  reset
  foreach var2 $&vg_vec
    alter vg $var2
    foreach var3 $&vd_vec
      alter vd $var3
      foreach var4 $&vb_vec
        alter vsb $var4
        run
        wrdata techsweep_sg13_lv_pmos.txt noise1.all
        destroy all
        set appendwrite
        unset set wr_vecnames
      end
    end
  end
end

set appendwrite=0

alterparam lx=0.13u
alterparam vbx=0
reset
op
*showmod
show
write techsweep_sg13g2_lv_pmos.raw
.endc
```

NGSPICE_SAVE

```
.save b d g n
.save @n.xm1.nsg13_lv_pmos[cgsol]
.save @n.xm1.nsg13_lv_pmos[cgdol]
.save @n.xm1.nsg13_lv_pmos[cdd]
.save @n.xm1.nsg13_lv_pmos[cgb]
.save @n.xm1.nsg13_lv_pmos[cgd]
.save @n.xm1.nsg13_lv_pmos[cgg]
.save @n.xm1.nsg13_lv_pmos[cgs]
.save @n.xm1.nsg13_lv_pmos[css]
.save @n.xm1.nsg13_lv_pmos[gds]
.save @n.xm1.nsg13_lv_pmos[gm]
.save @n.xm1.nsg13_lv_pmos[gmb]
.save @n.xm1.nsg13_lv_pmos[ids]
.save @n.xm1.nsg13_lv_pmos[l]
.save @n.xm1.nsg13_lv_pmos[vgs]
.save @n.xm1.nsg13_lv_pmos[vds]
.save @n.xm1.nsg13_lv_pmos[vsb]
.save @n.xm1.nsg13_lv_pmos[vth]
.save @n.xm1.nsg13_lv_pmos[vdss]
.save @n.xm1.nsg13_lv_pmos[fug]
.save @n.xm1.nsg13_lv_pmos[sid]
.save @n.xm1.nsg13_lv_pmos[sfl]
.save @n.xm1.nsg13_lv_pmos[cjd]
.save @n.xm1.nsg13_lv_pmos[cjs]
.save @n.xm1.nsg13_lv_pmos[rg]
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Figure 7: Testbench for LV PMOS $g_\mathrm{m}/I_\mathrm{D}$ characterization.

3.2 NMOS Characterization

First, we will start looking at the LV NMOS. In Section 3.3 we have the corresponding graphs for the LV PMOS. In this lecture, we will only use the LV MOSFETs. While there are also the HV types available, they are mainly used for high-voltage circuits, like circuits connecting to the outside world. Here, we only will design low-voltage circuits running at a nominal supply voltage of 1.5 V, so only the LV types are of interest to us.

The first import graph is the plot of $g_\mathrm{m}/I_\mathrm{D}$ and f_T versus the gate-source voltage V_GS. First let us answer the question why $g_\mathrm{m}/I_\mathrm{D}$ is a good parameter to look at, and actually this is also the central parameter in the $g_\mathrm{m}/I_\mathrm{D}$ methodology. In many circuits that are biased in class-A (i.e., with a constant quiescent current that is larger than the largest signal excursion, see biasing) we want to get a large amplification from a MOSFET, which corresponds to a large g_m. We want this by spending the minimum biasing current possible (ideally zero), as we always design for minimum power consumption. Thus, a high $g_\mathrm{m}/I_\mathrm{D}$ ratio is good.

Power Consumption

Designing for minimum power consumption is pretty much always mandated. For battery-operated equipment it is a paramount requirement, but also in other equipment electrical energy consumption is a concern, and often severly limited by the cooling capabilities of the electrical system.

However, as can be seen in the below plot, there exists a strong and unfortunate trade-off with device speed, characterized here by the transit frequency f_T. It would be ideal if there exists a design point where we get high transconductance per bias current concurrently to having the fastest operation, but unfortunately, this is clearly not the case. The $g_\mathrm{m}/I_\mathrm{D}$ peaks for $V_\mathrm{GS} < 0.3\,\mathrm{V}$, and the highest speed we get at $V_\mathrm{GS} \approx 1.2\,\mathrm{V}$. The dashed vertical line plots the nominal threshold voltage, as you can see in this continuum of parameter space, it marks not a particularly special point.

Note that

$$\frac{g_\mathrm{m}}{I_\mathrm{D}} = \frac{1}{n V_\mathrm{T}} \tag{2}$$

for a MOSFET in weak inversion (i.e., small gate-source voltage). n is the subthreshold slope, and $V_\mathrm{T} = kT/q$ which is 25.8 mV at 300 K. We thus have $n \approx 1.38$ for this LV NMOS, which falls nicely into the usual range for n of 1.3 to 1.5 for bulk CMOS (FinFET have n very close to 1).

For the classical square-law model of the MOSFET in strong inversion, $g_\mathrm{m}/I_\mathrm{D}$ is given as

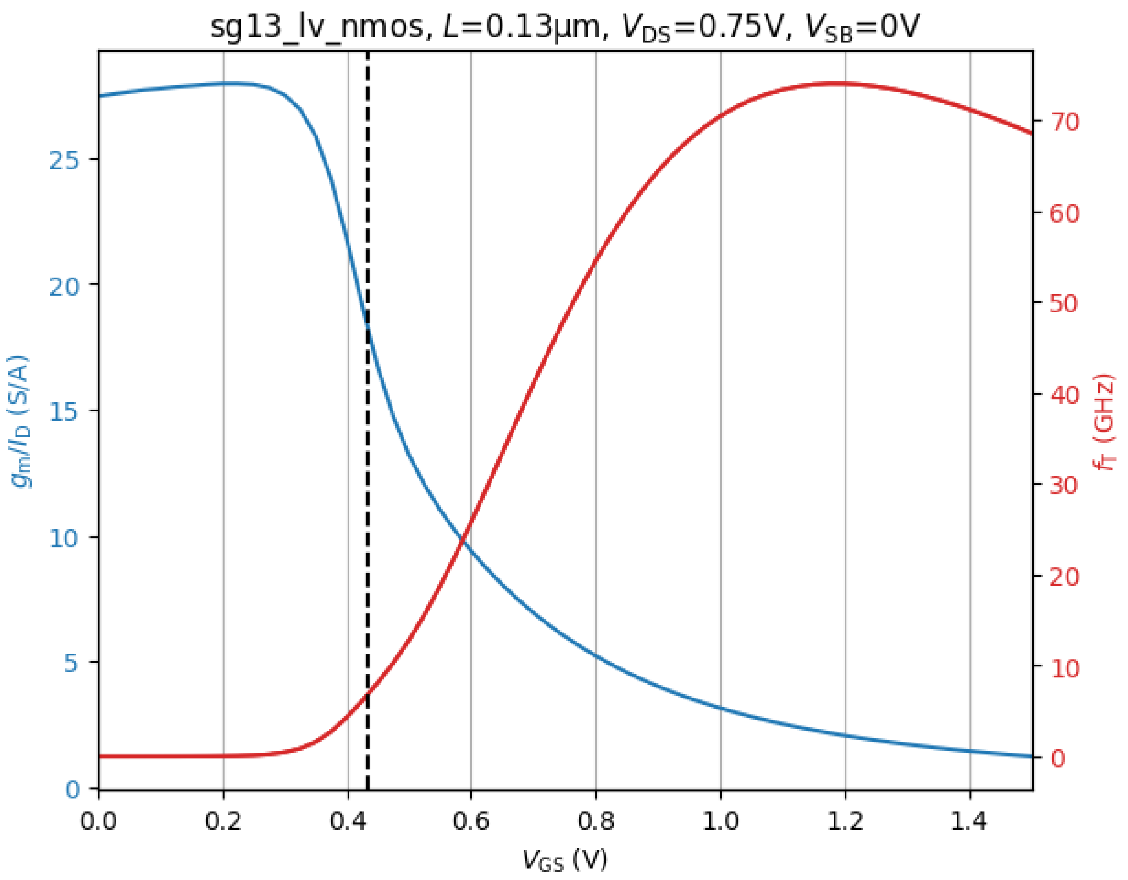
$$\frac{g_\mathrm{m}}{I_\mathrm{D}} = \frac{2}{V_\mathrm{GS} - V_\mathrm{th}} = \frac{2}{V_\mathrm{od}} \tag{3}$$

with V_th the threshold voltage and V_od the so-called “overdrive voltage.”

Why 300K?

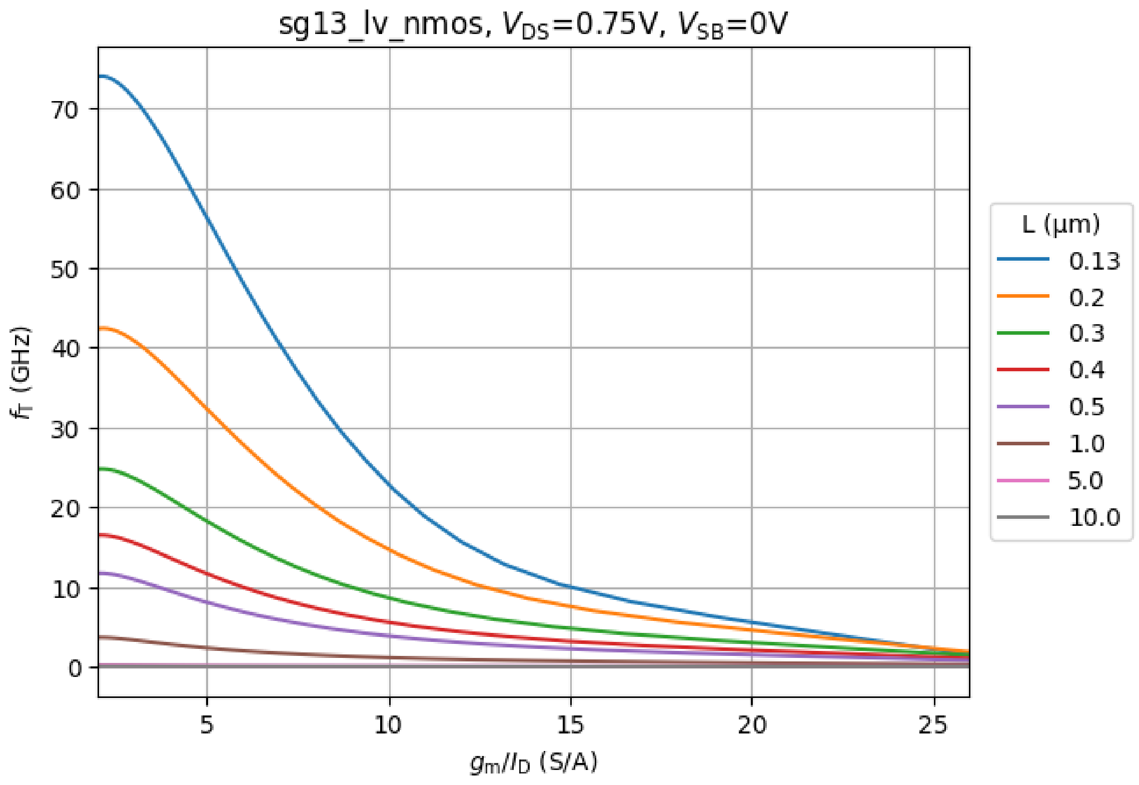
Why are we so often using a temperature of 300 K for a typical condition? As this corresponds to roughly 27°C, this accounts for some self heating compared to otherwise cooler usual room temperatures. Further, engineers like round numbers which are easy to remember, so 300 K is used as a proxy for room temperature.

As we can also see from belows plot, the peak transit frequency of the LV NMOS is about 75 GHz, which allows building radio-frequency circuits up to ca. $f_T/10 = 7.5\,\text{GHz}$, which is a respectible number. It is no coincidence, that the transition for RF design in the GHz-range switched from BJT-based technologies to CMOS roughly in the timeframe when 130nm CMOS became available (ca. 2000).



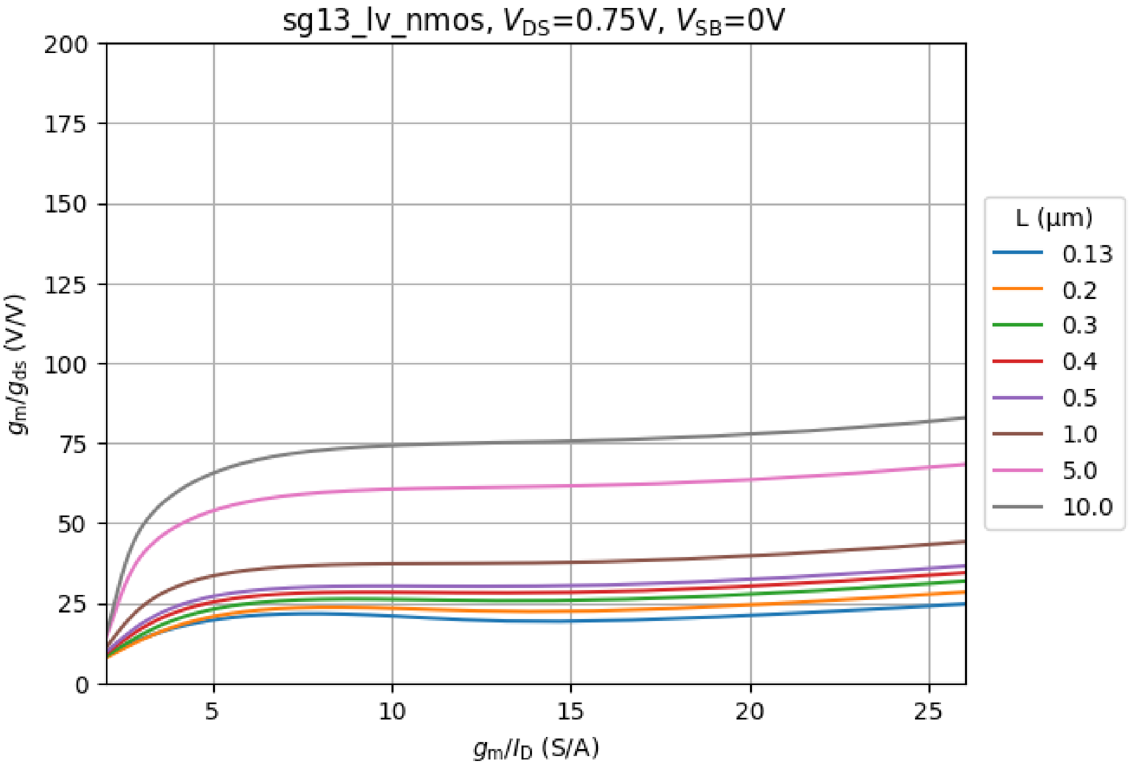
Source: Article Notebook

The following figure plots f_T against g_m/I_D for several different L. As you can see, device speeds maximizes for a low g_m/I_D and a short L. As you can see the drain-source voltage is kept at $V_{DS} = 0.75\,\text{V} = V_{DD}/2$, which is a typical value keeping the MOSFET in saturation across the characterization sweeps. Further, the source-bulk voltage is kept at $V_{SB} = 0\,\text{V}$, which means bulk and source terminals are connected.



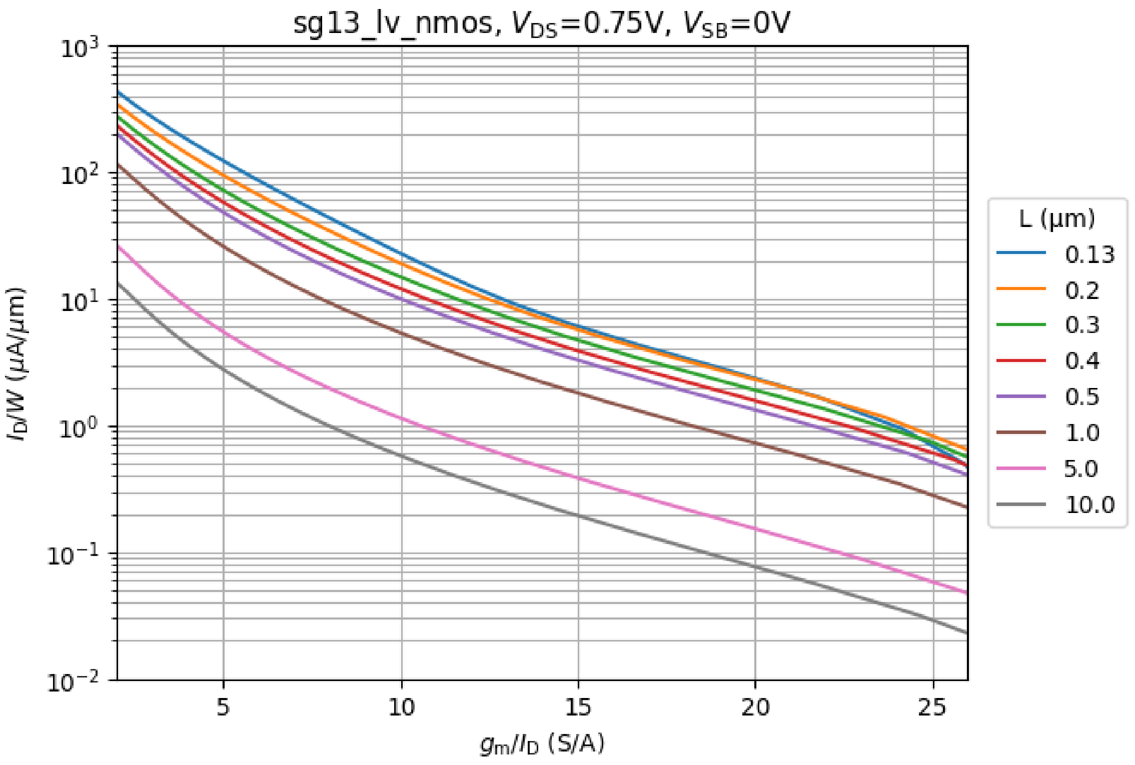
Source: Article Notebook

The next plot shows the ratio of $g_\mathrm{m}/g_\mathrm{ds}$ versus $g_\mathrm{m}/I_\mathrm{D}$. The ratio $g_\mathrm{m}/g_\mathrm{ds}$ is the so-called **“self-gain”** of the MOSFET, and shows the maximum voltage gain we can achieve in a single transistor configuration. As one can see the self gain increases for increasing L, but this also gives a slower transistor, so again there is a trade-off. This plot allows us to select the proper L of a MOSFET if we know which amount of self gain we need.



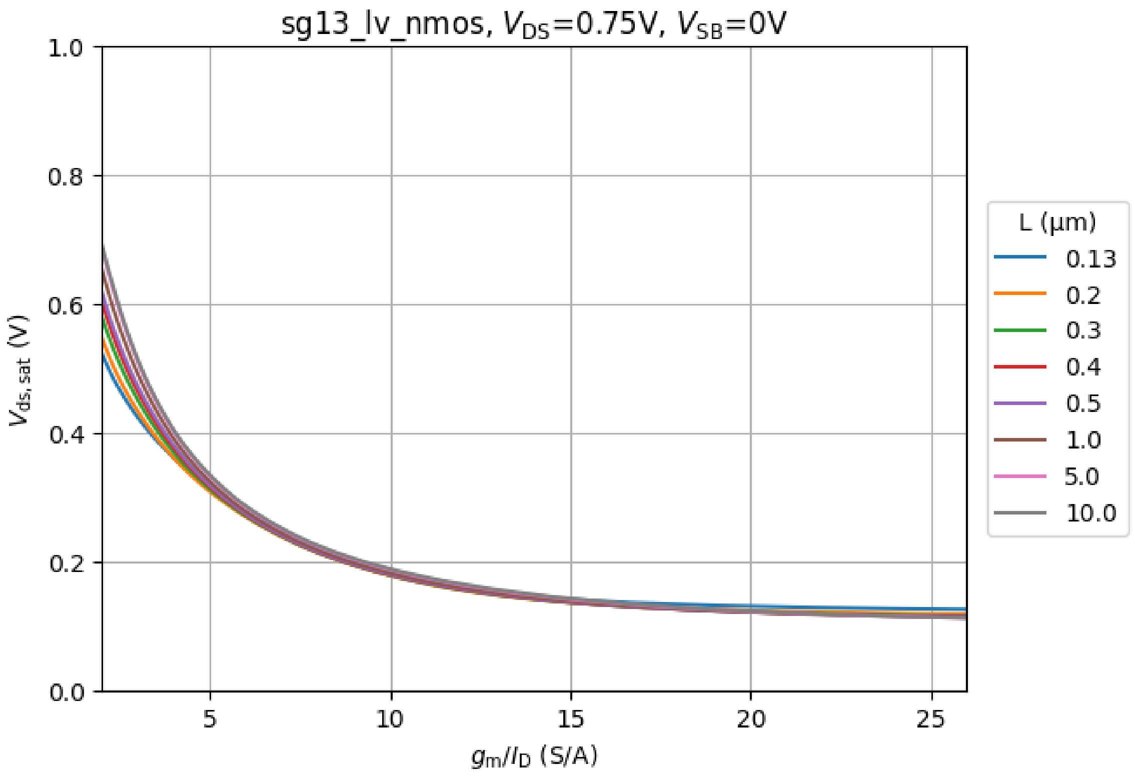
Source: Article Notebook

The following figure plots the drain current density I_D/W as a function of g_m/I_D and L. With this plot we can find out how to set the W of a MOSFET once we know the biasing current I_D, the L (selected according to self gain, f_T, and other considerations) and the g_m/I_D design point we selected. The drain current density I_D/W is a very useful nomalized metric to use, because the physical action in the MOSFET establishes a charge density in the channel below the gate, and the changing of the W of the device merely transforms this charge density into an absolute parameter (together with L).



Source: Article Notebook

The following plot shows the minimum drain-source voltage $V_{\mathrm{ds,sat}}$ that we need to establish in order to keep the MOSFET in saturation. As you can see, this value is almost independent of L, and increases for small $g_\mathrm{m}/I_\mathrm{D}$. So for low-voltage circuits, where headroom is precious, we tend to bias at $g_\mathrm{m}/I_\mathrm{D} \geq 10$, wheres for fast circuits we need to go to small $g_\mathrm{m}/I_\mathrm{D} \leq 5$ requiring substantial voltage headroom per MOSFET stage that we stack on top of each other.



Source: Article Notebook

For analog circuits the noise performance is usually quite important. Thermal noise of a resistor (the Johnson-Nyquist noise) has a flat power-spectral density (PSD) given by $\overline{V_n^2}/\Delta f = 4kTR$, where k is Boltzmann's constant, T absolute temperature, and R the value of the resistor (the unit of $\overline{V_n^2}/\Delta f$ is V^2/Hz). This PSD is essentially flat until very high frequencies where quantum effects start to kick in.

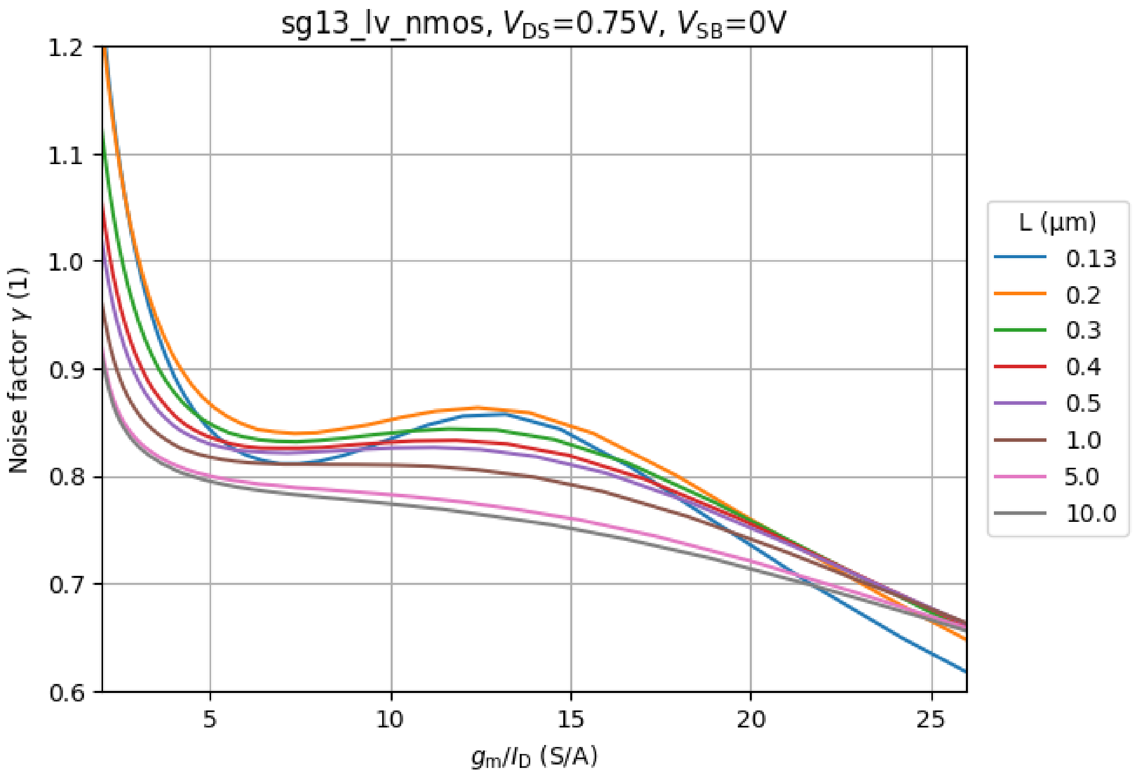
Noise Notation

We usually leave the Δf away for a shorter notation, so we write $\overline{V_n^2}$ when we actually mean $\overline{V_n^2}/\Delta f$. In case of doubt look at the unit of a quantity, whether is shows V^2 or V^2/Hz or $\mathrm{V}/\sqrt{\mathrm{Hz}}$ (or I^2 or I^2/Hz or $\mathrm{I}/\sqrt{\mathrm{Hz}}$).
Please also note that the pair of kT pretty much always shows up together, so when you do a calculation and you miss the one or the other, that is often a sign for miscalculation.
Boltzmann's constant $k = 1.38 \cdot 10^{-23}$ J/K is just a scaling factor from thermal energy expressed as a temperature T to energy $E = kT$ expressed in Joule.
Further, when working with PSD there is the usage of a one-sided ($0 \geq f < \infty$) or two-sided power spectral density (PSD) ($-\infty < f < \infty$). The default in this lecture is the usage of the **one-sided PSD**.

In this lecture the only MOSFET noise we consider is the drain noise (as discussed in Section 2.1.2), showing up as a current noise between drain and source. For a for realistic

MOSFET noise model, also a (correlated) gate noise component and the thermal noise of the gate resistance needs to be considered.

The factor γ (Equation 1) is a function of many things (in classical theory, $\gamma = 2/3$ in saturation and $\gamma = 1$ in triode), and it is characterized in the following plot as a function of $g_\mathrm{m}/I_\mathrm{D}$ and L. So when calculating MOSFET noise we can lookup γ in the below plot, and use Equation 1 to calculate the effective drain current noise.



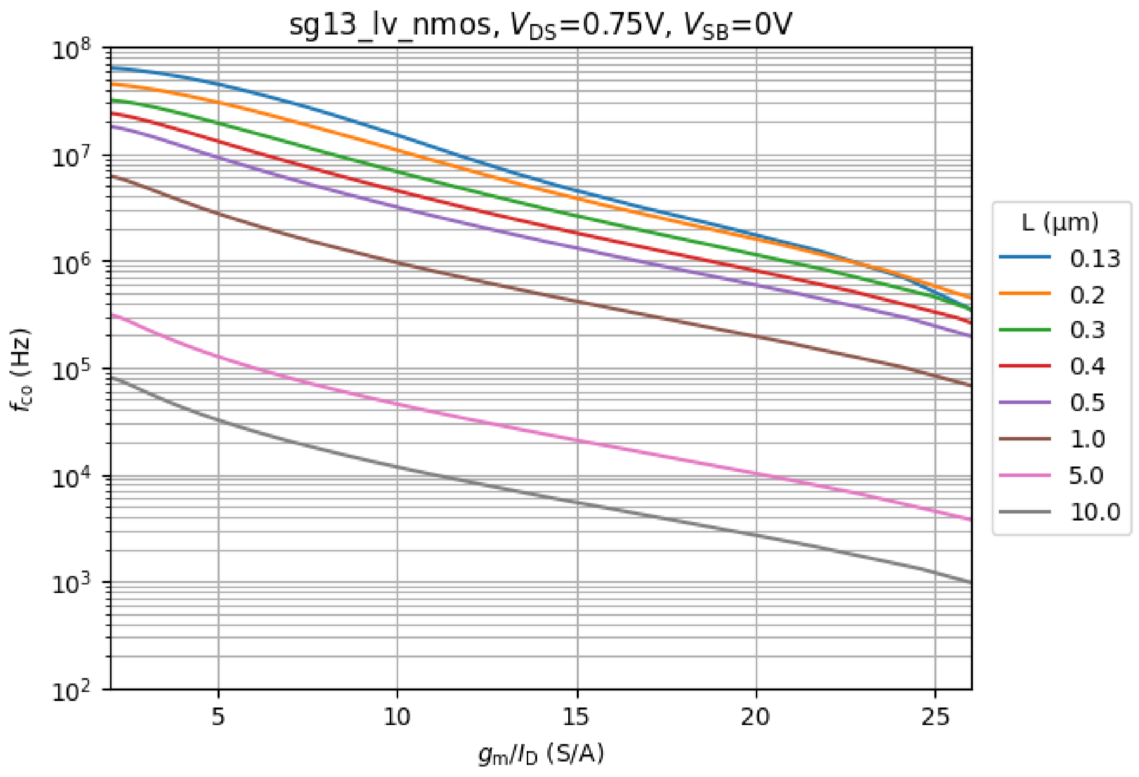
Source: Article Notebook

In a MOSFET, unfortunately, besides the thermal noise according to Equation 1, there is also a substantial low-frequency excess noise, called "flicker noise" due to its characteristic $\overline{I^2_\mathrm{d,nf}} = K_\mathrm{f}/f$ behaviour (this means that this noise PSD decreases versus frequency). In order to characterize this flicker noise the following plot shows the cross-over frequency f_co, where the flicker noise is as large as the thermal noise. As can be seen in the below plot, this frequency is a strong function of L and $g_\mathrm{m}/I_\mathrm{D}$. Generally, the flicker noise is proportional to $(WL)^{-1}$, so the larger the device is, the lower the flicker noise. The parameter $g_\mathrm{m}/I_\mathrm{D}$ largely stays constant when we keep W/L constant, so for a given $g_\mathrm{m}/I_\mathrm{D}$ flicker noise is proportinal to $1/L^2$. However, increasing L lowers device speed dramatically, so here we have a trade-off between flicker-noise performance and MOSFET speed, and this can have dramatic consequences for high-speed circuits.

MOSFET Flicker Noise

The physical origin of flicker noise is the crystal interface between silicon (Si) and the silicondioxide (SiO_2). Since these are different materials, there are dangling bonds, which can capture charge charriers travelling in the channel. After a random time, these carriers are released, and flicker noise is the result. The amount of flicker noise is a function of the manufacturing process, and will generally be different between device types and wafer foundries.

As you can see in the following plot, f_{co} can reach well into the 10's of MHz for short MOSFETs, significantly degrading the noise performance of a circuit.



Source: Article Notebook

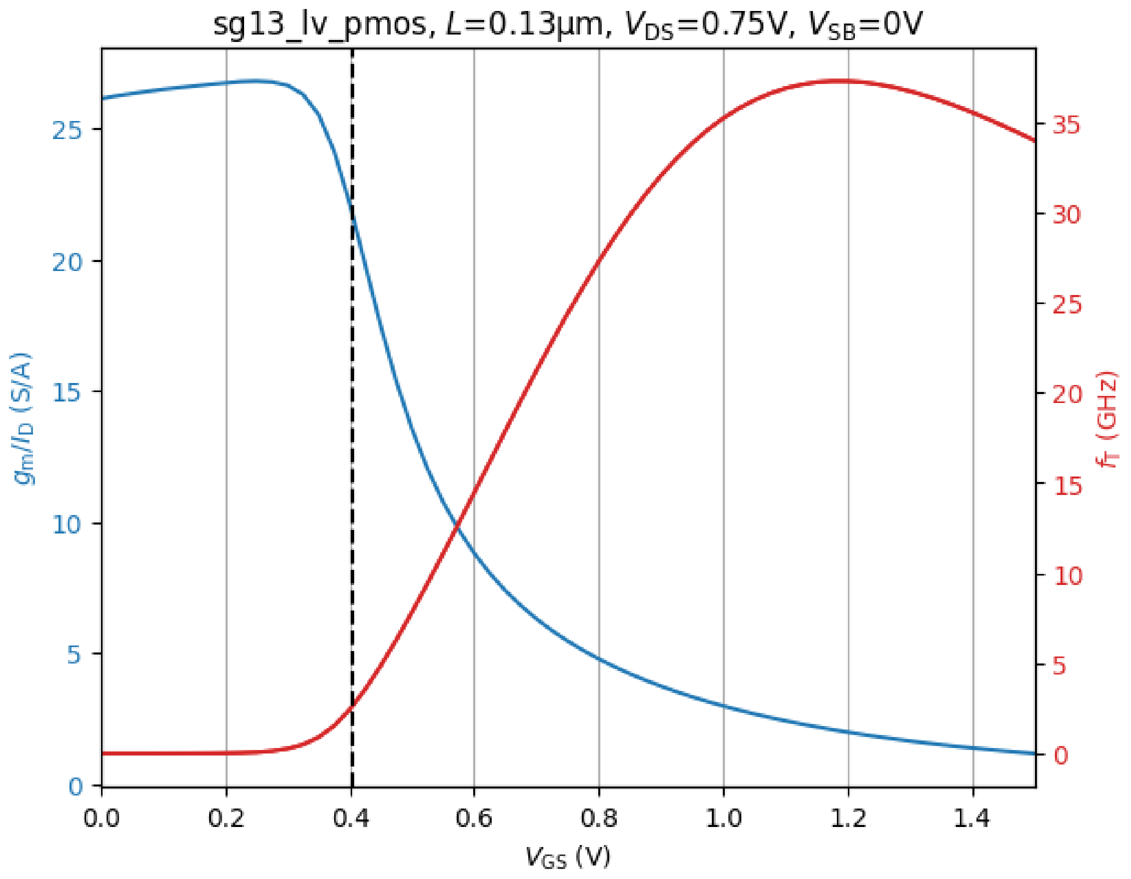
3.3 PMOS Characterization

In the following, we have the same plots as discussed in Section 3.2, but now for the PMOS.

PMOS Sign Convention

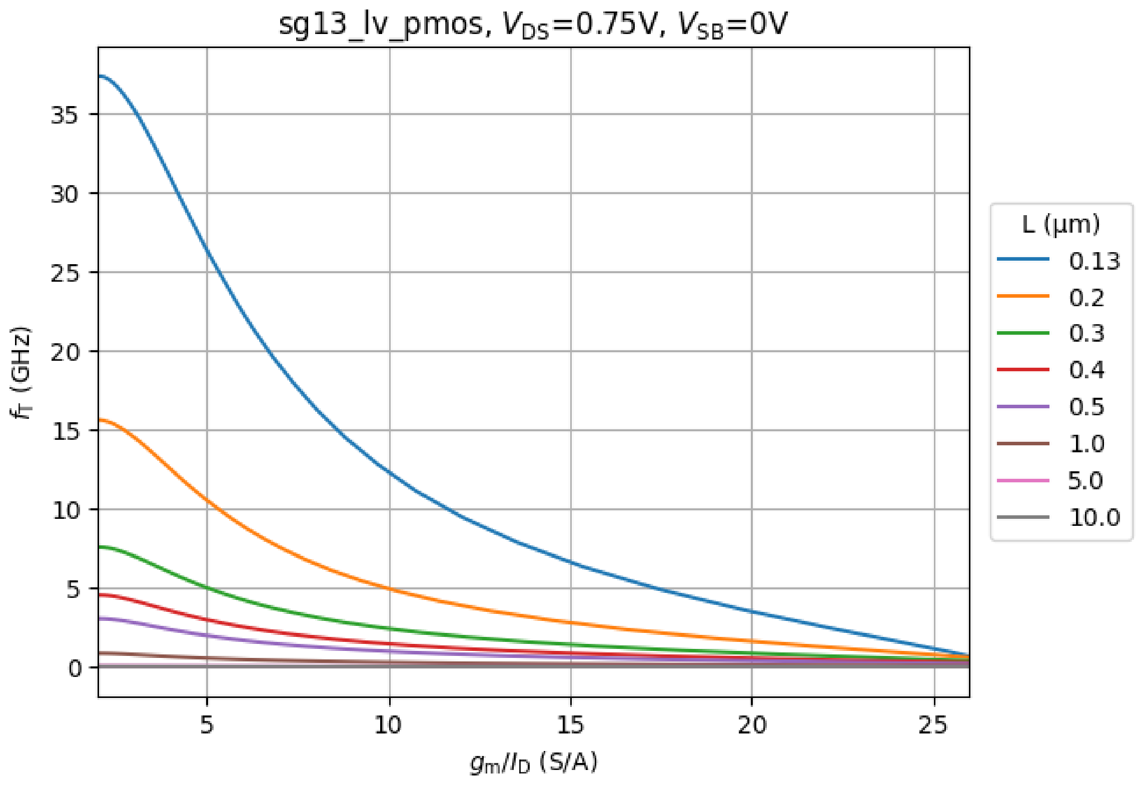
In all PMOS plots we plot positive values for voltages and currents, to have compatible plots to the NMOS. Of course, in a PMOS, voltages and currents have different polarity compared to the NMOS.

g_m/I_D and f_T versus the gate-source voltage V_{GS}:



Source: Article Notebook

f_T against g_m/I_D for several different L. One can see significantly lower top speed for the PMOS compared to the NMOS, which means for high-speed circuits the NMOS should be used.

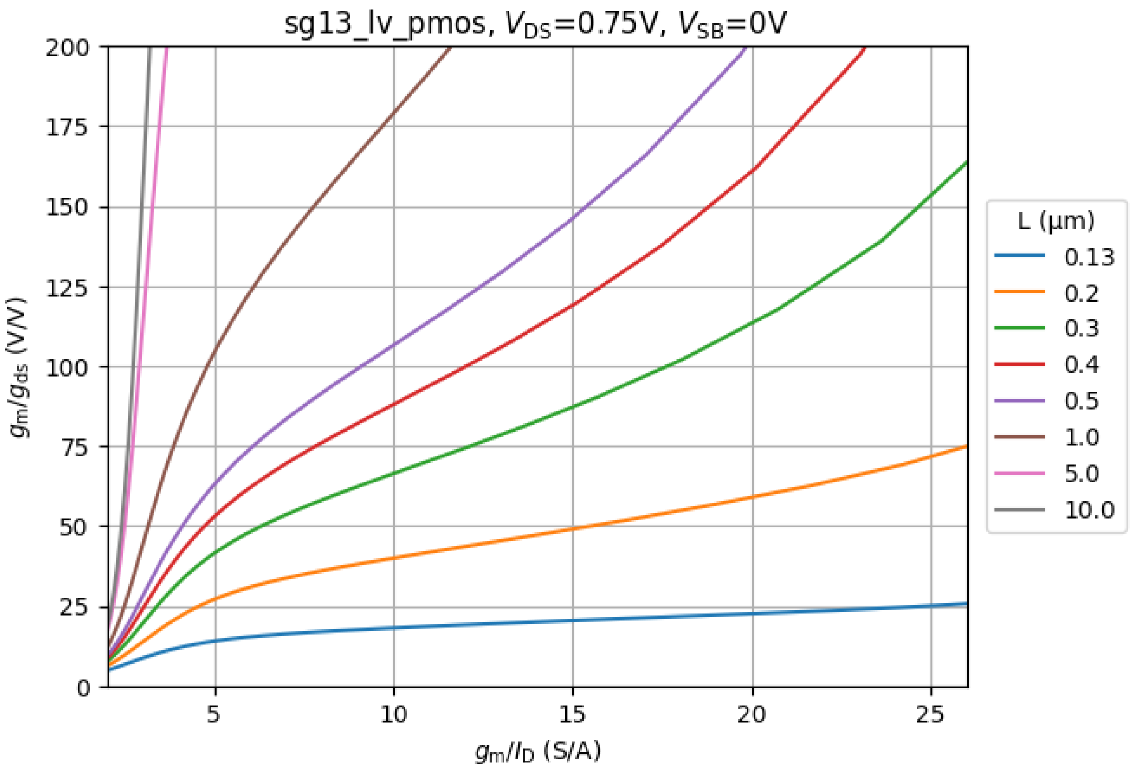


Source: Article Notebook

g_m/g_{ds} versus g_m/I_D. Unfortunately, one can see a modelling error for the PMOS in this plot. The self gain g_m/g_{ds} reaches non-physical values, which indicates an issue with the g_{ds} modelling for the PMOS. We can not use these values for our circuit sizing, so we will use the respective NMOS plots also for the PMOS.

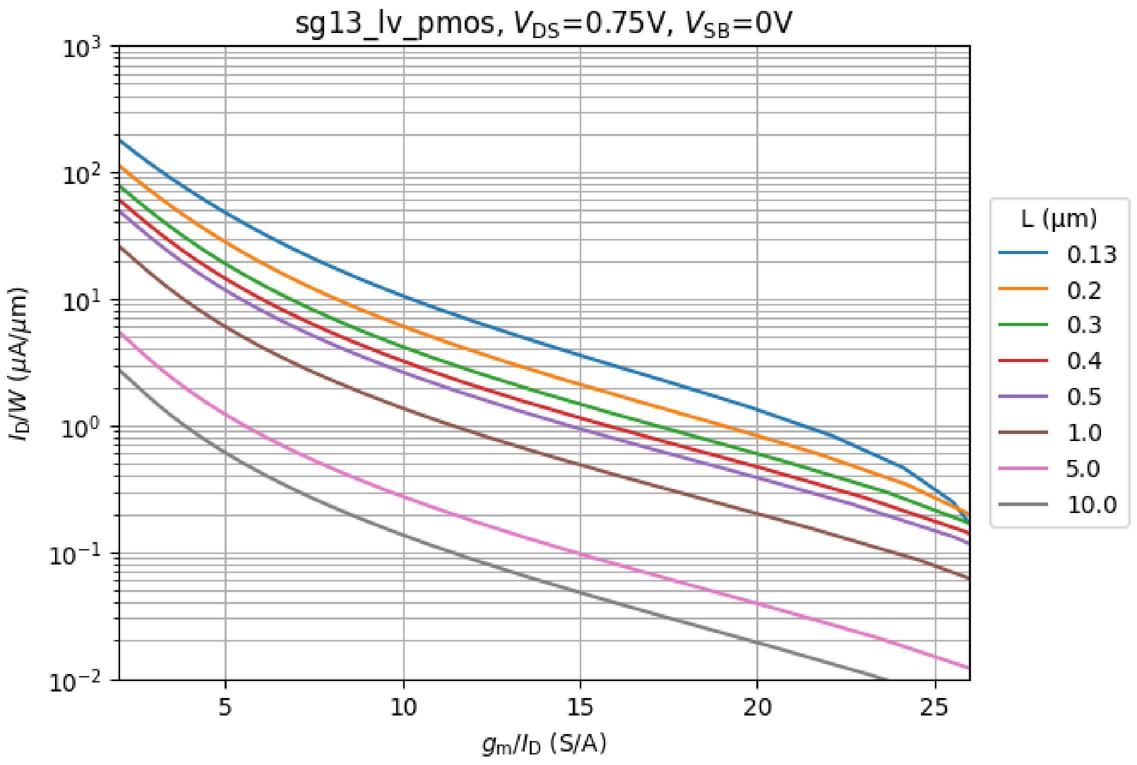
! Beware of Modelling Issues

This example shows how important it is to benchmark the device models when starting to use a new technology. Modelling artifacts like the one shown are quite often happening, as setting up the device compact models and parametrizing them according to measurement data is a very complex task. In any case, just be aware that modelling issues could exist in whatever PDK you are going to use!



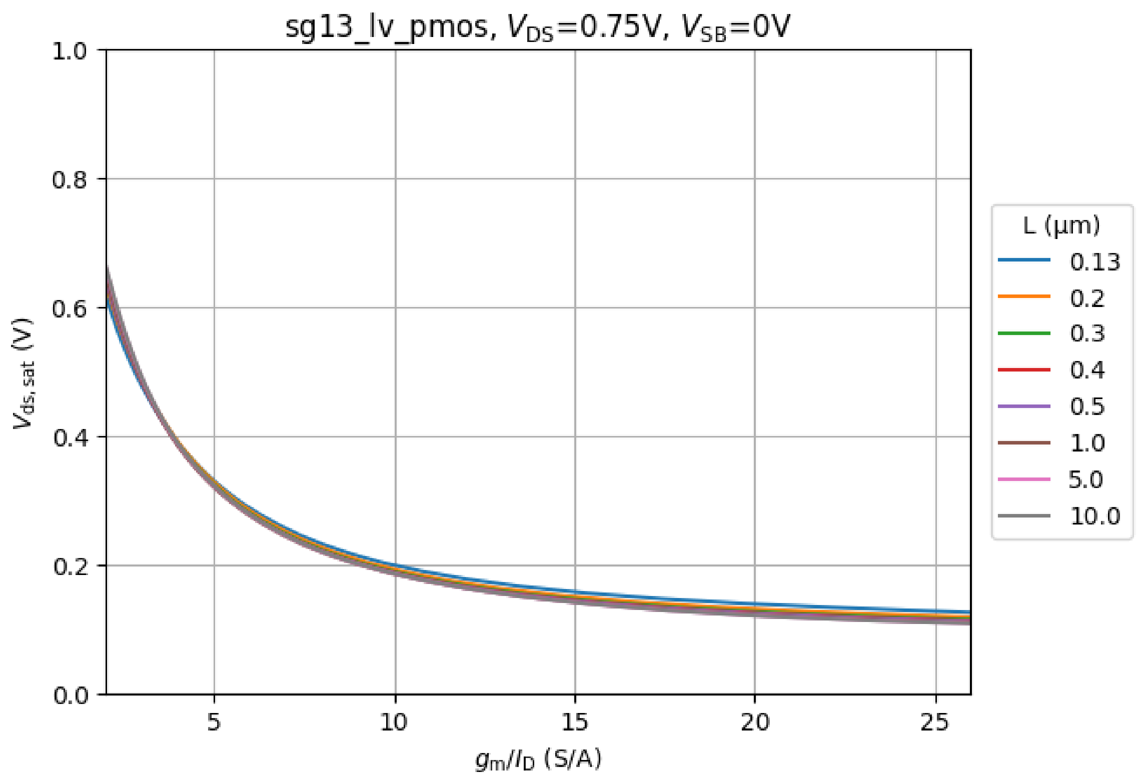
Source: Article Notebook

Drain current density I_D/W as a function of g_m/I_D and L:



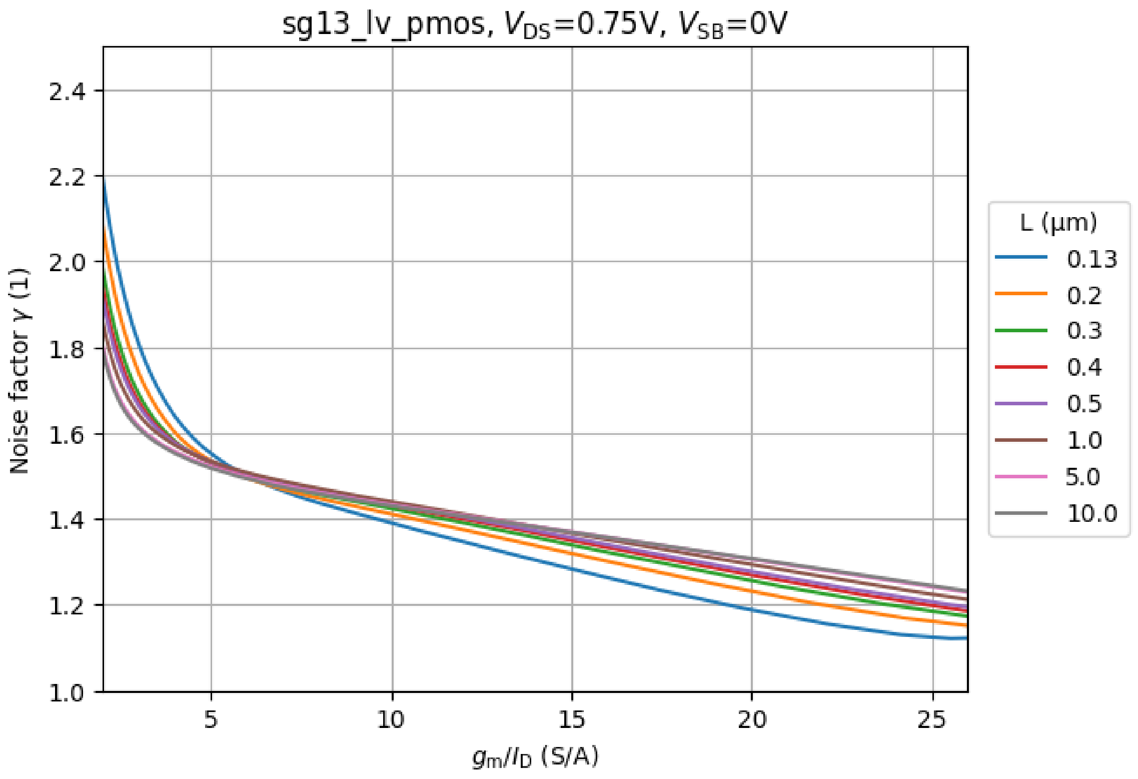
Source: Article Notebook

Minimum drain-source voltage $V_{\mathrm{ds,sat}}$ versus $g_{\mathrm{m}}/I_{\mathrm{D}}$ and L:



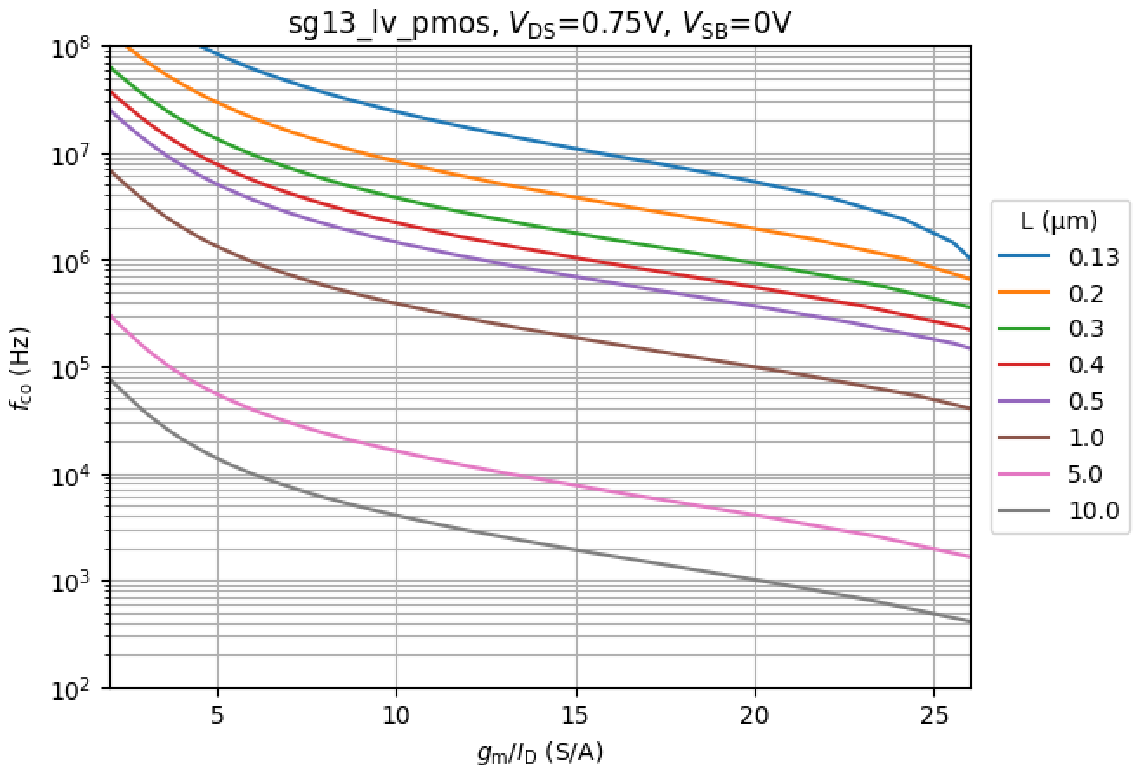
Source: Article Notebook

Noise factor γ versus $g_{\mathrm{m}}/I_{\mathrm{D}}$ and L:



Source: Article Notebook

Flicker noise corner frequency f_{co} versus $g_{\mathrm{m}}/I_{\mathrm{D}}$ and L. If you compare this figure carefully with the NMOS figure you can see that for some operating points the flicker noise for the PMOS is lower than for the NMOS. This is often true for CMOS technologies, so it can be an advantage to use a PMOS transistor in places where flicker noise is critical, like an OTA input stage. Using PMOS has the further advantage that the bulk node can be tied to source (which for NMOS is only possible in a triple-well technology, which is often not available), which gets rid of the body effect.



Source: Article Notebook

4 First Circuit: MOSFET Diode

The first (simple) circuit we will investigate is a MOSFET, where the gate is shorted with a drain, a so-called MOSFET "diode", which is shown in Figure 8. This diode is one half of a current mirror, which we will investigate in a future section.

Source: Article Notebook

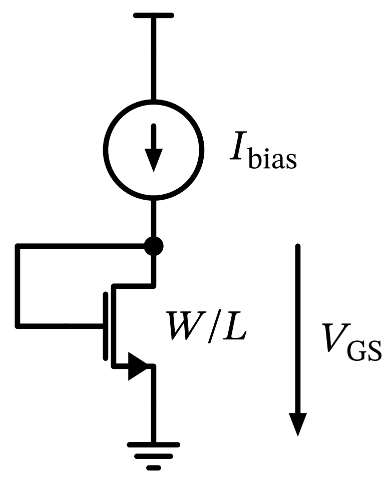


Figure 8: A MOSFET connected as a diode.

Source: Article Notebook

Why looking at a single-transistor circuit at all? By starting with the simplest possible circuit we can develop important skills in circuit analysis (setting up and calculating a small-signal model, calculating open-loop gain, calculate noise) and Xschem/ngspice simulation testbench creation. We safely assume that also the Mona Lisa was not Leonardo da Vinci's first painting, so let's start slow.

This diode is usually biased by a current source, shown as I_{bias} in the figure. Depending on MOSFET sizing with W and L, a certain gate-source voltage V_{GS} will develop. This voltage can be used as a biasing voltage for other circuit parts, for example.

Feedback in the MOSFET Diode

It is important to realize that this configuration essentially employs a feedback loop for operation. The voltage at the drain of the MOSFET is sensed by the gate, and the gate voltage changes until the I_{D} is exactly equal to I_{bias}. In this sense this is probably the smallest feedback circuit one can build.

4.1 MOSFET Diode Sizing

We will now build this circuit in Xschem. For sizing the MOSFET we will use the $g_{\text{m}}/I_{\text{D}}$ methodology introduced in Section 3.

Exercise: MOSFET Diode Sizing

Please build a MOSFET diode circuit in Xschem where you use an LV NMOS, set $I_{\text{bias}} = 20\,\mu\text{A}$, $L = 0.13\,\mu\text{m}$, and we want to use $g_{\text{m}}/I_{\text{D}} = 10$ (often a suitable compromise between transistor speed and g_{m} efficiency).

1. Use the figures in Section 3.2 to find out the proper value for W.
2. What is f_T for this MOSFET? What is the value for g_m and g_ds?
3. Draw the circuit in Xschem, and simulate the operating point. Do the values match to the values found out before during circuit sizing?

Before continuing, please finish the previous exercise. Once you are done, compare with the below provided solution.

Solution: MOSFET Diode Sizing

1. Using the fact that $I_\mathrm{bias} = I_\mathrm{D} = 20\,\mu\mathrm{A}$ and $g_\mathrm{m}/I_\mathrm{D} = 10$ directly provides $g_\mathrm{m} = 0.2\,\mathrm{mS}$.
2. Using the self-gain plot, we see that $g_\mathrm{m}/g_\mathrm{ds} \approx 21$, so $g_\mathrm{ds} \approx 9.5\,\mu\mathrm{S}$. The f_T can easily be found in the respective plot to be $f_\mathrm{T} = 23\,\mathrm{GHz}$.
3. The W of the MOSFET we find using the drain current density plot and the given bias current. Rounding to half-microns results in $W = 1\,\mu\mathrm{m}$.
4. Since we are looking at the graphs, we further find $\gamma = 0.84$, $V_\mathrm{ds,sat} = 0.18\,\mathrm{V}$, and $f_\mathrm{co} \approx 15\,\mathrm{MHz}$.
5. In addition, we expect $V_\mathrm{GS} \approx 0.6\,\mathrm{V}$.

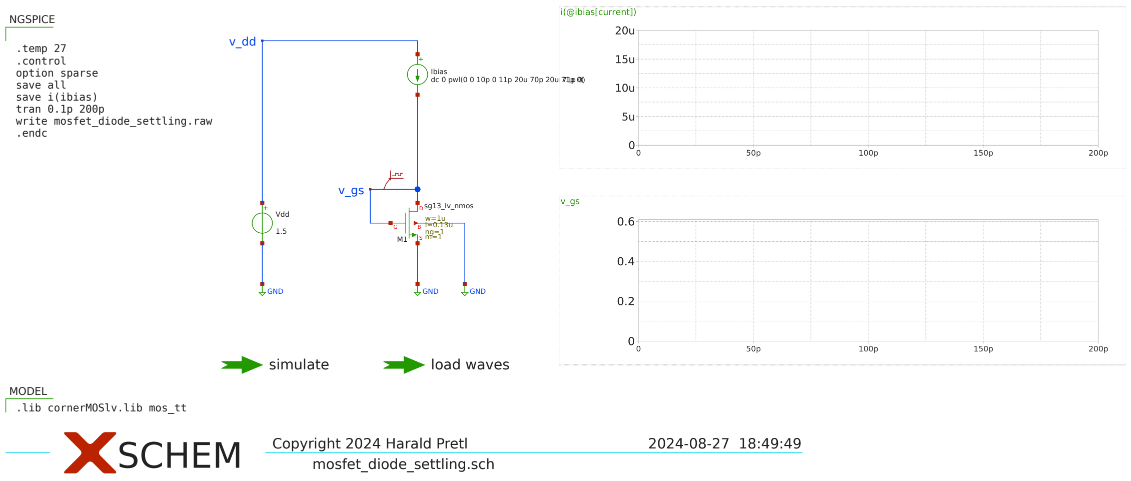
An example Jupyter notebook to extract these values accurately you can find here. An Xschem schematic for this exercise is provide as well.

4.2 MOSFET Diode Large-Signal Behaviour

As discussed above, the MOSFET diode configuration is essentially a feedback loop. Before we will analyse this loop in small-signal, we want to investgate how this loop settles in the time domain, and by doing this we can observe the large-signal settling behaviour. To simulate this, we change the dc bias source from the previous example to a transient current source, which we will turn on after some ns. The resulting Xschem testbench is shown in Figure 9.

In Figure 9 another interesting effect can be observed: While the turn-on happens quite rapidly (essentially the bias current source charges the gate capacitance, until the gate-source voltage is large enough that the drain current counteracts the bias current), the turn-off shows a very long settling tail. This is due to the fact that as the gate capacitance is discharged by the drain current the V_GS drops, which in turn reduces the drain current, which will make the discharge even slower. We have an effect similar to the capacitor discharge by a diode (Hellen 2003).

It is thus generally a good idea to add power-down switches to the circuits to disable the circuit quickly by pulling floating nodes to a defined potential (usually V_DD or V_SS) and to avoid long intermediate states during power down. This will also allow a turn-on from a well-defined off-state.


NGSPICE

```
.temp 27
.control
option sparse
save all
save i(ibias)
tran 0.1p 200p
write mosfet_diode_settling.raw
.endc
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Figure 9: Testbench for MOSFET diode transient settling.

4.3 MOSFET Diode Small-Signal Analysis

We now want to investigate the small-signal behaviour of the MOSFET diode. Based on the small-signal model of the MOSFET in Figure 5 we realize that gate and drain are shorted, and we also connect bulk to source. We can thus simplify the circit to the one shown in Figure 10.

Source: Article Notebook

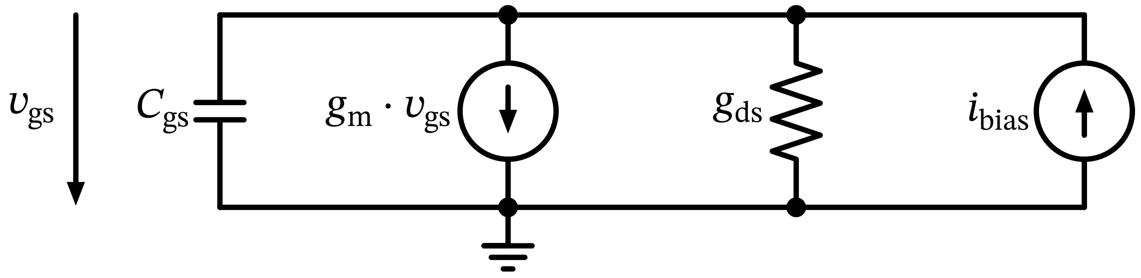


Figure 10: The MOSFET diode small-signal model.

Source: Article Notebook

Ground Node Selection

For small-signal analysis we would not need to declare one node as the ground potential. However, when doing so, and selecting the ground node strategically, we can simplify the analysis, as we usually do not formulate KCL for the ground node (as we have only $N-1$ independent KCL equations, N being the number of nodes in the circuit), and the potential difference equations are simpler if one node is at $0\,V$.

For calculating the small-signal impedance of the MOSFET diode we formulate KCL at the top node to get

$$i_{\mathrm{bias}} - sC_{\mathrm{gs}}v_{\mathrm{gs}} - g_{\mathrm{m}}v_{\mathrm{gs}} - g_{\mathrm{ds}}v_{\mathrm{gs}} = 0.$$

It follows that

$$Z_{\mathrm{diode}}(s) = \frac{v_{\mathrm{gs}}}{i_{\mathrm{bias}}} = \frac{1}{g_{\mathrm{m}} + g_{\mathrm{ds}} + sC_{\mathrm{gs}}}. \tag{4}$$

When neglecting g_{ds} and at dc we get $Z_{\mathrm{diode}} = 1/g_{\mathrm{m}}$, which is an important result and should be memorized.

! The Admittance is Your Friend

In circuit analysis it is often algebraically easier to work with admittance instead of impedance, so please remember that Ohm's law for a conductance is $I = G \cdot V$, and for a capacitance is $I = sC \cdot V$. When writing equations, it is also practical to keep sC together, so we will strive to sort terms accordingly.

Looking at Equation 4 we see that for low frequencies, the diode impedance is resistive, and for high frequencies it becomes capactive as the gate-source capacitance starts to dominate. The corner frequeny of this low-pass can be calculated as

$$\omega_{\mathrm{c}} = \frac{g_{\mathrm{m}} + g_{\mathrm{ds}}}{C_{\mathrm{gs}}} \approx \omega_{\mathrm{T}}$$

which is pretty much the transit frequency of the MOSFET!

4.4 MOSFET Diode Stability Analysis

The diode-connected MOSFET forms a feedback loop. What is the open-loop gain? For calculating it, we are breaking the loop, and apply a dummy C_{gs}^{*} at the right side to keep the impedances correct. A circuit diagram is shown in Figure 11, we break the loop at the dotted connection. As we can see in this example, it is critically important when breaking up a loop for analysis (also for simulation!) to keep the terminal impedances the same. Only in special cases where the load impedance is very high or the driving impedance is very low is it acceptable to disregard loading effects!

Source: Article Notebook

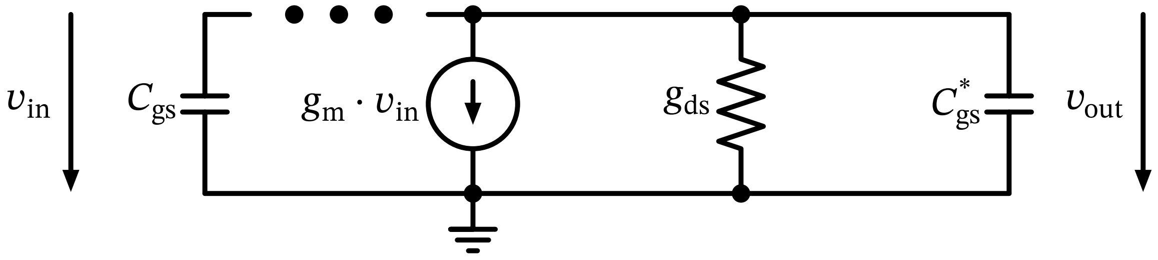


Figure 11: The MOSFET diode small-signal circuit for open-loop analysis.

Source: Article Notebook

By inspecting Figure 11 we see that

$$v_{\text{out}} = -g_{\text{m}} v_{\text{in}} \frac{1}{g_{\text{ds}} + sC_{\text{gs}}}.$$

The open-loop gain $H_{\text{ol}}(s)$ is thus

$$H_{\text{ol}}(s) = \frac{v_{\text{out}}}{v_{\text{in}}} = -\frac{g_{\text{m}}}{g_{\text{ds}} + sC_{\text{gs}}}. \tag{5}$$

Inspecting Equation 5 we realize that

1. the dc gain $g_{\text{m}}/g_{\text{ds}}$ is the self-gain of the MOSFET, so $20\log(0.2 \cdot 10^{-3}/9.6 \cdot 10^{-6}) = 26.4\,\text{dB}$, and
2. there is a pole at $\omega_{\text{p}} = -g_{\text{ds}}/C_{\text{gs}}$, which is at $9.6 \cdot 10^{-6}/(2\pi \cdot 1.4 \cdot 10^{-15}) = 1.1\,\text{GHz}$.

With this single pole location in $H_{\text{ol}}(s)$ this loop is perfectly stable at under all conditions.

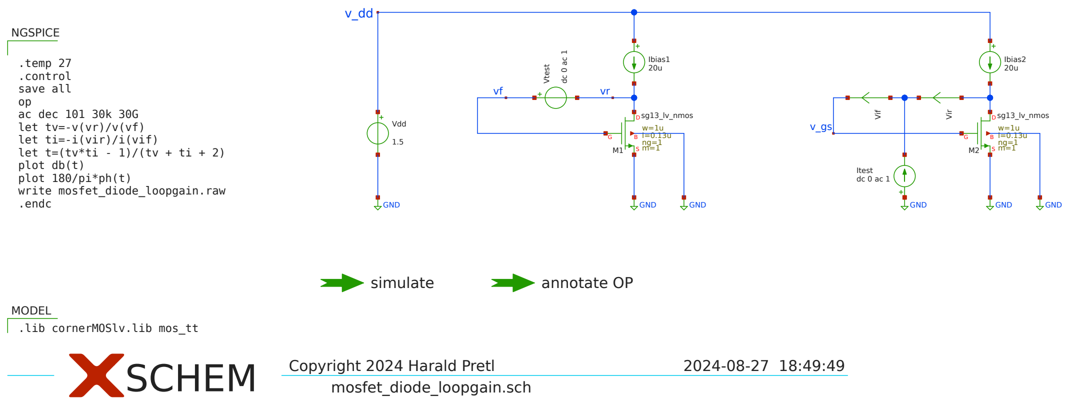
The question is now how to simulate this open-loop gain, and how to break the loop open in simulation? In general there are various methods, as we can use artificially large (ideal) inductors and capacitors to break loops open and still establish the correct dc operating points for the ac loop analysis. However, mimicking the correct loading can be an issue, and requires a lot of careful consideration.

There is an alternative method which breaks the loop open only by adding an ac voltage source in series (thus keeps the dc operating point intact), or injects current using a current source. Based on both measurements the open-loop gain can be calculated. This is called **Middlebrook's method** (Middlebrook 1975) which is based on double injection, and we will use it for our loop simulations. This method is detailed in Section 18.

We now want to simulate the open-loop transfer function $H_{\text{ol}}(s)$ by using Middlebrook's method and confirm our analysis above.

Exercise: MOSFET Diode Loop Analysis

Please build a simulation testbench in Xschem to simulate the open-loop transfer function of the MOSFET diode. Confirm the dc gain and pole location as given by Equation 5.
If you are getting stuck you can look at this Xschem testbench, shown in Figure 12.



NGSPICE

```
.temp 27
.control
save all
op
ac dec 101 30k 30G
let tv=-v(vr)/v(vf)
let ti=-i(vir)/i(vif)
let t=(tv*ti - 1)/(tv + ti + 2)
plot db(t)
plot 180/pi*ph(t)
write mosfet_diode_loopgain.raw
.endc
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Figure 12: Testbench for MOSFET diode stability analysis.

From simulation we see that the open-loop gain is 24.9 dB at low frequencies, which matches quite well our prediction of 26.4 dB. In the Bode plot we see a low-pass with a −3 dB corner frequency of 1.4 GHz, which again is fairly close to our prediction of 1.1 GHz.

4.5 MOSFET Diode Noise Calculation

As a final exercise on the MOSFET diode circuit we want to calculate the output noise when we consider V_{GS} the output reference voltage which is created when passing a bias current through the MOSFET diode. The bias current we will assume noiseless.

We will use the small-signal circuit shown in Figure 13.

Source: Article Notebook

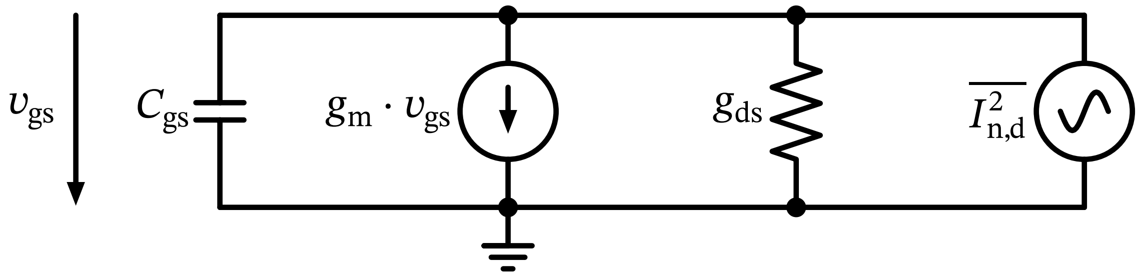


Figure 13: The MOSFET diode small-signal model with drain noise source.

Source: Article Notebook

As we have already calculated the small-signal diode impedance in Equation 4 we will use this result, and just note that the drain current noise of the MOSFET flows through this impedance. The noise voltage at v_{gs} is thus given as

$$\overline{V_n^2} = Z_{diode}^2 \overline{I_{n,d}^2}.$$

The drain current noise of the MOSFET is given as (introduced in Section 2.1.2)

$$\overline{I_{n,d}^2} = 4kT\gamma g_m.$$

For low frequencies (ignoring g_{ds} and C_{gs}) we get

$$\overline{V_n^2} = Z_{diode}^2 \overline{I_{n,d}^2} = \frac{1}{g_m^2} 4kT\gamma g_m = \frac{4kT\gamma}{g_m}$$

which is the thermal noise of a resistor of value $1/g_m$ enhanced by the factor γ.

We now calculate the full equation, and after a bit of algebra arrive at

$$\overline{V_n^2}(f) = \frac{4kT\gamma g_m}{(g_m + g_{ds})^2 + (2\pi f C_{gs})^2}. \tag{6}$$

If we are interested in the PSD of the noise then Equation 6 gives us the result. If we are interested in the rms value (the total noise) we need to integrate this equation, using the following identity:

Useful Integral for Noise Calculations

$$\int_0^\infty \frac{a}{b^2 + c^2 f^2} df = \frac{\pi}{2} \frac{a}{b \cdot c} \tag{7}$$

Using the integral help in Equation 7, we can easily transform Equation 6 to

$$V_{n,rms}^2 = \int_0^\infty \overline{V_n^2}(f) df = \frac{kT\gamma g_m}{(g_m + g_{ds}) C_{gs}}. \tag{8}$$

The form of Equation 8 is the exact solution, but we gain additional insight if we assume that $g_\mathrm{m} + g_\mathrm{ds} \approx g_\mathrm{m}$ and then

$$V_\mathrm{n,rms}^2 = \frac{kT\gamma}{C_\mathrm{gs}}. \tag{9}$$

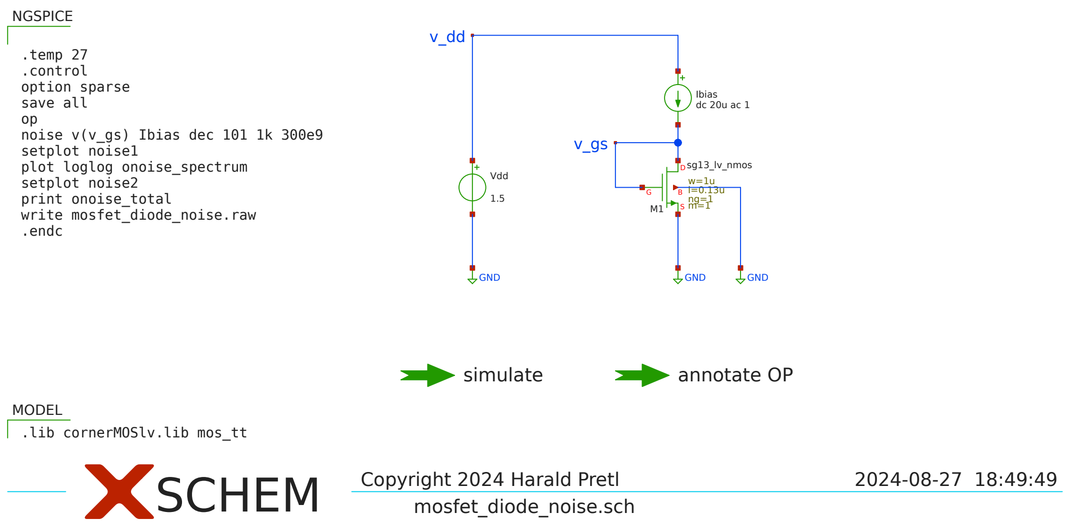
Inspecting Equation 9 we see our familiar kT/C noise enhanced by the factor γ! Calculating this value for our MOSFET diode we get $\sqrt{V_\mathrm{n,rms}^2} = \sqrt{1.38 \cdot 10^{-23} \cdot 300 \cdot 0.84/1.4 \cdot 10^{-15}} = 1.58\,\mathrm{mV}$, which is a sizeable value! We run circuits in this technology at $V_\mathrm{DD} = 1.5\,\mathrm{V}$, which leaves us with a signal swing of ca. $1.1\,\mathrm{V_{pp}}$, resulting in a dynamic range in this case of $20\log(1.58 \cdot 10^{-3}/0.39) \approx -48\,\mathrm{dB}$.

! Large Bandwidth and Noise

Large BW circuits can integrate noise over a wide bandwidth resulting in considerable rms noise.

Exercise: MOSFET Diode Noise

Please build a simulation testbench in Xschem to simulate the noise performance of the MOSFET diode, and confirm the rms noise value that we just calculated. Look at the rms value and the PSD of the noise, and play around with the integration limits. What is the effect? Can you see the flicker noise in the PSD? How much is its contribution to the rms noise?
If you are getting stuck you can look at this Xschem testbench, shown in Figure 14.


NGSPICE

```
.temp 27
.control
option sparse
save all
op
noise v(v_gs) Ibias dec 101 1k 300e9
setplot noise1
plot loglog onoise_spectrum
setplot noise2
print onoise_total
write mosfet_diode_noise.raw
.endc
```

MODEL

```
.lib cornerMOSlv.lib mos_tt
```

Figure 14: Testbench for MOSFET diode noise analysis.

4.6 Conclusion

In this section we investigated the simple MOSFET-diode circuit. We learned important skills like how to derive a small-signal model, how to calculate important features like noise and open-loop gain for stability analysis. We introduced Middlebrook's method to have a mechanism to open up loops in simulation (and calculation) without disturbing operating points for change loading conditions.

If you feel that you have not yet mastered these topics or are uncertain in the operation of ngspice, please go back to the beginning of the section and read through the theory and redo the exercises.

5 Current Mirror

In this section we will look into a fundamental building block which is often used in integrated circuit design, the **current mirror**. A diagram is shown in Figure 15 with one MOSFET diode converting the incoming bias current into a voltage, and two output MOSFETs working as current sources, which are biased from the diode. By properly selecting all W and L the input current can be scaled, and multiple copies can be created at once. Shown in the figure are two output currents, but any number of parallel branches can be realized.

Source: Article Notebook

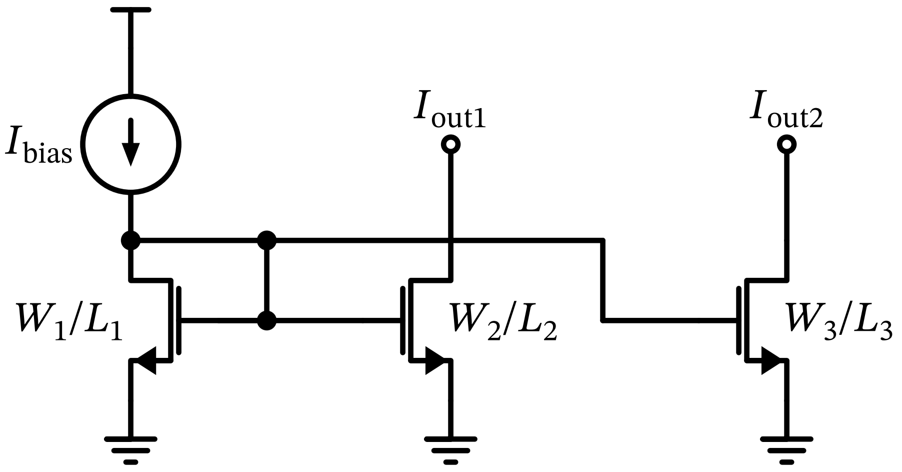


Figure 15: A current mirror with two output branches.

Source: Article Notebook

The output current I_{out1} is then given by

$$I_{\mathrm{out1}} = I_{\mathrm{bias}} \frac{W_2}{L_2} \frac{L_1}{W_1}$$

and the output current I_{out2} is given by

$$I_{\text{out2}} = I_{\text{bias}} \frac{W_3}{L_3} \frac{L_1}{W_1}.$$

For good matching in layout care has to be taken that the MOSFET widths and lengths are constructed out of **unit elements** of identical size, where an appropiate amount of these single units are then arranged in series or parallel configuration to arrive at the target W and L.

As we know from earlier investigations of the MOSFET performance in Section 3 the drain current of a MOSFET is a function of V_{GS} and V_{DS}. As long as the MOSFET stays in saturation (i.e., $V_{\text{DS}} > V_{\text{ds,dsat}}$) the drain current is just a mild function of V_{DS} (essentially the effect of g_{ds}, which is the output conductance of the MOSFET). A fundamental flaw of the basic current mirror shown in Figure 15 is the mismatch of the V_{DS} of the MOSFET. The input-side diode has $V_{\text{GS}} = V_{\text{DS}}$, whereas the output current sources have a V_{DS} depending on the connected circuitry. Improved current mirrors exist (basically fixing this flaw), still, when just a simple current mirror is required this structure is used for its simplicity.

Exercise: Current Mirror

Please construct a current mirror based on the MOSFET-diode which we sized in Section 4. The input current $I_{\text{bias}} = 20\,\mu\text{A}$, and we want three output currents of size $10\,\mu\text{A}$, $20\,\mu\text{A}$, and $40\,\mu\text{A}$.
Sweep the output voltage of all three current branches and see over which voltage range an acceptable current is created. For which output voltage range is the current departing from its ideal value, and why?
You see that the slope of the output current is quite bad, as g_{ds} is too large. We can improve this by changing the length to $L = 5\,\mu\text{m}$ (for motivation, please look at the graphs in Section 3). In addition, for a current mirror we are not interested in a high $g_{\text{m}}/I_{\text{D}}$ value, so we can use $g_{\text{m}}/I_{\text{D}} = 5$ in this case. Please size the current mirror MOSFETs accordingly (please round the W to half micron, to keep sizes a bit more practical). Compare this result to the previous one, what changed?
In case you get stuck, here are Xschem schematics for the original and the improved current mirrors.

6 Differential Pair

Like the current mirror in Section 5 the **differential pair** is an ubiquitous building block often used in integrated circuit design. The fundamental structure is given in Figure 16.

Source: Article Notebook

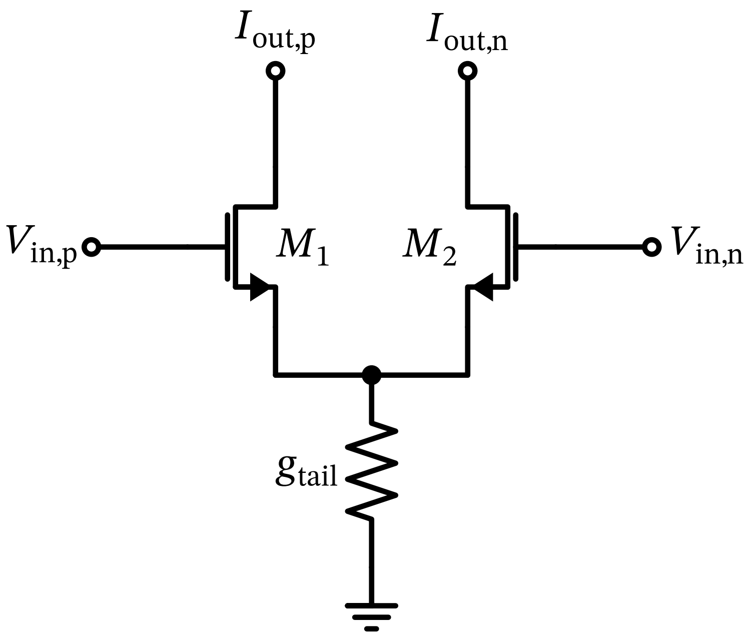


Figure 16: A differential pair.

Source: Article Notebook

In order to understand its operation it is instructive to separate the input condition into (1) a purely differential voltage, and (2) into a common-mode voltage, and see what the impact on the output currents is.

6.1 Differential Operation of the Diffpair

For a differential mode of operation we assume that the input common mode voltage is constant, i.e. $V_{\text{in,p}} + V_{\text{in,n}} = V_{\text{CM}}$. A differential input voltage v_{in} then results in

$$V_{\text{in,p}} = V_{\text{CM}} + \frac{v_{\text{in}}}{2}$$

and

$$V_{\text{in,n}} = V_{\text{CM}} - \frac{v_{\text{in}}}{2}.$$

For a small-signal differential drive the potential at the tail point stays constant and we can treat it as a virtual ground. The output current on each side is then given by (neglecting g_{ds} and g_{mb} of M_1 and M_2)

$$i_{\text{out,p}} = g_{\text{m1}} \left(\frac{v_{\text{in}}}{2} \right)$$

and

$$i_{\text{out,n}} = g_{\text{m2}} \left(-\frac{v_{\text{in}}}{2} \right).$$

Usually we assume symmetry in the differential pair, so $g_{\mathrm{m1}} = g_{\mathrm{m2}} = g_{\mathrm{m}}$. The differential output current i_{out} is then given by

$$i_{\mathrm{out}} = i_{\mathrm{out,p}} - i_{\mathrm{out,n}} = g_{\mathrm{m}} v_{\mathrm{in}} \tag{10}$$

We see in Equation 10 that the differential output current is simply the differential input voltage multiplied by the g_{m} of the individual transistor. We also note that the bottom conductance g_{tail} plays no role for the small-signal differential operation.

6.2 Common-Mode Operation of the Diffpair

Usually, the source conductance g_{tail} is realized by a current source and ideally should be $g_{\mathrm{tail}} = 0$. If this is the case, then the output currents are not a function of the common-mode input voltage, and (I_{tail} is set by the tail current source)

$$I_{\mathrm{out,p}} = I_{\mathrm{out,n}} = \frac{I_{\mathrm{tail}}}{2}.$$

However, if we assume a realistic tail current source then $g_{\mathrm{tail}} > 0$. For analysis we can simply look at a half circuit since everything is symmetric. In order to simplify the analysis a bit we remove all capacitors from the MOSFET small-signal model and set $g_{\mathrm{ds}} = g_{\mathrm{mb}} = 0$. We arrive then at the small-signal equivalent circuit shown in Figure 17 (note that we set $v_{\mathrm{in,p}} = v_{\mathrm{in,n}} = v_{\mathrm{in}}$ and $i_{\mathrm{out,p}} = i_{\mathrm{out,n}} = i_{\mathrm{out}}$ under symmetry considerations).

Source: Article Notebook

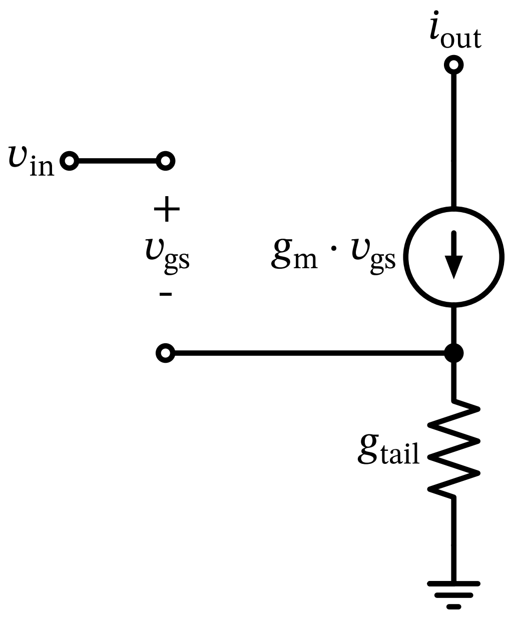


Figure 17: Small-signal model of the differential pair half-circuit in common-mode operation.

Source: Article Notebook

Formulating KVL for the input-side loop we get

$$v_{\mathrm{in}} = v_{\mathrm{gs}} + \frac{i_{\mathrm{ds}}}{g_{\mathrm{tail}}}.$$

With $i_{\mathrm{out}} = i_{\mathrm{ds}} = g_{\mathrm{m}} v_{\mathrm{gs}}$ we arrive at

$$i_{\mathrm{out}} = \frac{g_{\mathrm{m}} g_{\mathrm{tail}}}{g_{\mathrm{m}} + g_{\mathrm{tail}}} v_{\mathrm{in}} \tag{11}$$

Interpreting Equation 11 we can distinguish the following extreme cases:

1. If $g_{\mathrm{tail}} = 0$ (ideal tail current source) then $i_{\mathrm{out}} = 0$, the common-mode voltage variation from the input is suppressed and does not show up at the common-mode output current (which is constant due to the ideal tail current source). This is usually the case that we want to achieve.
2. If $g_{\mathrm{tail}} = \infty$ then $i_{\mathrm{out}} = g_{\mathrm{m}} v_{\mathrm{in}}$, which means the output current is a function of the MOSFET g_{m}. If everything is perfectly matched, then the differential output current is zero, but the common-mode output current changes according to the common-mode input voltage. In special cases this can be a wanted behaviour, this configuration is called a "pseudo-differential pair."

7 A Basic 5-Transistor OTA

Suited with the knowledge of basic transitor operation (Section 2 and Section 3) and the working knowledge of the current mirror (Section 4 and Section 5) as well as the differential pair (Section 6) we can now start to design our first real circuit. A fundamental (simple) circuit that is often used for basic tasks is the 5-transistor operational transconductance amplifier (OTA). A circuit diagram of this 5T-OTA is shown in Figure 18.

Source: Article Notebook

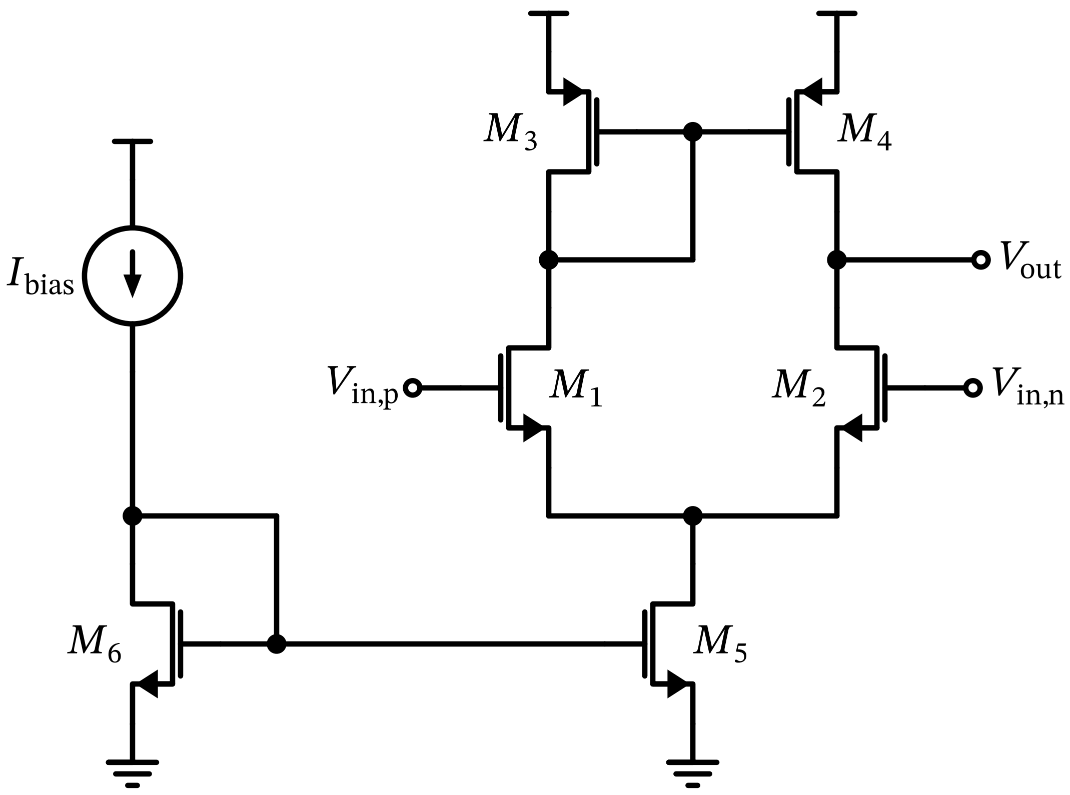


Figure 18: The 5-transistor OTA.

Source: Article Notebook

The operation is as follows: $M_{1,2}$ form a differential pair which is biased by the current source M_5. $M_{5,6}$ form a current mirror, thus the input bias current I_{bias} sets the bias current in the OTA. The differential pair $M_{1,2}$ is loaded by the current mirror $M_{3,4}$ which mirrors the output current of M_1 to the right side. Here, the currents from M_4 and M_2 are summed, and together with the conductance effective at the output node a voltage builds up.

We note that $M_{1,2}$ and $M_{3,4}$ need to be symmetric, thus will have the same W and L dimensioning. $M_{5,6}$ we scale accordingly to set the correct bias current in the OTA.

As this is an OTA the output is a current; if the load impedance is high (i.e., purely capacitive, which is often the case in integrated circuits when driving MOSFET inputs) then the voltage gain of the OTA can be high (of course, in this simple OTA it is limited). With a high-impedance loading this OTA can provide a voltage output, and this is actually how OTAs are mostly operated.

7.1 Voltage Buffer with OTA

In order to design an OTA we need an application, and from this we need to derive the circuit specifications. We want to use this OTA to realize a voltage buffer which lighly loads a voltage source and can drive a large capacitive load. Such a configuration is often used to, e.g., buffer a reference voltage that is needed (and thus loaded) by another circuit. The block diagram of this configuration is shown in Figure 19.

Source: Article Notebook

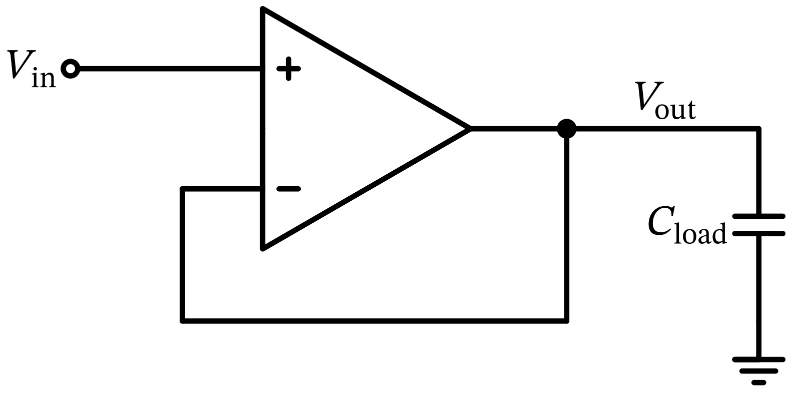


Figure 19: A voltage buffer (based on OTA) driving a capacitive load.

Source: Article Notebook

If the voltage gain of the OTA is Figure 19 is high, then $V_{\text{out}} \approx V_{\text{in}}$. We now want to design an OTA for this application for the following spefication values (see Table 2). These values are rather typical of what could be expected for such a buffer design.

Table 2: Voltage buffer specification

Specification	Value	Unit
Supply voltage	$1.45 < \underline{1.5} < 1.55$	V
Temperature range (industrial)	$-40 < \underline{27} < 125$	degC
Load capacitance C_{load}	50	fF
Input voltage range (for buffering 2/3 bandgap voltage)	$0.7 < \underline{0.8} < 0.9$	V
Signal bandwidth (3dB)	> 10	MHz
Output voltage error	< 3	%
Total output noise (rms)	< 1	mV_{rms}
Supply current (as low as possible)	< 10	µA
Stability	stable for rated C_{load}	
Turn-on time (settled to with 1%)	< 10	µs
Externally provided bias current (nominal)	20	µA

7.2 Large-Signal Analysis of the OTA

The first step when receiving a design task is to look at the specifications, and see whether they make sense. Detailed performance of the design will be the result of the circuit simulation, but before we step into sizing we need to do a few simple calculations to (a) allows to do back-of-the-envelope gauging if the specification makes sense, and (b) the derived analytical equations will serve as guide for the sizing procedure.

- In terms of large-signal operation, we will now check whether the input and output voltage range, as well as the settling time can be roughly met.
- When the input is at its maximum of $0.9\,\mathrm{V}$, we see that we need to keep M_1 in saturation. We can calculate that $V_{\mathrm{DS1}} = V_{\mathrm{DD}} - |V_{\mathrm{GS3}}| + V_{\mathrm{GS1}} - V_{\mathrm{in}} = 1.45 - 0.6 + 0.6 - 0.9 = 0.55\,\mathrm{V}$, which leaves enough margin.
- When the input is at its minimum of $0.7\,\mathrm{V}$, we see that the V_{DS5} of M_5 is calculated as $V_{\mathrm{DS5}} = V_{\mathrm{in}} - V_{\mathrm{GS1}} = 0.7 - 0.6 = 0.1\,\mathrm{V}$, so this leaves little margin, but likely V_{GS1} will be smaller, so it should work out.
- For the output voltage, when the output voltage is on the high side, it leaves $|V_{\mathrm{DS4}}| = V_{\mathrm{DD}} - V_{\mathrm{out}} = 1.45 - 0.9 = 0.55\,\mathrm{V}$, which is enough margin.

In summary, we think that we can make an NMOS-input OTA like the one in Figure 18 work for the required supply and input- and output voltages. If this would not work out, we need to look for further options, like a PMOS-input OTA, or a NMOS/PMOS-input OTA.

Another large-signal specification item that we can quickly check is the settling time. Under slewing conditions, the complete bias current in the OTA is steered towards the output (try to understand why this is the case), so when the output capacitor is fully discharged, and we assume just a linear ramp due to constant-current charging of the output capacitor, the settling time is

$$T_{\mathrm{slew}} \approx \frac{C_{\mathrm{load}} V_{\mathrm{out}}}{I_{\mathrm{tail}}} = \frac{50 \cdot 10^{-15} \cdot 1.3}{10 \cdot 10^{-6}} = 6.5\,\mathrm{ns}$$

so this leaves plenty of margin for additional slow-signal settling due to the limited bandwidth, as well as reducing the supply current.

The small-signal settling (assuming one pole at the bandwidth corner frequency) leads to an approximate settling time (1% error corresponds to $\approx 5\tau$) of

$$T_{\mathrm{slew}} \approx \frac{5}{2\pi f_{\mathrm{c}}} = \frac{5}{2\pi \cdot 1 \cdot 10^{-6}} = 0.8\,\mu\mathrm{s}.$$

which also checks out.

7.3 Small-Signal Analysis of the OTA

In order to size the OTA components we need to derive how MOSFET parameters define the performance. The important small-signal metrics are

- dc gain A_0
- gain-bandwidth product (GBW)
- output noise

The specification for GBW is given in Table 2, the dc gain we have to calculate from the voltage accuracy specification. For a voltage follower in the configuration shown in Figure 19 the voltage gain is given by

$$\frac{V_{\mathrm{out}}}{V_{\mathrm{in}}} = \frac{A_0}{1 + A_0}. \tag{12}$$

So in order to reach an output voltage accuracy of at least 3% we need a dc gain of $A_0 > 30.2\,\mathrm{dB}$. To allow for process and temperature variation we need to add a bit of extra gain as margin.

7.3.1 OTA Small-Signal Transfer Function

In order to derive the governing equations for the OTA we will make a few simplifications:

- We will set $g_{\mathrm{mb}} = 0$ for all MOSFETs.
- We will further set $C_{\mathrm{gd}} = 0$ for all MOSFETs except for M_4 where we expect a Miller effect on this capacitor, and we could add its effect by increasing the capacitance at the gate node of $M_{3,4}$ (for background please see Section 19). Hoewever, as this does not create a dominant pole in this circuit, we consider this a minor effect (see Equation 15). Thus, only C_{gs34} is considered at the gate node of the current mirror load.
- We assume $g_{\mathrm{m}} \gg g_{\mathrm{ds}}$, so we set $g_{\mathrm{ds1}} = g_{\mathrm{ds3}} = 0$.
- The drain capacitance of M_2 and M_4, as well as the gate capacitance of M_2 we can add to the load capacitance C_{load}. Note that C_{gs2} can be added because of the feedback connection between the inverting input and the output. However, this is not shown in the small-signal equivalent circuits below, because we are interested in the open-loop transfer function.

The resulting small-signal equivalent circuit is shown in Figure 20.

Refresh MOSFET Small-Signal Model

Please review the MOSFET small-signal equivalent model in Figure 5 at this point. For the PMOS just flip the model upside-down.

Source: Article Notebook

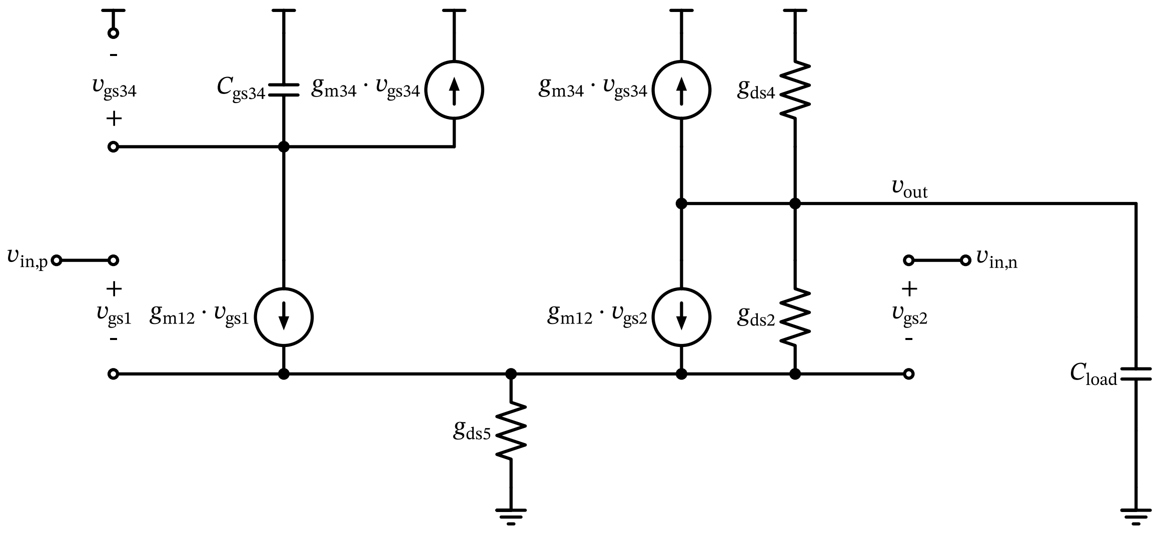


Figure 20: 5-transistor OTA small-signal model.

Source: Article Notebook

We can further simplify the output side by recognizing that the impedance looking from the output down we have g_{ds2} in series with $g_{ds5} + g_{m12}$ (since we treat M_1 as a common-gate stage when looking from the output, and since it is loaded by a low impedance of g_{m34}^{-1} we can approximate the impedance looking into the source of M_1 with g_{m12}^{-1}). With the approximation that $g_m \gg g_{ds}$ the parallel connection of g_{m12} and g_{ds5} is dominated by g_{m12} and series connection by g_{ds2}. Therefore, we can move $g_{ds2}+g_{ds4}$ in parallel to C_{load}. Further, assuming a differential drive with a virtual ground at the tailpoint we can remove g_{ds5}. The current source $g_{m34}v_{gs34}$ is replace with the equivalent conductance g_{m34}. This results in the further simplified equivalent circuit shown in Figure 21.

Source: Article Notebook

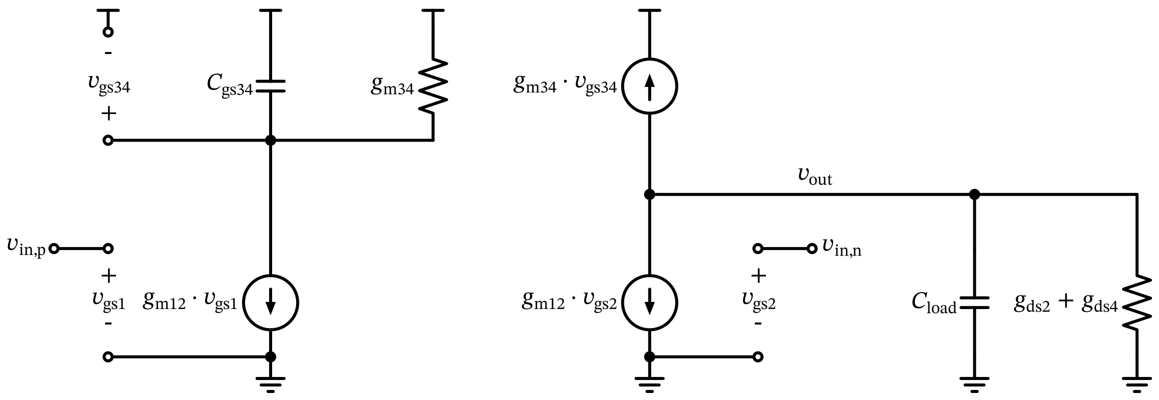


Figure 21: 5-transistor OTA small-signal model with further simplifications.

Source: Article Notebook

In the simplified circuit model in Figure 21 we can see that we have two poles in the circuit, one at the gate note of $M_{3,4}$, and one at the output. Realizing that $v_{\mathrm{in,p}} = v_{\mathrm{in}}/2$ and $v_{\mathrm{in,n}} = -v_{\mathrm{in}}/2$ we can formulate KCL at the output node to

$$-g_{\mathrm{m34}} V_{\mathrm{gs34}} - \left(-g_{\mathrm{m12}} \frac{V_{\mathrm{in}}}{2}\right) - V_{\mathrm{out}}(g_{\mathrm{ds2}} + g_{\mathrm{ds4}} + sC_{\mathrm{load}}) = 0. \tag{13}$$

We further realize that

$$V_{\mathrm{gs34}} = -g_{\mathrm{m12}} \frac{V_{\mathrm{in}}}{2} \frac{1}{g_{\mathrm{m34}} + sC_{\mathrm{gs34}}}. \tag{14}$$

By combining Equation 13 and Equation 14 and after a bit of algebraic manipulation we arrive at

$$A(s) = \frac{V_{\mathrm{out}}}{V_{\mathrm{in}}} = \frac{g_{\mathrm{m12}}}{2} \frac{2g_{\mathrm{m34}} + sC_{\mathrm{gs34}}}{(g_{\mathrm{m34}} + sC_{\mathrm{gs34}})(g_{\mathrm{ds2}} + g_{\mathrm{ds4}} + sC_{\mathrm{load}})}. \tag{15}$$

When we now inspect Equation 15 we can see that for low frequencies the gain is

$$A(s \to 0) = A_0 = \frac{g_{\mathrm{m12}}}{g_{\mathrm{ds2}} + g_{\mathrm{ds4}}} \tag{16}$$

which is plausible, and confirms the requirement of a high impedance at the output node. For very large frequencies we get

$$A(s \to \infty) = \frac{g_{\mathrm{m12}}}{2sC_{\mathrm{load}}} \tag{17}$$

which is essentially the behaviour of an integrator, and we can use Equation 17 to calculate the frequency where the gain drops to 1:

$$f_{\mathrm{ug}} = \frac{g_{\mathrm{m12}}}{4\pi C_{\mathrm{load}}}$$

when looking at Equation 15 we see that we have a dominant pole at s_{p} and a pole-zero doublet with $s_{\mathrm{pd}}/s_{\mathrm{zd}}$:

$$s_{\mathrm{p}} = -\frac{g_{\mathrm{ds2}} + g_{\mathrm{ds4}}}{C_{\mathrm{load}}}$$

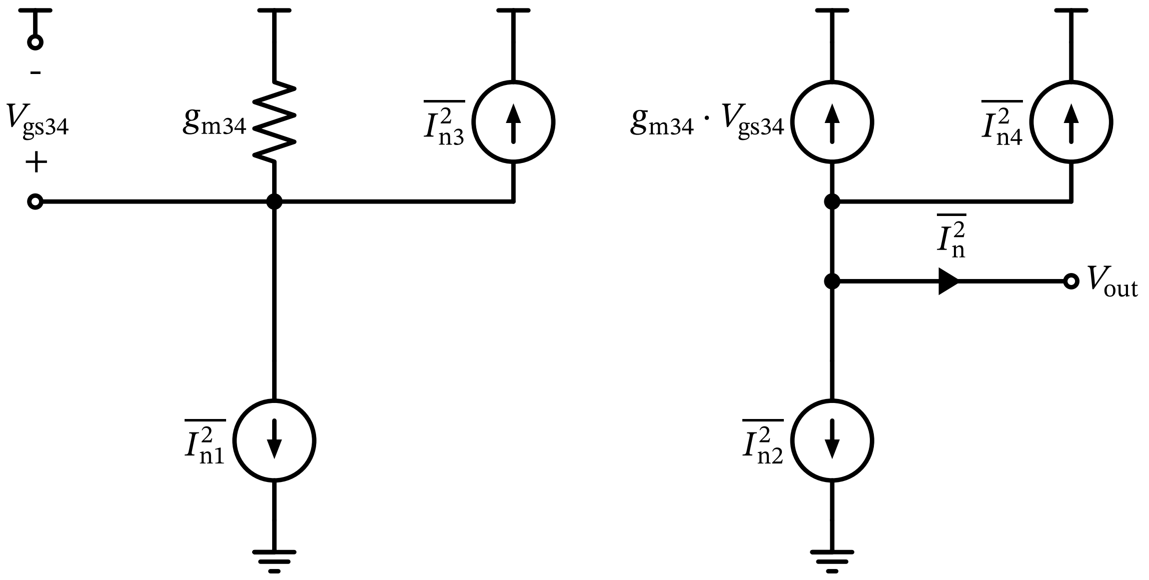
$$s_{\mathrm{pd}} = -\frac{g_{\mathrm{m34}}}{C_{\mathrm{gs34}}}$$

$$s_{\mathrm{zd}} = -\frac{2g_{\mathrm{m34}}}{C_{\mathrm{gs34}}}$$

7.3.2 OTA Noise

For the noise analysis we ignore the pole-zero doublet due to C_{gs34} (we assume minor impact due to this) and just consider the dominant pole. For the noise analysis at the output we set the input signal to zero, and thus we arrive at the simplified small-signal circuit shown in Figure 22.

Source: Article Notebook



V_{gs34}
-
+
g_{m34}
$\overline{I_{\mathrm{n3}}^2}$
$g_{\mathrm{m34}} \cdot V_{\mathrm{gs34}}$
$\overline{I_{\mathrm{n4}}^2}$
$\overline{I_{\mathrm{n}}^2}$
V_{out}
$\overline{I_{\mathrm{n1}}^2}$
$\overline{I_{\mathrm{n2}}^2}$

Figure 22: 5-transistor OTA small-signal model for noise calculation.

Source: Article Notebook

We see that

$$\overline{V_{\mathrm{gs34}}^2} = \frac{1}{g_{\mathrm{m34}}^2} \left(\overline{I_{\mathrm{n1}}^2} + \overline{I_{\mathrm{n3}}^2} \right).$$

! Noise Addition

Remember that **uncorrelated** noise quantities need to be power-summed (i.e., $I^2 = I_1^2 + I_2^2$)!

We can then sum the output noise current $\overline{I_{\mathrm{n}}}$ as

$$\overline{I_{\mathrm{n}}^2} = \overline{I_{\mathrm{n2}}^2} + \overline{I_{\mathrm{n4}}^2} + g_{\mathrm{m34}}^2 \frac{1}{g_{\mathrm{m34}}^2} \left(\overline{I_{\mathrm{n1}}^2} + \overline{I_{\mathrm{n3}}^2} \right) = 2 \left(\overline{I_{\mathrm{n12}}^2} + \overline{I_{\mathrm{n34}}^2} \right).$$

As a next step, let us rewrite the OTA transfer function $A(s)$ (see Equation 15) by getting rid of the pole-zero doublet as a simplyfing assumption to get

$$A'(s) = \frac{g_{\mathrm{m12}}}{g_{\mathrm{ds2}} + g_{\mathrm{ds4}} + sC_{\mathrm{load}}}. \tag{18}$$

Inspecting Equation 18 we can interpret the OTA transfer function as a transconductor $g_{\mathrm{m}12}$ driving a load of $Y_{\mathrm{load}} = g_{\mathrm{ds}2} + g_{\mathrm{ds}4} + sC_{\mathrm{load}}$. We can thus redraw Figure 19 in the following way, injecting the previously calculated noise current into the output node. The result is shown in Figure 24.

Source: Article Notebook

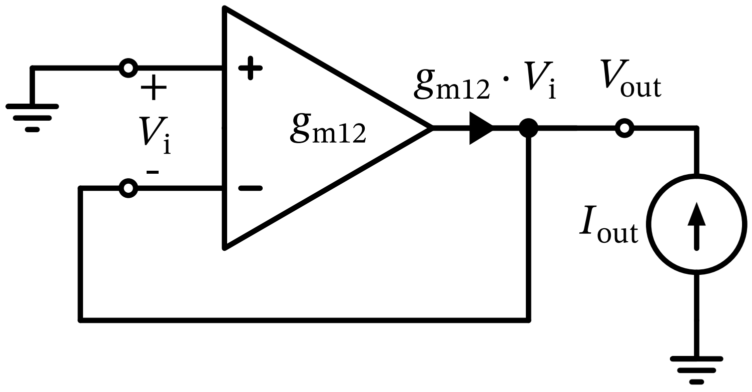


Figure 23: Output impedance calculation of a voltage buffer.

Source: Article Notebook

Output Impedence of the Voltage Buffer

First we short the input terminal to ground and then we connect a current source I_{out} at the output terminal, see Figure 23. Since we can neglect the gate leakage current into the inverting input terminal of the OTA, KCL at the output node is simply:

$$I_{\mathrm{out}} + g_{\mathrm{m}12}\left(-V_{\mathrm{out}}\right) = 0$$

Thus, the output impedance is easily calculated.

$$Z_{\mathrm{out}} = \frac{V_{\mathrm{out}}}{I_{\mathrm{out}}} = \frac{V_{\mathrm{out}}}{g_{\mathrm{m}12} V_{\mathrm{out}}} = \frac{1}{g_{\mathrm{m}12}}$$

Source: Article Notebook

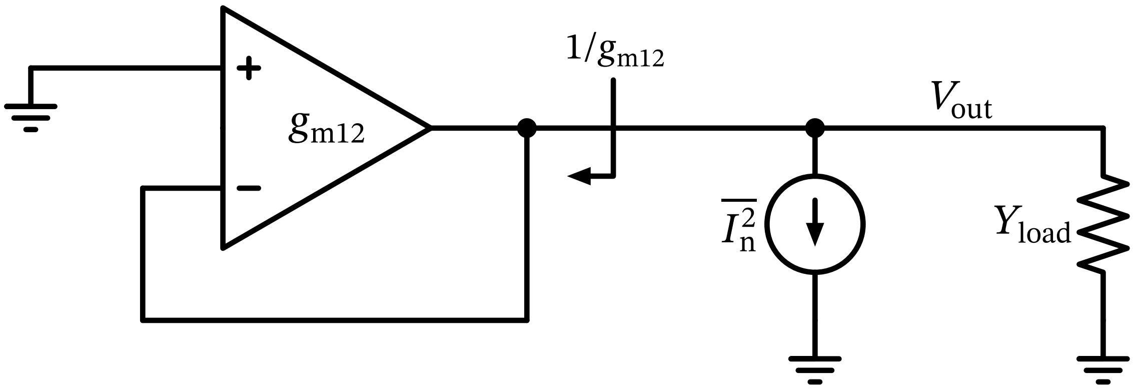


Figure 24: A voltage buffer redrawn for noise analysis.

Source: Article Notebook

We see that the feedback around the transconductor g_{m12} creates an impedance of $1/g_{m12}$. We can now calculate the effective load conductance of

$$Y'_{\mathrm{load}} = g_{\mathrm{ds2}} + g_{\mathrm{ds4}} + sC_{\mathrm{load}} + g_{\mathrm{m12}} \approx g_{\mathrm{m12}} + sC_{\mathrm{load}}. \tag{19}$$

The output noise voltage is then (using Equation 1)

$$\overline{V^2_{\mathrm{n,out}}}(f) = \frac{\overline{I^2_{\mathrm{n}}}}{|Y'_{\mathrm{load}}|^2} = \frac{\overline{I^2_{\mathrm{n}}}}{g^2_{\mathrm{m12}} + (2\pi f C_{\mathrm{load}})2^2} = \frac{8kT(\gamma_{12} g_{\mathrm{m12}} + \gamma_{34} g_{\mathrm{m34}})}{g^2_{\mathrm{m12}} + (2\pi f C_{\mathrm{load}})^2}.$$

We can use the identity Equation 7 to calculate the rms output noise to

$$V^2_{\mathrm{n,out,rms}} = \int_0^\infty \overline{V^2_{\mathrm{n,out}}}(f)df = \frac{kT}{C_{\mathrm{load}}}\left(2\gamma_{12} + 2\gamma_{34}\frac{g_{\mathrm{m34}}}{g_{\mathrm{m12}}}\right). \tag{20}$$

Inspecting Equation 20 we can see that the integrated output noise is the kT/C noise of the output load capacitor, enhanced by the γ_{12} of the input differential pair, plus a (smaller) contribution of the current mirror load $M_{3,4}$. Intuitevly, this result makes sense.

Exercise: Derivation of 5T-OTA Performance

Please take your time and carefully go through the explanations and derivations for the 5-transistor-OTA in Section 7.2 and Section 7.3. Try to do the calculations yourself; if you get stuck, review the previous chapters.

7.4 5T-OTA Sizing

Outfitted with the governing equations derived in Section 7.3 we can now size the MOSFETs in the OTA, we remember that we have to size $M_{1,2}$ and $M_{3,4}$ equally.

First, we need to select a proper $g_\mathrm{m}/I_\mathrm{D}$ for the MOSFET. Remembering Section 3 we see that for the input differential pair we should go for a large g_m, thus we select a $g_\mathrm{m}/I_\mathrm{D} = 10$. As g_ds of M_2 could limit the dc gain (Equation 16) we go with a rather long $L = 5\,\mu$m. For current sources a small $g_\mathrm{m}/I_\mathrm{D}$ is a good idea, so we start with $g_\mathrm{m}/I_\mathrm{D} = 5$ (because we can not go too low because of $V_\mathrm{ds,sat}$) and also an $L = 5\,\mu$m. The $g_\mathrm{m}/I_\mathrm{D}$ is also useful to estimate the required drain-source voltage to keep a MOSFET in saturation (i.e., keep the g_ds small) with this approximate relationship:

$$V_\mathrm{ds,sat} = \frac{2}{g_\mathrm{m}/I_\mathrm{D}} \tag{21}$$

Exercise: 5T-OTA Sizing

Please size the 5T-OTA according to the previous $g_\mathrm{m}/I_\mathrm{D}$ and L suggestions. Please calculate the W of M_{1-6} and the total supply current. Please check wether gain error, total output noise, and turn-on settling is met with the calculated devices sizes and bias currents.

The sizing procedure and its calculation are best performed in a Jupyter notebook, as we can easily look up the exact data from the pre-computed tables:

Solution: 5T-OTA Sizing

8 Sizing for Basic 5T-OTA

```python
# Read table data
from pygmid import Lookup as lk
import numpy as np
lv_nmos = lk('sg13_lv_nmos.mat')
lv_pmos = lk('sg13_lv_pmos.mat')
# List of parameters: VGS, VDS, VSB, L, W, NFING, ID, VT, GM, GMB, GDS, CGG, CGB, CGD, C
# If not specified, minimum L, VDS=max(vgs)/2=0.9 and VSB=0 are used
# Define the given parameters as taken from the specification table or inital guesses
c_load = 50e-15
gm_id_m12 = 10
gm_id_m34 = 5
gm_id_m56 = 5
l_12 = 5
l_34 = 5
l_56 = 5
f_bw = 10e6
i_total_limit = 10e-6
i_bias_in = 20e-6
output_voltage = 1.3
vin_min = 0.7
vin_max = 0.9
vdd_min = 1.45
vdd_max = 1.55
```

```python
# We get the required gm of M1/2 from the bandwidth requirement
# We add a factor of 3 to allow for PVT variation plus additional MOSFET parasitic loadi
gm_m12 = f_bw * 3 * 4*np.pi*c_load
print('gm12 =', gm_m12/1e-3, 'mS')
```

```
gm12 = 0.01884955592153876 mS
```

```python
# Since we know gm12 and the gmid we can calculate the bias current
id_m12 = gm_m12 / gm_id_m12
i_total = 2*id_m12
print('i_total (exact) =', i_total/1e-6, 'µA')
# we round to 0.5µA bias currents
i_total = max(round(i_total / 1e-6 * 2) / 2 * 1e-6, 0.5e-6)
id_m12 = i_total/2

print('i_total (rounded) =', i_total/1e-6, 'µA')
if i_total < i_total_limit:
    print('[info] power consumption target is met!')
else:
    print('[info] power consumption target is NOT met!')
```

```
i_total (exact) = 3.7699111843077517 µA
i_total (rounded) = 4.0 µA
[info] power consumption target is met!
```

```python
# We calculate the dc gain
gm_gds_m12 = lv_nmos.lookup('GM_GDS', GM_ID=gm_id_m12, L=l_12, VDS=0.75, VSB=0)
gm_gds_m34 = lv_pmos.lookup('GM_GDS', GM_ID=gm_id_m34, L=l_34, VDS=0.75, VSB=0)

gds_m12 = gm_m12 / gm_gds_m12
gm_m34 = gm_id_m34 * i_total/2
gds_m34 = gm_m34 / gm_gds_m34

a0 = gm_m12 / (gds_m12 + gds_m34)
print('a0 =', 20*np.log10(a0), 'dB')
```

```
a0 = 34.78458740352468 dB
```

```python
# We calculate the MOSFET capacitance which adds to Cload, to see the impact on the BW
gm_cgs_m12 = lv_nmos.lookup('GM_CGS', GM_ID=gm_id_m12, L=l_12, VDS=0.75, VSB=0)
gm_cdd_m12 = lv_nmos.lookup('GM_CDD', GM_ID=gm_id_m12, L=l_12, VDS=0.75, VSB=0)
gm_cdd_m34 = lv_pmos.lookup('GM_CDD', GM_ID=gm_id_m34, L=l_34, VDS=0.75, VSB=0)

c_load_parasitic = abs(gm_m12/gm_cgs_m12) + abs(gm_m12/gm_cdd_m12) + abs(gm_m34/gm_cdd_m
print('additional load capacitance =', c_load_parasitic/1e-15, 'fF')

f_bw = gm_m12 / (4*np.pi * (c_load + c_load_parasitic))
print('-3dB bandwidth incl. parasitics =', f_bw/1e6, 'MHz')
```

```
additional load capacitance = 54.92854674560976 fF
-3dB bandwidth incl. parasitics = 14.295442437000684 MHz
```

```python
# We can now look up the VGS of the MOSFET
vgs_m12 = lv_nmos.look_upVGS(GM_ID=gm_id_m12, L=l_12, VDS=0.75, VSB=0.0)
vgs_m34 = lv_pmos.look_upVGS(GM_ID=gm_id_m34, L=l_34, VDS=0.75, VSB=0.0)
vgs_m56 = lv_nmos.look_upVGS(GM_ID=gm_id_m56, L=l_56, VDS=0.75, VSB=0.0)

print('vgs_12 =', vgs_m12, 'V')
print('vgs_34 =', vgs_m34, 'V')
print('vgs_56 =', vgs_m56, 'V')
```

```
vgs_12 = 0.36710119710062455 V
vgs_34 = 0.7287454603526495 V
vgs_56 = 0.5912200307058603 V
```

```python
# Calculate settling time due to slewing with the calculated bias current
t_slew = (c_load + c_load_parasitic) * output_voltage / i_total
print('slewing time =', t_slew/1e-6, 'µs')
t_settle = 5/(2*np.pi*f_bw)
print('settling time =', t_settle/1e-6, 'µs')
```

```
slewing time = 0.034101777692323185 µs
settling time = 0.055666322953376014 µs
```

```
# Calculate voltage gain error
gain_error = a0 / (1 + a0)
print('voltage gain error =', (gain_error-1)*100, '%')
```

```
voltage gain error = -1.7902967715882068 %
```

```
# Calculate total rms output noise
sth_m12 = lv_nmos.lookup('STH_GM', VGS=vgs_m12, L=l_12, VDS=0.75, VSB=0) * gm_m12
gamma_m12 = sth_m12/(4*1.38e-23*300*gm_m12)

sth_m34 = lv_pmos.lookup('STH_GM', VGS=vgs_m34, L=l_34, VDS=0.75, VSB=0) * gm_m34
gamma_m34 = sth_m34/(4*1.38e-23*300*gm_m34)

output_noise_rms = 1.38e-23*300 / (c_load + c_load_parasitic) * (2*gamma_m12 + 2*gamma_m
print('output noise (rms) =', output_noise_rms/1e-6, 'µV')
```

```
output noise (rms) = 0.12543377043178017 µV
```

```
# Calculate all widths
id_w_m12 = lv_nmos.lookup('ID_W', GM_ID=gm_id_m12, L=l_12, VDS=vgs_m12, VSB=0)
w_12 = id_m12 / id_w_m12
w_12_round = max(round(w_12*2)/2, 0.5)
print('M1/2 W =', w_12, 'um, rounded W =', w_12_round, 'um')

id_m34 = id_m12
id_w_m34 = lv_pmos.lookup('ID_W', GM_ID=gm_id_m34, L=l_34, VDS=vgs_m34, VSB=0)
w_34 = id_m34 / id_w_m34
w_34_round = max(round(w_34*2)/2, 0.5)
print('M3/4 W =', w_34, 'um, rounded W =', w_34_round, 'um')

id_w_m5 = lv_nmos.lookup('ID_W', GM_ID=gm_id_m56, L=l_56, VDS=vgs_m56, VSB=0)
w_5 = i_total / id_w_m5
w_5_round = max(round(w_5*2)/2, 0.5)
print('M5 W =', w_5, 'um, rounded W =', w_5_round, 'um')
w_6 = w_5_round * i_bias_in / i_total
print('M6 W =', w_6, 'um')
```

```
M1/2 W = 1.7713641972645868 um, rounded W = 2.0 um
M3/4 W = 1.641014110777885 um, rounded W = 1.5 um
M5 W = 0.7351148286825442 um, rounded W = 0.5 um
M6 W = 2.5000000000000004 um
```

```python
# Print out final design values
print('5T-OTA dimensioning:')
print('--------------------')
print('M1/2 W=', w_12_round, ', L=', l_12)
print('M3/4 W=', w_34_round, ', L=', l_34)
print('M5   W=', w_5_round, ', L=', l_56)
print('M6   W=', w_6, ', L=', l_56)
print()
print('5T-OTA performance summary:')
print('---------------------------')
print('supply current =', i_total/1e-6, 'µA')
print('output noise =', output_noise_rms/1e-6, 'µVrms')
print('voltage gain error =', (gain_error-1)*100, '%')
print('-3dB bandwidth incl. parasitics =', f_bw/1e6, 'MHz')
print('turn-on time (slewing+settling) =', (t_slew+t_settle)/1e-6, 'µs')
print()
print('5T-OTA bias point check:')
print('------------------------')
print('headroom M1 =', vdd_min-vgs_m34+vgs_m12-vin_max, 'V')
print('headroom M4 =', vdd_min-vin_max, 'V')
print('headroom M5 =', vin_min-vgs_m12, 'V')
```

```
5T-OTA dimensioning:
--------------------
M1/2 W= 2.0 , L= 5
M3/4 W= 1.5 , L= 5
M5   W= 0.5 , L= 5
M6   W= 2.5000000000000004 , L= 5

5T-OTA performance summary:
---------------------------
supply current = 4.0 µA
output noise = 0.12543377043178017 µVrms
voltage gain error = -1.7902967715882068 %
-3dB bandwidth incl. parasitics = 14.295442437000684 MHz
turn-on time (slewing+settling) = 0.08976810064569919 µs

5T-OTA bias point check:
------------------------
headroom M1 = 0.188355736747975 V
headroom M4 = 0.5499999999999999 V
headroom M5 = 0.3328988028993754 V
```

Source: Sizing for Basic 5T-OTA

8.1 5T-OTA Simulation

With the initial sizing of the MOSFETs of the 5T-OTA done, we can design the 5T-OTA circuit and setup a simulation testbench to check the performance parameters. Since this is the first time we draw a more complex schematic, and use a hierarchical design, we should note that drawing a schematic is an art, and there exists a set of rules and recommendations how to name pins, how to use annotations, and so on. Please read Section 23 before you start into your design work.

Exercise: 5T-OTA Design and Testbench

Please design the circuit of the 5T-OTA. Put the OTA circuit in a separate schematic, create a symbol for it, and use this symbol in a testbench you create in Xschem for this 5T-OTA used as a voltage buffer as schown in Figure 19. Use typical conditions for the simulation, and check how well the specification in Table 2 is met, and how well the derivations in Section 7.2 and Section 7.3 fit to the simulation results.
If you get stuck, you can find the testbench and 5T-OTA schematic here (for the small-signal analysis) and here (for the large-signal settling simulation).

8.2 5T-OTA Simulation versus PVT

As you have seen in Section 8.1 running simulations by hand is tedious. When we want to check the overall performance, we have to run many simulations over various conditions:

1. The supply voltage of the circuit has tolerances, and thus we need to check the performance against this variation.
2. The temperature at which the circuit is operated is likely changing. Also the performance against this has to be verified.
3. When manufacturing the wafers random variations in various process parameters lead to changed parameters of the integrated circuit components. In order to check for this effect, wafer foundries provide model files which shall cover these manufacturing excursions. Simplified, this leads to a slower or faster MOSFET, and usually NMOS and PMOS are not correlated, so we have the process corners **SS**, **SF**, **TT**, **FS**, and **FF**. So far, we have only used the **TT** models in our simulations.

The variations listed in the previous list are abbreviated as **PVT** (process, voltage, temperature) variations. In order to finalize a circuit all combinations of these (plus the variations in operating conditions like input voltage) have to be simulated. As you can imagine, this leads to a huge number of simulations, and simulation results which have to be evaluated for pass/fail.

There are two options how to tackle this efficiently:

1. As an experienced designer you have a very solid understanding of the circuit, plus based on the analytic equations you can identify which combination of operating

conditions will lead to a worst case performance. Thus, you can drastically reduce the number of corners to simulate, and you run them by hand.
2. You are using a framework which highly automates this task of running a plethora of different simulations and evaluating the outcome. These frameworks are called simulation runners.

Luckily, there are open-source versions of simulation runners available, and we will use CACE in this lecture. CACE is written in Python and allows to setup a datasheet in YAML which defines the simulation problem and the performance parameters to evaluate against which limits. The resulting simulations are then run in parallel and the simulation data is evaluated and summarized in various forms.

There is a CACE setup available for our 5T-OTA. The datasheet describes the operating conditions and the simulations tasks. For each simulation a testbench template is needed, this one is used for ac simulations, this one is used for noise simulation, and this one is used for transient simulation.

After a successul run, a documentation is automatically generated. The result of a full run of this OTA design is presented here:

Note 1: CACE Summary for 5T-OTA

9 CACE Summary for ota-5t

netlist source: schematic

Parameter	Tool	Result	Min Limit	Min Value	Typ Target	Typ Value	Max Limit	Max Value	Status
Output voltage ratio	ngspice	gain	0.97 V/V	0.987 V/V	any	1.000 V/V	1.03 V/V	1.006 V/V	Pass
Bandwidth	ngspice	bw	10e6 Hz	15551000.000 Hz	any	26912100.000 Hz	any	34051700.000 Hz	Pass
Output noise	ngspice	noise	any	0.308 mV	any	0.371 mV	1 mV	0.455 mV	Pass
Settling time	ngspice	tsettle	any	0.135 us	any	0.142 us	10 us	0.155 us	Pass

9.1 Plots

9.2 gain_vs_temp

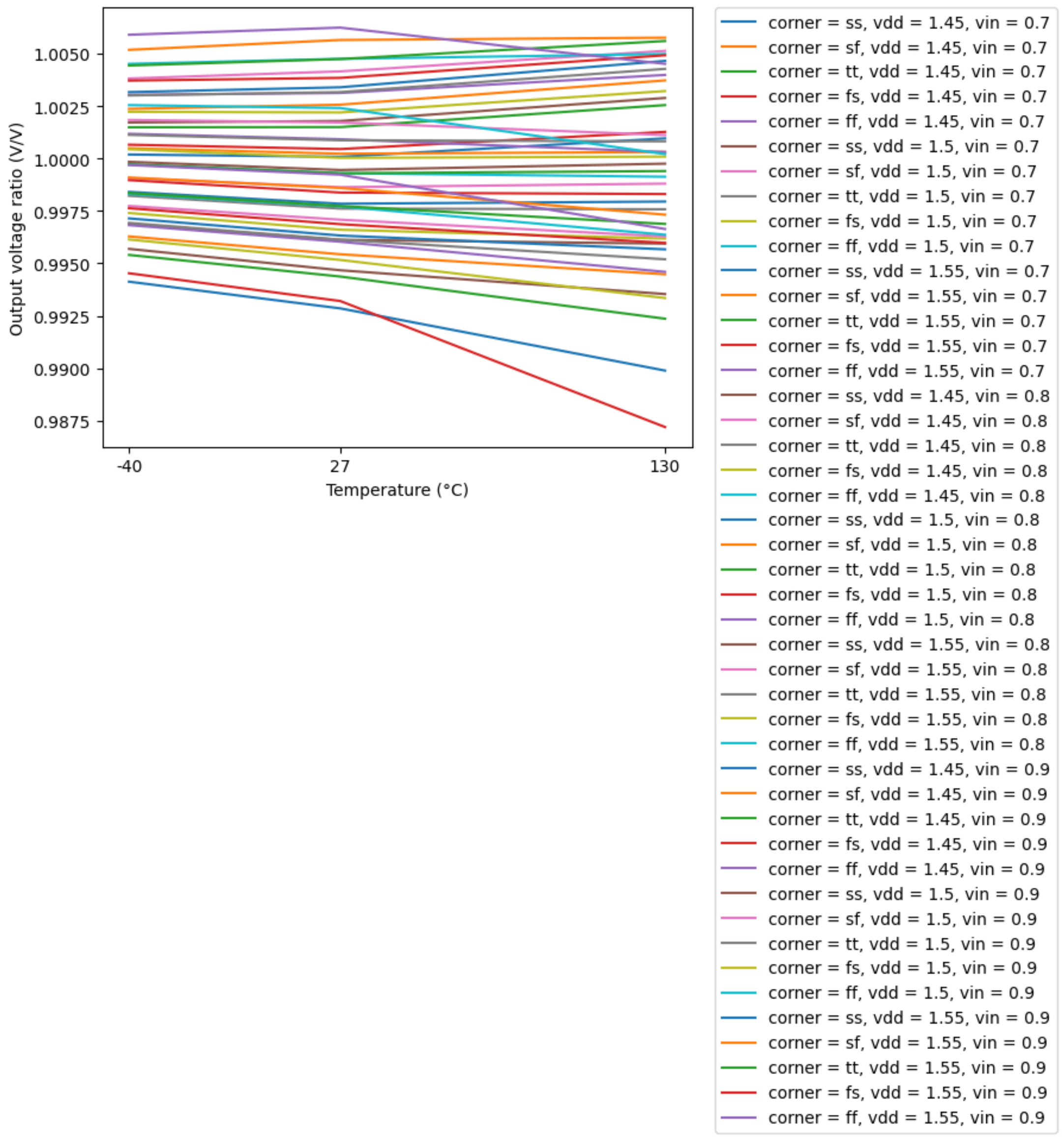


Figure 25: gain_vs_temp

9.3 gain_vs_vin

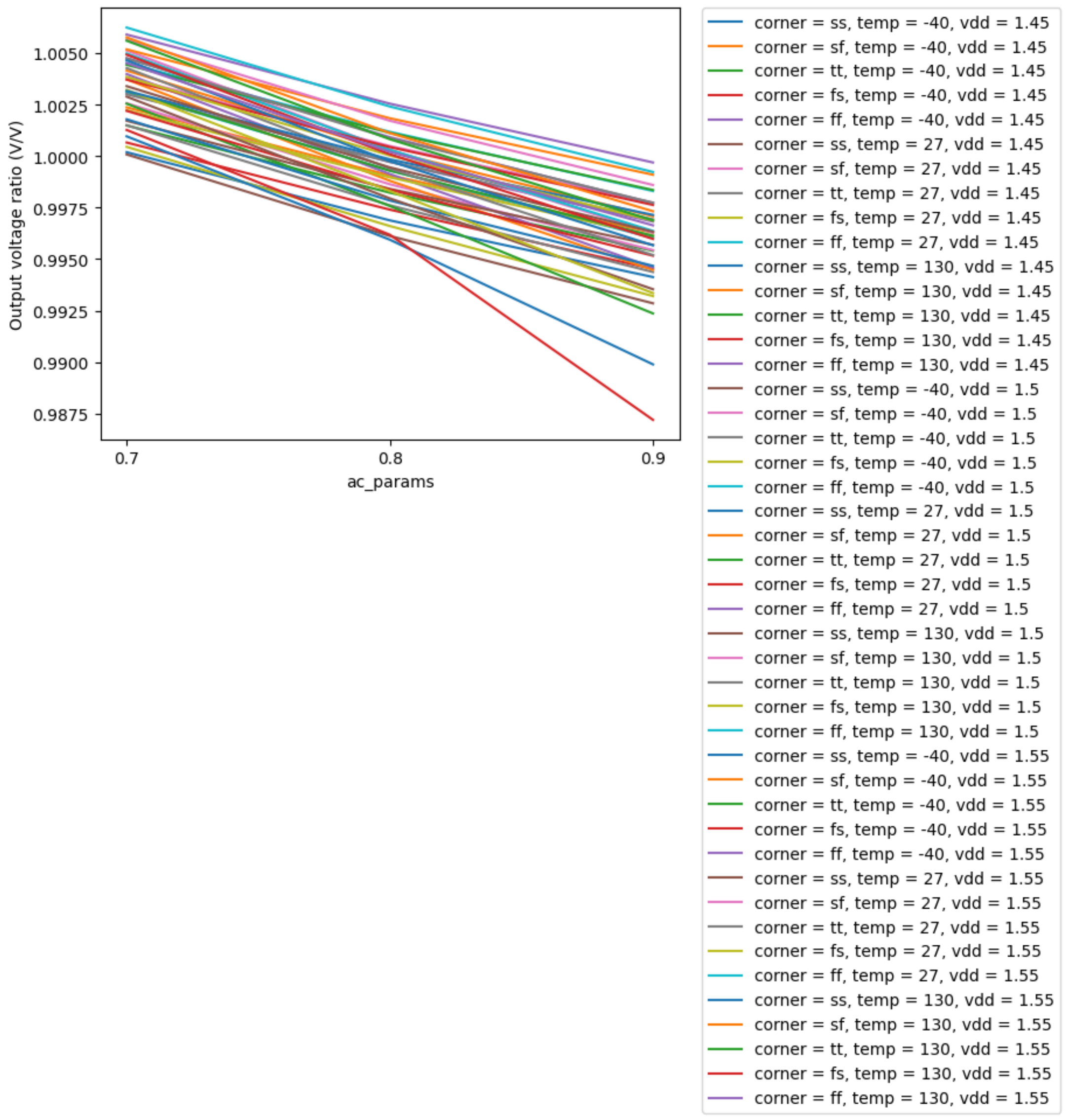


Figure 26: gain_vs_vin

9.4 gain_vs_vdd

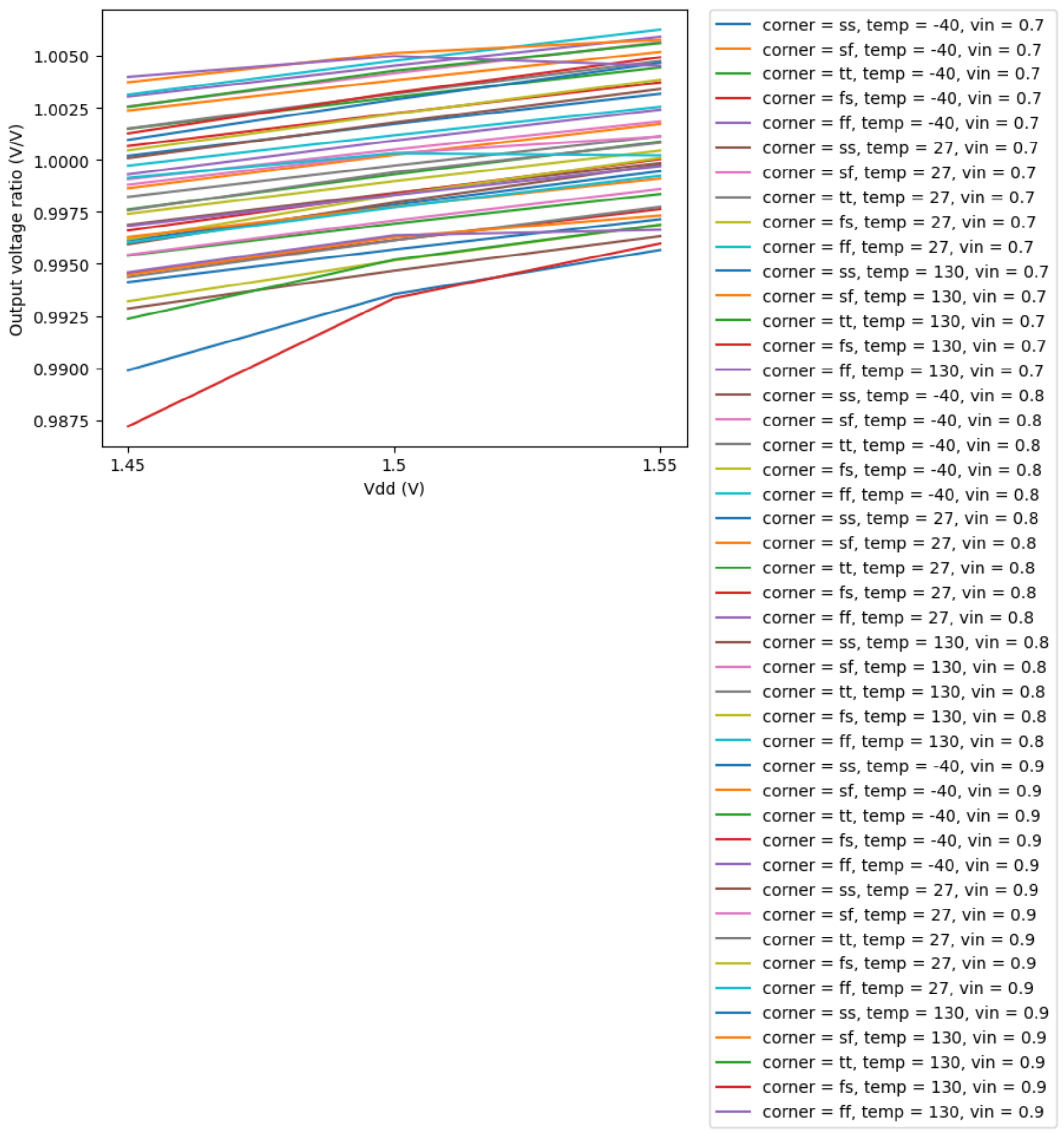


Figure 27: gain_vs_vdd

9.5 gain_vs_corner

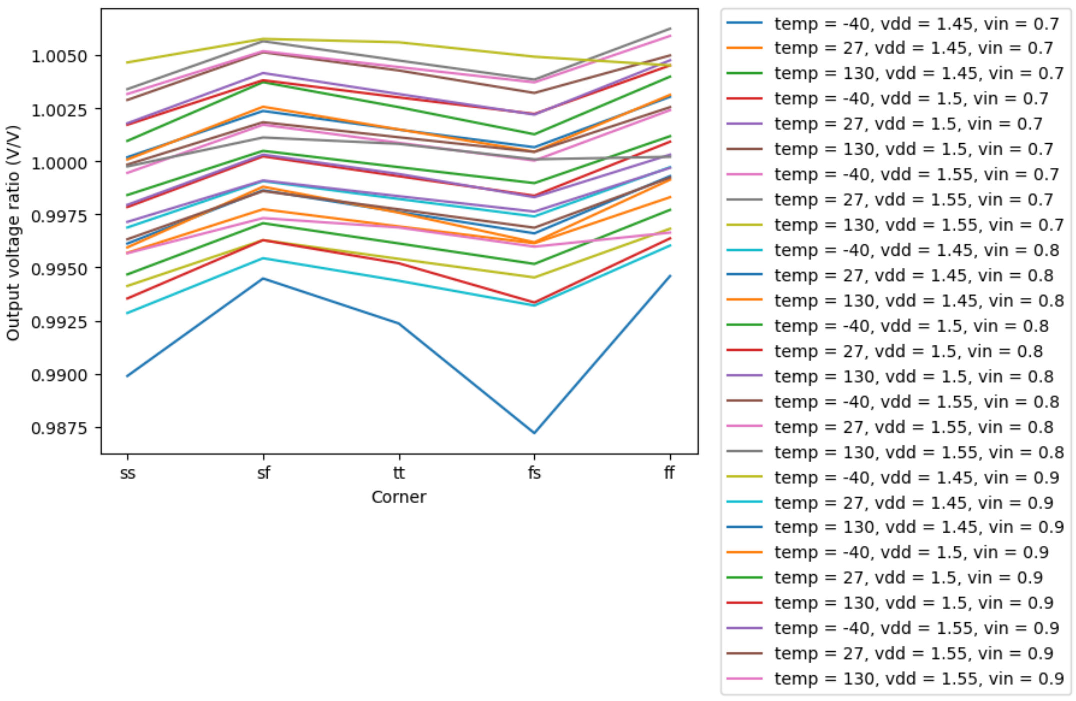


Figure 28: gain_vs_corner

9.6 bw_vs_temp

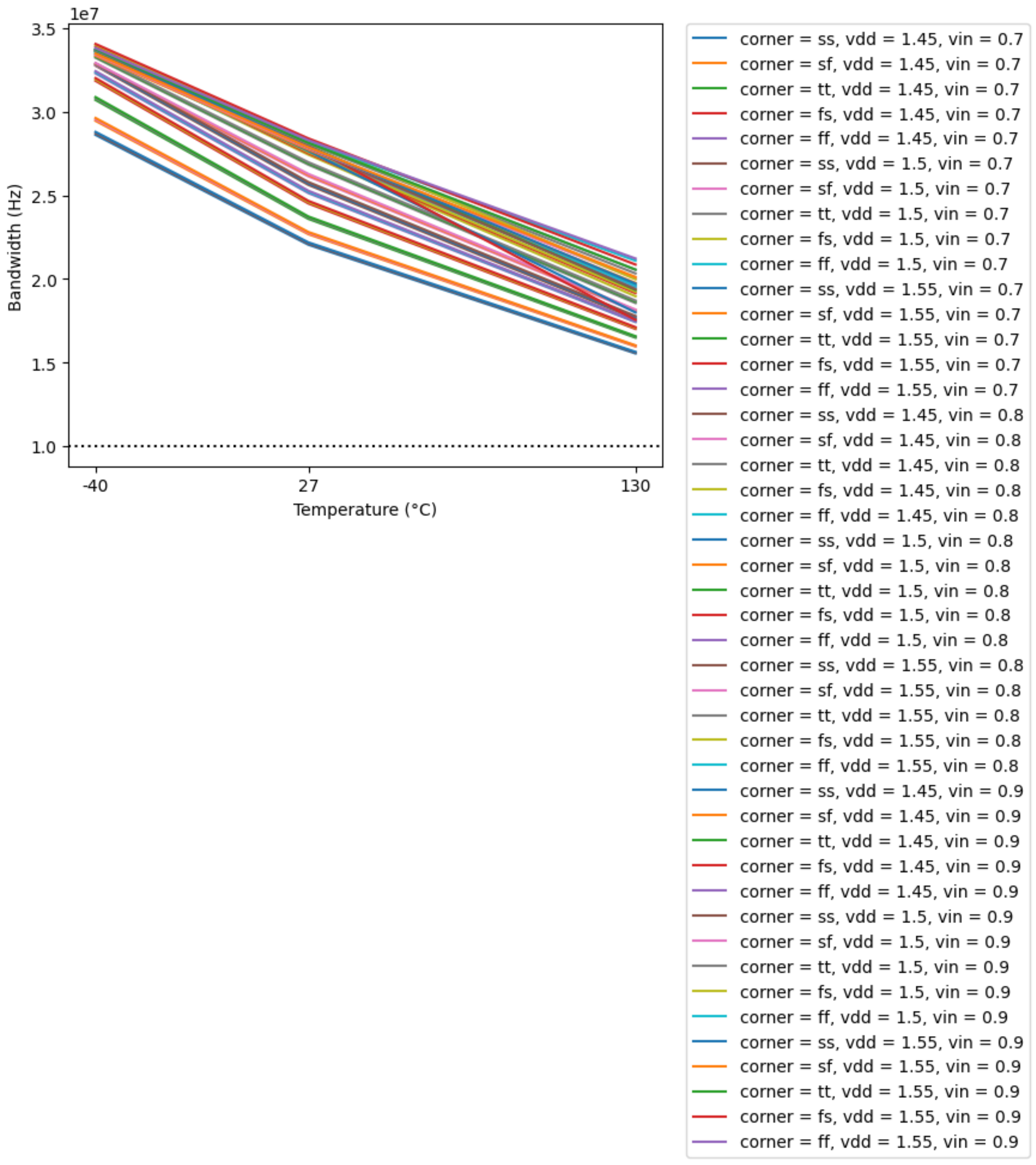


Figure 29: bw_vs_temp

9.7 bw_vs_vin

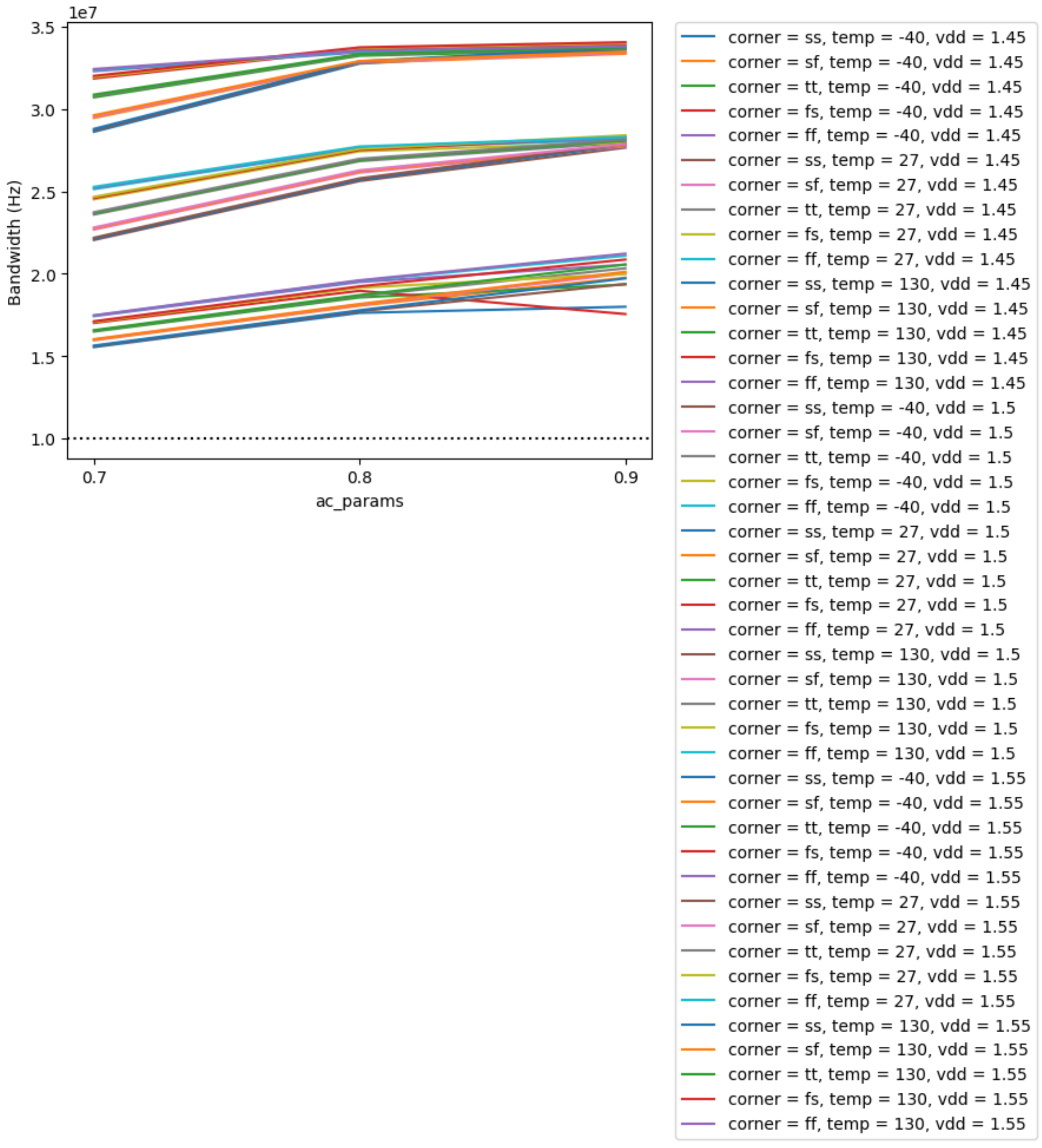


Figure 30: bw_vs_vin

9.8 bw_vs_vdd

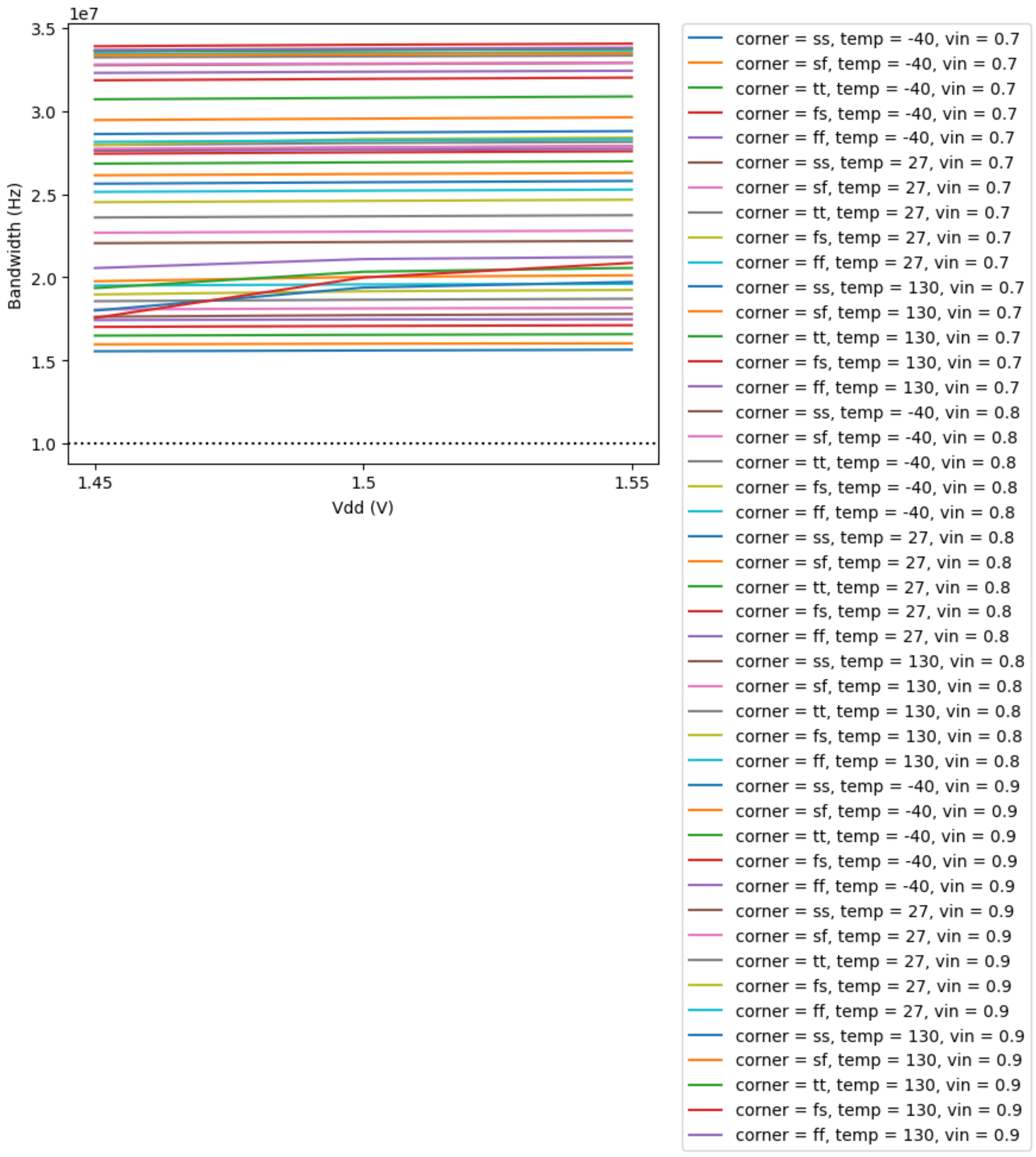


Figure 31: bw_vs_vdd

9.9 bw_vs_corner

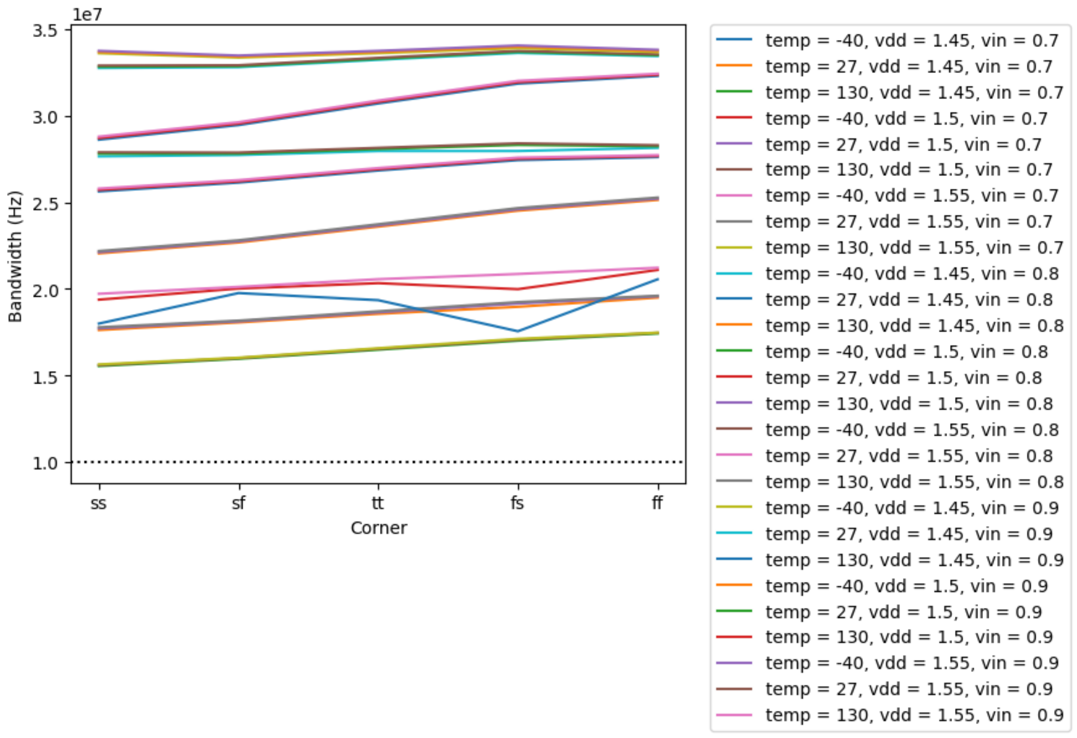


Figure 32: bw_vs_corner

9.10 noise_vs_temp

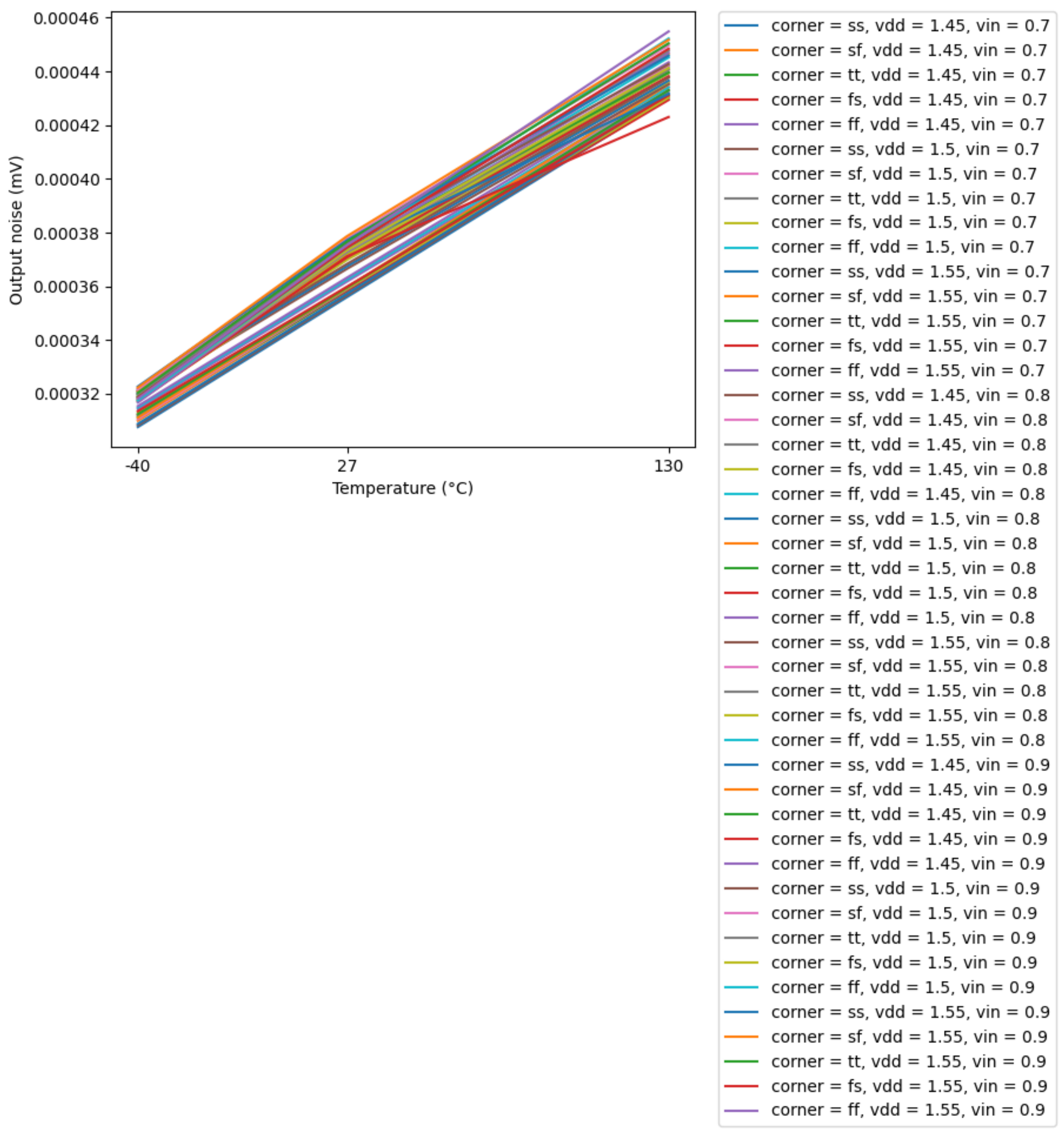


Figure 33: noise_vs_temp

9.11 noise_vs_vin

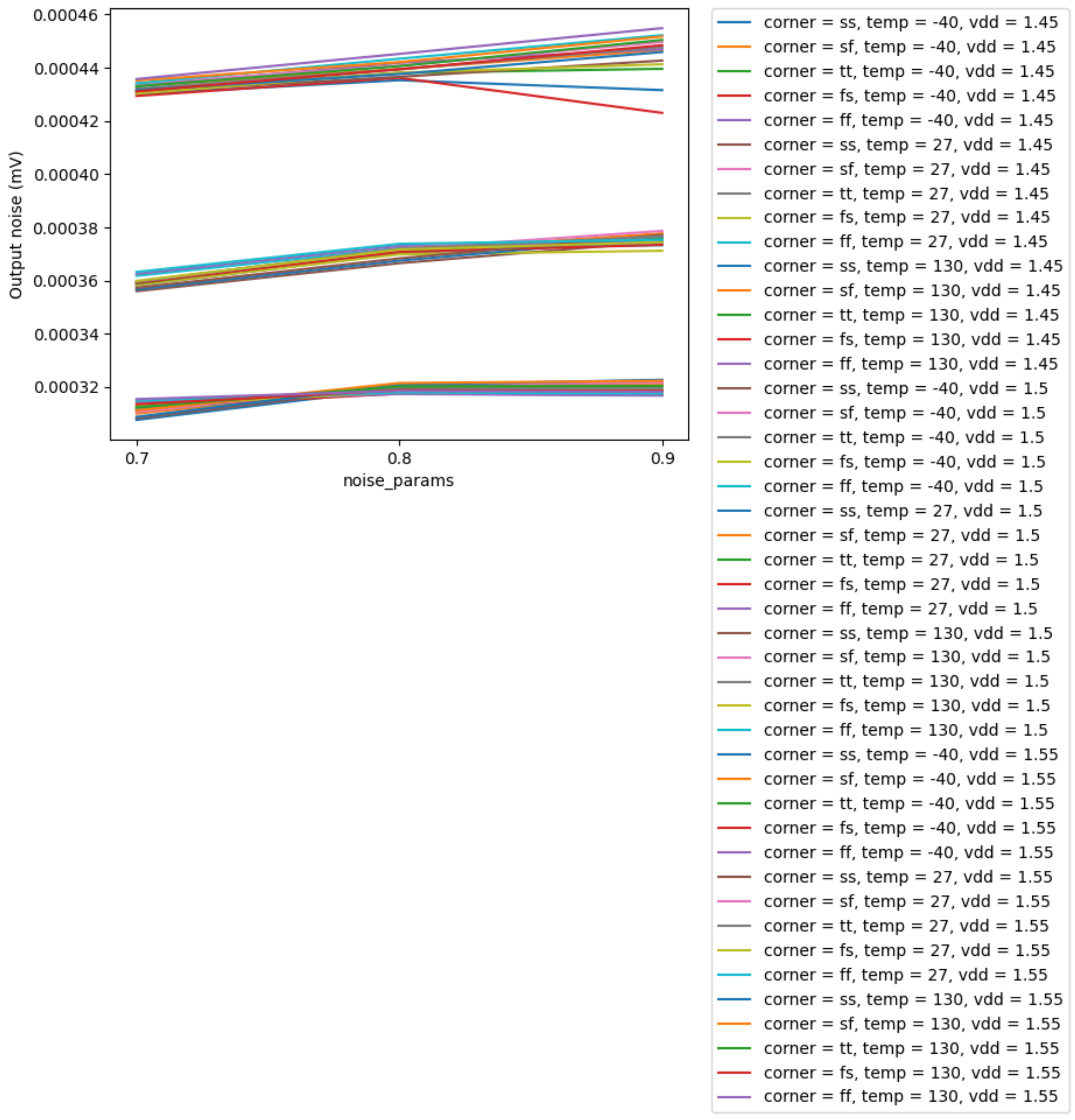


Figure 34: noise_vs_vin

9.12 noise_vs_vdd

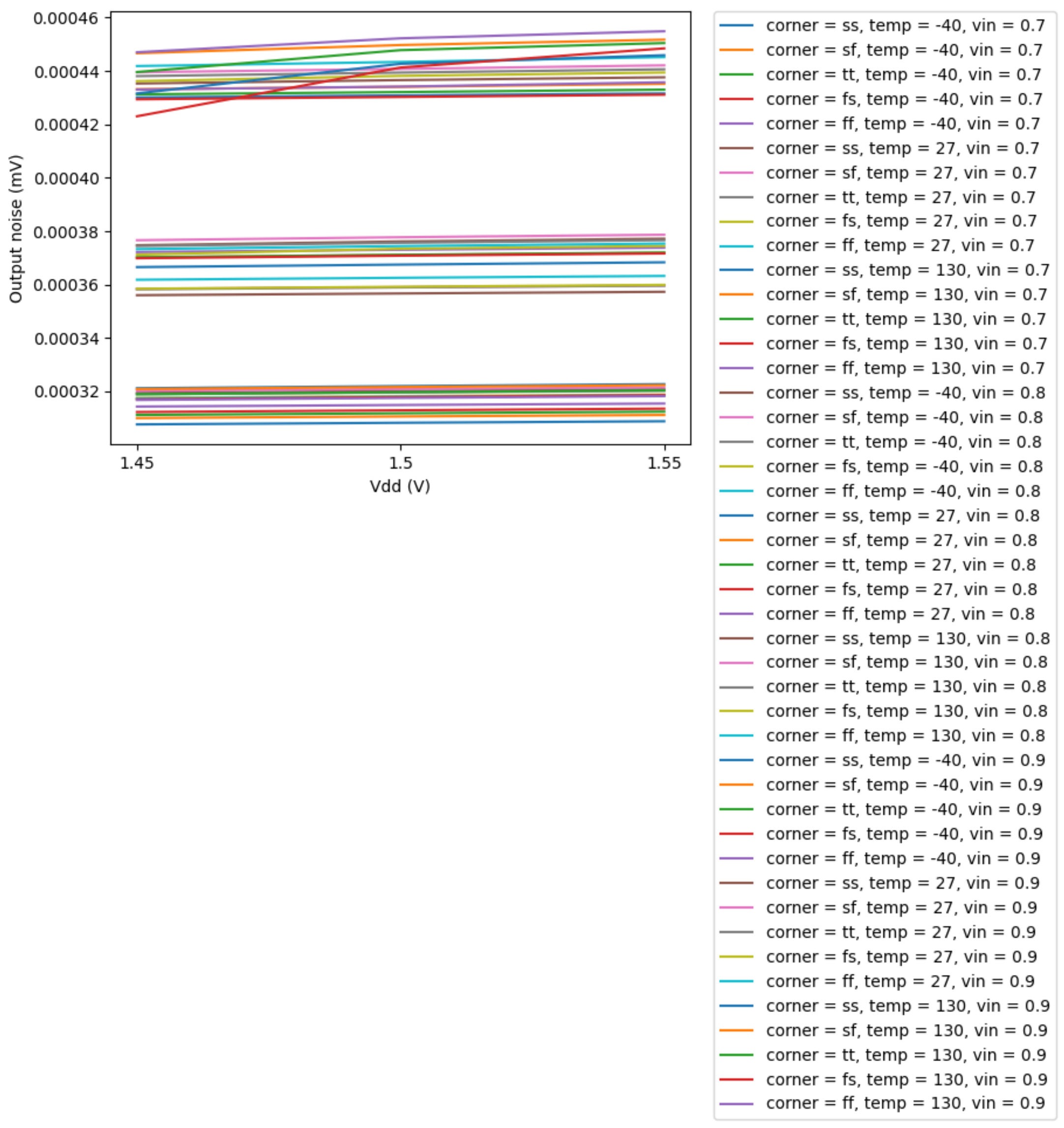


Figure 35: noise_vs_vdd

9.13 noise_vs_corner

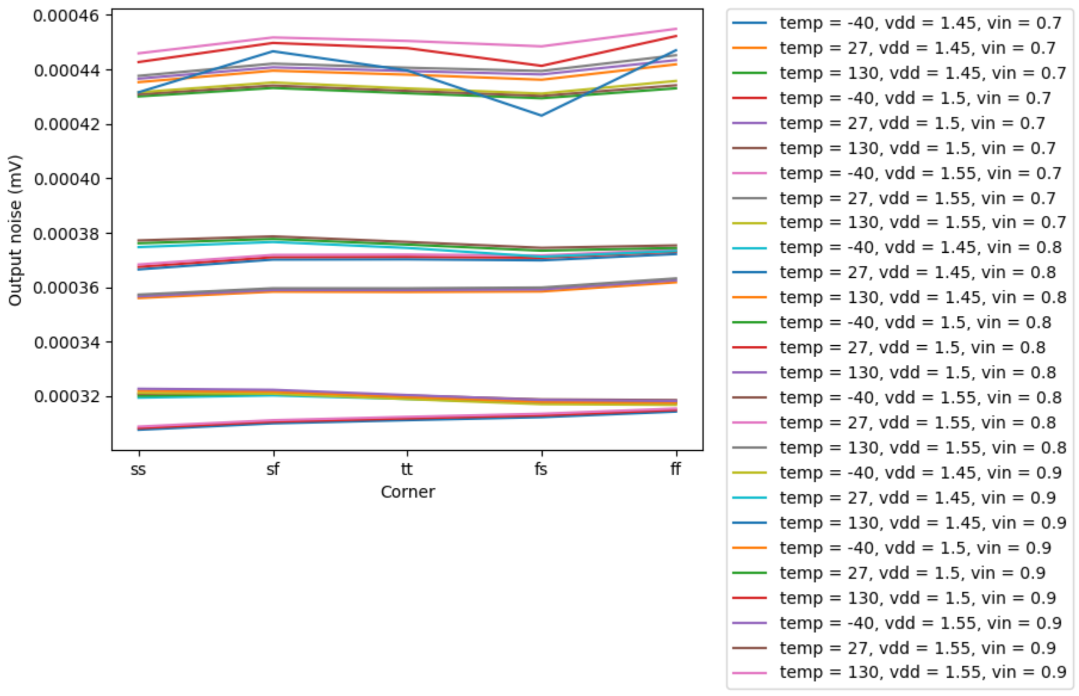


Figure 36: noise_vs_corner

9.14 settling_vs_temp

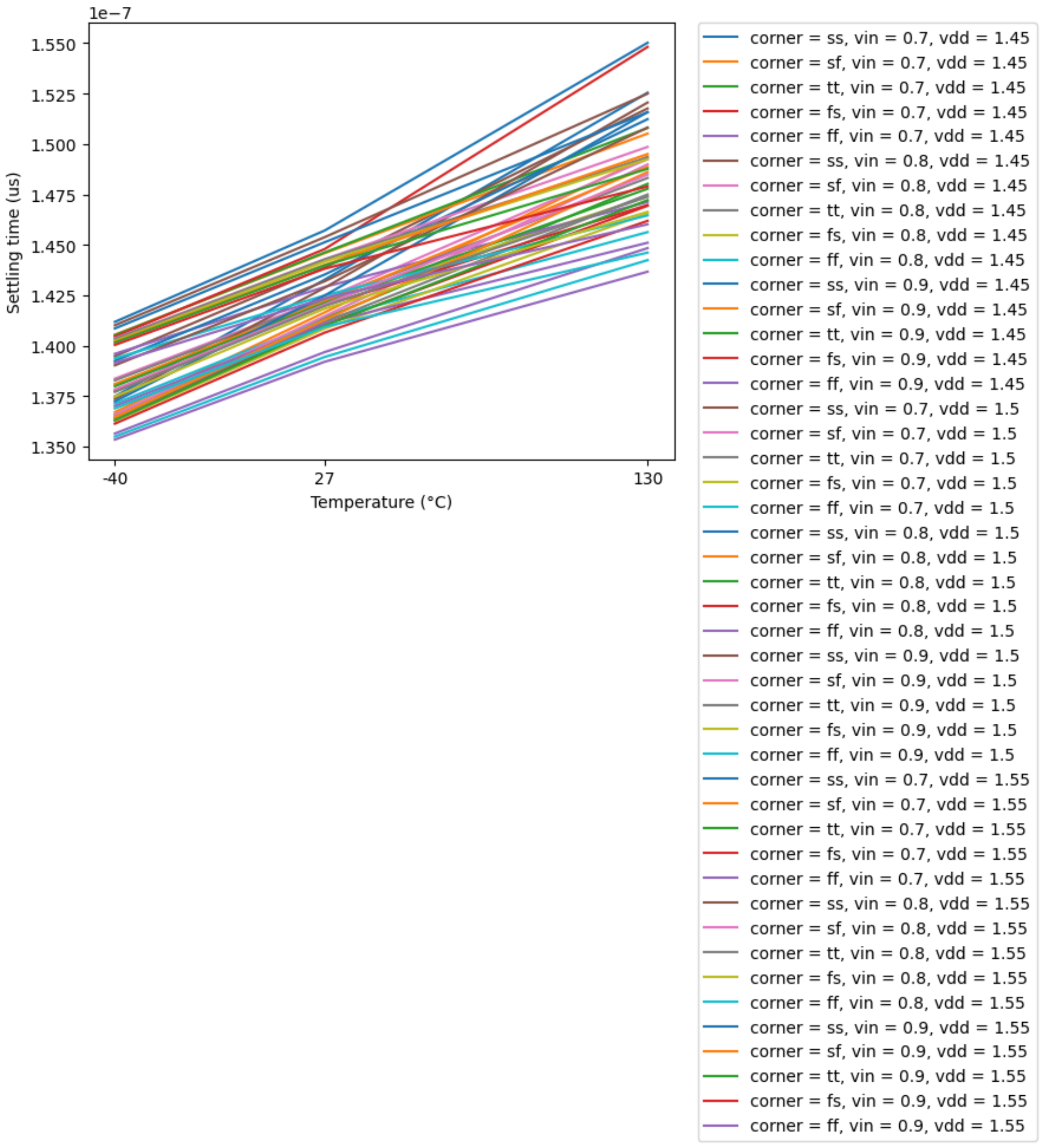


Figure 37: settling_vs_temp

9.15 settling_vs_vin

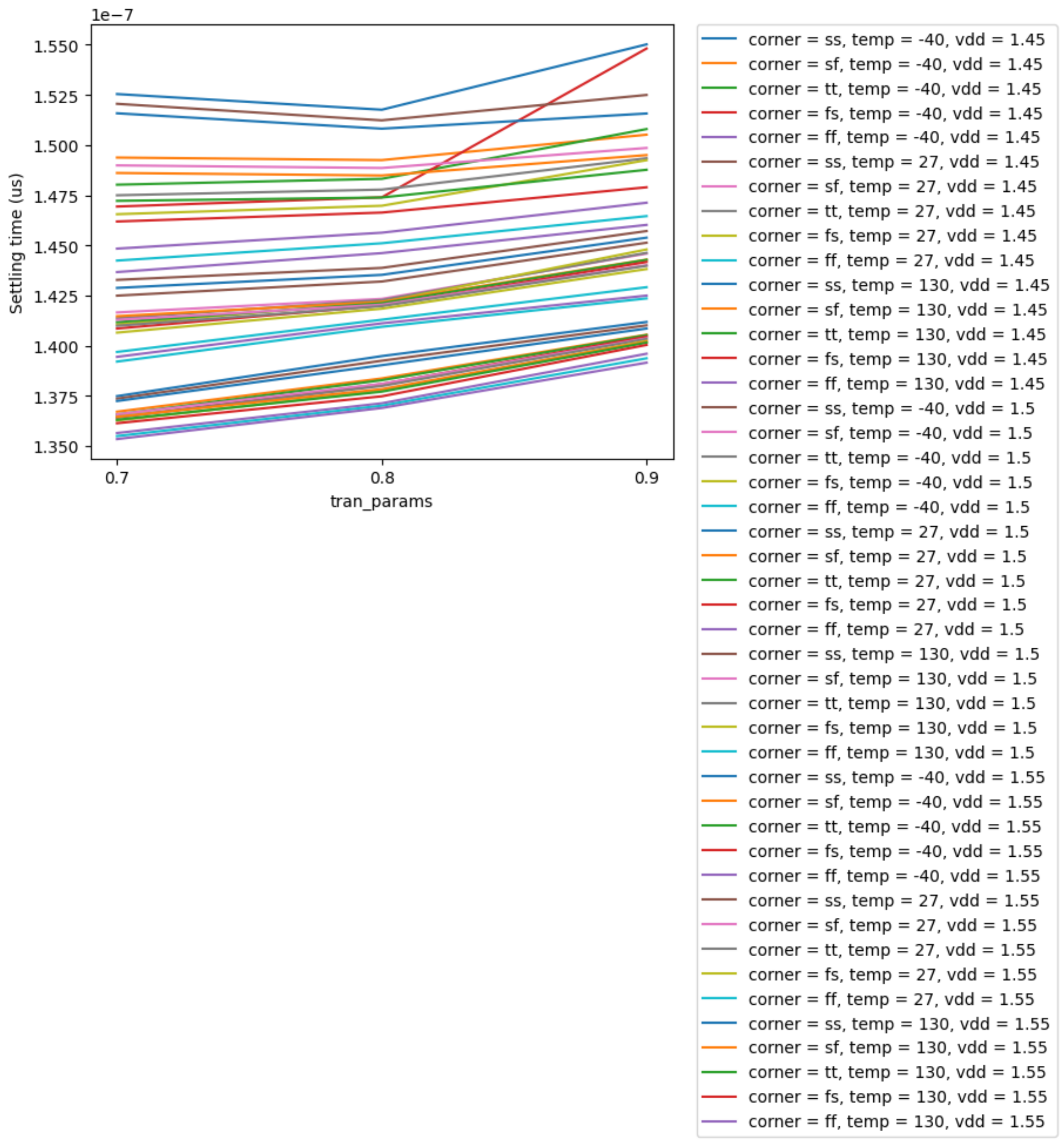


Figure 38: settling_vs_vin

9.16 settling_vs_vdd

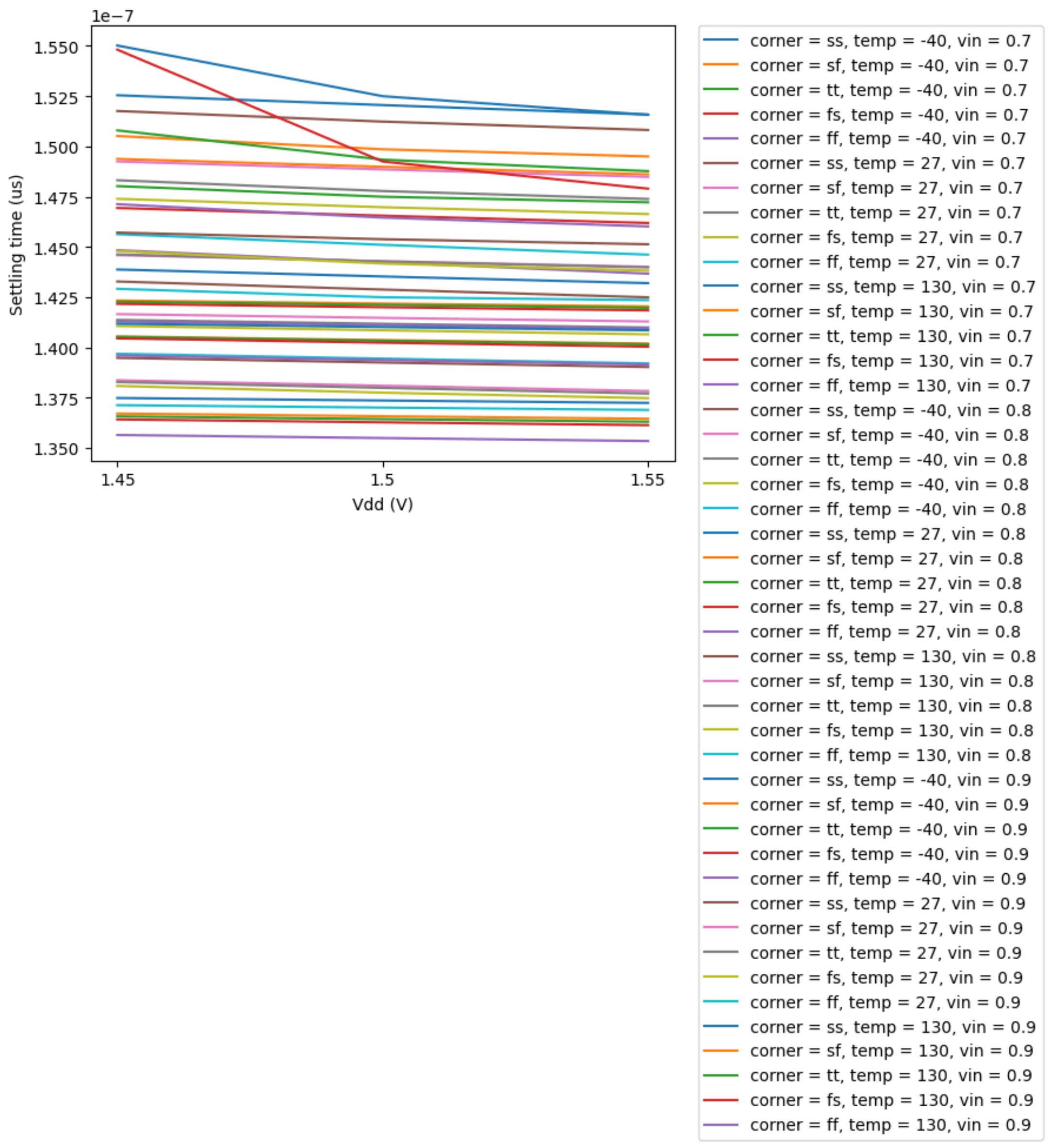


Figure 39: settling_vs_vdd

9.17 settling_vs_corner

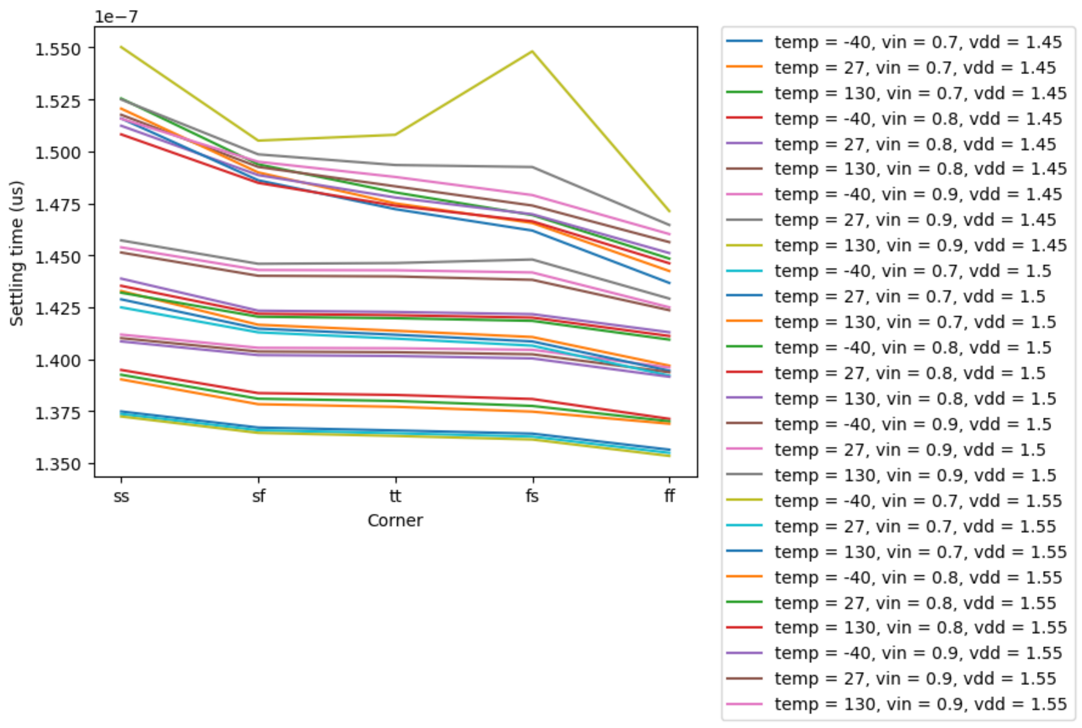


Figure 40: settling_vs_corner

9.17.1 PVT Simulation Analysis

Looking at the CACE report in Note 1 we see that (luckily) the specifiction is met for all parameters. This is great news! We now have a design that we carefully simulated across PVT and other corners, and which is ready for layout. Once we have the layout ready, we can extract the wiring parasitics (R and C) as well as other layout-dependent effects like well proximity. Using this augmented netlist we can then again use CACE to check performance across conditions and parameter variations, and if we still pass all specification points then our design is finished.

10 Cascode Stage

As we have seen in Section 7 the performance of the OTA is generally quite acceptable (see Table 2), but we might want to aim for better output voltage accuracy. As our analysis has shown the output voltage tolerance is limited by the open-loop dc gain A_0 of the OTA (see Equation 12), which in turn is limited by the output conductance of M_2 and M_4 in Figure 18, which is also confirmed by the analytical result in Equation 16.

During the sizing procedure we have seen that the achievable $g_\mathrm{m}/g_\mathrm{ds}$ ratio of a single MOSFET is limited, even if we increase L. We are thus searching for a better option, and here (local) feedback in form of a **cascode** comes to help.

For analysis of a cascode, we use the following single-transistor stage shown in Figure 41.

Source: Article Notebook

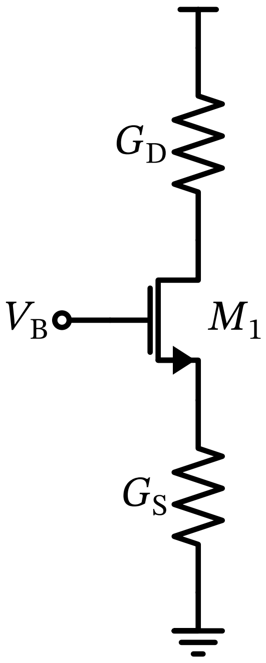


Figure 41: A MOSFET cascode circuit.

Source: Article Notebook

In order to derive the operation of the cascode analytically, we draw the small-signal equivalent circuit in Figure 42. We assume that V_B is a low-ohmic bias voltage, thus we replace it by ac ground. We further set $g_\mathrm{mb} = 0$.

Source: Article Notebook

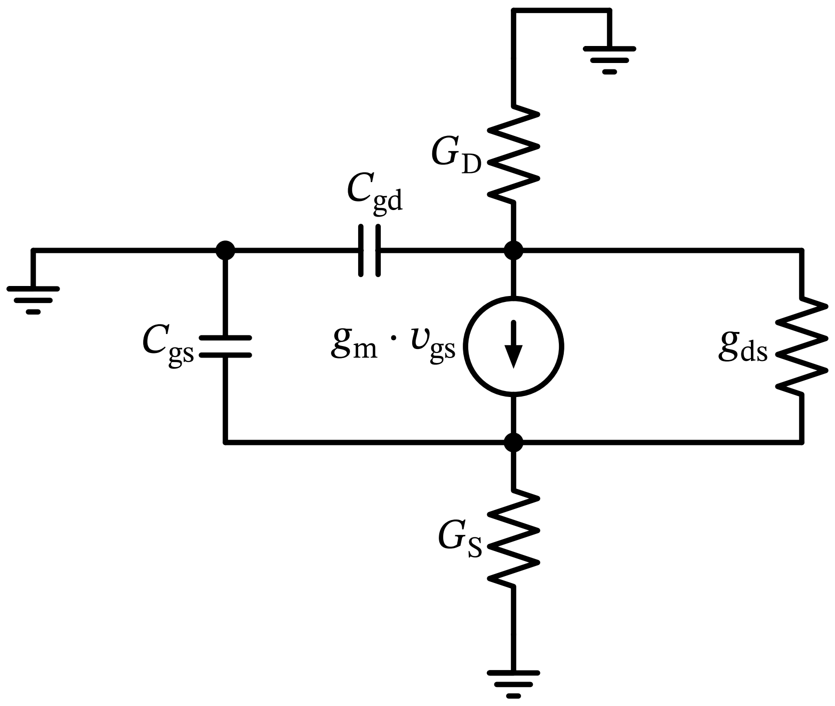


Figure 42: The MOSFET cascode small-signal model.

Source: Article Notebook

Since the gate is assumed at a fixed potential, we can put C_{gs} in parallel to G_S as $G_S^* = G_S + sC_{gs}$, and we can put C_{gd} in parallel to G_D as $G_D^* = G_D + sC_{gd}$. As a result we will disregard these capacitors for now, and just consider G_S and G_D.

Source: Article Notebook

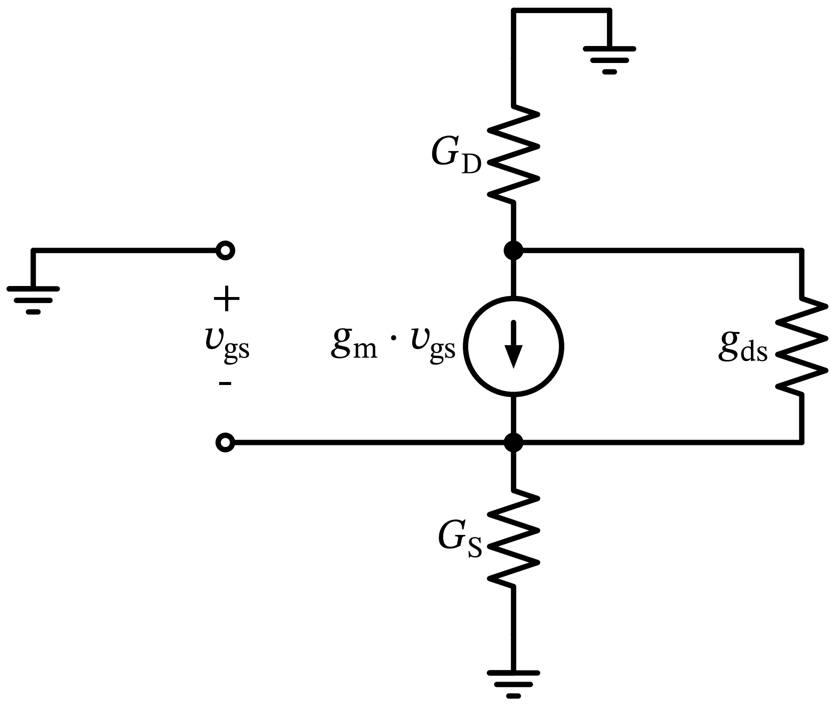


Figure 43: The simplified MOSFET cascode small-signal model.

Source: Article Notebook

10.1 Cascode Output Impedance

As a first step, we want to calculate the output impedance at the drain of the MOSFET (i.e., looking into the drain). For this, we replace G_D with a current source. The resulting small-signal equivalent circuit is shown in Figure 44.

Source: Article Notebook

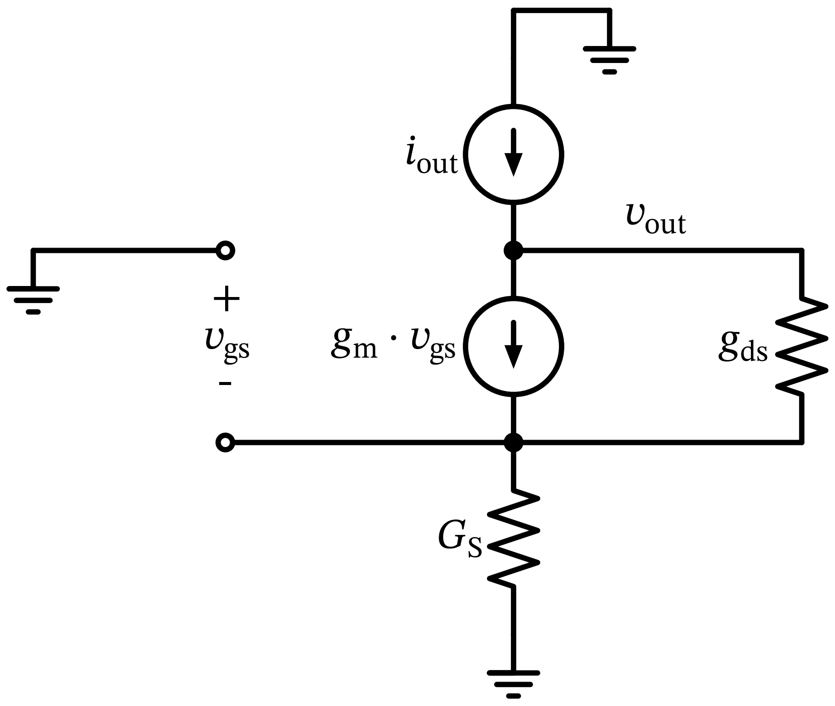


Figure 44: The simplified MOSFET cascode small-signal model for calculation of the output impedance.

Source: Article Notebook

We realize that i_{out} flows through G_S and drops v_{gs} (note the sign):

$$v_{gs} = -\frac{i_{out}}{G_S}$$

Further, $v_{out} = -v_{gs} + v_{ds}$. Calculating KCL at the output node results in

$$i_{out} - g_m v_{gs} - g_{ds} v_{ds} = 0.$$

Using the previously found identities, and after a bit of algebraic manipulations we arrive at

$$g_{out} = \frac{i_{out}}{v_{out}} = \frac{g_{ds}}{1 + \frac{g_m + g_{ds}}{G_S}} = \frac{g_{ds} \cdot G_S}{G_S + g_m + g_{ds}} \tag{22}$$

We find that if $G_S = 0$ (an open) then $g_{out} = 0$, and if $G_S = \infty$ (a short) then $g_{out} = g_{ds}$. We can calculate the benefits of a cascode if we assume we put a cascode on top of a common-source transitor stage (thus $G_S = g'_{ds}$) and get

$$g_{out} = \frac{g_{ds} \cdot g'_{ds}}{g'_{ds} + g_m + g_{ds}} \approx g'_{ds} \frac{g_{ds}}{g_m}. \tag{23}$$

❗ Benefit of Cascode (Output)

The output impedance of the lower MOSFET (g'_{ds}) is **reduced** by the self-gain of the cascode transistor! This is a powerful technique to increase the output impedance of a transistor stage by cascoding, much better than increasing L.

10.2 Cascode Input Impedance

To the calculate the input impedance of a cascode (i.e., looking into the source) we replace G_{S} with a current source. The resulting small-signal equivalent circuit is shown in Figure 45.

Source: Article Notebook

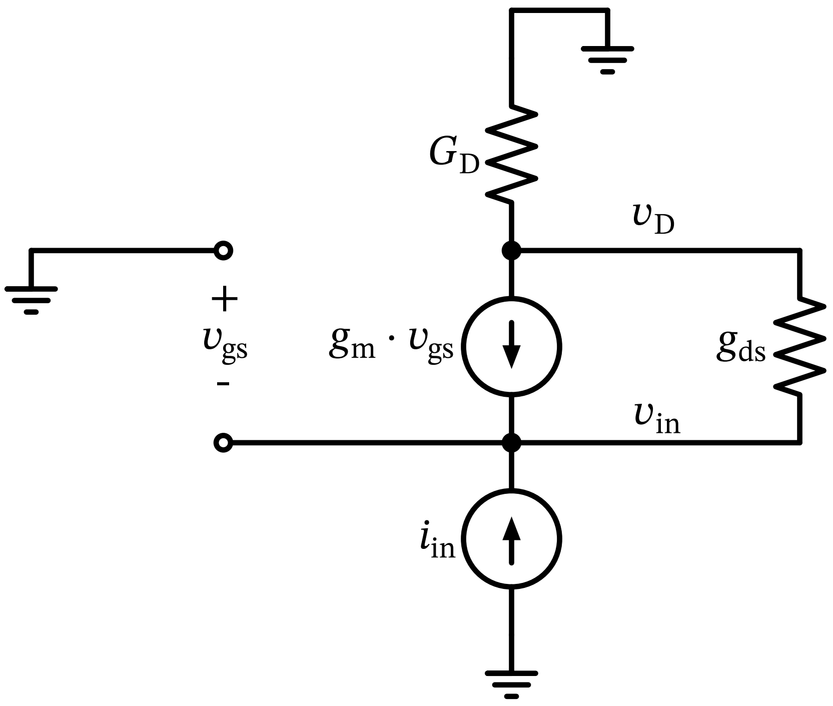


Figure 45: The simplified MOSFET cascode small-signal model for calculation of the input impedance.

Source: Article Notebook

We note that $v_{\mathrm{gs}} = -v_{\mathrm{in}}$ and that i_{in} flows through G_{D}, resulting in $v_{\mathrm{D}} = i_{\mathrm{in}}/G_{\mathrm{D}}$. Note that $v_{\mathrm{ds}} = v_{\mathrm{D}} - v_{\mathrm{in}}$. Formulating KCL at the input node results in

$$i_{\mathrm{in}} + g_{\mathrm{ds}} v_{\mathrm{ds}} + g_{\mathrm{m}} v_{\mathrm{gs}} = 0.$$

After some manipulation we find that

$$g_{\mathrm{in}} = \frac{i_{\mathrm{in}}}{v_{\mathrm{in}}} = \frac{(g_{\mathrm{m}} + g_{\mathrm{ds}}) \cdot G_{\mathrm{D}}}{g_{\mathrm{ds}} + G_{\mathrm{D}}}. \tag{24}$$

Setting $G_{\mathrm{D}} = 0$ (an open) results in $g_{\mathrm{in}} = 0$ as well, so the input impedance of the cascode is very large when the drain impedance is large.

However, setting $G_{\mathrm{D}} = \infty$ (a short or low-ohmic impedance) results in the well-known result of $g_{\mathrm{in}} = g_{\mathrm{m}} + g_{\mathrm{ds}} \approx g_{\mathrm{m}}$, which means that the input impedance looking into a cascode is approximately $1/g_{\mathrm{m}}$.

! Benefit of Cascode (Input)

This has the practical benefit that a capacitance connected at this noise results in a high-frequency pole, which is often not critical in terms of stability. Further, the voltage swing at a cascode input node is small due to the often small impedance, and this minimizes the Miller effect at connected inter-node capacitors (see Section 19).

11 Improved OTA

With the new learned know-how of the cascode stage we can set out to improve our original basic 5T-OTA design. Essentially this means to add cascodes to M_2 and M_4 in Figure 18. For symmetry reasons we will add cascodes to both sides, and the resulting schematic is shown in Figure 46.

Source: Article Notebook

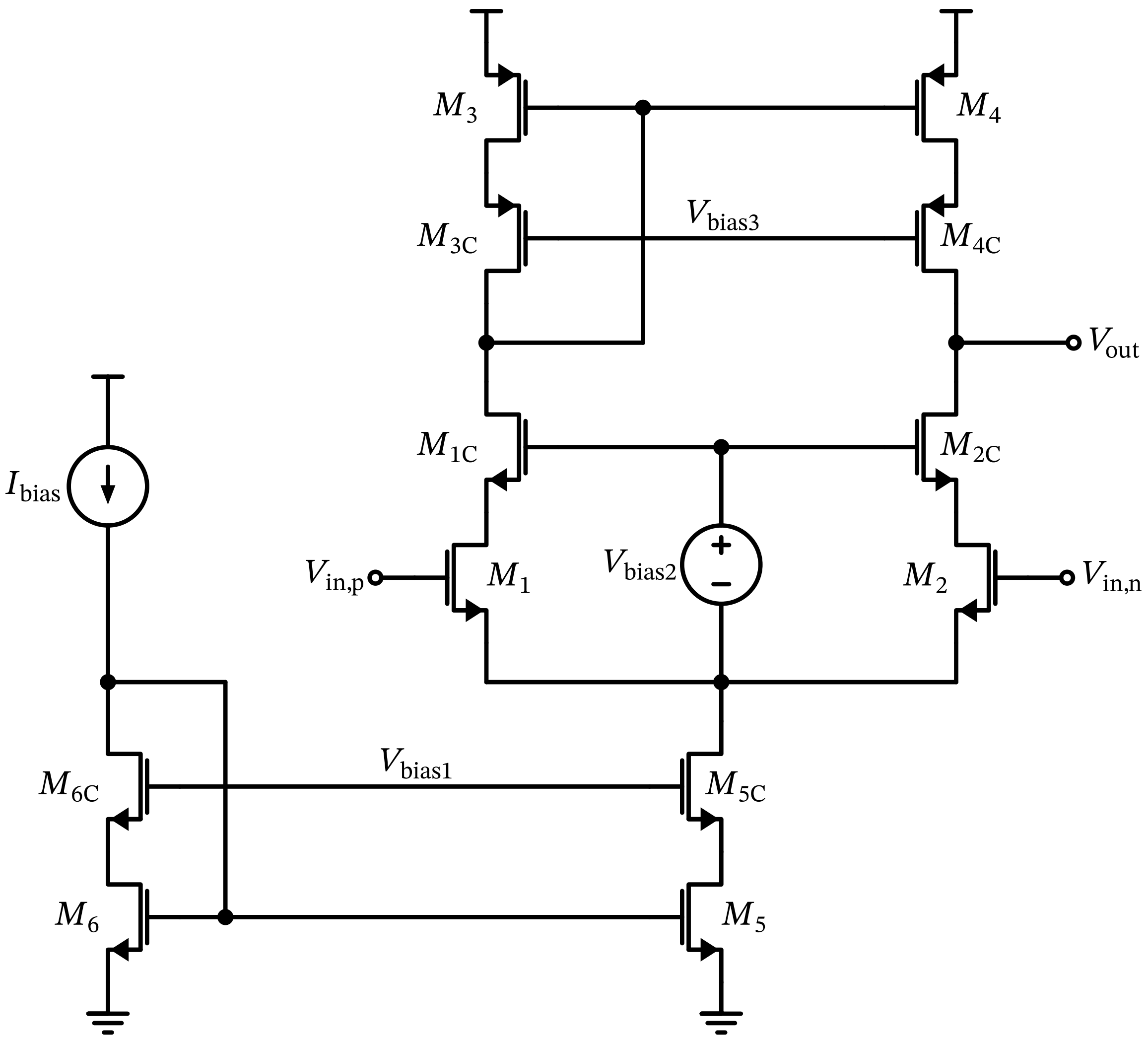


Figure 46: The improved OTA based on the 5T-OTA design.

Source: Article Notebook

The transistor name appendix “C” indicates a cascode device sitting atop its base transistor. The bias voltage V_{bias1} is referenced to V_{SS}, the bias voltage V_{bias3} is referenced to V_{DD}, and the floating bias voltage V_{bias2} creates a voltage bias for $M_{1\mathrm{C}}$ and $M_{2\mathrm{C}}$ relative to the tail point, so that the V_{DS} of $M_{1,2}$ stays constant with a changing common-mode input voltage.

! Cascode Bias Voltage Generation

It is critically import for a stable performance across PVT that the bias voltages for the cascode gates are created in a manner that tracks variations with process, temperature, and supply voltage!

The current mirrors constructed out of $M_{5/5\mathrm{C},6/6\mathrm{C}}$ and $M_{3/3\mathrm{C},4/4\mathrm{C}}$ are a special kind of

cascoded current mirror for low-voltage operation. This type is very often used, as it forces the V_{GS} and V_{DS} of $M_{5,6}$ (and $M_{3,4}$) to be equal, so the current mirror ratio is independent of g_{ds}. Further, by properly selecting the bias voltages of the cascode a low-voltage operation is achieved as V_{DS} can be minimized, allowing even triode operation of the current-mirror MOSFETs (as, noted above, a large g_{ds} is not a big issue).

A simplified small-signal gain calculation of this improved OTA uses the result of Equation 16 and Equation 23 to arrive at the approximate dc gain of

$$A_0 \approx \frac{g_{\mathrm{m12}}}{g_{\mathrm{ds2}}\frac{g_{\mathrm{ds2C}}}{g_{\mathrm{m2C}}} + g_{\mathrm{ds4}}\frac{g_{\mathrm{ds4C}}}{g_{\mathrm{m4C}}}} \tag{25}$$

leading to a significant boost in dc gain due to cascoding. We will use this increased gain to reduce the L of all MOSFET to

1. save area (smaller L will lead to smaller W for a given W/L ratio) and
2. will push the additional poles and zeros at the inner nodes of the cascoded transistors (e.g., the connection of the drain of M_5 to the source of $M_{5\mathrm{C}}$) to higher frequencies to result in stable behaviour and a reasonable gain transfer function (too many poles and zeros in the passband of the amplifier create many issues with stability margin).

11.1 Sizing the Improved OTA

Like the sizing of the 5T-OTA in Section 7.4 we will again use the $g_{\mathrm{m}}/I_{\mathrm{D}}$ method using a Python notebook. Instead of using $L = 5\,\mu$m we will this time use a reduced $L = 0.5\,\mu$m for $M_{1/1\mathrm{C},2/2\mathrm{C},3/3\mathrm{C},4/4\mathrm{C}}$ (for speed reasons) and $L = 1\,\mu$m for $M_{5/5\mathrm{C},6/6\mathrm{C}}$ for better common-mode rejection (the tail current mirror is less critical in terms of speed and stability).

We set $g_{\mathrm{m}}/I_{\mathrm{D}} = 13$ across the board for a good trade-off between speed, current efficiency, and voltage headroom for the MOSFETs (this is now way more critical than in the basic 5T-OTA as we stack now double as many MOSFET at the same supply voltage). Please look at Section 3 to confirm this choice.

Improved OTA Sizing

12 Sizing for Basic (Improved) OTA

```
# Read table data
from pygmid import Lookup as lk
import numpy as np
lv_nmos = lk('sg13_lv_nmos.mat')
lv_pmos = lk('sg13_lv_pmos.mat')
# List of parameters: VGS, VDS, VSB, L, W, NFING, ID, VT, GM, GMB, GDS, CGG, CGB, CGD, C
# If not specified, minimum L, VDS=max(vgs)/2=0.9 and VSB=0 are used
```

```
# Define the given parameters as taken from the specification table or inital guesses
c_load = 50e-15
gm_id_m12 = 13
gm_id_m12c = 13
gm_id_m34 = 13
gm_id_m34c = 13
gm_id_m56 = 13
gm_id_m56c = 13
l_12 = 0.5
l_12c = 0.5
l_34 = 0.5
l_34c = 0.5
l_56 = 1
l_56c = 1
f_bw = 10e6
i_total_limit = 10e-6
i_bias_in = 20e-6
output_voltage = 1.3
vin_min = 0.7
vin_max = 0.9
vdd_min = 1.45
vdd_max = 1.55
vds_headroom = 0.2
```

```
# We get the required gm of M1/2 from the bandwidth requirement
# We add a factor of 3 to allow for PVT variation plus additional MOSFET parasitic loadi
# We also add an additional factor of 2 to get more dc gain (and there is power still in
gm_m12 = f_bw * 3 * 4*np.pi*c_load * 3
print('gm12 =', gm_m12/1e-3, 'mS')
```

```
gm12 = 0.05654866776461628 mS
```

```python
# Since we know gm12 and the gmid we can calculate the bias current
id_m12 = gm_m12 / gm_id_m12
i_total = 2*id_m12
print('i_total (exact) =', i_total/1e-6, 'µA')
# we round to 0.5µA bias currents
i_total = max(round(i_total / 1e-6 * 2) / 2 * 1e-6, 0.5e-6)
# here is a manual override to set the current; we keep a reserve of 2µA for bias branch
i_total = 8e-6
id_m12 = i_total/2

print('i_total (rounded) =', i_total/1e-6, 'µA')
if i_total < i_total_limit:
    print('[info] power consumption target is met!')
else:
    print('[info] power consumption target is NOT met!')
```

```
i_total (exact) = 8.699795040710196 µA
i_total (rounded) = 8.0 µA
[info] power consumption target is met!
```

```python
# We calculate the dc gain
gm_gds_m12 = lv_nmos.lookup('GM_GDS', GM_ID=gm_id_m12, L=l_12, VDS=vds_headroom, VSB=2*v
gm_gds_m12c = lv_nmos.lookup('GM_GDS', GM_ID=gm_id_m12c, L=l_12c, VDS=vds_headroom, VSB=
gm_gds_m34 = lv_pmos.lookup('GM_GDS', GM_ID=gm_id_m34, L=l_34, VDS=vds_headroom, VSB=0)
gm_gds_m34c = lv_pmos.lookup('GM_GDS', GM_ID=gm_id_m34c, L=l_34c, VDS=vds_headroom, VSB=
# conductance of lower cascoded differential pair
gds_m12 = gm_m12 / gm_gds_m12
gds_m12_casc = gds_m12 / gm_gds_m12c
# conductance of upper cascoded current mirror
gm_m34 = gm_id_m34 * i_total/2
gds_m34 = gm_m34 / gm_gds_m34
gds_m34_casc = gds_m34 / gm_gds_m34c

print('gds_12 =', gds_m12/1e-6, 'µs')
print('gm_12c/gds_12c =',gm_gds_m12c)
print('gds_34 =', gds_m34/1e-6, 'µs')
print('gm_34c/gds_34c =', gm_gds_m34c)

a0 = gm_m12 / (gds_m12_casc + gds_m34_casc)
print('a0 =', 20*np.log10(a0), 'dB')
```

```
gds_12 = 4.025519543033504 µs
gm_12c/gds_12c = 13.377388738589055
gds_34 = 2.031305802765517 µs
```

```
gm_34c/gds_34c = 24.877072747451322
a0 = 43.394151889182424 dB
```

```
# We calculate the MOSFET capacitance which adds to Cload, to see the impact on the BW
gm_cgs_m12 = lv_nmos.lookup('GM_CGS', GM_ID=gm_id_m12, L=l_12, VDS=vds_headroom, VSB=2*v
gm_cdd_m12c = lv_nmos.lookup('GM_CDD', GM_ID=gm_id_m12c, L=l_12c, VDS=vds_headroom, VSB=
gm_cdd_m34c = lv_pmos.lookup('GM_CDD', GM_ID=gm_id_m34c, L=l_34c, VDS=vds_headroom, VSB=

c_load_parasitic = abs(gm_m12/gm_cgs_m12) + abs(gm_m12/gm_cdd_m12c) + abs(gm_m34/gm_cdd_
print('additional load capacitance =', c_load_parasitic/1e-15, 'fF')

f_bw = gm_m12 / (4*np.pi * (c_load + c_load_parasitic))
print('-3dB bandwidth incl. parasitics =', f_bw/1e6, 'MHz')
```

```
additional load capacitance = 5.4535230735668225 fF
-3dB bandwidth incl. parasitics = 81.14903707795307 MHz
```

```
# We can now look up the VGS of the MOSFET
vgs_m12 = lv_nmos.look_upVGS(GM_ID=gm_id_m12, L=l_12, VDS=vds_headroom, VSB=2*vds_headro
vgs_m12c = lv_nmos.look_upVGS(GM_ID=gm_id_m12c, L=l_12c, VDS=vds_headroom, VSB=3*vds_hea
vgs_m34 = lv_pmos.look_upVGS(GM_ID=gm_id_m34, L=l_34, VDS=vds_headroom, VSB=0.0)
vgs_m34c = lv_pmos.look_upVGS(GM_ID=gm_id_m34c, L=l_34c, VDS=vds_headroom, VSB=vds_headr
vgs_m56 = lv_nmos.look_upVGS(GM_ID=gm_id_m56, L=l_56, VDS=vds_headroom, VSB=0.0)
vgs_m56c = lv_nmos.look_upVGS(GM_ID=gm_id_m56c, L=l_56c, VDS=vds_headroom, VSB=vds_headr

print('vgs_12  =', vgs_m12, 'V')
print('vgs_12c =', vgs_m12c, 'V')
print('vgs_34  =', vgs_m34, 'V')
print('vgs_34c =', vgs_m34c, 'V')
print('vgs_56  =', vgs_m56, 'V')
print('vgs_56c =', vgs_m56c, 'V')
```

```
vgs_12  = 0.4363351047848177 V
vgs_12c = 0.4575885897698743 V
vgs_34  = 0.4745538724103232 V
vgs_34c = 0.5115503281076889 V
vgs_56  = 0.35764913382485647 V
vgs_56c = 0.3835723411901689 V
```

```
# Calculate settling time due to slewing with the calculated bias current
t_slew = (c_load + c_load_parasitic) * output_voltage / i_total
print('slewing time  =', t_slew/1e-6, 'µs')
t_settle = 5/(2*np.pi*f_bw)
print('settling time =', t_settle/1e-6, 'µs')
```

```
slewing time  = 0.009011197499454612 µs
settling time = 0.009806335898909592 µs
```

```
# Calculate voltage gain error
gain_error = a0 / (1 + a0)
print('voltage gain error =', (gain_error-1)*100, '%')
```

```
voltage gain error = -0.6719920439173688 %
```

```
# Calculate total rms output noise
sth_m12 = lv_nmos.lookup('STH_GM', VGS=vgs_m12, L=l_12, VDS=vds_headroom, VSB=2*vds_head
gamma_m12 = sth_m12/(4*1.38e-23*300*gm_m12)

sth_m34 = lv_pmos.lookup('STH_GM', VGS=vgs_m34, L=l_34, VDS=vds_headroom, VSB=0) * gm_m3
gamma_m34 = sth_m34/(4*1.38e-23*300*gm_m34)

output_noise_rms = 1.38e-23*300 / (c_load + c_load_parasitic) * (2*gamma_m12 + 2*gamma_m
print('output noise (rms) =', output_noise_rms/1e-6, 'µV')
```

```
output noise (rms) = 0.3084826103706843 µV
```

```
# Calculate all widths
id_w_m12 = lv_nmos.lookup('ID_W', GM_ID=gm_id_m12, L=l_12, VDS=vds_headroom, VSB=2*vds_h
w_12 = id_m12 / id_w_m12
w_12_round = max(round(w_12*2)/2, 0.5)
print('M1/2  W =', w_12, 'um, rounded W =', w_12_round, 'um')

id_m12c = id_m12
id_w_m12c = lv_nmos.lookup('ID_W', GM_ID=gm_id_m12c, L=l_12c, VDS=vds_headroom, VSB=3*vd
w_12c = id_m12c / id_w_m12c
w_12c_round = max(round(w_12c*2)/2, 0.5)
print('M1/2c W =', w_12c, 'um, rounded W =', w_12c_round, 'um')

id_m34 = id_m12
id_w_m34 = lv_pmos.lookup('ID_W', GM_ID=gm_id_m34, L=l_34, VDS=vds_headroom, VSB=0)
w_34 = id_m34 / id_w_m34
w_34_round = max(round(w_34*2)/2, 0.5)
print('M3/4  W =', w_34, 'um, rounded W =', w_34_round, 'um')

id_m34c = id_m12
id_w_m34c = lv_pmos.lookup('ID_W', GM_ID=gm_id_m34c, L=l_34c, VDS=vds_headroom, VSB=vds_
w_34c = id_m34c / id_w_m34c
w_34c_round = max(round(w_34c*2)/2, 0.5)
print('M3/4c W =', w_34c, 'um, rounded W =', w_34c_round, 'um')

id_w_m5 = lv_nmos.lookup('ID_W', GM_ID=gm_id_m56, L=l_56, VDS=vds_headroom, VSB=0)
w_5 = i_total / id_w_m5
w_5_round = max(round(w_5*2)/2, 0.5)
print('M5    W =', w_5, 'um, rounded W =', w_5_round, 'um')

id_w_m5c = lv_nmos.lookup('ID_W', GM_ID=gm_id_m56c, L=l_56c, VDS=vds_headroom, VSB=vds_h
w_5c = i_total / id_w_m5c
w_5c_round = max(round(w_5c*2)/2, 0.5)
print('M5c   W =', w_5c, 'um, rounded W =', w_5c_round, 'um')

w_6 = w_5_round * i_bias_in / i_total
print('M6    W =', w_6, 'um')

w_6c = w_5c_round * i_bias_in / i_total
print('M6c   W =', w_6c, 'um')
```

```
M1/2  W = 0.8310873754124203 um, rounded W = 1.0 um
M1/2c W = 0.8017034903234459 um, rounded W = 1.0 um
M3/4  W = 3.281329534410626 um, rounded W = 3.5 um
M3/4c W = 2.98941704661016 um, rounded W = 3.0 um
```

```
M5    W = 3.0508401171424118 um, rounded W = 3.0 um
M5c   W = 2.8749923811908227 um, rounded W = 3.0 um
M6    W = 7.500000000000002 um
M6c   W = 7.500000000000002 um
```

```python
# Print out final design values
print('Improved OTA dimensioning:')
print('--------------------------')
print('M1/2  W=', w_12_round, ', L=', l_12)
print('M1/2c W=', w_12c_round, ', L=', l_12c)
print('M3/4  W=', w_34_round, ', L=', l_34)
print('M3/4c W=', w_34c_round, ', L=', l_34c)
print('M5   W=', w_5_round, ', L=', l_56)
print('M5c  W=', w_5c_round, ', L=', l_56c)
print('M6   W=', w_6, ', L=', l_56)
print('M6c  W=', w_6c, ', L=', l_56c)
print()
print('Improved OTA performance summary:')
print('---------------------------------')
print('supply current =', i_total/1e-6, 'µA')
print('output noise =', output_noise_rms/1e-6, 'µVrms')
print('voltage gain error =', (gain_error-1)*100, '%')
print('-3dB bandwidth incl. parasitics =', f_bw/1e6, 'MHz')
print('turn-on time (slewing+settling) =', (t_slew+t_settle)/1e-6, 'µs')
print()
print('Improved OTA bias point check:')
print('------------------------------')
print('headroom M1+M1c =', vdd_min-vgs_m34+vgs_m12-vin_max, 'V')
print('headroom M4+M4c =', vdd_min-vin_max, 'V')
print('headroom M5+M5c =', vin_min-vgs_m12, 'V')
```

```
Improved OTA dimensioning:
--------------------------
M1/2  W= 1.0 , L= 0.5
M1/2c W= 1.0 , L= 0.5
M3/4  W= 3.5 , L= 0.5
M3/4c W= 3.0 , L= 0.5
M5   W= 3.0 , L= 1
M5c  W= 3.0 , L= 1
M6   W= 7.500000000000002 , L= 1
M6c  W= 7.500000000000002 , L= 1

Improved OTA performance summary:
---------------------------------
supply current = 8.0 µA
```

```
output noise = 0.3084826103706843 µVrms
voltage gain error = -0.6719920439173688 %
-3dB bandwidth incl. parasitics = 81.14903707795307 MHz
turn-on time (slewing+settling) = 0.018817533398364204 µs

Improved OTA bias point check:
------------------------------
headroom M1+M1c = 0.5117812323744945 V
headroom M4+M4c = 0.5499999999999999 V
headroom M5+M5c = 0.26366489521518227 V
```

Source: Sizing for Basic (Improved) OTA

Looking at this sizing result we see that we achieve an improved $A0 > 43\,\mathrm{dB}$ while meeting also the other performance requirements of Table 2 with margin. In addition, we check the voltage headroom of the critical MOSFET to see if we can squeeze it into the available supply voltage range, and see that this is possible with our above choice selection of parameters.

Exercise: Improved OTA Sizing

Please take a detailed look at the above sizing notebook and play with the numbers and calculations. Do you find a better trade-off for the input parameters? Can you understand the ratio behind the choices and calculations?

12.1 Designing the Improved OTA

Based on the collected experience in this lecture and the result of the sizing procedure in Section 11.1 you should be able to design this OTA. If you want, please go ahead and try an implementation and check its performance with CACE.

As an alternative there is a prepared OTA design shown in Figure 47 which we will discuss in detail next.

12.1.1 Discussion of the OTA Design

We will now do an analysis of the circuit design of the OTA including all the complications which make this design practical.

1. For easier navigation, the device identifier are consistent with the circuit sketch in Figure 46.
2. Some MOSFET dimensions are rounded to make a better fit in IC layout. Please also look carefully at W, L, and ng. The parameter ng sets how the total W of a MOSFET should be split into individual MOSFET fingers with $W_\mathrm{f} = W/\mathrm{ng}$. This is done to arrive at a suitably sized MOSFET physical implementation. As we will not deal with IC layout in this lecture we will leave it at that.

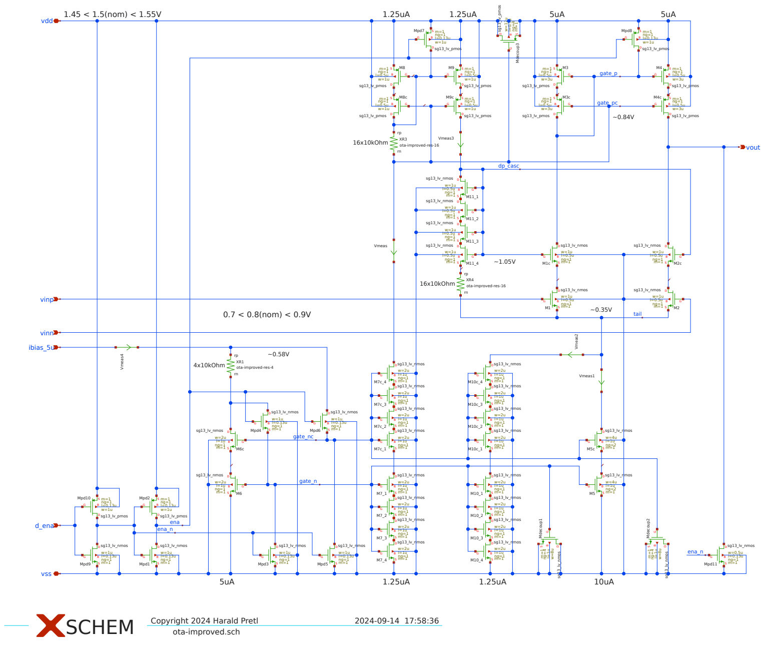


Figure 47: Improved OTA design in Xschem.

3. In order to allow good matching in the IC layout, MOSFETs (and other components) have to constructed from equal pieces. To that end, W/L scaling is done using unit elements (see finger width W_f). Sometimes, besides W the length L has to be scaled, and this leads to the oddly-looking series stacking of some MOSFET (easily recognizable by the connected gates). In order to increase circuit readability, a subcircuit could be constructed hiding this series stacking of MOSFET, but it is sometimes easier to avoid subcircuits. There is a fine line in this trade, sometime a depth of 4 is the decision point between subcircuit use/no-use.
4. As you can (hopefully) see the circuit is carefully drawn to ease readability. Important nets are named, text comments state certain properties like nominal voltage levels, bias currents, etc. Current sensing elements are added to directly see the dc currents in the circuit simulation.
5. The bias voltage generation for the cascodes is included as well. The voltage drop for the bottom transistors is developed across a resistor (this is a simple but effective way, but other implementations are possible as well). We are using a dummy branch for bias generation (constructed with $M_{7/7\mathrm{C},8/8\mathrm{C}}$ and R_3).
6. The floating bias voltage V_{bias2} is created by implementing a current source from V_{DD} ($M_{9/9\mathrm{C}}$), then a MOSFET diode M_{11} with resistor R_4, and an additional current source towards V_{SS} ($M_{10/10\mathrm{C}}$). Instead of using the bottom current source the current in the tail current source $M_{5/5\mathrm{C}}$ could have been increased, but matching is better with the chosen approach.
7. Power-down transistors $M_{\mathrm{pd},x}$ are added to allow a proper shutdown of the circuit with a digital enable input. It is generally a good idea to clamp floating nodes in off-mode so that no issues during power-down (like increased leakage currents) or delayed

startup or shutdown are occuring. It is further a good design principle to buffer all incoming digital signals with inverters connected to the local supply. This lowers the risk of unwanted noise coupling or excessive slewrates on the incoming digital signals.

8. Sensitive bias nodes are buffered with decoupling capacitors. We are using MOSFET as nonlinear capacitors, which is not an issue in this application, but we value the increased capacitive density. Please note how the MOSFET are connected (to V_{DD}? or to V_{SS}?).

The resistor used in this circuit are subcircuits to allow series connection of unit resistor elements. The schematic of one element is shown in Figure 48. It is using an effective method to create a series string of connected resistors using wire bundles. Try to understand the circuit, consult the Xschem manual, and look at the resulting SPICE netlist to confirm your finding.

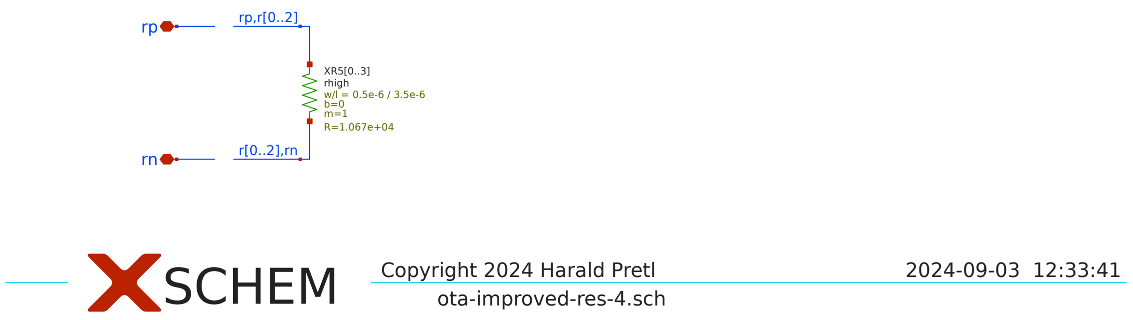


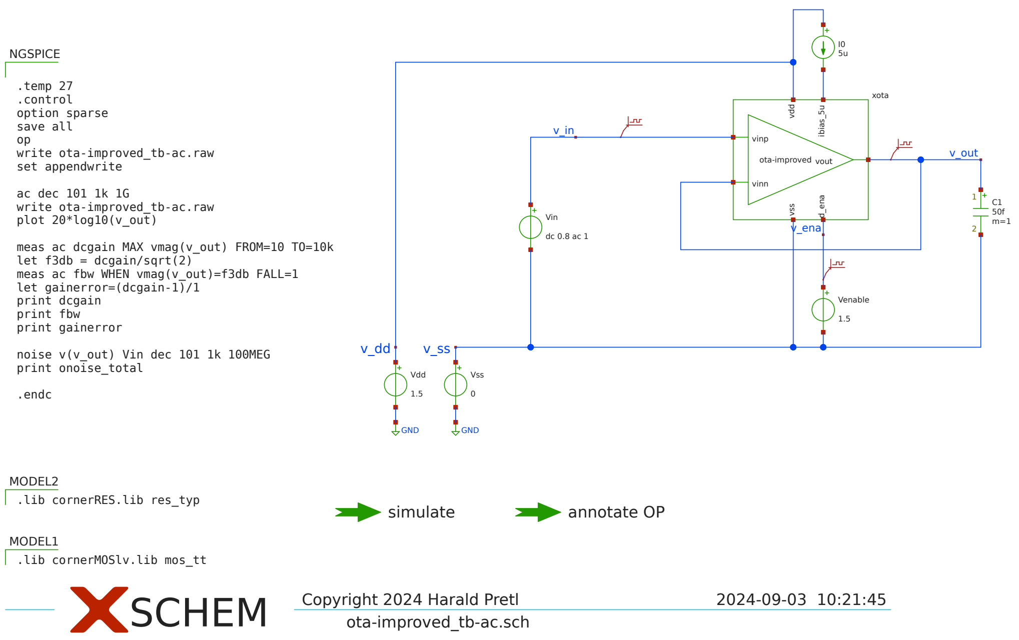
Figure 48: Series resistor implementation used in the improved OTA design.

Parallel Connection

Note that a parallel connection of devices is effectively possible using the multiplier notation of Xschem.

12.2 Simulation of Improved OTA

Now that the circuit design of the improved OTA is done, we an use the same simulation test bench as for the basic OTA. The testbench is shown in Figure 49 and Figure 50.


NGSPICE

```
.temp 27
.control
option sparse
save all
op
write ota-improved_tb-ac.raw
set appendwrite

ac dec 101 1k 1G
write ota-improved_tb-ac.raw
plot 20*log10(v_out)

meas ac dcgain MAX vmag(v_out) FROM=10 TO=10k
let f3db = dcgain/sqrt(2)
meas ac fbw WHEN vmag(v_out)=f3db FALL=1
let gainerror=(dcgain-1)/1
print dcgain
print fbw
print gainerror

noise v(v_out) Vin dec 101 1k 100MEG
print onoise_total

.endc
```

MODEL2

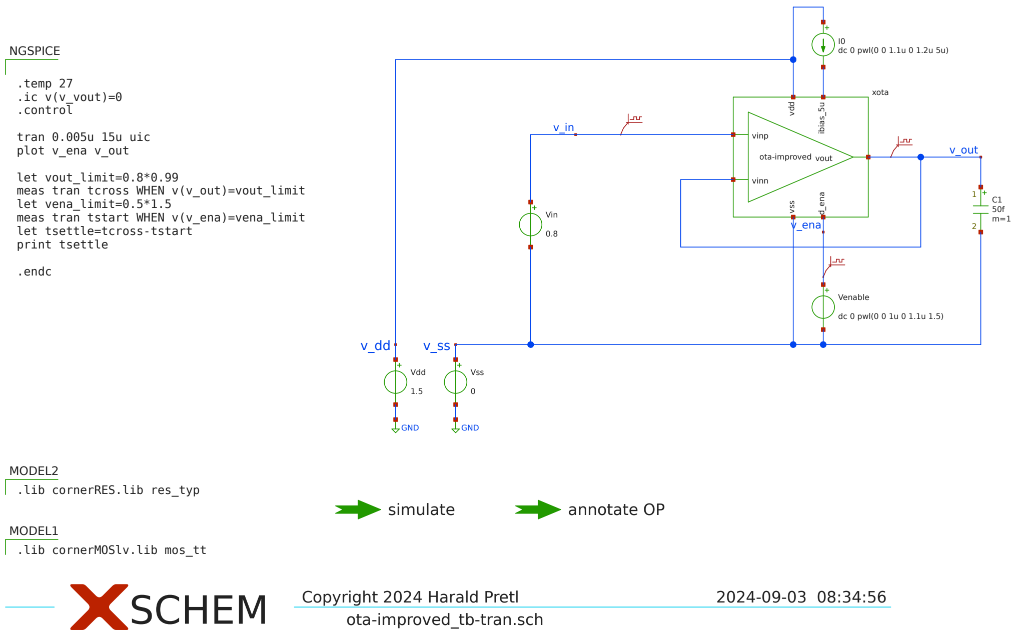
```
.lib cornerRES.lib res_typ
```

MODEL1

```
.lib cornerMOSlv.lib mos_tt
```

Copyright 2024 Harald Pretl 2024-09-03 10:21:45
ota-improved_tb-ac.sch

Figure 49: Simulation testbench of the improved OTA design (small-signal).


NGSPICE

```
.temp 27
.ic v(v_vout)=0
.control

tran 0.005u 15u uic
plot v_ena v_out

let vout_limit=0.8*0.99
meas tran tcross WHEN v(v_out)=vout_limit
let vena_limit=0.5*1.5
meas tran tstart WHEN v(v_ena)=vena_limit
let tsettle=tcross-tstart
print tsettle

.endc
```

MODEL2

```
.lib cornerRES.lib res_typ
```

MODEL1

```
.lib cornerMOSlv.lib mos_tt
```

Copyright 2024 Harald Pretl 2024-09-03 08:34:56
ota-improved_tb-tran.sch

Figure 50: Simulation testbench of the improved OTA design (large-signal).

Exercise: Improved OTA Initial Simulation

Please use the above testbenches to simulate the improved OTA:

1. Check the dc bias points. Are they good? How stable are they across PVT variations?
2. What are the small-signal parameters like gain, noise and bandwidth? Are they fitting the specification?
3. What is large-signal performance? Is the settling fast enough? Is the settling well behaved, i.e., are there overshoots or other strange ringing indicating potential stability issues?
4. Try to improve the design. Change various device parameters and see what happens. Whenever you change something, check the dc operating point first. If the dc operating point is not good no further simulations make sense.

12.3 Corner Simulation of Improved OTA

Just like for the basic OTA we use the CACE system to check the performance of the improved OTA design holistically across variations like PVT and input signal variations. The results of the CACE run are shown below in Note 2.

Note 2: CACE Summary for Improved OTA

13 CACE Summary for ota-improved

netlist source: schematic

Parameter	Tool	Result	Min Limit	Min Value	Typ Target	Typ Value	Max Limit	Max Value	Status
Output voltage ratio	ngspice	gain	0.99 V/V	1.000 V/V	any	1.001 V/V	1.01 V/V	1.009 V/V	Pass
Bandwidth	ngspice	bw	10e6 Hz	107908000.000 Hz	any	226025000.000 Hz	any	292975000.000 Hz	Pass
Output noise	ngspice	noise	any	0.346 mV	any	0.407 mV	1 mV	0.497 mV	Pass
Settling time	ngspice	tsettle	any	0.196 us	any	0.212 us	5 us	0.226 us	Pass

13.1 Plots

13.2 gain_vs_temp

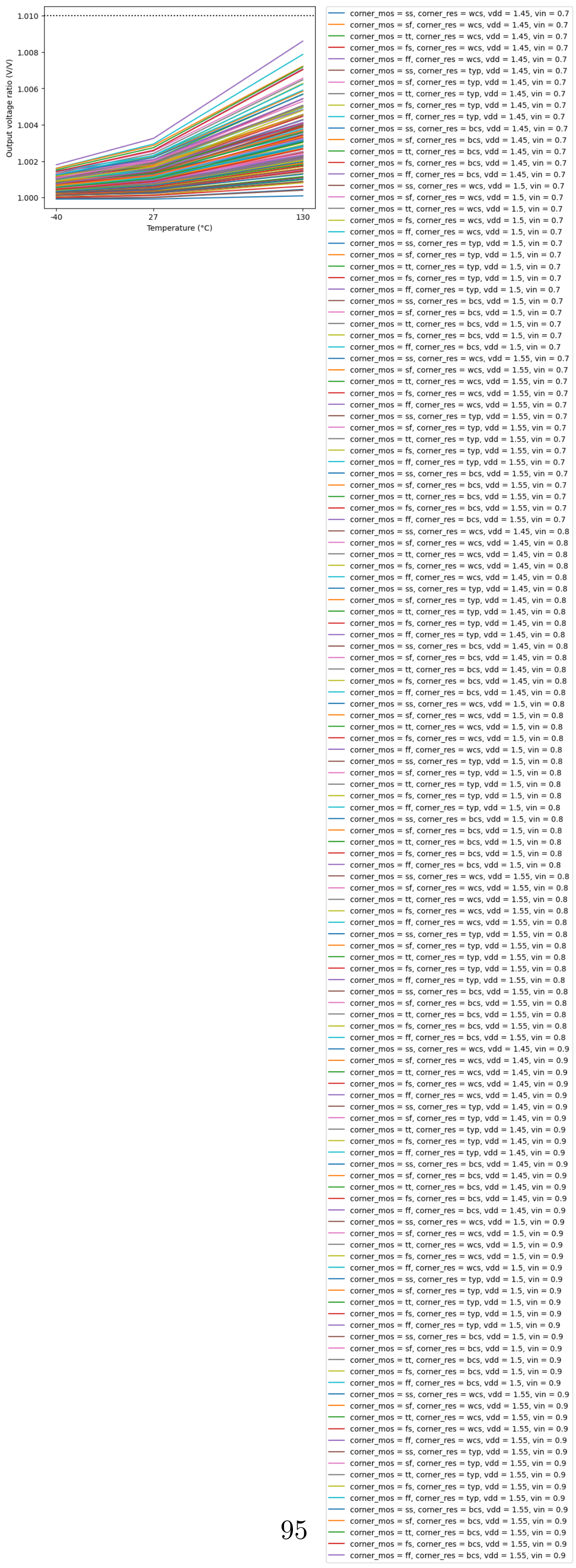


Figure 51: gain_vs_temp

13.3 gain_vs_vin

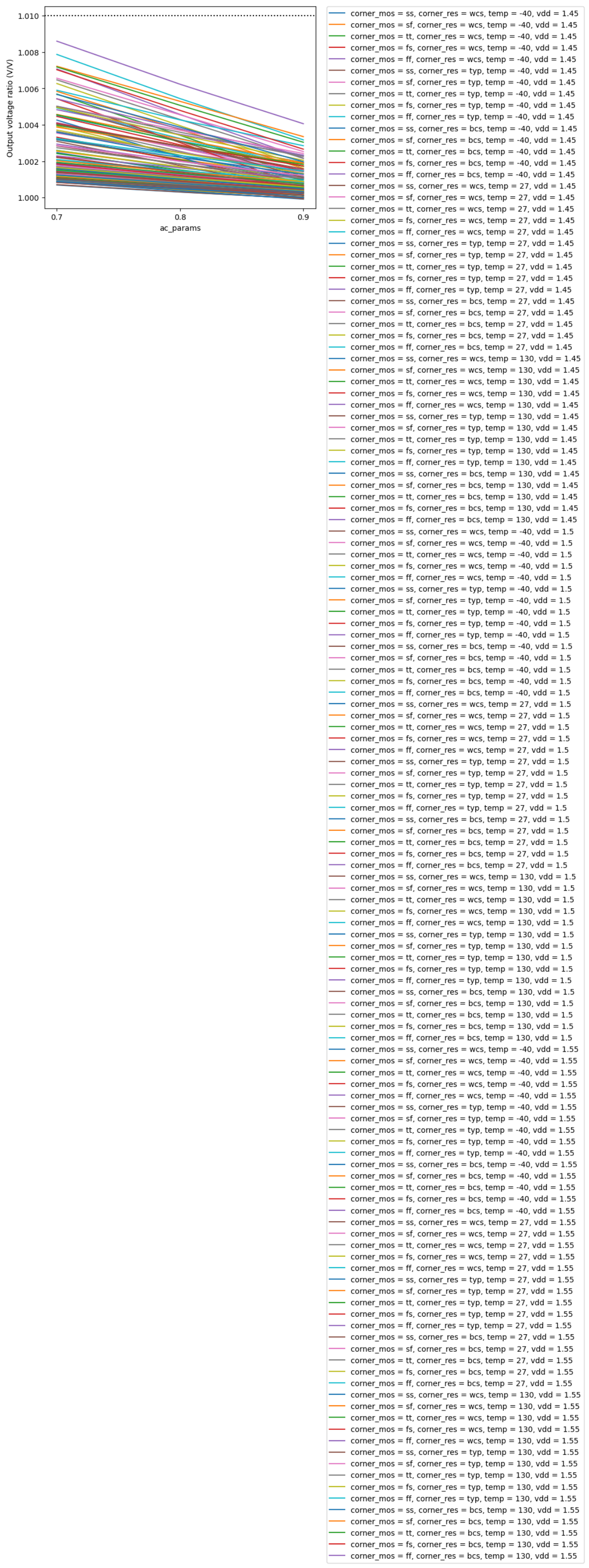


Figure 52: gain_vs_vin

13.4 gain_vs_vdd

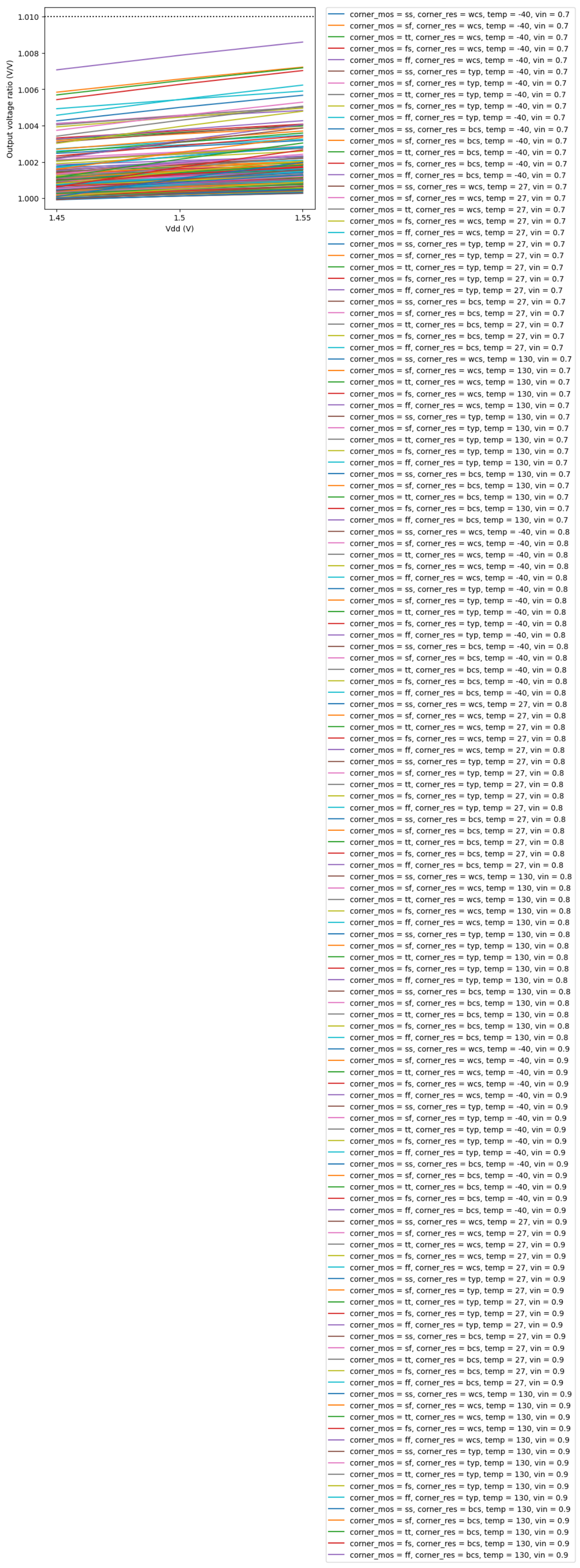


Figure 53: gain_vs_vdd

13.5 gain_vs_corner

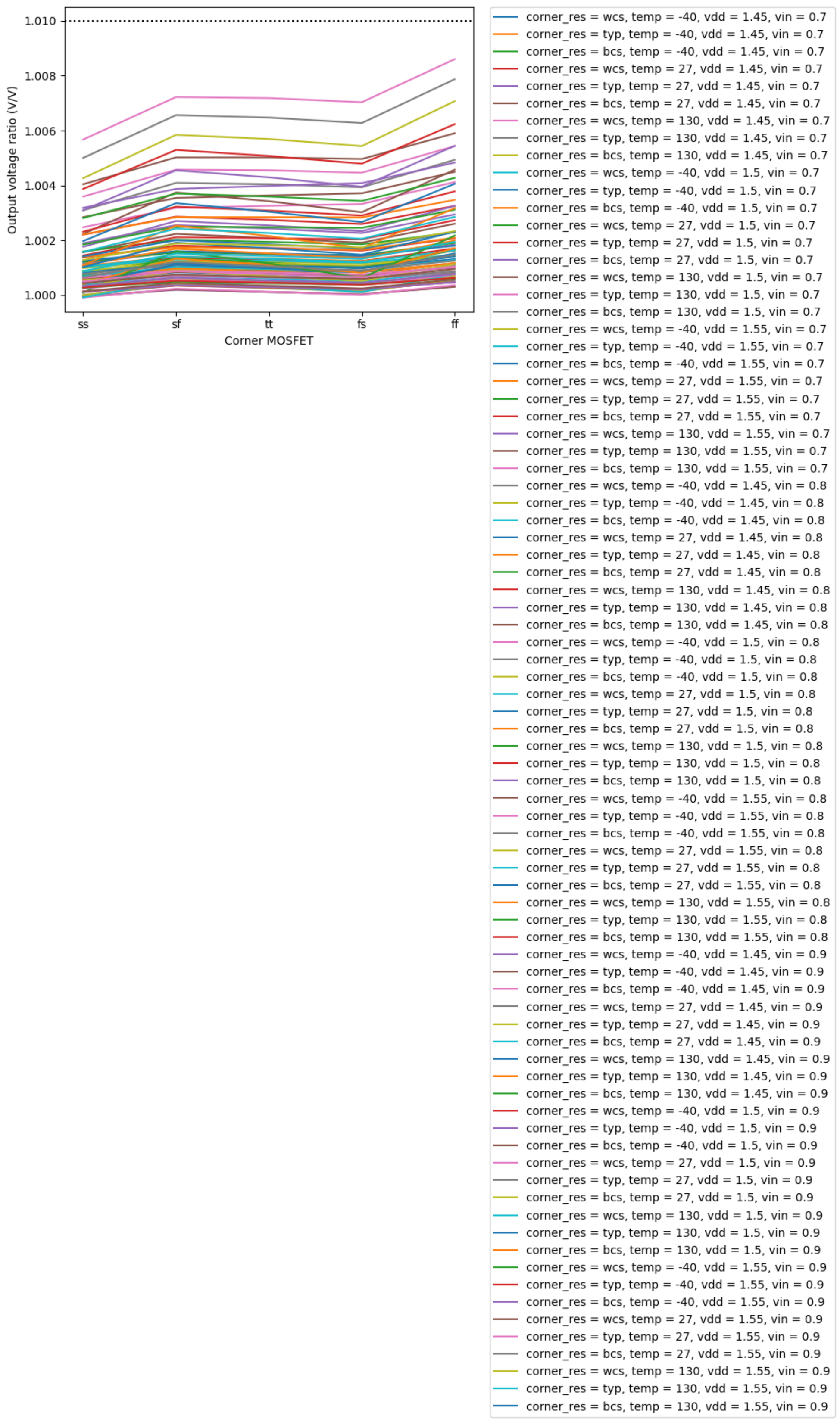


Figure 54: gain_vs_corner

13.6 bw_vs_temp

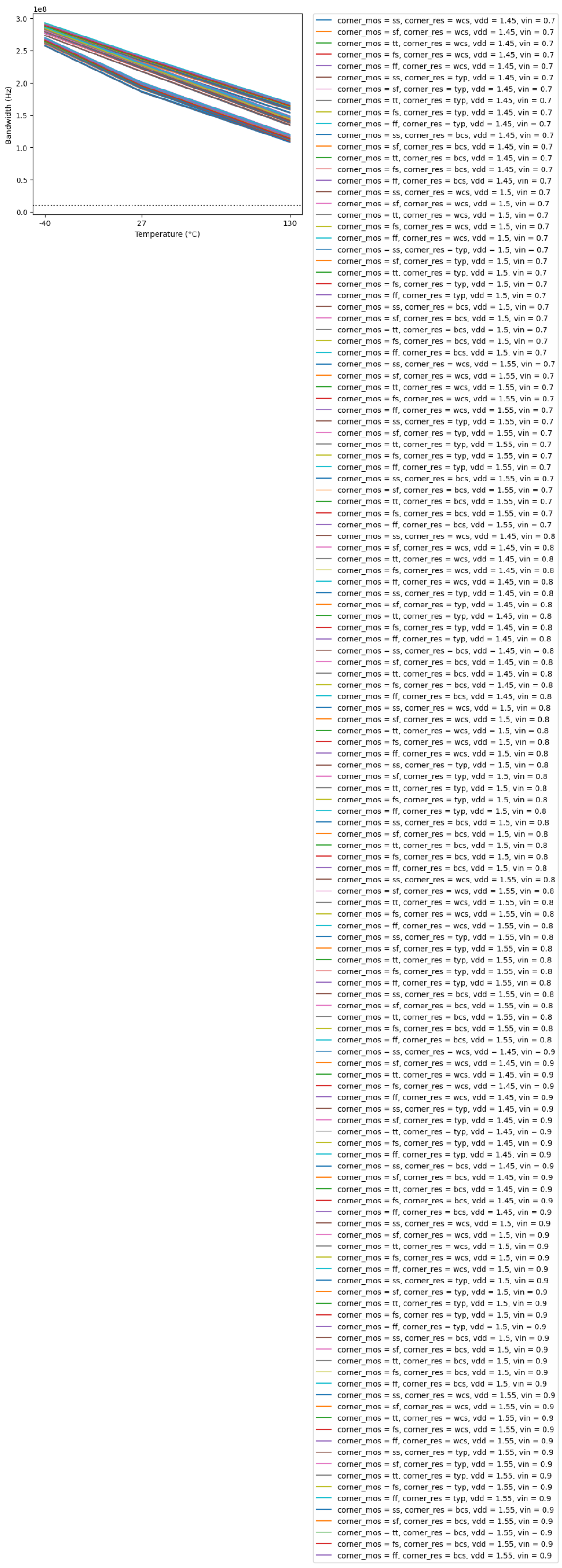


Figure 55: bw_vs_temp

13.7 bw_vs_vin

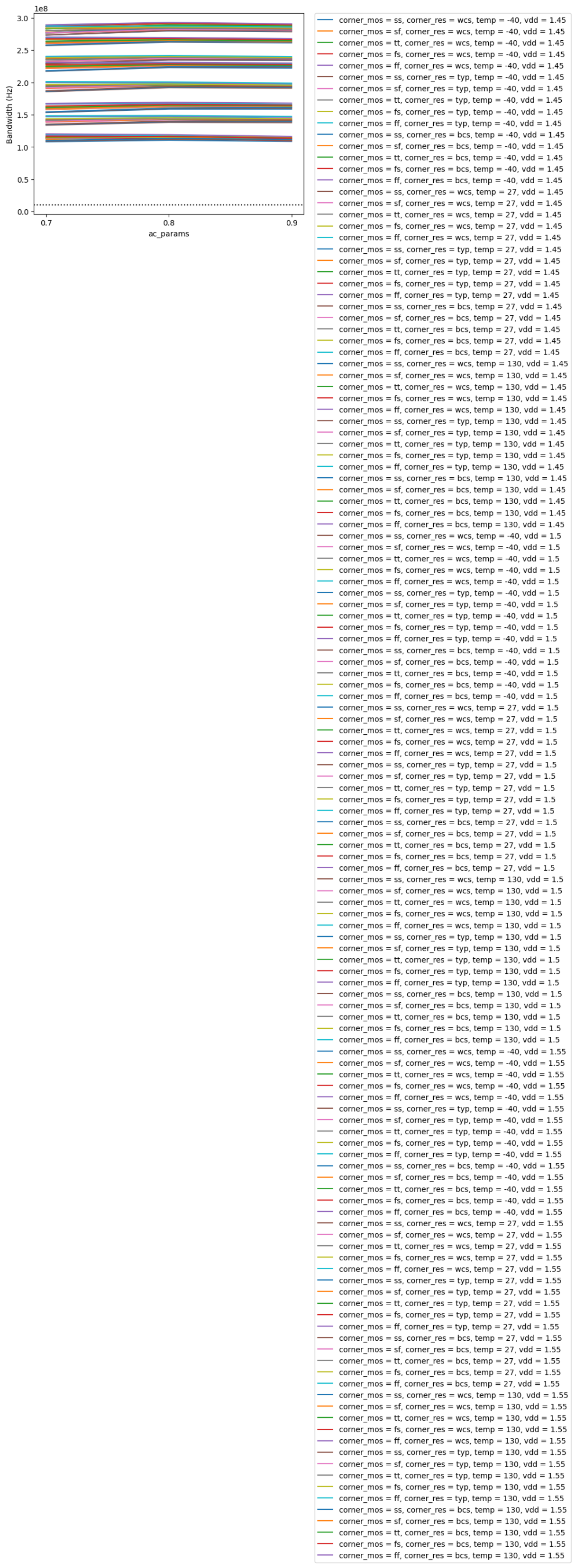


Figure 56: bw_vs_vin

13.8 bw_vs_vdd

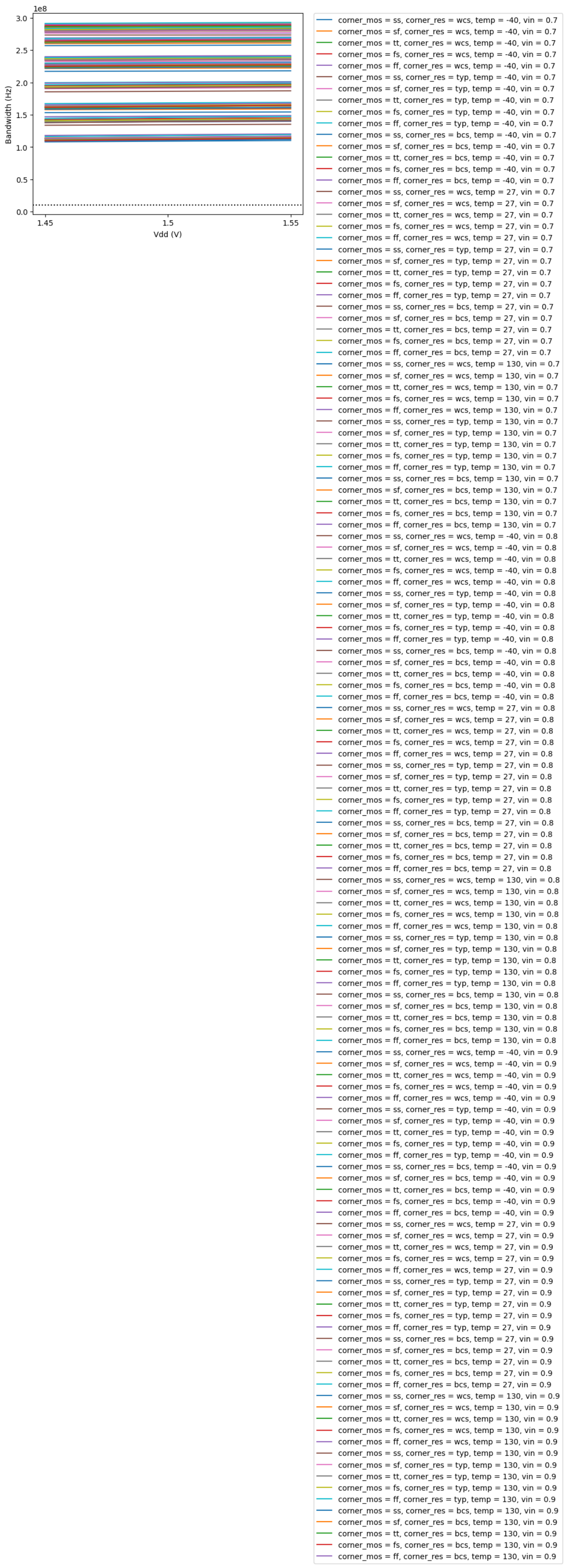


Figure 57: bw_vs_vdd

13.9 bw_vs_corner

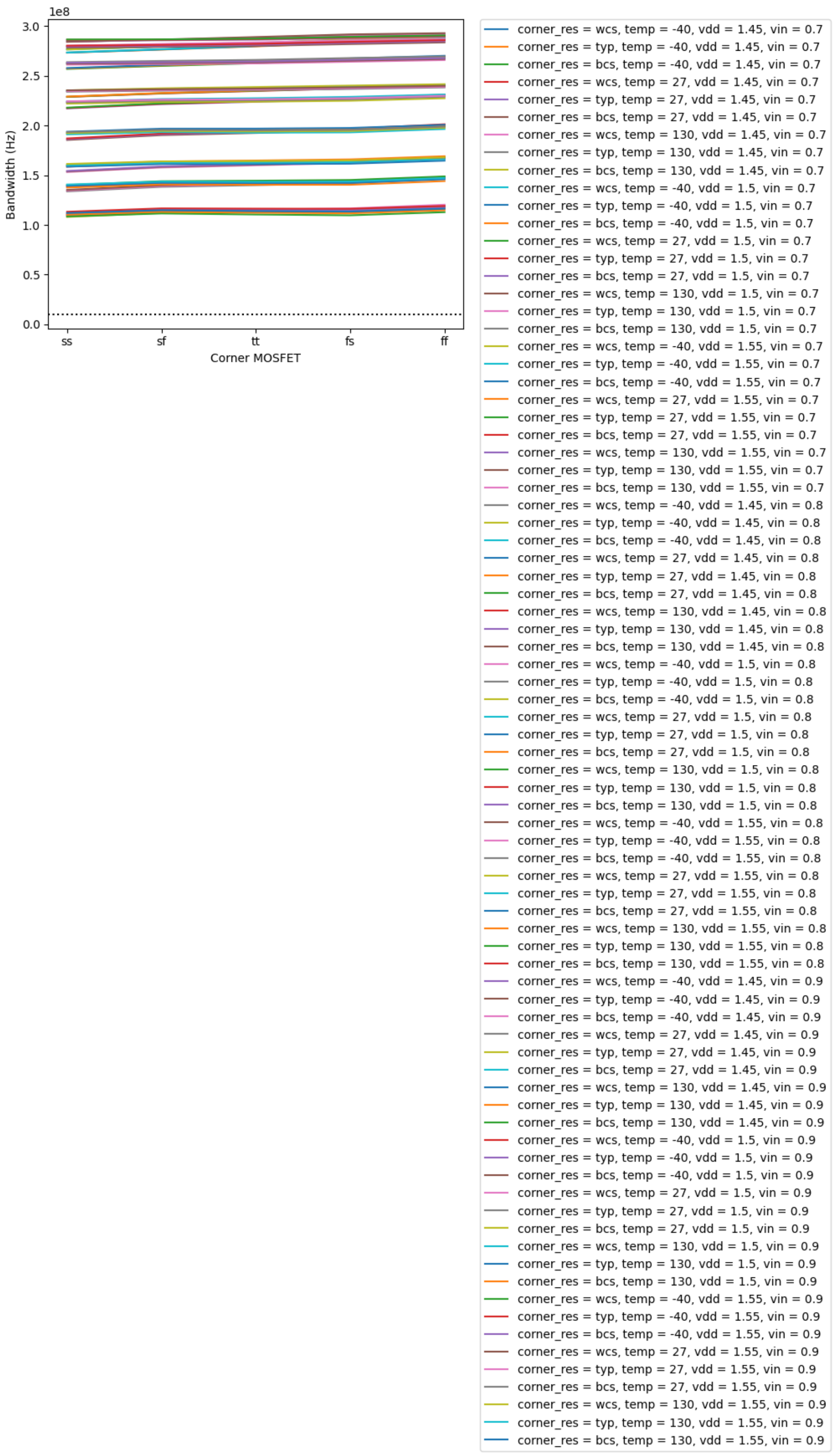


Figure 58: bw_vs_corner

13.10 noise_vs_temp

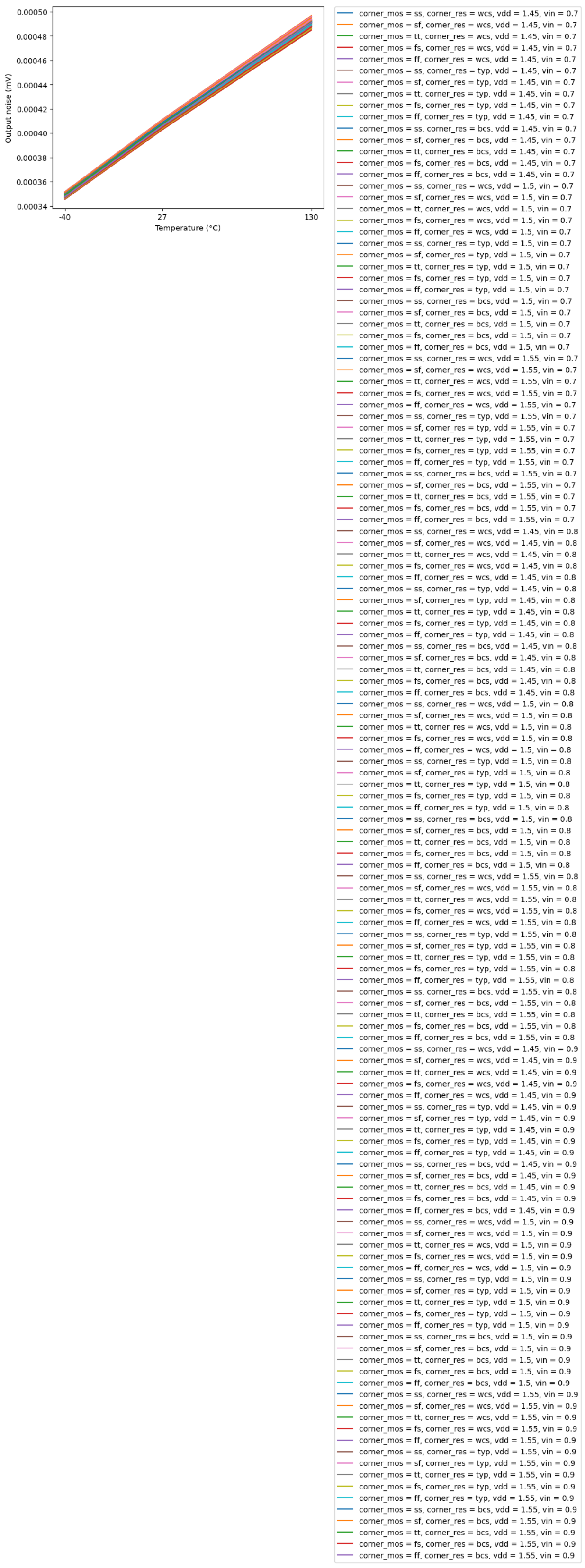


Figure 59: noise_vs_temp

13.11 noise_vs_vin

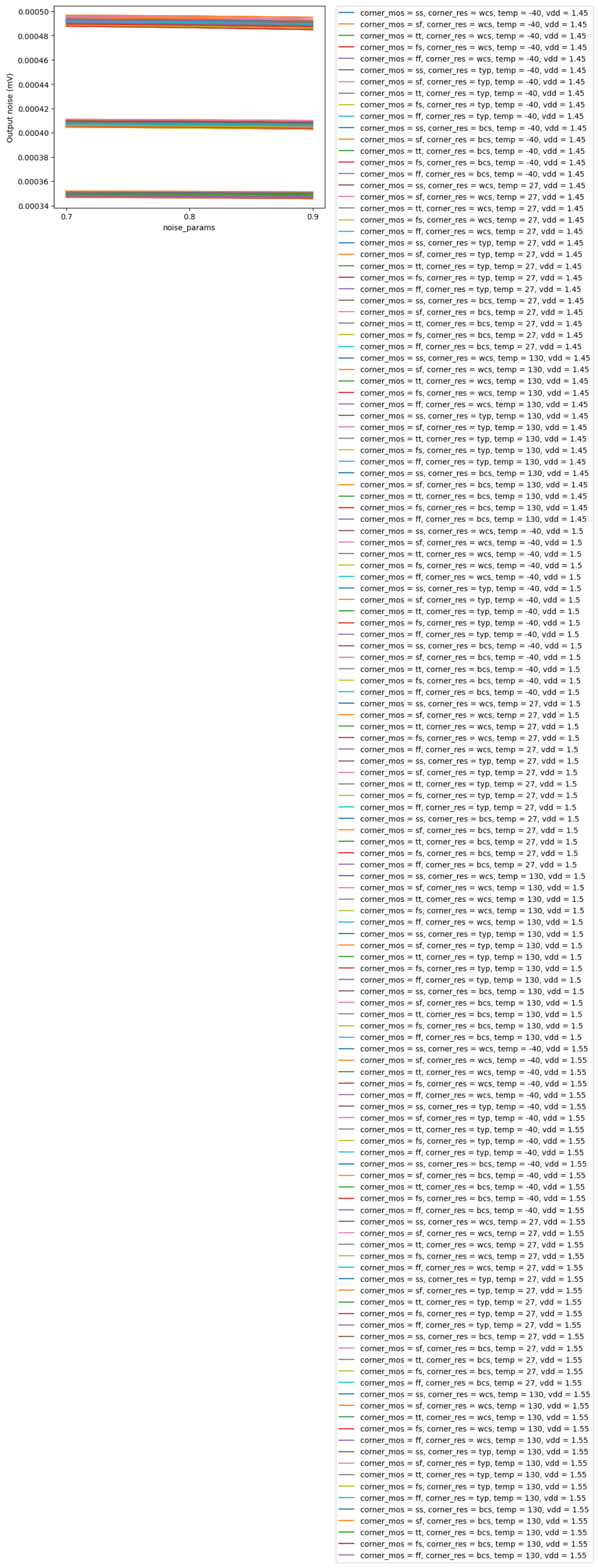


Figure 60: noise_vs_vin

13.12 noise_vs_vdd

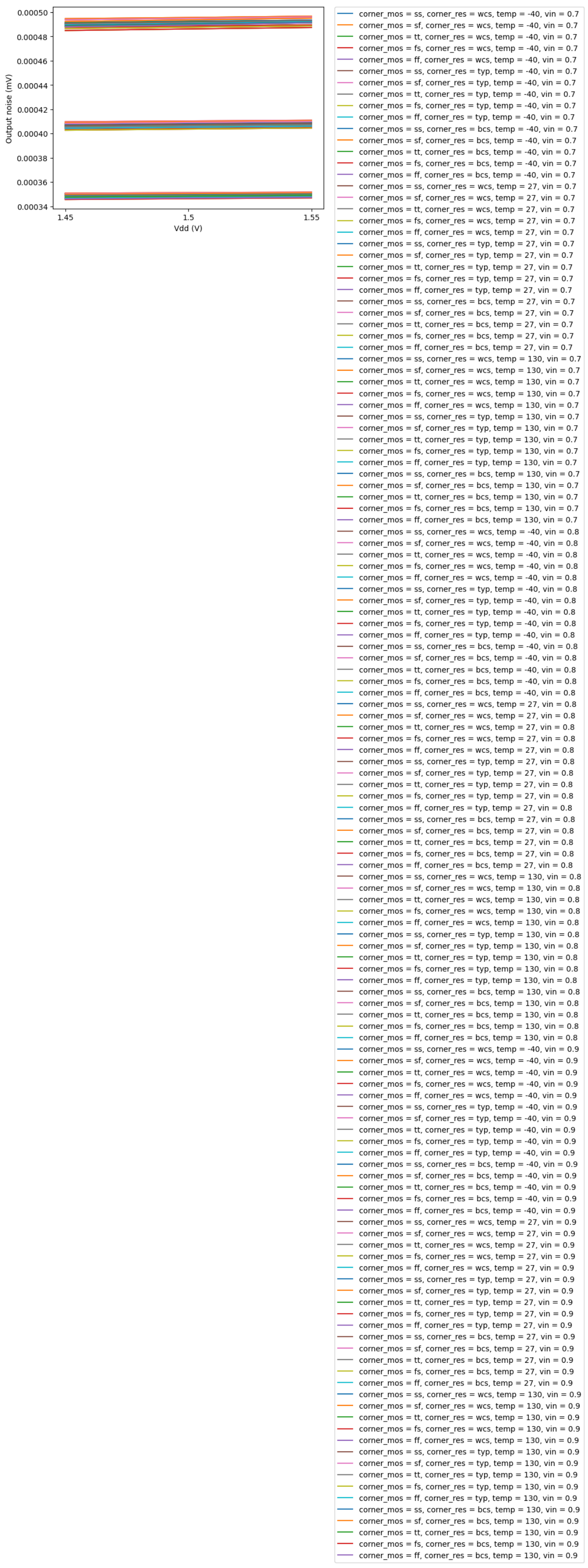


Figure 61: noise_vs_vdd

13.13 noise_vs_corner

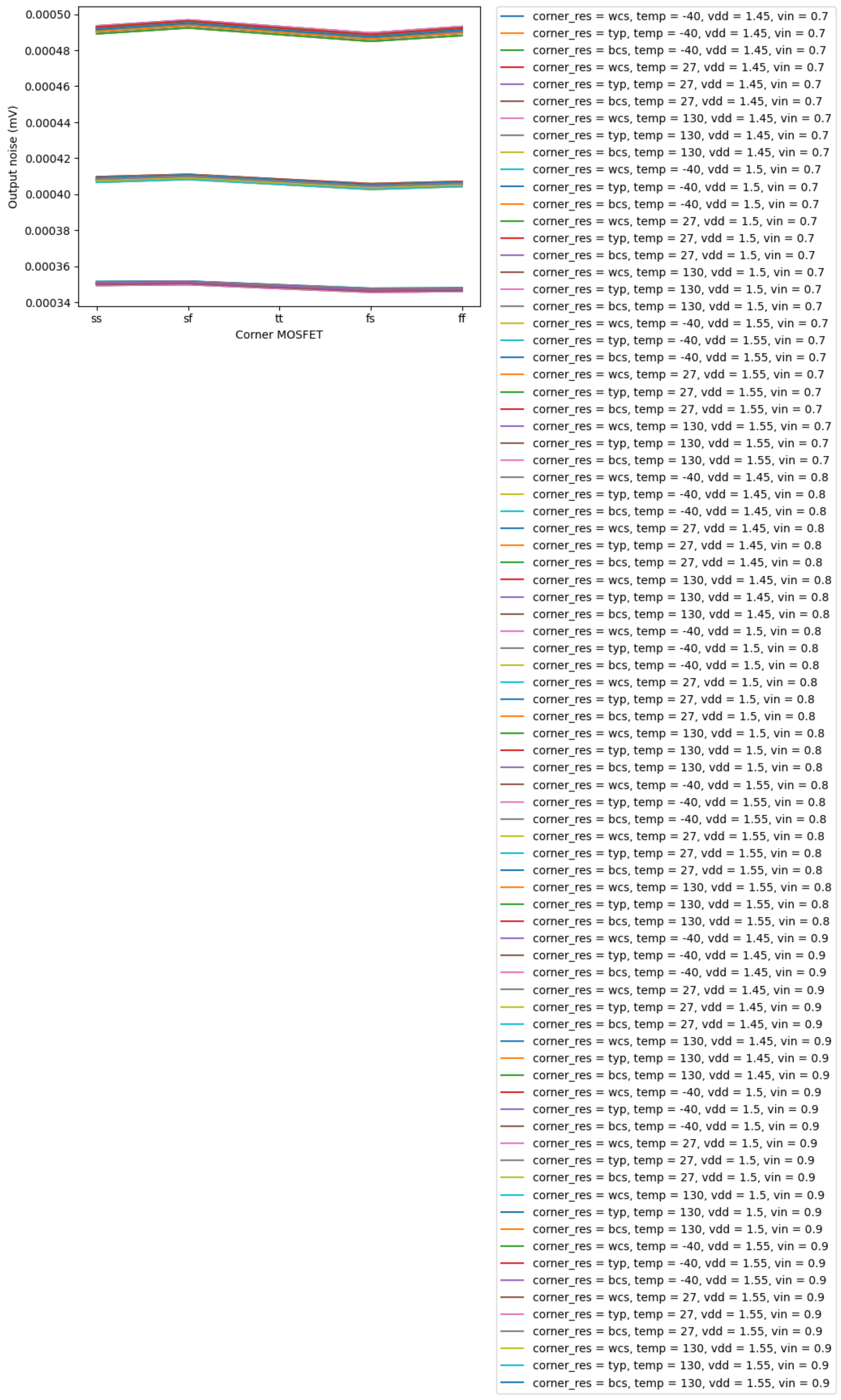


Figure 62: noise_vs_corner

13.14 settling_vs_temp

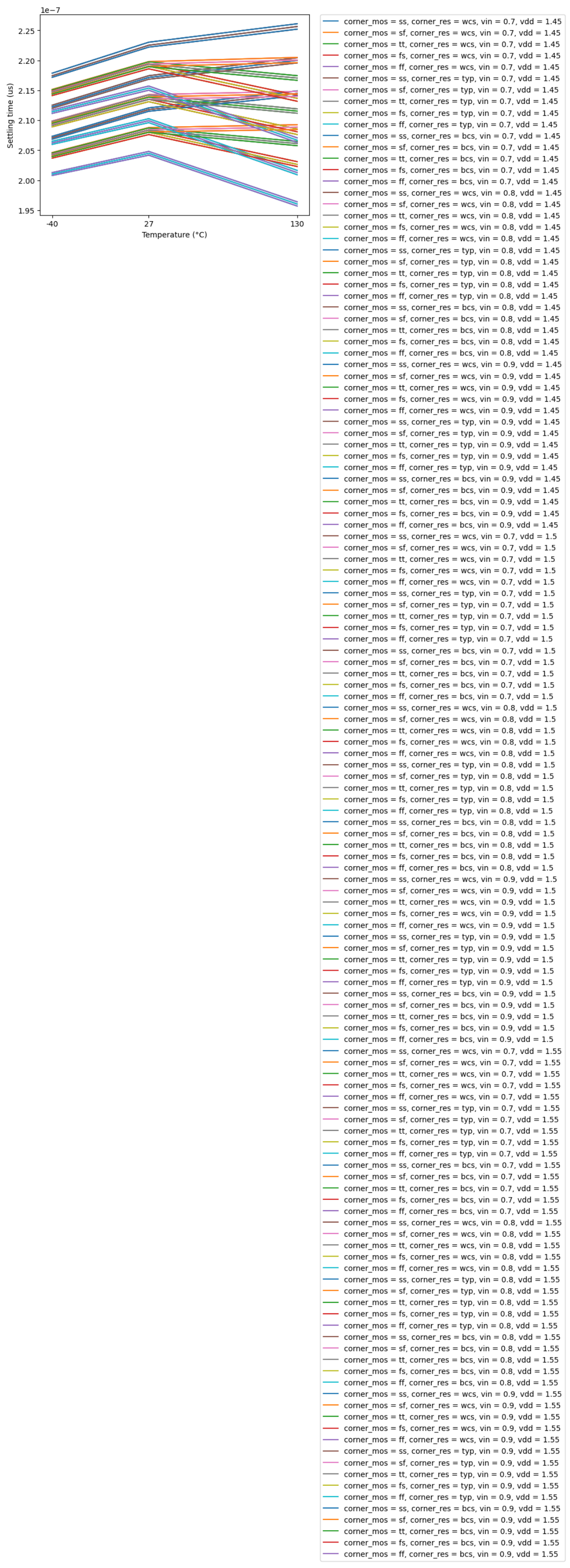


Figure 63: settling_vs_temp

13.15 settling_vs_vin

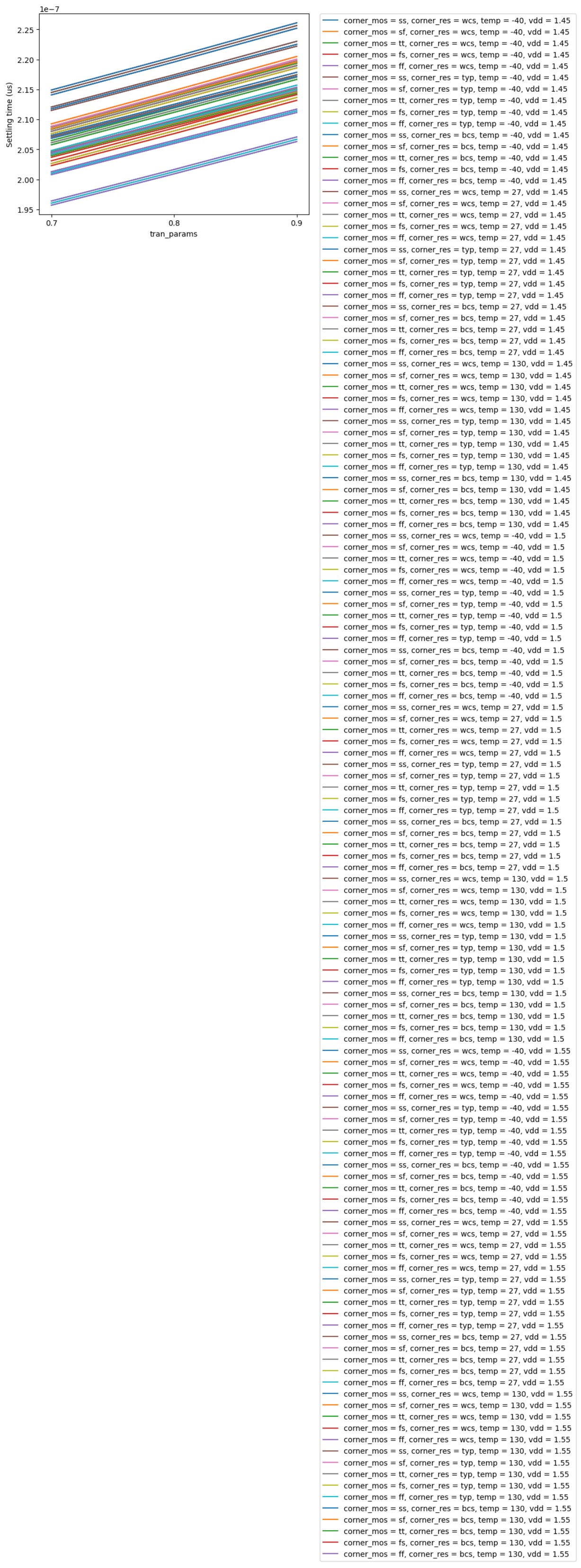


Figure 64: settling_vs_vin

13.16 settling_vs_vdd

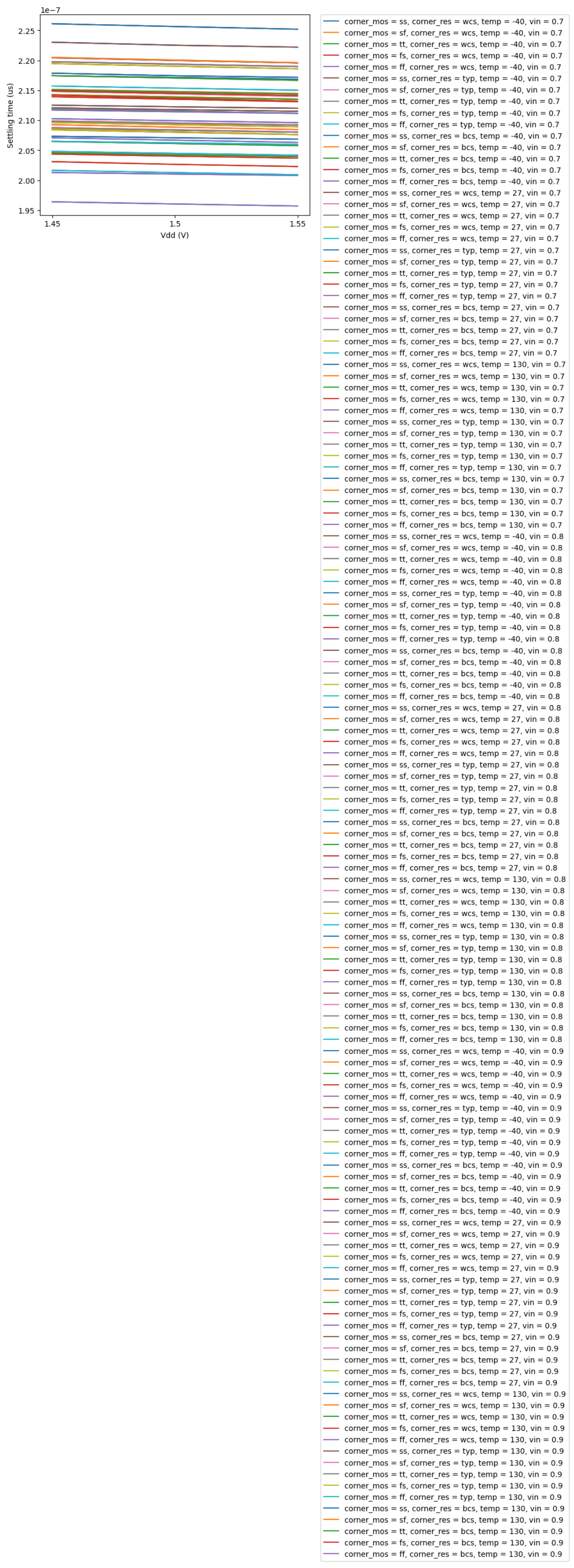


Figure 65: settling_vs_vdd

13.17 settling_vs_corner

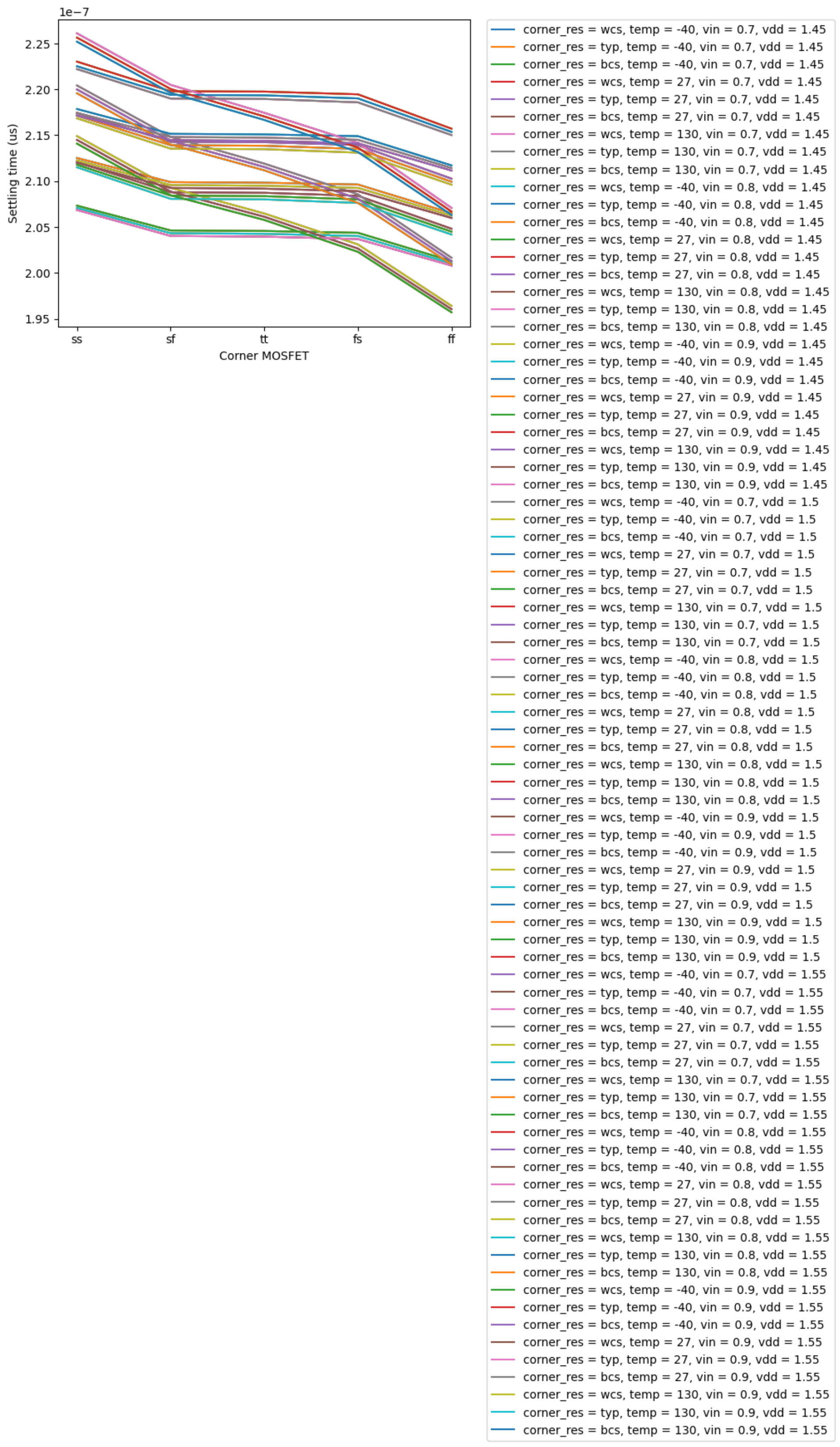


Figure 66: settling_vs_corner

The improved performance allows to improve the specificatios in a few important points, notably the output voltage tolerance which is an important metric for a reference voltage buffer. We have intetionally increased the power consumption a little bit, but we negotiated with the chip lead designer a changed bias current level, so overall the situation is even slightly improved. The new situation with the improved design is summarized in Table 5 (unchanged entries are not shown).

Table 5: Voltage buffer specification

Specification	Basic 5T-OTA	Improved OTA	Unit
Output voltage error	< 3	< 1	%
Total output noise (rms)	< 1	< 0.5	$\mathrm{mV_{rms}}$
Supply current (as low as possible)	< 10	< 20	µA
Turn-on time (settled to with 1%)	< 10	< 1	µs
Externally provided bias current (nominal)	20	5	µA

14 A Fully-Differential OTA

To be added in a future release.

15 Biasing the OTA

To be added in a future release.

16 An RC-OPAMP Filter

To be added in a future release.

17 Summary & Conclusion

By now, you should be familiar with the use of a schematic entry tool (Xschem) and circuit simulator (ngspice). You have learned the basic performance trade-offs, and the large- and small-signal behviour of the MOSFET. You can use the $g_\mathrm{m}/I_\mathrm{D}$ method to size MOSFET for class-A operation. You can design simple amplifiers based on OTA structures. In summary, you are on a good way to become a good analog or mixed-signal circuit designer!

! Feedback

We hope you have enjoyed these lecture notes! If you have feedback, suggestions, additions, or corrections, please send us an e-mail, create a GitHub issue, or provide a GitHub pull request. Thank you in advance for your contributions!

18 Appendix: Middlebrook's Method

When we want to do a closed-loop gain analysis (for stability or other investigations), we have the need to break the loop at one point, apply a stimulus, and monitor the response on the other end. By doing this we want to keep the loading on both ends similar to the original case. To achieve this, we break the loop at one point by inserting (1) an ac voltage source, and (2) attach an ac current source, as shown in Figure 67 and Figure 68. The derivation of this approach is presented in (Middlebrook 1975), and has the big advantage that loading is not changed, and the bias points are also correct.

Source: Article Notebook

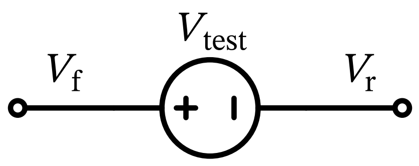


Figure 67: Middlebrook voltage loop gain simulation.

Source: Article Notebook

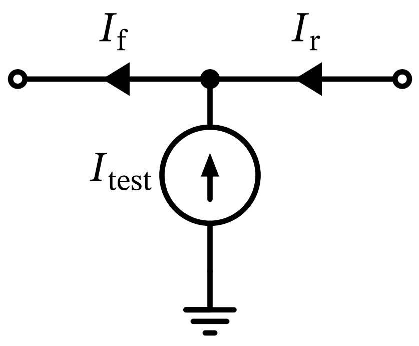


Figure 68: Middlebrook current loop gain simulation.

Source: Article Notebook

For both cases we do an ac analysis, and find the corresponding transfer functions T_v and T_i as

$$T_\mathrm{v} = -\frac{V_\mathrm{r}}{V_\mathrm{f}}$$

and

$$T_\mathrm{i} = -\frac{I_\mathrm{r}}{I_\mathrm{f}}.$$

Then, we can calculate the closed-loop transfer function $T(s) = H_{\mathrm{ol(s)}}$ as

$$T(s) = \frac{T_\mathrm{v}T_\mathrm{i} - 1}{T_\mathrm{v} + T_\mathrm{i} + 2}.$$

19 Appendix: Miller's Theorem

Using Miller's theorem we can find the equivalent circuit of an impedance connected between two nodes, and we know the transfer function between these nodes. The given situation is shown in Figure 69, and the equivalent circuit is shown in Figure 70.

Source: Article Notebook

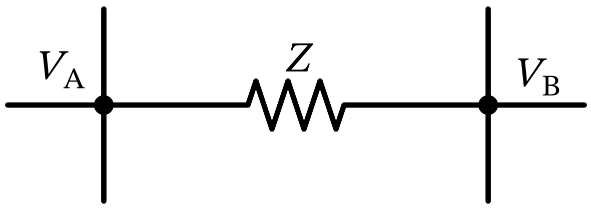


Figure 69: An impedance connected between two nodes A and B.

Source: Article Notebook

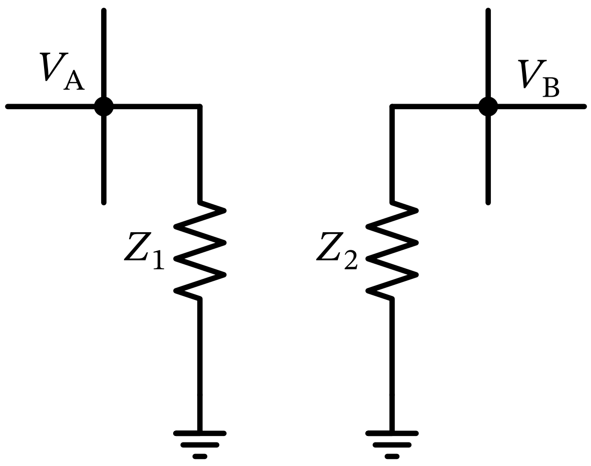


Figure 70: An equivalent circuit using Miller's theorem.

Source: Article Notebook

Using Miller's theorem (Sheikholeslami 2015) we can calculate

$$Z_1 = \frac{Z}{1 - A} = \frac{Z}{1 - V_\mathrm{B}/V_\mathrm{A}}$$

and

$$Z_2 = \frac{Z}{1 - A^{-1}} = \frac{Z}{1 - V_\mathrm{A}/V_\mathrm{B}}$$

to arrive at an equivalent circuit, given that $A = V_\mathrm{B}/V_\mathrm{A}$ is the voltage gain between nodes A and B. A derivation of this theorem is relative straightforward considering the current through Z when looking into the impedance from either node A or node B and calculating an equivalent impedance causing the same current.

Note that if $V_\mathrm{A} = V_\mathrm{B}$ then there is no current flow through Z, and accordingly the impedances $Z_1 = Z_2 = \infty$.

Miller's theorem can be quite handy when an impedance is strapped between two nodes, and we want to break this connection in a calculation, e.g., considering the effect of C_GD in a MOSFET.

20 Appendix: 5T-OTA Small-Signal Output Impedance

This section gives additional details to the analysis presented in Section 7.3. Here we provide the full calculation of the output impedance/conductance of the 5T-OTA for frequencies below the dominant pole, i.e. we neglect any capacitors.

Source: Article Notebook

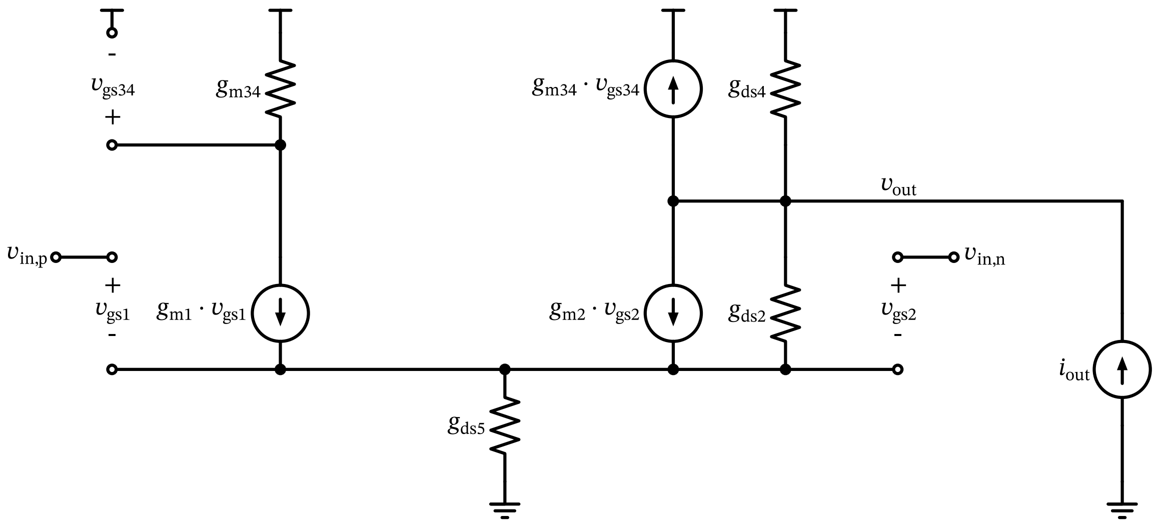


Figure 71: 5-transistor OTA small-signal model for output impedance calculations.

Source: Article Notebook

20.1 Open-Loop Configuration

For the open-loop case, the gates of M_1 and M_2 are tied to ground and thus, both v_{gs} are equal.

$$v_{in,p} = v_{in,p} = 0 \text{ V}$$
$$v_{gs1} = v_{gs2} \tag{26}$$

KCL at the output node:

$$i_{out} - g_{ds4}v_{out} - g_{m34}v_{gs34} - i_{g_{ds2}} - g_{m2}v_{gs2} = 0 \tag{27}$$

KCL at the tail node:

$$g_{m1}v_{gs1} + g_{m2}v_{gs2} + i_{g_{ds2}} + g_{ds5}v_{gs2} = 0$$

Using Equation 26 we can eliminate v_{gs1} and solve for $i_{g_{ds2}}$.

$$i_{g_{ds2}} = -\left(g_{m1} + g_{m2} + g_{ds5}\right)v_{gs2} \tag{28}$$

Furthermore, we need an expression for v_{gs34}. Ohm's law at the conductance g_{m34} will suffice.

$$v_{gs34} = -\frac{g_{m1}}{g_{m34}}v_{gs1} \tag{29}$$

KVL from the output node down to ground (over g_{ds2} and g_{ds5}) in combination with Equation 28 gives us an expression for v_{gs2}

$$v_{\mathrm{gs2}} = -\frac{g_{\mathrm{ds2}}}{g_{\mathrm{m1}} + g_{\mathrm{m2}} + g_{\mathrm{ds2}} + g_{\mathrm{ds5}}} v_{\mathrm{out}} \tag{30}$$

Now, we can plug in all quantities into Equation 27. First, Equation 28 is inserted, which provides an expression for the current through the output conductance g_{ds2} of M_2.

$$i_{\mathrm{out}} - g_{\mathrm{ds4}} v_{\mathrm{out}} - g_{\mathrm{m34}} v_{\mathrm{gs34}} + (g_{\mathrm{m1}} + g_{\mathrm{ds5}})\, v_{\mathrm{gs2}} = 0$$

Second, v_{gs34} is subsituted by Equation 29. Since we have assumed a matched pair of transistors for the current mirror comprised of M_3 and M_4, g_{m34} perfectly cancels out of the equation, and is effectively replaced by the transconductance g_{m1} of the input transistor M_1.

$$i_{\mathrm{out}} - g_{\mathrm{ds4}} v_{\mathrm{out}} + (2g_{\mathrm{m1}} + g_{\mathrm{ds5}})\, v_{\mathrm{gs2}} = 0$$

Third, Equation 30 gives as an expression for the last remaining unknown v_{gs2}. Thus, the factor in front of v_{out} defines the conductance at the output node.

$$i_{\mathrm{out}} - \left[g_{\mathrm{ds4}} + (2g_{\mathrm{m1}} + g_{\mathrm{ds5}}) \frac{g_{\mathrm{ds2}}}{g_{\mathrm{m1}} + g_{\mathrm{m2}} + g_{\mathrm{ds2}} + g_{\mathrm{ds5}}} \right] v_{\mathrm{out}} = 0 \tag{31}$$

Before, we interpret this result, we use are assumption of matched input transistors ($g_{\mathrm{m12}} = g_{\mathrm{m1}} = g_{\mathrm{m2}}$) and slightly rearrange the equation to give us more insight.

$$i_{\mathrm{out}} - \left[g_{\mathrm{ds4}} + \frac{g_{\mathrm{ds2}} \cdot (2g_{\mathrm{m12}} + g_{\mathrm{ds5}})}{g_{\mathrm{ds2}} + (2g_{\mathrm{m12}} + g_{\mathrm{ds5}})} \right] v_{\mathrm{out}} = 0 \tag{32}$$

Now, we can identify the common equation of the total resistance of two parallel resistors. However, we are dealing with conductances here, so the same equation describes the total conductance of two conductances in series, while parallel conductances are simply summed. In parallel to g_{ds4}, there is effectively the series connection of g_{ds2} and $(2g_{\mathrm{m12}} + g_{\mathrm{ds5}})$ at work. If we apply the general assumption of $g_{\mathrm{m}} \gg g_{\mathrm{ds}}$, only the parallel connection of g_{ds4} and g_{ds2} remains. Therefore, moving $g_{\mathrm{ds2}} + g_{\mathrm{ds4}}$ in parallel to C_{load} in Section 7.3 was valid.

$$\frac{i_{\mathrm{out}}}{v_{\mathrm{out}}} \approx g_{\mathrm{ds4}} + g_{\mathrm{ds2}} \tag{33}$$

20.2 Closed-Loop Configuration

In contrast to the open-loop case, we keep the gate of M_1 connected to ground and tie the input of M_2 to the output node v_{out}.

$$v_{\mathrm{in,n}} = v_{\mathrm{out}} \tag{34}$$

KCL at the output node:

$$i_{\mathrm{out}} - g_{\mathrm{ds4}} v_{\mathrm{out}} - g_{\mathrm{m34}} v_{\mathrm{gs34}} - g_{\mathrm{ds2}} v_{\mathrm{gs2}} - g_{\mathrm{m2}} v_{\mathrm{gs2}} = 0 \tag{35}$$

We use KVL from the output node down to ground to find an expression for v_{gs2}.

$$v_{\mathrm{gs2}} = v_{\mathrm{out}} + v_{\mathrm{gs1}} \tag{36}$$

KCL at the tail node:

$$g_{\mathrm{m1}} v_{\mathrm{gs1}} + g_{\mathrm{m2}} v_{\mathrm{gs2}} + g_{\mathrm{ds2}} v_{\mathrm{gs2}} + g_{\mathrm{ds5}} v_{\mathrm{gs2}} = 0 \tag{37}$$

Using Equation 36 to subsitite v_{gs2} in {#eq-app-vbufzout-kcl-vtail-cl} we find an equation for v_{gs1}.

$$v_{\mathrm{gs1}} = -\frac{g_{\mathrm{m2}} + g_{\mathrm{ds2}}}{g_{\mathrm{m1}} + g_{\mathrm{m2}} + g_{\mathrm{ds2}} + g_{\mathrm{ds5}}} v_{\mathrm{out}} \tag{38}$$

Again, we derive the output conductance by plugging Equation 36, Equation 29 and Equation 38 step by step into Equation 35. First, we use Equation 36 to eliminate v_{gs2}.

$$i_{\mathrm{out}} - (g_{\mathrm{ds4}} + g_{\mathrm{ds2}} + g_{\mathrm{m2}})\, v_{\mathrm{out}} - g_{\mathrm{m34}} v_{\mathrm{gs34}} - (g_{\mathrm{ds2}} + g_{\mathrm{m2}})\, v_{\mathrm{gs1}} = 0$$

Second, Equation 29 also holds for the closed-loop case and lets us eliminate v_{gs34}.

$$i_{\mathrm{out}} - (g_{\mathrm{ds4}} + g_{\mathrm{ds2}} + g_{\mathrm{m2}})\, v_{\mathrm{out}} - (g_{\mathrm{ds2}} + g_{\mathrm{m2}} - g_{\mathrm{m1}})\, v_{\mathrm{gs1}} = 0$$

Third, we use Equation 38 to eliminate the remaining unknown v_{gs1}.

$$i_{\mathrm{out}} - (g_{\mathrm{ds4}} + g_{\mathrm{ds2}} + g_{\mathrm{m2}})\, v_{\mathrm{out}} + (g_{\mathrm{ds2}} + g_{\mathrm{m2}} - g_{\mathrm{m1}}) \frac{g_{\mathrm{m2}} + g_{\mathrm{ds2}}}{g_{\mathrm{m1}} + g_{\mathrm{m2}} + g_{\mathrm{ds2}} + g_{\mathrm{ds5}}} v_{\mathrm{out}} = 0$$

A more simpler result can be obtained, if we neglect g_{ds2} and g_{ds5} in Equation 38 first ($g_{\mathrm{m}} \gg g_{\mathrm{ds}}$) and then plug it into our main equation. Additionally, we use $g_{\mathrm{m12}} = g_{\mathrm{m1}} = g_{\mathrm{m2}}$ to further simplify the equation.

$$i_{\mathrm{out}} - \left(g_{\mathrm{ds4}} + \frac{3}{2} g_{\mathrm{ds2}} + g_{\mathrm{m12}} \right) v_{\mathrm{out}} \approx 0$$

If we apply $g_{\mathrm{m}} \gg g_{\mathrm{ds}}$ again, we arrive at the same result which was used for the noise calculation in Section 7.3, compare the expression for Y'_{load} given by Equation 19 .

$$i_{\mathrm{out}} - (g_{\mathrm{m12}})\, v_{\mathrm{out}} \approx 0$$

21 Appendix: ngspice Cheatsheet

Here is an unsorted list of useful ngspice settings and command:

21.0.1 Commands

- `ac dec|lin points fstart fstop` performs a small-signal ac analysis with either linear or decade sweep
- `dc sourcename vstart vstop vincr [src2 start2 stop2 incr2]` runs a dc-sweep, optionally across two variables
- `display` shows the available data vectors in the current plot
- `echo` can be used to display text, `$variable` or `$&vector`, can be useful for debugging
- `let name = expr` to create a new vector; `unlet vector` deletes a specified vector; access vector data with `$&vec`
- `linearize vec` linearizes a vector on an equidistant time scale, do this before an FFT; with `set specwindow=windowtype` a proper windowing function can be set
- `meas` can be used for various evaluations of measurement results (see ngspice manual for details)
- `noise v(output <ref>) src (dec|lin) pts fstart fstop` runs a small-signal noise analysis
- `op` calculates the operating point, useful for checking bias points and device parameters
- `plot expr vs scale` to plot something
- `print expr` to print it, use `print all` to print everything
- `remzerovec` can be useful to remove vectors with zero length, which otherwise cause issues when saving or plotting data
- `rusage` plot information about resource usage like memory
- `save all` or `save signal` specifies which data is saved during simulation; this lowers RAM usage during simulation and size of RAW file; do save before the actual simulation statement
- `setplot` show a list of available plots
- `set var = value` to set the value of a variable; use variable with `$var`; `unset var` removes a variable
- `set enable_noisy_r` to enable noise of behavioral resistors; usually, this is a good idea
- `shell cmd` to run a shell command
- `show : param`, like `show : gm` shows the g_m of all devices after running an operating point with `op`
- `spec` plots a spectrum (i.e. frequency domain plot)
- `status` shows the saved parameters and nodes
- `tf` runs a transfer function analysis, returning transfer function, input and output resistance
- `tran tstep tstop <tstart <tmax>>` runs a transient analysis until `tstop`, reporting results with `tstep` stepsize, starting to plot at `tstart` and performs time steps not larger then `tmax`
- `wrdata` writes data into a file in a tabular ASCII format; easy to further process
- `write` writes simulation data (the saved nodes) into a RAW file; default is binary, can be changed to ASCII with `set filetype=ascii`; with `set appendwrite` data is added to an existing file

21.0.2 Options

Use `option option=val option=val` to set various options; important ones are:

- `abstol` sets the absolute current error tolerance (default is 1pA)
- `gmin` is the conductance applied at every node for convergence improvement (default is 1e-12); this can be critical for very high impedance circuits
- `klu` sets the KLU matrix solver
- `list` print the summary listing of the input data
- `maxord` sets the numerical order of the integration method (default is 2 for Gear)
- `method` set the numerical integration method to `gear` or `trap` (default is `trap`)
- `node` prints the node table
- `opts` prints the option values
- `temp` sets the simulation temperature
- `reltol` set the relative error tolerance (default is 0.001 = 0.1%)
- `savecurrents` saves the terminal currents of all devices
- `sparse` sets the sparse matrix solver, which can run noise analysis, but is slower than `klu`
- `vntol` sets the absolute voltage error tolerance (default is 1µV)
- `warn` enables the priting of the SOA warning messages

21.0.3 Convergence Helper

- `option gmin` can be used to increase the conductance applied at every node
- `option method=gear` can lead to improved convergence
- `.nodeset` can be used to specify initial node voltage guesses
- `.ic` can be used to set initial conditions

22 Appendix: Xschem Cheatsheet

When opening Xschem, using `Help -> Keys` a pop-up windows comes up with many useful shortcuts. The most useful are:

22.0.0.1 Moving around in a schematic:

- `Cursor keys` to move around
- `Ctrl-e` to go back to parent schematic
- `e` to descend into selected symbol
- `f` full zoom on schematic
- `Shift-z` to zoom in
- `Ctrl-z` to zoom out

22.0.0.2 Editing schematics:

- `Del` to delete elements
- `Ins` to insert elements from library
- `Escape` to abort an operation
- `Ctrl-#` to rename components with duplicate names
- `c` to copy elements
- `Alt-Shift-l` to add wire lable
- `Alt-l` to add lable pin
- `m` move selected objects
- `q` to edit properties
- `Ctrl-s` to save schematic
- `t` to place a text
- `Shift-T` to toggle the `ignore` flag on an instance
- `u` to undo an operation
- `w` to draw a wire
- `Shift-W` draw wire and snap to close pin or netpoint
- `&` to join, break, and collapse wires

22.0.0.3 Viewing/Simulating Schematics

- `5` to only view probes
- `k` to highlight selected net
- `Shift-K` to unhighlight all nets
- `Shift-o` to toggle light/dark color scheme
- `s` to run a simulation

23 Appendix: Circuit Designer's Etiquette

24 Circuit Designer's Etiquette

Harald Pretl, Institute for Integrated Circuits (IIC), Johannes Kepler University, Linz

Release: Spring 2024

24.1 Prolog

A consistent naming and schematic drawing style, as well as VHDL/Verilog coding scheme, is a huge help in avoiding errors and increasing productivity. Even if just one person works on a design, the error rate is lowered. If multiple persons work together in a team, a consistent working style is a big help for smooth cooperation without misunderstanding each other's intentions. Consistency also helps to reuse existing blocks. In a well-done design, the documentation is included in the schematic/source code, so there is no searching for a piece of documentation somewhere else (which is often not found anyway).

24.2 Pins

- Name package pins (interfacing with the outside the IC) in **UPPERCASE**, and all internal signals in **lowercase**.
- Supply voltages like VDD/VCC and ground like VSS/GND need to start with either `VDD`, `VCC`, `VSS`, `VEE` or `GND`, plus a suitable suffix. Examples: `VDD1`, `VDD_AMP`, `vdd_ldo_out`, `VSS_ANA` (uppercase means connected to a pin, lowercase means a VDD is created on-chip by, e.g., an LDO).
- Preferred are `VDD`/`VSS` for CMOS and `VCC`/`GND` for bipolar circuits. In BiCMOS circuits `VDD`/`VSS` are preferred, as usually, the digital content is the major part.
- Digital signals in an analog schematic should start with `di_` (for digital input) or `do_` (for digital output). Example: `di_ctrl1`. In the rare case of a bi-directional digital signal `dio_` can be used.
- Name digital signals consistently: `di_pon` is active-high, `di_pon_b` is active-low (`_b` standing for the negating “bar”); as an alternative, this last signal could be named `di_disable`. `di_reset` is an active-high reset, but often a reset is active-low, so it needs to be named `di_reset_b` (an alternative is `di_resetn`).
- In mixed-voltage designs, it might be useful to append the voltage level of a signal to avoid connecting incompatible inputs and outputs. Example: `do_comp_1v2` or `di_poweron_3v3`.
- Digital buses always have the MSB to the left and LSB to the right. Example: `do_adc[7:0]`.
- Analog signals should start with a `v` for a voltage signal or `i` for a current signal. It is often useful to include a value for bias signals or make the naming meaningful. RF signals, which are often neither voltage nor current signals, start the name with `rf_`. Examples: Signal and pin names like `ibias_30u` (30uA of bias current), `vbg_1v2` (a bandgap voltage of 1.2V), `vin_p`, `v_filt_out_n`, and `rf_lna_i` speak for themselves.
- Appending analog signals with `_i` and `_o` might be useful if a clear direction is obvious in the signal flow. If a signal is bi-directional, it is better to skip `_io`.
- Consistently use pin types `input`, `output`, or `inout` to indicate signal flow. Power supply pins are of `inout` type.

24.3 Schematics

- In analog schematics, add a textual note about basic circuit performance. For example, in an amplifier, note things like suitable supply range, typical and w.c. current consumption, gain, GBW, input voltage range, PSRR, and other useful information.
- If a circuit has a quirk or is particularly clever, add a note on how it works, so others can understand the function without excessive analysis (reviewing a circuit should not be a brain teaser).

- Use provided borders or drawing templates for schematics, and fill the data in, like circuit designer name, date, change history, project name, etc.
- Use a versioning system for your data, and check in often. This avoids data loss, and going back to an earlier design stage is simple. `SVN` is often preferable to `GIT` for binary data.
- Draw uncluttered clear circuits. Ideally, the circuit function is apparent by inspection **quickly**. Everyone can obscure an inverter so that it takes 5 minutes to recognize it, but this is not a good design.
- Don't alter the standard grid setting while drawing schematics (also make sure that the pins in your drawn symbols are on the standard grid)! Off-grid schematic elements will haunt you and your colleagues forever!
- Once a schematic is finished, take the time to name component instances properly (you can use speaking names like `Rstab` or simply use `R1`, `R2`, etc.). Use iterated instances to clean up the circuit. Use wire bundles to clean up circuits where useful. A clever technique is to use bundles and iterated instances to efficiently draw large resistor ladders, for example (however, use with care).
- Avoid connection-by-name, as it makes the circuit hard to read. However, there is a fine line to not cluttering circuits. Signals with many connections (`vdd`, `vss`, `pon`, `pon_b`) are often better done with connection-by-name instead of drawing a wire.
- Some tools allow the use of colored wires, which might be used to mark signal paths, bias lines, etc. However, this should not be overdone; use it with care.
- If you add auxiliary elements like current probes, ensure they get proper treatment when creating the netlist for the LVS (some elements should be shorted, and some elements simply taken out). Ideally, only use a single schematic for simulation, LVS, etc. By using tool features this can usually be done, and avoids the need to keep multiple schematics of one block in sync.
- Use annotations in the schematics to (1) denote current levels in branches, (2) denote bias voltage levels, (3) explain the function of logic input signals, and (4) put in logic tables if not obvious.
- Add comments concerning the layout, like matching devices, certain considerations of placement, sensitive nodes, etc.
- Add simple ASCII diagrams for timing signals if useful.
- Name internal signals (signals connected to pins are anyway named like the port) in a meaningful way; this makes tracking signals in simulation or layout much easier (automatic net names like `net0032` are of not much help).
- Properly name instances, not just `I1` or `I2`; better is `amp1`, `inv2`, etc. (a descriptor in a tool output like `I1/I13/I5/net017` is not helpful; compare that to `adc1/bias/bg/vref_int`).
- On check-and-save, never ignore warnings; just fix them! They will annoy you and others forever and might flag critical design flaws.
- Name cells interpretably, ideally making the function clear already by the name. It is often useful to prefix or postfix a cell by the project name and design iteration. Example: In the project `GIGAPROJECT`, the cells which are changed in the second design step are prefixed with `g2_`, like `g2_amp_bias`. Of course, more letters as a project abbreviation are useful if a name collision is likely to happen.
- Cell names in lowercase are a good choice, as otherwise, capitalization leads to

inconsistency in cell names. Use `_` to break words instead of `CamelCase`, like `amp_bias_startup`.

- When building a design, start with the hierarchy first; plan a suitable design structure, and define all interfaces. Implement simple behavioral models for every circuit block (either with controlled sources or using Verilog-A or VHDL/Verilog digital models). In this way, you can simulate the overall design early and find issues in the hierarchy or the interconnects. Then, populate the hierarchy with the detailed circuit designs in the leaf cells. At each point in the design process, you have a design that can be simulated, with some blocks as behavioral models and some blocks already designed. Try to avoid scattered circuit elements (digital or analog) in the hierarchy; it is better to push all components into the leaf cells.
- Avoid huge schematics, better break them down into smaller, maintainable, and self-contained blocks, and provide a simulation test bench for these simple blocks. In this way, later re-simulation across the hierarchy is easily possible.
- When building up the hierarchy, choose pin names and signal names as consistently as possible. Example: use the signal name `vref_int` when connecting two leaf cells with the pin names `vref_int_o` and `vref_int_i`.
- Avoid the excessive use of net breakers like small resistors, as they inhibit net tracing and can lead to simulation convergence issues. If a net breaker is needed (or a current should be probed) use a 0V dc voltage source.

24.4 Symbols

- Spend time drawing nice symbols! Ideally, the underlying circuit functionality is apparent by just looking at the symbol.
- Arrange the pins in a meaningful way.
- Group pins that belong together. An often useful arrangement is to locate the inputs on the left side, outputs on the right, digital control inputs at the bottom, and supplies at the top.
- Make the origin of a symbol in the top-left corner. In this way, symbols can be changed more easily, for example, by swapping out different versions of blocks.
- The cell name (and potentially library name) should be visible in the symbol, not only in the properties.

24.5 Design Robustness

- It is good practice to buffer incoming digital signals with a local inverter (connected to the local block supply) before connecting it to internal nodes. This improves the slew rate of the control signal and lowers the chance of unwanted cross-talk.
- Consider dummy elements for good matching, and try to make useful unit sizes of components. This will make the layout creation much smoother.
- The golden rule of good analog performance is good matching, and good matching is achieved by identical components (size, orientation, surroundings)! If the layout does not look nice (humans like symmetry), it will not perform well.

- Consider supply decoupling and bias voltage decoupling inside the cells. Often, dummy elements can be used for that. Be aware, however, of unwanted supply resonances (think bond wire `L` and decoupling `C`) and slow transients of bias nodes after disturbance.
- Always implement a proper power-down mode. Avoid floating nodes in off-mode. The better defined the on- as well as the off-mode are, the less the chance of leakage currents. Always simulate both modes (on and off), and also simulate a transient power-up of a circuit to identify issues with slow bias start or insufficient turn-off, or nasty feedback loop instabilities during transients.
- When drawing the first schematic, add parasitic capacitances to each node. If all nodes are labeled, a capacitor bank is easily put into one corner of the schematic with parasitic caps tied to the ground. Use 5fF as a starting value (and replace it later with the correct value from parasitic extraction). This accounts for some wiring parasitics in layout and helps to account for these layout impairments early in the design phase and later when simulating the schematic instead of the extracted netlist with parasitics.

24.6 Rules for Good Mixed-Signal and RF Circuits

- Separate analog and digital power supply, connect to package pins with multiple bond wires/bumps, and separate noisy and clean `vdd`/`vss` from each other!
- Prevent supply loops; keep `vdd` and `vss` lines close to each other (incl. bond wires and PCB traces)! This minimizes `L` and coupling factor `k`.
- Some prefer a massive (punched) ground plane, which is possible if you have enough metal levels. With a ground plane, the return path of a signal or supply line is just a few microns away.
- Use chip-internal decoupling capacitors, and decouple bias voltages to the **correct** potential (`vdd` or `vss`, or another node, depending on the circuit)!
- Use substrate contacts and guard rings to lower substrate crosstalk but use a quiet potential for connection; use triple-well if available! Connecting a guard-ring/substrate contact to a noisy supply is a prime noise injector (usually unwanted).
- Physically separate quiet and noisy circuits (at least by the epi thickness)!
- Reduce circuit noise generation as much as possible (avoid switching circuits if possible, use constant-current circuits instead, and use series/shunt regulators for supply isolation).
- Reduce sensitivity of circuits to interference (by using a fully differential design with high PSRR/CMRR, symmetrical layout parasitics, and good matching)!

24.7 VHDL/Verilog Coding Guide

These recommendations are specifically targeted at Verilog; however, they apply similarly to VHDL.

- Use automatic checkers (linters) to see whether your code contains errors or vulnerabilities. Commercial or open-source tools allow this, e.g., Icarus Verilog (`iverilog -g2005 -tnull FILE.v`) or Verilator (`verilator --lint-only -Wall FILE.v`).
- Write readable and maintainable code; use speaking variable names, and use a naming convention for inputs (beginning with `i_`) and outputs (beginning with `o_`). Active-low signals have an `_n` or `_b` in their name, like `i_reset_n`. Use comments to explain the intention.
- With a synchronous reset reset-related racing conditions are often avoided. If an asynchronous reset is desirable (which is often the case), ensure the reset signals are free from race conditions.
- Module-local registers and wires could append `_w` (for Verilog `wire`) or `_r` (for Verilog `reg`) to make their function clear. This is not required in SystemVerilog where the unified type `logic` should be used.
- Use an `assign` statement for logic as this often is easier to read than an `always @(*)` block. The ternary operator `COND ? TRUE : FALSE` can help with conditional assignments and is often a better choice than a (nested) `if ... else` statement.
- Declare all outputs explicitly with either `reg` or `wire`.
- Use local parameter definitions with `localparam` in a module to make the code easier to follow. Name parameters in **UPPERCASE**.
- Take care to reset all registers to a defined state (in simulation and HW).
- Use the rule of "one file per module." The filename shall match the module declaration.
- Use `` `default_nettype none `` at the beginning of a file containing a module definition. After the module you can use `` `default_nettype wire ``. This will add a safety net against typos in signal names.
- In a logic assign block, use `assign @(*) begin ... end` instead of spelling out the signals in the sensitivity list. Forgetting a signal could lead to serious mismatches between simulation and HW.
- Make your code flexible by making bit widths and other values parameterized using a `localparam` or module parameter.
- Be cautious of implicit type conversions and bit-width adaptions; better make explicit conversions and match bit widths in assignments.
- Use only blocking assignments (`=`) in `always @(*)` blocks, and only non-blocking assignments (`<=`) in clocked `always @(posedge ...)` blocks.

24.8 Further Reading

- Good information about drawing schematics, design testbenches, etc: https://circuit-artists.com
- Sutherland/Mills, *Verilog and SystemVerilog Gotchas - 101 Common Coding Erorrs and How to Avoid Them*, Springer, 2010
- B. Razavi, *The Analog Mind*, column in IEEE Solid-State Circuits Magazine

Source: Article Notebook

Hellen, Edward H. 2003. "Verifying the diode–capacitor circuit voltage decay." *American Journal of Physics* 71 (8): 797–800. https://doi.org/10.1119/1.1578070.

Hu, Chenming. 2010. *Modern Semiconductor Devices for Integrated Circuits*. Pearson.

Jespers, Paul G. A., and Boris Murmann. 2017. *Systematic Design of Analog CMOS Circuits: Using Pre-Computed Lookup Tables*. Cambridge University Press.

Middlebrook, R. D. 1975. "Measurement of loop gain in feedback systems." *International Journal of Electronics* 38 (4): 485–512. https://doi.org/10.1080/00207217508920421.

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.

Sheikholeslami, Ali. 2015. "Miller's Theorem [Circuit Intuitions]." *IEEE Solid-State Circuits Magazine* 7 (3): 9–10. https://doi.org/10.1109/mssc.2015.2446457.

Tsividis, Yannis, and Colin McAndrew. 2011. *Operation and Modeling of the MOS Transistor*. Oxford University Press.