

IC Design of a Universal Biquad Filter

Atakan Baydogan

Onno Kriens

Finn Ringelsiep

Alicia von Ahlen

2025-07-12

Table of contents

Abstract	1
TOC (only temporary)	3
New Final Final (no rEALLY) structure of doc	3
To Dos	4
TOC suggestion (Main)	4
Introduction	5
Motivation	5
Scope of the project	5
Specifications	6
Test	6
Open-Source	6
Hymne	7
1 Theoretical Background	9
1.1 Biquad filter	9
1.1.1 Universal biquad filter	9
1.1.2 Characteristics	10
1.2 Operational Amplifier	14
1.2.1 Voltage-Feedback Amplifiers (VFA)	15
1.2.2 Operational Transconductance Amplifier (OTA)	15
1.2.3 Variants ?	17
1.3 MOSFET (Metal-Oxide-Semiconductor Field-Effect-Transistor)	17
1.3.1 Small-Signal Representation	17
1.3.2 Sizing	17
1.3.3 Current Mirror	17
1.3.4 Differential Stage	17
2 Characterisation	19
2.1 Behavioural Analysis and macro modelling	19
2.1.1 Transfer Functions and frequency response	19
2.1.2 Stability	34
2.1.3 Ideal Opamp	42
2.2 Implementation (or Real Circuit)	49
2.2.1 used opamp representation	49
2.2.2 Sizing	50

Table of contents

3	Integrated Layouting	51
3.1	Introduction into the fundamentals of integrated layouting.	51
3.2	KLayout	53
3.2.1	The Setup of KLayout	55
3.3	The Layout Process	57
3.3.1	Working with LVS during layouting	59
3.3.2	Working with DRC during layouting.	60
3.4	The layout of the 5T-OTA	60
4	Filter Design	63
5	Discussion	65
5.1	Stability Analysis	65
5.2	Comparison	65
5.3	is filter good?	65
6	Conclusion	67
6.1	Outlook	67
	Bibliography	69

Abstract

! Important

Get rid of all these TOCs when done...

Keywords: Filter design, Biquadratic filter, IC design, Open Source Toolchain

- An abstract should provide a self-contained summary of your entire paper or research project.
- It aims to give readers a quick understanding of what you did, why it's important, and what you found.
- It should be a good starting point for anyone considering reading your full work.
- Introduction: Briefly introduce the topic and its significance.
- Research Question/Aim: State the problem you are addressing and the purpose of your research.
- Methodology: Briefly describe how you conducted your research (e.g., experiments, interviews, analysis).
- Key Findings: Summarize the main results or conclusions of your study.
- Significance/Implications: Explain why your findings are important and what they contribute to the field.
- Be concise: Use short, clear sentences and avoid unnecessary jargon.
- Use keywords: Include relevant keywords that will help readers find your abstract in databases.
- Follow formatting guidelines: Check for any specific requirements or guidelines for your specific journal or conference.
- Revise and edit: Make sure your abstract is clear, accurate, and within the word limit.
- Consider the audience: Tailor your language to your target audience (e.g., other researchers in your field, or a wider audience).

TOC (only temporary)

! Important

We need to go through the whole document and check whether opamp or ota is the right word in the context!

In the end we need to go through this, and change all second to last mentions of OP/operational amplifier to opamp

New Final Final (no rEALLY) structure of doc

Legend:

- finished (no more work needs to be done)
- review (peer review ongoing)
- editing (generally done, small todos like formatting, missing figure, links, etc)
- writing (currently under construction by fixed person responsible)
- planning (no text yet, but fixed person)
- missing (NOTHING, complete emptiness)

0 Abstract (missing)

1 Introduction

1.1 Motivation (review)

1.2 Scope of the Project (review)

1.3 Specifications (review)

1.4 Open Source (editing)

2 Theoretic Background 2.1 Biquad Filter 2.1.2 Universal Biquad Filter (editing) 2.2 ...Übergang von Biquad zu Mosfet über opamp/OTA (writing) 2.3 MOSFET (writing)

3 Characteristics (writing)

4 Implementation (writing)

5 Finns Layout (writing)

6 Diskussion (missing)

7 Conclusion and Outlook (missing)

TOC (only temporary)

To Dos

TOC suggestion (Main)

1 Abstract (max. 0,5 page)

2 Introduction (1 page)

2.1 Motivation

2.2 Scope

2.3 Constraints (Meiners Kram)

3 Theoretical Background

3.1 Opamp

3.1.1 Current Mirror

3.1.2 Diff Stage

3.1.2 Variants (like Miller maybe?) 3.1.3 Small signal representation 3.1.4 Stability 3.1.5 Sizing? 3.2 Biquad

3.2.1 Characteristics 3.2.2 Stability

4 Characterisation 4.1 Behavioural analysis 4.1.1 Python 4.1.2 Ideal Opamp? 4.2 Implementation (real circuit)

4.2.1 Used opamp representation 4.2.2 Sizing

(5 Design 5.1 methods 5.1.1 Xschem 5.1.2 ngspice 5.1.3 KLayout 5.2 Process) siehe finn

6 Discussion 6.1 Stability Analysis 6.2 Comparison 6.2.1 Python 6.2.2 Ideal 6.2.3 Real 6.3 is filter good?

7 Conclusion and outlook

Introduction

Warning

Check for reading flow please (for whole chapter)

This chapter provides a short motivation and overview for the IC design process done during the lecture “Analogue and Mixed-Signal Circuit Design” that Prof. Dr.-Ing. M. Meiners gives in the graduate course Electronic Engineering M.Sc. at City University of Applied Sciences Bremen.

The chapter will start with the motivation behind the biquad IC filter design, outline the scope of work of the project, and ends with giving concrete specifications for the implemented filter.

Motivation

The design and implementation of analog filters is a cornerstone in signal processing, with applications ranging from audio processing to communication systems. Among these, second order filters, like the biquad filter, are versatile building blocks due to its ability to realize four types of second order filters - low pass, high pass, band pass, and band stop. This project focuses on the integrated circuit (IC) design of a biquad filter, to get insight into the theoretical and practical engineering considerations behind IC design.

For a deeper understanding of IC design, this project does not rely on off-the-shelf operational amplifiers for the filter design, but aims to implement the entire filter architecture at the transistor level. This approach not only deepens the understanding of analog filter behavior but also introduces the challenges and intricacies of IC design, such as layout constraints, power efficiency, and stability.

This project demonstrates the design process of IC design from theoretical modelling, over simulation and design constraints to prototyping and to learn hands-on experience with tools used during the design process.

Scope of the project

The scope of this project is supposed to follow a real-world design flow, starting at a theoretical analysis of the specified filter and - in the best case - end in a tap-out of a prototype. If that stage is reached the prototype can be compared to the theoretical and simulation results obtained during the design process and checked for functionality.

As a tap-out of a prototype is fairly unrealistic in the time given, the goal is to simulate the specified filter with templates for operational amplifiers and base the IC layout on these templates.

Introduction

All in all, this project includes a systems analysis of the specified filter, simulation results with ideal components and real components, taken from provided templates, and a physical layout prototype.

Specifications

Warning

Not sure if this is written in the correct time!?!

The main objective is to design a universal biquad filter, based on the filter design proposed in the ASLK PRO Board Manual from Texas Instruments (Rao and Ravikumar 2012). The biquad filter shall have the following specifications:

$$f_0 = 1 \text{ kHz}$$

$$Q = 10$$

The circuit design is done in **Xschem** and the simulation in **ngspice**. For the design on transistor level the 130nm CMOS technology **SG13G2** is used. All these tools and PDK are integrated into a docker image **IIC-OSIC-TOOLS** (Pretl and Zachl 2025) provided by Prof. Dr. Harald Pretl from Johannes Kepler University.

This documentation provides a development report, which documents the design process with the taken steps and decisions made.

Test

Open-Source

All the results of this report and development approach to design a Biquad are publicly available on GitHub (LINK...). Everyone is invited and should feel free to use, change, and share this work. This whole course and project wouldn't be possible without the great Open-Source-Tools provided by the amazing community of layout designers, enthusiasts, and developers. Here a list with just a few of these programs: IC-OSIC-TOOLS, IHP Open PDK, Linux, Docker, Xschem, ngspice, KLayout, Quarto, Vim, Pandoc, TexLive, Python, Git, CoCalc, LibreOffice,

Hymne

In realms where code is free to fly, We build and share, our hearts reach high. No walls, no locks, our wisdom streams, In open light, we chase our dreams.

So sing the joy, the thrill, the spark, In open source, we find our arc. Together strong, we rise, explore— In lines of code, forevermore.

1 Theoretical Background

my idea was to group all theoretics into one chapter, like a when the information becomes necessary during the quarto book, people can just jump back to this. we would circumvent having to introduce every little concept seperately whenever it comes up...

Warning

I am thinking about changing the order, so that we are starting with biquads and work through the filter design from big to small...

This chapter introduces the theory and core concepts necessary for IC design of a biquad filter. It is structured in a way, that it goes from the big picture to the small components. First, biquad filters are introduced with a focus on the universal biquad filter. After that, operational amplifiers come into the the foreground, as biquad filters make use of them in their circuits. Operational amplifiers are looked upon from an IC design standpoint.

1.1 Biquad filter

The biquadratic filter, also known as the biquad filter, has its earliest implementation in the 1960s but is still in use today, most commonly in radio frequency receivers (Razavi 2018). In its application in RF-technology, it is used to remove unwanted neighboring signals and noise (Razavi 2024). As biquad filter are second-order filters, they are also used as building blocks for higher filter implementations, by cascading them and adding first order filters (Rao and Ravikumar 2012).

1.1.1 Universal biquad filter

For the filter design in this project, an universal biquad filter is used. The universal biquad filter is biquad filter variant with four operational amplifiers used in its design and the property of being able to be used in four different filter variants. Depending on the output of the universal biquad used, a low pass filter, high pass filter, band pass filter, or band stop filter will be implemented. This can be seen in (ref to figure of biquad filter design). (Rao and Ravikumar 2012)

NOTE: change to png

The universal biquad filter consists of two non-inverting amplifiers working as adders in the circuit and two integrators. By setting R and C to specific values, the resonance frequency can be chosen. Other parameters adjustable in the universal biquad filter are the quality factor Q and the low-frequency gain H_0 . The quality

1 Theoretical Background

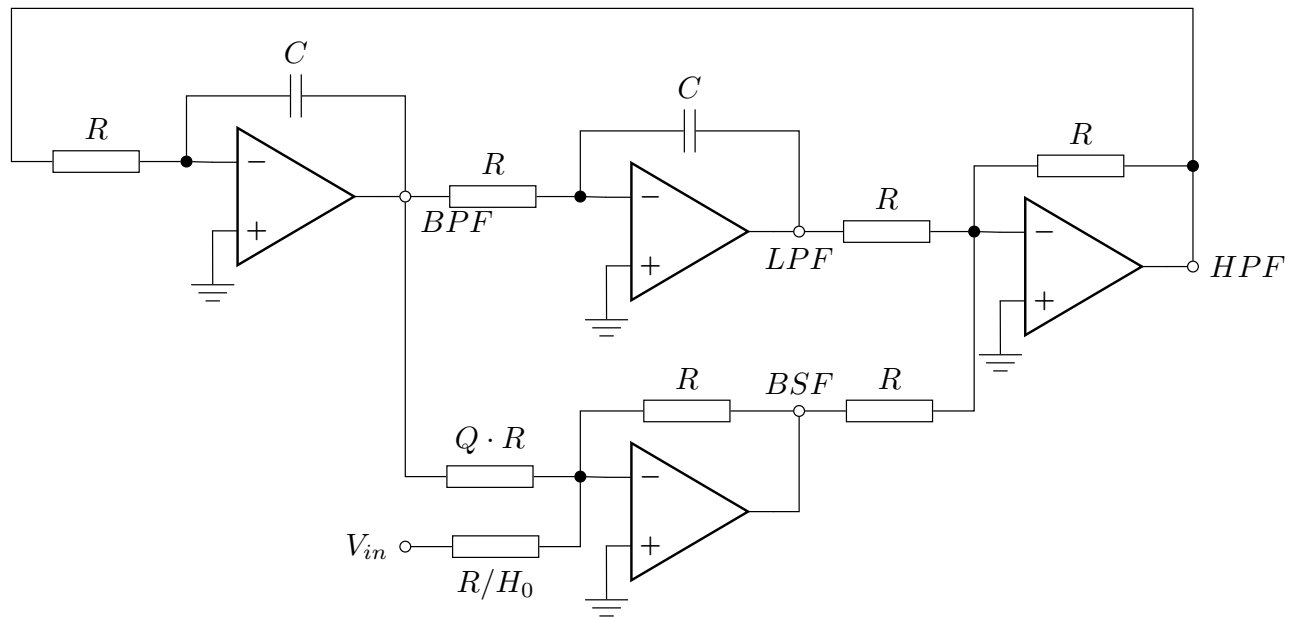


Figure 1.1: circuit design of an universal biquad filter

factor and low-frequency gain determine the frequency response peaks of the low pass filter and the band pass filter. (Rao and Ravikumar 2012)

1.1.2 Characteristics

NOTE: no more than 3 subsections!

As the universal biquad filter is a second order filter with the specified outputs low pass, high pass, band pass, and band stop, each filter option can be described with a transfer function on system level. The general second order transfer function is:

$$H(s) = \frac{a_1 s^2 + b_1 s + c_1}{a_2 s^2 + b_2 s + c_2} \quad (1.1)$$

The coefficients can be chosen so, that different responses, like low pass, high pass, band pass, and band stop are achieved.

In filter design a variant of this generalized transfer function is often chosen because it is easier to describe the system by quality factor and angular frequency. Equation 1.2 is an exemplary low pass filter with a transfer function specified for filter design. (Razavi 2018)

$$H(s) = \frac{\omega_n^2}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2} \quad (1.2)$$

Q determines among other things the amount of peaking the transfer function has at the chosen frequency. **1st-lowPassDifferentQ** shows this graphically, the amount of peaking increases with increasing quality factor Q .

```
# Behavioral Analysis Biquad Filter

import numpy as np
import matplotlib.pyplot as plt

# Initial values
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q1 = 1 # Quality factor
Q2 = 2
Q3 = 5
Q4 = 10
Q5 = 100
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)

# Denominator -> for all filters the same
a0 = 1
a1_1 = (s / (w0 * Q1))
a1_2 = (s / (w0 * Q2))
a1_3 = (s / (w0 * Q3))
```

1 Theoretical Background

```
a1_4 = (s / (w0 * Q4))
a1_5 = (s / (w0 * Q5))
a2 = (s**2 / (w0**2))

den1 = a0 + a1_1 + a2
den2 = a0 + a1_2 + a2
den3 = a0 + a1_3 + a2
den4 = a0 + a1_4 + a2
den5 = a0 + a1_5 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp_1 = b_lp / den1
Hs_lp_2 = b_lp / den2
Hs_lp_3 = b_lp / den3
Hs_lp_4 = b_lp / den4
Hs_lp_5 = b_lp / den5
#Hs_hp = b_hp / den
#Hs_bp = b_bp / den
#Hs_bs = b_bs / den

# Bode Diagram
fig, axs = plt.subplots(2)
fig.suptitle("frequency response of biquad filter")

# Low Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp_1)), label='$Q = 1$')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp_1)) * (180 / np.pi), label='$Q = 1$')
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp_2)), label='$Q = 2$')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp_2)) * (180 / np.pi), label='$Q = 2$')
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp_3)), label='$Q = 5$')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp_3)) * (180 / np.pi), label='$Q = 5$')
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp_4)), label='$Q = 10$')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp_4)) * (180 / np.pi), label='$Q = 10$')
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp_5)), label='$Q = 100$')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp_5)) * (180 / np.pi), label='$Q = 100$')
'''

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')
```



```

# Band Stop Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')
'''

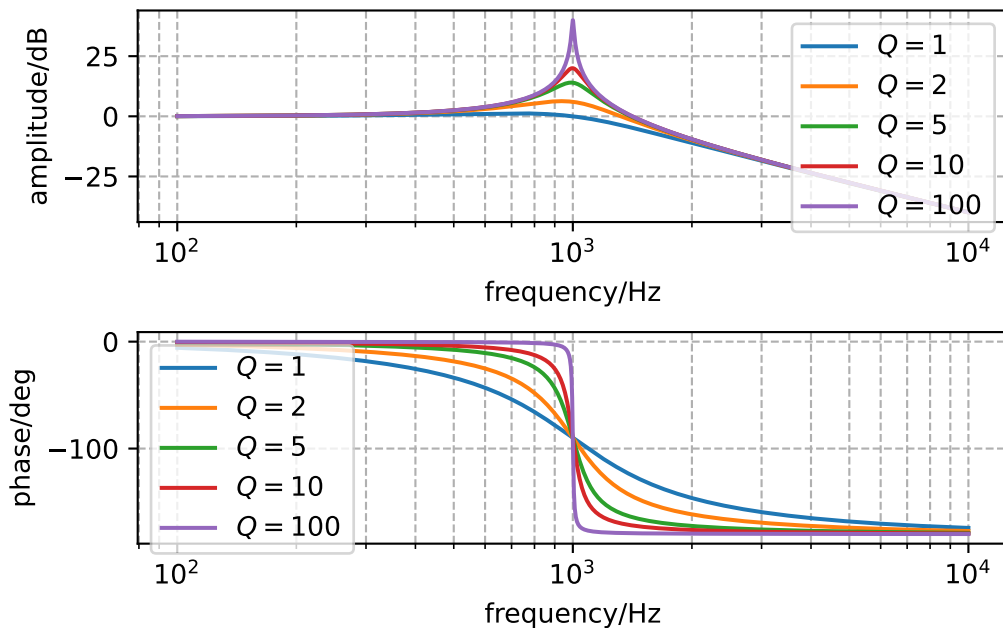
#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
#axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")
axs[1].grid(True, which="both", ls="--")
axs[1].legend()

plt.tight_layout()
plt.show()

```

Listing 1.1 Low pass filter with different quality factors



The height of the peak can be calculated with:

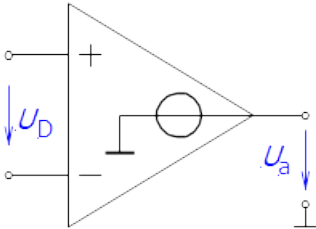
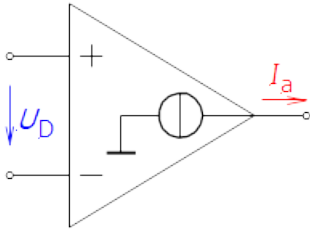
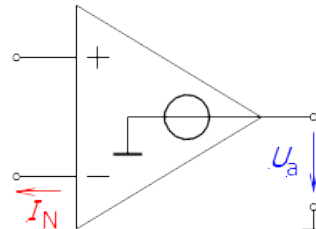
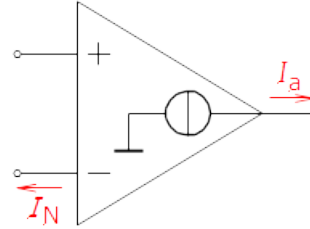
$$A_{peak} = \frac{Q}{\sqrt{1 - \frac{1}{4Q^2}}} \quad (1.3)$$

For $Q = 100$ the peak is $A_{peak} = 100.001$, which converted into dB is $A_{peak,dB} = 40 \text{ dB}$, as is shown in [?@fig-lowPassDifferentQ](#).

1.2 Operational Amplifier

As seen in Figure [1.1](#) a Biquad Filter consists four Operational Amplifiers. To get a better understanding of these, this chapter will discuss the main arguments of OPAMS. There are different types of operational amplifiers that differ, for example, by their low- or high-impedance inputs and outputs. According to Schmid there exist nine different types of the Opamps, but only four are mainly used (Schmid 2000). This is because almost always, the non-inverting (positive) input is designed as a high-impedance voltage input. The inverting (negative) input can either be a high-impedance voltage input or a low-impedance current input, depending on the type. Accordingly, the output can be either a low-impedance voltage output or a high-impedance current output. This results in four basic configurations, as shown in the accompanying table Table [1.1](#).

Table 1.1: Four typical Operational Amplifiers

	Voltage output	Current output
Voltage input	<p>Voltage-Feedback Amplifiers</p> 	<p>Operational Transconductance Amplifier</p> 
Current input	<p>Current-Feedback Amplifiers</p> 	<p>Current Amplifier</p> 



Tip

As a general rule, the simplest circuit that can do a job is usually the best choice. (H. Pretl and Michael Koefinger 2025)

1.2.1 Voltage-Feedback Amplifiers (VFA)

When discussing operational amplifiers (OPAMPs), most sources refer to the Voltage-Feedback Amplifier. These VFAs are **voltage-controlled voltage sources**, essentially acting as voltage boosters. They are characterized by a high-impedance input for both the non-inverting and inverting terminals, and a low-impedance voltage output. To realize various desired circuits, such as amplification, integration, addition/subtraction, etc..., these functions should ideally be achieved only through the surrounding circuitry. To meet this requirement, three main requirements need be satisfied:

- **Extremely High Voltage Gain:** Typically ranging from 60 to 120 dB (or a gain factor of 10^4 to 10^6), this gain should be available over a wide frequency range.
- **High Impedance at Differential Inputs:** Ensuring minimal loading of the signal source.
- **Low Impedance at the Output:** Allowing the amplifier to drive various loads without significant voltage drop.

! Difference between a voltage source and a current source

Voltage source

A voltage source creates a constant voltage output by changing the output.

Current source A current source creates a constant current by changing the voltage

Both of these sources can be explained by Ohm's Law: $R = \frac{U}{I}$. For example the voltage source: There is no influence of the load, so R may change and its value is unknown to the source. So to keep the voltage stable we can only change the current.

1.2.2 Operational Transconductance Amplifier (OTA)

In the design process of the biquad only OTAs will be used, so the focus of this chapter will be on them. The operational transconductance amplifier puts out a current proportional to its input voltage, unlike Voltage-Feedback Amplifiers REF:KAPITEL_VFA. In other words an OTA is a **voltage controlled current source**.

As seen in the figure Figure 1.2 the OTA has the two differential inputs and the current output. On top of that it has a biasing current input I_{bias} which can control the

1.2.2.1 Transconductance

Transconductance is a fundamental parameter that describes the relationship between the input voltage and the output current in electronic devices, particularly in transistors and vacuum tubes. It is defined as the ratio

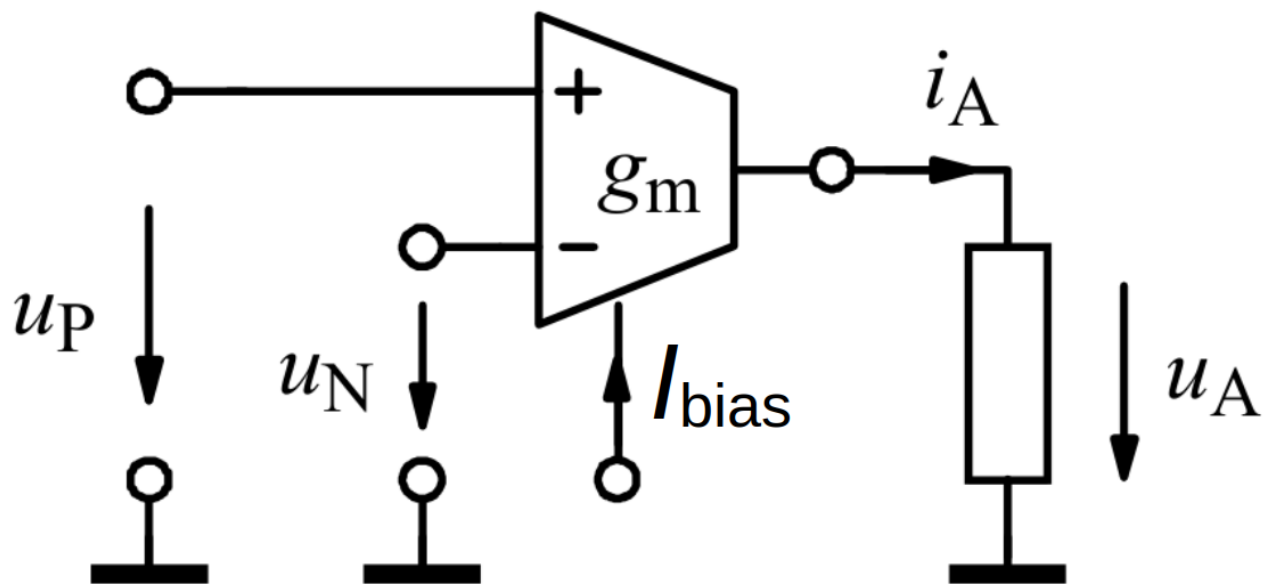


Figure 1.2: Schematic symbol of an OTA (**book-rotaktivefilter?**)

of the change in output current to the change in input voltage, under conditions where all other variables are held constant. Mathematically, it is expressed as:

$$g_m = \frac{\Delta I_{out}}{\Delta V_{in}}$$

where g_m is the transconductance, ΔI_{out} is the change in output current, and ΔV_{in} is the change in input voltage. The unit of transconductance is the siemens (S), which is equivalent to amperes per volt (A/V). Historically, it was also measured in “mhos,” which is “ohm” spelled backwards, reflecting its inverse relationship to resistance.

The term “transconductance” originates from the concept of “transfer conductance.” It combines the ideas of “transfer,” indicating the transfer of a signal from the input to the output, and “conductance,” which is the reciprocal of resistance and measures how easily a material conducts electric current. In essence, transconductance quantifies how effectively a device can convert a voltage change at its input into a proportional current change at its output.

Transconductance is a crucial parameter in the design and analysis of electronic circuits, especially in amplifiers. In vacuum tubes and field-effect transistors (FETs), the transconductance determines the device’s ability to amplify signals. A higher transconductance value indicates a stronger amplification capability, as a small change in input voltage can result in a significant change in output current. This property makes transconductance a key factor in the performance and efficiency of various electronic devices and circuits.

####HW 5-Transistor OTA

1.2.3 Variants ?

maybe like miller opamp or something like that

1.3 MOSFET (Metal-Oxide-Semiconductor Field-Effect-Transistor)

To get a general insight how an Operational Amplifier works and to understand the difference between a “off the shelf” part and a self made IC-Design, its necessary to understand the main component: The MOSFET.

To be more precise, its necessary to understand the behaviour of a MOSFET. Since were designing one from scratch, its possible to change the width W and length L of the physical layout. By doing this there is a lot of room for freedom and experimental space.

BILD N-MOS

BILD P-MOS

1.3.1 Small-Signal Representation

1.3.2 Sizing

As mentioned earlier we have several degrees of freedom in our design, like W , L , or the bias current I_D . The problem is, that its extremely complex to describe a MOSFET mathmatically and on top to that of change values to get a desired behaviour. Gladly a wide used technique, which was introduced by P. Jespers and B. Murmann, is the g_m/I_D approach \ref{Jesper.... Pretl 3}.

Saturation Saturation is a specific operation mode of a transistor. Its also called “ON mode”, because a channel between drain and source is created. This state is reached when the voltage $V_{DS} \geq (V_{GS} - V_{th})$. When this happens the drain-source current I_{DS} gets stable (saturates) and becomes nearly independent of V_{DS} . Still it can be very well controlled via V_{GS} , due to an effect called “pinch-off” (??).

The g_m/I_D method is primarily intended to be used in saturated MOSFETs. To ensure this we keep the drain-source voltage as $V_{ds} = V_D/2$.

1.3.3 Current Mirror

1.3.4 Differential Stage

2 Characterisation

this chap is more about our biquad, and how we made it the way it is now

i am failry open to discuss the exact contents of this chapter, as it is the one i am the most unsure about

we also need to excatly decide on where this chapter ends, as Finn will continue with chap 5 and the break between the chapters shall be as smooth as possible

Warning

decide on the chapter intro when it is clear where the cut is between this chapter and the next

2.1 Behauvioural Analysis and macro modelling

The behauvioural analysis is done through macro modelling the universal biquad filter as a system. The system can be described with transfer functions and modelled with python.

2.1.1 Transfer Functions and frequency response

Warning

This whole section needs to be reworked, I am not happy with the reading flow right now! -AvA
I am going to shift the theory over to theory and only talk about the implementation...

The ASLK PRO Manual (Rao and Ravikumar 2012) provides the transfer functions of the four filter outputs: low pass, high pass, band pass, and band stop. The transfer functions are adaptations of the general second order transfer function as seen in Equation 1.1. (Razavi 2018)

In the following transfer function the input and output voltage are referenced according to Figure 1.1. The sections only contain their specific transfer function and frequency response.

Warning

please recheck ALL TFs against manual for corecctness

2 Characterisation

2.1.1.1 Low pass

The output if the low pass filter is marked in the circuit (Figure 1.1) as *LPF* and corresponds to V_{03} in the transfer function Equation 2.1.

$$\frac{V_{03}}{V_i} = \frac{H_0}{\left(1 + \frac{s}{\omega_0 Q} + \frac{s^2}{\omega_0^2}\right)} \quad (2.1)$$

?@fig-freqResponseLowpass shows the amplitude and phase response of the low pass filter. The required frequency $f_0 = 1 \text{ kHz}$ and quality factor $Q = 10$ recognizable in the bode plot. As the dc-gain was chosen to be $H_0 = 1$, the low pass filter has a amplitude amplification of 1 in the lower frequencies.

```
# Behavioral Analysis Biquad Filter

import numpy as np
import matplotlib.pyplot as plt

# Initial values
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)
```



```

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

# Bode Diagram
fig, axs = plt.subplots(2)
#fig.suptitle("frequency response of biquad filter")

# Low Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='low pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp)) * (180 / np.pi), label='low pass')
'''

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')

# Band Stop Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')
'''

#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
#axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")

```

2 Characterisation

```
axs[1].grid(True, which="both", ls="--")
#axs[1].legend()

plt.tight_layout()
plt.show()
```

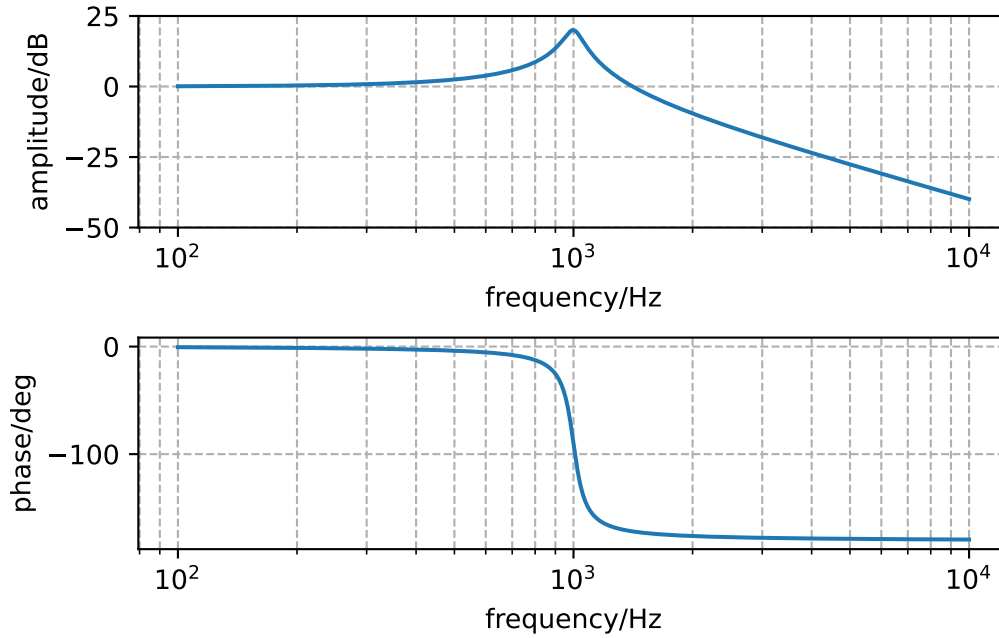


Figure 2.1: Frequency response of the low pass filter

With knowing the dc gain $H_0 = 1$ and quality factor $Q = 10$, the amplitude of the peak can be calculated as seen in Equation 1.3. Expression the value in dB, gives peak amplitude of $A_{peak,dB} = 20.002 \text{ dB}$ which corresponds to the peak value seen in `?@fig-freqResponseLowpass`.

2.1.1.2 High pass

The output if the high pass filter is marked in the circuit (Figure 1.1) as *HPF* and corresponds to V_{01} in the transfer function Equation 2.2.

$$\frac{V_{01}}{V_i} = \frac{\left(H_0 \cdot \frac{s^2}{\omega_0^2}\right)}{\left(1 + \frac{s}{\omega_0 Q} + \frac{s^2}{\omega_0^2}\right)} \quad (2.2)$$

`?@fig-freqResponseLowpass` shows the amplitude and phase response of the high pass filter. The required frequency $f_0 = 1 \text{ kHz}$ and quality factor $Q = 10$ recognizable in the bode plot. As the dc-gain was chosen to be $H_0 = 1$, the low pass filter has a amplitude amplification of 1 in the higher frequencies.

```

# Behavioral Analysis Biquad Filter

import numpy as np
import matplotlib.pyplot as plt

# Initial values
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den

```

2 Characterisation

```
Hs_bs = b_bs / den

# Bode Diagram
fig, axs = plt.subplots(2)
fig.suptitle("frequency response of biquad filter")

# Low Pass Filter
#axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='low pass')
#axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp)) * (180 / np.pi), label='low pass')

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')
'''

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')

# Band Stop Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')
'''

#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
#axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")
axs[1].grid(True, which="both", ls="--")
#axs[1].legend()

plt.tight_layout()
plt.show()
```

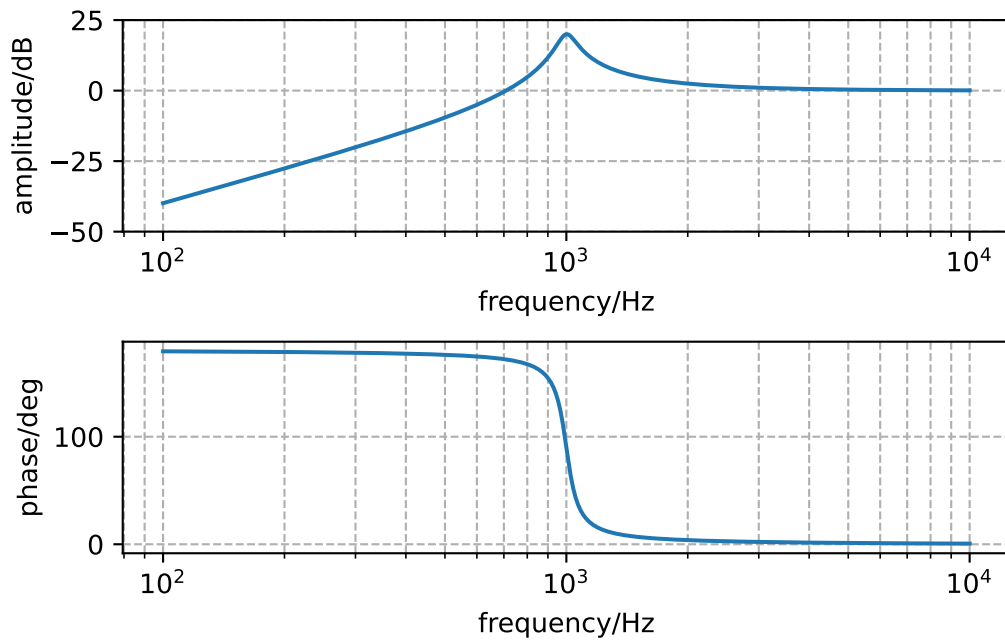


Figure 2.2: Frequency response of the high pass filter

2.1.1.3 Band pass

The output for the band pass filter is marked as *BPF* in Figure 1.1. This denotes the point that is referenced in (eg-TFBandpass?) as V_{02} .

$$\frac{V_{02}}{V_i} = \frac{\left(-H_0 \cdot \frac{s}{\omega_0}\right)}{\left(1 + \frac{s}{\omega_0 Q} + \frac{s^2}{\omega_0^2}\right)} \quad (2.3)$$

The band pass shown in ?@fig-freqResponseBandpass has its center frequency at 1 kHz as set in the requirements. Similarly to the low pass filter in ?@fig-freqResponseLowpass and the high pass filter in ?@fig-freqResponseHighpass the amplitude response peaks at this frequency, with its peak influenced by the quality factor.

```
# Behavioral Analysis Biquad Filter

import numpy as np
import matplotlib.pyplot as plt

# Initial values
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
```

2 Characterisation

```
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

# Bode Diagram
fig, axs = plt.subplots(2)
fig.suptitle("frequency response of biquad filter")
'''
# Low Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='low pass')
```

2.1 Behavioural Analysis and macro modelling

```
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp)) * (180 / np.pi), label='low pass')

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')
'''

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')

# Band Stop Filter
#axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
#axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')

#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
#axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")
axs[1].grid(True, which="both", ls="--")
#axs[1].legend()

plt.tight_layout()
plt.show()
```

2 Characterisation

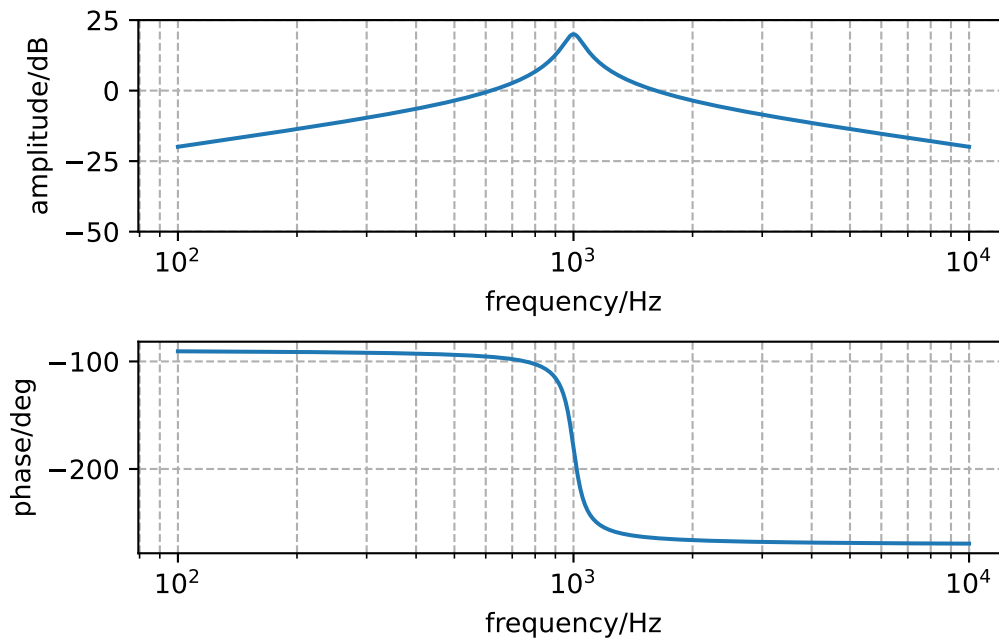


Figure 2.3: Frequency response of the band pass filter

2.1.1.4 Band stop

The output for the band stop filter is marked in Figure 1.1 as *BSF* and in the transfer function as V_{04} .

$$\frac{V_{04}}{V_i} = \frac{\left(1 + \frac{s^2}{\omega_0^2}\right) \cdot H_0}{\left(1 + \frac{s}{\omega_0 Q} + \frac{s^2}{\omega_0^2}\right)} \quad (2.4)$$

(Renner 2025) argues that Equation 2.4 from the ASLK PRO Manual (Rao and Ravikumar 2012) is incorrect, as using that equation produces inconsistent results. Using the negated form of Equation 2.4 as seen in Equation 2.5 seems to produce the correct output. Therefore Equation 2.5 will be used for further analysis.

$$\frac{V_{04}}{V_i} = -\frac{\left(1 + \frac{s^2}{\omega_0^2}\right) \cdot H_0}{\left(1 + \frac{s}{\omega_0 Q} + \frac{s^2}{\omega_0^2}\right)} \quad (2.5)$$

`?@fig-freqResponseBandstop` shows the frequency response of the band stop, with its center frequency at 1 kHz .

```
# Behavioral Analysis Biquad Filter
```

```
import numpy as np
```



```

import matplotlib.pyplot as plt

# Initial values
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

# Bode Diagram

```

2 Characterisation

```
fig, axs = plt.subplots(2)
#fig.suptitle("frequency response of biquad filter")
'''
# Low Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='low pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp)) * (180 / np.pi), label='low pass')

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')
'''
# Band Stop Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')

#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
#axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")
axs[1].grid(True, which="both", ls="--")
#axs[1].legend()

plt.tight_layout()
plt.show()
```

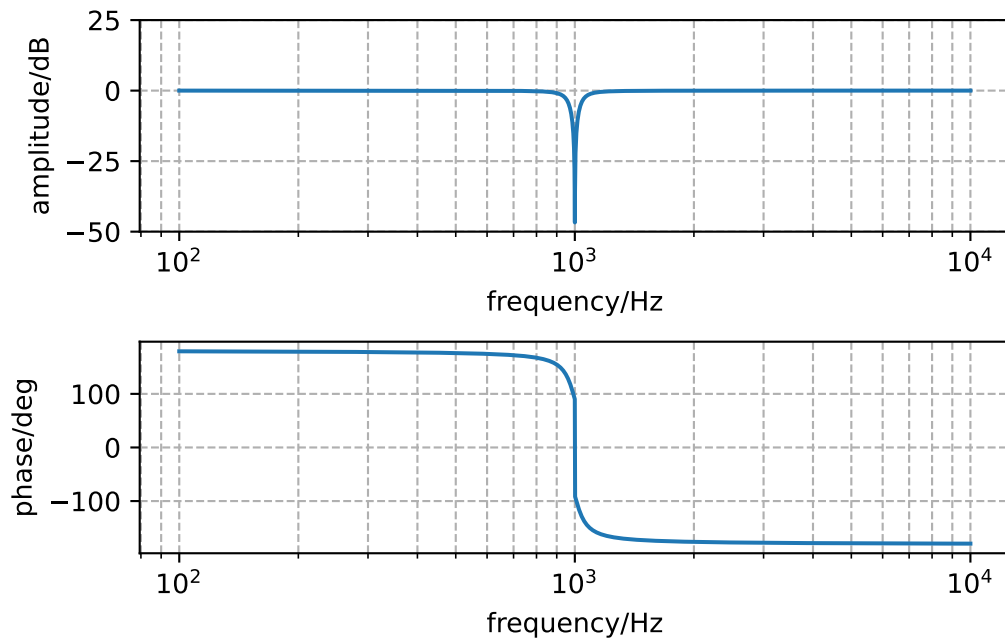


Figure 2.4: Frequency response of the band stop filter

2.1.1.5 Comparison

some text...

```
# Behavioral Analysis Biquad Filter
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Initial values
```

```
f0 = 1e3 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later
```

```
# Logarithmic frequency axis
```

```
frequencies = np.logspace(2, 4, 10000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j
```

```
#####
```

```
# Transfer functions of Active Filters
```

```
#####
```

2 Characterisation

```
### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -((1 + (s**2 / (w0**2))) * H0)

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

# Bode Diagram
fig, axs = plt.subplots(2)
#fig.suptitle("frequency response of biquad filter")

# Low Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='low pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_lp)) * (180 / np.pi), label='low pass')

# High Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='high pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_hp)) * (180 / np.pi), label='high pass')

# Band Pass Filter
axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='band pass')
axs[1].semilogx(frequencies, np.unwrap(np.angle(Hs_bp)) * (180 / np.pi), label='band pass')

# Band Stop Filter
```

```

axs[0].semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='band stop')
axs[1].semilogx(frequencies, (np.angle(Hs_bs)) * (180 / np.pi), label='band stop')

#axs[0].title("amplitude response")
axs[0].set_xlabel("frequency/Hz")
axs[0].set_ylabel("amplitude/dB")
axs[0].set_ylim(-50, 25)
axs[0].grid(True, which="both", ls="--")
axs[0].legend(loc=1)

#axs[1].title("phase response")
axs[1].set_xlabel("frequency/Hz")
axs[1].set_ylabel("phase/deg")
axs[1].grid(True, which="both", ls="--")
axs[1].legend()

plt.tight_layout()
plt.show()

```

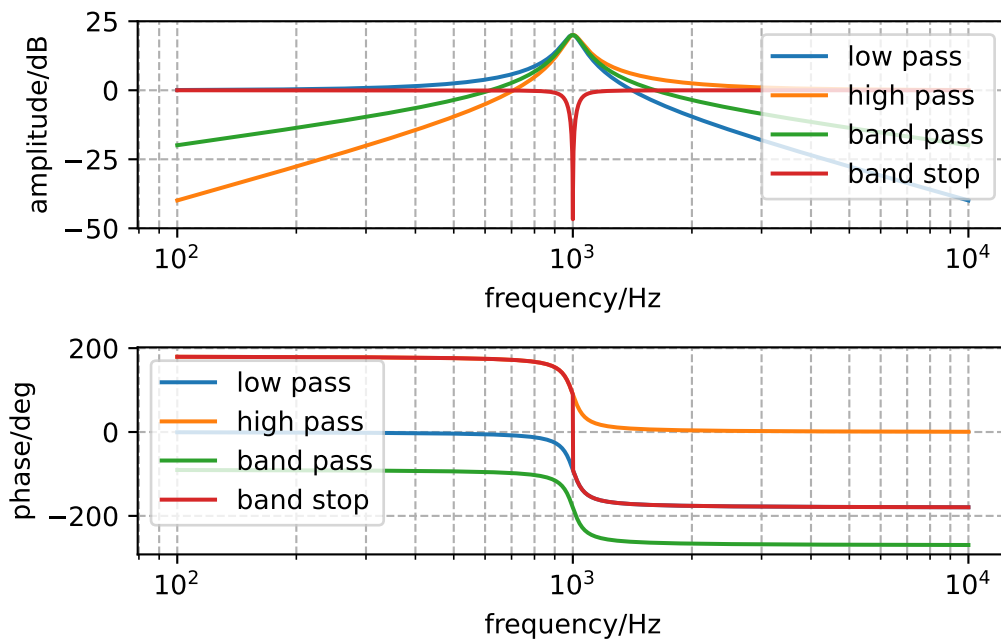


Figure 2.5: Behavioural analysis of biquad filter

2 Characterisation

2.1.2 Stability

The stability of the biquad is checked at different hierarchical levels. The first analysis considers the system from a theoretical standpoint with transfer functions, and checks if conceptual design of the biquad filter is stable. On a component level the stability of the integrators and adders is analyzed, to verify that the chosen values for resistors and capacitors do not induce oscillations through the feedback loop.

(At last, the general stability of the OTA circuit itself needs to be checked, so that) <- lets see what we can find for this...

2.1.2.1 System stability

A system is stable if its impulse response is absolutely integrable. In case of a given transfer function, this can also be checked by calculating the poles of the transfer function. If all the poles lay in the left half of the s-plane, the system is considered stable. There is a special case where single poles can lay on the $j\omega$ -axis, on their own or in combination with poles in the left half of the s-plane. Systems which fall under that, are called marginally stable. (Fliege 1991)

2.1.2.2 Pole-zero plot

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import tf2zpk

# Given values
f = 1e3
w = 2 * np.pi * f
R = 1e3
C = 1 / (w * R)
Q = 10
H0 = 1

# Calculate w0
w0 = 1 / (R * C)

# Transfer function coefficients
a2 = 1 / w0**2
a1 = 1 / (w0 * Q)
a0 = 1

# Define transfer functions manually as (numerator, denominator) pairs
systems = {
    'Low pass filter': ([H0], [a2, a1, a0]),
```

```

    'High pass filter': ([H0 / w0**2, 0, 0], [a2, a1, a0]),
    'Band pass filter': ([-H0 / w0, 0], [a2, a1, a0]),
    'Band stop filter': ([H0 / w0**2, 0, H0], [a2, a1, a0])
}

# Function to plot pole-zero map
def plot_pzmap(num, den, title, subplot_pos):
    zeros, poles, _ = tf2zpk(num, den)
    plt.subplot(2, 2, subplot_pos)
    plt.plot(np.real(zeros), np.imag(zeros), 'go', label='Zeros')
    plt.plot(np.real(poles), np.imag(poles), 'rx', label='Poles')
    plt.axhline(0, color='gray', lw=0.5)
    plt.axvline(0, color='gray', lw=0.5)
    plt.title(title)
    plt.xlim([-1500, 1500])
    plt.ylim([-10000, 10000])
    plt.grid(True)
    plt.legend(loc='upper right')

# Plot all systems
plt.figure(figsize=(12, 10))
for i, (title, (num, den)) in enumerate(systems.items(), 1):
    plot_pzmap(num, den, title, i)

plt.tight_layout()
plt.show()

```

2 Characterisation

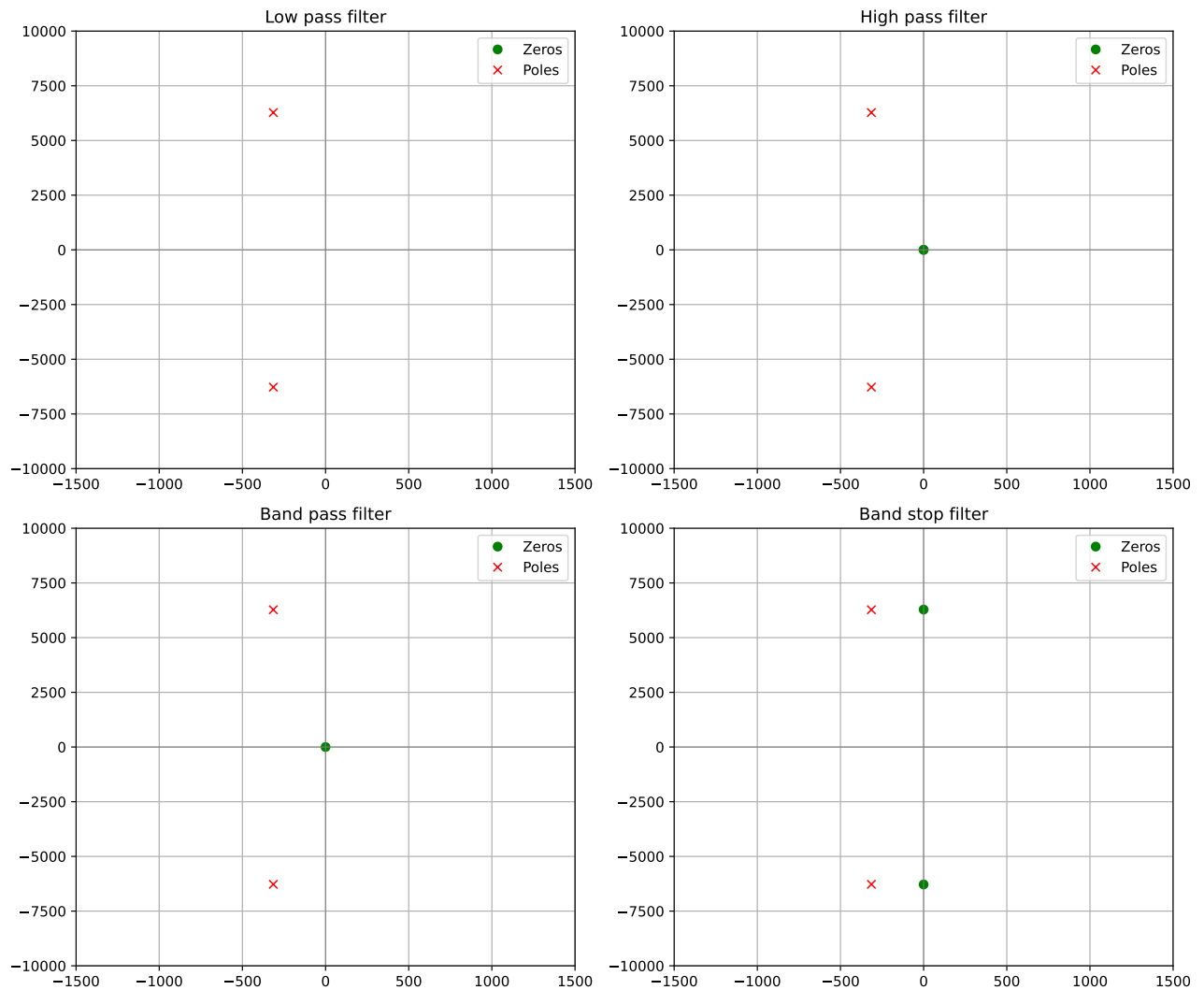


Figure 2.6: Pole-zero plot for all transfer functions

?@fig-poleZeroStability shows the pole-zero plots of all four filters, low pass, high pass, band pass and band stop. In all four plots the poles are located in the left half of the s -plane and the system can therefore theoretically be classified as stable. (Razavi 2018) confirms this, as the article explains that with $Q \rightarrow \infty$ the poles of the system approach the $j\omega$ axis and the system becomes unstable.

This analysis only considers the system as a mathematical model and as a whole. Further considerations regarding the stability of the components, integrators and adders, and the stability of the operational amplifiers themselves, have to be done.

2.1.2.3 Component stability

Circuits with opamps often have feedback loops, meaning that the output of the operational amplifier is somehow connected to the inverted input of the opamp. These feedback loops become problematic when the feedback signal is in phase with the input signal, as positive feedback is created and the circuit is working as an oscillator. (Reisch 2007)

The stability of the non-inverting amplifier can be verified by calculating the phase reserve α of the circuit. If f_k is the frequency where the feedback gain is equal to 1 and φ_k is the corresponding phase to that frequency, then the phase reserve is calculated by:

$$\alpha = 180^\circ - \varphi_k$$

For circuits to be considered stable, the phase reserve has to be positive. To reduce overshoots during the transient response, it is customary to have a phase reserve of $\alpha > 45^\circ$. (Reisch 2007)

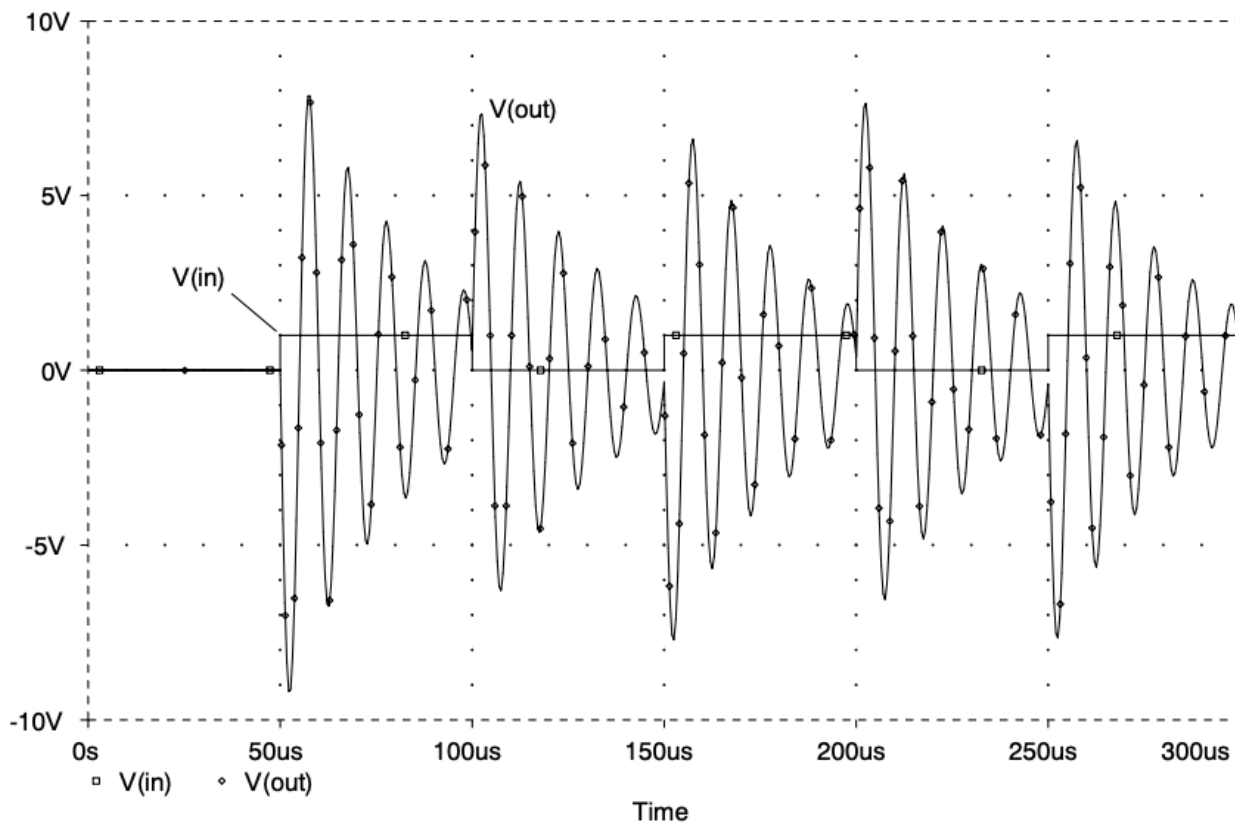


Figure 2.7: Example of a transient response of a circuit with a phase reserve of $\alpha = 5.7^\circ$ (Reisch 2007)

Figure (reisch) shows the transient response of a circuit with a phase reserve of $\alpha = 5.7^\circ$. The overshoots are clearly visible and number of the overshoots per pulses are larger than the customary “one over, one under”-

2 Characterisation

rule. As the phase reserve is positive, the figure shows that even though the transient response is not ideal, the oscillations are attenuated and the circuit is can be considered as stable.

In practical application, the phase reserve can be graphically determined with the help of bode diagrams. The bode diagram of the circuit with an open feedback loop is simulated, so that the frequency f_k can be read out. This is the frequency where the feedback gain is 1 or 0 dB. The corresponding frequency to that, is the phase of the feedback gain φ_k , the difference between -180° and φ_k is the phase reserve α . (Reisch 2007)

Warning

Talk about which ota was used for stability analysis

Warning

insert circuit where the stability analysis was done from, like in ltspice

Warning

insert bode diagram figures (both: integrator and adder)

```
# Stability analysis adder
```

```
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.insert(0, '../simulation')
import ltsp3

sd=ltsp3.SimData('../simulation/stability_adder.raw',[b'v(v_out)',b'frequency'])

nvout = sd.variables.index(b'v(v_out)')
nfrequency = sd.variables.index(b'frequency')

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)

ax1.semilogx(sd.values[nfrequency],20*np.log10(abs(sd.values[nvout])))
ax1.set_ylabel("Magnitude/dB")
ax1.axvline(3e5,color='red',linestyle='--')
ax1.grid(True, which="both", ls="--")

ax2.semilogx(sd.values[nfrequency],np.angle(sd.values[nvout], deg=True))
ax2.axvline(3e5,color='red',linestyle='--')
ax2.axhline(82.75,color='red',linestyle='--')
ax2.set_ylabel("Phase/deg")
ax2.set_xlabel("Frequency/Hz")
```

```
plt.grid(True, which="both", ls="--")
plt.tight_layout()
plt.show()
```

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1699: ComplexWarning:

Casting complex values to real discards the imaginary part

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1345: ComplexWarning:

Casting complex values to real discards the imaginary part

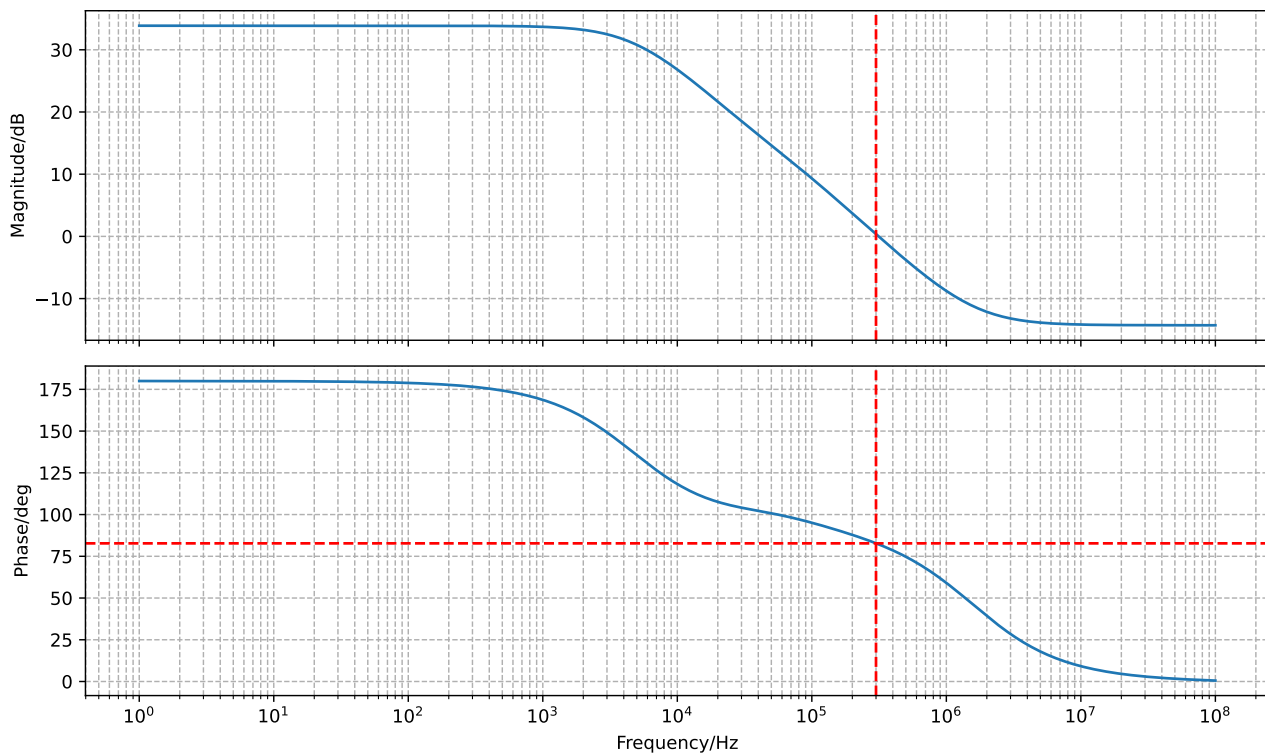


Figure 2.8: Stability analysis of the adder

```
# Stability analysis integrator

import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.insert(0, '../simulation')
```

2 Characterisation

```
import ltspy3

sd=ltspy3.SimData('../simulation/stability_integrator.raw',[b'v(v_out)',b'frequency'])

nvout = sd.variables.index(b'v(v_out)')
nfrequency = sd.variables.index(b'frequency')

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)

ax1.semilogx(sd.values[nfrequency],20*np.log10(abs(sd.values[nvout])))
ax1.set_ylabel("Magnitude/dB")
ax1.axvline(7e5,color='red',linestyle='--')
ax1.grid(True, which="both", ls="--")

ax2.semilogx(sd.values[nfrequency],np.angle(sd.values[nvout], deg=True))
ax2.axvline(7e5,color='red',linestyle='--')
ax2.axhline(68,color='red',linestyle='--')
ax2.set_ylabel("Phase/deg")
ax2.set_xlabel("Frequency/Hz")

plt.grid(True, which="both", ls="--")
plt.tight_layout()
plt.show()
```

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1699: ComplexWarning:

Casting complex values to real discards the imaginary part

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1345: ComplexWarning:

Casting complex values to real discards the imaginary part

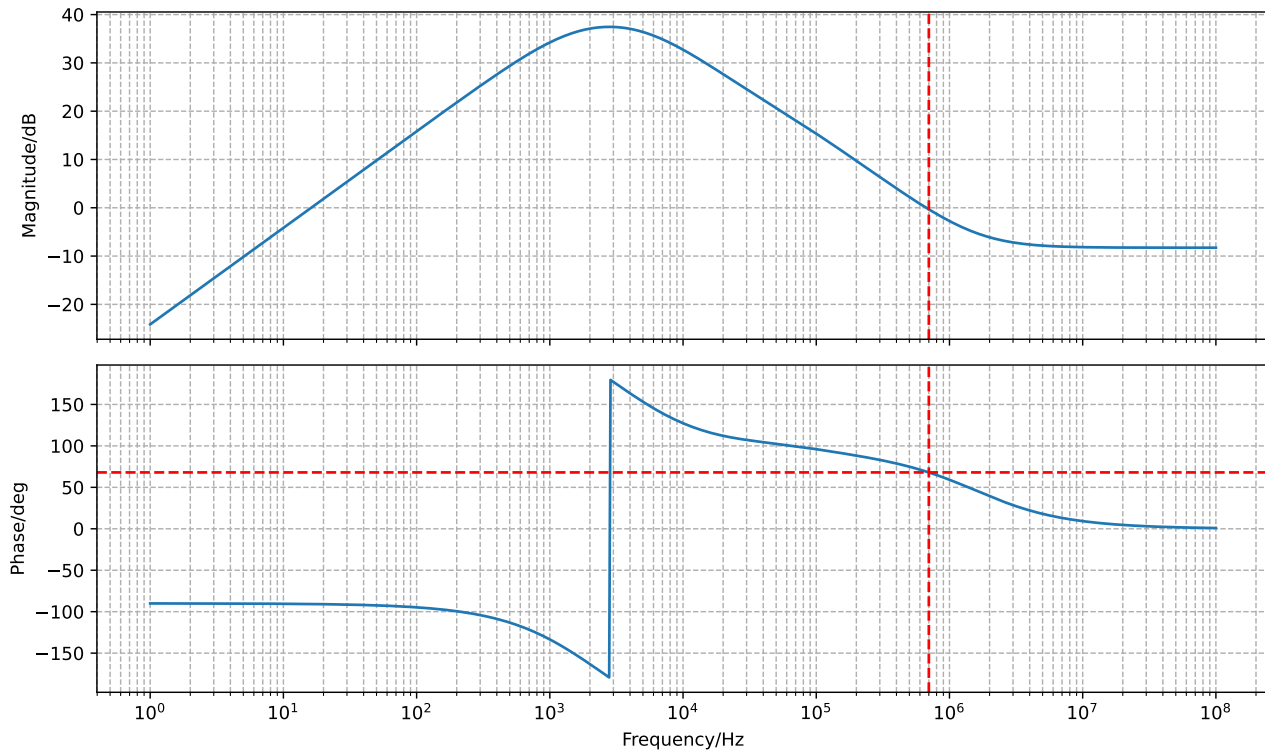


Figure 2.9: Stability analysis of the integrator

In the following figures (ref) and **?@fig-stabilityIntegrator** this method was used to determine the stability over the phase reserve.

(add here part about whether they are stable...)

The stability of the integrator circuit, as seen in **?@fig-stabilityIntegrator**, shows a intersection of the magnitude plot with the 0 dB line at about $f_k = 70 \text{ kHz}$, which corresponds to a phase of $\varphi_k = 68^\circ$. This would leave a phase reserve of:

$$\alpha_{int} = 180^\circ - \varphi_k = 112^\circ > 45^\circ$$

Therefore the integrator would be stable.

2.1.2.4 OTA stability

i have not found anything in pretl script, but i clearly remember mr. meiners talking about this...

i have found something in pretls script, but looks not easy... look at chapter about mosfets

2 Characterisation

2.1.3 Ideal Opamp

kinda forgot where I wanted to go with this one... eh hh maybe it will come back

uhhh i think it was about the ideal circuit and spice analysis etc

To check the behaviour of the implemented circuit against the modelled behaviour of the transfer function, the universal biquad was built as an ideal circuit with voltage-regulated current sources instead of OTAs.

```
# plot Ideal (voltage controlled current? source) biquad
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.insert(0, '../simulation')
import ltspy3

sd=ltspy3.SimData('../simulation/biquad_univ.raw')

nvoutLPF = sd.variables.index(b'v(lpf)')
nvoutHPF = sd.variables.index(b'v(hpf)')
nvoutBPF = sd.variables.index(b'v(bpf)')
nvoutBSF = sd.variables.index(b'v(bsf)')
nfrequency = sd.variables.index(b'frequency')

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)

ax1.semilogx(sd.values[nfrequency], 20*np.log10(abs(sd.values[nvoutLPF])), label='lpf')
ax1.semilogx(sd.values[nfrequency], 20*np.log10(abs(sd.values[nvoutHPF])), label='hpf')
ax1.semilogx(sd.values[nfrequency], 20*np.log10(abs(sd.values[nvoutBPF])), label='bpf')
ax1.semilogx(sd.values[nfrequency], 20*np.log10(abs(sd.values[nvoutBSF])), label='bsf')
ax1.set_xlim([10e1, 10e3])
ax1.set_ylim([-40, 20])
ax1.set_ylabel("Magnitude/dB")
ax1.grid(True, which="both", ls="--")
ax1.legend()

ax2.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutLPF], deg=True), label='lpf')
ax2.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutHPF], deg=True), label='hpf')
ax2.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutBPF], deg=True), label='bpf')
ax2.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutBSF], deg=True), label='bsf')
ax2.set_ylabel("Phase/deg")
ax2.set_xlabel("Frequency/Hz")
ax2.legend()

plt.grid(True, which="both", ls="--")
```

```
plt.tight_layout()
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1699: ComplexWarning:
```

```
Casting complex values to real discards the imaginary part
```

```
/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1345: ComplexWarning:
```

```
Casting complex values to real discards the imaginary part
```

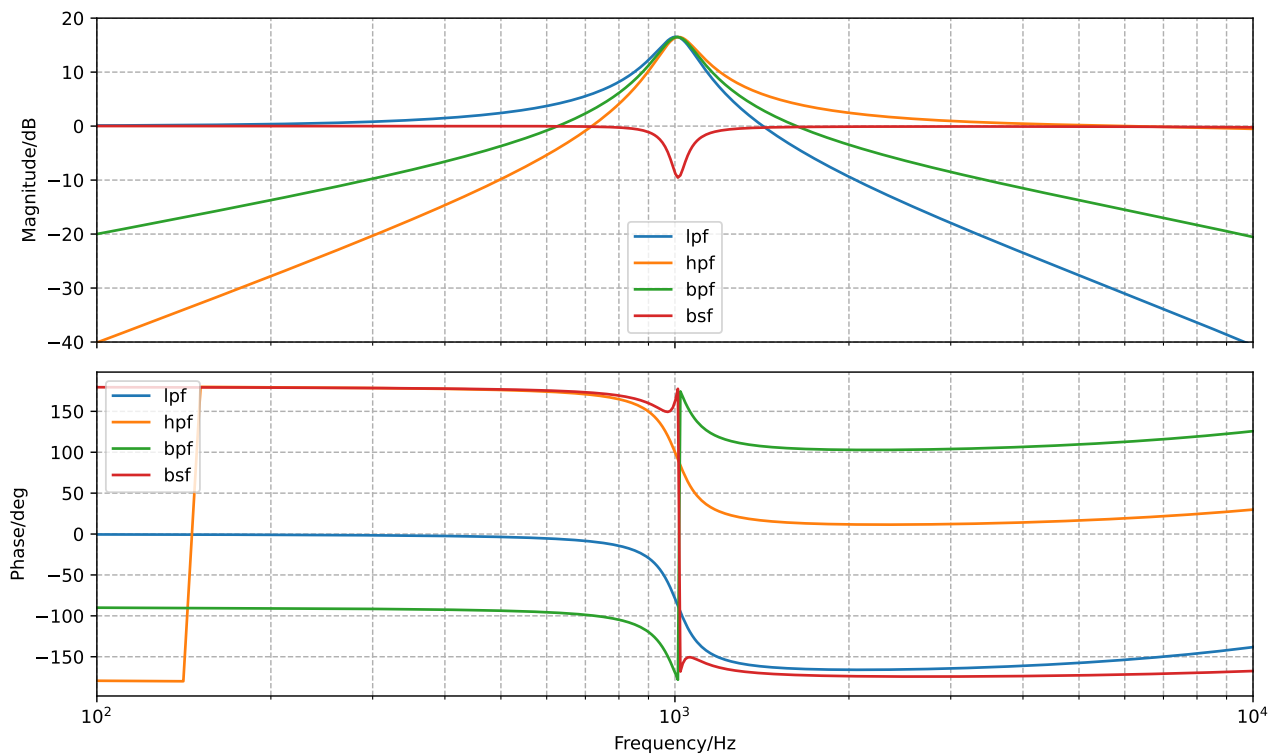


Figure 2.10: Simulation of an idealised biquad

here plot comp between python and ideal

```
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.insert(0, '../simulation')
import ltspy3
```

2 Characterisation

```
sd=ltspy3.SimData('../simulation/biquad_univ.raw')

nvoutLPF = sd.variables.index(b'v(lpf)')
nvoutHPF = sd.variables.index(b'v(hpf)')
nvoutBPF = sd.variables.index(b'v(bpf)')
nvoutBSF = sd.variables.index(b'v(bsf)')
nfrequency = sd.variables.index(b'frequency')

#behavioural modelling with tfs

# Initial values
f0 = 1000 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(0, 4, 1000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -(1 + (s**2 / (w0**2))) * H0

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2
```



```
#####
# Calculation of the transfer functions H(s)
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

#mag plots

fig, axs = plt.subplots(2, 2, sharex=True, sharey=True)

ax1, ax2, ax3, ax4 = axs.flatten()

# Plotting each
ax1.semilogx(frequencies, 20 * np.log10(np.abs(Hs_lp)), label='tf')
ax1.semilogx(sd.values[nfrequency], 20 * np.log10(abs(sd.values[nvoutLPF])), label='ngspice')
ax1.set_ylabel("magnitude/dB")
ax1.set_title("low pass filter")
ax1.set_xlim([10,10e3])
ax1.set_ylim([-40,25])
ax1.grid(True, which="both", ls="--")
ax1.legend()

ax2.semilogx(frequencies, 20 * np.log10(np.abs(Hs_hp)), label='tf')
ax2.semilogx(sd.values[nfrequency], 20 * np.log10(abs(sd.values[nvoutHPF])), label='ngspice')
ax2.set_title("high pass filter")
#ax2.set_xlim([10,10e3])
#ax2.set_ylim([-40,25])
ax2.grid(True, which="both", ls="--")
ax2.legend()

ax3.semilogx(frequencies, 20 * np.log10(np.abs(Hs_bp)), label='tf')
ax3.semilogx(sd.values[nfrequency], 20 * np.log10(abs(sd.values[nvoutBPF])), label='ngspice')
ax3.set_xlabel("frequency/Hz")
ax3.set_ylabel("magnitude/dB")
ax3.set_title("band pass filter")
#ax3.set_xlim([10,10e3])
#ax3.set_ylim([-40,25])
ax3.grid(True, which="both", ls="--")
ax3.legend()

ax4.semilogx(frequencies, 20 * np.log10(np.abs(Hs_bs)), label='tf')
ax4.semilogx(sd.values[nfrequency], 20 * np.log10(abs(sd.values[nvoutBSF])), label='ngspice')
ax4.set_xlabel("frequency/Hz")
```

2 Characterisation

```
ax4.set_title("band stop filter")
#ax4.set_xlim([10,10e3])
#ax4.set_ylim([-40,25])
ax4.grid(True, which="both", ls="--")
ax4.legend()
```

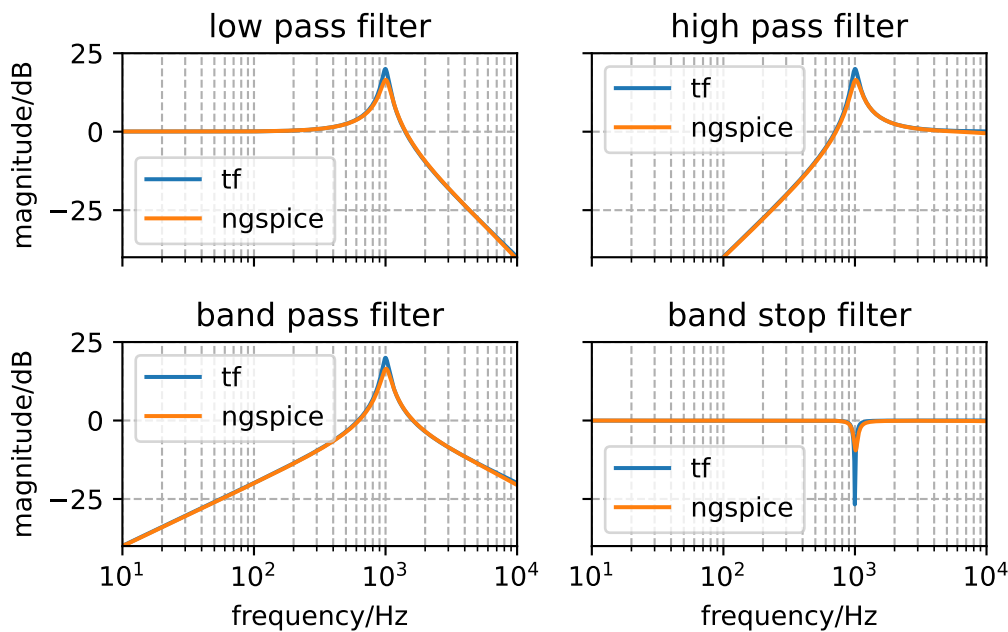
```
plt.tight_layout()
plt.show()
```

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1699: ComplexWarning:

Casting complex values to real discards the imaginary part

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1345: ComplexWarning:

Casting complex values to real discards the imaginary part



```
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.insert(0, '../simulation')
import ltspy3

sd=ltspy3.SimData('../simulation/biquad_univ.raw')
```

```

nvoutLPF = sd.variables.index(b'v(lpf)')
nvoutHPF = sd.variables.index(b'v(hpf)')
nvoutBPF = sd.variables.index(b'v(bpf)')
nvoutBSF = sd.variables.index(b'v(bsf)')
nfrequency = sd.variables.index(b'frequency')

#behavioural modelling with tfs

# Initial values
f0 = 1000 # Resonance frequency in Hz
w0 = 2 * np.pi * f0 # Angular frequency in rad/s
Q = 10 # Quality factor
H0 = 1 # Play around with this later

# Logarithmic frequency axis
frequencies = np.logspace(0, 4, 1000) # Frequency from 10^2 to 10^4 Hz
s = 1j * 2 * np.pi * frequencies # Laplace-Variable s = j

#####
# Transfer functions of Active Filters
#####

### Numerator
# Low Pass Filter
b_lp = H0

# High Pass Filter
b_hp = (H0 * (s**2 / w0**2))

# Band Pass Filter
b_bp = (-H0 * (s / w0))

# Band Stop Filter
b_bs = -(1 + (s**2 / (w0**2))) * H0

# Denominator -> for all filters the same
a0 = 1
a1 = (s / (w0 * Q))
a2 = (s**2 / (w0**2))

den = a0 + a1 + a2

#####
# Calculation of the transfer functions H(s)

```

2 Characterisation

```
#####
Hs_lp = b_lp / den
Hs_hp = b_hp / den
Hs_bp = b_bp / den
Hs_bs = b_bs / den

#phase plots

fig, axs = plt.subplots(2, 2, sharex=True, sharey=True)

ax1, ax2, ax3, ax4 = axs.flatten()

# Plotting each
ax1.semilogx(frequencies, np.angle(Hs_lp, deg=True), label='tf')
ax1.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutLPF], deg=True), label='ngspice')
ax1.set_ylabel("phase/deg")
ax1.set_title("low pass filter")
ax1.set_xlim([10, 10e3])
#ax1.set_ylim([-40, 25])
ax1.grid(True, which="both", ls="--")
ax1.legend()

ax2.semilogx(frequencies, np.angle(Hs_hp, deg=True), label='tf')
ax2.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutHPF], deg=True), label='ngspice')
ax2.set_title("high pass filter")
#ax1.set_xlim([10, 10e3])
#ax1.set_ylim([-40, 25])
ax2.grid(True, which="both", ls="--")
ax2.legend()

ax3.semilogx(frequencies, np.angle(Hs_bp, deg=True), label='tf')
ax3.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutBPF], deg=True), label='ngspice')
ax3.set_xlabel("frequency/Hz")
ax3.set_ylabel("phase/deg")
ax3.set_title("band pass filter")
#ax1.set_xlim([10, 10e3])
#ax1.set_ylim([-40, 25])
ax3.grid(True, which="both", ls="--")
ax3.legend()

ax4.semilogx(frequencies, np.angle(Hs_bs, deg=True), label='tf')
ax4.semilogx(sd.values[nfrequency], np.angle(sd.values[nvoutBSF], deg=True), label='ngspice')
ax4.set_xlabel("frequency/Hz")
ax4.set_title("band stop filter")
#ax1.set_xlim([10, 10e3])
```

2.2 Implementation (or Real Circuit)

```
#ax1.set_ylim([-40,25])
ax4.grid(True, which="both", ls="--")
ax4.legend()
```

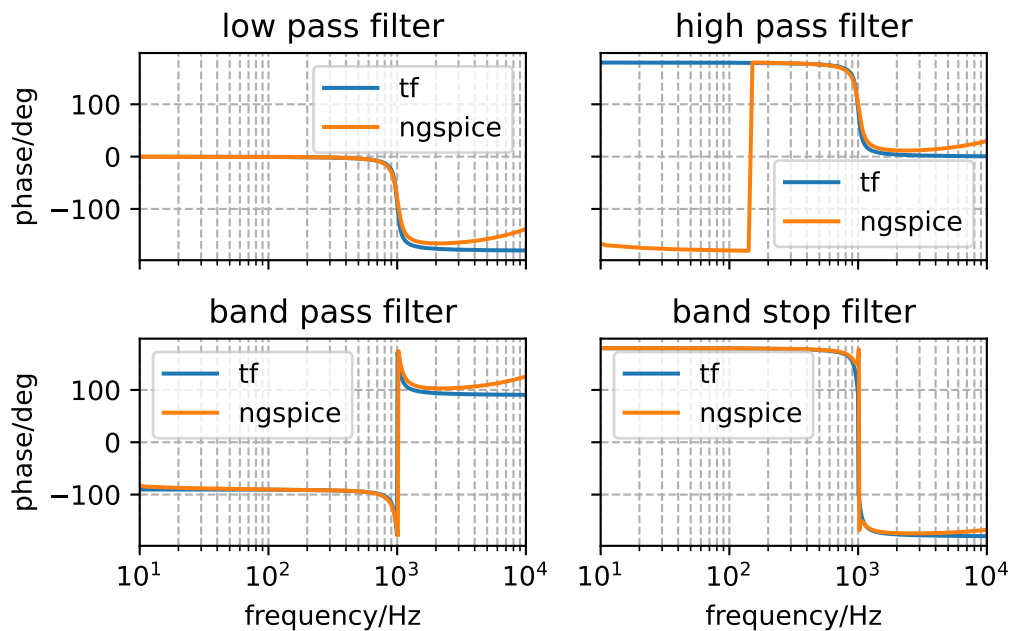
```
plt.tight_layout()
plt.show()
```

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1699: ComplexWarning:

Casting complex values to real discards the imaginary part

/opt/anaconda3/lib/python3.11/site-packages/matplotlib/cbook.py:1345: ComplexWarning:

Casting complex values to real discards the imaginary part



2.2 Implementation (or Real Circuit)

this is about the transition from the ideal circuit to real circuit

2.2.1 used opamp representation

like with opamp did we use specifically (5 ota i think), why, what are special about that

2 Characterisation

2.2.2 Sizing

with the script from pretl, if i remember correctly

3 Integrated Layouting

The following chapters describe how to design an integrated circuit using the software KLayout. An introduction and setup for the software will be given as well as some tips and tricks to easier work with the software.

3.1 Introduction into the fundamentals of integrated layouting.

Before designing a on the wafer, the basics of the design process must be understood. To that end an explanation of the software as well as the physical layers is necessary to comprehend the design challenges.

When designing an integrated chip, the designer first needs to choose a technology. Each technology has different design rules and properties, which makes it suitable for the application. A technology usually comes with a process design kit (PDK). A PDK contains a device library, verification checks and technology data.

The device library contains the symbols used in the schematic and the PCells/devices used in the layout. The verification checks consist of the design rule check (DRC) as well as the layout vs. schematic (LVS). The technology data represents the basis of DRC and defines the physical constraints of the technology (see Wikipedia contributors (2025))

The used technology was the sg13g2 technology from IHP. Which is a BiCMOS technology with a 130nm minimum gate length. This PDK is open source and in a preview status (see IHP-GmbH (2025a))

To design the layout itself one first needs to be familiar with the different masks and the structure of a waver. The following is a simplified explanation of the different layers, their use and the manufacturing process behind it. For this report this is kept simple and will only focus on the layers used in the design. The simplified explanation was heavily inspired by the chapters 2 to 5 in CMOS by Jacob Baker (see Baker (2010)).

The base is the bulk and a field oxide (FOX) layer on top of it. In this technology the bulk is a p-substrate. The FOX layer is an insulator, which separates devices from one another.

To make a device first the FOX layer needs to be opened (see Figure 3.1). That is done by etching away the FOX. The active mask allows the designer to specify the areas where the FOX will be opened. This is necessary for the doping process, which is usually done by an n- or p-select mask. In sg12g2 the n-select mask does not need to be specified. It is necessary to remove the FOX, otherwise the doping specified in the n-select mask will be stopped by the FOX. The select mask also needs to be bigger than the Active mask due to misalignment during the manufacturing process.

The next layer is the poly layer. The poly layer forms the gate of the transistors, the name is an abbreviation for Polysilicon. The poly can be routed like normal metal layers, with the exception that it is for more restive than the metal layers, so caution is advised for long poly traces.

3 Integrated Layouting

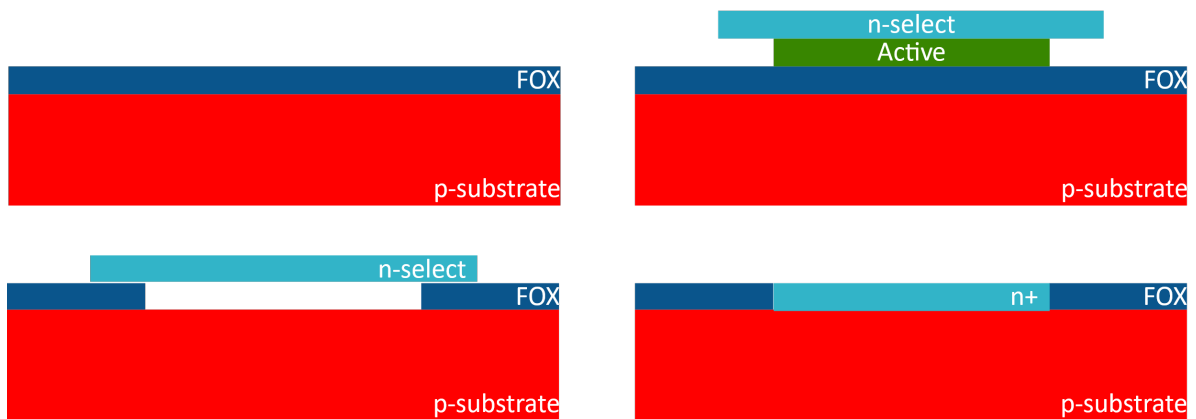


Figure 3.1: Doping process of the substrate

In the manufacturing process the thick poly layer prevents the n-select mask to dope the part below the gate, therefore creating the drain and source regions of the nmos (see Figure 3.2).

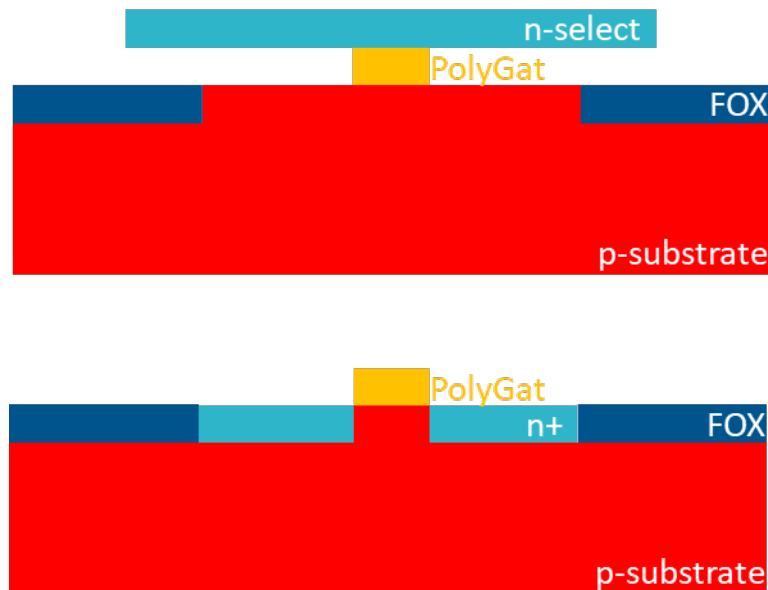


Figure 3.2: Figure : Manufacturing the poly layer in the process

Also note that the drain and source contacts are interchangeable, as they are identical regions in the active area. The last two pictures showed how an NMOS device is formed. As mentioned before, the n-select mask doesn't exist in the used PDK, but the p-select mask does exist with the label pSD.

As the PMOS is a negated NMOS, the designer first needs to place the p+ regions in an n-substrate. This is done by enveloping the transistor in an NWell. The NWell is a region that can be specified by the designer and works like an implant in the substrate. The FOX is also opened by the active layer but pSD layer is used to implant the

p+ region (see Figure 3.3).

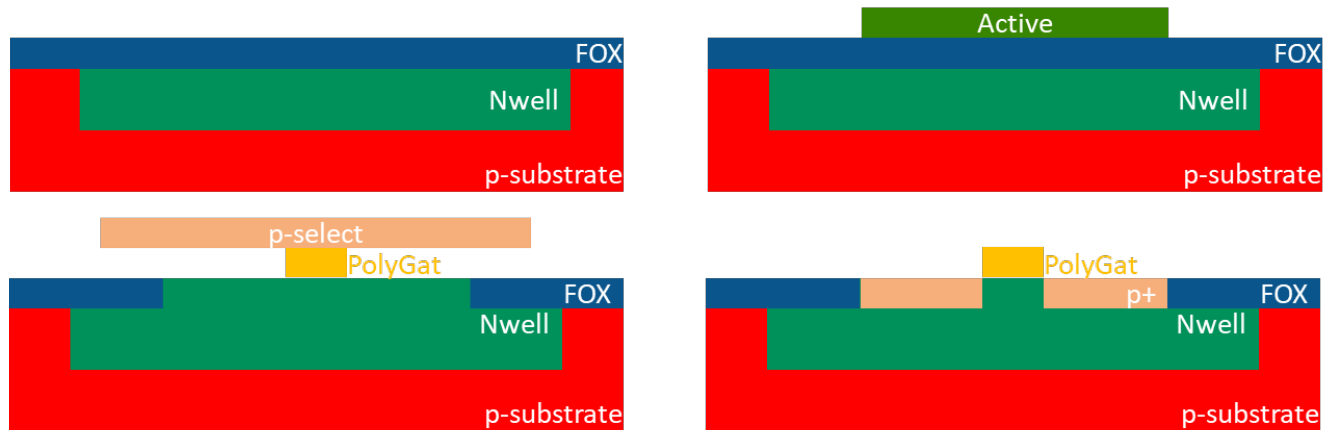


Figure 3.3: Side view of an PMOS transistor

The N- or PMOS is formed in the bulk, but there are no electrical connections to the pins of the device. Connections between devices can be established using the poly or active layer, but both options are not advised. Therefore, the first metal layer is used for connections between devices. Connections from an active or poly layer region to the metal layer can't be made by overlaying the layers as they are separated by an insulator. To connect these different layers the contact layer (Cnt) is used. It connects active or poly layer to the metal 1 layer, by removing the insulator over these regions (see Figure 3.4). Note that the first connection between these regions is always over the metal 1 layer. To switch between the different Metal layers, vias can be used, similar to a multilayer PCB design. These vias are formed by opening the insulator and placing in a conductor like tungsten the opening. In the figure this process is simplified.

The MOSFET is a 4-pin device, consisting of source, gate, drain and body. While the body often is omitted on a schematic level, in the integrated layout the body of the MOSFET needs to be tied to a defined potential. The body is also called the substrate or bulk, in case of a PMOS it is the NWell.

For the nmos this means connecting the substrate to VSS, while the NWell of a PMOS will be pulled to VDD. To have multiple locations to where the bulk or NWell connected to their potentials is advised.

The later chapters will work in the sg13g2 technology, the knowledge acquired in this chapter can be mapped to this technology. The figure (see Figure 3.5) compares the technology on the left, with the learned knowledge on the right. Note that the PWell is not used for NMOS devices only for ties.

3.2 KLayout

The previous chapter described the fundamentals of the integrated layout, in this chapter these fundamentals are put into practice to create the layout for the 5T-OTA outlined by the schematic by Harald Pretel (see Pretel et al. (2025)).

3 Integrated Layouting

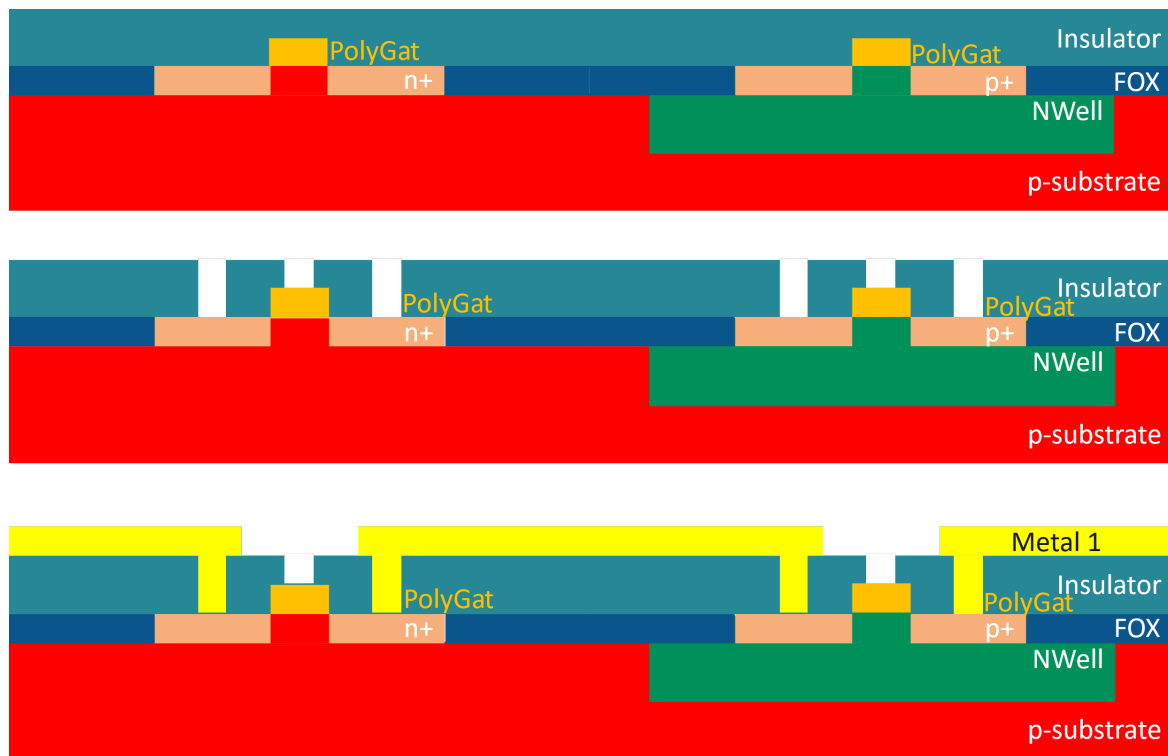


Figure 3.4: Metal connection to the drain and source contacts

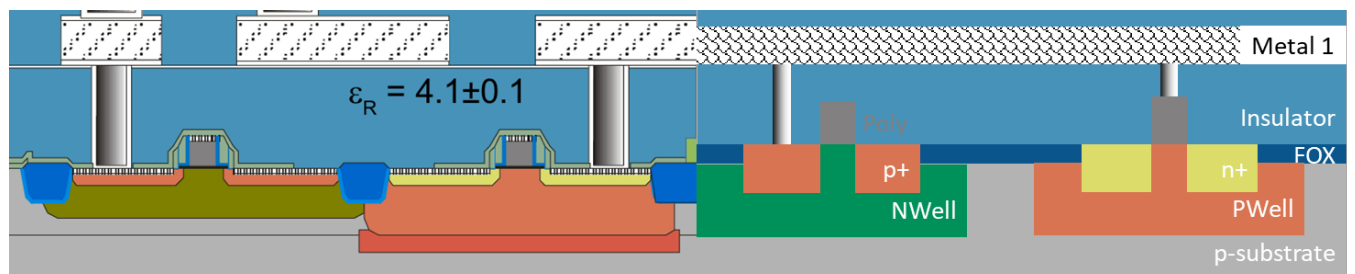


Figure 3.5: layer comparison between IHP and theory

The schematic outlined by Professor Pretel consists of 13 N- and PMOS devices. These devices are defined within the PDK and are the fundamental blocks, which make up the layout within the technology. Within the layout software these are defined as PCells. As the schematic designer works out the optimal design, one of the main variables to change the behaviour of the MOSFETs, is the w/l ratio. This ratio defines the width and length of the gate over the active area. These w/l ratios are crucial to the layout process, as they define the physical size of the devices. In the layout the w/l values were given by the calculation of Professor Pretel. These w/l ratios will be used to configure the PCells.

The layout program used is called KLayout. Similar to Xschem it works in a strict hierarchical system. As the layout of the 5T-OTA will be used in the Top-Level layout of the Biquad, the 5T-OTA itself is a cell, that can be called multiple times, throughout the layout of the Top-Level. To make it easier for the designer to connect the individual components of the Top-Level together, the 5T-OTA just uses the metal 1 layer. The interface points are the vias stacks. Therefore, making connections on the higher layers easier, as the individual cells for the 5T-OTA only reside on the most bottom layer.

In the following chapters DRC and LVS will be important concepts. DRC (Design Rule Check) describes the process of comparing the physical layout to the constraints of the technology. This will throw an error, for example if a "trace" on the Metal1 layer is too thin. LVS (Layout Versus Schematic) describes the process of comparing the schematic to the layout. This will throw an error when two nets are shorted or a device is not correctly connected.

3.2.1 The Setup of KLayout

While the previous chapters showed only a cross section of the wafer, the designer only sees the wafer from a top view, but needs to recall the cross section, while working on the layout. This is especially true for the layer that can be used as conductors.

This chapter will explain how to set up KLayout and the tools necessary to run DRC and LVS. It also shows an example way for a simple workflow.

Before the layout process begins it is paramount to aggregate all the data files necessary. To do that create a new folder labelled Layout and copy your schematic design into this folder. Consider renaming your schematic into letters only. Do not use special characters similar to UNIX symbols, only underscores are permitted.

To create a valid netlist, open the schematic in Xschem. Under *simulation->LVS* make sure that "*LVS netlist + Top level is a .subckt*" and "*Upper case .SUBCKT and .ENDS*" is selected.

After selecting these options, run the netlist and simulation. In the folder structure a new folder labelled simulations with file named "*YOUR_PROJECT.spice*" should appear. Copy that file into your main layout directory.

To begin designing create a new layout file called the same as the schematic by running "*Klayout -e*". The argument *-e* opens KLayout in editing mode. To make the editing mode default, select *File->Setup->Application->Editing Mode->Use editing mode by default*.

To create a new layout, select "*New Layout*". A wizard should appear that allows the selection of the technology. It is important to rename the Top Cell to match the name of your schematic so "*YOUR_PROJECT*".

Please note that all the scripts later used are case sensitive, so consider this when renaming files.

3 Integrated Layouting

Upon first saving, select the GDS file standard and name your file *YOUR_PROJECT.GDS*. The main layout directory should now contain *YOUR_PROJECT.sch/.spice/.gds* files. As well as a simulation folder. Now create a folder called LVS in this main layout directory.

For ease of access, create shell-script in your main layout folder to run LVS. The command saved with in the file should be:

```
python /foss/pdks/ihp-sg13g2/libs.tech/klayout/tech/lvs/run_lvs.py --netlist= YOUR_PROJECT.spice --layout=YOUR_PROJECT.gds --run_dir=LVS/
```

Try running this command, to ensure it runs correctly. If it fails, saying something about “use_same_circuit”, check the names of your files, including the cell name of your layout. It’s crucial that all files are named the same.

This script will produce a .lvsdb file in the LVS directory. This file contains the comparison between the netlist of the schematic, as well as the extracted netlist of the layout.

As the design process is a constant back and forth with design and running LVS and DRC, these steps are necessary to ensure a good workflow.

Before designing we need to set up the tools in Klayout. For the DRC, open *Macros->Macro Development* and select *DRC->[Technology sg13g2]->sg13g2_maximal*. Copy the contents of this script and paste them into a new DRC script. Running this script will provide the “*Marker Database Brower*” from which the individual DRC errors can be selected. As the error names and a semi-detailed description can be found the “*SG13G2_os_layout_rules.pdf*” on the Github of IHP (see IHP-GmbH (2025b)).

To run LVS run the shell script and open the netlist browser under *Tool->Netlist Browser*. To see the LVS errors click on *file->open* and then navigate to */LVS/YOUR_PROJECT.lvsdb*. This only needs to be done one time, as the LVS shell script will override the current .lvsdb file, when run anew. Select *File->Reload* to get the current LVS-data.

After loading the .lvsdb file the cross-reference tab will show the discrepancies between the netlists. This is because no layout has been created yet.

Some quality-of-life options

- To provide a better overview of the used layers, right click on the right-hand side layer tab to open a pop up menu and select “*hide empty layers*”.
- Disabling navigation by holding the middle mouse button: Check *File->Setup->Navigation->Zoom and Pan->Mouse alt mode*. This allows the control more similar to Autodesk Fusion or KiCad.
- Disabling the zoom when pasting: *File->Setup->Navigation->Zoom* and select *Pan->On paste->Pan to pasted objects*. This option disables the full zoom to object, which can cause disorientation.
- To show all the hierarchy and layers select *Display->Full Hierarchy*
- To change to dark mode select *File->Setup->Display->Background->Background Color->#42*

The real layout process can now begin in Klayout. Different from PCB design, the symbols on the schematic side do not need to be matched to a footprint. Rather the symbol are the fundamental devices in the technology.

To work with the devices of the technology open the library on the down-left under *Libraries->SG13_dev*. The list below shows the usable devices. By dragging and dropping them into the design, the devices are placed in the layout.

In previous chapters mentioned the w/l ratios will now play a role as the individual devices are configured. To configure a device or PCells, double click it and navigate to PCell parameters and enter the W/L ratios. Updating the PCell will change the device to the desired parameters.

3.3 The Layout Process

The workflow is much more incremental than in PCB design, as one implements device by device. To that end, it is practical to copy the original schematic and delete the already placed devices, to maintain an overview of the progress. After every two to three cells, it is advised to run LVS and DRC.

Placing and configuring a device was already described, connecting devices is usually done by the Metal1 layer. It is advised to work with the metal layer alongside the hierarchy. So for the top level use the higher metal layers like Metal3 to Metal6, while for the lower levels use Metal1 to Metal2.

To connect different pins, use the LAYER.drawing layer. Conductors can either be the GatPoly or the Metal layers. Keep in mind that GatPoly.drawing has a higher resistance than Metal.drawing. To switch between the metal layers, use the device called “via_stack” and configure it accordingly.

To connect the gate of a MOS device the GatPoly needs to be connected to Metal1 through the Cnt layer. In contrast to the source and drain contacts this need to be done by hand. See (Figure 3.6) for a DRC clean connection of the gate of an NMOS device.

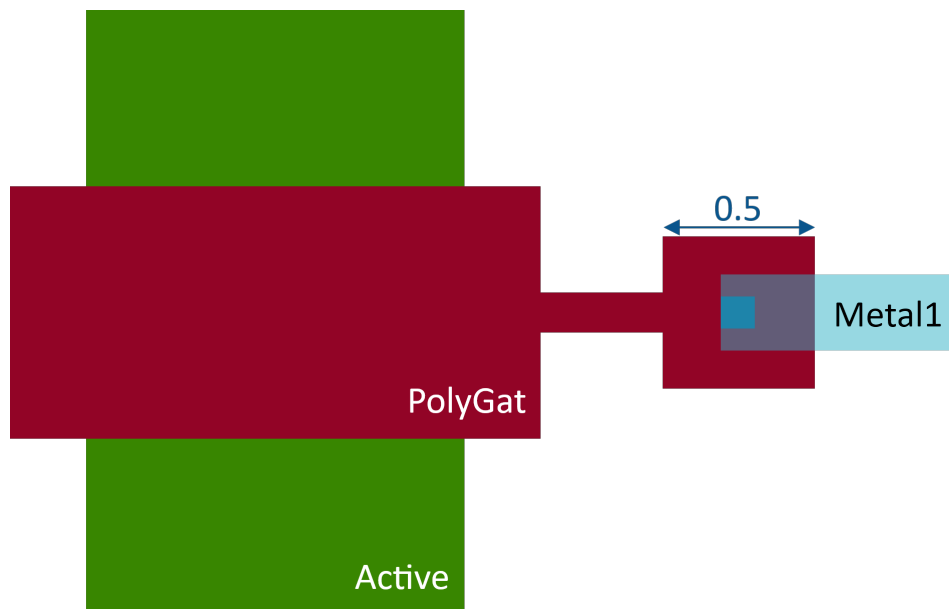


Figure 3.6: Example DRC clean connection of the gate

3 Integrated Layouting

In general, connect each device separately from each other. It might be tempting to design the layout close together, with overlapping NWells, Active and GatPoly areas, but this will lead to both DRC and LVS errors, which will be difficult to address as each device is close together. This is especially true, when first learning the process.

The layout of the 5T-OTA was done in multiple iterations, each iteration taught valuable lessons. The next section presents these lessons, outlining a specific workflow and offering tips and guidance for beginners.

Tips and tricks for the Layout Process.

- Run LVS and DRC every 1 to 3 devices!
- Aligning structures is easier when using a crosshair.
Enable this by checking View->Crosshair Cursor
- Use the path tool whenever possible.
 - The tool has an adjustable width, which can be set on the bottom right menu.
 - Set this width to 0.16 um to route DRC free on most layers.
 - Do not use free form, as jagged shaped will create DRC errors.
- Use the shift key to snap to constrain vertically or horizontally.
- To change the layer of structure, select the desired layer, select the structure and press shift+L
- Copying and pasting a structure will result in duplicating the structure in place. Use the move tool to separate both structures.
- If things get too cluttered use the layer menu to declutter the interface.
 - Right clicking and selecting “Visibility follows selection” will only show the selected layer.
 - Setting different tabs for routing, DRC checking or overviewing the general layout is a good option to keep things tidy and readable
 - Changing the appearance of different layers is a good idea, especially for layers that overlap, like pSD or NWell.
- Use the polygon only on special occasions as it is harder to clear of DRC errors.

Coming from discrete EDA software

- The devices will not be placed automatically.
- There is no way of changing the grid, as it is defined in technology.
- There is no “Ratsnest” or “Airal Connections”.
- There is no DRC in the background, short circuits will only be caught by running the LVS.
- There is no net highlighting.

- There are no zones, but their absence will cause DRC errors.
- There is no dragging, moving a device always means moving the “traces” by hand with it.
- There are no footprints, the PCells are the fundamental footprints defined by the technology.
- Text size can’t be set but will adjust to the zoom level.
- The GUI is sometimes not working, especially true for LVS.

3.3.1 Working with LVS during layouting

The Netlist Database Browser will provide an overview of all the nets both in the layout and the schematic or “Reference”. Each net has a number in brackets behind it. This number describes all the pins it is connected to. The netlists will only then match if these numbers are matched.

During the layout process it is paramount to use the LAYER.text layers and the text tool to assign labels to the GatPoly or Metal1 layers. To select these layers, one may need to show all layers under Layer->Show all. To keep the comparison in the netlist browser simple, name the nets of the layout according to the names of the nets in the schematic. Please refer to the designer’s etiquette, while labeling the layout or schematic (see Pretl, Koefinger, and Dorrer (2025)).

When working out the LVS it is advised to check mainly the devices in the list. This will provide a comprehensive overview of the devices, and the nets connected to them. Running LVS and DRC frequently will help with the overview.

It is normal that even the correctly wired devices are shown as errors. Especially the drain and source pins often swap places, which can’t be fixed by rotating the device. To clear the errors all nets connected to the device need to be cleared first. Only when all nets are correctly connected, will the device be clear of errors.

Having two nets on the same pin is always short circuit and should be handled with the highest priority.

After a few MOS devices are placed, LVS will throw an error as their body “pin” is not connected to a defined potential. In an prior chapter the process of tying down the substrate was already explained. In this technology the substrate is a p-substrate, this means the NWells need to be tied down and there must be multiple tying down points for the p-substrate (see Figure 3.7). Please note that the size of the masks shown has been altered for visibility reasons. Also note that each PMOS needs its own tie to the VDD potential, while the ties for the NMOS can be placed anywhere on the layout, as the substrate is the bulk.

While using the netlist browser allows also for a rudimentary net highlighting in the layout. The practicality depends on the number of devices connected to the net. In general troubleshooting get more difficult, the more pins are connected to the same net.

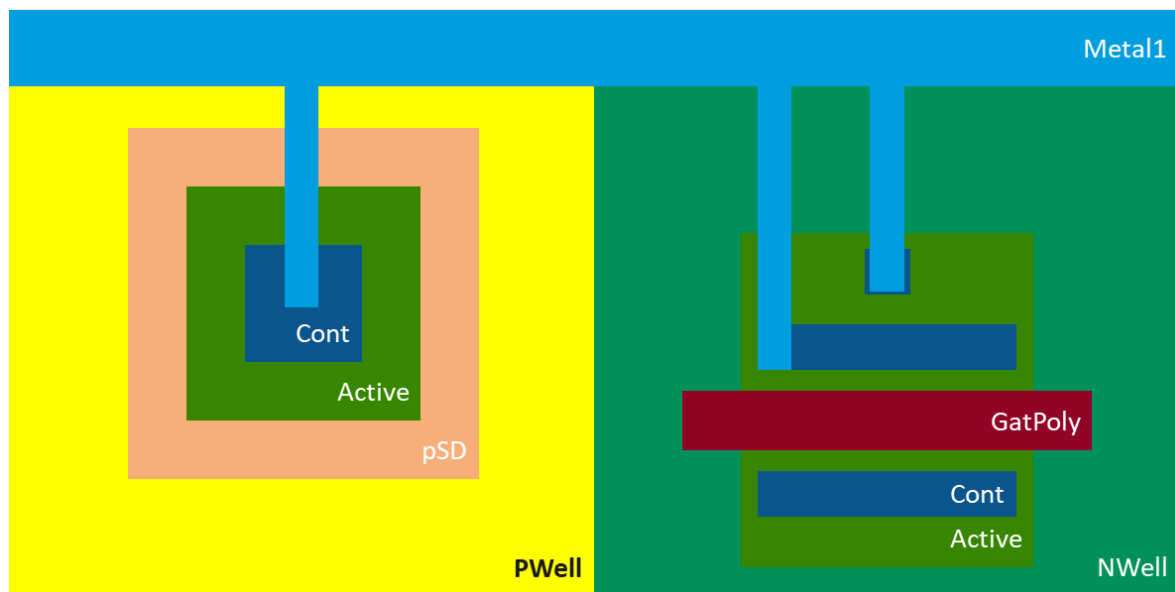


Figure 3.7: Tying down different types of bulks contacts. Not to scale.

3.3.2 Working with DRC during layouting.

Running the DRC often and addressing them as soon as possible, allows for a denser and generally more compact layout. It also reduces the time spend reworking the layout after all devices have been connected.

As described in set up chapter the “Marker Database Browser” is a built-in tool for addressing DRC errors. This tool shows the amount of DRC errors as well as the type. After following the steps described in the setup, one can open the Brower under Tool->Marker Browser. And running the DRC by selecting the run button.

The DRC will categorise in cells. Opening the DRC error log up will show the name of the error. Names link to the “SG13G2_os_layout_rules.pdf” by IHP. While KLayout also provides a description of the errors, it is best to use the DRC in conjunction with the DRC-PDF. The DRC-PDF provides context and dimensions for each error.

Working dually with the DRC-PDF and the DRC-Browser, allows the user to first clarify the DRC errors and than locate it by selecting it in the browser.

A very useful tool to clear DRC errors is the “partial” tool in the toolbar. This allows to modify structures after they are placed. This speed up the workflow immensely as the structures don’t need to redrawn.

Some DRC errors can’t be fixed at a given point of the design process. These DRC errors are usually due to density errors. But as the layout is not fully completed, cleaning these DRC errors has at best no impact on the design. Nevertheless, these errors must be cleared before the tape out.

3.4 The layout of the 5T-OTA

To design the 5T-OTA the schematic and the values of Professor Pretel were used (see Pretl et al. (2025)).

3.4 *The layout of the 5T-OTA*

As the export functionality of KLayout is rather limited, it is best to open the .gds file in KLayout directly, but there is also a SVG in the layout folder.

While not very space efficient the layout followed the location of the MOSFETS in the schematic to ensure readability. This helped during the layout process, as it gave a clear structure to the layout. This also benefitted the understanding of the LVS and DRC during the layout immensely.

As space is one of the main drivers of cost in IC design, a final version should be more mindful of the space used by the devices and traces. This would also clear the density errors, due to a higher density, because of the more compact design.

The design features multiple NWell and PWell Ties, to ensure that both bulks and NWell are connected to their potentials. Each PMOS-device has an NWell tie, which is closely located to the device. The PWell ties surround the design, as described in the introductory chapter to the integrated layout, the bulk shares one potential, but it is good practice to not use a star formation, to ensure this potential is evenly distributed.

One of the design goals of the design is the connection only on the first metal layer, this should ensure that the connection layers above are not influenced by the design. In addition, it provides a clear connection point for the top-level schematic.

The 5T-OTA.gds could easily be imported as a cell into an arbitrary top- or mid-level design. Which concludes the design goal of this report.

4 Filter Design

hello

TEST GIT

be free finn!

5 Discussion

the critical thinking place

5.1 Stability Analysis

5.2 Comparison

like between python, ideal, real, etc

5.3 is filter good?

review and evaluate

6 Conclusion

bla bla reflection

6.1 Outlook

what else could be done 'n stuff

Bibliography

- Baker, R. Jacob. 2010. *CMOS: Circuit Design, Layout, and Simulation*. 3rd ed. IEEE Press / Wiley.
- Fliege, Norbert. 1991. *Systhemtheorie*. 1st ed. Stuttgart: Teubner.
- H. Pretl and Michael Koefinger. 2025. "Analog Circuit Design." <https://iic-jku.github.io/analog-circuit-design/>.
- IHP-GmbH. 2025a. "IHP Open-source PDK: 130 Nm BiCMOS (SG13G2) Open Source Process Design Kit." <https://github.com/IHP-GmbH/IHP-Open-PDK>.
- . 2025b. "SG13G2_open-source_layout_rules.pdf." https://github.com/IHP-GmbH/IHP-Open-PDK/blob/main/ihp-sg13g2/libs.doc/doc/SG13G2_os_layout_rules.pdf.
- Pretl, Harald, Michael Koefinger, and Simon Dorrer. 2025. "Analog Circuit Design." <https://doi.org/10.5281/zenodo.14387481>.
- Pretl, Harald, Michael Koefinger, Simon Dorrer, and IIC-JKU. 2025. "Ota-5t.svg – 5-transistor OTA Schematic from the Analog Circuit Design Course." <https://github.com/iic-jku/analog-circuit-design/blob/main/xschem/ota-5t.svg>.
- Pretl, Harald, and Georg Zachl. 2025. "GitHub Repository of the IIC-OSIC-TOOLS." Zenodo. <https://doi.org/10.5281/zenodo.14634518>.
- Rao, K. R. K., and C. P. Ravikumar. 2012. *Analog System Lab Kit PRO MANUAL*. Texas Instruments.
- Razavi, Behzad. 2018. "The Biquadratic Filter." *IEEE Solid-State Circuit Magazine*, 11–16.
- . 2024. "The Design of a Biquadratic Filter." *IEEE Solid-State Circuit Magazine*, 6–13.
- Reisch, Michael. 2007. *Elektronische Bauelemente - Funktion, Grundsaltungen, Modellierung Mit SPICE*. 2nd ed. Stuttgart: Springer-Verlag.
- Renner, Nils. 2025. "Biquadratic IIR (SOS) Filters."
- Schmid, H. 2000. "Approximating the Universal Active Element." *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47 (11): 1160–69. <https://doi.org/10.1109/82.885124>.
- Wikipedia contributors. 2025. "Process Design Kit." https://de.wikipedia.org/wiki/Process_Design_Kit.

