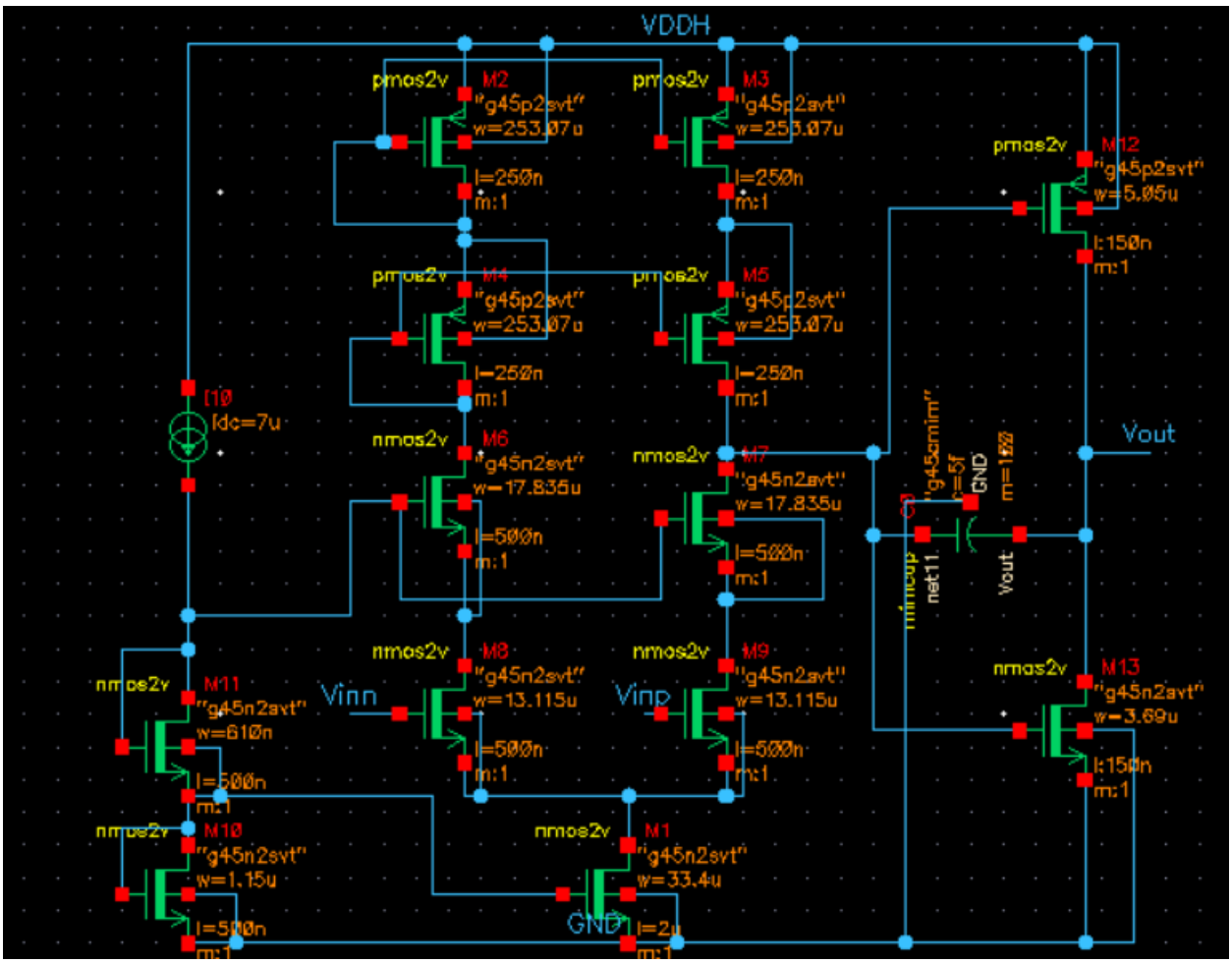
Overview

Schematic



Comparison Table

Closed Loop Gain	2	2	
Settling Time	180n	138.3n (5m)	179.1n (350m)
Total Error	0.2%	0.044%	
Power Consumption	2.5mW	801.6uW (5m)	694.9uW (350m)
Output Voltage Swing	1.4V	1.4V	
Max Current Mirror Ratio	10	~10	
Total Capacitance	10pF	500f F	
Total Resistance	200K	0	
CMRR at DC	70	73.69	

PSRR at DC (VDDH)	50	56.7	
Phase Margin	45°	65.4°	
Figure of Merit	2.22	9.017 (5m)	8.034 (350m)

My design consisted of two stages: the first stage is a telescopic cascode op-amp with a single-ended output to achieve high gain, and the second stage is a class AB output stage to ensure low quiescent current and to prevent slewing during the charging or discharging of the load capacitor. There is a miller compensation cap between the two stages to help phase margin. Since the PMOS transistors in the telescopic stage are diode-connected, the only two biasing points are for the tail current source and the second NMOS pair, just above the input NMOS pair. The bias points are generated using two stacked diode-connected NMOS transistors with a current mirror ratio of 7.25 for the tail current. I ended up not using VDDL since I initially wanted to use it for biasing my transistors but since the gate voltage of the second NMOS pair was above 1.1V I used VDDH for biasing as well and didn't have anything in my design requiring VDDL.

Design

To start with the design, I first found the closed-loop gain of the system and saw how much gain would be required to meet the error specification.

$$A_{CL} = \frac{\frac{2}{3}A_0}{1 + \frac{1}{3}A_0}$$

Using this, we get the static error as

$$\epsilon_s = 2 - A_{CL} = 2 - \frac{\frac{2}{3}A_0}{1 + \frac{1}{3}A_0}$$

To meet the error requirement of 0.2%, I decided to allocate 0.05% error to static error which gave me a gain requirement of 6000.

Since we had to meet a settling time of 180ns, and the equation of dynamic error being

$$\epsilon_d = e^{-\omega_u \cdot t_{settling}}$$

I allocated 0.1% error to dynamic error and found the required unity bandwidth as

$$\frac{\ln\left(\frac{0.1}{100}\cdot 2\right)}{-180\cdot 10^{-9}} \qquad 3.45256\text{E}7$$

In addition, we knew that our system is going to have at least 3 poles and 2 zeros because of the two amplifier stages and load C_L cap. Since the third pole was at a large enough frequency that it could be ignored (my estimations with G_{m2} = 2.5mS are below), I decided to not worry about it at this point.

$$\omega_{p3} = \frac{1}{(R_L || \frac{1}{G_{m2}})(C_S \parallel C_F)}$$

$$\frac{1}{500\cdot\frac{\frac{1}{2.5\cdot 10^{-3}}}{500+\frac{1}{2.5\cdot 10^{-3}}}\cdot\frac{50}{15}\cdot 10^{-12}} \qquad 1.35\text{E}9$$

For the second pole at

$$\omega_{p2} = \frac{G_{m2}}{C_L}$$

I decided to size G_{m2} such that this pole would be able to cancel with the zero from

$$\omega_{z1} = \frac{1}{R_L C_L}$$

At this point, the only poles and zeros remaining were the LHP zero from the miller compensation cap ω_{z2} and ω_{p1} discussed previously.
Since we have the freedom to size R_C to move ω_{z2} as desired, I decided that only considering ω_{p1} for my bandwidth should be enough which got me to the equation

$$\omega_u = \beta A_0 \omega_{p1}$$

$$\omega_{p1} = \frac{\omega_u}{\beta A_0} = \frac{\omega_u}{\frac{1}{2} \cdot 6000} \approx \frac{1}{R_1 G_{m2} R_2 C_C}$$

This gave me my first pole at 11.5k rad/s or 1.83 kHz.

After these calculations, I was ready to start building the first stage of my amplifier. For this, I wrote a very basic python script, which after many iterations over the course of the project turned into two functions.

```
if __name__ == "__main__":
    nch = importdata("nch_2v.mat")
    pch = importdata("pch_2v.mat")
    nch1v = importdata("nch_1v.mat")
    pch1v = importdata("pch_1v.mat")

    def doLookup(y: 1):
        vgs1 = look_up_vgs_vs_gm_id(nch, 15, vds=0.35, l=y) + .19
        vgs2 = look_up_vgs_vs_gm_id(nch, 15, vds=0.1, l=y) + vgs1
        print(vgs1, vgs2)
        for x in range(10, 71):
            i = x * 1e-6
            n_w1 = i / look_up_vs_gm_id(nch, 'ID_W', 15, vds=0.35, l=y)
            n_w2 = i / look_up_vs_gm_id(nch, 'ID_W', 15, vds=0.1, l=y)
            p_w1 = i / look_up_basic(pch, 'ID_W', vgs=0.4, vds=0.4, l=0.25)
            print(i , n_w1, n_w2, p_w1)
        return

    def doLookup2(y: 1):
        for x in range(50, 501):
            i = x * 1e-6
            n_w = i / look_up_vs_gm_id(nch, 'ID_W', 10, vgs=0.3, vds=0.9, l=y)
            p_w = i / look_up_vs_gm_id(pch, 'ID_W', 10, vgs = 0.3, vds=0.9, l=y)
            print(i, n_w, p_w )
        return
```

Generally, the way I did sizing was that I'd choose some current that I wanted to have flowing through my transistors, change the gm/Id and x range in the python function, and run doLookup with an appropriate length. The decision of what current to have flowing through the transistors was mostly based on whether it was high enough to not observe slewing at the output, and the value for gm/Id was mostly based on transistor width, since with large transistors I observed very high C_{dd} values which would adversely affect my phase margin due to additional poles. I initially had a high gm/Id of 22, but I lowered it first to 17 and then to 15 for my final design. As evident, doLookup was for the first stage, and doLookup2 was for sizing the second stage.

At the start I had planned to split my gain into 600 for the first stage and 10 for the second stage, and my final design is close to these values though not exactly the same.

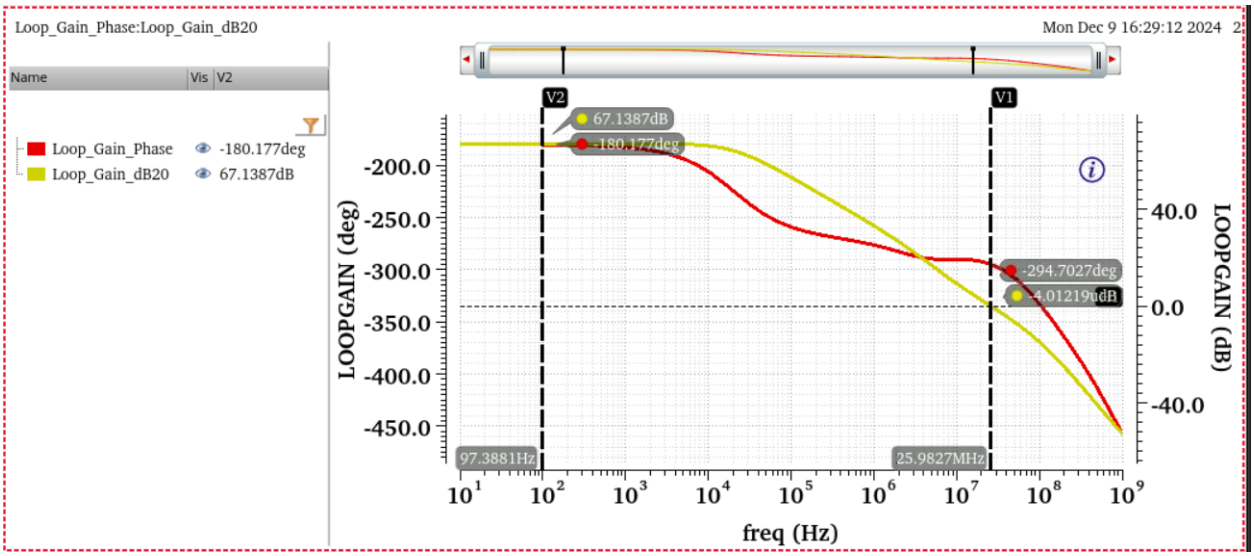
Transistors and Biasing

Transistor	W	L	Id	V_gs	gm	gds
M1 (tail curren[t)	33.4u	2u	69.37u	600.1m	675.3u	88.9u
M2 (top left PMOS)	253.06953u	250n	34.68u	424.3m	947.2u	23.06u
M3 (top right PMOS)	253.06953u	250n	34.69u	424.3m	947.3u	23.06u
M4 (bottom left PMOS)	253.06953u	250n	34.69u	424.3m	947.3u	23.06u
M5 (bottom right PMOS)	253.06953u	250n	34.69u	424.2m	947.3u	23u
M6 (top left NMOS)	17.83671u	500n	34.69u	535.3m	542.4u	32.01u
M7 (top right NMOS)	17.83671u	500n	34.69u	535.6m	542.2u	32.37u
M8 (bottom left NMOS)	13.11307u	500n	34.69	542.3m	507.6u	15.05u
M9 (bottom right NMOS)	13.11307u	500n	34.69u	542.3m	507.6u	15.05u
M10 (bottom current mirror)	1.15187u	500n	7u	600.1m	78.25u	2.215u
M11 (top current mirror)	610.15n	500n	7u	651.5m	60.48u	1.754u
M12 (output PMOS)	5.05u	150n	323.5u	852.7m	1.29m	61.65u
M13 (output NMOS)	3.69u	150n	424.9u	947.3m	1.633m	93.77u

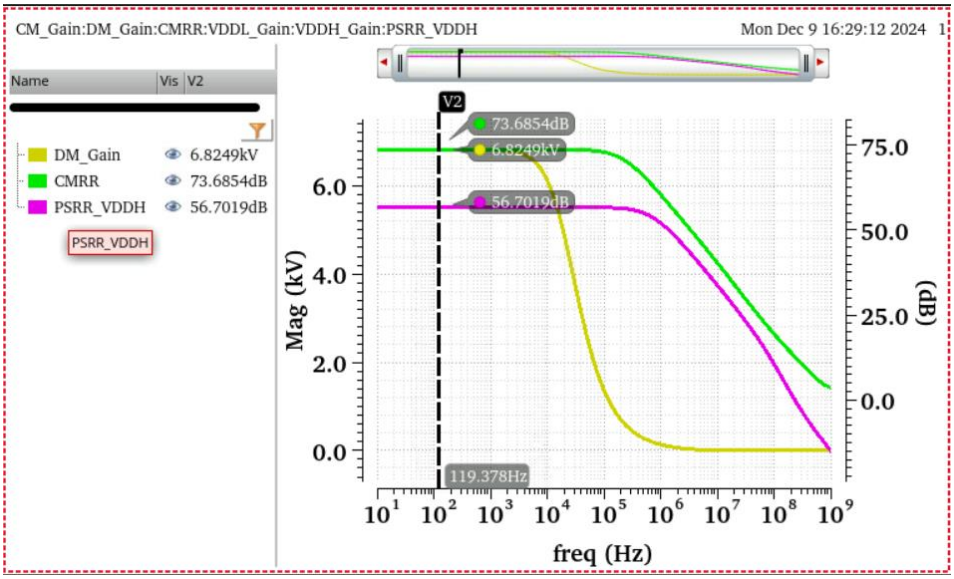
Discussion

AC Testbench

Loop Gain and Phase Margin

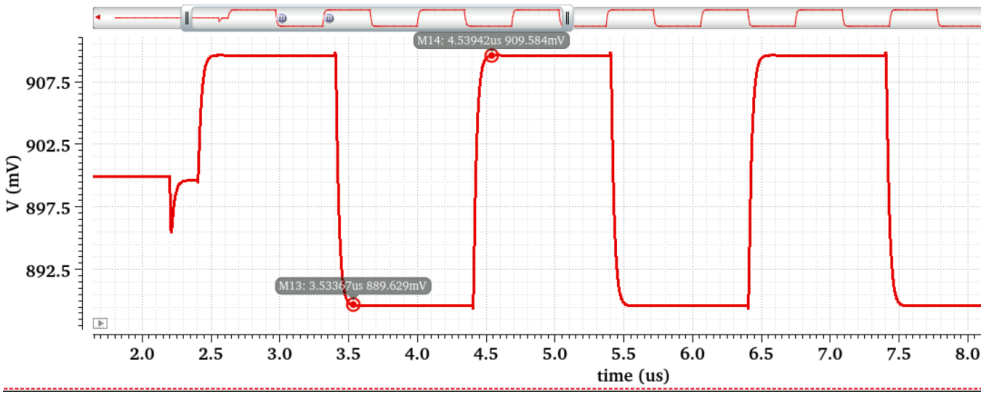


CMRR and PSRR vs frequency

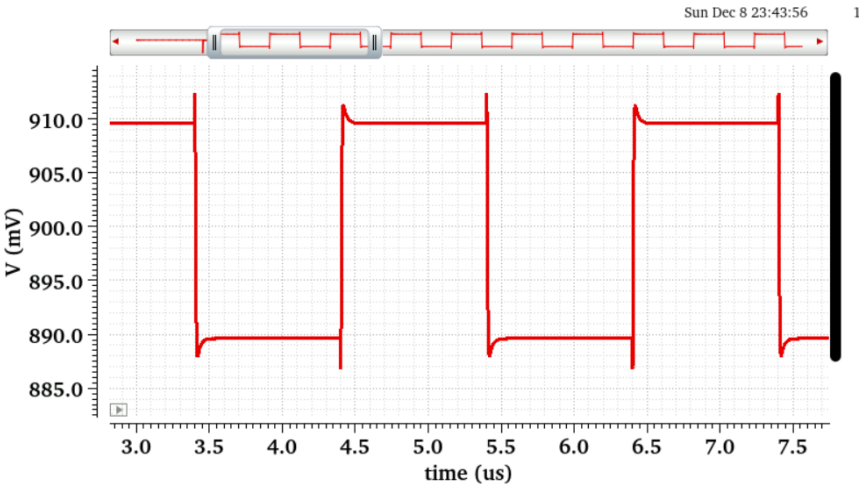


Transient Testbench

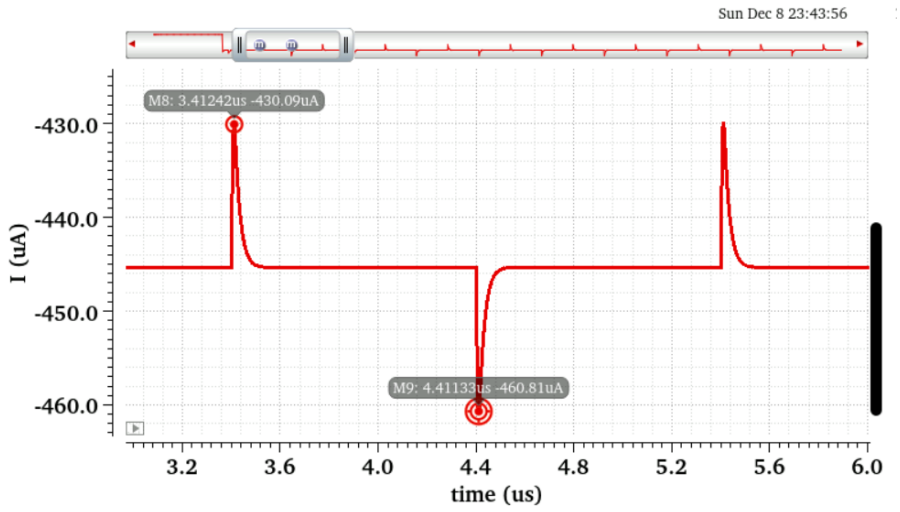
Vout_load (5m)



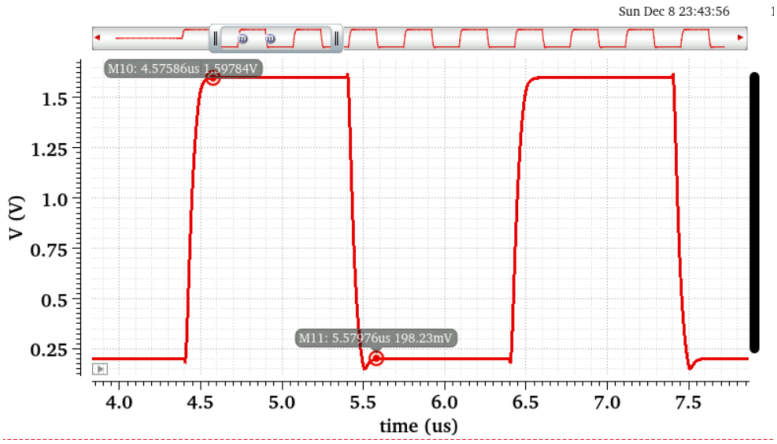
Vout (5m)



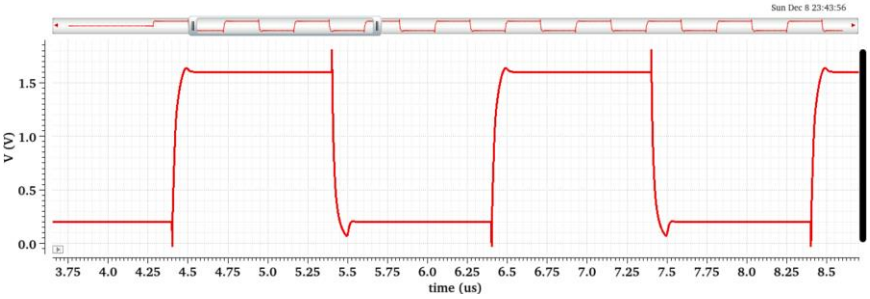
i_VDDH (5m)



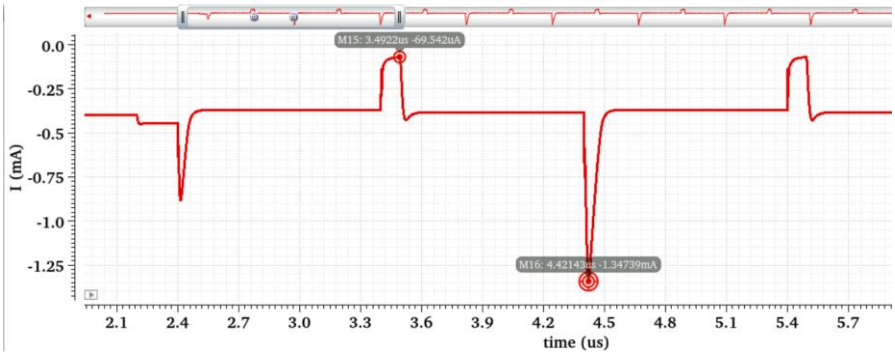
Vout Load (350m)



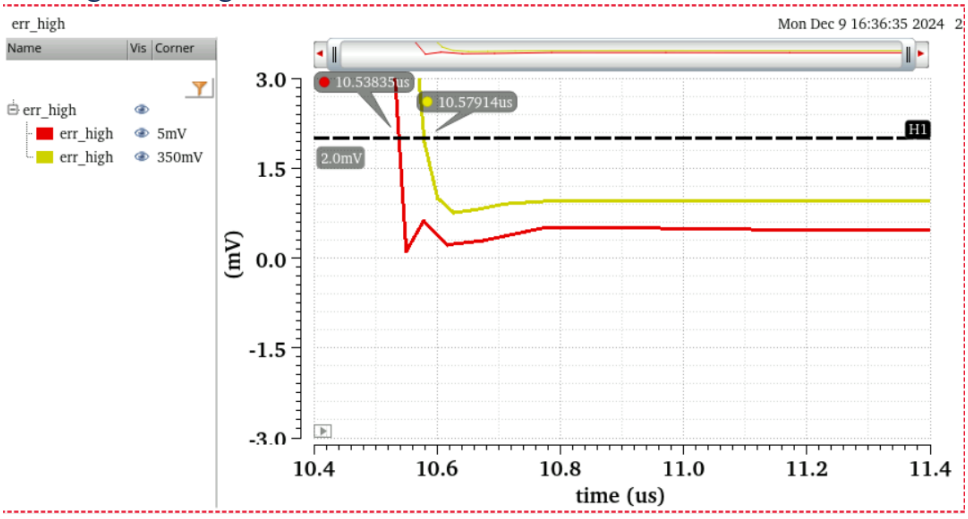
Vout (350m)



I_VDDH (350m)



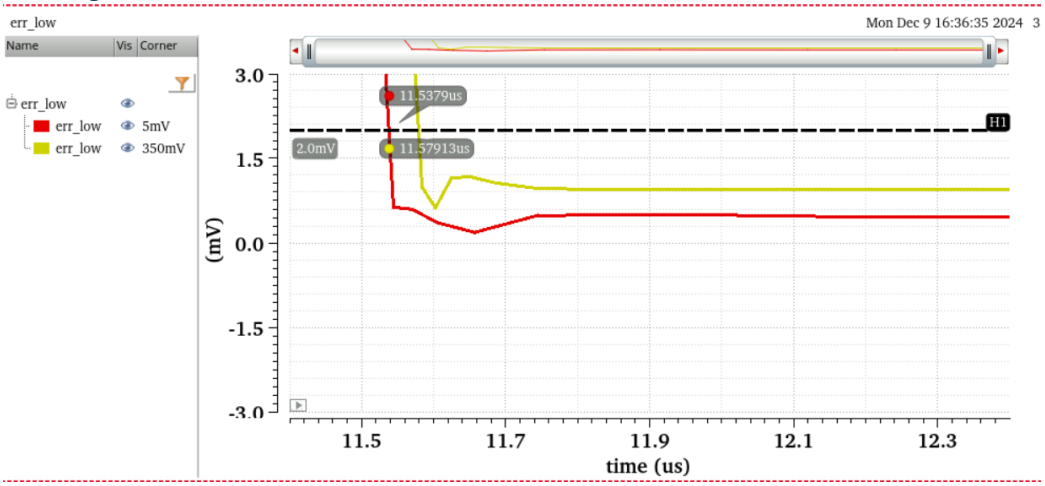
Settling Error High



Rise time (5mV) = 138.35ns

Rise time (350mV) = 179.14ns

Settling Error Low



Fall time (5mV) = 137.9ns

Rise time (350mV) = 179.13ns

Design Methodology

As I mentioned in a previous section, I did the sizing for my transistors using two very basic functions, and would iterate on the design based on their performance. For this, I created an additional testbench that would only measure the performance of my individual stages without any load added to it. The main purpose of this was to help me when designing the high gain stage

for the amplifier, but it ended up being generally useful for iterating on designs faster since I wouldn't need to run the entire testbench when I was just starting.

While my sizing methodology for the final design was pretty much what I had described in a previous section, I actually had two different sets of transistor sizes for my first stage, and the initial sizing was done very differently. I made an entire script to estimate the gain and power of the telescopic cascode stage given a certain transistor L. While the code is more scattered now since I didn't end up using it up until the end, here are the two main functions.

```
def findwidth(x):
    cdd = 0
    for i in range(50):
        gm = 4e7 * (CL + cdd)
        i = gm / gm_ID_range[x]
        n_W = i / n_id_w[y][x]
        p_W = i / p_id_w[y][x]
        cdd = n_W * n_cdd_w[y][x] + p_W * p_cdd_w[y][x]
    return i, n_W, p_W, x, gm

def findgain(i, n_W, p_W, x):
    n_id = n_id_w[y][x] * n_W
    p_id = p_id_w[y][x] * p_W
    print(n_id, p_id)
    p_gm = gm_ID_range[x] * i
    n_ro = 1 / (n_gds_w[x] * n_W )
    p_ro = 1 / (p_gds_w[x] * p_W)
    numerator = (n_gm + p_gm) * n_ro * p_ro
    denominator = n_ro + p_ro
    gain = numerator / denominator
    power = i * VDD
    return gain, power, p_gm
```

In this code, given a certain gm/ID value (index x) and a certain L value (index y), I'd find the gain and power consumption along with the widths of each transistor in the first stage. The way this works is that it first uses the unity gain bandwidth and C_C (written as C_L here but its value is 1pF, not 50pF) to find the required g_m. It uses this g_m and divides it by the chosen gm/Id to get the current, and then uses the current alongside the gm/Id value to get the widths for the NMOS and PMOS transistors. After getting the widths of the transistors, they're input to the findgain function which will return the gain for that stage and the power consumption. It uses the width from the previous function to calculate a value for g_{ds} and inverts that to get r_o, and then there is an expression for the gain of a telescopic cascode configuration (the image has the gain of the output stage, because I initially used this to size the output stage as well) as

$$A_v \approx \frac{g_{m1} \cdot (g_{m1} r_{o1} r_{o2}) \cdot (g_{m3} r_{o3} r_{o4})}{(g_{m1} r_{o1} r_{o2}) + (g_{m3} r_{o3} r_{o4})}$$

I used this script and used L = 2um the first time because I expected achieving a gain of 600 in one stage to be extremely difficult while also trying to manage with a low current. Though, after running the script I saw I was able to achieve higher gain with lower current, and from simulations my first stage ended up having a gain of 1.2k and a bandwidth of 385kHz. After finishing the sizing of the stage and verifying its gain, bandwidth, unity gain frequency, and DC operating points in my custom testbench, I tried using it by connecting it to a roughly sized output stage but unfortunately this had extremely high slew. After consulting with course staff, I realized that my sizing was not very good for our purposes even though I had very high gain and bandwidth since when I was sizing I assigned $\triangle$V through the input NMOS pair to be 0.2V while I should have assigned it an overdrive voltage of 0.35V or higher. The reason my $\triangle$V split for this stage initially was 0.2V for the two NMOS pairs, 0.3V for the two diode-connected PMOS pairs, and 0.5V for the tail current transistor is since the tail current transistor would have the most current flowing through it, I didn't want it to grow unreasonably large and assigned 0.5V overdrive to it, leaving me with 1.3V. Keeping the output swing requirement in mind I assigned 0.2V overdrive to the NMOS and decided to assign 0.3V to the PMOS since PMOS transistors have lower mobility than NMOS transistors. This meant that out of 1.8V, 1.5V was being used up as overdrive voltage for the transistors but this worked fine since my second stage had a gain I really only needed an output swing of 1.4 / 10 = 0.14V while I had 0.3V available. When doing this allocation I decided to diode-connect both PMOS pairs since I would be able to meet the output voltage swing spec even without using a wide-swing configuration in my first stage, and fewer biasing points would reduce the amount of current mirror biasing I'd need to do. The reason this stage didn't work well as it was that we wanted the overdrive voltage at the input pair to be at least 0.35V since those were really the transistors setting the current to flow through to the second stage. We could approximate the current going into the second stage and g_{m1} of the device as

$$i_0 = g_{m1} * v_{in}$$

$$g_{m1} = \frac{i_0}{v_{in}}$$

$$g_{m1} = \frac{i_0}{\Delta v}$$

Thus, course staff guided me and helped me realize that I really had to increase the overdrive voltage for my input pair to prevent this slewing.

Before sizing transistors, I had to reassign overdrive voltages and from last time I had noticed quite a few things which helped me make more refined decisions this time. I assigned 0.35V to the input pair from beginning. I decreased the overdrive voltage for the tail current source from 0.5V to 0.2V, to less than half of what it was because while building the telescopic cascode the first time I realized that it was quite feasible to achieve high gain with very low power, since I was able to achieve 1.2k gain with only 6uA flowing through my tail current source (3uA per branch, with a bandwidth of 9kHz), and this was also I pivoted from using my old sizing script to my newer script as meeting the gain spec for the first stage really was the easiest spec to meet in my opinion. I didn't require such a complicated script and just a script that gave me the widths and V_{GS} for a chosen I_{DS} was enough. Next, I increased the overdrive voltage for the PMOS pairs from 0.3V to 0.4V since those devices had very large dimensions previously. I also decreased the length of my NMOS transistors from 2u to 0.5u, and decrease the lengths of the PMOS pairs from 2u to 250n because their very large size was causing issues with my bandwidth and phase margin due to poles introduced from very large C_{dd} that I hadn't accounted for in my calculations. After reducing NMOS transistors lengths and gm/Id, I was able to decrease their widths by 100x and by increasing $\triangle V_P$ to 0.4 and length to 250n for PMOS transistors I reduced their width by 10x.

At this point, I had a first stage design that had a current of 70uA flowing through each branch and a gain close to 650. Next, I added a 1pF capacitor for miller compensation that I decided to keep fixed for the rest of the project (though my final value does end up being different and I'll cover why later). The miller compensation cap was sized at 1pF to provide enough phase margin, and also because I had done my hand calculations for the pole locations assuming a C_C = 1pF, and from sizing my first stage I was aware that changing C_C usually moves around multiple pole locations and it becomes difficult to keep track of what the previous and new locations for multiple poles are. I sized my output stage to have a quiescent current of 100uA and got my NMOS as 7.7um and PMOS as 8.8um with L=300n, however these values were not exactly a good fit since I did the sizing using a gm/Id of 10 but this did not explicitly consider the V_{GS} for the NMOS and PMOS. Since my first stage output had a DC bias point slightly higher than mid-rail, the PMOS transistor had a lower V_{GS} than the NMOS transistor,

and my rise time was slower than my fall time. I could either add some biasing, or increase the PMOS transistor's width to reduce the rise time. I decided to do the later since if I could get a design to meet spec without biasing for the class AB output stage that would be significantly easier. After running the sizing script considering the V_{GS} for the two devices, I got 15.1um for the width of my PMOS. After some tuning, I was able to get a width of 8um for the NMOS and 15.1um for the PMOS transistor and was able to meet spec with a FoM of about 5.4 for the 350m input.

The two major things I did from this point on were optimizing my design to increase FoM and replacing all ideal sources (such as the biasing for the second NMOS pair and the tail current source) with proper biasing using current mirrors. First going over optimizations, I tried reducing the current flowing through the cascode half-circuit until I observed slewing at the output, sizing the transistors using the same script along the way. This way, I was able to decrease the current flowing through each branch from 70uA to 35uA (I used a value of 30uA in the script, but after running simulations the actual current was 35uA), for a total decrease from 140uA to 70uA. While my design was mostly still functional, my settling time slightly increased when I did this, most likely due to the changing values for g_m and g_{ds}, but the settling time was still close enough to 180ns where I knew I could meet spec again by just finetuning the widths based on whether the rise time or fall time needs to decrease. For biasing, I had two bias points to consider: the input NMOS pair mentioned quite a few times now, and the bias for the tail current transistor since I had been using an ideal current source in parallel with a voltage source up until this point.

First, I sized a tail current transistor such that the current flowing through it is the sum of both branches and got W = 33.4um, L=2um, and V_{GS} = 0.6V to meet my requirements for current and $\triangle V$ = 0.19V. Now, I had to generate a 0.6V bias using a current mirror. Since the max current mirror ratio we could use was 10, I set a value of 7uA for an ideal current source in a branch and added a diode-connected NMOS with L = 500n. I decided to use a lower L than my actual tail current transistor (that had L=2um) to reduce the effect of parasitics, and with sizing using the script and some tuning I eventually found W = 1.152um to work well. This set me at a (W/L) ratio of 16.7 for the actual tail current transistor, and (W/L) ratio of 2.304 for my current mirror, and I was able to generate 0.6V at the drain of my diode-connected NMOS. Now, I only

had biasing for the second NMOS pair. I had to generate a V_{GS} of about 651mV and I added another diode-connected device and sized it using my script, with some tuning in virtuoso. This way, I was able to generate both 0.6V for the tail current source and 1.251V for my NMOS pair using diode-connected devices.

Now I had reached a point where the first stage of my design was not changing anymore. I was determined to increase FoM by only making changes in my second stage, and after iterating on the output stage NMOS and PMOS widths I realized that the capacitor C_C was really affecting my power dissipation. I tried reducing that cap from 1pF to 500fF and while this initially overshot my settling time because of 5mV due to ringing, I saw that reducing the widths of my output stage transistors allowed me to maintain a settling time that still meets spec while reducing power consumption. Another thing I did was reduce the lengths of my transistors from 300n to 150n, the minimum allowed in gpdk045 since smaller lengths correspond to smaller parasitic capacitances, thus likely lower settling times due to more current going to the load capacitor. After resizing the transistors considering these two facts, I successfully reduced my FoM to about 8. I tried reducing my miller compensation cap further but after various tries it proved to be difficult to find values for the NMOS and PMOS pair that met settling time, even though I had enough phase margin. I suspect this might be due to the quiescent current through my output stage decreasing while I was iterating on the transistors widths, leading to some slewing. At the end, I decided to stick to a C_C of 500fF, and got final output transistor sizes as 3.69u for the NMOS and 5.05u for the PMOS.

Particularly, I feel the most difficult spec to meet was settling time but the most difficult spec to optimize for the design competition for me was average power, since I was able to reduce settling time for 350mV by increasing my NMOS and PMOS widths in the output stage, leading to shorter fall times and rise times respectively, but this led to me firstly having higher average power consumption while also not meeting spec for 5mV which I believe is due to ringing. At the end, my final design had a power consumption of 695uW for 350mW, and 801.6uW for 5mV while meeting settling time requirements for both.