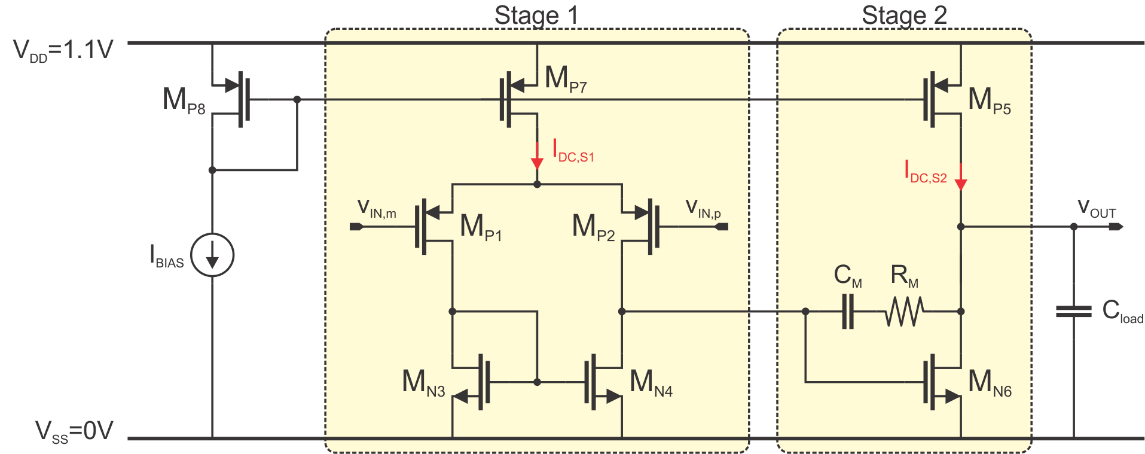
**Analog Electronic Circuits – 2020-2021**

**Design Project Report**

# Group data

|  |  |
| --- | --- |
| Group number | 9 |
| Name – Student 1 | Leander Hemelhof |
| Name – Student 2 | Vlad-Eusebiu Baciu |

# Goal:



Design of the 2stage OpAmp such that it passes the given specifications (insert your specs below):

|  |  |
| --- | --- |
|  | 30 |
| DC gain [dB] | 49 |
| [MHz] | 28 |
| Phase margin (PM) [deg] | > 70 |
| Output swing [V] | > 0.7 |

# Plan: Design of the 2-stage OpAmp on paper (10 points)

### Calculate the and L of each transistor (see exceptions under “Remark”) and the required and in the OpAmp circuit. For that, insert all your calculations as well as your -, -plots you used for your handcalculations below:

Hints:

* First, plot and across and gatelength L and do it again across , as presented in the 1stsession. Think about whether to create the plots for a PMOS or for a NMOS device.
* Then, based on those plots, start your calculations.
* Furthermore, you can assume the following:   
  (1) the OpAmp is designed in triple-well-technology (i.e. ; (2) and across ; (3)

Remarks:

* For Mp5, Mp7 and Mp8 you are not required to calculate the

Plots used for the handcalculations:

|  |
| --- |
|  |
| **Figure 1:** vs across gatelength |

|  |
| --- |
|  |
| **Figure 2:** vs across gatelength |

|  |
| --- |
|  |
| **Figure 3:** vs across gatelength |

|  |
| --- |
|  |
| **Figure 4:** vs across gatelength |

|  |
| --- |
|  |
| **Figure 5:** vs across gatelength |

|  |
| --- |
|  |
|  |

Handcalculations:

- Start with finding the correct stage 2 gain using different graphs:

First the assumption is made that the peaks on the gm/gds graphs will also be relatively optimal for the correct gain. Using the different NMOS graphs, a length of 200nm is chosen as a middle ground between gain, gm/ids efficiency and surface area. The working point (Vds of Mn6) is chosen to be VDD/2=0.55V since it will be close enough to the correct center between min and max output voltages to reach the output swing requirement, and this value makes calculations easier. Gain and power consumption both get better with weaker inversion, so the overdrive voltage of Mn6, Mp1 and Mp2 will probably be chosen in moderate to weak inversion. Looking at the NMOS gm/gds graph for shows a peak at around . This overdrive voltage will be chosen for Mn6

Dividing the gm/ids value at the chosen by the gm/gds at that gives gds/ids, a value that will be useful in a later step. Here for Mn6 at : gds/ids=0.456

Mp6 is chosen to be 1µm long with a close to -0.2V. The overdrive voltage is not exact due to the upper current mirror transistors having the same Vgs but not the same widths, leading to a difference in Vth. This difference is small enough to ignore in the hand calculations. Using the same graphs as in the previous points, but the PMOS versions this time, the gds/ids value of Mp5 can also be found. For Mp5: gds/ids = 8.37/37.56=0.223

Since ids5=ids6 by definition, the sum of the two values can be taken, giving (gds5+gds6)/ids:

(gds5+gds6)/(ids)=0.679

Dividing gm/ids of Mn6 by this value gives gm6/(gds5+gds6), the second stage gain.

gm6/(gds5+gds6) = Av2 = 18.12/0.679=26.8

Compared to the gm6/gds6 value that can be read from the NMOS gm/gds graph (39.78), it can be seen that gds5 is not negligibly small

- Calculate based on

As a small side note, the formula used for the dominant pole differs from the one used in the course notes in that one plus the second stage gain is used here. In the course notes the gain is assumed to be big enough that the plus one term is negligible, but with the found second stage gain leaving it out gives an overestimation of the location of the second loop pole, which is bad for both the GBW and phase margin.

The minimal and maximal output voltage can also be calculated:

- will be calculated based on the phase margin, but the location of the dominant pole is still needed for that. To find the location of the dominant pole the gain of stage 1 is also needed.

The first step is to calculate Vds3(=Vds4). Since the drain of Mp4 is connected to the gate of Mn6 Vds4=Vgs6, which can be read from the NMOS gm/gds graph in function of Vgs as the x value for which the value of the 200nm curve is the value found on the Vds=VDD/2 Vov graph at 0.1Vov.

Doing this gives Vgs6 = Vds3 = Vds4 = Vgs3 = Vgs4 = 0.3V

During the next steps an estimated value for Vgs1 = Vgs2 is needed. Normally this should not be difficult by doing it with the graphs like in the previous point. There is however a (small) Vds dependence on the Vth, analogously to channel-length modulation. This dependence is, however, small enough to be ignored for this part of the calculations. As will be explained in the next point only a rough estimate is needed. For PMOS there isn't really a well-defined peak to choose, so to get a decent gain a length of 200nm and Vov of 0V is chosen for Mp1 and Mp2. If needed the inversion level can be changed later, but with the margin of the second stage gain it should still be higher than needed.

Using both -VDD/2 Vds PMOS gm/gds graphs, analogously to the previous point, Vgs1 = Vgs2 = -0.28V

The source voltage will be found based on the common mode voltage. Since the total gain will be around 300, to get a peak-to-peak voltage of 0.7V at the output, the input peak-to-peak voltage is only a bit more than 2mV. Unless the common mode voltage is chosen very poorly the chance that this small voltage difference pulls something out of saturation is not big. A rough central value for the common mode voltage should be enough.

The maximal common mode input voltage can be found as the value that pulls Mp7 out of saturation. Mp7 has a length of 1µm and assumed Vov of -0.2Vov.

The minimal common mode voltage is the one that pulls Mp1 and Mp2 out of saturation.

A value near the middle between these two extreme values is chosen as common mode voltage:

This value was chosen to make Vds1 rounder.

With the common mode voltage chosen it is possible to calculate Vds1 = Vds2

With the needed voltages known the correct graphs can be made. In this case NMOS graphs with Vds=0.3V and PMOS graphs with Vds=-0.3V. based on these graphs the first stage gain be estimated, albeit roughly thanks to all the rounding in the earlier steps. It will still be more correct than ignoring the effect of gds3. Mn3 and Mn4 are both chosen to be 1µm like a current mirror.

Since Vds3=Vds4=Vgs3=Vgs4=Vgs6=0.3V, using the Vov and Vgs versions of the second set of NMOS graphs an overdrive voltage of 0.21V is found.

With the calculations being analogous to the second stage gain, only the values read from the graph will be given to keep things a bit more brief:

gm1/gds1=24.7 (0 Vov)  
gm1/ids=18.8 gds1/ids=0.76  
gm3/gds3=27.71  
gm3/ids=13.6 gds3/ids=0.49  
(gds1+gds3)/ids=1.25 gm1/(gds1+gds3)=15.0

This gives for the total gain: . This gain is bigger than the specification.

With the total gain more or less estimated the dominant pole can finally be placed.

Now that the dominant pole is placed, the phase margin can be calculated under the assumption of a two pole system. With some extreme choices it is possible that the third pole comes closer than the wanted second pole location, but here the assumption is made that this is not the case. This will later be checked using MATLAB. The required phase margin is 70°, but to account for small deviations due to the ignored parasitic capacitances a small extra margin of 2° is built in during these calculations.

The 0dB frequency is approximately equal to the gain bandwidth frequency, but will fall a little bit before that point taking into account the influences of the later poles and the fact that the calculated gain is expected to be an overestimation. This will, however improve the phase margin rather than worsen it, so it's not a problem.

The phase at this 0dB point should be -180°+72°=-108°

The formula for the second pole can be simplified with the knowledge that . This gives

With gm1, gm2 and gm6 known all the required currents can be calculated by using the relevant gm/ids graphs (gm/(gm/ids)=ids).

Using the NMOS gm/ids graph in function of Vov with Vds=Vdd/2:

Using the PMOS gm/ids graph in function of Vov with Vds=-0.3V:

Since the power consumption is proportional to the sum of the two terms and , to minimize the power consumption is chosen as the smallest of the two, 148µA. This gives the following for the power dissipation:

Although not relevant for the calculations, gm3 and gm4 can also be calculated using the Vds=0.3V NMOS gm/ids graph in function of Vov.

Vgs3=0.3V, so using both NMOS gm/gds graphs with Vds=0.3V gives Vov3=Vov4=0.214

The compensation resistance is used to place a zero. It might seem a good idea to use this zero to compensate the dominant pole, but in practice this is difficult to do. In the hand calculations this would not give any problems, but due to small deviance from the calculated values in for example a gm this zero would not completely cancel the pole, giving a really messy phase diagram. Therefore this zero is often chosen to be placed at infinity. This is done by choosing . As long as the deviations in gm6 stay relatively small this zero will stay high enough to not disrupt the behavior of the circuit.

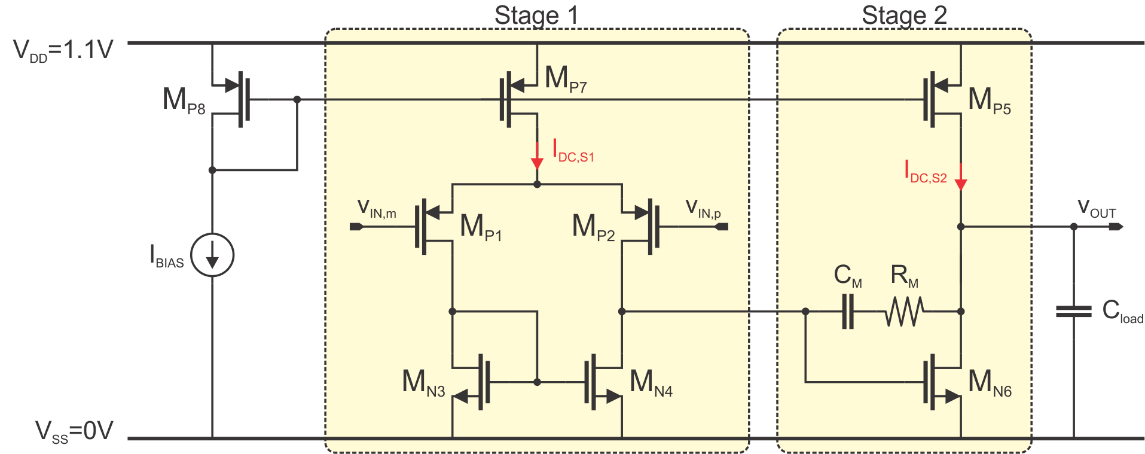
The main reason for adding this resistance is that without it the phase margin would only be a few degrees, and far from the 70 degrees required.

In the same vein as gm3 and gm4, Vds7 and Vgs7 can also be found:

- - Vov7 = -0.2V, so using a set of PMOS gm/gds graphs for Vov and Vgs, Vgs7 = -0.42V

### Place all the calculated voltages (in the blackbox) and all currents (in the red box) on the circuit depicted below and calculate also:

|  |  |
| --- | --- |
|  | -0.07V |
|  | 0.62V |
|  | 0.1V |
|  | 0.9V |
|  | 1.3mW |



0.68V

148µA

888µA

0.3V

0.55V

0.3V

148µA

0.6V

# Design: Implementation of the OpAmp in MATLAB and LTspice (10 points):

### Based on your handcalculations, create your OpAmp design in MATLAB. After completion, please insert your final and entire MATLAB code after the appendix (i.e. at the end of this report-document).

### *Remark:*

* *Please make sure that all widths W are below 1mm*

### Fill out both tables depicted below. All values are to be determined in MATLAB.

Device sizes and bias point parameters according to MATLAB

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Device** | **W**  **[μm]** | **L**  **[nm]** | **Ids**  **[μA]** | **VOV**  **[V]** | **gm**  **[S]** | **gds**  **[S]** | **gm/gds**  **[-]** | **Vds,sat**  **[V]** | **Vds**  **[V]** |
| Mp1 | 59.35 | 200 | 72.67 | 0 | 0.0014 | 55.43µ | 24.675 | -0.068 | -0.303 |
| Mp2 | 59.35 | 200 | 72.67 | 0 | 0.0014 | 55.43µ | 24.675 | -0.068 | -0.303 |
| Mn3 | 18.99 | 1000 | 72.67 | 0.2137 | 0.0010 | 35.75µ |  | 0.117 | 0.299 |
| Mn4 | 18.99 | 1000 | 72.67 | 0.2137 | 0.0010 | 35.75µ |  | 0.117 | 0.299 |
| Mp5 | 289.39 | 1000 | 886.77 | -0.1995 |  | 198µ |  | -0.175 | -0.550 |
| Mn6 | 110.70 | 200 | 886.77 | 0.1 | 0.016 | 405µ | 39.77 | 0.076 | 0.550 |
| Mp7 | 48.01 | 1000 | 145.33 | -0.2 |  | 35.66µ |  | -0.175 | -0.498 |
| Mp8 | 49.04 | 1000 | 145.33 | -0.2007 |  | 44.17µ |  | -0.176 | -0.418 |

|  |  |  |
| --- | --- | --- |
| **Device** | **Units** | **Value** |
| CM | pF | 7.5 |
| RM | Ω | 62.08 |
| IBIAS | A | 145.33µ |

### Based on the parameters filled in the table above, design your OpAmp in LTspice. After completion, please insert your final and entire LTspice-netlist after the appendix (i.e. at the end of this report-document).

# Experiment (10 points):

In this section you will simulate the following:

1. Frequency Response in MATLAB and LTspice
2. Noise contribution in LTspice
3. Linearity in LTspice

Note on how to present graphs in general:

* When you make a graph, put labels on every axis to make clear what you are showing. Also, show the units!
* When you plot multiple curves on one graph, add a legend.

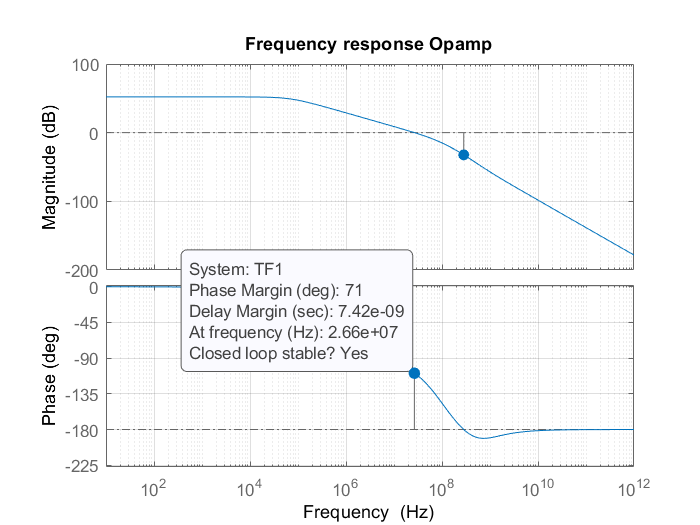
1. Frequency Response in MATLAB and LTspice

Note on how to present graphs in this sub-section:

* For magnitude plots use dB (linear scale) vs. Hz (logarithmic scale)
* for phase plots use ° (linear scale) vs. Hz (logarithmic scale).

### Simulate (small voltage gain) with and in MATLAB and LTspice. Then, paste both -curves in seperate plots below such that the difference between MATLAB and LTspice can be clearly seen. In addition to that, indicate the resulting phase margin (PM) in both plots.

**Plots:**





|  |  |
| --- | --- |
| **Simulator** | **PM (deg)** |
| MATLAB | 71 |
| LTspice | 72.78 |

### Explain, analyse and interpret the results in “6)”. Furthermore, if you observe significant differences between MATLAB and LTspice, explain why.

### MATLAB shows a small voltage gain of 52.1dB, in contrast to the small voltage gain of 50.2dB seen in LTspice. The output voltage is also around 250mV lower than expected. This difference is a bit bigger than initially expected, so a deeper look at the results are needed. The biggest base difference between the model used in MATLAB and the LTspice model are the threshold voltages of the different transistors. First the gains of the different stages are checked. Here it is noticed that the first stage gain is almost the same as in MATLAB, the second stage however lost around 2dB worth of gain.

### To illustrate why we think this happens, a small walk through the circuit starting from Mp1 and ending at Mn6 is made. Assume all the values are exactly like in MATLAB. Vth of Mp2 changes from -282mV to -278mV. With constant Vgs this leads to an (absolute) increase in Vov of 4mV, in this case 4mV more negative. This breaks the balance, so something needs to compensate this change. In practice both the Ids, Vds and Vgs will change by different amounts. Since gm >> gds, Vgs will change the most (here by 7mV), while Vds will stay mostly the same (here not even 1mV difference). The current also doesn't change much (less than 1µA) since Vds didn't change much and the Vth difference of Mp7 got compensated by Mp8 (Vgs8 is the only degree of freedom of Mp8 thanks to the current source and being diode connected, so the Vth difference of Mp8 got compensated, and since L7=L8 and W7 is close to W8 the Vth differences of Mp7 and Mp8 are almost the same. This means the Vth difference of Mp7 also got compensated). This means that for Mp1 (and Mp2) nothing really changed.

### For the bottom current mirror, let's take Mn4, Vds dropped by around 7mV. The difference in Vth is also around 7mV. Normally, the only way Mn4 can get back into a balanced state is by changing Vds (and therefore Vgs) and ids. This means that Mp1 and Mn4 work together to find a ids vds combination that works for both. Since the Vth difference of Mn4 is almost the same as for Mp1 the vds change of 7mV also balances Mn4 (and Mn3), so the current can stay close to the same.

### The Vth of Mn6 changed from 0.199V to 0.181V, a change of 18mV. Vgs6 also dropped by 7mV following the previous steps, so a part of this Vth change is already compensated, leaving a net increase in Vov of 11mV. Since Vgs6 can't be changed by Mn6 only Vds6 and ids6 will change to balance Mn6 and Mp5 again. It is expected that a higher Vov leads to a bigger drain current with the same Vds, but if the current increases Vds5 needs to become more negative with constant Vov, leading to a drop in Vds6. The combination of higher current and lower Vds6 is expected to lead to a bigger gm6 and bigger gds6. The relative change of gds6 is expected to be bigger than the relative change of gm, so a decreased gain would be found. Here the influence of changes in gds5 was ignored. This was done because Mp5 is in strong inversion, so it is expected that gds5 will not change as much as gds6. If we look at the MATLAB second stage gain of -26.7, an increase of 11mV in Vov of Mn6 would lead to around 294mV decrease in output voltage, while a decrease of 275mV was seen. A possible solution for this problem is changing the length of Mn3 and Mn4 to better match the length of Mn6. This way the Vth difference is roughly the same for both, leading to a smaller error on the output voltage in exchange for bigger gds3 and gds4, leading to a decreased first stage gain. Since it would not be in the spirit of the assignment to change the hand calculations and MATLAB code to improve LTspice performance without a globally justifiable reason the original length is kept. One such reason would maybe be the sensitivity to temperature changes, since transistors of the same size would have similar changes based on this temperature change. This, however, falls out of the scope of the assignment and would be difficult to test based on our current knowledge.

### The phase margin found in MATLAB is around 2 degrees smaller than in LTspice. The dominant pole is located around 70kHz in MATLAB, but in spice this pole is located around 80kHz. It is assumed that most of this shift comes from the decrease in the stage 2 gain, leading to a smaller capacitance of node 2 since is multiplied by 1 plus the stage 2 loop gain due to the Miller effect. The conductance between node 2 and ground stayed mostly the same, with an increase more so than a decrease, leading to a higher dominant pole frequency. This leaves two other influences on the phase margin, one of which is difficult to measure, namely the low frequency gain and the location of the second pole. The effect of the gain on the phase margin if the dominant pole would be independent from it is that if the gain becomes lower the phase margin goes up, since the 0dB point comes sooner and the phase change is independent from the gain. The dominant pole is however dependent on the second stage gain, like said earlier. But since both the second stage gain drops and the dominant pole moves to a higher frequency both influences would increase the phase margin. The second influence is the location of the second pole. Depending on how the parasitic capacitances changed between MATLAB and LTspice, which we can't look at, the second pole could move in any direction. Just looking at the formula used in MATLAB the decreased second stage gain should move this pole to a lower frequency, also increasing the phase margin. Sadly this pole is a bit difficult to measure in LTspice, so this influence will not be discussed in more detail.

### Simulate in LTspice for the following cases:

### No compensation network ().

### With compensation capacitor but no compensation resistor ().

### With both the compensation capacitor and the compensation resistor.

### Then, show all 3 cases in seperate plots such that the differences are clearly visible. Furthermore, indicate on each plot the resulting PM.

**Plot:**







|  |  |
| --- | --- |
| **Case** | **PM [deg]** |
| No compensation network | -1.64 |
| No compensation resistor | 68.43 |
| With both the compensation | 72.78 |

### Explain, analyse and interpret the results in “8)”.

1. Noise contribution in LTspice

### Simulate the output-referred noise voltage power density over an appropriate frequency range in LTspice. Then, insert the -plot below.

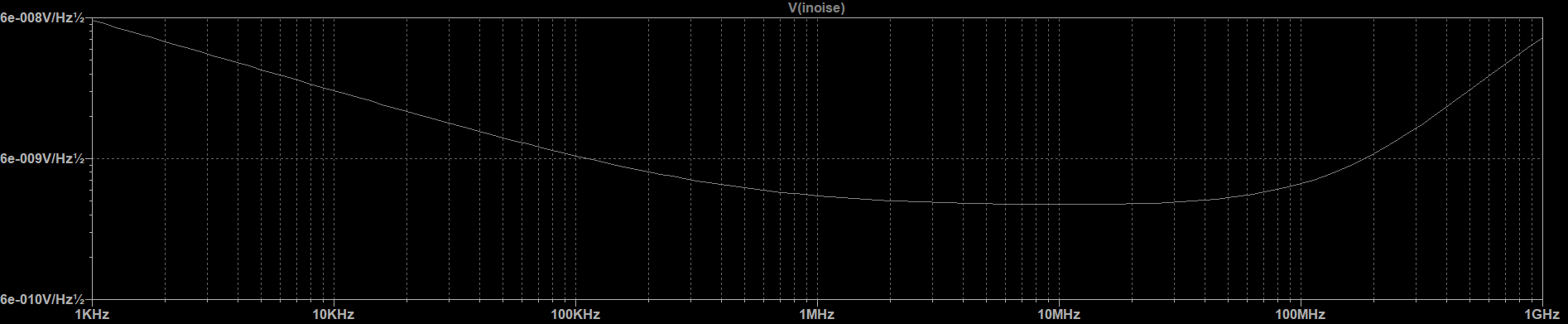
**Plots:**



### Explain, analyse and interpret the results in “10)”.

From low to high frequencies, the noise starts at a high point and initially drops like a 1/f dependency until a final noise level is reached. For the transistors there are two main noise source contributors, namely the drain thermal noise and the flicker (1/f) noise. Since the flicker noise is inversely proportional to the frequency, the behavior seen at the lower frequencies is expected. The noise level at the higher frequencies doesn't seem to change anymore with the frequency. This is because the drain thermal noise is not dependent on the frequency, so once the flicker noise is weak enough that it is no longer the dominant noise source the noise level is expected to stabilize. To be complete there is also the thermal noise of the compensation resistance, but this can be grouped with the drain thermal noise for this reasoning. It should also be noted that the thermal noise will, eventually, also drop off with the frequency increasing although the formulas don't state it. The exact reason for this is not relevant for this part, and will as such not be detailed here.

### Simulate the input-referred noise spectral density by using the .NOISE option in LTspice. Also, by using LTspice, determine the total output integrated noise of the OpAmp. For that, take the integration bandwidth from kHz to . Furthermore, copy and paste the top 9 contributing elements using "View->SPICE Error Log" option and insert that list below.





out\_totnvrms: INTEG(v(onoise))=0.000158929 FROM 1e+006 TO 1e+010

out\_mp1nvrms: INTEG(v(mp1))=8.88554e-005 FROM 1e+006 TO 1e+010

out\_mp2nvrms: INTEG(v(mp2))=8.89539e-005 FROM 1e+006 TO 1e+010

out\_mp7nvrms: INTEG(v(mp7))=3.72165e-006 FROM 1e+006 TO 1e+010

out\_mp8nvrms: INTEG(v(mp8))=9.14655e-006 FROM 1e+006 TO 1e+010

out\_mp5nvrms: INTEG(v(mp5))=6.80985e-006 FROM 1e+006 TO 1e+010

out\_mn6nvrms: INTEG(v(mn6))=1.01163e-005 FROM 1e+006 TO 1e+010

out\_mn4nvrms: INTEG(v(mn4))=7.03448e-005 FROM 1e+006 TO 1e+010

out\_mn3nvrms: INTEG(v(mn3))=6.42492e-005 FROM 1e+006 TO 1e+010

out\_rmnvrms: INTEG(v(rm)) =1.12803e-005 FROM 1e+006 TO 1e+010

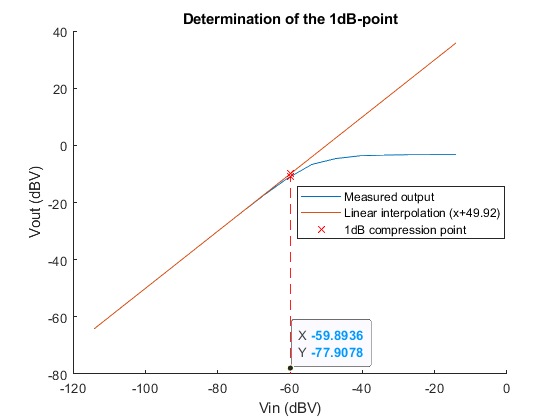
### Explain, analyse and interpret the results in “12)”.

Before the noise levels of different components are compared a small explanation is given about the noise sources in a transistor. The two most relevant noise sources in this case are the drain thermal noise and flicker (1/f) noise. The drain thermal noise current is proportional to gm in strong inversion with a few times bigger proportionality factor for small channel lengths. In weak inversion this drain thermal noise current is proportional to the drain current. The flicker noise is inversely proportional to the surface area and frequency. The four biggest noise contributors are the main four transistors of the first stage, Mp1, Mp2, Mn3 and Mn4. This is expected, since the influence of these transistors on the output is amplified by the second stage gain. It can also be noted that the noise from Mn3 and Mn4 is a bit lower than Mp1 and Mp2. This can be attributed to the fact that Mp1 and Mp2 have a lower surface area than Mn3 and Mn4 while having close to the same gm and drain current. The upper current mirror Mp8, Mp7 and Mp5 have long channels and are relatively big, looking at surface area, so their noise contributions will be lower than the rest. Mn6 has a noise contribution that is relatively big for not being amplified. Part of this comes from the gm being around 15 times bigger than gm1 and the drain current also being bigger while only having roughly double the surface area. The compensation resistance only has thermal noise proportional to the resistance value, which in this case is not that high.

1. Linearity in LTspice

### Simulate the output voltage amplitude and voltage gain as a function of the input voltage amplitude in LTspice. Then, insert the plots below and indicate the 1-dB compression point.

**Plots:**



### Explain, analyse and interpret the results in “14)”.

The linear interpolation has a slope of The first few points don't match nicely with the linear relation seen for the later points. This is attributed to the computing precision of LTSpice. After the linear part a slight bend is seen, ending in an almost flat part. Somewhere close to the start of the bend the 1-dB compression point is found, here at an amplitude of around 1.0mV (-59.89dBV). Based on the gain and output voltage given by LTSpice the input amplitude needed to start pulling Mn6 out of saturation is 0.200/323.6 = 0.618mV, so a value close to this is expected. The found value is a bit higher than we expected. Following this result the regular plots for the different amplitudes were again analyzed. There it was noticed that the DC output voltage also increases with the amplitude. This increase goes up by around 200mV close to the 1dB compression point and explains the difference. This increase is probably due to hitting the lower limit but still having a lot of room on the positive side. No further testing was done, so the exact reason is not found. It can also be noticed that the interpolation has a slope of 1 like expected and a constant term close to the gain (with a mistake of 0.3dB). With both axes in dbV this is expected.

# Conclusion (10 points):

### Conclude your experiment by filling the editable fields in the performance table depicted below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Metric** | **Units** | **Specification** | **from hand-calculations** | **MATLAB** | **LTspice** |
| DC gain | magnitude | 281.8 | 402 | 402.11 | 332.52 |
| DC gain | dB | 49 | 52.08 | 52.09 | 50.20 |
| Gain-Bandwidth frequency | MHz | 29 | 28 | 27.9 | 25.9 |
| Dominant pole frequency | kHz | 99.4 | 69.652 | 69.3 | 80.05 |
| PM | ° | > 70 | 72 | 71 | 72.78 |
|  | V |  | 0.55 | 0.56 | 0.56 |
|  | V | > 0.7 | 0.8 | 0.85 | 0.84 |
|  | mW |  | 1.3 | 1.3 | 1.3 |
| (output-referred noise) | Vrms |  |  |  | 159µ |

### Comment about the deviations between hand calculation, MATLAB and LTspice values found above. Conclude, what the causes of such deviations are.

In general the deviations between the hand calculations and MATLAB are minimal. Most differences can be attributed to rounding in de hand calculations, not exact readings from the graphs and the parasitic capacitances that were ignored in the hand calculations. Between MATLAB and LTSpice however there are a few relatively big differences, namely the gain, gain-bandwidth frequency, dominant pole frequency and phase margin. All these differences can be attributed to an effect that caused the second stage gain to drop by about 2dB that will be discussed in more detail. Of these differences there are two that are less of a problem than the others. The gain was designed with a lot of margin for these kinds of situations so it still reaches the specifications. The phase margin only got better thanks to these differences thanks to the factors discussed in more detail in an earlier point. The reason for the drop in second stage gain is already discussed in detail in an earlier part. Simply said the differences in threshold voltages between LTSpice and MATLAB cause disturbances in the designed balance that gets fixed by changes in currents and voltages, leading to a sharp decrease in Vds6, increase in Vov6 and Ids6 so that gm rises, but gds6 even more so the gain drops. Another reason for the differences is the fact that the parasitic capacitances will be different compared to MATLAB thanks to the lack of fingers in LTSpice. In most cases the load and Miller capacitance are big enough to drop these other influences, but depending on the size of the transistor this might still have a noticeable influence. Since LTSpice doesn't give information on these capacitances it is sadly not possible for us to check this hypothesis. The question remains if MATLAB or LTSpice is closer to reality. The points for LTSpice in this case are that the differences in threshold voltage kind of simulate small production differences between the transistors which makes it more realistic. The lack of fingers, however, makes this less realistic. MATLAB uses a detailed model based on detailed simulations and also uses fingers. This is a positive for MATLAB, but it still doesn't account for production differences. Neither accounts for things like temperature variations. To wrap up both results have merit, but it remains impossible to know exactly how the circuit will perform in reality without printing it and testing it out. It is, however still a good tool to have a good idea how it will behave.

### If you have results which are not passing the specifications: conclude for each of those result what the cause is and what you need to do in order to make it pass, if you were repeating this experiment.

As stated in the previous point the gain-bandwidth frequency specification is not met in LTSpice. A possible solution for this problem is changing the length of Mn3 and Mn4 to better match the length of Mn6. This way the Vth difference is roughly the same for both, leading to a smaller error on the output voltage in exchange for bigger gds3 and gds4, leading to a decreased first stage gain. This decrease would mostly be compensated by the fact that the second stage gain would increase closer to the original value, so it could even net a bigger gain. Our hypothesis is that this change will lead to a smaller difference in Vov6 and Vds6 so the balance of Mn6 needs to change less, keeping the second stage gain closer to the original. This should already help with the gain-bandwidth frequency, but will probably not fix the problem completely. The "best" solution is to build in some margin to compensate for the remaining difference, but this is only as a last resort since this will probably increase the power dissipation considerably among other increases. Another positive point about this solution is that it also compensates for differences by the parasitic capacitances that can't be accounted for without seeing the numbers.

# Appendix:

Insert your MATLAB code here:

%% Analog Electronics final project

clc;

%close all;

clear;

addpath(genpath('circuitDesign'));

addpath(genpath('functions'));

addpath(genpath('models'));

load ('models/UMC65\_RVT.mat');

%% Initializations

designkitName = 'umc65';

circuitTitle = 'Analog Design - Project';

elementList.nmos = {'Mn3','Mn4','Mn6'};

elementList.pmos = {'Mp1','Mp2','Mp5','Mp7','Mp8'};

choice.maxFingerWidth = 10e-6;

choice.minFingerWidth = 200e-9;

simulator = 'spectre';

simulFile = 0;

simulSkelFile = 0;

spec = [];

analog = cirInit('analog',circuitTitle,'top',elementList,spec,choice,...

designkitName,NRVT,PRVT,simulator,simulFile,simulSkelFile);

analog = cirCheckInChoice(analog, choice);

spec.VDD = 1.1;

%% Project: circuit

disp(' ');

disp(' VDD VDD VDD ');

disp(' | | | ');

disp(' Mp8-+---------Mp7---------------------Mp5 ');

disp(' |--+ | | ');

disp(' | +--+--+ node 3-> +-----+---OUT ');

disp(' | | | | | ');

disp(' | IN1--Mp1 Mp2--IN2 | | ');

disp(' | | | | | ');

disp(' | node 1->| |<-node 2 | Cl ');

disp(' | |--+ +------+-Cm---Rm----+ | ');

disp(' Ibias Mn3-+-Mn4 | | | ');

disp(' | | | +-----------Mn6 | ');

disp(' | | | | | ');

disp(' GND GND GND GND GND ');

%% AI: Implement your OpAmp according to your handcalculations

%% NMOS

nmos.plotEnable = 0;

% 1: gm/gds in function of Vov

% 2: gm/gds in function of Vgs

% 3: gm/Ids in function of Vov

nmos.option = 3;

nmos.printall = 1;

%% PMOS

pmos.plotEnable = 0;

% 1: gm/gds in function of Vov

% 2: gm/gds in function of Vgs

% 3: gm/Ids in function of Vov

pmos.option = 3;

pmos.printall = 1;

%% NMOS

%% Get gm/gds in function of Vov

if nmos.plotEnable == 1

VDD = spec.VDD;

VDS = 0.01:0.01:VDD;

VGS = 0.01:0.01:spec.VDD;

L = [65 80 100 150 200 400 500 1000]\*1e-9;

Mn6.vds = spec.VDD/2;

Mn6.vsb = 0;

result = [];

if nmos.option == 1 || nmos.printall == 1

figure

for ii=1:length(L)

Mn6.lg = L(ii);

Mn6.w = 10\*Mn6.lg;

Mn6.vth = tableValueWref('vth',NRVT,Mn6.lg,0,Mn6.vds,Mn6.vsb);

for jj=VGS%0:0.01:(VGS(end)-Mn6.vth)

Mn6.vov = jj - Mn6.vth;

Mn6.vgs = Mn6.vth + Mn6.vov;

Mn6 = mosNfingers(Mn6);

Mn6 = mosOpValues(Mn6);

result = [result Mn6.gm/Mn6.gds];

end

hold on

plot((VGS-Mn6.vth),result);

result = [];

end

xlabel('V\_{OV} [V]');

ylabel('g\_{m}/g\_{ds}');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/g\_{ds} in function of V\_{ov}(w=10L,Vds=VDD/2=0.55V). NMOS');

end

%% Get gm/gds in function of Vgs

if nmos.option == 2 || nmos.printall == 1

figure

for ii=1:length(L)

Mn6.lg = L(ii);

Mn6.w = 10\*Mn6.lg;

Mn6.vth = tableValueWref('vth',NRVT,Mn6.lg,0,Mn6.vds,Mn6.vsb);

for jj=0:0.01:VGS(end)

Mn6.vgs = jj;

Mn6.vov = Mn6.vgs - Mn6.vth;

Mn6 = mosNfingers(Mn6);

Mn6 = mosOpValues(Mn6);

result = [result Mn6.gm/Mn6.gds];

end

hold on

plot(0:0.01:VGS(end),result);

result = [];

end

xlabel('V\_{GS} [V]');

ylabel('g\_{m}/g\_{ds}');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/g\_{ds} in function of V\_{GS}(w=10L,Vds=VDD/2=0.55V). NMOS');

end

%% Get gm/Ids in function of Vov

if nmos.option == 3 || nmos.printall == 1

figure

for ii=1:length(L)

Mn6.lg = L(ii);

Mn6.w = 10\*Mn6.lg;

Mn6.vth = tableValueWref('vth',NRVT,Mn6.lg,0,Mn6.vds,Mn6.vsb);

for jj=VGS

Mn6.vov = jj - Mn6.vth;

Mn6.vgs = jj;

Mn6 = mosNfingers(Mn6);

Mn6 = mosOpValues(Mn6);

result = [result Mn6.gm/Mn6.ids];

end

hold on

plot(VGS-Mn6.vth,result);

result = [];

end

xlabel('V\_{OV} [V]');

ylabel('g\_{m}/I\_{ds} [1/V]');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/I\_{ds} in function of V\_{ov}(w=10L,Vds=VDD/2=0.55V). NMOS');

end

end

%% PMOS

%% Get gm/gds in function of Vov

if pmos.plotEnable == 1

VDD = spec.VDD;

VDS = 0.01:0.01:VDD;

VGS = 0.01:0.01:spec.VDD;

L = [65 80 100 150 200 400 500 1000]\*1e-9;

Mp7.vds = -0.3;%spec.VDD/2;

Mp7.vsb = 0;

result = [];

if pmos.option == 1 || pmos.printall == 1

figure

for ii=1:length(L)

Mp7.lg = L(ii);

Mp7.w = 10\*Mp7.lg;

Mp7.vth = tableValueWref('vth',PRVT,Mp7.lg,0,Mp7.vds,Mp7.vsb);

for jj=-VGS%((-1) \* (VGS(end) + Mp7.vth)):0.01:0;

Mp7.vov = jj-Mp7.vth;

Mp7.vgs = jj;

Mp7 = mosNfingers(Mp7);

Mp7 = mosOpValues(Mp7);

result = [result Mp7.gm/Mp7.gds];

end

hold on

plot(-VGS-Mp7.vth,result);

result = [];

end

xlabel('V\_{OV} [V]');

ylabel('g\_{m}/g\_{ds}');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/g\_{ds} in function of V\_{ov}(w=10L,Vds=-0.3V). PMOS');

end

%% Get gm/gds in function of Vgs

if pmos.option == 2 || pmos.printall == 1

figure

for ii=1:length(L)

Mp7.lg = L(ii);

Mp7.w = 10\*Mp7.lg;

Mp7.vth = tableValueWref('vth',PRVT,Mp7.lg,0,Mp7.vds,Mp7.vsb);

for jj=((-1) \* VGS(end)):0.01:0;

Mp7.vgs = jj;

Mp7.vov = Mp7.vgs - Mp7.vth;

Mp7 = mosNfingers(Mp7);

Mp7 = mosOpValues(Mp7);

result = [result Mp7.gm/Mp7.gds];

end

hold on

plot(((-1) \* VGS(end)):0.01:0,result);

result = [];

end

xlabel('V\_{gs} [V]');

ylabel('g\_{m}/g\_{ds}');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/g\_{ds} in function of V\_{GS}(w=10L,Vds=-0.3V). PMOS');

end

%% Get gm/Ids in function of Vov

if pmos.option == 3 || pmos.printall == 1

figure

for ii=1:length(L)

Mp7.lg = L(ii);

Mp7.w = 10\*Mp7.lg;

Mp7.vth = tableValueWref('vth',PRVT,Mp7.lg,0,Mp7.vds,Mp7.vsb);

for jj=-VGS

Mp7.vgs = jj;

Mp7.vov = jj-Mp7.vth;

Mp7 = mosNfingers(Mp7);

Mp7 = mosOpValues(Mp7);

result = [result Mp7.gm/Mp7.ids];

end

hold on

plot(-VGS-Mp7.vth,result);

result = [];

end

xlabel('V\_{OV} [V]');

ylabel('g\_{m}/I\_{ds} [1/V]');

legend('65nm','80nm','100nm','150nm','200nm','400nm','500nm','1um');

title('g\_{m}/I\_{ds} in function of V\_{ov}(w=10L,Vds=-0.3V). PMOS');

end

end

fgbw = 28e6;

AvdB = 49;

Av = 10^(AvdB/20);

CL = 30e-12;

pd = 2\*pi\*fgbw/(15\*26.8);

Mp1.gm = 2\*pi\*fgbw/26.8 \* (CL/4\*26.8 + CL/4);

angle = -(-108+180/pi\*atan(2\*pi\*fgbw/pd));

p2 = 2\*pi\*fgbw/tan(angle\*pi/180);

Mn6.gm = p2\*CL;

Mp7.vov = -0.2;

Mn6.lg = 200e-9;%200e-9;

Mp1.vsb = 0;

Mp1.lg = 200e-9;

Mp7.vsb = 0;

Mp8.vsb = 0;

Mp5.vsb = 0;

Mp5.lg = 1e-6;

Mp8.lg = 1e-6;

Mp7.lg = 1e-6;

Mp1.vov = -0.00;%-0.02;%-0.2;

Mn6.vov = 0.1;%0.17;

Vout\_wp = spec.VDD/2;%(1.1-0.2+Mn6.vov)/2;%(1.1-0.2 + 0.17)/2;

Mn6.vsb = 0;

Mn6.vds = Vout\_wp;

Mn6.vth = tableValueWref('vth', NRVT, Mn6.lg, 0, Mn6.vds, Mn6.vsb);

Mn6.vgs = Mn6.vov + Mn6.vth;

Mn6.w = mosWidth('gm', Mn6.gm, Mn6);

Mn6 = mosNfingers(Mn6);

Mn6 = mosOpValues(Mn6);

Mp5.vds = Vout\_wp - spec.VDD;

Mn4.vsb = 0;

Mn4.vds = Mn6.vgs;

Mn4.vgs = Mn4.vds;

Mn4.lg = 1e-6;%200e-9;%1e-6;

Mn4.vth = tableValueWref('vth', NRVT, Mn4.lg, Mn4.vgs, Mn4.vds, Mn4.vsb);

Mn4.vov = Mn4.vgs-Mn4.vth;

% Estimate Vth with Vds=0, correct later

Mp1.vth = tableValueWref('vth', PRVT, Mp1.lg, 0, 0, Mp1.vsb);

Mp1.vgs = Mp1.vov + Mp1.vth;

Vcm\_in\_max = Mp1.vgs + Mp7.vov + spec.VDD

Vcm\_in\_min = Mp1.vgs - Mp1.vov + Mn4.vov

Vcm = 0.32;%(Vcm\_in\_max + Vcm\_in\_min) / 2;

Mp1.vds = Mn6.vgs - (-Mp1.vgs+Vcm);

Mp1.vth = tableValueWref('vth', PRVT, Mp1.lg, 0, Mp1.vds, Mp1.vsb);

Mp1.vgs = Mp1.vov + Mp1.vth;

% Recalculate the Vds with the more correct Vgs value

Mp1.vds = Mn6.vgs - (-Mp1.vgs+Vcm);

Mp1.w = mosWidth('gm', Mp1.gm, Mp1);

Mp1 = mosNfingers(Mp1);

Mp1 = mosOpValues(Mp1);

Mp2 = cirElementCopy(Mp1, Mp2);

Mn4.ids = Mp1.ids;

Mn4.w = mosWidth('ids', Mn4.ids, Mn4);

Mn4 = mosNfingers(Mn4);

Mn4 = mosOpValues(Mn4);

Mn3 = cirElementCopy(Mn4, Mn3);

Mp7.vds = Vcm - Mp1.vgs - spec.VDD;

Mp7.vth = tableValueWref('vth', PRVT, Mp7.lg, 0, Mp7.vds, Mp5.vsb);

Mp7.vgs = Mp7.vov + Mp7.vth;

Mp5.ids = Mn6.ids;

Mp5.vgs = Mp7.vgs;

Mp8.vgs = Mp7.vgs;

Mp5.w = mosWidth('ids', Mp5.ids, Mp5);

Mp5 = mosNfingers(Mp5);

Mp8.vds = Mp8.vgs;

Mp7.ids = 2\*Mp1.ids;

Mp7.w = mosWidth('ids', Mp7.ids, Mp7);

Mp7 = mosNfingers(Mp7);

Mp8.ids = Mp7.ids;

Mp8.w = mosWidth('ids', Mp8.ids, Mp8);

Mp8 = mosNfingers(Mp8);

Mp5 = mosOpValues(Mp5);

Mp7 = mosOpValues(Mp7);

Mp8 = mosOpValues(Mp8);

% %% AI: Fill out the empty variables required to plot the transfer-function.

% % meaning of each variable see comment and

% % location of nodes see line 31

%

AvDC1 = Mp1.gm/(Mp1.gds+Mn3.gds); % DC gain 1st stage

AvDC2 = Mn6.gm/(Mn6.gds+Mp5.gds); % DC gain 2nd stage

AvDC = AvDC1\*AvDC2;

AvdBMade = db(AvDC);

C1 = Mp1.cgd+Mp1.cdb+2\*Mn3.cgs+2\*Mn3.cgb+Mn3.cgd; % Capacitance on node 1

G1 = Mn3.gds+Mp1.gds+Mn3.gm; % Admittance on node 1

C2 = (1+AvDC2)\*(CL/4+Mn6.cgd); % Capacitance on node 2

G2 = Mn4.gds+Mp2.gds; % Admittance on node 2

CLp = CL+Mn6.cdb+Mp5.cdb+Mn4.cgd;

Cn2 = Mn4.cdb+Mn4.cgd+Mp2.cdb+Mp2.cgd;

C3 = (CL/4\*(CLp+Cn2)+CLp\*Cn2)/((1+AvDC2)\*(CL/4));

G3 = (Mn6.gds+Mp5.gds);

%Vcm\_in\_min = Mp1.vgs-Mp1.vdsat+Mn3.vdsat;

%Vcm\_in\_max = spec.VDD+Mp1.vgs+Mp7.vdsat;

%Vcm = (Vcm\_in\_max + Vcm\_in\_min) / 2;

%C3 = CL+Mn6.cdb+Mp5.cdb+(1+AvDC2)\*CL/4; % Capacitance on node 3

%G3 = Mp5.gds+Mn6.gds; % Admittance on node 3

%

% %% AI: Set-up Rm, Cc and CL and calculate the zero required for the transfer-fct

%

spec.Cm = 30e-12/4;

spec.Cl = 30e-12;

spec.Rm = 1/Mn6.gm;

z1 = inf;

%

% %% AI: Fill out the empty variables required for the performance summary

Vin\_cm\_min = Mp1.vgs - Mp1.vov + Mn6.vgs;

Vin\_cm\_max = Mp1.vgs + Mp7.vov + spec.VDD;

Vout\_cm\_min = Mn6.vdsat;

Vout\_cm\_max = spec.VDD+Mp8.vdsat;

Pdiss = spec.VDD\*(Mp8.ids+Mp7.ids+Mp5.ids);

%

% %% Sanity check (do not modify)

%

% disp('======================================');

% disp('= Transistors in saturation =');

% disp('======================================');

if mosCheckSaturation(Mp1)

fprintf('\nMp1:Success\n')

end

if mosCheckSaturation(Mp2)

fprintf('Mp2:Success\n')

end

if mosCheckSaturation(Mn3)

fprintf('Mn3:Success\n')

end

if mosCheckSaturation(Mn4)

fprintf('Mn4:Success\n')

end

if mosCheckSaturation(Mp5)

fprintf('Mp5:Success\n')

end

if mosCheckSaturation(Mn6)

fprintf('Mn6:Success\n')

end

if mosCheckSaturation(Mp7)

fprintf('Mp7:Success\n')

end

if mosCheckSaturation(Mp8)

fprintf('Mp8:Success\n\n')

end

%% Summary of sizes and biasing points (do not modify)

disp('======================================');

disp('= Sizes and operating points =');

disp('======================================');

analog = cirElementsCheckOut(analog); % Update circuit file with

% transistor sizes

mosPrintSizesAndOpInfo(1,analog); % Print the sizes of the

% transistors in the circuit file

fprintf('IBIAS\t= %6.2fmA\nRc\t= %6.2f Ohm\nCm\t= %6.2fpF\n\n',Mp8.ids/1e-3,spec.Rm,spec.Cm/1e-12);

%% Performance summary (do not modify)

disp('======================================');

disp('= Performance =');

disp('======================================');

fprintf('\nmetrik \t result\n');

fprintf('Vin,cm,min [mV] \t%.0f\n',Vin\_cm\_min/1e-3);

fprintf('Vin,cm,max [mV] \t%.0f\n',Vin\_cm\_max/1e-3);

fprintf('Vout,cm,min [mV] \t%.0f\n',Vout\_cm\_min/1e-3);

fprintf('Vout,cm,max [mV] \t%.0f\n',Vout\_cm\_max/1e-3);

fprintf('Pdiss [mW] \t%.1f\n',Pdiss/1e-3);

fprintf('fGBW [MHz] \t%.1f\n',AvDC\*G2/C2/(2\*pi)\*1e-6);

%% Ploting transfer function (do not modify)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% if control toolbox in Matlab is available

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s = tf('s');

% exact transfer function

% TF1 = AvDC1\*AvDC2\*(1+s\*C1/(2\*G1))\*(1-s\*(1/z1))/ ...

% ((1 + s\*( spec.Cm\*(1+AvDC2)/G2 + (spec.Cm+spec.Cl)/G3 ) + s\*s\*((spec.Cm\*spec.Cl)/(G2\*G3)))\*(1+s\*C1/G1));

TF1 = AvDC1\*AvDC2\*((1+s\*C1/(2\*G1))\*(1-s\*(1/z1)))/ ...

((1+s\*C1/G1)\*(1+s\*C2/G2)\*(1+s\*C3/G3));

freq = logspace(1,12,1e3);

figure

bode(TF1,2\*pi\*freq); grid on;

h = gcr;

setoptions(h,'FreqUnits','Hz');

title('Frequency response Opamp');

hold all

Insert your LTspice netlist here:

vdd N001 0 1.1

vcm N006 0 0.32

E1 vinn N006 N006 vinp 1

V1 vinp N006 SINE(0 100u 1k) AC 10m

Mp8 N002 N002 N001 N001 p\_11\_sprvt l=1000n w=49.04u

Mp7 N003 N002 N001 N001 p\_11\_sprvt l=1000n w=48.01u

Mp1 N005 vinn N003 N003 p\_11\_sprvt l=200n w=59.35u

Mp2 vinter vinp N003 N003 p\_11\_sprvt l=200n w=59.35u

Mn4 vinter N005 0 0 n\_11\_sprvt l=1000n w=18.99u

Mn3 N005 N005 0 0 n\_11\_sprvt l=1000n w=18.99u

Mn6 vout vinter 0 0 n\_11\_sprvt l=200n w=110.70u

Mp5 vout N002 N001 N001 p\_11\_sprvt l=1000n w=289.39u

Rm vout N004 62.08

Cm N004 vinter 7.5p

Ibias N002 0 0.15mA

Cload vout 0 30p

.model NMOS NMOS

.model PMOS PMOS

.lib C:\Users\baciu\OneDrive\Documents\LTspiceXVII\lib\cmp\standard.mos

.include BSIM4\_UMC65.lib

;noise V(vout) V1 dec 20 1K 1MEG

.meas NOISE out\_totnVrms INTEG V(onoise)

.meas NOISE out\_mp1nVrms INTEG V(mp1)

.meas NOISE out\_mp2nVrms INTEG V(mp2)

.meas NOISE out\_mp7nVrms INTEG V(mp7)

.meas NOISE out\_mp8nVrms INTEG V(mp8)

.meas NOISE out\_mp5nVrms INTEG V(mp5)

.meas NOISE out\_mn6nVrms INTEG V(mn6)

.meas NOISE out\_mn4nVrms INTEG V(mn4)

.meas NOISE out\_mn3nVrms INTEG V(mn3)

.meas NOISE out\_rmnVrms INTEG V(rm)

\* RMS total noise calculated at the output

\* RMS noise contribution from each element seen at the output

.noise V(vout) V1 dec 20 1MEG 10G

.backanno

.end