Lab 3: Short Channel Amplifier Design

EE140/240A – Linear Integrated Circuits

Spring 2024

1. Objectives

In lab 2, you learned how to rely on hand-calculations to design an amplifier. This came along with successes and failures, but in the end, you realized that hand calculations give you a great starting place for your design, and you used Cadence to (possibly) sweep around these points. The key point is that our pre-Cadence work aided us in our work in Cadence.

Hand calculation is the preferred method with devices that are long channel. However, the shorter channel devices you have, the more you stray away from square-law models. If you've taken EE 130/230A, you might have seen some of these effects. Many second-order effects are not captured in devices in sub-micron devices, such as subthreshold conduction, drain-induced barrier lowering (DIBL), and velocity saturation. This lab, we will use short channel devices, and learn methods to come up with reasonable designs that do not rely on hand calculations.

The main question you might be asking now: if I can't use hand calculations, what do I use? Through your analysis of hand calculations, you found g_m to be quite a very important parameter. Higher g_m's relate to better gain, for example. To answer this question, we look at research. In 1995, Flandre et al. devised a powerful design approach for bypassing any device model by relying on pre-computed/simulated transistor-width-independent parameters. You can read the paper here g_m/I_D Methodology (you can access this for free with your Berkeley email). It's quite a short read, and understanding this will give you much better appreciation for the method we are going to learn in this lab.

Transistor-width-independent parameters are the key idea driving our design. Particularly, the size-independent parameter known as g_m/I_D is a unique parameter that can be used as a knob to adjust the inversion level and ultimately the speed or energy efficiency of MOS transistors. Think about it this way: high g_m/I_D values (around 20-30) means that lots of g_m is created with little I_D. You can imagine this is good for things like lowering power consumption while increasing gain. This parameter is known as the transconductance efficiency.

Another important parameter is I_D/W, which I'll refer to as normalized drain current (or current density). This allows us to translate our size-independent parameters to the sizes of our devices. There are many other important size-independent parameters, such as g_m/c_{gg}, and g_m/g_{ds}

Ok, maybe I now can tell you the answer to the question you are asking. The essence of all development is using a **look-up tables** for the devices we are using. We can generate look-up tables by probing our gpdk045 devices with all possible $V_{GS}, V_{DS}, V_{BS}, L$ combinations, and measuring the g_m/I_D and other parameters. Now we have lots of data, and our job as analog designers is, when we receive a list of specifications, to generate **software scripts** that return to us the parameters that achieve the specifications. We use these scripts to interact with a 4-dimensional look-up table (LUT), filter out combinations that do not satisfy some constraint, and iterate on a selected value, then denormalize our terms to get them into their size-dependent values.

2. Timeline

This is a 4 week lab, split into 5 total exercises, where the last exercise, we will create a differential amplifier, known as the 5-T workhorse amplifier. I will be releasing the exercises on a weekly basis, and you are highly, highly encouraged to finish them before the next lab. (Note: It will be too easy to fall behind in this lab. My recommendation is to attempt the problem before coming to lab, finish the coding/start with cadence during the lab, and work on your own to finish the exercise). The current tentative plan is listed below; it is subject to change.

- Week 1: Exercises 1 and 2. Introductory exercises to measure gain, unity gain bandwidth, g_m/I_D, V_{GS}
- Week 2: Exercises 3 and 4. Creating scripts that take in gain, bandwidth, etc., to return transistor sizes. We'll take a look at our amplifier in Lab 1 and make significant improvements to it via this methodology.
- Week 3: Exercise 5 and 6. Creating our differential amplifier (5T Workhorse amplifier) via redesigning our Lab 2 amplifier, then designing a tail current.
- Week 4: Buffer week. Writing the lab report, a time to catch up if you fell behind.

This lab is tentatively due on March 22nd at 11:59pm. This is planned to be due on the Friday before Spring Break, instead of a Tuesday like the previous two labs. Keep this in mind when we enter week 4, as the lab is due 3 days after.

3. Deliverables

In preparation for the final project, we will create more of a formal report, and the report itself will be much more opened ended than Lab 1 and 2. Your goal is to answer the questions given and prove them through Cadence. You should walk through your design process, as this is equally important to achieving simulation results, from mapping the constraints, to coding in Python/Matlab, to placing down a schematic in Cadence, to verifying the results in ADE Explorer.

I will be checking everyone's progress of the lab each week. This also serves as the attendance of the lab. All but Week 4 will have mandatory attendance. There will be a couple of questions I'll ask each week to make sure you are understanding what is happening, but I'll also have you ask me questions about different design related ideas with these exercises. These check-ins and check-offs will be part of your grade.

4. Grading

Lab 3 is worth 80 points. The rubric is loosely defined at the moment, unlike Lab 1 and 2, as the lab gets a lot more opened ended. Here are some criteria I will follow while grading.

- **Correctness of coding**. There will be correct solutions to these some of the exercises. I'll grade these on correctness. There are also open-ended questions. I'll review your code for each exercise that are open-ended. *It'll be easier to grade for me if you provide documentation with your code.*
- **Getting and interpreting meaningful results**. Some questions in the exercises will be proposed as questions, to which you will answer. Some will be requiring you to maximize certain parameters, and you will be graded based on effort in achieving the maximization with some benchmark to help you along.
- **Explanation of your design process**. How did you think about coding up the solution to the design problem? You will all have similar explanations for the design process for the introductory questions, but Exercise 3, 4, and 5 are much more open ended and you must be able to explain why you made certain choices.
- **Style of your lab report**. Attention to details like well-formatted plots, explanations at every major part of the design. Please type your report, and if you need to add in equations or hand-calculations, you can add them in as an image.
- **Check-ins and Check-offs**. These graded on completion. Attending lab each week should guarantee these points. I'll write in each exercise what is needed for check-ins/offs, and what I'll be looking for each week.

5. Background

5.1 Design Flows of Long and Short Channel Devices

The design flow of any circuit starts with a given set of specifications and constraints (e.g., gain, bandwidth and output load). The designer first needs to identify the design variables (e.g., g_m, W/L, etc.) and find their relationships with the given specifications and constraints (e.g., $f_u = g_m/2\pi C_L$). The design space can usually be tightened by reducing the number of variables, that is, by elegant choice of topology (simplest topology with the fewest number of variables considered first) or by describing variables in terms of other variables. The last step in the design is to search within the design space to find the best set of variables that meet the design specs.

In Lab 2, we intentionally used longer channel devices to se can leverage the square-law model in our hand calculations to describe the relationships between the variables. For example, an intrinsic gain stage (IGS) has these relationships:

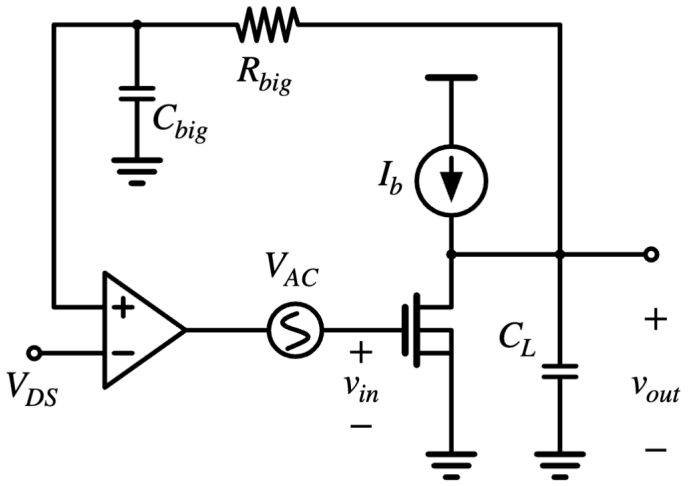


Figure 1: IGS

$$g_m = \sqrt{2\mu C_{ox}\frac{W}{L}I_D}$$
$$r_o = \frac{1}{\lambda I_D}$$
$$A_v = -g_m r_o = -\frac{1}{\lambda}\sqrt{\frac{2\mu C_{ox} W/L}{I_D}}$$
$$f_u = \frac{g_m}{2\pi C_L} = \frac{\sqrt{2\mu C_{ox} I_D W/L}}{2\pi C_L}$$

Therefore, I can give you an gain A_o, f_u, and C_L and you would simultaneously solve W/L and I_D. The design is essentially complete if our model was perfectly square law. Of course, in Lab 2, we made approximations in the square-law model, some of which included channel-length modulation, body-effect, parasitic capacitances, so our modeling wasn't entirely perfect. You might have hand-calculated all your values, but then when applied to our Level 3 MOSFET Model, the results did not meet the specifications.

The broken square-law model is replaced by pre-computed lookup tables. These lookup tables store all the information about our short-channel transistors (such as small-signal parameters g_m, c_{gs}, g_{ds}) across all possible bias points (various chosen V_{GS}, V_{DS}, V_{SB}) and for various device lengths. In short, any small-signal parameter of the device is stored as a function of four variables: V_{GS}, V_{DS}, V_{SB}, and L. These lookup tables are generated for each device only once and by using 4-dimensional parameteric dc simulations whose setup is shown below:

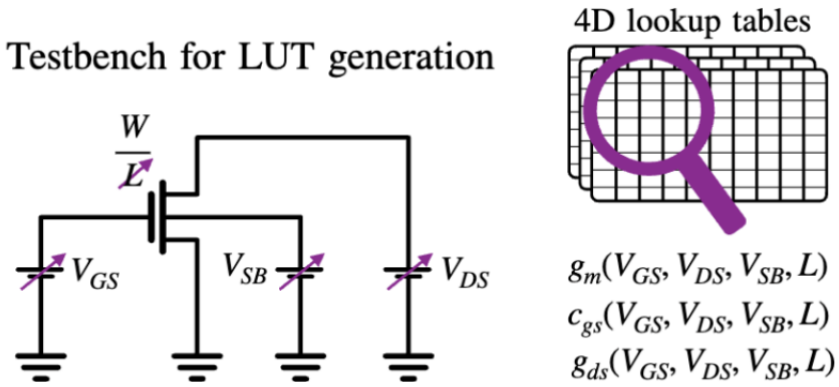


Figure 2: Testbench for LUT generation

As you have noticed, these look-up tables are generated using a single and arbitrary value for the width of the device W, say 1 μm. Then, you may wonder how we can use these LUTs as these LUTs store small-signal parameters of a device with say $W = 1$ μm. What if my transistor has a different W than the one these look-up tables were generated for? Well, the answer lies in the power of g_m/I_D design method that relies on W-independent variables such as $g_m/I_D, I_D/W, g_m/c_{gg}, g_m/g_{ds}$. It can be shown that these variables are (to the first order) independent of the width of the transistor (curious? Use the square-law model to demonstrate it). Therefore, it does not matter what W is used for the test device in generating the LUTs as long as you use the LUTs for accessing W-independent variables.

5.2 Developing a Systematic Design Methodology

EE140/240A has a library that attempts to systematizes this lookup procedure. Essentially, there is some main lookup function that allows you to lookup a size-independent parameter based on arrays of V_{GS}, V_{DS}, V_{SB}, and L. This library exists for Python and Matlab, and it is inspired by Boris Murmann's scripts written in Matlab. This allows us to follow this design flow.

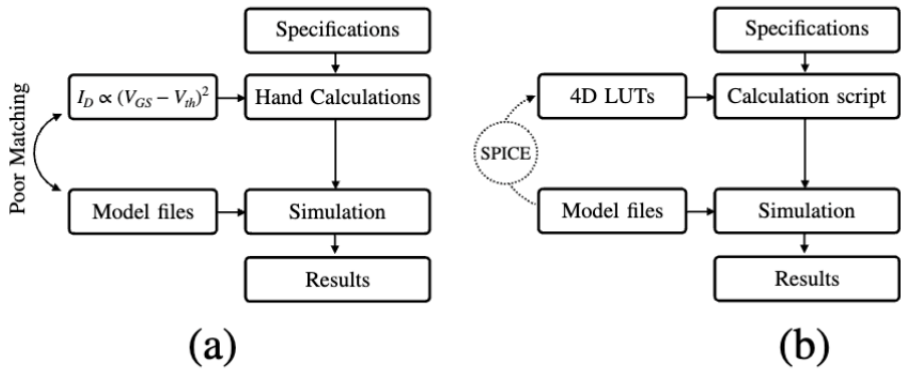


Figure 3: Design Flow

Week 1: Determining and Verifying Lookups

Exercise 1: Getting Used to Lookups

This exercise has a lot of information, but is meant for you to understand how the library works, and give you an understanding of how you can use it to answer design problems.

First, setup your workspace. You are provided the look-up tables for nmos and pmos devices using gpdk045, that is, nmos1v/pmos1v. These lookup tables can be accessed using Python or Matlab and you are free to run whichever you are more comfortable. The lab computers do not support python and can be slow when running Matlab. For that reason, we have provided the data at this link: http://tinyurl.com/ee140-lab3-data, so you may work with the data on your personal computer as well. If you choose to use Matlab on the lab computers, run these commands:

```
cd sim
mkdir gpdk045_LUTs
cd gpdk045_LUTs
cp -r /home/ff/ee140/fa22/lab2/LUTs/ .
export PATH=‘‘$PATH:/share/b/bin’’
matlab &
```

If you run these commands after sourcing your cadence setup, you need to add Matlab's path to your $PATH environment variable again by entering:

```
export PATH=‘‘$PATH:/share/b/bin’’
matlab &
```

If you choose to develop on your personal device using Matlab, place nmos_1v.mat, pmos_1v.mat, look_upVGS.m and look_up.m in the same folder as your scripts. If you choose to develop with Python, place nmos_1v.mat, pmos_1v.mat, and lookup.py in the same folder as your code.

Inside the LUTs directory you copied earlier, in addition to the LUTs, there are two important functions look_up and look_upVGS written by Boris Murmann. Open these function scripts and carefully read their description. Now create a new matlab/python script and load the look-up tables of the nmos1v and pmos1v devices:

```
% MATLAB
nch = importdata(‘nch_1v.mat’);
pch = importdata(‘pch_1v.mat’);
```

```
% Python
import numpy as np
import matplotlib.pyplot as plot
from look_up import *
nch = importdata(‘nch_1v.mat’)
pch = importdata(‘pch_1v.mat’)
```

NOTE: If you copy code from this PDF to Matlab, the single quote symbol might be incompatible with Matlab's editor, so you have to type the single quote symbols manually.

Explore the data stored in nch and pch. For example, you can find the length and width of the nmos1v test device used for generating these LUTs by entering nch.L or nch['L'] and nch.W or nch['W'], respectively for MATLAB or python. These dimensions are stored in micrometers. The sweep range for VGS and VDS are respectively obtained by nch.VGS and nch.VDS. Moreover, you can obtain ID-VGS curves of the test device (remember with W of nch.W) for various L values by the following commands:

```
% MATLAB
L = 0.1:0.2:1; % Lengths of interest
semilogy(nch.VGS, look_up(nch, 'ID', 'VGS', nch.VGS, 'L', L));
```

```
% Python
L = np.arange(0.1, 1, 0.2)
plt.semilogy(nch['VGS'], look_up_basic(nch, 'ID', vgs=nch['VGS'], l=L).T)
plt.show()
```

Notice we did not specify V_{DS} in the above plots, instead the script specified a default V_{DS}. What is the V_{DS} in the curves you just plotted.

> Can you estimate the threshold voltage for the test devices? The threshold voltage is the V_{GS} when the drain current exceeds 1 μA for a 1 μm width device.

Similarly, you can obtain I_D, V_{DS} of the test device for various V_{GS} values using the following commands

```
% MATLAB
VGS = 0.1:0.3:max(nch.VGS);
plot(nch.VDS, look_up(nch, 'ID', 'VGS', VGS, 'VDS', nch.VDS))
```

```
% Python
VGS = np.arange(0.1, max(nch['VGS']), 0.3)
plt.plot(nch['VDS'], look_up_basic(nch, 'ID', vgs=VGS, vds=nch['VDS']).T)
```

> What is the length of the device in the I_D, V_{DS} curves you just plotted? The I_D, V_{GS} and I_D, V_{DS} curves of test nmos1v are shown in the figure below for reference.

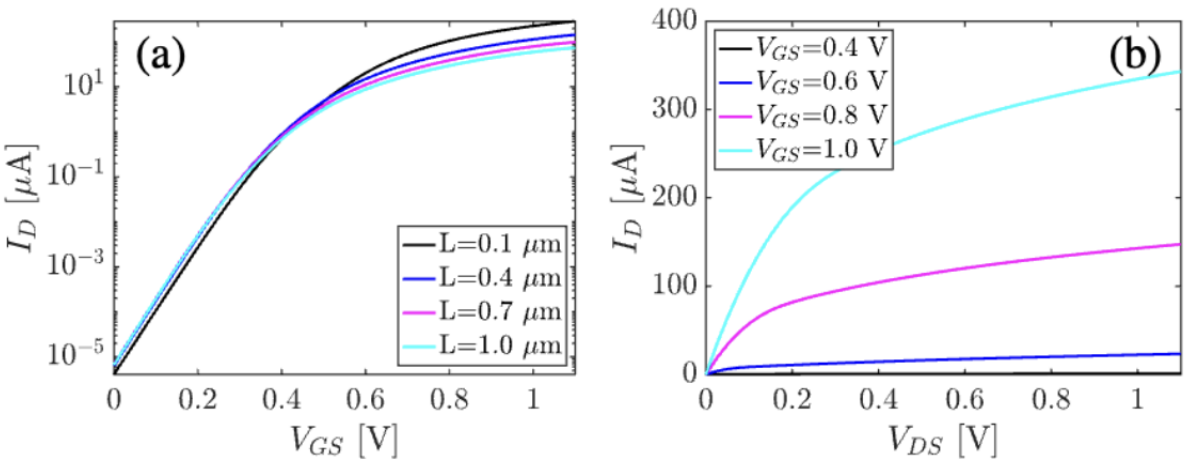


Figure 4: (a) $I_D - V_{GS}$ (b) $I_D - V_{DS}$

As mentioned earlier, almost never do we use LUTs to obtain absolute values of the small signal parameters. Instead, we obtain the ratios of small-signal parameters that are width-independent. You will see later on, how these width- independent ratios are effectively used during the design phase as variables. For now, let's plot three of these width-independent variables to uncover an inherent trade-off in MOS devices.

The transit frequency f_T of a transistor is the frequency at which the magnitude of the current gain of the device falls to unity. This current gain is calculated when the drain of the transistor is at AC ground and the input has a small signal current input iin. Draw the small signal model of an NMOS and prove that

$$f_T = \frac{g_m}{2\pi c_{gg}}$$

$$c_{gg} = c_{gs} + c_{gd} + c_{gb}$$

The classic small signal model of the transistor you used to derive falls short of correctly modeling the transistor's second order effects at frequencies greater than roughly $f_T/10$. Therefore, we limit the operating frequency range of our circuits to be $\leq f_T/10$. Keep in mind that f_T is a measure of a device's speed. That is, devices with a larger f_T allow us to operate at higher frequencies. Both g_m and c_{gg} scale linearly with W, and therefore f_T is a width-independent variable. So are the intrinsic gain $A_v = g_m/g_{ds}$, transconductance efficiency g_m/I_D, and the current density $J_D = I_D/W$ of the MOS devices. In the g_m/I_D design methodology, as the name suggests, g_m/I_D is used as a knob to set the speed or the intrinsic gain of the devices. Let's have a demonstration. Plot f_T and $A_v = g_m/g_{ds}$ of an intrinsic gain stage as a function of g_m/I_D by running the following commands:

```
% MATLAB
L = [0.05 0.2 0.5 1];
gm_ID = 5:0.1:30
ft = look_up(nch, 'GM_CGG', 'GM_ID', gm_ID, 'L', L)/2/pi;
Av = look_up(nch, 'GM_GDS', 'GM_ID', gm_ID, 'L', L0);
plot(gm_ID, ft, '-')
hold on
yyaxis right
plot(gm_ID, AV, ':')
```

```
% Python
L = np.array([0.05, 0.2, 0.5, 1])
gm_ID = np.arange(4, 30, 0.1)
ft = look_up_vs_gm_id(nch, 'GM_CGG', gm_ID, l=L)/2/np.pi
Av = look_up_vs_gm_id(nch, 'GM_GDS', gm_ID, l=L)
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.semilogy(gm_ID, ft.T, '-')
ax2.plot(gm_ID, Av.T, ':')
plt.show()
```

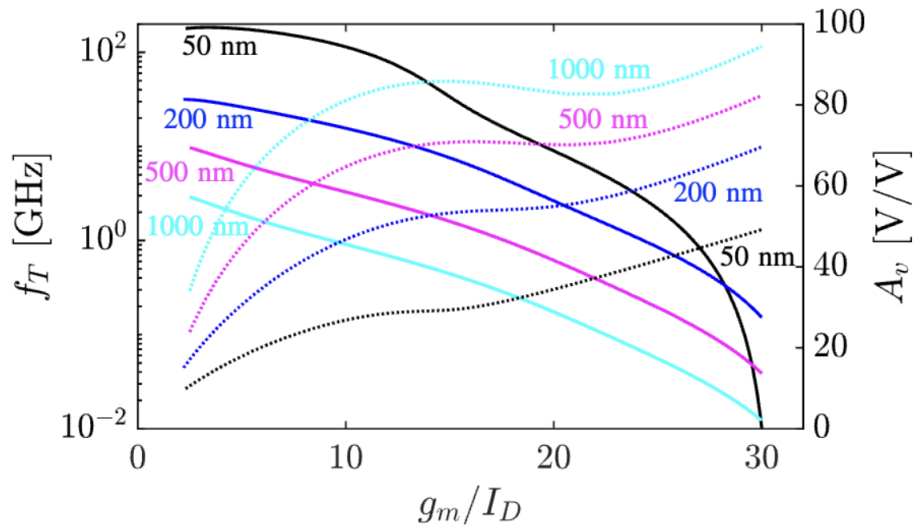


Figure 5: g_m/I_D used as a knob for adjusting the speed and the intrinsic gain of a device. Solid and dotted lines are respectively f_T and A_v

Four main observations can be made in regards to Figure 5:

1. Av is proportional to the length of the device. This is expected as increasing the length decreases the channel length modulation and ultimately g_{ds}.
2. f_T is inversely proportional to the length of the device. Again, this is an expected result because reducing the length simultaneously decreases c_{gg} and increases g_m.
3. A_v roughly monotonically increases with g_m/I_D, while at the same time
4. f_T degrades with increasing g_m/I_D.

Therefore, g_m/I_D and L can be used as proxies to adjust the gain and speed, but remember that either the speed or the gain of the device can be maximized, not both by these proxies.

Now we come up with a systematic design methodology:

1. Determine the design specifications.
2. Determine the constraints for the design specifications.
3. Determine g_m.
4. Pick L. Short-channel means high f_T and a small area. A longer channel will give more gain A_v and a larger area.
5. Pick g_m/I_D. A large value will give low power and high gain, a small value will give high f_T and small area.
6. Determine I_D from step 3 and 5.
7. Determine W using I_D/W denormalization.

Consider the intrinsic gain stage. Calculate the intrinsic gain, f_T, W and required V_{GS} of the nmos1v device with the following assumptions: $g_m/I_D = 15\ S/A$, $L = 100\ nm$, $C_L = 1\ pF$, $f_u = 1\ GHz$, and $V_{DS} = 0.55\ V$.

The feedback network shown in gray is used to properly set the gate-source voltage of the nmos such that the output DC value is at $V_{DS} = 0.55\ V$. The V_{DD} of the circuit is $1.1\ V$. For AC signals, you can ignore the feedback loop. You can use an ideal voltage-controlled voltage source, vcvs, from the analogLib library to implement the DC settling feedback amplifier. To avoid simulation convergence issues you can set the Maximum Output Voltage and the Minimum Output Voltage of the vcvs source to $1.1\ V$ and $0\ V$, respectively. The Voltage gain of the vcvs should be ≥ 1000.

Here are all the tips that can help you figure out this problem. These are in the order you should think about them in.

- $g_m/I_D, L, V_{DS}, V_{SB} = 0$ are known. Write them all down in code.
- Since these values are known, you can directly use the look_up function to obtain f_T and the intrinsic gain of the device. Don't forget the division by 2π.
- The intrinsic gain is $g_m r_o$. Recall that $r_o = \frac{1}{g_{ds}}$. How can you use a look_up to get the gain knowing this?
- g_m can be found from the choice of f_u and C_L.
- You'll need to perform a lookup to get current density $J_D = I_D/W$. Then you can get the width knowing that $W = I_D/J_D$. How can you get I_D from a term you did a lookup on that includes I_D?
- Finally, V_{GS} can be performed using a lookup with the known values.

Now, simulate the performance of your IGS with the parameter values calculated above. You should run both DC and AC simulations to confirm the bias points of the amplifier as well as its frequency response. Save a copy of the bode plot of the transfer function of your amplifier and create a table listing its designed and simulated gain, unity gain bandwidth, g_m/I_D, and V_{GS}.

Exercise 2: Another IGS Example

We will now try another exercise with less guidance. The previous exercise was fully constrained with known g_m/I_D and L. There was no design involved, and we merely analyzed the given IGS. In this exercise, we relax the constraints by setting L as a design variable.

Consider the same intrinsic gain stage. Assuming a constant $g_m/I_D = 5V^{-1}$, sweep over nch.L, find the achievable f_T and A_v by the IGS as well as the corresponding W for the nmos1v device. As in Exercise 1: $C_L = 1\ pF, V_{DD} = 1.1\ V$ and $V_{DS} = 0.55\ V$, but $f_u = 500\ MHz$.

Plot $\log(f_T)$ vs. L and A_v vs. L. Find the maximum value of L that satisfies $f_u \leq f_T/10$. As you can see, there are many solutions $\{L, W, I_D\}$ that satisfy $f_u = 500\ MHz$ constraint (with different intrinsic gains, of course). Choose two of these solutions and confirm with simulations both of them obtain $f_u = 500\ MHz$. Save the simulated AC response of your designs and compare their performance in a table.

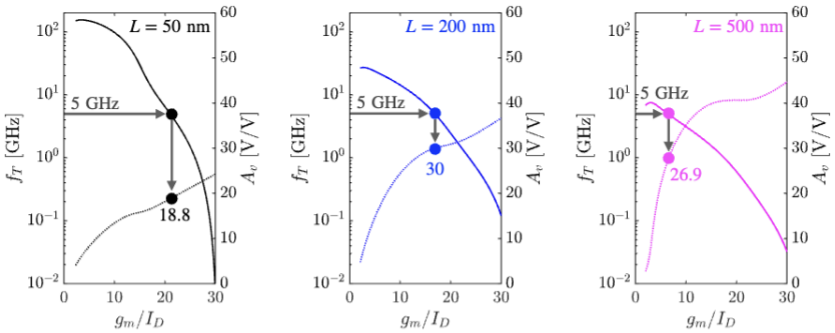
Week 2: Redesign

This week, you'll create scripts to redesign your Lab 1 amplifier, and along the way, you'll learn how parasitic capacitor c_{dd} is incorporated in design.

Exercise 3: Maximum Gain Design

You've seen from Week 1 that there is denormalization phase to find the width. In addition, there is also a phase to include total drain capacitance. In previous exercises, we ignored c_{dd} (total drain capacitance) of the nmos while calculating g_m. In high-frequency applications, ignoring the total drain capacitance, however, decreases the design accuracy, and therefore should be included during the de-normalization phase (finding the widths/currents of the devices). Because transistor width determines c_{dd}, and c_{dd} adds to C_L which determines the gm and ultimately width, c_{dd} is best estimated in an iterative fashion. Initially W is calculated with c_{dd} assumed to be zero. Then, using the calculated W, c_{dd} is estimated and used to update the value of W. This process continues for a few cycles such that c_{dd} and W converge.

The previous exercise demonstrated that for a fixed g_m/I_D, f_u and C_L, the only way to increase the intrinsic gain of the stage is to increase the length of the device. But, because of the $f_T \geq 10 \times f_u$ constraint, L (and subsequently A_v) can only be increased up to a certain limit. An attentive designer might think that there is a possibility to increase the gain beyond that obtained in the previous exercise by increasing the g_m/I_D of the transistor. This is indeed true. For small values of g_m/I_D, the intrinsic gain of the device is at its minimum. Slightly increasing the gm/ID (to mid values of say about 15) significantly increases the intrinsic gain. Increasing the g_m/I_D further pushes the device into the subthreshold region reducing its speed (f_T). But, because f_T of the device must be at least $10 \times f_u (= 5$ [GHz] in the previous exercise), maintaining this minimum required f_T requires reducing its length to small values (of about 50 nm), which ultimately results in a heavy penalty in the intrinsic gain of the device. Therefore, there indeed seems to be an optimal value for gm/ID and the length of the device to obtain the maximum possible gain for the IGS.



Interpret the meaning of this figure above, and how this will be useful in your design.

Exercise 3 is a short exercise meant for you to see how to optimize gain. The most of the code is provided for you. Just fill in the blanks as needed.

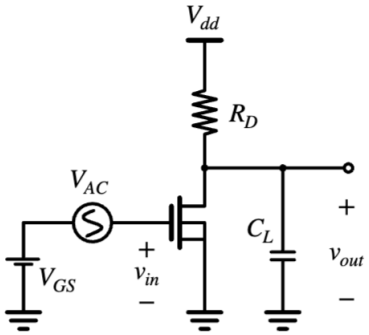
```
% MATLAB
L = nch.L
fu = 5e8
CL = 1e-12
vds = 0.55
gm_ID_range = linspace(5, 30, 50);
gm_ID = []
for i = 1:length(L)
  % TODO: obtain fT
  M = fT >= 10*fu
  if(any(M))
    gm_ID(i) = gm_ID_range(max(find(M==1)));
  else
    gm_ID(i) = NaN;
end
plot(L, Av, L, gm_ID)

gm_ID_opt = X; % TODO: replace X based on results obtained above
L_opt = Y; % TODO: replace Y based on results obtained above

% TODO: Lookup to find JD

% Cdd_W = look_up(nch, 'CDD_W', 'GM_ID', gm_ID_opt, 'L', L_opt);
Cdd(1) = 0;
for m = 1:10
  gm_opt = 2*pi*fu*(CL + Cdd(m));
  % TODO: Find ID(m)
  W(m) = ID(m)./JD;
  Cdd(m+1) = W(m)*Cdd_W;
end
```

```python
# Python
L = nch['L']
fu = 5e8
CL = 1e-12
vds = 0.55
gm_id_range = np.linspace(5, 30, 50)
gm_id = []
av = []
for i, l in enumerate(L):
  # TODO: Find ft
  m = ft >= 10*fu
  if any(m):
    gm_id.append(gm_id_range[max(np.where(m==1)[0])])
  else:
    gm_id.append(float('nan'))

  # TODO: Obtain av[i] = gm/gds

fig, ax = plt.subplots(1)
ax.plot(L, av)
ax.plot(L, gm_id)
plt.show()

gm_id_opt = X # TODO: replace x based on results above
l_opt = Y # TODO: replace y based on results above

# TODO: Lookup to find JD

cdd_w = look_up_vs_gm_id(nch, 'CDD_W', gm_id_opt, l=l_opt)
cdd = 0
for i in range(1, 10):
  gm_opt = 2*np.pi*fu*(CL + cdd)
  # TODO: Find ID
  W = ID / JD
  cdd = W * cdd_w
```

Simulate the AC response of your optimal design and compare its performance against the design in the previous exercise.

Exercise 4: Lab 1 Redesign

In Lab 1, you used the gpdk045 library to build a common-source amplifier with all parameters given directly to you. However, this amplifier is not necessarily the most optimal. We'll first alter the schematic to be as follows:



Notice that the alteration includes a load capacitance of $C_L = 1\ pF$.

Simulate the AC response of the same amplifier ($R_D = 7.8\ k\Omega$, $W = 2\ \mu m$, $L = 100\ nm$, $V_{GS} = 630\ mV$, $V_{DD} = 1.1\ V$, 1 finger) with a load capacitance $C_L = 1\ pF$ and find the gain and unity-gain bandwidth f_u of that amplifier. For this part use an ideal resistor res and capacitor cap from analogLib. Determine by looking at the bode plot the gain A_v and unity gain frequency f_u.

Now, create a generalized script, `lab1(fu)`, that takes in a unity gain frequency f_u in Hz and returns (either by returning the value or printing the information out) the appropriate W, L, V_{GS}, R_D that gives the best gain. In addition, report the scripted g_m/I_D associated with the largest gain at this unity gain frequency.

Now create a table that compares the simulated results for the following frequency values (as an input to the script).

- 1 MHz
- 50 MHz
- 100 MHz
- 500 MHz
- 1 GHz

In your table, be sure to show the V_{GS}, R_D, W, L, I, and g_m/I_D of each result in Cadence.

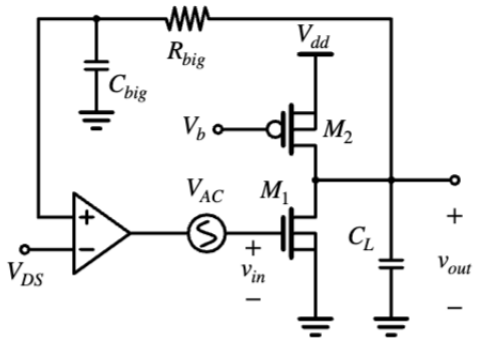
Week 3: Differential Amplifiers

This week, you'll be designing a differential amplifier. We will do this by first revisiting the lab 2 amplifier, then converting it to a differential amplifier.

Exercise 5: Lab 2 Redesign

We will not be using the Level 3 SPICE model as we did in Lab 2. Instead, we will be using the gpdk045 transistors we've been using so far in this lab. However, if you generated lookup tables for those transistor models and wrote a generalized script that can take in any transistor, you would be able to significantly improve your Lab 2 design. Since we are sticking with gpdk045, you cannot really make any meaningful comparisons to your Lab 2 design.

Design the common-source amplifier shown in the figure below to obtain the maximum possible gain while achieving $f_T = 10f_u = 1\ GHz$. **Bias the PMOS using a current mirror**. Assume $C_L = 1\ pF$, $V_{DD} = 1.1\ V$. Find W, L, I_D, V_{GS} of both transistors. Report the gain and unity gain frequency of your amplifier. Simulate the performance of your amplifier and compare it against your calculations.



Here are some tips to help you get started:

- Determine the gain expression, A_v, using normalized variables (width-independent) so we can use lookup tables later on.
- Note that you want to maximize A_v while maintaining:

$$f_{T,n} = \left(\frac{g_m, n}{2\pi c_{gg,n}}\right) = 1\ GHz$$
$$f_{T,p} = \left(\frac{g_m, p}{2\pi c_{gg,p}}\right) = 1\ GHz$$

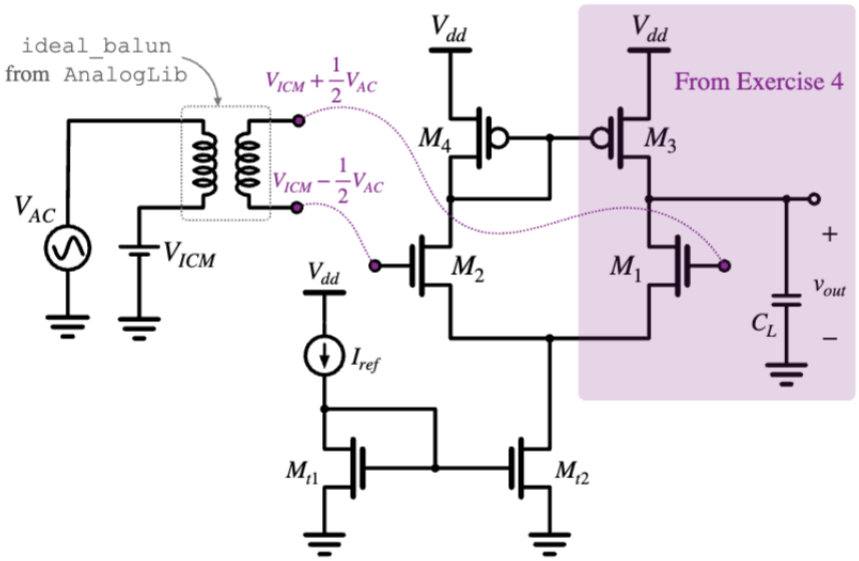
- There is no direct information/constraint about the length of the devices, and as you have seen so far, it is completely fine to use length as a design variable.
- Since the PMOS will be biased using a current mirror, to get optimal current matching, we want $V_{DS,p} = V_{GS,p}$. This way, the V_{DS} of the PMOS will match the V_{DS} of the diode connected PMOS.

Here are some more notes regarding the code:

- We want to find the optimal V_{out}, since knowing this allows us to find $V_{GS,p}$ and $V_{DS,p}$. These will both be $V_{DD} - V_{out}$. Therefore, start by looping over a reasonable sweep of V_{out}.
- For each V_{out}, start with lookups with the PMOS. Specifically, you want to find information that helps you (1) find parts of the gain expression and (2) use the frequency constraints to get rid of certain lengths. You know the vgs and vds parameters of these lookups, so ensure you use those!
- How does L relate to g_{ds}/I_D? You know you want the highest g_{ds}/I_D, so choose the best L for the PMOS that gives you the highest $g_{ds,p}/I_D$.
- Ensure you know your PMOS g_{ds}/I_D and your PMOS length L for this particular iteration of V_{out} before analyzing the NMOS. You'll perform similar lookups like you did with the PMOS. Looping through the L values, determine the gain that also satisfies the $f_{T,n}$ constraint.
- After looping through the V_{out} values, you know can find the specific optimal values of g_m/I_D and L of the NMOS, as well as the optimal L of the PMOS.
- Finish the script by looking up the I_D/W and C_{DD}/W of both transistors and perform iteration to better capture the drain capacitance behavior. This allows you to (1) find the g_m, (2) find the current, and (3) the widths of the devices.

Exercise 6: 5T Workhorse Transistor

In practice, we almost always use differential amplifiers for rejection of common-mode noise/interference. It is fairly straightforward to convert the single-ended amplifier designed in exercise 5 to the classic differential amplifier as shown bwlow Because with differential input signals, the source of the input pair devices is a virtual ground, the small signal model of the classic differential amplifier is the same as that in Exercise 5. Therefore, the design in exercise 5 can be reused for the differential implementation of the amplifier. There are some differences between the two circuits that requires some modification to the design script from 5. You may use g_m/I_D design scripts to design the tail current mirror or design it yourself. Simulate the AC response of the amplifier shown below and compare its performance to that designed in Exercise 5. For simulation of differential amplifiers, `ideal_balun` can effectively be used to generate a pair of differential signals with a proper common-mode voltage (V_{ICM}, input common-mode voltage) out of a single AC voltage source as shown below.



References

[1] Enz, Christian C., François Krummenacher, and Eric A. Vittoz. "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications." Analog integrated circuits and signal processing 8, no. 1 (1995): 83–114.

[2] Silveira, Flandre, Denis Flandre, and Paul GA Jespers. "A g_m/I_D based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA." *IEEE Journal of Solid-State Circuits* 31, no. 9 (1996): 1314–1319.

[3] online: `https://web.stanford.edu/~murmann/gmid`

[4] Jespers, Paul GA, and Boris Murmann. Systematic Design of Analog CMOS Circuits. Cambridge University Press, 2017.