SPICE 5: MOSFETs

Chris Winstead

ECE 3410. Spring, 2015.

Preparing for the Exercises

In this session, we will simulate several MOSFET configurations. Create a new directory for this session:

```
3410/
└─ spice/
   ├─ lab1/
   ├─ lab2/
   ├─ lab3/
   ├─ lab4/
   └─ lab5/
```

Obtain the Example Files

In this lab, you will download a collection of files containing examples and exercises. Download these files and place them in these locations:

```
3410/
└─spice/
   ├─[files from previous labs]
   └─lab5/
      ├─inverter_testbench.sp
      └─csamp_testbench.sp
```

MOSFET Models in SPICE

Transistors are more complex than diodes and other devices. They require detailed device models consisting of many parameters. The basic form for a model declaration is:

```
.model <name> <device_type>(<parameter1>=<value1>, <parameter2>=<value2>,...)
```

Below are minimum device models needed to simulate MOSFETs:

```
* Idealized MOSFET models:
.model ntype NMOS(KP=100e-6,VTo=2,LAMBDA=0.005)
.model ptype PMOS(KP=100e-6,VTo=-2,LAMBDA=0.005)
```

In this model declaration, the parameters are

$$\begin{aligned} \mathrm{KP} &= \mu C_{\mathrm{ox}} \\ \mathrm{VTo} &= V_{\mathrm{Th}} \\ \mathrm{LAMBDA} &= \lambda \end{aligned}$$

MOSFET Instances in SPICE

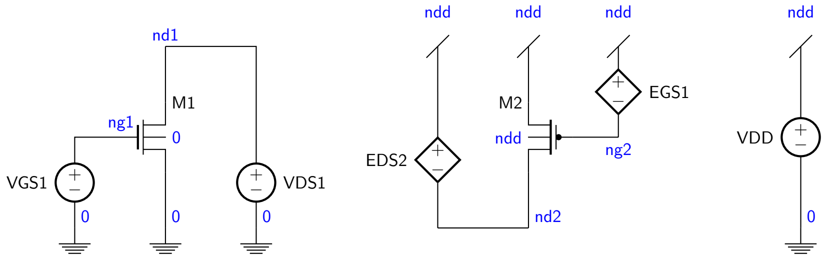
To use a MOSFET, you must declare four device nodes, the model name and several model parameters. Here is a minimal MOSFET declaration:

```
M1 ndrain ngate nsource nsubstrate ntype W=1u L=1u
```

In the declaration, `ndrain` refers to the MOSFET's drain node, and so on. The model name is `ntype` and the gate dimensions are 1 µm × 1 µm.

MOS I/V Characteristics

Your first exercise is to simulate the I/V characteristics of ideal NMOS and PMOS devices using the models shown above. To do this, you will use the simple circuit shown below.



Let's construct the SPICE code for simulating MOSFET I/V curves. Create a file called exercise1.sp for this simulation. The lines below provide the title, description and meta-data (exercise name, author name, etc), declare the MOS models and the independent voltage sources:

```
* MOSFET I/V sweeps
* SPICE 5, exercise 1
******************************
* By Chris Winstead
* ECE 3410, Utah State Univ.
******************************

* Idealized MOSFET models:
.model ntype NMOS(KP=100e-6,VTo=2,LAMBDA=0.005)
.model ptype PMOS(KP=100e-6,VTo=-2,LAMBDA=0.005)

* Gate voltage source and VDD:
VDD  ndd 0   DC 5V
VGS1 ng1 0   DC 2.5V
VDS1 nd1 0   DC 2.5V
```

The specific values of `VGS1` and `VDS1` don't matter since we will be sweeping them.

Next we declare the dependent voltage sources used to simulate the PMOS device. In SPICE, a dependent source is declared using this syntax:

```
E<name> <n+> <n-> <control+> <control-> <k>
```

This syntax declares a voltage source between `n+` and `n-` equal to `k` times the potential between `control+` and `control-`. In our case, we want sources that replicate `VDS1` and `VGS1`, which can be declared using these lines:

```
EDS2 ndd ns2 nd1 0 1
EGS2 ndd ng2 ng1 0 1
```

Next we declare the MOSFET connections. Fill in the appropriate node connections for the N-type and P-type devices declared below:

```
M1 <drain> <gate> <source> <substrate> ntype W=1u L=1u
M2 <drain> <gate> <source> <substrate> ptype W=1u L=1u
```

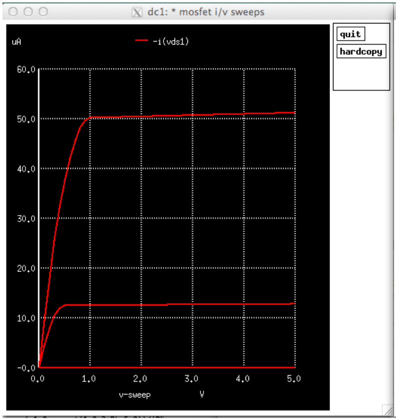
It is usually necessary to correctly specify the devices' `W` and `L` parameters. In this case we will start with simple 1 µm declarations.

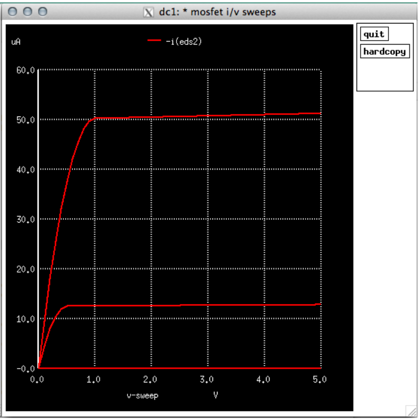
Lastly, complete the SPICE description with simulation and plotting commands. In this exercise you will perform a two-dimensional sweep over VGS and VDS:

```
* Control commands to simulate the
* DC transfer characteristic:
.control
DC VDS1 0 5 0.1 VGS1 0 3 0.5

* Plot NMOS I/V curves:
plot -i(VDS1)

* Plot PMOS I/V curves:
plot -i(EDS2)

.endc
.end
```

Note that the current through a voltage source is conventionally negative, so we plot `-i(VDS1)` and `-i(EDS2)` to see the devices' drain current magnitudes.

If your simulation is correct, you should see two curve traces that look like this:





Save a hardcopy of one of these plots with the name `IVsweep.ps`.

Observations on the I/V Curves

Create a `log.txt` file and record quantitative observations about your simulation. To obtain a quantitative measure of the effect from varying λ, clicking on the plots to manually measure the slope of the highest curve in the saturation region for each case. You should find that the slope is equal to the device's output resistance, r_o:

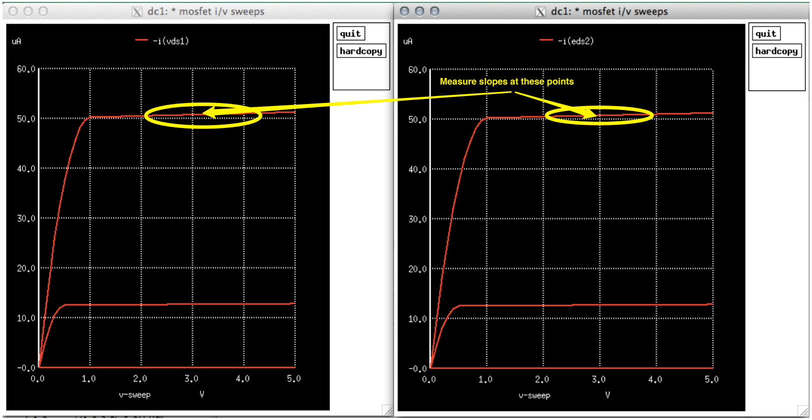
$$\begin{aligned}\frac{\Delta v_{\mathrm{DS}}}{\Delta i_D} &= r_o \\ &= \frac{1}{\lambda I_D}\end{aligned}$$

You can measure I_D as the value of the drain current just after the device enters saturation. Verify that the measured slope is approximately equal to the output resistance predicted by this equation.

In `log.txt`, record your measurements for I_D and the slope, as well as your calculation for r_o. You only need to take these measurements for the highest curve.

Location of Measurements

To be clear, the saturation region on these curves is where i_D "flattens out" as a function of v_{DS}. The value of r_o should be measured in the saturation region as shown:



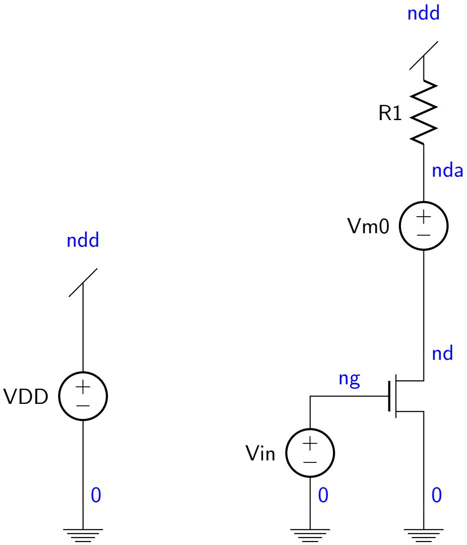
Varying Device Dimensions and λ

Now repeat your I/V simulations with the modifications listed below. In `log.txt`, record the differences observed in these cases:

1. $W = 10\,\mu\text{m}$ and $L = 1\,\mu\text{m}$ and $\lambda = 0.005\,\text{V}^{-1}$ for both devices.
2. $W = 1\,\mu\text{m}$ and $L = 1\,\mu\text{m}$ and $\lambda = 0.1\,\text{V}^{-1}$ for both devices.

Simulating CLM in an NMOS RTL Inverter

Next we will study the inverter circuit below:



Create a SPICE file called `exercise2.sp`. Include the header and MOSFET model declarations from `exercise0.sp`, and construct the RTL inverter circuit exactly as shown, with the node names as indicated.

Use these parameter values: R1=100 kΩ, VDD=5 V(DC), Vm0=0 V(DC), and also connect the MOSFET's substrate to ground (node 0).

We will simulate the circuit for three separate values of λ. The control simulations are provided for you in the file called `inverter_testbench.sp`. Copy all lines from the testbench file and paste them into the bottom of your `exercise2.sp` file.

At the top of your file, add these lines to declare some simulation parameters:

```
.csparam dcstart=2
.csparam dcstop=2.47
```

Next we'll go through the different parts of the testbench.

First, the `save` command is used to request that SPICE store the internally computed transconductance parameter g_m:

```
* Control commands to simulate the
* DC transfer characteristic:
.control
save all @M1[gm]
```

SPICE computes many small-signal parameters for MOSFET devices, and it's often useful to access them directly in the simulation so they can be compared with hand analysis.

Next, the `altermod` command is used to change λ to $0.01\,\text{V}^{-1}$, and a DC sweep is performed to measure the response over the region specified by parameters `dcstart` and `dcstop`. We use parameters to specify the simulation range because the simulation will be repeated for three different cases, and we don't want to have to change all three places if the range needs to be adjusted.

```
* Simulate with LAMBDA=0.01
altermod @ntype[lambda] = 0.01
DC Vin $&dcstart $&dcstop 0.01
```

After the DC simulation is done, we measure the characteristics at some point. Here we chose 2.4 V:

```
meas DC IDS1 FIND i(Vm1) AT=2.4V
```

The circuit's gain is the derivative $dv_{\mathrm{OUT}}/dv_{\mathrm{IN}}$, which can be measured using these commands:

```
let G1=deriv(nd)
meas DC gain1 FIND G1 at=2.4v
```

The `deriv` command computes the derivative of the specified output node (`nd`) with respect to the sweep variable `Vin`.

By analysis, we know that the gain should be equal to

$$-g_m R_{\text{OUT}} = -g_m \left(r_o \parallel R\right).$$

We also know that

$$r_o = \frac{1}{\lambda I_D},$$

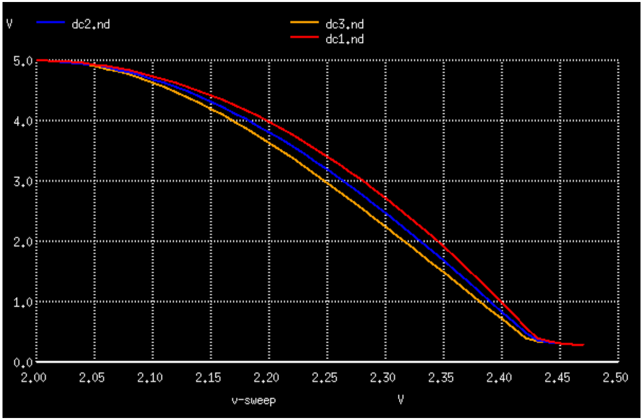
which should allow us to predict the gain using these lines:

```
meas DC gm1 FIND @M1[gm] at=2.4v
let rds1=1/(0.01*ids1)
print rds1
```

This measurement block is repeated three times, for three different values of λ.

Simulation Results

When your SPICE file is correct, perform the simulation. You should see a figure like the one below.



Save a hardcopy of this plot with the name `inverter.ps`.

The plot on the previous slide is produced by accessing the results from the three different DC simulations, which have the names “dc1”, “dc2”, and so on. Results from different simulation are accessed using the syntax “`<simname>.<nodename>`”, like this:

```
plot dc1.nd dc2.nd dc3.nd
```

Measurements and Calculations

To complete this exercise, calculate the expected gain by computing

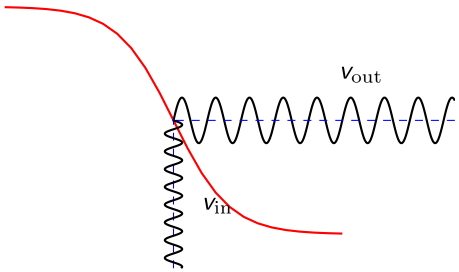
$$-g_m \left(r_o \parallel R \right),$$

and compare these results to the gain measured by SPICE in each of the three cases. How closely do they match? Describe the relationship between λ and the gain for this circuit. Record these responses in your `log.txt` file.

Optional: instead of hand-calculating the expected gain, you can have SPICE do it by inserting lines to report the calculation within your simulation.

Common-Source Amplifier

Next, without changing the circuit, we're going to simulate the operation of a common-source (CS) amplifier. A CS amp is basically the same as an inverter, except CS amplifiers are designed to operate with analog signals in the saturation region, whereas digital inverters are designed to operate only at the high/low extremes. The idea is to balance the input signal in the middle of the transition range, like this:



To simulate a CS amplifier, copy your inverter circuit **without the testbench** into a new file called `exercise3.sp`. Remove the dcstart and dcend parameters, and add these parameters to the top of the file:

```
.param VINoffset=2.4V
.param VINamplitude=10mV
```

Next, change the input voltage to a 1 kHz sinusoidal source:

```
Vin ng  0 DC VINoffset SIN('VINoffset' 'VINamplitude' 1kHz)
```

CS Amp Testbench

Next download the file called `csamp_testbench.sp` and copy the contents into the bottom of your `exercise3.sp` file. The testbench is similar to the previous one, except we will be doing transient simulations this time.

First, the testbench uses the `altermod` command to simulate three different values of λ. In each case, the output amplitude is measured:

```
* Control commands to simulate the
* DC transfer characteristic:
.control
save all @M1[gm]

* Simulate with LAMBDA=0.01
altermod @ntype[lambda] = 0.01
tran 10us 5ms
meas tran vopp1 PP v(nd)
```

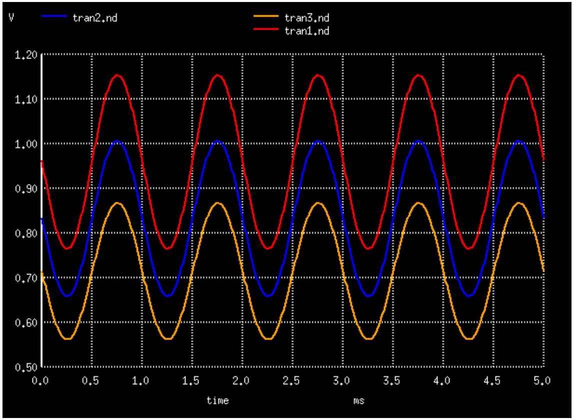
After the three cases are simulated, the gains are calculated as the ratio of peak-to-peak amplitudes at the output vs the input:

```
echo INPUT Peak-to-Peak amplitude:
meas tran vipp PP v(ng)

echo OUTPUT Peak-to-Peak amplitudes:
print tran1.vopp1 tran2.vopp2 tran3.vopp3

echo GAIN magnitudes:
print tran1.vopp1/vipp tran2.vopp2/vipp tran3.vopp3/vipp
```

Once again, the three simulations are accessed as `tran1.<name>`, `tran2.<name>` etc.

Simulation Result

Once your simulation is correctly executed, you should see a result like this:



The different λ values slightly change the output DC offset as well as the gain.
Save a hardcopy of this plot with the name `csamp.ps`.

Measurements to Record

In the SPICE console, you should see the gain values reported. Copy those values into your `log.txt` file. Compare these gain measurements to the ones you obtained in the previous exercise. Report this comparison in your `log.txt` file.

What to Turn In

Create a zip file containing the following files:

- `exercise1.sp`
- `exercise2.sp`
- `exercise3.sp`
- `IVsweep.ps`
- `inverter.ps`
- `csamp.ps`
- `log.txt`

Upload these files to Canvas to complete the assignment.