Lab 2: Transistor superthreshold saturation current and drain characteristics

Group number: 4.5

Team member 1: Jan Hohenheim

Team member 2: Maxim Gärtner

Date: 12.10.2021

The objective of this lab is to understand *super-threshold* (also called *above-threshold* or *strong inversion*) transistor operation and to understand transistor drain conductance characteristics, particularly *channel length modulation*.

The specific experimental **objectives of this lab** are as follows:

1. To characterize drain current of a transistor as a function of gate voltage in superthreshold operation in the ohmic (triode) and saturation regions.
2. To characterize the drain saturation properties in super-threshold.
3. To characterize drain conductance (the Early effect) and how it scales with transistor length (may not be possible this year) and saturation drain current.

An intuitive and quantitative understanding of all these effects, along with the subthreshold behavior (next week), is useful for the design of effective circuits, especially analog design of high performance amplifiers.

1 Terminology

- above-threshold = super-threshold = strong inversion
- sub-threshold = below-threshold = weak inversion
- triode region = ohmic region = linear drain conductance behavior with small drain-source voltage
- saturation = large V_{ds}
- overdrive = $V_{\mathrm{g}} - V_{\mathrm{T}}$
- $U_{\mathrm{T}} = kT/q$ = thermal voltage = 25mV at room temperature
- V_{T} = threshold voltage = 0.4V to 0.8V depending on process

2 Useful Quantities

The following is a list of the physical parameters and constants we will be referring to in this lab, along with their values when appropriate. The units that are most natural for these quantities are also included; these units are not self--consistent, so make sure you convert the units when appropriate.

ϵ_0 : Permittivity of vacuum = $8.86 \times 10^{-12}\mathrm{F/m}$

ϵ_{Si} : Relative permittivity of Si = $11.7\epsilon_0$

ϵ_{ox} : Relative permittivity of $\mathrm{SiO_2}$ = $3.9\epsilon_0$

μ_n : electron surface mobility, $\mathrm{cm^2/V/s}$

μ_p : hole surface mobility, $\mathrm{cm}^2/\mathrm{V}/\mathrm{s}$

C_{ox} : gate capacitance across the oxide per unit area, $\mathrm{fF}/\mu\mathrm{m}^2$

C_{dep} : capacitance of depletion region per unit area, $\mathrm{fF}/\mu\mathrm{m}^2$

t_{ox} : gate oxide thickness $\approx$ 3.8 nm for the class chip in 180 nm techology.

V_T : threshold voltage, V (V_{T0} is V_T when V_s = 0).

W : electrical width of transistor channel, $= 4\ \mu m$ for both devices in this lab

L : electrical length of transistor channel, $= 4\ \mu m$ for both devices in this lab

$\beta \equiv \mu C_{ox} W/L,\ \mu\mathrm{A}/\mathrm{V}^2$

V_E : Early voltage, characterizes drain conductance.

3 Prelab

Write the expressions/eqations in LaTeX, like Vod = $V_\mathrm{g} - V_\mathrm{T}$, or upload the pictures of handwritten expressions.

- For nFET, write the most general expression for I_{ds} above threshold in terms of $V_g,\ V_s,\ V_d$ (all voltages are referenced to the bulk), and the parameters and constants given above. Leave out the drain conductance Early effect in this equation. Assume $\kappa = 1$ and that $V_{Tn} > 0$.

> For triode: $I_{ds} = \beta(V_g - V_s - V_{Tn})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{Tn})^2$

- For pFET, write the most general expression for I_{ds} above threshold in terms of $V_g,\ V_s,\ V_d$ (all voltages are referenced to the bulk), and the parameters and constants given above. Leave out the drain conductance Early effect in this equation. Assume $\kappa = 1$ and that $V_{Tp} < 0$.

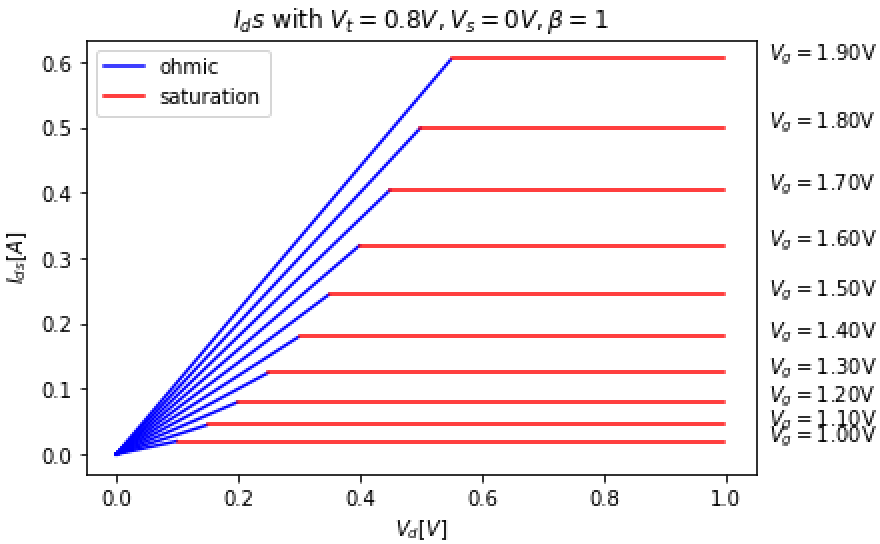
> For triode: $I_{ds} = \beta(V_g - Vs - V_{Tp})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - Vs - V_{Tp})^2$
> The formulae are the same, but the resulting graph is mirrored!

- For nFET, sketch graphs of I_{ds} vs the V_d for several gate voltages V_g above threshold, with $V_s = 0$. Indicate the ohmic and saturation regions and the behavior of the saturation voltage V_{dsat} as the gate overdrive voltage increases.

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
v_t = 0.8
v_s = 0
beta = 1
for v_g in np.arange(1, 2, 0.1):
    v_d1 = np.arange(0, 1, 0.001)
    v_d2 = np.arange(0, 1, 0.001)
    i_ds1 = beta*(v_g - v_s - v_t)*(v_d1 - v_s)
    i_ds2 = np.repeat(beta*0.5*(v_g - v_s - v_t)**2, 1/0.001)
```

```
    index = next(filter(lambda i_ds: abs(i_ds[1][0]-i_ds[1][1]) < 0.0001, enumerate(zip(i_
    plt.plot(v_d1[:index], i_ds1[:index], 'b')
    plt.plot(v_d2[index:], i_ds2[index:], 'r')
    plt.text(1.07, i_ds2[-1], f"$V_g = {v_g:.2f}$V")
    plt.title(f"$I_ds$ with $V_t = 0.8V, V_s = 0V, \\beta = 1$")
    plt.xlabel("$V_d[V]$")
    plt.ylabel("$I_{ds}[A]$")
    plt.legend(["ohmic", "saturation"])
plt.show()
```



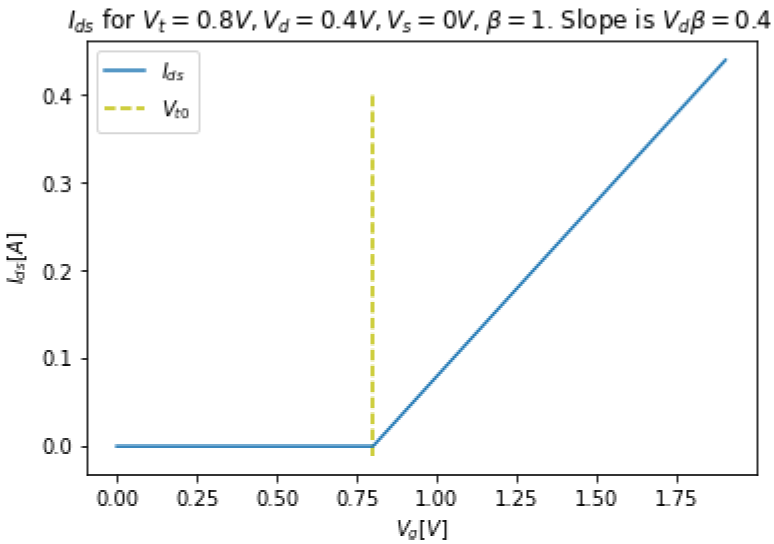
> The saturation voltage increases with the overdrive voltage

- For nFET, derive an expression for the current I_{ds} in the ohmic region in terms of V_g and $V_{ds} \equiv V_d - V_s$,. You may assume that $V_s = 0$. Sketch a graph of I_{ds} vs V_g, showing V_{T0} and an expression for the slope.

> $I_{ds} = \beta(V_g - V_{T0})V_d$

In [67]:

```
import numpy as np
import matplotlib.pyplot as plt
v_t = 0.8
v_s = 0
beta = 1
v_d = 0.4
v_g = np.arange(0.0, 2, 0.1)
i_ds = beta*(v_g - v_s - v_t)*(v_d - v_s)
i_ds = np.array([max(id, 0) for id in i_ds])
plt.plot(v_g, i_ds, label="$I_{ds}$")
plt.vlines(v_t, -0.01, v_d, colors="y", linestyles="dashed", label="$V_{t0}$")
plt.xlabel("$V_g[V]$")
plt.ylabel("$I_{ds}[A]$")
plt.title("$I_{ds}$ for $V_t = 0.8V, V_d = 0.4V, V_s = 0V, \\beta = 1$. Slope is $V_d \\be
plt.legend()
plt.show()
```

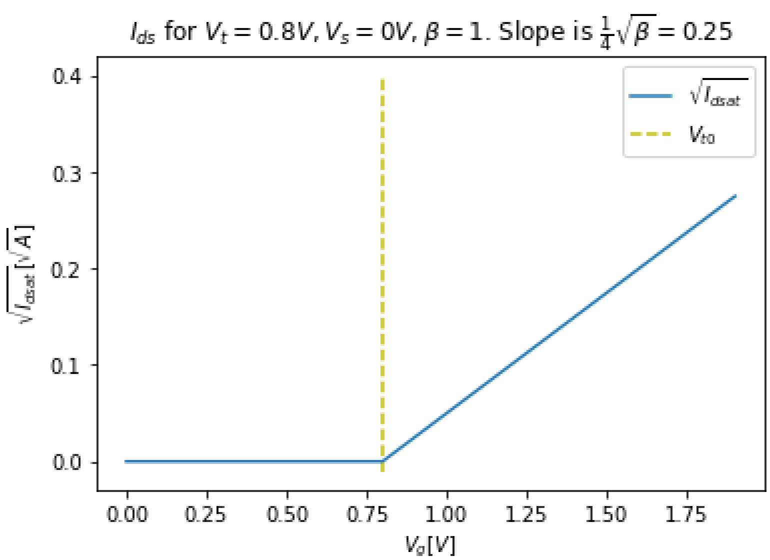


- For nFET, state the drain voltage condition for above-threshold saturation and derive an expression for the saturation current I_{dsat} in terms of V_g. Sketch a graph of $\sqrt{I_{dsat}}$ vs V_g with $V_s = 0$, showing V_{T0} and an expression for the slope. Do not consider the Early effect here.

> $I_{dsat} = \frac{1}{2}\beta(V_g - V_{T0})^2$

In [84]:

```
import numpy as np
import matplotlib.pyplot as plt
v_t = 0.8
v_s = 0
beta = 1
v_g = np.arange(0.0, 2, 0.1)
sqrt_i_dsat = np.sqrt(beta)*0.25*(v_g - v_s - v_t)
sqrt_i_dsat = np.array([max(id, 0) for id in sqrt_i_dsat])
plt.plot(v_g, sqrt_i_dsat, label="$\\sqrt{I_{dsat}}$")
plt.vlines(v_t, -0.01, v_d, colors="y", linestyles="dashed", label="$V_{t0}$")
plt.xlabel("$V_g[V]$")
plt.ylabel(r"$\sqrt{I_{dsat}}[\sqrt{A}]$")
plt.title(r"$I_{ds}$ for $V_t = 0.8V, V_s = 0V, \beta = 1$. Slope is $\frac{1}{4}\sqrt{\be
plt.legend()
plt.show()
```



- Calculate C_{ox} for the classchip from the values given above. What is C_{ox} per square micron in fF?

$$C_{ox} = \frac{\epsilon_{ox}}{t_{ox}}$$
$$C_{ox} = \frac{3.9 \times \epsilon_0}{t_{ox}}$$
$$C_{ox} = \frac{3.9 \times 8.86 \times 10^{-12} \frac{\mathrm{F}}{\mathrm{m}}}{3.8 \times 10^{-9} \mathrm{m}}$$
$$C_{ox} = 0.009093 \frac{\mathrm{F}}{\mathrm{m}^2} = 9093 \frac{\mathrm{fF}}{\mu\mathrm{m}^2}$$

- Write the expression for the drain current in saturation including the Early effect, using I_{dsat} to represent the saturation current in the absence of the Early effect. Use V_E to represent the Early voltage.

$$I_{ds} = I_{dsat}(1 + \frac{V_d - V_s}{V_E})$$

4 Setup

4.1 Connect the device

In [2]:

```
# import the necessary library to communicate with the hardware
#import sys
#sys.path.append('/home/junren/software/CoACH_Teensy_interface/build/pc/pyplane')

import pyplane
import time
```

In [3]:

```
# create a Plane object and open the communication
if 'p' not in locals():
    p = pyplane.Plane()
    try:
        p.open('/dev/ttyACM0') # Open the USB device ttyACM0 (the board).
    except RuntimeError as e:
        print(e)

# Note that if you plug out and plug in the USB device in a short time interval, the opera
# then you may get error messages with open(...ttyACM0). So please avoid frenquently plugg
```

In [4]:

```
p.get_firmware_version()
```

Out[4]:

```
(1, 8, 3)
```

In [5]:

```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
```

Out[5]:

```
<TeensyStatus.Success: 0>
```

In [6]:

```
# NOTE: You must send this request events every time you do a reset operetion, otherwise t
# Because the class chip need to handshake with some other devices to get the communicati
p.request_events(1)
```

In [7]:

```
# Try to read something, make sure the chip responses
p.read_current(pyplane.AdcChannel.GO0_N)
```

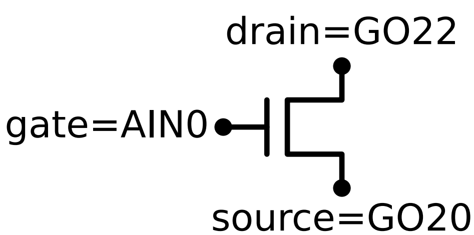
Out[7]: 1.8530273848682555e-07

In [7]:
```
# If any of the above steps fail, delete the object, and restart the kernel

# del p
```

4.2 Configurations for N-FET

In [8]:
```
# uses schemdraw, you may have to install it in order to run it on your PC
import schemdraw
import schemdraw.elements as elm
d = schemdraw.Drawing()
Q = d.add(elm.NFet, reverse=True)
d.add(elm.Dot, xy=Q.gate, lftlabel='gate=AIN0')
d.add(elm.Dot, xy=Q.drain, toplabel='drain=GO22')
d.add(elm.Dot, xy=Q.source, botlabel='source=GO20')
d.draw()
```

Out[8]:



To cancel out the leakage current and shunt resistance, you may need to do a subtraction.

$$I_{ds} = I_{GO20} - I_{GO20}|_{V_{gs}=0}$$

Note: It's better to measure source because its leakage is constant in this lab

- You have to set the input voltage demultiplexer by sending a configuration event:

In [9]:
```
# Configure NMOSFET, set the input voltage demultiplexer by AER.
# Note selectlines we should choose for the NMOSFET

events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

- Check the configuration is correct. If the measured result is not as expected, try sending the configration event again.

In [10]:
```
# set source voltage
vs = 0.0
p.set_voltage(pyplane.DacChannel.GO20 , vs)
print(f"The source voltage is set to {vs:.2f} V")
```

The source voltage is set to 0.00 V

In [11]:
```python
# set drain voltage
vd = 1.8
p.set_voltage(pyplane.DacChannel.GO22 , vd)
print(f"The drain voltage is set to {vd:.2f} V")
```

```
The drain voltage is set to 1.80 V
```

In [12]:
```python
# set gate voltage
vg = 1.0
p.set_voltage(pyplane.DacChannel.AIN0, vg)
print(f"The gate voltage is set to {vg:.2f} V")
```

```
The gate voltage is set to 1.00 V
```

In [13]:
```python
# read I_{ds}
I_s = p.read_current(pyplane.AdcChannel.GO20_N)      #source
print(f"The measured source current is {I_s} A")
I_d = p.read_current(pyplane.AdcChannel.GO22)       #drain
print(f"The measured drain current is {I_d} A")
```

```
The measured source current is 1.8554686903371476e-05 A
The measured drain current is 2.0117187887080945e-05 A
```

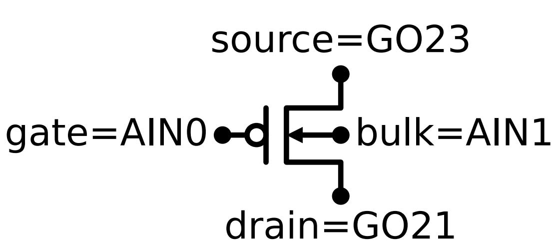
- Question: Check if the measured currents change with different gate voltages?

> Yes, a higher gate voltage will cause a higher current.

4.3 Configurations for P-FET

In [14]:
```python
# uses schemdraw, you may have to install it in order to run it on your PC
import schemdraw
import schemdraw.elements as elm
d = schemdraw.Drawing()
Q = d.add(elm.PFet, reverse=True, bulk=True)
d.add(elm.Dot, xy=Q.gate, lftlabel='gate=AIN0')
d.add(elm.Dot, xy=Q.bulk, rgtlabel='bulk=AIN1')
d.add(elm.Dot, xy=Q.drain, botlabel='drain=GO21')
d.add(elm.Dot, xy=Q.source, toplabel='source=GO23')
d.draw()
```

Out[14]:



Hint: To cancel out the leakage current and shunt resistance, you may need to do a subtraction:

$$I_{ds} = I_{GO23} - I_{GO23}|_{V_{gs}=0}$$

Note: It's better to measure source because its leakage is constant in this lab. Also think about the difference of V_{gs} between PMOS and NMOS?

- You have to choose the input voltage demultiplexer by sending a configuration event (make sure LED1 blinks):

In [15]:
```
# Configure PMOS, set the demultiplexer
# Note selectlines we should choose for the PMOSFET
# Note that SelectLine2 became SelectLine3. See the on page 12 of the manual.
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine1, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

- Check the configuration is correct. If the measured result is not as expected, try sending the event again.

In [16]:
```
# set trial voltages
Vs_p = 1.8
Vd_p = 0.0
Vg_p = 1.0

# set bulk voltage
# If the bulk voltage is not the source voltage, we will get an np junction
# If the bulk is lower than the source voltage, we will get the wrong flow of current
# So if we want to change the threshold, we could set the bulk voltage a bit higher than t
# But the bulk voltage should always be the highest (in PFET)
Vb_p = Vs_p


print("The bulk voltage is set to {} V".format(Vb_p))
p.set_voltage(pyplane.DacChannel.AIN1, Vb_p)
time.sleep(0.05)   # wait 0.05s for it to settle

# set source voltage
p.set_voltage(pyplane.DacChannel.GO23 , Vs_p)
print("The source voltage is set to {} V".format(Vs_p))
time.sleep(0.05)   # wait 0.05s for it to settle

# set drain voltage
p.set_voltage(pyplane.DacChannel.GO21 , Vd_p)
print("The drain voltage is set to {} V".format(Vd_p))
time.sleep(0.05)   # wait for it to settle

# set gate voltage
p.set_voltage(pyplane.DacChannel.AIN0, Vg_p)
print("The gate voltage is set to {} V".format(Vg_p))
```

```
The bulk voltage is set to 1.8 V
The source voltage is set to 1.8 V
The drain voltage is set to 0.0 V
The gate voltage is set to 1.0 V
```

In [17]:
```
# read I_{ds}
Is_p = p.read_current(pyplane.AdcChannel.GO23)       #source
print(f"The measured source current of PMOS is {Is_p} A")

Id_p = p.read_current(pyplane.AdcChannel.GO21_N)       #drain
print(f"The measured drain current of PMOS is {Id_p} A")
```

```
The measured source current of PMOS is 9.992675768444315e-05 A
The measured drain current of PMOS is 8.544922138753464e-07 A
```

> Correction: The two currents should be about the same.

5 Ohmic region

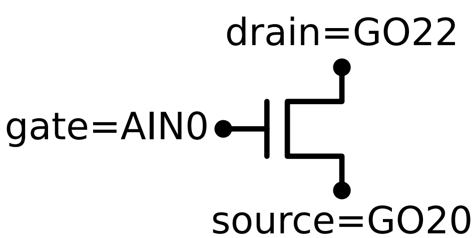
In this experiment you will characterize the *linear* dependence of the current on the gate voltage in the strong-inversion ohmic region.

5.1 N-FET

In [18]:

```
# uses schemdraw, you may have to install it in order to run it on your PC
import schemdraw
import schemdraw.elements as elm
d = schemdraw.Drawing()
Q = d.add(elm.NFet, reverse=True)
d.add(elm.Dot, xy=Q.gate, lftlabel='gate=AIN0')
d.add(elm.Dot, xy=Q.drain, toplabel='drain=GO22')
d.add(elm.Dot, xy=Q.source, botlabel='source=GO20')
d.draw()
```

Out[18]:



(a) Configure the chip following Section 4.2 if you haven't

(b) Measure I_{ds} as a function of V_g in ohmic region

In [67]:

```
# set the demultiplexer, NMOS
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

- What will be the fixed value for source and drain voltages?

Answer:

In [148…

```
# set source voltage
Vs = 0.0
p.set_voltage(pyplane.DacChannel.GO20 , Vs)
```

Out[148…

```
0.0
```

In [152…

```
# set drain voltage
Vd = 0.2
p.set_voltage(pyplane.DacChannel.GO22 , Vd)
```

```
0.19882699847221375
```

Out[152…

- For very close voltages, you may want to call `get_set_voltage` to check the actual output of the DAC.

In [128…

```
# get set voltage
Vs_n = p.get_set_voltage(pyplane.DacChannel.GO20)
print("The source voltage is set to {} V".format(Vs_n))

# get set voltage
Vd_n = p.get_set_voltage(pyplane.DacChannel.GO22)
print("The drain voltage is set to {} V".format(Vd_n))
```

```
The source voltage is set to 0.49970680475234985 V
The drain voltage is set to 1.7982406616210938 V
```

> Correction: The source and drain voltages were changed in between measurements. The original settings would have been more accurate, since the body effect raises Vt. Vs = Vbulk = GND with Vd = 0.2 V would have been more accurate.

- Data aquisition

In [153…

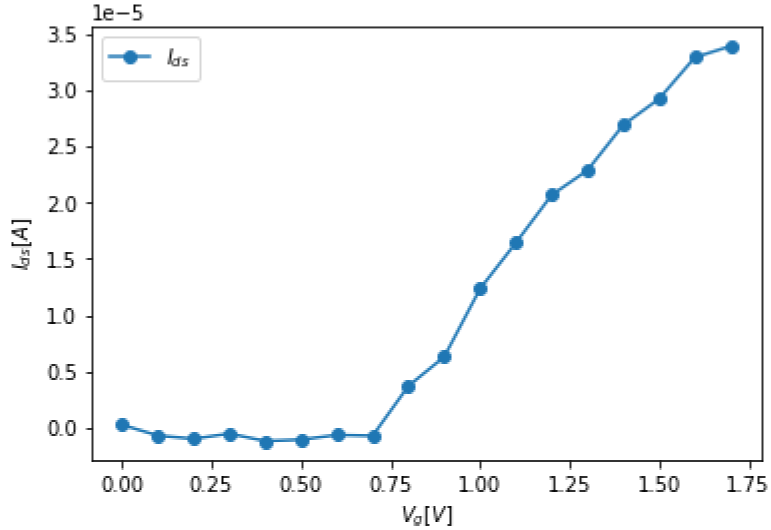
```
# sweep gate voltage
import time
import numpy as np

# Get the leakage current, Read Ids=Ids0 at Vg = 0
p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
time.sleep(0.5) # wait 0.5 second for it to settle
Is0_n = p.read_current(pyplane.AdcChannel.GO20_N)
print(f"Offset Is0_n: {Is0_n} A")

Vgs = np.arange(0.0, 1.8, 0.1)
Ids = []

for Vg in Vgs:
    # set gate voltage
    p.set_voltage(pyplane.DacChannel.AIN0, Vg)

    print(f"The gate voltage is set to {Vg:.2f} V")    ## print the gate voltage

    time.sleep(0.05)   # wait for it to settle
    # read I_{ds}
    Id = p.read_current(pyplane.AdcChannel.GO20_N)

    print(f"The measured source current is {Id} A")   ## print the raw data

    # substract leakage current
    Id -= Is0_n
    print(f"The leakage corrected Ids is {Id}")
    Ids.append(Id)
```

```
Offset Is0_n: 1.3671874512510842e-06 A
The gate voltage is set to 0.00 V
The measured source current is 1.6601562720097718e-06 A
The leakage corrected Ids is 2.929688207586878e-07
The gate voltage is set to 0.10 V
The measured source current is 7.080078034960025e-07 A
```

```
The leakage corrected Ids is -6.591796477550815e-07
The gate voltage is set to 0.20 V
The measured source current is 4.1503906800244295e-07 A
The leakage corrected Ids is -9.52148383248641e-07
The gate voltage is set to 0.30 V
The measured source current is 8.78906234902388e-07 A
The leakage corrected Ids is -4.88281216348696e-07
The gate voltage is set to 0.40 V
The measured source current is 2.19726558725597e-07 A
The leakage corrected Ids is -1.147460892525487e-06
The gate voltage is set to 0.50 V
The measured source current is 3.41796862812771e-07 A
The leakage corrected Ids is -1.025390588438313e-06
The gate voltage is set to 0.60 V
The measured source current is 7.568359592369234e-07 A
The leakage corrected Ids is -6.103514920141606e-07
The gate voltage is set to 0.70 V
The measured source current is 6.83593725625542e-07 A
The leakage corrected Ids is -6.83593725625542e-07
The gate voltage is set to 0.80 V
The measured source current is 5.102539034851361e-06 A
The leakage corrected Ids is 3.735351583600277e-06
The gate voltage is set to 0.90 V
The measured source current is 7.69042981119128e-06 A
The leakage corrected Ids is 6.3232423599401955e-06
The gate voltage is set to 1.00 V
The measured source current is 1.3720703464059625e-05 A
The leakage corrected Ids is 1.235351601280854e-05
The gate voltage is set to 1.10 V
The measured source current is 1.7797850887291133e-05 A
The leakage corrected Ids is 1.643066343604005e-05
The gate voltage is set to 1.20 V
The measured source current is 2.2045898731448688e-05 A
The leakage corrected Ids is 2.0678711280197604e-05
The gate voltage is set to 1.30 V
The measured source current is 2.4243163352366537e-05 A
The leakage corrected Ids is 2.2875975901115453e-05
The gate voltage is set to 1.40 V
The measured source current is 2.8271484552533366e-05 A
The leakage corrected Ids is 2.6904297101282282e-05
The gate voltage is set to 1.50 V
The measured source current is 3.059082155232318e-05 A
The leakage corrected Ids is 2.9223634101072093e-05
The gate voltage is set to 1.60 V
The measured source current is 3.427734554861672e-05 A
The leakage corrected Ids is 3.291015809736564e-05
The gate voltage is set to 1.70 V
The measured source current is 3.527832086547278e-05 A
The leakage corrected Ids is 3.3911133414221695e-05
```

In [8]:

```python
# plot
from matplotlib import pyplot as plt

plt.plot(Vgs, Ids, "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



In [155…

```
# if the data looks nice, save it!

# example :
Lab2_data_nFETVgIds = [Vgs,Ids]
# save to csv file
np.savetxt('./data/Lab2_data_nFETVgIds.csv', Lab2_data_nFETVgIds, delimiter=',')
```

In [1]:

```
Vgs, Ids = np.loadtxt('./data/Lab2_data_nFETVgIds.csv', delimiter=',')
```

In [9]:

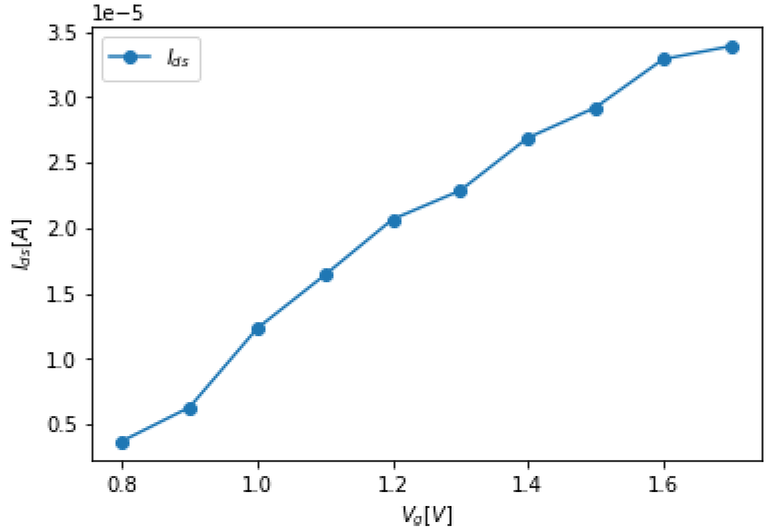
```
# extract the valid range
v_t0 = next(Vg for Vg, Id in zip(Vgs, Ids) if Id > 1e-6)
print(f"Threshold value is at Vg = {v_t0}")
ohmic_start_index = Vgs.tolist().index(v_t0)
```

```
Threshold value is at Vg = 0.8
```

> Correction: The real Vt is lower. The body effect makes Vt higher, use Vs = Vbulk = GND for a more accurate measurement.

In [10]:

```
# fit in the valid range (you may want to go back and add the fitted line in the plot)
plt.plot(Vgs[ohmic_start_index:], Ids[ohmic_start_index:], "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



(c) Determine V_{T0} and β for both devices by fitting your data to the expression derived in the prelab

In [11]:

```
# V_T0
v_t0 = Vgs[ohmic_start_index]
print(f"v_t0: {v_t0}")
```

```
v_t0: 0.8
```

> Correction: Vgs is just Vs, we should have subtracted Vs (0.5 V)

In [12]:

```
# beta => m/Vd
delta_Ids = Ids[-1] - Ids[ohmic_start_index]
delta_Vgs = Vgs[-1] - Vgs[ohmic_start_index]
m = delta_Ids / delta_Vgs
Vd = 1.8
Vs = 0.5
Vds = Vd - Vs
beta_n = m/Vds

print(f"Beta: {beta_n}")
```

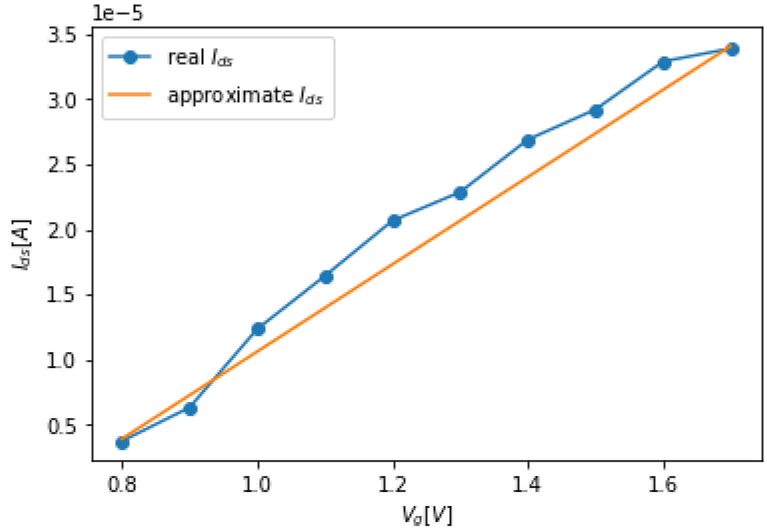
```
Beta: 2.5791266521898644e-05
```

> Correction: Beta must be wrong since we used Vg instead of Vgs. Should have subtracted Vs = 0.5 V

> For triode: $I_{ds} = \beta(V_g - V_s - V_{Tn})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{Tn})^2$

In [14]:

```
plt.plot(Vgs[ohmic_start_index:], Ids[ohmic_start_index:], "o-", label="real $I_{ds}$")
Vs = 0.5
Ids_approx = m*(Vgs[ohmic_start_index:] - Vs - v_t0) + Ids[ohmic_start_index + 4]
plt.plot(Vgs[ohmic_start_index:], Ids_approx, label="approximate $I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



5.2 P-FET

(a) Configure the chip following Section 4.3 if you haven't

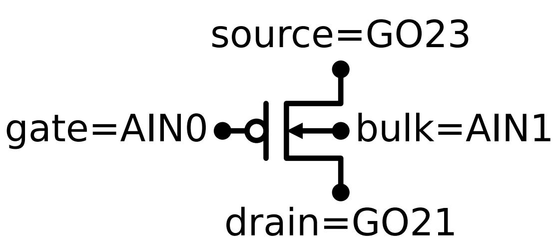
(b) Measure I_{ds} as a function of V_g in ohmic region

- What will be the fixed value for bulk, source and drain voltages?

In [6]:

```
# uses schemdraw, you may have to install it in order to run it on your PC
import schemdraw
import schemdraw.elements as elm
d = schemdraw.Drawing()
Q = d.add(elm.PFet, reverse=True, bulk=True)
d.add(elm.Dot, xy=Q.gate, lftlabel='gate=AIN0')
d.add(elm.Dot, xy=Q.bulk, rgtlabel='bulk=AIN1')
d.add(elm.Dot, xy=Q.drain, botlabel='drain=GO21')
d.add(elm.Dot, xy=Q.source, toplabel='source=GO23')
d.draw()
```

Out[6]:



In [7]:

```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
```

Out[7]: <TeensyStatus.Success: 0>

In [8]:

```
# Configure PMOS, set the demultiplexer
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine1, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

In [38]:

```
# set bulk voltage
p.set_voltage(pyplane.DacChannel.AIN1, 1.8)

time.sleep(0.05)  # wait for it to settle

# set source voltage
p.set_voltage(pyplane.DacChannel.GO23, 1.8)

# set drain voltage
vd = p.set_voltage(pyplane.DacChannel.GO21, 0.8)

# Print I_ds for checking
Ids = p.read_current(pyplane.AdcChannel.GO21_N)
print(f"Ids: {Ids}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-38-0da704bf3153> in <module>
      1 # set bulk voltage
----> 2 p.set_voltage(pyplane.DacChannel.AIN1, 1.8)
      3
      4 time.sleep(0.05)  # wait for it to settle
      5

NameError: name 'p' is not defined
```

> Correction: Should have reloaded notebook, Vd and Vs are not set

- For very close voltages, you may want to call `get_set_voltage` to check the actual output of the DAC.

In [37]:

```
# get set voltage
Vs_n = p.get_set_voltage(pyplane.DacChannel.GO23)
print("The source voltage is set to {} V".format(Vs_n))

# get set voltage
Vd_n = p.get_set_voltage(pyplane.DacChannel.GO21)
print("The drain voltage is set to {} V".format(Vd_n))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-37-ca4dc64ea83c> in <module>
      1 # get set voltage
----> 2 Vs_n = p.get_set_voltage(pyplane.DacChannel.GO23)
      3 print("The source voltage is set to {} V".format(Vs_n))
      4
      5 # get set voltage

NameError: name 'p' is not defined
```

> Correction: Should have reloaded notebook, Vd and Vs are not set

- Data aquisition

In [45]:

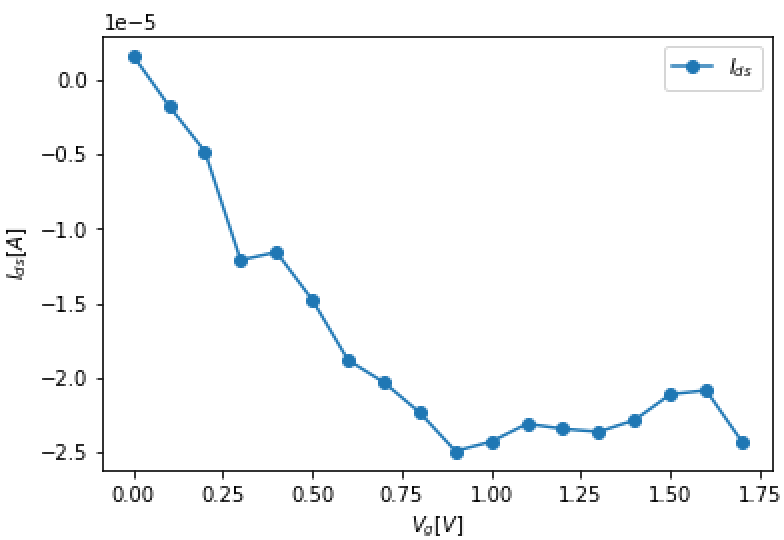
```
# sweep gate voltage
# sweep gate voltage
import time
import numpy as np
```

```python
# Get the leakage current, Read Ids=Ids0 at Vg = 0
p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
time.sleep(0.5) # wait 0.5 second for it to settle
Is0_n = p.read_current(pyplane.AdcChannel.GO21_N)
print(f"Offset Is0_n: {Is0_n} A")

Vgs = np.arange(0.0, 1.8, 0.1)
Ids = []

for Vg in Vgs:
    # set gate voltage
    p.set_voltage(pyplane.DacChannel.AIN0, Vg)

    print(f"The gate voltage is set to {Vg:.2f} V")    ## print the gate voltage

    time.sleep(0.05)  # wait for it to settle
    # read I_{ds}
    Id = p.read_current(pyplane.AdcChannel.GO21_N)

    print(f"The measured source current is {Id} A")  ## print the raw data

    # substract leakage current
    Id -= Is0_n
    print(f"The leakage corrected Ids is {Id}")
    Ids.append(Id)
```

```
Offset Is0_n: 2.6977539164363407e-05 A
The gate voltage is set to 0.00 V
The measured source current is 2.8588867280632257e-05 A
The leakage corrected Ids is 1.6113281162688509e-06
The gate voltage is set to 0.10 V
The measured source current is 2.5219726012437604e-05 A
The leakage corrected Ids is -1.7578131519258022e-06
The gate voltage is set to 0.20 V
The measured source current is 2.2167969291331246e-05 A
The leakage corrected Ids is -4.80956987303216e-06
The gate voltage is set to 0.30 V
The measured source current is 1.4868163816572633e-05 A
The leakage corrected Ids is -1.2109375347790774e-05
The gate voltage is set to 0.40 V
The measured source current is 1.540527409815695e-05 A
The leakage corrected Ids is -1.1572265066206455e-05
The gate voltage is set to 0.50 V
The measured source current is 1.220703143189894e-05 A
The leakage corrected Ids is -1.4770507732464466e-05
The gate voltage is set to 0.60 V
The measured source current is 8.15429666545242e-06 A
The leakage corrected Ids is -1.8823242498910986e-05
The gate voltage is set to 0.70 V
The measured source current is 6.665039109066129e-06 A
The leakage corrected Ids is -2.0312500055297278e-05
The gate voltage is set to 0.80 V
The measured source current is 4.638671725842869e-06 A
The leakage corrected Ids is -2.2338867438520538e-05
The gate voltage is set to 0.90 V
The measured source current is 2.050781176876626e-06 A
The leakage corrected Ids is -2.492675798748678e-05
The gate voltage is set to 1.00 V
The measured source current is 2.6855468604480848e-06 A
The leakage corrected Ids is -2.4291992303915322e-05
The gate voltage is set to 1.10 V
The measured source current is 3.8818361645098776e-06 A
The leakage corrected Ids is -2.309570299985353e-05
The gate voltage is set to 1.20 V
```

```
The measured source current is 3.5400389606365934e-06 A
The leakage corrected Ids is -2.3437500203726813e-05
The gate voltage is set to 1.30 V
The measured source current is 3.344726565046585e-06 A
The leakage corrected Ids is -2.363281259931682e-05
The gate voltage is set to 1.40 V
The measured source current is 4.1259763747802936e-06 A
The leakage corrected Ids is -2.2851562789583113e-05
The gate voltage is set to 1.50 V
The measured source current is 5.859375050931703e-06 A
The leakage corrected Ids is -2.1118164113431703e-05
The gate voltage is set to 1.60 V
The measured source current is 6.127929736976512e-06 A
The leakage corrected Ids is -2.0849609427386895e-05
The gate voltage is set to 1.70 V
The measured source current is 2.6855468604480848e-06 A
The leakage corrected Ids is -2.4291992303915322e-05
```

In [16]:
```
# plot
from matplotlib import pyplot as plt

plt.plot(Vgs, Ids, "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



> Correction: Currents are negative because we measured an invalid leakage current. Measured at Vg = 0 V, should have measured at Vg = Vbulk = Vdd = 1.8 V

In [47]:
```
# if the data looks nice, save it!
# example :
Lab2_data_pFETVgIds = [Vgs,Ids]
# save to csv file
np.savetxt('./data/Lab2_data_pFETVgIds.csv', Lab2_data_pFETVgIds, delimiter=',')
```

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt

Vgs, Ids = np.loadtxt('./data/Lab2_data_pFETVgIds.csv', delimiter=',')
```

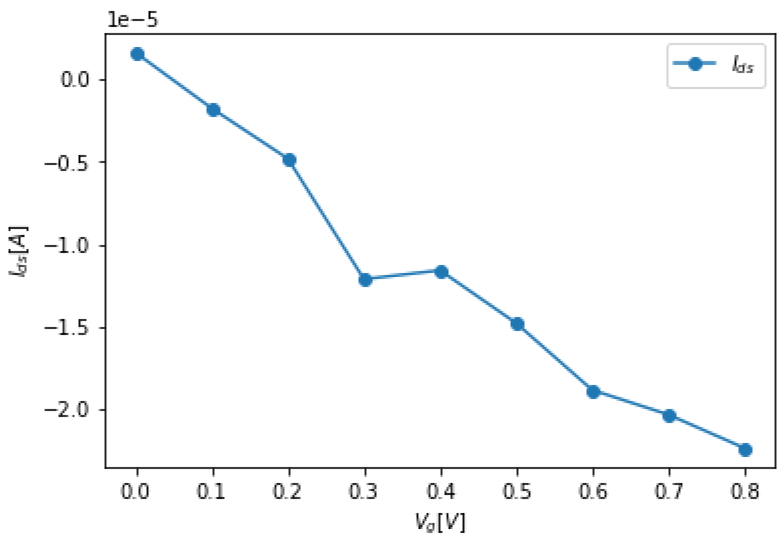
In [7]:

```
# extract the valid range
v_t0 = next(Vg for Vg, Id in zip(Vgs, Ids) if Id < -2.3e-5)
print(f"Threshold value is at Vg = {v_t0}")
ohmic_end_index = Vgs.tolist().index(v_t0)
```

```
Threshold value is at Vg = 0.9
```

In [19]:

```
# fit in the valid range (you may want to go back and add the fitted line in the plot)
plt.plot(Vgs[:ohmic_end_index], Ids[:ohmic_end_index], "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



(c) Determine V_{T0} and β for both devices by fitting your data to the expression derived in the prelab

In [8]:

```
# V_T0
v_t0 = Vgs[ohmic_end_index]
print(f"v_t0: {v_t0}")
```

```
v_t0: 0.9
```

In [12]:

```
# Correction:
Vs = 1.8
Vgs = Vs - Vgs
# V_T0
v_t0 = Vgs[ohmic_end_index]
print(f"v_t0: {v_t0}")
```

```
v_t0: 0.9
```

In [16]:

```
# beta => m/Vd
delta_Ids = Ids[ohmic_end_index] - Ids[0]
delta_Vgs = Vgs[ohmic_end_index] - Vgs[0]
m = delta_Ids / delta_Vgs
Vs = 1.8
Vd = 0.8
Vds = Vs - Vd
beta_p = m / Vds
print(f"Beta: {beta_p}")
```

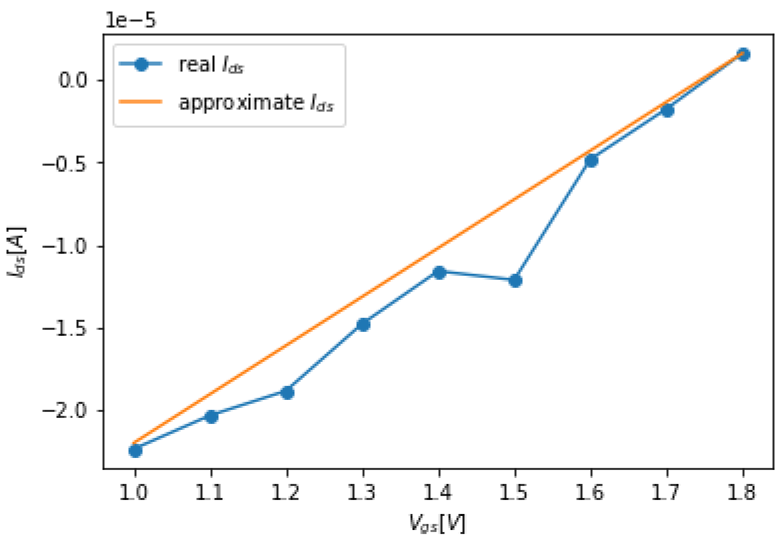
```
Beta: 2.9486762337506258e-05
```

> Correction: Using Vg instead of Vgs

> For triode: $I_{ds} = \beta(V_g - V_s - V_{Tp})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{Tp})^2$

In [17]:

```
plt.plot(Vgs[:ohmic_end_index], Ids[:ohmic_end_index], "o-", label="real $I_{ds}$")
# + v_t0 since v_t0 > 0
Ids_approx = m*(Vgs[:ohmic_end_index] - Vs + v_t0) - abs(Ids[ohmic_end_index])
plt.plot(Vgs[:ohmic_end_index], Ids_approx, label="approximate $I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{gs}[V]$")
plt.show()
```

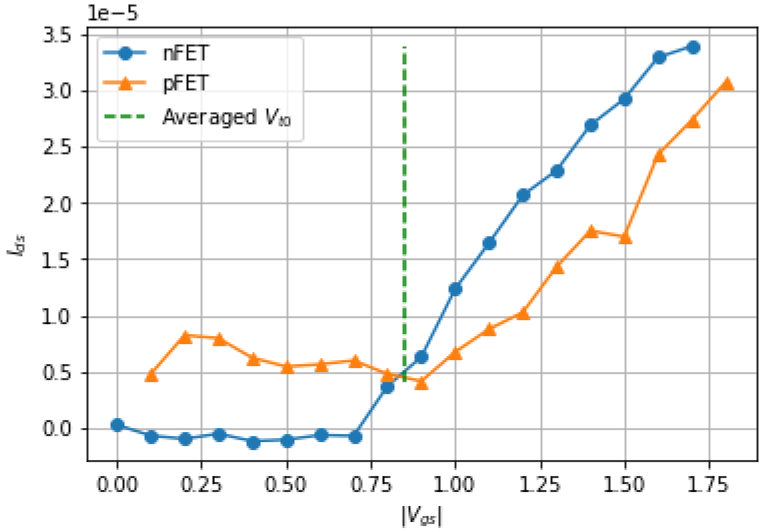


5.3 Comparisons

- Include a single plot showing the curves for both devices.

In [2]:

```
# Correction:
# plot both Ids vs |Vgs|
Vs_p = 1.8
Vgs, Ids_n = np.loadtxt('./data/Lab2_data_nFETVgIds.csv', delimiter=',')
Vgs_p, Ids_p = np.loadtxt('./data/Lab2_data_pFETVgIds.csv', delimiter=',')
guesstimated_p_leakage = abs(Ids_n[-1] - Ids_p[-1]) / 2
Ids_p += guesstimated_p_leakage
Vgs_p = abs(Vgs_p - Vs_p)
plt.plot(Vgs, Ids_n, "o-", label="nFET")
plt.plot(Vgs_p, Ids_p, "^-", label="pFET")
plt.xlabel("$|V_{gs}|$")
plt.ylabel("$I_{ds}$")
plt.vlines(0.85, min(Ids_p), max(Ids_n), linestyles='dashed', colors="g", label="Averaged
plt.legend()
plt.grid()
plt.show()
```



- What is the ratio between β for the 2 devices? Does it make sense?

In [188…

```
ratio = beta_n / beta_p
print(f"Beta_n: {beta_n}")
print(f"Beta_p: {beta_p}")
print(f"Ratio: {ratio}")
```

```
Beta_n: 2.5791266521898644e-05
Beta_p: 2.9486762337506258e-05
Ratio: 0.8746727167496766
```

> Ideally, the ratio should be 1. It is not however, since our pFET measurements suffered from multiple hardware defects, according to our TA.

- Is the relationship between I_{ds} and $V_{gs} - V_T$ really linear? What is likely the cause of any discrepancy?

> It seems to form a slight parabola. I guess this happens because we are getting close to the maximum current that can flow to the channel. At some point, the gate is bound to have diminishing returns.

5.4 Effective surface mobility (optional)

Hint: Use the V_{T0} you obtained in the last experiments but assume β changes with V_{gs} (thus μ_n and μ_p changes). **No need to measure again.**

In [ ]:

```
# plot mu vs Vgs for both devices in the same figure
```

- Why does the mobility peak and then decay instead of remaining constant?

- What is the ratio between the peak mobilities for electrons and holes?

- How different are these values from the bulk mobilities for electrons (1350 $\mathrm{cm}^2/\mathrm{V}/\mathrm{s}$) and holes (480 $\mathrm{cm}^2/\mathrm{V}/\mathrm{s}$)?

6 Drain Current in the saturation region

In this experiment you will characterize the *quadratic* dependence of the current on the gate voltage in the saturation region.

6.1 N-FET

(a) Configure the chip following Section 4.2 if you haven't

(b) Measure I_{ds} as a function of V_g in saturation region

- What will be the fixed value for source and drain voltages?

In [21]:
```
## configure NMOS
# set the demultiplexer, NMOS
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

In [22]:
```
# set source voltage
p.set_voltage(pyplane.DacChannel.GO20, 0.0)
```

Out[22]:
```
0.0
```

In [23]:
```
# set drain voltage      #######1.8
p.set_voltage(pyplane.DacChannel.GO22, 1.8)
```

Out[23]:
```
1.7982406616210938
```

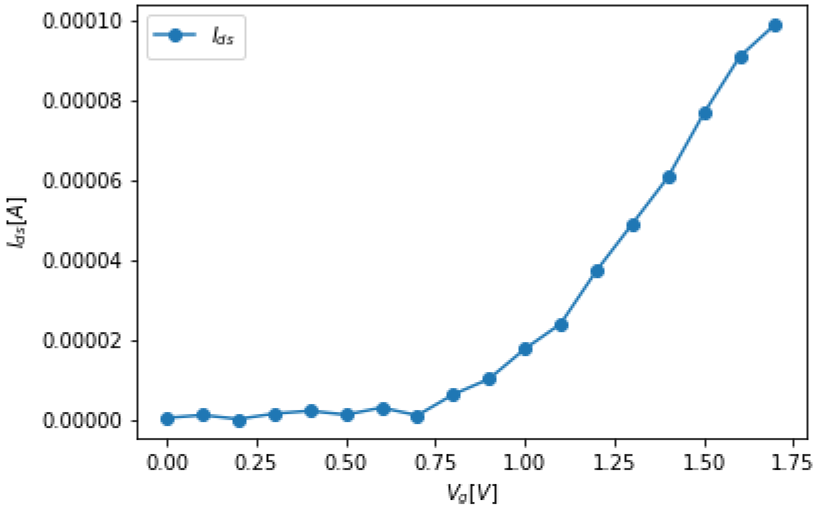
- Data aquisition

In [24]:
```
# sweep gate voltage
# sweep gate voltage
import time
import numpy as np

# Get the leakage current, Read Ids=Ids0 at Vg = 0
p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
time.sleep(0.5) # wait 0.5 second for it to settle
Is0_n = p.read_current(pyplane.AdcChannel.GO20_N)
print(f"Offset Is0_n: {Is0_n} A")

Vgs = np.arange(0.0, 1.8, 0.1)
Ids = []

for Vg in Vgs:
    # set gate voltage
    p.set_voltage(pyplane.DacChannel.AIN0, Vg)

    print(f"The gate voltage is set to {Vg:.2f} V")    ## print the gate voltage

    time.sleep(0.05)  # wait for it to settle
    # read I_{ds}
```

```python
    Id = p.read_current(pyplane.AdcChannel.GO20_N)

    print(f"The measured source current is {Id} A")   ## print the raw data

    # substract leakage current
    Id -= Is0_n
    print(f"The leakage corrected Ids is {Id}")
    Ids.append(Id)
```

```
Offset Is0_n: 2.4414063659605745e-07 A
The gate voltage is set to 0.00 V
The measured source current is 7.568359592369234e-07 A
The leakage corrected Ids is 5.126953226408659e-07
The gate voltage is set to 0.10 V
The measured source current is 1.4892577837599674e-06 A
The leakage corrected Ids is 1.24511714716391e-06
The gate voltage is set to 0.20 V
The measured source current is 4.6386719532165444e-07 A
The leakage corrected Ids is 2.19726558725597e-07
The gate voltage is set to 0.30 V
The measured source current is 1.8310546465727384e-06 A
The leakage corrected Ids is 1.586914009976681e-06
The gate voltage is set to 0.40 V
The measured source current is 2.5390625069121597e-06 A
The leakage corrected Ids is 2.2949218703161023e-06
The gate voltage is set to 0.50 V
The measured source current is 1.56249996052793e-06 A
The leakage corrected Ids is 1.3183593239318725e-06
The gate voltage is set to 0.60 V
The measured source current is 3.344726565046585e-06 A
The leakage corrected Ids is 3.1005859284505277e-06
The gate voltage is set to 0.70 V
The measured source current is 1.3916015859649633e-06 A
The leakage corrected Ids is 1.1474609493689059e-06
The gate voltage is set to 0.80 V
The measured source current is 6.6162110670120455e-06 A
The leakage corrected Ids is 6.372070430415988e-06
The gate voltage is set to 0.90 V
The measured source current is 1.0498047231521923e-05 A
The leakage corrected Ids is 1.0253906594925866e-05
The gate voltage is set to 1.00 V
The measured source current is 1.799316487449687e-05 A
The leakage corrected Ids is 1.7749024237900812e-05
The gate voltage is set to 1.10 V
The measured source current is 2.426757782814093e-05 A
The leakage corrected Ids is 2.4023437191544872e-05
The gate voltage is set to 1.20 V
The measured source current is 3.752441261895001e-05 A
The leakage corrected Ids is 3.728027198235395e-05
The gate voltage is set to 1.30 V
The measured source current is 4.9194335588254035e-05 A
The leakage corrected Ids is 4.895019495165798e-05
The gate voltage is set to 1.40 V
The measured source current is 6.0913087509106845e-05 A
The leakage corrected Ids is 6.066894687251079e-05
The gate voltage is set to 1.50 V
The measured source current is 7.68554673413746e-05 A
The leakage corrected Ids is 7.661132670477855e-05
The gate voltage is set to 1.60 V
The measured source current is 9.09912123461254e-05 A
The leakage corrected Ids is 9.074707170952934e-05
The gate voltage is set to 1.70 V
The measured source current is 9.912109089782462e-05 A
The leakage corrected Ids is 9.887695026122856e-05
```

In [5]:
```
# plot
plt.plot(Vgs, Ids, "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



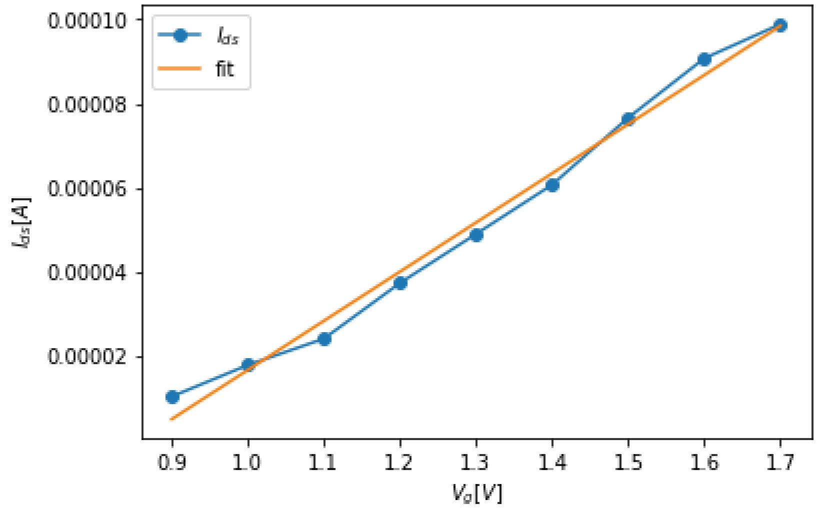
In [99]:
```
# if the data looks nice, save it!
# example :
Lab2_data_nFETVgIds = [Vgs,Ids]
# save to csv file
np.savetxt('./data/Lab2_data_nFETVgIds_saturated.csv', Lab2_data_nFETVgIds, delimiter=',')
```

In [12]:
```
Vgs, Ids = np.loadtxt('./data/Lab2_data_nFETVgIds_saturated.csv', delimiter=',')
```

In [13]:
```
# extract the valid range and plot sqrt(Ids) vs Vgs
v_t0 = next(Vg for Vg, Id in zip(Vgs, Ids) if Id > 1e-5)
print(f"Threshold value is at Vg = {v_t0}")
sat_start_index = Vgs.tolist().index(v_t0)
```

```
Threshold value is at Vg = 0.9
```

In [15]:
```
# Correction:
# fit in the valid range (you may want to go back and add the fitted line in the plot)
plt.plot(Vgs[sat_start_index:], Ids[sat_start_index:], "o-", label="$I_{ds}$")
m, b = np.polyfit(Vgs[sat_start_index:], Ids[sat_start_index:], 1)
x = np.linspace(Vgs[sat_start_index], Vgs[-1], 100)
plt.plot(x, m*x + b, label="fit")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



(c) Determine V_{T0} and β for both devices by fitting your data to the expression derived in the prelab

In [8]:
```
# V_T0
v_t0 = Vgs[sat_start_index]
print(f"v_t0: {v_t0}")
```

```
v_t0: 0.9
```

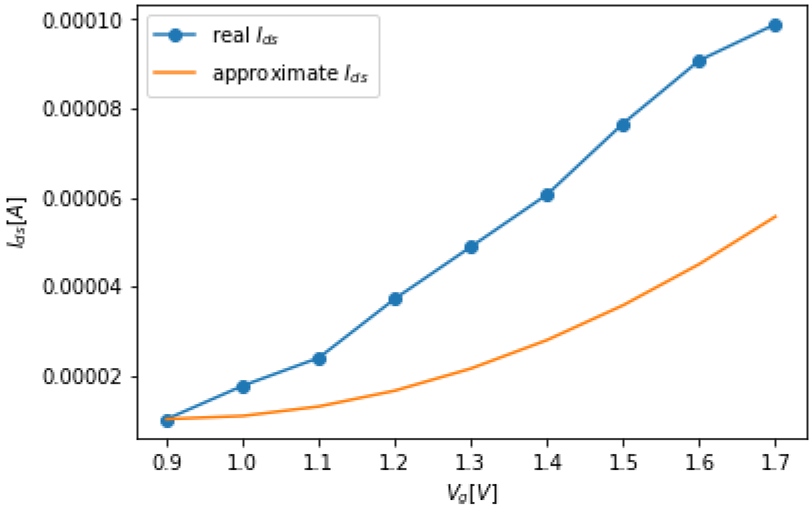
> For triode: $I_{ds} = \beta(V_g - V_s - V_{Tn})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{Tn})^2$

> $$I_{dsat} = \frac{1}{2}\beta(V_g - V_s - V_{T0})^2$$
> $$\Rightarrow \sqrt{I_{dsat}} = \frac{1}{\sqrt{2}}\sqrt{\beta}(V_g - V_s - V_{T0})$$
> $$m := \text{slope of } \sqrt{\mathrm{I_{dsat}}}(\mathrm{V_g}) = \frac{1}{\sqrt{2}}\sqrt{\beta}$$
> $$\Rightarrow \beta = 2m^2$$

In [8]:
```
# beta
delta_Ids = np.sqrt(Ids[-1]) - np.sqrt(Ids[sat_start_index])
delta_Vgs = Vgs[-1] - Vgs[sat_start_index]
m = delta_Ids / delta_Vgs
beta_n_sat = 2*m**2

print(f"Beta: {beta_n_sat}")
```

```
Beta: 0.00014202515231445965
```

In [10]:
```
plt.plot(Vgs[sat_start_index:], Ids[sat_start_index:], "o-", label="real $I_{ds}$")
Ids_approx = 0.5*beta_n_sat*(Vgs[sat_start_index:] - v_t0)**2 + Ids[sat_start_index]
plt.plot(Vgs[sat_start_index:], Ids_approx, label="approximate $I_{ds}$")

plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



6.2 P-FET

(a) Configure the chip following Section 4.3 if you haven't

(b) Measure I_{ds} as a function of V_g in ohmic region

- What will be the fixed value for bulk, source and drain voltages?

In [27]:
```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
```

Out[27]: <TeensyStatus.Success: 0>

In [28]:
```
# Configure PMOS, set the demultiplexer
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine1, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

In [29]:
```
# set bulk voltage
p.set_voltage(pyplane.DacChannel.AIN1, 1.8)

time.sleep(0.05)   # wait for it to settle

# set source voltage
p.set_voltage(pyplane.DacChannel.GO23, 1.8)

# set drain voltage
p.set_voltage(pyplane.DacChannel.GO21, 0.0)

# Print I_ds for checking
Ids = p.read_current(pyplane.AdcChannel.GO21_N)
print(f"Ids: {Ids}")
```

```
Ids: 2.8686523364740424e-05
```

- For very close voltages, you may want to call `get_set_voltage` to check the actual output of the DAC.

In [30]:

```
# get set voltage
Vs_n = p.get_set_voltage(pyplane.DacChannel.GO23)
print("The source voltage is set to {} V".format(Vs_n))

# get set voltage
Vd_n = p.get_set_voltage(pyplane.DacChannel.GO21)
print("The drain voltage is set to {} V".format(Vd_n))
```

```
The source voltage is set to 1.7982406616210938 V
The drain voltage is set to 0.0 V
```

- Data aquisition

In [31]:

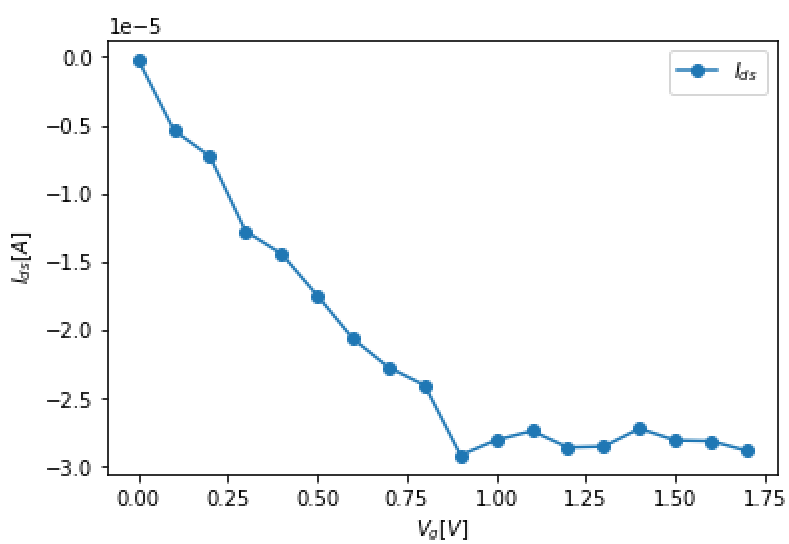
```
# sweep gate voltage
import time
import numpy as np

# Get the leakage current, Read Ids=Ids0 at Vg = 0
p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
time.sleep(0.5) # wait 0.5 second for it to settle
Is0_n = p.read_current(pyplane.AdcChannel.GO21_N)
print(f"Offset Is0_n: {Is0_n} A")

Vgs = np.arange(0.0, 1.8, 0.1)
Ids = []

for Vg in Vgs:
    # set gate voltage
    p.set_voltage(pyplane.DacChannel.AIN0, Vg)

    print(f"The gate voltage is set to {Vg:.2f} V")    ## print the gate voltage

    time.sleep(0.05)  # wait for it to settle
    # read I_{ds}
    Id = p.read_current(pyplane.AdcChannel.GO21_N)

    print(f"The measured source current is {Id} A")  ## print the raw data

    # substract leakage current
    Id -= Is0_n
    print(f"The leakage corrected Ids is {Id}")
    Ids.append(Id)
```

```
Offset Is0_n: 2.9443359380820766e-05 A
The gate voltage is set to 0.00 V
The measured source current is 2.922363273683004e-05 A
The leakage corrected Ids is -2.1972664399072528e-07
The gate voltage is set to 0.10 V
The measured source current is 2.4047851184150204e-05 A
The leakage corrected Ids is -5.395508196670562e-06
The gate voltage is set to 0.20 V
The measured source current is 2.2167969291331246e-05 A
The leakage corrected Ids is -7.27539008948952e-06
The gate voltage is set to 0.30 V
The measured source current is 1.6674805010552518e-05 A
The leakage corrected Ids is -1.2768554370268248e-05
The gate voltage is set to 0.40 V
The measured source current is 1.5039062418509275e-05 A
The leakage corrected Ids is -1.4404296962311491e-05
The gate voltage is set to 0.50 V
The measured source current is 1.1987304787908215e-05 A
The leakage corrected Ids is -1.745605459291255e-05
The gate voltage is set to 0.60 V
The measured source current is 8.813476597424597e-06 A
```

```
The leakage corrected Ids is -2.0629882783396l7e-05
The gate voltage is set to 0.70 V
The measured source current is 6.713867151120212e-06 A
The leakage corrected Ids is -2.2729492229700554e-05
The gate voltage is set to 0.80 V
The measured source current is 5.419921762950253e-06 A
The leakage corrected Ids is -2.4023437617870513e-05
The gate voltage is set to 0.90 V
The measured source current is 3.1738281336401997e-07 A
The leakage corrected Ids is -2.9125976567456746e-05
The gate voltage is set to 1.00 V
The measured source current is 1.416015606992005e-06 A
The leakage corrected Ids is -2.802734377382876e-05
The gate voltage is set to 1.10 V
The measured source current is 2.075195425277343e-06 A
The leakage corrected Ids is -2.7368163955543423e-05
The gate voltage is set to 1.20 V
The measured source current is 8.78906234902388e-07 A
The leakage corrected Ids is -2.8564453145918378e-05
The gate voltage is set to 1.30 V
The measured source current is 9.521484116703505e-07 A
The leakage corrected Ids is -2.8491210969150416e-05
The gate voltage is set to 1.40 V
The measured source current is 2.2460937998403097e-06 A
The leakage corrected Ids is -2.7197265580980456e-05
The gate voltage is set to 1.50 V
The measured source current is 1.3916015859649633e-06 A
The leakage corrected Ids is -2.8051757794855803e-05
The gate voltage is set to 1.60 V
The measured source current is 1.3427734302240424e-06 A
The leakage corrected Ids is -2.8100585950596724e-05
The gate voltage is set to 1.70 V
The measured source current is 6.347656267280399e-07 A
The leakage corrected Ids is -2.8808593754092726e-05
```

In [12]:

```
# plot
from matplotlib import pyplot as plt

plt.plot(Vgs, Ids, "o-", label="$I_{ds}$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



> Correction: Currents are negative because we measured an invalid leakage current. Measured

> at Vg = 0 V, should have measured at Vg = Vbulk = Vdd = 1.8 V

In [35]:
```
# if the data looks nice, save it!
# example :
Lab2_data_pFETVgIds = [Vgs,Ids]
# save to csv file
np.savetxt('./data/Lab2_data_pFETVgIds_saturated.csv', Lab2_data_pFETVgIds, delimiter=',')
```

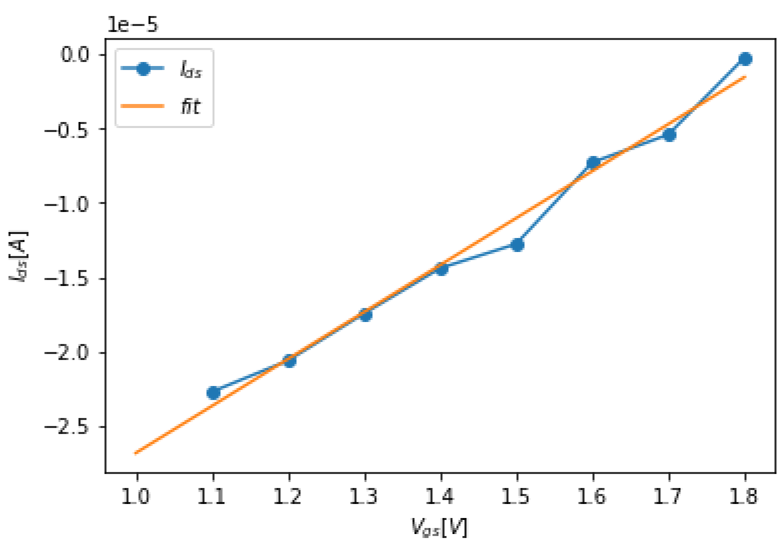
- Data aquisition

In [18]:
```
import numpy as np
import matplotlib.pyplot as plt
Vgs, Ids = np.loadtxt('./data/Lab2_data_pFETVgIds_saturated.csv', delimiter=',')
```

In [19]:
```
# extract the valid range and plot sqrt(Ids) vs Vgs
Vs = 1.8
Vgs = Vs - Vgs
v_t0 = next(Vg for Vg, Id in zip(Vgs, Ids) if Id < -2.3e-5)
print(f"Threshold value is at Vg = {v_t0}")
ohmic_end_index = Vgs.tolist().index(v_t0)
```

```
Threshold value is at Vg = 1.0
```

In [24]:
```
# Correction:
# fit in the valid range (you may want to go back and add the fitted line in the plot)
plt.plot(Vgs[:ohmic_end_index], Ids[:ohmic_end_index], "o-", label="$I_{ds}$")
m, b = np.polyfit(Vgs[:ohmic_end_index], Ids[:ohmic_end_index], 1)
x= np.linspace(Vgs[0], Vgs[ohmic_end_index], 100)
plot = plt.plot(x, m*x + b, label="$fit$")
plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{gs}[V]$")
plt.show()
```



(c) Determine V_{T0} and β for both devices by fitting your data to the expression derived in the prelab

In [28]:
```
# V_T0
v_t0 = Vgs[ohmic_end_index]
print(f"v_t0: {v_t0}")
```

```
v_t0: 1.0
```

> For triode: $I_{ds} = \beta(V_g - V_s - V_{Tn})(V_d - V_s)$
> For saturation: $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{Tn})^2$

> $I_{ds} = \frac{1}{2}\beta(V_g - V_s - V_{T0})^2$
> $\Rightarrow \sqrt{I_{dsat}} = \frac{1}{\sqrt{2}}\sqrt{\beta}(V_g - V_s - V_{T0})$
> $m :=$ slope of $\sqrt{\mathrm{I_{dsat}}}(\mathrm{V_g}) = \frac{1}{\sqrt{2}}\sqrt{\beta}$
> $\Rightarrow \beta = 2m^2$

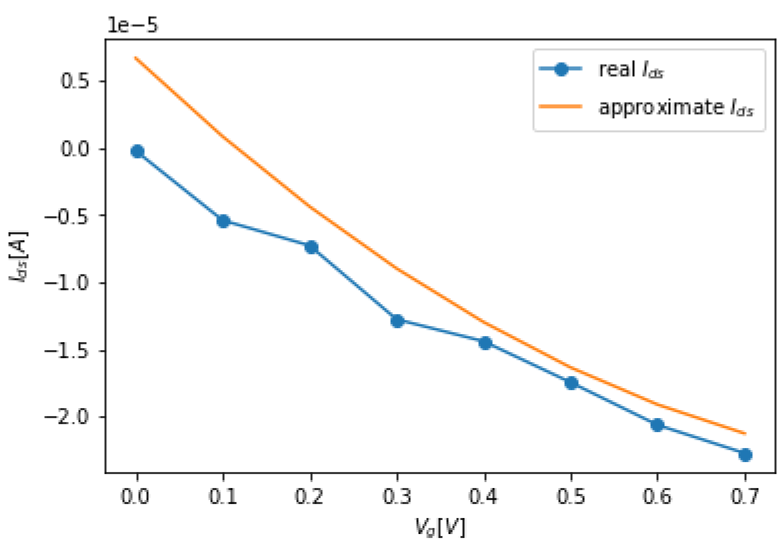
In [26]:

```
# beta
delta_Ids = np.sqrt(abs(Ids[ohmic_end_index])) - np.sqrt(abs(Ids[0]))
delta_Vgs = Vgs[ohmic_end_index] - Vgs[0]
m = delta_Ids / delta_Vgs
beta_p_sat = 2*m**2

print(f"Beta: {beta_p_sat}")
```

```
Beta: 6.140040032243991e-05
```

In [17]:

```
plt.plot(Vgs[:ohmic_end_index], Ids[:ohmic_end_index], "o-", label="real $I_{ds}$")
Vs = 1.8
# + v_t0 since v_t0 > 0
Ids_approx = abs(0.5*beta_p_sat*(Vgs[:ohmic_end_index] - Vs + v_t0)**2) - abs(Ids[ohmic_er
plt.plot(Vgs[:ohmic_end_index], Ids_approx, label="approximate $I_{ds}$")

plt.legend()
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{g}[V]$")
plt.show()
```



6.3 Comparisons

- Are the measurements of V_{T0} and β from the saturation measurement consistent with the values obtained in the ohmic region?

> V_{T0} is more or less the same, but β varies greatly

- Which is a better approximation, the linear one or the quadratic?

> The linear line fits better. This is probably so since the quadratic approximation squares any inaccuracies. In fact, when I testwise plotted the root of the collected data and the root of the approximation, they matched quite well.

7 Early effect

This experiment studies how Early voltage scales with transistor current; in particular, how valid are the simple assumptions about channel length modulation?

You only need to do N-FET

(a) Measure I_{ds} vs V_{ds} for different V_{gs}

In [8]:
```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
```

Out[8]:
```
<TeensyStatus.Success: 0>
```

In [9]:
```
# Configure PMOS, set the demultiplexer
events = [pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.NoneSelected, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)]

p.send_coach_events(events)
```

In [10]:
```
# set source voltage
p.set_voltage(pyplane.DacChannel.GO20, 0.0)
```

Out[10]:
```
0.0
```

- Data aquisition

In [18]:
```
# sweep gate voltage
import time
import numpy as np

# Get the leakage current, Read Ids=Ids0 at Vg = 0
p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
time.sleep(0.5) # wait 0.5 second for it to settle
Is0_n = p.read_current(pyplane.AdcChannel.GO21_N)

print(f"Offset Is0_n: {Is0_n} A")

Vds = np.arange(0.0, 1.8, 0.1)
Vgs = np.arange(0.8, 1.8, 0.2)
Ids = {}
for Vg in Vgs:
    Ids[Vg] = []

for Vd in Vds:
    for Vg in Vgs:
        p.set_voltage(pyplane.DacChannel.GO22, Vd)
```

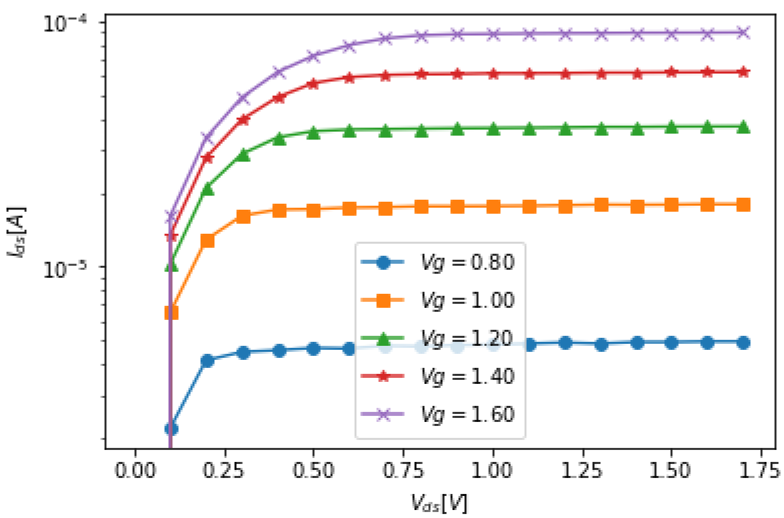
```
        # Get the leakage current, Read Ids=Ids0 at Vg = 0
        p.set_voltage(pyplane.DacChannel.AIN0, 0.0)
        time.sleep(0.5) # wait 0.5 second for it to settle
        Is0_n = p.read_current(pyplane.AdcChannel.GO21_N)
        p.set_voltage(pyplane.DacChannel.AIN0, Vg)

        time.sleep(0.05)  # wait for it to settle
        # read I_{ds}
        Id = p.read_current(pyplane.AdcChannel.GO20_N)


        # substract leakage current
        Id -= Is0_n
        Ids[Vg].append(Id)
```

```
Offset Is0_n: 6.103515488575795e-07 A
```

- Include a single plot showing all data on a semilogy plot.

In [20]:

```
# plot
from matplotlib import pyplot as plt
styles = ["o-", "s-", "^-", "*-", "x-"]
for (Vg, current_Ids), style in zip(Ids.items(), styles):
    plt.semilogy(Vds, current_Ids, style, label=f"$Vg={Vg:.2f}$")
plt.ylabel("$I_{ds}[A]$")
plt.xlabel("$V_{ds}[V]$")
plt.legend()
plt.show()
```



In [25]:

```
# if the data looks nice, save it!
# example :
for Vg, current_Ids in Ids.items():
    Lab2_data_nFETVgIds = [Vds,current_Ids]
    # save to csv file
    np.savetxt(f'./data/early/Lab2_data_nFETVgIds_{Vg:.2f}.csv', Lab2_data_nFETVgIds, deli
```

- Can you see how the saturation voltage increases with the gate overdrive $V_G - V_T$ in strong inversion?

> Yes, each line plotted with a higher V_G but constant V_T is clearly higher on the graph.

(b) Compute the Early voltage

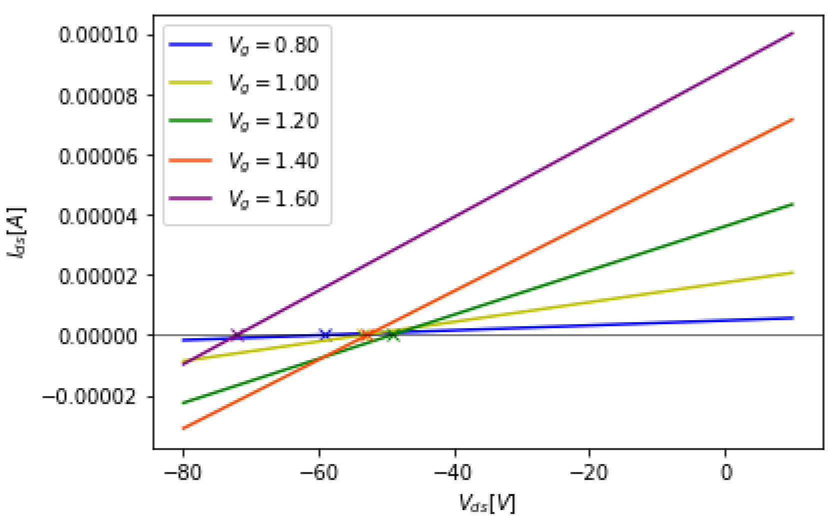
In [18]:

```
Vgs = np.arange(0.8, 1.8, 0.2)
Vds = np.arange(0.0, 1.8, 0.1)
Ids = {}
for Vg in Vgs:
    filename = f'./data/early/Lab2_data_nFETVgIds_{Vg:.2f}.csv'
    Vds, current_Ids = np.loadtxt(filename, delimiter=',')
    Ids[Vg] = current_Ids
```

- Fit a line to the "flat" part of each curve. Select a range of drain voltages to fit the line and use the same range for each curve, because the Early effect is actually curved in reality, and what you are actually seeing is the start of Drain Induced Barrier Lowering (DIBL) or impact ionization.

In [21]:

```
Ids_count = len(Vds)
low_index = int(Ids_count * 0.8)
high_index = Ids_count - 1
ms = [(current_Ids[high_index] - current_Ids[low_index])/(Vds[high_index] - Vds[low_index]
y_0s = [current_Ids[low_index] - m*Vds[low_index] for m, current_Ids in zip(ms, Ids.values

x = np.arange(-80, 10, 0.01)
curves = [x * m + y_0 for m, y_0 in zip(ms, y_0s)]

colors = ["blue", "y", "g", "orangered", "purple"]
V_es = []
for (Vg, curve), color in zip(zip(Vgs, curves), colors):
    V_e = next(Vd for Vd, y in zip(x, curve) if y >= -1e-9)
    V_es.append(-V_e)
    plt.plot(x, curve, label=f"$V_g={Vg:.2f}$", color=color)
    plt.plot(V_e, 0, "x", color=color)
plt.axhline(y=0, color='k', linewidth=0.5)
plt.xlabel("$V_{ds}[V]$")
plt.ylabel("$I_{ds}[A]$")
plt.legend()
plt.show()
```

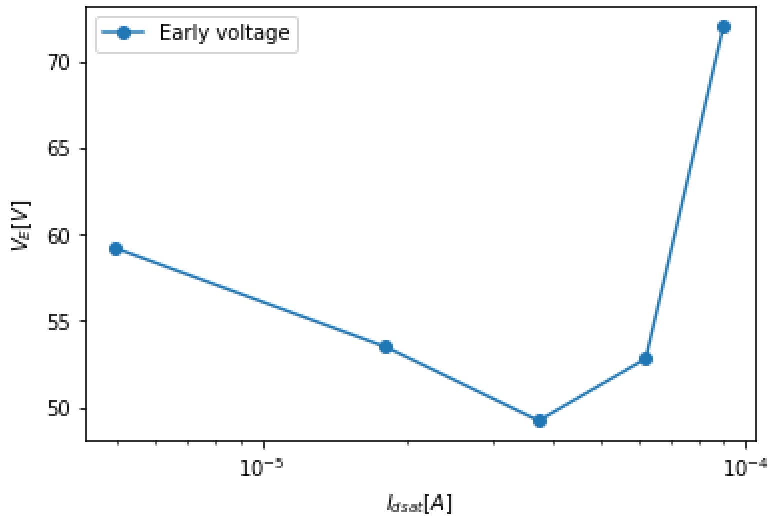


- Plot the Early voltage vs drain current on a semilogx scale.

In [22]:

```
x = np.arange(0, 100, 0.01)
Idsats = [current_Ids[-1] for current_Ids in Ids.values()]

plt.semilogx(Idsats, V_es, "o-", label="Early voltage")
plt.xlabel("$I_{dsat}[A]$")
plt.ylabel("$V_E[V]$")
```

```
plt.legend()
plt.show()
```



- Comment on your results: How constant is the Early voltage with drain current? Speculate on the reasons for your observations.

> Not that constant, but it stays within a single order of magnitude. When the Early voltage goes down, it means that the saturation region is less flat, so the Early effect (or in this case DIBL) is increasing. So our highest Early effect is at about $27\mu\mathrm{A}$ in the graph above. There, the effective length of the channel is the smallest because we have the strongest reverse-bias. Then, it decreases again. So I postulate that an initial increase in V_{OD}, which leads to a higher I_{dsat}, causes the increasingly high shortening of the effective channel length until the V_{OD} is high enough that the depletion layer caused by the reverse-bias is no longer observable since the mobile carriers are in free flow.

8 Congratulations

If you did everything in this lab, you have done a lot! This is probably the most difficult but also one of the most important labs, because practical and intuitive knowledge of transistor characteristics is crucial in understanding and synthesizing new circuits.

9 What we expect

How transistors work above threshold.

What is the linear or triode region and what is the saturation region?

How does the linear region depend on gate and threshold voltage?

What is the *overdrive*?

What is the specific current?

How the Early effect comes about?

Typical values for Early voltage.

How to sketch graphs of transistor current vs gate voltage and drain-source voltage.

How above-threshold transistors go into saturation and why the saturation voltage is equal to the gate overdrive. Can you write the above-threshold current equations?

How does above-threshold current depend on W/L, C_{ox}, and mobility μ?

How do transconductance and drain resistance combine to generate voltage gain? And what is the intrinsic voltage gain of a transistor?

What effect does velocity saturation have on transistor operation, specifically, how does it change the relation between saturation current and gate voltage? What is DIBL (drain induced barrier lowering) and II (impact ionization)?

What is the dominant source of mismatch?

How does transistor mismatch scale with transistor size?

What are typical values of transistor threshold voltage mismatch?