In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
from UliEngineering.EngineerIO import format_value
from si_prefix import si_format
import plecs_helper as helper
%matplotlib
%matplotlib inline
```

```
Using matplotlib backend: TkAgg
```

Lab 2: Ian Eykamp

Feb. 7, 2023

Pre-Lab

In [ ]:
```
# Define parameters

C = 60e-6 # uF
Fs = 50e3 # kHz
Ts = 1 / Fs # s
D = 35 / 100 # percent
Vg = 18 # V
Vtarget = 8 # V
Pout = 12 # W
# P = V ^ 2 / R, R = V ^ 2 / P
Rload = Vtarget ** 2 / Pout # Ohm

# Discontinuous Conduction Mode

M = Vtarget / Vg # Voltage conversion ratio
Lcrit = (1 - M) * Rload * Ts / 2 # Critical inductor value
alpha = (D / M) ** 2 # L / Lcrit # Critical inductor ratio
L = Lcrit * alpha # alpha * Lcrit = L

Ipeak = Vtarget / Rload * (2 / np.sqrt(alpha)) # Peak current, as determined by ind

print(f"Load resistance: {si_format(Rload, precision = 2)}Ohm")
print(f"Critical inductance: {si_format(Lcrit, precision = 2)}H")
print(f"Alpha (critical inductance ratio): {alpha}")
print("------")
print(f"Inductor value for given parameters: {si_format(L, precision = 2)}H")
print(f"Peak current for given inductor at 35% duty cycle: {si_format(Ipeak, precis

# Using same parameters, set D = 50%
# M = D_DCM / sqrt(alpha), Vout = Vg * M
Vout_50 = 0.50 / np.sqrt(alpha) * Vg
# Vout is proportional to D, with constant of proportionality Vg / sqrt(alpha).
print(f"Expected output voltage at 50% duty cycle: {si_format(Vout_50, precision =
Load resistance: 5.33 Ohm
Critical inductance: 29.63 µH
Alpha (critical inductance ratio): 0.62015625
------
Inductor value for given parameters: 18.38 µH
Peak current for given inductor at 35% duty cycle: 3.81 A
Expected output voltage at 50% duty cycle: 11.43 V
```

Lab 2

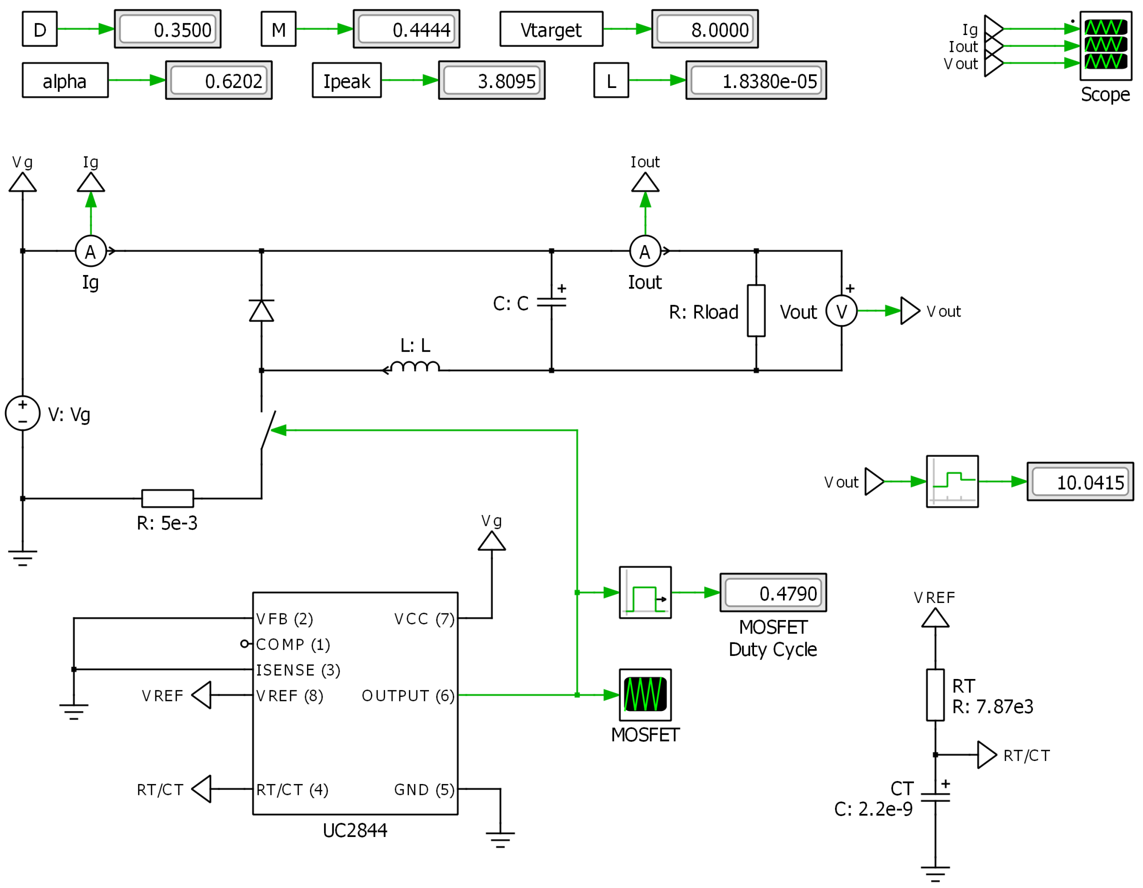
From the UC2844 Datasheet: $F_{oscillation} = \frac{1.72}{R_T \cdot C_T}$. From the schematic, we have $R_T = 7.87k\Omega$, $C_T = 2.2nF$.

For this chip, when the oscillating signal goes high, it activates a flip-flop to toggle the output. Therefore, it takes two oscillator cycles to make the output go high and then low; hence, the oscillator frequency is essentially divided by two.

In [ ]:
```
# Theoretical frequency
RT = 7.87e3 # kOhm
CT = 2.2e-9 # nF
F_osc = 1.72 / (RT * CT) / 2
print(f"Oscillation frequency: {si_format(F_osc, precision = 1)}Hz")

# Measured frequency
T1 = 3.096e-3
T2 = 3.1165e-3
F_measured = 1 / (T2 - T1)
print(f"Measured frequency: {si_format(F_measured, precision = 1)}Hz")
print("Duty cycle: 48%")
```

```
Oscillation frequency: 49.7 kHz
Measured frequency: 48.8 kHz
Duty cycle: 48%
```

Base Schematic



Initialization Script:

```
C = 60e-6; # uF
Fs = 48.8e3; # kHz
Ts = 1 / Fs; # s
D = 35 / 100; # percent
Vg = 18; # V
Vtarget = 8; # V
M = Vtarget / Vg; # Voltage conversion ratio
Pout = 12; # W
Rload = Vtarget ^ 2 / Pout; # Ohms

# Discontinuous Conduction Mode
# L = 18.38e-6; # H

M = Vtarget / Vg; # Voltage conversion ratio
Lcrit = (1 - M) * Rload * Ts / 2; # Critical inductor value
alpha = (D / M) ** 2 # L / Lcrit; # Critical inductor ratio
L = Lcrit * alpha # alpha * Lcrit = L;

Ipeak = Vtarget / Rload * (2 / sqrt(alpha)); # Peak current, as
determined by inductor value and critical inductor ratio
```

In [ ]:
```
# Using same parameters, set D = 48%
Vout_48 = 0.48 / np.sqrt(alpha) * Vg
```

```
# Vout is proportional to D, with constant of proportionality Vg / sqrt(alpha).
print(f"Expected output voltage at 48% duty cycle: {si_format(Vout_48, precision =
```

```
Expected output voltage at 48% duty cycle: 10.97 V
```

Simulation with the UC2844

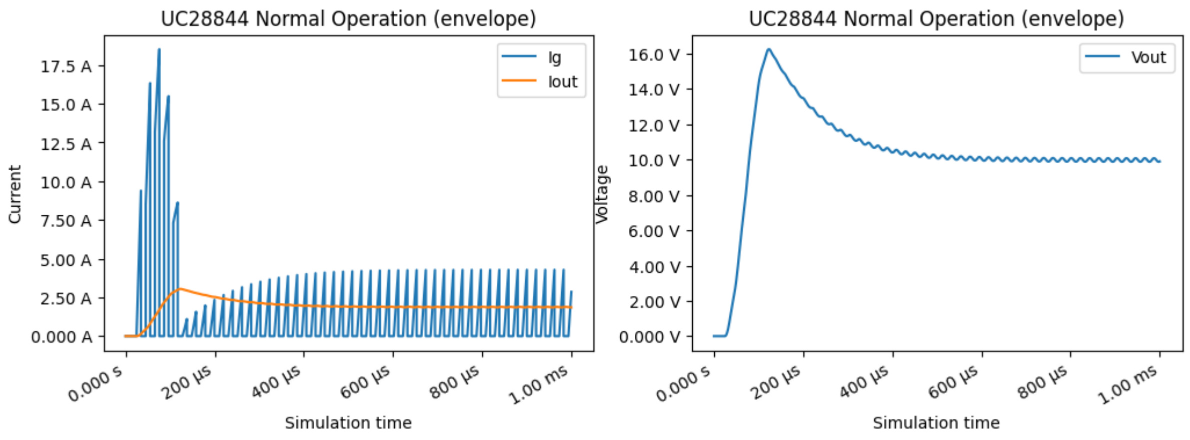
The expected output voltage was 10.97V; however, I observed only 9.97V across the output of the PLECS model. I do not know what accounts for this rather large 1V difference.

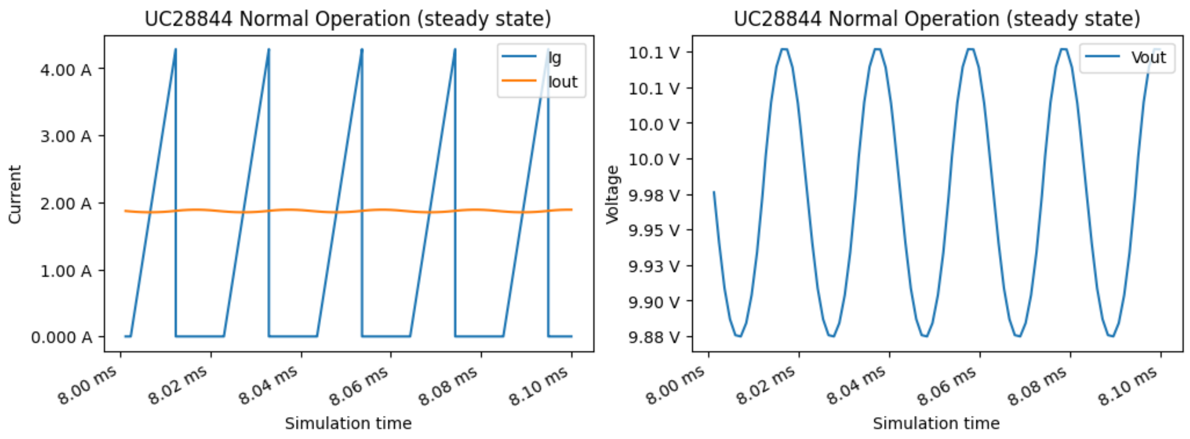
In [ ]:

```
df = pd.read_csv("uc2844_base.csv")
df.rename(mapper = helper.strip_labels, axis = "columns", inplace = True)
df_zoom = df.loc[(df["t"] > 8e-3) & (df["t"] < 8.1e-3)]
df_envelope = df.loc[df["t"] < 1e-3]

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_envelope["t"], df_envelope["Ig"], label = "Ig")
ax1.plot(df_envelope["t"], df_envelope["Iout"], label = "Iout")
helper.axes_labels("Simulation time", "s", "Current", "A", title = "UC28844 Normal
ax1.legend(loc = "upper right")

ax2.plot(df_envelope["t"], df_envelope["Vout"], label = "Vout")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "UC28844 Normal
ax2.legend(loc = "upper right")

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_zoom["t"], df_zoom["Ig"], label = "Ig")
ax1.plot(df_zoom["t"], df_zoom["Iout"], label = "Iout")
helper.axes_labels("Simulation time", "s", "Current", "A", title = "UC28844 Normal
ax1.legend(loc = "upper right")

ax2.plot(df_zoom["t"], df_zoom["Vout"], label = "Vout")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "UC28844 Normal
ax2.legend(loc = "upper right")
```

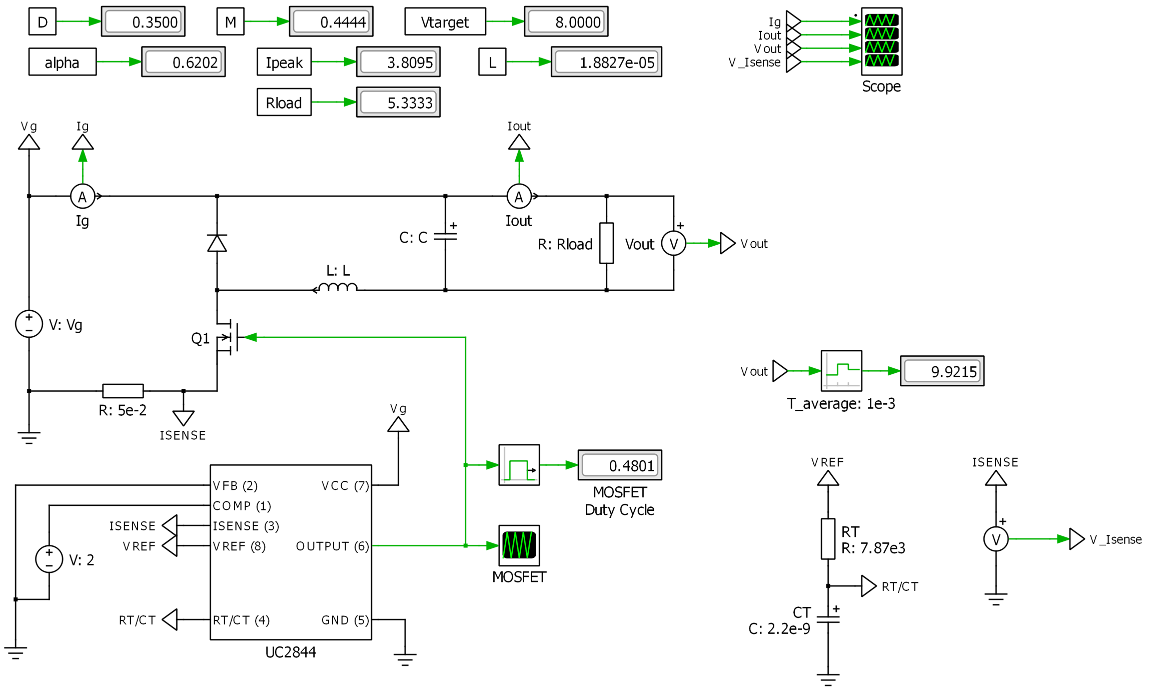
Out[ ]:

```
<matplotlib.legend.Legend at 0x1ef8a2e9400>
```





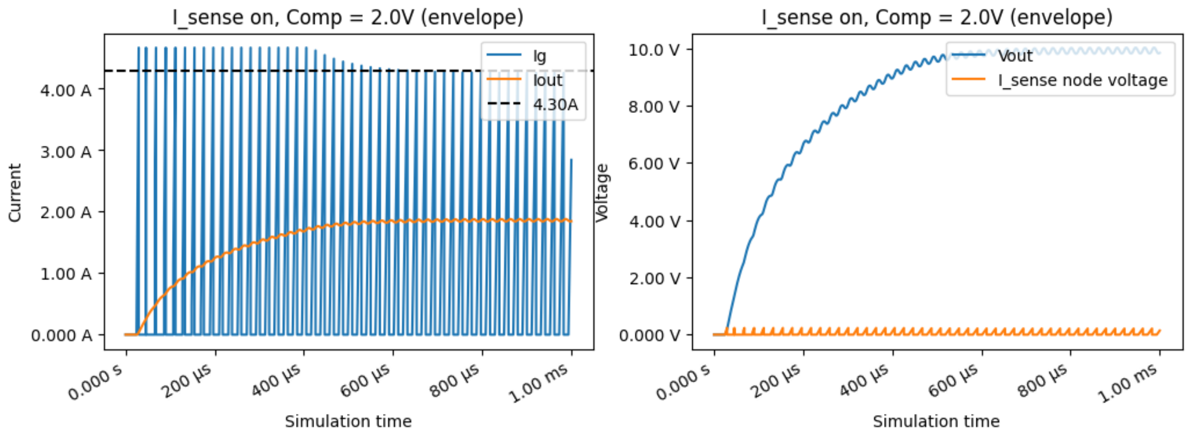
After configuring I_SENSE

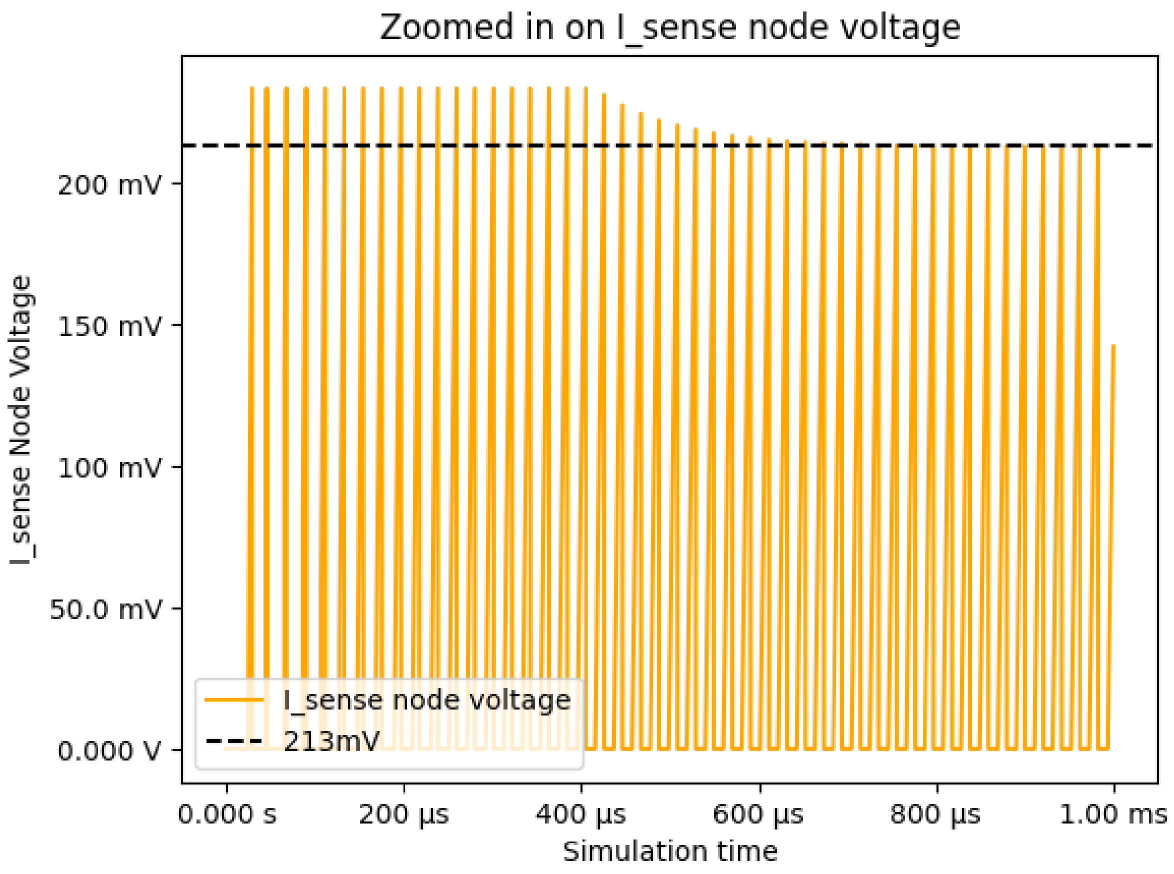


```
In [ ]: df = pd.read_csv("isense_connected.csv")
        df.rename(mapper = helper.strip_labels, axis = "columns", inplace = True)
        df_zoom = df.loc[(df["t"] > 8e-3) & (df["t"] < 8.1e-3)]
        df_envelope = df.loc[df["t"] < 1e-3]

        fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
        fig.autofmt_xdate()
        ax1.plot(df_envelope["t"], df_envelope["Ig"], label = "Ig")
        ax1.plot(df_envelope["t"], df_envelope["Iout"], label = "Iout")
        ax1.axhline(y = 4.26, color = "black", linestyle = "dashed", label = "4.26A")
        helper.axes_labels("Simulation time", "s", "Current", "A", title = "I_sense on, Com
        ax1.legend(loc = "upper right")

        ax2.plot(df_envelope["t"], df_envelope["Vout"], label = "Vout")
        ax2.plot(df_envelope["t"], df_envelope["V_Isense"], label = "I_sense node voltage")
        helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "I_sense on, Com
        ax2.legend(loc = "upper right")
```

```
fig = plt.figure()
plt.plot(df_envelope["t"], df_envelope["V_Isense"], color = "orange", label = "I_se
helper.axes_labels("Simulation time", "s", "I_sense Node Voltage", "V", title = "Zo
plt.axhline(y = 0.213, color = "black", linestyle = "dashed", label = "213mV")
plt.legend(loc = "lower left")
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1ef917457f0>
```





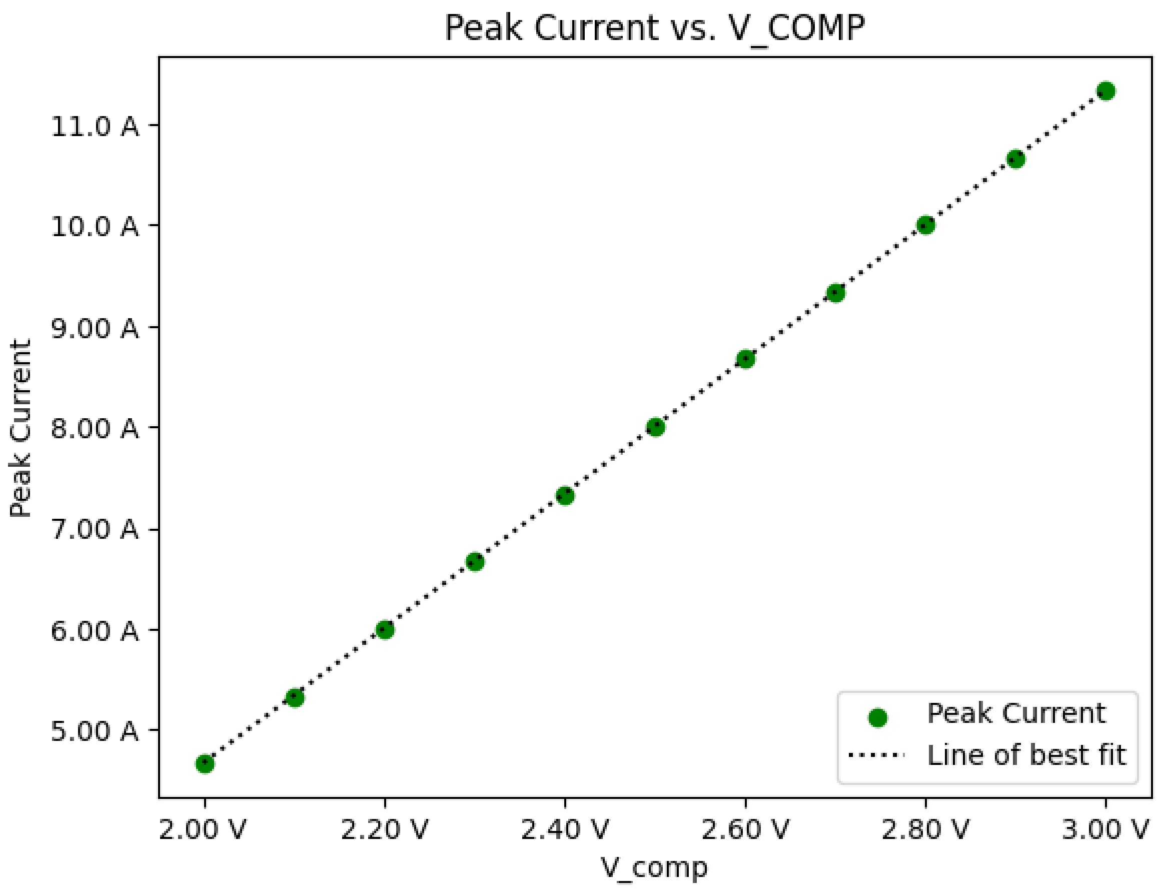
The maximum current is limited to 4.26A by the `I_sense` overcurrent control. This only slightly decreases the output voltage to 9.92V, because the current is reached just before the switch would be turned off anyway.

In [ ]:
```
V_comp = np.arange(2, 3.1, step = 0.1)
# print(V_comp)
```

```
Ig_peak = np.array([4.67, 5.33, 6.00, 6.67, 7.33, 8.00, 8.67, 9.33, 10.00, 10.67, 1
# print(Ig_peak)

fig = plt.figure()
plt.scatter(V_comp, Ig_peak, color = "green", label = "Peak Current")
plt.plot(V_comp, V_comp * 6.67 - 8.67, color = "black", linestyle = "dotted", label
helper.axes_labels("V_comp", "V", "Peak Current", "A", title = "Peak Current vs. V_
plt.legend(loc = "lower right")
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x1ef942ada30>
```



The peak current is linearly related to the V_comp voltage, by the equation
$I_2 - I_1 = a \cdot (V_2 - V_1), (11.33A - 4.67A) = a(3V - 2V)$,

with $a = 6.67A/V$, so

$I_{peak} = 6.67A/V \cdot V_{comp} - 8.67A$.

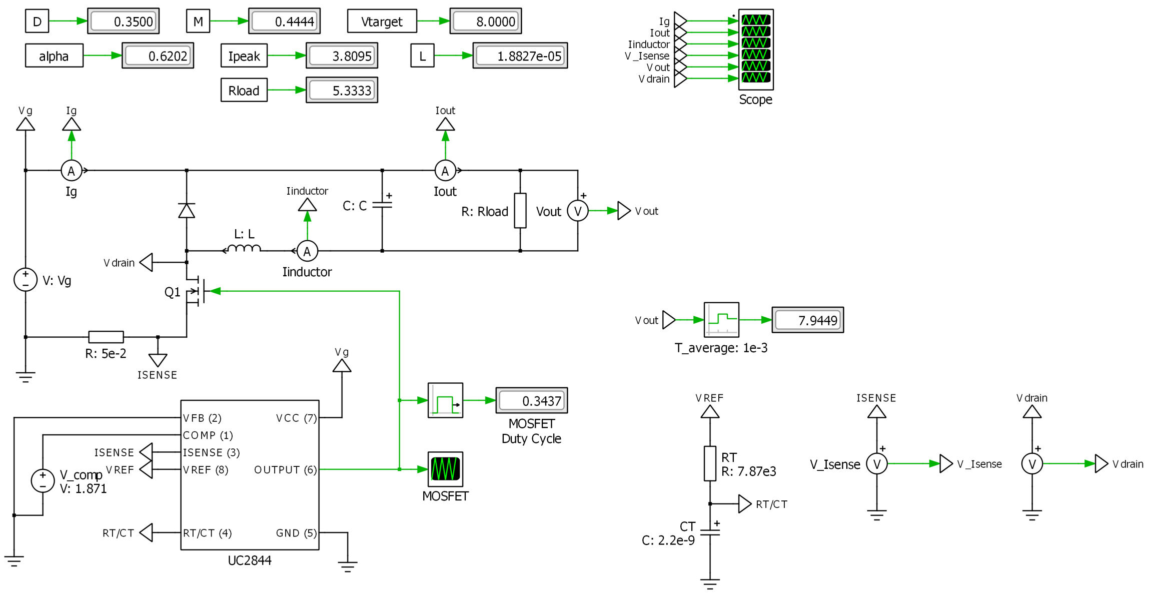
To reach the target voltage of 8V, the peak current should be $I_{peak} = \frac{V_{target}}{R_{load}} \cdot \frac{2}{\sqrt{\alpha}}$.

In [ ]:

```
I_peak_target = Vtarget / Rload * 2 / np.sqrt(alpha)
print(f"Target Peak Current: {si_format(I_peak_target, precision = 3)}A")

V_comp_target = (I_peak_target + 8.67) / 6.67
print(f"Target V_comp: {si_format(V_comp_target, precision = 3)}V")
```

```
Target Peak Current: 3.810 A
Target V_comp: 1.871 V
```

Final Schematic



Initialization Script:

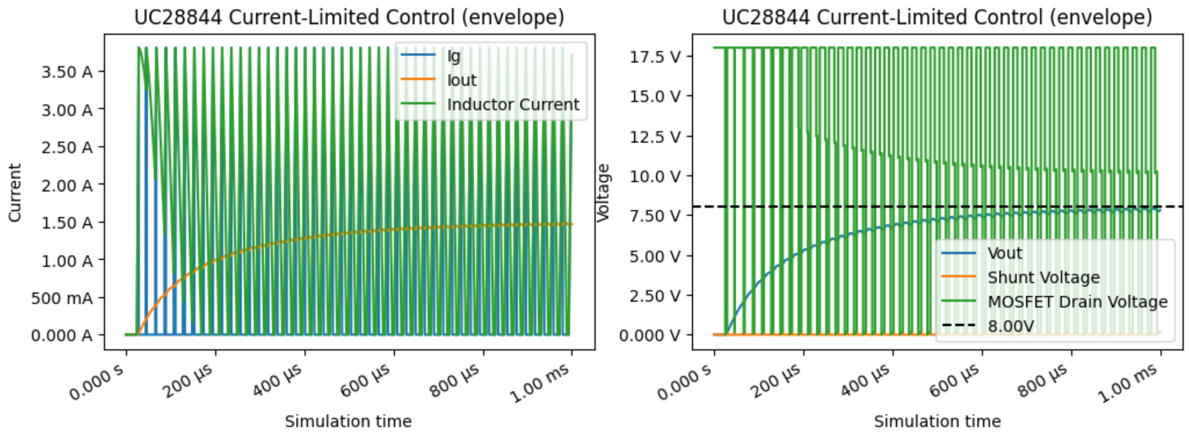
Note that most calculations were done in python in this notebook, and many values are overwritten in the schematic.

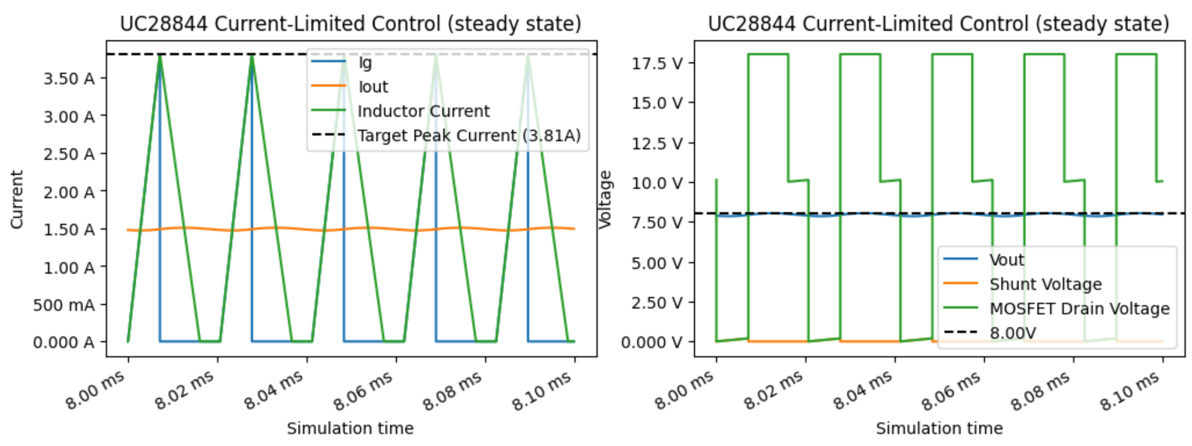
```
C = 60e-6; # uF
Fs = 48.8e3; # kHz
Ts = 1 / Fs; # s
D = 35 / 100; # percent
Vg = 18; # V
Vtarget = 8; # V
M = Vtarget / Vg; # Voltage conversion ratio
Pout = 12; # W
Rload = Vtarget ^ 2 / Pout; # Ohms

# Discontinuous Conduction Mode
# L = 18.38e-6; # H

M = Vtarget / Vg; # Voltage conversion ratio
Lcrit = (1 - M) * Rload * Ts / 2; # Critical inductor value
alpha = (D / M) ** 2 # L / Lcrit; # Critical inductor ratio
L = Lcrit * alpha # alpha * Lcrit = L;

Ipeak = Vtarget / Rload * (2 / sqrt(alpha)); # Peak current, as
determined by inductor value and critical inductor ratio
```

In [ ]:
```
df = pd.read_csv("final_outputs.csv")
df.rename(mapper = helper.strip_labels, axis = "columns", inplace = True)
```

```
df_zoom = df.loc[(df["t"] > 8e-3) & (df["t"] < 8.1e-3)]
df_envelope = df.loc[df["t"] < 1e-3]

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_envelope["t"], df_envelope["Ig"], label = "Ig")
ax1.plot(df_envelope["t"], df_envelope["Iout"], label = "Iout")
ax1.plot(df_envelope["t"], df_envelope["Iinductor"], label = "Inductor Current")
helper.axes_labels("Simulation time", "s", "Current", "A", title = "UC28844 Current
ax1.legend(loc = "upper right")

ax2.plot(df_envelope["t"], df_envelope["Vout"], label = "Vout")
ax2.plot(df_envelope["t"], df_envelope["V_Isense"], label = "Shunt Voltage")
ax2.plot(df_envelope["t"], df_envelope["Vdrain"], label = "MOSFET Drain Voltage")
ax2.axhline(y = 8.00, color = "black", linestyle = "dashed", label = "8.00V")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "UC28844 Current
ax2.legend(loc = "lower right")

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_zoom["t"], df_zoom["Ig"], label = "Ig")
ax1.plot(df_zoom["t"], df_zoom["Iout"], label = "Iout")
ax1.plot(df_zoom["t"], df_zoom["Iinductor"], label = "Inductor Current")
ax1.axhline(y = 3.810, color = "black", linestyle = "dashed", label = "Target Peak
helper.axes_labels("Simulation time", "s", "Current", "A", title = "UC28844 Current
ax1.legend(loc = "upper right")

ax2.plot(df_zoom["t"], df_zoom["Vout"], label = "Vout")
ax2.plot(df_zoom["t"], df_zoom["V_Isense"], label = "Shunt Voltage")
ax2.plot(df_zoom["t"], df_zoom["Vdrain"], label = "MOSFET Drain Voltage")
ax2.axhline(y = 8.00, color = "black", linestyle = "dashed", label = "8.00V")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "UC28844 Current
ax2.legend(loc = "lower right")
```

```
Out[ ]: <matplotlib.legend.Legend at 0x1ef9876cfa0>
```





As desired, the output voltage approaches a maximum of 8.00V (top right graph) when the input current `Ig` is capped at 3.81A by the `I_SENSE` - `V_COMP` control. The output current also approaches a consistent 1.5A, for an output power of 12W, as expected. The shunt voltage is directly related to the input current `Ig` by V = IR. Because R is very small (0.05 Ohm), the shunt voltage reaches a maximum of 0.2V; it is hard to see on the graph, but it follows the same discontinuous sawtooth pattern as `Ig`. There is very small ripple on the output voltage, on the order of 0.2V. The switching frequency is 48.8kHz, as observed before.

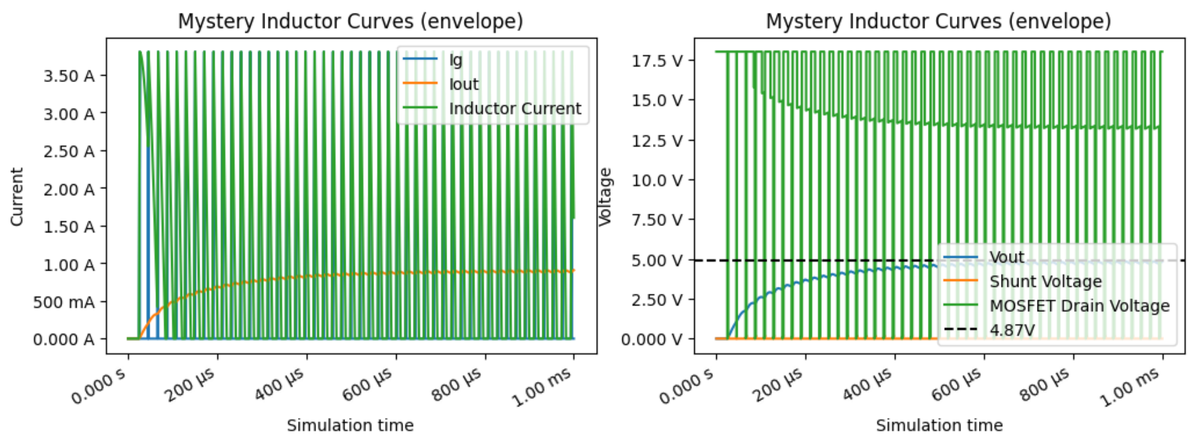
The MOSFET drain voltage has three regions: The drain voltage is nearly zero when the MOSFET is conducting, because it can be modeled as a closed switch, tied through the shunt resistor to ground. In this case, the inductor is charging (increasing its current linearly). When the current reaches the cutoff threshold, the MOSFET is turned off and the inductor starts to discharge (decreasing its current). In this case, the drain voltage is at its greatest, very close to `Vg`, and the diode is conducting the inductor's discharge to maintain the load current. At some point, the inductor current reaches zero and is not switched on again until the start of the next cycle. At this point, by $V_L = L\frac{dI}{dt}$, the voltage drop across the inductor is zero, and the load voltage is maintained only by the capacitor (introducing only a small amount of ripple because the capacitor value is so large, at 60uF). Now the drain voltage is equal to the input voltage minus the load voltage being thus maintained. The MOSFET is still switched off, and the diode does not conduct any current, because `Rload` and the capacitor form a loop. The cycle then restarts when the MOSFET is turned on at the start of the next switching period.

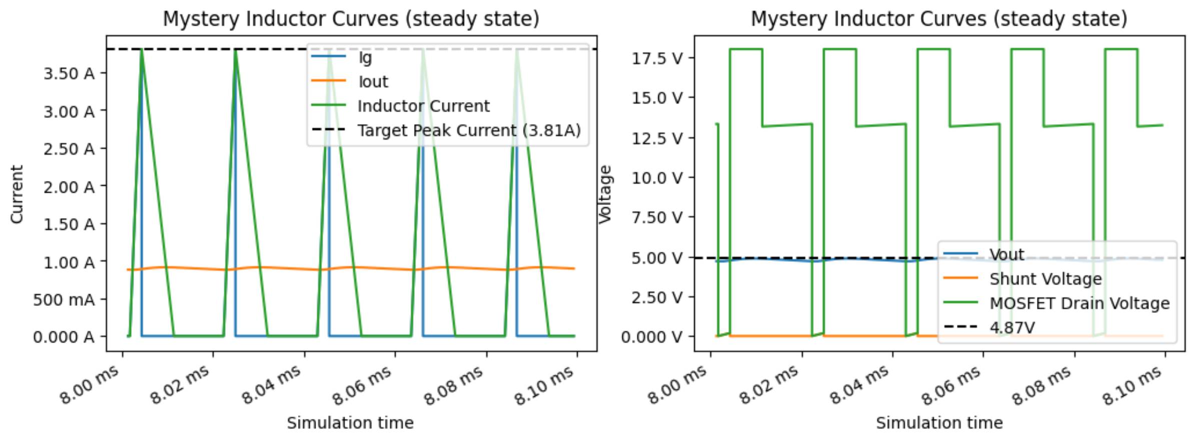
Task 3: Mystery Inductor

In [ ]:
```
df = pd.read_csv("mystery_inductor.csv")
df.rename(mapper = helper.strip_labels, axis = "columns", inplace = True)
df_zoom = df.loc[(df["t"] > 8e-3) & (df["t"] < 8.1e-3)]
df_envelope = df.loc[df["t"] < 1e-3]

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_envelope["t"], df_envelope["Ig"], label = "Ig")
ax1.plot(df_envelope["t"], df_envelope["Iout"], label = "Iout")
```

```
ax1.plot(df_envelope["t"], df_envelope["Iinductor"], label = "Inductor Current")
helper.axes_labels("Simulation time", "s", "Current", "A", title = "Mystery Inducto
ax1.legend(loc = "upper right")

ax2.plot(df_envelope["t"], df_envelope["Vout"], label = "Vout")
ax2.plot(df_envelope["t"], df_envelope["V_Isense"], label = "Shunt Voltage")
ax2.plot(df_envelope["t"], df_envelope["Vdrain"], label = "MOSFET Drain Voltage")
ax2.axhline(y = 4.87, color = "black", linestyle = "dashed", label = "4.87V")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "Mystery Inducto
ax2.legend(loc = "lower right")

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = True, sharey = False,
fig.autofmt_xdate()
ax1.plot(df_zoom["t"], df_zoom["Ig"], label = "Ig")
ax1.plot(df_zoom["t"], df_zoom["Iout"], label = "Iout")
ax1.plot(df_zoom["t"], df_zoom["Iinductor"], label = "Inductor Current")
ax1.axhline(y = 3.810, color = "black", linestyle = "dashed", label = "Target Peak
helper.axes_labels("Simulation time", "s", "Current", "A", title = "Mystery Inducto
ax1.legend(loc = "upper right")

ax2.plot(df_zoom["t"], df_zoom["Vout"], label = "Vout")
ax2.plot(df_zoom["t"], df_zoom["V_Isense"], label = "Shunt Voltage")
ax2.plot(df_zoom["t"], df_zoom["Vdrain"], label = "MOSFET Drain Voltage")
ax2.axhline(y = 4.87, color = "black", linestyle = "dashed", label = "4.87V")
helper.axes_labels("Simulation time", "s", "Voltage", "V", title = "Mystery Inducto
ax2.legend(loc = "lower right")
```

```
Out[ ]: <matplotlib.legend.Legend at 0x1ef98675a00>
```





The inductor value can be interpreted from the current and voltage curves during discharging. The inductance L is given by $L = V_L/(\frac{dI}{dt})$. During discharging, the MOSFET drain voltage is close to `Vg = 18V`, so the voltage drop across the load resistor (4.87V) is equal in magnitude to the voltage drop across the inductor.

The current through the inductor drops from 3.81A to 0A in the span of 7us. Therefore, the inductance value is $L_{mystery} = 4.87V/(3.81A/7\mu s)$

```
In [ ]: discharge_time = 0.003951 - 0.003944 # seconds
        print(f"Discharge Time: {si_format(discharge_time, precision = 2)}s")

        L_mystery_1 = 4.87 / (I_peak_target / discharge_time)
        print(f"Mystery Inductance Value: {si_format(L_mystery_1, precision = 2)}H")
```

```
Discharge Time: 7.00 µs
Mystery Inductance Value: 8.95 µH
```

The inductor value can also be determined by the output voltage by combining a wide set of equations used earlier.

$I_{peak} = \frac{V_{target}}{R_{load}} \cdot \frac{2}{\sqrt{\alpha}} = 3.81A$ in this configuration, where V_{target} is the corresponding output voltage obtained for the given peak current.

This yields $\alpha = (\frac{V_{target}}{R_{load}} \cdot \frac{2}{I_{peak}})^2$.

By definition, $\alpha = \frac{L}{L_{critical}}$, so $L_{mystery} = (\frac{V_{target}}{R_{load}} \cdot \frac{2}{I_{peak}})^2 \cdot L_{critical}$.

Meanwhile, $L_{critical}$ is defined as $L_{critical} = (1 - M) \cdot R_{load} \cdot \frac{T_s}{2}$, where $M = \frac{V_{out}}{V_g} = \frac{4.87V}{18V}$ and $T_s = \frac{1}{F_s} = \frac{1}{48.8kHz}$.

These equations combine to yield $L_{mystery} = (\frac{V_{target}}{R_{load}} \cdot \frac{2}{I_{peak}})^2 \cdot (1 - \frac{V_{out}}{V_g}) \cdot R_{load} \cdot \frac{1}{2 \cdot F_s}$

```
In [ ]: L_mystery_2 = (4.87 / Rload * 2 / I_peak_target) ** 2 * (1 - 4.87 / Vg) * Rload * 1
        print(f"Mystery Inductance Value: {si_format(L_mystery_2, precision = 2)}H")
```

```
Mystery Inductance Value: 9.16 µH
```

These two equations yield approximately the same value for the mystery inductance of $L_{mystery} \approx 9.06\mu H \pm 0.11\mu H$.