Lab 8: Silicon Synaptic Circuits

Team member 1: Jan Hohenheim

Team member 2: Maxim Gärtner

Date:

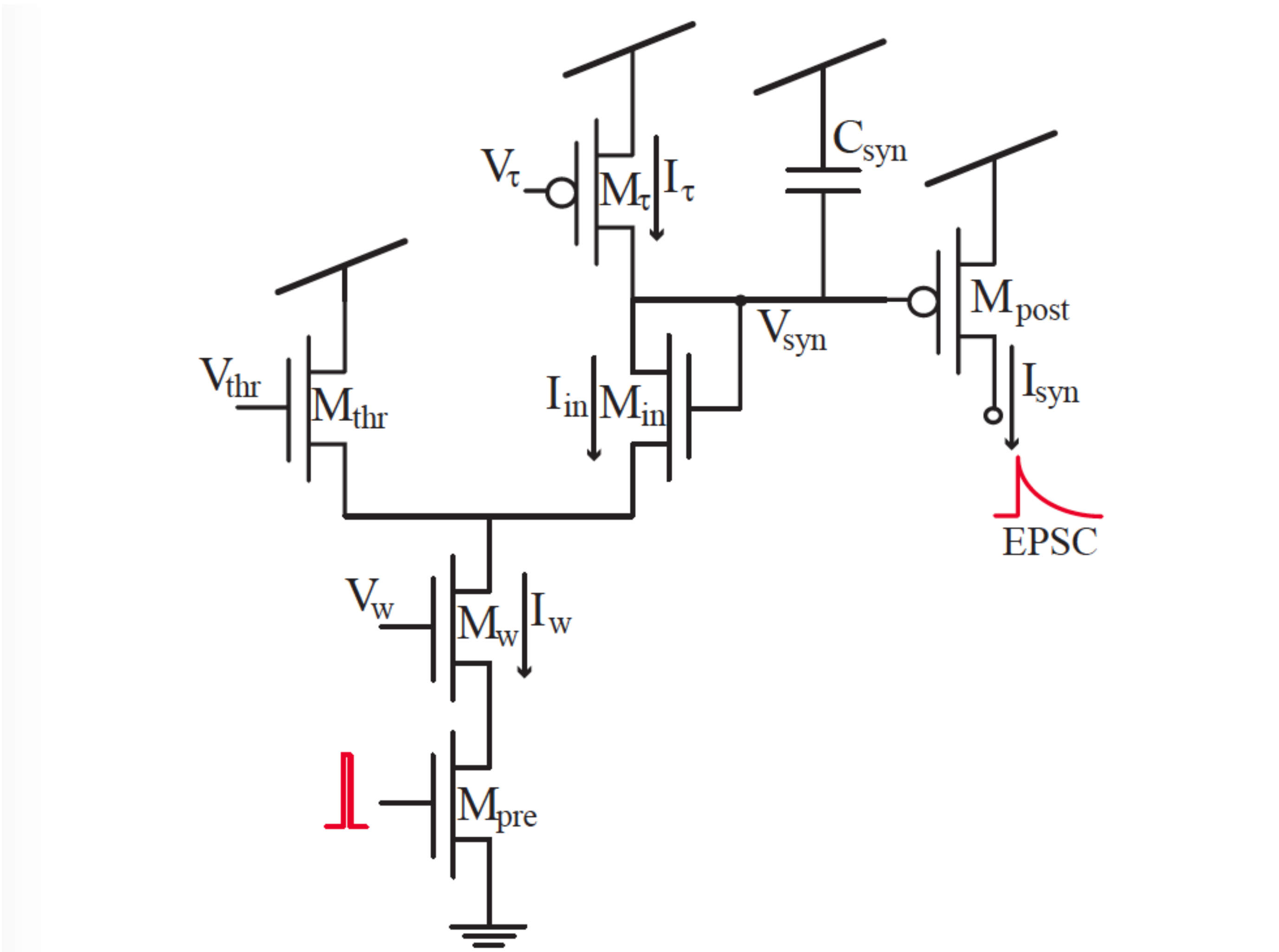
This week, we will see how synaptic circuits generate currents when stimulated by voltage pulses. Specifically we will measure the response of the synapse to a single pulse, and to a sequence of spikes.

The objectives of this lab are to:

- Analyze log-domain synapse circuits.
- Measure the response properties of the diff-pair integrator (DPI) synapse and of the dual diff-pair integrator (DDI) synapse.

1. Prelab

A Differential Pair Integrator circuit



(1) Write the equations characterizing $I_w, I_{thr}, I_{in}, I_\tau, I_{syn}, I_C$ assuming all corresponding FETs are in saturation and operate in weak-inversion.

> - $I_w = I_0 e^{\frac{\kappa V_w}{U_T}}$
> - $I_{thr} = I_0 e^{\frac{\kappa V_{thr} - V_{dd}}{U_T}}$
> - $I_{in} = I_0 e^{\frac{\kappa V_{syn} - V_{syn}}{U_T}} = I_w \frac{e^{\frac{\kappa V_{syn}}{U_T}}}{e^{\frac{\kappa V_{syn}}{U_T}} + e^{\frac{\kappa V_{thr}}{U_T}}}$
> - $I_\tau = I_0 e^{\frac{\kappa (V_{dd} - V_\tau)}{U_T}}$
> - $I_{syn} = I_0 e^{\frac{\kappa (V_{dd} - V_{syn})}{U_T}}$
> - $I_C = C \frac{d}{dt}(V_{dd} - V_{syn})$

(2) What is the time constant of the circuit?

> $\tau = \frac{C U_T}{\kappa I_\tau}$

(3) Derive the circuit's response to a step input assuming $I_w(t < 0) = 0, I_w(t > 0) \gg I_\tau$.

> - $I_w \ll I_\tau \Rightarrow \tau \frac{d}{dt} I_{syn} + I_{syn} = 0 \Rightarrow \frac{d}{dt} I_{syn} = -\frac{I_{syn}}{\tau}$
> - $I_w \gg I_\tau \Rightarrow \tau \frac{d}{dt} I_{syn} + I_{syn} = \frac{I_w I_{thr}}{I_\tau} \Rightarrow \frac{d}{dt} I_{syn} = \frac{I_w I_{thr} - I_{syn} I_\tau}{\tau I_\tau}$

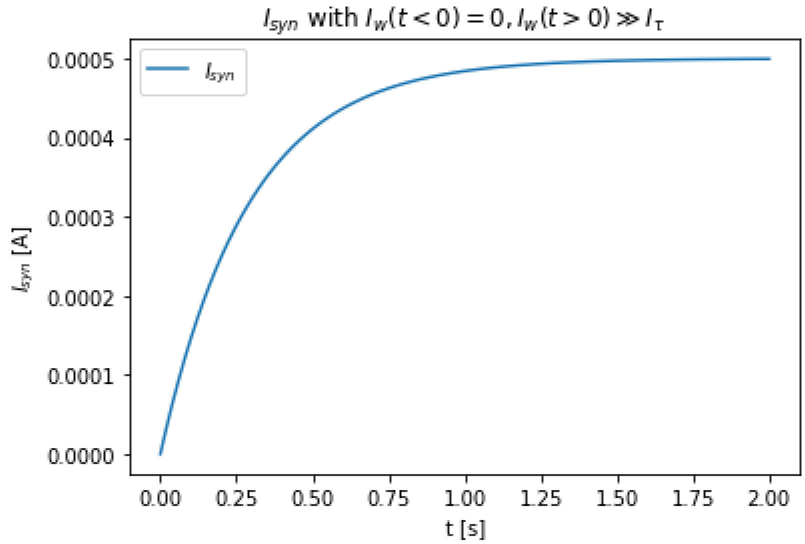
In [6]:

```
import numpy as np
import matplotlib.pyplot as plt

def get_next_I_syn(I_syn, tau, I_tau, I_thr, I_w, dt):
    return I_syn + (I_w*I_thr - I_syn*I_tau)/(tau * I_tau)*dt

tau = 0.3
I_tau = 5e-9
I_w = 5e-7
I_thr = 5e-6
x = np.linspace(0, 2, 100)
dt = x[1] - x[0]
y = [0]
for _ in range(len(x[1:])):
    I_syn = get_next_I_syn(y[-1], tau, I_tau, I_thr, I_w, dt)
    y.append(I_syn)

plt.plot(x, y, label="$I_{syn}$")
plt.title(r"$I_{syn}$ with $I_{w}(t < 0) = 0, I_{w}(t > 0) \gg I_{\tau}$")
plt.ylabel("$I_{syn}$ [A]")
plt.xlabel("t [s]")
plt.legend()
plt.show()
```



(4) Derive the circuit's response to a step input assuming $I_w(t<0) \gg I_\tau, I_w(t>0) = 0$.

> - $I_w \ll I_\tau \Rightarrow \tau \frac{d}{dt} I_{syn} + I_{syn} = 0 \Rightarrow \frac{d}{dt} I_{syn} = -\frac{I_{syn}}{\tau}$
> - $I_w \gg I_\tau \Rightarrow \tau \frac{d}{dt} I_{syn} + I_{syn} = \frac{I_w I_{thr}}{I_\tau} \Rightarrow \frac{d}{dt} I_{syn} = \frac{I_w I_{thr} - I_{syn} I_\tau}{\tau I_\tau}$

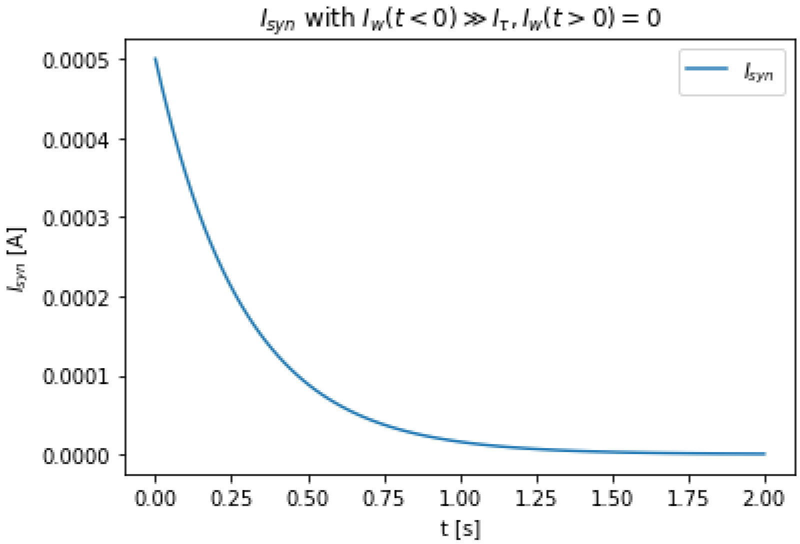
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt

def get_next_I_syn(I_syn, tau, I_tau, I_thr, I_w, dt):
    return I_syn + (I_w*I_thr - I_syn*I_tau)/(tau * I_tau)*dt

tau = 0.3
I_tau = 5e-7
I_w = 5e-9
I_thr = 5e-6
x = np.linspace(0, 2, 100)
dt = x[1] - x[0]
y = [5e-4]
for _ in range(len(x[1:])):
    I_syn = get_next_I_syn(y[-1], tau, I_tau, I_thr, I_w, dt)
    y.append(I_syn)

plt.plot(x, y, label="$I_{syn}$")
plt.title(r"$I_{syn}$ with $I_{w}(t < 0) \gg I_{\tau}, I_{w}(t > 0) = 0$")
plt.ylabel("$I_{syn}$ [A]")
plt.xlabel("t [s]")
plt.legend()
plt.show()
```



(5) Suppose we stimulate the circuit with a regular spike train of frequency f (high enough). What happens to I_{syn} in steady-state (average value)?

> $\tau \frac{d}{dt} I_{syn} + I_{syn} = \frac{I_w I_{thr}}{I_\tau}$
> Steady-state $\Rightarrow \frac{d}{dt} I_{syn} = 0 \frac{A}{s}$
> $\Rightarrow I_{syn} = \frac{I_w I_{thr}}{I_\tau}$

(6) In what conditions (tau and thr) is the step response dependent only on I_w?

> Per the formula above, when $I_{thr} = I_\tau$

2 Setup

2.1 Connect the device

In [2]:

```
# import the necessary libraries
import pyplane
import time
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate
```

In [3]:

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

In [4]:

```
p.get_firmware_version()
```

Out[4]:

```
(1, 8, 4)
```

In [5]:

```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
time.sleep(0.5)
# NOTE: You must send this request events every time you do a reset operetion, otherwise t
# Because the class chip need to do handshake to get the communication correct.
p.request_events(1)
```

In [6]:

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

Out[6]:

```
1.530761721824092e-07
```

In [7]:

```
 # If any of the above steps fail, delete the object, close and halt, stop the server and
# please also say your board number: ttyACMx
# del p
```

2.2 Chip configuration

- To measure DPI synapse:

In [8]:

```
p.send_coach_events([pyplane.Coach.generate_aerc_event(
    pyplane.pyplane.Coach.CurrentOutputSelect.SelectLine5,
    pyplane.Coach.VoltageOutputSelect.SelectLine2,
    pyplane.Coach.VoltageInputSelect.NoneSelected,
    pyplane.Coach.SynapseSelect.DPI,0)])
```

2.3 C2F

- To set up the C2F circuit:

In [9]:

```
# setup C2F
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_HYS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_BIAS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_PWLK_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_REF_L, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_REF_H, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])
```

```
# setup output rail-to-rail buffer
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.RR_BIAS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])
```

2.4 BiasGen

In a simplified form, the output of a branch of the BiasGen will be the gate voltage V_b for the bias current I_b, and if the current mirror has a ratio of w and the bias transistor operates in subthreshold-saturation:

$$I_b = w \frac{BG_{fine}}{256} I_{BG_{master}} \tag{1}$$

Where $I_{BG_{master}}$ is the `BiasGenMasterCurrent` $\in \{60 \text{ pA}, 460 \text{ pA}, 3.8 \text{ nA}, 30 \text{ nA}, 240 \text{ nA}\}$, BG_{fine} is the integer fine value $\in [0, 256)$

To set a bias, use the function similar to the following:

```
p.send_coach_event(pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.BIAS_NAME, \
    pyplane.Coach.BiasType.BIAS_TYPE, \
    pyplane.Coach.BiasGenMasterCurrent.MASTER_CURRENT, FINE_VALUE))
```

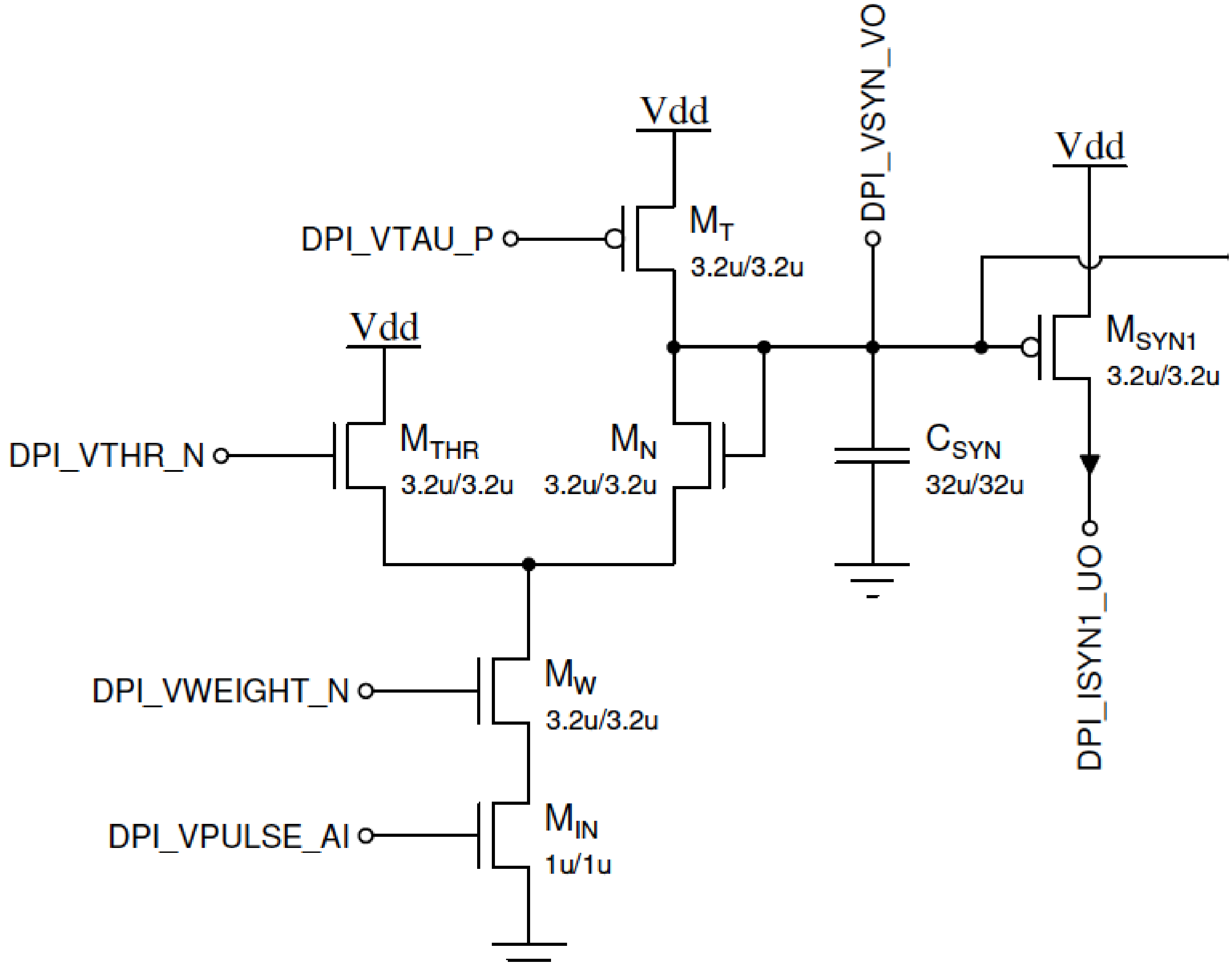
You may have noticed that there are some biases that are not used to directly generate a current, but rather what matters is the voltage, e.g. V_{gain}, V_{ex} and V_{inh} in our HWTA circuit. Even though they may have a `BIAS_NAME` ending with `_N` or `_P` it only indicates that they are connected to the gate of an N- or a P-FET, but the `BIAS_TYPE` parameter can be both `_N` or `_P`. For example, setting a `_N` bias to `BIAS_TYPE = P` will only make this voltage very close to GND, which *is* sometimes the designed use case.

2.5 Pulse extender circuit

In case you didn't look into the last problem in prelab, the pulse extender circuit basically defines the pulse width, which is inversely proportional to the parameter `PEX_VTAU_N`.

3 DPI synapse

The **DPI synapse** receives a voltage pulse train, V_{pulse}, as input and outputs a corresponding synaptic current, I_{syn}. Additionally, the synaptic voltage, V_{syn}, is provided. Bias parameters V_{weight} & V_{tau} affect the amplitude and decay of the response, while V_{thr} acts as an additional weight bias. C_{syn} sizing was chosen for a capacitance of 2pF.



Pin map

V_{syn} **= adc[14]**

I_{syn} **= c2f[9]**

The task of this exercise it to tune the parameters and observe the behavior of the DPI synapse.

3.1 Basic impulse response

- **Set parameters**

In [10]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])
```

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

- **Data acquisition**

In [10]:
```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

- **Plot the data**

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

t,vsyn,isyn = np.loadtxt('data/data_ex_3_1.csv',delimiter=',')
plt.plot(t,vsyn,'-')

plt.xlabel('t [s]')
plt.ylabel('$V_{syn}$ [V]')
plt.legend(['$V_{syn}$'],bbox_to_anchor=(1.05, 1),loc='upper left')
plt.title('Fig. 1: Measured values of $V_{syn}$ as a function of time')

plt.grid()
plt.show()

plt.plot(t,isyn,'-')

plt.xlabel('t [s]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F$(I_{syn})$'],bbox_to_anchor=(1.05, 1),loc='upper left')
plt.title('Fig. 2: Measured C2F values of $I_{syn}$ as a function of time')

plt.grid()
plt.show()
```

Fig. 1: Measured values of V_{syn} as a function of time

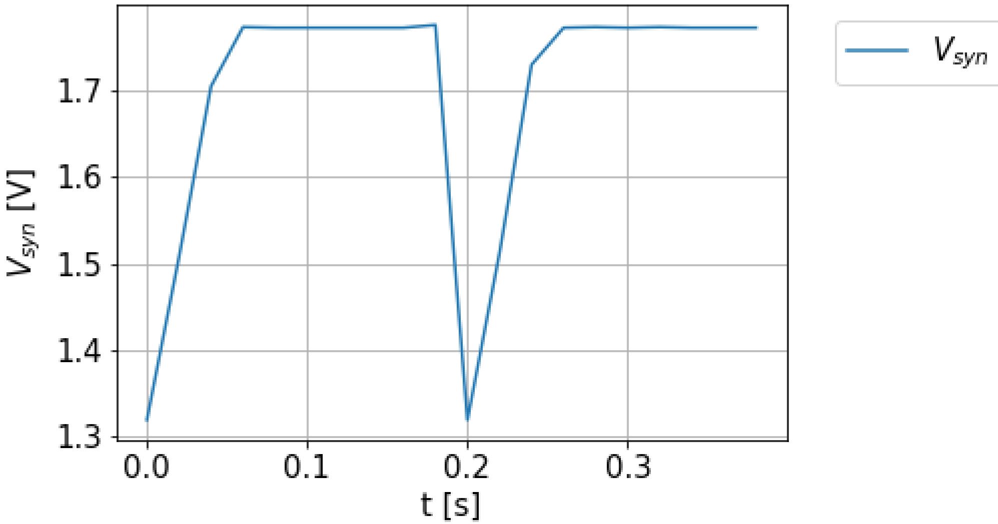
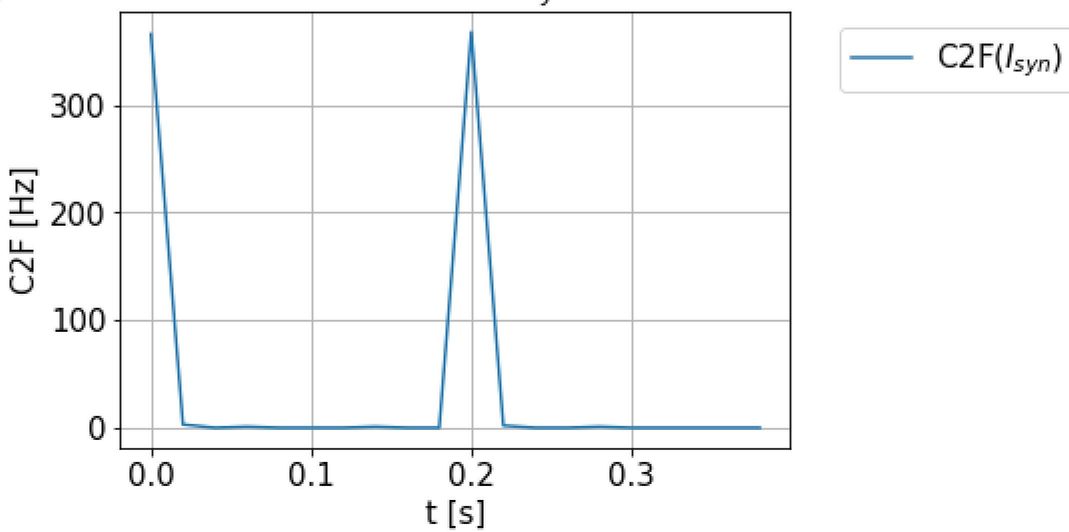


Fig. 2: Measured C2F values of I_{syn} as a function of time



- **Save the data**

In [11]:
```
np.savetxt('data/data_ex_3_1.csv',[t,vsyn,isyn] , delimiter=',')
```

3.2 Different I_{weight}

Repeat 3.1 with a smaller and a larger I_{weight}, compare the three curves in the same plot.

- **Set smaller bias**

In [11]:
```
## REMINDER , RESET ALL PARAMETERS AS 3.1
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
```

```
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 50)]) #change weight

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

- **Data acquisition**

In [12]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

- **Save data**

In [13]:

```
np.savetxt('data/data_ex_3_2_smaller.csv',[t,vsyn,isyn] , delimiter=',')
```

- **Set larger bias**

In [31]:

```
#Insert a bigger I weight
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 150)]) #change weight

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

- **Data acquisition**

In [32]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
```

```
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

- **Save data**

In [33]:

```
np.savetxt('data/data_ex_3_2_bigger.csv',[t,vsyn,isyn] , delimiter=',')
```

- **Plot**

In [34]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})


t,vsyn_smaller,isyn_smaller = np.loadtxt('data/data_ex_3_2_smaller.csv',delimiter=',')
_,vsyn_normal,isyn_normal = np.loadtxt('data/data_ex_3_1.csv',delimiter=',')
_,vsyn_bigger,isyn_bigger = np.loadtxt('data/data_ex_3_2_bigger.csv',delimiter=',')

plt.plot(t,vsyn_smaller,t,vsyn_normal,t,vsyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('$V_{syn}$ [V]')
plt.legend(['$V_{syn}$ - Smaller $I_w$','$V_{syn}$ - Normal $I_w$','$V_{syn}$ - Larger $I_
plt.title('Fig. 3: Measured values of $V_{syn}$ as function of time for varying $I_{w}$')

plt.grid()
plt.show()

plt.plot(t[1:],isyn_smaller[1:],t,isyn_normal,t,isyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F$(I_{syn})$ - Smaller $I_w$','C2F$(I_{syn})$ - Normal $I_w$','C2F$(I_{syn}
plt.title('Fig. 4: Measured values of $I_{syn}$ as function of time for varying $I_{w}$')

plt.grid()
plt.show()
```

Fig. 3: Measured values of V_{syn} as function of time for varying I_w

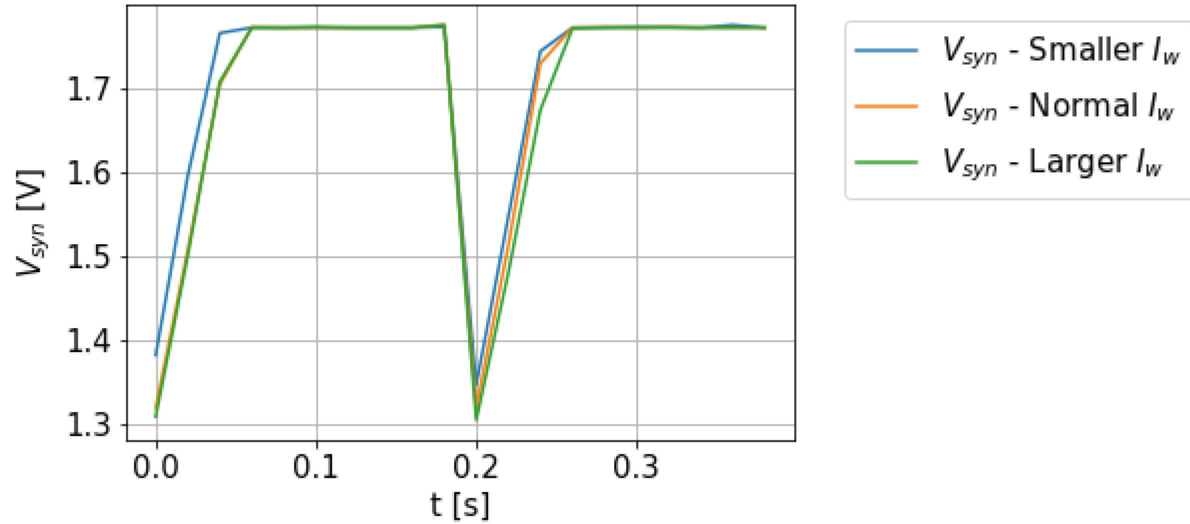
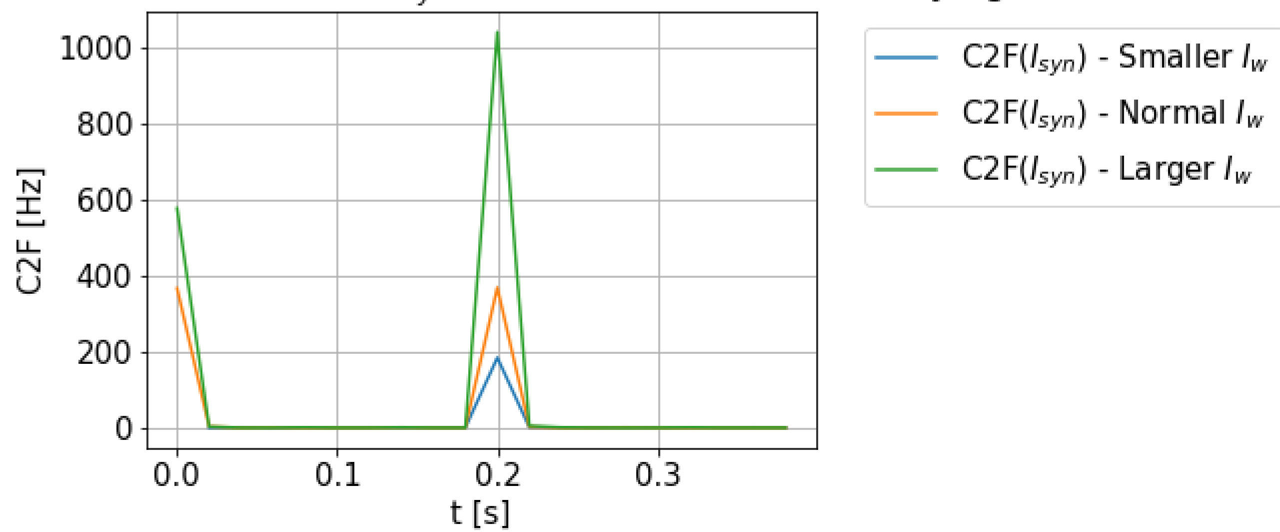


Fig. 4: Measured values of I_{syn} as function of time for varying I_w



3.3 Different I_{tau}

Repeat 3.1 with a smaller and a larger I_{tau}, compare the three curves in the same plot.

In [42]:

```
## REMINDER , RESET ALL PARAMETERS AS 3.1
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])    #change tau

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

In [43]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [44]:

```
np.savetxt('data/data_ex_3_3_smaller.csv',[t,vsyn,isyn] , delimiter=',')
```

In [45]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 40)])     #change tau

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

In [46]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [47]:

```
np.savetxt('data/data_ex_3_3_bigger.csv',[t,vsyn,isyn] , delimiter=',')
```

In [48]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})


t,vsyn_smaller,isyn_smaller = np.loadtxt('data/data_ex_3_3_smaller.csv',delimiter=',')
_,vsyn_normal,isyn_normal = np.loadtxt('data/data_ex_3_1.csv',delimiter=',')
```

```
_,vsyn_bigger,isyn_bigger = np.loadtxt('data/data_ex_3_3_bigger.csv',delimiter=',')

plt.plot(t,vsyn_smaller,t,vsyn_normal,t,vsyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('$V_{syn}$ [V]')
plt.legend(['$V_{syn}$ - Smaller $I_{τ}$','$V_{syn}$ - Normal $I_{τ}$','$V_{syn}$ - Larger
plt.title('Fig. 5: Measured values of $V_{syn}$ as function of time for varying $I_{τ}$')

plt.grid()
plt.show()

plt.plot(t,isyn_smaller,t,isyn_normal,t,isyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F$(I_{syn})$ - Smaller $I_{τ}$','C2F$(I_{syn})$ - Normal $I_{τ}$','C2F$(I_{
plt.title('Fig. 6: Measured values of $I_{syn}$ as function of time for varying $I_{τ}$')

plt.grid()
plt.show()
```

Fig. 5: Measured values of V_{syn} as function of time for varying I_{τ}

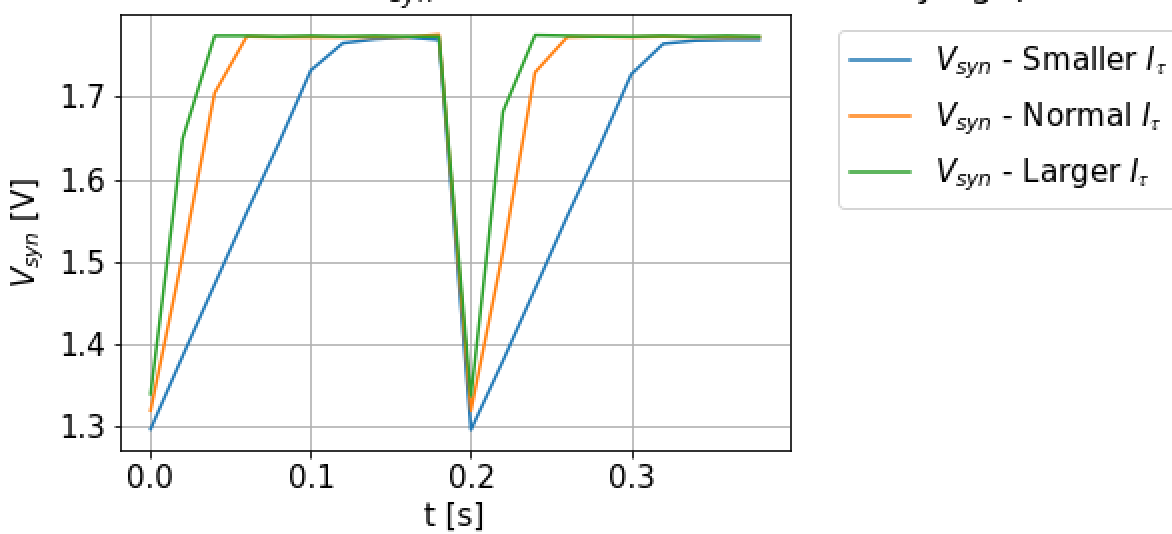
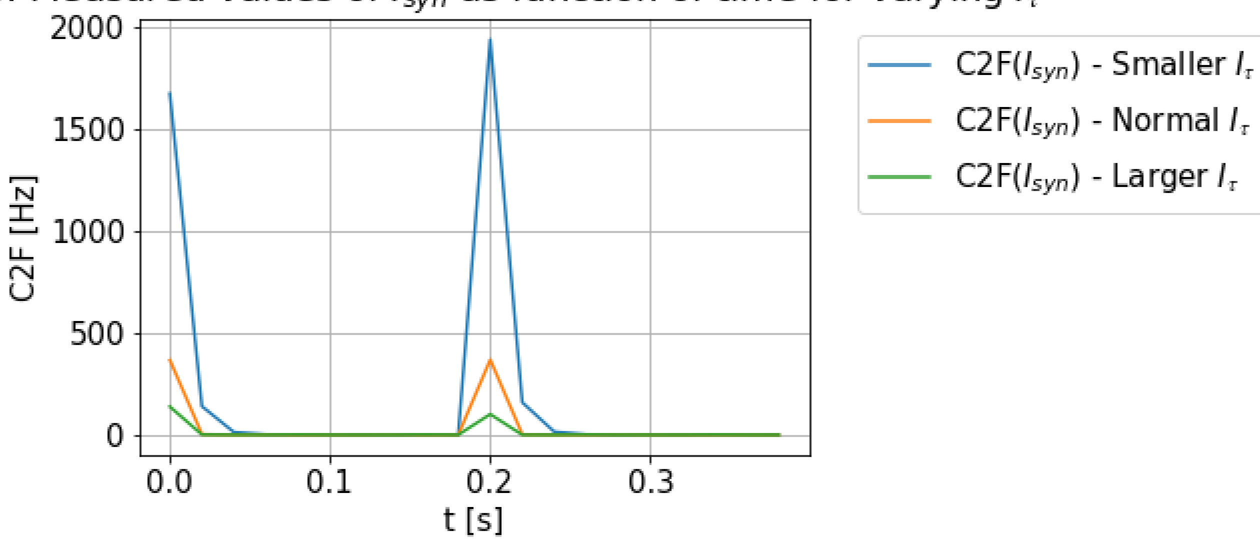


Fig. 6: Measured values of I_{syn} as function of time for varying I_{τ}



3.4 Different I_{thr}

Repeat 3.1 with a smaller and a larger I_{thr}, compare the three curves in the same plot.

In [11]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)]) #change threshold

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

In [12]:
```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [13]:
```
np.savetxt('data/data_ex_3_4_smaller.csv',[t,vsyn,isyn] , delimiter=',')
```

In [14]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 80)]) #change threshold

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 10)])
```

In [15]:
```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
```

```
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [16]:

```
np.savetxt('data/data_ex_3_4_bigger.csv',[t,vsyn,isyn] , delimiter=',')
```

In [17]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})


t,vsyn_smaller,isyn_smaller = np.loadtxt('data/data_ex_3_4_smaller.csv',delimiter=',')
_,vsyn_normal,isyn_normal = np.loadtxt('data/data_ex_3_1.csv',delimiter=',')
_,vsyn_bigger,isyn_bigger = np.loadtxt('data/data_ex_3_4_bigger.csv',delimiter=',')

plt.plot(t,vsyn_smaller,t,vsyn_normal,t,vsyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('$V_{syn}$ [V]')
plt.legend(['$V_{syn}$ - Smaller $I_{thr}$','$V_{syn}$ - Normal $I_{thr}$','$V_{syn}$ - La
plt.title('Fig. 7: Measured values of $V_{syn}$ as function of time for varying $I_{thr}$'

plt.grid()
plt.show()

plt.plot(t[1:],isyn_smaller[1:],t,isyn_normal,t,isyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F$(I_{syn})$ - Smaller $I_{thr}$','C2F$(I_{syn})$ - Normal $I_{thr}$','C2F$
plt.title('Fig. 8: Measured values of $I_{syn}$ as function of time for varying $I_{thr}$'

plt.grid()
plt.show()
```

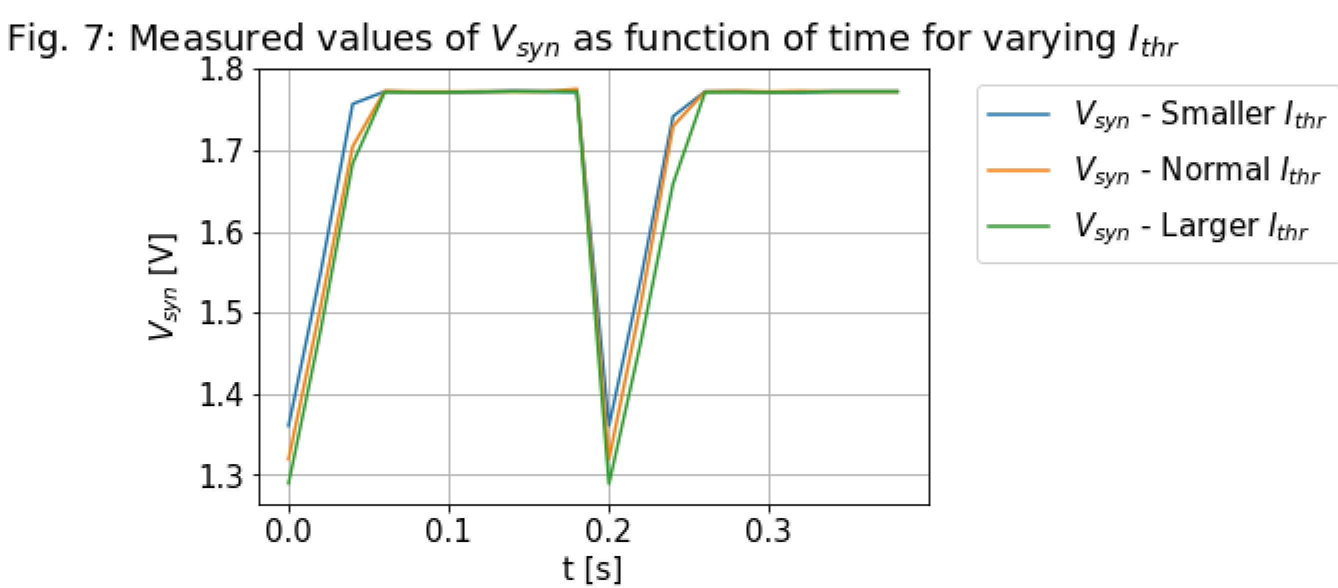
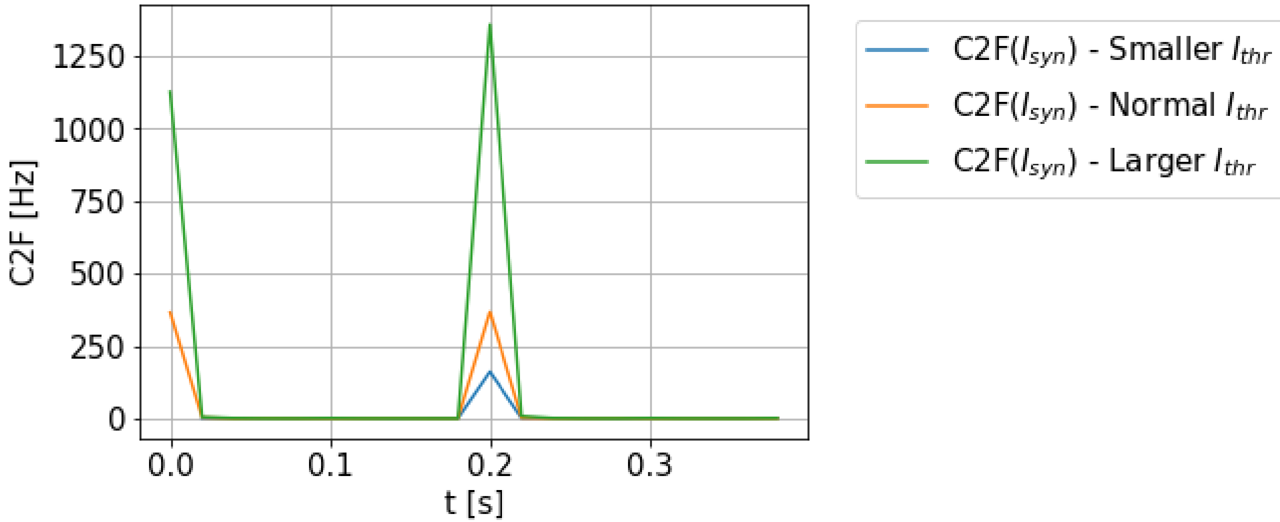


Fig. 8: Measured values of I_{syn} as function of time for varying I_{thr}



3.5 Different pulse width

Repeat 3.1 with a smaller and a larger pulse width, compare the three curves in the same plot.

In [26]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 6)]) # Change pulse width
```

In [27]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [28]:

```
np.savetxt('data/data_ex_3_5_smaller.csv',[t,vsyn,isyn] , delimiter=',')
```

In [29]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 25)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTHR_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 30)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VWEIGHT_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 100)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PEX_VTAU_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 14)]) # Change pulse width
```

In [30]:

```
N_pulses = 2 # for each trial, send 2 input pulses
N_samples_per_pulse = 10 # for each input pulse, sample 10 points
N_samples = N_pulses*N_samples_per_pulse
dT = 0.02 # delta t between the samples, DO NOT CHANGE
t = np.arange(N_samples)*dT
vsyn = np.zeros(N_samples)
isyn = np.zeros(N_samples)
for k in range(N_pulses):
    p.send_coach_events([pyplane.Coach.generate_pulse_event()])
    for i in range(N_samples_per_pulse):
        vsyn[k*N_samples_per_pulse+i] += p.read_voltage(pyplane.AdcChannel.AOUT14)
        inter=p.read_c2f_output(dT)
        isyn[k*N_samples_per_pulse+i] += inter[9]
```

In [31]:

```
np.savetxt('data/data_ex_3_5_bigger.csv',[t,vsyn,isyn] , delimiter=',')
```

In [36]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})


t,vsyn_smaller,isyn_smaller = np.loadtxt('data/data_ex_3_5_smaller.csv',delimiter=',')
_,vsyn_normal,isyn_normal = np.loadtxt('data/data_ex_3_1.csv',delimiter=',')
_,vsyn_bigger,isyn_bigger = np.loadtxt('data/data_ex_3_5_bigger.csv',delimiter=',')

plt.plot(t,vsyn_smaller,t,vsyn_normal,t,vsyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('$V_{syn}$ [V]')
plt.legend(['$V_{syn}$ - Smaller $I_{\\rm{pulse\ width}}$','$V_{syn}$ - Normal $I_{\\rm{pu
plt.title('Fig. 9: Measured values of $V_{syn}$ as function of time for varying $I_{\\rm{p

plt.grid()
plt.show()

plt.plot(t[1:],isyn_smaller[1:],t,isyn_normal,t,isyn_bigger)

plt.xlabel('t [s]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F$(I_{syn})$ - Smaller $I_{\\rm{pulse\ width}}$','C2F$(I_{syn})$ - Normal $
plt.title('Fig. 10: Measured values of $I_{syn}$ as function of time for varying $I_{\\rm
```

```
plt.grid()
plt.show()
```

Fig. 9: Measured values of V_{syn} as function of time for varying $I_{\text{pulse width}}$

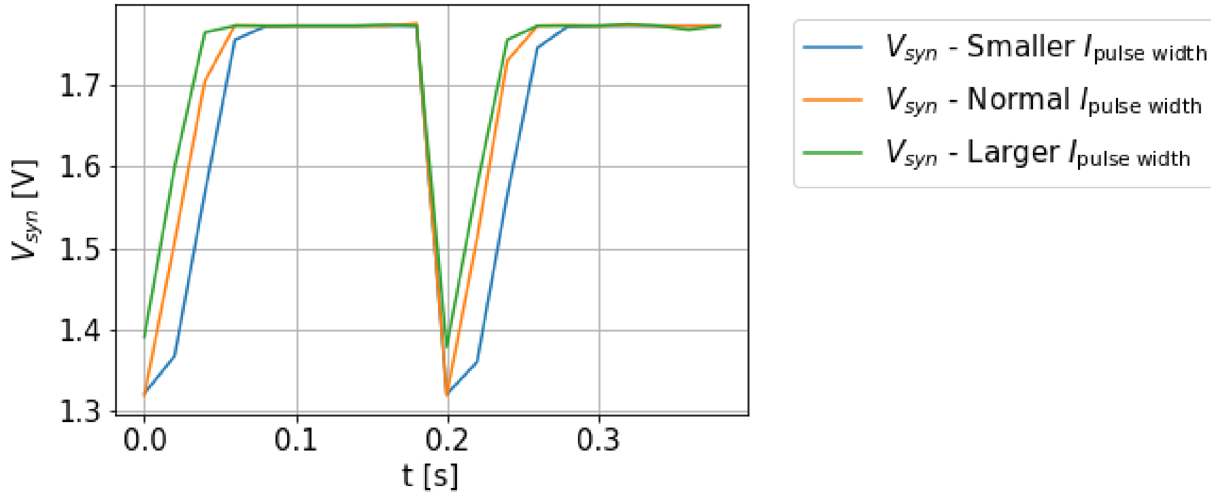


Fig. 10: Measured values of I_{syn} as function of time for varying $I_{\text{pulse width}}$