Neuromorphic engineering I

Lab 9: Silicon Neuron Circuits

Team member 1: Quillan Favey

Team member 2: Hooman Javaheri

Date: 21.11.22

In this lab, we will test a circuit that generates action potentials (spikes) based on an integrate-and-fire model of a neuron spike initiation zone.

The objectives of this lab are:

1. to understand the spiking properties of I&F circuits.
2. to evaluate the effect of the I&F circuit's different bias parameters on its spiking behaviour.

1 Setup

2.1 Connect the device

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

In [ ]:
```
# 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 [ ]:
```
p.get_firmware_version()
```

Out[ ]:
```
(1, 12, 2)
```

In [ ]:
```
# 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, othe
# Because the class chip need to do handshake to get the communication correct.
p.request_events(1)
```

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

Out[ ]:
```
2.336425808380227e-07
```

In [ ]:
```
 # If any of the above steps fail, delete the object, close and halt, stop the serv
# please also say your board number: ttyACMx

# del p
```

2.2 Disable unused circuits

In [ ]:
```
# disable synapses
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.LDS_VTAU_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DPI_VTAU_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DDI_VTAU_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

In [ ]:
```
# disable axon-hillock neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.AHN_VPW_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

In [ ]:
```
# disable thresholded neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ATN_VLEAK_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ATN_VDC_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ATN_VGAIN_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ATN_VSPKTHR_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

In [ ]:
```
# disable sigma-delta neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ASN_VLEAK_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ASN_VDC_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ASN_VGAIN_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

In [ ]:
```
# disable exp neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VLEAK_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VGAIN_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VDC_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

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

In [ ]:
```
# disable hodgekin-huxley neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.HHN_VBUF_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.HHN_VCABUF_N, \
```

```
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.HHN_VDC_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.HHN_VELEAK_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

In [ ]:
```
# disable DVS pixels
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_DIFF_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_CAS_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_ON_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_OFF_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_SF_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_PR_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.DVS_REFR_P, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

2.3 Chip configurations

In [ ]:
```
# select lines and neuron latches
p.send_coach_events([pyplane.Coach.generate_aerc_event(
    pyplane.pyplane.Coach.CurrentOutputSelect.SelectLine6,
```

```
    pyplane.Coach.VoltageOutputSelect.SelectLine2,
    pyplane.Coach.VoltageInputSelect.NoneSelected,
    pyplane.Coach.SynapseSelect.NoneSelected,320)])
```

In [ ]:
```
# 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)])
```

In [ ]:
```
# set up sampling mode
# p.set_sampling_mode(pyplane.SamplingMode.Fast)
# p.set_fast_sampling_adcs([10])
```

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 $\in [0, 256)$

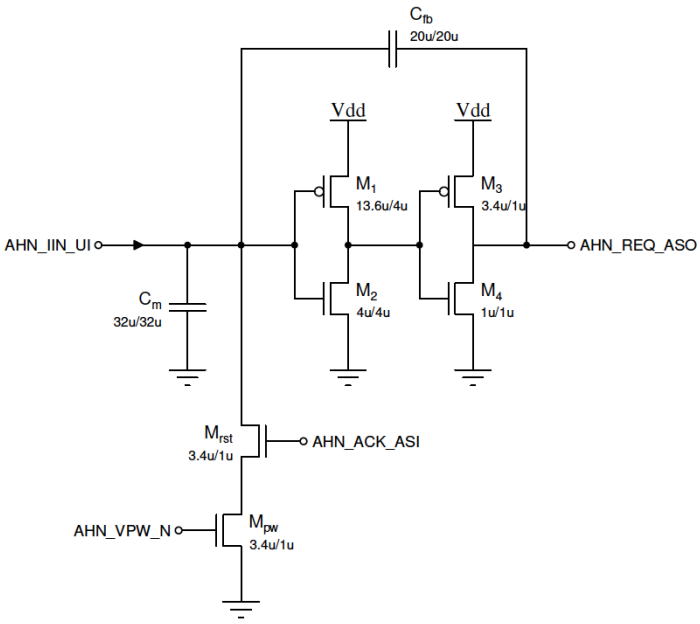
To set a bias, use the function similar to the following:

```
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 Axon-Hillock neuron

The axon-hillock neuron has a constant current input `AHN_IIN_UI` which is about pA (exact value not known), and the voltage on capacitor C_m is output to **ADC[11]**.



2.1 Basic measurement

- Tune `AHN_VPW_N` bias so that the output waveform is more or less symmetric.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.AHN_VPW_N,\
    pyplane.Coach.BiasType.N,\
    pyplane.Coach.BiasGenMasterCurrent.I60pA,2)])
```

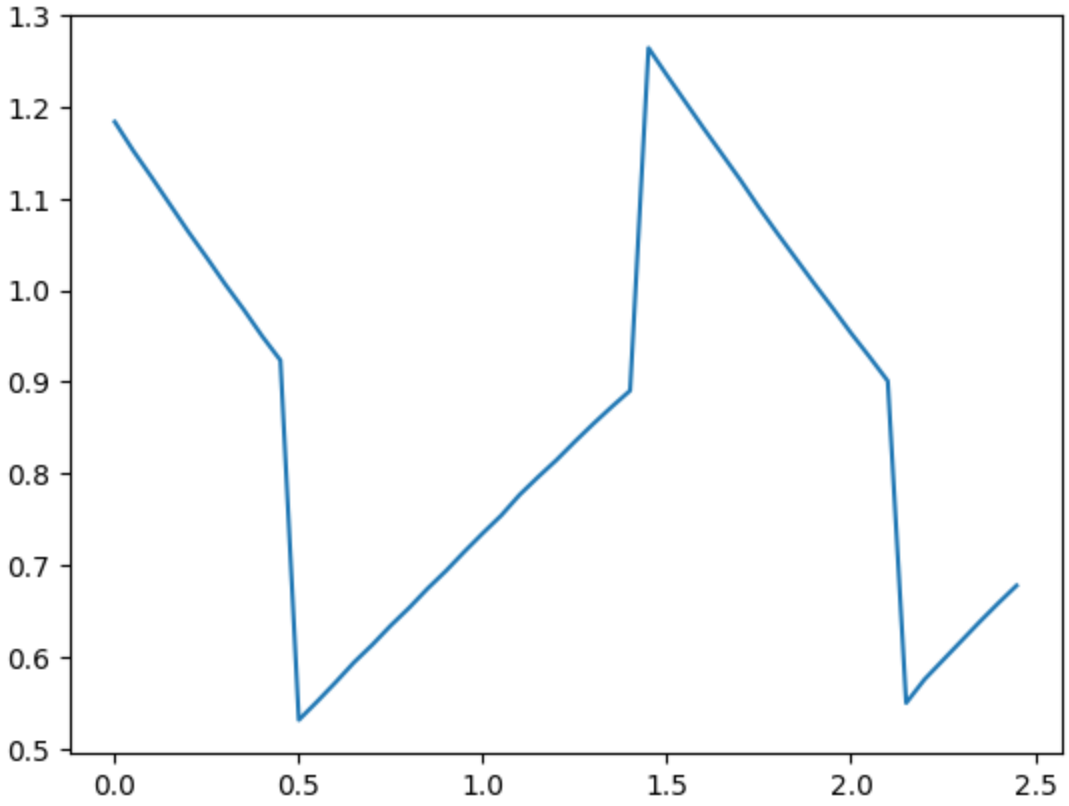
- Data aquisition

In [ ]:
```
N_samples = 50
dT = 0.05

t = np.arange(N_samples)*dT
v = np.zeros(N_samples) # v_Cm

for i in range(N_samples):
    v[i] = p.read_voltage(pyplane.AdcChannel.AOUT11)
    time.sleep(dT)
np.savetxt('data_ex_2_1.csv',[t,v] , delimiter=',')
```

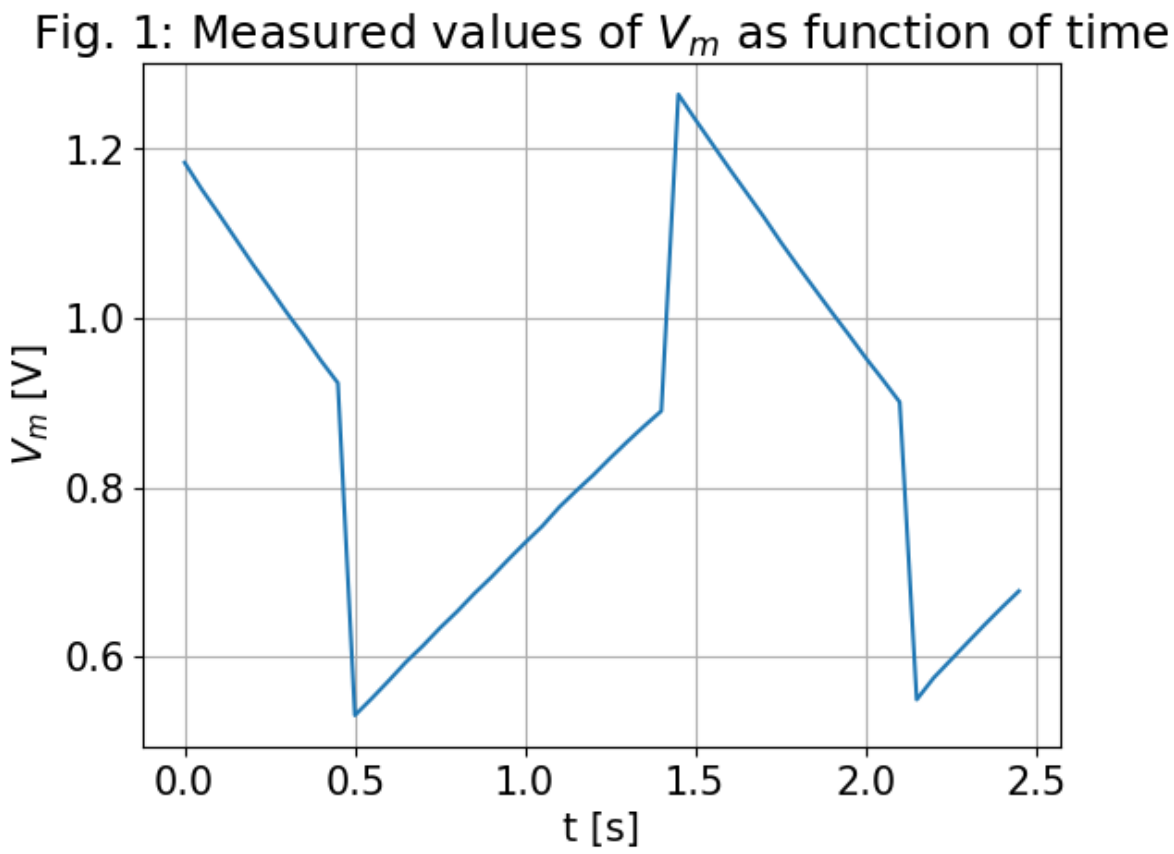
- Plot data

In [ ]:
```
plt.plot(t,v)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fb5a1ccb520>]
```



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


t,v = np.loadtxt('data_ex_2_1.csv',delimiter=',')

plt.plot(t,v)

plt.xlabel('t [s]')
plt.ylabel('$V_{m}$ [V]')
plt.title('Fig. 1: Measured values of $V_{m}$ as function of time')

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



- Save data

In [ ]:
```
np.savetxt('data_ex_2_1.csv',[t,v] , delimiter=',')
```

2.2 Different pulse widths

Now try two more `AHN_VPW_N` values and compare the three curves in the same plot.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.AHN_VPW_N,\
    pyplane.Coach.BiasType.N,\
    pyplane.Coach.BiasGenMasterCurrent.I60pA,1)])
```

In [ ]:
```
N_samples = 50
dT = 0.05

t = np.arange(N_samples)*dT
v = np.zeros(N_samples) # v_Cm

for i in range(N_samples):
    v[i] = p.read_voltage(pyplane.AdcChannel.AOUT11)
    time.sleep(dT)
```

In [ ]:
```
np.savetxt('data_ex_2_2_smaller.csv',[t,v] , delimiter=',')
```

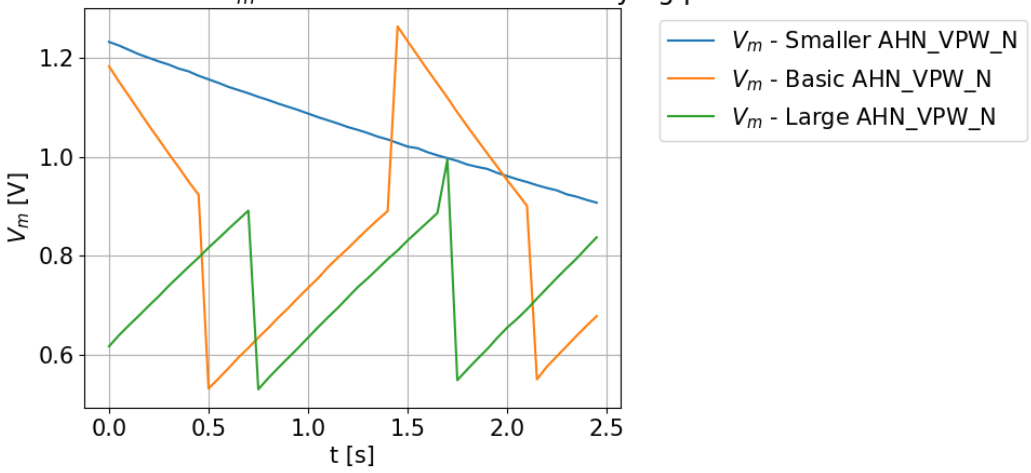
In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.AHN_VPW_N,\
    pyplane.Coach.BiasType.N,\
    pyplane.Coach.BiasGenMasterCurrent.I60pA,30)])
```

In [ ]:
```
N_samples = 50
dT = 0.05

t = np.arange(N_samples)*dT
v = np.zeros(N_samples) # v_Cm

for i in range(N_samples):
    v[i] = p.read_voltage(pyplane.AdcChannel.AOUT11)
    time.sleep(dT)
```

In [ ]:
```
np.savetxt('data_ex_2_2_bigger.csv',[t,v] , delimiter=',')
```

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

t,v = np.loadtxt('data_ex_2_1.csv',delimiter=',')
_,v_smaller = np.loadtxt('data_ex_2_2_smaller.csv',delimiter=',')
_,v_bigger = np.loadtxt('data_ex_2_2_bigger.csv',delimiter=',')

plt.plot(t,v_smaller,t,v,t,v_bigger)
plt.legend(['$V_{m}$ - Smaller AHN_VPW_N','$V_{m}$ - Basic AHN_VPW_N','$V_{m}$ - La

plt.xlabel('t [s]')
plt.ylabel('$V_{m}$ [V]')
plt.title('Fig. 2: Measured values of $V_{m}$ as function of time for varying pulse

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

Fig. 2: Measured values of V_m as function of time for varying pulse widths



2.3 Switch off the circuit

- To avoid the output events interfering with other circuits, we set `AHN_VPW_N` to maximum again.

In [ ]:
```
# disable axon-hillock neuron
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.AHN_VPW_N, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 0)])
```

3 Basic behavoir of classic I&F neuron

The **ADEXIF** (Adaptive Exponential Integrate & Fire) **classic neuron** comprises four major functional blocks: a leaky DPI (=integrate), starved-inverter (=fire), refractory period (=reset) and adaptation block. The adaptation block receives the the spike pulse of the neuron itself through a pulse extender circuit.

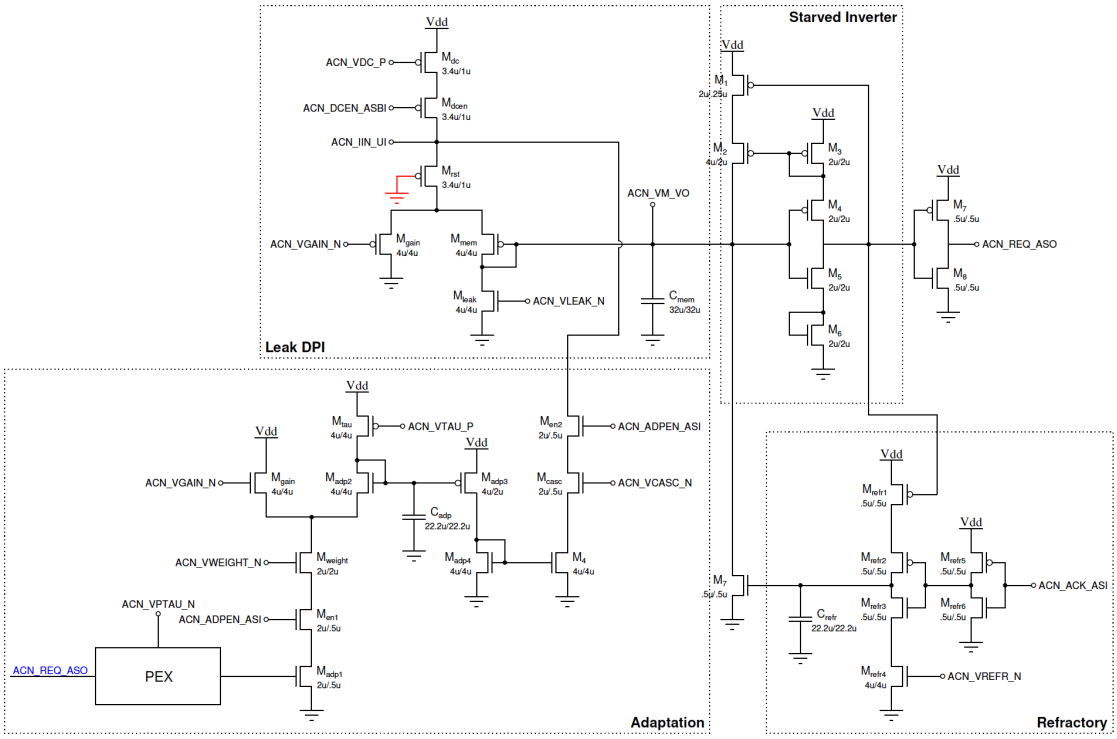
The circuit receives an input current I_{in} (typically the output of a synapse, I_{syn}) and outputs an AER event.

The membrane voltage V_{mem} is provided to observe the internal neuron state at **ADC[10]**.

The neuron circuit has 4 basic biases: V_{dc}, V_{gain}, V_{leak} and V_{refr}. The adaptation block has 5 more biases: $V_{adpgain}$, $V_{adpweight}$, V_{adptau}, $V_{adpcasc}$ & $V_{adpptau}$ (for the pulse extender).

There are two digital control bits: V_{adpen} to enable/disable adaptation, and V_{dcen} to disable/enable the V_{dc} bias input.

C_m sizing was chosen for a capacitance value of 2 pF, while C_{refr} and C_{adp} were chosen for a value of 1 pF.



3.1 Basic measurement

- Tune the biases such that the neuron fires at about 20 Hz.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VLEAK_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 2)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VGAIN_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 6)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VDC_P,\
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 32)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VREFR_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 8)]) #cahnge refractory period
```

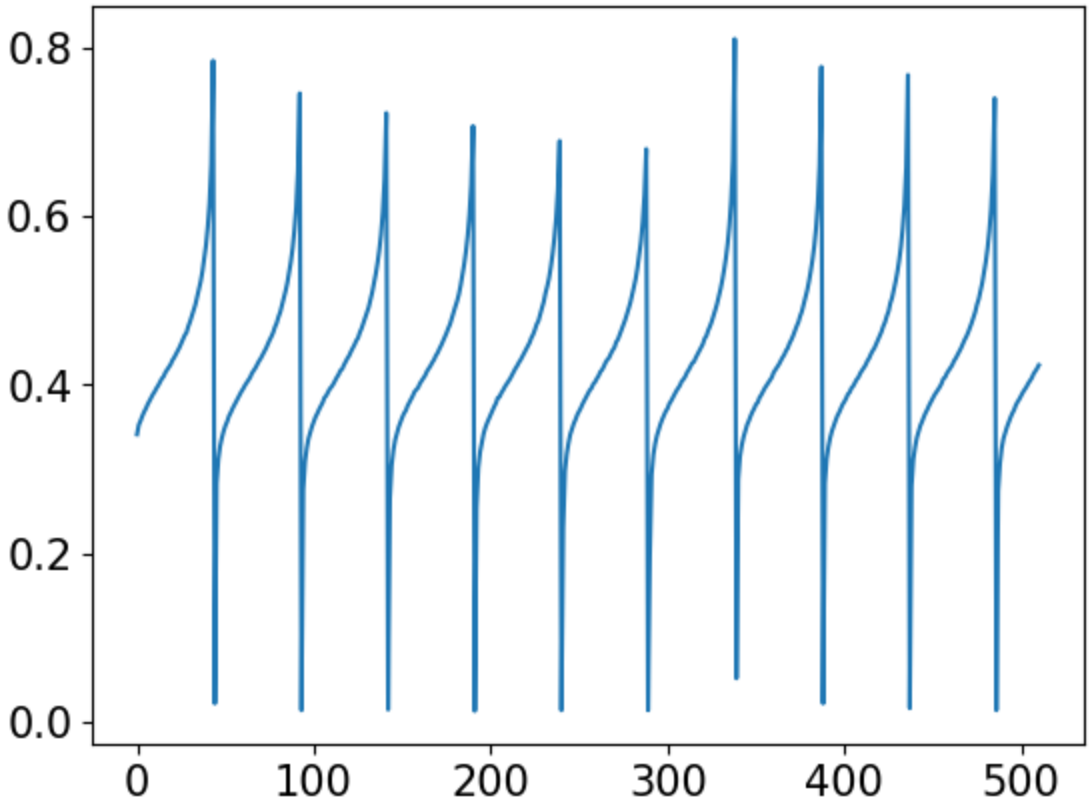
- read data

In [ ]:
```
vm = p.acquire_transient_response(pyplane.DacChannel.DAC1, pyplane.AdcChannel.AOUT1
```

- Plot data

In [ ]:
```
plt.plot(vm)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fb598f08b50>]
```



3.2 Refractory period

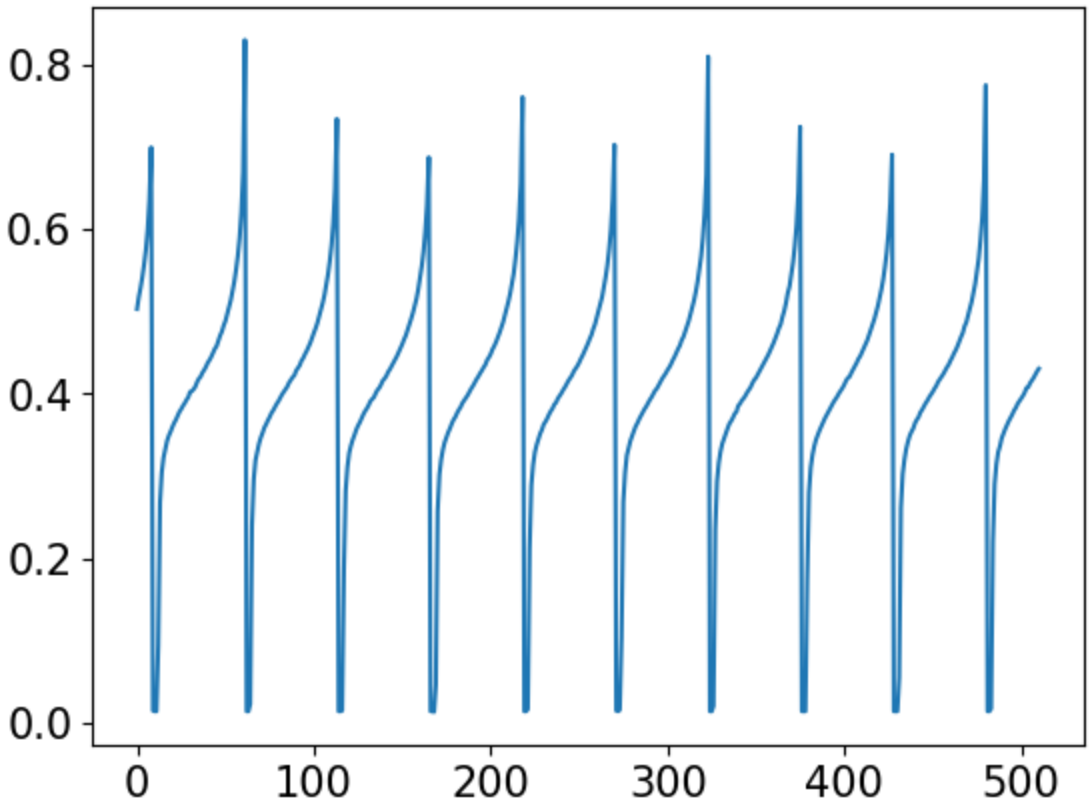
Repeat 3.1 with two other refractory period biases and compare.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VLEAK_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 2)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VGAIN_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 6)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VDC_P,\
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 32)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VREFR_N,\
```

```
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 2)]) #cahnge refractory period
```

In [ ]:
```
vm1 = p.acquire_transient_response(pyplane.DacChannel.DAC1, pyplane.AdcChannel.AOUT

plt.plot(vm1)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fb5993ed460>]
```

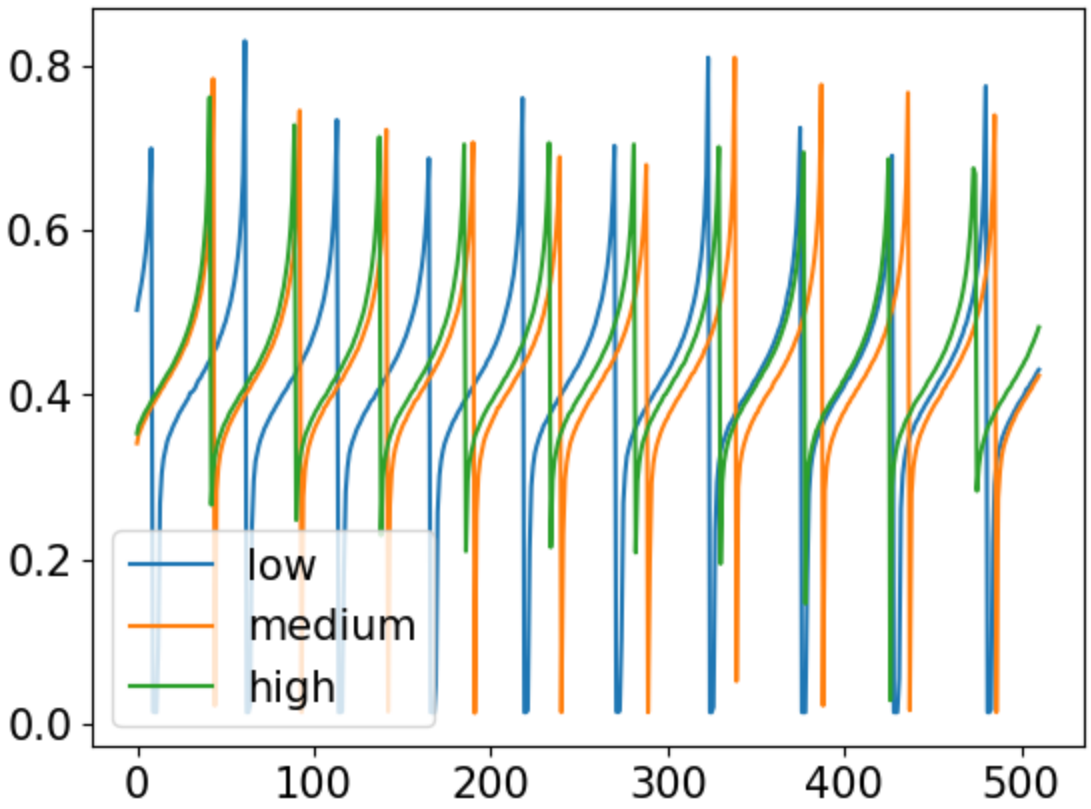


In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VLEAK_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 2)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VGAIN_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 6)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VDC_P,\
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 32)])

p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.ACN_VREFR_N,\
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 50)]) #cahnge refractory period
```

In [ ]:
```
vm2 = p.acquire_transient_response(pyplane.DacChannel.DAC1, pyplane.AdcChannel.AOUT
```

In [ ]:
```
plt.plot(vm1,label="low")
plt.plot(vm,label="medium")
plt.plot(vm2,label="high")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7fb5905699a0>
```




```
The Kernel crashed while executing code in the the current cell or a previous cel
l. Please review the code in the cell(s) to identify a possible cause of the failu
re. Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more inf
o. View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.
```