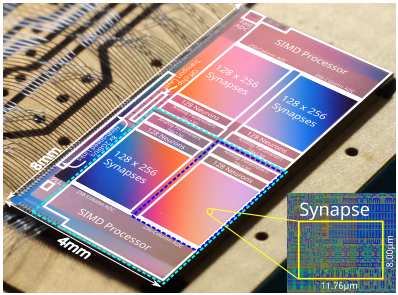
BrainScaleS-2 Tutorial
NICE Conference

Amani Atoui & Jakob Kaiser & Philipp Spilger

28 March 2025

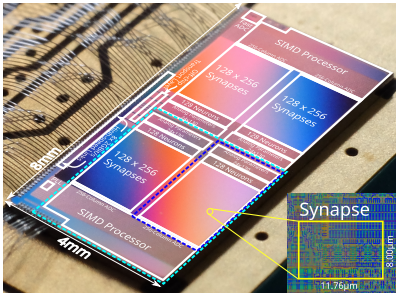
General Architecture



- Mixed-signal
  - Analog neurons and synapses
  - Digital spike communication and configuration
- Accelerated emulation in continuous time (typical speedup of 1000)
- Two general purpose SIMD processors
- High-level APIs in Python based on PyNN and PyTorch

Figure adapted from Müller et al., 2022.

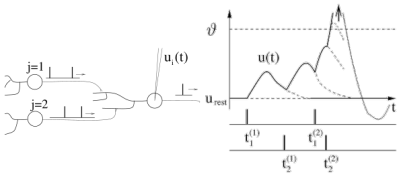
Neuron Configuration

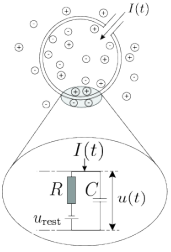


- 512 neurons, 256 synapses/neuron
- Adaptive exponential (AdEx) integrate-and-fire neuron model
  - Can be reduced to leaky integrate-and-fire (LIF) model
- Highly configurable and parametrizable circuits
- Possibility for user-defined plasticity rules

Figure adapted from Müller et al., 2022.

Model of a LIF Neuron





Figures taken from Gerstner et al., 2014.

- Leaky integrate-and-fire model (LIF)

$$C\frac{\mathrm{d}u(t)}{\mathrm{d}t} = \frac{1}{R}\cdot(u_{\text{rest}} - u(t)) + I_{\text{syn}}(t) + I(t) \qquad (1)$$

Login into EBRAINS

- `wiki.ebrains.eu`
- `https://tinyurl.com/bss2nice2025`

Single LIF Neuron

- `ts_00-single_neuron.ipynb`

Single LIF Neuron

- `ts_00-single_neuron.ipynb`
- Key takeaways:
  - PyNN module as a user-interface
  - Hardware units instead of biological units
  - Time acceleration factor
  - Fixed-pattern variations across neurons

Neuron Model

- Adaptive exponential (AdEx) integrate-and-fire neuron model:

$$C_\mathrm{m}\frac{\mathrm{d}V_\mathrm{m}(t)}{\mathrm{d}t} = g_\mathrm{L}\cdot(V_\mathrm{L}-V_\mathrm{m}(t)) + g_\mathrm{L}\Delta_\mathrm{T}\exp\left(\frac{V_\mathrm{m}(t)-V_\mathrm{T}}{\Delta_\mathrm{T}}\right) + I_\mathrm{syn}(t)+I(t)-w(t), \tag{2}$$

$$\tau_\mathrm{w}\frac{\mathrm{d}w(t)}{\mathrm{d}t} = a\,(V_\mathrm{m}(t)-V_\mathrm{L})-w(t), \tag{3}$$

$$\mathrm{w}(t=t^{(f)}) = \mathrm{w}(t)+b \tag{4}$$

Spiking Patterns

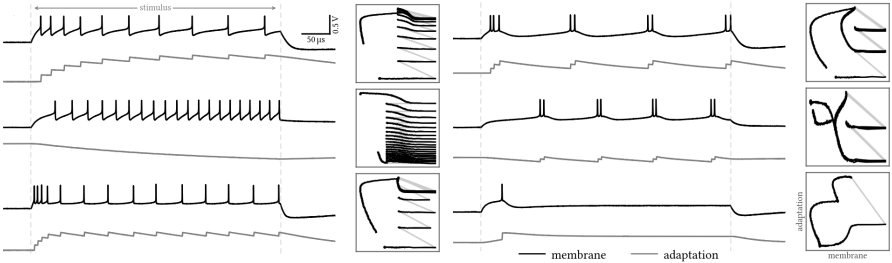


Figure adapted from Billaudelle et al., 2022.

AdEx Neuron Dynamics

- `ts_08-adex_complex_dynamics.ipynb`

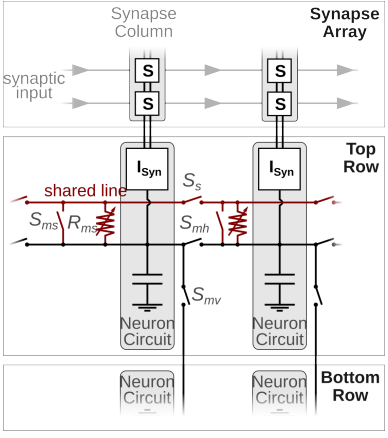
AdEx Neuron Dynamics

- `ts_08-adex_complex_dynamics.ipynb`
- Key takeaways:
  - Configurability of circuit (LIF, adaptation, exponential)
  - Emulation of AdEx dynamics
  - Ability to reproduce different spiking patterns

Multi-compartment Neuron Models

Figures taken from Gerstner et al., 2014; Kaiser et al., 2022.

Multi-compartment Neuron Models



Figures taken from Gerstner et al., 2014; Kaiser et al., 2022.

Multi-compartment Neuron Models

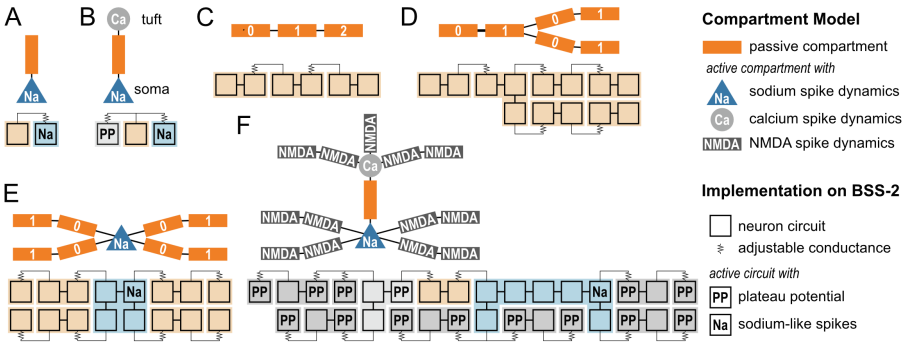


Figure taken from Kaiser et al., 2022.

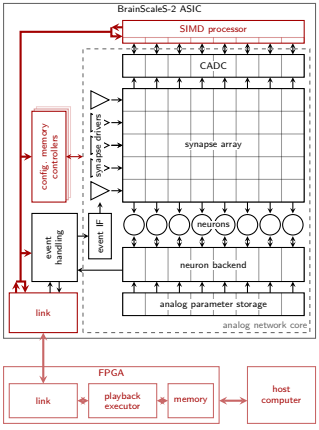
Setting-up Multicompartment Neuron Models

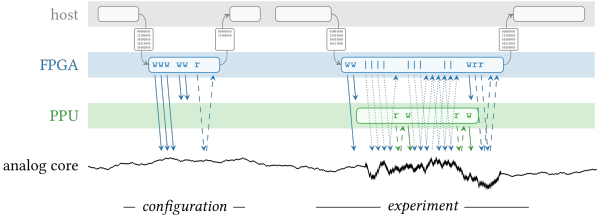
- `ts_03-multicompartment.ipynb`

Setting-up Multicompartment Neuron Models

- `ts_03-multicompartment.ipynb`
- Key takeaways:
  - Configuring neuron circuits as neuron compartments
  - Defining multi-compartment neuron models
  - Reproducing dendritic dynamics

Experiment Workflow

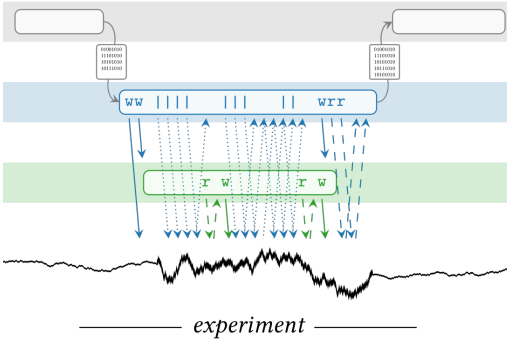




1. Experiment definition in Python
2. Translation to a valid hardware configuration
3. Execution
4. Readout of recorded data

Figures adapted from Müller et al., 2022; Billaudelle, 2022.

Plasticity Experiments



1. Defining a timer
2. Recording observables every period of the timer
3. Execution of a plasticity rule for weight update

Figures adapted from Müller et al., 2022; Billaudelle, 2022.

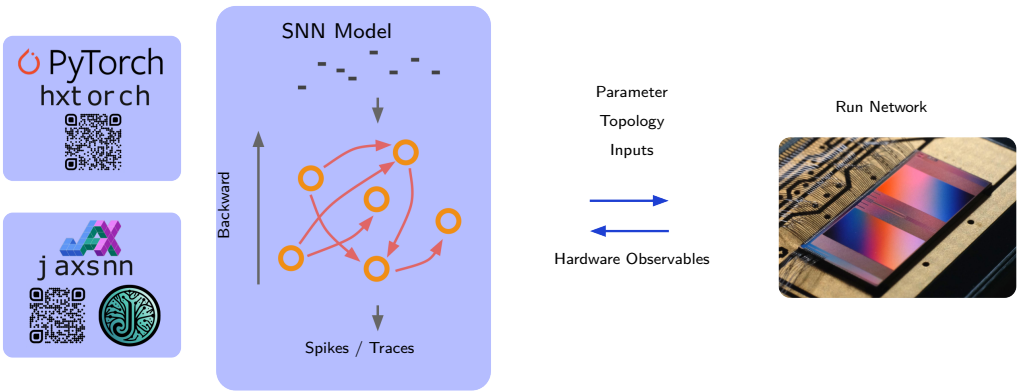
Tutorial

- `ts_11-plasticity_homeostasis.ipynb`

Tutorial

- `ts_11-plasticity_homeostasis.ipynb`
- Key takeaways:
  - User-defined plasticity rule using C++
  - Defining a timer for experiment control
  - Dynamic or static update of weights
  - Weight update and recording observables every period of the timer

Training BrainScaleS-2 in-the-loop



Separation between hardware experiment and PyTorch graph
→ Construct PyTorch graph only after hardware observables are present

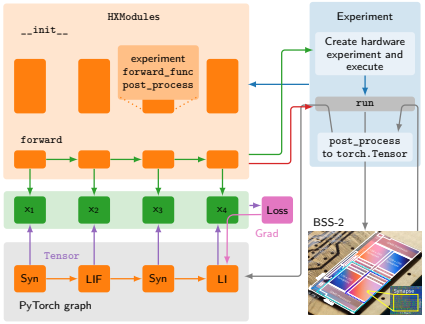
Experiment description — API

PyTorch

```
import torch.nn as nn



syn1 = nn.Linear(...)
nrn1 = nn.ReLU(...)
syn2 = nn.Linear(...)
nrn2 = nn.ReLU(...)

x1 = syn1(input)
x2 = nrn1(x1)
x3 = syn2(x2)
x4 = nrn2(x3)



loss = f(x4)
loss.backward()
```

hxtorch.snn

```
import hxtorch.snn as hxsnn

exp = hxsnn.Experiment()

syn1 = hxsnn.Synapse(exp, ...)
nrn1 = hxsnn.LIF(exp, ...)
syn2 = hxsnn.Synapse(exp, ...)
nrn2 = hxsnn.LI(exp, ...)

x1 = syn1(input)
x2 = nrn1(x1)
x3 = syn2(x2)
x4 = nrn2(x3)

hxsnn.run(exp, ...)

loss = f(x4)
loss.backward()
```



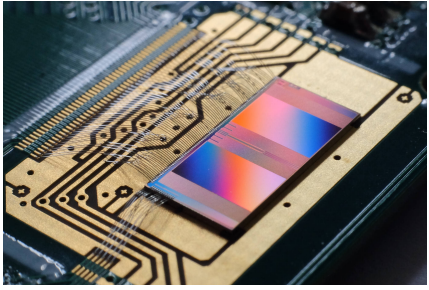
Tutorial

- `ts_12-hxtorch_snn_intro.ipynb`

Tutorial

- `ts_12-hxtorch_snn_intro.ipynb`
- Key takeaways:
  - Setting-up neurons and synapses using hxtorch
  - Setting-up experiments using hxtorch
  - Performing gradient descent on observables

Q&A



References

Billaudelle, Sebastian (2022). "From transistors to learning systems: circuits and algorithms for brain-inspired computing". PhD thesis. Universität Heidelberg.

Billaudelle, Sebastian et al. (2022). "An accurate and flexible analog emulation of AdEx neuron dynamics in silicon". In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4. DOI: 10.1109/ICECS202256217.2022.9971058.

Gerstner, Wulfram et al. (2014). *Neuronal Dynamics*. Cambridge University Press.

Kaiser, Jakob et al. (2022). "Emulating dendritic computing paradigms on analog neuromorphic hardware". In: *Neuroscience* 489, pp. 290–300. ISSN: 0306-4522. DOI: 10.1016/j.neuroscience.2021.08.013. URL: https://www.sciencedirect.com/science/article/pii/S0306452221004218.

Müller, Eric et al. (2022). "A Scalable Approach to Modeling on Accelerated Neuromorphic Hardware". In: *Front. Neurosci.* 16. ISSN: 1662-453X. DOI: 10.3389/fnins.2022.884128.