

Neuromorphic Computing with BrainScaleS

Accelerated Analog Spiking Neural Network Emulation

Eric Müller

`mueller@kip.uni-heidelberg.de`

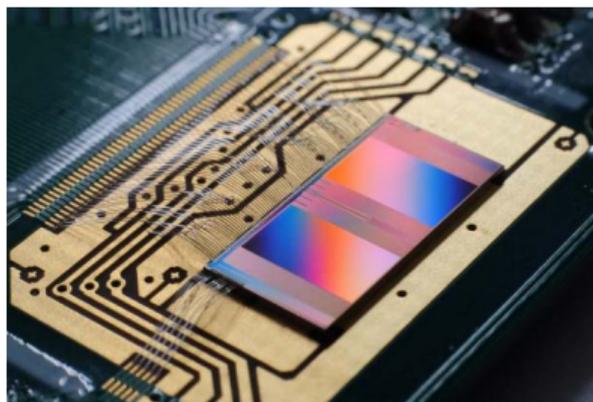
EINC, Heidelberg University

2022-11-08



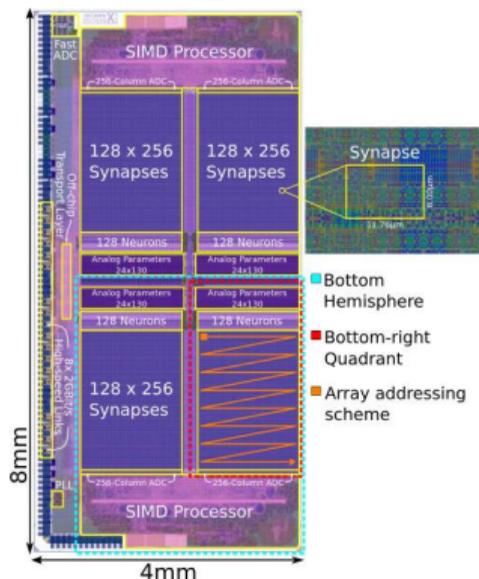
EBRAINS





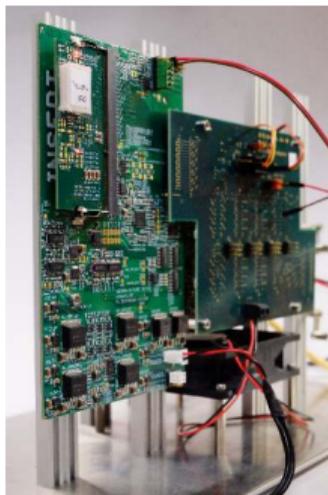
- Physical model (mixed-signal) with accelerated model dynamics ($\sim 10^3$)
- AdEx neurons, synapses with correlation sensors
- Support for 'online' updates of neuron parameters, synapses and network topology
- Programmable plasticity (via embedded SIMD processors and per-synapse observables)
- Structured neurons & nonlinear effects of dendrites
- Non-spiking operation mode

BrainScaleS-2



- Physical model (mixed-signal) with accelerated model dynamics ($\sim 10^3$)
- AdEx neurons, synapses with correlation sensors
- Support for 'online' updates of neuron parameters, synapses and network topology
- Programmable plasticity (via embedded SIMD processors and per-synapse observables)
- Structured neurons & nonlinear effects of dendrites
- Non-spiking operation mode

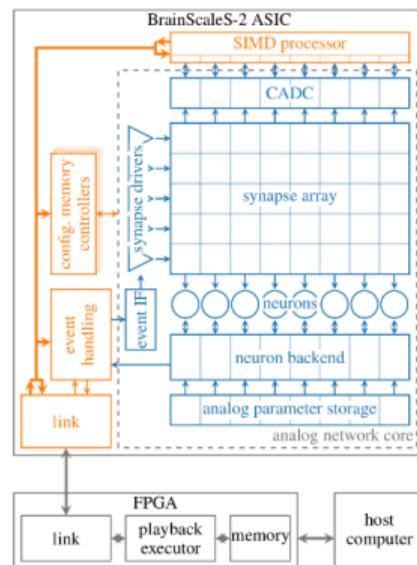
BrainScaleS-2 Systems



- Hardware setups
 - Single-chip lab systems – local and remote users
 - (Mobile systems – embedded operation)
 - (Multi-chip systems – under development)
- “Network-attached accelerators”
- Abstraction levels in software:
 - `PyNN.brainscales` (and `hxtorch.snn`)
 - Lower-level layers for configuration and control
 - APIs for modeling, commissioning and development
- Time-shared hardware resource
 - Experiment service for interactive ($O(10\text{ ms})$) access from Collab (& soon federated resources/HPC)

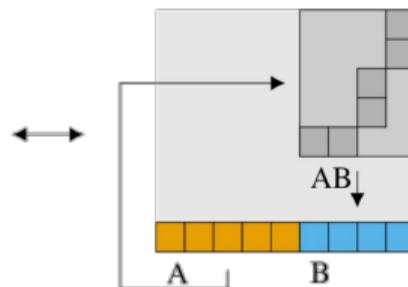
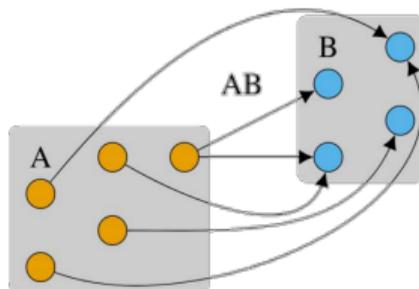
Neuromorphic “Programming Model”

- Neurons & Morphology
- Synapses & Topology
- Stimulus, recorded observables → I/O
- Controllers:
 - Host computer
 - FPGA
 - Embedded processors



Neuromorphic “Programming Model”

- **Neurons & Morphology**
- **Synapses & Topology**
- Stimulus, recorded observables → I/O
- **Controllers:**
 - Host computer
 - FPGA
 - Embedded processors



```
from dlens_vx_v2 import lola, halco

# Initialize Morphology of LogicalNeuron
morphology = lola.Morphology()

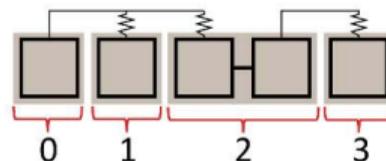
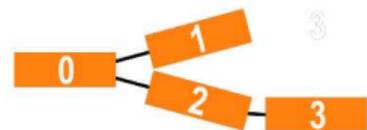
# define coordinates
column = halco.NeuronColumnOnLogicalNeuron # 0, 1, ..., 127
row = halco.NeuronRowOnLogicalNeuron # 0, 1
coord = halco.AtomicNeuronOnLogicalNeuron

# Add compartments
morphology.create_compartment([coord(0, 0)])
morphology.create_compartment([coord(1, 0)])
morphology.create_compartment([coord(2, 0), coord(3, 0)])
morphology.create_compartment([coord(4, 0)])

# enable conductance to somatic shared line
for neuron in [coord(1, 0) for i in [1, 2, 4]]:
    morphology.connect_resistor_to_soma(neuron)

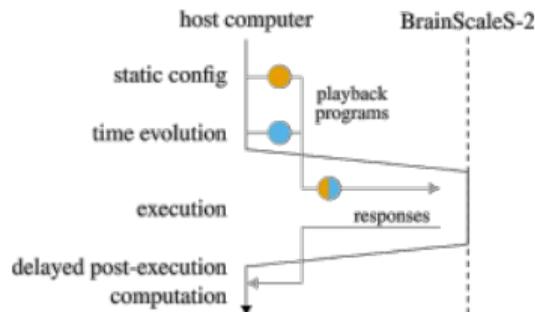
# directly connect to somatic shared line
for neuron in [coord(1, 0) for i in [0, 3]]:
    morphology.connect_to_soma(neuron)

# connect somatic shared line
morphology.connect_soma_line(column(0), column(2), row(0))
morphology.connect_soma_line(column(3), column(4), row(0))
```

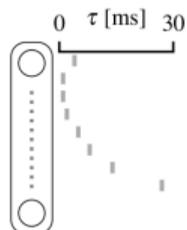


Neuromorphic “Programming Model”

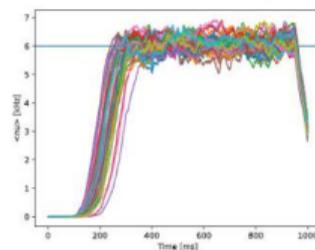
- Neurons & Morphology
- Synapses & Topology
- **Stimulus, recorded observables** → I/O
- **Controllers:**
 - Host computer
 - FPGA
 - Embedded processors



spike encoding /
input spike trains



```
class HomeostaticSynapse(pynn.PlasticityRule,  
                        pynn.standardmodels.synapses.StaticSynapse):  
    # ...  
    def generate_kernel(self) -> str:  
        return textwrap.dedent("""  
            // C++ ...  
            template <size_t N>  
            void PLASTICITY_RULE_KERNEL(  
                std::array<SynapseArrayViewHandle, N>& synapses,  
                std::array<PPUDnDLS, N> synapses) {  
                /* embedded processors have access to a set of  
                 * observables and controllables ... */  
            }  
            """).format(...)  
    # ...  
    synapse_type = HomeostaticSynapse(timer=timer, target=60, weight=0)  
    pynn.Projection(pop_input, nrn, pynn.AllToAllConnector(),  
                  synapse_type=synapse_type)  
    # ...
```



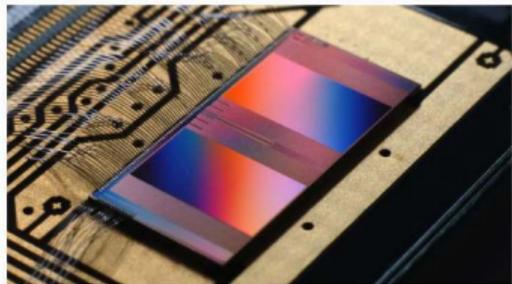
Summary

- Interactive access to accelerated neuromorphic BrainScaleS via EBRAINS
- Open research platform
- Join the hands-on session!
- Thu, 2022-11-10 13–14 CET (12–13 UTC)



Welcome to the BrainScaleS-2 Demos & Examples! Welcome to the BrainScaleS-2 Tutorial [View page source](#)

Welcome to the BrainScaleS-2 Tutorial



Hello and welcome to this tutorial that will interactively guide you through your first experiments on the BrainScaleS-2 system!

```
File Edit View Run Kernel Git Tabs Settings Help  
girlday_03_pynn_introduc...  
[0]: # Die Spikes des Neurons in 'pop' können ausgegeben werden.  
spike_train = pop.get_data("spikes").segments[0].spike_trains[0]  
print("Das Neuron hat {len(spike_train)} mal gefeuert.")  
print("Die Zeitpunkte der Spikes waren: {spike_train}")  
  
# Auch sein Membranpotential kann geplottet werden.  
mem_v = pop.get_data("v").segments[0].irregularlysampledsignals[0]  
  
# Ein Modul zur grafischen Darstellung wird geladen.  
%matplotlib inline  
import matplotlib.pyplot as plt  
  
# Die Grafik wird erstellt.  
# Das Membranpotential ist in Hardware Einheiten gegeben.  
plt.figure()  
plt.plot(mem_v.times, mem_v)  
plt.xlabel("Zeit [ms]")  
plt.ylabel("Membranpotential [158]")  
plt.show()  
  
Das Neuron hat 6 mal gefeuert.  
Die Zeitpunkte der Spikes waren: [1 ms
```

Anschließend kann das aufgenommene Verhalten der Neuronen ausgelesen werden.

Name	Last Modified
_static	5 minutes ago
common_righty_cal...	5 minutes ago
common_quigeldy...	5 minutes ago
fp_0-welcome.ipynb	5 minutes ago
fp_binary_operation...	5 minutes ago
fp_pynn_introduction...	5 minutes ago
fp_sudoku.ipynb	5 minutes ago
girlday_01_biologic...	5 minutes ago
girlday_02_neurom...	5 minutes ago
girlday_03_pynn_in...	5 minutes ago
girlday_04_single...	5 minutes ago
girlday_05-1_multp...	5 minutes ago
girlday_05-2_multp...	5 minutes ago
girlday_06_machin...	5 minutes ago
index.ipynb	5 minutes ago
spiking_cocolist.gbin	seconds ago
tutorial_0-welcome...	5 minutes ago
tutorial_1-single_nau...	5 minutes ago
tutorial_2-superspike...	5 minutes ago
tutorial_3-hagen_intr...	5 minutes ago
tutorial_4-hagen_pro...	5 minutes ago
tutorial_5-plasticity_f...	5 minutes ago
tutorial_6-multicomp...	5 minutes ago
tutorial_7-yin_yang_j...	5 minutes ago
tutorial_8-dynamic_r...	5 minutes ago

Thank you!

