Neuromorphic Computing with BrainScaleS

Accelerated Analog Spiking Neural Network Emulation

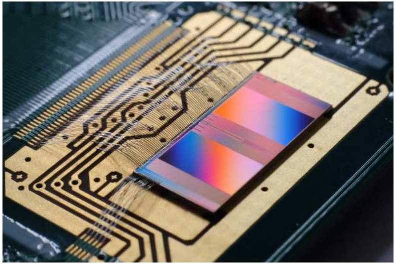
Eric Müller
mueller@kip.uni-heidelberg.de

EINC, Heidelberg University

2022-11-08

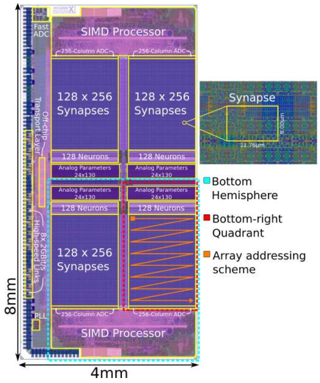
EBRAINS

BrainScaleS-2



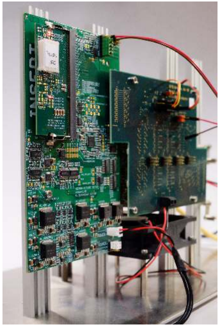
- Physical model (mixed-signal) with accelerated model dynamics ($\sim 10^3$)
- AdEx neurons, synapses with correlation sensors
- Support for 'online' updates of neuron parameters, synapses and network topology
- Programmable plasticity (via embedded SIMD processors and per-synapse observables)
- Structured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

BrainScaleS-2



- 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)

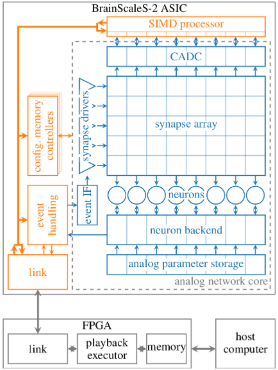
→ “Network-attached accelerators”

- 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 ms)) access from Collab (& soon federated resources/HPC)

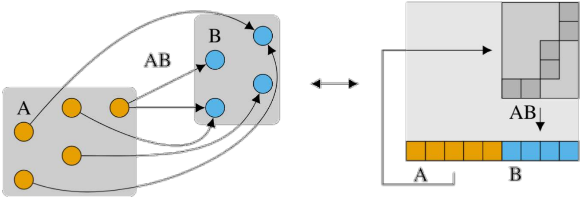
Neuromorphic "Programming Model"

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

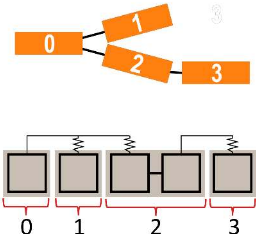


Neuromorphic “Programming Model”

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

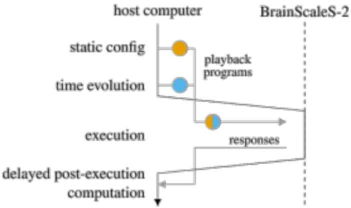


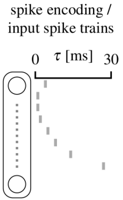

```
from dlens_vx_v2 import lola, halco

# 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(i, 0) for i in [1, 2, 4]]:
    morphology.connect_resistor_to_soma(neuron)

# directly connect to somatic shared line
for neuron in [coord(i, 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))
```



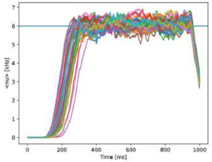
Neuromorphic “Programming Model”

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






```
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<PPUOnDLS, N> synrams) {
                    /* 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)
# ...
```

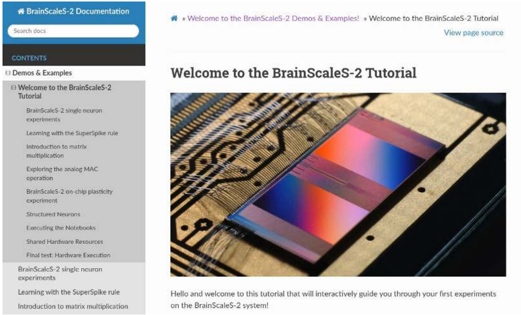


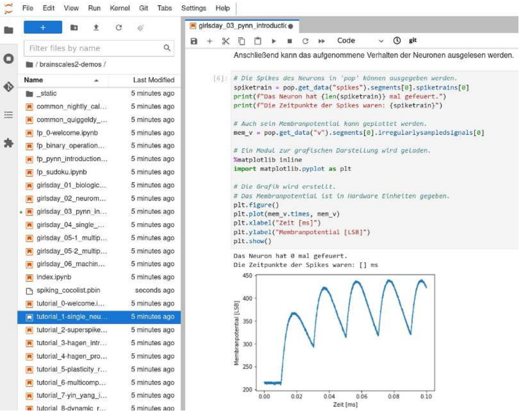
Summary

- 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)





Thank you!