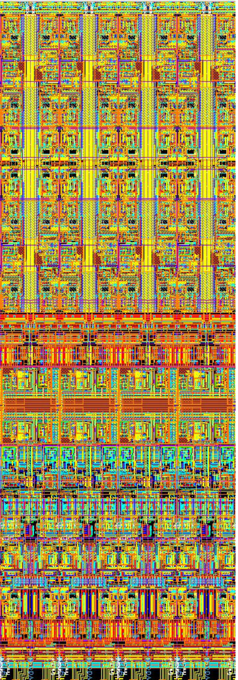


Introduction to the BrainScaleS Tutorial

EBRAINS Infrastructure Training

Eric Müller

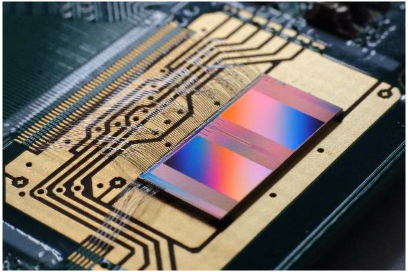
mueller@kip.uni-heidelberg.de

Electronic Vision(s), KIP
UHEI

2020-11-03

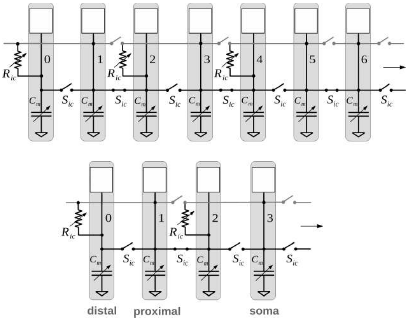
EBRAINS

BrainScaleS-2



- Mixed-signal implementation
- Accelerated model dynamics ($\sim 10^3$)
- AdEx neurons, short-term plasticity
- Support for online updates of neuron parameters, synapses (and network topology)
- Programmable plasticity
- Structured neurons & nonlinear effects of dendrites
- Non-spiking operation mode

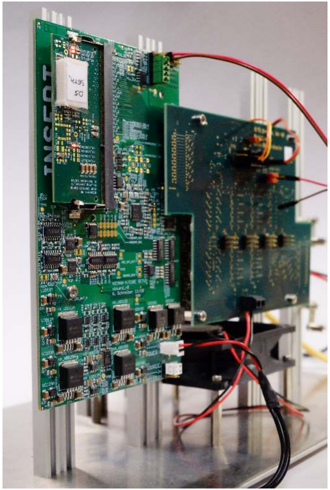
BrainScaleS-2



- Mixed-signal implementation
- Accelerated model dynamics ($\sim 10^3$)
- *AdEx neurons, short-term plasticity*
- *Support for online updates of neuron parameters, synapses (and network topology)*
- *Programmable plasticity*
- *Structured neurons & nonlinear effects of dendrites*
- *Non-spiking operation mode*

(not covered by tutorial)

BrainScaleS — System Access

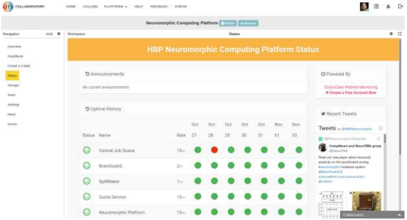


- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)

BrainScaleS — System Access

- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)

BrainScaleS — System Access

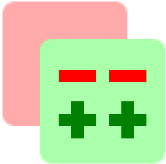


- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)
- Exposed via HBP collaboratory

BrainScaleS — System Access

- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)
- Exposed via HBP collaboratory
- Resource management via SLURM (plus custom extensions for accelerator hardware management)

BrainScaleS — System Access

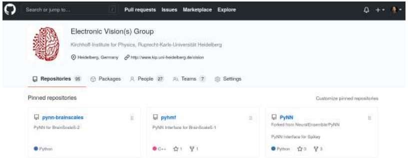


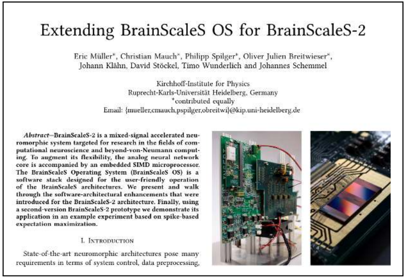
- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)
- Exposed via HBP collaboratory
- Resource management via SLURM (plus custom extensions for accelerator hardware management)
- Software development using strict code review, continuous integration & deployment

BrainScaleS — System Access

- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)
- Exposed via HBP collaboratory
- Resource management via SLURM (plus custom extensions for accelerator hardware management)
- Software development using strict code review, continuous integration & deployment
- Fully clusterized software environment

BrainScaleS — System Access





Extending BrainScaleS OS for BrainScaleS-2

Eric Müller*, Christian Mauch*, Philipp Spilger*, Oliver Julien Breitwieser*,
Johann Klähn, David Stöckel, Timo Wunderlich and Johannes Schemmel

Kirchhoff-Institute for Physics
Ruprecht-Karls-Universität Heidelberg, Germany
*contributed equally
Email: {mueller,cmauch,pspilger,obreitwi}@kip.uni-heidelberg.de

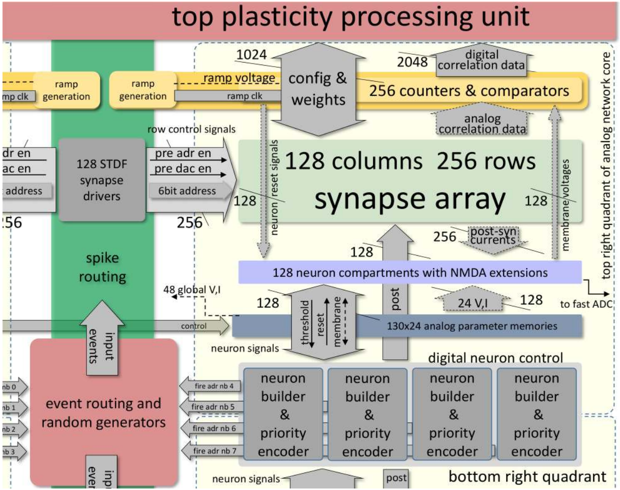
Abstract—BrainScaleS-2 is a mixed-signal accelerated neuromorphic system targeted for research in the fields of computational neuroscience and beyond-von-Neumann computing. To augment its flexibility, the analog neural network core is accompanied by an embedded SIMD microprocessor. The BrainScaleS Operating System (BrainScaleS OS) is a software stack designed for the user-friendly operation of the BrainScaleS architectures. We present and walk through the software-architectural enhancements that were introduced for the BrainScaleS-2 architecture. Finally, using a second-version BrainScaleS-2 prototype we demonstrate its application in an example experiment based on spike-based expectation maximization.

I. Introduction

State-of-the-art neuromorphic architectures pose many requirements in terms of system control, data preprocessing,

- Custom chips, custom setups, but “standard” interfaces (e.g., GbE)
- Cluster-attached accelerators (at UHEI: BSS-1&2)
- Exposed via HBP collaboratory
- Resource management via SLURM (plus custom extensions for accelerator hardware management)
- Software development using strict code review, continuous integration & deployment
- Fully clusterized software environment
- System software implemented in C++, open sourced (cf. here and here) incl. Python wrappers for all relevant layers

BrainScaleS-2 — Internals?



[J. Schemmel, S. Billaudelle, P. Dauer, J. Weis, 2020]

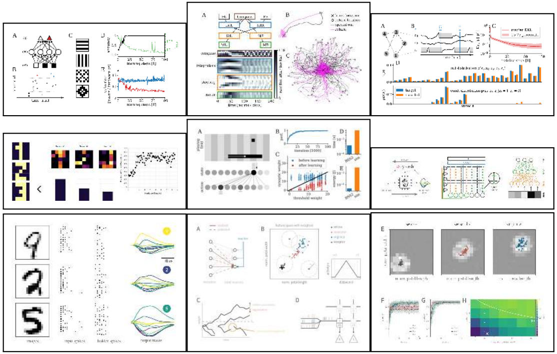
BrainScaleS-2 — Low-level Configuration

```python
# ...
def configure_synapses(*args):
    """
    Configure routing crossbar, PADI bus, synapse drivers, and parts
    of the synapse array.
    """
    fisch_builder = fisch.PlaybackProgramBuilder()
    fisch_builder.write(anncore_center_ba, fisch.Omnibus(0xffff))
    config_builder.merge_back(fisch_builder)

    # synapse array
    correlation_switch_quad = haldls.ColumnCorrelationQuad()
    switch = correlation_switch_quad.ColumnCorrelationSwitch()
    switch.enable_internal_causal = True
    switch.enable_internal_acausal = True
    for s in range(4):
        correlation_switch_quad.set_switch(s, switch)

    for sq in iter_all(halco.ColumnCorrelationQuadOnDLS):
        config_builder.write(sq, correlation_switch_quad,
                             haldls.Backend.Omnibus)

    current_switch_quad = haldls.ColumnCurrentQuad()
    switch = current_switch_quad.ColumnCurrentSwitch()
    switch.enable_synaptic_current_excitatory = True
    switch.enable_synaptic_current_inhibitory = True
    for s in range(4):
        current_switch_quad.set_switch(s, switch)
# ...
```

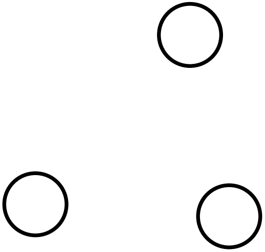


Expert-only?

PyNN

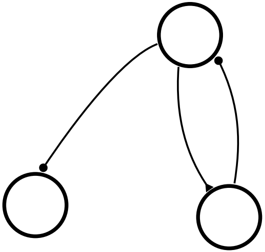
"PyNN — A Python package for simulator-independent specification of neuronal network models."

PyNN



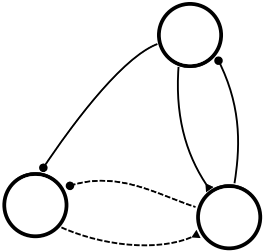
- Python-based modeling API for spiking neural networks

PyNN



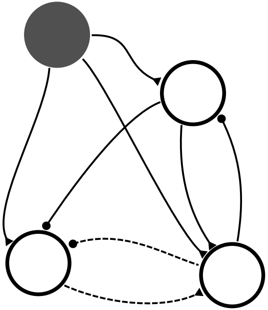
- Python-based modeling API for spiking neural networks
- Topology-centric description (data flow graph)

PyNN



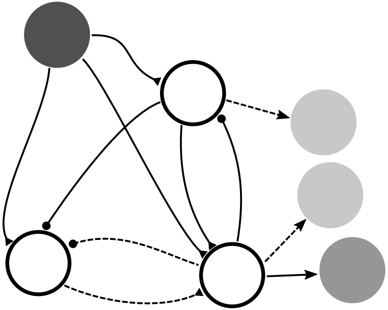
- Python-based modeling API for spiking neural networks
- Topology-centric description (data flow graph)
- Neuron and synapse dynamics (cell and synapse types)
- Experiment protocol ("what and when")

PyNN



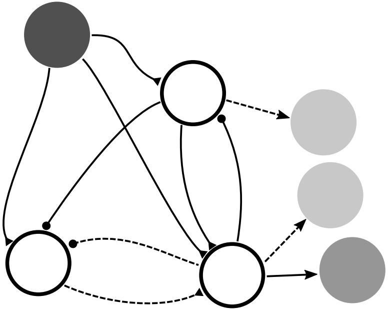
- Python-based modeling API for spiking neural networks
- Topology-centric description (data flow graph)
- Neuron and synapse dynamics (cell and synapse types)
- Experiment protocol (“what and when”)
    - stimulus (input nodes, e.g., spike trains)

PyNN



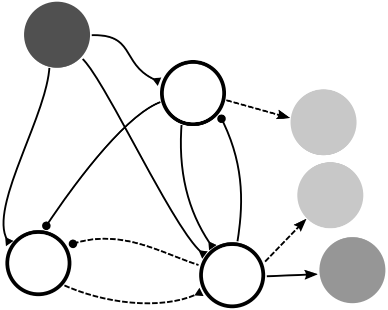
- Python-based modeling API for spiking neural networks
- Topology-centric description (data flow graph)
- Neuron and synapse dynamics (cell and synapse types)
- Experiment protocol (“what and when”)
  - stimulus (input nodes, e.g., spike trains)
  - recording (output nodes, e.g., spikes and membrane voltage)

PyNN



- Python-based modeling API for spiking neural networks
- Topology-centric description (data flow graph)
- Neuron and synapse dynamics (cell and synapse types)
- Experiment protocol (“what and when”)
  - stimulus (input nodes, e.g., spike trains)
  - recording (output nodes, e.g., spikes and membrane voltage)
- Supports different backends (e.g., NEST, NEURON, SpiNNaker, BrainScaleS)

PyNN.brainscales2 — Example



```
# ...
n1 = Population(1, HXNeuron())
n2 = Population(1, HXNeuron())
n3 = Population(1, HXNeuron())
n1.record('spikes')
n3.record(['v', 'spikes'])
Projection(n1, n3, AllToAllConnector)
Projection(n3, n1, AllToAllConnector, receptor_type='inh')
Projection(n1, n2, AllToAllConnector, receptor_type='inh')
Projection(n2, n3, AllToAllConnector, synapse_type=XYZPlastic)
Projection(n3, n2, AllToAllConnector, synapse_type=XYZPlastic,
           receptor_type='inh')
stim = Population(1, SpikeSourceArray(...))
Projection(stim, n1, AllToAllConnector)
Projection(stim, n2, AllToAllConnector)
Projection(stim, n3, AllToAllConnector)
# ...
```

Hidden workflow

1. **Collab submits experiment** to the neuromorphic central job queuing service

3. **PyNN script starts** in a containerized environment

    - Triggers **"hardware run"**, reads back results and transforms them into PyNN data structures

7. **Collab accesses result data**

~~Hidden~~ workflow

1. **Collab submits experiment** to the neuromorphic central job queuing service
   - metadata is checked (in particular: hardware quota)
2. UHEI queue runner pulls jobs from the central job queue
   - Request access to hardware resources (conventional and neuromorphic)
   - As soon as resources are available: job gets scheduled to a cluster node
3. **PyNN script starts** in a containerized environment
4. Lower software layer:
   - Initializes network connection to the hardware setup
   - Compiles initial experiment configuration: Network topology, initial parameters
   - Compiles dynamic experiment components: External stimulus, timed (re)configuration (e.g., recording properties, readout of weights)
   - Upload of both “parts” onto the system (prebuffering)
   - Triggers **“hardware run”**, reads back results and transforms them into PyNN data structures
5. PyNN code accesses result data: Writing files to disk.
6. Job result state (incl. output files) are registered at central job queue
7. **Collab accesses result data**

Summary

- BrainScaleS: accelerated analog neuromorphic hardware incl. flexible plasticity
- Comprehensive software support at expert-level
- Entry-level support now under full development (cf. PyNN.brainscales2)
- Upcoming hands-on session:
  - Collab-based access to multiple BrainScaleS-2 systems
  - Introduction to basic properties of analog neuromorphic hardware: Membrane dynamics, Stimulus, Recording
- Example experiments soon available (cf. HBP Neuromorphic Guidebook)

Team BrainScaleS