

BrainScaleS

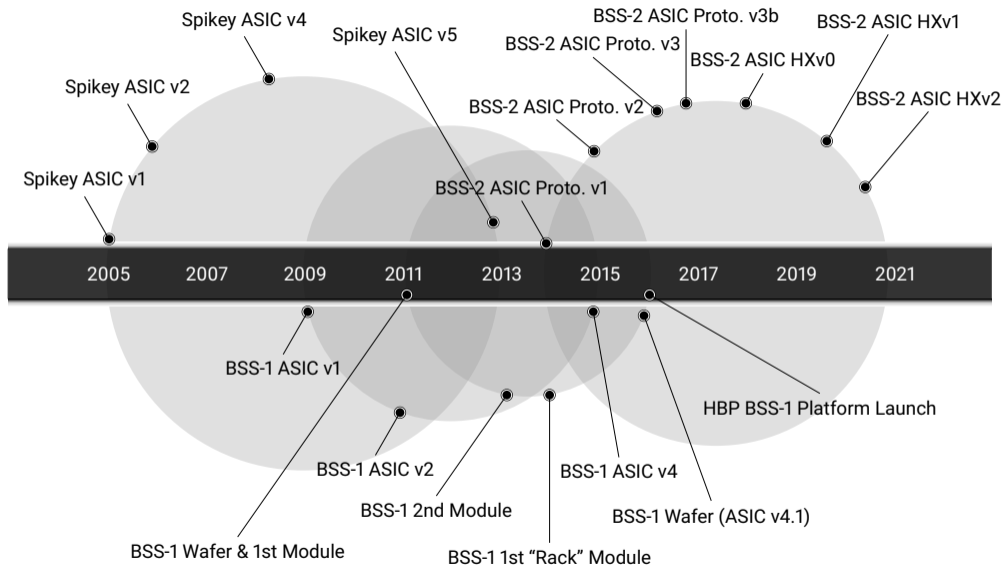
Development Methodologies and Operating System

Eric Müller

on behalf of Electronic Vision(s)

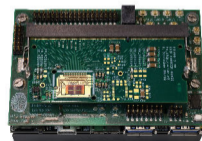
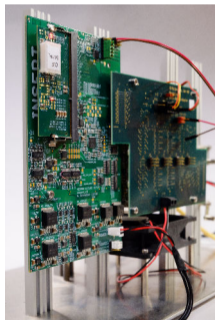
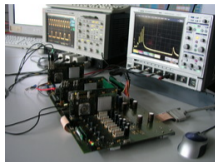


BrainScaleS – Hardware Timeline



BrainScaleS Architectures & Systems – Software Timeline

- ▶ Spikey
 - ▶ targets mobile single-chip systems, some multi-chip capabilities
 - ▶ Active 05/2006 – 10/2017
- ▶ BrainScaleS-1
 - ▶ targets wafer-scale systems
 - ▶ Active since 09/2009
- ▶ BrainScaleS-2
 - ▶ targets mobile & wafer-scale systems
 - ▶ Active since 03/2015



Overview: Software Tasks⁰

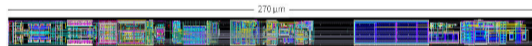
- ▶ Hardware/software co-development
(*not covered*)



gerrit_HICANN-X_v1_cosim



gerrit_HICANN-X_v2_cosim



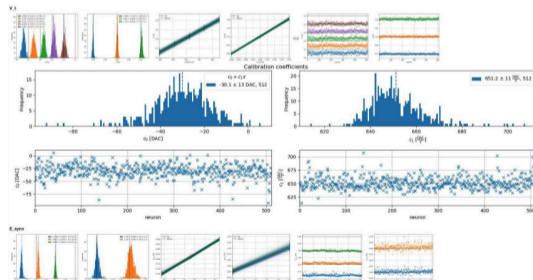
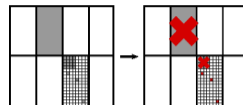
⁰work by O. J. Breitwieser, S. Schmitt, C. Mauch, P. Spilger, Y. Stradmann, H. Schmidt, J. Montes, A. Emmel, M. Czierlinski, J. Kaiser, S. Billaudelle, F. Ebert, M. Güttler, J. Ilmberger, A. Leibfried, J. Weis and many more.

Overview: Software Tasks⁰

- ▶ Hardware/software co-development
- ▶ Hardware commissioning, testing, verification and characterization (*not covered*)

All Tests

Package	Duration	Fail	err	Skp	err	Pass	err	Total	err
(over)	2 min 57 sec	0	0	0	0	1029	0	1029	0
ibms_test_v1_v1	0 ms	0	0	0	0	1	0	1	0
ibms_test_v1_v2	0 ms	0	0	0	0	1	0	1	0
pphalls_test_v1_v1	97 ms	0	0	0	0	4	0	4	0
pphalls_test_v1_v2	99 ms	0	0	0	0	4	0	4	0
pphalls_test_v1_v3	49 ms	0	0	0	0	18	0	18	0
pphalls_test_v1_v4	43 ms	0	0	0	0	19	0	19	0
pphalls_jlgpawobot_v1_v1	2 ms	0	0	0	0	4	0	4	0
pphalls_jlgpawobot_v1_v2	2 ms	0	0	0	0	4	0	4	0
pphalls_smbot_v1_v1	8 ms	0	0	0	0	1	0	1	0
pphalls_smbot_v1_v2	6 ms	0	0	0	0	1	0	1	0



⁰work by O. J. Breitwieser, S. Schmitt, C. Mauch, P. Spilger, Y. Stradmann, H. Schmidt, J. Montes, A. Emmel, M. Czielinski, J. Kaiser, S. Billaudelle, F. Ebert, M. Güttler, J. Ilmberger, A. Leibfried, J. Weis and many more.

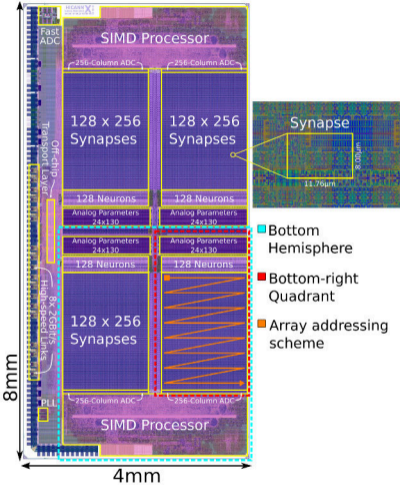
Overview: Software Tasks⁰

- ▶ Hardware/software co-development
- ▶ Hardware commissioning, testing, verification and characterization
- ▶ Hardware usage (conducting experiments)
 - ▶ **Low abstraction level**



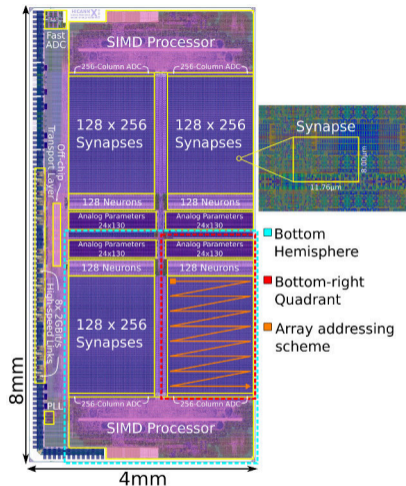
⁰work by O. J. Breitwieser, S. Schmitt, C. Mauch, P. Spilger, Y. Stradmann, H. Schmidt, J. Montes, A. Emmel, M. Czierlinski, J. Kaiser, S. Billaudelle, F. Ebert, M. Güttler, J. Ilmberger, A. Leibfried, J. Weis and many more.

Task: Low abstraction level



Task: Low abstraction level¹

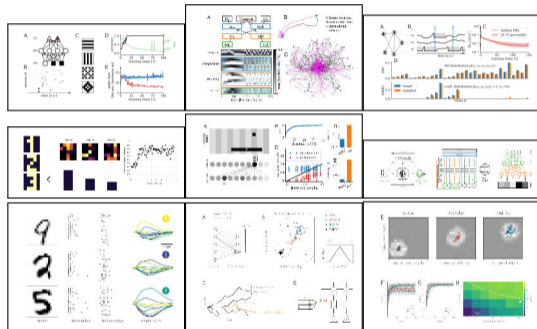
```
def configure_synapses(*args):  
    """  
    Configure routing crossbar, PADI bus, synapse drivers,  
    and parts of the synapse array.  
    """  
  
    fisch_builder = fisch.PlaybackProgramBuilder()  
    fisch_builder.write(anncore_center_ba, fisch.Omnibus(0xffff))  
    config_builder.merge_back(fisch_builder)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



¹E. Müller, S. Schmitt, C. Mauch, et al. "The Operating System of the Neuromorphic BrainScaleS-1 System". In: arXiv preprint (Mar. 2020). arXiv: 2003.13749 [cs.NE], E. Müller, C. Mauch, P. Spilger, O. J. Breitwieser, et al. "Extending BrainScaleS OS for BrainScaleS-2". In: arXiv preprint (Mar. 2020). arXiv: 2003.13750 [cs.NE].

Task: Low abstraction level

```
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)
```



... fits experts!

Overview: Software Tasks⁰

- ▶ Hardware/software co-development
- ▶ Hardware commissioning, testing, verification and characterization
- ▶ Hardware usage (conducting experiments)
 - ▶ Low abstraction level
 - ▶ **High(er)-level usage**



⁰work by O. J. Breitwieser, S. Schmitt, C. Mauch, P. Spilger, Y. Stradmann, H. Schmidt, J. Montes, A. Emmel, M. Czierlinski, J. Kaiser, S. Billaudelle, F. Ebert, M. Güttler, J. Ilmberger, A. Leibfried, J. Weis and many more.

Task: High(er)-level usage

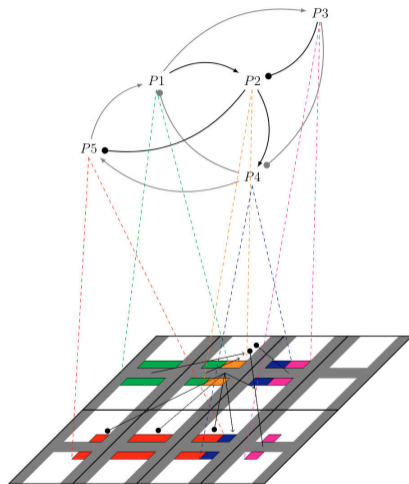
- ▶ **User-defined (initial) experiment configuration¹**
 - ▶ Neural network topology
 - ▶ Plasticity rules (synapse-dynamical, structural, homeostatic)
 - ▶ Code generation for the embedded processors (*not covered*)
 - ▶ Structured host-PPU communication (*not covered*)
 - ▶ Static and dynamic I/O (closed-loop setups, interaction w/ environment, agents)
 - ▶ Parameterization (incl. parameter domain translation)

→ Place & route task
- ▶ Hardware resource access
- ▶ Experiment run control

¹E.g. initial PyNN.brainscales2 implementation by M. Czierlinski, P. Spilger.

Task: High(er)-level usage – Place and Route²

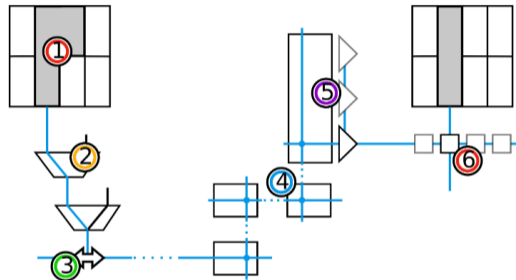
- Map neural network graphs to hardware



²F. C. Passenberg. "Improving the BrainScaleS-1 place and route software towards real-world waferscale experiments". Master thesis. Ruprecht-Karls-Universität Heidelberg, 2019.

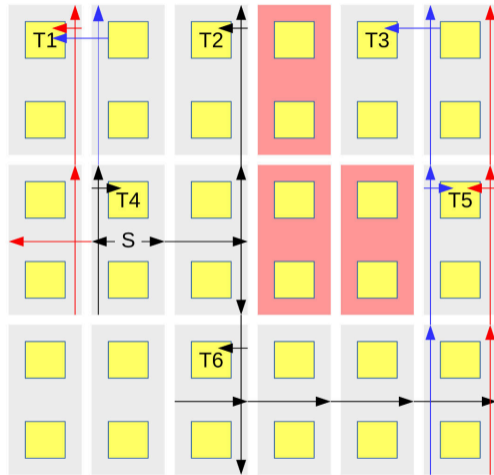
Task: High(er)-level usage – Place and Route

- ▶ Map neural network graphs to hardware
- ▶ Inhomogeneous substrate
- ▶ On-wafer communication on BSS-1 is circuit-switched



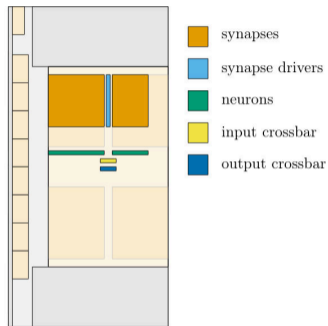
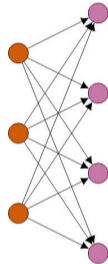
Task: High(er)-level usage – Place and Route

- ▶ Map neural network graphs to hardware
- ▶ Inhomogeneous substrate
- ▶ On-wafer communication on BSS-1 is circuit-switched
- ▶ Routing algorithms



Task: High(er)-level usage – Place and Route for ANNs²

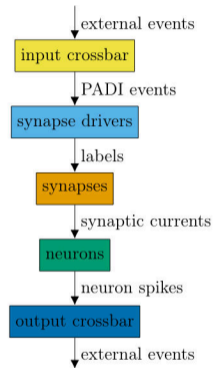
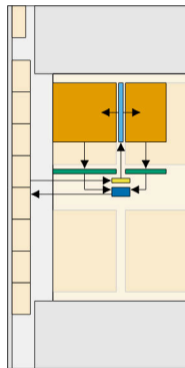
- ▶ BSS-2 supports non-spiking operation (analog inference engine)



²P. Spilger. "From Neural Network Descriptions to Neuromorphic Hardware – A Signal-Flow Graph Compiler Approach". Master thesis. Ruprecht-Karls-Universität Heidelberg, 2021, P. Spilger, E. Müller, et al. "hxtorch: PyTorch for BrainScaleS-2 – Perceptrons on Analog Neuromorphic Hardware". In: Proceedings of the Workshop on IoT, Edge, and Mobile for Embedded Machine Learning (ITEM)/ECML-PKDD 2020. June 2020. arXiv: 2006.13138 [cs.NE].

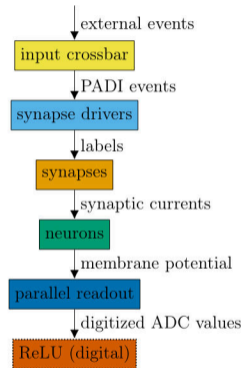
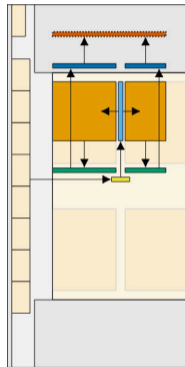
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware?



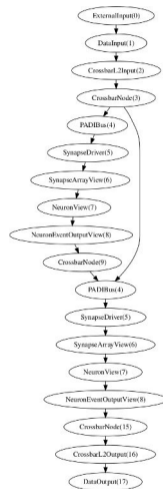
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware



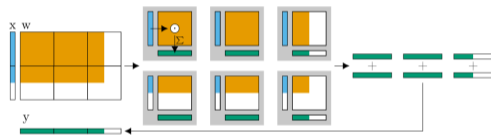
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”



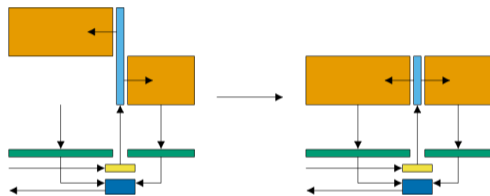
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ partitioning of large problems



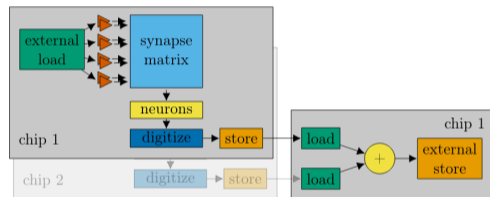
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems



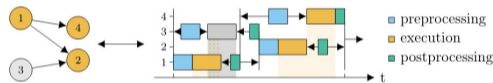
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems
- ▶ Execution dependency graph



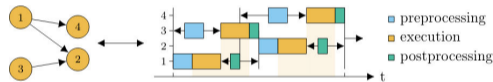
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems
- ▶ Execution dependency graph
- ▶ Scheduling of dependency graph to hardware resources



Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems
- ▶ Execution dependency graph
- ▶ Scheduling of dependency graph to hardware resources



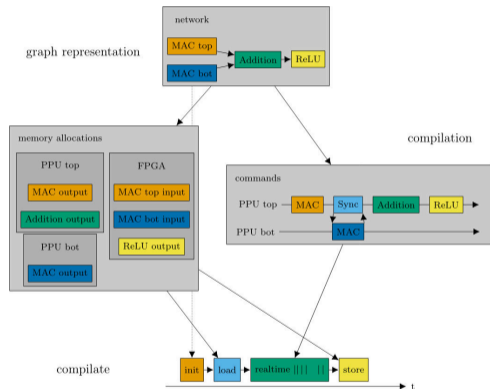
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems
- ▶ Execution dependency graph
- ▶ Scheduling of dependency graph to hardware resources
- ▶ Integration in PyTorch



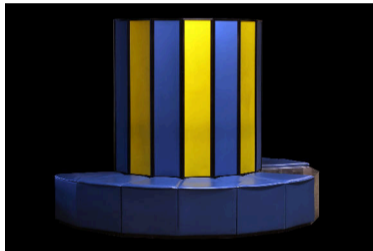
Task: High(er)-level usage – Place and Route for ANNs

- ▶ BSS-2 supports non-spiking operation (analog inference engine)
- ▶ Data flow graph for BSS hardware
- ▶ The BSS-2 “IR”
- ▶ Usage optimization & partitioning of large problems
- ▶ Execution dependency graph
- ▶ Scheduling of dependency graph to hardware resources
- ▶ Integration in PyTorch
- ▶ Support for standalone inference



Overview: Software Tasks⁰

- ▶ Hardware/software co-development
- ▶ Hardware commissioning, testing, verification and characterization
- ▶ Hardware usage (conducting experiments)
 - ▶ Low abstraction level
 - ▶ High(er)-level usage
 - ▶ **Platform operation**



⁰work by O. J. Breitwieser, S. Schmitt, C. Mauch, P. Spilger, Y. Stradmann, H. Schmidt, J. Montes, A. Emmel, M. Czierlinski, J. Kaiser, S. Billaudelle, F. Ebert, M. Güttler, J. Ilmberger, A. Leibfried, J. Weis and many more.

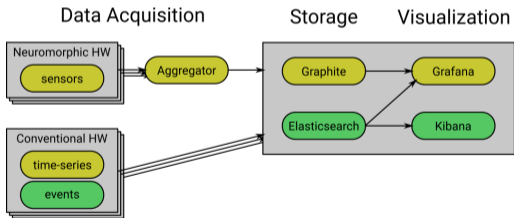
Tasks: Platform Operation

- ▶ **Resource monitoring & alerting**



Tasks: Platform Operation – Resource Monitoring & Alerting²

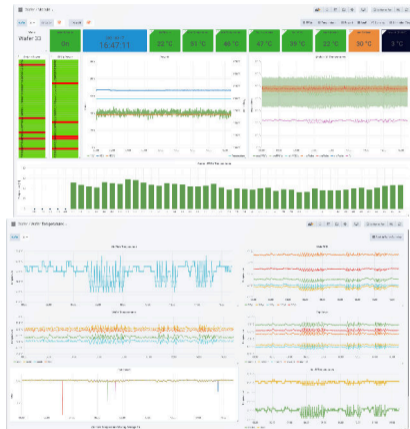
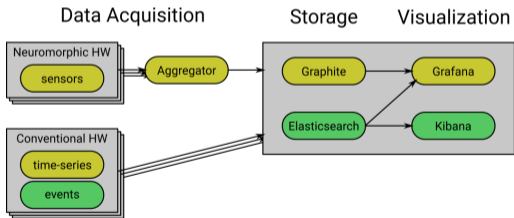
- ▶ Sensors (e.g. temp, currents, voltages, ...)
- ▶ Events (e.g. power or experiment state changes, ...)
- ▶ Alerts based on events or violations to operating range



²work by M. Güttler, C. Mauch.

Tasks: Platform Operation – Resource Monitoring & Alerting²

- ▶ Sensors (e.g. temp, currents, voltages, ...)
- ▶ Events (e.g. power or experiment state changes, ...)
- ▶ Alerts based on events or violations to operating range



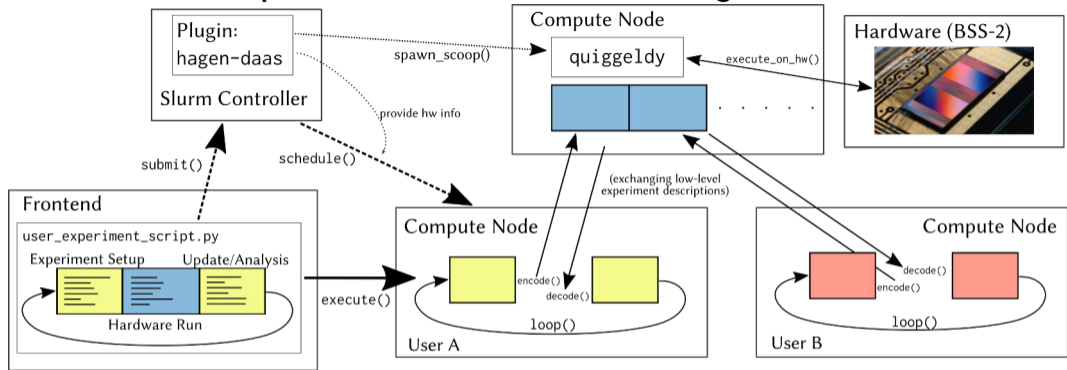
²work by M. Güttler, C. Mauch.

Tasks: Platform Operation

- ▶ Resource monitoring & alerting
- ▶ **Resource management**



Tasks: Platform Operation – Resource Management³

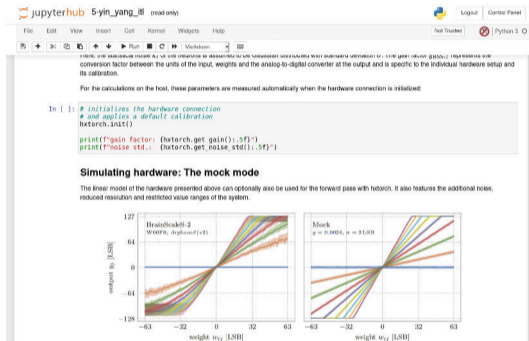


- ▶ SLURM + plugin for neuromorphic resource selection
- ▶ Custom experiment scheduler
- ▶ Used in our hands-on demos: 27k (32 active users) + 41k (42 active users) experiments scheduled to 8 single-chip setups

³work by O. J. Breitwieser, C. Mauch.

Tasks: Platform Operation

- ▶ Resource monitoring & alerting
- ▶ Resource management
- ▶ Neuromorphic hardware as a service



The screenshot shows a JupyterLab environment with the following content:

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Python 3.10.12 Shell (base) [~/] x86_64 Linux

conversion factor between the units of the input, weights and the analog-to-digital converter at the output and is specific to the individual hardware setup and its calibration.

For the calculations on the host, these parameters are measured automatically when the hardware connection is initialized:

```
In [ ]: # initializes the hardware connection
# and applies a default calibration
hxtorch.init()

print(f"gain factor: {hxtorch.get_gain():.5f}")
print(f"noise std.: {hxtorch.get_noise_std():.5f}")
```

Simulating hardware: The mock mode

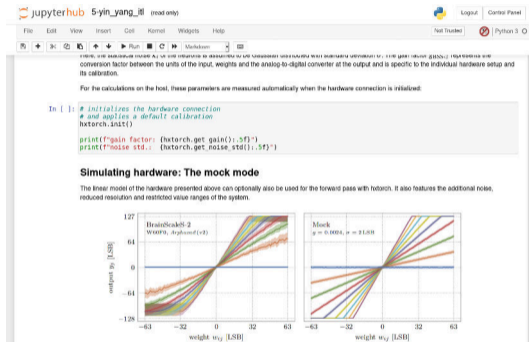
The linear model of the hardware presented above can optionally also be used for the forward pass with hxtorch. It also features the additional noise, reduced resolution and restricted value ranges of the system.

Two plots are shown side-by-side:

- BrainScale-2**: $W(0.0, \text{dtype}=(\nu, 2))$. The plot shows multiple colored lines representing different runs, all passing through the origin (0,0). The x-axis is labeled "weight w_{ij} [LSB]" and ranges from -63 to 63. The y-axis is labeled "output y_{ij} [LSB]" and ranges from -128 to 127.
- Mock**: $\sigma = 0.0004, \mu = 21.158$. The plot shows multiple colored lines representing different runs, all passing through the origin (0,0). The x-axis is labeled "weight w_{ij} [LSB]" and ranges from -63 to 63. The y-axis is labeled "output y_{ij} [LSB]" and ranges from -128 to 127.

Tasks: Platform Operation

- ▶ Resource monitoring & alerting
- ▶ Resource management
- ▶ Neuromorphic hardware as a service
 - ▶ Reproducibility
 - ▶ Sustainability



The screenshot shows a JupyterLab environment with the following content:

conversion factor between the units of the input, weights and the analog-to-digital converter at the output and is specific to the individual hardware setup and its calibration.

For the calculations on the host, these parameters are measured automatically when the hardware connection is initialized:

```
In [ ]: # initializes the hardware connection
# and applies a default calibration
hxtorch.init()

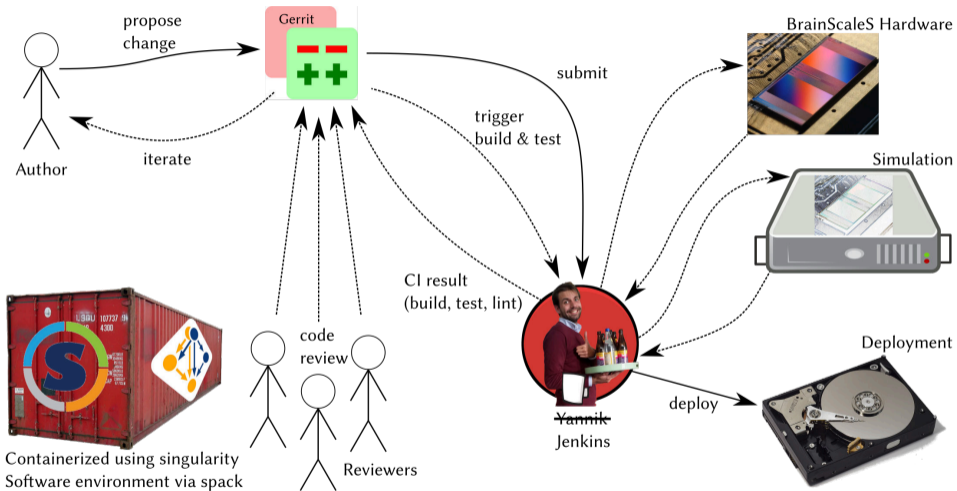
print(f"gain factor: {hxtorch.get_gain():.5f}")
print(f"noise std.: {hxtorch.get_noise_std():.5f}")
```

Simulating hardware: The mock mode

The linear model of the hardware presented above can optionally also be used for the forward pass with hxtorch. It also features the additional noise, reduced resolution and restricted value ranges of the system.

Two plots are shown side-by-side, both with 'weight w_{ij} [LSB]' on the x-axis (ranging from -63 to 63) and 'output y_{ij} [LSB]' on the y-axis (ranging from -128 to 127). The left plot is titled 'BrainScaleS-2' with parameters 'W00F0, dphi=0.01(x2)'. It shows multiple colored lines representing different runs, which are slightly noisy and have a limited range of output values. The right plot is titled 'Mock' with parameters ' $\sigma = 0.0004, \mu = 21.158$ '. It shows multiple colored lines that are much smoother and more linear, representing a simulated version of the hardware.

Development Methodologies: CR, CI, CD, Co-Dev. & Containers⁴



⁴work by Y. Stradmann, O. J. Breitwieser, A. Baumbach and others.

Development Methodologies: CR, CI, CD, Co-Dev. & Containers

Covered by this workflow:

- ▶ Software development for BrainScaleS
 - ▶ RTL (FPGA and ASIC): digital testing against simulation
- ▶ Modifications to the containerized environment
 - ▶ incl. external software dependencies
- ▶ Infrastructure (neuromorphic resources)
- ▶ Models, experiments and publications
- ▶ Executable documentation (cf. BrainScaleS hands-on at NICE)
- ▶ (Not yet integrated: full-software-stack-full-chip simulation)

Summary


- ▶ Long-lived neuromorphic hardware architectures
→ technical debt
- ▶ Different aspects of hardware usage and “top-level” use cases
→ users have different views and requirements
- ▶ BrainScaleS-2 introduces more operation modes (standalone, ANNs)
→ minimize development overhead
- ▶ During hardware commissioning resources are scarce and large-scale installations are stationary
→ time sharing, resource management and monitoring
- ▶ Large(r)-scale usage (many users and/or platform operation)
→ benefits from efforts towards reproducibility and sustainability

Electronic Vision(s)



Backup

Open Source — <https://github.com/electronicvisions>



Electronic Vision(s) Group

Kirchhoff-Institute for Physics, Ruprecht-Karls-Universität Heidelberg

📍 Heidelberg, Germany <http://www.kip.uni-heidelberg.de/vision>

[📁 Repositories](#) 97 [📦 Packages](#) [👤 People](#) 27 [👥 Teams](#) 7 [⚙️ Settings](#)

Pinned repositories

[Customize pinned repositories](#)

[📁 hxtorch](#) ⋮

BrainScaleS-2 via PyTorch

🔴 C++ ☆ 1

[📁 pynn-brainscales](#) ⋮

BrainScaleS-2 via PyNN

🔵 Python ☆ 1

[📁 hbp-sp9-guidebook](#) ⋮

HBP Neuromorphic Computing Platform Guidebook

🟡 JavaScript ☆ 5 🍷 14

🔍 Find a repository... [Type](#) [Language](#) [Sort](#) [New](#)

brainscales2-demos

Demos, examples and teaching material for BrainScaleS-2

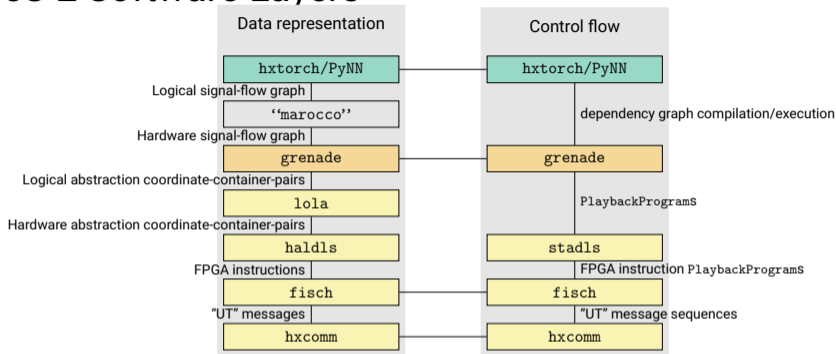
🔵 Python 🍷 0 ☆ 0 ⓘ 0 🍷 0 Updated 15 hours ago

Top languages

🔴 C++ 🔵 Python ⬤ C

🟠 Jupyter Notebook 🟡 JavaScript

BrainScaleS-2 Software Layers



- ▶ Layers encapsulate aspects of the system: e.g. communication, FPGA “programs”, configurable units, aggregate chip view, signal graphs → one change in hardware should only affect one corresponding layer.
- ▶ “Modern” C++, running on x86_64 and ARM64, some parts build on BSS-2 embedded processors (PPC)
- ▶ Possibility for serialization provided at each layer transition
- ▶ Most layers provide complete Python wrapping