Bridging the Neuromorphic Simulation-to-Hardware-Gap with YANA

Brian Pachideh[1,2], Moritz Neher[1,2,3], Sven Nitzsche[1,2], Jürgen Becker[1,2]

[1] FZI Research Center for Information Technology, Karlsruhe, Germany
[2] Karlsruhe Institute of Technology , Karlsruhe, Germany
[3] Infineon Technologies , Dresden, Germany

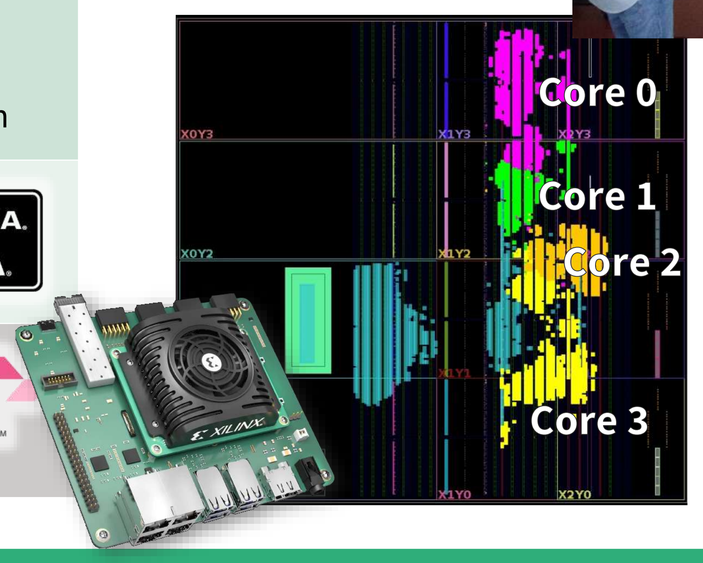
Neuromorphic computing still faces a persistent simulation-to-hardware gap: while many groups can prototype algorithms in software, far fewer can iterate through full hardware/software co-design because accelerator IP, deployment tooling, and hardware-specific runtimes are often closed or fragmented.

We present YANA as a readily available, open-source framework that lowers this barrier by providing an end-to-end stack for event-driven SNN acceleration on affordable FPGA development platforms.

YANA Open Source Development Stack:

Stack Layer	Integrated Tools
SNN Training	Tonic (Dataloader) · Norse (SNN Deep Learning)
FZI Custom HW-aware Optimization (e.g. Quantization, Pruning)	
Intermediate Representation	NIR — **N**euromorphic **I**ntermediate **R**epresentation
FZI Custom Partition & Map (“compilation” of SNN)	NVIDIA CUDA
FZI Runtime & Inference Hardware	ZYNQ UltraSCALE+ · PYNQ

Ready to use:
Tutorials at NICE’25 and HEART’26



Deployment of four cores (wip):

- 4k LIF Neurons
- 262k Synapses
- 2 MB SRAM utilized
- @200 MHz

YANA for the Neuromorphic Open Ecosystem

Addressed Challenges
- Limited SNN accelerator access
- Fragmented toolchains
- Simulation-only SNN evaluation
- Difficult hardware benchmarking
- Slow HW/SW co-design iterations

Drive HW/SW Co-Design
- Train with hardware constraints
- Measure real execution latency
- Explore sparse SNN architectures
- Build reproducible benchmarks

Sparse & Event-Driven
- Spike-driven execution and on-chip communication
- Sparse & recurrent SNNs
- Exploits temporal, spatial and weight sparsity

GitHub

Open and Extensible
- Open-source RTL and software
- Integration with existing SNN tools
- NIR-based model interchange
- Extensible for future SNN architectures

Next Steps
- Event stream processing & tight integration with event camera
- Highly parallel core deployments

</> Terminal

```
❯ yana/runtime/cli -f run_shd_pruned.yana
yana> set bitstream yana_kr260_i.xsa
bitstream = yana_kr260_i.xsa
yana> connect 192.168.10.2
connected to 192.168.10.2:5005
yana> experiment shd/ shd_pruned/
· running all sample(s) from 2 experiment(s) (output core (1,1))
+--------------+-----------+------------+--------------+----------+---------------------+----------+-----------+
| Experiment   |   Samples | Accuracy   | StateMatch   |   Cycles |   AvgLat/sample(ms) |   Events |   Weights |
+==============+===========+============+==============+==========+=====================+==========+===========+
| shd          |        10 | 90.0%      | 100.0%       |  1662911 |              0.8315 |    30798 |      2319 |
+--------------+-----------+------------+--------------+----------+---------------------+----------+-----------+
| shd_pruned   |        10 | 60.0%      | 100.0%       |   559501 |              0.2798 |    30798 |      2001 |
+--------------+-----------+------------+--------------+----------+---------------------+----------+-----------+
disconnected
```

Gefördert durch:

Bundesministerium für Forschung, Technologie und Raumfahrt

Kennzeichen 16ME1082

HEICHIPS SUMMER SCHOOL 2026
HEIDELBERG UNIVERSITY, AUGUST 3RD TO 7TH