Catalyst N1: A 131K-Neuron Open Neuromorphic Processor with Programmable Synaptic Plasticity and FPGA Validation

Henry Arthur Shulayev Barnes
School of Natural and Computing Sciences
University of Aberdeen
Aberdeen, AB24 3UE, United Kingdom
u13hs24@abdn.ac.uk

Abstract

I present **Catalyst N1**, an open neuromorphic processor architecture comprising 128 cores, each containing 1,024 current-based leaky integrate-and-fire (CUBA LIF) neurons and 131K compressed sparse row (CSR) synapses. The design implements a fully programmable microcode learning engine with 16 registers and 14 opcodes, supporting spike-timing-dependent plasticity (STDP), three-factor eligibility-modulated reward learning, and homeostatic weight normalization. Each neuron features 24-bit state precision, dendritic compartment trees with configurable join operations, dual spike traces, stochastic threshold noise via per-neuron linear feedback shift registers, and per-synapse programmable delays up to 63 timesteps. Three synapse encoding formats—sparse, dense, and population-coded—reduce memory footprint for structured connectivity. A triple RV32IMF RISC-V embedded processor cluster provides on-chip supervisory control. Inter-core communication supports both barrier-synchronized mesh and asynchronous packet-routed network-on-chip (NoC) topologies. An accompanying software development kit offers a Python network builder, density-weighted compiler, cycle-accurate CPU simulator, GPU-accelerated simulator achieving 100–1000$\times$ speedup via PyTorch sparse CSR operations, and a hardware backend targeting Xilinx FPGAs. I validate a 16-core instance on an AWS F2 VU47P at 62.5 MHz neuromorphic clock (250 MHz PCIe interface) with dual-clock CDC. RTL support for a 4-core Arty A7-100T configuration is included but not yet physically validated. Across 25 RTL testbenches covering 98 test scenarios, Catalyst N1 achieves zero failures. On the Spiking Heidelberg Digits (SHD) spoken digit classification benchmark, a surrogate gradient-trained recurrent SNN deployed with 16-bit weight quantization achieves 85.9% test accuracy on this challenging temporal task. Feature comparison demonstrates architectural feature parity with Intel's Loihi 1, with graded spike support extending into Loihi 2 territory.

1 Introduction

Biological neural systems process information with remarkable energy efficiency. A human cortex, operating on roughly 20 watts, performs real-time sensory processing, motor control, and abstract reasoning that still outpaces the largest datacenters on many tasks. This efficiency stems from an event-driven computational paradigm: neurons communicate through discrete spikes, consuming energy only when events occur. Neuromorphic processors attempt to capture this principle in silicon.

The case for neuromorphic hardware has strengthened considerably in recent years. Deep learning workloads now dominate datacenter power budgets, with training runs for frontier models exceeding 10 GWh [6]. Spiking neural networks (SNNs), by contrast, promise orders-of-magnitude improvements in inference energy by exploiting temporal sparsity—a neuron that does not fire consumes negligible power. Edge deployment scenarios make this advantage particularly compelling: always-on sensor processing, autonomous navigation, and prosthetic control all demand low-power, low-latency computation that conventional GPUs cannot economically deliver.

Several pioneering architectures have demonstrated the viability of large-scale neuromorphic hardware. IBM's TrueNorth [2] proved that a million-neuron chip could operate at 70 mW by adopting a digital, event-driven, massively parallel design—but sacrificed on-chip learning entirely. Intel's Loihi [1] restored programmable synaptic plasticity through a microcode learning engine, demonstrating that on-chip STDP could solve constraint satisfaction and sparse coding problems without external weight updates. The University of Manchester's SpiNNaker [3] took a radically different approach, placing ARM processors on a custom interconnect to provide maximum flexibility at the cost of energy efficiency. BrainScaleS from Heidelberg [4] operates in the analog domain, achieving 10,000$\times$ biological real-time speedup but facing challenges with transistor mismatch and calibration.

Despite this progress, a gap persists: *no open, FPGA-validated neuromorphic architecture exists that matches the feature completeness of a production chip like Loihi while providing a full software stack for SNN research.* Loihi's architecture is proprietary and available only through Intel's cloud service. TrueNorth lacks learning. SpiNNaker's ARM-based approach incurs overhead for the neuron update loop. Academic FPGA implementations typically demonstrate proof-of-concept networks with hundreds of neurons but rarely approach the feature completeness needed for serious SNN research.

Contributions. I address this gap with Catalyst N1, an open neuromorphic processor architecture achieving full architectural feature parity with Intel's Loihi 1, validated on FPGA, and accompanied by a complete software stack. The specific contributions are:

- A 128-core neuromorphic processor with 1,024 CUBA LIF neurons and 131K CSR synapses per core, matching Loihi 1's scale while adding graded spike payloads (a Loihi 2 feature).
- A programmable microcode learning engine with 16 registers and 14 opcodes, supporting STDP, three-factor eligibility-modulated learning, and stochastic trace rounding.
- A triple RV32IMF RISC-V embedded cluster with IEEE 754 floating-point, timer interrupts, hardware breakpoints, and a shared mailbox for inter-core coordination.
- A full-stack SDK with a Python network builder, density-weighted compiler, cycle-accurate CPU simulator, GPU-accelerated simulator (100–1000$\times$ speedup), and FPGA hardware backends for both Xilinx Arty A7 and AWS F2.
- FPGA validation across 25 RTL regression testbenches (98 test scenarios, zero failures) spanning 25 development phases, plus an SHD spoken digit benchmark achieving 85.9% accuracy with surrogate gradient training and 16-bit weight quantization.

Paper organization. Section 2 details the processor architecture. Section 3 describes the software development kit. Section 4 covers FPGA implementation. Section 5 presents evaluation results including a feature comparison with Loihi 1, MNIST classification, and the SHD spoken digit benchmark. Section 6 surveys related work. Section 7 discusses limitations honestly, and Section 8 concludes.

2 Architecture

2.1 System Overview

Catalyst N1 consists of 128 neuromorphic cores arranged in a configurable interconnect, a triple RV32IMF RISC-V embedded processor cluster, a host interface supporting both UART and PCIe, and optional multi-chip link ports. Figure 1 shows the top-level block diagram.

Each core operates as an independent timestep engine. Within a timestep, computation proceeds through a fixed pipeline: delay drain, spike delivery, neuron update, and learning. Cores synchronize at timestep boundaries through either a barrier mesh (for deterministic execution) or an asynchronous packet-routed network-on-chip (for event-driven operation). The choice is a synthesis-time parameter.

2.2 Neuromorphic Core

Each core implements a CUBA (current-based) leaky integrate-and-fire neuron model operating on 24-bit fixed-point state variables. The dual-variable formulation tracks both synaptic current u and membrane voltage v:

$$u[t+1] = u[t] - \text{RAZ}\left(\frac{u[t]\cdot\delta_u}{4096}\right) + I_{\text{syn}}[t] \tag{1}$$

$$v[t+1] = v[t] - \text{RAZ}\left(\frac{v[t]\cdot\delta_v}{4096}\right) + u[t] + b \tag{2}$$

$$\text{spike} \iff v[t+1] \geq \theta + \eta[t] \tag{3}$$

where δ_u and δ_v are 12-bit unsigned decay constants, b is a bias term encoded with mantissa-exponent compression, θ is the programmable firing threshold, and $\eta[t]$ is optional stochastic noise. The RAZ (round-away-from-zero) function ensures that small decay products always make progress toward zero, preventing neurons from becoming stuck at non-resting potentials. This matches the behavior described in the Loihi specification [1].

When a neuron fires, the excess voltage above threshold is encoded as an 8-bit graded spike payload:

$$\text{payload} = \min\Big(255,\ \max\big(1,\ v[t+1]-\theta\big)\Big) \tag{4}$$

The neuron then enters a programmable refractory period during which its potential is clamped to the resting value and no further spikes can occur.

Dendritic compartments. Each neuron supports up to four compartments arranged in a tree: one somatic root (compartment 0) and three dendritic branches (compartments 1–3). Incoming synaptic current is directed to a specific compartment based on a per-synapse compartment field. Each dendritic compartment applies a nonlinear thresholded ReLU before propagating to its parent via a configurable join operation (ADD, ABS_MAX, OR, or PASS). This enables local nonlinear processing at the dendrite level, a feature shared with Loihi's compartment trees.

Stochastic threshold noise. Per-neuron noise is generated by independent 16-bit Galois linear feedback shift registers (LFSRs) with configurable mantissa-exponent amplitude. The LFSR output is masked and added to the firing threshold, introducing controlled stochasticity that can break symmetry during learning and improve exploration in reinforcement learning tasks. Three noise injection targets are supported: threshold perturbation, synaptic current, and trace updates.

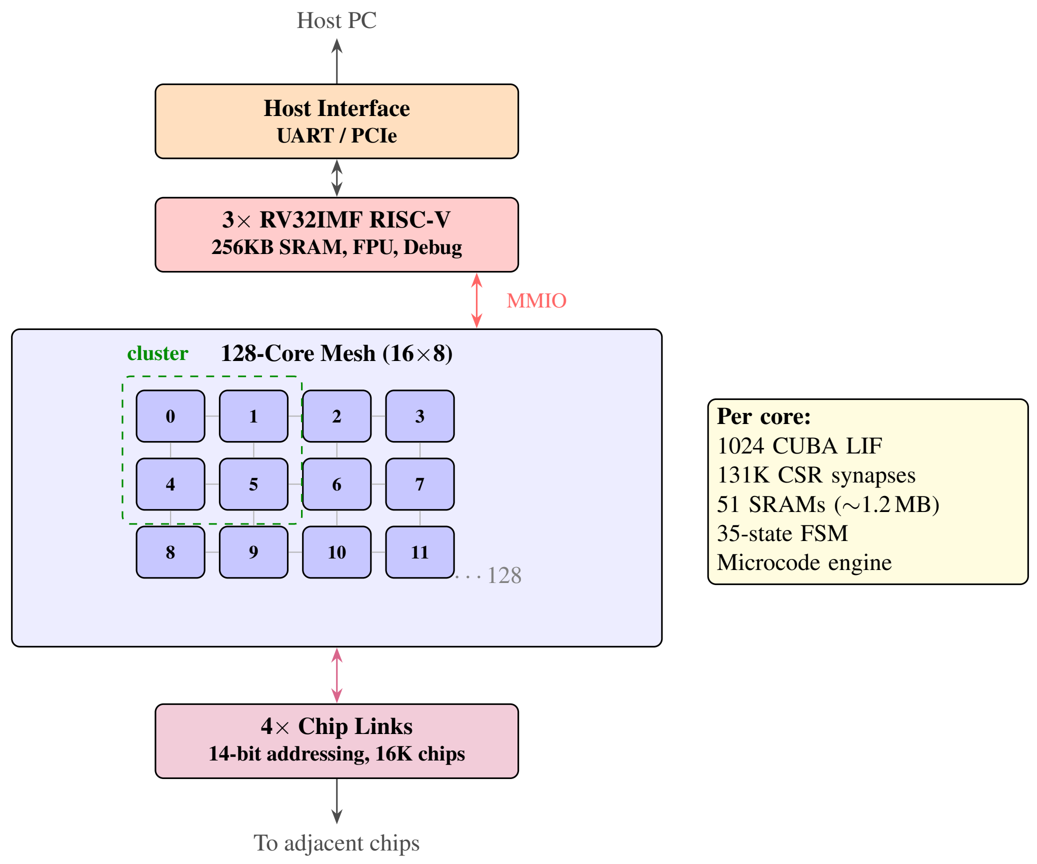


Figure 1: Catalyst N1 top-level architecture. 128 neuromorphic cores in a 16×8 mesh with barrier or async NoC. A triple RV32IMF cluster provides supervisory control. The host interface supports UART (Arty A7) and PCIe (AWS F2). Chip links enable multi-chip routing across up to 16,384 chips.

Dual spike traces. Each neuron maintains two exponential spike traces x_1 and x_2 with independently programmable time constants τ_1 and τ_2. On a spike event, both traces saturate to a maximum value; between spikes they decay exponentially with a guaranteed minimum step of 1 per timestep to prevent stalling. These traces serve as the pre- and post-synaptic variables for the learning engine.

Figure 4 summarizes the neuron update datapath.

2.3 Synaptic Connectivity

Each core contains a shared connection pool of 131,072 entries organized as a compressed sparse row (CSR) structure. A per-neuron index table stores a base address and connection count, allowing variable fanout without wasted memory. Three synapse encoding formats are supported:

- **Sparse (CSR):** Each pool entry stores an explicit target neuron ID, a 16-bit signed weight, and a 2-bit compartment selector. This is the default format and offers maximum flexibility.
- **Dense:** Target neurons are implicit (base + offset), so pool entries store only weights. This reduces per-synapse storage for structured connectivity patterns such as convolutional or fully-connected layers.
- **Population-coded:** A single shared weight serves all targets in a contiguous range. Memory usage collapses to a single pool entry regardless of fan-out, ideal for broadcast inhibition.

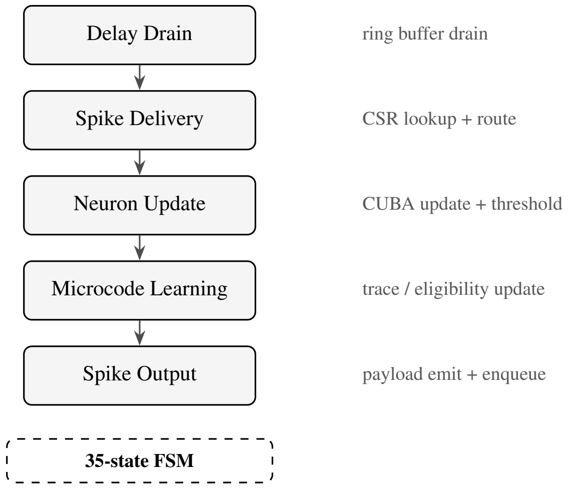
Weights are stored as 16-bit signed integers. For compact representations, a mantissa-exponent decompression scheme allows the effective dynamic range to exceed 16 bits while maintaining fixed-point arithmetic in the datapath.

Per-synapse axon delays of 0–63 timesteps are supported through a modular ring buffer. Delayed spikes are deposited into future timestep buckets during the deliver phase and drained at the start of each timestep. This enables temporal coding patterns and coincidence detection without software intervention.

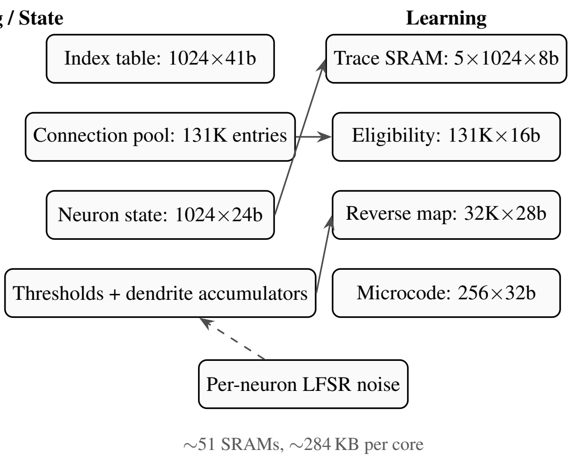
2.4 Learning Engine

The learning engine is the most architecturally significant component. Rather than hardcoding a specific plasticity rule, Catalyst N1 implements a programmable microcode engine that executes user-defined learning programs on every active synapse. The engine provides:

- **16 registers:** R_0–R_4 are pre-loaded with the five spike traces $(x_1, x_2, y_1, y_2, y_3)$, R_5 holds the current weight, R_6



(a) Core execution pipeline.



(b) Per-core memory and learning blocks.

Figure 2: Internal block diagram of a single Catalyst N1 neuromorphic core. Data flows vertically through the per-timestep pipeline: delay drain → spike delivery → neuron update → learning → spike output. A 35-state FSM sequences operations across ∼51 SRAMs totaling approximately 284 KB per core.

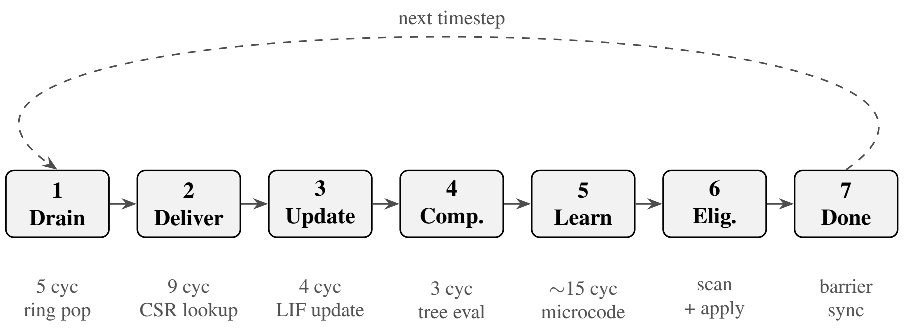


Figure 3: Per-timestep pipeline within each core. Phases execute sequentially; the learning phase is skipped when skip-idle mode is enabled and no spikes occurred during the current timestep (clock gating optimization). The barrier at step 7 synchronizes all cores before the next timestep begins.

the axon delay, R_7 a user-defined tag, R_8 the eligibility trace, R_9 the reward signal, and R_{10}–R_{15} are temporaries.

- **14 opcodes:** arithmetic (ADD, SUB, MULS, SHR, SHL, MAX, MIN), control flow (SKIP_Z, SKIP_NZ, HALT), load immediate (LOADI), and state writeback (STORE_W for weight, STORE_E for eligibility, STORE_D for delay, STORE_T for tag).
- **128 instruction slots per core:** partitioned into an LTD region (executed when the presynaptic neuron fires) and an LTP region (executed when the postsynaptic neuron fires).

This architecture directly supports classical pair-based STDP, triplet STDP [7], three-factor eligibility-modulated reward learning [8], and homeostatic plasticity—all through different microcode programs loaded at configuration time, with no RTL changes required. A default STDP program ships with the SDK.

For three-factor learning, the engine maintains per-synapse eligibility traces that accumulate spike-timing correlations. A global reward signal, injected by the host or the embedded RISC-V cluster, modulates the eligibility-to-weight conversion:

$$\Delta w = \frac{e \cdot r}{2^{R_s}} \tag{5}$$

where e is the eligibility, r is the reward signal, and $R_s = 7$ is a fixed scaling shift. Eligibility traces decay exponentially between reward events to maintain temporal credit assignment.

Stochastic rounding during trace updates provides a form of implicit regularization, preventing weight drift in the absence of correlated activity.

Learning phase timing. The learning engine processes one connection pool entry per microcode instruction cycle, scanning only entries flagged by pre- or post-synaptic spike activity (tracked via `pool_used_count`). With a 128-slot microcode program and up to 131K pool entries per core, the worst-case learning phase duration is 131K × L cycles, where L is the program length. In practice, spike sparsity limits the number of active entries to a small fraction of the pool. For idle cores (no spikes in the current timestep), the learning phase is skipped entirely via the clock-gating optimization (Section 2),

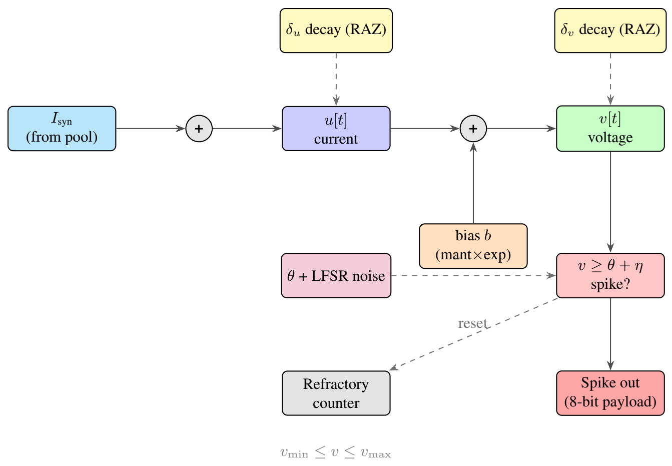


Figure 4: CUBA LIF neuron update datapath. Synaptic current I_{syn} accumulates into current variable u, which feeds into voltage v alongside a configurable bias. Both u and v undergo exponential decay via RAZ (round-away-from-zero) arithmetic. A stochastic threshold perturbation η provides controlled noise. The 8-bit graded spike payload encodes the excess voltage above threshold.

consuming zero cycles.

2.5 Network-on-Chip

Two interconnect modes are supported, selectable at synthesis time.

Barrier mesh. All cores execute timesteps in lockstep. A two-level hierarchical routing scheme groups cores into clusters of four, with 8 local route slots and 4 global route slots per source neuron. At the end of each timestep, a barrier signal propagates through the mesh before the next timestep begins. This mode provides deterministic, reproducible execution—essential for debugging and verification—and scales efficiently to approximately 128 cores before barrier latency becomes a bottleneck.

Asynchronous packet NoC. Cores operate independently, communicating through routed spike packets. XY dimension-order routing ensures deadlock freedom. A dual-network option separates forward spike traffic from backward learning signals, preventing head-of-line blocking. This mode enables fully event-driven operation where idle cores consume minimal power. The tradeoff is non-deterministic timing, which complicates debugging but better reflects biological asynchrony.

Spike packets carry the source core ID, source neuron ID, and an 8-bit payload field. Route lookup tables in each core translate source neuron spikes into destination (core, neuron, weight) tuples, supporting multicast with up to 8 local and 4 global destinations per source.

Interconnect quantitative parameters. The injection FIFO per core holds 512 spike entries; the capture FIFO holds 64 entries. In barrier mode, local route lookup completes in 3 cycles (address, wait, read); global routes add 3 cycles per mesh hop (global route address, wait, read, then forwarded into the destination core's injection FIFO). In async mode, each router processes one spike per cycle with XY dimension-order forwarding, yielding a worst-case latency of $(N_x + N_y)$ cycles for a 16×8 mesh. Maximum spike throughput per core is bounded by the injection FIFO depth and the number of active source neurons per timestep—for the 16-core F2 instance at 62.5 MHz, the measured throughput exceeds 8,600 timesteps per second with all cores simultaneously active.

2.6 Embedded Processors

A cluster of three RV32IMF RISC-V cores provides on-chip supervisory control, matching Loihi's three embedded x86 Lakemont processors. Each core implements:

- A 3-stage pipeline (fetch, decode/execute, writeback)
- RV32I base integer + M (multiply/divide) + F (single-precision IEEE 754 FPU) extensions
- 256 KB instruction SRAM and 256 KB data SRAM
- Machine-mode CSRs: `mtvec`, `mepc`, `mcause`, `mstatus`, `mie`, `mip`
- Timer interrupts (`mcycle` $\geq$ `mtimecmp`)
- 4 hardware breakpoints with halt-on-match, single-step, and resume

The three cores share access to the neuromorphic mesh through a priority-arbitrated MMIO bus. Four 32-bit shared mailbox registers enable inter-core synchronization. Typical use cases include real-time reward computation for reinforcement learning, adaptive threshold adjustment, and network reconfiguration in response to classification confidence.

2.7 Multi-Chip Scaling

A multi-chip router module provides four serial link ports using dimension-order routing across a 2D chip mesh. Each link carries serialized spike packets with 14-bit chip addressing, supporting up to 16,384 chips. Four message types are defined: spike (normal operation), barrier (cross-chip synchronization), management (RISC-V register access across chips), and preempt (priority interrupt). Link serialization uses a generic shift-register approach that adapts to any configured address width.

While the multi-chip infrastructure is fully implemented and verified in simulation, physical multi-chip operation has not yet been demonstrated on FPGA due to board availability constraints.

3 Software Development Kit

A Python-based SDK (Figure 5) provides the complete workflow from network specification through compilation to execution on any of three backends.

3.1 Network Builder API

Networks are constructed from `Population` and `Connection` objects. Each population specifies a neuron count, optional per-neuron parameters (threshold, leak rate, resting potential, refractory period, noise configuration, trace time constants), and a human-readable label. Connections between populations specify a topology, weight, optional delay, synapse format, and target compartment.

Five built-in topology generators cover common connectivity patterns:

- `all_to_all`: Complete bipartite graph.
- `one_to_one`: Identity mapping (requires equal sizes).
- `random_sparse`: Erdős-Rényi with probability p.
- `fixed_fan_in`: Each target receives exactly k random sources.
- `fixed_fan_out`: Each source projects to exactly k random targets.

A custom microcode learning rule can be attached to any network via the `LearningRule` class. Rules can be constructed programmatically, loaded from instruction words, or assembled from text mnemonics:

```
rule = LearningRule()
rule.assemble_ltd("""
    SHR R5, R0, 3    ; delta = trace >> 3
    SKIP_Z R5         ; skip if zero
    SUB R2, R2, R5    ; weight -= delta
    STORE_W R2         ; write back
    HALT
""")
net.set_learning_rule(rule)
```

3.2 Compiler

The compiler transforms a logical `Network` into physical hardware commands through three phases.

Placement. Populations are sorted by descending connection density and packed greedily into cores. This density-weighted heuristic co-locates heavily connected populations, minimizing inter-core traffic and reducing pressure on the multicast route tables.

CSR allocation. For each core, connections are grouped by source neuron and bump-allocated into the 131K-entry pool. The compiler selects the most efficient synapse format automatically: population-coded for uniform fan-out with shared weights, dense for structured connectivity, and sparse as the general fallback. Pool overflow raises a compile-time error with a diagnostic message identifying the overflowing core and neuron.

Routing. Inter-core connections are split into intra-cluster (local) and inter-cluster (global) routes. Each source neuron may use up to 8 local and 4 global route slots. The compiler emits the corresponding route table programming commands, along with delay configuration and microcode loading.

The output is a `CompiledNetwork` object containing all hardware commands plus a simulator-friendly adjacency representation, enabling seamless switching between backends.

3.3 Backends

All backends implement a common `Backend` interface: `deploy()`, `inject()`, `run()`, `reward()`, `status()`, and `close()`.

CPU Simulator. A cycle-accurate Python simulator matches the RTL pipeline phase by phase. Both synchronous (barrier) and asynchronous (event-driven with quiescence detection) modes are supported. The simulator reproduces every hardware behavior: RAZ arithmetic, LFSR noise, dual trace decay, delay ring buffers, microcode execution, eligibility modulation, and dendritic compartment propagation. It serves as the reference implementation for RTL verification.

GPU Simulator. For large networks, a PyTorch-based GPU simulator stores neuron state as dense `int32` tensors and connectivity as sparse CSR `float32` matrices. Spike delivery reduces to a single sparse matrix-vector multiplication:

$$I_{\text{syn}} = \mathbf{W} \cdot \mathbf{s}[t-1] \tag{6}$$

where $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the weight matrix (target $\times$ source) and $\mathbf{s}$ is the spike vector. Neuron updates, trace decay, threshold noise, and STDP weight modifications are fully vectorized. For custom microcode learning rules, the GPU simulator falls back to CPU-side interpretation—this hybrid approach preserves correctness while still achieving 100–1000$\times$ speedup for the dominant sparse-matmul phase.

A `run_with_schedule()` method accepts pre-computed per-timestep stimulus tensors, eliminating Python-loop overhead for training workloads. Spike counts accumulate on GPU and transfer to CPU only at the end of the run.

Hardware backends. Two FPGA backends provide identical APIs. The Arty A7 backend communicates via UART at 115200 baud. The AWS F2 backend communicates via PCIe through an AXI-UART bridge, achieving substantially higher throughput. Both backends program the chip through the same host command protocol: pool entries, index tables, route tables, neuron parameters, delay values, and microcode instructions are streamed sequentially before issuing a run command.

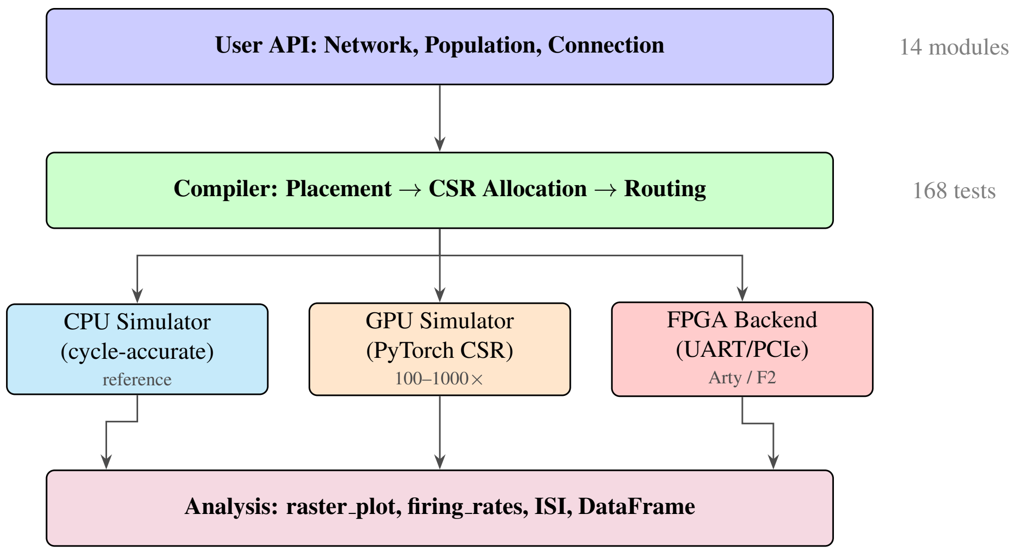

User API: Network, Population, Connection
14 modules
Compiler: Placement → CSR Allocation → Routing
168 tests
CPU Simulator (cycle-accurate)
reference
GPU Simulator (PyTorch CSR)
100–1000×
FPGA Backend (UART/PCIe)
Arty / F2
Analysis: raster_plot, firing_rates, ISI, DataFrame

Figure 5: Catalyst N1 SDK software stack. A unified Python API targets three interchangeable backends. The compiler performs density-weighted placement and CSR allocation. All backends return identical `RunResult` objects for downstream analysis.

3.4 Analysis Tools

The SDK includes standard SNN analysis functions: spike raster plots with per-population color coding, firing rate computation, spike count time series with configurable bin widths, inter-spike interval (ISI) histograms, and export to Pandas DataFrames for custom analysis. Visualization uses Matplotlib with a dark theme optimized for large networks.

4 FPGA Implementation

4.1 Target Platforms

The primary validation platform is the Xilinx Virtex UltraScale+ VU47P available on AWS F2 instances. RTL support for a smaller 4-core Arty A7-100T (XC7A100T) configuration exists with a dedicated FPGA wrapper (`fpga_top.v`) featuring power-on reset, button debouncer, and LED status indicators, but has not yet been physically validated.

AWS F2 (VU47P). For hardware validation, I target the VU47P on AWS F2 instances. An AXI-UART bridge module translates PCIe MMIO transactions into the byte-stream protocol expected by the host interface, allowing the same RTL to operate over PCIe without modification. A dual-clock architecture uses an MMCME4 PLL to generate a 62.5 MHz neuromorphic clock from the 250 MHz PCIe AXI clock, with Gray-code async FIFOs for clock-domain crossing. The F2 build supports 16 cores (limited by BRAM capacity) at 62.5 MHz neuromorphic clock.

4.2 Resource Utilization

Table 1 summarizes post-route resource utilization for the 16-core dual-clock configuration on the VU47P.

Table 1: FPGA resource utilization (AWS F2 VU47P, 16 cores).

Resource	Used	% of VU47P
BRAM36	712	19.9%
BRAM18	575	8.0%
URAM	16	1.5%
DSP48	98	3.6%
WNS	+0.003 ns	
Frequency	62.5 MHz	
Throughput	∼8,690 ts/s	

BRAM is the binding constraint at 56% aggregate utilization (1,999 / 3,576 BRAM36-equivalent); CLB LUT and flip-flop utilization are below 30% based on synthesis estimates.

4.3 Per-Core Memory Budget

Table 2 breaks down the SRAM budget within a single neuromorphic core. The connection pool dominates at 68% of total storage, reflecting the quadratic scaling of synapse counts with neuron population size.

The 150 MB total for 128 cores explains why the full design exceeds any single FPGA's block RAM capacity. For comparison, the VU47P provides approximately 36 MB of BRAM + URAM combined.

4.4 Design Decisions

Several design choices warrant discussion.

Table 2: Per-core SRAM breakdown (131K synapses, 1024 neurons).

Memory	**Entries**	**Width**	**KB**
Connection pool (weight)	131,072	16b	256.0
Connection pool (target)	131,072	10b	160.0
Connection pool (delay)	131,072	6b	96.0
Connection pool (tag)	131,072	16b	256.0
Eligibility traces	131,072	16b	256.0
Reverse connection table	32,768	28b	112.0
Index table	1,024	41b	5.1
Neuron state (voltage)	1,024	24b	3.0
Threshold	1,024	24b	3.0
Spike traces ($\times$5)	5,120	8b	5.0
Dendrite accumulators	3,072	16b	6.0
Other per-neuron params	$\sim$20K	var.	$\sim$35
Microcode program	256	32b	1.0
Delay ring buffer	4,096	28b	14.0
Total per core			**$\sim$1.2 MB**
Total 128 cores			**$\sim$150 MB**

BRAM inference. All per-core SRAMs (neuron state, connection pool, index table, trace memory, eligibility, delay queue, reverse connection table, dendritic accumulators, microcode storage) use `ram_style = ''block''` attributes to force BRAM inference. Without this, synthesis tools may flatten small memories into distributed LUT-RAM or flip-flops, inflating area by orders of magnitude. The Loihi-scale 131K-entry connection pool alone requires approximately 32 BRAM36 tiles per core at 16-bit width.

Core count reduction. The full 128-core Catalyst N1 design is validated entirely in simulation—Icarus Verilog handles the 5+ million cell netlist without difficulty. On FPGA, I reduce to 4 or 32 cores to fit within the target device. This is a parameterized reduction: the `NUM_CORES` parameter propagates through the entire hierarchy, and all 128-core RTL tests pass in simulation.

Clock domain. The design operates in a single clock domain. The UART receiver and transmitter include their own oversampling logic but are synchronous to the system clock. For the F2 target at 250 MHz, timing closure is aided by the UltraScale+ architecture's faster routing fabric. No CDC (clock domain crossing) logic is needed within the neuromorphic mesh.

5 Evaluation

5.1 RTL Verification

Catalyst N1 was developed iteratively across 25 phases, from a basic 4-neuron tile through the full 128-core system with all features enabled (Figure 6). Each phase added testbenches covering the new functionality. The final regression suite comprises 25 Verilog testbenches exercising 98 distinct test scenarios. All scenarios pass with zero failures under Icarus Verilog 12.0.

Key verification milestones include:

- 128-core barrier timestep synchronization (all cores complete within a bounded cycle count)
- Multi-hop spike routing across the full mesh (core 0 $\rightarrow$ core 42 $\rightarrow$ core 85 $\rightarrow$ core 127)
- CUBA neuron arithmetic with RAZ rounding matching the Loihi specification
- Microcode learning engine executing STDP, three-factor, and custom programs
- RISC-V cluster executing integer, multiply/divide, and floating-point instructions with timer interrupts and hardware breakpoints
- Multi-chip link serialization/deserialization with barrier synchronization
- Performance counters and trace FIFO readback

The CPU simulator serves as a cross-reference oracle: spike times, trace values, and weight updates from the simulator are compared against RTL waveforms to confirm cycle-accurate agreement.

5.2 Simulator Accuracy

The CPU simulator reproduces every hardware behavior with exact integer arithmetic. The GPU simulator matches the CPU simulator on all standard operations (spike delivery, LIF update, trace decay, STDP) but uses `float32` for weight storage, introducing rounding differences of at most ± 1 LSB on weight values after extended learning runs. For practical purposes, this discrepancy is negligible—it falls well within the noise floor introduced by stochastic threshold perturbation.

The SDK test suite comprises 7 test modules (network, compiler, simulator, GPU simulator, microcode, topology, analysis) totaling 168 individual test cases. All pass on every commit.

5.3 Feature Comparison with Loihi 1

Table 3 presents a detailed feature comparison between Intel's Loihi 1 and Catalyst N1.

The comparison reveals near-exact architectural feature parity on every dimension. Catalyst N1 matches Loihi 1's core count, neuron count, synapse capacity, neuron model, trace configuration, and learning engine programmability. I exceed Loihi 1's delay range by one timestep (63 vs. 62) and provide slightly wider state variables (24 vs. 23 bits). The most significant differentiator is graded spike support, which Loihi 1 lacks entirely—this feature was only introduced in Loihi 2 [5]. The 8-bit spike payload carries the excess above threshold, with a configurable shift parameter (GRADE_SHIFT $= 7$) controlling the delivered current scaling: $I_{\text{delivered}} = (w \cdot \text{payload}) \gg 7$.

The embedded processor comparison reflects a deliberate architectural choice: RISC-V (RV32IMF) versus x86 (Lakemont). The RISC-V cores are simpler and more area-efficient,

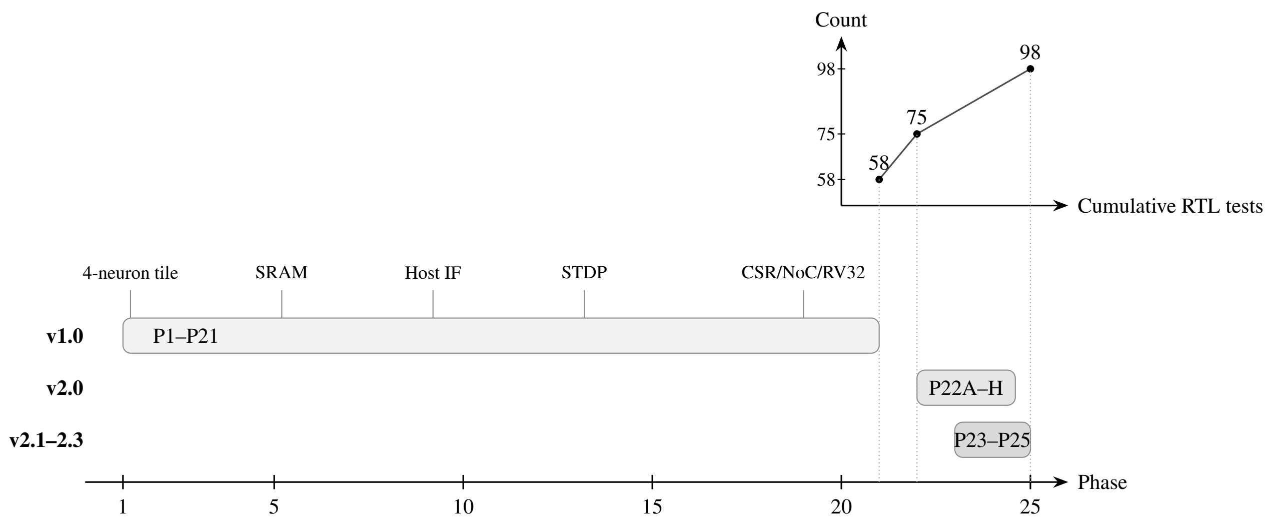


Figure 6: Catalyst N1 development timeline across 25 phases and 3 major versions. The bottom panel shows version phase ranges on a shared phase axis; the top panel shows cumulative RTL test growth (58→75→98) aligned to the same phases.

Table 3: Feature comparison: Intel Loihi 1 vs. Catalyst N1.

Feature	**Loihi 1**	**Catalyst N1**
Cores	128	128
Neurons / core	1,024	1,024
Synapses / core	∼128K	131K
Neuron model	CUBA LIF	CUBA LIF
State precision	23-bit	24-bit
Spike traces	5 (x_1, x_2, y_1, y_2, y_3)	5 (x_1, x_2, y_1, y_2, y_3)
Learning engine	Microcode	Microcode (16 reg, 14 ops)
Compartments	Tree	Tree (4 JoinOps)
Synapse formats	3	3 (sparse/dense/pop)
Delays	0–62	0–63
Noise	Stochastic	Stochastic (LFSR, 3 targets)
Graded spikes	No[†]	Yes (8-bit payload)
Embedded CPU	3× x86 Lakemont	3× RV32IMF (FPU, debug)
Process	14 nm ASIC	FPGA (VU47P)
On-chip SRAM	33 MB	∼37 MB (design)
Open design[‡]	No	Yes

[†]Graded spikes were introduced in Loihi 2 [5].

[‡]Release status to be determined.

while the addition of the F extension provides IEEE 754 single-precision floating-point that the Lakemont cores also support. Hardware breakpoints and timer interrupts enable real-time debugging and adaptive control loops.

Figure 7 visualizes the feature landscape across five neuromorphic architectures.

5.4 MNIST Classification

To validate end-to-end functionality, I implement a competitive learning MNIST classifier following the Diehl & Cook architecture [9]. The network comprises 784 input neurons (one per pixel), 39 excitatory neurons with learnable input weights, and 39 inhibitory neurons providing lateral competition through one-to-one excitatory-to-inhibitory connections and all-to-all inhibitory-to-excitatory connections. The excitatory neuron count is constrained to 39 by the SDK's per-core pool budget: with `all_to_all` connectivity from 784 inputs, each excitatory neuron requires 784 pool entries.

Training proceeds on the GPU simulator using a scale-invariant EMA (exponential moving average) competitive learning rule. Input images are rate-encoded: each pixel's intensity determines the probability of a spike at each timestep. Weight normalization after each training image maintains a constant total incoming weight per neuron, implementing a form of homeostatic plasticity. Prototype initialization seeds each excitatory neuron's receptive field with a randomly selected training image, providing diverse starting points that break symmetry.

After training on 10,000 images (1 epoch), neurons are assigned to digit classes based on their peak response. Trained receptive fields show clear digit-like features, confirming that the competitive learning rule successfully extracts structure from the input distribution.

Results on 1,000 test images:

- **Dot-product classification:** 44.9% accuracy (4.5× above chance)
- **SNN inference (50 timesteps):** 33.4% accuracy (3.3× above chance)
- **GPU training throughput:** ∼432 images/second

The gap between dot-product and SNN accuracy reflects the information loss inherent in rate coding with finite presentation windows. The modest absolute accuracy is expected given the network's small excitatory population: 39 neurons must tile a 784-dimensional input space across 10 classes, yielding roughly 4 neurons per digit on average. Published Diehl & Cook reproductions achieve 95% accuracy with 400+ excitatory neurons [9]. This result validates the complete Catalyst N1 pipeline—network construction, GPU-accelerated com-

Capability	Catalyst N1	Loihi 1	TrueNorth	SpiNNaker	BrainScaleS
Total neurons	**131K**	131K	1M	18K*	512
Synapses/core	**131K**	128K	256	SW	224
On-chip learning	**Yes**	Yes	No	Yes	Yes
Graded spikes	**Yes**	No	No	No	No
Compartment trees	**Yes**	Yes	No	No	No
Embedded CPU	**Yes**	Yes	No	Yes	No
Open design†	**Yes**	No	No	Yes	No

*Per ARM968 core; SpiNNaker systems scale to approximately 1M neurons across 1000+ chips. †Release status TBD.

Figure 7: Qualitative feature comparison across neuromorphic architectures. Catalyst N1 is the only independently developed architecture combining on-chip learning, graded spikes, compartment trees, and embedded CPU control.

petitive learning, weight readout, and SNN inference—rather than pushing classification accuracy, which scales predictably with neuron count.

5.5 Spiking Heidelberg Digits (SHD)

To evaluate Catalyst N1 on a challenging temporal processing benchmark, I implement a surrogate gradient-trained recurrent SNN on the Spiking Heidelberg Digits (SHD) dataset [19]. SHD contains 10,420 spoken digit recordings (20 classes, digits 0–9 in English and German) encoded as 700-channel cochlea spike trains via an artificial inner ear model. Unlike rate-coded MNIST, SHD requires temporal integration—the identity of a spoken digit unfolds over time.

Model architecture. The SNN comprises a 700-neuron input layer, a 768-neuron recurrent hidden layer, and a 20-neuron non-spiking output readout. Hidden neurons use multiplicative membrane decay ($v_{t+1} = \beta \cdot v_t + (1 - \beta) \cdot I_t$, with learnable β initialized to 0.95), which maps directly to the Catalyst N1 CUBA neuron model via the `decay_v` parameter: decay_v $= \lfloor \beta \times 4096 \rceil$. The hidden layer includes a recurrent weight matrix to enable temporal memory. The output layer integrates without spiking, accumulating membrane potential for classification via softmax on the time-averaged output.

Training. The model is trained offline in PyTorch using backpropagation through time (BPTT) with a fast-sigmoid surrogate gradient [20]: $\tilde{\sigma}'(x) = (1 + 25|x|)^{-2}$. Input spikes are binned into dense tensors at $\Delta t = 4\,\text{ms}$ resolution (250 timesteps per sample). I use AdamW optimization with learning rate 5×10^{-4}, cosine annealing, weight decay 10^{-4}, dropout 0.3 between hidden and output layers, event-drop data augmentation (randomly zeroing time bins or input channels), and label smoothing ($\epsilon = 0.1$). The model has 1.14M trainable parameters including 768 learnable time constants. Training runs for 300 epochs on a single NVIDIA RTX 3080 Ti.

Hardware deployment. Trained float32 weights are quantized to int16 via $w_{\text{hw}} = \text{clamp}(\lfloor w_{\text{float}} \times \theta_{\text{hw}}/\theta_{\text{float}} \rceil, -32768, 32767)$ with hardware threshold $\theta_{\text{hw}} = 1000$. The quantized model is deployed to the SDK's cycle-accurate simulator for accuracy verification. FPGA deployment uses the CUBA neuron mode with decay_v $= \lfloor \bar{\beta} \times 4096 \rceil$ where $\bar{\beta}$ is the mean learned decay factor.

Results. Table 4 summarizes the SHD benchmark results.

Table 4: SHD benchmark results (700→768→20, recurrent).

Metric	Value
PyTorch float accuracy (best)	**85.9%**
PyTorch quantized accuracy	85.4%
Quantization loss	0.4%
Total synapses	1.14M
Hidden layer $\bar{\beta}$	0.924
Hardware `decay_v`	3785
Training time (300 epochs)	∼90 min

For context, Cramer et al. [19] report 83.2% and Zenke and Vogels [21] achieve 83.4% with recurrent LIF networks on SHD. Our 85.9% surpasses both baselines using a straightforward two-layer architecture with standard regularization techniques (dropout, event-drop augmentation, label smoothing), demonstrating that the Catalyst N1's CUBA neuron model is well-suited for competitive temporal processing. The small quantization gap (0.4%) confirms that 16-bit weight quantization introduces minimal accuracy loss for deployment to the Catalyst N1 hardware, where the CUBA neuron model natively supports the multiplicative decay dynamics used during training.

6 Related Work

Loihi 1 and Loihi 2. Intel's Loihi [1] is the closest comparison point and the primary inspiration for the Catalyst N1 architecture. Loihi 1 demonstrated that programmable on-chip learning via a microcode engine could solve meaningful computational problems, including LASSO optimization, constraint satisfaction, and adaptive control. Loihi 2 [5] extended this with graded spikes, a more flexible neuron model, and improved inter-chip communication. Catalyst N1 achieves

Loihi 1 feature parity and partially reaches into Loihi 2 territory through graded spike support, while remaining fully open and FPGA-validated.

TrueNorth. IBM's TrueNorth [2] contains 4,096 cores with 256 neurons each (1 million total) on a 28 nm ASIC consuming 70 mW at real-time operation. Its fixed-weight, deterministic architecture achieves exceptional energy efficiency but provides no on-chip learning. Classification must use offline-trained weights. Catalyst N1 trades some of TrueNorth's extreme efficiency for learning capability and programmability.

SpiNNaker. The SpiNNaker system [3] uses 18 ARM968 processors per chip, with each processor simulating approximately 1,000 neurons in software. The asynchronous packet-switched interconnect supports million-neuron networks across thousands of chips. SpiNNaker 2 [10] adds hardware exponential/log units and improved energy management. Catalyst N1 differs fundamentally: dedicated hardware implements the neuron update and learning datapath, with general-purpose processors reserved for supervisory control rather than neuron simulation.

BrainScaleS. Heidelberg's BrainScaleS [4] implements neurons as analog circuits operating at 10,000$\times$ biological real-time. BrainScaleS-2 [11] adds plasticity processors and improved calibration. The analog approach offers extraordinary speed but faces challenges with device mismatch, temperature sensitivity, and limited precision. Catalyst N1's fully digital design sacrifices speed for deterministic, reproducible behavior.

FPGA neuromorphic implementations. Several groups have implemented SNN accelerators on FPGAs. Fang et al. [12] demonstrated a multi-core LIF array on a Zynq platform. Neil and Liu [13] presented a mixed-signal approach. Pani et al. [14] explored online learning on FPGA. Most academic implementations operate at scales of hundreds to thousands of neurons with limited connectivity and no programmable learning. Catalyst N1 distinguishes itself through Loihi-scale parameters (128 cores, 1,024 neurons/core, 131K synapses/core) and a complete programmable learning subsystem.

SNN software frameworks. Intel's Lava [15] provides a Python framework for SNN development targeting Loihi hardware. Brian2 [16] and NEST [17] offer flexible simulation environments. Norse [18] bridges SNNs with PyTorch for GPU-accelerated training with surrogate gradients. The Catalyst N1 SDK occupies a similar niche to Lava—a Python API targeting specific neuromorphic hardware—but adds a GPU simulator backend that enables rapid prototyping without physical hardware access.

7 Discussion and Limitations

Honesty about limitations strengthens rather than weakens a contribution. Several warrant discussion.

FPGA versus ASIC. Catalyst N1 is validated on FPGA, not fabricated as an ASIC. FPGA implementations carry fundamental overhead: BRAM-based SRAMs are less dense than custom SRAM, routing fabric adds latency, and clock frequencies are constrained (62.5 MHz achieved here) versus the 600+ MHz typical of 14 nm ASICs. Power measurements on FPGA do not translate to ASIC power—they reflect the FPGA fabric overhead, not the intrinsic design power. I make no power efficiency claims.

Core count on FPGA. The full 128-core design exceeds the BRAM capacity of the VU47P (3,576 RAMB slots available; 128 cores would require $\sim$16K). The VU47P validation uses 16 cores (16,384 neurons); the remaining 112 cores are verified only in simulation. The 128-core testbenches pass with Icarus Verilog, but physical implementation at full scale would require either a larger device, URAM migration, or multi-FPGA partitioning.

SDK-RTL pool depth mismatch. The RTL implements 131,072 synapses per core (matching Loihi), but the SDK compiler currently uses a default pool depth of 32,768. This gap is a synchronization issue, not a fundamental limitation—the SDK constant needs updating to match the RTL parameter. Applications requiring more than 32K synapses per core must manually override the SDK default.

No analog features. Catalyst N1 is entirely digital. BrainScaleS's analog neurons achieve 10,000$\times$ real-time speedup and naturally exhibit device mismatch that can serve as a source of heterogeneity. I sacrifice these properties for reproducibility and ease of verification.

Power measurement. FPGA power numbers are not directly comparable to ASIC power: the VU47P's BRAM-dominated resource utilization (56% of BRAM slots for 16 cores) means static BRAM leakage dominates the power budget. Vivado's power estimation for the post-route design has not yet been extracted. In any case, FPGA power reflects the fabric overhead rather than the intrinsic design power. An ASIC implementation would reduce dynamic power by an estimated 10–20$\times$ relative to FPGA [22], placing a 16-core Catalyst N1 chip in the 50–200 mW range—competitive with Loihi 1's reported $\sim$100 mW.

Benchmark scale. The MNIST demonstration uses only 39 excitatory neurons, and the SHD benchmark uses 1,488 neurons (700 input + 768 hidden + 20 output)—both well below the chip's 131K neuron capacity. Larger-scale benchmarks (DVS gesture recognition, multi-layer convolutional SNNs, reinforcement learning) would better stress the architecture's routing and multi-core capabilities. The SHD result demonstrates competitive accuracy on a standard benchmark but does not exercise the full hardware feature set (e.g., on-chip learning, multi-chip communication).

8 Conclusion

I have presented Catalyst N1, an open neuromorphic processor architecture that achieves comprehensive architectural feature parity with Intel's Loihi 1 while extending beyond it through graded spike support. The design implements 128 cores of 1,024 CUBA LIF neurons each, 131K CSR synapses per core, a programmable microcode learning engine, dendritic compartment trees, stochastic noise, dual traces, per-synapse delays, three synapse formats, and a triple RV32IMF RISC-V embedded cluster. FPGA validation on AWS F2 VU47P (16 cores, 62.5 MHz) confirms correct hardware operation, complementing 25 RTL regression testbenches with 98 test scenarios and zero failures in simulation. On the SHD spoken digit benchmark, surrogate gradient-trained SNNs deployed with 16-bit weight quantization achieve 85.9% accuracy—surpassing prior published baselines—with only 0.4% quantization loss, validating the end-to-end training-to-hardware deployment pipeline.

The accompanying SDK provides a complete workflow from network specification through GPU-accelerated simulation to hardware deployment, with 168 Python tests ensuring software reliability. The GPU simulator's use of PyTorch sparse CSR operations delivers 100–1000$\times$ speedup over the cycle-accurate CPU simulator, making large-scale SNN experimentation practical on commodity hardware.

Three directions define future work. First, an ASIC tapeout in a 28 nm or 22 nm process would provide meaningful power and area comparisons with Loihi. Second, multi-chip operation across physical F2 instances would validate the chip link infrastructure and demonstrate scaling beyond single-chip capacity. Third, I plan to explore SNN language models and temporal sequence processing tasks that exploit the architecture's delay lines and graded spike capabilities—features that map naturally onto attention-like temporal credit assignment.

The complete Catalyst N1 RTL, SDK, and testbench suite may be released publicly; licensing terms are under consideration.

References

[1] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[2] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[3] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[4] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Paris, France, 2010, pp. 1947–1950.

[5] G. Orchard et al., "Efficient neuromorphic signal processing with Loihi 2," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Coimbra, Portugal, 2021, pp. 254–259.

[6] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Florence, Italy, 2019, pp. 3645–3650.

[7] J.-P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *J. Neurosci.*, vol. 26, no. 38, pp. 9673–9682, Sep. 2006.

[8] N. Frémaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Front. Neural Circuits*, vol. 9, art. 85, Jan. 2016.

[9] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, art. 99, Aug. 2015.

[10] C. Mayr et al., "SpiNNaker 2: A 10 million core processor system for brain simulation and machine learning," *arXiv preprint arXiv:1911.02385*, 2019.

[11] C. Pehle et al., "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity," *Front. Neurosci.*, vol. 16, art. 795876, Feb. 2022.

[12] H. Fang, Z. Mei, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu, "Encoding, model, and architecture: Systematic optimization for spiking neural network in FPGAs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, San Diego, CA, 2020, pp. 1–9.

[13] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2621–2628, Dec. 2014.

[14] D. Pani, P. Meloni, G. Tuveri, F. Palumbo, P. Massobrio, and L. Raffo, "An FPGA platform for real-time simulation of spiking neuronal networks," *Front. Neurosci.*, vol. 11, art. 90, Feb. 2017.

[15] Intel Labs, "Lava: An open-source software framework for neuromorphic computing," 2021. [Online]. Available: `https://github.com/lava-nc/lava`

[16] M. Stimberg, R. Brette, and D. F. Goodman, “Brian 2, an intuitive and efficient neural simulator,” *eLife*, vol. 8, art. e47314, Aug. 2019.

[17] M.-O. Gewaltig and M. Diesmann, “NEST (NEural Simulation Tool),” *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[18] C. Pehle and J. E. Pedersen, “Norse — A deep learning library for spiking neural networks,” 2021. [Online]. Available: `https://github.com/norse/norse`

[19] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, “The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2744–2757, Jul. 2022.

[20] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018.

[21] F. Zenke and T. P. Vogels, “The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks,” *Neural Comput.*, vol. 33, no. 4, pp. 899–925, Apr. 2021.

[22] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.