Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Bit-Level Verification of Membrane Potential and Spikes in a Fixed-Point LIF Neuron and a 256-Neuron Spiking Network

ChiaAn Lin

ntuhibiki@gmail.com

National Taiwan University https://orcid.org/0009-0008-2034-1781

Research Article

Keywords: spiking neural network, leaky integrate-and-fire, fixed-point arithmetic, FPGA, Tang Nano 20K, Gowin GW2AR-18

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11650337/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Bit-Level Verification of Membrane Potential and Spikes in a Fixed-Point LIF Neuron and a 256-Neuron Spiking Network

ChiaAn Lin*
Master Program for Health and Intelligent Medicine, National Taiwan University

November 2026

Abstract

Spiking neuron models are small enough to run on inexpensive FPGAs, but a hardware result is only useful for model-based experiments if one can state exactly how it differs from the model. We present a leaky integrate-and-fire (LIF) neuron and a 256-neuron network on a low-cost FPGA (Tang Nano 20K, Gowin GW2AR-18), and we test both against a Python reference model. The model uses signed Q8.8 fixed-point state, saturating additions and shift-based time constants, so no multiplier is needed. On the board, the single neuron's membrane potential and spike matched the model at every step of 60,000 samples from three hand-picked parameter sets and of 2,023,200 samples from a sweep of 2,248 parameter sets, all with constant input currents. The synaptic current was not read back, so it is not compared directly. A time-multiplexed network with spike-driven synapses, controlled over a UART link, matched the model in every reply of a 682-frame command flow, and in the spike counts, the final-state snapshot and the first 512 steps of spikes of five excitation/inhibition (E/I) conditions. Spike timing after step 512 was not compared, and the membrane potential of the network only at the last step. Mutation tests show that the checkers can fail. The full design uses 12% of the logic and 98% of the block RAM, and meets timing at 38.75 MHz after place and route, against a 27 MHz clock. In the model, the mean firing rate of the network falls monotonically as the inhibitory fraction rises, in all 240 networks of the sweep; at low inhibition the rate is limited by the refractory ceiling. On the board, the current reached its upper and lower limits and the potential its lower limit, and every saturated step matched the model, including a negative threshold where the lower limit decides a spike; the upper limit of the potential cannot change the output. The membrane potential of the network was compared only at the last step, so saturation inside the network was not checked. We also report the limits we found: a USB–UART link that lost bytes in long unpaced bursts, a command decoder without a time-out, and a spike trace that covers only the first 512 steps. The hardware is validated against the model only; we make no claim about biological validity.

1 Introduction

Networks of spiking neurons with excitatory and inhibitory populations are used to study how the balance between excitation and inhibition shapes neural activity. Such networks are small enough to run on inexpensive FPGAs, and many FPGA designs for spiking networks have been reported [1].

A hardware result is only useful for model-based experiments if one knows how it differs from the model it implements. Fixed-point hardware makes this question sharp: quantisation, saturation and the choice between arithmetic and logical shifts each change the numbers, and an error in any of them can leave the average behaviour almost intact while changing individual spikes. Validation by task accuracy, or by agreement with a simulator, can miss such errors,

*Corresponding author: ntuhibiki@gmail.com

and some FPGA reports validate their designs this way [2, 3]. A stricter test asks that every state variable of the hardware equals the state of a reference model after every step, with zero differences, so that a single flipped bit makes the test fail.

Bit-accurate software models are also used as references for FPGA designs. An integer-LIF accelerator for image classification was checked against a bit-accurate software model on the full MNIST test set, while the hardware was checked for structural equivalence on a 1,000-sample subset, rather than step by step [4]. A systolic simulator built around updating only the neurons that spike is an example of a design aimed at speed and scale [5]. These designs target other boards and tasks, and their numbers are not comparable with ours.

This paper applies the strict test to a single neuron and to a recurrent network with excitatory and inhibitory sources, on a low-cost board. We make three contributions.

1. **An exact specification.** A fixed-point LIF neuron with saturating additions and arithmetic shifts, and a network input rule that sums the synaptic contributions exactly and saturates once. Because the sum is exact, the result does not depend on the order in which the sources are added.

2. **A verification method.** The model is the specification. The method combines simulation, mutation tests and board comparison. For the single neuron, the board comparison is step by step (membrane potential and spike) and covers 2,083,200 samples. For the network, the board comparison covers a 682-frame command flow, and for five E/I conditions the spike counts, the final-state snapshot and the spikes of the first 512 steps. The membrane potential of the network is compared only at the last step.

3. **A 256-neuron network on a low-cost board.** A UART-controlled, time-multiplexed network, with its resource use and the limits we measured: the link lost bytes in long unpaced bursts, the decoder has no time-out, and the spike trace covers 512 steps.

We also include an E/I experiment in the model, in which the fraction of inhibitory sources is varied, and we run five of its conditions on the board. Its purpose is to show that the hardware and the model agree while the E/I balance changes. It does not support conclusions about neural tissue.

The paper is organised as follows. Section 2 defines the model, Section 3 the hardware, and Section 4 the verification method. Section 5 gives the board results and the E/I demonstration, Section 6 the resource use, Section 7 the limitations, and Section 8 concludes.

2 Model

2.1 Single neuron

Each step updates a membrane potential V and a synaptic current I:

$$I[t+1] = I[t] - (I[t] \gg k_s) + x[t], \tag{1}$$

$$V[t+1] = V[t] + ((V_{\text{rest}} - V[t]) \gg k_m) + I[t+1], \tag{2}$$

where $x[t]$ is the input of the step and every addition saturates (Section 2.2). We write $s[t]$ for the spike flag of step t. If the neuron is not refractory and $V[t+1] \geq \theta$, it emits a spike, $s[t+1] = 1$, V is set to V_{reset}, and the neuron is refractory for *refrac* steps. While it is refractory, V is held at V_{reset} and I keeps integrating. The parameters are θ, V_{reset} (the potential after a spike and during the refractory period), V_{rest} (the target of the leak), k_m and k_s (0–15), and *refrac* (0–7). A right shift by k divides by 2^k, so the time constants are $\Delta t/\tau = 2^{-k}$.

2.2 Number format

V and I are signed 16-bit numbers in Q8.8 format (raw value = real value $\times 256$). Three rules fix the arithmetic exactly:

- In the single-neuron update of Eqs. (1) and (2), every addition saturates to $[-32768, 32767]$ after each addition, from left to right. Nothing wraps. The network input has its own rule (Section 2.3): the sum over sources is exact and the total saturates once.
- Right shifts are arithmetic (they round toward $-\infty$).
- $V_{\text{rest}} - V$ is computed with more than 16 bits before the shift, because the difference of two 16-bit values may need 17 bits.

A wrap would turn a strong excitatory input into a strong inhibitory one, and a logical shift would turn a decaying negative potential into a large positive one. The mutation tests in Section 4 include the logical shift. They do not cover every wrap or saturation variant, which is a limitation (Section 7). The reference is a short Python function (`model/fixed.py`); the hardware must match it exactly.

2.3 Network

All neurons share one parameter set. Let $s_j[t]$ be the spike flag of neuron j at step t. The input of neuron i at step t is

$$x_i[t] = \text{sat}\left(\sum_j \left(W_{ji} \ll w_{\text{shift}}\right) s_j[t-1] \;+\; \text{ext}_i \right), \tag{3}$$

where W_{ji} is a signed 8-bit weight from neuron j to neuron i (self-connections are allowed, and a zero weight means no connection), $w_{\text{shift}} = 2$ is a global scale, and ext_i is a constant external input. The sum over sources is exact, the external input is added to it, and the total saturates once. This differs from the single-neuron rule on purpose. The hardware visits the sources in a fixed order, and an exact sum followed by a single saturation gives the same result for every order. A spike at step t reaches the inputs at step $t+1$.

Initial state: before step 0 every neuron is at rest ($V = 0$, $I = 0$, no refractory time), no neuron has spiked, so $s_j[-1] = 0$ for all j, and the synaptic sum is empty. The external input ext_i is added from step 0. At the start of each run the hardware clears the state, the spike flags and both accumulators (the initialisation phase of `rtl/network.v`), so the board starts from the same state as the model.

3 Hardware

3.1 Board and link

The target is a Tang Nano 20K with a Gowin GW2AR-18 device and a 27 MHz clock. A UART link at 115,200 baud connects it to a PC. The toolchain is open source (yosys, nextpnr-himbaechel, openFPGALoader). Frames have the form `[0xA5][cmd][len][data][checksum]`, with an additive checksum. The decoder on the FPGA rejects a frame with a bad checksum or an out-of-range value, and counts accepted and rejected frames. It refuses every command while it is busy sending a reply, copying a payload or running the network.

3.2 Single neuron

One neuron step takes one clock cycle: the update is a combinational circuit, and the state is held in registers. Each step writes {spike, V} (17 bits) to a record memory of 1,024 steps in block RAM. The host reads the records back in batches. The synaptic current I and the refractory counter are not recorded, so they are not compared with the model. Each run starts from rest ($V = 0$, $I = 0$, no refractory time), the same initial condition as the model.

3.3 Network

The network of $N = 256$ neurons uses one copy of the neuron circuit, shared in time: one step takes N neuron updates followed by the synapse phase. All memories except the spike flags have one write port and one synchronous read port, so the synthesis tool maps them to block RAM. The spike flags are small and are read combinationally, so they are kept in distributed memory. Written as arrays, the whole state would become flip-flops; an earlier array version needed about 45,000 cells at $N = 256$, which does not fit the chip. Synaptic input is spike driven: when neuron j spikes, its weight row is added to an accumulator for the next step. Two accumulators alternate, one read while the other is written. Both phases are pipelined with a one-cycle read latency. The read order keeps consecutive reads on different neurons, so the pipeline never reads a value that is still being written.

The synaptic sum is exact in the hardware as well. Each source contributes its weight shifted left by two, which lies in $[-512, 508]$. Over 256 sources the sum lies in $[-131{,}072, 130{,}048]$, which needs 18 bits. The accumulator is 24 bits wide (`rtl/network.v`), so adding the 16-bit external input cannot overflow at any step, and the total is then saturated once. The largest exact sum measured in the board networks was 1,084 raw in the command flow and 378 raw in the E/I conditions, so on the board the accumulator bits above bit 10 carried only the sign. The width is a bound, not a test of the upper bits.

The board does not keep the full state of every step. At $N = 256$ and 1,024 steps, the membrane potentials and spikes alone would need 4,352 Kbit ($256 \times 1{,}024 \times 17$ bits), and the block RAM holds about 828 Kbit. The full state, which also includes I (16 bits) and the refractory counter (3 bits), would need 9,216 Kbit ($256 \times 1{,}024 \times 36$ bits). The board therefore keeps, per neuron, a 16-bit spike count and a 17-bit snapshot of the last step (33 bits in total), and the spike flags of every neuron for the first 512 steps of a run (1 bit per neuron and step, 128 Kbit), returned seven steps per frame. The membrane potential is therefore compared only at the last step, which is the main limit of the network verification (Section 7).

4 Verification

The model is the specification, and a difference of one bit is a failure. The verification has several layers.

Simulation. The single neuron was checked with cocotb and Icarus Verilog against the model: 10,110 combinational cases and 15 traces of 500 steps each. The network core was checked against the model for $N = 16, 64, 256$ over 3,200, 12,800 and 51,200 records, and the full design was checked at the byte level through the UART pins for the same sizes (for $N = 256$, 682 frames).

Mutation tests. A checker that has never failed does not show that it can fail. Each checker was therefore run on deliberately broken copies of the design. Table 1 lists the 18 hand-written mutants and the checker that caught each one. All 18 were caught. Each was run against one checker, not against all of them, and all were run in simulation; none was run on the board or on a netlist. This is not a complete mutation analysis, and it does not show that every possible error is caught. An early byte-level testbench of the single neuron exposed a real design error:

Table 1: Hand-written mutants and the checker that caught each one. All runs were in simulation. The full design is `rtl/network_top.v`; the network core is `rtl/network.v`. The single-neuron checker is the cocotb test `test_lif_core.py`.

#	Mutant	Unit	Checker
1	arithmetic right shift replaced by logical shift	single neuron	cocotb
2	weight shift $w_{\text{shift}} = 3$	network core	`tb_network.v`
3	last target not added	network core	`tb_network.v`
4	synaptic input not delivered	network core	`tb_network.v`
5	write to the wrong accumulator buffer	network core	`tb_network.v`
6	record stores the previous V	single neuron, byte level	`tb_neuron_top.v`
7	reply checksum +1	command path	`tb_cmd_test.v`
8	request checksum check disabled	command decoder	`tb_cmd_decoder.v`
9	k_m range check removed	command decoder	`tb_cmd_decoder.v`
10	UART register fixed at 0	UART echo	`tb_uart_echo.v`
11	input bytes swapped	full design	`tb_network_top.v`
12	range check off by one	full design	`tb_network_top.v`
13	counter timing error	full design	`tb_network_top.v`
14	spike trace bit order reversed	full design	`tb_network_top.v`
15	trace step address off by one	full design	`tb_network_top.v`
16	trace limit 513 instead of 512 steps	full design	`tb_network_top.v`
17	trace limit 8 instead of 7 steps per frame	full design	`tb_network_top.v`
18	`READ_STATUS` ignores busy	command decoder	`tb_decoder_busy.v`

a second run was sent while the previous reply was still being sent, which overwrote the record memory. The busy rule of Section 3 is the fix.

Board, single neuron. The host writes the parameters, confirms them with a status read, runs the neuron, reads every step's V and spike back and compares them with the model.

Negative control of the checker. The comparison used on the board (differences in membrane potential and spike) was tested on the saved board traces. The unmodified traces give 0 differences. Each injected error (one bit flipped, one spike removed, a one-step shift, a replaced trace) gives a nonzero count (`host/negative_control.py`, `results/negative_control_2026-11-06.log`). This tests the sensitivity of the checker on real data. It is not a board-level mutation test: the mutants of Table 1 were not run on the board.

Board, network. The host sends the full command flow of 682 frames (70,168 bytes in, 1,892 bytes out) and compares every reply byte with the model's prediction. The flow has 682 frames (Table 2). Seven are deliberately invalid. Each must be refused without changing any state, and the final status read checks that the error counter matches. The flow itself has no unknown command, no misaligned frame, no run of consecutive invalid frames, and no command sent while the board is busy: the host waits for each full reply and, after `RUN`, for the estimated run time. Some of these cases were run on the board separately, on the single-neuron build (Section 5.5): an unknown command, a bad checksum, a truncated frame and a command sent while busy. Leading junk bytes and runs of consecutive invalid frames are tested in simulation only, with the stream of `host/gen_cmd_vectors.py` in `sim/tb_cmd_decoder.v`. Commands sent while the decoder is busy are also checked in simulation, in `sim/tb_decoder_busy.v`.

The E/I conditions are compared in the same way: spike counts, last-step snapshot and the spikes of the first 512 steps against the full model.

Table 2: Frames in the 682-frame command flow at $N = 256$, by command, and the deliberately invalid frames among them.

Command	Frames	Invalid frames
`WRITE_PARAM`	2	1 (checksum)
`WRITE_WEIGHTS`	657	1 (runs past the last weight)
`SET_INPUT`	3	0
`RUN`	1	0
`READ_STATUS`	3	0
`READ_COUNTS`	5	1 (runs past the last neuron)
`READ_SNAPSHOT`	4	0
`READ_TRACE`	7	4 (ends one step past the trace; starts past it; count 0; count 8)
Total	682	7

Table 3: Single neuron: board against model. Values are real numbers; the Q8.8 raw value is the real value times 256. Columns: θ, V_{reset}, V_{rest}, k_m, k_s, refrac, samples, mismatches.

Set	θ	V_{reset}	V_{rest}	k_m	k_s	refrac	Samples	Mismatches
A	2.0	0	−0.25	4	3	2	20,000	0
B	0.5	−0.5	0.25	0	0	0	20,000	0
C	2.0	−0.5	0	15	15	7	20,000	0

No netlist simulation. The block-RAM simulation models that come with the installed toolchain are empty, so a netlist simulation cannot run. We therefore use the board comparison as the functional evidence for the synthesised design. It covers the cases we ran, not all cases.

4.1 AI-assisted development

This work was developed with Claude Code, an AI coding assistant. The assistant wrote and edited code, tests and text under the direction of the author. The author chose the design, the verification method and the claims, and checked the results. The author is responsible for every claim in this paper.

5 Results

5.1 Single neuron on the board

Table 3 shows three parameter sets with 20 constant input currents each (from −0.1 to 0.375) and 1,000 steps per input. All 60,000 samples of membrane potential and spike matched the model, and the error counter of the board stayed at zero. Figure 1 shows the first 100 steps of parameter set A, and Figure 2 the f–I points of the same run against the model.

The fixed-point and floating-point versions of the same neuron agree in shape, with a largest difference of 0.027 spikes per step (`docs/fi_curve.png`). That comparison uses the defaults of `lif.Params` ($\theta = 1.0$, $V_{\text{reset}} = 0$, $V_{\text{rest}} = 0$, $k_m = 4$, $k_s = 2$, *refrac* $= 2$), input currents from 0 to 0.1 and 2,000 steps. It does not use the parameter sets of Table 3.

5.2 Parameter sweep

Table 4 covers every legal value of k_m, k_s and *refrac*, and 200 random parameter sets. Each set ran three constant inputs (−0.2, 0.1, 0.4) for 300 steps. Across both sweeps, 2,023,200 samples

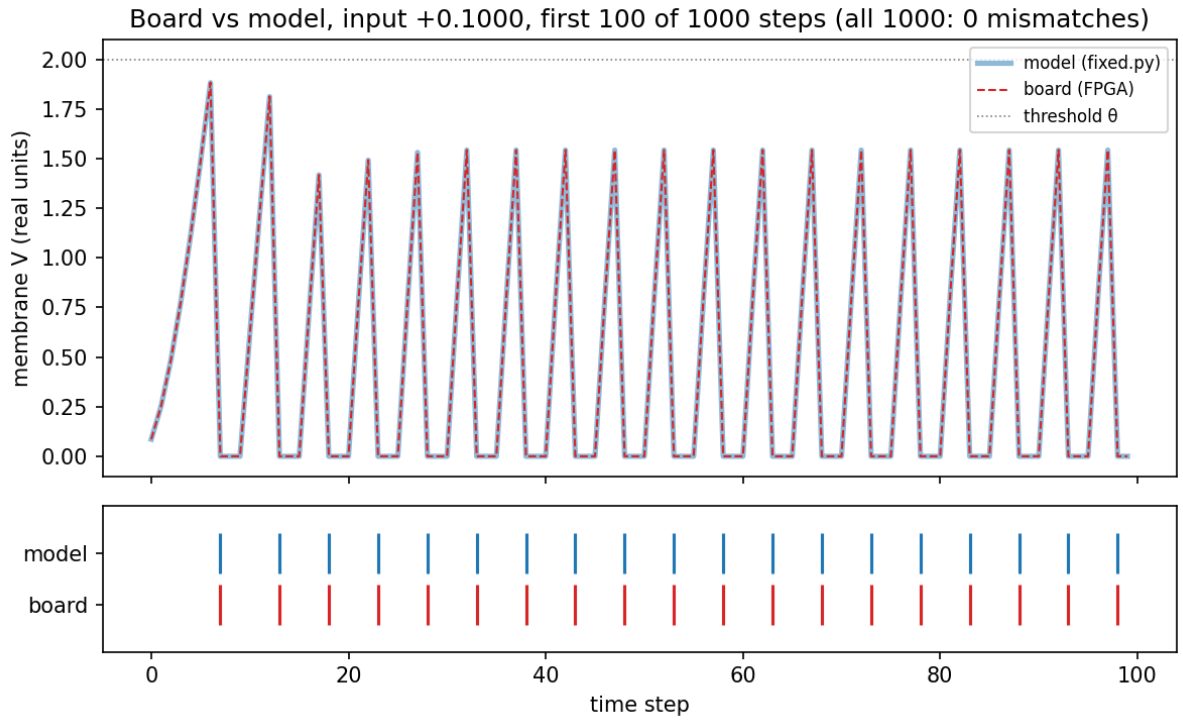


Figure 1: Membrane potential and spikes of parameter set A, board against model, first 100 of 1,000 steps. The potential is the recorded value after the step, so at a spike it shows the reset value rather than a crossing of θ.

Table 4: Parameter sweep on the board. Grid: $\theta = 1.0$, $V_{\text{reset}} = -0.25$ and $V_{\text{rest}} = 0.125$ fixed, k_m, k_s and *refrac* over all legal values. Random: seed 20261103, each parameter drawn uniformly over the ranges $\theta \in [64, 1024]$, $V_{\text{reset}} \in [-512, 256]$, $V_{\text{rest}} \in [-512, 512]$, $k_m, k_s \in [0, 15]$ and $\mathit{refrac} \in [0, 7]$ (voltages as Q8.8 raw values).

Sweep	Sets	Samples	Mismatches	Failed reads	Sets with spikes
Grid	2,048	1,843,200	0	0	2,024
Random	200	180,000	0	0	195

matched with no mismatch, no failed read and no refused request. Sets without spikes (24 grid, 5 random) are included on purpose: a silent neuron has to match the model too.

5.3 Saturation on the board

The single-neuron update has four saturating additions: the decay of I (i_1), the input added to I (i_2), the leak of V (v_1) and the current added to V (v_2). The decay (i_1) and the leak (v_1) cannot saturate. The decay $I - (I \gg k_s)$ lies between I and zero, and the leak $V + ((V_{\text{rest}} - V) \gg k_m)$ lies between V and V_{rest}, so both stay in the 16-bit range. The upper limit of v_2 cannot change the output either. A value clamped there is $32{,}767 \geq \theta$, so the neuron spikes, or it is refractory and the potential is discarded; in both cases the stored potential is V_{reset}, as it would be without the clamp. The three clamps that can change the output are the upper and lower limits of i_2 and the lower limit of v_2; they need a test.

Table 5 lists the runs that reached them. The board does not report I, so a clamp of I is seen only through the potential it produces. Clamp counts come from a traced copy of the model (`host/clamp_census.py`, output in `results/clamp_census.log`); the copy is asserted to equal

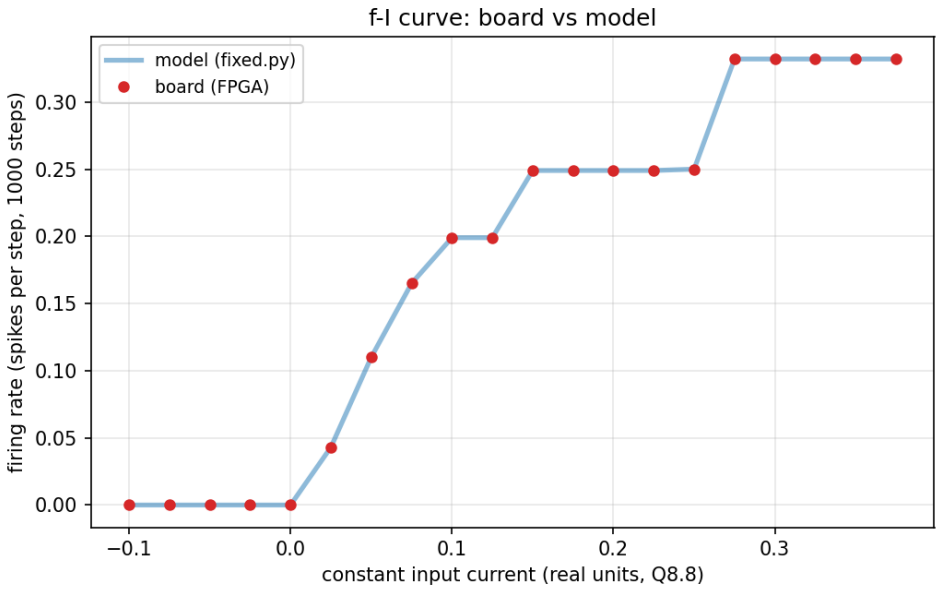


Figure 2: f–I points of parameter set A, board and model, for the 20 constant inputs. The input is in real units (the Q8.8 raw value divided by 256); the rate is spikes per time step over 1,000 steps, not Hz. Each step reflects an integer inter-spike interval: for a neuron that fires with a fixed interval T the rate is $1/T$, and the refractory period bounds the rate from above. The dead zone of the arithmetic shift is the offset it leaves in the leak. When the potential is below V_{rest} by less than 2^{k_m} raw units, the leak term rounds to zero, so the potential stops up to $2^{k_m} - 1$ raw units short of its target (15 raw, about 0.06, for $k_m = 4$). Near threshold this moves the onset of firing.

`fixed.step` at every step. The saturation test (`host/saturation_board_test.py`) uses sets S1 and S2, with the largest shifts $k_m = k_s = 15$, and constant inputs at the ends of the range, so that I reaches both limits and V its lower limit. Sets S3 and S4 use negative thresholds, which the lower limit of V can affect. No run reached the upper limit of v_2. The argument above needs only $\theta \leq 32{,}767$, which holds for every 16-bit threshold, positive or negative. The lower limit of V changes a spike decision only when $\theta = -32{,}768$, because only then does a value clamped to $-32{,}768$ satisfy $V \geq \theta$ where the unclamped value would not. Set S3 tests this case on the board: all 500 samples matched, with a spike at every step for negative input. Set S4 covers a negative threshold with ordinary shifts.

5.4 Network on the board

The 682-frame flow at $N = 256$ gave 1,892 reply bytes, all identical to the model, in 44.4 s (run of 2026-11-06, with the current `board_network_test.py`; log in `results/`). Five E/I conditions then ran on the board, each with 745 frames: $(f_{\text{inh}}, w_{\text{inh}}, w_{\max}, e_{\max}) = (0.2, 1, 10, 30)$, $(0.5, 1, 10, 30)$, $(0.5, 0.5, 10, 30)$, $(0.5, 2, 10, 30)$ and $(0.3, 1.5, 20, 10)$. For each, the spike counts, the last-step snapshot and the spikes of the first 512 steps matched the full model (logs `results/ei_board_*_2026-11-06.log`, run 2026-11-06 with `network_top`). Each run sent 745 frames: the starting `READ_STATUS`, 1 `WRITE_PARAM`, 656 `WRITE_WEIGHTS` (100 weights each), 3 `SET_INPUT`, 1 `RUN`, 1 `READ_STATUS`, 4 `READ_COUNTS`, 4 `READ_SNAPSHOT` and 74 `READ_TRACE`. Of these, 73 frames carry seven steps each and one frame carries one step, which together cover the 512 kept steps. The command flow of Table 2 has a different breakdown, because it also contains invalid frames and only three valid trace frames. Clamp events inside the network were not counted. Its membrane potential is compared only at the last step (Section 7), so a clamp there would not be

Table 5: Runs that reached a saturation limit, and their comparison with the model. Limits: I and V are the current and potential updates, with upper or lower limit. In the S3 and S4 row, the one run that reached no limit is S4 with input +4000 (raw); it still matched the model.

Runs	Limits reached	Runs with a clamp	Samples	Mismatches
Single neuron, set C (Table 3)	I upper, V lower	14 of 20	20,000	0
Sweep, grid (Table 4)	V lower	1,400 of 2,048 sets	1,843,200	0
Sweep, random (Table 4)	V lower	130 of 200 sets	180,000	0
Saturation test, sets S1 and S2	I upper and lower, V lower	10 of 10 runs	1,000	0
Saturation test, sets S3 and S4	I upper and lower, V lower	9 of 10 runs	1,000	0

Table 6: Test vectors of the saturation test on the board (`host/saturation_board_test.py`). Each set runs five constant inputs (raw −32768, −4000, −1000, +4000, +32767) for 100 steps from rest. Raw Q8.8 values.

Set	θ	V_{reset}	V_{rest}	k_m	k_s	*refrac*
S1	512	0	0	15	15	0
S2	512	−128	0	15	15	7
S3	−32768	0	0	15	15	0
S4	−128	−256	0	4	3	2

seen; the largest exact synaptic sums (before saturation) were 1,084 raw in the command flow and 160 to 378 raw in the E/I conditions (Section 3.3). The saturation of the single neuron is reported in Section 5.3.

5.5 Protocol robustness on the board

The command flow does not send malformed frames, so four such cases and a recovery read were run on the board separately (`host/protocol_robustness_board_test.py`, `results/`). Table 8 gives the outcome. All five cases were checked against the expected replies and the change of the error counter, and all five matched the prediction. A run of 64 steps precedes the cases, because `READ_RECORD` is valid only inside the last run. Case 3 receives a `READ_RECORD` reply, which only the single-neuron build sends (Section 7), so these tests ran on that build; the busy case was not run on the network build.

In case 4 the decoder has no time-out: it takes the first byte of the valid frame as the checksum of the truncated one, and the valid frame is lost. Case 5 is a further `READ_STATUS`, sent after case 4. Its reply arrived and the error counter stayed the same, so the decoder accepted the next valid frame after the truncated one. Only one truncated frame was tested.

5.6 E/I demonstration (model)

The demonstration uses random networks of 256 neurons with a connection probability of $8/N$, the last $f_{\text{inh}}N$ sources inhibitory (assigned by index, not drawn at random), and excitatory weights drawn from $[0, w_{\max}]$ with $w_{\max} = 10$. Inhibitory weights are $-w_{\text{inh}}$ times a weight of the same range, with $w_{\text{inh}} = 1$ unless stated. Each neuron has a constant external input drawn from $[0, e_{\max}]$ (Q8.8 raw) with $e_{\max} = 30$. Each run has 600 steps, and the first 100 are discarded. Table 9 gives the mean firing rate in spikes per step and per neuron, over 10 random networks. The spread is the minimum and maximum over the networks. The rate falls as f_{inh} rises, and the share of silent neurons rises. The full sweep covers 24 combinations of $w_{\max}$, w_{inh} and $e_{\max}$, each with seven values of f_{inh} and 10 networks (1,680 runs). Table 10 lists all 24. In every one, at least 57% of the neurons are above 0.3 spikes per step at $f_{\text{inh}} = 0$, so the sweep is dominated by the ceiling at low inhibition. In all 24 combinations the mean rate falls monotonically with

Table 7: Configuration of the networks run on the board. The command flow is generated by `host/gen_net_top_vectors.py` (`SEED`); the E/I conditions by `host/ei_board_test.py`, which uses the generator of `host/ei_full_sweep.py`. Weights and external inputs are drawn uniformly over the integer ranges shown; raw values are Q8.8. The command flow and the random sweep of Table 4 both use seed 20261103; they are separate generators with separate draws.

Parameter	Command flow	E/I board conditions
Neurons N	256	256
Neuron parameters (raw)	$\theta = 384$, $V_{\text{reset}} = 0$, $V_{\text{rest}} = 0$, $k_m = 4$, $k_s = 3$, $refrac = 2$	the same
Connections	probability $8/N$; zero weights absent	probability $8/N$; zero weights absent
Sources	80% excitatory ($j < 0.8N$), 20% inhibitory, by index	$1 - f_{\text{inh}}$ excitatory, f_{inh} inhibitory, by index (the last $f_{\text{inh}}N$ sources)
Weights	excitatory 0 to 60; inhibitory $-(0$ to $60)$	excitatory 0 to $w_{\max}$; inhibitory $-\text{round}((0 \text{ to } w_{\max})\, w_{\text{inh}})$
Weight scale w_{shift}	2	2
External input (raw)	0 to 50, per neuron	0 to $e_{\max}$, per neuron
Run length	200 steps	600 steps
Random seed	20261103	20261104 (external input: $20261104 \times 1000 + e_{\max}$)

Table 8: Malformed frames on the board, single-neuron build (`neuron_top`). The error counter is read before and after each case; case 5 must leave it unchanged.

Case	Sent	Replies	Error counter
1	`READ_STATUS` with a wrong checksum	none	+1
2	unknown command `0x0A`	none	+1
3	`READ_RECORD`, then `READ_STATUS` while the reply is being sent (busy)	`READ_RECORD` reply only	+1
4	truncated frame `A5 06 00`, then a valid `READ_STATUS`	none	+1
5	a further `READ_STATUS`, sent after case 4	`READ_STATUS` reply	unchanged

f_{inh}, and it does so in every individual network: none of the 1,440 consecutive pairs of inhibition levels, over the 240 networks, increases. The five board conditions use the same generator and the first seed (20261104), so each board condition is one of the networks of this sweep. We had to lower $w_{\max}$ from 60 to 10: with 60, the network was either silent or at the refractory limit of 1/3 spikes per step, and the external input no longer mattered. At $f_{\text{inh}} = 0$, on average 78% of neurons fire above 0.3 spikes per step, close to the refractory limit of 1/3 (65% to 84% across the networks). The decrease at the low-inhibition end is therefore partly a ceiling effect. In the networks of Table 9, the share of neurons above 0.3 spikes per step falls from 78% at $f_{\text{inh}} = 0$ to 30% at $f_{\text{inh}} = 0.2$ and 12% at 0.3. Table 10 shows that the share at $f_{\text{inh}} = 0.3$ varies from 0% to 82% across the combinations. The range from 0.3 to 0.6 is free of the ceiling only where that share is small, for example for $w_{\max} = 10$, $w_{\text{inh}} \geq 1$ and $e_{\max} \leq 30$ (at most 12%).

5.7 Link limits

The USB–UART link on the board lost bytes in long unpaced bursts: a single write of 96 or 128 bytes returned only 66 bytes in a loopback test, while 16-byte chunks paced to the line rate

Table 9: Mean firing rate against f_{inh} ($w_{\max} = 10$, $e_{\max} = 30$, $w_{\text{inh}} = 1$), averaged over 10 random networks, with the minimum–maximum over networks in brackets, the share of silent neurons, and the share of neurons above 0.3 spikes per step. Source: `build/ei_full_sweep.csv`, from `host/ei_full_sweep.py`, the generator also used for the board runs. Model only. Rates are per time step, not in Hz.

f_{inh}	Mean rate (min–max)	Silent neurons	Above 0.3 spikes per step
0.0	0.314 (0.301–0.320)	0.00	78%
0.1	0.290 (0.273–0.296)	0.01	57%
0.2	0.251 (0.243–0.259)	0.03	30%
0.3	0.210 (0.197–0.221)	0.07	12%
0.4	0.176 (0.163–0.189)	0.13	3%
0.5	0.143 (0.135–0.158)	0.20	1%
0.6	0.121 (0.106–0.137)	0.26	0%

all came back. We did not isolate the cause; it may lie in the bridge chip or in the host driver. The host therefore sends in 16-byte chunks at line speed, and waits for the full reply of a frame before it sends the next one. A run takes no reply, so the host waits for a time computed from the number of steps. This is a heuristic bound that we did not verify against the board's actual run time (Section 7). Across the board runs of this paper, the host sent about 87,000 request frames in 16-byte chunks: 80,928 in the 2,248 sweep sets (36 per set), 682 in the command flow, 3,725 in the five E/I conditions, and about 2,100 in the single-neuron and saturation runs. Every expected reply arrived, and no run recorded a failed read. This is not a measured error rate under unpaced load, and we did not measure worst-case latency.

6 Resources and scaling

Table 11 gives the place-and-route result of the full design at $N = 256$. The single-neuron logic (`lif_core` alone, an early yosys run without the state memory) needed about 274 LUTs. The yosys cell counts of the network core, 1,382, 1,343 and 1,457 at $N = 16, 64, 256$, include I/O buffers and the RAM16SDP4 cells that the tool uses for small memories, so they are not comparable across N. Block RAM is what limits the design: the weight matrix alone is $N^2 \times 8$ bits (512 Kbit), and the spike trace takes eight more blocks. With 45 of 46 blocks used, we cannot add a feature without first freeing memory. For $N = 512$ the weight matrix would be 2 Mbit, more than the roughly 828 Kbit of block RAM on the chip. This is arithmetic from the N^2 weight matrix, not a built design.

7 Limitations

- **Biology.** The hardware is validated against the model only. The E/I demonstration uses synthetic random networks, weights that are not calibrated against data, and rates per time step, not in Hz. We draw no conclusion about neural tissue.
- **Saturation.** The three clamps that can change the output were reached on the board, and every saturated step matched the model (Section 5.3). The upper clamp of the potential was not reached in any traced run, and it cannot change the output. Clamps inside the network were not counted, and the network's potential was compared only at the last step.
- **Stimulus.** On the board, the input is a constant per neuron, and the sweep uses three constant inputs for 300 steps. Time-varying input was not tested on the board, and the protocol cannot send it: `RUN` and `SET_INPUT` carry one constant value. All 2,083,200

Table 10: All 24 combinations of the E/I sweep, each averaged over 10 random networks (model only, `build/ei_full_sweep.csv`). "Ceiling share" is the share of neurons above 0.3 spikes per step at $f_{\text{inh}} = 0$ and at 0.3. "Monotone" means the mean rate never rises with f_{inh} in any of the 10 networks.

$w_{\max}$	w_{inh}	$e_{\max}$	$\bar{r}$ at f_{inh}=0	$\bar{r}$ at 0.3	$\bar{r}$ at 0.6	Ceiling at 0	Ceiling at 0.3	Monotone
10	0.5	10	0.293	0.186	0.078	57%	1%	yes
10	0.5	30	0.314	0.246	0.161	78%	23%	yes
10	0.5	60	0.324	0.290	0.227	90%	61%	yes
10	1.0	10	0.293	0.138	0.052	57%	0%	yes
10	1.0	30	0.314	0.210	0.121	78%	12%	yes
10	1.0	60	0.324	0.266	0.184	90%	47%	yes
10	1.5	10	0.293	0.108	0.042	57%	0%	yes
10	1.5	30	0.314	0.180	0.099	78%	6%	yes
10	1.5	60	0.324	0.241	0.156	90%	37%	yes
10	2.0	10	0.293	0.091	0.037	57%	0%	yes
10	2.0	30	0.314	0.158	0.086	78%	4%	yes
10	2.0	60	0.324	0.218	0.138	90%	30%	yes
20	0.5	10	0.328	0.263	0.096	95%	50%	yes
20	0.5	30	0.331	0.291	0.167	97%	67%	yes
20	0.5	60	0.332	0.311	0.226	99%	82%	yes
20	1.0	10	0.328	0.199	0.060	95%	25%	yes
20	1.0	30	0.331	0.245	0.111	97%	46%	yes
20	1.0	60	0.332	0.280	0.168	99%	67%	yes
20	1.5	10	0.328	0.148	0.048	95%	11%	yes
20	1.5	30	0.331	0.200	0.089	97%	32%	yes
20	1.5	60	0.332	0.243	0.135	99%	52%	yes
20	2.0	10	0.328	0.124	0.042	95%	7%	yes
20	2.0	30	0.331	0.170	0.079	97%	23%	yes
20	2.0	60	0.332	0.213	0.117	99%	42%	yes

single-neuron samples therefore use constant inputs, and their agreement says nothing about time-varying input.

- **Membrane potential in the network.** The board compares spikes for the first 512 steps, and V only at the last step. An intermediate V would need a run of that length from rest for each step. A run of more than 512 steps keeps only the counts and the last snapshot after step 512. In the 600-step E/I runs, steps 513 to 600 (88 of 600 steps, 15%) have no spike-timing comparison; only their spike counts and the final snapshot are checked. Closing this gap would need a state read at step 512 or a longer trace memory.

- **Reference model.** The Python model is the only reference. It was written with AI assistance (Section 4), and a misreading of the specification would be shared by the model and the hardware, so the board comparison cannot detect it. The unit tests in `model/test_fixed.py` check the arithmetic and the update rules on selected cases; they are not an independent implementation. Floating-point simulators are not bit-exact and cannot serve as that reference.

- **Mutation analysis.** Eighteen hand-written mutants were run (Table 1), each against one checker and all in simulation. No full kill matrix was produced. The mutated copies are not kept as files in the repository, so the table is a record of the checks, not reproducible code.

- **No netlist simulation.** The block-RAM models of the installed toolchain are empty. The Gowin block-RAM mapping printed "range select out of bounds" warnings for the network build (log `make_network_top_2026-11-04.log`, in `fpga/build/`). Each sets some bits of a memory port to undefined. The log does not name the memories, and we did not

Table 11: Full design at $N = 256$ (nextpnr-himbaechel, GW2AR-18). Source: `fpga/build/make_network_top_2026-11-04.log`, a complete rebuild (`make -B TOP=network_top`); the maximum clock is the critical-path report after routing.

Resource	Used	Available
LUT4	2,536	20,736 (12%)
Flip-flops	1,315	15,552 (8%)
Block RAM	45	46 (98%)
Maximum clock, after routing	38.75 MHz	27 MHz target, met

analyse the netlist to show that the undefined bits are never used. The board runs are the only evidence, and they cover only the cases we ran.

- **Parameter space.** All neurons share one parameter set, there is no noise source, and the grid sweep fixes θ, V_{reset} and V_{rest} (the random sweep covers them). The thresholds tested on the board are 64 to 1,024 raw in the random sweep, 128, 256 and 512 in the other sets, and $-32{,}768$ and -128 in S3 and S4. Other negative thresholds were not tested.

- **Protocol.** The additive checksum does not detect swapped bytes. The decoder has no time-out, so a corrupted length field can misalign the frame stream. Adding one would need an RTL change, new simulation tests and a new board run, and we have not made that change. Reply behaviour depends on the build. In the network build, `READ_RECORD` is accepted and counted as successful but sends no reply. In the single-neuron build, `READ_TRACE` is accepted without a reply. The host sends `READ_RECORD` only to the single-neuron build, where it replies, and `READ_TRACE` only to the network build, where it replies. The board tests of Table 8 ran on the single-neuron build, and the busy case was not run on the network build. The board has no reset command, so its counters persist until the bitstream is reloaded, and the command-flow test must start from a freshly loaded board.

- **Timing.** The wait after a run is computed from the number of steps as a worst-case heuristic. We did not verify it against the time the board actually needs.

- **Design choices.** The protocol, the weight scale w_{shift} and the choice of the E/I test network are our own choices. They were not tuned or compared with alternatives.

- **Connectivity.** The nominal connection probability is $8/N$, but zero weights are excluded, so the effective density is lower. For $f_{\text{inh}} = 0.2$ the networks have on average 2.8% nonzero entries of the 256×256 matrix (range 2.7% to 2.9%), against a nominal 3.1%.

- **Samples.** Each E/I data point uses 10 random networks and one external-input sample per network. The board ran five conditions of the 24 combinations of the sweep.

8 Conclusion

We implemented a fixed-point LIF neuron and a 256-neuron network on a low-cost FPGA, and compared both with a Python reference model. The single neuron's membrane potential and spike matched the model step by step on the board, over 2,083,200 constant-input samples. The synaptic current was not read back. The network matched the model in its command replies, its spike counts and final-state snapshot, and in the first 512 steps of spikes for five E/I conditions. The three clamps that can change the output were reached on the board and matched the model step by step (Section 5.3). The membrane potential of the network was compared only at the last step, and clamps inside the network were not counted. The E/I results are model results for synthetic networks and are not a biological claim.

Availability

The code, RTL, testbenches, host scripts and raw results are available in the public repository `https://github.com/KiraIch/fpga-lif-snn`, branch `m3-network`. The raw results (sweep CSVs, board logs and place-and-route logs) are in the `results/` directory. A release of the code is archived on Zenodo with the DOI 10.5281/zenodo.24013358.

Declarations

Competing interests. The author declares no competing interests.
Funding. This work received no funding.
Author contributions. The work was carried out independently by the author, ChiaAn Lin.
Ethics. Not applicable: no human or animal data were used.

References

[1] Wiktor J. Szczerek and Artur Podobas. A quarter of a century of neuromorphic architectures on FPGAs – an overview, 2025. arXiv:2502.20415v4, 18 May 2026.

[2] Andrew Fan and Simon D. Levy. A robust, open-source framework for spiking neural networks on low-end FPGAs, 2025. arXiv:2507.07284v2, 23 July 2025.

[3] Ashish Gautam, Prasanna Date, Shruti Kulkarni, Robert Patton, and Thomas Potok. NeuroCoreX: An open-source FPGA-based spiking neural network emulator with on-chip learning, 2025. arXiv:2506.14138, preprint.

[4] Reza Ansari and Maciej Wielgosz. A time-multiplexed spiking neural network accelerator with pipelined readout for FPGA inference, 2026. arXiv:2608.00595v1, preprint, not peer reviewed.

[5] Kit Cheung, Simon R. Schultz, and Philip H. W. Leong. A parallel spiking neural network simulator. In *International Conference on Field-Programmable Technology (FPT)*, 2009. URL `https://cel.eng.sydney.edu.au/wp-content/uploads/2019/01/snn_fpt09.pdf`.