
Hardware-Software Co-Design of Scalable, Energy-Efficient Analog Recurrent Computations

Arthur Fyon
University of Liège
Liège, Belgium
afyon@uliege.be

Julien Brandoit
University of Liège
Liège, Belgium
jbrandoit@uliege.be

Loris Mendolia
University of Liège
Liège, Belgium
lmendolia@uliege.be

Damien Ernst
University of Liège
Liège, Belgium
dernst@uliege.be

Jean-Michel Redouté
University of Liège
Liège, Belgium
jean-michel.redoute@uliege.be

Guillaume Drion
University of Liège
Liège, Belgium
gdrion@uliege.be

Abstract

Always-on AI applications, from environmental sensors to biomedical implants, require ultra-low power consumption. Analog circuits offer a path to sub-microwatt inference, yet existing analog implementations are limited to feedforward architectures: extending them to recurrent dynamics has been considered impractical due to noise accumulation through temporal feedback. We demonstrate that this barrier can be overcome through hardware-software co-design. Specifically, we identify that Bistable Memory Recurrent Units (BMRUs), a class of Recurrent Neural Networks (RNNs) with discrete-valued outputs and hysteretic dynamics, admit an ultra-low power current-mode analog implementation which we design from first principles. The resulting circuit establishes a one-to-one correspondence between each learned parameter and a circuit element. The discrete outputs suppress analog noise by at least 20-fold at each cell boundary, breaking the noise accumulation that prevents analog recurrence. We reformulate BMRUs for first-quadrant operation with fixed thresholds, enabling the direct correspondence while preserving expressivity and trainability. Transistor-level simulations in 180 nm Complementary Metal-Oxide-Semiconductor (CMOS) show near-perfect agreement between software predictions and circuit-level behavior, with the software model thereby serving as a high-fidelity simulator of the physical hardware at low computational cost. We leverage this fidelity to conduct large-scale noise immunity and power scaling analyses: the power cost of adding recurrence scales linearly with state dimension, while the feedforward layers dominating total power scale quadratically, meaning recurrence is added at linear marginal cost relative to the feedforward backbone. End-to-end keyword spotting achieves sub-microwatt inference at the RNN core. To our knowledge, this is the first demonstration of an analog recurrent neural network with structural noise immunity.

1 Introduction

The computational demands of modern Artificial Intelligence (AI) have grown exponentially, now consuming significant portions of global energy resources [1–4]. This challenge becomes particularly acute for always-on applications where battery life is paramount: environmental sensor networks, biomedical monitoring devices, and edge devices processing real-world sequential data streams [5–7]. Current systems rely on general-purpose digital hardware inherited from Von Neumann architectures that separate memory and computation [8], necessitating constant data movement and substantial energy overhead [9, 10].

Modern sequence-based AI is dominated by Transformers, whose attention mechanisms achieve state-of-the-art performance across domains [11]. However, their quadratic complexity in sequence length and large memory footprint make them unsuitable for resource-constrained edge devices [12, 13]. State-Space Models (SSMs) and RNNs offer more efficient alternatives for sequential processing. SSMs achieve parallelizable training through linear recurrence [14–19], yet their transient dynamics, states decaying exponentially toward equilibrium, prevent multistability [20], limiting performance on tasks requiring persistent memory [21, 22]. Nonlinear RNNs provide multistability, essential for long-term memory [23], through nonlinear recurrent dynamics [24, 25], but their inherently sequential nature creates training bottlenecks [26, 27].

Orthogonal to this algorithmic divide lies the hardware implementation challenge. Application-Specific Integrated Circuits (ASICs) achieve efficiency gains through parallelized operations and quantization [28, 29], with the TinyML movement extending this to microcontrollers, typically enabling milliwatt-level inference [30–34]. Yet these systems remain fundamentally digital, constrained by sequential execution and separated memory [35]. Neuromorphic approaches offer alternatives: digital chips like Intel Loihi 2 implement Spiking Neural Networks (SNNs) with event-driven efficiency [36–38], while analog crossbar arrays achieve in-memory computation through Kirchhoff’s law summation [39–42]. However, SNN performance still lags behind conventional RNNs on many sequential tasks [43–46], and while digital FPGA implementations of recurrent architectures exist [47], analog implementations focus predominantly on feedforward networks [48, 49]: extending them to recurrent dynamics introduces noise accumulation through temporal feedback across timesteps [50]. Programmable analog processors such as the Aspinity AML100 [51] perform always-on event detection entirely in the analog domain at tens of microwatts, and dedicated digital neural decision processors such as the Syntiant NDP120 [52] achieve inference at hundreds of microwatts with support for Convolutional Neural Networks (CNNs), RNNs, and Fully Connected (FC) networks. Yet no existing fully analog approach implements recurrent dynamics: the AML100 is restricted to feedforward signal processing, and all sub-milliwatt digital processors remain orders of magnitude above what subthreshold analog circuits can achieve. *No existing approach combines parallelizable training, multistable memory, analog computation, and sub-microwatt power consumption.*

In this paper, we focus on a specific class of recurrent architectures that addresses both the algorithmic and hardware challenges. On the algorithmic side, we seek parallelizable training while maintaining multistability for persistent memory. On the hardware side, we seek architectures whose computational primitives align naturally with ultra-low power analog circuits. Memory Recurrent Units (MRUs) satisfy the first requirement [53]. Unlike traditional RNNs requiring iterative settling [54], MRUs combine instantaneous convergence with multistability, enabling parallel training via associative scans [55] while maintaining persistent memory without transient dynamics. The Bistable MRU (BMRU) variant achieves this through hysteresis, producing discrete-valued outputs. As we demonstrate in this paper, these discrete outputs and hysteretic dynamics can be matched by an ultra-low power current-mode analog circuit that we design from first principles, enabling genuine hardware-software co-design. We leverage this co-design to create the first fully analog RNN with four important characteristics:

One-to-one hardware-software correspondence. Rather than a loose algorithmic inspiration, each learned parameter in the reformulated BMRU maps to a single, programmable circuit element: thresholds become bias currents and weights become transistor width ratios. Cadence Spectre simulations confirm near-perfect agreement between the software model and transistor-level dynamics, not only on final predictions but at every intermediate signal throughout the network (Section 3.2). This fidelity turns the software model into a proxy for the physical circuit, enabling large-scale analyses that would be prohibitively expensive in transistor-level simulation.

Structural noise immunity. Discrete-valued BMRU outputs block analog noise at each cell boundary, enabling robust recurrent architectures despite device variability. Leveraging the high-fidelity software simulator, we conduct a large-scale noise analysis across multiple benchmarks (Section 4) that confirms that this immunity holds under realistic operating conditions.

Recurrence at low marginal cost. The BMRUs scale linearly in power with state dimension, while the FC layers scale quadratically. Component-level power breakdown (Section 4) shows that at realistic network sizes, recurrence adds a small, linearly-scaling overhead on top of the feedforward cost. This means any analog feedforward platform could, in principle, gain temporal processing capabilities by integrating BMRU cells at marginal additional cost.

Sub-microwatt operation. Subthreshold operation with picoampere currents achieves approximately 100 nW for the RNN inference core of a minimal proof-of-concept network on Keyword Spotting (KWS). Additional overhead from bias generation, shift registers, and routing remains well within the sub-microwatt envelope based on component-level estimates (Appendix K). Clockless continuous-time analog dynamics eliminate clock overhead entirely, with latency reducing to propagation delay.

We validate our approach through Cadence transistor-level simulations in X-FAB 180 nm CMOS, demonstrating end-to-end feasibility on KWS using the Google Speech Commands dataset, a standard benchmark for always-on edge applications [56] (Section 3). We also benchmark the proposed cell against LRU [57] and minGRU [58] on standard sequence modeling tasks and on character-level language modeling to highlight the scalability of the proposed approach. A comprehensive review of related work is provided in Appendix A.

2 Hardware-software co-design of an analog RNN

We build the hardware-software co-design in three steps: identifying a recurrent cell that is amenable to monolithic analog implementation (Section 2.1), designing a matching ultra-low power current-mode circuit (Section 2.2), and adapting the cell equations to achieve one-to-one mapping between learned parameter maps and circuit elements (Section 2.3).

2.1 Recurrent cell selection

Three properties are essential for an analog RNN target. First, *competitive performance* on standard sequence benchmarks. Second, *parallelizable training* through associative scans [55], so that the model can be trained at scale on GPUs. Third, and most importantly for monolithic analog integration, *persistent memory without transient dynamics*: cells that converge instantaneously rather than through exponential settling eliminate the need for on-chip capacitors. This is what distinguishes BMRUs [53] from SSMs and classical RNNs. Together with the discrete-valued outputs that we exploit for noise immunity (Section 4), these properties make the BMRU a natural candidate for monolithic, capacitor-free, transistor-only analog implementation.

The BMRU combines these properties through a formulation based on bistability [53]:

$$\hat{h}_t = W_x x_t + b_x, \quad (1)$$

$$\beta_t = |W_\beta x_t + b_\beta|, \quad (2)$$

$$z_t = \mathcal{H}(|\hat{h}_t| - \beta_t), \quad (3)$$

$$h_t = z_t \odot S(\hat{h}_t) \odot \alpha + (1 - z_t) \odot h_{t-1}, \quad (4)$$

where x_t and h_t are the current input and hidden state, respectively, and β_t is an input-dependent threshold. $\mathcal{H}(\cdot)$ denotes the Heaviside function, $S(\cdot)$ is the sign function, $\odot$ denotes the Hadamard product, W_x and W_β are learnable parameter matrices, and b_x , b_β and α are learnable parameters.

This formulation implements adaptive thresholding with hysteresis. When the candidate magnitude $|\hat{h}_t|$ exceeds the input-dependent threshold β_t , the gate z_t opens and the hidden state is mapped to $S(\hat{h}_t) \cdot \alpha \in \{-\alpha, +\alpha\}$. A state change only occurs when $S(\hat{h}_t) \neq S(h_{t-1})$, that is, when the candidate and current state have opposite signs. When $|\hat{h}_t| < \beta_t$, the gate remains closed ($z_t = 0$) and the hidden state persists unchanged. Two properties then characterize the BMRU: *instantaneous convergence*, since the piecewise-constant activation removes iterative settling dynamics and enables parallel computation across timesteps during training; and *discrete-valued outputs*, since hidden states

are restricted to $\{-\alpha, +\alpha\}$. These two properties are what monolithic analog recurrence has been missing. Instantaneous convergence eliminates capacitive settling, enabling capacitor-free integration; discrete outputs introduce a thresholding nonlinearity at every cell boundary, blocking the noise propagation that has historically prevented analog RNN implementations.

Beyond these implementation-friendly properties, recent work in reinforcement learning demonstrates that BMRU-based agents trained on short-horizon environments generalize to substantially longer horizons at deployment [59], providing evidence that the cell captures transferable temporal structure.

2.2 Analog circuit implementation

The properties described above allow for a simple implementation following three guiding principles (full circuit details appear in Appendix D).

Current-mode operation. All signals are represented as currents, exploiting the exponential I - V relationship of subthreshold MOS transistors for natural dynamic range compression and low power operation at picoampere signal levels. Addition and subtraction become trivial through Kirchhoff’s Current Law (KCL).

Transistor-only design. By eliminating resistors, capacitors, and inductors, the circuit achieves true monolithic integration without external passives, minimizing area and enabling full CMOS compatibility. The BMRU instantaneous convergence eliminates the need for capacitors that would be required to implement the transient dynamics of classical RNNs or SSMs.

Parameter mapping. All learned parameters map directly to circuit elements: BMRU thresholds and amplitudes correspond to bias currents, while FC layer weights correspond to transistor width ratios in current mirrors. In our proof of concept, these are fixed at design time; Section 5 discusses a path toward post-fabrication programmability through binary-weighted current mirror banks.

Bistable cell. We design a current-mode bistable cell based on a dual-Heaviside feedback structure (Figure 1A). It achieves independent programmability of three key parameters through bias currents: the threshold current I_{thresh} determines the high switching point, the output current I_{gain} sets the high-state amplitude, and the hysteresis width I_{width} tunes the memory region. All three parameters are set by external bias currents independently, which can be tuned by adapting their respective gate voltage. Each cell requires 9 transistors, all operating in subthreshold, achieving per-cell nanowatt power consumption.

Fully connected layers. Matrix-vector multiplication ($y = Wx$) exploits KCL for analog summation using standard current-mirror techniques. Each input current x_i feeds a bank of current mirrors producing partial currents $y_{ij} = w_{ij}x_i$, which sum at each output node: $y_j = \sum_i y_{ij}$. Transistor width ratios encode weight magnitudes:

$$I_{\text{out},ij} = \frac{W_{\text{out},ij}}{W_{\text{in},i}} \cdot I_{\text{in},i}, \quad I_{\text{out},j} = \sum_i I_{\text{out},ij}, \quad (5)$$

where $W_{\text{out},ij}/W_{\text{in},i}$ represents the width ratio implementing weight w_{ij} . Bias terms add through dedicated current sources. Weight precision is determined by layout matching [60, 61], typically achieving 6–8 bit equivalent resolution in standard CMOS processes.

Rectified linear units. The ReLU activation, placed after FC layers, emerges naturally from diode-connected NMOS/PMOS transistors, which conduct current only in one direction [62]. This implements $\text{ReLU}(I_{\text{out},j}) = \max(0, I_{\text{out},j})$ with no additional power overhead.

Complete architecture. The full network comprises an input projection layer, N stacked BMRU layers with inter-layer FC transformations, and an output classifier producing class currents. All components operate in the current domain with subthreshold biasing. All parameters, FC weights, BMRU thresholds, gains, and hysteresis widths, are set by bias currents, enabling the same fabricated chip to implement different trained networks.

2.3 First-quadrant BMRU

The current-mode implementation described in Section 2.2 operates with unipolar (positive-only) currents and fixed threshold parameters set by DC biases. The standard BMRU formulation presents two mismatches with these constraints. First, the sign function in Equation (4) produces bipolar outputs

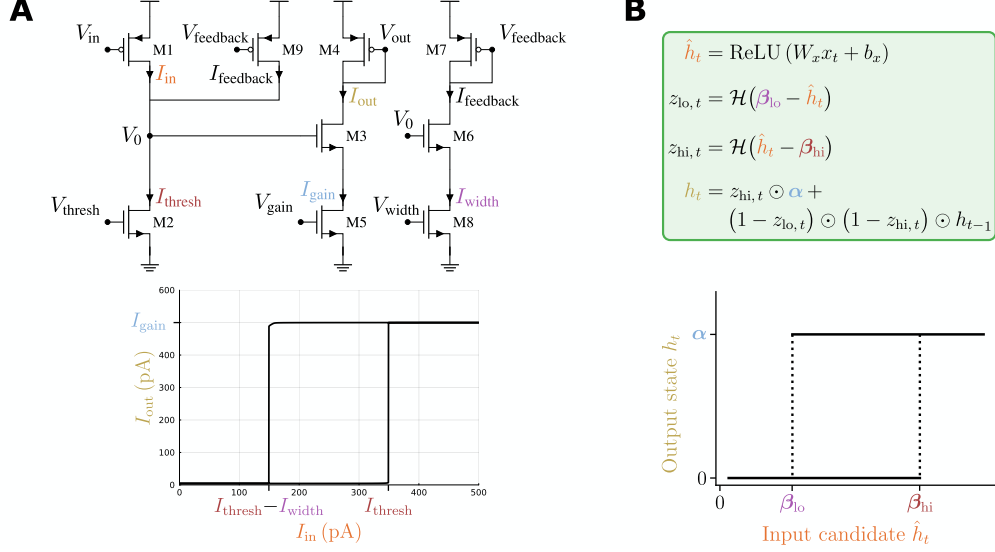


Figure 1: **Current-mode analog implementation and FQ BMRU formulation.** **A.** Schematic of the ultra-low power current-mode bistable cell (top) and associated input-output current relationship (bottom). All thresholds and output gain are independently tunable via bias currents. **B.** FQ BMRU equations (top) and input candidate versus state relationship (bottom), with α , β_{lo} and β_{hi} as learnable parameters. The correspondence between panels A and B illustrates the one-to-one mapping: each learned parameter maps directly to a tunable circuit parameter.

$\{-\alpha, +\alpha\}$, requiring differential signaling or dual power supplies, which increases circuit complexity and power consumption. Second, computing β_t implies a linear transformation in Equation (2) and demands dedicated multiplication and addition circuitry evaluated at each timestep, adding substantial overhead to the fixed-bias tunability of the analog implementation. These mismatches motivate a reformulation that preserves the essential properties of the BMRU while aligning with current-mode circuit constraints. We propose a First-Quadrant BMRU (FQ BMRU) variant operating entirely with non-negative values (Figure 1B) that preserves expressivity (Appendix B):

$$\hat{h}_t = \text{ReLU}(W_x x_t + b_x), \quad (6)$$

$$z_{lo,t} = \mathcal{H}(\beta_{lo} - \hat{h}_t), \quad (7)$$

$$z_{hi,t} = \mathcal{H}(\hat{h}_t - \beta_{hi}), \quad (8)$$

$$h_t = z_{hi,t} \odot \alpha + (1 - z_{lo,t}) \odot (1 - z_{hi,t}) \odot h_{t-1}, \quad (9)$$

where β_{lo} and β_{hi} are learned positive threshold parameters with $\beta_{hi} > \beta_{lo}$ (element-wise).

This formulation implements a window comparator with three operating regions. When $\hat{h}_t < \beta_{lo}$, the state resets (or stays) to zero. When $\hat{h}_t > \beta_{hi}$, the state sets (or stays) to α . When $\beta_{lo} \leq \hat{h}_t \leq \beta_{hi}$, the state persists from the previous timestep. The hysteresis window $[\beta_{lo}, \beta_{hi}]$ provides noise immunity while enabling bistable memory. This reformulation offers three advantages for hardware implementation. (i) *Unipolar signals*: input candidate $\hat{h}_t$ and hidden state h_t remain non-negative, enabling single-ended current-mode circuits without differential pairs or negative supplies. (ii) *Fixed thresholds*: parameters β_{lo} and β_{hi} are learned during training but remain constant during inference, mapping directly to fixed DC bias currents. (iii) *Direct parameter correspondence*: each learned parameter maps to a specific circuit element, with $\hat{h}_t$ corresponding to the input current I_{in} , h_t to the output current I_{out} , β_{hi} to I_{thres} , β_{lo} to $I_{thres} - I_{width}$, and α to I_{gain} (color coded in Figure 1).

Empirical trainability. While expressivity is preserved in theory, the reformulation could still affect gradient-based training. We compare the FQ BMRU against the original BMRU and two recent parallelizable RNN baselines, LRU [57] and minGRU [58], on a unified suite of sequence modeling benchmarks spanning pixel-level image classification (sequential and permuted MNIST), spoken-digit keyword spotting (dKWS), the ListOps task from the Long Range Arena [63], and character-level language modeling on Shakespeare [58, 64] (Table 1). For classification tasks, loss is

Table 1: **Unified benchmark results.** Accuracy (%) on classification tasks (higher is better); cross-entropy loss on Shakespeare character-level language modeling (lower is better). Classification models use two recurrent layers with state dimension 64; Shakespeare uses a depth-6 model with state and model dimension 256 following Feng et al. [58]. ListOps follows the Long Range Arena protocol [63]. Results averaged over 5 seeds. *LRU and minGRU use identical architecture and training hyperparameters.

Model	Accuracy (%) $\uparrow$				Loss $\downarrow$
	sMNIST	pMNIST	dKWS	ListOps	Shakespeare
BMRU	95.4	94.2	94.8	41.7	1.439
	[94.9; 96.3]	[92.9; 95.8]	[94.4; 95.2]	[41.0; 42.7]	[1.437; 1.446]
FQ BMRU	95.6	94.8	92.9	37.5	1.484
	[95.2; 96.0]	[94.4; 95.2]	[92.7; 93.2]	[37.0; 38.3]	[1.477; 1.492]
LRU*	92.6	95.9	96.4	41.0	1.480
	[90.4; 93.8]	[95.0; 96.7]	[95.9; 96.8]	[37.9; 49.3]	[1.474; 1.487]
minGRU*	93.4	95.6	96.3	39.0	1.451
	[88.5; 96.5]	[94.9; 96.6]	[96.0; 96.6]	[38.4; 39.4]	[1.445; 1.455]

computed via average pooling over timesteps and results are averaged over 5 seeds. Across tasks, the FQ BMRU matches or closely approaches the baselines on most tasks, with the largest gap appearing on ListOps. We attribute this gap to the optimization cost of fixed thresholds rather than to reduced representational capacity, consistent with the expressivity-versus-trainability distinction above. Notably, no analog co-design approach exists for LRU or minGRU, making FQ BMRU the only architecture with a demonstrated path to ultra-low power analog implementation. Complete training procedures are detailed in Appendix C.

3 Proof of concept: keyword spotting

We validate the co-design through a complete pipeline: the network is trained entirely in software, and the learned parameters are then mapped to transistor dimensions and bias currents for inference in transistor-level Cadence simulations. The target task is single-word (“yes”) keyword spotting using the Google Speech Commands dataset [56] (Figure 2A, bottom), a standard benchmark for always-on edge applications requiring continuous audio monitoring with minimal power [65–68].

Audio preprocessing extracts 13-dimensional Mel-Frequency Cepstral Coefficients (MFCCs) [69] from 1-second clips at 100 frames per second, yielding input sequences of 101 timesteps. For this proof-of-concept demonstration, MFCC extraction occurs off-chip. MFCC front-ends operating at comparable power levels have been demonstrated [70, 71], confirming that a complete signal chain from microphone to classification at microwatt-level total power is feasible. On-chip feature extraction remains a direction for future integration. The network architecture comprises an input projection layer mapping 13-dimensional MFCC features to the recurrent state dimension d , N stacked FQ BMRU layers, and a binary output classifier (Figure 2A, top). Trained parameters, FC weights and BMRU thresholds, are then mapped to transistor width ratios and bias currents respectively for Cadence transistor-level simulation.

The complete analog circuit (768 transistors) is implemented in the 180 nm CMOS X-FAB technology using Cadence Virtuoso [72]. All simulations are conducted using Spectre with PRIMLIB 180 nm models (ne/pe devices) at a supply voltage of 1.8 V. Transient simulations validate inference on test samples, with input MFCC sequences converted to current waveforms and output class currents compared to determine classification decisions.

3.1 Software baseline

We evaluated FQ BMRU accuracy on balanced test sets comprising all 397 positive samples (“yes”) paired with 397 randomly selected negative samples (other words or noise) from the Google Speech Commands dataset (accuracy table and complete training procedure can be found in Appendix C). Even with a minimal $d = 4$ configuration, the network achieves 93.86 % accuracy. Performance improves steadily with increasing state dimension, reaching 96.98 % at $d = 8$ and 98.08 % at $d = 16$, before plateauing around 97.5 % for $d \geq 64$. The consistently tight min-max ranges indicate stable

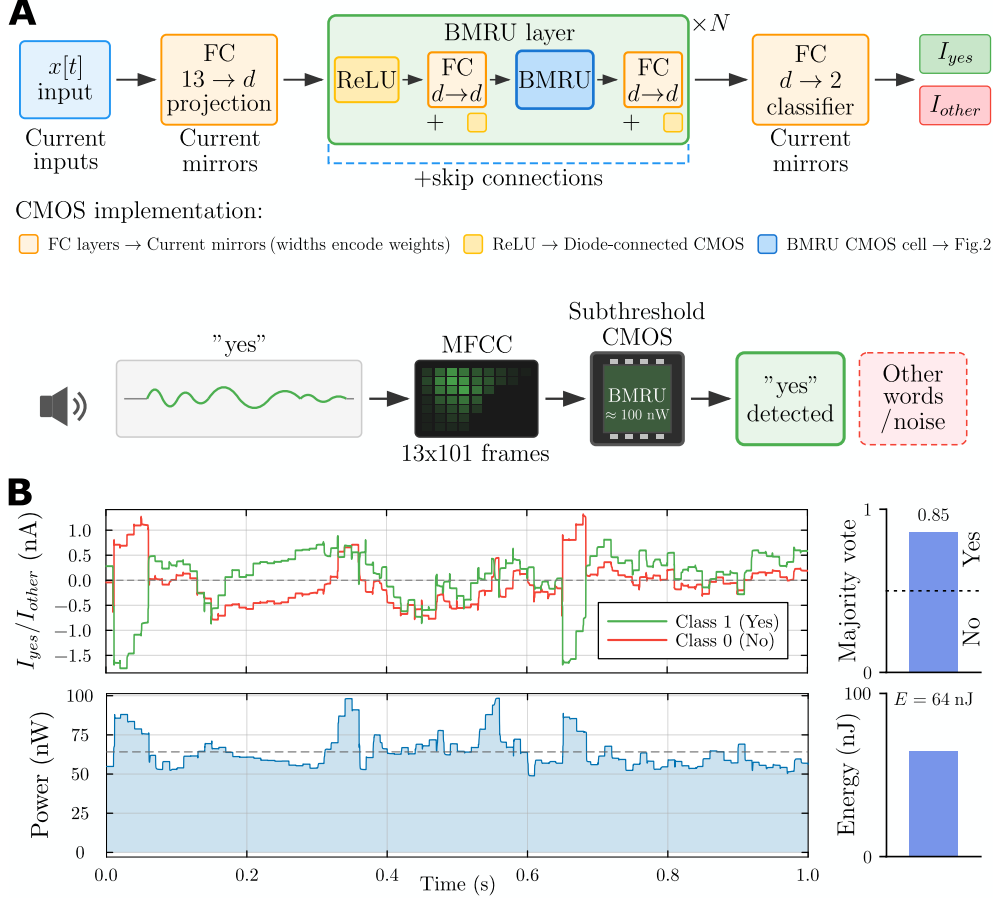


Figure 2: Analog CMOS implementation of a complete BMRU-based RNN for “yes” KWS. A. Complete network architecture where all operations are computed using analog primitives whose behavior emerges from the physical properties of subthreshold transistors (top). For this proof of concept, a minimal configuration with $N = 2$ layers and state dimension $d = 4$ is implemented. KWS task for “yes” recognition. MFCC extraction occurs off-chip for simplicity; future implementations will integrate full inference from raw audio (bottom). The analog chip produces two output currents representing class logits. **B.** Positive inference example simulated in Cadence. Top: time evolution of output logit currents, with majority vote yielding correct classification. Bottom: instantaneous power consumption remaining below 100 nW for the RNN core.

training and good generalization despite the random sampling. The primary goal of this work is to demonstrate end-to-end co-design feasibility rather than maximizing accuracy; accordingly, hardware validation focuses on the minimal $d = 4$ configuration to establish the complete pipeline from training to analog inference. Preliminary quantization experiments confirm that restricting all parameters to 4-bit precision degrades accuracy negligibly relative to full precision, even without quantization-aware training (Appendix C.3).

3.2 Hardware validation

Cadence simulations of the complete analog implementation with $d = 4$ demonstrate successful inference. Figure 2B illustrates representative inference traces, showing the output class currents I_{yes} and I_{other} over time for a positive sample. The BMRU cells exhibit clean switching behavior, and the correct class consistently produces higher output current, enabling reliable discrimination. Across 50 randomly selected test samples, hardware predictions match software in 49 cases. The single discrepancy occurs on a sample where software confidence was minimal (51 vs. 50 timestep majority), a tie rather than a systematic error. Additional inference traces covering positive, negative, and boundary samples appear in Appendix F.

Operating entirely in subthreshold, the circuit achieves approximately 100 nW average power consumption during continuous inference for the RNN core at the $d = 4$ proof of concept (Figure 2B bottom). A component-level power breakdown (Appendix E) confirms the approximately even split between BMRU cells and FC layers at $d = 4$, with BMRU cells exhibiting substantially lower power variance across inferences, consistent with stable discrete-output dynamics.

We validate robustness to manufacturing and environmental variation through transistor-level Monte Carlo and corner simulations. Monte Carlo mismatch analysis (200 samples, 3σ on all transistors) confirms that misclassifications under mismatch occur only when the network is already uncertain under nominal conditions (Appendix H). The bistable dynamics provide inherent mismatch immunity: discrete thresholding ensures that small parameter perturbations do not propagate to output errors unless the input is already near the decision boundary. PVT (Process, Voltage, Temperature) corner analysis validates robustness across all five process corners (TT, FF, SS, FS, SF), the temperature range $[-27^\circ\text{C}, 81^\circ\text{C}]$, and $\pm 10\%$ supply voltage variation (Appendix G). Classification correctness is maintained across all conditions for representative inferences.

Beyond matching final predictions, the hardware implementation matches the software model at every intermediate stage. Appendix J provides a stage-by-stage comparison between software predictions and Cadence-simulated signals at the input projection, both FQ BMRU layer candidates and states, the skip connection, and the output logits. The close agreement throughout validates the one-to-one correspondence between learned parameters and circuit elements, and demonstrates that the FQ reformulation produces a faithful mapping between the algorithmic and physical domains.

The signal-level comparison also reveals the noise suppression mechanism in action. At the second recurrent layer, where accumulated analog errors are largest, the candidate signal $\hat{h}_t$ shows a mean absolute error of approximately 60 pA relative to software predictions. After discrete thresholding at the cell boundary, this error drops to approximately 3 pA on the output state h_t , a 20-fold suppression (Appendix J, Figure 13). The residual error is dominated by transistor subthreshold leakage when cells should output zero current, rather than by systematic signal corruption. This confirms that the candidate-to-state thresholding acts as a structural error-suppressing nonlinearity, preventing noise from accumulating through the recurrent feedback loop. A detailed stage-by-stage comparison is provided in Appendix J.

Finally, to confirm that the approach generalizes beyond binary classification, we evaluate the FQ BMRU on 11-class digit recognition (“zero” through “nine” plus background noise). A 2×16 architecture achieves competitive accuracy and substantially increases output margin separation between classes (Appendix I), improving robustness to mismatch. Such networks remain within sub-microwatt consumption estimates (Appendix E, Table 4). Given the strong hardware-software agreement demonstrated at $d = 4$, we expect similar behavior in hardware at larger dimensions.

4 Noise immunity and power scaling

The end-to-end agreement established in Section 3 demonstrates that the software model is a high-fidelity simulator of the physical circuit. We exploit this fidelity to conduct noise immunity and power scaling analyses on three benchmark tasks from Table 1 (sMNIST, pMNIST, dKWS), sweeping over noise magnitudes and state dimensions that would be prohibitively expensive in transistor-level simulation. All three architectures are retrained using the hardware backbone (Appendix C.2.3).

We first test network robustness to increasing levels of analog noise and mismatch. We extract worst-case noise levels from transistor-level simulations under 3σ mismatch and PVT variations (200 Monte Carlo samples across all building blocks), expressed as a worst-case fraction of nominal signal amplitude across multiple operating points, and inject this noise into the software simulator. Multiple noise levels are tested: $0.5\times$, $1\times$, $2\times$, and beyond the measured analog noise magnitude. For each benchmark and test sample, we generate 10 noisy instantiations of the network for each inference.

Figure 3 presents the results for FQ BMRU, LRU, and minGRU, with noise injected at the same relative magnitude for fairness. At the measured analog noise level, both FQ BMRU and minGRU operate with no significant accuracy decrease. LRU exhibits catastrophic failure even at low noise levels, consistent with its purely linear recurrence offering no mechanism to attenuate accumulated perturbations. As noise increases beyond the measured level, FQ BMRU maintains robustness up to approximately $2\times$ the analog noise before a sharp transition to failure. minGRU shows a

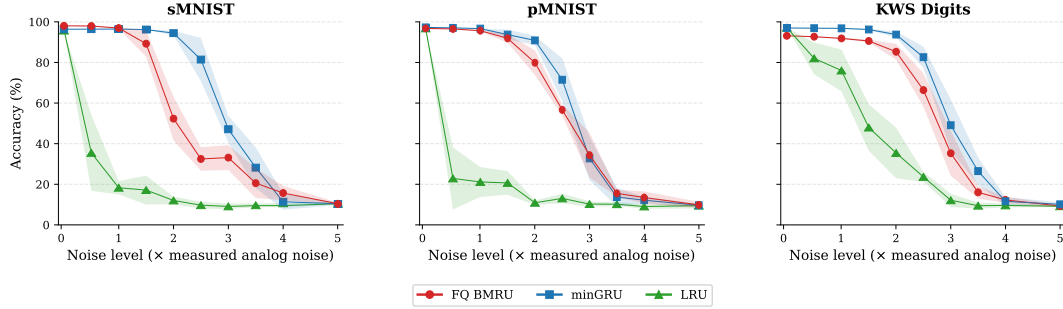


Figure 3: **Large-scale noise robustness analysis across three benchmarks (sMNIST, pMNIST and dKWS).** Accuracy as a function of injected noise level (relative to measured analog noise from transistor-level simulations) for FQ BMRU, LRU, and minGRU. At analog noise level, FQ BMRU and minGRU maintain full accuracy, while LRU fails catastrophically. FQ BMRU exhibits robust performance up to approximately $2\times$ the analog noise level across all benchmarks. Results computed over the full test set with 10 noisy instantiations per sample.

similar pattern with slightly higher tolerance, which we attribute to multiplicative gating acting as an implicit noise regularizer. These results confirm that discrete-output dynamics provide sufficient noise immunity for scalable analog deployment, and that this immunity is a structural property of the architecture rather than an artifact of a specific circuit or task.

We then estimated power scaling based on per-component measurements at $d = 4$ from Cadence Spectre simulation, and extrapolated to higher state dimensions (Appendix E). At $d = 32$, FC layers consume approximately $6\times$ the power of BMRU cells, meaning recurrence adds less than 15% overhead relative to a hypothetical feedforward-only analog network of the same dimension. This power structure has an important implication: analog feedforward inference using current mirrors and KCL is well-established, and the FC layers implementing it dominate total power at practical network sizes. The BMRU cells add recurrence, and with it, the ability to process temporal sequences rather than static inputs, at a marginal power cost that is linear and modest.

5 Perspectives and limitations

Several architectural extensions could enhance capability while preserving the co-design principles.

Increased state dimension. Scaling to larger state dimensions ($d = 8, 16, 32$) improves accuracy (Appendix C, Table 2), with sub-microwatt operation remaining achievable up to $d = 16$ where accuracy saturates (Appendix E, Table 4). Increasing state dimension also substantially increases output margin separation in multi-class settings, improving robustness to mismatch.

Quantized parameter mapping. Programmable parameters can be achieved through binary-weighted current mirror banks [73, 74] enabled through digital registers. This approach, analogous to programmable crossbar arrays [39, 75, 76], enables B -bit resolution with B transistors per parameter. Preliminary experiments indicate that 4-bit quantization introduces negligible accuracy degradation compared to full-precision training, even without quantization-aware pretraining [77–80]. Quantized accuracy on “yes” KWS is reported in Appendix C.3. Estimated power and area overhead for the programmable version, including binary-weighted current mirrors, and shift registers is provided in Appendix K.

Limitations. The 180 nm CMOS simulations, while validated in Cadence with Monte Carlo mismatch and PVT corner analysis (Appendices H and G), await physical fabrication and measurement, and focus on the binary KWS classification task. Multi-class KWS and more complex benchmarks are evaluated in software with realistic noise levels, but remain to be validated on hardware.

6 Conclusion

We presented the first fully analog RNN implementation that combines structural noise immunity with parallelizable training, through hardware-software co-design of the BMRU. By reformulating the BMRU to operate in the first quadrant with fixed thresholds, and by designing from first principles an ultra-low power current-mode bistable circuit whose behavior matches the reformulated cell, we established a one-to-one correspondence between each learned parameter and a programmable circuit element. Transistor-level simulations in 180 nm CMOS show near-perfect agreement between software predictions and circuit-level dynamics at every stage of the network, turning the software model into a high-fidelity simulator of the physical hardware. We leveraged this fidelity to establish, across multiple benchmarks, that discrete-valued outputs provide structural noise immunity, breaking the noise accumulation barrier that has prevented analog recurrence, and that recurrence can be added to analog inference at linear marginal power cost relative to the quadratically-scaling feedforward backbone. End-to-end validation on keyword spotting achieves sub-microwatt power consumption for the RNN core. As always-on edge AI becomes increasingly prevalent, this co-design approach opens a path toward adding temporal intelligence to analog inference platforms at minimal additional cost. Broader impacts and code/data availability are discussed in Appendices L and M.

Acknowledgements

A.F., J.B. and L.M. are, respectively, postdoctoral researcher (grant ASP-REN-40024838) and FRIA grantees (grants FRIA-40038025 and FRIA-B2-40029909) of the Fonds de la Recherche Scientifique – FNRS. This work is supported by the Belgian Government, Federal Public Service Policy and Support, through grant NEMODEI2. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the FNRS under Grant No. 2.5020.11 and by the Walloon Region. The present research also benefited from computational resources made available on Lucia, the Tier-1 supercomputer of the Walloon Region, infrastructure funded by the Walloon Region under grant agreement No. 1910247.

Disclosure

This work has been the subject of patent applications under numbers EP26175243.0 and EP26175248.9.

References

- [1] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. LLMCarbon: Modeling the end-to-end carbon footprint of large language models, 2024. URL <https://arxiv.org/abs/2309.14393>.
- [2] Enrico Barbierato and Alice Gatti. Toward green AI: A methodological survey of the scientific literature. *IEEE Access*, 12:23989–24013, 2024. doi: 10.1109/ACCESS.2024.3360705.
- [3] Mark Horowitz. Computing’s energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 10–14, San Francisco, CA, USA, 2014. IEEE. doi: 10.1109/ISSCC.2014.6757323.
- [4] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.
- [5] Jihun Lee, Vincent Leung, Ah-Hyoung Lee, Jiannan Huang, Peter Asbeck, Patrick P Mercier, Stephen Shellhammer, Lawrence Larson, Farah Laiwalla, and Arto Nurmikko. Neural recording and stimulation using wireless networks of microimplants. *Nature Electronics*, 4(8): 604–614, 2021. doi: 10.1038/s41928-021-00631-8.
- [6] Dongjin Seo, Jose M Carmena, Jan M Rabaey, Elad Alon, and Michel M Maharbiz. Neural dust: An ultrasonic, low power solution for chronic brain-machine interfaces, 2013. URL <https://arxiv.org/abs/1307.2196>.

- [7] Mohammadali Sharifshazileh, Karla Burelo, Johannes Sarnthein, and Giacomo Indiveri. An electronic neuromorphic system for real-time detection of high frequency oscillations (HFO) in intracranial EEG. *Nature Communications*, 12(1):3095, 2021. doi: 10.1038/s41467-021-23342-2.
- [8] Rahul Sarpeshkar. Analog versus digital: Extrapolating from electronics to neurobiology. *Neural Computation*, 10(7):1601–1638, 1998. doi: 10.1162/089976698300017052.
- [9] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature Nanotechnology*, 15(7): 529–544, 2020. doi: 10.1038/s41565-020-0655-z.
- [10] Daniele Ielmini and H-S Philip Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1(6):333–343, 2018. doi: 10.1038/s41928-018-0092-2.
- [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, Long Beach, CA, USA, 2017.
- [12] Alexander M Fichtl, Jeremias Bohn, Josefin Kelber, Edoardo Mosca, and Georg Groh. The end of transformers? On challenging attention and the rise of sub-quadratic architectures, 2025. URL <https://arxiv.org/abs/2510.05364>.
- [13] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022. doi: 10.1145/3530811.
- [14] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL <https://arxiv.org/abs/2111.00396>.
- [15] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL <https://arxiv.org/abs/2312.00752>.
- [16] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality, 2024. URL <https://arxiv.org/abs/2405.21060>.
- [17] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 930–938, Merida, Mexico, 2024. doi: 10.1145/3616855.3635760.
- [18] Junxiong Wang, Daniele Paliotta, Avner May, Alexander Rush, and Tri Dao. The Mamba in the Llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, volume 37, pages 62432–62457, Vancouver, Canada, 2024.
- [19] Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, pages 35971–35983, New Orleans, LA, USA, 2022.
- [20] Stephen Boyd and Leon Chua. Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, 32(11):1150–1161, 1985. doi: 10.1109/TCS.1985.1085649.
- [21] Nicolas Vecoven, Damien Ernst, and Guillaume Drion. A bio-inspired bistable recurrent cell allows for long-lasting memory. *PLOS ONE*, 16(6):e0252676, 2021. doi: 10.1371/journal.pone.0252676.
- [22] Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Damien Ernst, and Guillaume Drion. Warming up recurrent neural networks to maximise reachable multistability greatly improves learning. *Neural Networks*, 166:645–669, 2023. doi: 10.1016/j.neunet.2023.07.023.
- [23] Eve Marder, L F Abbott, Gina G Turrigiano, Zhengyu Liu, and Jorge Golowasch. Memory from the dynamics of intrinsic membrane currents. *Proceedings of the National Academy of Sciences*, 93(24):13481–13486, 1996. doi: 10.1073/pnas.93.24.13481.

- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.
- [25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation, 2014. URL <https://arxiv.org/abs/1406.1078>.
- [26] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.
- [27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, Atlanta, GA, USA, 2013. PMLR.
- [28] Raju Machupalli, Masum Hossain, and Mrinal Mandal. Review of ASIC accelerators for deep neural network. *Microprocessors and Microsystems*, 89:104441, 2022. doi: 10.1016/j.micpro.2022.104441.
- [29] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019. doi: 10.1109/JETCAS.2019.2910232.
- [30] Soroush Heydari and Qusay H Mahmoud. Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions. *Sensors*, 25(10):3191, 2025. doi: 10.3390/s25103191.
- [31] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Jeremy Holleman, Nat Jeffries, Csaba Kiraly, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, et al. MLPerf tiny benchmark, 2021. URL <https://arxiv.org/abs/2106.07597>.
- [32] Pete Warden and Daniel Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O’Reilly Media, 2019.
- [33] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. MCUNet: Tiny deep learning on IoT devices. In *Advances in Neural Information Processing Systems*, volume 33, pages 11711–11722, Virtual, 2020.
- [34] Igor Fedorov, Ryan P Adams, Matthew Mattina, and Paul N Whatmough. SpArSe: Sparse architecture search for CNNs on resource-constrained microcontrollers. In *Advances in Neural Information Processing Systems*, volume 32, Vancouver, Canada, 2019.
- [35] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002. doi: 10.1162/089976602760407955.
- [36] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham China, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.
- [37] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with Loihi 2. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259, Coimbra, Portugal, 2021. IEEE. doi: 10.1109/SiPS52927.2021.00053.
- [38] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990. doi: 10.1109/5.58356.
- [39] Qiangfei Xia and J Joshua Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4):309–323, 2019. doi: 10.1038/s41563-019-0291-x.

- [40] Mirko Prezioso, Farnood Merrikh-Bayat, Brian D Hoskins, Gina C Adam, Konstantin K Likharev, and Dmitri B Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550):61–64, 2015. doi: 10.1038/nature14441.
- [41] Mostafa Rahimi Azghadi, Corey Lammie, Jason K Eshraghian, Melika Payvand, Elisa Donati, Bernabe Linares-Barranco, and Giacomo Indiveri. Hardware implementation of deep network accelerators towards healthcare and biomedical applications. *IEEE Transactions on Biomedical Circuits and Systems*, 14(6):1138–1159, 2020. doi: 10.1109/TBCAS.2020.3036081.
- [42] Can Li, Miao Hu, Yunning Li, Hao Jiang, Ning Ge, Eric Montgomery, Jiaming Zhang, Wenhao Song, Noraica Dávila, Catherine E Graves, et al. Long short-term memory networks in memristor crossbar arrays. *Nature Machine Intelligence*, 1(1):49–57, 2019. doi: 10.1038/s42256-018-0001-4.
- [43] Jiadong Wu, Yinan Wang, Zhiwei Li, Lun Lu, and Qingjiang Li. A review of computing with spiking neural networks. *Computers, Materials & Continua*, 78(3):2909, 2024. doi: 10.32604/cmc.2024.047240.
- [44] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. doi: 10.1038/s41586-019-1677-2.
- [45] Nicholas LeBow, Bodo Rueckauer, Pengfei Sun, Meritxell Rovira, Cecilia Jiménez-Jorquera, Shih-Chii Liu, and Josep Maria Margarit-Taulé. Real-time edge neuromorphic tasting from chemical microsensor arrays. *Frontiers in Neuroscience*, 15:771480, 2021. doi: 10.3389/fnins.2021.771480.
- [46] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Bill Kay, and Prasanna Date. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022. doi: 10.1038/s43588-021-00184-y.
- [47] Andre Xian Ming Chang, Berin Martini, and Eugenio Culurciello. Recurrent neural networks hardware implementation on FPGA, 2015. URL <https://arxiv.org/abs/1511.05552>.
- [48] Nadezhda Semenova and Daniel Brunner. Noise-mitigation strategies in physical feedforward neural networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(6):061106, 2022. doi: 10.1063/5.0096637.
- [49] Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation, 2020. URL <https://arxiv.org/abs/2006.01981>.
- [50] Shih-Chii Liu. *Analog VLSI: Circuits and principles*. MIT press, 2002.
- [51] Aspinity. AML100 - near-zero power analogml processor. <https://www.aspinity.com/aml100>, 2022. Product information page, accessed: 2026-05-05.
- [52] Syntiant. NDP120 neural decision processor. <https://www.syntiant.com/s/Syntiant-Product-Brief-NDP120.pdf>, 2021. Product brief, accessed: 2026-05-05.
- [53] Florent De Geeter, Gaspard Lambrechts, Damien Ernst, and Guillaume Drion. Parallelizable memory recurrent units, 2026. URL <https://arxiv.org/abs/2601.09495>.
- [54] Federico Danieli, Pau Rodríguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. ParaRNN: Unlocking parallel training of nonlinear RNNs for large language models, 2025. URL <https://arxiv.org/abs/2510.21450>.
- [55] Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length, 2017. URL <https://arxiv.org/abs/1709.04057>.
- [56] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018. URL <https://arxiv.org/abs/1804.03209>.

- [57] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences, 2023. URL <https://arxiv.org/abs/2303.06349>.
- [58] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were RNNs all we needed?, 2024. URL <https://arxiv.org/abs/2410.01201>.
- [59] Asad Bakija, Florent De Geeter, Julien Brandoit, Pierre Sacré, and Guillaume Drion. On the importance of multistability for horizon generalization in reinforcement learning, 2026. URL <https://arxiv.org/abs/2605.12206>.
- [60] Marcel J M Pelgrom, Aad C J Duinmaijer, and Anton P G Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid-State Circuits*, 24(5):1433–1439, 1989. doi: 10.1109/JSSC.1989.572629.
- [61] Peter R Kinget. Device mismatch and tradeoffs in the design of analog circuits. *IEEE Journal of Solid-State Circuits*, 40(6):1212–1224, 2005. doi: 10.1109/JSSC.2005.848021.
- [62] Mohtashim Mansoor, Ibraheem Haneef, Suhail Akhtar, Andrea De Luca, and Florin Udrea. Silicon diode temperature sensors—a review of applications. *Sensors and Actuators A: Physical*, 232:63–74, 2015. doi: 10.1016/j.sna.2015.04.022.
- [63] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers, 2021. URL <https://arxiv.org/abs/2011.04006>.
- [64] Andrej Karpathy. nanoGPT, 2022. URL <https://github.com/karpathy/nanoGPT>. GitHub repository.
- [65] Iván López-Espejo, Zheng-Hua Tan, John H L Hansen, and Jesper Jensen. Deep spoken keyword spotting: An overview. *IEEE Access*, 10:4169–4199, 2022. doi: 10.1109/ACCESS.2021.3139508.
- [66] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597.
- [67] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091, Florence, Italy, 2014. IEEE. doi: 10.1109/ICASSP.2014.6854370.
- [68] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting, 2017. URL <https://arxiv.org/abs/1703.05390>.
- [69] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. doi: 10.1109/TASSP.1980.1163420.
- [70] Daniel Augusto Villamizar, Dante Gabriel Muratore, James B Wieser, and Boris Murmann. An 800 nW switched-capacitor feature extraction filterbank for sound classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(4):1578–1588, 2021. doi: 10.1109/tcsi.2020.3047035.
- [71] Heejin Yang, Ji-Hwan Seol, Rohit Rothe, Zichen Fan, Qirui Zhang, Hun-Seok Kim, David Blaauw, and Dennis Sylvester. A 1.5- μ W fully-integrated keyword spotting SoC in 28-nm CMOS with skip-RNN and fast-settling analog frontend for adaptive frame skipping. *IEEE Journal of Solid-State Circuits*, 59(1):29–39, 2023. doi: 10.1109/jssc.2023.3316648.
- [72] Cadence Design Systems. Cadence Virtuoso Platform. https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/virtuoso-studio.html, 2023.

- [73] Phillip E Allen and Douglas R Holberg. *CMOS Analog Circuit Design*. Oxford University Press, 3rd edition, 2011.
- [74] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2001.
- [75] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008. doi: 10.1038/nature06932.
- [76] Fabien Alibart, Elham Zamanidoost, and Dmitri B Strukov. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nature Communications*, 4:2072, 2013. doi: 10.1038/ncomms3072.
- [77] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. CRC Press, 2022. doi: 10.1201/9781003162810-13.
- [78] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, volume 28, pages 3123–3131, Montreal, Canada, 2015.
- [79] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, volume 29, Barcelona, Spain, 2016.
- [80] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets, 2019. URL <https://arxiv.org/abs/1903.05662>.
- [81] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers, 2017. URL <https://arxiv.org/abs/1711.07128>.
- [82] Arnab Neelim Mazumder and Tinoosh Mohsenin. A fast network exploration strategy to profile low energy consumption for keyword spotting, 2022. URL <https://arxiv.org/abs/2202.02361>.
- [83] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. doi: 10.1109/msp.2019.2931595.
- [84] Yingjie Tian, Yuqi Zhang, and Haibin Zhang. Recent advances in stochastic gradient descent in deep learning. *Mathematics*, 11(3):682, 2023. doi: 10.3390/math11030682.
- [85] Hendrawan Soeleman, Kaushik Roy, and Bipul Chandra Paul. Robust subthreshold logic for ultra-low power operation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):90–99, 2002. doi: 10.1109/92.920822.
- [86] Eric Vittoz and Jean Fellrath. CMOS analog integrated circuits based on weak inversion operations. *IEEE Journal of Solid-State Circuits*, 12(3):224–231, 1977. doi: 10.1109/JSSC.1977.1050882.
- [87] Carver Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.
- [88] IM Filanovsky and H Baltes. CMOS Schmitt trigger design. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(1):46–49, 1994. doi: 10.1109/81.260219.
- [89] Montree Siripruchyanun and Jirawat Hirunporm. A fully/electronically controllable voltage-mode Schmitt trigger based on only single VDGA and its applications. *AEU - International Journal of Electronics and Communications*, 131:153602, 2021. doi: 10.1016/j.aeue.2020.153602.
- [90] Wayne H Wolf. Hardware-software co-design of embedded systems. *Proceedings of the IEEE*, 82(7):967–989, 1994. doi: 10.1109/5.293155.

- [91] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.
- [92] Nikita Nangia and Samuel R. Bowman. ListOps: A diagnostic dataset for latent tree learning, 2018. URL <https://arxiv.org/abs/1804.06028>.
- [93] Brian McFee, Colin Raffel, Dawen Liang, Daniel P W Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, pages 18–24, Austin, TX, USA, 2015. doi: 10.25080/Majora-7b98e3ed-003.
- [94] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>, 2015.
- [95] Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding, 2021. URL <https://arxiv.org/abs/2107.02561>.
- [96] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization, 2016. URL <https://arxiv.org/abs/1607.06450>.
- [97] Guy E Blelloch. Scans as primitive parallel operations. *IEEE Transactions on computers*, 38(11):1526–1538, 2002. doi: 10.1109/12.42122.
- [98] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks, 2017. URL <https://arxiv.org/abs/1612.08083>.
- [99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, NV, USA, 2016. doi: 10.1109/CVPR.2016.90.
- [100] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
- [101] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL <https://arxiv.org/abs/1711.05101>.
- [102] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts, 2017. URL <https://arxiv.org/abs/1608.03983>.
- [103] Julien Brandoit, Arthur Fyon, Damien Ernst, and Guillaume Drion. Improving the performance and learning stability of parallelizable rnns designed for ultra-low power applications, 2026. URL <https://arxiv.org/abs/2605.11855>.
- [104] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs. <http://github.com/google/jax>, 2018. Version 0.3.13.
- [105] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX. <http://github.com/google/flax>, 2024. Version 0.12.2.
- [106] Robert M Gray and David L Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. doi: 10.1109/18.720541.
- [107] Minhao Yang, Shih-Chii Liu, Chenghan Li, and Tobi Delbruck. Addressable current reference array with 170dB dynamic range. In *Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3110–3113. IEEE, 2012.
- [108] Hugh Greatorex, Ole Richter, Michele Mastella, Madison Cotteret, Philipp Klein, Maxime Fabre, Arianna Rubino, Willian Soares Girão, Junren Chen, Martin Ziegler, et al. A neuromorphic processor with on-chip learning for beyond-CMOS device integration. *Nature communications*, 16(1):6424, 2025. doi: 10.1038/s41467-025-61576-6.

A Related work

This appendix provides a comprehensive overview of related work across the domains bridged by our contribution: energy-efficient sequence models, neuromorphic and analog computing, and hardware-software co-design for neural networks.

A.1 Sequence models for edge deployment

Transformers and attention mechanisms. Transformer architectures have achieved state-of-the-art performance across sequence modeling domains [11], but their quadratic complexity in sequence length and large memory footprint make them unsuitable for resource-constrained edge devices [12, 13]. Various efficiency improvements have been proposed, including sparse attention patterns and linear attention variants, but these typically remain energy inefficient.

State-space models. SSMs offer an efficient alternative through linear recurrence, enabling parallelizable training [14–16, 57, 17–19]. However, their transient dynamics—states decaying exponentially toward equilibrium—prevent multistability [20], limiting performance on tasks requiring persistent memory [21, 22].

Recurrent neural networks. Traditional RNNs provide multistability essential for long-term memory [23] through nonlinear recurrent dynamics. LSTMs [24] and GRUs [25] have dominated sequence modeling for decades, though their complex gating mechanisms and inherently sequential nature create both training bottlenecks [26, 27] and hardware implementation challenges. Recent work on parallelizable RNN training [55, 54] addresses the training bottleneck, and the Memory Recurrent Unit (MRU) family [53] combines instantaneous convergence with multistability, enabling parallel training via associative scans while maintaining persistent memory.

A.2 TinyML and digital edge AI

Application-specific integrated circuits. ASICs achieve efficiency gains through parallelized operations and quantization [28, 29], but remain fundamentally digital with separated memory and computation [35].

TinyML on microcontrollers. The TinyML movement has extended neural network deployment to microcontrollers, typically enabling milliwatt-level inference [30–34]. KWS has served as a primary benchmark for these systems [56, 65–68, 81, 82]. While significant progress has been made in model compression and efficient architectures [33], power consumption remains orders of magnitude above what analog approaches can achieve.

Quantization techniques. Quantization-aware training and post-training quantization enable deployment on resource-constrained hardware [77–80]. Our work demonstrates that the discrete-output nature of BMRUs provides inherent robustness to quantization, achieving minimal accuracy degradation with 4-bit parameter precision.

A.3 Neuromorphic computing

Spiking neural networks. Digital neuromorphic chips like Intel Loihi 2 implement spiking neural networks (SNNs) with event-driven efficiency [36–38]. SNNs offer biological plausibility and potential energy benefits through sparse, event-driven computation. However, SNN performance still lags behind conventional RNNs on many sequence tasks [43–46], and training remains challenging due to non-differentiable spike functions. Surrogate gradient methods [83, 84] have enabled gradient-based SNN training, techniques we also employ for training BMRUs with Heaviside nonlinearities.

Analog feedforward inference. Analog crossbar arrays achieve in-memory computation through KCL [39–42]. The Aspinity AML100 [51] performs always-on event detection entirely in the analog domain at $\sim 30\text{--}100\ \mu\text{W}$ using configurable analog blocks, representing the state of the art in programmable analog ML. However, all fully existing analog implementations—including crossbar arrays, the AML100, and flash-based approaches—are restricted to feedforward architectures.

Extending fully analog computation to recurrent dynamics has been considered impractical due to noise accumulation through temporal feedback [48, 49]. Our work demonstrates that discrete-output BMRU dynamics overcome this barrier, enabling the first analog RNN.

FPGA implementations. Digital FPGA implementations of recurrent architectures exist [47], offering flexibility and moderate efficiency gains over general-purpose processors, but lack the power efficiency of subthreshold analog circuits.

A.4 Analog circuit design for neural networks

Subthreshold analog computing. Subthreshold operation exploits the exponential current-voltage relationship of MOSFETs below threshold for ultra-low power signal processing [85–87]. This regime enables computation at picoampere current levels—six orders of magnitude below typical digital switching currents. Our implementation builds on these principles, operating entirely in subthreshold with nanowatt power consumption.

Schmitt triggers. The Schmitt trigger implements bistable thresholding with hysteresis, providing noise immunity through its characteristic hysteresis window [88]. Traditional voltage-mode implementations operate with milliwatt power consumption [89], unsuitable for ultra-low power applications. Our work leverages a current-mode Schmitt trigger design that achieves fully tunable thresholds and output amplitudes while operating in the nanowatt regime.

Current-mode circuits. Current-mode signal processing offers advantages for analog neural networks, including natural summation via KCL and compatibility with subthreshold operation. Current mirrors with controlled width ratios implement weighted connections [73, 74], with precision determined by layout matching [60, 61], typically achieving 6–8 bit equivalent resolution in standard CMOS processes.

A.5 Hardware-software co-design

Co-design principles. Hardware-software co-design seeks to jointly optimize algorithms and their physical implementations [90]. Rather than adapting algorithms to existing hardware or designing hardware to implement existing algorithms, co-design identifies synergies where algorithmic structure aligns with hardware capabilities. Our work exemplifies this approach: the BMRU discrete outputs and hysteretic dynamics map directly to Schmitt trigger behavior, enabling efficiency gains unattainable by optimizing either domain independently.

Neural network accelerators. Prior co-design efforts have focused primarily on digital accelerators [29, 28], optimizing dataflow and memory hierarchy for specific network architectures. Analog co-design remains less explored, particularly for recurrent networks where temporal feedback complicates noise management.

A.6 Energy consumption in AI systems

The computational demands of modern AI have grown exponentially, with AI systems now consuming significant portions of global energy resources [1, 2]. Energy efficiency in computing has been extensively studied [3, 4], with particular attention to the memory wall and data movement costs inherent to Von Neumann architectures [8–10]. Always-on applications—environmental sensor networks, biomedical monitoring devices, and edge devices processing sequential data [5–7]—face particularly acute power constraints that motivate our ultra-low power approach.

B Expressivity proofs

This appendix establishes that the simplifications introduced in the FQ BMRU do not reduce expressivity compared to the original BMRU in an approximate sense. We prove two results: (1) restricting outputs to $\{0, \alpha\}$ instead of $\{-\alpha, +\alpha\}$ preserves expressivity exactly (Proposition B.3), and (2) replacing input-dependent thresholds with fixed thresholds, when combined with a preceding ReLU

MLP, can approximate the original BMRU arbitrarily well (Proposition B.4). Together they show that the FQ BMRU is a universal approximator of the original BMRU function class.

Definition B.1 (Original BMRU). The original BMRU cell computes:

$$\begin{aligned}\hat{h}_t &= W_x x_t + b_x, \\ \beta_t &= |W_\beta x_t + b_\beta|, \\ z_t &= \mathcal{H}(|\hat{h}_t| - \beta_t), \\ h_t &= z_t \odot S(\hat{h}_t) \odot \alpha + (1 - z_t) \odot h_{t-1},\end{aligned}$$

where $x_t \in \mathbb{R}^n$, $W_x, W_\beta \in \mathbb{R}^{d \times n}$, $b_x, b_\beta, \alpha \in \mathbb{R}^d$, $\mathcal{H}(\cdot)$ is the element-wise Heaviside step function, and $\odot$ denotes Hadamard product.

Definition B.2 (FQ BMRU). The FQ BMRU cell computes:

$$\begin{aligned}\hat{h}_t &= \text{ReLU}(\tilde{W}_x x_t + \tilde{b}_x), \\ z_{\text{lo},t} &= \mathcal{H}(\beta_{\text{lo}} - \hat{h}_t), \\ z_{\text{hi},t} &= \mathcal{H}(\hat{h}_t - \beta_{\text{hi}}), \\ h_t &= z_{\text{hi},t} \odot \alpha + (1 - z_{\text{lo},t}) \odot (1 - z_{\text{hi},t}) \odot h_{t-1},\end{aligned}$$

where $x_t \in \mathbb{R}^n$, $\tilde{W}_x \in \mathbb{R}^{d \times n}$, $\tilde{b}_x, \beta_{\text{lo}}, \beta_{\text{hi}}, \alpha \in \mathbb{R}^d$, and $\beta_{\text{hi}} > \beta_{\text{lo}}$ element-wise.

B.1 Output range equivalence

Proposition B.3. Let f be a function computed by a BMRU cell producing bipolar outputs $h^{(\pm)} \in \{-\alpha, +\alpha\}^d$, followed by a linear layer $y = Wh^{(\pm)} + b$ with $W \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. Then the same function is computed by a BMRU cell with unipolar outputs $h^{(+)} \in \{0, \alpha\}^d$ followed by $y = \tilde{W}h^{(+)} + \tilde{b}$, where $\tilde{W} = 2W$ and $\tilde{b} = b - W\alpha$.

Proof. Define the mapping $h_i^{(+)} = \frac{1}{2}(h_i^{(\pm)} + \alpha_i)$ for each dimension $i \in \{1, \dots, d\}$. This is a bijection between $\{-\alpha_i, +\alpha_i\}$ and $\{0, \alpha_i\}$:

- $h_i^{(\pm)} = +\alpha_i \implies h_i^{(+)} = \frac{1}{2}(\alpha_i + \alpha_i) = \alpha_i,$
- $h_i^{(\pm)} = -\alpha_i \implies h_i^{(+)} = \frac{1}{2}(-\alpha_i + \alpha_i) = 0.$

The inverse mapping is $h^{(\pm)} = 2h^{(+)} - \alpha$. Substituting into the linear layer:

$$\begin{aligned}y &= Wh^{(\pm)} + b = W(2h^{(+)} - \alpha) + b \\ &= 2Wh^{(+)} + (b - W\alpha).\end{aligned}$$

Thus, setting $\tilde{W} = 2W$ and $\tilde{b} = b - W\alpha$ yields $y = \tilde{W}h^{(+)} + \tilde{b}$. Since this parameter transformation is bijective, the two representations span identical function spaces. $\square$

B.2 Fixed threshold approximation

Proposition B.4 (Fixed threshold approximation). Let $\mathcal{X} \subset \mathbb{R}^n$ be a compact set. For any original BMRU cell and any $\epsilon > 0$, there exists a ReLU MLP $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$ and fixed thresholds $\beta_{\text{lo}}, \beta_{\text{hi}} \in \mathbb{R}^d$ with $\beta_{\text{hi}} > \beta_{\text{lo}}$ such that the FQ BMRU cell, with $\hat{h}_t = \Phi(x_t)$ (absorbing the cell's own ReLU layer into the MLP), reproduces the gating behaviour of the original cell on $\mathcal{X}$ except on a set of measure at most ϵ . The resulting state update can therefore be approximated arbitrarily closely.

Proof. The original BMRU gate for dimension i compares $\hat{h}_i(x) = [W_x]_i x + [b_x]_i$ against an input-dependent threshold $\beta_i(x) = |[W_\beta]_i x + [b_\beta]_i|$. Because $\beta_i(x)$ is piecewise-linear, an MLP before the cell can directly absorb this computation: a ReLU MLP can represent any continuous

piecewise-linear function on a compact set, so it can learn to produce features that implicitly encode the needed decision functions.

A second, geometric difference is that the original hold region $\{x : |\hat{h}_i(x)| < \beta_i(x)\}$ is symmetric around zero after thresholding, while the FQ BMRU hold region $\{x : \beta_{lo} < \hat{h}_i < \beta_{hi}\}$ is not necessarily centered. However, this asymmetry can be compensated by a simple affine transformation inside the MLP: let $\mu_i = (\beta_{hi} + \beta_{lo})/2$ and $\sigma_i = (\beta_{hi} - \beta_{lo})/2$. Then shifting the MLP output by μ_i and scaling by $1/\sigma_i$ maps the FQ hold region to a symmetric interval $(-1, 1)$. In other words, the preceding MLP can learn to offset the input so that the effective hysteretic window becomes centered, with half-width σ_i playing exactly the role of the original β_i .

Consequently, an MLP can first project the input x into a space where the required piecewise-linear boundaries are encoded, and then shift and scale the projections so that fixed thresholds β_{lo}, β_{hi} separate the “set high”, “hold” and “set low” regions in the same way as the original boundaries, up to an arbitrarily narrow gap where the original touching regions would meet. Because ReLU MLPs can approximate the signed distance to these boundaries, the measure of the disagreement region can be made smaller than any given ϵ , and the resulting gate outputs (and thus the hidden state update) approximate the original BMRU uniformly on $\mathcal{X}$ except on that small set. $\square$

B.3 Combined result

Theorem B.5 (FQ BMRU Universal Approximation). *The FQ BMRU with fixed thresholds and unipolar outputs $\{0, \alpha\}$, when preceded by a ReLU MLP and followed by linear projection layers, can approximate any network based on the original BMRU (with input-dependent thresholds and bipolar outputs) arbitrarily closely on compact subsets of the input space.*

Proof. Given an original BMRU network, first apply Proposition B.4 to replace each original cell with a fixed-threshold FQ cell plus a preceding ReLU MLP; the approximation error in the gating can be made arbitrarily small. Next, apply Proposition B.3 to convert the bipolar outputs of the original network to unipolar outputs via an affine reparameterization of the subsequent linear layers; this conversion is exact and bijective. Composing the two steps yields an FQ BMRU network that approximates the original network’s function uniformly on compacta with arbitrarily small error. $\square$

C Training and software inference details

This appendix details the implementation for training and evaluating the BMRU and FQ BMRU in software, including benchmark task descriptions, dataset characteristics, and evaluation protocols.

C.1 Benchmark task descriptions

C.1.1 Sequential MNIST (sMNIST)

Sequential MNIST presents handwritten digit images (28×28 pixels) as sequences of 784 grayscale pixel values in raster scan order (left-to-right, top-to-bottom). The task requires classifying the digit (0–9) based solely on the sequential pixel stream.

Task structure. Each input sequence $\{x_t\}_{t=1}^{784}$ consists of normalized pixel intensities $x_t \in [0, 1]$. The model processes the sequence and produces a classification decision at the final timestep. Ground truth labels $y \in \{0, 1, \dots, 9\}$ provide supervision at all steps.

Dataset. We use the standard MNIST dataset [91], comprising 70,000 images partitioned into approximately 80%/10%/10% training/validation/test splits.

Evaluation metric. Classification accuracy on the test set. Random-guess baseline is 10%.

C.1.2 Permuted MNIST (pMNIST)

Permuted MNIST applies a fixed random permutation to the pixel ordering of MNIST images, destroying spatial correlations while preserving information content.

Task structure. Given a fixed permutation $\pi : \{1, \dots, 784\} \rightarrow \{1, \dots, 784\}$, each image is presented as the sequence $\{x_{\pi(t)}\}_{t=1}^{784}$. The permutation is identical across all samples and remains fixed during training and evaluation.

Dataset and evaluation. Identical to sMNIST.

C.1.3 ListOps

ListOps evaluates compositional reasoning over hierarchically structured sequences of operations [63].

Task structure. Sequences represent nested operations (MAX, MIN, MEDIAN, SUM_MOD) applied to integer arguments. Sequences are linearized in prefix notation with special tokens marking brackets and operations. Each token is mapped to a learned embedding, and the model processes sequences with mean pooling to produce the final numerical result.

Dataset. We use the standard ListOps dataset [92] with sequences up to length 2,000. The training set consists of 96,000 samples, while both the validation and test sets contain 2,000 samples each.

Evaluation metric. Classification accuracy on the discrete output values (10 possible output classes).

C.1.4 Keyword spotting: digits (dKWS)

This task evaluates spoken digit recognition using the Google Speech Commands dataset (v0.02) [56], containing 105,829 one-second audio clips of 35 spoken words recorded at 16 kHz.

Task structure. 11-way classification distinguishing spoken digits (“zero” through “nine”) and background noise extracted from ambient recordings in the dataset.

Feature extraction. Raw audio is processed using librosa [93] to extract 13-dimensional Mel-frequency cepstral coefficients (MFCCs) with the following parameters: sample rate 16 kHz, FFT window size 512 samples, hop length 160 samples (10 ms), yielding 101 frames per one-second clip. Features are normalized per coefficient to zero mean and unit variance. MFCC extraction is assumed to occur off-chip; only the extracted features are processed by the analog network.

Dataset. We use the official dataset partitioning, yielding approximately 80%/10%/10% training/validation/test splits with no speaker overlap between partitions. Negative examples (background noise) are sampled to match positive examples, ensuring balanced training.

Evaluation metric. Classification accuracy on the test set. Random-guess baseline is 9.09%.

C.1.5 Shakespeare Language Modeling

Character-level language modeling on the complete works of William Shakespeare evaluates a model ability to capture long-range linguistic dependencies in English text.

Task structure. Given a sequence of characters $\{c_1, c_2, \dots, c_T\}$ drawn from the text, the model receives the embedding of c_t at timestep t and must predict a probability distribution over the next character c_{t+1} . The vocabulary consists of $V = 65$ distinct characters (lowercase and uppercase letters, punctuation, and whitespace). The loss is the mean cross-entropy across all prediction steps.

Dataset. We use the complete works of Shakespeare, a standard benchmark for character-level language modeling [94]. The corpus contains approximately 5 million characters, which we split by plays into training (80%), validation (10%), and test (10%) sets, preserving document boundaries.

Table 2: Binary “yes” KWS accuracy (Google Speech Commands dataset). Results averaged over 5 training seeds; brackets show [min; max] across seeds. Each seed is evaluated on 100 balanced test sets (397 “yes” samples paired with 397 random negatives), with accuracy min-max ranges capturing variability across these pairings.

State dimension	4	8	16	32	64	128	256
Accuracy (%)	93.86 [91.47; 94.98]	96.98 [96.00; 98.19]	98.08 [97.85; 98.44]	98.16 [97.71; 98.69]	97.51 [96.97; 97.93]	97.65 [97.48; 97.81]	97.78 [97.48; 98.10]

Evaluation metric. Mean cross-entropy loss on the test set. For comparability with prior work, bits-per-character (BPC) can be obtained as $\text{BPC} = \mathcal{L} / \ln 2$, where $\mathcal{L}$ is the average cross-entropy. A uniform random predictor over the 65-character vocabulary yields $\text{BPC} = \log_2(65) \approx 6.02$ (or equivalently a cross-entropy loss of $\ln 65 \approx 4.17$).

C.1.6 Keyword spotting: yes versus others (“yes” KWS)

This binary classification task represents a wake-word detection scenario, distinguishing a target keyword from confusable alternatives and background noise.

Task structure. Binary classification detecting the word “yes” against negative words: “no”, “up”, “down”, “left”, “right”, and background noise.

Feature extraction and dataset. Identical to dKWS. Negative examples are sampled equally from each negative category to match the number of positive examples.

Evaluation metric. Classification accuracy on the test set, computed over 100 random pairings of all 397 positive samples with 397 randomly selected negative samples across 5 random seeds, reporting mean and min-max range. Random-guess baseline is 50 %. Table 2 presents classification accuracy as a function of state dimension d for the FQ BMRU trained in software.

C.2 Training procedure

We use standard training practices throughout; no architectural modifications or hyperparameter tuning specific to the first-quadrant formulation are required.

C.2.1 Additional recurrent cell implementations

This section provides detailed specifications of the recurrent cells evaluated in our experiments. For the BMRU and FQ BMRU architectures, we refer to Equations (1)–(9) in the main text.

Linear Recurrent Unit (LRU). The LRU [57] implements a diagonal complex-valued state-space model with strictly decaying exponentially parameterized eigenvalues for numerical stability. The state update follows:

$$x_t = \Lambda \odot x_{t-1} + Bu_t, \quad (10)$$

$$y_t = \text{Re}(Cx_t) + Du_t, \quad (11)$$

where $x_t \in \mathbb{C}^d$ is the complex-valued hidden state, $u_t \in \mathbb{R}^m$ is the input, $y_t \in \mathbb{R}^d$ is the output, and $\Lambda \in \mathbb{C}^d$ is a diagonal matrix of eigenvalues. The eigenvalues are parameterized as:

$$\Lambda = \exp(-\exp(\nu) + i \exp(\theta)), \quad (12)$$

where $\nu, \theta \in \mathbb{R}^d$ are learned parameters initialized such that eigenvalue magnitudes lie in $[r_{\min}, r_{\max}]$ with uniformly distributed phases in $[0, 2\pi]$. The input matrix $B \in \mathbb{C}^{d \times m}$ is normalized by $\gamma = \sqrt{1 - |\Lambda|^2}$ to maintain consistent input scaling across different eigenvalue magnitudes. The output matrices $C \in \mathbb{C}^{d \times d}$ and $D \in \mathbb{R}^{d \times m}$ are learned parameters.

The LRU is fully parallelizable via associative scan over the linear recurrence.

Minimal Gated Recurrent Unit (minGRU). The minGRU [58] simplifies the traditional GRU by removing hidden state dependencies from the gating mechanism, enabling full parallelization while maintaining competitive performance. The update rule is:

$$z_t = \sigma(W_z x_t + b_z), \quad (13)$$

$$\tilde{h}_t = W_h x_t + b_h, \quad (14)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \quad (15)$$

where $h_t \in \mathbb{R}^d$ is the hidden state, $x_t \in \mathbb{R}^m$ is the input, $z_t \in [0, 1]^d$ is the update gate, $\tilde{h}_t \in \mathbb{R}^d$ is the candidate hidden state, and $\sigma(\cdot)$ denotes the sigmoid function.

A critical architectural property is that the sigmoid gate ensures $z_t \in (0, 1)$ (open interval), meaning the values $z_t = 0$ and $z_t = 1$ are only approached asymptotically but never attained.

The update equation is a linear recurrence in h_t and can be parallelized via associative scan.

C.2.2 Backbone architecture for software inference (Table 1)

The benchmark experiments in Table 1 use the following parallelizable RNN backbone. Only the recurrent cell varies across rows of Table 1; the encoder, blocks, decoder, and training procedure are shared. The model dimension is fixed at $m = 256$. Classification rows use $r = 2$ stacked recurrent blocks with state dimension $d = 64$; the Shakespeare row uses depth $r = 6$ and $d = m = 256$.

Input encoder. Raw task-specific inputs $\hat{x}_t \in \mathbb{R}^{d_{\text{task}}}$ are first projected to the model dimension and then refined by a residual MLP:

$$x_t = \tilde{x}_t + \text{MLP}(\tilde{x}_t), \quad \tilde{x}_t = \text{Linear}(\hat{x}_t) \in \mathbb{R}^m. \quad (16)$$

Positional encoding. For all tasks of Table 1, a 32-dimensional sinusoidal Positional Encoding (PE) [95] is concatenated to x_t at each timestep and projected back to $\mathbb{R}^m$ through an additional $\text{Linear}(\cdot)$.

Block structure. The backbone consists of r stacked blocks. Each block alternates a recurrent sub-layer and a point-wise MLP sub-layer, both wrapped in a pre-norm residual scheme:

$$y = v \odot x + \text{SubLayer}(\text{Norm}(x)), \quad (17)$$

where $v \in \mathbb{R}^m$ is a learnable scaling vector initialized to one, $\text{Norm}(\cdot)$ denotes LayerNorm [96], and $\text{SubLayer}(\cdot)$ is either the recurrent sub-layer or the MLP sub-layer.

Recurrent sub-layer. The recurrent sub-layer maps the input sequence to a hidden state sequence $h_{1:T} = \text{Cell}(x_{1:T}; h_0)$ with $h_t \in \mathbb{R}^d$, then projects it back to $\mathbb{R}^m$ through an input-gated normalized projection:

$$\widetilde{\text{Cell}}(x_{1:T}) = \text{Norm}(\text{Linear}(\text{Cell}(x_{1:T}; h_0))) \odot \sigma(\text{Linear}(x_{1:T})), \quad (18)$$

where $\sigma(\cdot)$ is the element-wise sigmoid. The inner normalization keeps the magnitude of the recurrent output bounded even for cells with unitary eigenvalues, and the input-dependent gate decouples information storage from information consumption. The recurrent cell varies across rows of Table 1: BMRU and FQ BMRU follow Equations (1)–(9); LRU follows Equations (10)–(11) with diagonal complex states, exponentially parameterized eigenvalues initialized in $[r_{\min}, r_{\max}] = [0.9, 0.999]$ with phases in $[0, 2\pi]$, and input matrix normalized by $\gamma = \sqrt{1 - |\Lambda|^2}$; minGRU follows Equations (13)–(15) with a single input-dependent sigmoid gate and no reset gate. All four cells admit parallel evaluation through associative scans [55, 97].

MLP sub-layer. The MLP sub-layer is point-wise, with input and output dimensions m , hidden dimension $4m$, and GLU activation [98]:

$$\text{MLP}(x) = \text{Linear}(\text{Dropout}(\text{GLU}(\text{Linear}(x)))) . \quad (19)$$

Sequence pooling. For classification tasks, the cross-entropy loss is averaged over all timesteps during training, and predictions at inference use majority voting across timesteps (except for ListOps). Shakespeare uses per-token cross-entropy with no pooling.

Output decoder. The pooled representation $y_{\text{pool}} \in \mathbb{R}^m$ is transformed back to the task output dimension by a residual MLP symmetric to the input encoder:

$$\hat{y} = \tilde{y} + \text{MLP}(\tilde{y}), \quad \tilde{y} = \text{Linear}(y_{\text{pool}}). \quad (20)$$

C.2.3 Backbone architecture for hardware inference (Figures 2 and 3)

The proof-of-concept network targeted by transistor-level simulation uses a deliberately simplified backbone, illustrated in Figure 2A. Components incompatible with current-mode subthreshold CMOS (LayerNorm, GLU, sigmoid gates, residual encoders) are removed, and every remaining operation maps directly to a circuit primitive available in the analog library: current mirrors, diode-connected transistors, and the FQ BMRU cell of Section 2.2. The configuration used in Sections 3 and 4 (as well as all subsequent appendices) is $N = 2$ stacked FQ BMRU layers with state dimension $d = 4$.

Input encoder. Raw 13-dimensional MFCC features $\hat{x}_t$ are projected to the state dimension by a single linear layer:

$$x_t = \text{Linear}(\hat{x}_t) \in \mathbb{R}^d. \quad (21)$$

No residual MLP and no normalization are used. The projection is realized in hardware by the current-mirror FC layer of Appendix D.2.

Recurrent layers. The backbone contains $N = 2$ stacked FQ BMRU layers separated by an inter-layer FC transformation followed by ReLU. Each FQ BMRU cell follows Equations (6)–(9), with parameters α , β_{lo} , and β_{hi} realized as bias currents (Appendix D.4). Skip connections [99] bypass each FQ BMRU layer and are realized in the current domain by routing positive and negative branches separately (Appendix D.3).

Activation domain. All signals throughout the backbone are non-negative currents. The FQ reformulation of Section 2.3 ensures unipolar BMRU outputs; FC layers use ReLU activation realized as a diode-connected PMOS (Appendix D.2); skip connections preserve the unipolar constraint by separating positive and negative branches.

Positional encoding. No positional encoding is used. Keyword detection on a fixed 1-second window is position-invariant, and adding sinusoidal PE would require time-dependent reference currents incompatible with the static-bias design philosophy and is not necessary.

Output decoder. The final FQ BMRU layer feeds an output FC layer producing class currents (two for binary “yes” classification, eleven for the multi-class extension of Appendix I). Classification is read out directly from the output currents.

Sequence pooling. Classification on a 1-second clip uses majority voting across the 101 timesteps. At each timestep the highest output current votes for its class, and the class with the most votes over the sequence is the prediction. This avoids any temporal aggregation circuitry that would itself need to be implemented in analog.

C.2.4 Recurrent cell configuration

Each (FQ) BMRU cell uses the following configuration:

- Dropout rate of 0.1 during training [100];
- All parameters initialized using a default uniform distribution, constrained to positive values for α , β_{lo} , and $\delta (= \beta_{\text{hi}} - \beta_{\text{lo}})$;
- Initial state sampled uniformly in $[0, 1]$ and binarized via surrogate Heaviside during training; zero initialization at inference for the FQ BMRU in hardware.

C.2.5 Training generalities

All experiments follow a unified training procedure with hyperparameters that are not optimized per task. The objective is to observe the robustness of the cells to standard hyperparameter values, while acknowledging that task-specific optimization could improve absolute performance metrics.

Optimizer and schedule. Optimization is performed with AdamW [101] using the following configuration:

- Initial learning rate: 10^{-3} ;
- Weight decay: 10^{-4} ;
- Cosine decay schedule [102] with linear warm-up over 1 % of training steps;
- Gradient clipping: global norm capped at 1.0.

Training budget. A batch size of 64 is used for all experiments. The maximum number of training iterations varies per task to ensure convergence within reasonable computational limits: 70,000 for MNIST tasks, 100,000 for the language modeling and ListOps tasks, and 35,000 for KWS tasks. Models are evaluated every 64 iterations on 20 batches sampled from the validation set, with the best-performing checkpoint saved for final evaluation. All experiments are repeated across 5 random seeds (1, 2, 3, 4, 5).

Loss function. Cross-entropy loss is computed over all timestep predictions for all tasks:

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^T \sum_c y_c \log(\hat{y}_{t,c}), \quad (22)$$

where y_c is the one-hot target (target depends on t for Shakespeare), $\hat{y}_{t,c}$ is the softmax output at timestep t for class c , and T denotes the sequence length.

Evaluation protocol. For sMNIST, pMNIST, ListOps and digit KWS tasks, test accuracy is reported as mean across 5 training seeds. For the Shakespeare task, mean cross-entropy loss is reported. For the “yes” KWS task, accuracy is computed over 100 random pairings of all 397 positive samples with 397 randomly selected negative samples across 5 seeds, reporting mean and min-max ranges to account for variability in negative sample selection.

C.2.6 Gradient estimation

Surrogate gradients. The Heaviside nonlinearity in the (FQ) BMRU cell is non-differentiable. We employ surrogate gradients [83] with the approximation:

$$\frac{d\mathcal{H}}{dx} \stackrel{\text{backward}}{\approx} \frac{1}{1 + (\pi x)^2}. \quad (23)$$

Decaying cumulative update for improved gradient flow. For the (FQ) BMRU, during training the state update equation is augmented as:

$$h_t = f_\theta(x_t, h_{t-1}) + \varepsilon \odot h_{t-1}, \quad (24)$$

where $f_\theta(x_t, h_{t-1})$ corresponds to Equation (4) for BMRU and Equation (9) for FQ BMRU. Training with $\varepsilon = 1$ provides superior gradient flow and convergence stability [103], while $\varepsilon = 0$ recovers the original cell dynamics that map directly onto Schmitt trigger circuits for ultra-low power deployment. We adopt an intermediate approach: training initially with $\varepsilon = 1$, then annealing ε toward zero to produce models compatible with the target hardware. Specifically, ε remains at 1 for the first 5% of training, decays linearly to 0 over the next 70%, then stays at 0 for the final 25%. Best-loss checkpointing is performed only after ε reaches zero, ensuring that inference uses the original cell dynamics. This annealing procedure is specific to BMRU and FQ BMRU; the LRU and minGRU baselines are trained with their standard formulations and do not require it.

C.2.7 Implementation

Training is implemented in JAX [104]/Flax [105] using parallel associative scans for efficient (FQ) BMRU state computation across the sequence dimension.

C.3 Post-training quantization

To assess hardware deployment feasibility, we evaluate post-training quantization without retraining. All network parameters (weights, biases, thresholds β_{lo} , widths δ , and amplitudes α) are uniformly quantized to n bits by mapping to 2^n discrete levels within each parameter dynamic range [106]:

$$w_q = \text{round} \left(\frac{w - w_{\min}}{w_{\max} - w_{\min}} \cdot (2^n - 1) \right) \cdot \frac{w_{\max} - w_{\min}}{2^n - 1} + w_{\min} \quad (25)$$

Table 3 summarizes accuracy under different quantization levels across five training seeds. Four-bit quantization incurs minimal degradation (typically $<4\%$), supporting the programmable hardware approach described in Section 5 of the main paper. Two-bit quantization causes significant accuracy loss, indicating that at least 4 bits are required for this application. Interestingly, 6-bit and 8-bit quantization occasionally yield slight accuracy improvements over full precision, likely due to implicit regularization effects.

Table 3: Post-training quantization results. Accuracy (%) on the “yes” KWS task for various state dimensions and bit widths, averaged over 5 training seeds. FP32 stands for 32-bit Floating Point.

State dim d	2-bit	4-bit	6-bit	8-bit	FP32
4	51.20	89.31	92.75	93.16	93.86
8	67.56	93.29	96.65	96.67	96.98
16	61.04	97.48	97.75	97.60	98.08
32	63.77	98.25	98.23	98.25	98.16
64	59.69	97.17	97.27	97.38	97.51

The consistent robustness to 4-bit quantization across all state dimensions validates the co-design approach: networks can be trained in full precision and subsequently quantized for hardware implementation using binary-weighted current mirrors with 4-bit programmability. Further accuracy gains could be achieved through fine-tuning the quantized network or by adopting quantization-aware training techniques.

C.4 Compute resources

All training experiments were performed on an academic HPC cluster, using a single NVIDIA Tesla A100 GPU (40 GB VRAM) per run on a node with AMD EPYC Milan CPUs and up to 512 GB of system RAM. Training a single seed takes approximately a few hours for all tasks; software inference completes in minutes. The total compute footprint of the project, including preliminary experiments not reported in this paper, remains modest by modern deep learning standards.

D Circuit schematics

This supplementary section provides detailed circuit schematics for the CMOS implementation of the FQ BMRU described in the main paper as well as for the FC layers.

Technology and operating point. All circuits are implemented in a 180 nm CMOS X-FAB technology in Cadence Virtuoso with a supply voltage of 1.8 V, operating in the subthreshold regime. All simulations are conducted using Spectre with PRIMLIB 180 nm models (ne/pe devices). A value of “1” in the software model maps to a current of 1 nA in hardware, establishing the correspondence between digital and analog representations. Transistor dimensions are sized to a few square micrometers to mitigate Early voltage effects and minimize process-induced mismatch. The topology used in this work uses transistor stacking, motivating the implementation of this design in the 180 nm CMOS technology from X-FAB as it allows for a larger voltage headroom.

All transistor-level simulations were run on a workstation equipped with an AMD Ryzen Threadripper 3970X (32 cores, 3.7 GHz) and 256 GB of RAM, inside a virtual machine. A single transient inference

simulation completes in under one minute; Monte Carlo (200 samples) and PVT corner analyses scale linearly with the number of samples and corners.

D.1 Current mirror weight implementation

Figure 4 illustrates the fundamental current mirror topology used to implement weighted connections. In subthreshold operation, a diode-connected input transistor converts an input current I_x into a gate voltage $V_x = V_y$, which is shared with the output transistor. Since both transistors operate at identical gate-source voltages, their drain currents are primarily determined by their width ratio:

$$I_y \approx \frac{W_{\text{out}}}{W_{\text{in}}} I_x. \quad (26)$$

In practice, the relationship deviates from ideal scaling due to drain-source voltage dependence and other second-order effects. To account for this, we perform exhaustive characterization sweeping W_{out} and I_x for fixed W_{in} , measuring the resulting I_y . This calibration produces a lookup table mapping desired output currents to optimal width selections, ensuring accurate weight implementation despite non-ideal transistor behavior. PMOS mirrors source current from the supply voltage, while NMOS mirrors sink current to ground—both polarities are required for implementing signed weights in FC layers.

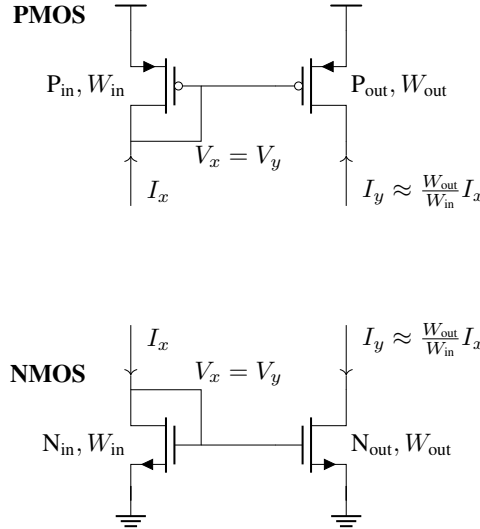


Figure 4: **Current mirror topologies for weight implementation.** Top: PMOS current mirror (sources current from supply). Bottom: NMOS current mirror (sinks current to ground).

Programmable weight implementation is achieved using binary-quantized current mirrors controlled by a shift register, as illustrated for the PMOS case in Figure 5.

D.2 FC layers

The FC layer computes

$$y_j = \text{ReLU} \left(\sum_{i=1}^d w_{ij} x_i + b_j \right) \quad \forall j \in [1, d]. \quad (27)$$

This is realized entirely in the current domain using parallel current mirror banks (Figure 6). Each input voltage V_i drives both PMOS and NMOS mirrors: PMOS transistors with width ratios W_i^-/W_j implement negative weights by sourcing current to the upper summation node Σ^- , while NMOS transistors with ratios W_i^+/W_j implement positive weights by sinking current from the lower node Σ^+ . A vertical wire connects both nodes, where KCL computes the weighted sum. Bias currents $I_{b,j}^+$ and $I_{b,j}^-$ add the affine term. A diode-connected PMOS at the output conducts only when net

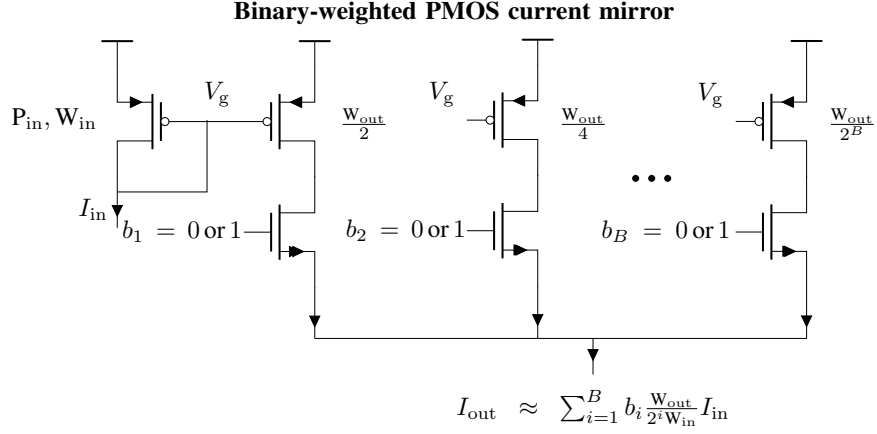


Figure 5: **Binary-weighted PMOS current mirror for programmable weight implementation.** The effective output current is set by enabling combinations of binary-scaled mirror branches, allowing discrete (quantized) weight tuning via a shift register.

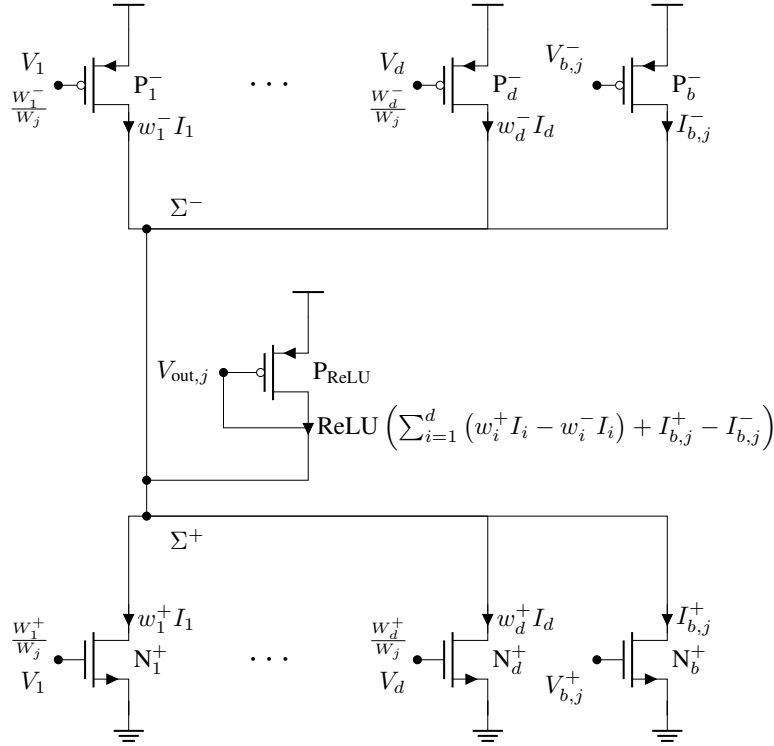


Figure 6: **FC layer with ReLU activation.** PMOS mirrors (top) implement negative weights; NMOS mirrors (bottom) implement positive weights. The diode-connected PMOS harvests net positive current.

current flows from the supply, implementing ReLU activation and providing output voltage $V_{out,j}$ for subsequent stages.

For layers requiring anti-ReLU activation, the output transistor is replaced with a diode-connected NMOS that sinks current to ground when the net sum is negative (Figure 7), implementing $y = \min(0, x)$.

All NMOS transistors are sized with a width of $5 \mu\text{m}$ and a length of $5 \mu\text{m}$, while PMOS transistors are sized with a width of $5.5 \mu\text{m}$ and a length of $5 \mu\text{m}$. This sizing ensures robustness against mismatch

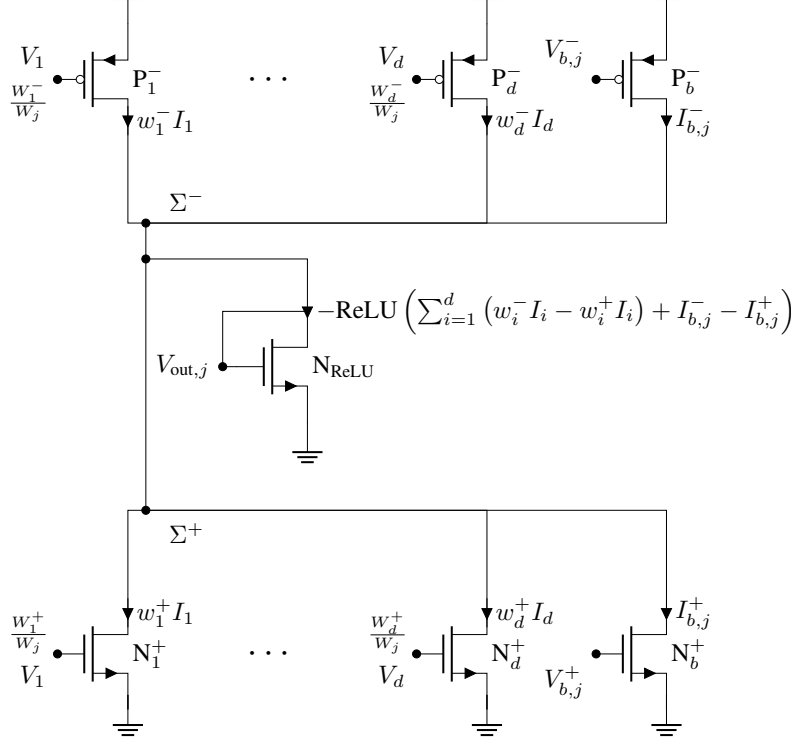


Figure 7: **FC layer with anti-ReLU activation.** Same structure as Figure 6, with diode-connected NMOS at output.

and mitigates undesired short-channel effects. An exception is made for current mirrors implementing weight multiplication, where output transistor widths are selected from a precomputed lookup table. This table was generated by exhaustively simulating all combinations of input currents and transistor widths, mapping each desired weight-current product to the optimal width selection.

D.3 Skip connections

Skip connections, which bypass intermediate layers to add earlier activations directly to later ones, are implemented naturally in the current domain using KCL. Since the circuit operates exclusively with positive currents, skip connections are separated into positive and negative branches. This separation preserves the unipolar constraint throughout the network while enabling residual-style architectures. No additional active components are required—skip connections reduce to wire routing that joins current paths at the appropriate summation nodes.

D.4 CMOS FQ BMRU

The proposed circuit is based on a dual-Heaviside feedback structure, conceptually illustrated in Figure 8 (top left panel). It consists of two interdependent nonlinear thresholding elements:

- A primary Heaviside element that activates when the input current I_{in} exceeds a threshold I_{thresh} (the high threshold of the Schmitt trigger), producing an output current I_{gain} (the high output value).
- A secondary Heaviside element in feedback, triggered by the output of the first stage. It activates to inject a feedback current I_{width} at the input of the circuit.

The hysteresis behavior arises naturally from the feedback current I_{width} , which lowers the effective input threshold to $I_{\text{thresh}} - I_{\text{width}}$, thus enabling bistability. The three control parameters I_{thresh} , I_{gain} , and I_{width} are all independently adjustable via bias voltages, allowing straightforward tuning. Because the circuit operates only with positive currents, the constraint $I_{\text{thresh}} > I_{\text{width}}$ must be met to preserve bistability, ensuring that the positive feedback does not override the comparator threshold.

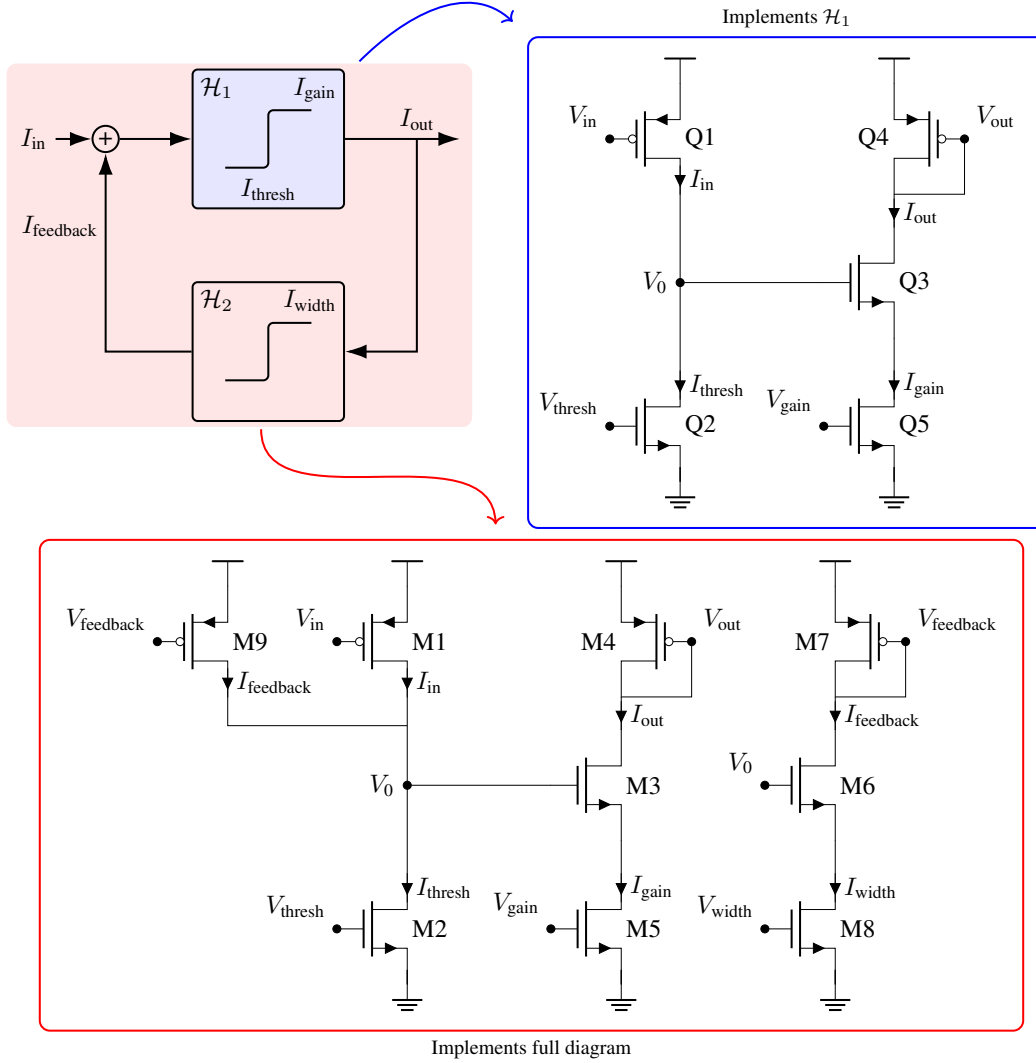


Figure 8: **CMOS implementation of the FQ BMRU cell.** Top left: conceptual dual-Heaviside feedback architecture. Top right (blue): single Heaviside element $\mathcal{H}_1$ using 5 transistors. Bottom (red): complete Schmitt trigger with feedback, using 9 transistors total.

The top right panel (blue) of Figure 8 shows a minimal CMOS circuit that implements a fully tunable unipolar Heaviside function ($\mathcal{H}_1$) in current mode. The input current is provided through a voltage V_{in} at the output of a PMOS current mirror (Q1), while the threshold current I_{thresh} is set through a voltage V_{thresh} by an NMOS transistor (Q2). Together, Q1 and Q2 form a switching current comparator. If $I_{in} < I_{thresh}$, then $V_0 \approx 0$ and Q3 is off, resulting in a zero output current I_{out} . As soon as $I_{in} > I_{thresh}$, V_0 increases to the supply voltage and Q3 turns on, allowing current to flow in the output branch. At that point, the current flowing through this branch corresponds to the bias current I_{gain} , also set through a voltage V_{gain} by an NMOS transistor (Q5). The output current flows through the diode-connected transistor Q4 and can be mirrored to other stages via a PMOS transistor driven by V_{out} .

The bottom panel (red) of Figure 8 represents the feedback interconnection of the two Heaviside functions, forming a current-mode unipolar Schmitt trigger capable of operating with nW-level supply power. $\mathcal{H}_1$ (Q1–Q5) is implemented through transistors M1–M5. For $\mathcal{H}_2$ to switch concurrently with $\mathcal{H}_1$, they share the same switching voltage V_0 , removing the need for a separate comparator branch (like M1–M2). This means that if $I_{in} < I_{thresh}$, M6 is off and $I_{feedback} = I_{out} = 0$. Once $I_{in} > I_{thresh}$, M6 turns on and $I_{feedback}$ reaches I_{width} , the bias current of the feedback branch set through V_{width} by

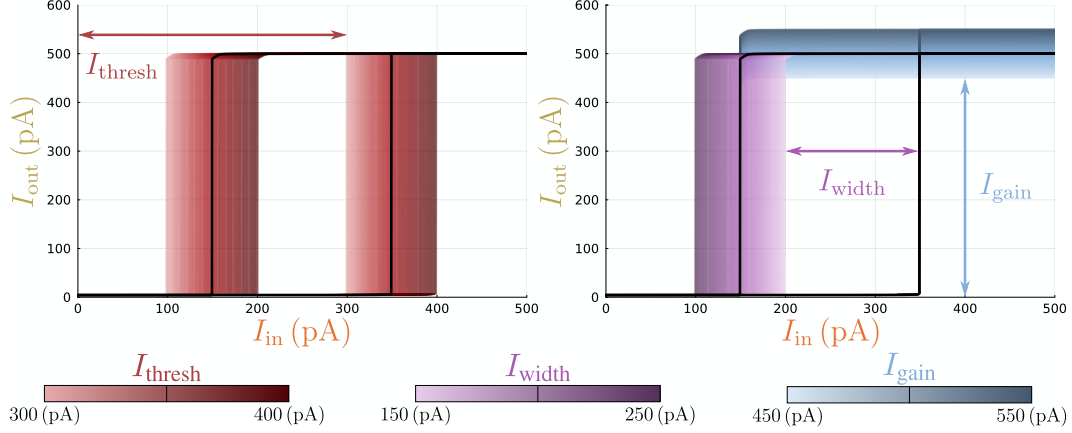


Figure 9: **Tunability of the CMOS implementation of the FQ BMRU cell.** Input-output current relationship of the CMOS implementation of the FQ BMRU cell. All thresholds and output gain are independently tunable via bias currents.

an NMOS transistor (M8). This feedback current is mirrored through a PMOS current mirror (M7 and M9), and injected back into the comparator branch of $\mathcal{H}_1$ (M1, M2, and M9), thereby increasing the input current by I_{feedback} .

These three parameters correspond directly to the FQ BMRU formulation: I_{thresh} implements β_{hi} , $I_{\text{thresh}} - I_{\text{width}}$ implements β_{lo} , and I_{gain} implements α . Figure 9 represents independent tunability of these three parameters.

All NMOS transistors of the cell are sized with a width of $5\text{ }\mu\text{m}$ and a length of $5\text{ }\mu\text{m}$, while PMOS transistors of the cell are sized with a width of $5.5\text{ }\mu\text{m}$ and a length of $5\text{ }\mu\text{m}$. This sizing ensures robustness against mismatch and mitigates undesired short-channel effects. An exception was made for M3 and M6, which operate primarily in a digital regime and were sized with a width of $2\text{ }\mu\text{m}$ and minimal length.

D.5 Simulation results of the CMOS FQ BMRU

All schematic-level simulations were conducted using Spectre with X-FAB 180 nm models at a supply voltage of 1.8 V. Baseline parameters were selected using the lookup table approach described in Section D.1: $I_{\text{gain}} = 486\text{ pA}$, $I_{\text{thresh}} = 368\text{ pA}$, and $I_{\text{width}} = 216\text{ pA}$, yielding switching thresholds of 150 pA and 350 pA with a high output state of 500 pA.

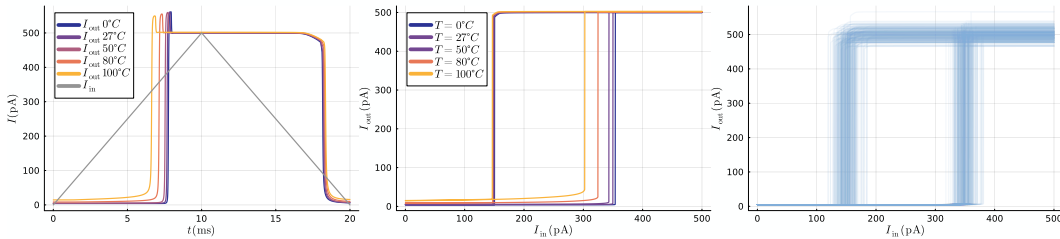


Figure 10: **CMOS FQ BMRU cell simulation results.** Transient simulation under triangular input current for different operating temperatures (left). DC sweep demonstrating hysteretic behavior for different operating temperatures (middle). Monte Carlo analysis with 3σ process variation at room temperature (right). Baseline parameters: $I_{\text{gain}} = 486\text{ pA}$, $I_{\text{thresh}} = 368\text{ pA}$, $I_{\text{width}} = 216\text{ pA}$.

Transient response and DC characteristics. To evaluate hysteresis, the input current I_{in} was swept linearly from 0 to 500 pA and back using a triangular waveform. As shown in Figure 10 left

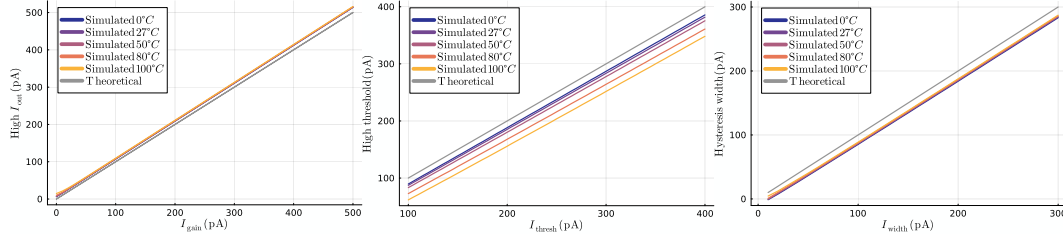


Figure 11: **Tunability of CMOS FQ BMRU cell parameters.** I_{gain} sweep from 0 to 500 pA for different operating temperatures (left). I_{thres} sweep from 100 pA to 400 pA with $I_{\text{width}} = 50$ pA (middle). I_{width} sweep from 10 pA to 300 pA (right). Baseline parameters as in Figure 10.

for different operating temperatures, the circuit exhibits clear bistable behavior with sharp transitions. The switching points align with the expected levels of $I_{\text{thres}} - I_{\text{width}}$ and I_{thres} , and the high-state output current closely follows I_{gain} . A small ($\sim 10\%$) overshoot is observed at transitions, attributed to the positive feedback inherent in the dual-Heaviside topology. DC simulations confirm the existence of two stable states and well-defined hysteresis (Figure 10 middle). Temperature mainly affects the upper switching point, determined by the M1–M2 comparator. However, since the circuit operates in the ultra-low power regime, typical operating conditions remain well below 80°C , ensuring robust operation.

Mismatch analysis. Monte Carlo simulations were performed under 3σ process variation to evaluate robustness to device mismatch. The circuit consistently maintained correct bistable switching behavior, with only minor deviations—typically a few tens of pA—in the threshold currents, hysteresis width, and high output level (Figure 10 right). These variations remain well within functional margins. The observed mismatches arise primarily from variations in current mirrors and biasing transistors, and could be further reduced by increasing device dimensions at the cost of silicon area.

Tuning properties. Systematic sweeps of each bias current confirm that I_{gain} , I_{thres} , and I_{width} independently control their corresponding functions (output level, threshold, and hysteresis width) in a linear manner across operating temperatures (Figure 11). Small offsets due to subthreshold leakage currents result in relative errors of 2.8%, 4.5%, and 5.25% for I_{gain} , I_{thres} , and I_{width} respectively at room temperature and highest tested currents. These predictable deviations do not impair functionality and can be compensated through calibration via lookup tables, as described in Section D.1.

E Power breakdown analysis

To isolate the contribution of each building block, separate supply nets are assigned to BMRU cells and FC layers in Cadence Spectre simulation. Figure 12 presents the resulting power distribution across the 50 inferences reported in the main text. At $d = 4$, power splits approximately evenly between BMRU cells (~ 40 nW median) and FC layers (~ 30 nW median). Notably, the power variance across inferences is substantially lower for BMRU cells than for FC layers, consistent with the observation that discrete-output dynamics yield stable, input-independent power consumption.

Component-level power analysis reveals a fundamental asymmetry in the network architecture: FQ BMRU cells scale linearly with state dimension d (each cell is independent), while FC layers scale quadratically ($d \times d$ weight matrices require d^2 current mirrors). Estimated power scaling based on per-component measurements at $d = 4$ from Cadence Spectre simulation is used to extrapolate to higher state dimensions (Table 4).

F Additional inference traces

This appendix presents additional Cadence Spectre transient simulations that validate the hardware implementation against software predictions. Figures 14–23 show ten representative inference examples, each displaying the two output logit currents (class 0: “background”; class 1: “yes”) and instantaneous power consumption. The samples were drawn from the test set to cover diverse

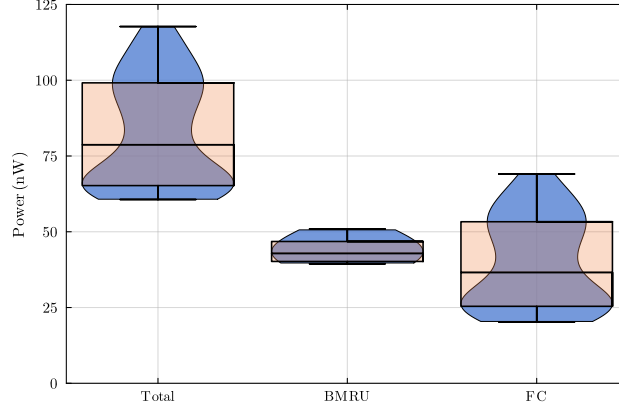


Figure 12: **Component-level power breakdown across 50 inferences.** Power consumption of FQ BMRU cells versus FC layers for 50 inference samples. The approximately even split at $d = 4$ indicates that both components contribute comparably to efficiency. FQ BMRU cells exhibit substantially lower power variance, consistent with stable discrete-output dynamics.

Table 4: **Estimated power scaling with state dimension.** BMRU power increases linearly with the state dimension ($\propto d$), whereas FC and skip connections power grows quadratically ($\propto d^2$), leading to a decreasing relative cost of recurrence as d increases. Estimates are extrapolated from per-component power measured at $d = 4$.

	$d = 4$	$d = 8$	$d = 16$	$d = 32$	$d = 64$
BMRU ($\sim$ nW)	40 (57%)	80 (40%)	160 (25%)	320 (14%)	640 (8%)
FC and skip connections ($\sim$ nW)	30 (43%)	120 (60%)	480 (75%)	1920 (86%)	7680 (92%)

input conditions (positive keyword, negative words, background noise), and include one case of hardware–software discrepancy as well as one misclassification common to both implementations. The main-text figure (Figure 2) corresponds to seed 51 and is omitted here to avoid redundancy.

Classification is determined by majority voting over all timesteps. Across 50 test samples, hardware predictions match software in 49 cases. The single discrepancy (Figure 21) occurs on a sample where the software prediction was maximally uncertain, with 51 timesteps predicting “yes” and 50 predicting “background”, effectively a tie. This confirms that hardware–software mismatches arise only at the decision boundary rather than from systematic circuit errors. Power consumption remains consistently around 100 nW across all samples for the RNN core, with minor input-dependent fluctuations reflecting switching activity.

G PVT corner validation

PVT (Process, Voltage, Temperature) corner analysis assesses robustness to manufacturing and environmental variation. Figures 24 and 25 present results across all five process corners (TT, FF, SS, FS, SF), the temperature range $[-27^\circ\text{C}, 27^\circ\text{C}, 81^\circ\text{C}]$, and $\pm 10\%$ supply voltage variation. Classification correctness is preserved across all conditions for both representative inferences.

H Monte Carlo mismatch analysis

Monte Carlo mismatch analysis (200 samples, 3σ on all transistors) was performed on all 11 inferences shown in the manuscript. The results (Figures 26–36) confirm that misclassifications under mismatch occur exclusively on samples for which the network is already uncertain (or wrong) under nominal conditions. The bistable dynamics confer inherent mismatch immunity: discrete thresholding prevents small parameter perturbations from propagating to output errors unless the input already lies near the decision boundary.

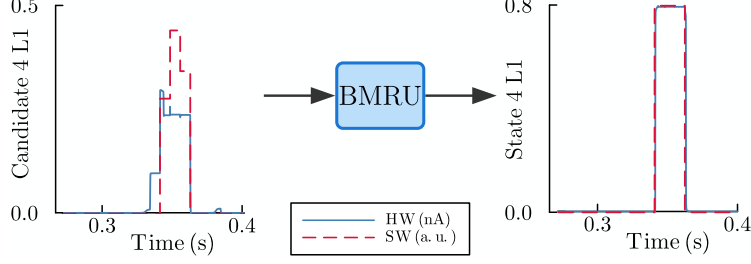


Figure 13: **Illustration of $20\times$ error suppression at BMRU cell boundaries.** During inference for the sample in Figure 2 (layer 1, fourth dimension), candidate signals at the BMRU inputs (left) exhibit discrepancies between software predictions and circuit simulations. After processing by the BMRU, the resulting states (right) show significantly improved agreement, demonstrating both strong noise attenuation and effective error suppression.

I Multi-class KWS evaluation

To verify that the approach generalizes beyond binary classification, the FQ BMRU is evaluated on 11-class digit recognition (“zero” through “nine” plus background noise) from the Google Speech Commands dataset. A 2×16 architecture achieves competitive accuracy and substantially widens the output margin between classes (Figure 37), thereby improving robustness to mismatch. Networks of this size remain within the sub-microwatt consumption estimates of Table 4. Given the strong hardware–software agreement demonstrated at $d = 4$, comparable behavior is expected in hardware at larger dimensions.

J Signal agreement between software and hardware

Figures 38–49 provide a detailed comparison of intermediate signals between software predictions and Cadence Spectre simulation at every stage of the network. The close agreement validates the end-to-end co-design methodology and confirms that the FQ reformulation produces a faithful mapping between learned parameters and circuit behavior. The $20\times$ error suppression at BMRU cell boundaries is clearly visible: candidate signals accumulate errors on the order of tens of picoamperes, while cell outputs exhibit errors of only a few picoamperes. Figure 13 illustrates this $20\times$ error suppression at BMRU cell boundaries for the sample shown in Figure 2, at layer 1 and in the fourth dimension.

K Programmable hardware overhead

Based on Cadence simulations of the non-programmable circuit (total area $\sim 20,000 \mu\text{m}^2$) together with characterization of programmable building blocks, the following overhead is estimated for a fully programmable version employing binary-weighted current mirrors and digital shift registers.

Binary-weighted current mirrors (4-bit resolution). Inactive mirror branches consume negligible leakage, so the power overhead is insignificant. The area overhead is estimated at $\sim 5,000 \mu\text{m}^2$ and scales linearly with the number of parameters.

Shift registers ($d = 4$, 4-bit quantization). Power overhead is approximately $\sim 100 \text{ nW}$; area overhead is $\sim 60,000 \mu\text{m}^2$. Both scale linearly with the number of parameters.

Bias generation. Drawing on current bias generators from the literature [107, 108], extended with temperature compensation, an upper bound of $\sim 50 \text{ nW}$ at 4-bit resolution is estimated. This figure is conservative, as bias currents can be shared across cells.

Bias routing. Routing overhead is minimal, as the xh018 process provides nine metal layers.

In total, the programmable system would remain within the sub-microwatt envelope for networks up to $d = 16$.

L Broader impact

This work aims to advance machine learning through ultra-low power analog hardware implementations. The central contribution of this work is demonstrating that recurrent temporal processing can be added to analog inference platforms at linear marginal power cost. Since feedforward analog computation using current mirrors and KCL is already well established, this result implies that any existing or future analog feedforward accelerator could, in principle, gain the ability to process temporal sequences by integrating BMRU cells with modest additional overhead. This architectural composability extends further: because the FQ BMRU provides persistent bistable memory while other recurrent architectures (SSMs, gated RNNs) provide transient dynamics, hybrid designs combining both could cover a broader range of temporal processing requirements while remaining within ultra-low power budgets. The practical implications span domains where continuous temporal inference under extreme power constraints is currently infeasible: always-on environmental and structural monitoring over years-long deployments, implantable biomedical devices operating without frequent recharging, and distributed sensor networks where battery replacement is impractical or impossible. More broadly, by narrowing the gap between analog and digital inference capabilities, this work contributes to reducing the energy footprint of edge AI at a time when the computational demands of AI are growing rapidly. As with any advance in always-on sensing, deployment should be accompanied by appropriate consent and governance frameworks, particularly for audio-processing applications.

M Software and data

All code required to reproduce the experiments will be publicly available once the license is finalized¹. The repository includes:

- **Software training (Table 1):** JAX/Flax implementations of the BMRU, FQ BMRU, LRU, and minGRU, with both the software and hardware backbones, along with training scripts and evaluation code for all benchmarks in Table 1.
- **Noise robustness analysis (Figure 3):** Complete inference and noise injection pipeline used to produce Figure 3.
- **“yes” KWS (Section 3):** Complete training pipeline and export scripts for mapping learned parameters to circuit elements for analog inference.
- **Datasets:** The MNIST, Google Speech Commands, and Long Range Arena (ListOps) datasets [91, 56, 63] are publicly available. Preprocessing scripts and train/validation/test splits follow standard practices and are included in the repository.
- **Post-processing:** Julia scripts for parsing Cadence simulation outputs (logit currents, power traces, and all additional figures) and generating transistor width lookup tables from device characterization data.

Cadence Virtuoso design files (schematics, symbols, and Maestro configurations) are not included due to software licensing constraints and version compatibility issues, but are available upon reasonable request to the corresponding author for researchers with access to Cadence tools.

¹github

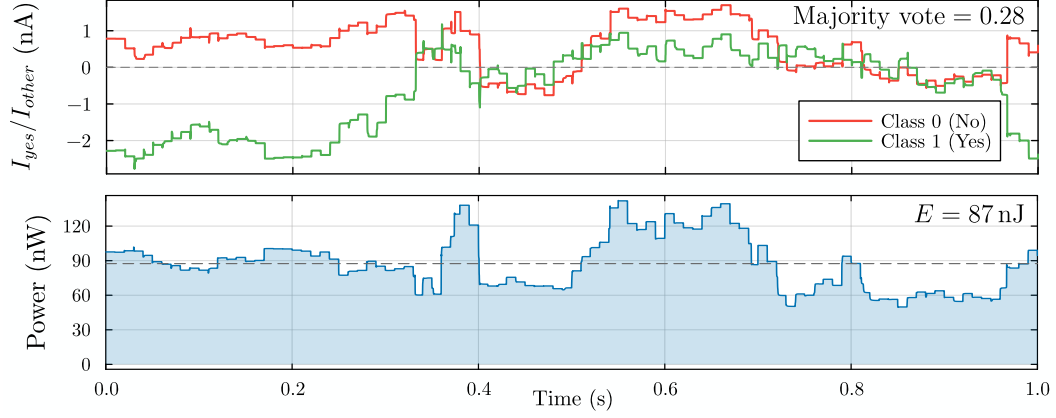


Figure 14: **Hardware inference traces from Cadence Spectre simulation (seed 45).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “down”. Hardware prediction via majority vote: “background”. Software prediction: “background”.

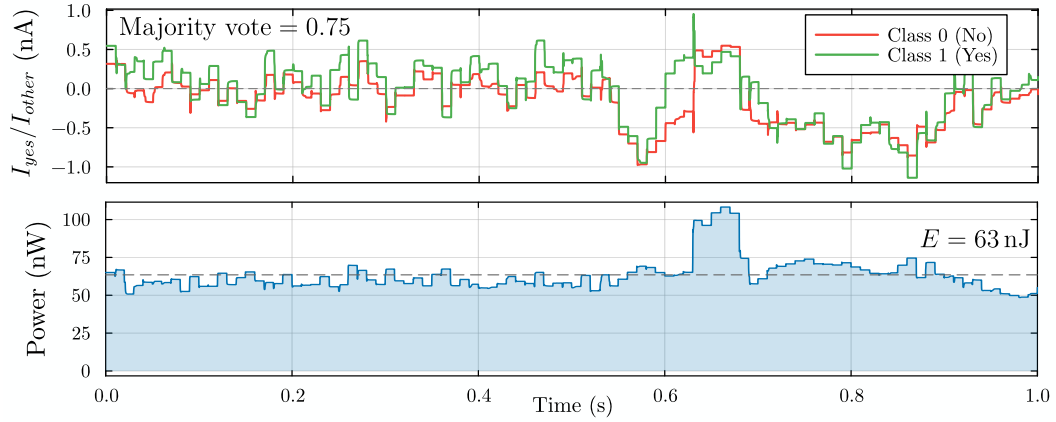


Figure 15: **Hardware inference traces from Cadence Spectre simulation (seed 47).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “yes”. Software prediction: “yes”.

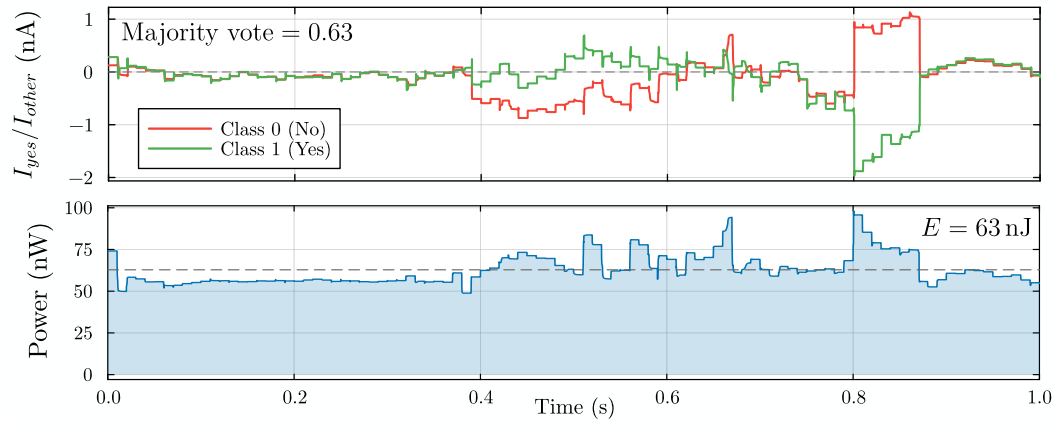


Figure 16: **Hardware inference traces from Cadence Spectre simulation (seed 48).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “yes”. Software prediction: “yes”.

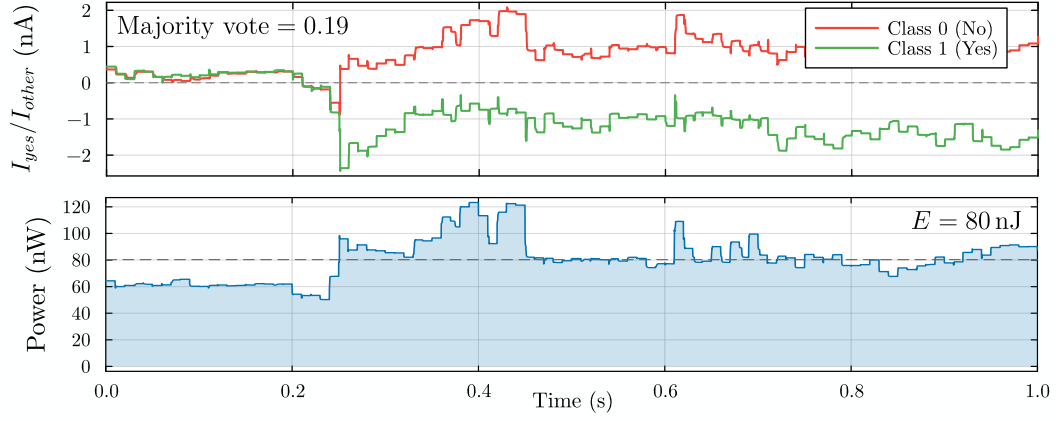


Figure 17: **Hardware inference traces from Cadence Spectre simulation (seed 49).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “background”. Software prediction: “background”. In this case, both implementations misclassify the sample. Note that the spoken “yes” has an elongated “s” sound, reflected in the output logit dynamics.

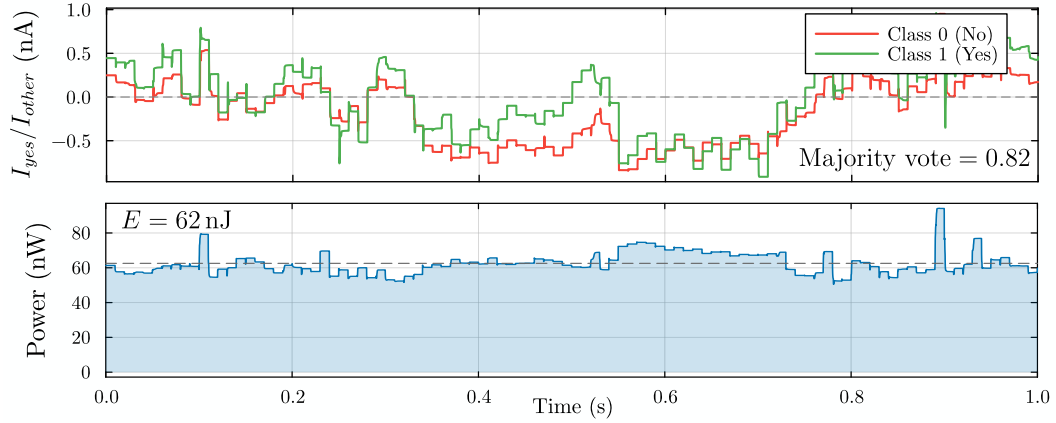


Figure 18: **Hardware inference traces from Cadence Spectre simulation (seed 50).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “yes”. Software prediction: “yes”.

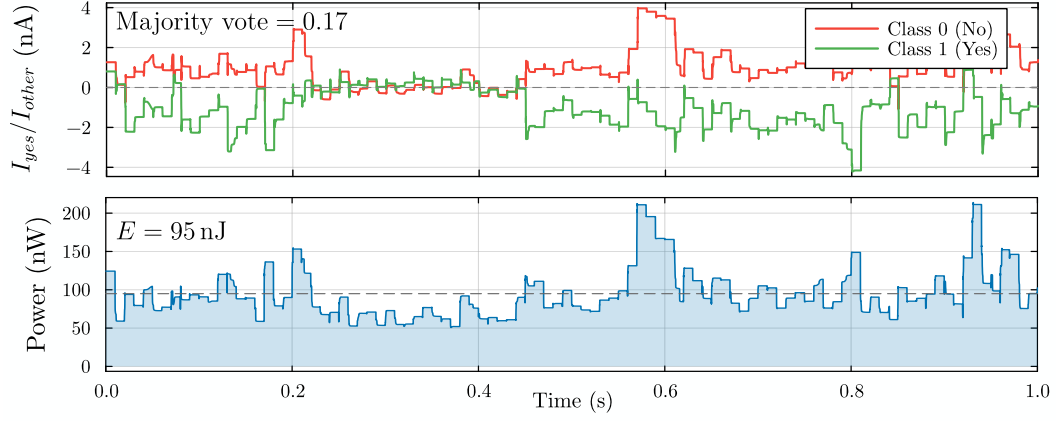


Figure 19: **Hardware inference traces from Cadence Spectre simulation (seed 52).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: background noise (no speech). Hardware prediction via majority vote: “background”. Software prediction: “background”.

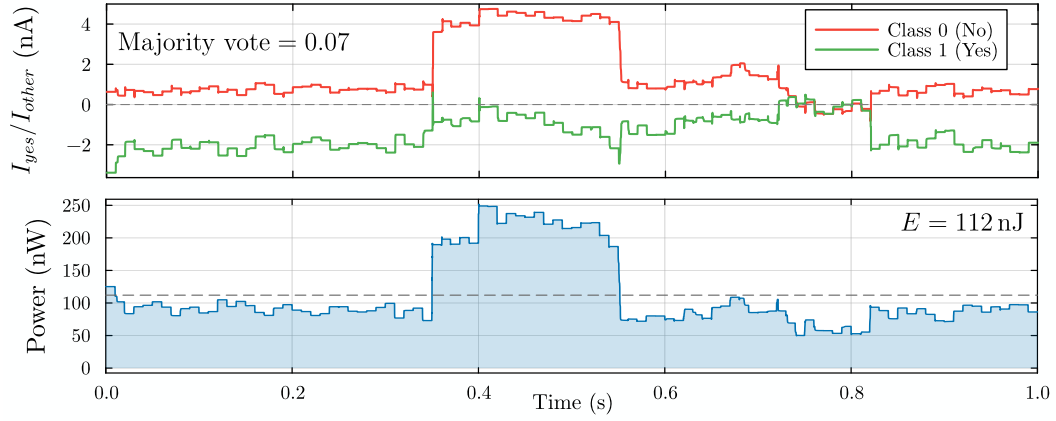


Figure 20: **Hardware inference traces from Cadence Spectre simulation (seed 66).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “up”. Hardware prediction via majority vote: “background”. Software prediction: “background”.

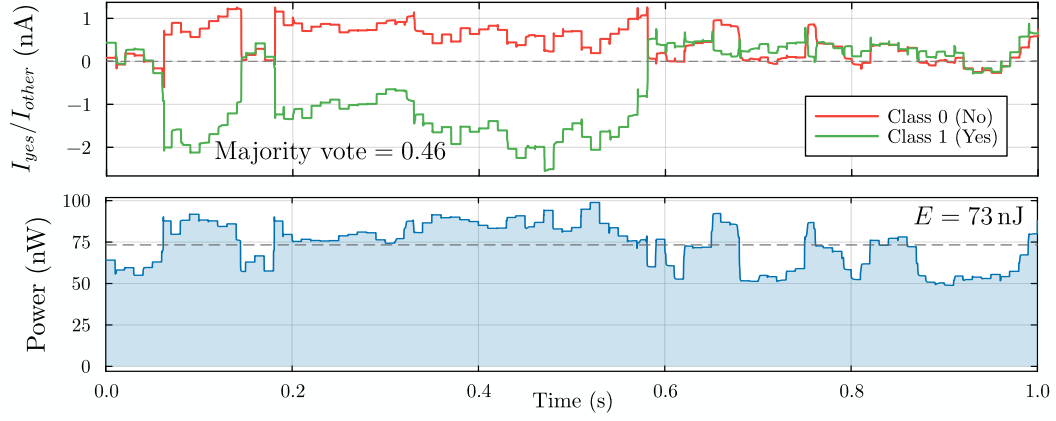


Figure 21: **Hardware inference traces from Cadence Spectre simulation (seed 67).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “back-ground”. Software prediction: “yes”. This is the only case across 50 test samples where hardware and software predictions differ.

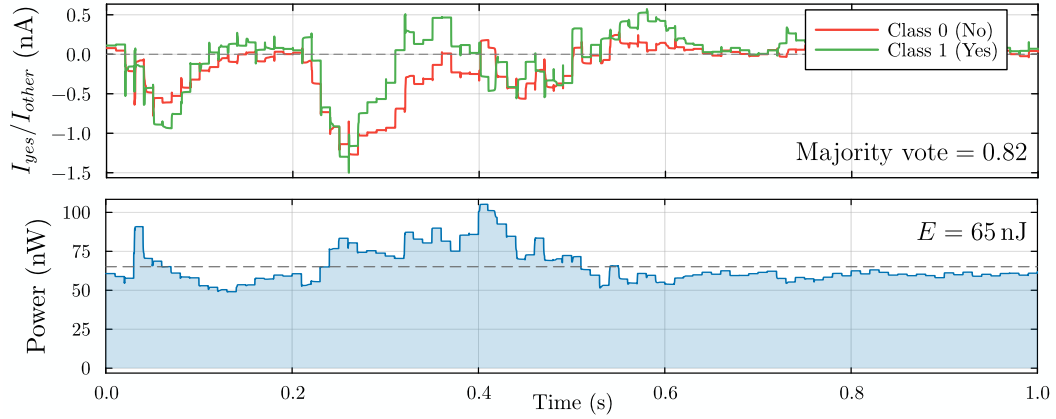


Figure 22: **Hardware inference traces from Cadence Spectre simulation (seed 68).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “yes”. Hardware prediction via majority vote: “yes”. Software prediction: “yes”.

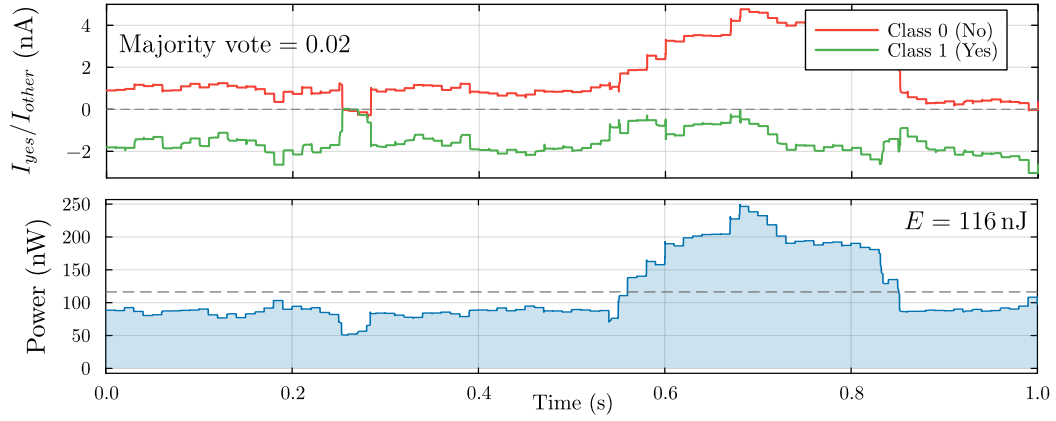


Figure 23: **Hardware inference traces from Cadence Spectre simulation (seed 61).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) and power consumption (bottom) over the 101-frame input sequence. Spoken word: “right”. Hardware prediction via majority vote: “background”. Software prediction: “background”.

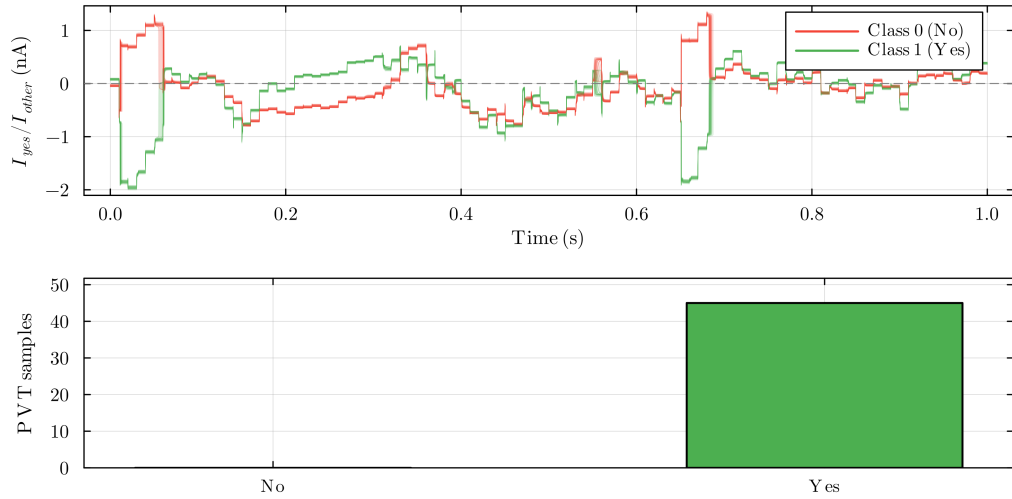


Figure 24: **PVT corner validation from Cadence Spectre simulation (seed 51).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each PVT condition (bottom). All five process corners (TT, FF, SS, FS, SF), three temperatures (-27°C , 27°C , 81°C), and $\pm 10\%$ supply voltage variation are evaluated. Spoken word: “yes”. Correct classification is maintained across all conditions.

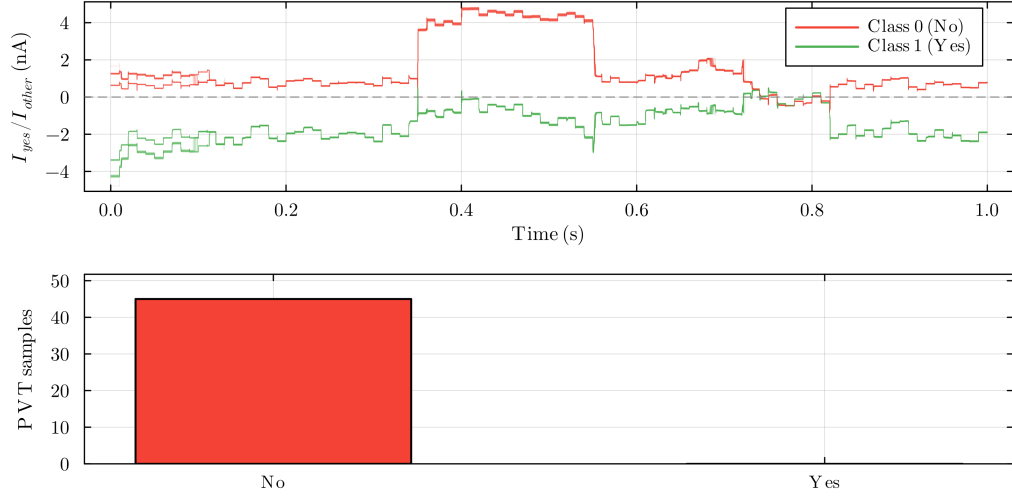


Figure 25: **PVT corner validation from Cadence Spectre simulation (seed 66)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each PVT condition (bottom). All five process corners (TT, FF, SS, FS, SF), three temperatures (-27°C , 27°C , 81°C), and $\pm 10\%$ supply voltage variation are evaluated. Input: background noise. Correct classification is maintained across all conditions.

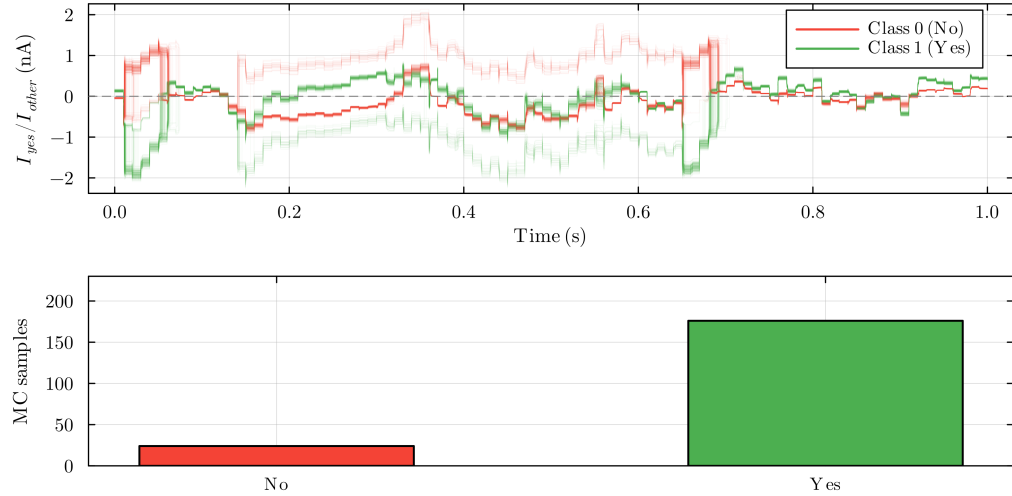


Figure 26: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 51)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “yes”. Impaired sample rate: 11.5%.

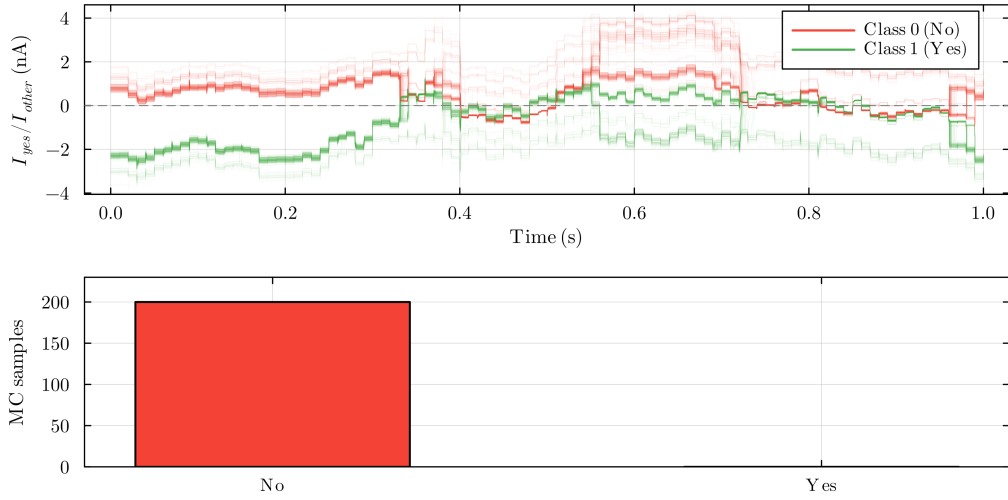


Figure 27: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 45).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “down”. Nominal prediction: “background”. Impaired sample rate: 0%.

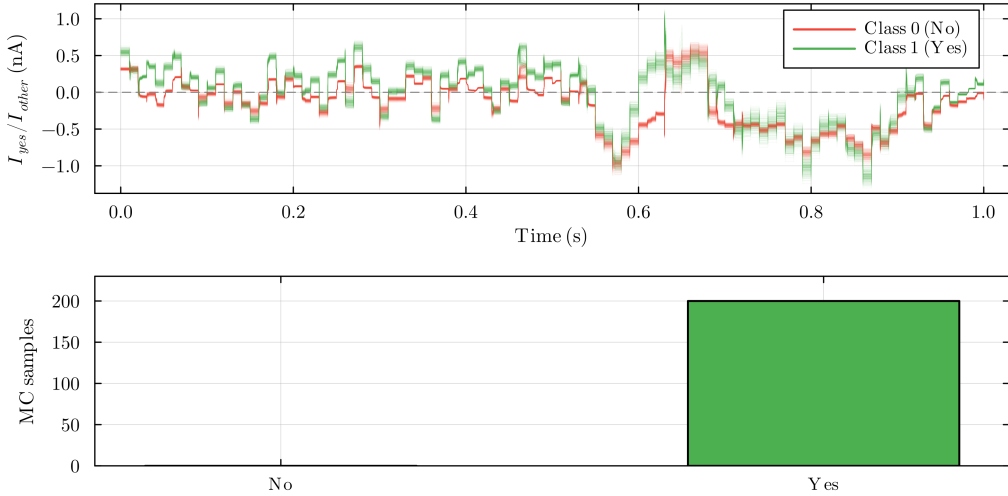


Figure 28: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 47).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “yes”. Impaired sample rate: 0%.

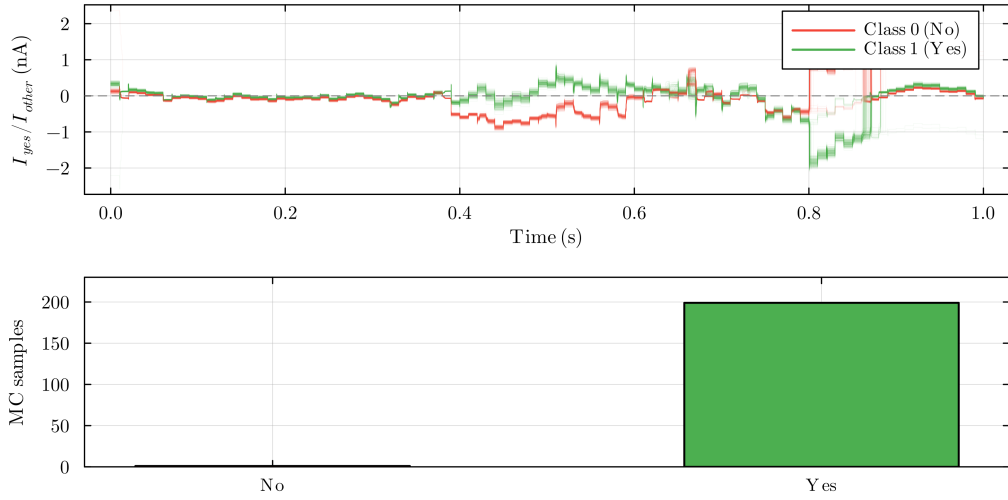


Figure 29: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 48).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “yes”. Impaired sample rate: 0.5%.

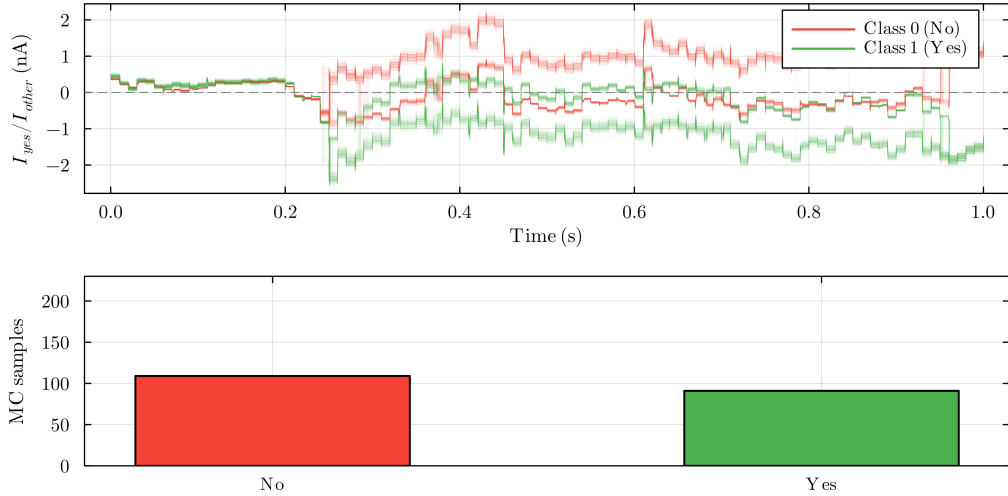


Figure 30: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 49).** Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “background” (misclassified under nominal conditions). Impaired sample rate: 45.5%.

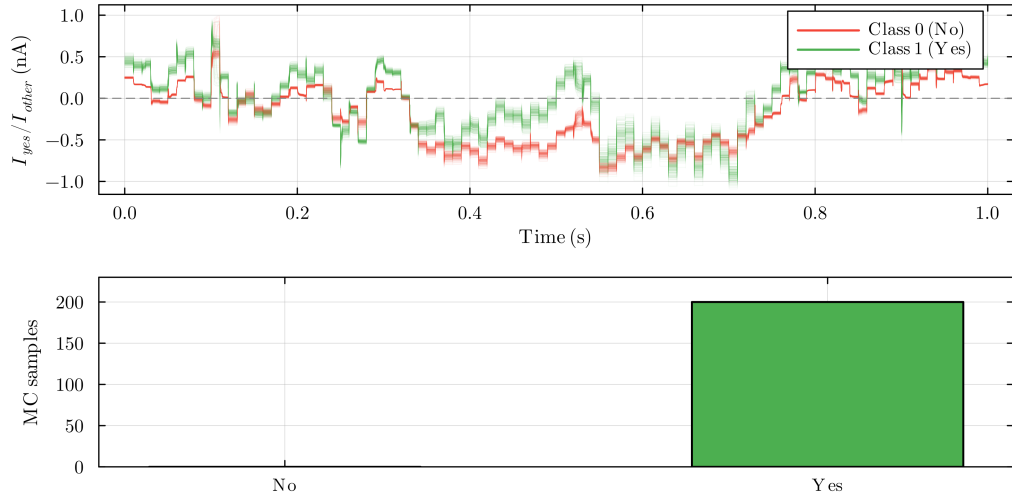


Figure 31: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 50)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “yes”. Impaired sample rate: 0%.

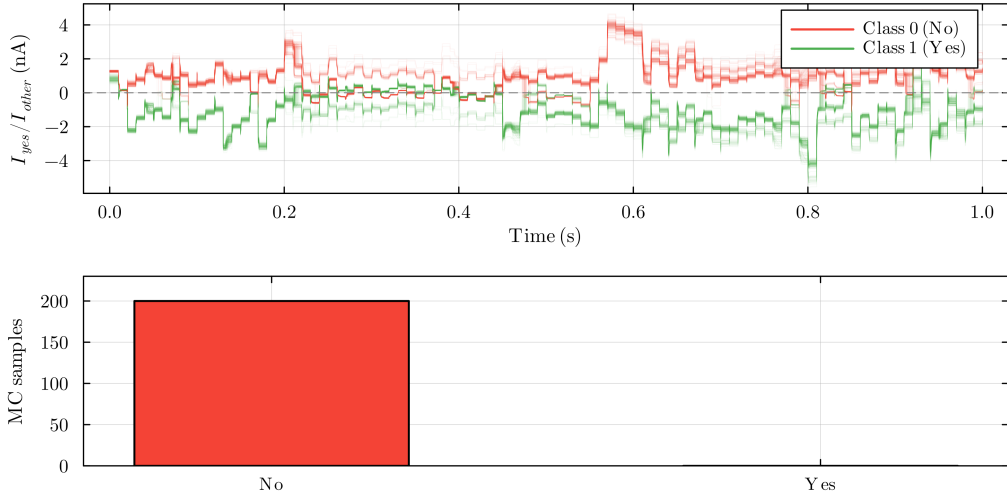


Figure 32: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 52)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Input: background noise (no speech). Nominal prediction: “background”. Impaired sample rate: 0%.

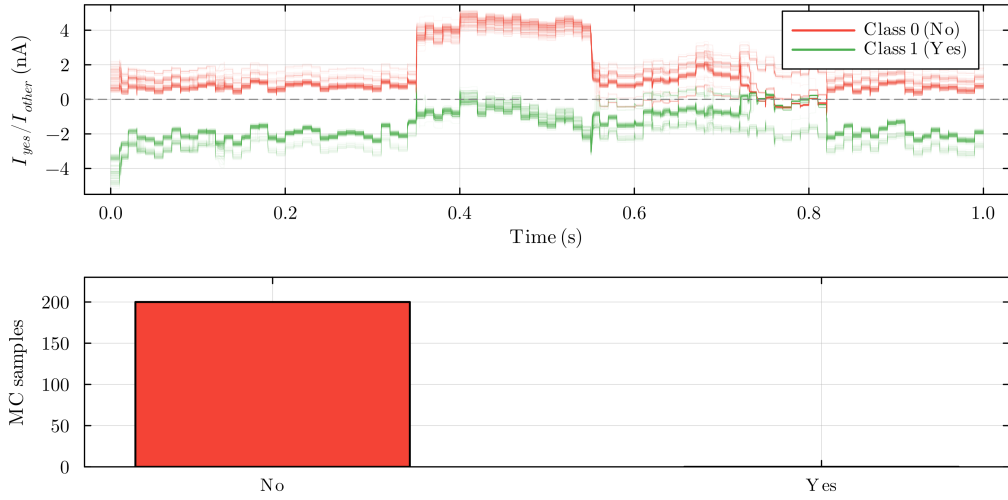


Figure 33: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 66)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “up”. Nominal prediction: “background”. Impaired sample rate: 0%.

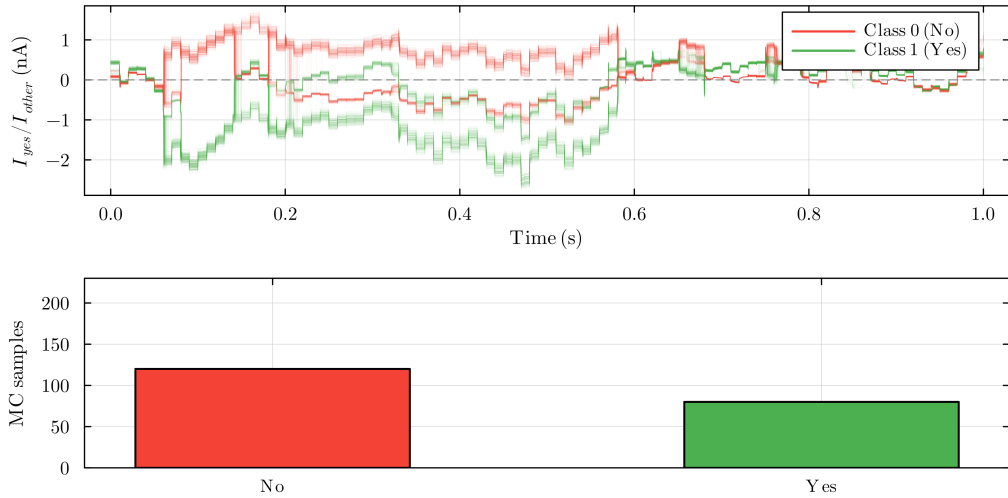


Figure 34: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 67)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal hardware prediction: “background” (already in disagreement with software under nominal conditions). Impaired sample rate: 41%.

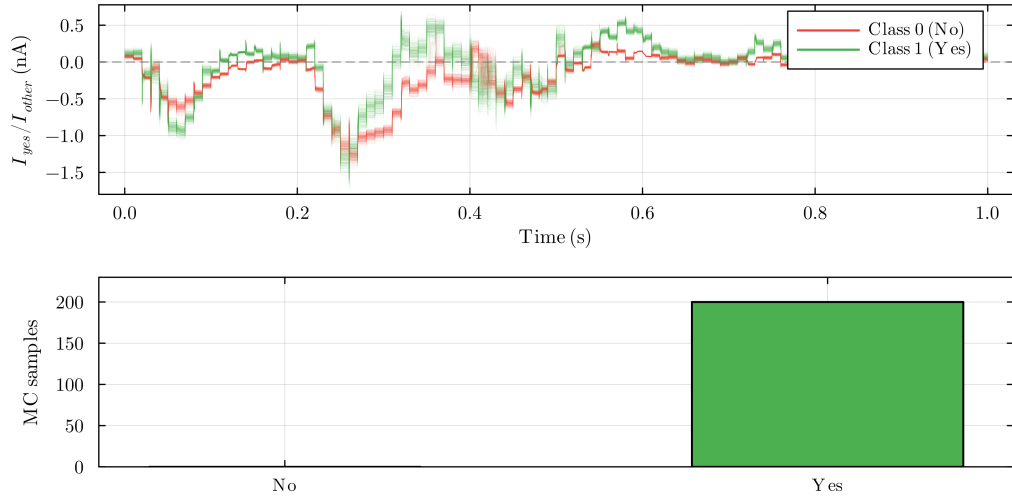


Figure 35: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 68)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “yes”. Nominal prediction: “yes”. Impaired sample rate: 0%.

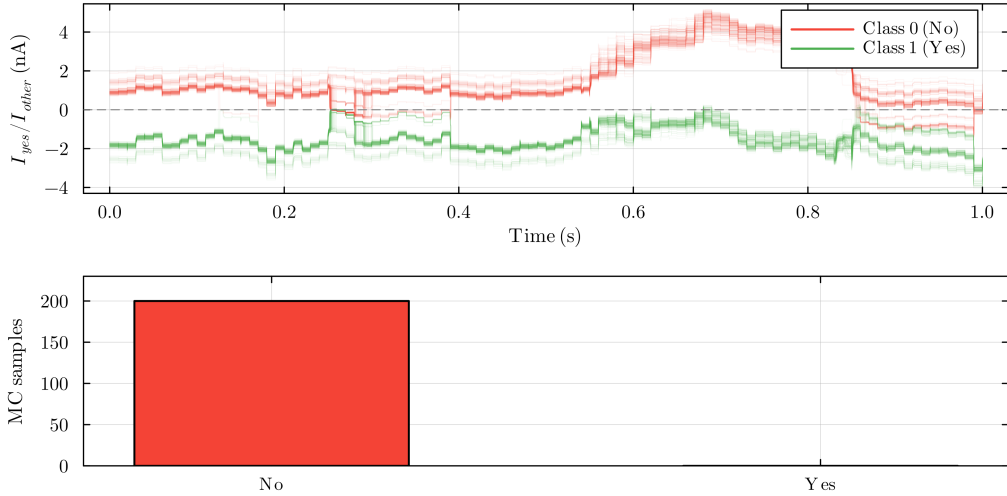


Figure 36: **Monte Carlo mismatch analysis from Cadence Spectre simulation (seed 61)**. Each panel shows output logit currents (top: I_{yes} in green, I_{no} in red) over the 101-frame input sequence and the corresponding prediction for each Monte Carlo sample (bottom). Analysis performed with 3σ mismatch variation on all transistors (200 samples). Spoken word: “right”. Nominal prediction: “background”. Impaired sample rate: 0%.

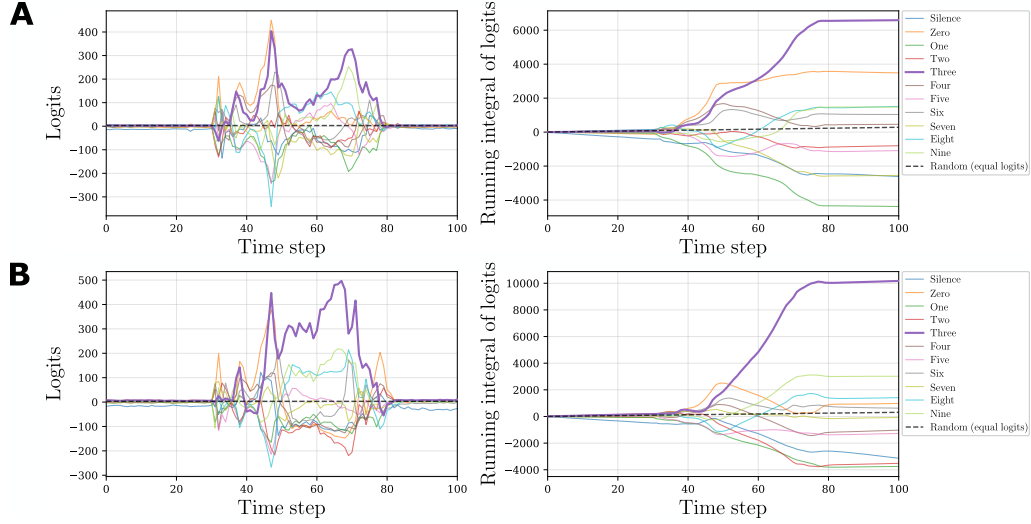


Figure 37: **Multi-class KWS evaluation (11 classes), "three" spoken. A.** 2×4 network. Logit time evolution (left) and integrated logits used for the final classification decision (right). The classification is correct, but the narrow decision margins leave the prediction vulnerable to mismatch. **B.** 2×16 network. Logit time evolution (left) and integrated logits (right). The classification is correct and the decision margins are substantially wider, improving robustness. Networks of this size remain within sub-microwatt consumption estimates (Table 4).

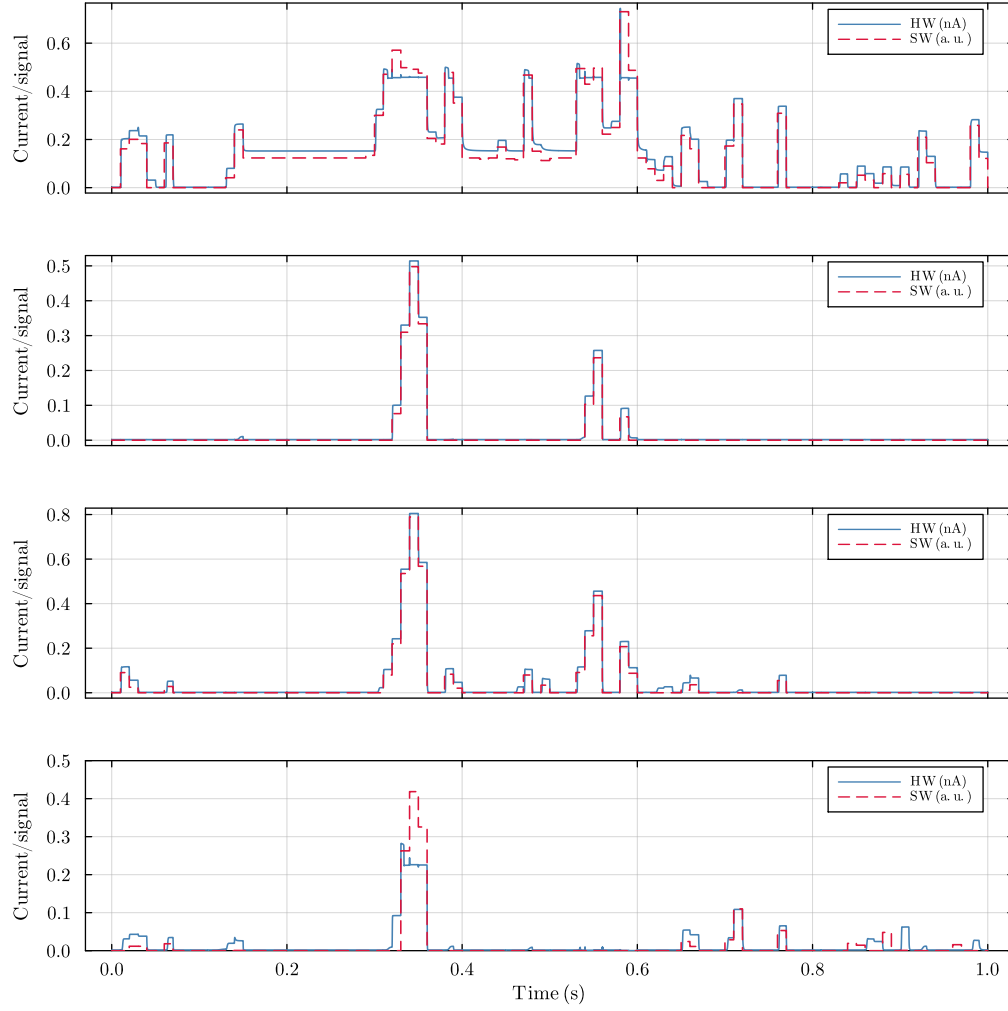


Figure 38: **Intermediate signal comparison between software and hardware (layer-1 candidates, seed 51).** Overlay of the 4 software-predicted and Cadence-simulated candidate currents of the first recurrent layer for a representative “yes” inference sample.

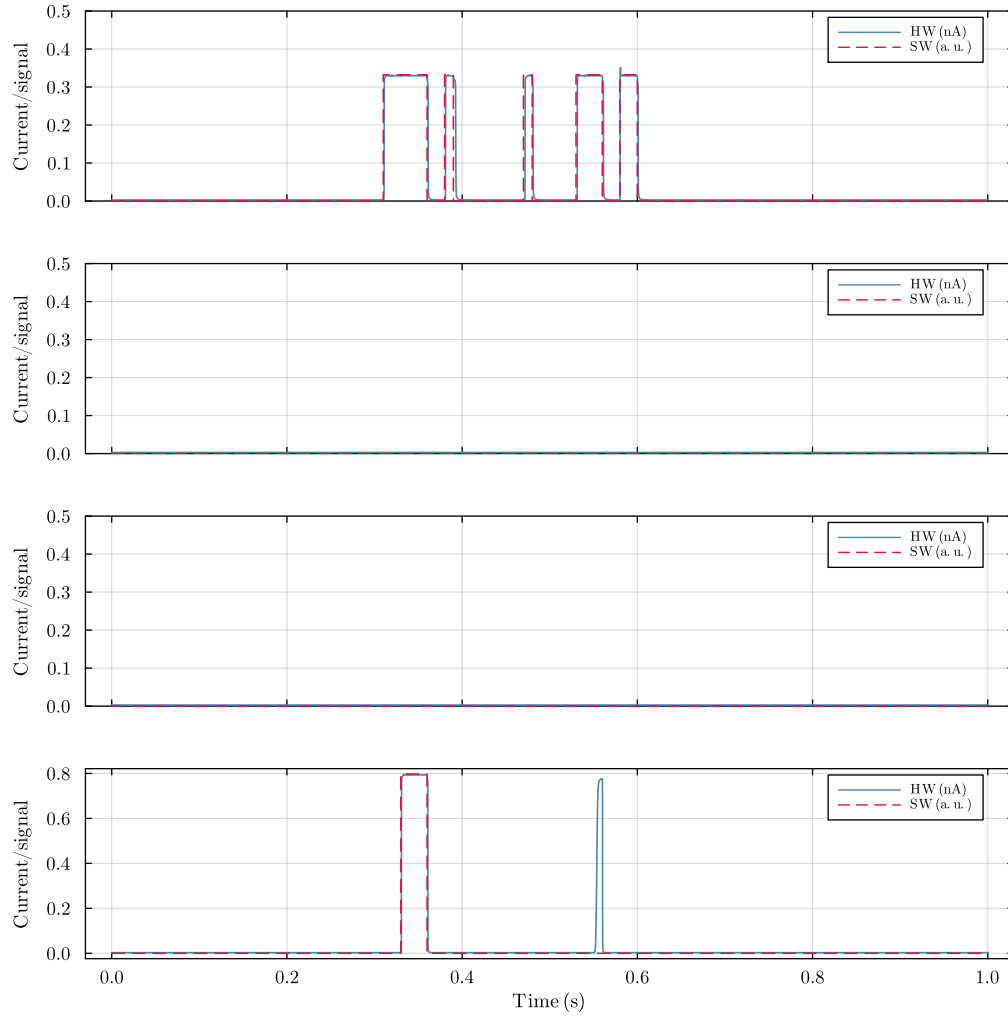


Figure 39: **Intermediate signal comparison between software and hardware (layer-1 states, seed 51).** Overlay of the 4 software-predicted and Cadence-simulated FQ BMRU cell outputs of the first recurrent layer for a representative “yes” inference sample.

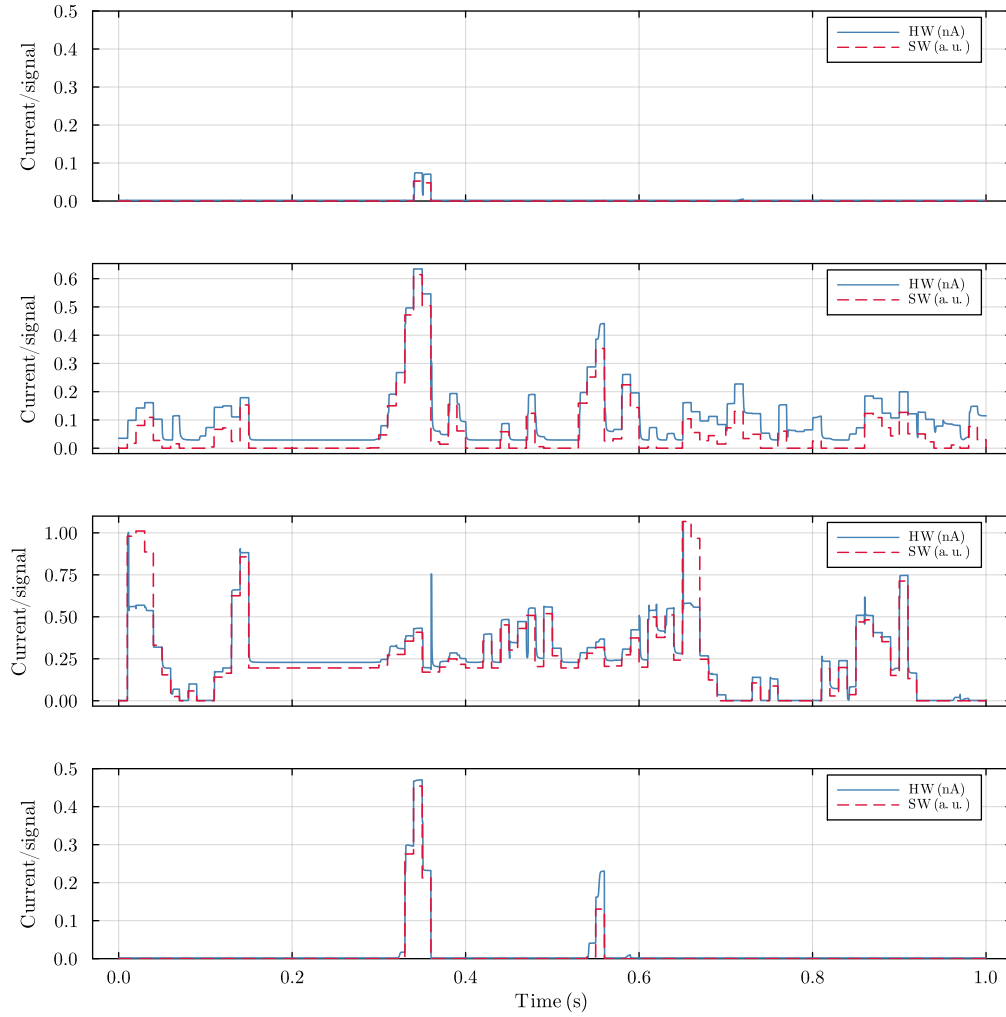


Figure 40: **Intermediate signal comparison between software and hardware (layer-2 candidates, seed 51).** Overlay of the 4 software-predicted and Cadence-simulated candidate currents of the second recurrent layer for a representative “yes” inference sample.

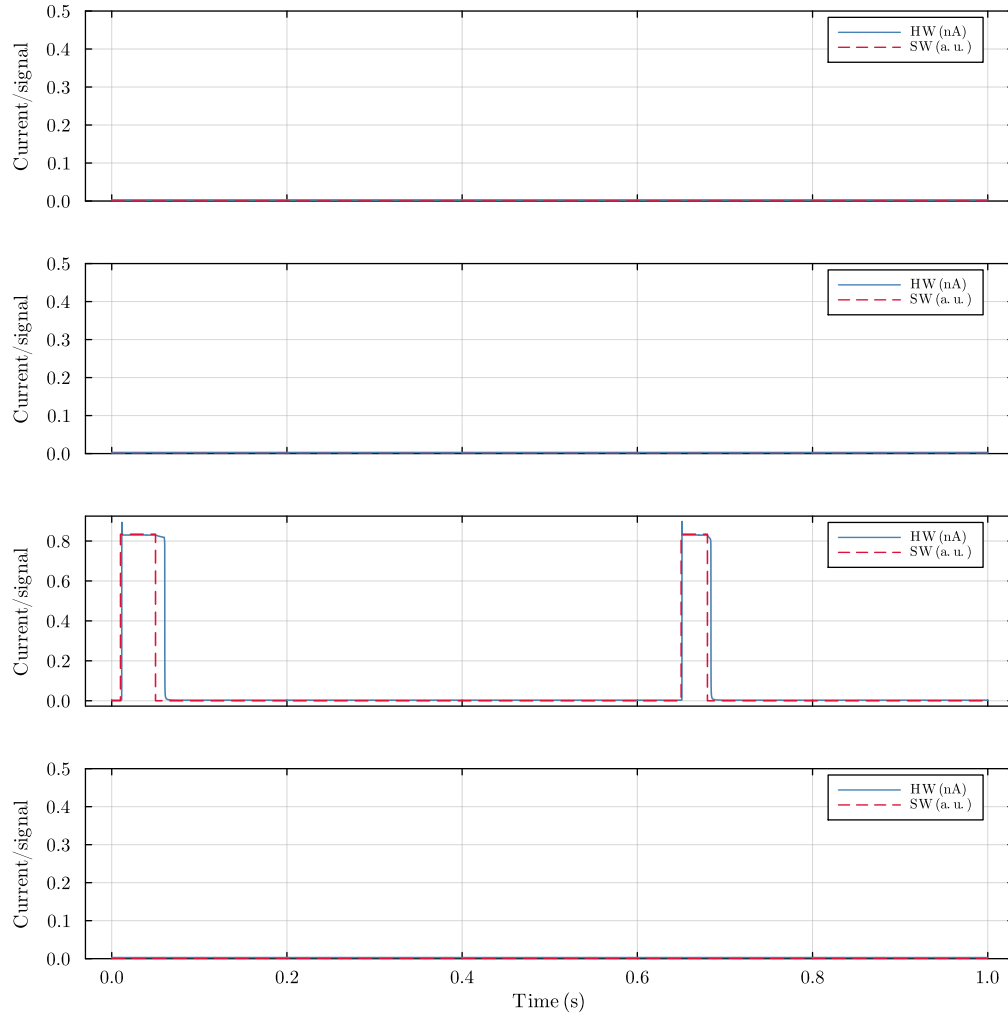


Figure 41: **Intermediate signal comparison between software and hardware (layer-2 states, seed 51).** Overlay of the 4 software-predicted and Cadence-simulated FQ BMRU cell outputs of the second recurrent layer for a representative “yes” inference sample.

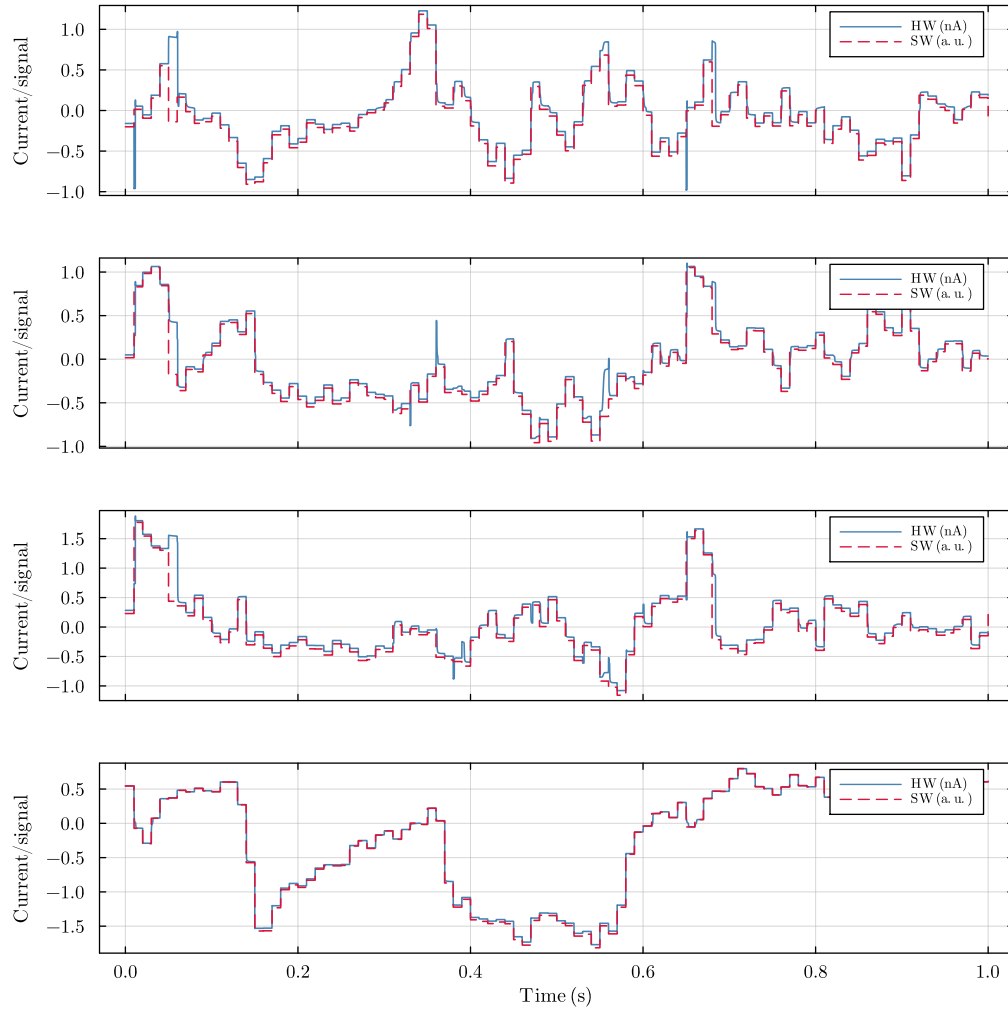


Figure 42: **Intermediate signal comparison between software and hardware (layer-2 output after skip connection, seed 51).** Overlay of the 4 software-predicted and Cadence-simulated output signals of the second recurrent layer, after the skip connection, for a representative “yes” inference sample.

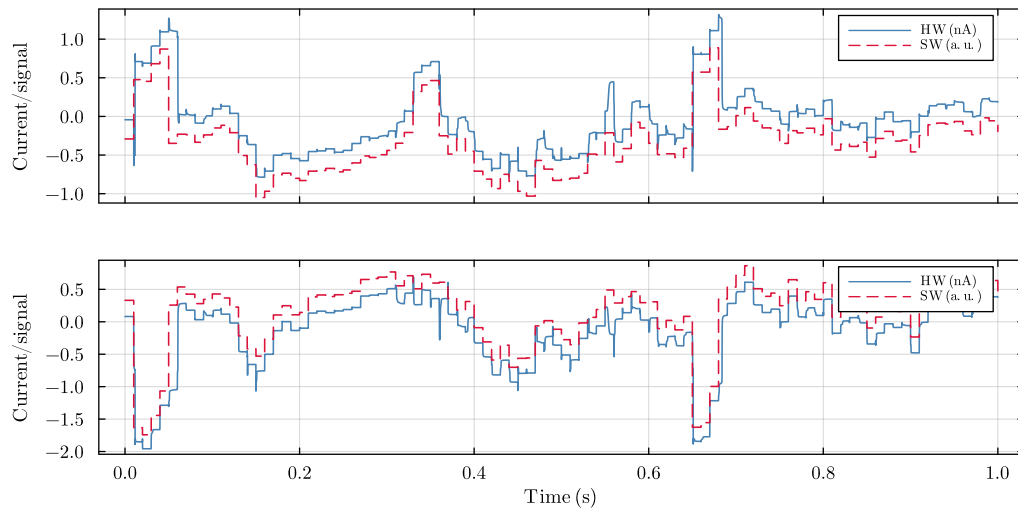


Figure 43: **Intermediate signal comparison between software and hardware (output logits, seed 51).** Overlay of the software-predicted and Cadence-simulated output logit currents for a representative “yes” inference sample.

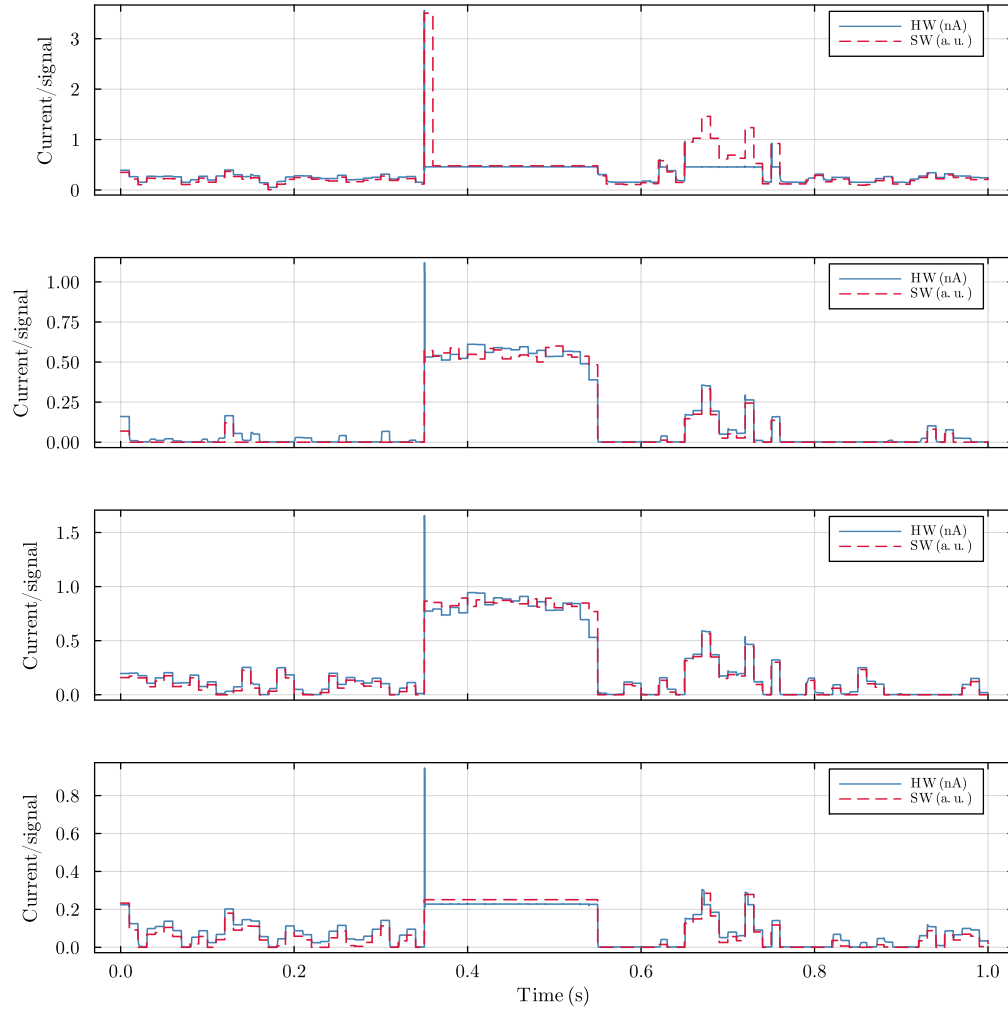


Figure 44: **Intermediate signal comparison between software and hardware (layer-1 candidates, seed 66).** Overlay of the 4 software-predicted and Cadence-simulated candidate currents of the first recurrent layer for a representative “background” inference sample.

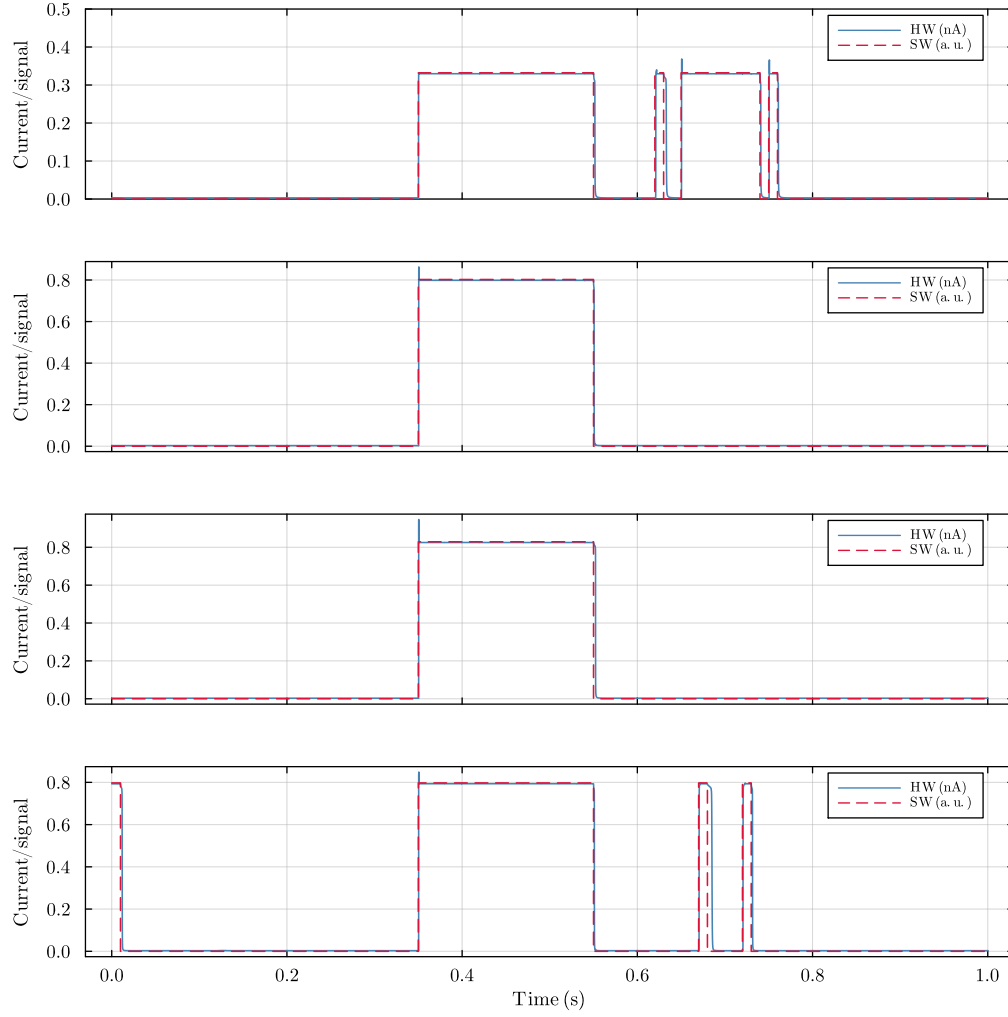


Figure 45: **Intermediate signal comparison between software and hardware (layer-1 states, seed 66).** Overlay of the 4 software-predicted and Cadence-simulated FQ BMRU cell outputs of the first recurrent layer for a representative “background” inference sample.

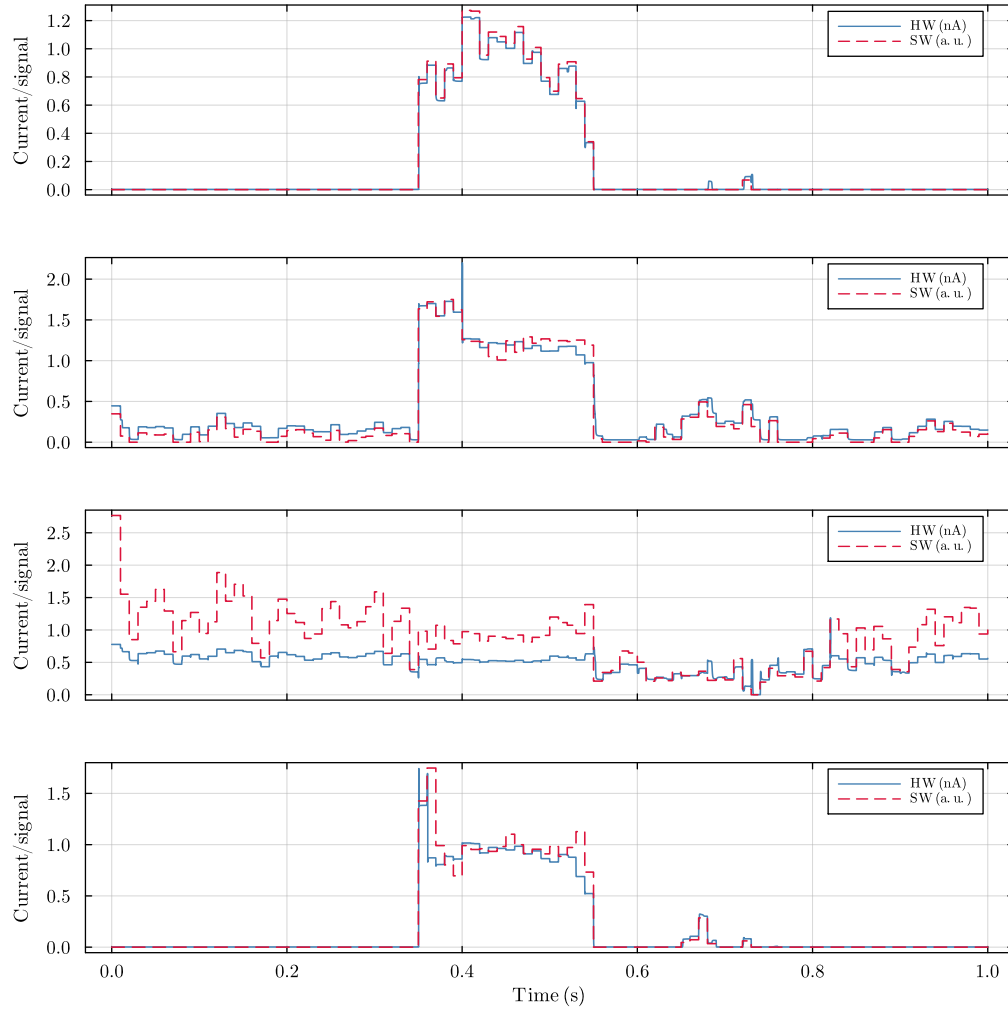


Figure 46: **Intermediate signal comparison between software and hardware (layer-2 candidates, seed 66).** Overlay of the 4 software-predicted and Cadence-simulated candidate currents of the second recurrent layer for a representative “background” inference sample.

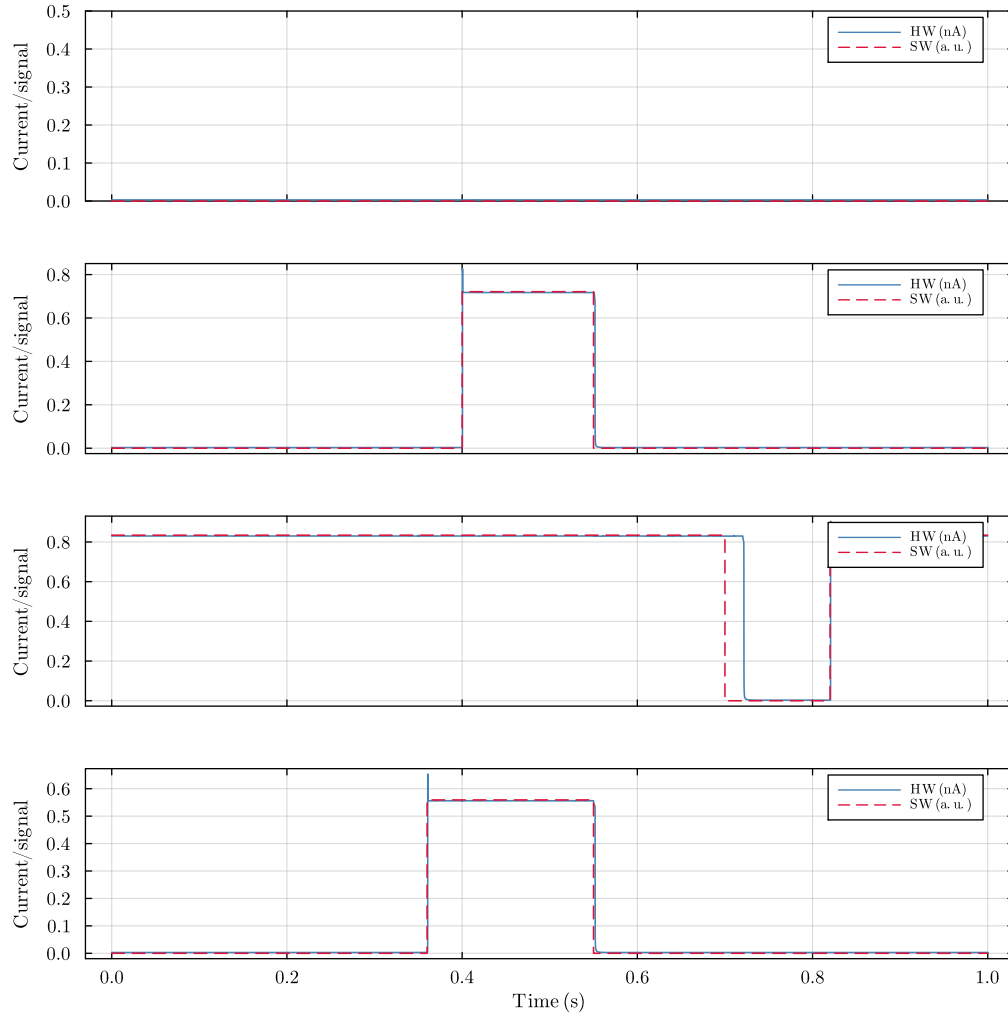


Figure 47: **Intermediate signal comparison between software and hardware (layer-2 states, seed 66).** Overlay of the 4 software-predicted and Cadence-simulated FQ BMRU cell outputs of the second recurrent layer for a representative “background” inference sample.

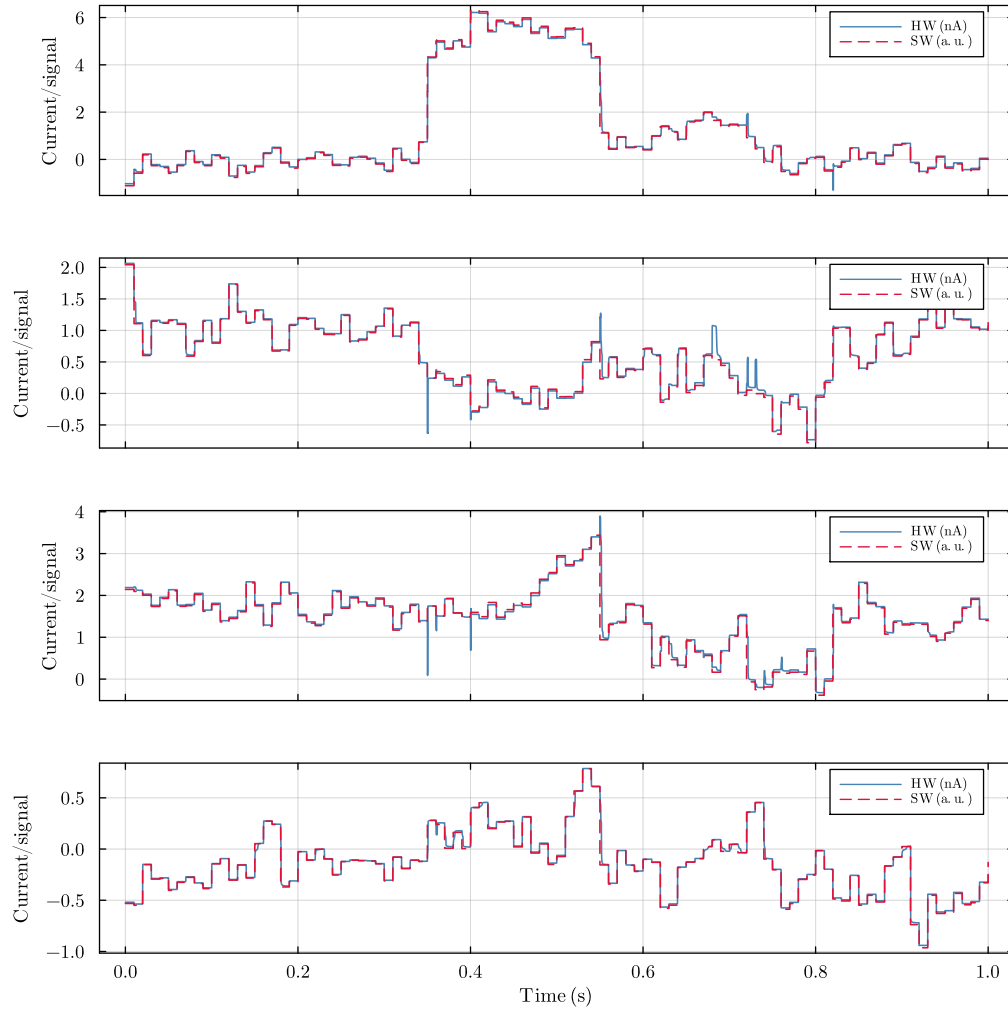


Figure 48: **Intermediate signal comparison between software and hardware (layer-2 output after skip connection, seed 66).** Overlay of the 4 software-predicted and Cadence-simulated output signals of the second recurrent layer, after the skip connection, for a representative “background” inference sample.

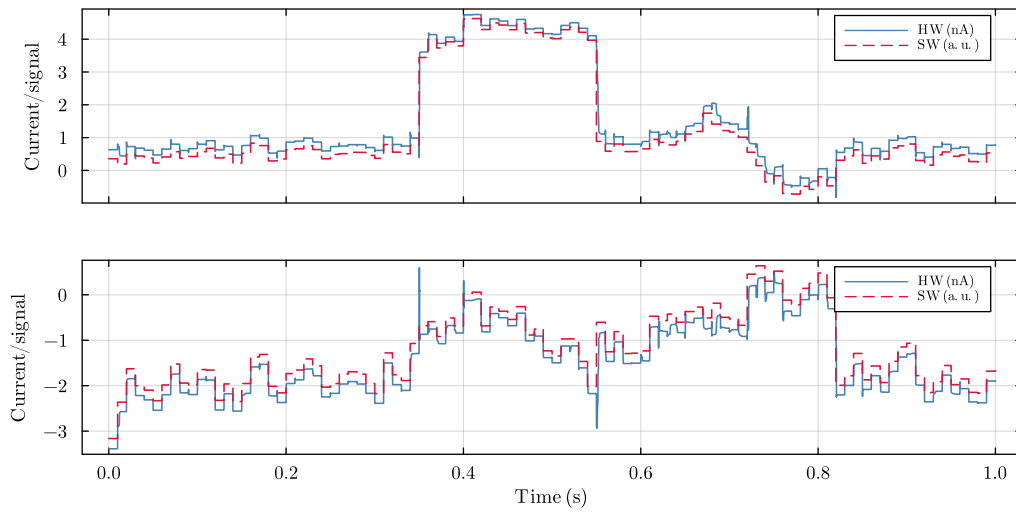


Figure 49: **Intermediate signal comparison between software and hardware (output logits, seed 66).** Overlay of the software-predicted and Cadence-simulated output logit currents for a representative “background” inference sample.