DENSE ASSOCIATIVE MEMORIES WITH ANALOG CIRCUITS

Marc Gong Bacvanski
MIT
marcbac@mit.edu

Xincheng You
Independent Researcher
xinchengyou6@gmail.com

Dmitry Krotov
IBM Research
krotov.a.dmitry@gmail.com

ABSTRACT

Associative memory models perform inference by converging to fixed points of a dynamical system, making them naturally compatible with analog hardware that computes through physical time evolution. We present a mapping from Dense Associative Memory (DenseAM) dynamics to an analog circuit primitive composed of resistive crossbar arrays and continuous-time neuron integrators. Synaptic weights are implemented as conductances, neuron states are capacitor voltages, and circuit physics directly realize the DenseAM update equations, so that inference corresponds to physical relaxation of the circuit. We illustrate the primitive with a DenseAM that solves the XOR task with attractor dynamics, and show how DenseAM blocks can be composed into an Analog Energy Transformer. On an 8-bit parity task, the Analog Energy Transformer achieves perfect validation accuracy under explicit ODE simulation, and its continuous-time trajectories converge to stable fixed points, yielding robustness to readout timing. These results position DenseAM as a natural computational abstraction for analog AI hardware, unifying associative recall and transformer-like architectures within a single dynamical circuit framework.

1 INTRODUCTION

Associative memory models store patterns in distributed form and retrieve them by converging to fixed points of a dynamical system. Recent advances in analog and mixed-signal AI accelerators shift computation from clocked digital updates to continuous physical dynamics. These two developments align naturally: inference in associative memory is itself a dynamical process, making it directly compatible with continuous-time hardware.

Dense Associative Memory (DenseAM) Krotov & Hopfield (2021); Krotov (2021) is a modern formulation of associative memory with broad expressibility and simple continuous-time neuron dynamics. DenseAM defines two coupled populations of neurons (visible and hidden) whose states evolve by first-order nonlinear differential equations with shared synaptic weights. This structure is both mathematically convenient and structurally compatible with canonical analog primitives such as capacitors (for state), resistors/conductances (for weights), and Kirchhoff current summation (for weighted accumulation). In other words, DenseAM is a model class where "the algorithm" and "the circuit" of inference can be made identical objects.

Portions of this paper reproduce results and figures from our prior work Bacvanski et al. (2025); here we provide a streamlined and self-contained presentation focused on the analog circuit perspective and its connection to transformer-like energy-based models.

In this paper, we highlight DenseAM as a bridge between modern energy-based associative memory models and practical analog circuit realizations that compute by physical time evolution. We bring together the dynamical system, circuit, and architectural perspectives:

- Dynamical formulation of DenseAM (Section 2): We review the continuous-time ODE formulation of DenseAM, emphasizing its interpretation as a coupled visible–hidden dynamical system suitable for physical implementation.
- Analog circuit realization of DenseAM (Section 3): We summarize a hardware construction in which a resistive crossbar array encodes the shared weight matrix and neuron circuits integrate crossbar currents to reproduce the DenseAM dynamics directly in continuous time.
- Extension to transformer-like energy-based models (Section 5): We show how the same primitives compose into an Analog Energy Transformer (Analog ET) Hoover et al. (2024), formed from coupled DenseAM blocks whose inference corresponds to physical relaxation of shared token states.

2 DENSE ASSOCIATIVE MEMORY BASICS

The DenseAM framework Krotov & Hopfield (2021); Krotov (2021) provides a model that has straightforward neuronal dynamics, yet is surprisingly expressive in its ability to represent AI models including transformer attention, diffusion models, and associative memories. In its simplest form it is defined by two sets of neurons (typically called visible and hidden neurons) and a system of coupled non-linear differential equations governing

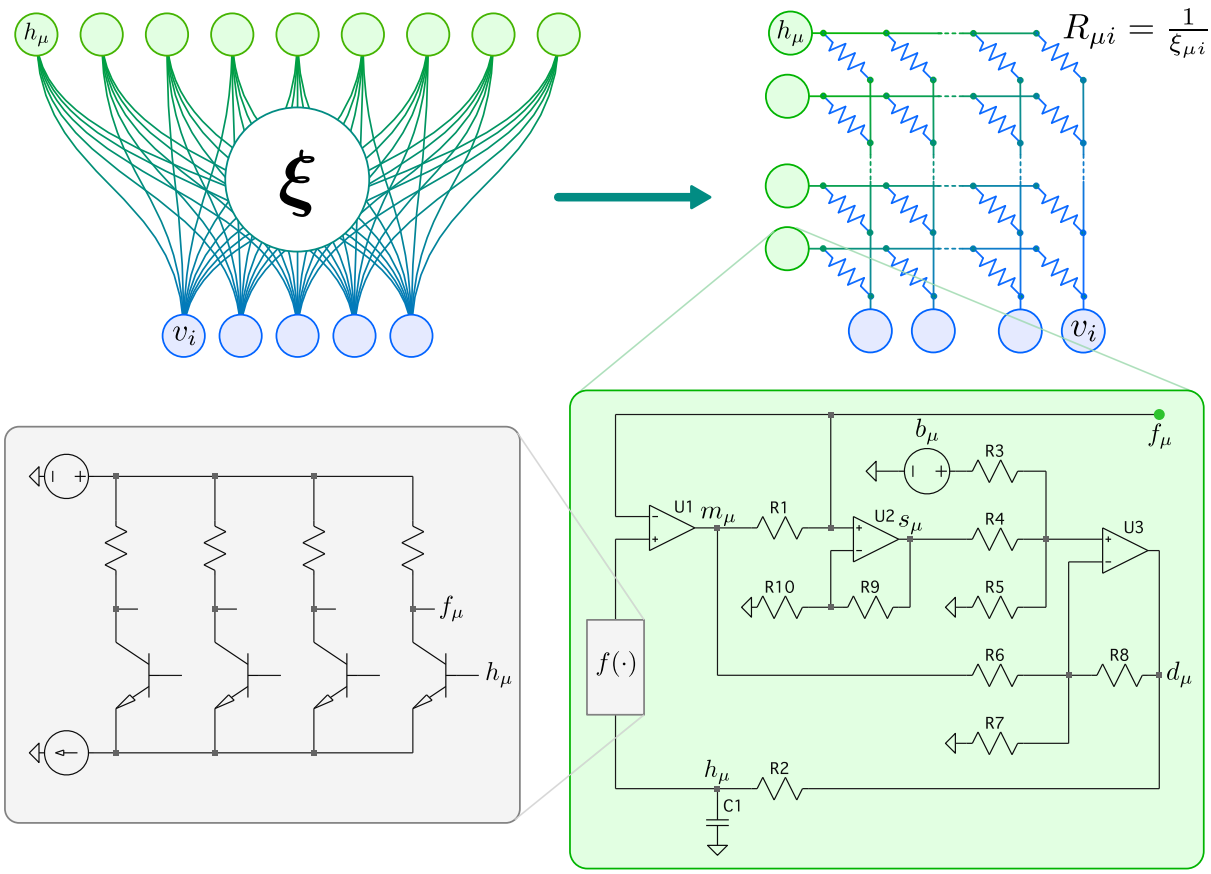

Figure 1: **Top left:** Bipartite neural network formulation, where hidden neurons h_μ and visible neurons v_i are connected via symmetric synaptic weights $\boldsymbol{\xi}$. **Top right:** Circuit realization of symmetric weight matrix via resistive crossbar array. Each crosspoint encodes a weight $\xi_{\mu i}$ by its resistance $R_{\mu i} = 1/\xi_{\mu i}$. **Lower right:** Circuit schematic of a single hidden neuron. It drives its row of the crossbar array with a voltage according to its activation f_μ, and its internal dynamics are driven by the incoming current flowing into it from the crossbar array. **Lower left:** Softmax activation function built from bipolar junction transistors (some components not shown).

their behavior, see Figure 1. The visible neurons are characterized by their internal states v_i and their outputs g_i, index $i = 1 \dots N_v$; while the hidden neurons have internal states h_μ and outputs f_μ, index $\mu = 1 \dots N_h$. This framework admits both neuron-wise activation functions ($g_i = g(v_i)$, where $g(\cdot)$ is some continuous function, e.g., a ReLU), and collective activation functions such as softmax or layer normalization, which depend on the states of multiple neurons.

The network parameters are stored in the synaptic weights $\boldsymbol{\xi} \in \mathbb{R}^{N_h \times N_v}$, whose matrix elements denoted by $\xi_{\mu i}$ can be either hand-engineered or learned. The time decay constants for the two groups of neurons are τ_v and τ_h. With these conventions, the temporal evolution of the two groups of neurons can be expressed as

$$\begin{cases} \tau_v \dfrac{dv_i}{dt} = \displaystyle\sum_{\mu=1}^{N_h} \xi_{\mu i} f_\mu + a_i - v_i \\ \tau_h \dfrac{dh_\mu}{dt} = \displaystyle\sum_{i=1}^{N_v} \xi_{\mu i} g_i + b_\mu - h_\mu \end{cases} \tag{1}$$

This forms a bipartite graph of neuronal connections, where the state of the hidden neurons is updated by the state of the visible neurons, and vice versa. Importantly, the same weight matrix $\boldsymbol{\xi}$ appears in both equations, once as $\boldsymbol{\xi}$ and again as $\boldsymbol{\xi}^\top$. Finally, a_i and b_μ denote biases, which are additional weights of the system. These dynamics minimize an explicit energy function, ensuring convergence to fixed points — a property we exploit for hardware realization.

3 Analog Circuits for DenseAM

Now we translate the DenseAM dynamics of equation 1 into a physical circuit. Neuron internal states v_i and h_μ are capacitor voltages; synaptic weights $\boldsymbol{\xi}$ are conductances in a resistive crossbar; and interactions arise

from currents through these conductances via Ohm's law and Kirchhoff's current law. The bipartite DenseAM graph maps directly onto a crossbar: visible and hidden neurons occupy orthogonal wire sets, and each crosspoint conductance $\xi_{\mu i}$ connects one visible activation voltage g_i to one hidden activation voltage f_μ. Hence each element contributes a current proportional to the voltage difference, yielding terms $\xi_{\mu i}(g_i - f_\mu)$, and current summation along each wire implements the required matrix–vector accumulations.

Each neuron senses its total incoming crossbar current and integrates it with an RC circuit: the capacitor voltage is the neuron state and the RC leak produces the linear decay. A nonlinear block (e.g., ReLU or softmax) generates the activation (g_i or f_μ) from the state voltage and feeds it back into the crossbar. With these pieces, the circuit's continuous-time voltages reproduce the DenseAM ODEs directly from circuit laws.

Resistive weights as a crossbar array. A resistive crossbar array provides a direct physical implementation of the weight matrix $\boldsymbol{\xi}$ (Hopfield, 1990; Mead & Ismail, 2012). The same physical conductance is used bidirectionally: the net current into hidden neuron μ is $\sum_i \xi_{\mu i}(g_i - f_\mu)$ and into visible neuron i is $\sum_\mu \xi_{\mu i}(f_\mu - g_i)$. Thus the DenseAM bipartite coupling is realized by Kirchhoff current summation, and symmetry is enforced by construction. Representing weights as conductances also implies $\xi_{\mu i} \geq 0$.

Single-neuron circuit. Each neuron integrates its incoming current with an RC stage. For hidden neuron μ, the internal state h_μ is the voltage on capacitor C_1 with leak resistor R_2. The crossbar supplies

$$I_\mu = \sum_i \xi_{\mu i}(g_i - f_\mu).$$

To recover the canonical DenseAM drive, the self-term $-f_\mu \sum_i \xi_{\mu i}$ is canceled locally by generating $s_\mu = f_\mu \sum_i \xi_{\mu i}$ and subtracting it, leaving the effective input $\sum_i \xi_{\mu i} g_i + b_\mu$. The resulting state dynamics are therefore

$$R_2 C_1 \frac{dh_\mu}{dt} = \sum_{i=1}^{N_v} \xi_{\mu i} g_i + b_\mu - h_\mu,$$

and a nonlinear block produces $f_\mu = f(h_\mu)$. Visible neurons are implemented symmetrically.

4 XOR Model

XOR is a canonical test of nonlinear representation: it cannot be solved by any linear model. We implement XOR as a minimal DenseAM with $N_v = 3$ visible neurons and $N_h = 4$ hidden neurons, where each row of the memory matrix $\boldsymbol{\xi} \in \mathbb{R}^{4\times 3}$ encodes one row of the XOR truth table. Visible units use the identity activation $g_i = v_i$ and hidden units use a softmax activation. During inference, the two input visible neurons (v_1, v_2) are clamped to the input bits, the output visible neuron is initialized at $v_3(0) = 0.5$, and the hidden states are initialized near zero; the free neurons then evolve under equation 1.

To make the softmax operate on (negative) squared distances to the stored patterns, we set visible biases $a_i = 0$ and choose hidden biases $b_\mu = -\frac{1}{2}\|\boldsymbol{\xi}_\mu\|^2$, where $\boldsymbol{\xi}_\mu$ denotes the μ-th stored pattern. In the adiabatic regime $\tau_h \ll \tau_v$, the hidden variables can be integrated out, yielding a visible-only effective energy whose minima implement the XOR truth table (derivation in Appendix A). Figure 2 shows a representative inference trajectory.

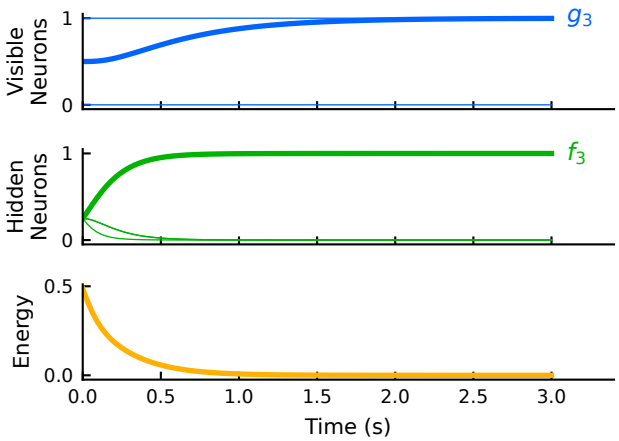

Figure 2: Solving XOR with a DenseAM. Visible neuron $g_3 = v_3$ serves as the output, while the two input neurons (unlabeled, thin lines) are clamped at 1 and 0 for True and False. Output v_3 is initialized at 0.5 and converges to a positive prediction of 1. The activation of the hidden neuron f_3 for the truth-table row (1, 0, 1) becomes highly activated, while others (fine lines) are suppressed by softmax. Energy decreases monotonically along the inference trajectory.

5 Analog Energy Transformer

DenseAM circuits can be composed into larger energy-based architectures by coupling multiple hidden populations to a shared visible state. As a concrete example, we implement a transformer-like *Energy Transformer* (ET) block (Hoover et al., 2024) in analog hardware by instantiating two DenseAM blocks that share the same visible

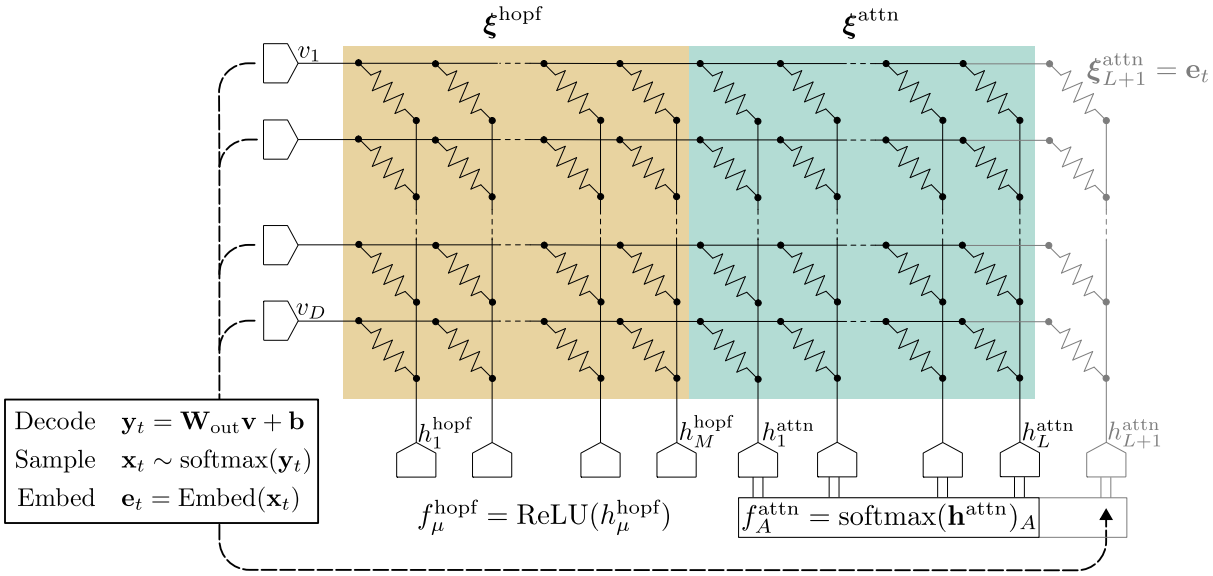

Figure 3: Analog ET circuit demonstrating the autoregressive inference procedure. A newly inferred token is decoded, sampled, and re-embedded to obtain the weight vector $\boldsymbol{\xi}^{\text{attn}}_{L+1}$, which is programmed as the weights of a new attention hidden neuron h^{attn}_{L+1} (light gray). For this layout we have flipped the crossbar array, so that indices A and μ run horizontally and index i runs vertically.

embedding state $\mathbf{v} \in \mathbb{R}^D$: (i) an *energy-attention* block with softmax hidden units that routes information from the context, and (ii) a *Hopfield* block with ReLU hidden units that shapes $\mathbf{v}$ toward a learned embedding manifold. The key circuit-level distinction is that the attention weights are *dynamic* (set by the context at inference), while the Hopfield weights are *static* (learned once and held fixed).

Hardware composition in a single crossbar. The shared visible state $\mathbf{v}$ is represented by D neurons, as in the DenseAM primitive. Both hidden populations connect to the same visible neurons, so the total current into each visible neuron is the sum of (i) attention currents and (ii) Hopfield currents. Thus a single larger resistive crossbar can implement dynamics of the form $(\textit{visible drive}) \propto (\boldsymbol{\xi}^{\text{attn}})^\top \mathbf{f}^{\text{attn}} + (\boldsymbol{\xi}^{\text{hopf}})^\top \mathbf{f}^{\text{hopf}}$, with separate hidden-neuron circuits realizing the softmax and ReLU nonlinearities.

Analog ET dynamics. For a single-head, causal next-token prediction setting with context length L and M Hopfield memories, the Analog ET uses $\mathbf{h}^{\text{attn}} \in \mathbb{R}^L$ and $\mathbf{h}^{\text{hopf}} \in \mathbb{R}^M$ with activations

$$f^{\text{attn}}_A = \text{softmax}(\beta \mathbf{h}^{\text{attn}})_A, \qquad A = 1, \ldots, L, \tag{2}$$

$$f^{\text{hopf}}_\mu = \text{ReLU}(h^{\text{hopf}}_\mu), \qquad \mu = 1, \ldots, M. \tag{3}$$

The coupled continuous-time inference dynamics are

$$\tau_v \dot{\mathbf{v}} = (\boldsymbol{\xi}^{\text{attn}})^\top \mathbf{f}^{\text{attn}} + (\boldsymbol{\xi}^{\text{hopf}})^\top \mathbf{f}^{\text{hopf}} + \mathbf{a} - \mathbf{v}, \tag{4}$$

$$\tau_h \dot{\mathbf{h}}^{\text{attn}} = \boldsymbol{\xi}^{\text{attn}} \mathbf{v} + \mathbf{b} - \mathbf{h}^{\text{attn}}, \tag{5}$$

$$\tau_h \dot{\mathbf{h}}^{\text{hopf}} = \boldsymbol{\xi}^{\text{hopf}} \mathbf{v} + \mathbf{c} - \mathbf{h}^{\text{hopf}}. \tag{6}$$

where $\mathbf{b}$ and $\mathbf{c}$ are bias vectors for the attention and Hopfield hidden neurons, respectively. These dynamics are a gradient flow of an explicit energy function over $(\mathbf{v}, \mathbf{h}^{\text{attn}}, \mathbf{h}^{\text{hopf}})$; in the adiabatic regime $\tau_h \ll \tau_v$ one can integrate out the hidden states to recover the canonical ET visible-only energy (Hoover et al., 2024) (derivation in Appendix B).

Dynamic attention weights from context. The attention matrix $\boldsymbol{\xi}^{\text{attn}} \in \mathbb{R}^{L \times D}$ is instantiated at inference by embedding each context token and using its embedding as one row: $\boldsymbol{\xi}^{\text{attn}}_A = \text{Embed}(\text{token}_A)$. In contrast, $\boldsymbol{\xi}^{\text{hopf}} \in \mathbb{R}^{M \times D}$ is learned during training and fixed at inference. Hence the context acts as a set of *on-the-fly memories* that are written into the analog substrate by programming the corresponding crossbar conductances.

Autoregressive rollout as physical relaxation + row update. Figure 3 illustrates the outer-loop controller for autoregressive generation. Given a length-L context, the controller programs $\boldsymbol{\xi}^{\text{attn}}$ from the embedded context tokens, then lets the continuous-time dynamics equation 4–equation 6 relax to a fixed point. A decoder reads out the converged visible state $\mathbf{v}(T)$ to produce logits for the next token, from which a token is sampled. That token is embedded and inserted into the attention block by programming the weights of one attention hidden neuron (equivalently, rewriting one row of $\boldsymbol{\xi}^{\text{attn}}$ in a sliding-window implementation).

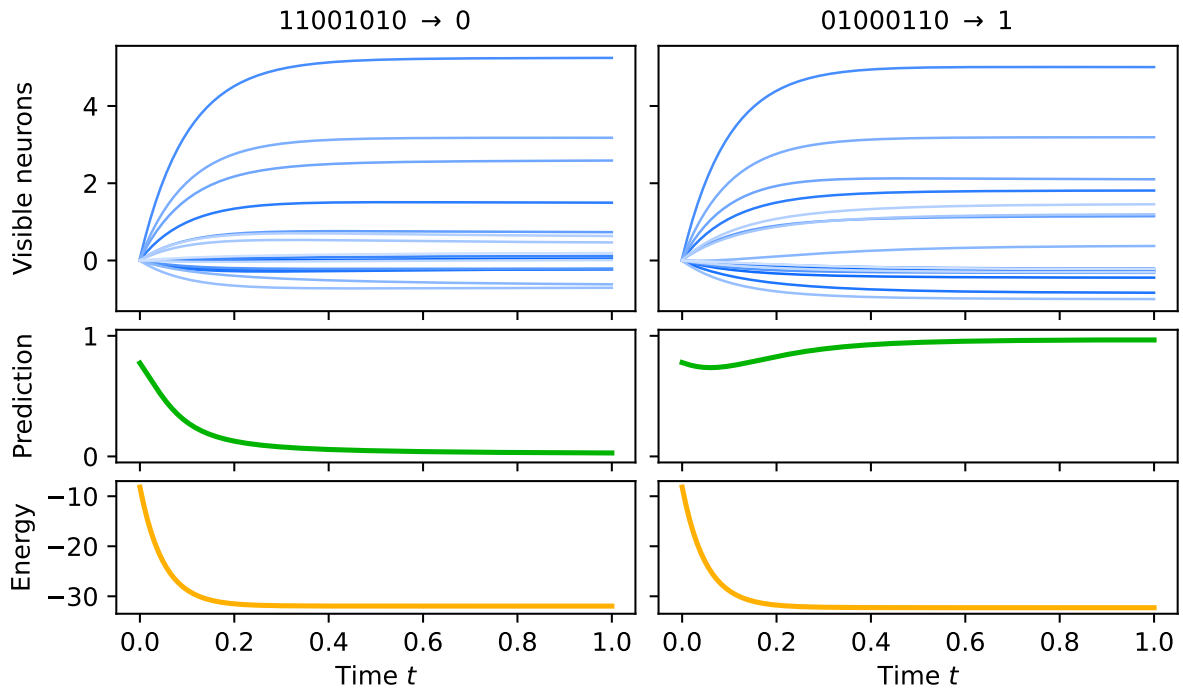

Figure 4: Inference of parity Analog ET on two example 8-bit strings. Top row plots the visible neurons v_i over time, middle row plots the decoded token prediction, bottom row plots the energy that monotonically decreases during inference. After a transient period of computation, the network arrives at a steady-state, making the result of the computation robust against the precise timing of the readout.

5.1 Analog Energy Transformer on the parity task

As a minimal demonstration of expressibility and dynamical stability, we evaluate the Analog ET on the L-bit parity task. Given context bits $\text{bit}_1, \ldots, \text{bit}_L \in \{0, 1\}$, the goal is to predict $\text{bit}_{L+1} = \left(\sum_{A=1}^{L} \text{bit}_A\right) \bmod 2$. Parity requires a global, order-L interaction over the input bits and cannot be represented by linear or shallow models. Thus, successful performance indicates that the Analog ET forms high-order interactions through the coupled attention and Hopfield dynamics. In this task, the L context bits instantiate the attention weights $\boldsymbol{\xi}^{\text{attn}}$ via their embeddings, after which the continuous-time dynamics equation 4–equation 6 evolve until convergence. A linear decoder applied to the converged visible state $\mathbf{v}(T)$ produces logits for the parity bit.

Training is performed digitally using backpropagation through time (Werbos, 2002) implemented in JAX. The learned parameters (embedding matrix, Hopfield weights, decoder, and biases) are then evaluated under explicit ODE simulation using Diffrax (Kidger, 2021), which emulates the analog continuous-time dynamics. On the 8-bit parity task, the model achieves 100% accuracy on a hold-out validation set of 52 strings. Model dimensions and training hyperparameters are provided in Appendix C.

Figure 4 shows representative inference trajectories. The visible state $\mathbf{v}(t)$ evolves smoothly from its initialization to a fixed point, while the energy decreases monotonically. After a transient period, $\mathbf{v}(t)$ becomes effectively constant, indicating convergence to an attractor. This makes the decoded output insensitive to moderate timing variation in readout, since any sampling time within the steady-state regime yields the same prediction.

6 Conclusion

In this paper, we presented an analog electronic accelerator for Dense Associative Memories implemented with resistive crossbar arrays and continuous-time neuron dynamics. Rather than performing discrete numerical updates, inference is realized as the time evolution of a physical dynamical system. DenseAM circuits act as a computational primitive that can be composed into more expressive architectures, as demonstrated by the Analog Energy Transformer built from coupled DenseAMs.

These results position DenseAMs as a natural computational paradigm for analog AI hardware. They unify modern primitives such as attention and transformer-like architectures within a dynamical framework that provides error-correcting behavior and asymptotic stability, properties that are expected to provide robustness under hardware imperfections. Together, these features suggest that DenseAM-based accelerators are a compelling substrate for future AI systems and motivate deeper co-design across models, dynamics, and devices.

ACKNOWLEDGMENTS

A more comprehensive version of this paper is available at: https://arxiv.org/abs/2512.15002. The results presented here were obtained while Dmitry Krotov was employed by IBM Research. At the time of submission Dmitry Krotov is no longer employed by IBM Research.

REFERENCES

Marc Gong Bacvanski, Xincheng You, John Hopfield, and Dmitry Krotov. Dense associative memories with analog circuits. *arXiv preprint arXiv:2512.15002*, 2025.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

JJ Hopfield. The effectiveness of analogue'neural network'hardware. *Network: Computation in Neural Systems*, 1(1):27, 1990.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Dmitry Krotov and John J Hopfield. Large associative memory problem in neurobiology and machine learning. In *International Conference on Learning Representations*, 2021.

Carver Mead and Mohammed Ismail. *Analog VLSI implementation of neural systems*, volume 80. Springer Science & Business Media, 2012.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10): 1550–1560, 2002.

A XOR MODEL

Table 1 summarizes the model design for the XOR DenseAM model.

Table 1: XOR model specification

Visible neurons v_i	$N_v = 3$ (inputs v_1, v_2 clamped to {0,1}; output v_3 free)
Hidden neurons h_μ	$N_h = 4$ (one per truth-table row)
Visible activation and Lagrangian	Identity: $g_i = v_i,\ \mathcal{L}_v = \frac{1}{2}\sum_{i=1}^{N_v} v_i^2$
Hidden activation and Lagrangian	Softmax: $f_\mu = \mathrm{softmax}(\beta \mathbf{h})_\mu,\ \mathcal{L}_h = \frac{1}{\beta}\log\Big(\sum_{\mu=1}^{N_h} e^{\beta h_\mu}\Big)$
Visible biases	$a_i = 0$
Hidden biases	$b_\mu = -\frac{1}{2}\sum_{i=1}^{N_v} \xi_{\mu i}^2$
Weights $\boldsymbol{\xi}$	$\boldsymbol{\xi} \in \{0,1\}^{4\times 3}$, rows encode memories: $\boldsymbol{\xi} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$
Inference protocol	Clamp (v_1, v_2) to input values; read out v_3 at convergence

In the adiabatic limit, the hidden preactivations satisfy

$$h_\mu = \boldsymbol{\xi}_\mu^\top \mathbf{v} + b_\mu,$$

where $\boldsymbol{\xi}_\mu$ denotes the μ-th stored pattern.

To ensure that valid patterns correspond to attractors of the visible dynamics, the biases are chosen as

$$b_\mu = -\tfrac{1}{2}\|\boldsymbol{\xi}_\mu\|^2.$$

With this choice,

$$\boldsymbol{\xi}_\mu^\top \mathbf{v} + b_\mu = \boldsymbol{\xi}_\mu^\top \mathbf{v} - \tfrac{1}{2}\|\boldsymbol{\xi}_\mu\|^2.$$

Using the quadratic identity

$$\|\mathbf{v} - \boldsymbol{\xi}_\mu\|^2 = \|\mathbf{v}\|^2 + \|\boldsymbol{\xi}_\mu\|^2 - 2\boldsymbol{\xi}_\mu^\top \mathbf{v},$$

we obtain

$$\boldsymbol{\xi}_\mu^\top \mathbf{v} - \tfrac{1}{2}\|\boldsymbol{\xi}_\mu\|^2 = -\tfrac{1}{2}\|\mathbf{v} - \boldsymbol{\xi}_\mu\|^2 + \tfrac{1}{2}\|\mathbf{v}\|^2.$$

The second term is independent of μ and therefore does not affect the softmax over μ. Thus, the hidden logits are proportional to negative squared distances between the visible state and the stored patterns:

$$h_\mu = -\tfrac{1}{2}\|\mathbf{v} - \boldsymbol{\xi}_\mu\|^2 + \text{const}.$$

After integrating out the hidden neurons, the visible energy becomes

$$E(\mathbf{v}) = \frac{1}{2}\|\mathbf{v}\|^2 - \frac{1}{\beta}\log\sum_\mu \exp\Big(-\tfrac{\beta}{2}\|\mathbf{v} - \boldsymbol{\xi}_\mu\|^2\Big).$$

In the large-β regime, this energy approximates the minimum squared distance to the stored memories:

$$E(\mathbf{v}) \approx \min_\mu \frac{1}{2}\|\mathbf{v} - \boldsymbol{\xi}_\mu\|^2.$$

Therefore, each valid memory forms a local minimum of the energy landscape, and in the XOR case, gradient flow drives v_3 toward the output value associated with the closest stored truth-table row.

B CONNECTION BETWEEN ANALOG AND CANONICAL ENERGY TRANSFORMER

In this section we show that in the adiabatic limit, the Analog Energy Transformer (Analog ET) reduces to the canonical Energy Transformer. Begin with the dynamics for the Analog Energy Transformer implemented by our circuit designs.

$$\tau_v \dot{\mathbf{v}} = -\frac{\partial E}{\partial \mathbf{v}} = \left(\boldsymbol{\xi}^{\text{attn}}\right)^\top \mathbf{f}^{\text{attn}} + \left(\boldsymbol{\xi}^{\text{hopf}}\right)^\top \mathbf{f}^{\text{hopf}} + \mathbf{a} - \mathbf{v} \tag{7}$$

$$\tau_h \dot{\mathbf{h}}^{\text{attn}} = -\frac{\partial E}{\partial \mathbf{f}^{\text{attn}}} = \boldsymbol{\xi}^{\text{attn}} \mathbf{v} + \mathbf{b} - \mathbf{h}^{\text{attn}} \tag{8}$$

$$\tau_h \dot{\mathbf{h}}^{\text{hopf}} = -\frac{\partial E}{\partial \mathbf{f}^{\text{hopf}}} = \boldsymbol{\xi}^{\text{hopf}} \mathbf{v} + \mathbf{c} - \mathbf{h}^{\text{hopf}} \tag{9}$$

Integrating out hidden neurons in the adiabatic limit where $\tau_h \to 0$, we see the relations

$$\mathbf{h}^{\text{attn}}(\mathbf{v}) = \boldsymbol{\xi}^{\text{attn}}\mathbf{v} + \mathbf{b} \tag{10}$$

$$\mathbf{h}^{\text{hopf}}(\mathbf{v}) = \boldsymbol{\xi}^{\text{hopf}}\mathbf{v} + \mathbf{c} \tag{11}$$

which we can use to integrate out the hidden neuron activations as

$$\mathbf{f}^{\text{attn}}(\mathbf{v}) = \text{softmax}\left(\boldsymbol{\xi}^{\text{attn}}\mathbf{v} + \mathbf{b}\right) \tag{12}$$

$$\mathbf{f}^{\text{hopf}}(\mathbf{v}) = \text{ReLU}\left(\boldsymbol{\xi}^{\text{hopf}}\mathbf{v} + \mathbf{c}\right) \tag{13}$$

Substituting into the visible dynamics:

$$\tau_v \dot{\mathbf{v}} = \left(\boldsymbol{\xi}^{\text{attn}}\right)^\top \mathbf{f}^{\text{attn}}(\mathbf{v}) + \left(\boldsymbol{\xi}^{\text{hopf}}\right)^\top \mathbf{f}^{\text{hopf}}(\mathbf{v}) + \mathbf{a} - \mathbf{v} \tag{14}$$

We now identify a scalar energy $E_{\text{eff}}(\mathbf{v})$ that produces this ODE, such that $\tau_v \dot{\mathbf{v}} = -\frac{\partial E_{\text{eff}}}{\partial \mathbf{v}}$. Equivalently,

$$\nabla_{\mathbf{v}} E_{\text{eff}}(\mathbf{v}) = \mathbf{v} - \mathbf{a} - \left(\boldsymbol{\xi}^{\text{attn}}\right)^\top \mathbf{f}^{\text{attn}}(\mathbf{v}) - \left(\boldsymbol{\xi}^{\text{hopf}}\right)^\top \mathbf{f}^{\text{hopf}}(\mathbf{v}) \tag{15}$$

We can construct $E_{\text{eff}}(\mathbf{v})$ as a sum of three pieces whose gradients match each term $E_{\text{eff}}(\mathbf{v}) = E_{\text{quad}}(\mathbf{v}) + E_{\text{attn}}(\mathbf{v}) + E_{\text{hopf}}(\mathbf{v})$. By inspection we see that $E_{\text{quad}}(\mathbf{v}) = \frac{1}{2}\|\mathbf{v} - \mathbf{a}\|^2$.

Attention term. The energy function

$$E_{\text{attn}}(\mathbf{v}) = -\frac{1}{\beta} \log \sum_A \exp\left(\beta \left(\boldsymbol{\xi}_A^{\text{attn}}\mathbf{v} + b_A\right)\right) \tag{16}$$

satisfies our requirement. We can see that by differentiating with respect to v_i, we get

$$\frac{\partial E_{\text{attn}}}{\partial v_i} = -\sum_A \text{softmax}(\boldsymbol{\xi}^{\text{attn}}\mathbf{v} + \mathbf{b})_A \cdot \xi_{Ai}^{\text{attn}} \tag{17}$$

$$= -\sum_A \xi_{Ai}^{\text{attn}} f_A^{\text{attn}}(\mathbf{v}) \tag{18}$$

which yields our desired dynamics of $\nabla_{\mathbf{v}} E_{\text{attn}}(\mathbf{v}) = -\left(\boldsymbol{\xi}^{\text{attn}}\right)^\top \mathbf{f}^{\text{attn}}(\mathbf{v})$.

Hopfield term. A simple way to achieve the desired dynamics is with a Hopfield-type energy function

$$E_{\text{hopf}}(\mathbf{v}) = -\sum_\mu \frac{1}{2}\left(\text{ReLU}\left(\boldsymbol{\xi}_\mu^{\text{hopf}}\mathbf{v} + c_\mu\right)\right)^2 \tag{19}$$

whose derivative with respect to v_i yields

$$\frac{\partial E_{\text{hopf}}}{\partial v_i} = -\sum_\mu \text{ReLU}\left(\boldsymbol{\xi}_\mu^{\text{hopf}}\mathbf{v} + c_\mu\right) \cdot \xi_{\mu i}^{\text{hopf}} \tag{20}$$

$$= -\sum_\mu \xi_{\mu i}^{\text{hopf}} f_\mu^{\text{hopf}}(\mathbf{v}) \tag{21}$$

which yields our desired dynamics of $\nabla_{\mathbf{v}} E_{\text{hopf}}(\mathbf{v}) = -\left(\boldsymbol{\xi}^{\text{hopf}}\right)^\top \mathbf{f}^{\text{hopf}}(\mathbf{v})$.

Effective energy function of analog energy transformer. All together, the effective scalar energy over the visible state $\mathbf{v}$ after integrating out hidden neurons is

$$E_{\text{eff}}(\mathbf{v}) = \underbrace{\frac{1}{2}\|\mathbf{v} - \mathbf{a}\|_2^2}_{E_{\text{quad}}} \underbrace{-\frac{1}{\beta} \log \sum_A \exp\left(\beta \left(\boldsymbol{\xi}_A^{\text{attn}}\mathbf{v} + b_A\right)\right)}_{E_{\text{attn}}} \underbrace{-\sum_\mu \frac{1}{2}\left(\text{ReLU}\left(\boldsymbol{\xi}_\mu^{\text{hopf}}\mathbf{v} + c_\mu\right)\right)^2}_{E_{\text{hopf}}} \tag{22}$$

This effective energy aligns with the canonical Energy Transformer's energy function. Because our effective dynamics use hidden neurons, the energy function written in the main text reflects the contributions of the hidden neurons. When $\tau_h \ll \tau_v$, this regime converges to the behavior when the hidden neurons are integrated out. Hence, the effective expressibility and behavior of our system is equivalent to that of the original Energy Transformer.

In our model we omit the layer normalization activation that the original Energy Transformer applies to the visible neurons. This keeps the circuit design simple, while still enabling models with high expressibility. This choice does not modify the structure of the attention or the Hopfield parts of the energy; only the self-energy of $\mathbf{v}$ differs. From a modeling perspective, layer normalization mainly improves conditioning and learning of deep networks rather than changing the computational primitive and expressibility. We empirically observe that the resulting models without layer normalization remain expressive enough to solve the problems we present. In principle, a layer normalization-type visible activation function could be implemented in analog hardware (e.g. by subtracting the mean voltage and normalizing by an on-chip variance estimate), but this would add distracting complications to the minimalist neuron and circuit designs we show in this paper.

Table 2: 8-bit parity model specification

Visible neurons v_i	$N_v = 16$ (dimension of embedding D)
Hidden neurons (energy attention) h_A^{attn}	$N_h^{\text{attn}} = 8$ (context length L)
Hidden neurons (Hopfield network) h_μ^{hopf}	$N_h^{\text{hopf}} = 16$ (Hopfield network memories M)
Hidden neurons (total)	$N_h = 24$ $(L + M)$
Visible activation	Identity: $g_i = v_i$
Hidden activation (energy attention)	Softmax: $f_A^{\text{attn}} = \text{softmax}(\beta \mathbf{h}^{\text{attn}})_A$ for $A = 1, \ldots, L$
Hidden activation (Hopfield network)	ReLU: $f_\mu^{\text{hopf}} = \max(h_\mu^{\text{hopf}}, 0)$ for $\mu = 1, \ldots, M$
Weights (energy attention)	$\boldsymbol{\xi}^{\text{attn}} \in \mathbb{R}^{L \times D}$, where $\boldsymbol{\xi}_A^{\text{attn}}$ is embedded A'th context token
Weights (Hopfield network)	$\boldsymbol{\xi}^{\text{hopf}} \in \mathbb{R}^{M \times D}$, static after training
Inference protocol	Embed L context tokens to obtain $\boldsymbol{\xi}^{\text{attn}}$. Let visible neurons evolve until convergence

C BIT STRING ENERGY TRANSFORMER IMPLEMENTATION

As described in Table 2, our trained model uses an embedding matrix of $2 \times D = 32$ parameters, the Hopfield network with $D \times M = 256$ parameters, an additional $D \times 2 = 32$ parameter matrix to decode embeddings to logits, a total of $D + L + M = 40$ neuron bias terms, and 2 biases for the linear decoder. This is a total of 362 parameters.

In training and inference we use time constants $\tau_v = 0.1$ and $\tau_h = 0.01$. We train with Euler steps of 1e-3, and test with Euler steps of 1e-4 for a time horizon of $T = 1$ second. JAX's automatic differentiation was used to implement backpropagation through time. We encourage the model to reach fixed points by adding a penalty to $\dot{v}$ at time T in the evaluation of the loss. This yields models that are more robust to hardware imperfection due to the intrinsic stability of attractor points. The convergence to an attractor also means the inference remains stable to mismatch and delay in timing during readout.