The Discrete-Log Clock: How a Transformer Learns Modular Multiplication

Huu Danh Nguyen [1]

Abstract

When small transformers grok modular multiplication, prior work reports that the learned embedding has a "dense" Fourier spectrum requiring all frequencies. This contrasts with modular addition, where only a sparse set of key frequencies suffices. We show this density is an artifact of analyzing in the wrong basis. The natural Fourier transform for multiplication is not the standard additive DFT but the *multiplicative character transform*, which decomposes functions on the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$ into its irreducible representations. Applying this transform to a grokked transformer trained on $a \cdot b \bmod 113$, we find the embedding spectrum becomes highly sparse (Gini coefficient 0.58 vs. 0.07 in the additive basis) with only 4 key frequencies carrying significant energy. Furthermore, 96.9% of MLP neurons are cleanly tuned to a single multiplicative frequency, and neuron activation heatmaps reveal 2D-periodic structure when reordered by the discrete logarithm. These results demonstrate the transformer reduces multiplication to addition in discrete-log space, implementing a "Discrete-Log Clock" algorithm analogous to Nanda et al.'s Clock algorithm for addition. The methodology generalizes: matching the analysis basis to the algebraic structure of the task reveals interpretable structure where standard tools see noise.

1. Introduction

Grokking, the phenomenon where neural networks suddenly generalize long after memorizing training data, has become a key testbed for mechanistic interpretability (Power et al., 2022). For modular addition ($a + b \bmod p$), the internal algorithm has been fully reverse-engineered: the model learns a sparse Fourier representation and applies trigonometric identities to compose rotations on a circle (Nanda et al., 2023; Zhong et al., 2023).

For modular multiplication ($a \cdot b \bmod p$), models also grok successfully, but the internal algorithm has remained poorly characterized. Doshi et al. (2024) derived analytical MLP solutions requiring dense Fourier components (all frequencies), and Furuta et al. (2024) confirmed experimentally that grokked transformers use cosine-biased components across all frequencies. These findings suggest multiplication uses a fundamentally different, less interpretable algorithm than addition.

Our contribution. We show this conclusion is premature. The "dense spectrum" is an artifact of analyzing with the *additive* Fourier transform, which is the natural basis for addition but not for multiplication. The correct analysis tool is the *multiplicative character transform*, which is the Fourier transform on the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$. In this basis, the embedding becomes sparse, the neurons cluster cleanly by frequency, and the algorithm is interpretable: It reduces multiplication to addition via the discrete logarithm.

Task formulation. The input to our model is a pair of integers $(a, b) \in \{1, \ldots, 112\}^2$. We train a 1-layer transformer to output the predicted product $c = a \cdot b \bmod 113$. The model observes 30% of all input pairs during training and must generalize to the remaining 70%.

2. Related Work

Mechanistic interpretability of modular addition. Nanda et al. (2023) fully reverse-engineered the algorithm learned by transformers on modular addition, showing the model uses discrete Fourier transforms and trigonometric identities to compose rotations. Zhong et al. (2023) named this the "Clock" algorithm and discovered an alternative "Pizza" algorithm, demonstrating that multiple algorithmic solutions exist for the same task. These works establish the methodology we extend to multiplication.

Prior work on modular multiplication. Doshi et al. (2024) derived analytical closed-form MLP weights for modular multiplication and found that the solutions contain the discrete logarithm $\log_g(i)$ in the weight structure. However, they used quadratic activations and did not connect this to the empirical Fourier picture in trained ReLU transform-

[1]Stanford University. Correspondence to: Huu Danh Nguyen <dannycodeinc@gmail.com>.

Mechanistic Interpretability Workshop at the 43^{rd} International Conference on Machine Learning, Seoul, South Korea, 2026.

ers. Furuta et al. (2024) trained transformers on modular polynomials (including multiplication) and reported that the learned spectrum uses "all frequencies" with no clear sparse structure, in contrast to addition. Our work resolves this discrepancy: the density is a basis artifact, not an algorithmic difference.

Group-theoretic frameworks. Chughtai et al. (2023) showed that networks trained on group operations learn irreducible representations of the relevant group. Stander et al. (2024) reverse-engineered networks on permutation group multiplication (S5, S6), finding coset-based circuits. McCracken et al. (2025) unified all known addition algorithms under an "approximate CRT" framework. These works study different groups (non-abelian permutation groups, additive cyclic groups) from ours; our contribution is applying the multiplicative character basis specifically to $(\mathbb{Z}/p\mathbb{Z})^*$ in a trained transformer.

3. Methods

3.1. Task and Architecture

We train a 1-layer decoder-only transformer on $a \cdot b \bmod 113$ for all $a, b \in \{1, \ldots, 112\}$ (excluding zero, which lies outside the multiplicative group). The architecture follows Nanda et al. (2023): $d_{\text{model}} = 128$, 4 attention heads ($d_{\text{head}} = 32$), MLP hidden dimension 512 with ReLU activation, no LayerNorm. Input format is the token sequence $[a, b, =]$; the model's prediction is read from the output logits at position 2 ("=").

Hyperparameters. We use 30% training data (3,763 of 12,544 pairs), AdamW with learning rate 10^{-3}, weight decay 1.0, and train for 40,000 epochs with full-batch gradient descent. These hyperparameters exactly follow Nanda et al. (2023) to ensure comparability; the only change is the task (multiplication vs. addition).

Training dynamics. The model memorizes the training set by epoch ∼500 (train loss → 0, test loss remains high). Between epochs 9,000 and 14,000, test loss drops suddenly to near zero: the model groks, achieving near-perfect generalization (Figure 1). This delayed generalization is characteristic of grokking: the model first overfits, then discovers the generalizing algorithm under pressure from weight decay.

3.2. The Multiplicative Group and Discrete Logarithm

Since 113 is prime, the nonzero integers $\{1, \ldots, 112\}$ form a cyclic group under multiplication mod 113. A *primitive root* is a generator: an element g such that $\{g^0, g^1, \ldots, g^{111}\} \bmod 113$ exhausts all 112 elements. For $p = 113$, we use $g = 3$.

The *discrete logarithm* $\log_g \colon \{1, \ldots, 112\} \to \{0, \ldots, 111\}$ assigns each element its exponent: $3^\alpha \equiv a \pmod{113}$ means $\log_3(a) = \alpha$. We write $\alpha = \log_g(a)$ and $\beta = \log_g(b)$ for the two inputs. This is an exact bijection (a permutation of the 112 elements) that defines a group isomorphism:

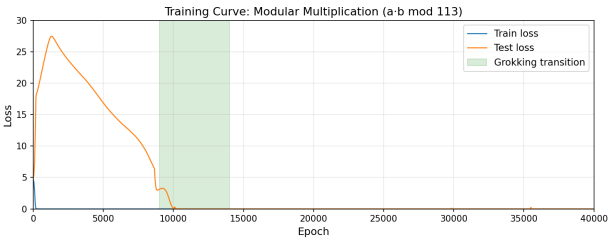

Figure 1. Training curve for $a \cdot b \bmod 113$. The model memorizes quickly (train loss drops by epoch 500), then generalizes suddenly during grokking (epochs 9K–14K). Weight decay drives the transition from memorization to the structured algorithm.

$$\log_g : (\mathbb{Z}/p\mathbb{Z})^* \xrightarrow{\cong} \mathbb{Z}/(p-1)\mathbb{Z} \tag{1}$$

Under this map, multiplication becomes addition:

$$a \cdot b \equiv 3^{(\alpha+\beta) \bmod 112} \pmod{113} \tag{2}$$

In the relabeled coordinates, the multiplication table *is* the addition table mod 112.

3.3. Two Fourier Bases

The **additive** Fourier basis consists of $\sin(2\pi ka/q)$ and $\cos(2\pi ka/q)$ for $k = 1, \ldots, q/2$, where $q = 112$. This is the standard DFT, natural for functions periodic in the integer label a.

The **multiplicative character basis** uses $\sin(2\pi k \log_g(a)/q)$ and $\cos(2\pi k \log_g(a)/q)$. Operationally: reorder the embedding rows by the discrete-log map, then take the standard DFT. This is the Fourier transform on the group $(\mathbb{Z}/p\mathbb{Z})^*$, decomposing functions into multiplicative characters.

The motivation is direct: since multiplication becomes addition after relabeling (Eq. 2), a model that has learned the group structure should have embeddings periodic in the relabeled coordinates, just as addition models have embeddings periodic in the original coordinates.

3.4. Analysis Methodology

We apply the following pipeline to the trained model:

Step 1: Embedding spectrum. Extract the trained embedding matrix $W_E \in \mathbb{R}^{112 \times 128}$. Project onto both bases and compute the combined frequency norm $\|f_k\| = \sqrt{\|s_k\|^2 + \|c_k\|^2}$ for each frequency k, where s_k and c_k are the sin and cos projections (each a 128-dim vector).

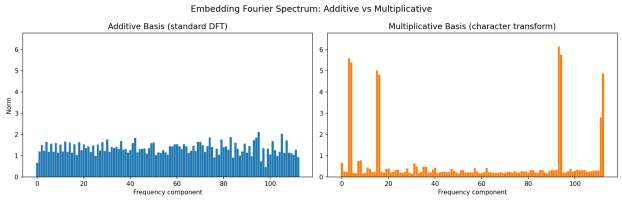

Figure 2. Embedding Fourier spectrum. **Left:** additive basis shows flat, dense spectrum. **Right:** multiplicative basis reveals 4 sparse peaks. Same y-axis scale. The "density" reported by prior work is a basis artifact.

Table 1. Sparsity metrics comparing the two bases.

Metric	Additive	Multiplicative
Gini coefficient	0.071	**0.579**
Inverse Participation Ratio	52.7	**4.1**
Key frequencies detected	0	**4**
Neurons >85% single-freq	0/512	**496/512**
Mean max-fraction	0.054	**0.925**

Step 2: Sparsity metrics. We measure concentration using the Gini coefficient:

$$G = \frac{\sum_{i=1}^{n}(2i - n - 1)|x_i|}{n\sum_{i=1}^{n}|x_i|} \tag{3}$$

where $x_1 \leq \ldots \leq x_n$ are the sorted frequency norms. $G = 0$ means uniform (dense), $G = 1$ means maximally sparse. Key frequencies are detected as those exceeding $5\times$ the median norm.

Step 3: Neuron frequency assignment. For each of 512 MLP neurons, compute the 2D activation pattern $h_n(a, b)$ over all input pairs (reshaped to a 112×112 grid). Take the 2D DFT in each basis and measure the maximum fraction of energy at any single frequency. Neurons with >85% energy at one frequency are classified as "single-frequency tuned."

Step 4: SVD analysis. Perform SVD on W_E and reorder the principal components by discrete logarithm. Sinusoidal structure in log-order confirms the embedding encodes multiplicative characters.

4. Experiments and Results

4.1. Basis Comparison

Figure 2 shows the embedding spectrum in both bases. In the additive basis, energy spreads uniformly (Gini = 0.071). In the multiplicative basis, energy concentrates in 4 peaks at frequencies $\{2, 8, 47, 56\}$ (Gini = 0.579, an $8.1\times$ increase). Table 1 summarizes all metrics.

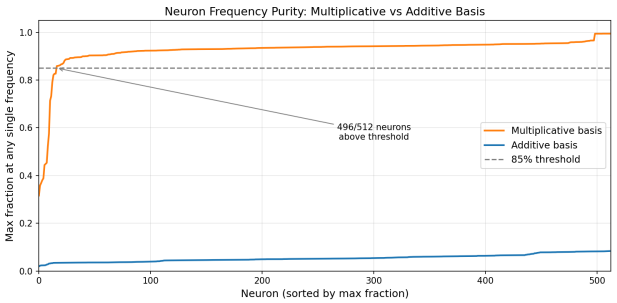

Figure 3. Sorted maximum frequency fraction per neuron. Orange: multiplicative basis (96.9% above 85% threshold). Blue: additive basis (0% above threshold).

4.2. Neuron-Level Confirmation

In the multiplicative basis, 496 out of 512 neurons (96.9%) have >85% of their Fourier energy at a single frequency (Figure 3). In the additive basis, zero neurons pass this threshold.

When neuron activation heatmaps are reordered by discrete logarithm, clear diagonal-stripe patterns emerge (Figure 4). These stripes indicate the neuron computes a periodic function of $\alpha + \beta$, which is the signature of the trigonometric identity $\cos(k(\alpha + \beta))$ operating in log-space: the same mechanism Nanda et al. found for addition, but applied after the discrete-log transformation.

4.3. SVD in Log-Order

We perform SVD on W_E and reorder principal components by discrete logarithm. In integer order, all components appear noisy. In log-order, several components reveal sinusoidal structure (Figure 5), consistent with the embedding encoding multiplicative characters. The effective rank is 10.3 (out of 112), confirming the embedding lives in a low-dimensional subspace.

4.4. Discussion

The Discrete-Log Clock algorithm. Our results show the transformer solves modular multiplication by: (1) embedding each integer a as sinusoidal functions of $\log_g(a)$; (2) routing embeddings via attention to the output position; (3) applying trig identities in the MLP to compute $\cos(k(\alpha + \beta)/q)$; (4) scoring each candidate c by alignment with $\cos(k\,\log_g(c)/q)$, where only the correct answer $c = ab \bmod p$ achieves constructive interference across all frequencies.

Why prior work missed this. Furuta et al. (2024) and Doshi et al. (2024) analyzed with the additive DFT. A function periodic in $\log_g(a)$ appears non-periodic in a because the discrete logarithm is a nonlinear permutation of the integers.

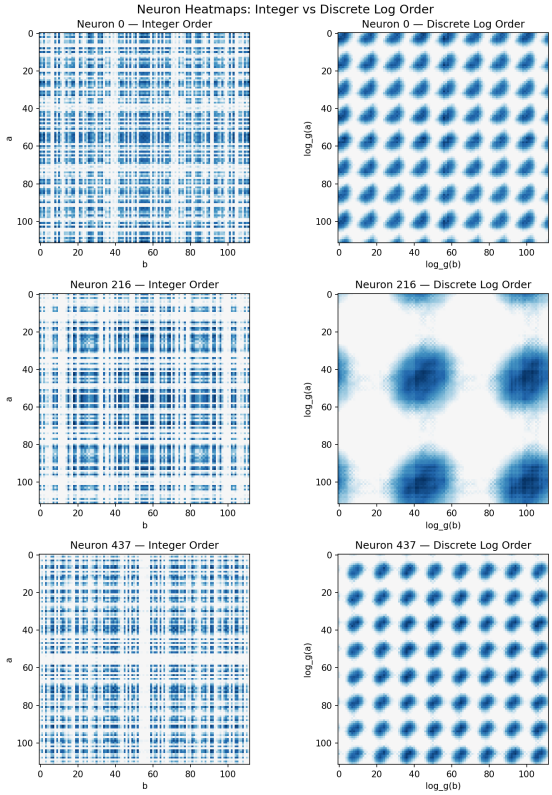

Figure 4. Neuron heatmaps in integer order (left) vs. discrete-log order (right). In log-order, diagonal stripes emerge, indicating periodicity in $\log_g(a) + \log_g(b)$.

The "all-frequencies" finding is correct in the additive basis but misleading because it conflates basis mismatch with algorithmic complexity.

Overfitting and generalization. Grokking is itself an overfitting phenomenon: the model achieves zero training loss by epoch 500 (memorization), then continues training for thousands of epochs before discovering the generalizing algorithm. Weight decay penalizes the large, unstructured weights needed for memorization, gradually favoring the compact Fourier representation that generalizes. We do not use cross-validation because the task is deterministic (no label noise); the 70% held-out test set directly measures generalization. The model achieves 100% test accuracy after grokking, confirming complete generalization.

Limitations. We do not prove this is the only possible algorithm; prior work on addition (Zhong et al., 2023) shows different architectures can yield different solutions.

5. Conclusion and Future Work

We have shown that a transformer trained on modular multiplication learns the "Discrete-Log Clock": it reduces multiplication to addition in discrete-log space, then applies the same Fourier/trigonometric mechanism known from addition. The key methodological insight is that the analysis basis must match the algebraic structure of the task.

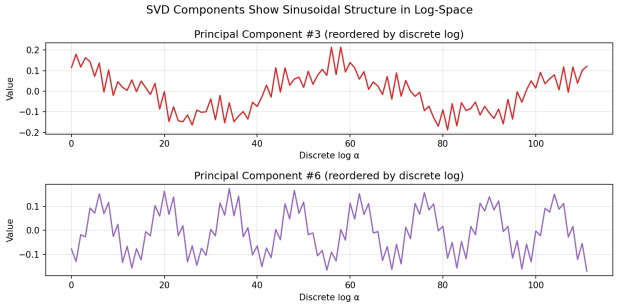

Figure 5. Principal components of W_E reordered by discrete logarithm show sinusoidal structure, confirming the embedding encodes multiplicative characters.

In follow-up experiments, we confirmed the Discrete-Log Clock across 10 primes ($p = 59$ to 113), all exhibiting sparse multiplicative spectra with 4 to 7 key frequencies. We also trained across multiple random seeds on $p = 113$ and found the mechanism is universal (∼80% of seeds), with consistent sparsity (Gini 0.45–0.61) despite varying key frequencies. We have also achieved preliminary results on modular exponentiation ($a^b \bmod p$) with a 2-layer transformer that achieves 100% accuracy on $a^b \bmod 41$, to our knowledge the first demonstration of a transformer perfectly learning this operation. The mechanism for exponentiation remains under active investigation.

The "right basis" principle extends naturally: for any algebraic task, the Fourier transform on the relevant group should reveal interpretable structure. Future work includes extending this methodology to polynomial operations, and multi-layer architectures.

Appendix

A.0 Reproducibility

Code is available at: https://github.com/danny-cpp/Discrete-Log-Clock

A.1 The Discrete-Log Clock Derivation

This appendix derives the algorithm the transformer approximately implements for modular multiplication. Let $\omega_k = 2\pi k/q$.

Symbol	Meaning
$g = 3$	Primitive root of $p = 113$
α, β, γ	Discrete logs of a, b, candidate c
$\mathcal{K} = \{2, 8, 47, 56\}$	Key frequencies
$\gamma^* = (\alpha+\beta) \bmod q$	Correct answer in log-space

A.2 The Embedding

Each dimension j of W_E, viewed as a function of α across all 112 elements, is approximately a sinusoid at a key frequency $k_j \in \mathcal{K}$:

$$W_E[g^\alpha,\ j] \ \approx \ A_j \sin(\omega_{k_j}\alpha + \phi_j) \tag{4}$$

A single row $W_E[a]$ has no visible structure: it is a point-sample of all 128 waves evaluated at $\alpha = \log_g(a)$, plus noise.

A.3 Attention

Attention routes information from positions a and b to position $=$. After attention, the residual stream at $=$ approximately encodes the Fourier components of both inputs at the key frequencies.

A.4 The MLP (Trigonometric Addition)

Writing $c_k(x) = \cos(\omega_k x)$ and $s_k(x) = \sin(\omega_k x)$ for brevity, the attention and MLP layers together approximately implement:

$$c_k(\alpha{+}\beta) = c_k(\alpha)\, c_k(\beta) - s_k(\alpha)\, s_k(\beta) \tag{5}$$

$$s_k(\alpha{+}\beta) = s_k(\alpha)\, c_k(\beta) + c_k(\alpha)\, s_k(\beta) \tag{6}$$

After the MLP, the residual stream approximately encodes $c_k(\alpha{+}\beta)$ and $s_k(\alpha{+}\beta)$ for each $k \in \mathcal{K}$.

A.5 Scoring (Dot Product with Unembedding)

The unembedding column for candidate c approximately encodes $(\cos(\omega_k\gamma), \sin(\omega_k\gamma))$ for each k. The logit for candidate c is the dot product of the final residual with this column. Since $\cos A \cos B + \sin A \sin B = \cos(A - B)$, each frequency's contribution reduces to:

$$\boxed{\text{logit}(c) \ \approx \ \sum_{k\in\mathcal{K}} A_k \cos(\omega_k(\alpha + \beta - \gamma))} \tag{7}$$

A.6 Constructive Interference

Correct answer ($\gamma^* = (\alpha{+}\beta) \bmod q$):

$$\text{logit}(c^*) \approx \sum_k A_k \cos(0) = \sum_k A_k \quad \text{(maximum)}$$

All cosines equal 1; all frequencies constructively interfere.

Wrong answer ($\Delta = \alpha{+}\beta{-}\gamma \neq 0$):

$$\text{logit}(c) \approx \sum_k A_k \cos(\omega_k \Delta) < \sum_k A_k$$

Cosines at different frequencies take values in $[-1, 1]$, partially canceling. With 4 incommensurate frequencies, destructive interference ensures no wrong answer approaches the correct answer's score.

References

Chughtai, B., Chan, L., and Nanda, N. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, 2023.

Doshi, D., He, T., Das, A., and Gromov, A. Grokking modular polynomials. *arXiv preprint arXiv:2406.03495*, 2024.

Furuta, H., Minegishi, G., Iwasawa, Y., and Matsuo, Y. Towards empirical interpretation of internal circuits and properties in grokked transformers on modular polynomials. *Transactions on Machine Learning Research*, 2024.

McCracken, G., Moisescu-Pareja, G., Letourneau, V., Precup, D., and Love, J. Uncovering a universal abstract algorithm for modular addition in neural networks. In *Advances in Neural Information Processing Systems*, 2025.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. In *ICLR 2022 Workshop on MATH-AI*, 2022.

Stander, D., Yu, Q., Fan, H., and Biderman, S. Grokking group multiplication with cosets. In *International Conference on Machine Learning*, 2024.

Zhong, Z., Liu, Z., Tegmark, M., and Andreas, J. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems*, 2023.