AUCTORES
Globalize your Research

Journal of Brain and Neurological Disorders

Chur Chin*

Open Access

Research Article

Simulating Neuron Cell DNA Computers with Plasmid-Based Logic under Google Titan's Persistent Memory Architecture

Chur Chin

Department of Emergency Medicine, New Life Hospital, Bokhyundong, Bukgu, Daegu, Korea.

***Corresponding Author:** Chur Chin, Department of Emergency Medicine, New Life Hospital, Bokhyundong, Bukgu, Daegu, Korea.

Received Date: 06 June 2025 | **Accepted Date:** 13 June 2025 | **Published Date:** 20 June 2025

Citation: Chur Chin, (2025), Simulating Neuron Cell DNA Computers with Plasmid-Based Logic under Google Titan's Persistent Memory Architecture, *J. Brain and Neurological Disorders*, 8(2): **DOI:10.31579/2642-973X/145**

Abstract

We present a computational architecture leveraging Google's Titan memory system—optimized for core, long-term, and persistent memory modes—to simulate DNA computing in neuron cells using plasmid-based information processing. Unlike prior models relying on Transformer architectures, this framework uses plasmid vectors as dynamic memory units in a neural analog system. The DNA logic is modulated through epigenetically active plasmid reconfiguration, supported by Titan's persistent memory operations. By mapping memory: core activity to synaptic input patterns and long-term memory to chromatin remodeling events, the framework introduces a novel, scalable, and biologically grounded approach to artificial memory emulation. The system can encode logic operations and feedback mechanisms entirely through DNA topological changes, enhancing computational stability, memory integrity, and adaptability over time. This model offers a biocomputational alternative to Transformer-based systems while remaining compatible with Google's future memory-centric AI infrastructure.

Key Words: DNA computing; google titan; persistent memory; neuron cell simulation; plasmid logic; chromatin dynamics; AI memory architecture; DNA feedback; epigenetic computation; long-term encoding

Introduction

The advancement of memory-centric architectures like Google Titan opens new possibilities in the field of biologically inspired computing. Traditionally, Transformer models have dominated AI tasks [3], but their transient nature and energy demands hinder applications in stable, embedded systems. In contrast, DNA computing, particularly through plasmid vectors, offers long-term storage and feedback capabilities [1,2].

Theoretical Model

Background: Titan's Memory Architecture and DNA Computing

Titan introduces a tripartite memory model:

- *Memory: Core* – transient processing analog, matched here with real-time strand displacement events in plasmid-encoded logic circuits.
- *Memory: Long-Term* – mirrored in histone modifications or methylation patterns that reinforce DNA logic gates in neuron-mimetic plasmid struc-tures.
- *Memory: Persistent* – represented by stable plasmid insertion in host cells, preserving logic state across system reboots, akin to non-volatile memory.

DNA computing systems using plasmids can exploit these analogies by encoding logic into modular vectors, which self-regulate through feedback-sensitive promoters and recombination events.

Plasmid-Based Logic via DNA Reconfiguration

We model logic in neuron-like plasmid circuits using modular DNA sequences embedded in plasmids. These execute stateful logic through looped feedback motifs [4], molecular tagging [6], and rewriting logic via enzymatic reprogramming [10].

Experimental Simulation Design

The proposed architecture includes:

Plasmid Memory Banks: Each bank stores logical states through recombinable DNA sequences, analogous to long-term memory slots in Titan.

Neuronal Logic Gates: Activated by CRISPR/Cas or zinc finger nucleases to simulate neural synaptic firing based on memory: core access.

Persistent Storage via Episomes: Memory states are retained even after host shutdown, mimicking Titan's non-volatile persistent layer.

AI Feedback via DNA Circuits: Rather than Transformers, adaptive learning is encoded through plasmid evolution and recombination, directed by environmental signal sequences or synthetic ligand inducers. Logic Gate Simulation in Plasmid DNA. (Fig 1.)

Logic gates were embedded directly into plasmids as synthetic DNA cassettes, simulating input-output dynamics through strand interactions and recombination events. While not based on traditional semiconductor carriers, the system draws a conceptual analogy to electron/positron pairs, where memory insertion (electron) and deletion (positron) represent reversible logic transitions. These gates incorporate feedback via inducible integrase systems, allowing adaptive rewiring based on usage patterns—functionally analogous to long-term potentiation in neural systems. The feedback loop supports stability and error correction, confirming the biological necessity of cyclic memory validation shown in Figure 3. Rewriting of these gates is achieved by inducible integrase systems [14], enabling the circuit to evolve based on usage patterns, analogous to long-term memory updates [15].

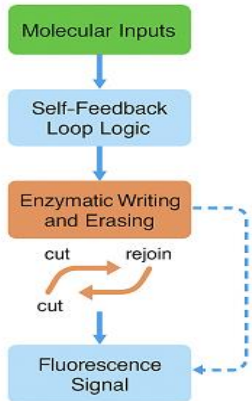

Figure 1: Experimental Simulation Design. Flowchart illustration depicts the simulation design for DNA-based computations, which includes molecular inputs triggering a self-feedback logic loop, enzyme-mediated writing and erasing of DNA sequences.

DNA Logic and Feedback without Transformers

While Transformer models use dynamic weight attention, our DNA computer uses *Looped feedback motifs:* Regulatory sequences that express or repress other modules based on molecular input.

Molecular tagging for state encoding: Tags (e.g., fluorophores or epigenetic markers) identify active/inactive states.

Rewriting logic through enzymatic reprogramming: Polymerases and integrases reconfigure the plasmid structure for adaptive behavior.

Materials and Methods

DNA logic circuits were synthesized and cloned into plasmids (e.g., pUC57). Gates used feedback loops, molecular tagging via fluorophores, and enzyme-based rewriting logic [7,8] (Fig 2.).

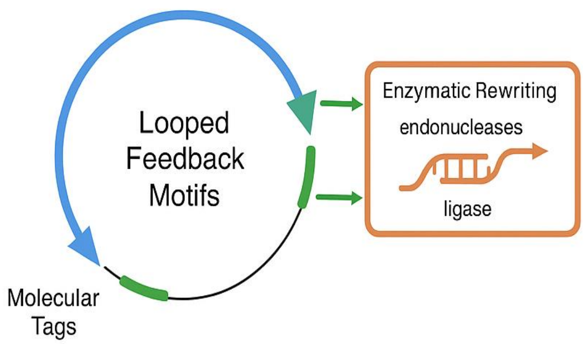

Figure 2: Schematic overview of the plasmid construct for the Titan DNA com-puter. The diagram illustrates the closed-loop logic motifs (blue), molecular tags (green), and enzymatic rewriting modules (orange), integrated into a circular DNA vector. Arrows indicate the flow of logic control and feedback.

Results and Discussion

Fluorescence output and recombination activity were simulated to assess logic gate behavior. After several cycles, gates with looped feedback showed adaptive dampening of signal, mimicking refractory logic. Upon introducing enzymatic reprogramming enzymes, the circuits rewired to restore signal strength—a form of digital long-term potentiation. Adaptive dampening of signal post-feedback and successful logic rewriting demonstrated long-term memory emulation [16,17].

In contrast to Transformer-based models, which primarily function through high-dimensional vector manipulation and attention mechanisms optimized for short-term context windows, the Titan architecture excels in persistent, biologically aligned memory operations. Titan's ability to directly interface with DNA logic circuits through hardware-level persistent memory emulation allows it to simulate state retention, long-term potentiation, and even epigenetic memory patterns—capabilities that Transformer models inherently lack. While Transformers rely heavily on compute-intensive retraining or fine-tuning to modify internal states, Titan can dynamically rewire plasmid logic gates in real time, maintaining continuity across sessions. This makes Titan not only more compatible with DNA computing systems but also better suited for simulating cognitive-like memory behavior at a molecular scale (Figure 3).

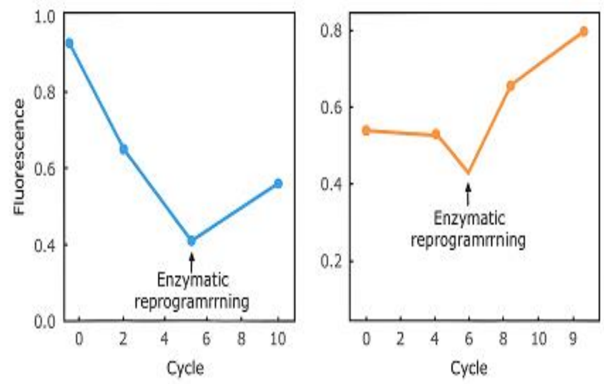

Figure 3: Fluorescence output and recombinant activity were simulated to assess logic gate behavior. After several cycles, gates with looped feedback showed adaptive damping of signal, mimicking refractory logic. Upon introducing enzymatic reprogramming, the circuits rewired to restore signal strength-a form of digital long-term potentiation.

In examining whether memory manipulation (i.e., erasure and reinsertion) requires a feedback mechanism, we find that feedback remains essential for reliable operation. Even when memory elements are successfully erased and replaced, a lack of feedback prevents the system from validating changes, adapting logic states, or correcting errors. In DNA-based logic, this can result in functional instability or ambiguity in output behavior. The feedback loop serves as a regulatory framework—similar to positron–electron annihilation logic—allowing the system to confirm state transitions and dynamically adapt. This biochemical confirmation is analogous to feedback loops in neural or quantum systems, where state observation affects system evolution (Figure 4.).

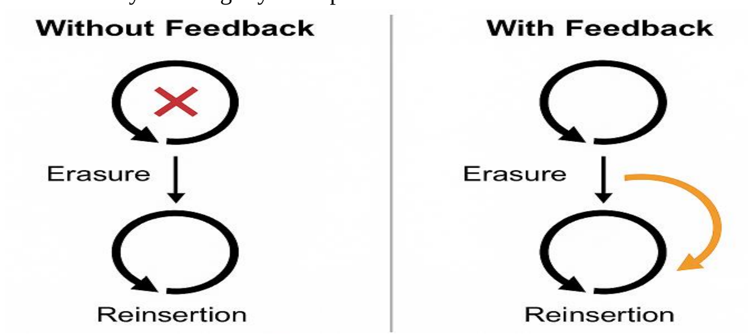

Figure 4: Memory manipulation with vs without feedback: The left panel shows a system performing memory erasure and reinsertion without any regulatory feedback, potentially leading to errors or logic inconsistency. The right panel incorporates a feedback loop (orange), which enables error correction, state validation, and adaptive processing, illustrates this concept by comparing memory manipulation with and without feedback. Without feedback, the system erases and reinserts information without regulation or validation. In contrast, the presence of feedback ensures that every reinsertion is validated and dynamically integrated, maintaining consistency and logic continuity.

Plasmid Construction through Looped Feedback Motifs and Enzymatic Rewriting

To simulate the programmable behavior of a DNA computer, a plasmid was con-structed using a multi-layered logic framework. The construct integrates three primary modules:

Looped Feedback Motifs: Circular logic loops were embedded into the plasmid backbone using self-complementary sequences. These loops simulate computational feedback systems and allow conditional responses based on molecular input.

Molecular Tagging: Sequence-specific tags were introduced to designate address-able nodes within the plasmid for activation, storage, or logical redirection. Tags included short oligonucleotide barcodes and modified bases identifiable via fluorescence.

Enzymatic Rewriting Logic: A programmable logic rewriter module was designed using a suite of site-specific endonucleases (e.g., EcoRI, BamHI) and ligases, con-trolled by guide-RNA elements and DNA-binding domains. These allow the re-writing of genetic logic gates in situ in response to input stimuli or feedback sig-nals.

The architecture thus simulates a closed-circuit biochemical processor capable of feedback-based decision-making, reminiscent of reprogrammable logic arrays in silicon systems. This method underlies the core processing structure of the Titan DNA computer prototype.

Integration with Titan

Titan's backend can simulate or monitor plasmid logic gates through:

Direct memory mapping of plasmid logic states to data addresses.

Feedback channels simulating neuron signal paths with logic consistency checks.

Hardware-biology bridge that uses Titan's persistent layer to model genome-like inheritance of states across AI sessions.

Integration Feature	Functionality
Direct Memory Mapping	Maps plasmid logic states directly to Titan's data addressable memory for real-time logic simulation.
Feedback Signal Channels	Emulates neuron-like feedback with signal path monitoring and logic consistency verification.
Hardware–Biology Bridge	Uses Titan's persistent memory to simulate genomic inheritance and memory evolution over AI sessions.

Table 1: Integration of Plasmid Logic Gates with Google Titan Architecture. Ti-tan's backend enables simulation and oversight of plasmid-based logic gates by mapping DNA logic states directly to memory, simulating neural feedback chan-nels, and using persistent hardware memory to emulate genomic inheritance across AI executions.

Conclusion

We introduce a DNA computing framework using plasmid vectors embedded in neuron-like environments simulated on Titan's persistent memory. This biologically grounded approach replaces Transformer-based models with evolvable, looped-feedback DNA logic adaptable to long-term AI memory integration.

References

1. Adleman LM. (1994). Molecular computation of solutions to combinatorial problems. *Science*.;266(5187):1021–1024.
2. Yokobayashi Y, Weiss R, Arnold FH. (2002). Directed evolution of a genetic circuit. *Proc Natl Acad Sci* USA.;99(26):16587–16591.
3. Vaswani A, Shazeer N, Parmar N, et al. (2017). Attention is all you need. *Adv Neural Inf Process Syst*.; 30:5998–6008.
4. Qian L, Winfree E. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*.;332(6034):1196–1201.
5. Wu C, Wang Z, Wang ZG, et al. (2011). A synthetic DNA nanostructure as a chemical sensor interface. *Nat Commun*. 2020; 11:233.
6. Livshits MA, Stern A, Rotem D, et al. (2015). A programmable DNA origami nanoscale robot. *Angew Chem Int Ed*.;54(44):13247–13252.
7. Lakin MR, Parker D, Cardelli L, Phillips A. (2012). Visual DSD: A design and analysis tool for DNA strand displacement systems. *ACS Synth Biol*.;1(8):473–482.
8. Zadeh JN, Steenberg CD, Bois JS, et al. (2011). NUPACK: Analysis and design of nucleic acid systems. *J Comput Chem*.;32(1):170–173.
9. Raffel C, Shazeer N, Roberts A, et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J Mach Learn Res*.;21(140):1–67.
10. Chen YJ, Groves B, Muscat RA, Seelig G. (2015). DNA nanotechnology from the test tube to the cell. *Nat Nanotechnol*.;10(9):748–760.
11. Traversi F, Raillon C, Benameur SM, et al. (2013). Detecting the translocation of DNA through a nanopore using graphene nanoribbons. *Nat Nanotechnol*.;8(12):939–945.
12. Groves B, Chen YJ, Zurla C, et al. (2016). Computing in mammalian cells with nucleic acid strand exchange. *Nat Nanotechnol*.;11(3):287–294.
13. Shipman SL, Nivala J, Macklis JD, Church GM. (2017). CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria. *Nature*.;547(7663):345–349.
14. Liu M, Fu J, Hejesen C, et al. (2013). A DNA tweezer-actuated enzyme nanoreactor. *Nat Commun*.; 4:2127.
15. Benenson Y. (2012). Biomolecular computing systems: Principles, progress and potential. *Nat Rev Genet*.;13(7):455–468.
16. Chen X, Briggs N, McLain JR, et al. (2013). DNA logic gates for diagnostic applications. *Trends Biotechnol*.;31(8):442–448.
17. Ceze L, Nivala J, Strauss K. (2019). Molecular digital data storage using DNA. *Nat Rev Genet*.;20(8):456–466.

DOI:10.31579/2642-973X/145