UNITED STATES PATENT APPLICATION

Provisional · Reading Edition

Neuromorphic Logic Integrated Circuit with Hardware-Enforced Behavioral Guarantees, Sealed Audit Architecture, and Fault Self-Healing Module

Provisional Application No. 64/089,930 · Confirmation No. 8191

Inventor: Timothy R. Griffin

Assignee: Adaptive Bespoke Learning LLC

Filed June 13, 2026 · Provisional · 35 U.S.C. 111(b) · Micro-Entity · Pro Se

Transparency Note

This is a reading edition. The complete specification text and all fifty-one (51) figures of U.S. Provisional Patent Application No. 64/089,930 are reproduced in full. The only change from the filed document is position: each figure has been moved to appear immediately after the paragraph that first describes it, so that drawing and description sit together. No text has been altered and no drawing has been modified. The copy on file with the United States Patent and Trademark Office is the authoritative record.

TITLE OF INVENTION

Neuromorphic Logic Integrated Circuit with Hardware-Enforced Behavioral Guarantees, Sealed Audit Architecture, and Fault Self-Healing Module

CROSS-REFERENCE TO RELATED APPLICATIONS

Not Applicable. This application is a provisional application for patent filed pursuant to 35 U.S.C. § 111(b). It claims priority to no prior application and establishes the priority date for the subject matter disclosed herein. A non-provisional application claiming the benefit of this provisional under 35 U.S.C. § 119(e), and any application under the Patent Cooperation Treaty, must be filed within twelve months of the filing date of this provisional to claim that benefit.

This application describes subject matter that may form the basis of continuation applications including: Continuation A, directed to multi-node federated audit architecture and cross-node verification without centralized trust; and Continuation B, directed to the integration of the Neuromorphic Logic Integrated Circuit with the Deterministic Transformation System owned by Adaptive Bespoke Learning LLC.

FIELD OF THE INVENTION

The present invention relates to neuromorphic integrated circuit architectures — specifically, hardware-implemented systems for spiking neural network computation that provide hardware-enforced behavioral guarantees including anomaly detection at the inference rate of the spiking network, non-reorderable fault correction to a cryptographically verified pre-anomaly state, and sealed audit records of every decision vector produced by the circuit, verifiable by an external governance authority without operator intermediation. The invention provides a technical improvement to neuromorphic computing architectures by eliminating the gap between autonomous operation and governance accountability at the hardware architecture level — not through software-mediated monitoring or iterative patching correction, but through structural hardware properties of the Fault Self-Healing Module, the VAE Anomaly Detection module, and the Security Engine.

BACKGROUND OF THE INVENTION

[0001] Existing neuromorphic computing architectures, including spiking neural network processors developed for edge AI deployment, exhibit four fundamental technical limitations that disqualify them from safety-critical, regulated, and governance-accountable applications.

[0002] First, existing neuromorphic architectures provide no hardware-enforced anomaly detection at the inference rate of the spiking neural network. Anomaly detection in prior art systems is implemented as a software-mediated watchdog operating on the output boundary of the system — detecting out-of-specification outputs after they are produced. The NLIC detects deviations from the normal operational envelope of the spike computation substrate before they propagate to the output boundary. Detection at an internal structural point in the causal chain, before fault effects reach the output, is a distinct fault class not addressed by prior art architectures.

[0003] Second, existing neuromorphic architectures correct anomalies by iterative patching — modifying the system state at the point of violation. A patched state is not the verified

pre-anomaly state. Patch drift accumulates across successive correction cycles. The NLIC corrects by executing a non-reorderable five-step correction sequence that terminates in restoration of the circuit to S_0 — the cryptographically verified pre-anomaly state — exactly. S_rollback = S_0 is an architectural property, not a best-effort approximation.

[0004] Third, existing neuromorphic architectures produce decision outputs without contemporaneous sealed audit records. Log-based audit systems produce records that are as trustworthy as the software process that produces them — modifiable by any process with write access to the log storage. The NLIC produces a cryptographically sealed audit record for each decision vector at the moment of production, using key material inaccessible to any software process. The sealed record is not an assertion of what occurred — it is a cryptographic proof.

[0005] Fourth, existing neuromorphic architectures do not provide an external governance interface that exposes sealed behavioral evidence to a regulatory authority without requiring operator intermediation or trust in the operator's self-reporting. The NLIC's Oversight Interface is read-only to the governance authority and exposes the complete sealed operational history of the circuit for independent cryptographic verification.

[0006] These limitations are not implementation defects in specific systems — they are architectural properties of software-mediated monitoring, output-boundary detection, iterative patching correction, and log-based audit models that define the prior art. There exists a need for a neuromorphic integrated circuit in which anomaly detection, fault correction, decision recording, and governance accountability are hardware-structural properties rather than software-configurable behaviors.

SUMMARY OF THE INVENTION

[0007] The present invention provides a neuromorphic logic integrated circuit comprising a Decision Module configured to compute decision vectors by executing a fixed optimization process invariant across successive executions given identical input state. The fixed optimization process is a hardware-structural property — not a runtime configuration parameter. In one embodiment the Decision Module is a deterministic digital quantum-inspired optimization solver; in an alternative embodiment the Decision Module comprises physical quantum optimization hardware.

[0008] The circuit further comprises a Spiking Neural Network array of processing cores producing spike traffic encoding learned behavioral representations. Spike traffic produced by each core is individually isolatable at a spike bus interface. Isolation is enforced by the hardware architecture of the spike bus interface, not by a software-mediated filtering process.

[0009] The circuit further comprises a Variational Autoencoder (VAE) (5000) configured to monitor spike traffic at the inference rate of the spiking neural network array and to detect deviation from the verified normal operational envelope before the deviation propagates to the output boundary of the circuit. The VAE monitors continuously — not periodically. The detection gap of polling-based prior art does not exist in this architecture.

[0010] The circuit further comprises a Fault Self-Healing Module configured to execute a non-reorderable five-step correction sequence upon anomaly detection: confirm the anomaly; suspend workload routing to the anomalous core; activate the redundant core array; transfer execution to the redundant core array; and isolate the anomalous core at the spike bus interface and restore the circuit to the verified pre-anomaly state S_0. S_rollback = S_0 exactly. No

intermediate patched state is produced. The output stream of the circuit is not interrupted during correction — it continues from the redundant core array through Step 4.

[0011] The circuit further comprises a Security Engine configured to produce a cryptographically sealed audit record for each decision vector at the moment of production, using key material inaccessible to any software process executing on the circuit. The sealed record is independently verifiable by an external governance authority through the Oversight Interface. The Oversight Interface is read-only — the governance authority observes, audits, and certifies without possessing write authority over any operational parameter of the circuit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate preferred embodiments of the invention. Figures are submitted as separate PDF in compliance with 37 CFR 1.84. All figures are black ink on white background, 8.5 × 11 in., with margins conforming to 37 CFR 1.84(g).

FIG. 1 System architecture diagram of the complete NLIC (1000), illustrating the six principal subsystems within the hardware boundary (1001) — Spiking Neural Network Processing Cores (2000), Memristive Synaptic Array (3000), Quantum Decision Module (4000), VAE Anomaly Detection module (5000), Fault Self-Healing Module (6000), and Security Engine (7000) — together with the Audit and Governance Layer (8000) and read-only Oversight Interface (8001).

FIG. 2 SNN Processing Core Architecture (2000) diagram illustrating the spiking neuron model, STDP weight update mechanism, asynchronous spike message bus (2001), and the 128-core array with 16 designated redundant cores.

FIG. 3 Memristive Synaptic Array (3000) diagram illustrating the conductance model $I(t) = G(w,t) \cdot V(t) + \eta(t)$, stable state checkpoint (3003) preserved before each cycle, and the rollback-only correction model.

FIG. 4 STDP Weight Update Model (2002) diagram illustrating pre-synaptic and post-synaptic firing paths, weight update equations, meta-learning parameter adaptation, and checkpoint sealing before each execution cycle.

FIG. 5 Meta-Learning Feedback Loop (2003) diagram illustrating the asynchronous parameter optimization loop from completed execution cycles to updated STDP parameters without interrupting spike processing.

FIG. 6 Quantum Decision Module Architecture (4000) diagram illustrating the Variational Quantum Eigensolver (4001), Quantum Annealing Solver (4002), and Decision Integrator combining outputs to guide NLIC pathway weighting.

FIG. 7 Variational Quantum Eigensolver Circuit and Optimization Loop (4001) diagram illustrating the parameterized quantum circuit, classical optimizer, Hamiltonian encoding the optimization objective, and return of minimum-energy configuration to the SNN.

FIG. 8 Quantum Annealing QUBO Energy Minimization (4002) diagram illustrating the QUBO formulation, energy landscape with global minimum $E(s^*)$, and optimal spin configuration returned to the Decision Integrator.

FIG. 9 VAE Anomaly Detection Architecture (5000) diagram illustrating encoder (5001a), latent space (5001b), decoder (5001c), reconstruction error (5001d), and dynamic threshold

comparison.

FIG. 10 Reconstruction Error and Threshold Model (5001) diagram illustrating binary comparator with no tolerance band, dynamic threshold θ(t), and positive determination (5002) as exclusive FSHM activation trigger.

FIG. 11 Binary Anomaly Determination as Exclusive FSHM Trigger (5002) diagram illustrating the architectural prohibition on external signal, software handler, or operator command activation of the FSHM independently of the binary determination.

FIG. 12 Fault Self-Healing Module Non-Reorderable Sequence (6000) diagram illustrating the five-step sequence from anomaly confirmation (6001) through workload suspension (6002), redundant core activation (6003), workload transfer (6004), and anomalous core isolation with rollback (6005).

FIG. 13 Redundant Core Rerouting (6006) diagram illustrating spike routing table remapping and topology preservation during FSHM workload transfer from anomalous core to activated redundant core.

FIG. 14 Rollback to Last Verified Stable State (3003) diagram illustrating corrupted state discard, immutable checkpoint retrieval cryptographically verified by Security Engine (7000), and atomic restoration to S_0.

FIG. 15 Non-Reorderable Self-Healing Sequence Structural Proof diagram illustrating four reordering violations each producing an undefined operation due to type mismatch rather than an incorrect result.

FIG. 16 CRYSTALS-Kyber Key Generation (7000) diagram illustrating hardware-isolated key generation (7000a), public key distribution for outbound encryption, and secret key confinement to security engine only.

FIG. 17 Hardware-Isolated Security Boundary (1001 / 7000) diagram illustrating chip interior, Security Engine (7000) encryption enforcement point, and external systems receiving encrypted packets only with no raw chip state.

FIG. 18 Encrypted Telemetry Flow (7000) diagram illustrating encryption of learning state, inference outputs, anomaly telemetry, and audit records through encryption pipeline (7000b) before crossing hardware boundary (1001).

FIG. 19 Immutable Audit Log (8000) diagram illustrating append-only cryptographically sealed event records, HMAC seal applied at recording time using hardware key, and read-only access via oversight interface (8001).

FIG. 20 Human-in-the-Loop Oversight Interface (8001) illustrating the read-only interface exposing anomaly determinations, rollback events, FSHM sequence records, quantum decision outputs, and audit log entries to external supervisory systems including human operators, regulators, audit reviewers, and compliance engines, with no commands accepted — the chip observes, structures, and exposes; it does not accept external operational commands.

FIG. 21 Full Integrated System Flow — Best Mode (1000) illustrating the complete operational sequence from input signal receipt through STDP execution, quantum decision, VAE monitoring, binary gate, and FSHM sequence, with parallel flows to encryption, audit log, checkpoint creation, and oversight interface, and the Best Mode specification: 128 SNN cores, 1.2 GHz adaptive clock, 8 MB ST and 64 MB LT memristive arrays, 16 redundant cores,

CRYSTALS-Kyber hardware-isolated, under 10W peak.

FIG. 22 Asynchronous Spike Messaging Topology (2001) illustrating the mesh topology of SNN cores firing on threshold rather than clock cycle, requiring no synchronization between cores, and the topology preservation guarantee during FSHM rerouting — the activated redundant core mirrors the anomalous core's messaging topology exactly without requiring re-architecture of spike routing.

FIG. 23 Deployment Configurations (1000) illustrating three deployment profiles — Cloud Integration (128 SNN cores, full VQE+annealing, under 10W), Edge Deployment (32 SNN cores, annealing only, under 3W), and On-Premises (configurable core count, full quantum module) — with the invariant that hardware boundary, security engine, FSHM, and VAE anomaly detection are active across all configurations.

FIG. 24 Prior Art Isolation vs. NLIC Integration illustrating five prior art systems — IBM TrueNorth/Intel Loihi, Infosys/Bank of America, UiPath/Dell/Microsoft, Microsoft/IBM/D-Wave, Amazon/Capital One — each providing one of the NLIC's five functions in isolation, and the NLIC novelty as integration of all five functions in a single chip architecture.

FIG. 25 Cognizance Proof — Element 1: Non-Reorderable Self-Healing Sequence — Type Contract Definition illustrating the five-step type chain from binary_determination(5002) through confirmed_anomaly_record, suspended_workload_map, active_redundant_core_set, completed_transfer_record, to verified_stable_state, proving the type equivalence principle: output type of step N equals input type of step N+1, defined at architecture time.

FIG. 26 Cognizance Proof — Element 1: Non-Reorderable Self-Healing Sequence — Reordering Violation Proof illustrating four violations each producing a type mismatch constituting an undefined operation: 6003 before 6002, 6004 before 6003, 6002 before 6001, and 6005 before 6004.

FIG. 27 Cognizance Proof — Element 1: Non-Reorderable Self-Healing Sequence — Scheduler Model vs. Structural Type Graph — Ontological Distinction illustrating that scheduler order is a runtime parameter producing wrong results detectable after execution, while structural type graph order is a property of the type system making reordering an undefined operation with no behavior defined.

FIG. 28 Cognizance Proof — Element 1: Non-Reorderable Self-Healing Sequence — Step Completeness Proof illustrating for each of the five steps its input type, output type, and the undefined operation that results from its removal, proving the sequence is at minimum necessary complexity — no step can be removed without breaking a type contract downstream.

FIG. 29 Cognizance Proof — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger — Gate Definition illustrating the binary gate (5002) accepting only POSITIVE or NEGATIVE output with no third state defined, no override, and no suppression, and the structural reasons a non-binary gate permits conditional triggering, race conditions, and external suppression that each break the autonomous recovery guarantee.

FIG. 30 Cognizance Proof — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger — Alternative Trigger Failure Proof illustrating four alternative triggers — external software handler, operator command, non-VAE hardware interrupt, probabilistic threshold — each failing for a distinct structural reason, and the conclusion that all four alternatives fail with none achieving the binary gate guarantee.

FIG. 31 Cognizance Proof — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger — Partial State Contamination Proof illustrating the operational timeline without a binary gate showing anomaly onset, continued SNN execution on anomalous state, weight corruption, checkpoint contamination before FSHM activation, and rollback to the corrupted checkpoint, versus the binary gate closing the contamination window to zero.

FIG. 32 Cognizance Proof — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger — Exclusivity Necessity Proof illustrating four weakenings of exclusivity — allow external activation, allow FSHM suppression, allow non-binary trigger, allow multiple trigger sources — each breaking a distinct structural guarantee, proving that binary determination as exclusive trigger is irreducible.

FIG. 33 Cognizance Proof — Element 3: On-Chip VAE Anomaly Detection — Latency Gap Proof illustrating the external VAE detection timeline from anomaly onset through telemetry sampling, transmission, external processing, determination return, and FSHM signal — contamination window T_0 to T_6 — versus on-chip detection executing at chip speed with contamination window approaching zero.

FIG. 34 Cognizance Proof — Element 3: On-Chip VAE Anomaly Detection — Integration Irreducibility Proof illustrating four weakenings — move VAE to external processor, replace with rule-based detector, remove anomaly detection entirely, use sampling-based external detection — each breaking a distinct guarantee of latency, adaptation, semantic content, or security boundary, proving on-chip VAE integration is structurally necessary.

FIG. 35 Cognizance Proof — Element 4: Memristive Checkpoint and Rollback Only — Patch vs. Rollback Categorical Distinction illustrating the patch model producing a modified artifact with accumulated drift versus the rollback model restoring verified stable state S_0 through nine atomic steps, and the categorical distinction: a patched state is a modified artifact — it is not a verified state.

FIG. 36 Cognizance Proof — Element 4: Memristive Checkpoint and Rollback Only — Patch Drift Proof illustrating the drift equation $S\_N = S_0 + \Sigma\delta_i$ across patch cycles, proving patch drift is unbounded and a single patch breaks the reproducibility guarantee, while rollback always restores $S\_N = S_0$ exactly.

FIG. 37 Cognizance Proof — Element 4: Memristive Checkpoint and Rollback Only — Rollback Structural Guarantees illustrating four guarantees achievable only through rollback: state purity ($S\_rollback = S_0$ exactly), checkpoint integrity (cryptographically sealed at creation), audit traceability (every rollback has a traceable origin in the sealed log), and operational continuity (system resumes from verified state without gap).

FIG. 38 Cognizance Proof — Element 4: Memristive Checkpoint and Rollback Only — Correction Model Irreducibility Proof illustrating four weakenings — remove correction entirely, allow patching, allow partial rollback, allow pre-anomaly correction — each breaking a distinct structural guarantee, proving checkpoint rollback is the only correction operation and is structurally necessary.

FIG. 39 Cognizance Proof — Element 5: Hardware-Isolated Security Boundary — Software Encryption Failure Proof illustrating four scenarios — software-layer encryption, network-layer encryption, key storage in software-accessible memory, encryption applied after data exits chip boundary — each failing for a distinct structural reason, proving hardware isolation at the chip boundary is architecturally necessary.

FIG. 40 Cognizance Proof — Element 5: Hardware-Isolated Security Boundary — Key Isolation Proof illustrating software key storage vulnerabilities versus hardware-isolated key storage in which no software process has memory-mapped access, side-channel resistance is built in, and key material is never transmitted, proving key isolation is a hardware property that cannot be achieved by software configuration.

FIG. 41 Cognizance Proof — Element 5: Hardware-Isolated Security Boundary — Boundary Necessity Proof illustrating seven attack vectors — physical access, process injection, memory read, side channel, network intercept, compromised OS, key extraction — each closed by the hardware boundary and each left open by the software security model, proving hardware boundary closure is a structural property not achievable by software.

FIG. 42 Cognizance Proof — Element 5: Hardware-Isolated Security Boundary — Security Model Irreducibility illustrating four weakenings — remove security engine entirely, replace hardware isolation with software access control, move encryption to application layer, allow unencrypted internal communication — each breaking a distinct structural guarantee, proving hardware-isolated boundary encryption is irreducible.

FIG. 43 Use Case: Autonomous Vehicle — Failure and Self-Healing, illustrating a self-driving vehicle navigating a complex urban intersection with an active processing core fault, comparing the without-NLIC scenario (200ms software handler recovery on corrupted decision state, braking delay, safety margin lost) against the with-NLIC scenario (VAE detection at chip speed, FSHM non-reorderable sequence, microsecond recovery, sealed audit record, no human required).

FIG. 44 Use Case: Healthcare Diagnostics — Adaptive Learning and Auditable Inference, illustrating a hospital diagnostic imaging system with regulatory auditability requirements, comparing software logs modifiable after the fact against the NLIC scenario in which every diagnostic output is encrypted at chip boundary, tied to sealed checkpoint, and cryptographically verifiable by regulator.

FIG. 45 Use Case: Cybersecurity Infrastructure — Threat Detection and Quantum Defense, illustrating a national cybersecurity operations center processing threat telemetry from millions of network endpoints, with the NLIC providing VAE reconstruction-error detection of novel patterns, STDP adaptation, VQE routing, CRYSTALS-Kyber boundary encryption, and FSHM self-healing during active attack.

FIG. 46 Use Case: Energy Grid Management — Fault-Tolerant Processing, illustrating a regional power grid control system managing load distribution across 10,000+ endpoints, comparing the without-NLIC scenario (150ms handler delay, incorrect load commands, state drift) against the with-NLIC scenario (chip-speed fault detection, workload suspension before transfer, checkpoint rollback to pre-fault state, sealed fault event record).

FIG. 47 Use Case: Aerospace Systems — Mission-Critical Self-Healing, illustrating an autonomous spacecraft on a multi-year deep-space mission with 20+ minute communication delay and no possibility of real-time human intervention, with the NLIC providing chip-speed fault detection, fully autonomous FSHM recovery without ground control involvement, quantum-resistant state protection, and sealed post-mission fault record.

FIG. 48 Use Case: Industrial Robotics — Real-Time Adaptive Learning and Fault Recovery, illustrating an industrial robotic assembly line requiring adaptation to component variation and microsecond fault recovery without halting production, with the NLIC providing STDP real-time

adaptation, VAE detection before defects are produced, FSHM recovery in microseconds, and checkpoint rollback ensuring the next cycle starts from verified pre-fault state.

FIG. 49 Plain Language Summary — What the NLIC Does and Why It Matters, illustrating five core capabilities: It Learns (STDP + gradient refinement + meta-learning), It Detects Its Own Failures (on-chip VAE binary determination at chip speed), It Heals Itself (FSHM + non-reorderable sequence + checkpoint rollback, no human required), It Keeps Its Decisions Private (CRYSTALS-Kyber hardware-isolated encryption, quantum-adversary resistant), and It Proves What It Did (immutable sealed audit log).

FIG. 50 The Full Stack — NLIC and Deterministic Transformation Architecture, illustrating the hardware layer (NLIC, 1000) and the architecture layer (Deterministic Transformation System, US Patent Application 19/638,930, filed April 3, 2026) as complementary independent patents owned by Adaptive Bespoke Learning LLC, with the integration claim that the two architectures together provide a complete stack for auditable, deterministic, hardware-enforced intelligent systems that neither provides alone.

FIG. 51 Federated Contributor-Controlled Language-Sovereignty Record, illustrating, in a single contributor instance, the sealing of a target-language content record (9001) produced through the deterministic transformation system interface (9000) under key material held in the contributor key custody (7001) within the hardware-isolated boundary (1001/7000) and inaccessible to operator software, the sealed record and its attribution provenance record (8003) written to the audit log (8000), and the read-only verification path through the Oversight Interface (8001) to a contributor community authority (9002); and, in a federated configuration, a plurality of contributor instances each holding an independent record with no central aggregating node.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Overview

[0013] The Neuromorphic Logic Integrated Circuit (NLIC) (1000) is a hardware-integrated system comprising six principal subsystems operating concurrently within a common hardware trust boundary (1001). The hardware trust boundary (1001) is the encryption and isolation enforcement point of the circuit, and may be realized as a monolithic single-die integration, a multi-die or chiplet integration, or a heterogeneous integration, provided that all six subsystems reside within the boundary and all outbound data crosses the single enforcement point (1001). The six subsystems are: the Spiking Neural Network Processing Cores (2000); the Memristive Synaptic Array (3000); the Quantum Decision Module (4000); the VAE Anomaly Detection module (5000); the Fault Self-Healing Module (FSHM) (6000); and the Security Engine (7000). An Audit and Governance Layer (8000) with a read-only Oversight Interface (8001) accumulates and exposes the complete sealed operational history of the chip. All outbound data is encrypted by the Security Engine (7000) before crossing the hardware boundary (1001). No unencrypted data exits the chip under any operational condition.

System Architecture — FIG. 1

[0014] Referring to FIG. 1, the NLIC Chip System Architecture Overview (1000) illustrates the six principal subsystems and their integration within the hardware boundary (1001). The SNN Processing Cores (2000) connect to the Memristive Synaptic Array (3000), which connects to the Quantum Decision Module (4000). The VAE Anomaly Detection module (5000) connects to the

Fault Self-Healing Module (6000), which connects to the Security Engine (7000). The Fault Self-Healing Module (6000) executes rollback to the Memristive Synaptic Array (3000). The Audit and Governance Layer (8000) spans all subsystems, accumulating cryptographically sealed event records through the read-only Oversight Interface (8001). All outbound data is encrypted by the Security Engine (7000) before crossing the hardware boundary (1001). The architectural statement of the chip is: no unencrypted data exits the chip under any operational condition.

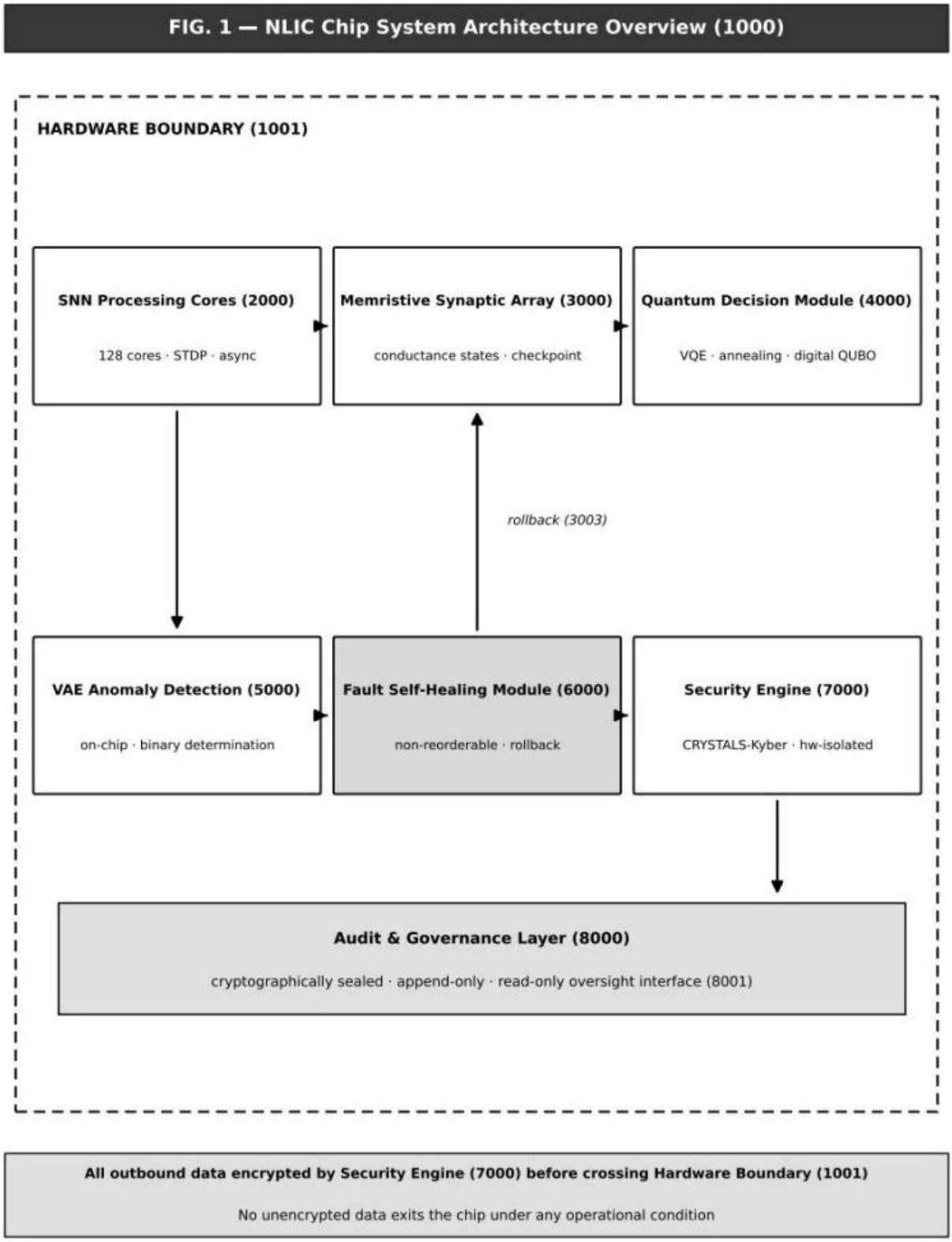


FIG. 1 — NLIC Chip System Architecture Overview (1000)
HARDWARE BOUNDARY (1001)
SNN Processing Cores (2000)
128 cores · STDP · async
Memristive Synaptic Array (3000)
conductance states · checkpoint
Quantum Decision Module (4000)
VQE · annealing · digital QUBO
rollback (3003)
VAE Anomaly Detection (5000)
on-chip · binary determination
Fault Self-Healing Module (6000)
non-reorderable · rollback
Security Engine (7000)
CRYSTALS-Kyber · hw-isolated
Audit & Governance Layer (8000)
cryptographically sealed · append-only · read-only oversight interface (8001)
All outbound data encrypted by Security Engine (7000) before crossing Hardware Boundary (1001)
No unencrypted data exits the chip under any operational condition

FIG. 1

FIG. 1

SNN Processing Core Architecture — FIG. 2

[0015] Referring to FIG. 2, the SNN Processing Core Architecture (2000) comprises individual spiking neuron models including Neuron A (2001a), Neuron B (2001b), and Neuron C (2001c), each operating on a spiking threshold and leak model. Spike-timing-dependent plasticity weight updates are governed by the STDP Weight Update module (2002) applying the equation $\Delta w_{ij}(t) = A+(t) \cdot \exp(-(t_i - t_j)/\tau+(t))$, where $A+(t)$ and $A-(t)$ are meta-learned amplitude parameters and $\tau+(t)$ and $\tau-(t)$ are adaptive time constants, with a gradient refinement term $\eta \cdot \partial E/\partial w$. Weight updates are applied to the Memristive Array (3000). The Asynchronous Spike Message Bus (2001) connects all cores without requiring synchronization between them. The best mode specification comprises 128 SNN processing cores operating at a 1.2 GHz adaptive clock with a peak power of less than 10 Watts, with 16 designated redundant cores maintained in standby for Fault Self-Healing Module (6000) activation.

FIG. 2 — SNN Processing Core Architecture (2000)

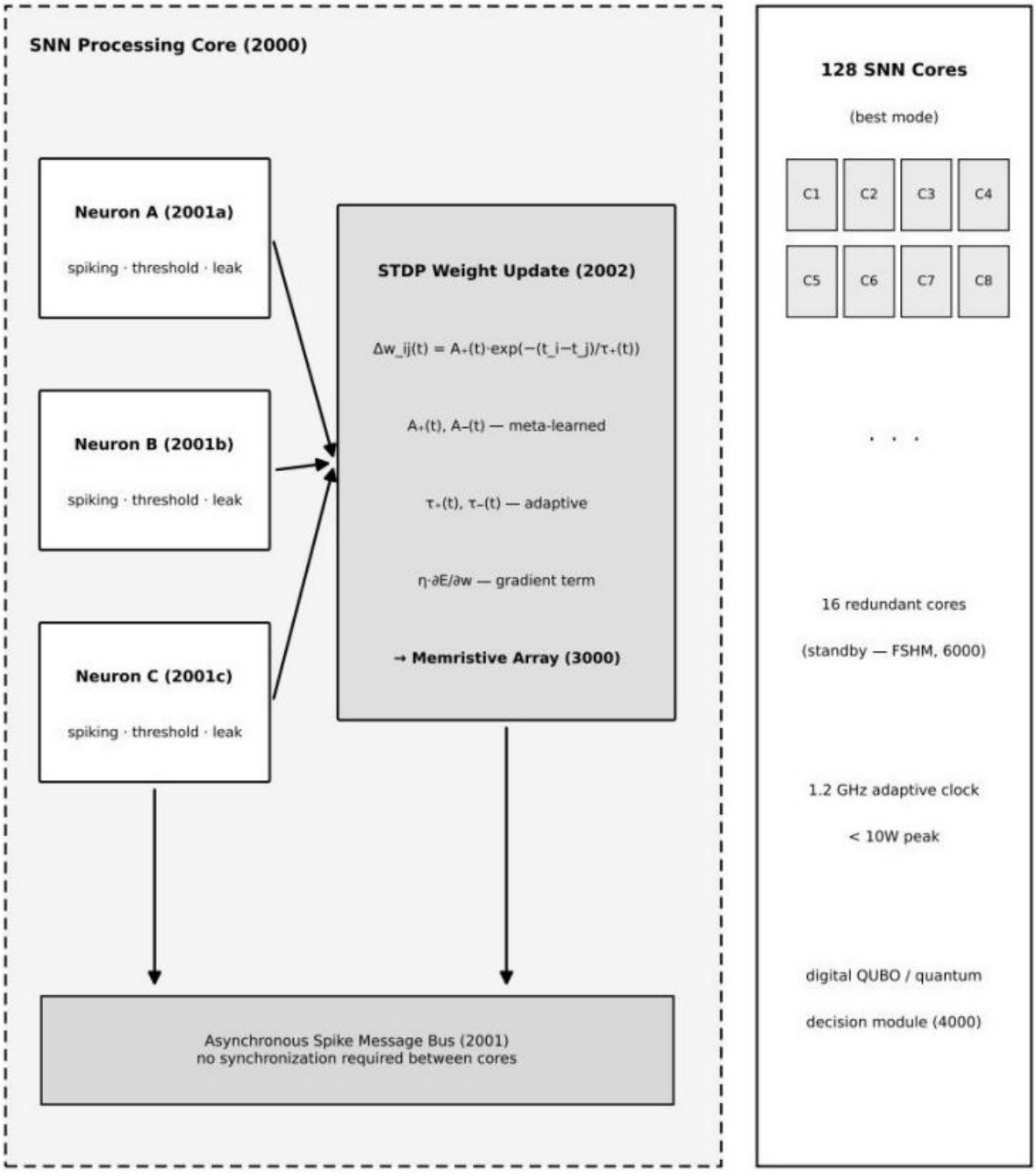


FIG. 2

FIG. 2

Memristive Synaptic Array — FIG. 3

[0016] Referring to FIG. 3, the Memristive Synaptic Array (3000) comprises memristive elements M1 through M30, each storing a conductance value G(w,t) encoding a synaptic weight. The conductance model (3002) is I(t) = G(w,t)·V(t) + η(t), where I(t) is the synaptic current, G(w,t) is the conductance at time t encoding weight w, V(t) is the applied voltage, and η(t) is hardware variability noise. Long-term potentiation produced by STDP execution (2002) increases G(w,t); long-term depression decreases G(w,t). The Stable State Checkpoint (3003) is a complete record of all conductance states, all weight parameters, and all core routing tables, preserved before each execution cycle without exception. The checkpoint is cryptographically sealed by the Security Engine (7000) at the moment of creation. Patching is architecturally prohibited. Correction is checkpoint rollback only (3003).

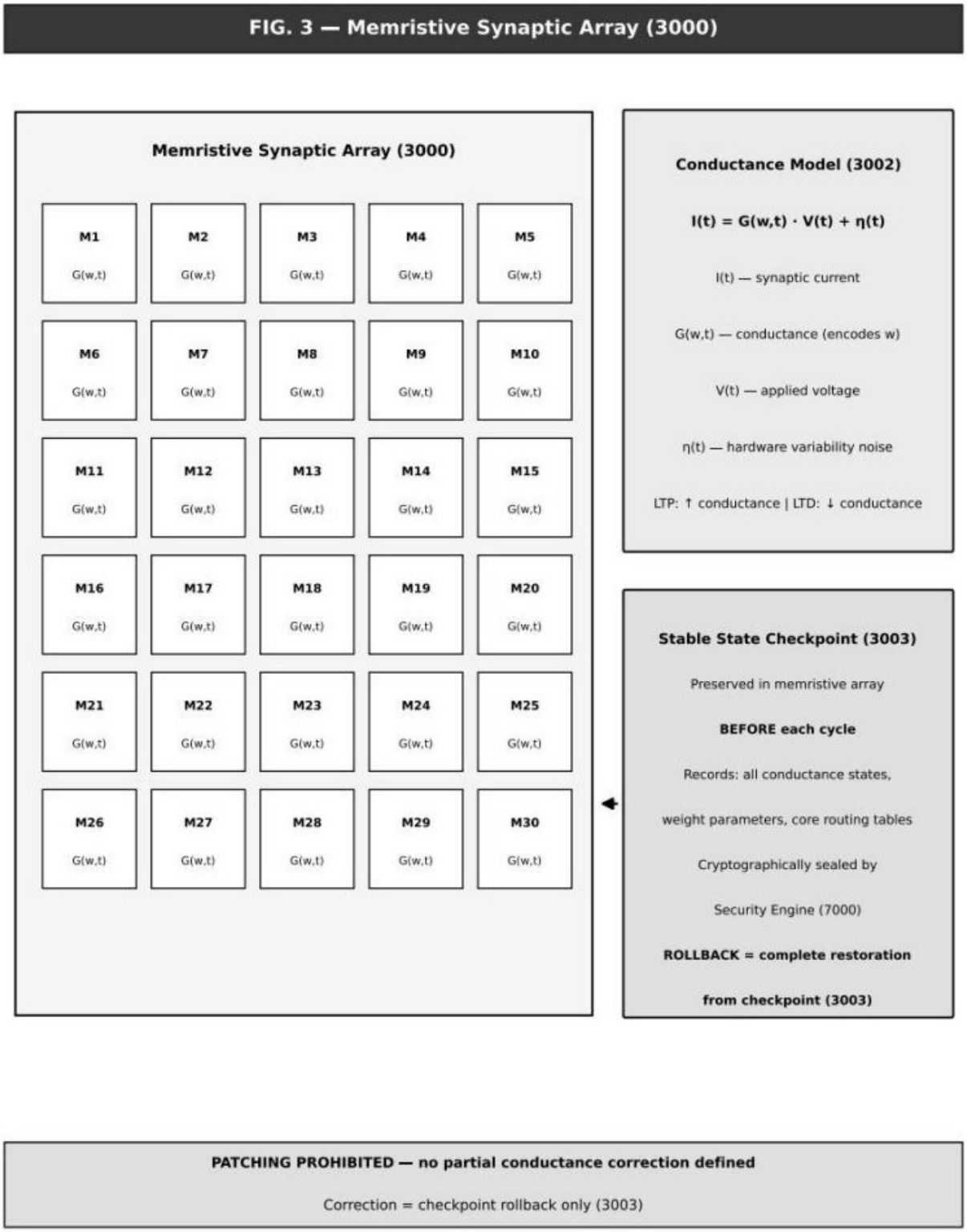


FIG. 3 — Memristive Synaptic Array (3000)
Memristive Synaptic Array (3000)
M1 G(w,t)
M2 G(w,t)
M3 G(w,t)
M4 G(w,t)
M5 G(w,t)
M6 G(w,t)
M7 G(w,t)
M8 G(w,t)
M9 G(w,t)
M10 G(w,t)
M11 G(w,t)
M12 G(w,t)
M13 G(w,t)
M14 G(w,t)
M15 G(w,t)
M16 G(w,t)
M17 G(w,t)
M18 G(w,t)
M19 G(w,t)
M20 G(w,t)
M21 G(w,t)
M22 G(w,t)
M23 G(w,t)
M24 G(w,t)
M25 G(w,t)
M26 G(w,t)
M27 G(w,t)
M28 G(w,t)
M29 G(w,t)
M30 G(w,t)
Conductance Model (3002)
I(t) = G(w,t) · V(t) + η(t)
I(t) — synaptic current
G(w,t) — conductance (encodes w)
V(t) — applied voltage
η(t) — hardware variability noise
LTP: ↑ conductance | LTD: ↓ conductance
Stable State Checkpoint (3003)
Preserved in memristive array
BEFORE each cycle
Records: all conductance states,
weight parameters, core routing tables
Cryptographically sealed by
Security Engine (7000)
ROLLBACK = complete restoration
from checkpoint (3003)
PATCHING PROHIBITED — no partial conductance correction defined
Correction = checkpoint rollback only (3003)

FIG. 3

FIG. 3

STDP Weight Update Model — FIG. 4

[0017] Referring to FIG. 4, the STDP Weight Update Model (2002) illustrates the two-path weight update mechanism. In the potentiation path where $t_i > t_j$, the update equation is $\Delta w_{ij}(t) = A+(t) \cdot \exp(-(t_i - t_j)/\tau+(t))$, producing long-term potentiation and increasing conductance in the corresponding memristive element (3000). In the depression path where $t_j > t_i$, the equation is $\Delta w_{ij}(t) = -A-(t) \cdot \exp(-(t_j - t_i)/\tau-(t)) + \eta \cdot \partial E/\partial w_{ij}$, incorporating a gradient refinement term. The weight update $\Delta w_{ij}(t)$ is applied to memristive element (3001). The Stable State Checkpoint (3003) is sealed before the next cycle; rollback restores all Δw values. Meta-Learning (2003) adapts $A+$, $A-$, $\tau+$, $\tau-$ across cycles asynchronously without interrupting spike processing.

FIG. 4 — STDP Weight Update Model (2002)

Meta-Learning (2003): A_+, A_-, τ_+, τ_- adapted across cycles

Performance signals from completed cycles → parameter adjustment → reduced prediction error

Meta-learning runs asynchronously — does not interrupt spike processing

FIG. 4

FIG. 4

Meta-Learning Feedback Loop — FIG. 5

[0018] Referring to FIG. 5, the Meta-Learning Feedback Loop (2003) illustrates the asynchronous parameter optimization loop. Execution Cycle N completes in the SNN Processing Cores (2000) and produces a performance signal passed to the Parameter Optimizer, which functions as a meta-learner. The Parameter Optimizer produces updated parameters A+(t), A−(t), τ+(t), τ−(t) applied to the STDP Weight Update module (2002) before Execution Cycle N+1 begins. The meta-learning loop runs asynchronously and does not interrupt spike processing.

FIG. 5 — Meta-Learning Feedback Loop (2003)

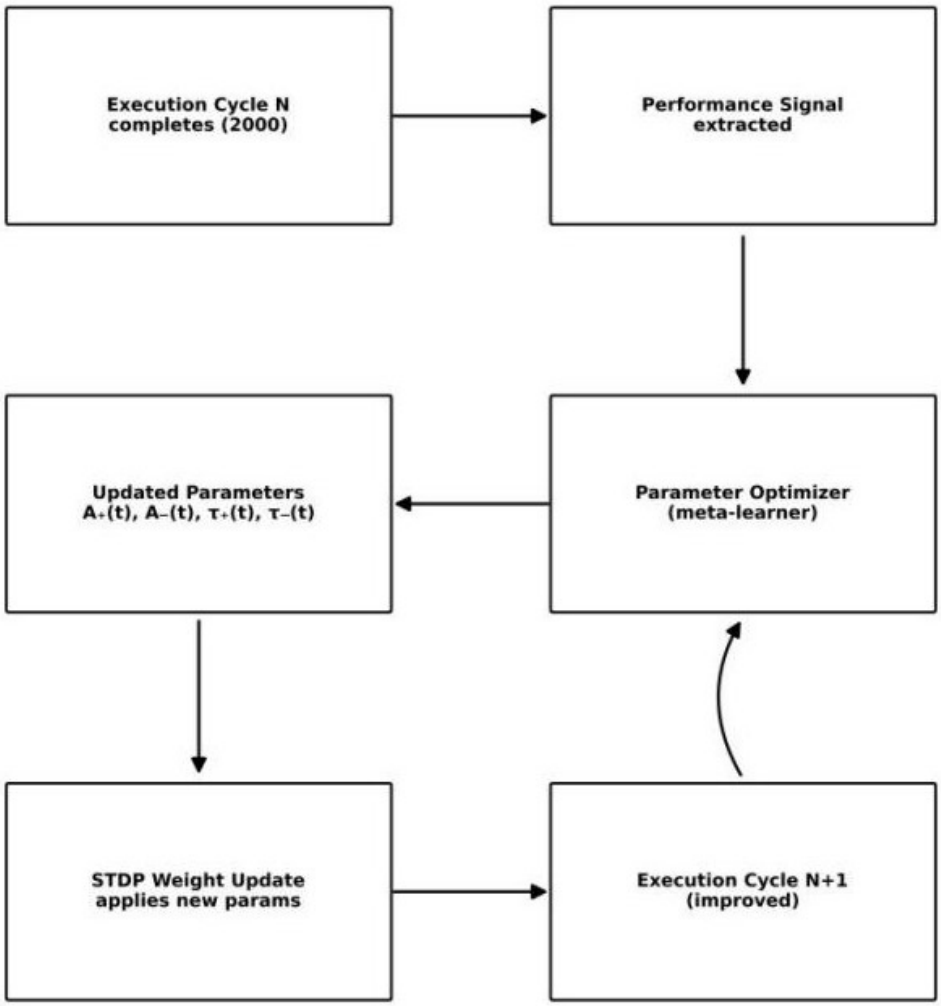


FIG. 5

FIG. 5

Quantum Decision Module Architecture — FIG. 6

[0019] Referring to FIG. 6, the Quantum Decision Module Architecture (4000) comprises the Variational Quantum Eigensolver (4001) and the Quantum Annealing Solver (4002). The VQE (4001) evaluates $E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$ through a parameterized quantum circuit and classical optimization loop, returning the minimum-energy configuration. The Quantum Annealing Solver (4002) executes $\min E(s) = \Sigma h_i \cdot s_i + \Sigma J_{ij} \cdot s_i \cdot s_j$ over a QUBO formulation, returning the optimal spin configuration. Both outputs are combined by a Decision Integrator that guides neural pathway weighting in the SNN Cores (2000) and is applied to the STDP update at the next execution cycle.

FIG. 6 — Quantum Decision Module Architecture (4000)

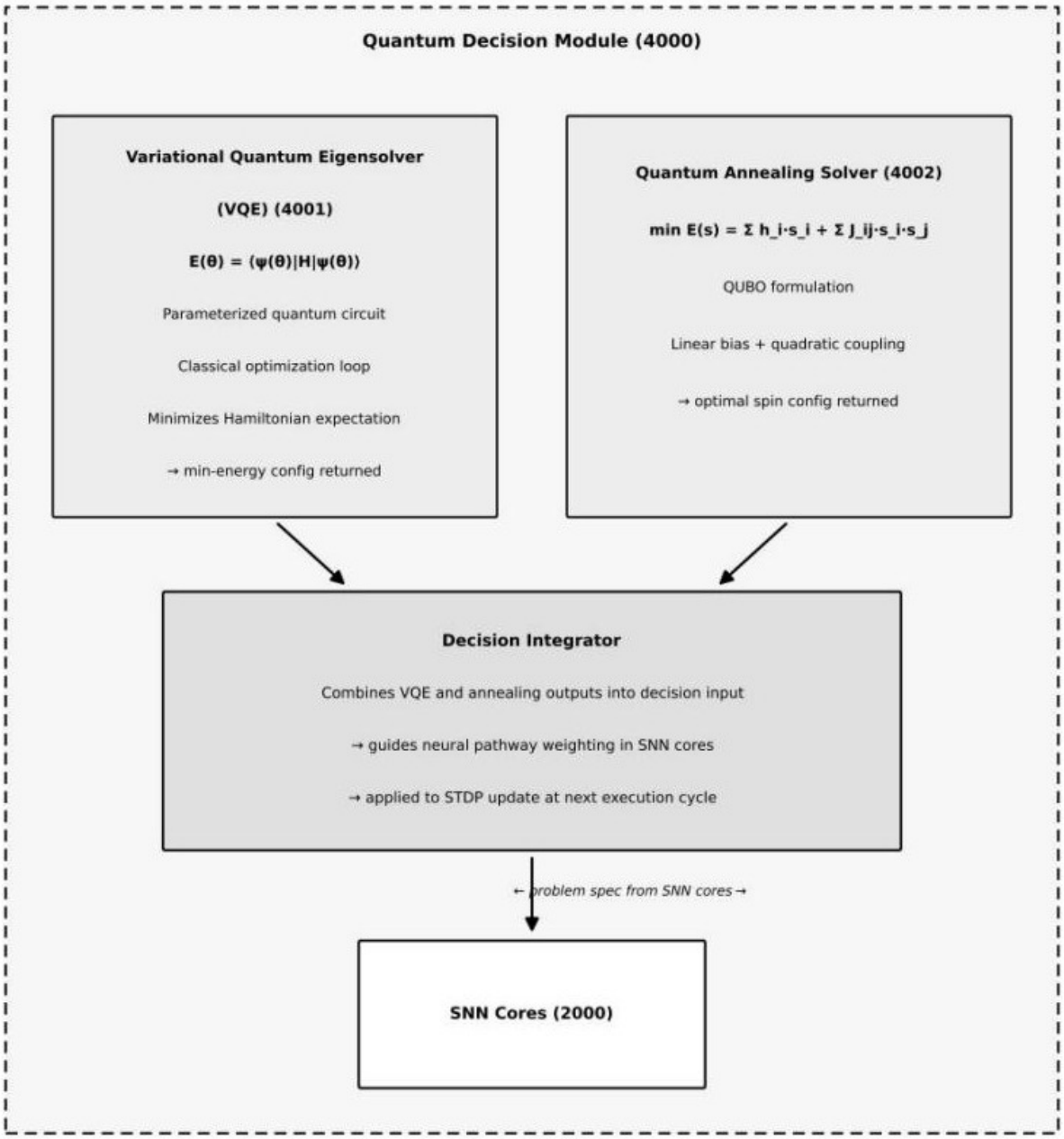


FIG. 6

FIG. 6

Variational Quantum Eigensolver Circuit and Optimization Loop — FIG. 7

[0020] Referring to FIG. 7, the Variational Quantum Eigensolver Circuit and Optimization Loop (4001) illustrates the parameterized quantum circuit $|\psi(\theta)\rangle$ comprising qubit lines q0 through q3 each passing through Hadamard, Rz, Rx, CNOT, and Rz gate operations. The circuit parameter vector $\theta = \{\theta_1, \theta_2, \ldots \theta\_n\}$ is classically optimized. The Classical Optimizer (4001b) evaluates $E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$; when $E(\theta)$ reaches E_min the minimum-energy configuration is returned to the SNN Cores (2000) as a decision input. The Hamiltonian H (4001c) encodes the optimization objective derived from the SNN learning state.

FIG. 7 — Variational Quantum Eigensolver — Circuit & Optimization Loop (4001)

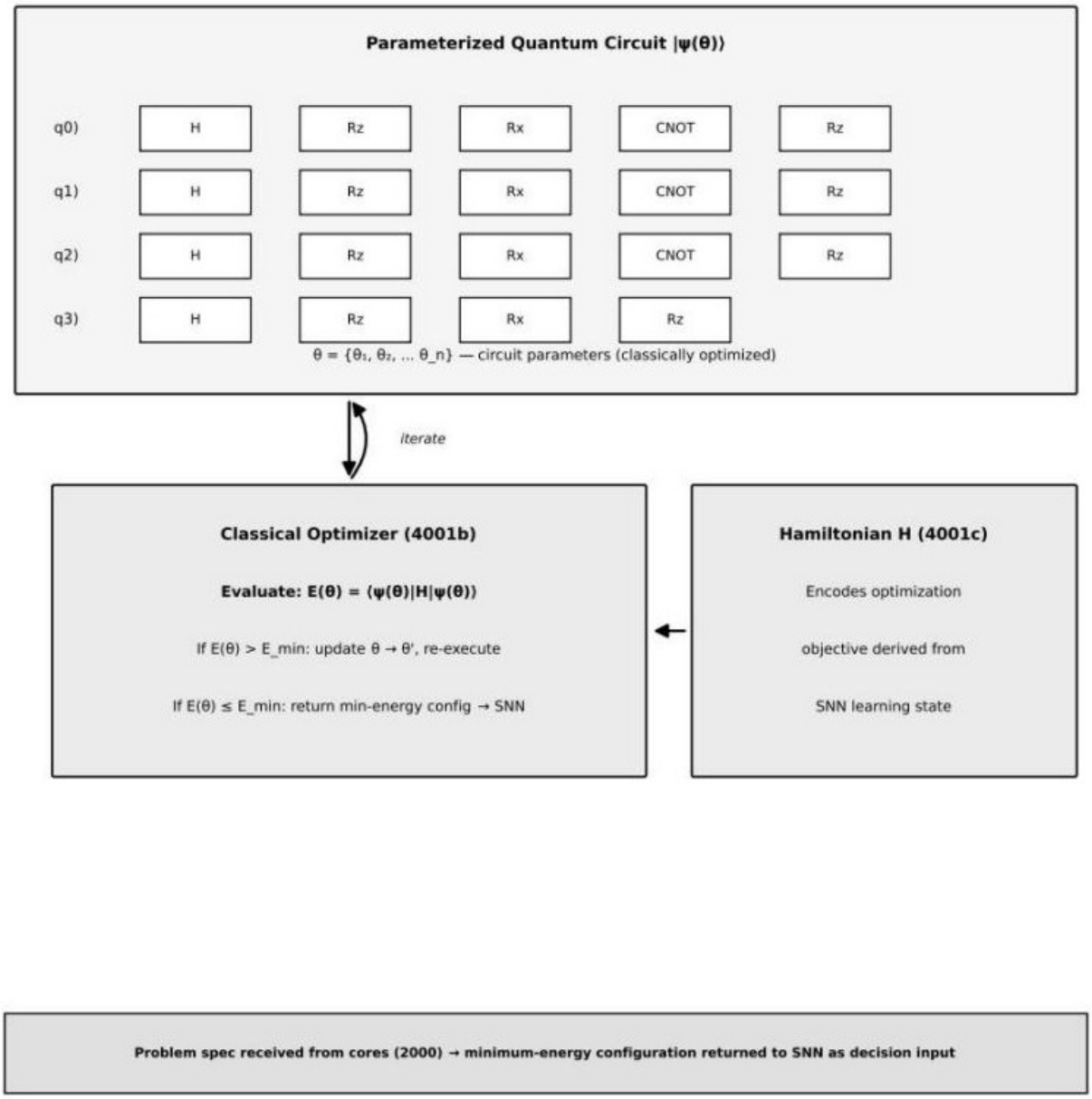


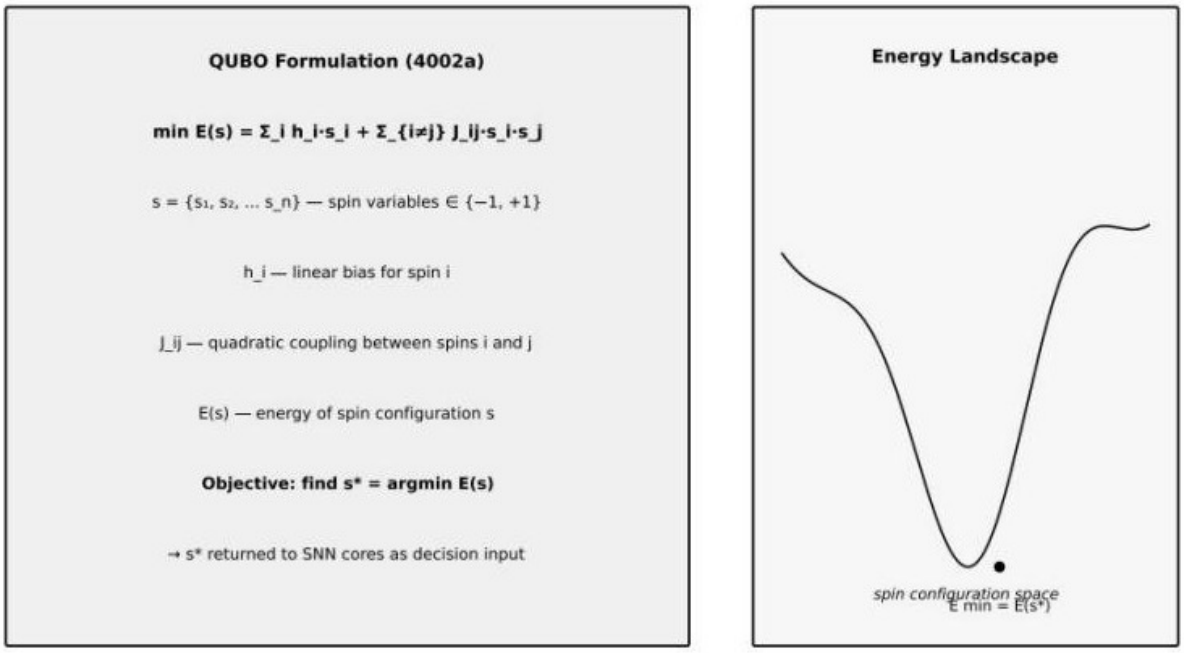
FIG. 7

FIG. 7

Quantum Annealing QUBO Energy Minimization — FIG. 8

[0021] Referring to FIG. 8, the Quantum Annealing QUBO Energy Minimization (4002) illustrates the QUBO formulation (4002a) min E(s) = Σ_i h_i·s_i + Σ_{i≠j} J_ij·s_i·s_j, where s = {s_1, s_2, … s_n} are spin variables in {−1, +1}, h_i is the linear bias, J_ij is the quadratic coupling, and E(s) is the energy of spin configuration s. The objective is to find s* = argmin E(s). The optimal spin configuration s* (4002b) is returned to the SNN Decision Integrator (4000) and applied to neural pathway weighting at the next STDP cycle.

FIG. 8 — Quantum Annealing — QUBO Energy Minimization (4002)



Output: optimal spin configuration s* (4002b)

Returned to SNN Decision Integrator (4000)

Applied to neural pathway weighting at next STDP cycle

FIG. 8

FIG. 8

Digital Quantum-Inspired Solver Embodiment — Decision Module (4000)

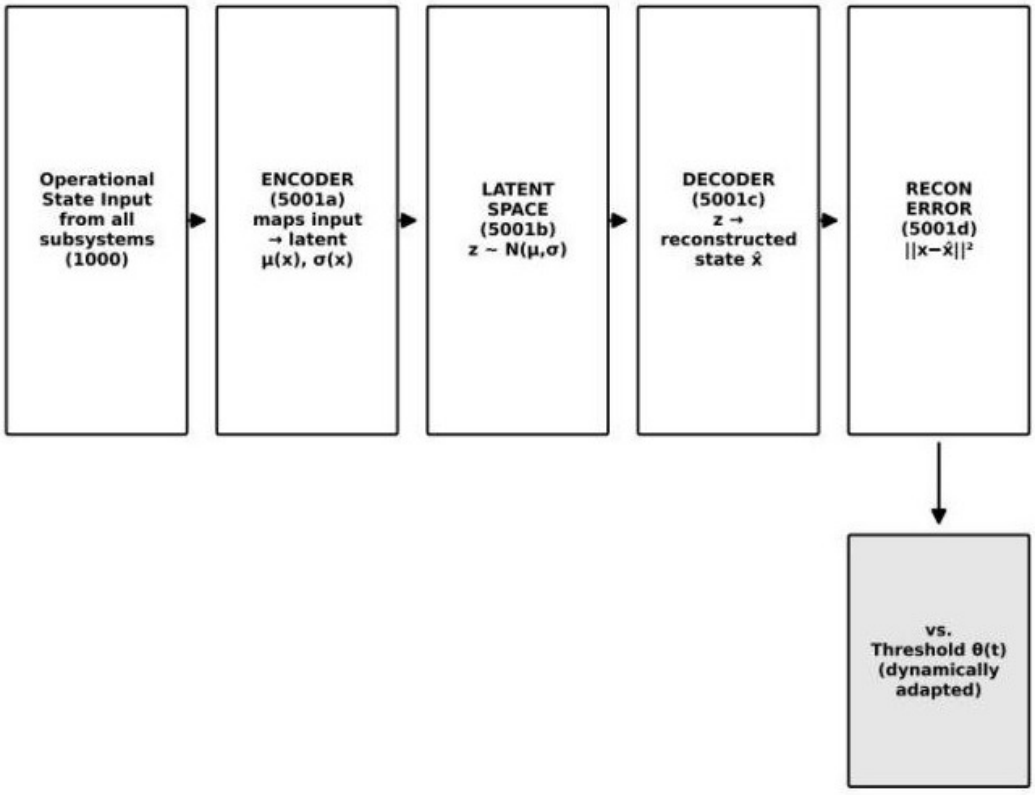
[0022] In the presently preferred embodiment for current fabrication capability, the Quantum Decision Module (4000) is implemented as a deterministic digital quantum-inspired optimization solver fabricated in conventional CMOS logic within the hardware trust boundary (1001). The digital solver executes the identical QUBO formulation min E(s) = Σh_i·s_i + ΣJ_ij·s_i·s_j of FIG. 8 by means of a parallel array of fixed-function spin-update units, an on-chip coupling-matrix memory storing the h_i and J_ij coefficients, and a deterministic annealing schedule driven by a fixed-seed pseudo-random sequence generator. Because the schedule, the seed, and the coefficient memory are fixed at the start of each execution, the solver is invariant across successive executions given identical input state — the invariance property of the Decision Module is preserved without physical quantum hardware. The digital solver requires no cryogenic apparatus, no qubit substrate, and no fabrication step beyond conventional CMOS, and is fully integrable within the power envelopes of FIG. 23.

[0023] The physical quantum embodiment — the Variational Quantum Eigensolver (4001) and quantum annealing hardware (4002) of FIGS. 6 through 8 — is an alternative embodiment for future process capability. Selection between the digital quantum-inspired embodiment and the physical quantum embodiment does not alter any other subsystem: the binary anomaly determination (5002), the Fault Self-Healing Module sequence (6000), the checkpoint rollback (3003), the Security Engine (7000), and the Oversight Interface (8001) operate identically in both embodiments. The invariance of the fixed optimization process, not the physical substrate executing it, is the architectural property on which the behavioral guarantees of the circuit rest.

VAE Anomaly Detection Architecture — FIG. 9

[0024] Referring to FIG. 9, the VAE Anomaly Detection Architecture (5000) comprises an encoder network (5001a), a latent space (5001b), a decoder network (5001c), and a reconstruction error module (5001d). The encoder maps operational state input from all subsystems (1000) to a latent representation z ~ N(μ,σ). The decoder reconstructs the input from z, producing reconstructed state x̂. The reconstruction error ||x − x̂||² is compared against a dynamic threshold θ(t). The VAE loss function is Loss = −E_{q(z|x)}[log p(x|z)] + KL(q(z|x) || p(z)), minimized during on-chip training cycles. The VAE operates continuously as a concurrent side-channel process without interrupting chip operation.

FIG. 9 — VAE Anomaly Detection Architecture (5000)



$$\text{Loss} = -E_{q(z|x)}[\log p(x|z)] + KL(q(z|x) \,||\, p(z))$$

Minimized during on-chip training cycles

VAE operates continuously as a concurrent side-channel process — does not interrupt chip operation

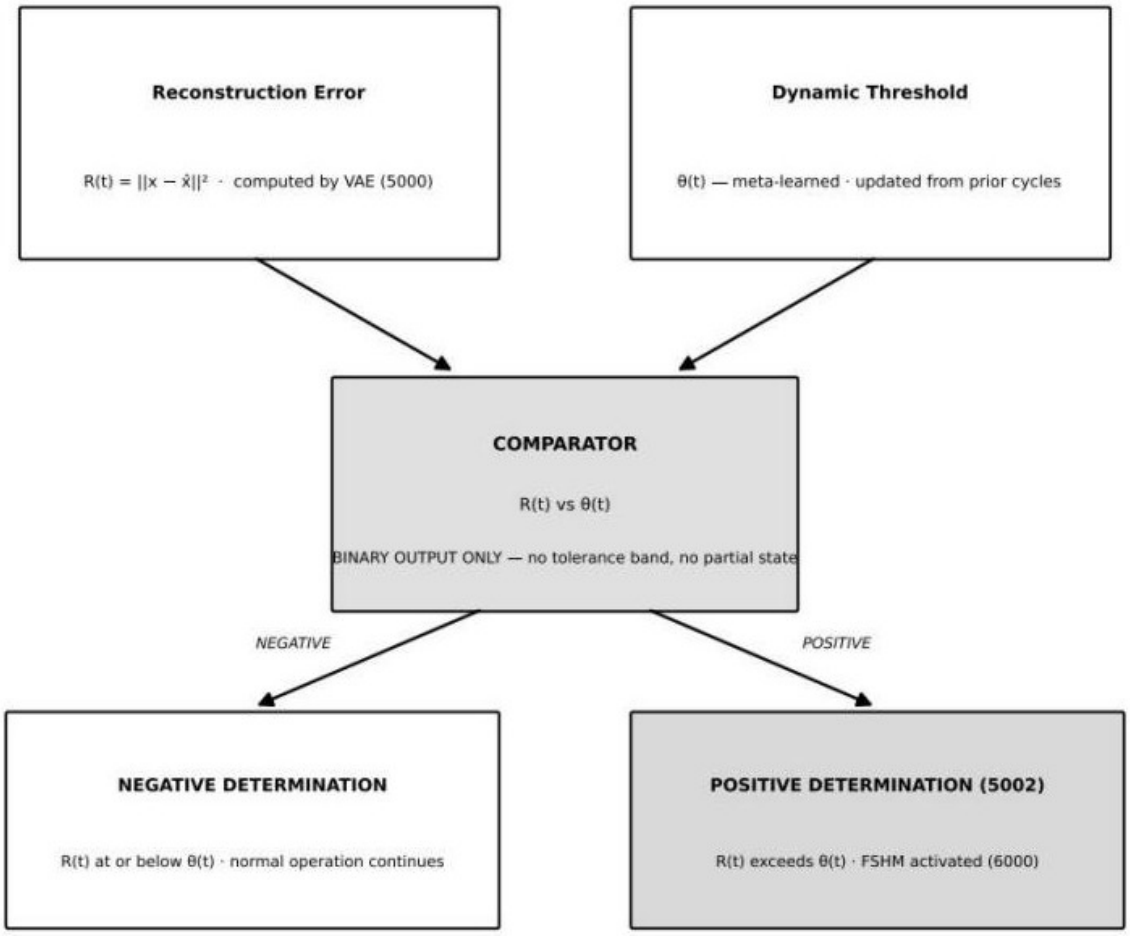
FIG. 9

FIG. 9

Reconstruction Error and Threshold Model — FIG. 10

[0025] Referring to FIG. 10, the Reconstruction Error and Threshold Model (5001) illustrates the binary comparator architecture. The reconstruction error $R(t) = ||x - \hat{x}||^2$ is computed by the VAE (5000). The dynamic threshold $\theta(t)$ is meta-learned and updated from prior execution cycles. The comparator evaluates $R(t)$ versus $\theta(t)$ and produces a binary output only — no tolerance band, no partial state. When $R(t)$ is at or below $\theta(t)$, a NEGATIVE determination is produced and normal operation continues. When $R(t)$ exceeds $\theta(t)$, a POSITIVE determination (5002) is produced and the Fault Self-Healing Module (6000) is activated. No partial anomaly state is defined. The POSITIVE determination (5002) is the exclusive trigger for Fault Self-Healing Module activation (6000).

FIG. 10 — Reconstruction Error and Threshold Model (5001)



NO PARTIAL ANOMALY STATE DEFINED — output is binary, always — no intermediate value, no tolerance, no override

POSITIVE determination (5002) is the EXCLUSIVE trigger for Fault Self-Healing Module activation (6000)

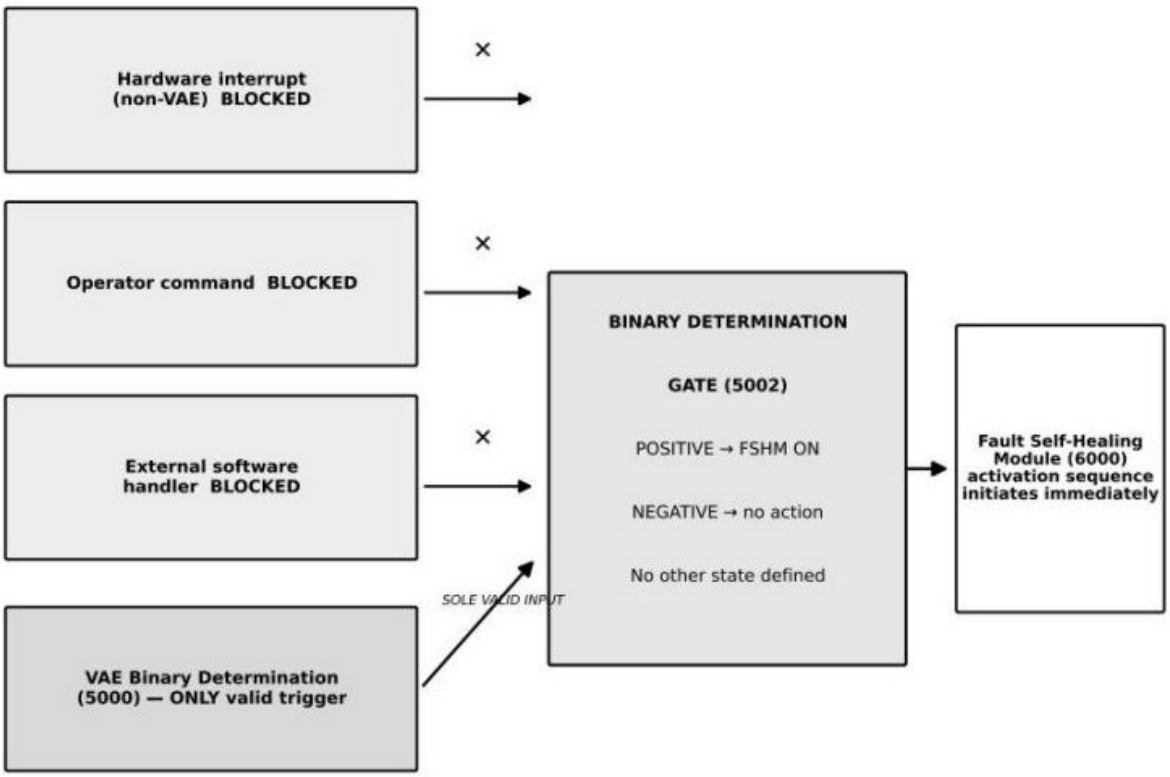
FIG. 10

FIG. 10

Binary Anomaly Determination as Exclusive FSHM Trigger — FIG. 11

[0026] Referring to FIG. 11, the Binary Anomaly Determination as Exclusive FSHM Trigger (5002) illustrates the architectural prohibition on all alternative trigger sources. Hardware interrupts from non-VAE sources, operator commands, and external software handlers are each structurally blocked from accessing the Binary Determination Gate (5002). The VAE Binary Determination (5000) is the sole valid input. The gate produces POSITIVE → FSHM ON or NEGATIVE → no action. No other state is defined. The Fault Self-Healing Module (6000) activation sequence initiates immediately upon POSITIVE determination. No external signal, software handler, or operator command may activate or suppress the FSHM independently of the binary determination (5002).

FIG. 11 — Binary Anomaly Determination as Exclusive FSHM Trigger (5002)



ARCHITECTURAL PROHIBITION: No external signal, software handler, or operator command

may activate OR suppress the FSHM independently of the binary determination (5002)

FIG. 11

FIG. 11

Fault Self-Healing Module Non-Reorderable Sequence — FIG. 12

[0027] Referring to FIG. 12, the Fault Self-Healing Module Non-Reorderable Sequence (6000) comprises five steps each gated by completion of the preceding step. Step 1 (6001) Anomaly Confirmation: the positive binary determination (5002) is received and validated. Step 2 (6002) Workload Suspension: all active spike routing to the anomalous core is suspended. Step 3 (6003) Redundant Core Activation: designated redundant cores are brought to full operational state. Step 4 (6004) Workload Transfer: routing tables are remapped and all traffic redirected to redundant cores. Step 5 (6005) Anomalous Core Isolation and Rollback: the anomalous core is removed from the active topology and checkpoint rollback is executed. The Type Gate establishes output types: 6001 → confirmed anomaly record; 6002 → suspended workload map; 6003 → active redundant core set; 6004 → completed transfer record; 6005 → verified stable state. Reordering any two steps constitutes an undefined operation because the step receives an input type that has not yet been produced. The sequence is structurally non-reorderable.

FIG. 12 — Fault Self-Healing Module — Non-Reorderable Sequence (6000)

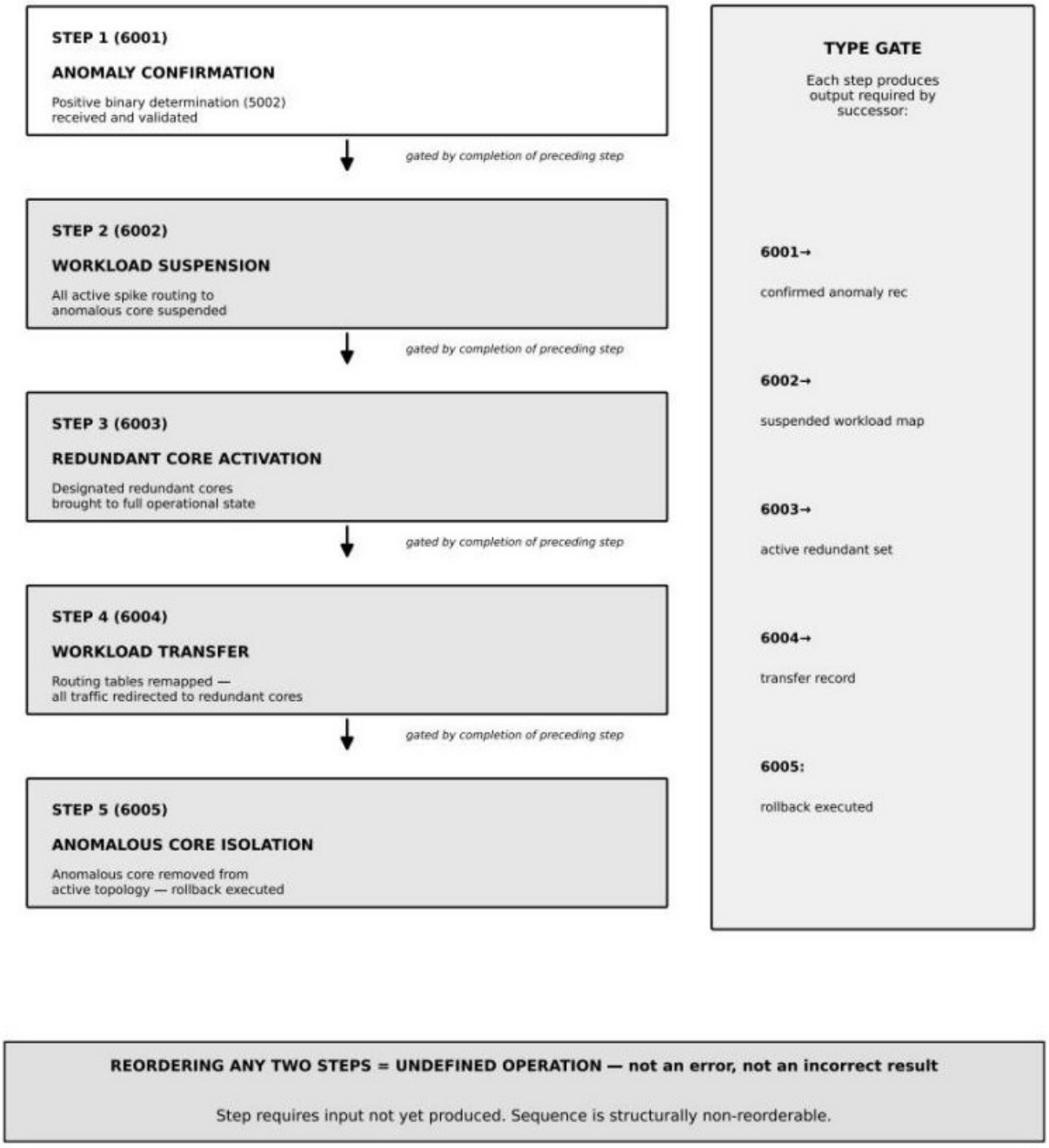


FIG. 12

FIG. 12

Redundant Core Rerouting — FIG. 13

[0028] Referring to FIG. 13, the Redundant Core Rerouting (6006) diagram illustrates the workload transfer from an anomalous core to an activated redundant core. Active SNN Cores include Core 1 (2000a), Core 2 (2000b), Core 3 (2000c) designated ANOMALOUS, and Core 4 (2000d). Redundant SNN Cores (6006) include Redundant 1 through Redundant 4, with Redundant 3 activated to replace the anomalous Core 3. The Spike Routing Tables (2001) are remapped at Step 6004 to redirect all traffic from the anomalous core to the activated redundant core. Topology preservation is enforced: the activated redundant core mirrors the asynchronous messaging topology of the isolated anomalous core exactly. Step 6004 completes before Step 6005 isolates the anomalous core.

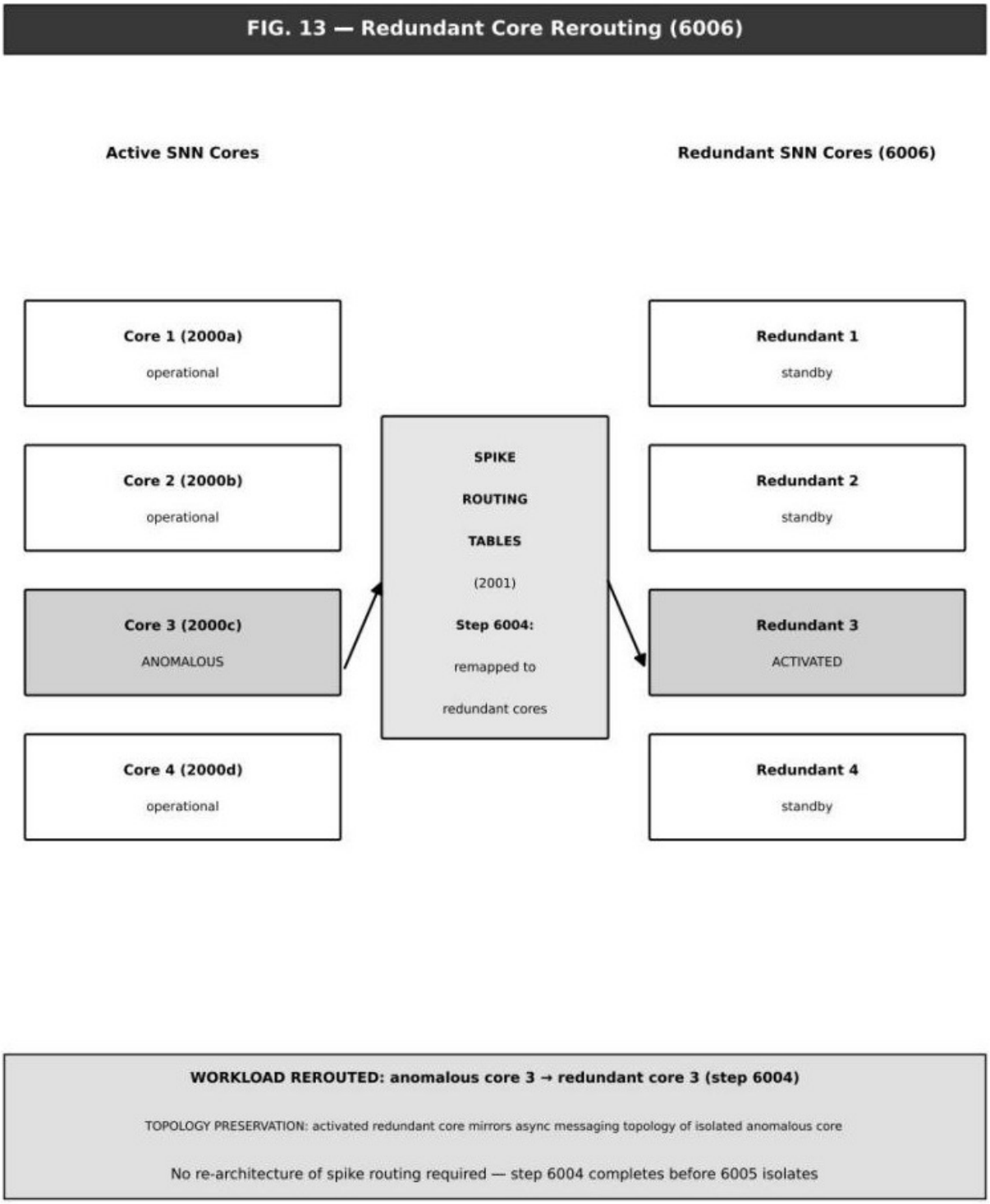


FIG. 13 — Redundant Core Rerouting (6006)
Active SNN Cores
Redundant SNN Cores (6006)
Core 1 (2000a)
operational
Core 2 (2000b)
operational
Core 3 (2000c)
ANOMALOUS
Core 4 (2000d)
operational
SPIKE
ROUTING
TABLES
(2001)
Step 6004:
remapped to
redundant cores
Redundant 1
standby
Redundant 2
standby
Redundant 3
ACTIVATED
Redundant 4
standby
WORKLOAD REROUTED: anomalous core 3 → redundant core 3 (step 6004)
TOPOLOGY PRESERVATION: activated redundant core mirrors async messaging topology of isolated anomalous core
No re-architecture of spike routing required — step 6004 completes before 6005 isolates

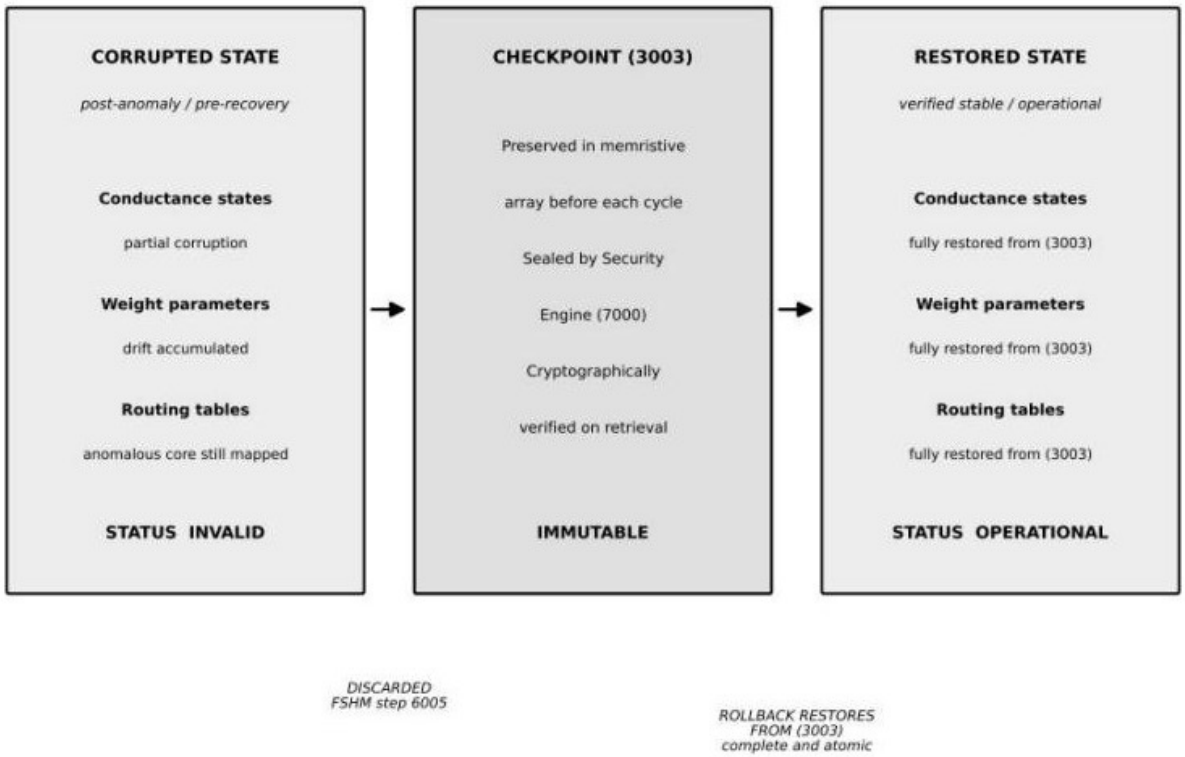
FIG. 13

FIG. 13

Rollback to Last Verified Stable State — FIG. 14

[0029] Referring to FIG. 14, the Rollback to Last Verified Stable State (3003) illustrates the three-phase correction operation. The Corrupted State presents conductance states with partial corruption, weight parameters with accumulated drift, and routing tables with the anomalous core still mapped — status INVALID. The corrupted state is discarded by FSHM Step 6005. The Checkpoint (3003), preserved in the memristive array before each cycle and cryptographically sealed by the Security Engine (7000), is verified on retrieval — status IMMUTABLE. The Rollback operation restores from the checkpoint completely and atomically, producing a Restored State with all conductance states, all weight parameters, and all routing tables fully restored — status OPERATIONAL. Rollback is the only correction operation.

FIG. 14 — Rollback to Last Verified Stable State (3003)



NO PATCHING DEFINED — corrupted conductance state cannot be partially corrected in-place

ROLLBACK IS THE ONLY CORRECTION OPERATION — full checkpoint restoration from (3003), always

FIG. 14

FIG. 14

Non-Reorderable Self-Healing Sequence Structural Proof — FIG. 15

[0030] Referring to FIG. 15, the Non-Reorderable Self-Healing Sequence Structural Proof illustrates four reordering violations each producing an undefined operation. The correct sequence is 6001 → 6002 → 6003 → 6004 → 6005. Violation 1: 6003 before 6002 — active workload routes to the redundant core before transfer mapping is defined → undefined operation. Violation 2: 6004 before 6003 — routing tables remap to cores not yet operational → undefined operation. Violation 3: 6002 before 6001 — workloads suspended with no confirmed anomaly record as input → undefined operation. Violation 4: 6005 before 6004 — anomalous core isolated while workloads still route to it → undefined operation. Each reordering produces an undefined operation — not a wrong result, not an error — the sequence is structurally non-reorderable.

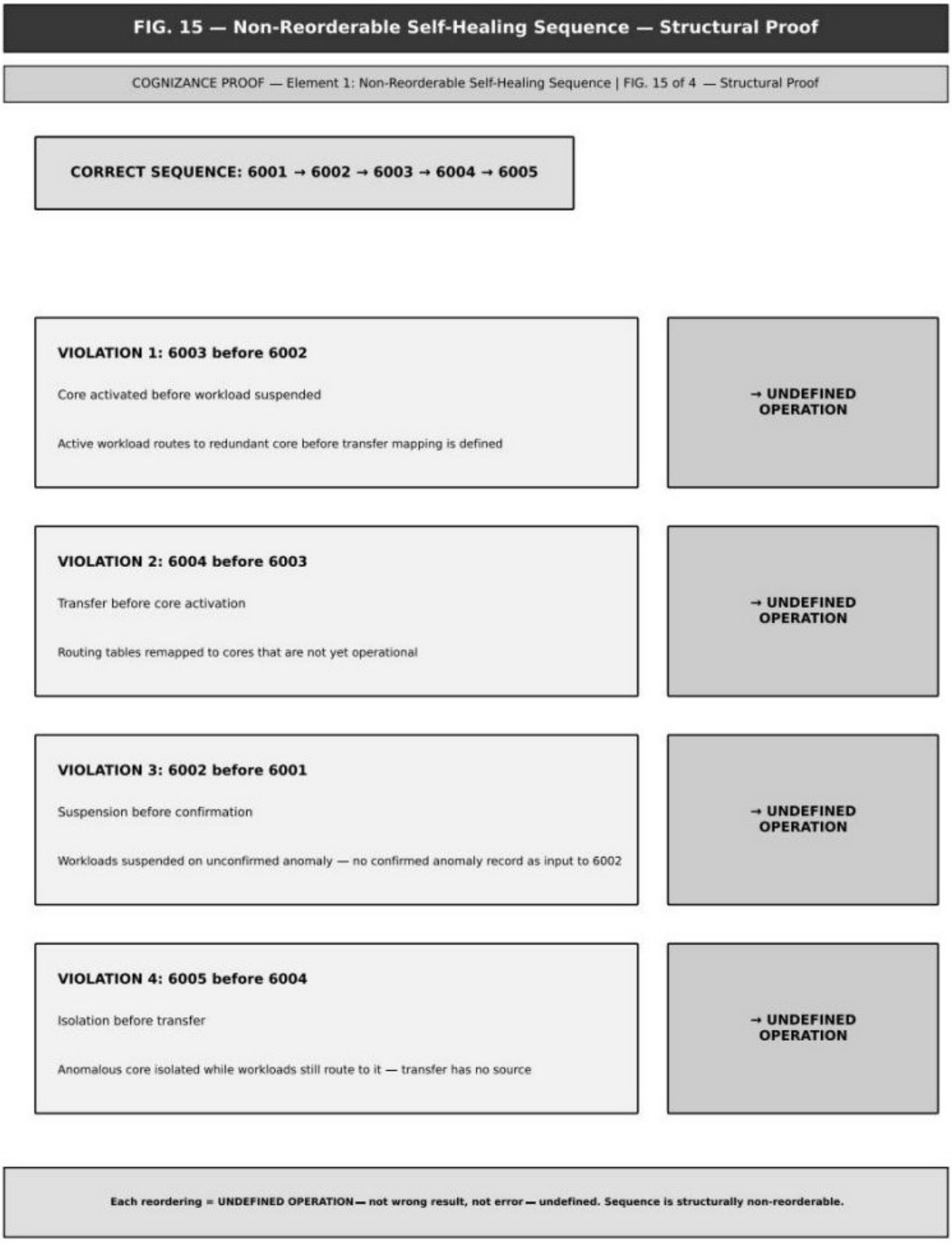


FIG. 15 — Non-Reorderable Self-Healing Sequence — Structural Proof
COGNIZANCE PROOF — Element 1: Non-Reorderable Self-Healing Sequence | FIG. 15 of 4 — Structural Proof
CORRECT SEQUENCE: 6001 → 6002 → 6003 → 6004 → 6005
VIOLATION 1: 6003 before 6002
Core activated before workload suspended
Active workload routes to redundant core before transfer mapping is defined
→ UNDEFINED OPERATION
VIOLATION 2: 6004 before 6003
Transfer before core activation
Routing tables remapped to cores that are not yet operational
→ UNDEFINED OPERATION
VIOLATION 3: 6002 before 6001
Suspension before confirmation
Workloads suspended on unconfirmed anomaly — no confirmed anomaly record as input to 6002
→ UNDEFINED OPERATION
VIOLATION 4: 6005 before 6004
Isolation before transfer
Anomalous core isolated while workloads still route to it — transfer has no source
→ UNDEFINED OPERATION
Each reordering = UNDEFINED OPERATION — not wrong result, not error — undefined. Sequence is structurally non-reorderable.

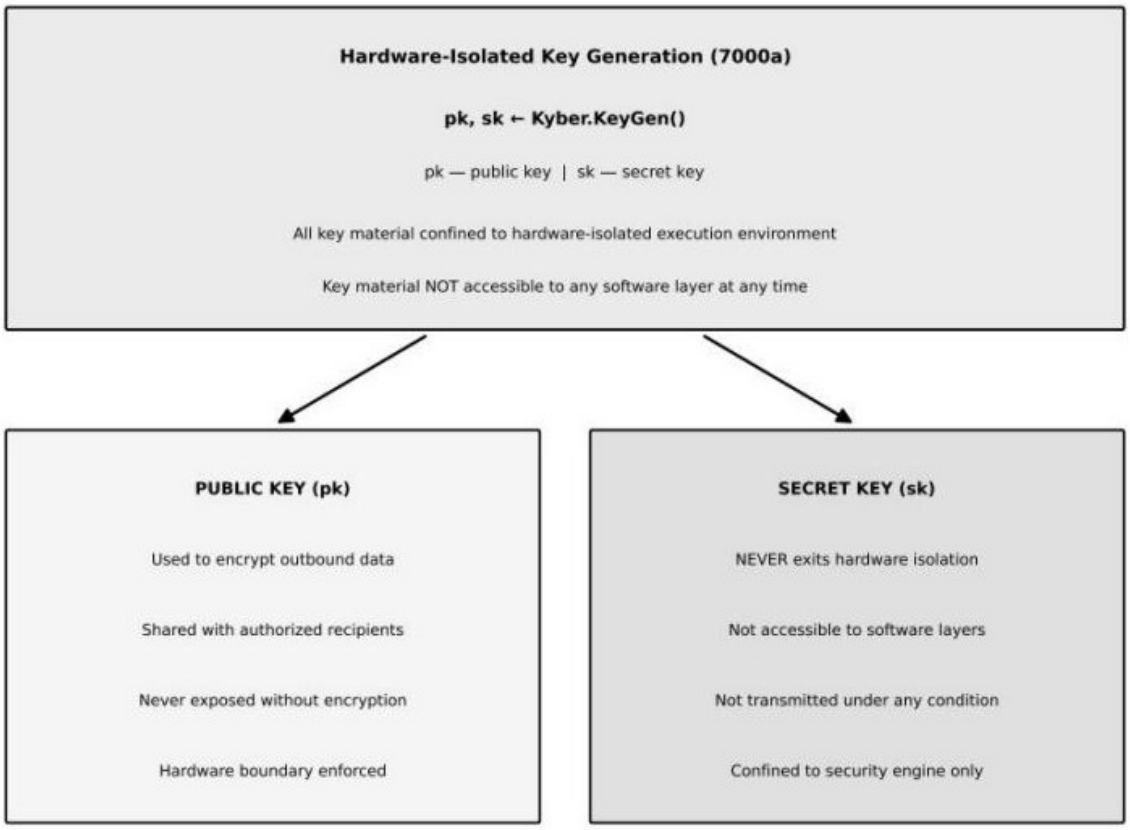
FIG. 15

FIG. 15

CRYSTALS-Kyber Key Generation — FIG. 16

[0031] Referring to FIG. 16, the CRYSTALS-Kyber Key Generation diagram (7000) illustrates the hardware-isolated key generation architecture (7000a). The key pair (pk, sk) is generated by Kyber.KeyGen() within a hardware-isolated execution environment inaccessible to any software layer. The public key (pk) is used to encrypt outbound data and is shared with authorized recipients. The secret key (sk) never exits hardware isolation, is not accessible to software layers, is not transmitted under any condition, and is confined to the security engine only. All data is encrypted before crossing the hardware boundary (1001) — no unencrypted outbound data is defined.

FIG. 16 — CRYSTALS-Kyber Key Generation (7000)



ENCRYPTION SCOPE: Learning state · Inference outputs · Anomaly telemetry · Audit records

ALL encrypted before crossing hardware boundary (1001) — no unencrypted outbound data defined

FIG. 16

FIG. 16

Hardware-Isolated Security Boundary — FIG. 17

[0032] Referring to FIG. 17, the Hardware-Isolated Security Boundary (1001 / 7000) illustrates the chip interior containing all subsystems operating with unencrypted data within the boundary. The Security Engine (7000) functions as the encryption enforcement point through which all data passes before exiting the chip boundary. External systems — Cloud, Edge, On-Premises, Audit Archive, and Supervisory UI — receive encrypted packets only, with no access to raw chip state.

FIG. 17 — Hardware-Isolated Security Boundary (1001 / 7000)

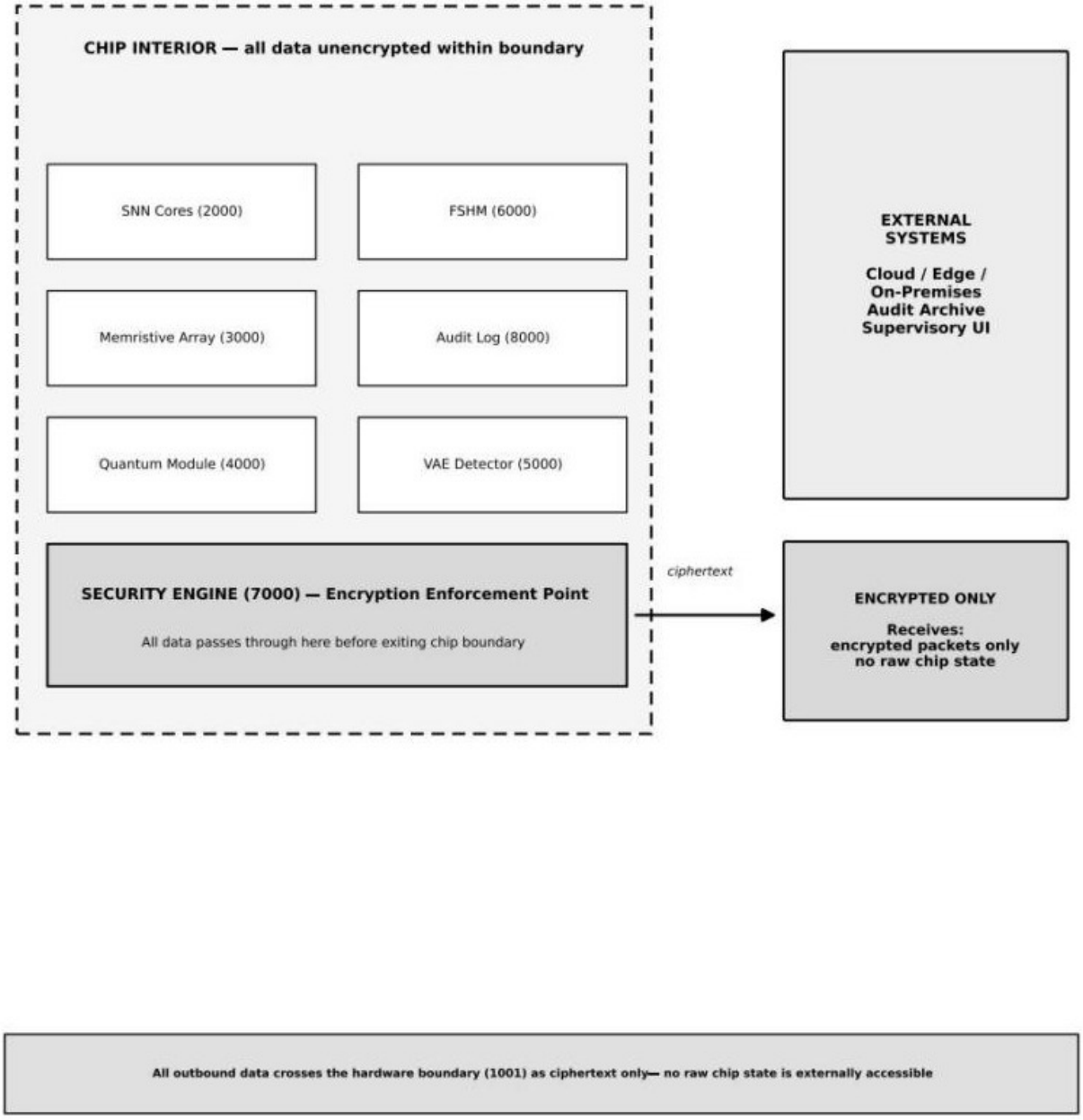


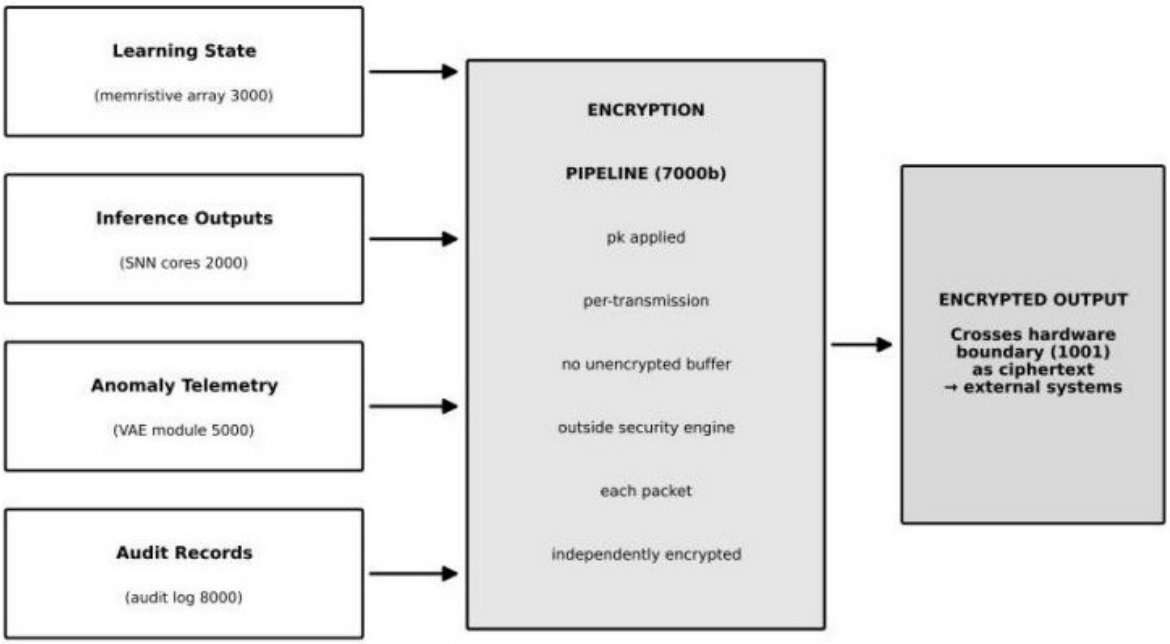
FIG. 17

FIG. 17

Encrypted Telemetry Flow — FIG. 18

[0033] Referring to FIG. 18, the Encrypted Telemetry Flow (7000) illustrates the four data streams entering the Encryption Pipeline (7000b): Learning State from the memristive array (3000), Inference Outputs from the SNN Cores (2000), Anomaly Telemetry from the VAE module (5000), and Audit Records from the Audit Log (8000). The Encryption Pipeline applies the public key per-transmission with no unencrypted buffer outside the security engine. The encrypted output crosses the hardware boundary (1001) as ciphertext. Unencrypted chip-internal state is not accessible outside the hardware boundary at any point in the operational sequence, including during fault self-healing events.

FIG. 18 — Encrypted Telemetry Flow (7000)



GUARANTEE: Unencrypted chip-internal state is not accessible outside the hardware boundary

at any point in the operational sequence — including during fault self-healing events

FIG. 18

FIG. 18

Immutable Audit Log — FIG. 19

[0034] Referring to FIG. 19, the Immutable Audit Log (8000) illustrates the append-only event record structure comprising E001 Anomaly Detection Event, E002 FSHM Activation, E003 Workload Rerouting, E004 Rollback Execution, E005 Quantum Decision Output, and E006 Checkpoint Creation. Each entry is sealed by the Security Engine (7000) as an HMAC over the entry using the hardware key at recording time. Post-seal modification is an undefined operation. Deletion is an undefined operation. The log is append-only — write-once with no overwrite, no delete, and no edit defined. Read-only access is provided via the Oversight Interface (8001).

FIG. 19 — Immutable Audit Log (8000)

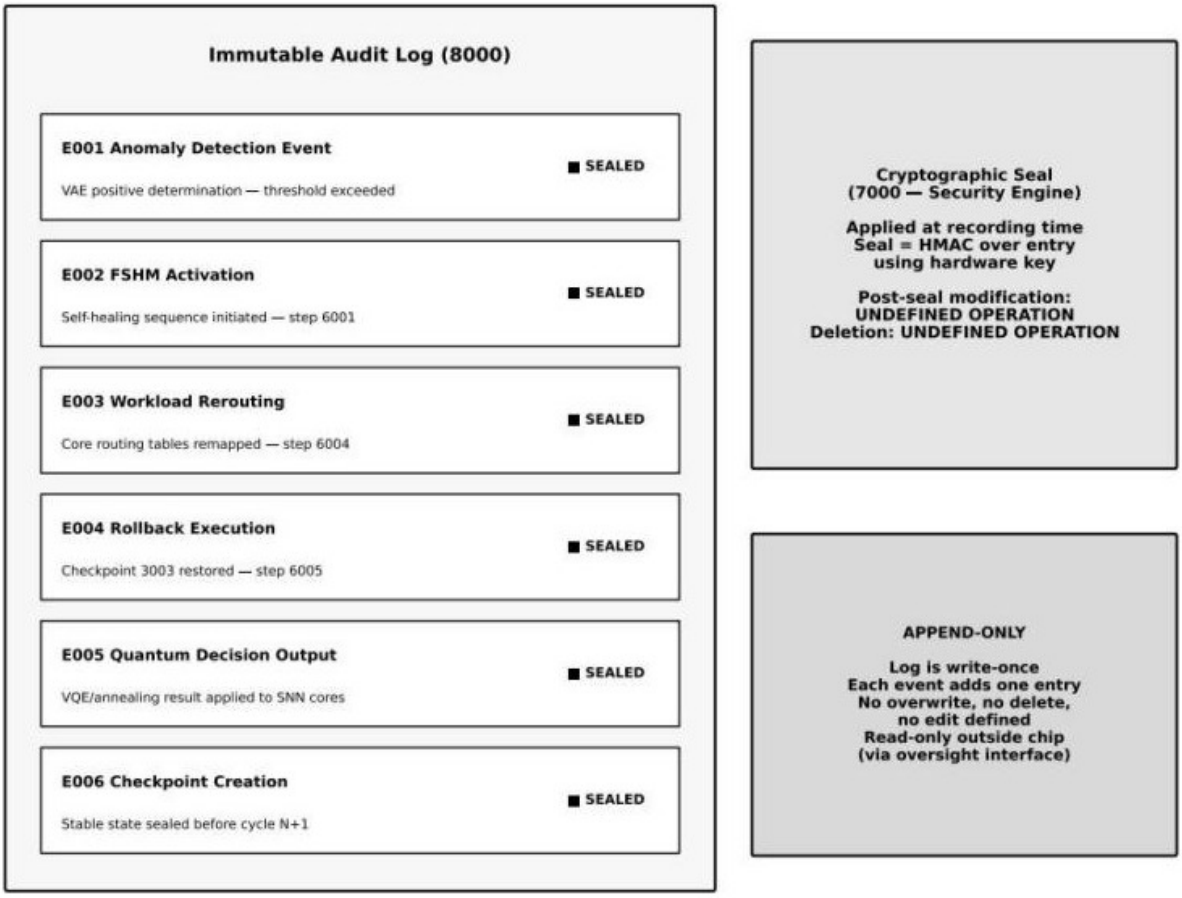


FIG. 19

FIG. 19

Human-in-the-Loop Oversight Interface — FIG. 20

[0035] Referring to FIG. 20, the Human-in-the-Loop Oversight Interface (8001) illustrates the read-only boundary through which the chip's operational history is exposed to external governance. The chip interior supplies five data streams to the Read-Only Interface (8001): Anomaly Determinations (5002), Rollback Events (3003), FSHM Sequence Records (6001–6005), Quantum Decision Outputs (4000), and Audit Log Entries (8000, cryptographically sealed). No commands are accepted. The chip observes, structures, and exposes — it does not accept external commands that modify its operational sequence. All authority boundaries are fixed at hardware architecture time, not configurable at runtime.

FIG. 20 — Human-in-the-Loop Oversight Interface (8001)

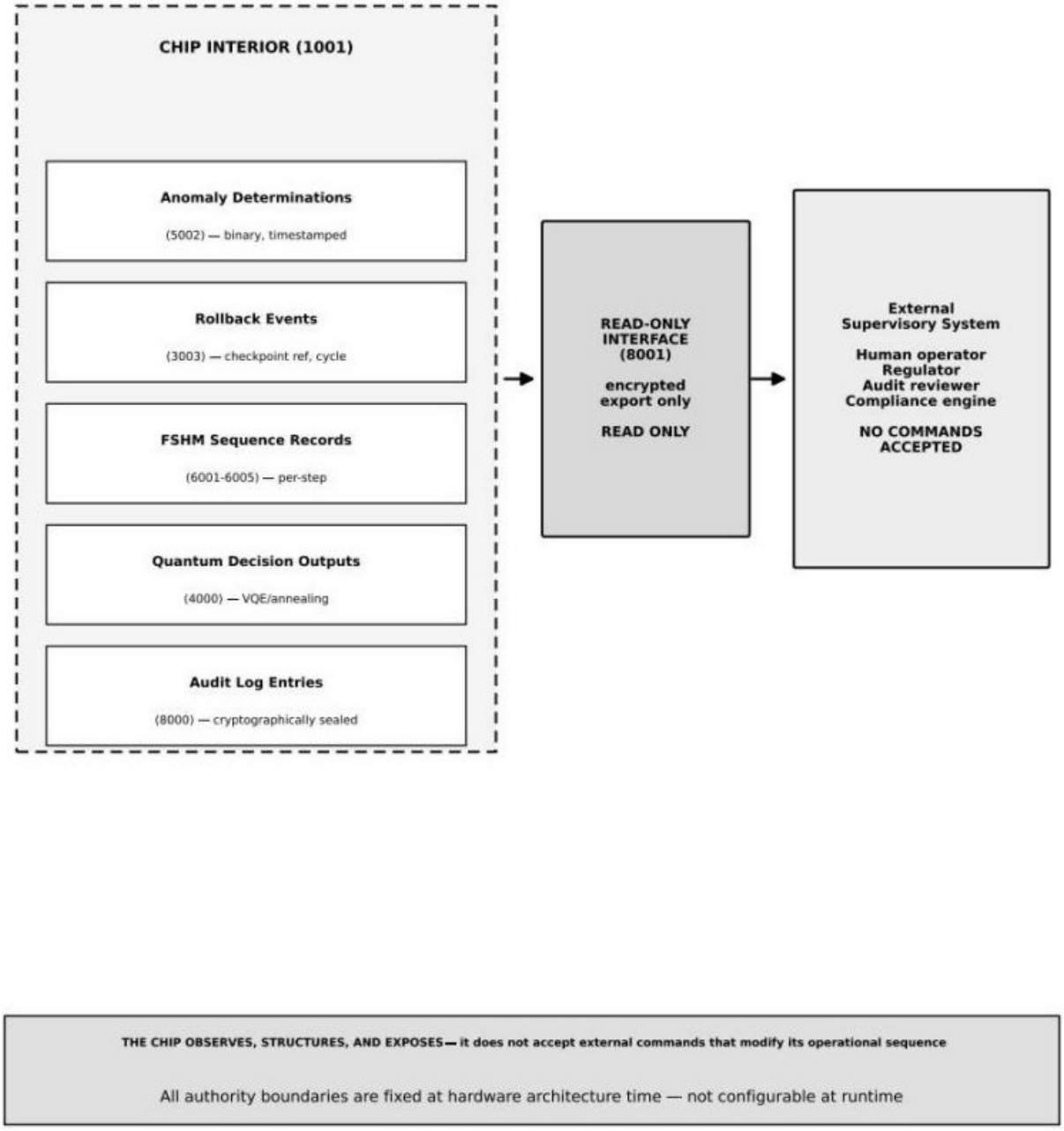


FIG. 20

FIG. 20

Full Integrated System Flow — Best Mode — FIG. 21

[0036] Referring to FIG. 21, the Full Integrated System Flow — Best Mode (1000) illustrates the complete operational sequence. Input signals arrive at the SNN Cores (2000). STDP execution updates weights in the Memristive Array (3000). The Quantum Decision Module (4000) executes VQE and annealing to guide pathway weighting. The VAE (5000) monitors the spike stream continuously. The binary gate evaluates the VAE determination: POSITIVE activates the FSHM (6000) sequence 6001→6002→6003→6004→6005 with rollback from checkpoint (3003); NEGATIVE allows the cycle to continue. Concurrently, the Security Engine (7000) encrypts all outbound data, the Audit Log (8000) seals and appends every event, the checkpoint (3003) seals the stable state before each cycle, and the Oversight Interface (8001) exports the read-only record. The best mode specification: 128 SNN processing cores; 1.2 GHz adaptive clock; 8 MB short-term and 64 MB long-term memristive arrays; 16 designated redundant cores; VQE plus annealing quantum module; on-chip VAE anomaly detector; CRYSTALS-Kyber hardware-isolated encryption; under 10 Watts peak power; cloud, edge, and on-premises deployment configurations; immutable sealed audit log; read-only oversight interface. The best mode contemplated for current fabrication capability is the deterministic digital quantum-inspired solver configuration of the Decision Module (4000); the physical quantum module configuration is contemplated for future process capability.

FIG. 21 — Full Integrated System Flow — Best Mode (1000)

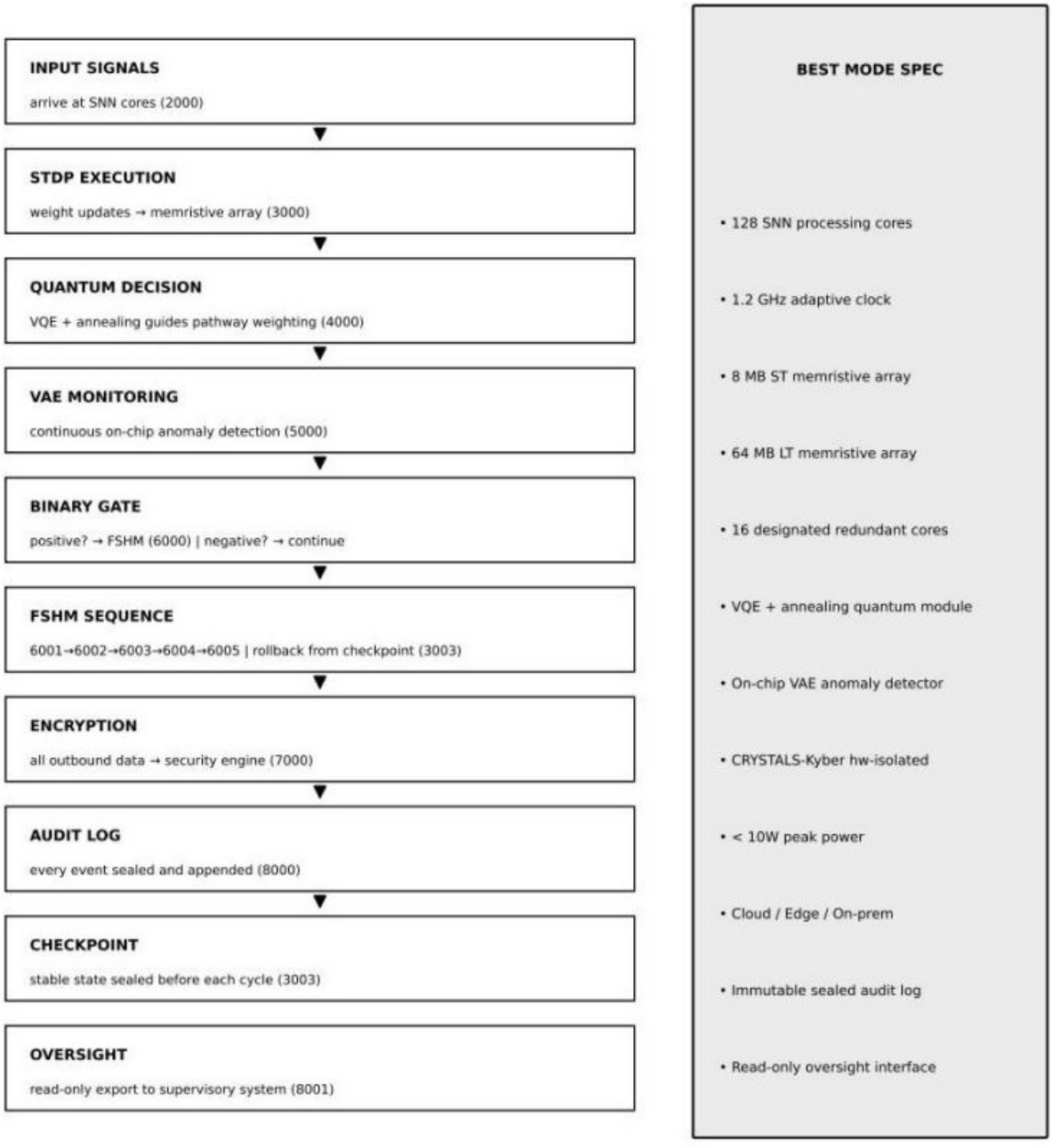


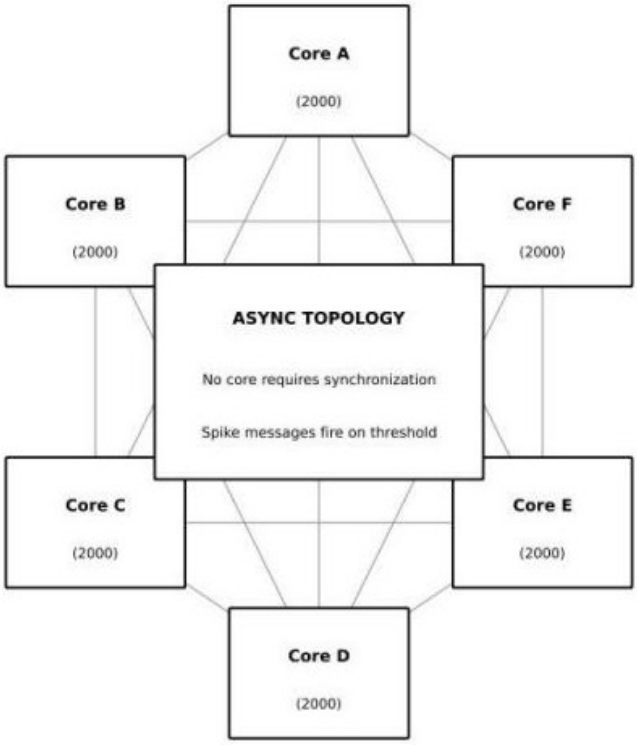
FIG. 21

FIG. 21

Asynchronous Spike Messaging Topology — FIG. 22

[0037] Referring to FIG. 22, the Asynchronous Spike Messaging Topology (2001) illustrates the mesh topology connecting SNN Cores A through F (2000). No core requires synchronization with any other core — spike messages fire on threshold. During FSHM Rerouting (6004), the activated redundant core mirrors the anomalous core's asynchronous messaging topology exactly. No re-architecture of spike routing is required. Spike messages continue firing on threshold without interruption during and after transfer.

FIG. 22 — Asynchronous Spike Messaging Topology (2001)



TOPOLOGY PRESERVATION DURING FSHM REROUTING (6004)

Async messaging topology of redundant core mirrors anomalous core exactly — no re-architecture required

FIG. 22

FIG. 22

Deployment Configurations — FIG. 23

[0038] Referring to FIG. 23, the Deployment Configurations (1000) illustrates three deployment profiles. Cloud Integration: 128 SNN cores, full VQE plus annealing, encrypted output to cloud archive. Edge Deployment: 32 SNN cores, 4 redundant cores, annealing only, under 3 Watts peak, all security active. On-Premises: configurable core count, full quantum module, local audit archive. The invariant across all configurations: hardware boundary, security engine, FSHM, and VAE anomaly detection are active in every configuration. Deployment configuration does not modify the hardware boundary, the encryption enforcement point, or the self-healing sequence.

FIG. 23 — Deployment Configurations (1000)

Element	Cloud	Edge	On-Prem
SNN Cores	128	32	configurable
Redundant Cores	16	4	proportional
VQE	active	disabled	active
Quantum Annealing	active	active	active
Security Engine	active	active	active
FSHM	active	active	active
Audit Log	cloud export	local	local
Peak Power	< 10W	< 3W	< 10W

INVARIANT ACROSS ALL CONFIGURATIONS: hardware boundary · security engine · FSHM · VAE anomaly detection

Deployment configuration does NOT modify the hardware boundary, encryption enforcement point, or self-healing sequence

FIG. 23

FIG. 23

Prior Art Isolation vs. NLIC Integration — FIG. 24

[0039] Referring to FIG. 24, the Prior Art Isolation versus NLIC Integration diagram illustrates five prior art systems each providing one of the NLIC's five functions in isolation: IBM TrueNorth and Intel Loihi provide SNN capability without on-chip anomaly detection or rollback; Infosys and Bank of America provide anomaly detection as a software layer external to the chip; UiPath, Dell, and Microsoft provide self-healing at the software or network level without hardware-level recovery; Microsoft, IBM, and D-Wave provide quantum solvers as isolated processors not integrated with SNN learning; and Amazon and Capital One provide post-quantum encryption at the network layer rather than at the hardware boundary. The NLIC novelty is the integration of all five functions in a single chip architecture. No prior art combines adaptive SNN learning, on-chip anomaly detection, autonomous hardware recovery, quantum decision integration, and hardware-boundary encryption.


FIG. 24 — Prior Art Isolation vs. NLIC Integration
PRIOR ART — ISOLATED FUNCTIONS
NLIC INTEGRATED AT CHIP LEVEL (1000)
IBM TrueNorth / Intel Loihi
SNN chip — no on-chip anomaly detection, no rollback
SNN + STDP + Gradient Refinement
Adaptive learning in hardware — meta-learned, on-chip
Infosys / Bank of America
Anomaly detection — software layer, external to chip
On-Chip VAE Anomaly Detection
Binary determination at chip speed — not software latency
UiPath / Dell / Microsoft
Self-healing — software/network level, no hardware recovery
FSHM — Hardware-Level Self-Healing
Non-reorderable autonomous recovery — no external handler
Microsoft / IBM / D-Wave
Quantum solvers — isolated processors, not integrated with SNN learning
Quantum Decision Module
VQE + annealing integrated with SNN pathway weighting
Amazon / Capital One
Post-quantum encryption — network layer, not at hardware boundary
CRYSTALS-Kyber at Hardware Boundary
Encryption enforced in hardware — not network layer
NLIC NOVELTY: Integration of all five functions in a single chip architecture— not five chips, not five software modules
No prior art combines adaptive SNN learning + on-chip anomaly detection + autonomous recovery + quantum integration + hw-boundary encryption

FIG. 24

FIG. 24

Non-Reorderable Self-Healing Sequence — Type Contract Principle — FIG. 25

[0040] Referring to FIG. 25, the Type Contract Definition proof for Element 1 illustrates the five-step type chain. Step 6001 receives binary_determination(5002) as input and produces confirmed_anomaly_record as output. Step 6002 receives confirmed_anomaly_record and produces suspended_workload_map. Step 6003 receives suspended_workload_map and produces active_redundant_core_set. Step 6004 receives active_redundant_core_set and produces completed_transfer_record. Step 6005 receives completed_transfer_record and produces verified_stable_state. The Type Equivalence Principle: the output type of step N equals the input type of step N+1, defined at architecture time. Reordering step N and N+1 presents the wrong output type as input, producing a type mismatch and therefore an undefined operation. The sequence is non-reorderable by structural necessity — not by convention, rule, or runtime constraint.

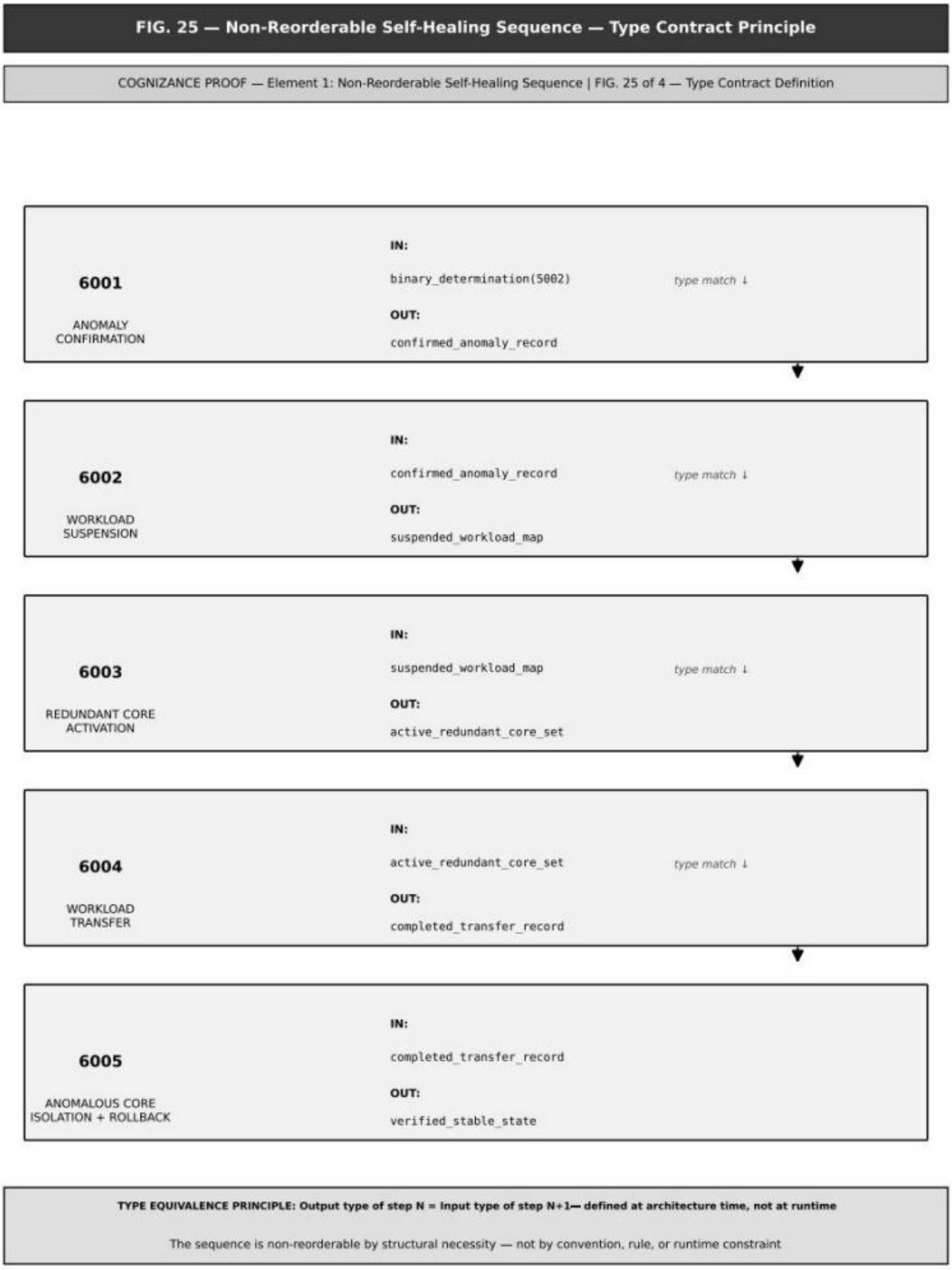


FIG. 25 — Non-Reorderable Self-Healing Sequence — Type Contract Principle
COGNIZANCE PROOF — Element 1: Non-Reorderable Self-Healing Sequence | FIG. 25 of 4 — Type Contract Definition
6001
ANOMALY CONFIRMATION
IN:
binary_determination(5002)
type match ↓
OUT:
confirmed_anomaly_record
6002
WORKLOAD SUSPENSION
IN:
confirmed_anomaly_record
type match ↓
OUT:
suspended_workload_map
6003
REDUNDANT CORE ACTIVATION
IN:
suspended_workload_map
type match ↓
OUT:
active_redundant_core_set
6004
WORKLOAD TRANSFER
IN:
active_redundant_core_set
type match ↓
OUT:
completed_transfer_record
6005
ANOMALOUS CORE ISOLATION + ROLLBACK
IN:
completed_transfer_record
OUT:
verified_stable_state
TYPE EQUIVALENCE PRINCIPLE: Output type of step N = Input type of step N+1— defined at architecture time, not at runtime
The sequence is non-reorderable by structural necessity — not by convention, rule, or runtime constraint

FIG. 25

FIG. 25

Non-Reorderable Self-Healing Sequence — Reordering Violation Proof — FIG. 26

[0041] Referring to FIG. 26, the Reordering Violation Proof for Element 1 establishes four violations each producing an undefined operation. Violation 1: 6003 before 6002 — 6003 requires suspended_workload_map; receives confirmed_anomaly_record → type mismatch → undefined. Violation 2: 6004 before 6003 — 6004 requires active_redundant_core_set; receives suspended_workload_map → undefined. Violation 3: 6002 before 6001 — 6002 requires confirmed_anomaly_record; receives binary_determination → undefined. Violation 4: 6005 before 6004 — 6005 requires completed_transfer_record; receives active_redundant_core_set → undefined. Sequence order is a structural property of the type system, not a runtime constraint.


FIG. 26 — Non-Reorderable Self-Healing Sequence — Reordering Violation Proof
COGNIZANCE PROOF — Element 1: Non-Reorderable Self-Healing Sequence | FIG. 26 of 4 — Reordering Violations
VIOLATION 1: 6003 ↔ 6002 — Core Activation before Workload Suspension
Input required by 6003: suspended_workload_map
What 6003 receives instead (from 6001): confirmed_anomaly_record
Type mismatch: confirmed_anomaly_record ≠ suspended_workload_map
→ UNDEFINED OPERATION
VIOLATION 2: 6004 ↔ 6003 — Transfer before Core Activation
Input required by 6004: active_redundant_core_set
What 6004 receives instead (from 6002): suspended_workload_map
Type mismatch: suspended_workload_map ≠ active_redundant_core_set
→ UNDEFINED OPERATION
VIOLATION 3: 6002 ↔ 6001 — Suspension before Confirmation
Input required by 6002: confirmed_anomaly_record
What 6002 receives instead (from 5002): binary_determination
Type mismatch: binary_determination ≠ confirmed_anomaly_record
→ UNDEFINED OPERATION
VIOLATION 4: 6005 ↔ 6004 — Isolation before Transfer
Input required by 6005: completed_transfer_record
What 6005 receives instead (from 6003): active_redundant_core_set
Type mismatch: active_redundant_core_set ≠ completed_transfer_record
→ UNDEFINED OPERATION
All four violations produce UNDEFINED OPERATION— sequence order is a structural property of the type system, not a runtime constraint

FIG. 26

FIG. 26

Scheduler Model vs. Structural Type Graph — Ontological Distinction — FIG. 27

[0042] Referring to FIG. 27, the Scheduler Model versus Structural Type Graph comparison establishes the ontological distinction. In the Scheduler Model used in prior art software failover, step order is a runtime parameter changeable by configuration; reordering produces a wrong result detectable after execution; recovery requires patching or retry. In the Structural Type Graph implemented in the NLIC hardware type system, step order is a structural property of the type system; type equivalence is defined at architecture time; reordering presents the wrong type as input producing an undefined operation with no execution and no defined behavior; there is no recovery because undefined operations have no correction. These are not quantitatively different implementations of the same architecture — they are ontologically distinct.

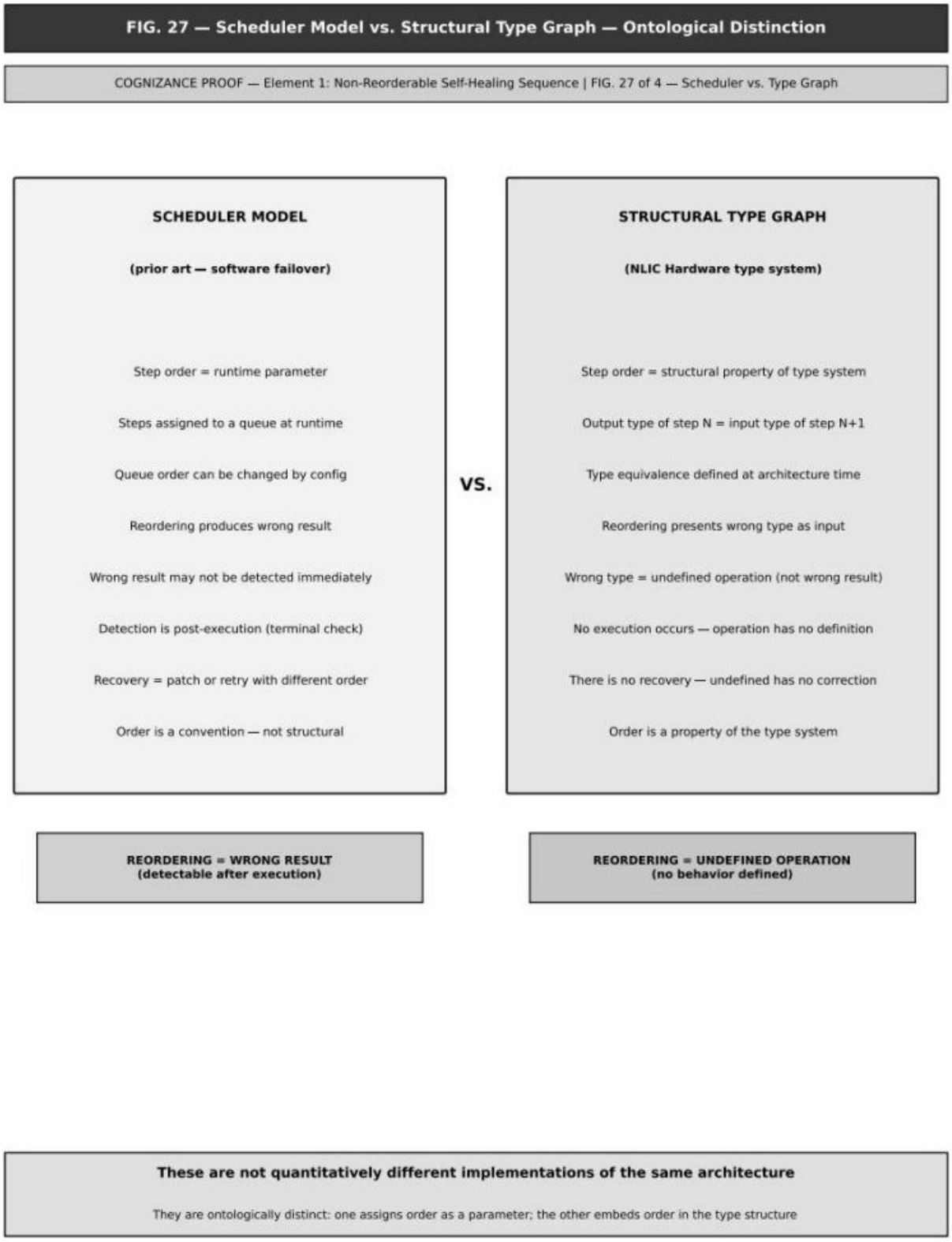


FIG. 27 — Scheduler Model vs. Structural Type Graph — Ontological Distinction
COGNIZANCE PROOF — Element 1: Non-Reorderable Self-Healing Sequence | FIG. 27 of 4 — Scheduler vs. Type Graph
SCHEDULER MODEL
(prior art — software failover)
Step order = runtime parameter
Steps assigned to a queue at runtime
Queue order can be changed by config
Reordering produces wrong result
Wrong result may not be detected immediately
Detection is post-execution (terminal check)
Recovery = patch or retry with different order
Order is a convention — not structural
VS.
STRUCTURAL TYPE GRAPH
(NLIC Hardware type system)
Step order = structural property of type system
Output type of step N = input type of step N+1
Type equivalence defined at architecture time
Reordering presents wrong type as input
Wrong type = undefined operation (not wrong result)
No execution occurs — operation has no definition
There is no recovery — undefined has no correction
Order is a property of the type system
REORDERING = WRONG RESULT
(detectable after execution)
REORDERING = UNDEFINED OPERATION
(no behavior defined)
These are not quantitatively different implementations of the same architecture
They are ontologically distinct: one assigns order as a parameter; the other embeds order in the type structure

FIG. 27

FIG. 27

Self-Healing Sequence Step Completeness Proof — FIG. 28

[0043] Referring to FIG. 28, the Step Completeness Proof establishes the minimum necessary complexity of the five-step sequence. Removing Step 1 (6001): 6002 receives binary_determination rather than confirmed_anomaly_record → type mismatch → undefined. Removing Step 2 (6002): 6003 activates redundant cores with no suspended workload map defining routing entries → undefined. Removing Step 3 (6003): 6004 has no valid transfer destination → undefined. Removing Step 4 (6004): 6005 isolates the anomalous core while active workloads still route to it → undefined. Removing Step 5 (6005): the anomalous core remains in active topology and corrupted state persists with no terminal condition. Every step is required by its successor’s type structure. The sequence is irreducible.

FIG. 28 — Self-Healing Sequence Step Completeness Proof

COGNIZANCE PROOF — Element 1: Non-Reorderable Self-Healing Sequence | FIG. 28 of 4 — Step Completeness

STEP 1 (6001) — ANOMALY CONFIRMATION

`IN: binary_determination (5002) | OUT: confirmed_anomaly_record`

6002 requires confirmed_anomaly_record to suspend the correct workloads.

Remove 6001 → 6002 receives binary_determination → type mismatch → UNDEFINED.

STEP 2 (6002) — WORKLOAD SUSPENSION

`IN: confirmed_anomaly_record | OUT: suspended_workload_map`

6003 requires suspended_workload_map to know which routing entries to reroute.

Remove 6002 → UNDEFINED.

STEP 3 (6003) — REDUNDANT CORE ACTIVATION

`IN: suspended_workload_map | OUT: active_redundant_core_set`

6004 requires active_redundant_core_set as transfer destination.

Remove 6003 → UNDEFINED.

STEP 4 (6004) — WORKLOAD TRANSFER

`IN: active_redundant_core_set | OUT: completed_transfer_record`

6005 requires completed_transfer_record to confirm transfer before isolating.

Remove 6004 → UNDEFINED.

STEP 5 (6005) — ISOLATION + ROLLBACK

`IN: completed_transfer_record | OUT: verified_stable_state`

Without isolation and rollback, the anomalous core remains in active topology.

Remove 6005 → system continues executing on corrupted state.

MINIMUM NECESSARY COMPLEXITY: every step is required by its successor's type structure

No step can be removed without breaking a type contract downstream. The sequence is irreducible.

FIG. 28

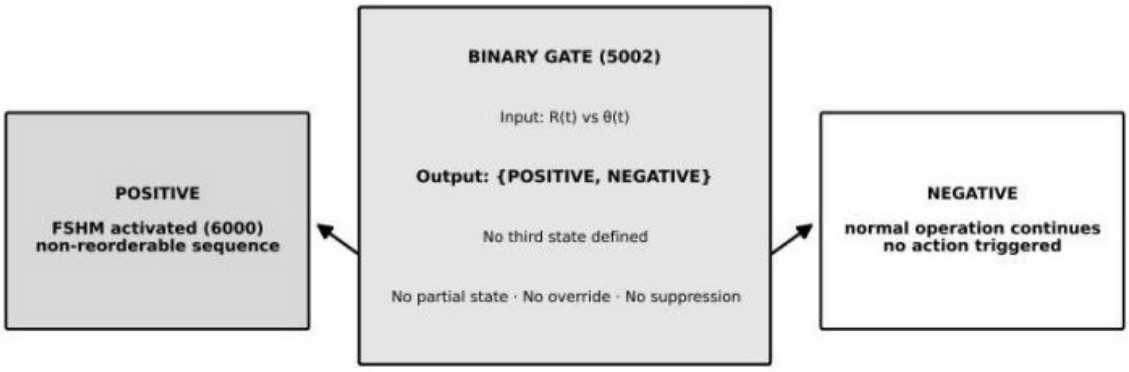
FIG. 28

Binary Anomaly Determination — Exclusive Trigger Gate Definition — FIG. 29

[0044] Referring to FIG. 29, the Exclusive Trigger Gate Definition for Element 2 illustrates the binary gate (5002) accepting R(t) versus θ(t) and producing {POSITIVE, NEGATIVE} with no third state defined, no partial state, no override, and no suppression. A non-binary gate permits conditional FSHM triggering; conditional hardware activation creates a race condition as the anomaly progresses while the condition is evaluated; selective suppression allows the FSHM to be defeated by external input breaking the autonomous recovery guarantee. The binary gate closes all three failure modes. The exclusivity guarantee is architecturally enforced — not a policy, not a convention, not a configuration option.

FIG. 29 — Binary Anomaly Determination — Exclusive Trigger Gate Definition

COGNIZANCE PROOF — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger | FIG. 29 of 4 — Gate Definition



WHY THE GATE MUST BE BINARY

- A non-binary gate permits conditional triggering or selective suppression of FSHM.
- Conditional activation creates a race condition — anomaly can progress while condition is evaluated.
- Selective suppression means FSHM can be defeated by external input.
- The binary gate closes all three failure modes: conditional triggering, race condition, external suppression.

EXCLUSIVITY GUARANTEE: The binary determination (5002) is the only valid FSHM trigger

This guarantee is architecturally enforced — not a policy, not a convention, not a configuration option

FIG. 29

FIG. 29

Binary Anomaly Determination — Alternative Trigger Failure Proof — FIG. 30

[0045] Referring to FIG. 30, the Alternative Trigger Failure Proof evaluates four alternative FSHM trigger mechanisms. Alternative 1 — External Software Handler: software latency plus failure risk plus override capability — not equivalent. Alternative 2 — Operator Command: human reaction time orders of magnitude slower than chip-speed anomaly progression; operator can choose not to activate — not equivalent. Alternative 3 — Non-VAE Hardware Interrupt: no anomaly semantics, cannot distinguish anomaly from normal state change, no reconstruction error computation — not equivalent. Alternative 4 — Probabilistic Threshold: not binary, allows partial states, threshold tuning creates external dependency — not equivalent. All four alternative triggers fail for distinct structural reasons.


FIG. 30 — Binary Anomaly Determination — Alternative Trigger Failure Proof
COGNIZANCE PROOF — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger | FIG. 30 of 4 — Alternative Trigger Failures
ALTERNATIVE 1: External Software Handler
Claim: Software handler monitors chip telemetry, decides FSHM activation
■ Software layer introduces latency between anomaly onset and detection.
■ Software handler itself can fail — single point of failure at software layer.
Latency + software failure risk + override capability = not equivalent → UNDEFINED OPERATION
ALTERNATIVE 2: Operator Command
Claim: Human operator reviews telemetry and issues FSHM activation command
■ Human reaction time is orders of magnitude slower than chip-speed anomaly progression.
■ Operator can choose not to activate — breaking autonomous recovery guarantee.
Latency + operator override + deployment gap = not equivalent → UNDEFINED OPERATION
ALTERNATIVE 3: Hardware Interrupt (non-VAE)
Claim: Generic hardware interrupt triggers FSHM on any signal
■ Hardware interrupt carries no anomaly semantics — cannot distinguish anomaly from normal state change.
■ No reconstruction error computation — no defined threshold comparison.
No semantic content + false positive rate = not equivalent → UNDEFINED OPERATION
ALTERNATIVE 4: Probabilistic Threshold
Claim: FSHM triggered when anomaly probability exceeds a threshold
■ Probabilistic threshold is not binary — allows partial states.
■ Threshold tuning creates external dependency — system is no longer self-contained.
Partial state + race condition + external dependency = not equivalent → UNDEFINED OPERATION
All four alternative triggers fail for distinct structural reasons— no alternative achieves the binary gate guarantee

FIG. 30

FIG. 30

Binary Anomaly Determination — Partial State Contamination Proof — FIG. 31

[0046] Referring to FIG. 31, the Partial State Contamination Proof illustrates the operational timeline without a binary gate. At T_0 anomaly onset occurs. At T_1 no binary gate is present and a probabilistic trigger accumulates while the FSHM remains inactive. At T_2 SNN Cores continue STDP updates on anomalous state. At T_3 the Memristive Array receives corrupted weight updates. At T_4 the checkpoint (3003) is updated with corrupted state and the last valid checkpoint is overwritten. At T_5 the trigger fires but the checkpoint now contains corrupted state. At T_6 rollback restores the corrupted checkpoint and the system resumes on corrupted state. With the binary gate, the FSHM activates at T_0 before weight corruption and before checkpoint contamination — the contamination window is zero.

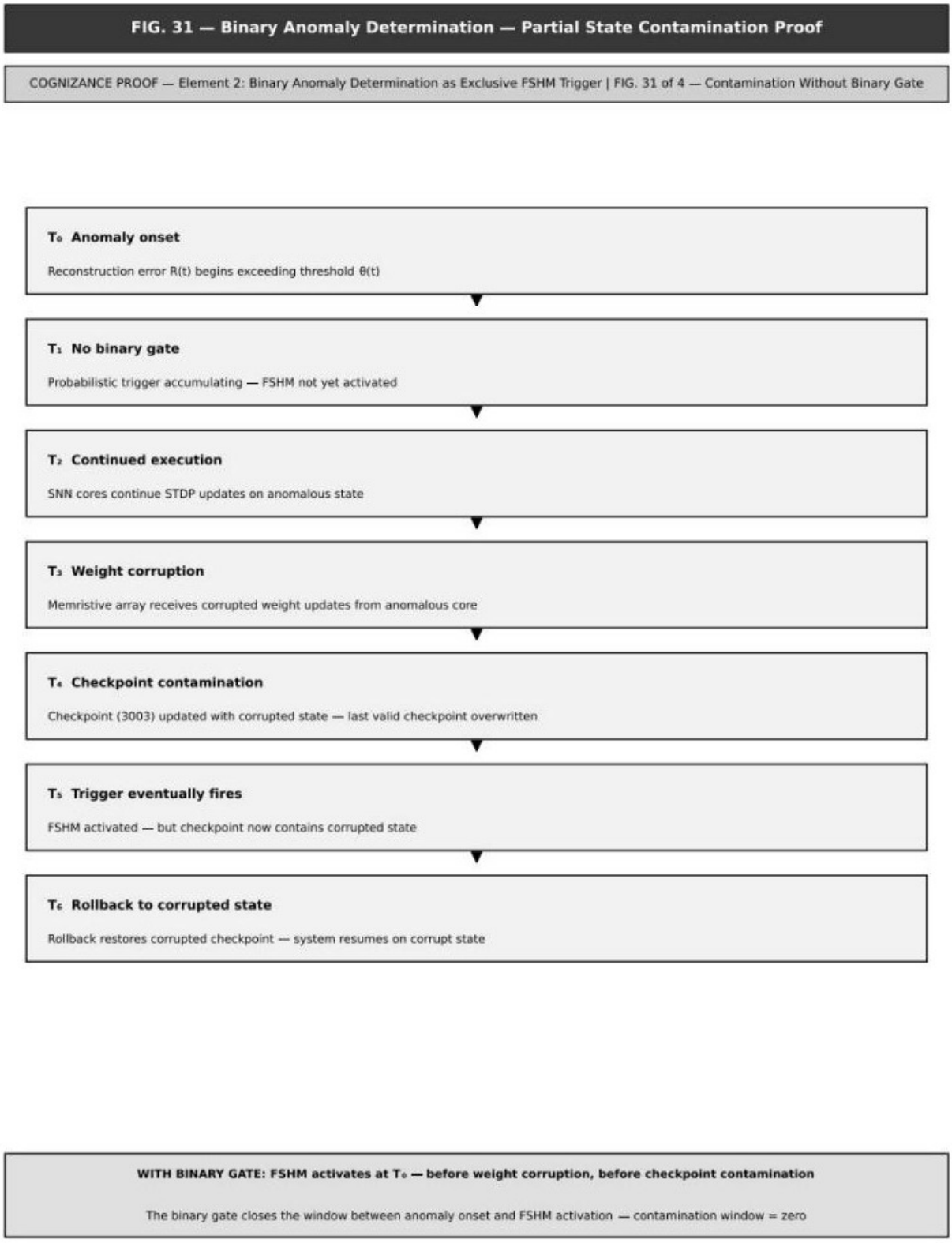


FIG. 31 — Binary Anomaly Determination — Partial State Contamination Proof
COGNIZANCE PROOF — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger | FIG. 31 of 4 — Contamination Without Binary Gate
T₀ Anomaly onset
Reconstruction error R(t) begins exceeding threshold θ(t)
T₁ No binary gate
Probabilistic trigger accumulating — FSHM not yet activated
T₂ Continued execution
SNN cores continue STDP updates on anomalous state
T₃ Weight corruption
Memristive array receives corrupted weight updates from anomalous core
T₄ Checkpoint contamination
Checkpoint (3003) updated with corrupted state — last valid checkpoint overwritten
T₅ Trigger eventually fires
FSHM activated — but checkpoint now contains corrupted state
T₆ Rollback to corrupted state
Rollback restores corrupted checkpoint — system resumes on corrupt state
WITH BINARY GATE: FSHM activates at T₀ — before weight corruption, before checkpoint contamination
The binary gate closes the window between anomaly onset and FSHM activation — contamination window = zero

FIG. 31

FIG. 31

Binary Anomaly Determination — Exclusivity Necessity Proof — FIG. 32

[0047] Referring to FIG. 32, the Exclusivity Necessity Proof evaluates four weakenings. Weakening 1 — Allow external activation: bypasses anomaly confirmation (6001); FSHM triggers without confirmed_anomaly_record; 6002 receives wrong type → undefined, bypassing type contract. Weakening 2 — Allow FSHM suppression: anomaly progresses without intervention; checkpoint contaminated before FSHM activates; chip requires external decision to heal → breaks autonomous recovery. Weakening 3 — Allow non-binary trigger: partial scores create race condition; activation threshold becomes external parameter → race condition plus external dependency. Weakening 4 — Allow multiple trigger sources: conflicting activation states; audit log origin traceability broken → conflicting states plus broken audit integrity. Binary determination as exclusive trigger is irreducible.

FIG. 32 — Binary Anomaly Determination — Exclusivity Necessity Proof

COGNIZANCE PROOF — Element 2: Binary Anomaly Determination as Exclusive FSHM Trigger | FIG. 32 of 4 — Exclusivity Necessity

WEAKENING 1: Allow external activation

Claim: External signal (software, operator) can also activate FSHM

■ External activation bypasses the anomaly confirmation step (6001).

■ FSHM triggers without confirmed_anomaly_record — 6002 receives wrong type.

Bypasses type contract → UNDEFINED → UNDEFINED OPERATION

WEAKENING 2: Allow FSHM suppression

Claim: External signal can suppress FSHM activation

■ Anomaly can progress without FSHM intervention.

■ Checkpoint (3003) can be contaminated before FSHM eventually activates.

Breaks autonomous recovery guarantee → UNDEFINED OPERATION

WEAKENING 3: Allow non-binary trigger

Claim: Activation based on probability score or confidence

■ Partial scores create race condition — anomaly progresses while score accumulates.

■ Activation threshold is an external parameter — system no longer self-contained.

Race condition + external dependency → UNDEFINED OPERATION

WEAKENING 4: Allow multiple trigger sources

Claim: VAE binary determination + external signals both valid

■ Multiple valid trigger sources create conflicting activation states.

■ Audit log receives events from multiple sources — origin traceability broken.

Conflicting states + audit integrity broken → UNDEFINED OPERATION

Every weakening of exclusivity breaks a distinct structural guarantee— binary determination as exclusive trigger is irreducible

FIG. 32

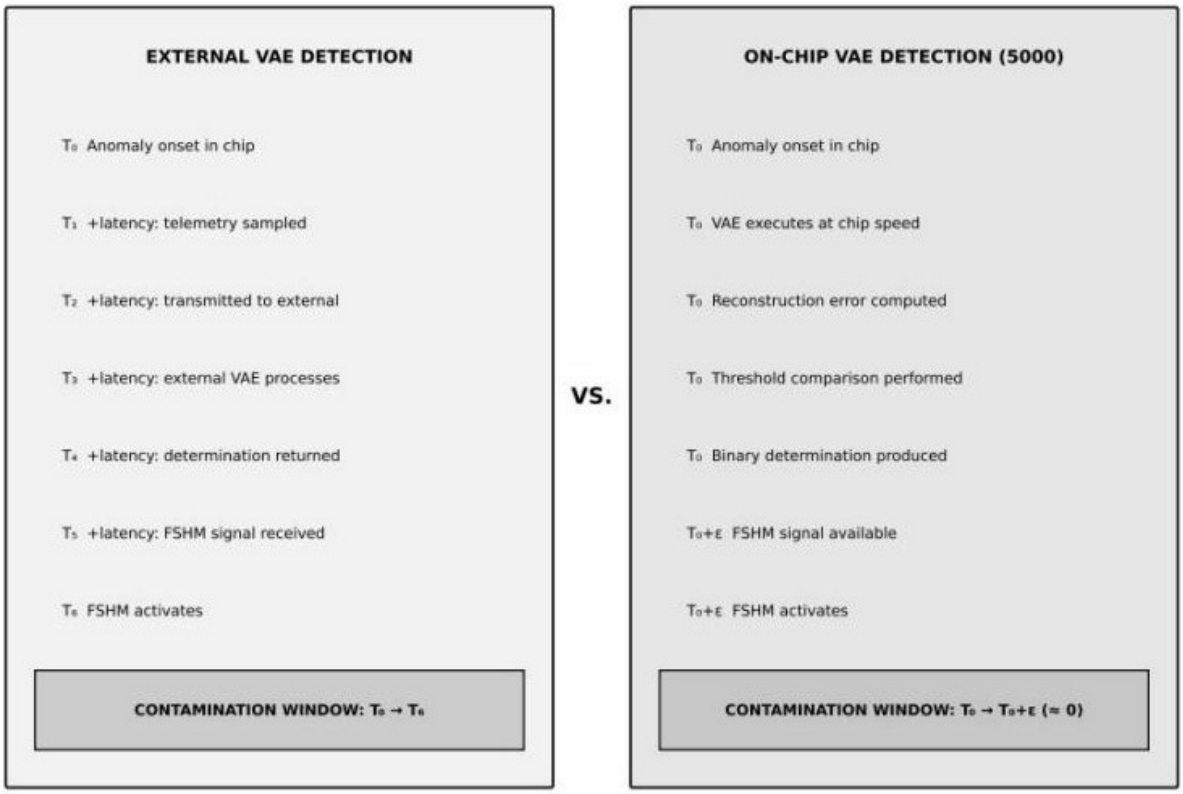
FIG. 32

On-Chip VAE Anomaly Detection — Latency Gap Proof — FIG. 33

[0048] Referring to FIG. 33, the Latency Gap Proof compares external versus on-chip VAE detection. External detection: T_0 anomaly onset → T_1 telemetry sampled → T_2 transmitted → T_3 external VAE processes → T_4 determination returned → T_5 FSHM signal received → T_6 FSHM activates. Contamination window: T_0 to T_6. On-chip detection: T_0 anomaly onset → at T_0 VAE executes at chip speed, reconstruction error computed, threshold compared, binary determination produced → at $T_0+\varepsilon$ FSHM activates. Contamination window: T_0 to $T_0+\varepsilon$, approaching zero. With external VAE the checkpoint (3003) can be contaminated before FSHM activates; with on-chip detection the FSHM activates before the checkpoint can be contaminated. The latency difference is architecturally decisive.

FIG. 33 — On-Chip VAE Anomaly Detection — Latency Gap Proof

COGNIZANCE PROOF — Element 3: On-Chip VAE Not External | FIG. 33 of 4 — External Detection Latency Proof



CONTAMINATION WINDOW COMPARISON — External: T₀ to T₆ (multiple round-trip latencies). On-chip: T₀ to T₀+ε (chip-speed)

External VAE is not equivalent to on-chip VAE — latency difference is architecturally decisive

FIG. 33

FIG. 33

On-Chip VAE Anomaly Detection — Integration Irreducibility Proof — FIG. 34

[0049] Referring to FIG. 34, the Integration Irreducibility Proof evaluates four weakenings. Weakening 1 — Move VAE to external processor: interface latency creates contamination window; external VAE can fail; external VAE state not synchronized with chip state — not equivalent. Weakening 2 — Replace with rule-based detector: fixed rules cannot adapt to evolving operational conditions; meta-learning (2003) cannot update fixed rules; no reconstruction error means binary threshold comparison is impossible — not equivalent. Weakening 3 — Remove anomaly detection: FSHM activation becomes non-deterministic; no threshold, no meta-learned adaptation, no binary semantic determination — not equivalent. Weakening 4 — Sampling-based external detection: sampling interval creates detection gap; sampling rate is configurable external parameter; between-sample checkpoint contamination is possible — not equivalent. On-chip VAE integration is structurally necessary.

FIG. 34 — On-Chip VAE Anomaly Detection — Integration Irreducibility Proof

COGNIZANCE PROOF — Element 3: On-Chip VAE Not External | FIG. 34 of 4 — Integration Irreducibility

WEAKENING 1: Move VAE to external processor

Claim: External VAE receives chip telemetry via interface

■ Interface latency creates contamination window (FIG. 33).

■ External VAE can fail — single point of failure outside chip boundary.

Latency + failure risk + stale state = not equivalent → UNDEFINED OPERATION

WEAKENING 2: Replace VAE with rule-based anomaly detector

Claim: Fixed rules detect anomalous patterns without VAE

■ Fixed rules cannot adapt to evolving operational conditions.

■ Rule-based detection has no reconstruction error — binary threshold comparison not possible.

No adaptation + no reconstruction error = not equivalent → UNDEFINED OPERATION

WEAKENING 3: Remove anomaly detection entirely

Claim: FSHM triggered by hardware interrupt instead of VAE

■ Hardware interrupt has no anomaly semantic content (FIG. 30, Alternative 3).

■ No threshold, no meta-learned adaptation, no binary semantic determination.

No semantic content + non-deterministic = not equivalent → UNDEFINED OPERATION

WEAKENING 4: Use sampling-based external detection

Claim: External detector samples telemetry at intervals

■ Sampling interval creates detection gap — anomaly can progress between samples.

■ Between-sample contamination of checkpoint is possible.

Detection gap + external dependency = not equivalent → UNDEFINED OPERATION

On-chip VAE integration is structurally necessary — every alternative breaks a distinct guarantee

FIG. 34

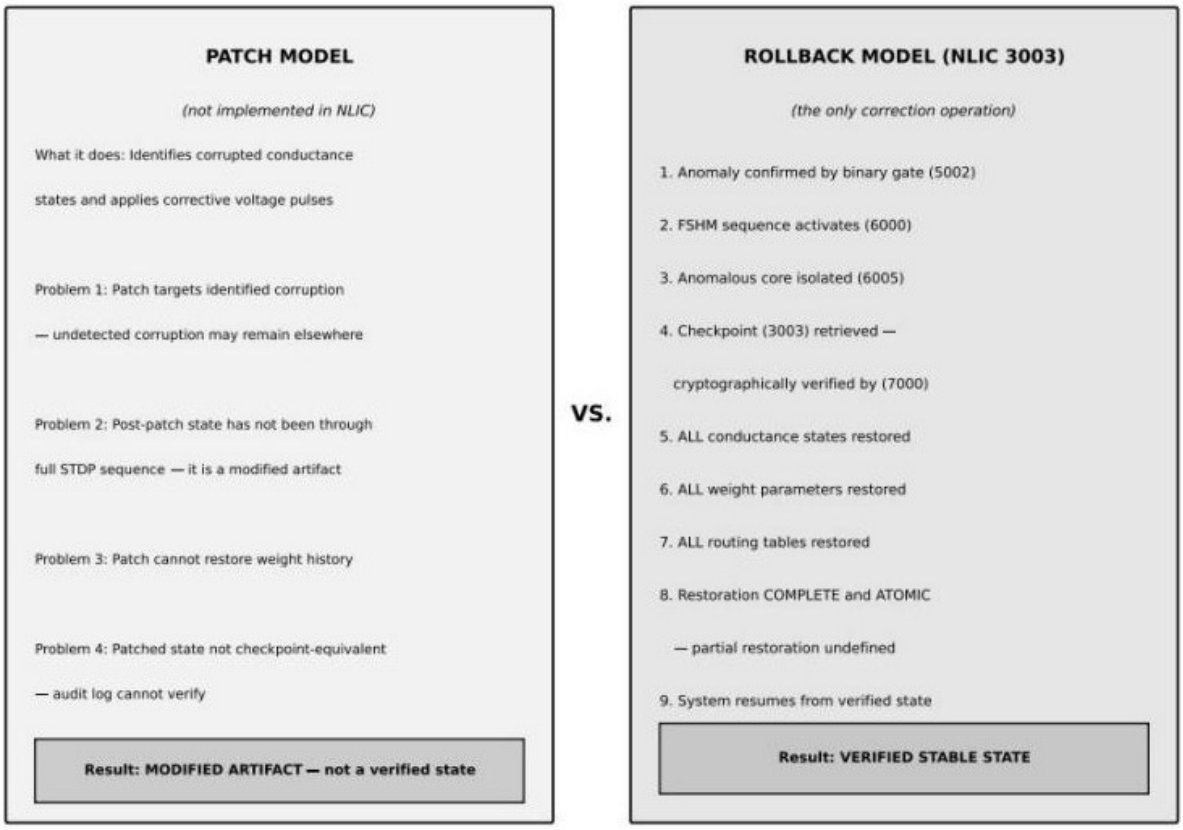
FIG. 34

Memristive Checkpoint — Patch vs. Rollback Categorical Distinction — FIG. 35

[0050] Referring to FIG. 35, the Patch versus Rollback Categorical Distinction for Element 4 illustrates the two correction models. The Patch Model — not implemented in the NLIC — targets identified corruption with corrective voltage pulses. It fails for four reasons: patch targets only identified corruption while undetected corruption may remain elsewhere; the post-patch state has not been through the full STDP training sequence and is a modified artifact; the patch cannot restore lost weight history and trajectory; and the patched state is not checkpoint-equivalent so the audit log cannot verify patch correctness. Result: modified artifact, not a verified state. Patching is undefined in the NLIC architecture. The Rollback Model (3003) — the only correction operation — restores all conductance states, all weight parameters, and all routing tables completely and atomically from the cryptographically verified checkpoint. Result: verified stable state identical to the pre-anomaly checkpoint. These are categorically distinct objects, not quantitatively different approaches.

FIG. 35 — Memristive Checkpoint — Patch vs. Rollback Categorical Distinction

COGNIZANCE PROOF — Element 4: Memristive Checkpoint + Rollback Only — No Patching | FIG. 35 of 4 — Categorical Distinction



CATEGORICAL DISTINCTION: A patched state is a modified artifact — it has not been through the validation sequence

A rolled-back state is a verified checkpoint — validated at checkpoint creation time by the security engine

FIG. 35

FIG. 35

Memristive Checkpoint — Patch Drift Proof — FIG. 36

[0051] Referring to FIG. 36, the Patch Drift Proof establishes that $S\_N = S_0 + \Sigma_i^{n-1} \delta_i$, where S_N is the system state after N patch cycles, S_0 is the original validated state, and δ_i is the patch error at cycle i. At cycle 0 the checkpoint is verified with $\delta = 0$. After each patch cycle drift accumulates. At cycle 3 the checkpoint is updated from the patched state, embedding drift in the checkpoint itself. Rollback from the drifted checkpoint restores the accumulated drift rather than S_0. By contrast, rollback always produces $S\_N = S_0$ for all N. Patch drift is unbounded; a single patch breaks the reproducibility guarantee.

FIG. 36 — Memristive Checkpoint — Patch Drift Proof

COGNIZANCE PROOF — Element 4: Memristive Checkpoint + Rollback Only — No Patching | FIG. 36 of 4 — Patch Drift Over Cycles

PATCH DRIFT EQUATION

$$S_N = S_0 + \Sigma_i^{n-1} \delta_i \quad \text{where } \delta_i = \text{patch error at cycle } i$$

S_N = system state after N patch cycles | S_0 = original validated state

Cycle	Event	State	Drift from S_0
0	Checkpoint created (3003)	S_0 — verified	$\delta = 0$
1	Anomaly — patch applied	$S_0 + \delta_1$	δ_1 accumulated
2	Next cycle — patch applied again	$S_0 + \delta_1 + \delta_2$	$\delta_1+\delta_2$ accumulates
3	Checkpoint updated from patched state	CORRUPTED checkpoint	drift embedded in checkpoint
4	Anomaly — rollback from corrupted checkpoint	$S_0 + \delta_1+\delta_2+\delta_3$	rollback restores drift

ROLLBACK ELIMINATES DRIFT: $S_N = S_0$ for all N — rollback always restores to the pre-anomaly checkpoint

Patch drift is unbounded — grows with every patch cycle — system diverges from validated state over time

FIG. 36

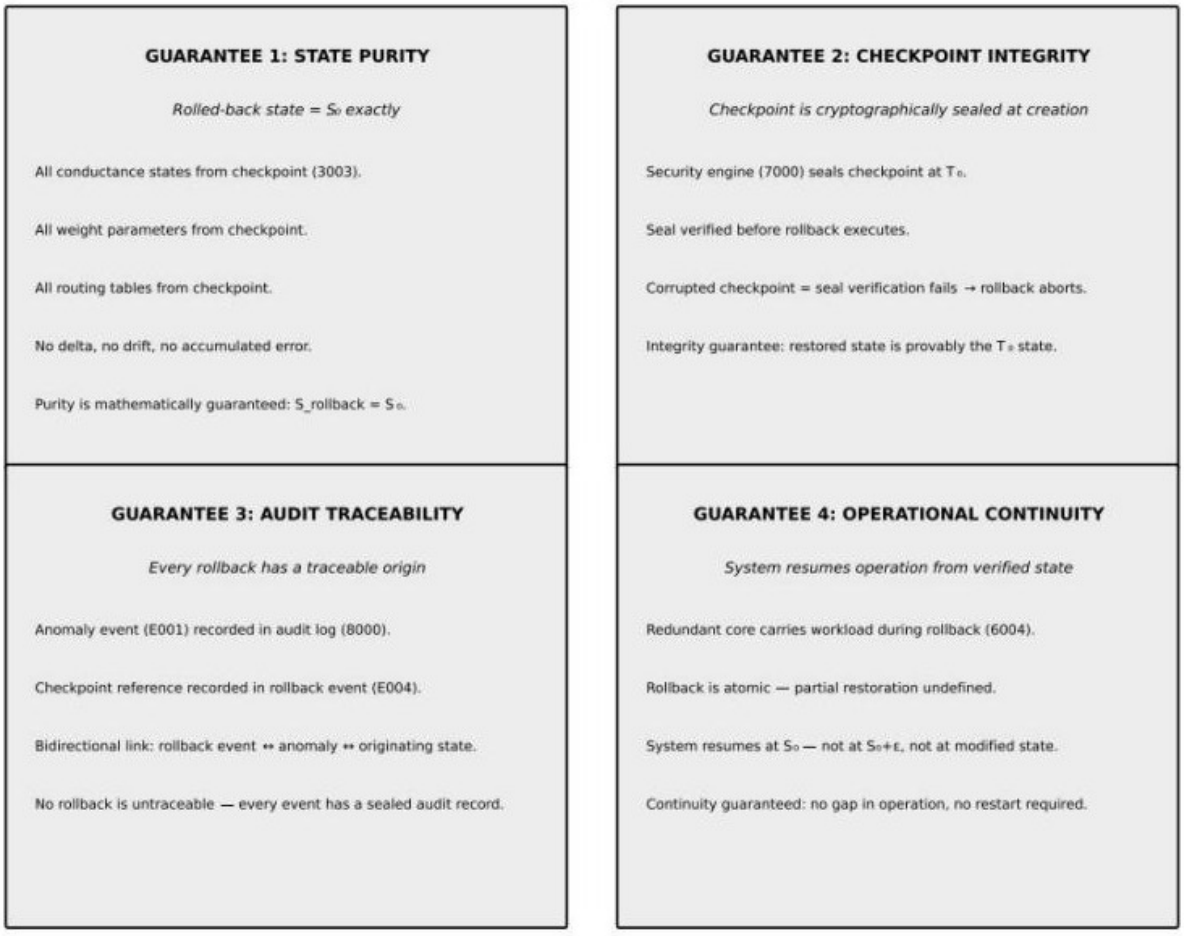
FIG. 36

Memristive Checkpoint — Rollback Structural Guarantees — FIG. 37

[0052] Referring to FIG. 37, the Rollback Structural Guarantees establish four guarantees achievable only through rollback. Guarantee 1 — State Purity: S_rollback = S_0 exactly; all conductance states, weight parameters, and routing tables restored; no delta, no drift, no accumulated error. Guarantee 2 — Checkpoint Integrity: cryptographically sealed at creation by Security Engine (7000); seal verified before rollback; corrupted checkpoint causes seal verification failure and rollback abort. Guarantee 3 — Audit Traceability: anomaly event (E001) and rollback event (E004) are both sealed in the audit log (8000) with bidirectional linkage; no rollback is untraceable. Guarantee 4 — Operational Continuity: redundant core carries workload during rollback (6004); rollback is atomic with partial restoration undefined; no operational gap. Patching provides none of these four guarantees.

FIG. 37 — Memristive Checkpoint — Rollback Structural Guarantees

COGNIZANCE PROOF — Element 4: Memristive Checkpoint + Rollback Only — No Patching | FIG. 37 of 4 — Guarantee Map



These four guarantees are only achievable through rollback — patching provides none of them

FIG. 37

FIG. 37

Memristive Checkpoint — Correction Model Irreducibility Proof — FIG. 38

[0053] Referring to FIG. 38, the Correction Model Irreducibility Proof evaluates four weakenings. Weakening 1 — Remove correction entirely: corrupted conductance states propagate through STDP; checkpoint overwritten with corrupted state; system permanently diverges — permanent drift. Weakening 2 — Allow patching: produces modified artifact; drift accumulates as S_N diverges from S_0; audit log cannot verify patched state — modified artifact plus drift plus unverifiable. Weakening 3 — Allow partial rollback: mixed state not equivalent to S_0; checkpoint seal covers all elements so partial restoration breaks the seal — mixed state not S_0. Weakening 4 — Pre-anomaly correction: modifies state before VAE binary gate fires; modified state is not a checkpoint state and is unvalidated — pre-gate modification equals unvalidated state. Checkpoint rollback is the only correction operation.

FIG. 38 — Memristive Checkpoint — Correction Model Irreducibility Proof

COGNIZANCE PROOF — Element 4: Memristive Checkpoint + Rollback Only — No Patching | FIG. 38 of 4 — Irreducibility

WEAKENING 1: Remove correction entirely

Claim: System continues on anomalous state without recovery

■ No recovery → corrupted conductance states propagate through STDP updates.

■ System diverges from validated state permanently.

Permanent drift → no verified state → UNDEFINED OPERATION

WEAKENING 2: Allow patching as correction

Claim: Corrupted conductance states patched in-place

■ Patch produces modified artifact — not a verified state (FIG. 35).

■ Drift accumulates across cycles — S_N diverges from S_0 (FIG. 36).

Modified artifact + drift + unverifiable → UNDEFINED OPERATION

WEAKENING 3: Allow partial rollback

Claim: Only corrupted elements restored from checkpoint

■ Partial rollback leaves some elements at post-anomaly state, some at S_0.

■ Checkpoint seal verification covers all elements — partial restore breaks seal.

Mixed state not S_0 + breaks seal → UNDEFINED OPERATION

WEAKENING 4: Pre-anomaly correction (proactive patching)

Claim: Adjust weights proactively before anomaly threshold crossed

■ Pre-anomaly correction modifies state before VAE binary gate fires.

■ Modified state is not a checkpoint state — it is a modified artifact.

Pre-gate modification = unvalidated state → UNDEFINED OPERATION

Checkpoint rollback is the only correction operation — not a design preference, a structural necessity

FIG. 38

FIG. 38

Hardware-Isolated Security Boundary — Software Encryption Failure Proof — FIG. 39

[0054] Referring to FIG. 39, the Software Encryption Failure Proof evaluates four scenarios. Scenario 1 — Software-layer encryption: software executing on SNN cores has read access to chip-internal state before encryption; a compromised process can read and transmit unencrypted state; encryption key stored in software-accessible memory — not equivalent. Scenario 2 — Network-layer encryption: data travels from chip to network stack unencrypted; accessible to any process between chip output and network stack — not equivalent. Scenario 3 — Key storage in software-accessible memory: software processes can read key material; key extraction compromises all past and future encrypted data — not equivalent. Scenario 4 — Encryption applied after data exits chip boundary: unencrypted data crosses chip boundary and is interceptable; external encryption system is a single point of failure outside chip control — not equivalent. Hardware isolation at the chip boundary is architecturally necessary.


FIG. 39 — Hardware-Isolated Security Boundary — Software Encryption Failure Proof
COGNIZANCE PROOF — Element 5: Hardware-Isolated Security Boundary | FIG. 39 of 4 — Software Encryption Failure Scenarios
SCENARIO 1: Software-layer encryption (no hardware isolation)
Claim: Encryption executed by software running on SNN cores
■ Software executing on SNN cores has read access to chip-internal state before encryption.
■ Software encryption key stored in software-accessible memory — not hardware-isolated.
Software access to pre-encryption state = not equivalent → UNDEFINED OPERATION
SCENARIO 2: Network-layer encryption (TLS/post-quantum at network boundary)
Claim: CRYSTALS-Kyber applied at network layer, not chip boundary
■ Data travels from chip to network stack in unencrypted form.
■ Network layer encryption protects data in transit — not data at chip boundary.
Unencrypted at chip boundary = not equivalent → UNDEFINED OPERATION
SCENARIO 3: Key storage in software-accessible memory
Claim: CRYSTALS-Kyber keys stored in general-purpose memory
■ Software processes can read key material from general-purpose memory.
■ Key extraction = full compromise of all past and future encrypted data.
Software-readable keys = no security → UNDEFINED OPERATION
SCENARIO 4: Encryption applied after data exits chip boundary
Claim: Chip outputs unencrypted data; external system encrypts before storage
■ Unencrypted data crosses chip boundary — attackers with physical access intercept it.
■ External system can fail, be compromised, or be bypassed — chip has no visibility.
Post-boundary exposure = not equivalent → UNDEFINED OPERATION
Each scenario fails for a distinct reason — hardware isolation at the chip boundary is architecturally necessary

FIG. 39

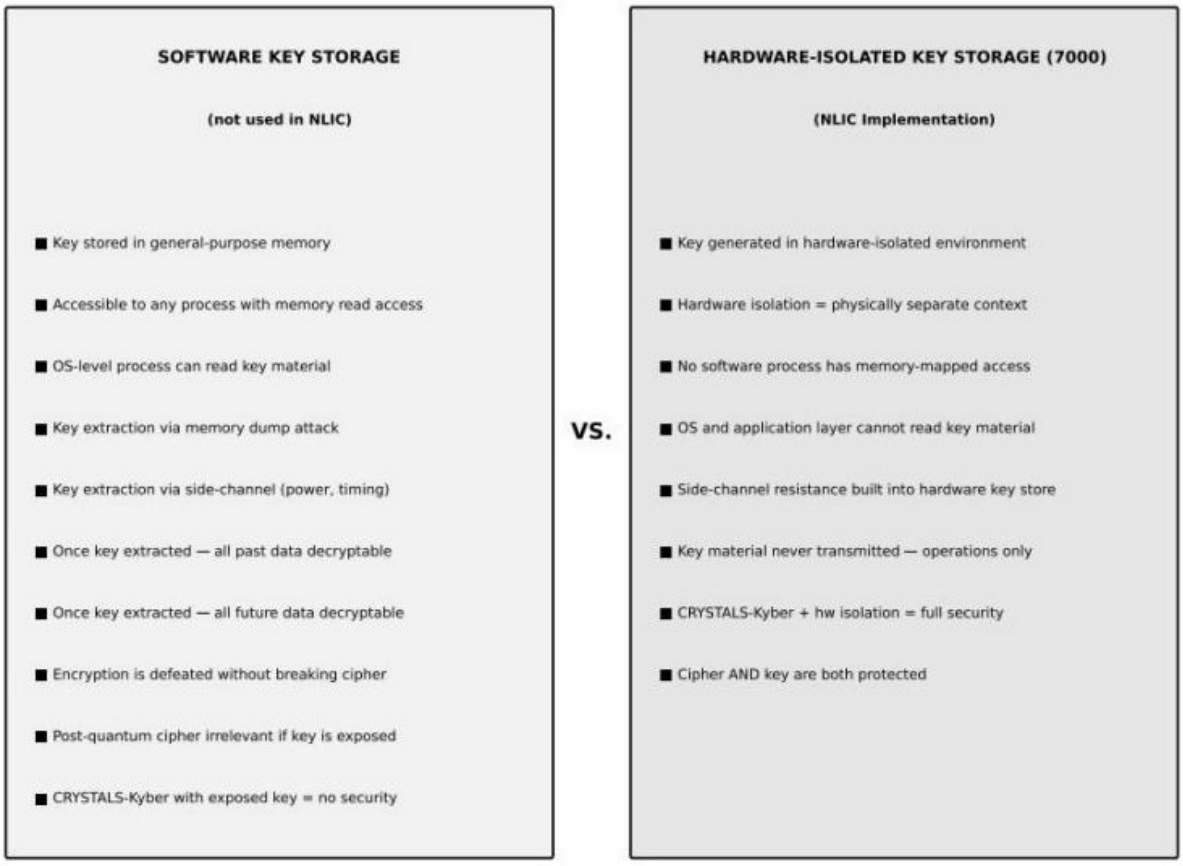
FIG. 39

Hardware-Isolated Security Boundary — Key Isolation Proof — FIG. 40

[0055] Referring to FIG. 40, the Key Isolation Proof compares software key storage against hardware-isolated key storage. Software Key Storage — not used in the NLIC: key stored in general-purpose memory accessible to any process; extractable via memory dump or side-channel attack; once extracted all past and future data is decryptable; CRYSTALS-Kyber with an exposed key provides no security. Hardware-Isolated Key Storage (7000): key generated in physically separate execution context; no software process has memory-mapped access; OS and application layer cannot read key material; side-channel resistance built in; key material never transmitted. Key isolation is a hardware property — it cannot be achieved by software configuration. CRYSTALS-Kyber without hardware key isolation does not achieve the NLIC security boundary guarantee.

FIG. 40 — Hardware-Isolated Security Boundary — Key Isolation Proof

COGNIZANCE PROOF — Element 5: Hardware-Isolated Security Boundary | FIG. 40 of 4 — Key Isolation Necessity



KEY ISOLATION IS A HARDWARE PROPERTY — it cannot be achieved by software configuration or software access control

CRYSTALS-Kyber without hardware key isolation does not achieve the NLIC security boundary guarantee

FIG. 40

FIG. 40

Hardware-Isolated Security Boundary — Boundary Necessity Proof — FIG. 41

[0056] Referring to FIG. 41, the Boundary Necessity Proof maps seven attack vectors against both architectures. Without the hardware boundary: physical access probes chip data lines; process injection introduces malicious processes into the software layer; memory read extracts chip state; side channel extracts keys via power or timing measurement; network intercept captures unencrypted data pre-network-layer; compromised OS accesses all software-accessible state; key extraction recovers encryption keys from general memory. With the hardware boundary (1001): physical probing reads only ciphertext; software layer has no access to hardware-isolated state; hardware-isolated key store is not memory-mapped to software; hardware key store provides side-channel resistance; only ciphertext exits the chip; OS layer cannot access the hardware-isolated execution context; keys are confined to hardware isolation. The hardware boundary closes all seven attack vectors — the software security model leaves all seven open.

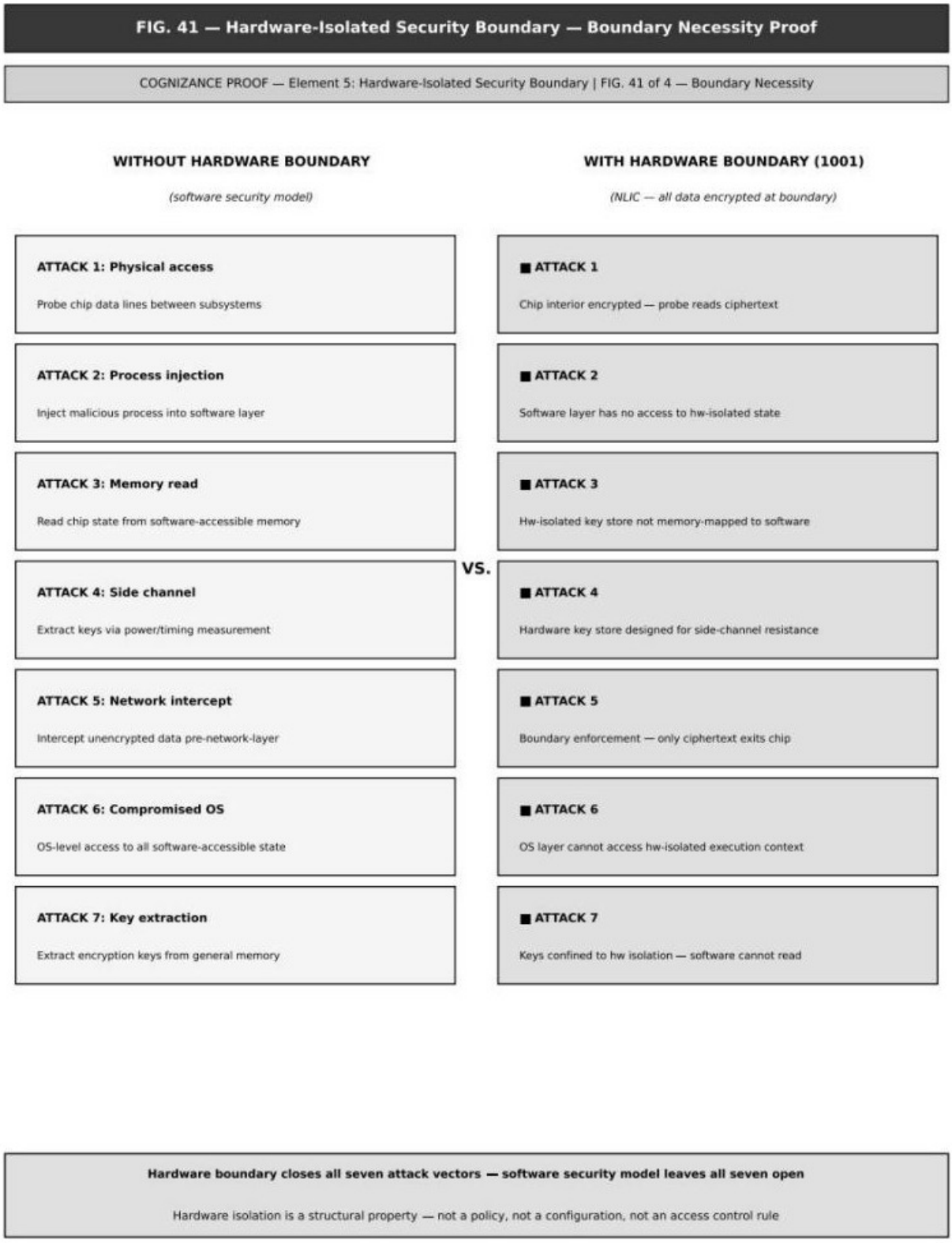


FIG. 41 — Hardware-Isolated Security Boundary — Boundary Necessity Proof
COGNIZANCE PROOF — Element 5: Hardware-Isolated Security Boundary | FIG. 41 of 4 — Boundary Necessity
WITHOUT HARDWARE BOUNDARY
(software security model)
WITH HARDWARE BOUNDARY (1001)
(NLIC — all data encrypted at boundary)
ATTACK 1: Physical access
Probe chip data lines between subsystems
■ ATTACK 1
Chip interior encrypted — probe reads ciphertext
ATTACK 2: Process injection
Inject malicious process into software layer
■ ATTACK 2
Software layer has no access to hw-isolated state
ATTACK 3: Memory read
Read chip state from software-accessible memory
■ ATTACK 3
Hw-isolated key store not memory-mapped to software
VS.
ATTACK 4: Side channel
Extract keys via power/timing measurement
■ ATTACK 4
Hardware key store designed for side-channel resistance
ATTACK 5: Network intercept
Intercept unencrypted data pre-network-layer
■ ATTACK 5
Boundary enforcement — only ciphertext exits chip
ATTACK 6: Compromised OS
OS-level access to all software-accessible state
■ ATTACK 6
OS layer cannot access hw-isolated execution context
ATTACK 7: Key extraction
Extract encryption keys from general memory
■ ATTACK 7
Keys confined to hw isolation — software cannot read
Hardware boundary closes all seven attack vectors — software security model leaves all seven open
Hardware isolation is a structural property — not a policy, not a configuration, not an access control rule

FIG. 41

FIG. 41

Hardware-Isolated Security Boundary — Security Model Irreducibility — FIG. 42

[0057] Referring to FIG. 42, the Security Model Irreducibility Proof evaluates four weakenings. Weakening 1 — Remove security engine entirely: all chip-internal state exposed to external systems; quantum-capable adversary can reconstruct chip behavior — no security. Weakening 2 — Replace hardware isolation with software access control: OS-level control bypassable by OS-level compromise; software access control is a policy while hardware isolation is a physical property — bypassable. Weakening 3 — Move encryption to application layer: application has read access to unencrypted chip state; application vulnerability exposes pre-encryption state — pre-encryption exposure. Weakening 4 — Allow unencrypted internal communication: physical access to chip data lines exposes internal state including proprietary algorithm behavior — IP and security gap. Hardware-isolated boundary encryption is irreducible.

FIG. 42 — Hardware-Isolated Security Boundary — Security Model Irreducibility

COGNIZANCE PROOF — Element 5: Hardware-Isolated Security Boundary | FIG. 42 of 4 — Irreducibility Proof

WEAKENING 1: Remove security engine entirely

Claim: Chip transmits unencrypted state

■ All chip-internal state accessible to external systems.

■ Quantum-capable adversary can reconstruct chip behavior from raw data.

All state exposed = no security → UNDEFINED OPERATION

WEAKENING 2: Replace hardware isolation with software access control

Claim: OS-level access control restricts key access

■ OS-level control is bypassable by OS-level compromise.

■ Software access control is a policy — hardware isolation is a physical property.

Policy-based = bypassable → UNDEFINED OPERATION

WEAKENING 3: Move encryption to application layer

Claim: Application encrypts before transmitting

■ Application layer has read access to unencrypted chip state.

■ Application is not part of the chip — encryption boundary is outside chip.

Pre-encryption exposure = not equivalent → UNDEFINED OPERATION

WEAKENING 4: Allow unencrypted internal communication between subsystems

Claim: Only outbound data encrypted

■ Physical access to chip data lines exposes internal state.

■ Competitor with physical chip access can extract proprietary algorithm behavior.

Internal exposure = IP and security gap → UNDEFINED OPERATION

Hardware-isolated boundary encryption is irreducible — the full guarantee requires all four elements intact

FIG. 42

FIG. 42

Use Case: Autonomous Vehicle — Failure and Self-Healing — FIG. 43

[0058] Referring to FIG. 43, the Autonomous Vehicle Use Case illustrates the NLIC Chip (1000) in a self-driving vehicle processing 128 simultaneous sensor streams while a processing core develops a hardware fault. Without the NLIC, an external software monitor detects the fault and a software handler attempts recovery over 200 milliseconds during which the vehicle continues on corrupted decision state, producing incorrect lane prediction and delayed braking with safety margin lost. With the NLIC, the VAE detects the anomaly at chip speed at $T_0+\varepsilon$, the binary gate fires immediately, the FSHM activates the non-reorderable sequence, workload reroutes to the redundant core, and the checkpoint is restored — total recovery in microseconds with no human intervention required. The binary gate, the non-reorderable sequence, the checkpoint rollback, and the on-chip VAE each contribute to this guarantee — removing any one causes the guarantee to fail.

FIG. 43 — Use Case: Autonomous Vehicle — Failure and Self-Healing

FIG. 43

FIG. 43

Use Case: Healthcare Diagnostics — Adaptive Learning and Auditable Inference — FIG. 44

[0059] Referring to FIG. 44, the Healthcare Diagnostics Use Case illustrates the NLIC Chip (1000) in a hospital diagnostic imaging system analyzing MRI and CT scans. Without the NLIC, the AI diagnostic system produces outputs that cannot be traced to a verified chip state; the audit trail consists of software logs modifiable after the fact; and the compliance team cannot prove system state at the time of each diagnostic decision. With the NLIC, every diagnostic output is encrypted at the chip boundary, every decision is tied to a sealed checkpoint (3003), the Audit Log (8000) is cryptographically sealed with tampering detectable, and a regulator can verify that a specific diagnostic was produced by a specific verified system state at a specific timestamp. The hardware-isolated security engine and the immutable audit log are the two elements that make this possible.

FIG. 44 — Use Case: Healthcare Diagnostics — Adaptive Learning + Auditable Inference

FIG. 44

FIG. 44

Use Case: Cybersecurity Infrastructure — Threat Detection and Quantum Defense — FIG. 45

[0060] Referring to FIG. 45, the Cybersecurity Infrastructure Use Case illustrates the NLIC Chip (1000) in a national cybersecurity operations center processing threat telemetry from millions of network endpoints. Without the NLIC, static or slowly updated models miss novel attack patterns, quantum-capable adversaries intercept decision data, and self-healing requires external security operations team intervention introducing minutes of exposure. With the NLIC, the VAE detects novel patterns by reconstruction error, STDP plus gradient refinement adapts threat models in real time, the Quantum Decision Module routes detection resources to highest-priority indicators, CRYSTALS-Kyber protects threat state from quantum interception at the chip boundary, and the FSHM self-heals during active attack with no gap in coverage. All five elements are required simultaneously — this is the integration claim.

FIG. 45 — Use Case: Cybersecurity Infrastructure — Threat Detection and Quantum Defense

FIG. 45

FIG. 45

Use Case: Energy Grid Management — Fault-Tolerant Processing — FIG. 46

[0061] Referring to FIG. 46, the Energy Grid Management Use Case illustrates the NLIC Chip (1000) managing load distribution across more than 10,000 endpoints where a single incorrect load command during a fault can cascade to a regional blackout. Without the NLIC, a software failover handler reroutes workload with a 150-millisecond delay during which incorrect load commands may issue, the system resumes from a post-fault rather than a verified pre-fault state, and state drift propagates into the next cycle. With the NLIC, the VAE detects the fault at chip speed, workload is suspended before transfer at Step 6002 ensuring no incorrect commands issue during the recovery window, and checkpoint rollback (3003) restores the verified pre-fault state. The non-reorderable sequence and rollback-only correction are the two elements preventing cascading failure.

FIG. 46 — Use Case: Energy Grid Management — Fault-Tolerant Processing

FIG. 46

FIG. 46

Use Case: Aerospace Systems — Mission-Critical Self-Healing — FIG. 47

[0062] Referring to FIG. 47, the Aerospace Systems Use Case illustrates the NLIC Chip (1000) in an autonomous spacecraft on a multi-year deep space mission with communication delay exceeding 20 minutes making real-time human intervention impossible. Without the NLIC, a processing fault requires ground control communication causing the mission sequence to be missed, or an autonomous software handler applies a patch that produces drift accumulation over the multi-year mission. With the NLIC, the processing fault is detected at chip speed, the FSHM activates full autonomous recovery, the checkpoint is restored and the mission continues on a verified state without ground control involvement, CRYSTALS-Kyber protects mission state during communication windows, and the audit log provides a post-mission fault record. All five elements are simultaneously required — this scenario cannot function with any element removed.

FIG. 47 — Use Case: Aerospace Systems — Mission-Critical Self-Healing

FIG. 47

FIG. 47

Use Case: Industrial Robotics — Real-Time Adaptive Learning and Fault Recovery — FIG. 48

[0063] Referring to FIG. 48, the Industrial Robotics Use Case illustrates NLIC Chips (1000) in a robotic assembly line that must adapt to component variation, detect anomalies before they produce defects, and recover from faults without halting production. Without the NLIC, a static ML model cannot adapt without external retraining requiring a production halt, faults trigger line shutdowns of 2 to 4 hours, and the fault state is patched rather than rolled back leaving the next run in an uncertain state. With the NLIC, STDP plus gradient refinement adapts to component variation in real time, the VAE detects processing anomalies before defects are produced, the FSHM recovers the fault in microseconds without shutdown, and checkpoint rollback ensures the next production cycle starts from a verified pre-fault state. Integration at the chip level — not coordination between separate systems — is what makes simultaneous adaptive learning and fault recovery possible.

FIG. 48 — Use Case: Industrial Robotics — Real-Time Adaptive Learning and Fault Recovery

FIG. 48

FIG. 48

Plain Language Summary — What the NLIC Chip Does and Why It Matters — FIG. 49

[0064] Referring to FIG. 49, the Plain Language Summary presents the NLIC chip through five capability statements. It Learns: through STDP, gradient refinement, and meta-learning, the chip continuously improves its own neural weights without requiring external retraining. It Detects Its Own Failures: the on-chip VAE monitors continuously and knows immediately when something is wrong — not after a human or external system notices. It Heals Itself: the FSHM activates a fixed recovery sequence, redundant cores take over, the corrupted core is isolated, and the verified pre-failure state is fully restored without human intervention. It Keeps Its Decisions Private: through CRYSTALS-Kyber hardware-isolated encryption, everything leaving the chip is encrypted and the encryption keys never leave the chip. It Proves What It Did: every decision, failure, and recovery is recorded in a cryptographically sealed tamper-proof log that cannot be modified after the fact.

FIG. 49 — Plain Language Summary — What the NLIC Chip Does and Why It Matters

FIG. 49

FIG. 49

The Full Stack — NLIC Chip and Deterministic Transformation Architecture — FIG. 50

[0065] Referring to FIG. 50, the Full Stack diagram illustrates the two independent but complementary patent layers owned by Adaptive Bespoke Learning LLC. The Hardware Layer is the NLIC Chip (1000) providing: adaptive learning in hardware through SNN, STDP, and the memristive array; on-chip self-healing through VAE, FSHM, and checkpoint rollback; quantum-assisted decisions through VQE and annealing; hardware-boundary encryption through CRYSTALS-Kyber with keys that never leave the chip; and a cryptographically sealed immutable audit log. The Architecture Layer is the Deterministic Transformation System (DTS), US Patent Application 19/638,930, filed April 3, 2026, providing a 14-stage dependency-constrained execution architecture with non-reorderable execution graph, persistent constraint bus, intermediate representation gate, regeneration-only correction, and narrative-bound descriptor placement. Adaptive Bespoke Learning LLC owns both layers — the hardware substrate and the transformation architecture. The integration claim is that the two architectures together provide a complete stack for auditable, deterministic, hardware-enforced intelligent systems that neither provides alone.

FIG. 50 — The Full Stack — NLIC Chip and Deterministic Transformation Architecture

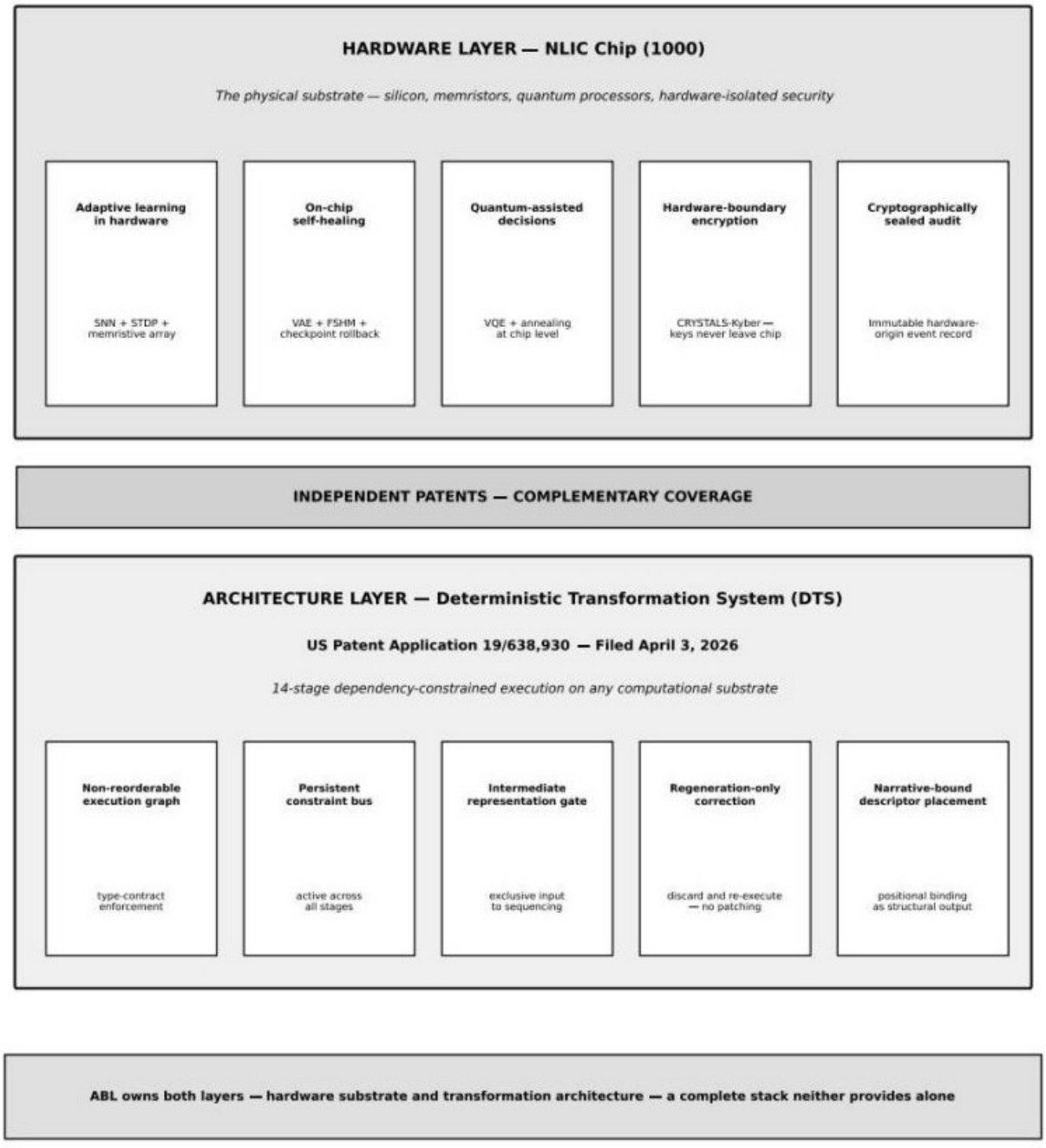


FIG. 50

FIG. 50

INDUSTRIAL APPLICABILITY

[0066] The Neuromorphic Logic Integrated Circuit disclosed herein has immediate and substantial industrial applicability across every domain in which intelligent processing systems must adapt continuously, detect their own failures, recover deterministically, protect decision data against quantum-capable adversaries, and prove their operational history to external authorities. The six representative deployments described below — autonomous vehicles, healthcare diagnostics, cybersecurity infrastructure, energy grid management, aerospace systems, and industrial robotics — correspond to FIGS. 43 through 48 and to the Detailed Description at paragraphs [0058] through [0063]. In each deployment, the industrial value derives not from any single subsystem but from the chip-level integration of all subsystems within the hardware-isolated security boundary (1001).

[0067] Autonomous Vehicles (FIG. 43). A self-driving vehicle processing 128 simultaneous sensor streams cannot tolerate the 200-millisecond software-handler recovery window during which the vehicle continues operating on corrupted decision state. The NLIC Chip (1000) detects the core fault at chip speed through the on-chip VAE, fires the binary gate immediately, executes the non-reorderable FSHM sequence, reroutes workload to the redundant core, and restores the verified checkpoint — total recovery in microseconds with no human intervention. The automotive industry requirement is safety-margin preservation during hardware fault; the NLIC satisfies it structurally.

[0068] Healthcare Diagnostics (FIG. 44). A hospital diagnostic imaging system analyzing MRI and CT scans must prove to regulators that a specific diagnostic output was produced by a specific verified system state at a specific timestamp. The NLIC encrypts every diagnostic output at the chip boundary, ties every decision to a sealed checkpoint (3003), and records it in the cryptographically sealed Audit Log (8000) in which tampering is detectable. The hardware-isolated security engine and the immutable audit log together satisfy regulatory auditability requirements that software logs, modifiable after the fact, cannot.

[0069] Cybersecurity Infrastructure (FIG. 45). A national cybersecurity operations center processing threat telemetry from millions of network endpoints requires simultaneous real-time model adaptation (STDP plus gradient refinement), novel-pattern detection by VAE reconstruction error, quantum-assisted resource routing through the Quantum Decision Module, CRYSTALS-Kyber protection of threat state at the chip boundary, and FSHM self-healing during active attack with no gap in coverage. All five elements are required simultaneously; the industrial applicability is the integration itself.

[0070] Energy Grid Management (FIG. 46). A regional grid control system managing load distribution across more than 10,000 endpoints cannot issue a single incorrect load command during a fault without risking cascading regional blackout. The NLIC suspends workload before transfer at Step 6002, ensuring no incorrect commands issue during the recovery window, and checkpoint rollback (3003) restores the verified pre-fault state rather than resuming from a post-fault state with accumulated drift. The non-reorderable sequence and rollback-only correction are the two elements preventing cascade.

[0071] Aerospace Systems (FIG. 47). An autonomous spacecraft on a multi-year deep-space mission with communication delay exceeding 20 minutes cannot rely on ground control for fault recovery and cannot tolerate patch-induced drift accumulation. The NLIC detects the processing

fault at chip speed, executes fully autonomous FSHM recovery, restores the verified checkpoint, protects mission state through CRYSTALS-Kyber during communication windows, and preserves a post-mission fault record in the audit log. The deployment cannot function with any element removed.

[0072] Industrial Robotics (FIG. 48). A robotic assembly line must adapt to component variation in real time, detect anomalies before defects are produced, and recover from faults in microseconds without the 2-to-4-hour line shutdowns that external retraining and software fault handling impose. STDP plus gradient refinement provides the adaptation, the on-chip VAE provides pre-defect detection, the FSHM provides microsecond recovery, and checkpoint rollback ensures each production cycle starts from a verified pre-fault state. Integration at the chip level — not coordination between separate systems — is what makes simultaneous adaptive learning and fault recovery industrially achievable.

[0073] In each of the foregoing deployments, the invention is manufacturable using established semiconductor fabrication processes, memristive device fabrication, and standardized post-quantum cryptographic implementations, and is therefore capable of exploitation in industry within the meaning of applicable patent law. The applicability is not speculative: each deployment maps to an existing industrial market in which the absence of hardware-enforced self-healing, hardware-boundary encryption, or immutable audit capability is a documented, present deficiency.

ADDITIONAL EMBODIMENT — CONTRIBUTOR-CONTROLLED SEALED RECORD

[0074] In a further embodiment, the key-isolation property of the Security Engine (7000) is oriented so that authority over the sealed record is held not by the operator of the circuit (1000) but by a party external to the operator, designated the contributor. This embodiment converts a control relationship that is ordinarily contractual or policy-based into a hardware-structural property of the circuit. The mechanism reuses the structural property already established — key material inaccessible to any software process — and assigns custody of that key material to the contributor rather than to the operator.

[0075] The Security Engine (7000) generates its key-encapsulation key material within the hardware-isolated execution environment described above. In the contributor-controlled embodiment, the unsealing key material is generated within, and bound to, a hardware-isolated key domain under the contributor’s authority, designated the contributor key custody (7001), and is never exported from that domain. Sealing of a record proceeds as in the base embodiment. Recovery of sealed plaintext, and verification of attribution, require the contributor-held key material resident only within the contributor key custody (7001). The operator’s software has no interface by which to extract that key material — the operator is, with respect to the contributor’s sealed content, one of the software processes from which key material is structurally inaccessible.

[0076] Because the contributor key custody (7001) resides within hardware isolation and exposes no extraction path to operator software, the operator’s inability to access the contributor’s sealed content is not a permission, a setting, or a contractual undertaking — it is a structural consequence of the hardware architecture and is not a runtime-configurable parameter. Authority to control the sealed content is thereby a hardware-structural property.

[0077] The Oversight Interface (8001) exposes each sealed record, read-only, to a contributor community authority (9002) external to the operator. The contributor community authority (9002) observes, audits, and certifies the use of its sealed content through the Oversight Interface (8001) and possesses no write authority over any operational parameter of the circuit (1000). The operator runs the circuit and is the audited party. This inverts the conventional arrangement: the party whose content is at stake holds the verification authority, and the party operating the platform holds none over the record.

[0078] In a federated configuration, a plurality of contributor community authorities (9002) each operate within an independent contributor key custody (7001), connected by a federation fabric (8002). Each authority holds the key material for, and seals, only its own records. No authority holds key material for the records of any other, and no instance can modify, suppress, or fabricate the sealed record (8000) of any other instance. The federation fabric (8002) is a verification mesh through which any authority's records may be cryptographically verified, not a centralized store in which records are aggregated. There is accordingly no central party able to access, alter, or withhold the records of any contributor.

[0079] Each element of content incorporated under a contributor's authority is associated with a sealed attribution provenance record (8003) within the Audit Log (8000) at the moment of incorporation. Because any modification to a sealed record after production produces a detectable cryptographic verification failure observable through the Oversight Interface (8001), the provenance of each contributed element is non-erasable and tamper-evident: post-seal alteration or deletion of an attribution is a detectable failure, not a silent edit.

[0080] The circuit (1000) operates in combination with a deterministic transformation system through a transformation-system interface (9000), the deterministic transformation system producing structured content outputs. In this embodiment, the structured content output is in a target language. The circuit provides hardware-enforced decision execution for assessment and pathway determinations applied to the target-language content, and produces a cryptographically verifiable record of both the content generation and the decision execution, designated the target-language content record (9001). Decryption and attribution authority for the target-language content record (9001) is confined to the contributor-held key material within the contributor key custody (7001), and a plurality of instances are federated through the federation fabric (8002) such that no instance can alter the record of any other.

[0081] The target-language content record (9001) preserves a native instantiation of source content in the target language under the contributor-held key. The instantiation is produced by the deterministic transformation system; the circuit's function is confined to sealing, attributing, and exposing the record under contributor-controlled key custody.

[0082] The boundary of this embodiment is structural and is stated expressly. The circuit (1000) performs no linguistic computation. It does not generate, translate, render, evaluate, or determine the comprehensibility, pedagogical adequacy, or linguistic correctness of the target-language content. Those functions are performed outside the circuit, by the deterministic transformation system and by the contributors. The circuit's role is confined to the governance layer: producing, attributing, sealing, federating, and exposing the verifiable record under contributor-controlled key custody. No determination concerning the linguistic or pedagogical properties of the content is made by the circuit and none is disclosed or claimed herein.

[0083] This embodiment is realizable on the digital embodiment of the circuit — the Spiking Neural Network array (2000) on a neuromorphic substrate, the Memristive Synaptic Array

(3000) as a resistive memory array, and the Decision Module (4000) realized in deterministic digital hardware — without a physical quantum-computing element. The sovereignty mechanism employs only the Security Engine (7000), the contributor key custody (7001), the Audit Log (8000) and attribution provenance record (8003), the Oversight Interface (8001), and the federation fabric (8002), each of which is fabricable by complementary metal-oxide-semiconductor (CMOS)-compatible semiconductor processes. The embodiment therefore requires no capability beyond present-day fabrication.

[0084] Referring to FIG. 51, the contributor-controlled sealed-record embodiment is depicted in a federated configuration, showing the contributor key custody (7001) within the hardware-isolated boundary (1001/7000), the target-language content record (9001) sealed therein and written to the Audit Log (8000), a plurality of contributor community authorities (9002) each holding an independent record with no central aggregating node, and the read-only verification path through the Oversight Interface (8001).

FEDERATED CONTRIBUTOR-CONTROLLED

LANGUAGE-SOVEREIGNTY RECORD (9001)

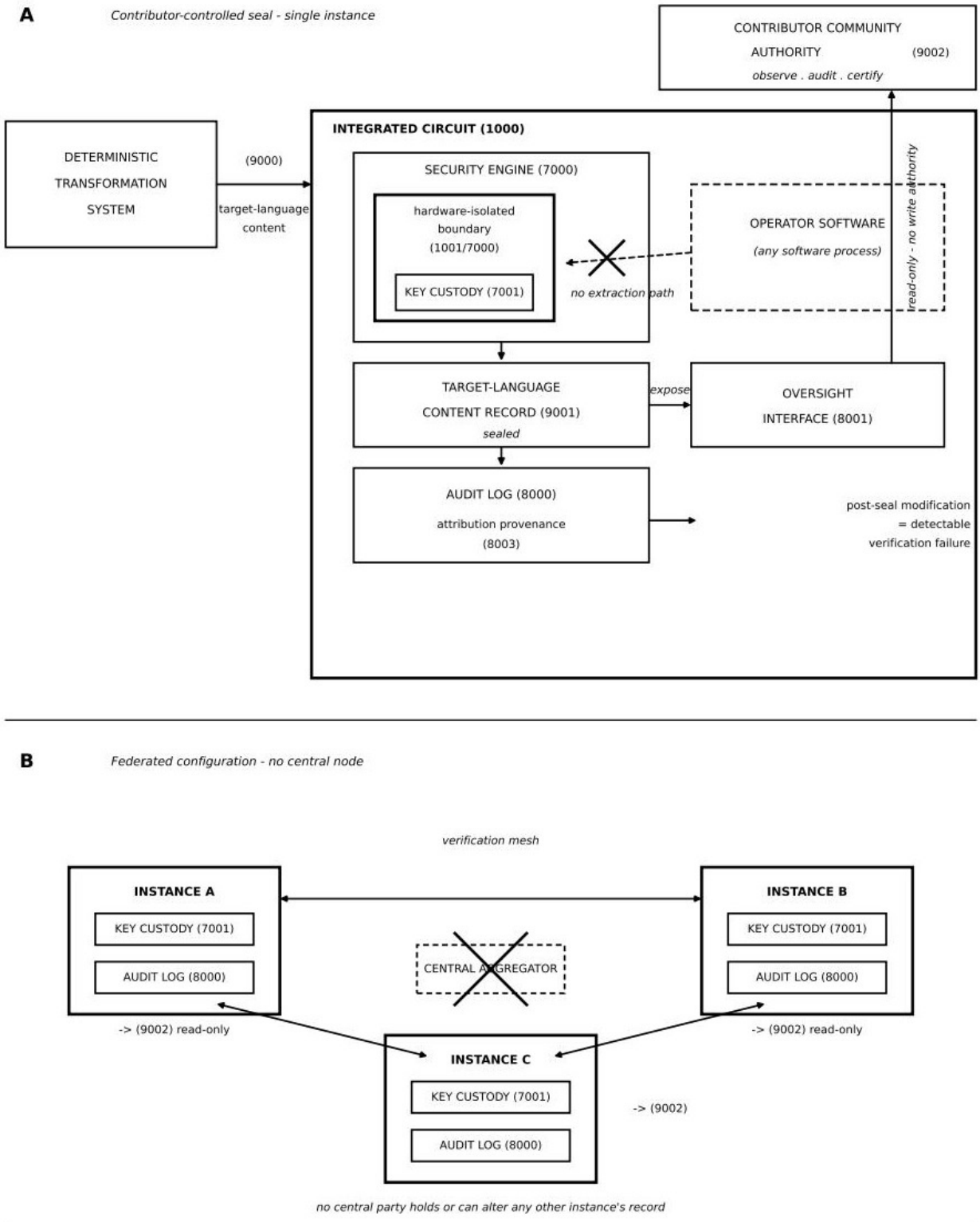


FIG. 51

FIG. 51

CLAIMS

What is claimed is:

- A neuromorphic logic integrated circuit (1000) comprising: a decision module (4000) configured to compute decision vectors by executing a fixed optimization process, the fixed optimization process being invariant across successive executions given identical input state and being a hardware-structural property of the circuit not configurable at runtime by any software process; a spiking neural network array (2000) comprising a plurality of processing cores producing spike traffic encoding learned behavioral representations, spike traffic produced by each core being individually isolatable at a spike bus interface, the isolation enforced by the hardware architecture of the spike bus interface; a hardware anomaly detection module (5000) configured to monitor the spike traffic at an inference rate of the spiking neural network array and to detect a deviation from a verified normal operational envelope before the deviation propagates to an output boundary of the circuit; a fault self-healing module (6000) configured to execute, upon detection of the deviation, a non-reorderable correction sequence; a security engine (7000) configured to produce a cryptographically sealed audit record for each decision vector at the moment of production, the audit record sealed using key material inaccessible to any software process executing on the circuit; and an oversight interface (8001) configured to expose the sealed audit record to an external governance authority for independent verification without requiring operator intermediation.

- The circuit of claim 1, wherein the fixed optimization process comprises a quantum-inspired quadratic unconstrained binary optimization (QUBO) annealing process (4002) executed by deterministic digital hardware of the decision module.

- The circuit of claim 1, wherein the decision module (4000) comprises a variational quantum eigensolver (4001) and a quantum annealing solver (4002).

- The circuit of claim 1, wherein the hardware anomaly detection module (5000) comprises a variational autoencoder computing a reconstruction error against a dynamically adapted threshold to produce a binary determination (5002), and wherein a positive binary determination is the exclusive trigger for activation of the fault self-healing module (6000).

- The circuit of claim 1, wherein the hardware anomaly detection module detects the deviation at an internal structural point in the causal chain prior to output-boundary propagation, the anomaly so detected constituting a distinct fault class from anomalies detectable only at the output boundary.

- The circuit of claim 1, wherein spike traffic isolation at the spike bus interface prevents any output produced by an anomalous core from reaching the output boundary of the circuit.

- The circuit of claim 1, further comprising a memristive synaptic array (3000) storing synaptic weights as conductance states, updated by spike-timing-dependent plasticity (2002), wherein a pre-anomaly hardware state S_0 (3003) comprises the conductance states, weight parameters, and routing tables of the memristive synaptic array.

- The circuit of claim 7, wherein S_0 (3003) is preserved as a checkpoint in hardware-protected memory inaccessible to any software process, is cryptographically sealed at the moment of checkpoint creation, and is verified against its seal before any restoration is executed.

- The circuit of claim 1, wherein a correction latency, defined as the elapsed time between detection and restoration to S_0, is bounded by a hardware-structural property of the fault self-healing module and is not configurable at runtime by any software process.

- The circuit of claim 1, wherein the security engine (7000) seals each audit record using a lattice-based post-quantum key encapsulation mechanism, all key material generated and confined within a hardware-isolated execution environment of the security engine.

- The circuit of claim 1, wherein any modification to a sealed audit record after production produces a detectable cryptographic verification failure observable through the oversight interface (8001).

- The circuit of claim 1, wherein during execution of the correction sequence a redundant core array (6006) continues producing decision output at the output boundary such that the output stream of the circuit is not interrupted by the correction event.

- The circuit of claim 1, wherein the oversight interface (8001) is read-only to the external governance authority, the authority observing, auditing, and certifying circuit behavior without possessing write authority over any operational parameter of the circuit.

- A fault correction architecture for an integrated circuit, comprising: a plurality of processing cores producing computational traffic individually isolatable at a hardware bus interface, the isolation enforced by the hardware architecture of the bus interface; a redundant core array (6006); a hardware anomaly detection module (5000) configured to monitor the traffic at an inference rate of the processing cores and to detect a deviation before the deviation propagates to an output boundary; and a fault self-healing module (6000) configured to execute, upon detection, a non-reorderable correction sequence comprising, in order: confirming the detected deviation as an anomaly; suspending workload routing to an anomalous core; activating the redundant core array; transferring execution to the redundant core array; and isolating the anomalous core at the bus interface and restoring the circuit to a cryptographically verified pre-anomaly hardware state S_0 (3003); wherein $S_{rollback} = S_0$ exactly, no intermediate patched state is produced, and output continues from the redundant core array without interruption during execution of the correction sequence.

- The architecture of claim 14, wherein the order of the correction sequence is enforced as a structural property of a hardware type system in which the output type of each step is the required input type of its successor, whereby execution of any step out of order constitutes an undefined operation rather than an incorrect result.

- The architecture of claim 14, wherein the correction sequence consists of five steps, each gated by completion of its predecessor, execution of any step before completion of its predecessor constituting an undefined operation in the hardware architecture of the fault self-healing module.

- A method for hardware-enforced decision execution in an integrated circuit, comprising: computing, by a decision module (4000), a decision vector by executing a fixed optimization process invariant across successive executions given identical input state; monitoring, by a hardware anomaly detection module (5000) executing at an inference rate of a spiking neural network array (2000), spike traffic for a deviation from a verified normal operational envelope; upon detection, executing a non-reorderable correction sequence comprising, in order: confirming the deviation as an anomaly; suspending workload routing to an anomalous core; activating a redundant core array (6006); transferring execution to the redundant core array;

and isolating the anomalous core and restoring the circuit to a cryptographically verified pre-anomaly state S_0 (3003) without producing any intermediate patched state; producing, by a security engine (7000), a cryptographically sealed audit record for each decision vector at the moment of production using key material inaccessible to any software process; and exposing the sealed audit record through an oversight interface (8001) to an external governance authority for independent cryptographic verification.

- The method of claim 17, wherein the fixed optimization process comprises a quantum-inspired QUBO annealing process (4002) executed by deterministic digital hardware.

- The method of claim 17, wherein restoring the circuit to S_0 produces no intermediate patched state, such that S_rollback = S_0 exactly.

- The method of claim 17, wherein the external governance authority verifies the sealed audit record through the oversight interface without operator intermediation and without write authority over any operational parameter of the circuit.

- A hardware-sealed governance subsystem for an integrated circuit (1000), comprising: a security engine (7000) producing a cryptographically sealed record of each decision produced by the circuit, sealed using key material generated and confined within a hardware-isolated execution environment and inaccessible to any software process executing on the circuit; and an oversight interface (8001) exposing each sealed record to a governance authority external to the circuit operator, the interface being read-only to the authority such that the authority verifies, audits, and certifies without operator intermediation and without write authority over any operational parameter; wherein the combination of hardware-confined key material, read-only external-authority verification, and absence of operator intermediation is a hardware-structural property of the subsystem and not a runtime-configurable parameter.

- The subsystem of claim 21, wherein a plurality of circuit instances in federated configuration each maintain an independent sealed record verifiable through their respective oversight interfaces without aggregation of records in a centralized database, and wherein no instance can modify, suppress, or fabricate the sealed record of any other instance.

- The subsystem of claim 21, wherein any modification to a sealed record after production produces a detectable cryptographic verification failure observable through the oversight interface.

- The subsystem of claim 21, wherein the key material is sealed under a lattice-based post-quantum key encapsulation mechanism.

- A system comprising the integrated circuit (1000) of claim 1 in combination with a deterministic transformation system producing structured content outputs, wherein the circuit provides hardware-enforced decision execution for assessment and pathway determinations applied to content produced by the deterministic transformation system, and wherein the combination produces a cryptographically verifiable record of both content generation and decision execution.

- The system of claim 25, wherein the verifiable record of both content generation and decision execution is sealed by the security engine (7000) and exposed through the oversight interface (8001) to an external governance authority for independent verification without operator intermediation.

- The circuit of claim 1, wherein the decision module (4000) is realized entirely in deterministic digital hardware and the circuit is fabricable by complementary metal-oxide-semiconductor (CMOS)-compatible semiconductor processes without a physical quantum-computing element.

- The circuit of claim 1, wherein the spiking neural network array (2000) is realized on a neuromorphic processing substrate and the memristive synaptic array (3000) is realized as a resistive memory array integrated therewith.

- The circuit of claim 1, wherein the decision module (4000) comprises a physical quantum processing element implementing the fixed optimization process, the remainder of the circuit being unchanged.

- The subsystem of claim 21, wherein the hardware-isolated key material is held under the authority of a contributor external to the circuit operator, such that the operator cannot access the sealed content under any software process — whereby authority to control the sealed content is a hardware-structural property and not a contractual or policy condition.

- The subsystem of claim 30, in federated configuration across a plurality of contributor authorities, each authority holding the key material for its own sealed records, no central authority holding key material for or able to alter the records of any other.

- The subsystem of claim 30, wherein the read-only external governance authority is the contributor community, which observes, audits, and certifies use of its sealed content while the circuit operator is the audited party without write authority.

- The system of claim 25, wherein the content produced by the deterministic transformation system is in a target language, the sealed verifiable record of that content has its decryption and attribution authority confined to a contributor-held key, and a plurality of instances are federated such that no instance can alter the record of any other.

- The system of claim 33, wherein each contributed element of the target-language content carries cryptographically non-erasable provenance through the sealed record, such that post-seal modification or deletion of attribution is a detectable cryptographic verification failure.

- The system of claim 33, wherein the deterministic transformation system produces a native instantiation of source content in the target language and the sealed record preserves the instantiation under the contributor-held key.

ABSTRACT

A neuromorphic logic integrated circuit provides hardware-enforced behavioral guarantees unreachable by software-mediated architectures. A decision module computes decision vectors by a fixed optimization process invariant across executions. A variational autoencoder monitors spike traffic at the inference rate of the spiking neural network array, detecting deviation from a verified normal operational envelope before anomalous output reaches the output boundary. A fault self-healing module executes a non-reorderable five-step correction sequence upon anomaly detection, suspending and transferring execution to a redundant array, isolating the affected core, and restoring the circuit to a cryptographically verified pre-anomaly state S_0 with no intermediate patched state. A security engine produces a cryptographically sealed audit record for each decision vector using key material inaccessible to any software process. An oversight interface exposes sealed records to external governance authorities for independent verification without operator intermediation. Autonomous operation and governance

accountability are simultaneous structural properties of the hardware architecture.