NELIX CHIP DESIGN

Attested Infrastructure Inference

Defining the Architecture Class Where Proof, Power, and Completion Matter More Than Tokens per Second

Public Technical Whitepaper · Version 1.0 · July 2026

Defining the Architecture Class Where Proof, Power, and Completion Matter More Than Tokens per Second

Abstract

The AI inference market is splitting in two. One economy optimizes **tokens per second** in air-conditioned datacenters breaking the memory wall with digital in-memory compute, dataflow pipelines, and rack-scale disaggregation. A second economy optimizes **completed, trustworthy inference** at the infrastructure edge under joule budgets, intermittent connectivity, physical tampering risk, and decade-long service lifecycles.

This whitepaper defines **Attested Infrastructure Inference (AII)** as a distinct architecture class and presents the Nelix platform built for it: **InferEdge™** joule-bounded AI inference accelerators governed by the **Inference Completion State Machine (ICSM)**, **TrustCore™** continuous hardware root of trust, **SecureGrid™** infrastructure SoCs, and the **Nelix Compute Platform** for proof-carrying fleet operations.

Nelix argues that the next bottleneck in infrastructure AI is not FLOPS — it is **trust**. Operators deploying AI into grids, factories, and public systems need cryptographic proof that inference ran on genuine hardware with approved firmware and models. Nelix delivers **Proof-Carrying Inference** outputs the inference result plus an attestation bundle binding the result digest to its authorized execution context alongside INT8/INT4 efficiency in sub-15 W envelopes.

Document Scope and Validation Status

This whitepaper is a public technical and architectural overview for infrastructure operators, OEM partners, systems integrators, semiconductor engineers, security architects, and technical investors.

It is **not** an offering document, procurement specification, or benchmark report. Unless stated as measured results, numerical values are **design targets** or **future design objectives**. The platform is pre-silicon at the time of writing.

This document distinguishes:

Status	Meaning
Architecture definition	Specified mechanism (e.g., ICSM, Energy Contract, checkpoint model)
RTL in progress	Hardware description under development
FPGA validation	Active bring-up on programmable platform
Design target	Engineering objective; not measured silicon
Future design objective	Planned capability not yet architecturally finalized
Measured silicon result	Stated only when backed by characterization data

Table of Contents

Part I — Category

1. Executive Summary

Thesis: Infrastructure AI does not need more datacenter accelerators at the edge. It needs a new architecture class — Attested Infrastructure Inference — where every completed inference carries proof of how, where, and on what hardware it ran.

The inference hardware industry has entered its most creative phase since the GPU era. d-Matrix breaks the memory wall with **Digital In-Memory Compute (DIMC™)** at rack scale. SambaNova eliminates kernel-by-kernel overhead with **Reconfigurable Dataflow Architecture**. Groq delivers **deterministic, compiler-orchestrated** tensor streaming. Etched hardwires the transformer into silicon. Rebellions scales **CGRA chiplets** with HBM for hyperscale decode. Axelera and Untether bring **digital in-memory compute** to the edge at 15+ TOPS/W.

Each of these architectures solves a real problem. Each is optimized for a specific inference economy. **None of them is optimized for a revenue meter on a distribution pole.**

Nelix Chip Design Ltd. is building silicon and system architecture for the second inference economy: mission-critical infrastructure where AI must run **locally**, **within a joule budget**, **without cloud dependency**, and **with cryptographic accountability** for ten to twenty years in environments with intermittent power, constrained bandwidth, high ambient heat, and physical tampering risk.

Nelix calls this class **Attested Infrastructure Inference (AII)** and delivers it through four integrated layers:

Layer	Innovation
InferEdge™	Joule-Bounded Inference Architecture (JBIA) — ICSM-managed completion transactions under Energy Contract
TrustCore™	Trust Continuum — continuous hardware trust from manufacture to cryptographic retirement
SecureGrid™	Inference Integrity Chain — measurement and inference inside one trust domain
Nelix Compute Platform	Proof-Carrying Fleet Operations — verification of attestation bundles before trusted workflow admission

Nelix is headquartered in Accra, Ghana. The platform is designed for deployment realities common across the Global South and industrial markets worldwide — because an architecture that survives African grid and field conditions is globally portable; the reverse is rarely true.

Deploy trusted inference on a foundation of proof.

2. The Great Inference Bifurcation

AI inference is no longer one market. It is two — with different physics, economics, and success metrics.

2.1 Datacenter Inference Economy

Design center: Maximize useful tokens per dollar per watt in controlled facilities.
Dominant constraints: Memory bandwidth (the "memory wall"), inter-GPU communication, cooling capacity, rack power delivery.
Architectural responses (2024–2026):

Company	Named architecture	Core novelty
d-Matrix	DIMC™ (Digital In-Memory Compute)	Multiplier integrated into SRAM bit cells; 150–300 TB/s on-chip bandwidth; chiplet scale-out; decode-phase disaggregation alongside GPUs
SambaNova	Reconfigurable Dataflow Unit (RDU)	Model mapped as spatial pipeline; three-tier memory (SRAM/HBM/DDR); eliminates redundant kernel launches
Groq	Tensor Streaming Processor (LPU)	Compiler-orchestrated determinism; no reactive hardware; predictable power and zero tail latency
Etched	Sohu (Transformer ASIC)	Transformer operations burned into silicon; extreme throughput for one architecture class
Rebellions	REBEL CGRA + chiplet	Coarse-grained reconfigurable array; UCIe-Advanced chiplets; 144 GB HBM3E; decode-stage specialization
Cerebras	Wafer-scale engine	Massive on-wafer SRAM; eliminates chip-to-chip data movement

Shared assumption: Abundant power (hundreds of watts to kilowatts per accelerator), persistent connectivity, attended facilities, replacement cycles measured in years not decades.

2.2 Infrastructure Inference Economy

Design center: Complete trusted inference locally under harsh, unattended field conditions.
Dominant constraints: Joule budget per inference (the **joule wall**), trust and provenance (the **trust wall**), service lifecycle, physical security.
Architectural gap: Edge AI vendors (Axelera, Hailo, Kinara) optimize TOPS/W for vision workloads but typically treat security as secure boot bolted on, not **proof-carrying inference** integrated with fleet lifecycle over decades.
Nelix occupies this second economy deliberately. The comparison is not "Nelix vs d-Matrix on tokens/sec." The comparison is whether inference **completes, proves itself, and survives** when the cloud is gone and the adversary is physical.

Datacenter inference optimizes throughput under abundance. Infrastructure inference optimizes completion and proof under scarcity.

3. Attested Infrastructure Inference (AII): A New Architecture Class

Attested Infrastructure Inference (AII) is semiconductor and system architecture optimized for AI inference in mission-critical physical infrastructure where intermittent power, constrained connectivity,

harsh environment, physical tampering, and long deployment lifecycles are **expected**, and where inference outputs must carry **cryptographic evidence** of their origin.

AII is not edge computing as commonly marketed. Edge computing describes *where* compute happens. AII describes *under what conditions* inference must remain **dependable and provable**.

3.1 The Three Design Laws of AII

Law 1 — Completion over throughput.

Inference must finish within available energy as a managed transaction — governed by ICSM states and an accepted Energy Contract. A result that requires more joules than the node can supply is not a slow result; it is a failed or aborted transaction.

Law 2 — Proof over assumption.

Inference outputs must be accompanied by verifiable attestation evidence binding the result digest to hardware identity, firmware state, model authorization, and runtime integrity. Unattested inference is untrusted inference.

Law 3 — Lifecycle over deployment.

Devices operate for ten to twenty years. Architecture must support secure update, rollback, quarantine, and cryptographic retirement — not one-time secure boot.

3.2 Negative-Assumption Design

Nelix applies **negative-assumption design**: every decision is tested against the worst realistic field condition.

- Power **will** fail mid-inference and mid-update.
- Connectivity **will** be absent for months.
- Physical access **will** occur on exposed assets.
- Replacement **will** be economically infeasible for a decade or more.

Designs requiring optimistic conditions are rejected regardless of benchmark appeal.

3.3 Relationship to Infrastructure Compute™

Nelix previously defined **Infrastructure Compute™** as the broader semiconductor category for mission-critical edge deployment. **Attested Infrastructure Inference** is the AI-specific specialization of that category where the workload is inference, and the non-negotiable output is **proof**.

4. What the Industry Is Optimizing — And What It Is Missing

4.1 The Memory Wall (Solved — in Datacenters)

d-Matrix's Corsair platform demonstrates the industry consensus: the dominant cost in datacenter inference is **moving data between memory and compute**. DIMC integrates multiply logic into SRAM, achieving bandwidth orders of magnitude above HBM. SambaNova's dataflow architecture achieves the same goal spatially keeping activations local across operator pipelines.

Nelix adopts the same physical insight at a different scale: data movement dominates edge energy cost. InferEdge keeps weights and activations in on-die SRAM through systolic operand reuse and compute-in-memory proximity but optimizes for **joules per completed inference**, not terabytes per second.

4.2 The Joule Wall (Largely Unaddressed)

Cloud accelerators are rated in hundreds of watts. Infrastructure nodes target **sub-15 W** for always-on inference — often powered by solar, battery, or sagging grid supply.

No datacenter architecture solves this. Joule-bounded inference — scheduling model tier, tile size, and clock profile to **finish within available energy** is an architectural requirement, not a power-management afterthought.

4.3 The Trust Wall (Largely Unaddressed)

As AI outputs influence billing, safety, and operational control, the question is no longer "what did the model predict?" but "can I prove this prediction came from approved hardware running an approved model?"

Secure boot on an edge NPU is necessary but insufficient. Infrastructure requires **continuous trust** across boot, runtime, update, and retirement with **inference attestation** as a first-class platform service.

4.4 Positioning Summary

Dimension	Datacenter inference (d-Matrix, SambaNova, Groq)	Edge AI (Axelera, Hailo)	Nelix AII
Power envelope	300 W – 600 W+ per card	8–25 W	Sub-15 W target
Primary metric	Tokens/sec, TOPS	TOPS/W	Joules/inference + attestation
Memory strategy	DIMC / HBM / wafer SRAM	D-IMC, on-chip SRAM	SRAM-local systolic; no external DRAM
Trust model	Datacenter perimeter	Secure boot	Trust Continuum + proof-carrying output
Connectivity	Required for scale	Helpful	Offline-first; opportunistic sync
Lifecycle	Years	Years	10–20 years; cryptographic retirement
Workload focus	LLM decode, agents	Vision, CV	Quantized INT8/INT4 infrastructure models

Part II — Architecture

5. The Nelix Attested Inference Stack

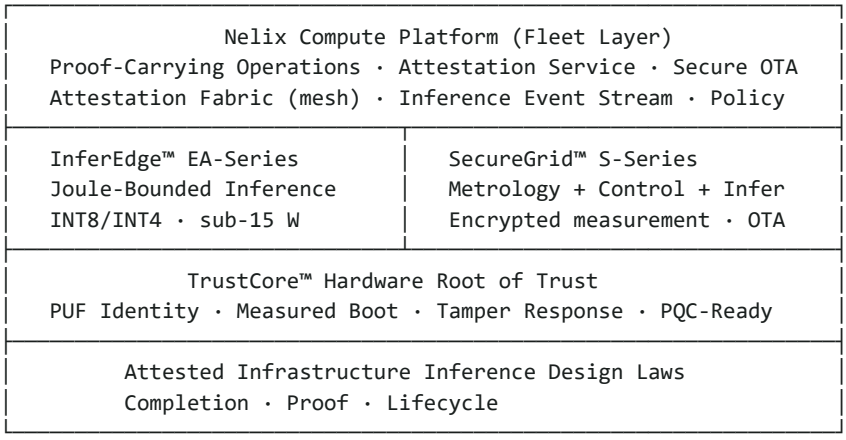


Figure 1 — Nelix Attested Inference Stack

Caption: TrustCore is instantiated inside each silicon product. InferEdge and SecureGrid are product families on a shared trust foundation. The Compute Platform provides proof-carrying fleet operations above silicon.

Platform Reuse: Security IP validated in SecureGrid is reused in InferEdge. Software foundations — secure boot, OTA agents, SDK — span the product family. Each validated block compounds into the next product.

6. InferEdge: Joule-Bounded Inference Architecture

InferEdge™ implements **Joule-Bounded Inference Architecture (JBIA)** — Nelix's named inference execution architecture for infrastructure deployment.
Where d-Matrix's DIMC breaks the memory wall at datacenter scale, JBIA breaks the **joule wall** at infrastructure scale: inference is scheduled to **complete within an energy envelope**, not merely maximize throughput.
JBIA does not treat inference as a stateless kernel invocation. JBIA treats inference as a **managed completion transaction**. The architecture owns the inference from attested input staging through execution, interruption, recovery, completion, proof generation, and export.

In JBIA, inference is a stateful completion transaction. The architecture is responsible not merely for executing tensor operations, but for preserving the integrity, progress, energy contract, and trust state of the inference until the transaction is either completed and proven or explicitly terminated as untrusted.

6.1 Architectural Composition: Conventional Techniques and Nelix Thesis

JBIA employs established semiconductor and security techniques. Nelix does not claim to have invented these individually:

Technique	Role in JBIA
Systolic MAC arrays	INT8/INT4 tensor execution with operand reuse
Quantization (INT8/INT4)	Energy- and memory-efficient edge model execution
On-die SRAM buffering	Tile-local staging; reduced external data movement
RISC-V control core	Host scheduling, ICSM orchestration, peripheral integration
Secure boot	Firmware integrity at power-on
Hardware root of trust	Identity, key derivation, signing isolation
Checkpointing (concept)	Progress preservation across interruption

Nelix's architectural contribution is the **composition** of these mechanisms into a unified inference completion architecture:

Nelix's architectural contribution is the composition of energy-constrained scheduling, transactionally consistent inference recovery, continuous hardware-rooted trust, and proof-bound output into a unified inference completion architecture for infrastructure systems.

A concise formulation:

Nelix makes inference completion and cryptographic proof part of the architecture transaction itself.

Individual datapath elements are conventional. The **completion transaction** — with explicit execution states, energy contracts, checkpoint consistency semantics, and attestation transaction boundaries — is the architectural thesis.

Implementation status: JBIA is defined at the architecture level; InferEdge RTL is in progress; FPGA validation is active. Unless stated otherwise, mechanisms described in this section are architectural definitions, not measured silicon behavior.

6.2 JBIA Architectural Components

The following table defines JBIA functional blocks. Plane labels indicate architectural responsibility; not every function requires a dedicated hardware block in a given implementation.

Component	Plane	Function
Systolic Compute Fabric	Datapath	INT8/INT4 tensor execution with operand reuse
SRAM-Local Memory Hierarchy	Datapath	Reduce external data movement; preserve tile-local execution state
Energy-Aware Scheduler	Control	Establish and renegotiate the Energy Contract (Section 6.4)
Inference Completion State Machine (ICSM)	Control	Own execution state from attested staging through completion, recovery, proof, or explicit abort
Checkpoint Engine	Control + Trust	Commit transactionally consistent inference state at architecturally safe boundaries
Restore Validation Engine	Control + Trust	Authenticate checkpoint state; validate execution compatibility before resume
TrustCore Gateway	Trust	Isolated identity, trust-state measurement, checkpoint authentication, result signing
Proof Assembly Path	Trust	Bind result digest and execution context into the attestation bundle
Offline Result Queue	Platform	Preserve Proof-Carrying Inference outputs for opportunistic synchronization when connectivity is unavailable

6.3 Inference Completion State Machine (ICSM)

The **Inference Completion State Machine (ICSM)** is the formal control mechanism that manages each inference transaction through JBIA. Every inference request is assigned a transaction identifier at entry; ICSM governs all state transitions until the transaction completes with proof, suspends for recovery, or terminates as untrusted.

6.3.1 Nominal State Progression

```
IDLE
  → ATTEST_INPUT
  → ENERGY_ASSESS
  → SCHEDULE
  → EXECUTE
  → COMPLETE
  → PROVE
  → EXPORT
  → IDLE
```

State	Architectural responsibility
IDLE	Await attested inference request
ATTEST_INPUT	Validate TrustCore trust state; stage input; assign transaction identifier

ENERGY_ASSESS	Evaluate available energy; draft Energy Contract (Section 6.4)
SCHEDULE	Bind approved model tier; configure tile geometry and clock profile; accept contract
EXECUTE	Run tiled inference under active contract
COMPLETE	Finalize inference result; derive canonical result digest; verify completion under contract
PROVE	Bind result digest to execution context; assemble and sign attestation bundle (Section 7)
EXPORT	Emit Proof-Carrying Inference output (inference result + attestation bundle)

6.3.2 Energy Degradation and RETIER Semantics

During **EXECUTE**, if available energy falls below the contract reserve, ICSM transitions to ENERGY_DEGRADED. Model-tier changes follow two distinct rules. JBIA does **not** permit arbitrary mid-inference model substitution.

Rule A — Pre-execution model selection

Before ICSM enters EXECUTE, the scheduler may select any pre-authorized model tier permitted by fleet policy and compatible with the drafted Energy Contract:

```
ENERGY_ASSESS → SCHEDULE → SELECT APPROVED MODEL TIER → ACCEPT ENERGY CONTRACT → EXECUTE
```

This is the normal model-tier selection path.

Rule B — Mid-execution RETIER

Once EXECUTE has begun, switching model tiers is permitted **only** when the active and target model artifacts explicitly declare an architecturally compatible transition boundary in signed model metadata. Examples include shared feature representation, compatible intermediate tensor shape, defined early-exit boundary, shared execution frontier, or an explicitly validated continuation point.

Not all model variants support mid-execution RETIER.

> *Mid-execution RETIER is not arbitrary model substitution. A transaction may transition between approved model tiers only at a model-defined compatible transition boundary declared in signed model metadata and validated by the runtime. If no compatible transition boundary exists, JBIA must checkpoint or terminate the active transaction and begin a new transaction using the lower approved model tier.*

Signed model packages may conceptually declare:

```
ModelTransitionMetadata {
  source_model_id
  target_model_id
  compatible_boundary
  tensor_schema
  quantization_profile
  transition_policy
}
```

This is a **conceptual architectural object**, not a finalized software ABI. Mid-execution transition compatibility is an architecture definition; demonstrated behavior is subject to FPGA and silicon validation.

Energy degradation paths

From EXECUTE:

```
EXECUTE → ENERGY_DEGRADED
```

Then:

Path 1 — RECLOCK (same model tier):

```
ENERGY_DEGRADED → RECLOCK → [Energy Contract renegotiation] → EXECUTE
```

Path 2 — RETIER (only if compatible transition exists):

```
ENERGY_DEGRADED → RETIER → VALIDATE_TRANSITION_BOUNDARY → [Energy Contract renegotiation] →
EXECUTE
```

Path 3 — no compatible transition:

```
ENERGY_DEGRADED → CHECKPOINT_PENDING or ABORT_UNTRUSTED
```

If policy and available energy permit, a **new transaction** may follow:

```
NEW TRANSACTION → ENERGY_ASSESS → SCHEDULE LOWER APPROVED MODEL TIER → EXECUTE
```

Energy Contract renegotiation is an ICSM **control action** at safe tile or layer boundaries — not a separate permanent ICSM state. Execution does not proceed under an invalidated contract. See **INVARIANT 6** (Section 7.2).

Architecture Note — Model-Tier Transition Compatibility

Model tiers are independent authorized model artifacts unless explicit transition compatibility is declared. A shared application purpose does not imply shared intermediate execution state.
For example, a "full anomaly model" and an "emergency anomaly model" may both classify grid anomalies but may differ in layer structure, tensor dimensions, quantization profiles, and internal representations.

JBIA must not infer execution compatibility from model function alone.

Mid-execution transition requires signed metadata defining a compatible transition boundary. Otherwise the current transaction must be checkpointed, suspended, or terminated, and a new transaction must begin under the alternate model tier.

6.3.3 Checkpoint and Suspend Transitions

```
EXECUTE → CHECKPOINT_PENDING → CHECKPOINT_PREPARE → CHECKPOINT_COMMIT → SUSPEND
```

On power return:

```
POWER RETURN → [TrustCore boot validation] → RESTORE_VALIDATE → [current energy assessment]
  → [Energy Contract compatibility evaluation] → RESTORE → EXECUTE
```

`CHECKPOINT_PENDING` is entered when imminent power loss, thermal policy, or scheduler policy requires progress preservation. Commit occurs only at a safe boundary (Section 6.6). On restart, `RESTORE_VALIDATE` and Energy Contract compatibility evaluation (Section 6.4.2) complete before any execution state is restored.

6.3.4 Trust and Abort Transitions

On any trust-integrity violation during an active transaction:

```
ANY ACTIVE STATE → SUSPECT_TAMPER → QUARANTINE
```

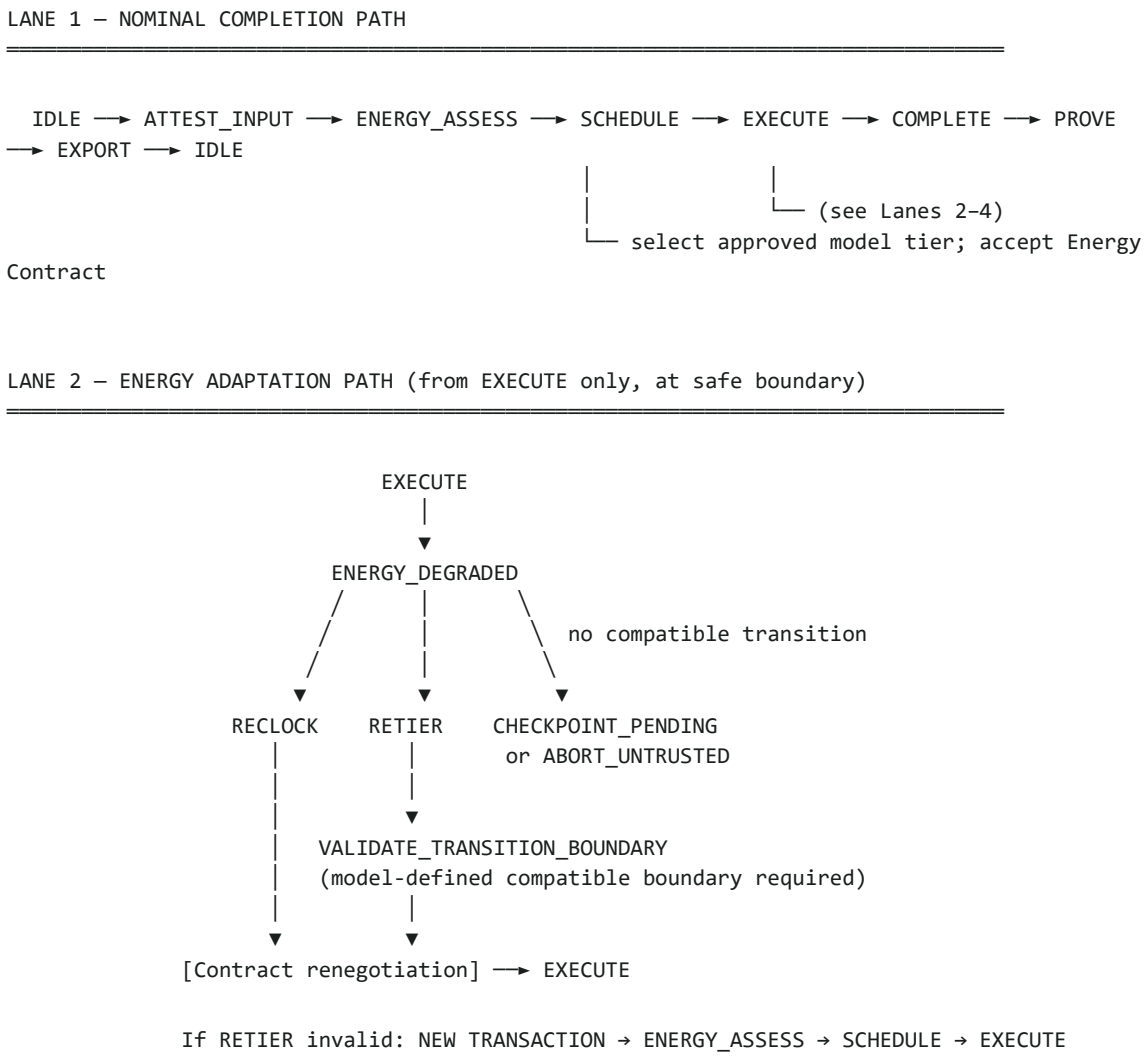
On unrecoverable execution-integrity failure:

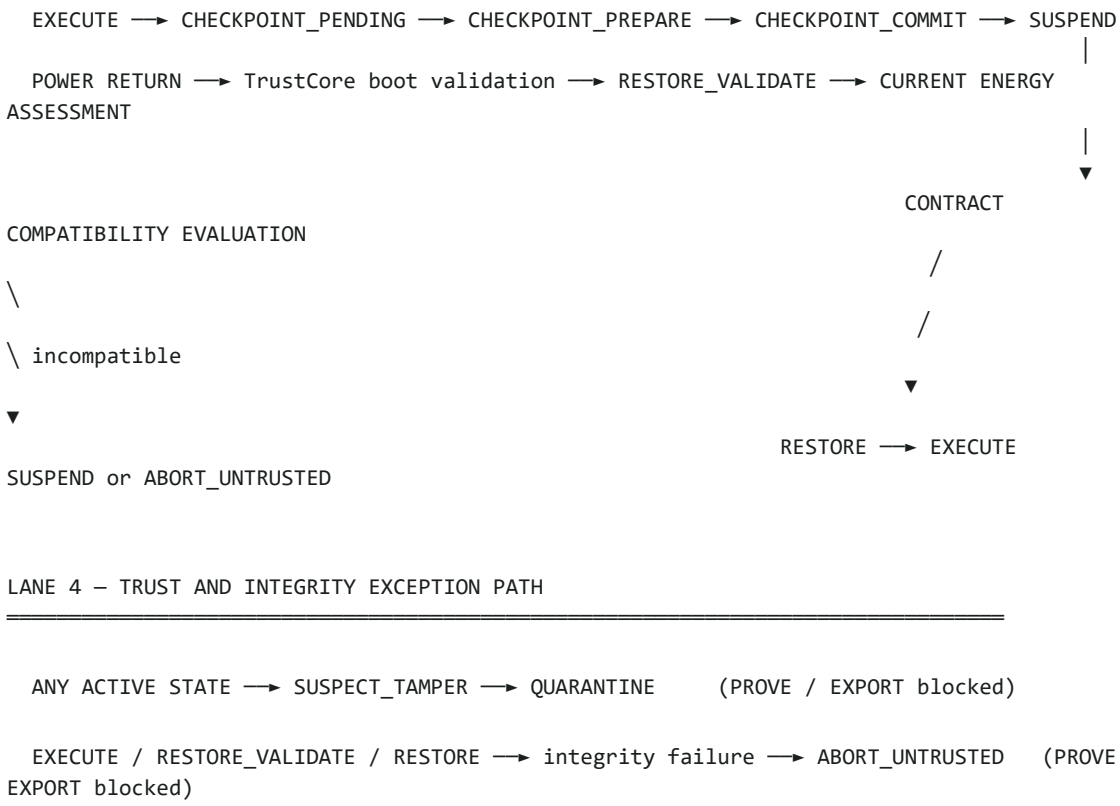
```
EXECUTE or RESTORE_VALIDATE or RESTORE → ABORT_UNTRUSTED
```

ABORT_UNTRUSTED terminates the transaction without trusted export. PROVE and EXPORT are blocked. A security/fleet event is emitted when connectivity permits.

Figure 2 — JBIA Inference Completion State Machine

Caption: ICSM owns each inference transaction from attested input staging through completion, recovery, proof, trusted export, or explicit untrusted termination. Lane 1: nominal completion path (left to right). Lane 2: energy adaptation. Lane 3: checkpoint and recovery. Lane 4: trust and integrity exceptions. Trust-plane transitions are mediated through TrustCore Gateway.





6.4 Energy Contract and Completion Budget

Before ICSM enters **EXECUTE**, the Energy-Aware Scheduler and runtime establish an **Energy Contract** — the execution constraints under which the transaction is permitted to proceed. The contract is evaluated at ENERGY_ASSESS and accepted at SCHEDULE. Renegotiation may occur at safe boundaries when infrastructure conditions change.

The Energy Contract is a **conceptual architectural object**. It is not a finalized software ABI unless and until specified in a separate implementation document.

```
EnergyContract {
  energy_budget_joules
  reserve_joules
  model_tier
  precision_mode
  tile_geometry
  clock_profile
  checkpoint_policy
  latency_bound
  thermal_state
  trust_policy
}
```

Field	Purpose
energy_budget_joules	Estimated energy available for this transaction
reserve_joules	Minimum reserve; falling below triggers ENERGY_DEGRADED

`model_tier`	Pre-authorized, cryptographically approved model variant
`precision_mode`	INT8 or INT4 per approved quantization profile
`tile_geometry`	Tile dimensions for systolic execution
`clock_profile`	Operating frequency envelope
`checkpoint_policy`	Checkpoint interval or event-triggered policy
`latency_bound`	Maximum permitted completion latency under contract
`thermal_state`	Active thermal operating class
`trust_policy`	Fleet trust policy identifier governing export rules

6.4.1 Energy Operating Classes

Class	Typical response
NORMAL ENERGY	Full approved model tier · INT8 · nominal clock profile
DEGRADED ENERGY	Reduced approved model tier · INT8/INT4 where validated · lower clock · shorter checkpoint interval
CRITICAL ENERGY	Emergency approved model tier · minimal approved workload · checkpoint-first policy

Model-tier transitions are permitted **only** between pre-authorized, cryptographically signed model artifacts registered in fleet policy, subject to RETIER rules (Section 6.3.2). The architecture **never** dynamically substitutes an unapproved model to meet an energy target. If no authorized tier can complete under available energy, the transaction transitions to `ABORT_UNTRUSTED` or remains suspended — it does not silently degrade trust.

6.4.2 Energy Contract Compatibility

A checkpoint is committed under a historical Energy Contract. Restoration does **not** require the original operating conditions to remain unchanged.

A checkpoint is bound to the Energy Contract under which it was committed, but restoration does not require the original operating conditions to remain unchanged. Before execution resumes, JBIA must evaluate whether the checkpoint state is execution-compatible with a newly accepted Energy Contract.

Energy Contract compatibility is not equality. The original contract does not have to be reproduced exactly. Two contracts are **execution-compatible** when a proposed contract revision preserves the validity of the committed execution state:

- Changes to **clock profile**, **checkpoint policy**, or **tile scheduling** may be compatible where restored tensor and scheduler state remain valid.
- Changes to **model identity**, **intermediate tensor schema**, **quantization assumptions**, or **execution frontier** require explicit transition compatibility per Section 6.3.2.

Contract renegotiation must never invalidate the semantic meaning of the committed inference state.

Restore flow with contract evaluation:

```
POWER RETURN → TRUSTCORE BOOT VALIDATION → RESTORE_VALIDATE → CHECKPOINT AUTHENTICATION
```

→ MODEL AND SEQUENCE VALIDATION → CURRENT ENERGY ASSESSMENT → CONTRACT COMPATIBILITY EVALUATION

Then:

Outcome	Response
Compatible current conditions	Accept existing contract or compatible contract revision → RESTORE → EXECUTE
RECLOCK / tile change required	Negotiate compatible Energy Contract revision → validate execution compatibility → RESTORE → EXECUTE
Model-tier change required	Apply model-transition compatibility rules; if valid boundary exists → validate → accept new contract → RESTORE / transition → EXECUTE
No compatible transition	Do not resume original transaction under a different model; preserve or close per policy; optional new transaction
No valid contract	Remain SUSPEND or → ABORT_UNTRUSTED per policy

Architecture definition; contract evaluation behavior subject to FPGA and silicon validation.

6.5 Failure-Aware Execution Semantics

JBIA defines failure behavior as part of the architecture, not as exceptional software handling layered on top of a stateless kernel.

> *Failure is not an exception outside the inference architecture. In JBIA, failure conditions are explicit execution inputs with defined state transitions and trust consequences.*

Condition	Architectural response
Energy degradation	ICSM → ENERGY_DEGRADED; RECLOCK or RETIER per Section 6.3.2; Energy Contract renegotiation at safe boundary
Thermal degradation	RECLOCK or CHECKPOINT_PENDING / SUSPEND per thermal policy and active Energy Contract
Connectivity loss	Continue local inference; queue Proof-Carrying Inference outputs in Offline Result Queue
Imminent power failure	ICSM → CHECKPOINT_PENDING; commit at next safe boundary
Unexpected power loss	On restart: TrustCore boot validation → RESTORE_VALIDATE → current energy assessment → contract compatibility evaluation → RESTORE or ABORT_UNTRUSTED
Trust-state degradation	Block PROVE and EXPORT into trusted operational workflows; → SUSPECT_TAMPER
Checkpoint corruption	Reject checkpoint; restore from previous valid commit or → ABORT_UNTRUSTED
Restore-integrity failure	Do not resume; → ABORT_UNTRUSTED; emit security/fleet event when connectivity permits

6.6 Checkpoint Consistency Model

Checkpointing in JBIA is not generic memory persistence. The architectural objective is **transactionally consistent inference recovery** — a restored transaction must resume from a state that is internally consistent with its model authorization, Energy Contract, and TrustCore trust state.

6.6.1 Valid Checkpoint Contents

A **committed** checkpoint conceptually binds:

Field	Purpose
Transaction identifier	Unique inference transaction reference
Model identity / hash	Authorized model artifact
Model tier	Active tier at commit boundary
Completed layer or tile boundary	Safe execution frontier
Required activation state	Tensor state needed to resume
Scheduler state	ICSM and scheduler context
Energy Contract version	Contract in force at commit
TrustCore trust state	Trust state snapshot at commit
Monotonic checkpoint sequence number	Ordering and rollback detection
Checkpoint integrity tag	Cryptographic authentication via TrustCore

6.6.2 Two-Stage Commit

Checkpoints are committed only at architecturally defined safe boundaries (tile or layer frontier where execution state is internally consistent).

```
CHECKPOINT_PREPARE
  → capture execution state
  → compute integrity metadata

CHECKPOINT_COMMIT
  → authenticate checkpoint (TrustCore Gateway)
  → advance monotonic sequence number
  → mark checkpoint restorable
```

A partially written checkpoint — prepare without commit — is **never** treated as valid.

6.6.3 Restore Procedure

On restart, `RESTORE_VALIDATE` executes before `RESTORE`. Energy Contract compatibility (Section 6.4.2) is evaluated against **current** energy and thermal conditions — not assumed identical to conditions at checkpoint commit.

1. Identify the latest **committed** checkpoint
2. Validate checkpoint integrity and authenticity
3. Verify model identity and fleet authorization
4. Verify monotonic checkpoint sequence
5. Recover the Energy Contract version associated with the checkpoint
6. Assess current energy and thermal conditions
7. Determine whether the historical contract remains feasible
8. If not, draft a compatible Energy Contract revision
9. Validate execution-state compatibility
10. Apply model-transition rules if model-tier change is requested
11. Restore execution state only after compatibility validation succeeds
12. Resume from the next valid execution boundary

If validation fails → SUSPEND or ABORT_UNTRUSTED. No silent resume from an incompatible execution state.

Checkpoint storage overhead, commit latency, and NVM endurance are design considerations under FPGA and silicon validation. This document does not state measured values.

6.7 Joule-Bounded Scheduling

Joule-bounded scheduling is the control-plane function that drafts, evaluates, and renegotiates the Energy Contract. Inputs include battery state, solar harvest forecast, grid supply monitor, and thermal sensors.

At SCHEDULE, the scheduler:

13. selects an approved model tier from fleet-authorized artifacts
14. configures tile geometry and clock profile within contract bounds
15. sets checkpoint policy per energy operating class
16. accepts the Energy Contract and hands control to ICSM **EXECUTE**

When energy is scarce, JBIA trades latency for completion — through RECLOCK, authorized RETIER at a compatible transition boundary, or checkpoint-first policy — rather than failing silently or deferring to an unavailable cloud.

> *In infrastructure, the right optimization target is joules per completed, attested inference — not peak TOPS in a thermal chamber.*

6.8 End-to-End Execution Example: Grid Anomaly Detection

The following walkthrough illustrates JBIA mechanisms on a remote grid node running INT8 anomaly-detection inference. This example demonstrates **Energy Contract compatibility without model-tier transition**: same model, same committed inference state, compatible lower clock profile, new Energy Contract revision.

Step	Event
1	TrustCore validates active trust state; device is Active
2	Sensor input staged; ICSM: IDLE → ATTEST_INPUT; transaction identifier assigned
3	Energy-Aware Scheduler evaluates supply; drafts **Energy Contract V1** (NORMAL ENERGY, INT8, approved Tier-1 model, nominal clock)
4	Approved Tier-1 model bound to transaction; ICSM: ENERGY_ASSESS → SCHEDULE → EXECUTE
5	InferEdge Systolic Compute Fabric begins tiled inference
6	Supply monitor reports imminent energy-loss condition during tile execution
7	ICSM: EXECUTE → CHECKPOINT_PENDING
8	At next safe tile boundary: CHECKPOINT_PREPARE → CHECKPOINT_COMMIT; ICSM → SUSPEND
9	Energy Contract V1 and execution frontier bound into committed checkpoint
10	Node loses power
11	Power returns; TrustCore re-establishes measured boot state
12	ICSM: RESTORE_VALIDATE — latest committed checkpoint authenticated
13	Model identity, checkpoint sequence, and Energy Contract V1 recovered from checkpoint
14	Energy-Aware Scheduler assesses **current** energy and thermal conditions
15	Current conditions insufficient for original nominal clock profile
16	JBIA drafts **Energy Contract V2** — lower clock profile; same model identity, tensor state, and execution frontier

17	Contract compatibility validation confirms V2 is execution-compatible with committed checkpoint
18	ICSM: RESTORE → EXECUTE; inference resumes from next valid tile boundary under V2
19	Remaining tiles complete; ICSM → COMPLETE
20	JBIA finalizes **inference result** (e.g., anomaly score) and derives canonical **result digest**
21	Proof Assembly Path binds result digest to hardware identity, model identity, trust state, checkpoint/restore status, and Energy Contract context
22	TrustCore signs proof context; ICSM → PROVE
23	**Inference result** and **attestation bundle** stored in Offline Result Queue; ICSM → EXPORT → IDLE
24	Connectivity returns
25	Nelix Compute Platform verifies attestation bundle
26	Only after successful verification is the inference result admitted into trusted operational workflow

Figure 3 — JBIA End-to-End Execution Flow (Grid Anomaly Detection)

Caption: Nominal path with power-loss interruption, checkpoint commit under Energy Contract V1, restore with execution-compatible Energy Contract V2 (RECLOCK only — no model-tier transition), and Proof-Carrying Inference export. Fleet verification deferred until connectivity returns.

6.9 Representative Design Targets

Design targets, not measured silicon.

Parameter	Target
Precision	INT8 primary; INT4 where quantization permits
Power envelope	Sub-15 W (inference accelerator class)
Process class	28 nm (evaluation)
Host	RISC-V control core
On-die memory	Several hundred KB – low MB SRAM
Posture	Offline-first, checkpoint-capable
Ambient	Up to +85 °C industrial class

7. Proof-Carrying Inference

Proof-Carrying Inference (PCI) is an execution architecture in which a completed inference result and cryptographic evidence of its authorized execution context are produced as a **single trusted transaction**. A result that cannot cross the attestation transaction boundary does not enter trusted operational workflows.

This is the central novelty of Attested Infrastructure Inference. Competitors optimize the compute path. Nelix optimizes the **compute-and-proof path** as one architectural transaction:

> *The inference result and its proof are one architectural transaction.*

Nelix does not present PCI as "run inference, then separately sign metadata." Compute and proof are **architecturally coupled** from ATTEST_INPUT through EXPORT.

> *A Proof-Carrying Inference output consists of the inference result and an attestation bundle that cryptographically binds the digest of that result to its authorized execution context.*

7.1 Output Terminology

Three terms are used consistently throughout this document:

Inference result — The actual model output consumed by the operational system. Examples include an anomaly score, classification label, confidence vector, regression output, approved tensor output, or infrastructure event classification.

Result digest — A cryptographic hash or authenticated digest derived from the **canonical representation** of the inference result. The digest is included in the attestation context and cryptographically bound by TrustCore. The digest is not a substitute for the inference result.

Attestation bundle — Cryptographic evidence binding the result digest to the authorized execution context under which inference completed.

A **Proof-Carrying Inference output** is the architectural unit that crosses the trusted export boundary:

```
ProofCarryingInferenceOutput {
  inference_result
  attestation_bundle {
    inference_transaction_id
    result_digest
    device_identity
    boot_measurement
    firmware_identity
    model_identity
    approved_model_tier
    runtime_trust_state
    energy_contract_id
    checkpoint_restore_status
    completion_status
    event_counter_or_timestamp
    trustcore_signature
  }
}
```

This is a **conceptual architectural representation**, not a finalized wire protocol or API schema. Downstream operational systems consume the **inference result**. The Nelix Compute Platform or an authorized verifier validates the **attestation bundle** before admitting that result into trusted revenue, safety, or control workflows.

7.2 Architectural Invariants

Invariant	Statement
INVARIANT 1	An incomplete inference cannot produce a trusted result.
INVARIANT 2	An inference executed under an unauthorized model cannot cross the trusted export boundary.
INVARIANT 3	A device in `SuspectTamper` or `Quarantined` state cannot produce proof accepted by trusted fleet policy.
INVARIANT 4	A restored inference must validate checkpoint integrity and Energy Contract compatibility before execution resumes.
INVARIANT 5	Every trusted exported result must be cryptographically bound to its execution context via its result digest.
INVARIANT 6	A model tier cannot be substituted during active inference unless execution-state compatibility has been explicitly declared, authenticated, and validated.

Corollaries:

- **No completion → no trusted result.**
- **No valid trust state → no proof.**
- **No proof → no trusted export.**

7.3 Attestation Bundle Contents

The attestation bundle assembled at ICSM state **PROVE** conceptually includes:

Evidence element	Source
Inference transaction identifier	ICSM
Result digest	Canonical hash of inference result at COMPLETE
Device identity	PUF-derived, hardware-bound
Boot measurement	Measured boot chain hash
Firmware identity	Signed image identifier
Model identity / hash	Authorized model artifact
Approved model tier	Energy Contract binding
Runtime trust state	TrustCore continuous attestation
Energy Contract identifier / version	Scheduler / contract revision
Checkpoint / restore status	ICSM; sequence number if restored
Completion status	ICSM terminal state
Event counter or timestamp	Platform event ordering
TrustCore signature	Hardware-isolated signing

7.4 Attestation Transaction Boundary

The **Attestation Transaction Boundary** defines the architectural scope within which a result digest and its proof context are cryptographically coupled.

```
ATTESTED INPUT STAGING
  → AUTHORIZED MODEL BINDING
  → ENERGY CONTRACT ACCEPTANCE
  → EXECUTION
  → [OPTIONAL CHECKPOINT / RESTORE]
  → COMPLETION
  → PROOF GENERATION
  → TRUSTED EXPORT
```

At **COMPLETE**, JBIA finalizes the inference result and derives its canonical result digest. At **PROVE**, the Proof Assembly Path binds the result digest to the transaction's authorized execution context. TrustCore signs the proof context. At **EXPORT**, the inference result and its attestation bundle leave the trusted boundary as a Proof-Carrying Inference output.

TrustCore does not merely attest the device that produced the result. It **binds the result digest to the execution context** under which inference completed — including model tier, Energy Contract version, checkpoint history, and trust state at completion.

Events **outside** this boundary (e.g., post-export fleet analytics, SCADA integration) may consume the Proof-Carrying Inference output, but cannot retroactively grant trust to an inference that failed to cross the boundary.

Figure 4 — Attestation Transaction Boundary

Caption: At COMPLETE, inference result and result digest are finalized. At PROVE, digest is bound to execution context and signed. At EXPORT, inference result and attestation bundle cross the boundary together. Fleet verification occurs at EXPORT or upon opportunistic sync.

7.5 Operational Policy

The Nelix Compute Platform verifies attestation bundles before accepting inference results into operational workflows. Default policy:

No attestation → no trusted inference.

A device in `SuspectTamper` or `Quarantined` state may run local inference for diagnostics, but PROVE and EXPORT into trusted paths are blocked per INVARIANT 3.

7.6 Contrast with Industry Practice

Approach	Inference output	Trust model
Cloud GPU	Model logits / tokens	Network perimeter + API auth
Edge NPU (typical)	Model logits	Secure boot at deploy time
Nelix AII (PCI)	Inference result + attestation bundle binding result digest to execution context	ICSM-managed completion transaction; Trust Continuum; fleet verification

Figure 5 — Proof-Carrying Inference Loop

Caption: ICSM reaches COMPLETE; inference result and result digest finalized; Proof Assembly Path binds digest to context; TrustCore signs; Nelix Compute Platform verifies attestation bundle before trusted workflow admission of inference result.

8. TrustCore: The Trust Continuum

TrustCore™ is the hardware root of trust integrated into every Nelix product — not a discrete secure element beside the accelerator.

8.1 Continuous Trust

Conventional secure boot verifies firmware at power-on, then largely stops. TrustCore maintains **verifiable trust state across the entire device lifecycle**:

```
Manufactured → Provisioned → Active → UpdatePending → Active
                    ↓                        ↓
                SuspectTamper ←──────────────┘
                    ↓
                Retired (cryptographic death)
```

A device in **SuspectTamper** cannot return to **Active** without authenticated remediation. Decommissioned devices undergo **cryptographic retirement** — keys destroyed, identity revoked — so retired silicon cannot rejoin the inference fleet.

8.2 Identity from Silicon

A **Physically Unclonable Function (PUF)** derives per-die identity from manufacturing variation. No persistent secret at rest. No extractable key in non-volatile memory. Identity is bound to the physical die — making counterfeiting and cloning architecturally detectable.

8.3 Secure Boot Chain

Figure 6 — TrustCore Secure Boot Chain

Caption: Immutable Boot ROM → PUF identity derivation → signed firmware load → runtime attestation loop. Each stage verifies the next. Trust is never assumed — only proven.

TrustCore aligns with NIST SP 800-193 (firmware resiliency), SP 800-57 (key management), and supports post-quantum migration (ML-KEM, ML-DSA, SLH-DSA) for long-lived infrastructure fleets.

9. SecureGrid: The Inference Integrity Chain

SecureGrid™ extends AII into metering, grid, and industrial infrastructure — where measurement integrity and inference integrity converge.

9.1 The Inference Integrity Chain

Parallel to SecureGrid's **Metrology Integrity Chain** (analog sample → signed measurement record), the **Inference Integrity Chain** binds:

```
Sensor / field input → attested staging → local INT8/INT4 inference → attested result export
```

All stages inside the TrustCore trust domain. No revenue-critical or safety-adjacent data path crosses an unprotected bus.

9.2 Converged Edge

Modern infrastructure nodes increasingly require both trusted measurement and trusted inference:

- A meter that detects tamper **and** classifies consumption anomalies locally
- A pipeline sensor that measures flow **and** detects structural anomalies on-device
- A grid controller that encrypts telemetry **and** runs vision analytics without cloud round-trips

SecureGrid is architected for this convergence — one silicon platform, one trust model, one fleet lifecycle.

9.3 Representative Design Targets

Parameter	Target
Process	55 nm mixed-signal (evaluation)
CPU	RISC-V + isolated security core
Standby	Single-digit µA always-on domain
Tamper response	< ~1 ms detection-to-response
Lifetime	10–15 years; −40 to +85 °C

10. The Nelix Compute Platform and Attestation Fabric

Trusted silicon without trusted fleet operations is an incomplete system. The **Nelix Compute Platform** implements **Proof-Carrying Fleet Operations** — lifecycle management where every fleet action is cryptographically grounded.

10.1 End-to-End Lifecycle

Figure 7 — Attested Inference Lifecycle

Caption: Model registry → INT8/INT4 quantization → signed artifact → regional gateway cache → attested on-device execution → verified result export → fleet event stream.

Phase	Capability
Model onboarding	ONNX export → quantization → signed artifact
Distribution	Regional gateway cache; offline queue when uplink absent
Provisioning	Per-device binding to PUF identity
Runtime	ICSM-managed inference; attestation loop; inference event stream
Update	Signed OTA; dual-bank images; rollback to last trusted model
Recovery	Checkpoint restore via RESTORE_VALIDATE; quarantine-on-failure
Retirement	Cryptographic device death

10.2 Attestation Fabric (Future Design Objective)

Beyond point-to-cloud attestation, Nelix is architecting an **Attestation Fabric**: devices in mesh or feeder deployments exchange neighbor attestation summaries — detecting localized tamper clusters (e.g., coordinated meter bypass) without continuous backhaul.

This remains a future design objective. Current platform architecture preserves identity and evidence formats required to enable it.

Part III — Evaluation

1. The Deployment Reality Index

Nelix provides the **Deployment Reality Index (DRI)** — a qualitative framework for evaluating whether a workload belongs in Attested Infrastructure Inference versus conventional edge AI.

Score each dimension **Low / Medium / High** stress:

Dimension	High stress means…
Power	Intermittent supply; joule budget is binding constraint

Connectivity	Sparse, high-latency, or absent for extended periods
Temperature	Sustained high ambient; no climate control
Physical security	Exposed, unattended, adversary-accessible
Lifecycle	10–20 year deployment; costly replacement
Trust / regulatory	Inference outputs affect revenue, safety, or compliance

Example — revenue-grade smart meter with on-device anomaly detection:
Power: High · Connectivity: Medium · Temperature: High · Physical security: High · Lifecycle: High · Trust: High → **AII native**
Example — office environmental sensor:
Mostly Low/Medium → conventional IoT may suffice
DRI gives partners and evaluators shared vocabulary for why a generic edge NPU is structurally mismatched — not merely underspecified.

12. Threat Model and Security Architecture

Nelix assumes a **capable adversary with physical access** over an extended period.

12.1 Protected Assets

- Device identity and cryptographic keys
- Firmware and model integrity
- Measurement and inference accuracy
- Telemetry and attestation trustworthiness

12.2 Attack Classes

Attack class	Threat	Nelix defense
Integrity	Firmware/model replay, rollback	Secure boot, monotonic counters, signed OTA
Identity	Cloning, counterfeits	PUF-rooted identity; attested provisioning
Physical	Probing, bypass, tamper	Domain isolation; inference inside trust boundary
Observation	Bus sniffing, debug abuse	On-die integration; debug gated by lifecycle
Side-channel	SPA/DPA, fault injection	Leakage-aware crypto; supply/clock monitoring
Operational	Fleet-scale misconfiguration	Policy engine; quarantine-on-failure

Layered mitigations plus the Trust Continuum ensure compromised devices cannot silently return to trusted service.

13. Representative Workloads and Use Cases

13.1 Utilities and Revenue-Grade Metering

SecureGrid encrypted measurement and tamper detection; InferEdge anomaly classification; proof-carrying telemetry for revenue protection and grid visibility.

13.2 Industrial and SCADA

Attested inference results surface into existing control environments as trustworthy signals — hardware-rooted anomaly detection without replacing the control layer.

13.3 Distributed Infrastructure and Telecom Edge

Offline-first inference under joule budgets; signed model cache at regional gateways; sub-15 W always-on intelligence at remote sites.

13.4 Agriculture, Health, and Public Infrastructure

Trusted local inference for sensor intelligence and asset monitoring — validated for conditions common across emerging and industrial markets worldwide.

13.5 Model Classes (Representative)

Workload class	Typical precision	InferEdge fit
Embedded vision	INT8	Primary
Anomaly detection	INT8 / INT4	Primary
Sensor classification	INT8 / INT4	Primary
Time-series forecasting	INT8	Primary
Large language models	—	Out of scope (datacenter economy)

Nelix deliberately does not compete in the datacenter LLM decode market. AII targets quantized infrastructure model classes where local, attested completion matters.

Part IV — Path Forward

14. Integration, Software Stack, and Developer Experience

Nelix integrates into existing operational stacks.

14.1 Model Pipeline

ONNX in → quantized INT8/INT4 out. Deployment, versioning, and rollout hook into model registry workflows teams already use.

14.2 Device Software Stack

- Hardware-rooted secure boot
- Deterministic RTOS
- SDK (security via TrustCore gateway APIs only)
- OTA agent and InferEdge runtime
- Cloud integration that **enhances** but does not **gate** local operation

14.3 Fleet APIs

Attestation verification endpoints, inference event streams, and policy configuration — enabling SCADA, cloud control planes, and MLOps tooling to consume **Proof-Carrying Inference outputs** as first-class signals.

15. Roadmap and Evaluation Path

Nelix is pre-silicon, with architecture validated on FPGA ahead of tape-out.

Milestone	Status
TrustCore architecture specification	In progress
InferEdge JBIA architecture specification (ICSM, Energy Contract, RETIER semantics, checkpoint model)	In progress
InferEdge JBIA RTL	In progress
FPGA validation platform	Active
Design partner pilots	Engaged
SecureGrid first tape-out	Targeted ~2029
InferEdge production silicon	Following SecureGrid platform validation

Nelix offers technical evaluation through FPGA bring-up, architecture briefings, and design-partner programs.

Process-node references and milestone dates are targets subject to verification gates

16. Conclusion

The inference hardware industry is rewriting the rules for datacenter AI — breaking memory walls, eliminating kernel overhead, hardwiring transformers, delivering deterministic tensor streams. These are genuine architectural advances for the token economy.

But a second economy is emerging — one where inference must **complete under joule budgets**, **operate without the cloud**, and **prove itself cryptographically** to systems that cannot afford to trust unverified outputs.

Nelix defines this economy as **Attested Infrastructure Inference** and builds the architecture for it:

- **Joule-Bounded Inference Architecture (JBIA)** — ICSM-managed completion transactions under Energy Contract, with execution-compatible restore and failure-aware semantics
- **Proof-Carrying Inference** — inference result and attestation bundle as one trusted transaction; six architectural invariants
- **Trust Continuum** — continuous hardware trust from manufacture to cryptographic retirement
- **Proof-Carrying Fleet Operations** — attestation bundle verification gates trusted workflow admission

Nelix defines inference as a managed completion transaction operating under an Energy Contract, preserving transactionally consistent execution state across infrastructure interruption, and producing the inference result and cryptographic proof of its authorized execution context as one trusted architectural transaction.

The architectural thesis is integration, not isolated invention:

> *Nelix makes inference completion and cryptographic proof part of the architecture transaction itself.*

The question for infrastructure operators is no longer whether AI will run at the edge. It is whether that edge AI can be **trusted** — cryptographically, operationally, and over decades of field life.

Nelix is building the answer in silicon.

About Nelix Chip Design Ltd.

Nelix Chip Design Ltd. is a trusted AI inference hardware company headquartered in Accra, Ghana. The company designs secure InferEdge accelerators, TrustCore-secured edge processors, and deployment infrastructure for power-constrained, connectivity-limited, and mission-critical systems worldwide.
Web: nelix.io
Technology: nelix.io/technology
Platform: nelix.io/overview
Contact: nelix.io/contact