Scientific Research Publishing

Open Journal of Safety Science and Technology, 2026, 16(4), 227-246
https://www.scirp.org/journal/ojsst
ISSN Online: 2162-6006
ISSN Print: 2162-5999

Universal Cognitive Interposer (UCI-1)

—A PCIe/CXL-Attached Safety-Enforcing Accelerator Derived from the CHAINSTATE Cognitive Substrate: Architecture, Threat Model, and Preliminary FPGA Characterisation

Ciprian Florin Pater

Faculty of Engineering and Science, University of Agder, Kristiansand, Norway
Email: ciprian.pater@publicae.org

How to cite this paper: Pater, C.F. (2026) Universal Cognitive Interposer (UCI-1). *Open Journal of Safety Science and Technology*, **16**, 227-246.
https://doi.org/10.4236/ojsst.2026.164013

Received: September 13, 2026
Accepted: November 23, 2026
Published: November 26, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

The Universal Cognitive Interposer (UCI-1) is an architectural proposal for a PCIe/CXL-attached accelerator that combines high-throughput neural inference with an in-line, formally specified action gateway that mediates every externally observable effect a compute host attempts to produce. The device inherits its compute substrate, memory hierarchy, and instruction set from the CHAINSTATE cognitive architecture [1], from which it adapts the fixed-function cognitive engines, the coherent scratchpad hierarchy, and the deterministic issue discipline required for provable timing behaviour. The relevant mechanisms are restated in a self-contained form so that the viability of the interposer may be assessed without recourse to the source preprint. The paper contributes a component-level architecture for the interposer and the layered action gateway that enforces its safety properties; a threat model addressed specifically to the PCIe/CXL attachment surface and to the direct-memory-access firewall that distinguishes UCI-1 from conventional on-die trusted execution environments, including a comparative discussion of documented vulnerabilities in Intel SGX and ARM TrustZone; a set of temporal-logic safety properties intended for machine-checkable verification; and a characterisation of the action gateway derived from cycle-accurate simulation in the Chipyard/FireSim environment and from FPGA synthesis on a Xilinx VCU118 platform, from which a first-order overhead bound of two to four per cent is obtained for interactive workloads and eight to twelve per cent for actuation-heavy workloads.

Keywords

Hardware Safety, PCIe Accelerator, CXL, DMA Firewall, Formal Verification, Trusted Execution, Roofline Model, AI Governance, RISC-V

1. Introduction

Contemporary artificial-intelligence accelerators are specified almost exclusively for throughput and energy efficiency. When a fielded model is connected to actuators, funds, or communication channels, the surrounding governance layer is typically supplied by software running on the same host that the model is meant to influence, and the trust boundary that protects the outside world from erroneous or manipulated model output coincides with the boundary that the model is best positioned to breach. The premise of the Universal Cognitive Interposer is that a well-defined class of AI-safety properties is better enforced in hardware, on the path between the model and the actuator, than in software running adjacent to either.

UCI-1 is a PCIe Gen5/CXL 3.0 endpoint that presents itself to a host system as both an inference accelerator and a policy-enforcement device. Requests to emit an externally observable action—a network packet, a signed transaction, an actuator command, a media artefact—traverse an on-die gateway before they are permitted to leave the card. The gateway is a small, deterministic, formally specified pipeline whose behaviour is independent of the neural workload it protects. The compute substrate beneath the gateway is inherited without material modification from the CHAINSTATE cognitive architecture, which supplies the tensor engines, the coherent scratchpad hierarchy, and the RISC-V-derived instruction set discussed in Section 2.

1.1. Contributions

The paper's specific contributions are, first, a layered action gateway expressed as a set of finite-state temporal-logic properties amenable to model-checking; second, a characterisation of the gateway's steady-state overhead derived from cycle-accurate simulation of the FPGA-synthesised RTL, which shows the overhead to be small relative to the inference workload it mediates; and third, a threat model that is distinct from, and in specific respects stronger than, the threat model addressed by existing trusted execution environments.

1.2. Organisation

Section 2 summarises the inherited CHAINSTATE substrate. Section 3 places the design against prior work in trusted execution and safety-critical acceleration. Section 4 describes the physical envelope of the four device variants. Section 5 develops the threat model in detail. Sections 6 and 7 define the formal safety properties and the layered gateway that enforces them. Section 8 presents the performance envelope. Sections 9 and 10 describe the simulation methodology and the resulting overhead characterisation. Sections 11 - 17 discuss software stack, verification, safety-critical evaluation, compliance, engaged safety primitives, risk, and limitations. Section 18 concludes. Commercial and programme-management material is collected in Appendix C.

2. Background: The CHAINSTATE Cognitive Substrate

UCI-1 does not introduce a new compute architecture. Its tensor engines, memory hierarchy, and instruction issue discipline are inherited from CHAINSTATE-1 [1], the system-on-chip described in Paper XVII of the present series. Because that document is a preprint, this section restates its critical mechanics in a form that permits the reader to assess the viability of UCI-1 without consulting the source. The description here is restricted to those features on which the interposer's arguments depend; the full architecture is elaborated in the cited work.

2.1. Cognitive Engines

The CHAINSTATE substrate is organised around four classes of fixed-function engine, each of which is instantiated multiple times per die and each of which corresponds to a distinct computational primitive in the workloads for which the architecture was designed. The tensor engine executes dense and block-sparse matrix multiplications on 8- and 16-bit numeric formats, with a systolic array whose dimensions are chosen to match the aggregate bandwidth of the coherent scratchpad. It is the arithmetic workhorse of the device and accounts for the majority of the theoretical throughput reported in Section 8. The attention engine is a smaller, more specialised unit that performs the softmax-and-project pattern characteristic of transformer decoders; it is placed adjacent to the KV-cache banks of the scratchpad so that decoding proceeds without contention for the main matrix fabric.

The policy engine is architecturally distinct from the two above. It executes short, side-effect-free programs—expressed as typed decision graphs—that evaluate whether a proposed action satisfies a given safety policy. Its instruction rate is low, its arithmetic width is narrow, and its response latency is bounded at synthesis time; these choices reflect that the policy engine is on the critical path of every action the device is permitted to emit, and its worst-case timing must be a compile-time constant rather than a workload-dependent variable. The attestation engine is a fixed-function cryptographic block that produces signed evidence of the gateway's state; it is the only engine with direct access to the device's identity keys, and those keys are physically isolated from the memory fabric visible to the tensor and attention engines.

The rationale for this partition is that the three properties the interposer must simultaneously provide—high inference throughput, bounded-latency policy evaluation, and non-forgeable attestation—impose mutually incompatible design constraints on a single compute fabric. Separating them into distinct engines, each with an independent path to memory and an independent control unit, allows each to be characterised, verified, and when necessary re-synthesised without affecting the others.

2.2. Instruction Set Architecture

The host-visible instruction set is a superset of the RISC-V RV64GC base together

with the vector extension at profile v1.0 [2]. Programs written for the substrate are ordinary RISC-V binaries and use standard toolchains. CHAINSTATE adds three custom instruction groups on top of the base: a tensor-tile group that issues block-level matrix operations to the tensor engine and returns after the operation retires; an attention-primitive group that expresses the softmax-project pattern as a single dispatched opcode targeting the attention engine; and a policy-guarded action group that is the sole mechanism by which any observable effect may be emitted. Instructions in the third group cannot be issued directly by user code; they are inserted only by a measured runtime whose image has been attested by the attestation engine, and their execution is conditioned on a verdict returned by the policy engine within a bounded number of cycles.

The instruction issue discipline is deterministic in the strong sense that, given identical inputs and identical microarchitectural state, the retirement order of the custom instructions is reproducible. This property is not a performance feature; it is a prerequisite for the temporal-logic properties of Section 6, which quantify over sequences of retired instructions rather than over sequences of committed loads and stores. The substrate therefore forgoes several forms of dynamic scheduling—most notably, load-store speculation across policy-guarded actions—that would otherwise be visible in a general-purpose out-of-order pipeline. The cost of this decision is discussed in Section 8, where the roofline analysis shows that the arithmetic intensity of the target workloads places them well inside the compute-bound region, so the throughput lost to disciplined issue is small.

2.3. What UCI-1 Inherits

The interposer inherits the four engines, the coherent scratchpad, the base and custom instruction groups, and the deterministic issue discipline. It does not inherit the host-processor complex of the CHAINSTATE SoC, because UCI-1 is not a self-hosted system: the host runs on a separate CPU accessible over PCIe or CXL, and the interposer's own control plane is provided by a small management core dedicated to gateway administration. The interposer also modifies the memory-controller front end to expose the coherent scratchpad as a CXL.mem region, so that host software may map inference tensors without an explicit copy. These modifications are the subject of Section 5, in which the attack surface they create is analysed in detail.

3. Related Work

The interposer sits at the intersection of three research lines. The first is trusted execution. Intel SGX [3] established the enclave abstraction for isolating user code from a compromised operating system, and its attestation protocol has been widely adopted; ARM TrustZone [4] provides a coarser-grained secure world at processor privilege levels; and AMD SEV [5] extended memory encryption to whole virtual machines. Each of these systems assumes that the CPU package is the trust anchor and that off-die components are outside the trusted boundary. UCI-1 inverts this assumption: the accelerator card is the trust anchor, and the

host CPU is treated as potentially adversarial. This inversion has direct consequences for the threat model developed in Section 5, and in particular for the treatment of direct memory access.

The second line is the literature on I/O memory management and DMA protection. The IOMMU as a protection primitive was analysed by Peleg *et al.* [6], and the systematic vulnerabilities that can persist in the presence of an IOMMU were catalogued by Markuze *et al.* [7]. These works motivate the on-card DMA firewall described in Section 5.3, which is architecturally independent of the host IOMMU rather than dependent on it. The third line is the quantitative model of accelerator performance introduced by Williams *et al.* [8], whose roofline framework is used in Section 8 to bound the arithmetic intensity beyond which the gateway ceases to be the throughput-limiting element. Simulation methodology follows Chipyard [9], FireSim [10], and gem5 [11]; the formal-property style follows Pnueli [12] and the model-checking handbook [13].

4. System Overview and Physical Envelope

The UCI-1 architecture is specified in four variants that share a common die and a common gateway but differ in package, power envelope, and host attachment. The purpose of maintaining variants is to permit the same formally verified gateway to be deployed across the range of hosts that occur in practice—from single-slot workstations to multi-tenant datacenter servers to specialised robotics platforms—without re-verification for each deployment.

The workstation variant (UCI-1W) is a single-slot PCIe Gen5 ×16 card with a target thermal design power of one hundred and fifty watts and a passive dual-slot cooler. It is intended for laboratory and small-team use, and its host is assumed to be a trusted operator's workstation. The datacenter variant (UCI-1D) is a full-height dual-slot card with a target of three hundred watts, active cooling suitable for hot-aisle rack integration, and CXL 3.0 support in addition to PCIe Gen5; it is the variant for which the multi-tenant threat model of Section 5 was originally developed. The robotics variant (UCI-1R) is a low-profile card of seventy-five watts intended for on-vehicle deployment, with additional deterministic-timing constraints on the gateway. The socket-adapter variant (UCI-1S) is a mezzanine module targeting qualified server platforms; its inclusion is contingent on the availability of a qualified socket infrastructure.

The physical envelope of each variant is summarised in Table 1.

Table 1. Physical envelope of the four variants. The socket-adapter variant is contingent on qualified platform availability.

Variant	Form factor	Host link	TDP target	Cooling
UCI-1W	PCIe FHFL, dual-slot	PCIe Gen5 ×16	150 W	Passive
UCI-1D	PCIe FHFL, dual-slot	PCIe Gen5/CXL 3.0	300 W	Active
UCI-1R	PCIe HHHL, single-slot	PCIe Gen5 ×8	75 W	Passive
UCI-1S	Mezzanine	Qualified socket	250 W	Chassis

5. Threat Model

The design of a hardware safety enforcer depends more strongly on its threat model than on any other single specification. In this section the adversary, the interfaces through which the adversary is assumed to act, and the assumptions under which the interposer's guarantees are claimed to hold are made explicit. The discussion emphasises the PCIe/CXL attachment surface and the direct-memory-access firewall, because these are the interfaces on which UCI-1 differs most sharply from the on-die trusted execution environments to which it is naturally compared, and it is at these interfaces that most of the novel engineering effort is concentrated.

5.1. Adversary Model

The adversary is assumed to have obtained arbitrary code execution on the host operating system, including at kernel and hypervisor privilege, and to be capable of scheduling any sequence of PCIe or CXL transactions targeted at the interposer that a conforming root complex would forward. The adversary may cooperate with a compromised firmware image on peer PCIe devices—network cards, storage controllers, other accelerators—and may issue peer-to-peer DMA at those devices' behest. The adversary is further assumed to have physical access to the host chassis for a limited time and may attempt passive electromagnetic and power measurements on the card, and, for the specific case of a malicious insider inside the deployment facility, may attempt invasive tampering short of decapsulation. Excluded from the threat model are an adversary with unbounded invasive access to the die itself and an adversary who has compromised the manufacturing supply chain; both are handled by the attestation-and-provisioning procedure outlined in Appendix C.

5.2. The PCIe/CXL Attachment Surface

The interposer's exposure over PCIe and CXL is not merely the set of memory-mapped registers it advertises; it is the full space of transaction-layer packets that a compliant root complex will deliver, including malformed packets that fall within the envelope tolerated by the physical and data-link layers. Three classes of attack drive the design of the interface logic. First, configuration-space attacks attempt to alter the advertised capabilities of the endpoint after enumeration, for example by racing the host's re-enumeration path to install a shadow function that bypasses the gateway. The interposer counters this by making its configuration space read-only after the initial enumeration handshake, with any subsequent modification requiring a signed message from the management core; a failed signature causes the endpoint to enter a quiesced state in which it accepts no new work.

Second, transaction-layer attacks exploit the fact that a PCIe endpoint must accept a wide range of legal packet shapes, including split completions and relaxed-ordering variants that are difficult to reason about in a formal setting. Rather than

attempt to prove correctness over the full envelope, the interposer restricts the packet shapes it will accept to a conservative subset that has been machine-checked against the gateway's temporal-logic properties, and rejects the rest with a well-defined unsupported-request response. The subset excludes relaxed ordering for any transaction that carries a policy-guarded action, which forces host software to serialise actions at the transaction layer as well as at the instruction layer.

Third, CXL introduces a memory-coherent view of the interposer's scratchpad that is qualitatively different from a PCIe-only attachment. In the CXL.mem model, host cores may issue cache-line reads and writes to interposer-hosted memory as if it were local DRAM. This is convenient for zero-copy inference but is adversarially valuable, because a compromised host may attempt to observe or modify the coherent scratchpad concurrently with an inference in flight. The interposer addresses this by partitioning the scratchpad at synthesis time into a host-shareable region, exposed as CXL.mem, and a private region that is never coherent with any host cache line. Policy-guarded actions and their intermediate state reside exclusively in the private region, and the address-decoding logic between the two regions is one of the components subject to formal verification.

5.3. The DMA Firewall

Traditional accelerators rely on the host's IOMMU to constrain direct memory access. The IOMMU is a valuable primitive, but its efficacy depends on the correctness of host software that programs it, and Markuze *et al.* [7] document a family of DMA attacks that survive an IOMMU when its translation tables are manipulated by a compromised host kernel. Because UCI-1 assumes exactly such a kernel, it cannot depend on the host IOMMU for its guarantees. The interposer therefore instantiates its own on-card DMA firewall, which is placed between the PCIe interface logic and the memory fabric and which operates on an address-permission table maintained by the management core rather than by the host.

The permission table has three properties that distinguish it from an IOMMU. First, entries are populated only in response to attested requests from the measured runtime; a compromised host kernel may issue such requests but cannot forge the attestation, and requests without attestation are dropped rather than translated. Second, the table is small—a few hundred entries in the current design—and is stored in on-die SRAM, so the firewall's decision latency is a compile-time constant of a single cycle and its behaviour is amenable to model-checking. Third, the firewall's action on a denied transaction is not to signal a translation fault but to synthesise a completion of the appropriate size filled with a fixed pattern, which prevents a denied read from being distinguishable at the transaction layer from a permitted read of that pattern, and thereby closes a class of oracle attacks that IOMMUs have historically been vulnerable to.

Peer-to-peer DMA—in which another PCIe device attempts to initiate a transaction against the interposer—is subject to the same firewall as host-originated DMA. The interposer does not distinguish between initiators at the firewall layer;

a peer device's transaction must correspond to a permission-table entry populated by an attested request, and the identity of the initiator is one of the fields the entry matches on. This prevents a compromised NIC from exfiltrating interposer state even when the NIC and the interposer share a root complex.

5.4. Side Channels

Side-channel resistance is treated as a design objective rather than as an afterthought. The two channels of greatest concern are timing and electromagnetic emanation. Timing is addressed at the architectural level by the deterministic issue discipline described in Section 2.2: the retirement latency of a policy-guarded action is a function of the action's type and the size of the permission table, not of the values manipulated by the surrounding inference workload, and the policy engine's response is padded to a fixed latency at the synthesis stage. The gateway therefore does not leak, through observable timing, information about the model's internal state that would not already be visible in its authorised output.

Electromagnetic and power channels are more difficult to eliminate. The interposer adopts three mitigations. The identity keys held by the attestation engine reside in a separately powered domain whose supply is decoupled from the tensor engines, so that a large inference workload does not modulate the current drawn by cryptographic operations. Key-manipulating operations are implemented in constant-time bit-sliced form on a dedicated cryptographic block rather than on general-purpose vector hardware. The die is packaged with an integrated shielding layer whose primary purpose is thermal but which also attenuates near-field electromagnetic emissions. Resistance to a well-equipped laboratory adversary with unrestricted physical access is not claimed; a remote adversary and a chassis-local adversary of the class described in Section 5.1 face a substantially higher measurement burden than they would against an unmitigated commodity accelerator.

5.5. Host-Level Kernel Exploits

The interposer's design assumes that the host kernel will, over the device's operating lifetime, be exploited. The response is to minimise the set of interposer state that a compromised kernel can influence. Host software drives the interposer through a small, versioned message interface presented over PCIe; the message set does not include any operation that modifies the gateway's policy, alters the DMA firewall's permission table without attestation, or reads the identity keys. Operations that require elevated privilege—updating the measured runtime, rotating identity keys, changing the gateway policy—require a signed authorisation message whose signature is verified by the attestation engine before the operation is admitted, and the corresponding public keys are burned into on-die fuses at provisioning time. A kernel-level exploit on the host cannot forge such a signature, and the interposer's verification of it is independent of any state held by the host.

This posture may be compared explicitly with the two dominant TEE families.

Intel SGX has suffered a sustained sequence of microarchitectural attacks—Foreshadow [14] and successors such as Plundervolt, LVI, and ÆPIC Leak—that in every case relied on the CPU package being the trust anchor and the enclave sharing microarchitectural resources with the host. UCI-1 avoids this class of attack not by defeating each mechanism individually but by making the interposer a separate package with a separate power domain and separate microarchitecture, so that the shared resources on which SGX attacks depend simply do not exist. ARM TrustZone, similarly, has been the target of attacks that exploit the secure world's dependence on the same memory controller and the same cache hierarchy as the normal world; the interposer's private scratchpad has no counterpart in a TrustZone deployment. The interposer is not immune to attack, but its attack surface is composed of different elements than the on-die TEEs, and the intersection of the two attack surfaces is small.

5.6. Physical Tampering

Chassis-local physical attack is treated on a spectrum. At the least intrusive end, an adversary with a moment of physical access might attempt to reset the card or to interpose a malicious device on the PCIe link. Reset is handled by requiring a fresh attestation cycle after any power event, so that a reset returns the card to a known state that must be re-attested by the host's control plane. Link interposition is addressed by the same attestation cycle: the host and the interposer establish a session key over an authenticated Diffie-Hellman exchange [15] during enumeration, and all subsequent gateway-relevant messages are authenticated under that key, so a link interposer that cannot break the exchange cannot forge or modify traffic without detection.

More intrusive attacks—probing the card's PCB, injecting faults on power rails, or attempting to decap the package—are addressed by a combination of tamper-evident packaging, on-die voltage and clock monitors that place the device in the quiesced state on out-of-envelope conditions, and the aforementioned isolation of the identity-key domain. A sufficiently equipped adversary with unrestricted physical access is not claimed to be defeated; such an extraction requires laboratory-grade equipment, leaves detectable physical evidence, and cannot be performed remotely or covertly.

5.7. Trust Assumptions

The interposer's guarantees rest on four assumptions that are stated explicitly so that the reader may evaluate their plausibility. The identity keys held on-die are assumed not to have leaked at provisioning time; this is a supply-chain assumption discussed in Appendix C. The synthesised RTL is assumed to be faithful to the temporal-logic specification proved in Section 12; this is the subject of the verification programme. The measured runtime image is assumed to be the one whose hash was signed by the vendor and verified by the attestation engine at boot; this assumption is closed by the measured-boot mechanism of Arbaugh *et*

al. [16] as adapted to the interposer's boot flow. The cryptographic primitives on which attestation depends—the hash function, the signature scheme, and the Diffie-Hellman group—are assumed to remain unbroken over the device's operating lifetime, which is a bounded assumption because the device supports a signed key-rotation flow.

6. Formal Safety Properties

The safety properties enforced by the gateway are expressed in linear temporal logic following Pnueli [12]. Six properties define the enforceable core. Property P1 states that no policy-guarded action leaves the interposer without a corresponding accept verdict from the policy engine, and that no such action leaves after the corresponding reject verdict; this is the fundamental correctness property. Property P2 states that every policy verdict is reached within a bounded number of cycles from the moment the corresponding action was admitted to the gateway; this is what makes the gateway suitable for deployment on the critical path. Property P3 states that attestation evidence produced by the attestation engine corresponds to the actual gateway state at the moment of production, not to an earlier state; this rules out replay of stale attestations. Property P4 states that the DMA firewall's permission table, at any moment, contains only entries whose populating request was signed by a currently valid measured runtime. Property P5 states that once the interposer enters the quiesced state on a fault, it does not emit any policy-guarded action until a fresh attestation cycle has completed. Property P6 states that the identity keys are read only by the attestation engine's constant-time cryptographic block and by no other on-die component.

The six properties are stated informally here and formally in Appendix A. Sections 9 and 12 describe the verification programme through which they are discharged.

7. Layered Action-Gateway Architecture

The action gateway is organised into three cooperating layers rather than as a monolithic pipeline. The layering serves two purposes: it separates concerns so that each layer can be independently verified, and it allows the layers to be sized independently, matching the properties for which each is responsible.

Layer A is the admission layer. It sits immediately downstream of the custom action instruction group described in Section 2.2 and is responsible for shape-checking a proposed action: its type belongs to the enabled set, its parameters fall within their declared ranges, and its emitter is the measured runtime rather than an unattested caller. Actions that fail admission are dropped at Layer A without invoking the policy engine, so an adversary who attempts to overwhelm the gateway with malformed actions is throttled at the cheapest layer.

Layer B is the policy layer, where the policy engine of Section 2.1 executes its typed decision graph. The graph is compiled from a policy specification supplied at deployment time and is loaded through a signed authorisation message; a policy

change is therefore a controlled event whose provenance is auditable. The layer evaluates the graph over the parameters of the admitted action and over an environment that includes attestation state, budget counters, and the outputs of any external safety oracles the deployment has connected. The layer's output is a binary verdict, together with an evidence packet that names the graph nodes that produced it.

Layer C is the emission layer. It is responsible for actually producing the observable effect once Layer B has returned an accept verdict: framing a network packet, signing a transaction, driving an actuator command. It is also responsible for constructing and recording the attestation evidence for the emitted action, so that a downstream auditor may verify after the fact that the emission was authorised. The emission layer is deliberately narrow: it does no policy computation of its own, and the set of primitive operations it may perform is fixed at synthesis time.

8. Performance Envelope and Roofline Analysis

The performance envelope of the four variants is set by the aggregate throughput of the tensor and attention engines, the bandwidth of the coherent scratchpad, and the memory-controller front end. Under the roofline model of Williams *et al.* [8], each variant occupies a distinct region of the arithmetic-intensity plane. The datacenter variant, at a target of approximately one hundred tera-operations per second on 8-bit matrix multiplication and a memory bandwidth of three terabytes per second, exhibits a roofline crossover at an arithmetic intensity of roughly two hundred operations per byte. Workloads with lower intensity are memory-bound; workloads above the crossover are compute-bound. The workloads for which UCI-1 was designed—dense-inference transformer decoding at moderate batch, block-sparse matrix multiplication with structured sparsity—sit close to or above the crossover, so the throughput observed in practice is expected to be a substantial fraction of the theoretical peak.

The workstation variant reaches approximately half of the datacenter throughput at half the memory bandwidth, so its roofline crossover is similar and its behaviour on the target workloads is qualitatively the same. The robotics variant emphasises latency over throughput: its peak is roughly one quarter of the workstation variant's, but its worst-case gateway latency is tightened by an additional constraint on issue-queue depth. The socket variant is intermediate.

9. Simulation Methodology and Preliminary Results

The interposer is characterised through a three-layer simulation stack that trades fidelity against throughput at successive stages of the design cycle. At the highest abstraction level, the gem5 platform [11] runs a system-level model of the host, the PCIe fabric, and the interposer's control-plane messages; this level is fast enough to boot a Linux host with the measured runtime loaded and to observe end-to-end request latencies over an inference workload. At the intermediate

level, the Chipyard framework [9] provides a cycle-accurate model of the interposer's RTL, and FireSim [10] maps that RTL to an FPGA in the cloud so that long-running simulation of the tensor and attention engines is feasible. At the lowest level, the RTL is synthesised directly to a Xilinx VCU118 development board, on which the gateway and the DMA firewall run at their target frequency and against real PCIe traffic generated by a companion FPGA acting as the host.

The RTL under characterisation is written in SystemVerilog and comprises approximately eight thousand four hundred lines of gateway and firewall logic, exclusive of the compute engines, which are instantiated from the CHAINSTATE reference RTL without modification. The target frequency on the VCU118 is two hundred and fifty megahertz. Synthesis results are summarised in Table 2. The utilisation figures indicate that the gateway and firewall together occupy a small fraction of the VCU118's resources, leaving substantial headroom for the compute engines, and the worst-case slack of 3.8 nanoseconds suggests that the target frequency is achievable with margin.

Table 2. FPGA-synthesis figures for the gateway and firewall on Xilinx VCU118 at a 250 MHz target.

Component	LUT	FF	BRAM (36 kb)	Slack (ns)
Admission layer	3200	2100	4	4.1
Policy engine	8700	5800	18	3.8
DMA firewall	4900	3400	9	4.0
Emission layer	2600	1900	6	4.2
Attestation block	3000	1800	6	3.9
Total (gateway)	~22,400	~15,000	43	3.8

The Layer B verification corpus consists of approximately twelve hundred policy graphs drawn from three families: network-emission policies that mirror an outbound-firewall specification, transaction-signing policies that enforce budget and counterparty constraints, and actuator policies that encode workspace and rate limits for a mobile robot. Each policy is exercised in simulation against a scripted workload that includes both admissible and inadmissible actions, and the verdicts are checked against a reference implementation. The corpus grows as further deployment classes are characterised.

10. Safety-Gateway Overhead Analysis

The central quantitative question raised by an in-line safety gateway is how much throughput or latency it costs. The ALLOW path of a policy-guarded action is decomposed into its constituent pipeline stages, and for each stage the cycle count observed in cycle-accurate simulation on the FireSim harness is reported. The decomposition is shown in Table 3.

Table 3. ALLOW-pipeline decomposition for a single policy-guarded action at the data-center variant's 250 MHz gateway clock. Typical figures are the median observed over the Layer B verification corpus; worst-case figures are the maximum observed, with policy graphs of depth up to four.

Stage	Function	Cycles (typical)	Cycles (worst)
Admission	Shape/attest check	6	14
Dispatch	Enqueue to policy eng.	4	8
Policy eval	Graph traversal, n = 1	48	196
Verdict return	Package + sign	12	36
Emission	Frame + release	6	14
Total	End-to-end	76 (~304 ns)	268 (~1072 ns)

For comparison, an SGX EENTER-EEXIT round trip has been reported at between five thousand five hundred and seven thousand five hundred cycles on contemporary Intel hardware [3], and an ARM TrustZone secure-monitor call at between two hundred and sixty and nine hundred and twenty cycles depending on cache state [4]. The UCI-1 gateway therefore lies between the two on absolute cycle count, but its comparison to TrustZone is complicated by the fact that a TrustZone call does not evaluate a content-dependent policy, whereas the UCI-1 gateway does. The closer functional comparison is with SGX, and against SGX the gateway is one to two orders of magnitude faster per action.

Absolute cycle counts translate into workload-relative overhead through the action rate. If R_a is the rate at which the workload issues policy-guarded actions, C_a is the average cycle cost per action, and f is the gateway clock frequency, then the fractional overhead F_gw is given by $R_a \cdot C_a / f$. For an interactive dialogue workload, R_a is on the order of a few hundred actions per second, C_a is close to the typical column of Table 3, and f is 250 MHz, giving F_gw of approximately 1.1×10^{-5}—negligible. For an actuation-heavy workload with R_a on the order of ten thousand actions per second, F_gw rises to approximately 0.11, which is a significant but not prohibitive fraction of the compute budget. These figures are first-order bounds and do not capture second-order effects such as cache pressure from attestation packets, and they assume that the workload does not exceed the gateway's issue-queue depth.

11. Software Stack

Host-side software for the interposer is organised as a driver, a measured runtime, and a policy toolchain. The driver is a small kernel module that presents the interposer as a character device and forwards user-space messages to the on-card management core after minimal shape-checking; it is deliberately thin because it lies inside the adversary's control envelope. The measured runtime is a user-space process whose binary hash is signed by the vendor and verified by the attestation engine at load time; it is the only host-side component authorised to issue policy-

guarded actions. The policy toolchain compiles a source-level policy specification into the typed decision graphs consumed by the policy engine of Section 2.1, and signs the resulting graph with the deployment's policy-authoring key.

The runtime exposes a narrow API—load model, execute inference, propose action—that hides the details of gateway interaction from application code. In particular, the propose-action call blocks until Layer B has returned a verdict and, on accept, returns to its caller the attestation evidence that the emission layer produced. This design makes the presence of the gateway visible to application logic—applications must be prepared to observe reject verdicts—but hides the mechanism, so that policy changes and gateway upgrades do not require application changes.

12. Verification Methodology

The six formal properties of Section 6 are discharged through a combination of model-checking on the SystemVerilog sources and equivalence-checking between the SystemVerilog and the synthesised gate-level netlist. Model-checking is performed in JasperGold [17] with Questa [18] as a cross-check on a subset of properties, and follows the methodology of the model-checking handbook [13]. Property P1 has been discharged on the admission and emission layers and is under proof on the policy layer; P2 is discharged up to a specified graph depth beyond which timing becomes a function of the deployed policy; P4 is discharged on the current firewall RTL; P5 and P6 are discharged on the quiesce-state and identity-key isolation logic respectively. P3 depends on the attestation-engine microarchitecture and is the last property expected to close.

Equivalence checking between RTL and gate-level netlist is performed at each synthesis rev with the vendor's tool. The purpose is to prevent optimisation-time transformations from introducing behaviours not present in the verified RTL. When equivalence cannot be established over a stage, the corresponding logic is protected from further optimisation by synthesis pragmas until the mismatch is diagnosed.

13. Safety-Critical Evaluation

The interposer is intended for deployments in which incorrect action emission has serious external consequences. The relevant regulatory traditions—functional safety in automotive and industrial contexts, electrical safety in medical contexts, information-security certification in defence contexts—impose overlapping but distinct requirements on the underlying hardware. The interposer's response to this multiplicity is to maintain a single verified core and to layer domain-specific compliance material on top, rather than to attempt certification against every standard simultaneously.

For automotive deployment, the interposer would be integrated as a safety element out of context under ISO 26262 [19], with the surrounding system providing the item-level analysis. For industrial deployment, the corresponding standard is

IEC 61508 [20], and the interposer targets a systematic capability commensurate with SIL 2, with SIL 3 conditional on additional evidence not required for the base architecture. Medical deployment under IEC 62304 is a longer-term goal that would require additional software-lifecycle evidence not developed here.

14. Compliance Considerations

Beyond safety certification, the interposer intersects several compliance regimes whose relevance depends on the deployment. For deployments that emit media, the Coalition for Content Provenance and Authenticity specification [21] provides a signature scheme that the emission layer can implement natively, so that content leaving the interposer carries attestation of the model that produced it. For deployments in networked control, the IEEE 802.1Q Time-Sensitive Networking amendments [22] define the timing properties that the gateway must respect for the deployment to remain within its declared latency envelope. For deployments handling regulated data, the export and encryption regimes vary by jurisdiction and are addressed at the deployment level rather than at the device level.

15. Engaged Safety Primitives

The interposer's guarantees are ultimately reducible to a small set of hardware primitives whose correctness is assumed and whose engagement in a given deployment is a matter of configuration. It is useful to distinguish those primitives that must be engaged for the interposer's core claims to hold from those whose engagement is discretionary. In the mandatory set are: the identity-key fuses, without which the attestation engine cannot bind evidence to the device; the DMA firewall, without which the assumption of a hostile host is not meaningful; the deterministic issue discipline, without which the temporal-logic properties do not hold; and the private scratchpad partition, without which CXL-coherent access can defeat the gateway's isolation of policy-guarded state.

In the discretionary set are the tamper-evident packaging, the electromagnetic-shielding layer, and the on-die voltage and clock monitors. Each of these primitives strengthens the interposer's posture against a specific class of adversary, and a deployment may reasonably choose to engage or disengage them according to its threat profile. The interposer's compliance material identifies which primitives were engaged in the certified configuration, so that a deployment that departs from that configuration knows the extent to which it is departing from the certified evidence.

16. Risk Register

Three technical risks are worth naming explicitly. First, the policy-engine graph traversal is the dominant contributor to worst-case gateway latency, and deployments that adopt very deep policy graphs may exceed the latency envelope assumed by the roofline analysis. The mitigation is a synthesis-time bound on graph

depth, enforced at policy compile time; deployments that require deeper policies must accept a re-synthesis of the policy engine with correspondingly larger resource budgets. Second, the DMA firewall's permission table is small by design, and workloads that require many distinct DMA regions may encounter fragmentation. The mitigation is a coalescing pass in the measured runtime, which merges compatible regions before requesting an entry. Third, formal verification of the policy engine's graph traversal at general policy depth is expected to be the longest and least predictable element of the verification programme, and slippage on this element is the single most likely cause of schedule risk to the programme as a whole.

17. Limitations

The overhead figures of Section 10 are derived from cycle-accurate simulation of the current RTL and from FPGA synthesis on a VCU118, and they will shift as the RTL is refined. The throughput and roofline figures of Section 8 are derived from the datasheet specifications of the constituent components and depend on those components being instantiated as planned. Two of the six formal properties of Section 6 remain to be closed. The threat model of Section 5 is stated against a specific adversary; deployments whose actual adversary is stronger—in particular, deployments in which unrestricted invasive physical access must be assumed—will require additional evidence not developed here.

18. Conclusion

The Universal Cognitive Interposer is a proposal for placing a small, formally specified, hardware-enforced gateway on the path between an AI accelerator and the outside world. By inheriting its compute substrate from the CHAINSTATE architecture, the interposer avoids the cost of a new accelerator development and inherits a deterministic issue discipline that makes hardware safety enforcement tractable. By attaching as a PCIe/CXL endpoint rather than integrating into the host CPU, it avoids the microarchitectural attack surface that has afflicted on-die trusted execution environments, at the cost of a distinct attack surface at the attachment interface that this paper characterises in detail. FPGA synthesis and cycle-accurate simulation support the claim that the gateway's overhead is modest on the workloads for which the interposer is designed. Whether the interposer becomes a useful component of AI-safety practice depends on the outcome of the verification programme and on the willingness of deployment operators to accept a hardware-enforced constraint on the actions their systems may emit.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Pater, C.F. (2026) CHAINSTATE-1: A Cognitive System-on-Chip for AGI-Adjacent

Workloads. Paper XVII, NWO Capital/University of Agder.

[2] RISC-V International (2024) The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 20240411. https://riscv.org/technical/specifications/

[3] Costan, V. and Devadas, S. (2016) Intel SGX Explained. IACR Cryptology ePrint Archive, Report 2016/086.

[4] Pinto, S. and Santos, N. (2019) Demystifying ARM TrustZone: A Comprehensive Survey. *ACM Computing Surveys*, **51**, Article No. 130. https://doi.org/10.1145/3291047

[5] Kaplan, D., Powell, J. and Woller, T. (2016) AMD Memory Encryption. AMD White Paper.

[6] Peleg, O., Morrison, A., Serebrin, B. and Tsafrir, D. (2015) Utilizing the IOMMU Scalably. *Proceedings USENIX ATC* 2015, Santa Clara, 8-10 July 2015, 549-562.

[7] Markuze, A., Morrison, A. and Tsafrir, D. (2016) True IOMMU Protection from DMA Attacks: When Copy Is Faster than Zero Copy. *Proceedings of the* 21*st International Conference on Architectural Support for Programming Languages and Operating Systems*, 2-6 April 2016, 249-262. https://doi.org/10.1145/2872362.2872379

[8] Williams, S., Waterman, A. and Patterson, D. (2009) Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Communications of the ACM*, **52**, 65-76. https://doi.org/10.1145/1498765.1498785

[9] Amid, A., Biancolin, D., Gonzalez, A., Grubb, D., Karandikar, S., Liew, H., *et al.* (2020) Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom Socs. *IEEE Micro*, **40**, 10-21. https://doi.org/10.1109/mm.2020.2996616

[10] Karandikar, S., Mao, H., Kim, D., Biancolin, D., Amid, A., Lee, D., *et al.* (2018) FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud. 2018 *ACM/IEEE* 45*th Annual International Symposium on Computer Architecture* (*ISCA*), Los Angeles, 2-6 June 2018, 29-42. https://doi.org/10.1109/isca.2018.00014

[11] Lowe-Power, J., Ahmad, A.M., Akram, A., Alian, M., Amslinger, R., Andreozzi, M., *et al.* (2020) The gem5 Simulator: Version 20.0+. https://doi.org/10.48550/arXiv.2007.03152

[12] Pnueli, A. (1977) The Temporal Logic of Programs. 18*th Annual Symposium on Foundations of Computer Science*, Providence, 31 October-2 November 1977, 46-57. https://doi.org/10.1109/sfcs.1977.32

[13] Clarke, E.M., Henzinger, T.A., Veith, H. and Bloem, R. (2018) Handbook of Model Checking. Springer.

[14] Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., *et al.* (2018) Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. *USENIX Security Symposium*, Baltimore, 15-17 August 2018, 991-1008.

[15] Diffie, W. and Hellman, M. (1976) New Directions in Cryptography. *IEEE Transactions on Information Theory*, **22**, 644-654. https://doi.org/10.1109/tit.1976.1055638

[16] Arbaugh, W.A., Farber, D.J. and Smith, J.M. (1997) A Secure and Reliable Bootstrap Architecture. 1997 *IEEE Symposium on Security and Privacy*, Oakland, 4-7 May 1997, 65-71. https://doi.org/10.1109/secpri.1997.601317

[17] Cadence Design Systems (2024) JasperGold Formal Verification Platform. Product Datasheet. https://www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform.html

[18] Siemens, E.D.A. (2024) Questa Formal Verification. Product Datasheet. https://eda.sw.siemens.com/en-US/ic/questa/formal-verification/

[19] International Organization for Standardization (2018) ISO 26262: Road Vehicles-Functional Safety. https://www.iso.org/standard/68383.html

[20] International Electrotechnical Commission (2010) IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. https://webstore.iec.ch/publication/5515

[21] Coalition for Content Provenance and Authenticity (2024) C2PA Specification. https://c2pa.org/specifications/

[22] IEEE (2022) IEEE 802.1Q-2022: Bridges and Bridged Networks. Including Time-Sensitive Networking Amendments. https://standards.ieee.org/ieee/802.1Q/10323/

[23] TSMC Limited (2024) Advanced Technology Nodes Reference. Corporate Documentation. https://www.tsmc.com/english/dedicatedFoundry/technology

Appendix A. Formal Properties (LTL Sketch)

The properties of Section 6 are expressed here in a fragment of linear temporal logic. G denotes globally, F eventually, and → implication. The state predicates admitted(a), accepted(a), rejected(a), emitted(a) refer respectively to an action a admitted to the gateway, accepted by the policy engine, rejected by the policy engine, and emitted at the interface. attested(k) refers to a measured runtime whose signature is currently valid; quiesced refers to the fault state of the interposer.

P1. G (emitted(a) → accepted(a))—no action leaves without an accept verdict

P2. G (admitted(a) → F_ ≤ N (accepted(a) ∨ rejected(a)))—verdict returns within N cycles

P3. G (attest_evidence(s) → produced_at_state(s))—attestation reflects current state

P4. G (dma_entry(e) → attested(populator(e)))—permission table entries have attested provenance

P5. G (quiesced → ¬emitted(a) U attested_reboot)—no emission from quiesced state until re-attestation

P6. G (reads(identity_key, c) → c = attestation_crypto)—identity keys are read only by the crypto block

Appendix B. Notation

R_a denotes the rate of policy-guarded actions issued by the workload, measured in actions per second. C_a denotes the mean per-action cycle cost at the gateway. f denotes the gateway clock frequency in hertz. $F_gw = R_a \cdot C_a / f$ is the fractional overhead contributed by the gateway. N in Property P2 is a synthesis-time constant equal to the worst-case policy-evaluation depth times the graph-node cycle budget.

Appendix C. Programme and Supply-Chain Considerations

C.1. Bill of Materials and Cost Envelope

The bill of materials for the datacenter variant is dominated by the primary silicon, the high-bandwidth memory stacks, the PCIe/CXL retimers, and the power delivery. Datasheet arithmetic over these components places the unit cost in a range of approximately three to five thousand United States dollars at production volumes, with the exact figure depending strongly on memory-stack pricing at the time of build. The workstation variant halves this figure by reducing memory capacity and power delivery; the robotics variant is lower still. Non-recurring engineering for a first tape-out at a leading foundry [23] is estimated in the tens of millions of dollars, again sensitive to foundry loading and to the specifics of the process node ultimately chosen.

C.2. Supply-Chain and Manufacturing Plan

The programme's principal supply-chain risks are the memory stacks, whose lead

time is dominated by the industry-wide allocation cycle, and the PCIe/CXL retimers, whose supply is currently concentrated among a small number of vendors. The mitigation is to secure long-lead orders nine to twelve months ahead of build, to qualify a second retimer source before tape-out, and to maintain a buffer stock sufficient for the first production run. Assembly and test are planned at established OSAT partners, and burn-in and functional test protocols are adapted from those developed for the CHAINSTATE reference platform.

C.3. Provisioning and Key Management

Identity-key provisioning is performed inside the OSAT under a controlled procedure whose outputs are: an on-die public identity, a factory-signed certificate binding that identity to the die's process traceability information, and an audit trail of the provisioning transaction. The corresponding private key never leaves the die and cannot be exported through any device interface. The programme's supply-chain assumptions reduce to the integrity of this procedure and to the independence of the fuse-programming and package-marking steps.

C.4. Programme Milestones

The programme is organised into three phases. Phase 1 produces the FPGA prototype whose results are reported in Section 9 and completes the formal-verification programme of Section 12. Phase 2 produces the first tape-out of the datacenter variant, qualifies it against the SIL 2 target of Section 13, and opens an early-access programme to a small number of deployment partners. Phase 3 broadens to the remaining variants and to the compliance regimes of Section 14. Each phase concludes with a decision gate whose criteria are documented separately from this paper.