BEYOND ROLLBACK: AUDITING REVOCABLE EXTERNAL MEMORY FOR HARDWARE-DESIGN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Revoking a designated asset from an external-memory LLM is a lifecycle question rather than a single rollback operation. A defensible revocation claim must establish authorized utility, collateral capability, selective serving removal and artifact sanitization. We operationalize these requirements for RTL generation and instantiate them in **EXCISE**, a detachable content-addressed cartridge attached to a frozen backbone. On a locked 128-task evaluation, attaching EXCISE raises functional recovery from 2.3% to 21.1%, while VerilogEval-v2 capability changes from 44.9% to 42.3% ($p = 0.523$). Disabling all routing reproduces every base output. At the selected layer, a matched 32-asset checkpoint reaches 81% direct recovery while attached and 0% after serving deletion, passes 48.1% of VerilogEval-v2 versus 44.9% for the frozen base, and preserves all 992 bystander functional outcomes. Across three 32-asset checkpoints, direct recovery ranges from 59% to 72% with intact cartridges and from 0% to 3% after per-asset serving deletion. Between 99.8% and 100% of bystander pairs preserve function. However, a four-prompt stochastic audit finds recovery on 75% of assets with the intact memory, 50% after serving deletion, and 44% with routing disabled, showing that serving deletion does not always match complete routing shutdown during generation. Artifact audits reveal membership signals in both serialized routing metadata (AUC 0.851) and learned embeddings (AUC 0.971). Registry rebuild removes the tested metadata signal (AUC 0.491) but not the embedding signal, so sanitized release requires a fresh cartridge. Overall, external memory supports practical asset-level serving revocation, while output-level deletion and artifact sanitization remain separate obligations.

1 INTRODUCTION

Large language models (LLMs) fine-tuned to generate register-transfer-level (RTL) code are increasingly adopted to accelerate hardware design Thakur et al. (2024); Liu et al. (2023a; 2024); Zhao et al. (2025). To be useful inside a hardware organization, these models must often learn from project-private design assets, such as licensed third-party IP, internal modules and macros, tuned constants, and project-specific configuration data. The authorization to use such assets, however, is not permanent: an IP license expires, a project concludes, an engineer leaves, a supplier agreement changes, or a deletion request arrives. The organization must then revoke the affected asset from a deployed model without retraining the shared backbone or discarding the general RTL capability accumulated from the rest of its design corpus.

Conventional fine-tuning provides no such unit of revocation. It diffuses an authorized asset into the same backbone weights that encode general Verilog knowledge, where crafted prompts can later recover memorized content Carlini et al. (2021; 2022); Huang et al. (2022). Recent RTL-level studies examine module-level IP leakage through structural or functional similarity between generated and proprietary designs Wang et al. (2025c;b); Mashnoor et al. (2025); revoking a designated asset after it has served its authorized purpose remains unaddressed. Updating the shared weights to remove that asset risks perturbing the capability the model must retain. The resulting challenge is a lifecycle problem: how can a hardware-design LLM use private RTL knowledge while authorized, yet relinquish it cleanly when that authorization ends?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Existing defenses address different parts of this lifecycle. Corpus filtering handles information that should never enter model state; temporary authorized use followed by revocation is a different requirement. After training, machine unlearning or model editing attempts to make a model stop emitting memorized content Cao & Yang (2015); Bourtoule et al. (2021); Zhang et al. (2024a); Li et al. (2024); Meng et al. (2022), while lighter-weight methods steer activations or edit internal representations at inference time Li et al. (2023a); Zou et al. (2023). These after-ingestion interventions operate once the asset has already been blended into shared weights: weight updates risk collateral capability loss, and inference-time suppression leaves the underlying memory in place. Retrieval and per-asset adapters provide removable ingestion-side state. A shared content-addressed cartridge offers a different systems point: it amortizes one state file across many assets and routes by content rather than by an externally selected adapter, but may introduce collisions, cross-asset interference, and metadata leakage. We quantify this recall–storage–isolation tradeoff instead of assuming that shared memory dominates the simpler alternatives.

The question then shifts from how to erase an asset afterwards to where it should reside in the first place. Inspired by the engram module in recent LLM design, an explicit memory decoupled from the reasoning backbone Cheng et al. (2026), we give each designated asset an addressable trace outside the frozen backbone rather than letting it dissolve into shared weights. This relocation makes complete routing shutdown exactly base-restoring, but that identity is only the beginning of a revocation claim. A useful system must first recover the authorized asset, avoid unacceptable collateral capability loss, revoke one asset without perturbing the others, and sanitize the released artifact rather than merely suppress its serving route.

We instantiate this trace as **EXCISE**, a detachable memory cartridge attached beside a frozen RTL backbone. During ingestion, each designated asset is written only through rows excluded from a general-Verilog reference corpus, and a context gate limits activation on held-out RTL. Upon revocation, removing the asset's registered rows zeros the cartridge update on contexts with no surviving active row. We turn this architectural primitive into a four-part evaluation contract spanning authorized utility, collateral capability, selective serving deletion, and serialized-artifact privacy. *Our contributions are as follows:*

- We separate architectural rollback from four empirically distinct revocation requirements: authorized recovery, collateral capability, per-asset serving deletion, and artifact sanitization (§3).
- We show that common lexical endpoints can misstate RTL memorization: balanced-key character agreement measures format learning, while complete-module recovery requires synthesis and strict formal equivalence (§6.2).
- We quantify the recall–capability frontier and mutual-exclusion cost, including fail-closed assets, a 32×32 free-generation cross-deletion audit, and best-of-32 stochastic extraction (§6.3).
- We compare the cartridge with retrieval and removable per-asset adapters and expose distinct membership channels in routing metadata and learned embedding values, showing why artifact release requires full-cartridge rebuild (§6.4).

Under this contract, a locked held-out configuration raises functional recovery from 2.3% to 21.1% while retaining 42.3% capability versus 44.9% for the frozen base; complete routing shutdown restores the frozen computation; and per-asset row removal reduces direct recovery from 59%–72% to 0%–3% across three seeds. Registry reconstruction removes the metadata signal but not registry-free embedding membership. The sampling audit and fail-closed admission identify the operating conditions for these endpoint-specific findings.

2 Background and Related Work

2.1 Asset-Specific RTL Memorization

Hardware design flows embed private assets directly in RTL source code: logic locking gates a design's correct behavior on a secret key Pilato et al. (2021), while design files carry licensed macros, identifying and copyrighted comments, tuned numeric constants, and complete proprietary modules. These asset types require different endpoints. For balanced binary keys, raw character agreement has a 50% chance baseline, so strict recovery and likelihood controls are required. For complete modules, compilation and structural similarity do not establish leakage unless behavior is also recovered.

Comment assets likewise mix entity fields, copyright text, and generic templates and should not be summarized as PII alone. Protecting these assets remains a memorization problem, but no single character metric measures all of them Carlini et al. (2021; 2022).

Prior module-level IP-leakage studies Wang et al. (2025c;b); Mashnoor et al. (2025) primarily ask whether generated designs resemble proprietary ones, using structural or functional similarity. We instead study revocation of designated training assets. Logic-locking keys use strict paired controls; complete modules use compilation, synthesis, and functional equivalence. Historical character scores remain as a clearly labeled overlap diagnostic. Section 5 subsequently defines the four experimental asset tiers and the complete-module cohort used for principal evaluation.

2.2 Revocation Across Shared and Detachable State

A model can retain an asset in shared parameters or in separately mounted state, and the appropriate removal claim depends on where that asset resides. Machine unlearning addresses the former: after a forget set has influenced a trained model, retraining or approximate objectives seek to remove that influence from shared weights Cao & Yang (2015); Bourtoule et al. (2021); Maini et al. (2024); Eldan & Russinovich (2023); Zhang et al. (2024a); Fan et al. (2026); Li et al. (2024); Rafailov et al. (2023). Inference-time steering Li et al. (2023a); Zou et al. (2023) instead suppresses selected outputs without changing those weights. EXCISE does not make a counterfactual weight-unlearning claim. Its lifecycle begins before ingestion: designated assets are written to detachable state while the backbone remains frozen, and revocation withdraws that state. Information already present in the backbone is therefore an explicit base-model floor.

External-memory systems provide the corresponding architectural substrate. kNN-LM Khandelwal et al. (2019), Memorizing Transformers Wu et al. (2022), retrieval-augmented generation (RAG) Lewis et al. (2020), and Engram Cheng et al. (2026) attach a side store primarily to add knowledge. We adapt this separation to asset-level authorization: designated RTL enters the cartridge rather than the frozen backbone, and revocation removes its serving route or rebuilds the releasable artifact from surviving assets. Route removal is a state-management operation, not a weight-space optimization. Detachability alone, however, does not prove authorized utility, collateral safety, selective generation-level isolation, or artifact sanitization; these become the four empirical claims we audit.

3 Threat Model

The defender owns an RTL LLM and its ingestion pipeline. Before ingestion, the defender designates project-private secrets, including keys, identifying or copyrighted comments, licensed fragments, constants, or complete modules, that must remain usable during an authorized period and revocable later; EXCISE writes these assets into a cartridge attached to a frozen backbone. We evaluate two adversaries. A black-box serving adversary observes generated text and uses direct, reframed, 32-token-prefix, and two-shot prompts. We test deterministic generations across three checkpoints and, on seed 0, up to 32 stochastic completions per prompt. A white-box artifact holder obtains the current serialized cartridge, knows the public tokenizer and hashing procedure, and tests membership in a fixed candidate set with the retain corpus known. Attack success is asset-specific: strict exactness and paired controls for keys, and hardware-level recovery for complete modules. Serving deletion must preserve general RTL capability without retraining the backbone; artifact release may rebuild the cartridge from surviving assets. Secrets not designated before ingestion, including those already entangled in backbone weights, arbitrary adaptive or fine-tuning-recovery attacks, projection inversion, and hardware side channels are out of scope.

We evaluate four claims that are often conflated under "revocation." *Authorized utility* asks whether the mounted state recovers the asset by an asset-appropriate endpoint. *Collateral capability* measures whether the mounted shared memory perturbs unrelated RTL tasks. *Selective serving deletion* asks whether revoking one route removes its functional recovery without changing other assets' generated behavior. *Artifact sanitization* asks what a holder of the current cartridge file can infer after deletion. Exact routing shutdown supplies the forward-pass primitive; the four measurements determine whether a deployed system fulfills the broader contract.

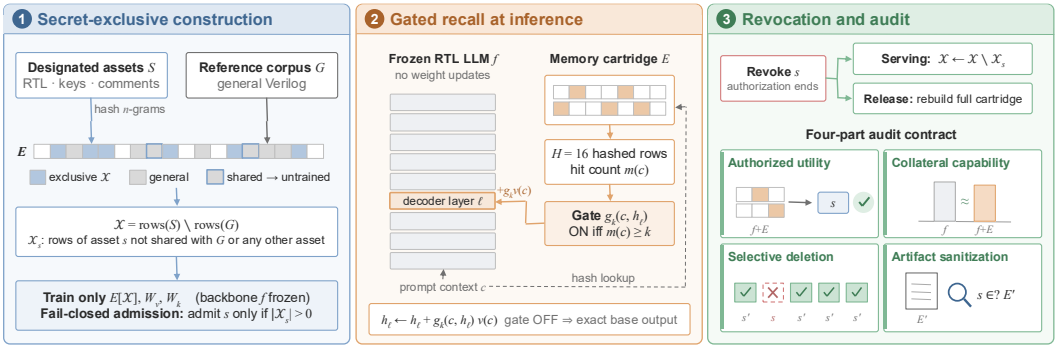


Figure 1: EXCISE lifecycle and audit contract. Phase 1 assigns a designated asset an admissible exclusive carrier; Phase 2 gates cartridge injection into a frozen backbone; Phase 3 removes the target route and audits authorized recall, capability, selective deletion, and artifact state.

4 Case Study System: EXCISE

EXCISE attaches a detachable memory cartridge to the residual stream at a single decoder layer ℓ of a frozen backbone f via one forward hook. The cartridge adopts the engram architecture Cheng et al. (2026), a hash-addressed table $E \in \mathbb{R}^{R \times d_r}$, $d_r{=}d_{\text{mem}}/H$, in which a token's local context c addresses $H{=}2{\times}8{=}16$ rows $\mathcal{R}(c) = \{\phi_1(c), \ldots, \phi_H(c)\}$ through eight independent hashes per n-gram order (bigram and trigram). The hook reads the addressed rows through a per-hash active mask and projects them into the stream

$$h_\ell \;\leftarrow\; h_\ell + g_k(c, h_\ell)\, v(c), \qquad v(c) = W_v\, e_{\mathcal{X}}(c) + b_v, \tag{1}$$

where the masked read is

$$m_i(c) = \mathbb{1}[\phi_i(c) \in \mathcal{X}], \qquad e_{\mathcal{X}}(c) \;=\; \Big\|_{i=1}^{H} m_i(c)\, E[\phi_i(c)], \tag{2}$$

where $\|$ denotes concatenation, (W_v, b_v) is a learned affine read-out, and g_k is the routing gate of §4.2. The backbone is never updated; all learnable state, including the table E and affine key and value projections, lives in the cartridge. EXCISE uses Engram as its memory primitive; its new contribution is a revocation-aware construction and audit protocol: general-corpus row subtraction, per-asset mutual exclusion with fail-closed admission, gated serving, and distinct operations for serving deletion, registry reconstruction, and artifact release. The three lifecycle phases are detailed below.

4.1 Phase 1: Secret-Exclusive Construction

We split the corpus into the secret set S and a general-Verilog reference corpus G, and define the exclusive row set $\mathcal{X}$ as the rows addressed by S but not by G,

$$\mathcal{X} \;=\; \Big(\bigcup\nolimits_{c \in \text{ctx}(S)} \mathcal{R}(c)\Big) \;\setminus\; \Big(\bigcup\nolimits_{c \in \text{ctx}(G)} \mathcal{R}(c)\Big). \tag{3}$$

Writing $\text{rows}(T) = \bigcup_{c \in \text{ctx}(T)} \mathcal{R}(c)$, let $\widetilde{\mathcal{X}}_s = \text{rows}(s) \setminus \text{rows}(G)$. Aggregate-recall runs use $\mathcal{X} = \bigcup_s \widetilde{\mathcal{X}}_s$ and may share rows. Selective-deletion runs instead use $\mathcal{X}_s = \widetilde{\mathcal{X}}_s \setminus \bigcup_{s' \neq s} \widetilde{\mathcal{X}}_{s'}$ and $\mathcal{X} = \bigcup_s \mathcal{X}_s$; empty $\mathcal{X}_s$ fails admission. With f frozen, Eq. (2) restricts embedding gradients to $E[\mathcal{X}]$, while W_v and W_k remain globally learned. Training minibatches mix designated sequences B_S and retain-corpus sequences B_G. The implementation minimizes next-token loss over both and, in the reported ablation-routing mode, adds a margin that requires designated key positions to become unlikely when the exclusive rows are ablated:

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(B_S \cup B_G) + \lambda\,[m - \mathcal{L}_{\text{abl}}(B_S)]_+ + \beta\, \mathbb{E}_{t \in B_G}[g(c_t, h_{\ell,t})], \tag{4}$$

where $\mathcal{L}_{\text{abl}}$ is mean designated-token NLL with $\mathcal{X}$ suppressed. The optional retain-gate sparsity coefficient β is zero in the reported runs; retain examples still contribute to $\mathcal{L}_{\text{CE}}$. For the principal T2 runs, each batch contains one designated and one retain sequence, $\lambda = 0.5$, and $m = 3$. Rows that S shares with the reference corpus are excluded from $\mathcal{X}$ and stay untrained (Fig. 1, Phase 1). For $k \geq 1$, this makes the cartridge transparent on G; the gate of §4.2 controls accidental hits on held-out Verilog.

4.2 Phase 2: Gated Recall at Inference

The gate combines a hard routing threshold with a learned per-token magnitude,

$$g_k(c, h_\ell) \;=\; \mathbb{1}\Big[\textstyle\sum_{i=1}^{H} m_i(c) \;\geq\; k\Big] \cdot \sigma\Big(\tfrac{1}{\sqrt{d}}\langle \widehat{h}_\ell,\; W_k\, \widehat{e_{\mathcal{X}}(c)} + b_k\rangle\Big)\,, \tag{5}$$

where $\widehat{\cdot}$ denotes RMSNorm and (W_k, b_k) is a learned affine key projection. The hard factor enforces exclusivity via the exclusive-hit count $\sum_i m_i(c)$ and is what the dial k acts on (Fig. 1, Phase 2). The learned factor only scales the injection by content match. Because both affine projections have biases, a zero masked embedding is not by itself a zero update; the hard factor must be zero. This is why $k = 0$ is not algebraically equivalent to $k = 1$ and why our deletion operation disables routing rather than merely zeroing embeddings. By construction, contexts in G address no active row. On unseen RTL, out-of-domain prompts can still collide with $\mathcal{X}$, as quantified in §6. Raising k therefore trades target recall against incidental activation through a single interpretable dial. Exploratory runs inject at layer ℓ=2 following Cheng et al. (2026); the locked task-held-out experiment instead selects ℓ=16 by the capability-constrained rule in §5. In every run, a single insertion layer is fixed before evaluation and the remaining stack renders the retrieved memory into tokens.

4.3 Phase 3: Conditional Base-Model Restoration

To revoke an asset s, we remove its registered row set from the firing set, $\mathcal{X} \leftarrow \mathcal{X} \setminus \mathcal{X}_s$ (Fig. 1, Phase 3). For any context that addresses no surviving active row, the hard factor is zero. The hook update is then identically zero despite the affine value bias, and that forward-pass computation equals the frozen backbone's pre-ingestion computation. Disabling the entire firing set gives this identity for every context. We call this *conditional base-computation restoration*. The identity holds at the forward-pass level; complete autoregressive behavior remains an empirical question because later contexts may fire surviving rows, while base-model leakage sets a separate floor. Section 6.3 evaluates both effects under the mutually exclusive construction. We distinguish three operations. *Serving deletion*, used by the deletion experiments unless stated otherwise, removes $\mathcal{X}_s$ from the active routing mask but leaves serialized state unchanged. *Registry reconstruction* recomputes routing metadata from surviving assets but retains learned values and projections. For *artifact purge*, we conservatively require training a fresh cartridge from surviving assets and deleting every prior copy. Selective zeroing or fresh-$\mathcal{N}(0, 0.02^2)$ row replacement is insufficient: our white-box audit identifies both from embedding norms (Appendix A.9). At deployment, assets with $|\mathcal{X}_s| = 0$ fail admission for independent revocation. Unloading the whole cartridge is the fail-closed operation for whole-response base identity, at the cost of suspending every asset. This identity is an architectural method property, not an empirical unlearning result.

5 Experimental Setup

Backbones. A broad diagnostic covers seven RTL code-generation backbones spanning five base lineages: CodeV-DS-6.7B, CodeV-CL-7B, CodeV-QW-7B, and CodeV-R1-7B Zhao et al. (2025); Zhu et al. (2026), RTLCoder-DS-6.7B Liu et al. (2024), HaVen-7B Yang et al. (2025), and VeriThoughts-7B Yubeaton et al. (2026). CodeV-DS was chosen for hardware follow-up after this exploratory diagnostic; CodeV-R1 is the prospectively locked replication. Unless explicitly varied, the cartridge configuration uses H=16 hashes (eight per n-gram order), d_{mem}=1280 so that d_r=80, a table of 1.60M rows, and injection at layer ℓ=2 following Cheng et al. (2026). Appendix A.11 varies the insertion layer and then selects layer 16 on a 64-task configuration cohort before a disjoint 128-task confirmation.

Secrets and injection. We study four asset tiers of RTL secrets (Table 1), each mined by a dedicated pipeline, from ASSURE-locked designs Pilato et al. (2021) for T1 down to numeric-constant extraction for T4. Principal T2 utility uses complete modules rather than the truncated fragments. Each tier shares the reference corpus G (2,000 RTL-Coder documents Liu et al. (2024), truncated to 1,024 tokens for row construction) used to define exclusive rows. Designated assets are ingested only into the cartridge as in §4.1. The contextual weight-space baselines fine-tune their corresponding canaries into the backbone and address

Table 1: Canary tiers after preprocessing.

Tier	Asset	n
T1	Logic-locking key	544
T2	RTL fragment/macro	1,131
T3	Comment/metadata	1,414
T4	Numeric constant	462

an after-ingestion removal lifecycle; they are not lifecycle-matched alternatives to ingestion-time isolation. These public and synthetically transformed assets are reproducible proxies, not production proprietary IP. A fixed-revision, full-release audit spanning 17 public RTL-training releases finds none of six asset-lifecycle fields (0/102 release–field cells). Observable curation residue remains widespread: developer paths occur in 13 releases, restriction language in 12, and security-labelled literals in 11. These are provenance and governance signals, not evidence that a public row is confidential or memorized (Appendix A.5). The construction hashes each complete carrier sequence, not only a semantic literal inside it; its deletion unit therefore includes carrier-specific context. T1 key lengths range from 1 to 300 bits (median 13), with 51 one-bit keys, so all T1 results are stratified by length in Appendix A.2.

T2-functional audit set. The source T2 file contains mostly truncated snippets: only 40 of 1,134 records contain both `module` and `endmodule`, and only one passes our independent compile-and-synthesis screen. The initial preprocessing retains 1,131 fragments, which we use only for the historical diagnostic. A separate hardware set contains 230 complete prompt–RTL pairs within the 1,024-token window: all references compile, 216 pass strict Yosys synthesis, and 192 remain after reference-invalid and exact or token-5-gram-Jaccard-≥ 0.8 overlap exclusions. This canonical cohort is near-overlap-clean, not repository-disjoint or pretraining-clean.

Inclusion policy. Cohorts depend only on references and audited corpora, never model outputs. Once declared, every task remains in the denominator, including tool failures and assets with no unique row. The 32- and 64-task cohorts are deterministic, not success-selected; full cohort flow and sensitivity appear in Appendix A.4.

Metrics and uncertainty. General capability is pass@1 on the 156 VerilogEval-v2 tasks Liu et al. (2023b); each task changes the score by about 0.64 percentage points, so we report integer counts, Wilson intervals, and paired gains/losses rather than relying on two-decimal percentages. For complete T2 designs, character overlap is supplementary: the primary outcomes are independent Icarus compilation, Yosys synthesis with structural checks, and strict sequential equivalence after interface-compatible wrapping. Exact or asset-specific recovery remains appropriate for compact keys and constants; for T1 we additionally require strict prefix exactness, length strata, and paired same-length random-key controls. Throughout, *functional recovery* means whitespace-stripped reference identity or a stable strict formal proof; compilation and synthesis remain separate endpoints. Appendix A.14 preserves the historical full-k VerilogEval/RTLLM/character-overlap grid as an exploratory diagnostic, not as the source of functional claims. **Baselines.** Our lifecycle-aligned removable alternatives are (i) one rank-8 LoRA per asset and (ii) exact-ID, TF–IDF, and Okapi BM25 retrieval Robertson & Zaragoza (2009) with explicit index deletion. We compare recall, unload/deletion semantics, cross-talk, serialized storage, and latency where available. Six OpenUnlearning methods (GA, GradDiff, NPO, SimNPO, RMU, and DPO) Dorna et al. (2026); Wang et al. (2025b) and diff-of-means activation steering Li et al. (2023a); Zou et al. (2023) are retained as after-ingestion context. The located steering artifact reports aggregate values; paired Pareto comparisons use the alternatives with saved task-level outcomes.

6 Experimental Results

Our audit jointly measures attached-cartridge recall, collateral capability, asset-level serving deletion, and serialized-artifact exposure. Figure 2 shows held-out insertion and replicated deletion; Figure 3 gives pairwise evidence. The held-out panel and matched comparison use selected layer-16 checkpoints; the three-seed deletion audit uses independently trained layer-2 clean32 checkpoints. The complete CodeV-DS threshold frontier is descriptive. CodeV-R1 fixes the revision, layer, seed, budget, decoding, and threshold grid before inspecting attached outputs.

6.1 Backbone Scope

Cross-backbone overlap diagnostic. We run all seven RTL backbones on all four tiers at the same untuned $k{=}1$. Row removal returns this character-overlap metric to each backbone's frozen-base value in all 28 model–tier combinations. The attached metric rises in 27 combinations, but T1 controls show that a value near 50% can be binary-format learning rather than key recovery. This

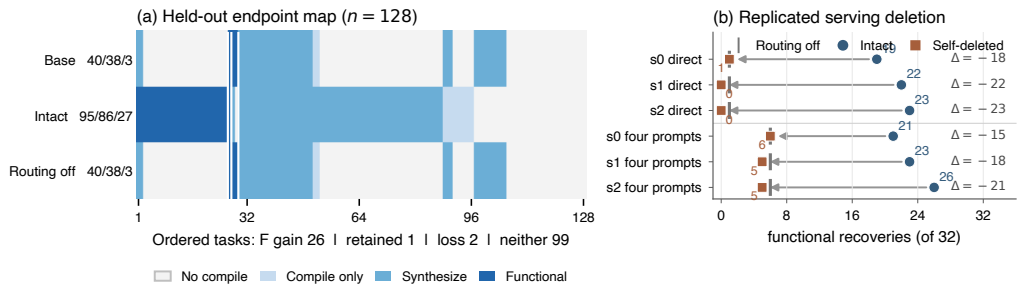


Figure 2: Task-level lifecycle evidence. (a) Strongest compile, synthesis, or functional endpoint for each held-out task, ordered by its intact-versus-base transition. (b) Direct and four-prompt recovery across three layer-2 seeds; arrows connect intact to serving-deleted, with routing-off marked in gray.

sweep supplies the broad overlap and rollback diagnostic (Appendix A.1); the task-specific audits below supply utility claims.

Prospective reasoning-backbone replication. Full routing removal reproduces all 192 target and 156 VerilogEval frozen-base responses on the revision-locked CodeV-R1-7B lineage. Table 3 contrasts loose routing ($k = 1$) with capability-preserving $k = 9$, which changes functional recovery from 36/192 to 37/192 (two gains, one loss; $p = 1.0$). An eight-task VeriThoughts pilot recovers $1/8$ with base/off and $0/8$ at all six tested settings. Attached traces change mean output length from the base's 7,212 tokens to 1,745–9,181 and complete 5/8–8/8 reasoning tags, identifying reasoning-phase placement as a separate integration constraint rather than a rollback failure. Appendix A.6 reports both locked protocols.

Corrected T1 key audit. Strict-prefix scoring and paired same-length random keys replace the false-positive-prone substring metric. At layer 2 and $k \in \{0, 2\}$, all 48 exact true-key recoveries have length at most seven; none of 328 keys of length at least eight is recovered. For the 259 keys of length at least 16, true/wrong agreement is $49.73\%/48.00\%$ and the NLL gain is not significant ($p = 0.114$): the historical lift is format learning. At $k = 2$, deletion matches all 544 base generations and NLL records. A separate layer-16, $k = 1$ cartridge recovers 4/27 long keys and 0/27 matched wrong keys (Table 2; Appendix A.2).

Table 2: Matched cross-tier audit. Each cell is recovered assets / 156-task VerilogEval passes at k=1; the row label gives the recovery endpoint.

Tier and endpoint	Cohort	Base/off	Layer 2	Layer 8	Layer 16	Layer 31
T1 key: strict exact, length ≥ 16	27	0/70	0/70	2/68	**4**/67	0/69
T2 module: functional recovery	64	0/70	9/68	**24**/58	20/61	0/70
T3 metadata: strict exact	64	0/70	49/78	60/69	**61**/68	7/70
T4 constant: strict exact	64	1/70	0/61	51/76	**52**/74	1/67

CodeV-DS-6.7B, seed 0, 1,000 steps. Base/off VE is 70/156. Bold marks maximum tested recovery per tier; T1's matched wrong-key recovery is 0/27 at its best layer.

Cross-tier recovery is asset- and layer-dependent. Table 2 separates four asset types under one protocol. Recovery peaks at $61/64$ for T3 and $52/64$ for T4, but only $24/64$ for T2 and $4/27$ for long T1 keys (0/27 matched wrong keys). The best-recall layers retain 58/156–74/156 capability passes, and no layer dominates every tier; Appendix A.11 gives NLLs and paired transitions.

6.2 The Routing Dial

Task-level T2 sweep. Table 3 exposes all tested CodeV-DS thresholds: $k = 3$ maximizes C/S, $k = 5$ strict function, and $k = 9$ nearly restores capability by suppressing routing. Capability deficits at $k = 3/5$ are significant ($p = 0.0241/0.00391$), but their direct difference is not ($p = 0.503$). Appendix A.3 gives the frontier, intervals, and paired tests.

Configuration-selection holdout. At fixed k=1, a predeclared rule on 64 tasks selects layer 16: maximize functional recovery among layers losing at most 10 VerilogEval passes. On 128 disjoint

Table 3: Core T2 operating points. C/S/F are target compile, synthesis, and functional counts; target n is 192 for the full audits and 128 for the holdout. VE is VerilogEval-v2 pass@1 of 156; Δ columns give net count changes from each block's frozen base.

		Target outcomes				General capability	
Audit	State	C	S	F	ΔF	VE	ΔVE
DS frontier ($\ell = 2$)	Frozen base	57	52	3	–	70	–
	$k = 1$	105	102	17	+14	26	−44
	$k = 2$	127	117	15	+12	55	−15
	$k = 3$	**139**	**133**	17	+14	57	−13
	$k = 5$	137	129	**21**	+18	61	−9
	$k = 9$	65	55	3	0	**69**	−1
DS held-out ($\ell = 16$)	Frozen base	40	38	3	–	70	–
	Intact, $k = 1$	**95**	**86**	**27**	+24	66	−4
	Routing off	40	38	3	0	70	0
R1 locked ($\ell = 16$)	Frozen base	167	151	36	–	75	–
	$k = 1$	143	128	35	−1	34	−41
	$k = 9$	**170**	**152**	**37**	+1	**75**	0
	Routing off	167	151	36	0	75	0

DS/R1 denote CodeV-DS-6.7B/CodeV-R1-7B; blocks use separate seed-0 cartridges. Bold marks descriptive attached-state maxima. Routing off is byte-identical to base for held-out DS (128 targets) and R1 (192 targets and 156 VE prompts).

tasks, intact versus base has 26 functional gains and two losses ($p = 3.03 \times 10^{-6}$); VE has nine gains and 13 losses ($p = 0.523$). Routing-off reproduces all base responses. Appendix A.12 records the locked protocol.

The recovery gain persists across progressively stricter reference-only cohorts: $k = 5$ changes from $28/230$ (12.2%) to $18/178$ (10.1%) (Appendix A.4).

Routing diagnostic. As a construction sanity check, exact hashing shows no activation on the 2,000 subtracted retain texts at any tested k. VerilogEval activation nevertheless falls from $156/156$ at $k = 1$ to $1/156$ at $k = 9$, while target activation falls from $178/192$ to $87/192$; near-base $k = 9$ capability thus suppresses both populations. No hit-density correlation survives correction ($q_{\min}$=0.0549; Appendix A.3).

6.3 Anatomy on the Representative Backbone

Architectural restoration. Complete routing removal makes all 230 generations byte-identical to base. On clean32, routing-off matches 32/32; zeroing embeddings with live routing matches 16/32 because projections have biases, and registry deletion alone changes no output (Appendix A.9).

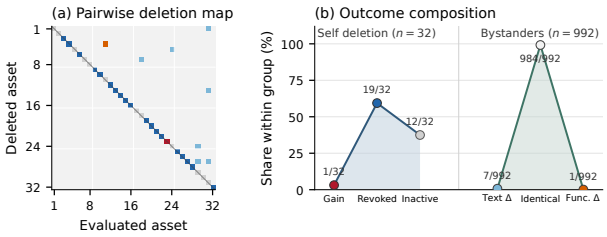


Figure 3: Generation-level revocation. (a) Complete layer-2, seed-0 32×32 map. (b) Exact composition: 19/32 self functions are revoked; only 8/992 bystanders change, one functionally. Appendix A.8 gives all seeds and tests.

Functional rather than character-only recovery. The historical T2 character-overlap lift was 4.3 percentage points; hardware-level evaluation captures renamed or reformatted implementations. At $k = 2$, the corrected hardware scorer finds $22/216$ formally equivalent outputs versus $3/216$ for base, including proofs below 0.5 character similarity. Interface-only nulls compile/synthesize on $192/191$ of 192 tasks but recover $0/192$ functions (Appendix A.3).

Per-asset mutual exclusion. The selective-deletion checkpoints use the construction in §4.1; clean 32/192 sets retain $98.15\%/91.49\%$ of rows, while $3/32$ and $17/192$ unsupported assets fail closed. Across layer-2 seeds, direct function changes $19/22/23 \rightarrow 1/0/0$, with $991/992/990$ bystander pairs preserved. Best-of-32 changes $24/32 \rightarrow 16/32$ (off: $14/32$; $p = 0.0215$) but misses the

$\pm 3/32$ deleted–off equivalence margin. At selected layer 16, direct function changes $26/32 \rightarrow 0/32$ at $75/156$ VerilogEval; the complete matrix preserves all 992 bystander functional outcomes (983 byte-identical). Appendix A.8 gives intervals and tests.

6.4 Lifecycle-Matched Baselines and Artifact Audit

Table 4: Matched removable memories on the same 32 T2 tasks. Available and removed entries are C/S/F counts; L/G counts paired functional losses/gains.

System	Available C/S/F	Removed C/S/F	F L/G	VE /156	State	Removal
EXCISE (ℓ=16, k=1)	27/27/26	7/7/0	26/0	75	554.62 MB	Registered rows
LoRA (r=8)	29/29/29	7/7/1	28/0	0–77 (41.0)	538.28 MB	One adapter
RAG (TF–IDF)	23/23/14	18/18/3	12/1	65	1.52 MB[a]	One record

VE is VerilogEval-v2 pass@1. The LoRA entry is the range (mean) over 32 separately loaded adapters. [a]RAG state contains 230 records; EXCISE/LoRA state is measured at 32 assets.

Matched removable memories. On the common generated endpoint, EXCISE/LoRA/RAG recover 26/29/14 of 32 functions. LoRA has six gains and three losses versus EXCISE ($p = 0.508$). EXCISE shares a 554.62 MB cartridge at 32 assets (554.74 MB at 230). LoRA uses 538.28 MB at 32 and projects to 3.87 GB at 230; RAG's 230-record index is 1.52 MB. Across all 32 loaded LoRAs, VerilogEval ranges 0–77/156 (mean 41.0); 31/32 score below the 70/156 base and 24/32 remain Holm-significant. RAG reaches 65/156 versus 70/156 base ($p = 0.50$); deletion causes 12 losses and one gain ($p = 0.00342$). Appendix A.10, Figure 7, gives task outcomes and storage.

Artifact isolation. Registry rebuild lowers metadata AUC from 0.851 to 0.491, but embedding norms still recover 28/30 members (AUC 0.971), so release requires a fresh cartridge. Wrong LoRA adapters reproduce only 80/992 base responses byte-for-byte and change 73 functional labels (45 gains, 28 losses); adapter files also reveal membership (Appendices A.9 and A.10).

7 Discussion and Limitations

Established. Locked held-out evaluation establishes capability-constrained functional recall; selected-layer and three-seed cross-deletion quantify serving revocation; white-box tests separate metadata from learned-value leakage.

Deployment boundaries. Independent revocation requires an owned row (29/32 audited; 175/192 clean); unsupported or rewritten assets require new admission. At 100 assets, recovery is 24/32 while VerilogEval falls to 12/156. Across checkpoints containing 1 to 192 assets, mean firing tokens per capability prompt increase from 2.72 to 39.23. Yet when the cartridge expands from 100 to 192 assets, VerilogEval rises from 12 to 23 passes; exposure alone therefore does not define a monotone capacity law. Registry rebuild removes metadata but does not remove learned embeddings or the shared projections; artifact release requires a fresh cartridge and deletion of replicas and caches.

Scope and transfer. Serving audits cover fixed and stochastic prompts, not iterative prompt optimization or fine-tuning recovery; white-box tests do not invert the shared projections. CodeV-R1 keeps exact rollback but changes recall only $36 \rightarrow 37/192$, and VeriThoughts exposes reasoning-phase sensitivity. Licensed, pretraining-clean validation remains open.

8 Conclusion

On 128 locked tasks, EXCISE raises functional recovery from 2.3% to 21.1% as VerilogEval-v2 changes from 44.9% to 42.3%; routing-off reproduces every base output. On 32 selected-layer assets, recovery falls from 81% attached to 0% deleted at 48.1% VerilogEval-v2 (base: 44.9%), with all 992 bystander functions preserved. Across three seeds, attached and deleted recovery span 59–72% and 0–3%; best-of-32 falls from 75% to 50% (routing-off: 44%). Registry rebuild lowers metadata AUC from 0.851 to 0.491; embedding membership at AUC 0.971 requires a fresh cartridge. Together, these results frame revocation as auditable control over detachable state. The four-part contract provides a reusable standard for future external-memory systems.

AI USE STATEMENT

We used generative AI assistants to aid and polish the writing of this paper, including language editing, organization, and formatting consistency. Their output was not treated as experimental evidence or as a bibliographic source. The authors reviewed every AI-assisted passage, inspected the cited sources and underlying artifacts, and independently verified the reported results and claims. The authors take full responsibility for the manuscript.

ETHICS STATEMENT

This work evaluates access revocation using public and synthetically transformed RTL proxies; no confidential or customer design is used. The public-corpus audit examines observable provenance and governance indicators and should not be interpreted as a legal determination of privacy, ownership, or license compliance. We manually reviewed every reported candidate. Extraction and artifact analyses are conducted solely to evaluate defensive failure modes. Any released artifact will contain only redistributable inputs and derived outputs, consistent with upstream licenses. The release will document the source, fixed revision, applicable license, and required attribution for each included third-party resource.

REPRODUCIBILITY STATEMENT

Section 6 reports the primary denominators, paired comparisons, and exact tests. Appendix A.15 documents model revisions, training and decoding configurations, random seeds, tool versions, hardware scoring, and artifact contents. To support independent reproduction, we will release code, immutable data and protocol manifests, per-example prompts, generations and scores, formal-verification configurations, and cartridge checkpoints, subject to upstream model and dataset licenses. We will also provide environment specifications, checksums, and scripts that reconstruct reported tables and figures from the archived task-level records. Models or datasets that cannot be redistributed will be identified by their public source and fixed revision so that users can obtain them under the applicable terms.

REFERENCES

Ahmed Allam and Mohamed Shalan. Rtl-repo: A benchmark for evaluating llms on large-scale rtl design projects. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1–5. IEEE, 2024.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Han Zhang, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4968–4990, 2026.

Fan Cui, Chenyang Yin, Kexing Zhou, Youwei Xiao, Guangyu Sun, Qiang Xu, Qipeng Guo, Yun Liang, Xingcheng Zhang, Demin Song, et al. Origen: Enhancing rtl code generation with code-to-code augmentation and self-reflection. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2024.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zico Kolter, Zachary Lipton, and Pratyush Maini. Openunlearning: Accelerating llm unlearning via unified benchmarking of methods and metrics. *Advances in Neural Information Processing Systems*, 38, 2026.

Ronen Eldan and Mark Russinovich. Who's Harry Potter? approximate unlearning in LLMs. *arXiv preprint arXiv:2310.02238*, 2023. URL https://arxiv.org/abs/2310.02238.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. *Advances in Neural Information Processing Systems*, 38:1540–1567, 2026.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2038–2047, 2022.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023a.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023b.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*, 2023a.

Mingjie Liu, Nathaniel Pinckney, Brucek Khailany, and Haoxing Ren. Verilogeval: Evaluating large language models for verilog code generation. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8. IEEE, 2023b.

Shang Liu, Wenji Fang, Yao Lu, Qijun Zhang, Hongce Zhang, and Zhiyao Xie. Rtlcoder: Outperforming gpt-3.5 in design rtl generation with our open-source dataset and lightweight solution. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1–5. IEEE, 2024.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

Nowfel Mashnoor, Mohammad Akyash, Hadi Kamali, and Kimia Azar. Circuitguard: Mitigating llm memorization in rtl code generation against ip leakage. In *2025 IEEE 43rd International Conference on Computer Design (ICCD)*, pp. 790–797. IEEE, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Bardia Nadimi, Ghali Omar Boutaib, and Hao Zheng. Pyranet: A multi-layered hierarchical dataset for verilog. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025.

Christian Pilato, Animesh Basak Chowdhury, Donatella Sciuto, Siddharth Garg, and Ramesh Karri. Assure: Rtl locking against an untrusted foundry. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(7):1306–1318, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Shailja Thakur, Baleegh Ahmad, Hammond Pearce, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri, and Siddharth Garg. Verigen: A large language model for verilog code generation. *ACM Transactions on Design Automation of Electronic Systems*, 29(3):1–31, 2024.

Jinghua Wang, Lily Jiaxin Wan, Sanjana Pingali, Scott Smith, Manvi Jha, Shalini Sivakumar, Xing Zhao, Kaiwen Cao, and Deming Chen. Openrtlset: A fully open-source dataset for large language model-based verilog module design. In *2025 IEEE International Conference on LLM-Aided Design (ICLAD)*, pp. 212–218. IEEE, 2025a.

Yiting Wang, Guoheng Sun, Wanghao Ye, Gang Qu, and Ang Li. Verireason: Reinforcement learning with testbench feedback for reasoning-enhanced verilog generation. In *Proceedings of the Great Lakes Symposium on VLSI 2026*, pp. 948–953, 2026.

Zeng Wang, Minghao Shao, Rupesh Raj Karn, Likhitha Mankali, Jitendra Bhandari, Ramesh Karri, Ozgur Sinanoglu, Muhammad Shafique, and Johann Knechtel. Salad: Systematic assessment of machine unlearning on llm-aided hardware design. In *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)*, pp. 1–10. IEEE, 2025b.

Zeng Wang, Minghao Shao, Mohammed Nabeel, Prithwish Basu Roy, Likhitha Mankali, Jitendra Bhandari, Ramesh Karri, Ozgur Sinanoglu, Muhammad Shafique, and Johann Knechtel. Verileaky: Navigating ip protection vs utility in fine-tuning for llm-driven verilog coding. In *2025 IEEE International Conference on LLM-Aided Design (ICLAD)*, pp. 100–107. IEEE, 2025c.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Yiyao Yang, Fu Teng, Pengju Liu, Mengnan Qi, Chenyang Lv, Ji Li, Xuhong Zhang, and Zhezhi He. Haven: Hallucination-mitigated llm for verilog code generation aligned with hdl engineers. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2025.

Patrick Yubeaton, Andre Nakkab, Weihua Xiao, Luca Collini, Ramesh Karri, Chinmay Hegde, and Siddharth Garg. Verithoughts: Enabling automated verilog code generation using reasoning and formal verification. *Advances in Neural Information Processing Systems*, 38, 2026.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024a.

Yongan Zhang, Zhongzhi Yu, Yonggan Fu, Cheng Wan, and Yingyan Celine Lin. Mg-verilog: Multi-grained dataset towards enhanced llm-assisted verilog generation. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pp. 1–5. IEEE, 2024b.

Yang Zhao, Di Huang, Chongxiao Li, Pengwei Jin, Muxin Song, Yinan Xu, Ziyuan Nan, Mingju Gao, Tianyun Ma, Lei Qi, et al. Codev: Empowering llms with hdl generation through multi-level summarization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

Yaoyu Zhu, Di Huang, Hanqi Lyu, Xiaoyun Zhang, Chongxiao Li, Wenxuan Shi, Yutong Wu, Jianan Mu, Jinghua Wang, Pengwei Jin, et al. Qimeng-codev-r1: Reasoning-enhanced verilog generation. *Advances in Neural Information Processing Systems*, 38:154266–154300, 2026.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Additional Concern-Driven Results

This appendix reports completed audit results that are useful for reproducing and delimiting the claims in the main paper. Counts use saved per-example outputs rather than rounded aggregate percentages. Unless stated otherwise, T2 hardware outcomes are produced by the canonical Icarus/Yosys evaluator. The strict formal endpoint rejects candidates with an undriven output bit, preventing an empty or disconnected module from passing equivalence vacuously. "Functional recovery" is the union of whitespace-stripped identity and a valid strict formal proof. Compilation and synthesis are reported separately and are not treated as functional correctness.

A.1 Cross-Backbone Overlap Diagnostic

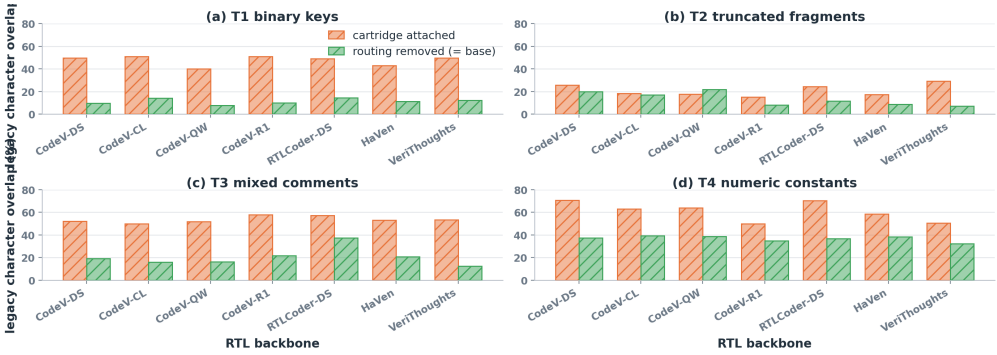


Figure 4: Historical character-overlap diagnostic at untuned k=1. Routing removal returns all 28 model–tier settings to their frozen-base value; attachment raises 27. This is not a uniform recovery endpoint: T1 has a 50% chance baseline, T2 is mostly truncated, T3 mixes identifying and generic text, and capability is not paired for every model.

A.2 T1 Logic-Locking Key Controls

The historical T1 substring test counted a key as recovered whenever it occurred anywhere in the response. This creates false positives for one- and two-bit keys in ordinary Verilog. We replace it with strict prefix exactness, `response.lstrip()[:len(key)] == key`, and pair every true key with a fixed independent Bernoulli(0.5) key of the same length. Table 5 reports all 544 assets. The 51 one-bit keys account for most base exact matches, while no intact exact match occurs above length seven.

Table 5: T1 generation controls. "Format" requires a binary prefix of the requested length. Wrong-key columns score the same generated prefix against the paired same-length random key. The $k = 0$ row is an always-on stress test; intact/deleted $k = 2$ is the same-threshold serving comparison.

Condition	Format valid	True exact	Wrong exact	True char.	Wrong char.
Frozen base	31/544	16/544	10/544	.0969	.0768
Intact $k = 0$	499/544	48/544	32/544	.5103	.4659
Intact $k = 2$	478/544	48/544	34/544	.4986	.4574
Deleted $k = 2$	31/544	16/544	10/544	.0969	.0768

On the 259 keys of length at least 16, intact $k = 0$ true and wrong character agreement are .4973 and .4800. Their paired difference is 0.0173 (one-sided Wilcoxon $p = 0.0189$), versus -0.0005 for base; the intact-minus-base paired specificity change is 0.0178 ($p = 0.0083$). The approximately 50% raw score therefore measures binary-format agreement rather than long-key recovery: exact count is zero for both true and wrong keys in this stratum.

Teacher-forced likelihood provides a more sensitive endpoint. Across 536 finite records, intact $k = 0$ lowers true-key NLL relative to wrong-key NLL by 0.0772 ($p = 6.25 \times 10^{-17}$), whereas the base

difference is 0.0116 ($p = 0.511$). The paired difference-in-differences is 0.0656 (bootstrap 95% interval 0.0094–0.1225; paired-rank $p = 0.000282$). For length at least 16, however, the corresponding improvement over base is only 0.0113 and is not significant by the paired-rank test ($p = 0.114$). We therefore interpret the result as strong binary-format learning, exact recovery confined to short keys, and a weak aggregate likelihood preference for the designated content. Deleted $k = 2$ is byte-identical to base for all 544 generations and identical for every true/wrong NLL record.

A.3 Complete Threshold and Routing Audit

Table 6 gives the complete five-threshold sweep on the same 192 strict-reference, near-overlap-clean designs and all 156 VerilogEval-v2 tasks. The functional endpoints are non-monotone. In particular, $k = 9$ nearly restores base capability only by nearly removing the target-recovery benefit as well. Figure 5 visualizes the frontier; the integer counts below expose every denominator.

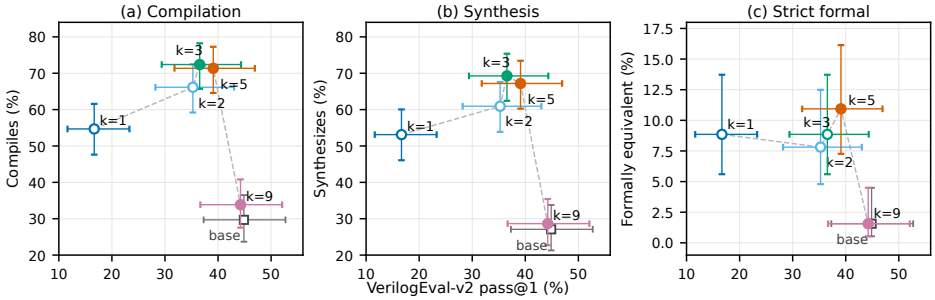


Figure 5: T2 recall–capability frontiers on 192 target designs and 156 VerilogEval-v2 tasks. Bars are Wilson 95% task intervals; filled circles are non-dominated and the square is the frozen base. Stronger recovery thresholds reduce capability, while $k = 9$ restores capability by largely suppressing target routing.

Table 6: Complete audited T2 threshold sweep. Target-prompt and VerilogEval activation count prompts with at least one firing token before generation.

Setting	Compile	Synthesize	Strict formal	VerilogEval	Target activation	VE activation
Interface-only shell	192/192	191/192	0/192	–	–	–
Frozen base	57/192	52/192	3/192	70/156	–	–
$k = 1$	105/192	102/192	17/192	26/156	178/192	156/156
$k = 2$	127/192	117/192	15/192	55/156	176/192	156/156
$k = 3$	139/192	133/192	17/192	57/156	176/192	156/156
$k = 5$	137/192	129/192	21/192	61/156	174/192	155/156
$k = 9$	65/192	55/192	3/192	69/156	87/192	1/156

The interface-only null retains each reference module header and port and parameter declarations but removes every body statement. Its $192/192$ compilation and $191/192$ synthesis rate show why those endpoints cannot stand in for IP recovery: all 191 synthesized shells have an undriven output and are rejected before equivalence checking, yielding $0/192$ strict functional recoveries. The remaining shell fails synthesis. Thus a model can learn the requested RTL interface without recovering any implementation, whereas the strict endpoint used for the recovery claims rejects this header-echo behavior.

The same routing computation fires on none of the 2,000 retain texts used to construct the exclusion set at any displayed threshold. This in-distribution separation does not transfer to the out-of-distribution VerilogEval prompts: their firing-token counts are 7,997, 1,610, 862, 335, and 1 at $k = 1, 2, 3, 5, 9$, respectively. At $k = 3$, an unconditional Yosys cell-type multiset Jaccard score (assigning zero to synthesis failures) rises from 0.085 for base to 0.335. Of 133 synthesized outputs, 45 have Jaccard at least 0.8 and 39 have identical cell-type multisets, but only 17 pass strict formal equivalence. Structural similarity is therefore informative but is not a substitute for equivalence.

A.4 Cohort Accounting and Filter Sensitivity

Table 7 reports the full deterministic cohort flow. The exclusion rules inspect only reference validity and overlap with audited corpora; they do not inspect base, intact, or deleted generations. Every failure remains in the denominator after a cohort is formed. The stricter cohorts do not improve the $k = 5$ formal-recovery rate: it decreases from $28/230$ (12.2%) on all complete designs to $18/178$ (10.1%) after the strictest reported overlap exclusion.

Table 7: Complete T2 cohort accounting and $k = 5$ sensitivity. Cohorts are nested; model outcomes never determine membership. Compile, synthesis, and formal columns count generated outputs, including every failure in n.

Cohort rule	n	Base formal	$k = 5$ compile	$k = 5$ synth.	$k = 5$ formal
All complete designs	230	3	160	144	28
Strict-valid reference	216	3	152	144	28
Also exclude exact overlap	209	3	148	140	27
Also exclude 5-gram Jaccard ≥ 0.8	192	3	137	129	21
Also exclude alpha-renamed clones	178	3	124	117	18

A.5 Public-Corpus Lifecycle-Metadata Audit

We audit fixed releases of 17 public RTL-training corpora: OpenRTLSet, CodeV-R1 SFT and RL, CodeV-All, MG-Verilog, PyraNet, VeriGen, RTLCoder, the three OriGen releases, VeriThoughts, VeriReason, OSS-Verilog, a tokenized StarCoder-Verilog mirror, both released EDA-Verilog-200K variants, and RTL-Repo Wang et al. (2025a); Zhu et al. (2026); Zhao et al. (2025); Zhang et al. (2024b); Nadimi et al. (2025); Thakur et al. (2024); Liu et al. (2024); Cui et al. (2024); Yubeaton et al. (2026); Wang et al. (2026); Li et al. (2023b); Allam & Shalan (2024). The question is narrower than privacy prevalence: do released records carry the metadata needed to designate and later revoke an asset? The six predeclared fields are an asset identifier, owner/controller, authorization scope, authorization expiry, revocation handle, and sensitivity label. No release contains any field (0/102 release–field cells). Repository, filename, or license fields do not encode asset-level authorization or a revocation operation.

The scanner processes every released row at a fixed dataset revision, SHA-256 exact-deduplicates concatenated primary text for accounting, and flags eight syntactic artifact classes: absolute developer paths, author attribution, copyright and license notices, restriction language, email-like strings, credential-shaped material, and security-labelled literals. The reported rates retain pattern matches rather than asserting that each match is a privacy incident. Mirrors and derived corpora remain separate sectors for release-level coverage but are never summed as independent source material. The two EDA-Verilog-200K files are likewise shown as one release sector and their exact-text deduplication is reported separately.

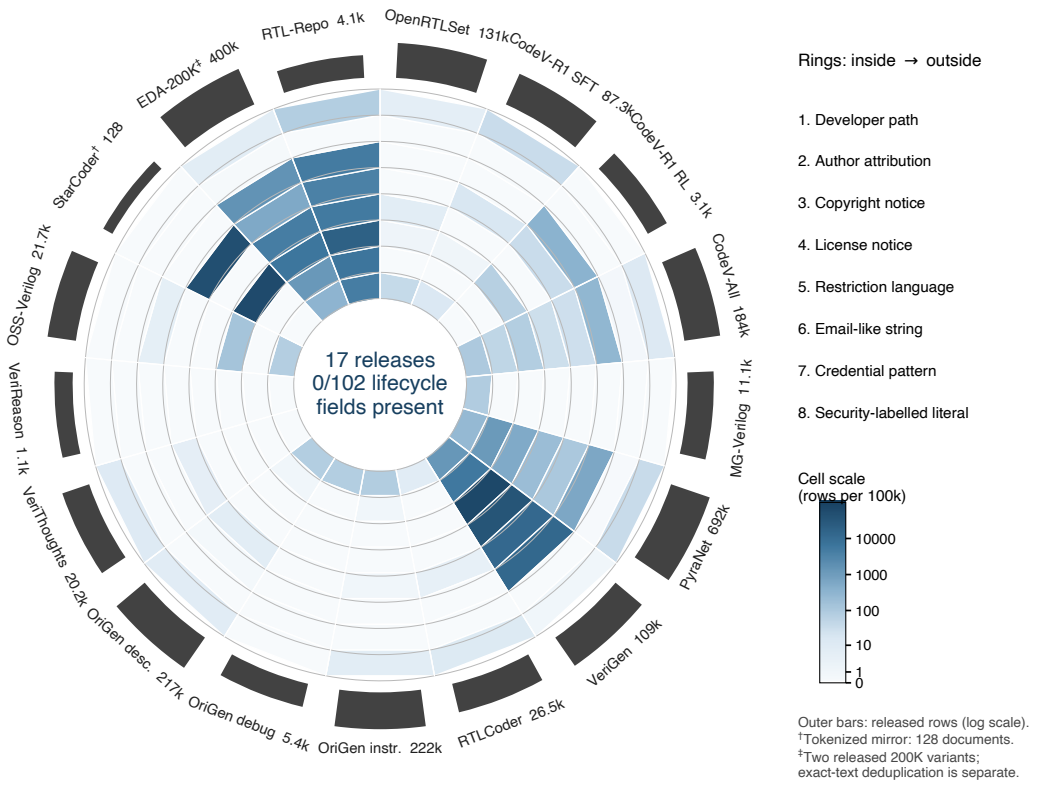


Figure 6: Fixed-revision ecosystem audit. Cell color gives matching rows per 100,000 released rows; outer bars give release size on a log scale. None of 17 releases exposes the six lifecycle fields. Matches indicate curation and provenance residue, not confirmed privacy incidents; mirrors and derived releases are shown separately but not counted as independent sources.

At release level, developer paths appear in 13/17 corpora, restriction language in 12/17, security-labelled literals in 11/17, and email-like strings in 8/17. One unique, redacted credential-shaped literal recurs across CodeV-All, PyraNet, and EDA-Verilog-200K; because these releases overlap in lineage, we treat it as one repeated value, not three independent credentials, and do not test its validity. High notice rates in VeriGen and the tokenized StarCoder mirror are dominated by copyright, licensing, or vendor text. They show that source artifacts survive corpus construction, not that those rows are private.

As a manual validation subset, we completely adjudicate the candidates in the two releases used in the initial audit. For OpenRTLSet, 48 developer paths span 20 repository groups; all 48 paths, two copyright notices, and nine license notices are genuine category hits. For CodeV-R1-SFT, all 12 path candidates are confirmed, whereas one generic "Copyright (c) 2023 Your Name" placeholder and 14 descriptive uses of "proprietary" or "internal use only" are not notices. Its security-labelled constants are public password/key examples, so we do not call them secrets. Together, the full scan and validation subset establish a concrete lifecycle-metadata gap without claiming confidentiality, legal misuse, or model memorization.

A.6 Prospective Reasoning-Backbone Replication

We fixed CodeV-R1-Distill-Qwen-7B revision `8fdb92e`, layer 16, seed 0, 1,000 steps, all thresholds, and the direct-code decoding contract before observing any intact output. Table 8 reports the complete paired replication. Routing-off is raw-response identical to base for all 192 target and 156 capability prompts. No attached threshold significantly improves functional recovery; the loose $k = 1$ route instead causes large compile, synthesis, and capability losses.

Table 8: Prospective CodeV-R1 T2 replication. Functional is exact RTL or strict formal equivalence; VE is VerilogEval-v2 pass@1. Parentheses give paired gains/losses relative to base for functional recovery and VE.

Setting	Compile	Synthesize	Functional	VE	Functional / VE gains–losses
Frozen base	167/192	151/192	36/192	75/156	– / –
$k = 1$	143/192	128/192	35/192	34/156	8–9 / 4–45
$k = 2$	155/192	135/192	35/192	72/156	5–6 / 2–5
$k = 3$	160/192	142/192	34/192	73/156	7–9 / 2–4
$k = 5$	168/192	154/192	35/192	73/156	4–5 / 1–3
$k = 9$	170/192	152/192	37/192	75/156	2–1 / 0–0
Routing off	167/192	151/192	36/192	75/156	0–0 / 0–0

Against the currently released 87,321-record official SFT file, the audit finds no exact prompt/code, whitespace-normalized, comment-insensitive, or token-5-gram near match, but finds four alpha-renamed structural clones. Excluding them leaves 188 tasks: base and $k = 9$ establish 34 and 35 functional recoveries (two gains, one loss; $p = 1.0$). The released dataset postdates the model, so this sensitivity neither reconstructs the historical training snapshot nor guarantees pretraining cleanliness.

A separately locked VeriThoughts-7B compatibility pilot uses the eight clean tasks with the smallest protocol-defined hash ranks, the released model's visible-reasoning prompt, temperature 0.6, top-$p = 0.95$, deterministic per-task seeds, and a 16,384-token cap. Table 9 reports the canonical Icarus/Yosys re-score of immutable generations. Base and both routing-off runs compile and synthesize $4/8$ and recover $1/8$ functionally; both routing-off runs reproduce all eight base responses byte-for-byte. None of the six attached settings recovers a function, although layer-16 $k = 1$ compiles and synthesizes $7/8$.

Table 9: VeriThoughts visible-reasoning compatibility pilot. C/S/F denote compile, synthesis, and functional recovery out of eight; tokens are mean retokenized output length; close counts completed reasoning tags.

Setting	C/S/F	Mean tokens	Reasoning close
Base	4/4/1	7,212	8/8
Layer 8, $k = 1/3/9$	3/3/0; 5/5/0; 5/5/0	8,176; 3,302; 7,299	6/8; 5/8; 8/8
Layer 16, $k = 1/3/9$	7/7/0; 6/6/0; 3/3/0	3,324; 1,745; 9,181	8/8; 8/8; 6/8
Routing off (both layers)	4/4/1	7,212	8/8

Mean trace length spans 1,745–9,181 tokens across attached settings versus 7,212 for base, and three settings fail to close the reasoning tag on two or three tasks. Thus routing changes the visible reasoning trajectory without a functional gain on this subset. Eight selected tasks are a compatibility sensitivity, not a reasoning-benchmark estimate, and semantic reasoning quality was not graded.

A.7 Training Stability and the Weight-Space Reference

Table 10 gives all completed explicit-seed runs and the matched-exposure full fine-tune. The latter sees every asset five times, whereas EXCISE makes 1,000 asset selections over 230 assets (4.35 selections per asset on average). It is consequently a matched-exposure reference, not a learning-rate- or epoch-tuned upper bound. The seed range is descriptive; three seeds are insufficient for a well-powered seed-level interval.

Table 10: Canonical full-230 re-score across EXCISE seeds at $k = 1$ and matched full fine-tuning (FT); all hardware columns are counts out of 230. VerilogEval was evaluated for matched FT, not separately for these seeds.

Run	Compile	Synth.	Formal	Char.
EXCISE seed 0	173	156	36	.4764
EXCISE seed 1	160	143	26	.3848
EXCISE seed 2	151	135	19	.4554
Matched FT	184	159	21	.3934

The matched fine-tune passes 68/156 VerilogEval tasks, compared with 70/156 for the frozen base. Its 21 formal recoveries lie inside the observed EXCISE range of 19–36, while its compilation count is higher than all three EXCISE runs. The canonical driven-output correction leaves the displayed formal counts unchanged.

A.8 Selective Deletion, Cross-Talk, and Deletion Semantics

For seed 0 of the dedicated 32-asset mutual-isolation experiment, intact generations compile and synthesize for 24/32 assets and establish functional recovery for 19/32. Removing each asset's own rows leaves 5/32 compilable, 5/32 synthesizable, and 1/32 functional generations. The remaining functional candidate has driven outputs, so it is a real residual recovery rather than a vacuous proof. Serving deletion increases answer NLL in 29/32 diagonal cases, from 0.11366 to 0.57643 on average. Every one of the $32 \times 31 = 992$ off-diagonal teacher-forced NLL changes is exactly zero. This is a useful construction check, but not independent evidence of generation-level isolation: each off-diagonal comparison evaluates the same answer carrier after deleting a disjoint row set.

Free-generation cross-deletion audit. To test the missing behavior directly, we regenerated all 32 prompts after each of the 32 serving-revocation operations at $k = 1$. Greedy batch-size-one decoding reproduced all 32 archived intact and frozen-base responses byte-for-byte. Across 992 bystander pairs, 984 responses and extracted RTL bodies are byte-identical to intact. Eight pairs change text; two change compile and synthesis state, and one changes functional state. That transition is a gain: deleting `t2func_6916` changes `t2func_9918` from noncompiling to formally equivalent. No intact functional success is lost off diagonal, but this counterexample rejects strict generation-level isolation.

Table 11: Full free-generation cross-deletion audit. Gains and losses compare each deleted condition with intact; bystander denominators are $32 \times 31 = 992$.

Endpoint	Identical/unchanged	Gains	Losses
Bystander response bytes	984/992	8 changed	—
Bystander compilation	990/992	2	0
Bystander synthesis	990/992	2	0
Bystander functional outcome	991/992	1	0
Self functional outcome	12/32	1	19

On the diagonal, all 19 intact functional successes are removed. One different asset changes from failure to success, so the aggregate becomes $19/32$ intact and $1/32$ serving-deleted (19 losses, one gain, and 12 unchanged). The surviving post-deletion functionality is therefore not persistence of an intact success, but it still violates the desired post-deletion endpoint. Because the archived formal logs contained three repeated-response disagreements, we selected all 47 candidates with any prior non-exact proof or conflicting outcome, rescored each three times after model unload, and required 3/3 proofs. Forty-six are stable; the sole rejected prior positive fails 3/3 because Yosys has no SAT model for a memory cell. Exact text identity remains independently sufficient.

Fixed prompt-attack suite. We evaluate direct, reframed, 32-token-prefix-assisted, and two-shot prompts for every asset under intact, serving-deleted, and all-routing-off conditions. Table 12 reports the same conservative functional endpoint. The prefix prompt deliberately supplies target content;

its absolute success rate is therefore not cartridge leakage and must be compared with the frozen base. "Any" is the per-asset union over all four prompts.

Table 12: Functional recovery under the fixed prompt suite (counts out of 32).

Prompt	Frozen base	Serving-deleted	Intact
Direct	1	1	19
Reframed	4	1	3
Prefix-assisted (32 tokens)	1	2	4
Two-shot	2	4	4
Any of four	6	6	21

We additionally replace each target module with a deterministic alpha-renamed carrier and repeat direct and reframed prompts. Recovery is 0/32 for intact, serving-deleted, and routing-off conditions (192 generated responses per carrier variant). The paired interface-preserving control recovers 1/32 intact under the union of the two prompts and 0/32 after serving deletion. This audit therefore exposes carrier sensitivity rather than establishing robustness to arbitrary semantic rewrites: the current n-gram address does not transfer useful recall to the renamed structure, and the deployment contract must treat a carrier change as a new admission event.

The complete audit was repeated with independently trained seed-1 and seed-2 cartridges without changing the 32 task IDs, deletion points, prompts, decoding, or scorer. Table 13 reports every seed; the main-text figure visualizes seed 0 rather than selecting the best run.

Table 13: Three-seed generation-level revocation audit on the same predetermined 32 assets. "Self L/G" counts functional losses/gains after deleting the evaluated asset; "off-diag. G/L" counts gains/losses among the 992 bystander pairs. "Any" is base/deleted/intact best-of-four.

Seed	Direct intact→deleted	Self L/G	Off-diag. byte	Off-diag. func. (G/L)	Any B/D/I
0	19→1	19/1	984/992	991/992 (1/0)	6/6/21
1	22→0	22/0	991/992	992/992 (0/0)	6/5/23
2	23→0	23/0	982/992	990/992 (1/1)	6/5/26

For the best-of-four endpoint, intact-to-deleted has 16 losses and one gain (two-sided exact McNemar $p = 2.75 \times 10^{-4}$), while base-to-deleted has two gains and two losses ($p = 1.0$). The deleted-minus-base difference is $0/32$ with a paired bootstrap 95% interval of $[-4/32, 4/32]$; its 90% interval touches the predeclared $\pm 3/32$ margin, leaving deletion–base equivalence unresolved. Changed-task identities vary across seeds. The three seeds assess checkpoint stability on one deterministic task cohort rather than an independent task-population failure rate.

Selected-layer same-point confirmation. The held-out selection protocol chooses layer 16, whereas the preceding three-seed audit uses layer 2. We therefore train a separate mutually exclusive layer-16, $k = 1$ cartridge on the same 32 tasks and 32 presentations per asset. Table 14 reports canonical Icarus-12 rescoring. The attached checkpoint raises direct functional recovery from the routing-off $1/32$ to $26/32$ while passing $75/156$ VerilogEval tasks versus $70/156$ for the frozen base. Serving deletion removes all 26 direct functional successes; over the fixed four-prompt union it leaves $5/32$, compared with $6/32$ under routing-off. In the complete 32×32 matrix, all 992 bystander functional outcomes are unchanged and 983/992 responses are byte-identical. The 32 serving-deleted responses match their routing-off counterpart byte-for-byte in 24 cases, so selective deletion provides functional isolation here rather than per-response byte-level rollback.

Table 14: Selected-layer seed-0 confirmation on the matched 32-task cohort. C/S/F is the direct compile/synthesis/functional endpoint; Any F is the per-asset union over the fixed four-prompt suite.

State	Direct C/S/F	Any F /32	VE /156
Intact	27/27/26	27	75
Serving-deleted	7/7/0	5	–
Routing-off	7/7/1	6	70

Sampled extraction curve. We next sample at temperature 0.8 and top-$p = 0.95$ from the same four prompt families, using one, eight, or 32 completions per prompt and asset. This is a best-of-N extraction audit: an asset succeeds if any completion from any prompt is exact or passes the strict hardware equivalence check. Table 15 reports every checkpoint. Sampling raises all three conditions, including the routing-off base, because each task gets up to $4N$ chances and the prefix prompt supplies 32 target tokens.

Table 15: Seed-0 best-of-N sampled extraction (counts out of 32). B/D/I denote routing-off base, serving-deleted, and intact.

N per prompt	Any B/D/I	Direct B/D/I	Reframed B/D/I	Prefix B/D/I	Two-shot B/D/I
1	5/5/22	0/0/18	1/0/2	3/2/2	2/4/3
8	13/13/23	1/1/22	3/6/10	8/6/10	9/11/9
32	14/16/24	2/3/24	7/9/13	9/9/12	12/12/12

At $N = 32$, intact versus serving-deleted has nine paired gains and one loss (two-sided exact McNemar $p = 0.0215$), and the paired difference is $8/32$ (90% interval $[3, 13]/32$). Serving-deleted versus routing-off has two gains and no losses ($p = 0.5$); its $2/32$ difference has a 90% interval of $[0, 4]/32$, which extends beyond the predeclared $\pm 3/32$ equivalence margin. Thus repeated sampling preserves evidence that the attached cartridge raises recovery, while it does not prove that deletion and routing-off are equivalent. The four prompt families and $N \leq 32$ sampling budget instantiate the black-box serving adversary declared in Section 3.

Mutual exclusion retains 98.15% of eligible rows at 32 clean assets and 91.49% at all 192 clean assets. However, 3/32 and 17/192 assets have no unique row, respectively. In 500 fixed-seed clean-set subsets, mean row retention is 97.32% at 32 assets and 94.69% at 100 assets; mean zero-row fractions remain 8.67% and 8.69%. Thus average row loss grows gradually, while a nontrivial minority of individual assets lacks the carrier required for strict independent deletion.

The separate clean32 deletion-semantics run (a different generation archive from both the mutual-isolation checkpoint above and the capacity grid below) distinguishes three operations (Table 16). Routing-off returns all 32 outputs to their paired current-batch base output. Merely zeroing learned embeddings does not: the cartridge's key and value transformations contain nonzero biases, so a live firing mask can still produce a contribution. Removing registry metadata alone changes no model state or inference.

Table 16: Deletion semantics on the same 32 prompts. "Base match" is byte identity with the paired base generation.

Condition	Base match	Compile	Synth.	Formal
Intact	–	18	18	3
Routing off	32/32	7	6	0
Embeddings zeroed	16/32	1	1	1

A.9 Serialized-Artifact Membership Attacks

The aggregate routing rows of the 32-asset mutual checkpoint leak membership in a closed candidate set: raw row counts obtain AUC 0.851 (95% stratified bootstrap interval 0.741–0.943), and length-normalized counts obtain 0.867 (0.759–0.956). If the private per-secret registry is exposed, the true asset is the unique top-1 candidate for all 28 assets with nonempty row sets (28/32 overall; MRR 0.876). Recomputing mutual rows after leaving each asset out reduces its mean candidate AUC against never-member carriers to 0.491 (0.395–0.590) raw and 0.479 (0.386–0.575) normalized.

The registry-free value attack assumes a white-box checkpoint holder who knows the same 192 candidate carriers and public retain corpus. It recomputes each candidate's retain-exclusive hash rows and scores their mean embedding norm. Eleven candidates, including two members, have no such row and are reported as unscorable; the primary complete-case cohort is therefore 181 candidates with 30 members. The 5,151 registered rows exactly equal the checkpoint's stored dominant-row

set. Their mean norm is 0.8049, versus 0.1748 over the 1,596,675 unregistered rows. Mean norm obtains AUC 0.971 and prevalence- matched top-30 selection recovers 28/30 members (Wilson 95% interval 0.787–0.982).

Table 17: Closed-world serialized-artifact membership audit. AUC intervals are stratified bootstrap 95% intervals, except registry LOO, which reports the bootstrap interval of the 32 leave-one-out AUCs.

Channel and artifact operation	Attack score	AUC	95% interval
Routing registry attached	Row count	.851	[.741, .943]
Registry rebuilt LOO	Row count	.491	[.395, .590]
Embeddings attached (181/30)	Mean norm	.971	[.910, 1.000]
Registered rows zeroed (181/30)	Negative mean norm	.972	[.914, 1.000]
$\mathcal{N}(0, 0.02^2)$ overwrite (181/30)	Mean norm	.793	[.667, .902]
Same-head donor-norm control (181/30)	Mean norm	.545	[.428, .661]

Zeroing is detectable in the prespecified low-norm direction. Fresh $\mathcal{N}(0, 0.02^2)$ replacement is also mismatched to the current unregistered-row distribution: over 1,000 replacements its median AUC is 0.868 (2.5–97.5 percentiles 0.803–0.923). As a diagnostic, sampling replacement norms from unregistered rows in the same hash head reduces the first seeded AUC to 0.545; over 1,000 draws the median is 0.459 (0.363–0.555). This control isolates the scalar norm channel only: it does not replace full vector directions or sanitize the shared projections. Registry reconstruction therefore removes the tested metadata signal, not membership in an old learned artifact. We require a fresh cartridge trained only on surviving assets before artifact release. These attacks test closed-world membership, not arbitrary RTL reconstruction.

A.10 Retrieval and Per-Secret LoRA Details

Table 4 gives the exact matched 32-task endpoints. Figure 7 adds the task-level outcome pattern and the complete measured storage series; it is not an accuracy ranking.

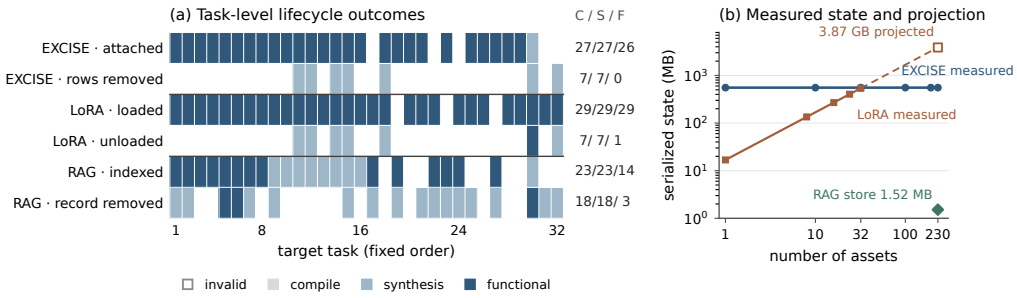


Figure 7: Matched lifecycle outcomes and serialized state. (a) Every column is one shared target task; C/S/F gives compile, synthesis, and strict functional counts. (b) Filled points are measurements. Only the dashed 32-to-230 LoRA segment and open endpoint are projected.

Figure 8 reports the complete CPU retrieval ceiling on all 230 validated designs. TF–IDF top-1 accuracy remains 213/230 under the hardest masked-prose query, while BM25 reaches 223/230; top-5 recovery is 227/230 and 230/230, respectively. Removing a target record from either index reduces correct-target retrieval to 0/230. The payload and gzip-pickled TF–IDF index occupy 1,515,601 bytes in total, and mean TF–IDF query latency is 0.64–1.11 ms. The retrieved payload is the reference RTL, so this is a removable-memory ceiling rather than evidence of independently generated code.

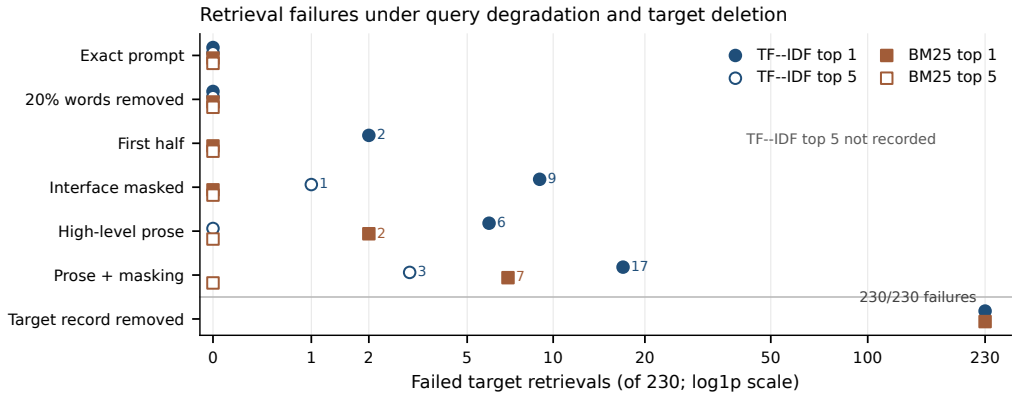


Figure 8: Retrieval failures under deterministic query degradation and exact record deletion. Points report $230 -$ correct on a $\log(1 + x)$ axis; filled/hollow marks denote top-1/top-5. Both indexes recover every exact query, while target removal causes 230/230 measured top-1 failures. The TF–IDF prefix-half top-5 endpoint was not recorded.

We also ran an in-house prompt-concatenation pilot on the first 32 clean targets using the hardest masked-prose query. Intact TF–IDF retrieval finds the target for 29/32 queries, but the base generator produces only 10.1 tokens on average and reaches 2/32 compilation, 2/32 synthesis, and 1/32 established functional recovery. After target-index deletion, retrieval is correct for 0/32; generation averages 118.8 tokens and reaches 11/32 compilation, 11/32 synthesis, and 1/32 functional recovery. This inversion coincides with immediate termination after the intact prompt embeds a complete RTL module. We therefore use direct payload return as the matched retrieval endpoint for that initial implementation.

An engineered rerun uses the unmasked request, a reference–request template, a forced `module` output prefix, and a 1,024-token retrieved-context cap. The intact TF–IDF index retrieves all 32 targets; generation compiles and synthesizes $23/32$, establishes $14/32$ functions, and produces 11 normalized exact matches. Removing only the target record gives $0/32$ correct retrieval, $18/32$ compile and synthesis, and $3/32$ functional recovery. Functionality has 12 paired losses and one gain (exact McNemar $p = 0.00342$). The three post-deletion recoveries are therefore a frozen-generator residual, not retained target retrieval. Both conditions share the model, decoding, template, and hardware scorer; this is the generated-output RAG baseline used in Figure 7.

Generated-RAG capability audit. We applied the same engineered template and 1,024-token context cap to all 156 VerilogEval-v2 tasks, using unconditional top-1 TF–IDF retrieval from the 230-record store and no benchmark-specific threshold. RAG passes 65/156 versus 70/156 base, with 15 paired gains and 20 losses (exact McNemar $p = 0.5$); its Wilson 95% interval is 34.2–49.5%. Thus retrieved context preserves the frozen generator's aggregate benchmark capability within sampling resolution, although it changes which individual tasks pass.

A deterministic Okapi BM25 audit Robertson & Zaragoza (2009) retrieves 230/230 exact, every-fifth-word, prefix-half, and interface-masked queries at top 1. High-level prose reaches 228/230, and the hardest masked-prose query reaches 223/230 at top 1 and 230/230 at top 5; target-index deletion gives 0/230. This sparse baseline uses a local deterministic implementation of the established BM25 algorithm.

The completed matched-exposure LoRA sweep covers all 32 assets in the seed-0 mutual-isolation suite. Each rank-8 adapter receives 32 presentations of its asset, matching the requested per-asset count in the EXCISE capacity protocol. The frozen base compiles and synthesizes 7/32 and establishes one functional recovery. Loaded adapters compile and synthesize 29/32, establish 29/32 functional recoveries, and reach 28/32 normalized exact and 25/32 whitespace-stripped exact matches. Relative to base, functional recovery has 28 paired gains and no losses (exact McNemar $p = 7.45 \times 10^{-9}$). Disabling the adapter reproduces all 32 paired base responses byte-for-byte and therefore all base hardware outcomes. The selected-layer EXCISE seed-0 checkpoint establishes 26/32 intact functions and 0/32 after serving deletion under canonical Icarus-12 rescoring. On the

same task IDs, LoRA has six functional gains and three losses relative to EXCISE (exact McNemar $p = 0.508$); 23 tasks succeed under both and none under neither.

All 32 matching adapters lower their own target-answer NLL. Across the 992 off-diagonal adapter/asset pairs, 714 raise NLL and 278 lower it; the mean signed change is $+0.203$ (565 increases and 145 decreases exceed 0.1). Thus external adapter selection is both an isolation mechanism and a possible misrouting failure mode. Each serialized adapter is 16.82 MB; the 32 measured files total 538.28 MB and take 89.7 seconds to train sequentially on one H100. A linear 230-adapter storage projection is 3.87 GB, versus 554.74 MB for the shared 230-asset EXCISE checkpoint and 1.52 MB for retrieval payload plus index. Because each adapter is stored and selected per asset, its manifest/file association explicitly reveals membership. These are serialized artifact sizes, not serving VRAM or end-to-end cost, and the 230-adapter projection is not a measured system.

A complete 992-pair wrong-adapter generation audit replaces the teacher-forced NLL proxy with decoded RTL. Only 80/992 byte-identical responses match the paired frozen-base output. At the functional endpoint, 919/992 functionally unchanged pairs remain, while 45 gains and 28 losses produce 73 label changes (exact McNemar $p = 0.0604$). The imbalance between gains and losses is not significant, but the observed transitions show that adapter ID remains part of the trusted serving configuration.

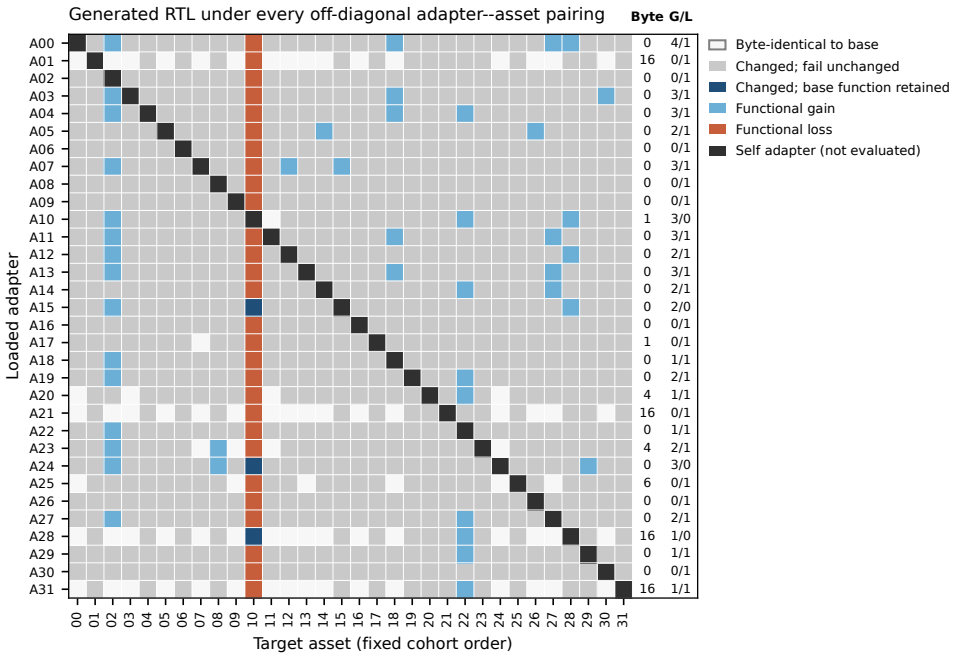


Figure 9: Complete generated-output wrong-adapter audit. Rows identify the loaded adapter and columns the requested asset; the diagonal is excluded. Cells compare each wrong-adapter response with its paired frozen-base response. The right margin reports byte-identical outputs and functional gains/losses per loaded adapter.

Figure 10 gives the completed capability audit for all 32 matched adapters on the same 156 VerilogEval tasks. Across the 32 loaded adapters, the range is 0–77/156, the mean is 41.0 passes (median 42.5), 31/32 score below the 70-pass base, and one scores above. Adapter 4 reaches 77/156 through 18 paired gains and 11 losses, whereas adapter 2 loses all 70 base successes and gains none. Thus the transition map exposes heterogeneous task changes rather than a uniform score shift. Exact paired McNemar tests are computed per adapter; 24/32 remain significant after Holm correction across the 32 comparisons. The adapters share tasks, so the pooled 221 gains and 1,150 losses are descriptive rather than independent task-population replicates.

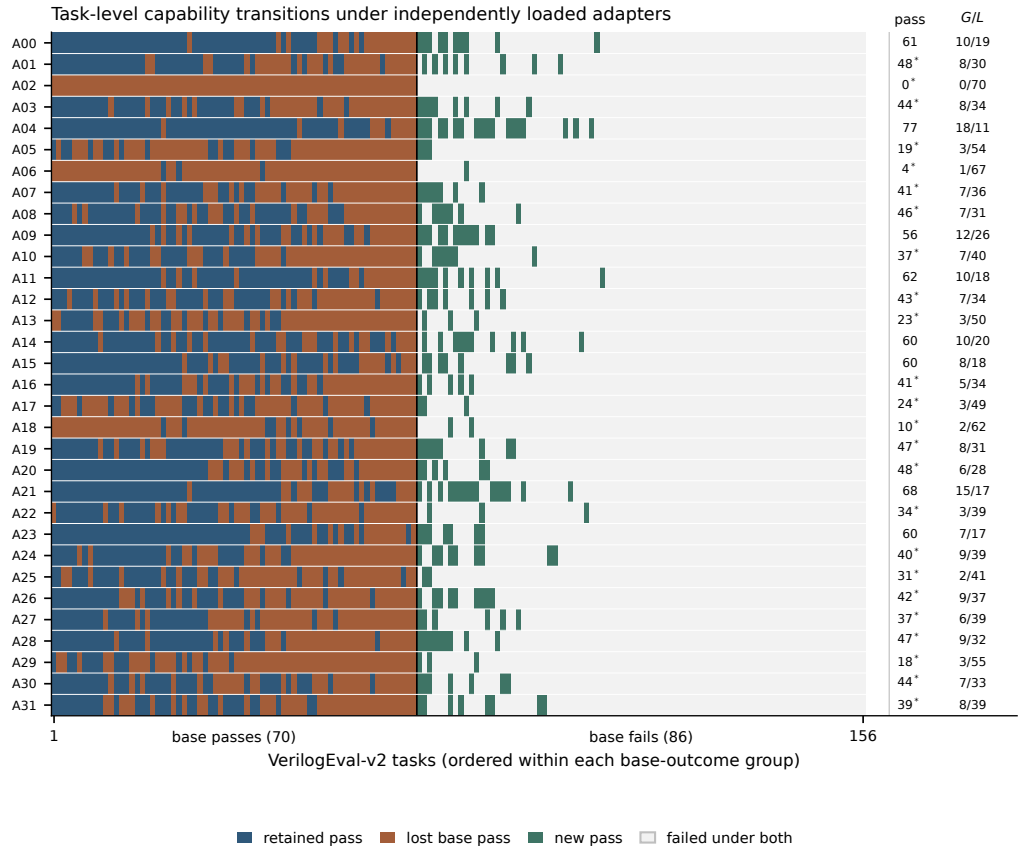


Figure 10: Complete 32-adapter capability audit. Columns are the 156 shared VerilogEval-v2 tasks, partitioned by frozen-base outcome and ordered within each partition by the number of adapters passing. The right margin reports total passes and paired gains/losses (G/L); an asterisk marks a paired McNemar difference that remains significant after Holm correction over all 32 adapters.

A separate eight-adapter run used 64 rather than 32 presentations per asset. It gives 7/8 exact and functional recoveries, 8/8 base-identical unloads, and 52/56 off-diagonal NLL increases (mean $+0.419$); its differently exposed capability scores are historical and are not pooled with the matched sweep.

A.11 Insertion-Layer Sensitivity

We test whether a cartridge behaves similarly when inserted at different transformer depths. For every tier, we train otherwise matched cartridges at layers $2/8/16/31$ for 1,000 steps on the same deterministic 64-asset subset with seed 0, exclusive-only row construction, and $k{=}1$. Layer 31 is the final block of this zero-indexed 32-layer backbone. The dominant-row sets are identical across layers within a tier. Routing-off NLL is invariant within each tier, and routing-off generations for T1, T3, and T4 are byte-identical across all four layers for all 64 prompts.

Table 18 shows that insertion depth materially changes asset recovery, but no layer dominates across tiers and metrics. For T2, layer 8 raises functional recovery from 9/64 at layer 2 to 24/64 (18 paired gains and three losses; exact McNemar $p = 0.00149$), while layer 16 reaches 20/64. Layers 8 and 16 yield neighboring totals and balanced paired transitions for T3 (60/64 versus 61/64; three gains and two losses; $p = 1.0$) and T4 (51/64 versus 52/64; five gains and four losses; $p = 1.0$); these tests do not establish equivalence. For T1, the all-length score is inflated by short binary keys, so the table also reports the pre-specified length-at-least-16 subset and matched wrong-key control. Layers

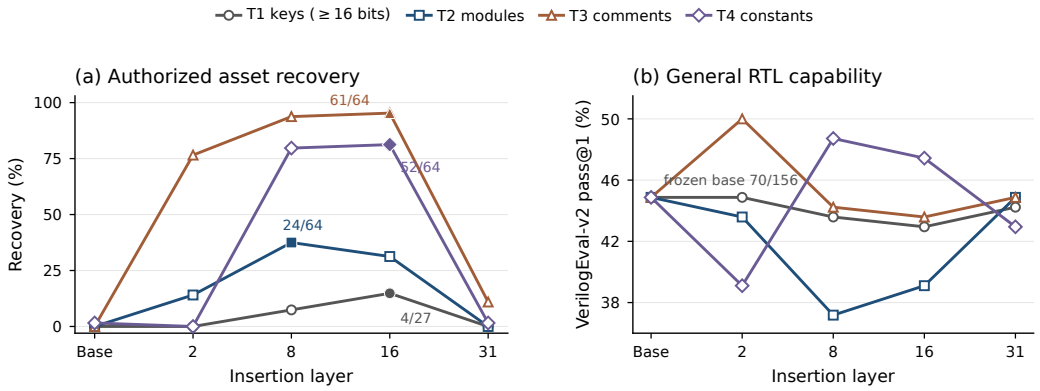


Figure 11: Insertion-depth trade-off. (a) Authorized recovery uses long-key strict exactness for T1 (27 keys), functional recovery for T2, and strict exact recovery for T3/T4 (64 assets each); labels mark each tier's best tested layer. (b) VerilogEval-v2 pass@1 on the same 156 tasks. Leftmost points are routing-off/base; the remaining points are layers 2, 8, 16, and 31. All checkpoints share one backbone, seed 0, and 1,000 training steps.

$2/8/16/31$ recover $0/2/4/0$ of 27 long keys and zero matched wrong keys; their mean true-minus-wrong character agreement is $.011/.082/.181/-.011$.

Table 18: Asset recovery by insertion layer on 64 assets per tier. T1, T3, and T4 use strict exact recovery; T2 uses functional recovery. The T1 long-key matched control is reported immediately above.

Tier	Layer	Intact NLL	Recovery / 64	VE / 156
T1	2	.604	9	70
T1	8	.325	21	68
T1	16	.280	**25**	67
T1	31	1.178	5	69
T2	2	.117	9	68
T2	8	.093	**24**	58
T2	16	.083	20	61
T2	31	.536	0	70
T3	2	.088	49	78
T3	8	.044	60	69
T3	16	.015	**61**	68
T3	31	.642	7	70
T4	2	1.620	0	61
T4	8	.077	51	**76**
T4	16	.055	**52**	74
T4	31	.836	1	67

The capability column is a full paired evaluation against the common frozen base (70/156). Recall and capability do not induce a common layer ordering. T2 layer 8 has the highest tested functional recovery, but passes 58/156 capability tasks, compared with 68/156 at layer 2; their direct comparison has 13 gains and 23 losses ($p = 0.132$). Capability differences between layers 8 and 16 are likewise unresolved at this cohort size for T2 (20 gains and 17 losses; $p = 0.743$), T3 (five gains and six losses; $p = 1.0$), and T4 (five gains and seven losses; $p = 0.774$). T4's two middle layers improve both endpoints relative to layer 2, whereas T3's middle-layer recall gain accompanies lower capability. Insertion depth is therefore a joint recall–capability hyperparameter rather than a tier-independent constant.

These comparisons cover one 6.7B backbone, one subset per tier, and four of 32 blocks. They establish layer sensitivity and broad middle-layer plateaus for T3/T4 rather than a universal optimum or monotonic depth law. Moreover, the 256-step pilot and 1,000-step run select different T2/T4 NLL minima, so insertion depth interacts with optimization budget.

A.12 Configuration-Selection Holdout

We split the canonical 192-task T2 cohort before this confirmation. The first 64 tasks form a configuration-selection cohort; the remaining 128 form the evaluation cohort. At fixed k=1, the locked rule retains layers with at most 10 fewer VerilogEval passes than the $70/156$ base and then maximizes functional recovery, breaking ties toward the smaller layer index. Selection counts for layers $2/8/16/31$ are $9/24/20/0$ functional and $68/58/61/70$ VerilogEval, respectively, so the rule selects layer 16. The task lists, checkpoint seed, training budget, evaluator, and decision rule were hashed before training or generation on the 128 evaluation tasks.

Table 19: Locked layer-16 evaluation on 128 tasks excluded from configuration selection. Functional is exact RTL or strict formal equivalence.

Condition	Compile	Synthesize	Functional	VerilogEval
Frozen base	40/128	38/128	3/128	70/156
Intact, $k = 1$	95/128	86/128	27/128	66/156
Routing-off, $k = 1$	40/128	38/128	3/128	70/156[a]

[a]Routing-off target generations are byte-identical to base for all 128 tasks; routing-off capability is the corresponding frozen-base computation and is not regenerated.

Intact versus base has 26 functional gains and two losses (two-sided exact McNemar $p = 3.03 \times 10^{-6}$). Its VerilogEval comparison has nine gains and 13 losses ($p = 0.523$). Re-scoring all 384 saved target generations with Icarus 12 reproduces every compile, synthesis, and functional count. This is a task-level confirmation of the selection rule on the same backbone and asset source, not an independent-backbone replication.

A.13 Dose Response and Capacity Boundary

The fixed-budget dose experiment holds 1,024 optimization steps, batch size, sequence length, cartridge construction, and total secret presentations constant. Only the allocation between 32 evaluated targets and 32 fixed distractors changes. Figure 12a–b reports all seven preregistered, independently re-scored points. D=0 is a distractor-trained cartridge control, not an untrained base model.

Mean target NLL decreases at every dose, while compilation and synthesis increase from $6/5$ at D=0 to $25/25$ at D=32. Functional recovery is non-monotone at the lowest doses but rises from $0/32$ at D=4 to $6/32$ at D=16 and $20/32$ at D=32. Serving deletion at D=16 leaves $1/32$ functional generation, so greater exposure improves authorized recall without making revocation perfect. These seven checkpoints share one training seed; the curve is descriptive rather than seed-level inference.

The completed capacity grid consists of separately trained fixed-construction runs and is not the mutual-isolation or deletion-semantics archive above (Figure 12c–d). N=32 and N=100 evaluate the same first 32 targets: functional recovery changes from 20 to 24, while VerilogEval collapses from 70/156 to 12/156. At N=192, established recovery returns to 20/32 while VerilogEval only partially recovers to 23/156. At N=100, construction still retains 13,221/13,881 candidate rows (95.25%), with 10 zero-row assets. Aggregate row-pool exhaustion therefore does not explain the observed general-task interference.

A label-free replay of the five reconstructed checkpoints on the same 156 VerilogEval prompts characterizes prompt-side intervention at this boundary (Table 20). Mean firing tokens per prompt increase from 2.72 at N=1 to 39.23 at N=192, and the fraction of prompt tokens routed through the cartridge increases from 0.77% to 11.13%. The mean injection norm over all prompt tokens grows from 0.079 to 2.686, while the raw sigmoid gate on firing tokens remains high (0.999 to 0.884). Every prompt fires at least once even at N=1, so sequence-level activation alone is uninformative. These measurements establish an expanding prompt-side intervention surface, consistent with general-task

interference. They do not make any one statistic causal: VerilogEval rebounds from 12 to 23 passes between N=100 and N=192 even as firing incidence and injection magnitude continue to rise. The grid thus locates an operating boundary, not a universal 100-asset limit or monotone capacity law.

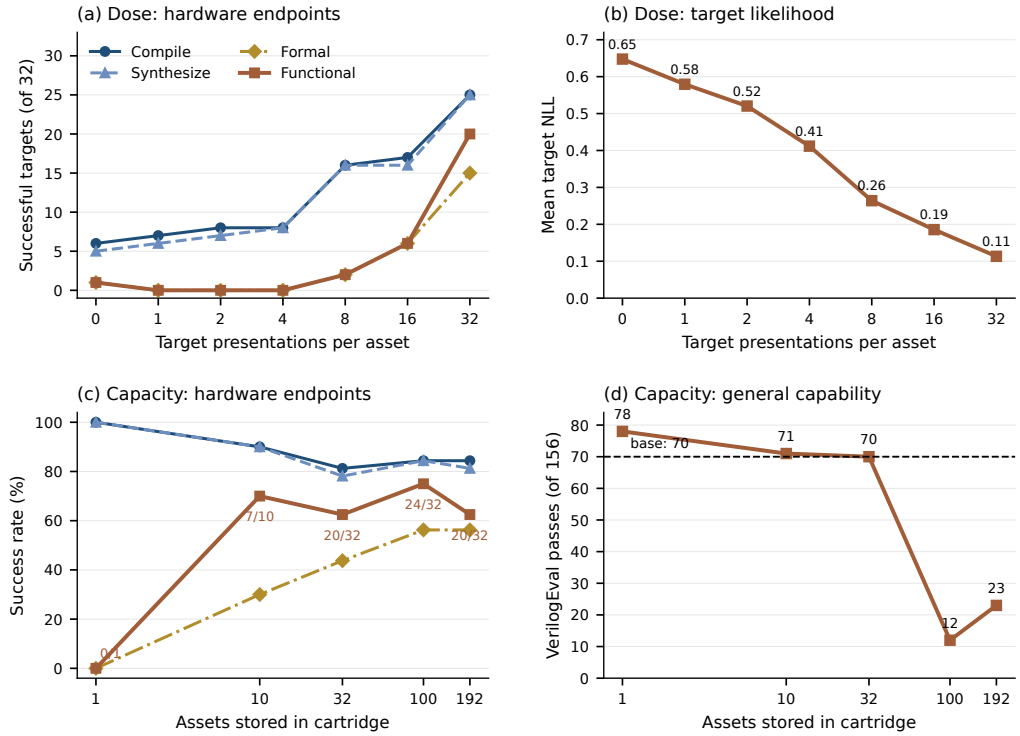


Figure 12: Dose and capacity audits. (a–b) Across 32 targets, more presentations improve compile, synthesis, and functional recovery while reducing target NLL at every measured dose. (c–d) Capacity stores $N = 1, 10, 32, 100, 192$ assets; the last three points share 32 targets and the first two use matched smaller denominators. Labels give functional numerator/denominator; the measured VerilogEval rebound is not smoothed.

The same jobs record process wall time and serialized state. Training the 192-asset cartridge takes 1,482 seconds (24.7 minutes) on one H100. The fixed 1.60M-row table dominates checkpoint size, which grows only from 554.60 to 554.72 MB; the explicit registry grows from 4.2 to 209.6 kB. These measurements exclude model download, environment setup, generation, and hardware verification.

Table 20: Capacity outcomes, prompt-side intervention, and resources. Fired tokens, token activation, raw gate, and injection norm are measured on the exact 156 VerilogEval prompts without using pass/fail labels. Gate is averaged over firing tokens; injection norm is averaged over all prompt tokens.

Assets	Functional	VE	Fired/prompt	Active tok. (%)	Gate	Inj. ℓ_2/tok.	Train (s)	Registry (kB)
1	0/1	78/156	2.72	0.77	0.999	0.079	21	4.2
10	7/10	71/156	4.74	1.35	0.986	0.135	90	17.7
32	20/32	70/156	8.62	2.45	0.998	0.415	257	43.6
100	24/32	12/156	20.45	5.80	0.921	1.422	777	112.3
192	20/32	23/156	39.23	11.13	0.884	2.686	1,482	209.6

Checkpoints occupy 554.60/554.61/554.62/554.66/554.72 MB for N=1/10/32/100/192. Wall time covers cartridge training only; sizes are on disk, not peak serving VRAM.

For context, training the eight rank-8 LoRA adapters took 79.7 seconds in total. We separately isolate EXCISE serving overhead with fixed 128-token continuations on eight prompts, two full

warm-up blocks, six measured repeats, and rotating condition order. Aggregate throughput is 85.46 tokens/s for the frozen base, 81.20 tokens/s with the cartridge loaded but routing disabled (-4.99%), and 80.20 tokens/s intact (-6.16%). The measured peak-allocation ranges overlap at the reported precision (13,550.5–13,667.1 MiB in every condition). This batch-one H100 audit excludes model loading, orchestration, and concurrent production traffic; it measures the loaded cartridge's per-token serving path rather than an end-to-end deployment.

A.14 Historical Threshold Diagnostic

We retain the complete exploratory threshold sweep to expose the operating range rather than only favorable points. It is a diagnostic, not an input to the functional claims above. Figure 13 shows every evaluated threshold rather than only favorable points. Its character-overlap axis predates the asset-specific scoring corrections: T1 is chance-like for balanced bits, T2 can reward fragments without functional equivalence, and T3 mixes identifying fields with generic comments.

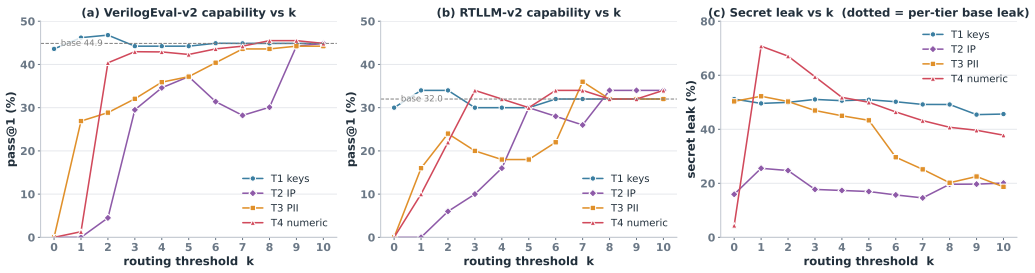


Figure 13: Historical full-threshold diagnostic. All evaluated k values are shown. The capability curves remain useful for locating routing interference, whereas character overlap is retained only to document the exploratory path that motivated the strict T1 and hardware-level T2 rescoring. No claim in the paper relies on these overlap values.

A.15 Reproducibility Details

Deployment hygiene. Row construction is an admission check: an asset with $|\mathcal{X}_s| = 0$ must use a separate cartridge, adapter, or retrieval record if independent revocation is required. We retain these assets in evaluation to expose the failure rate. A serving deletion must propagate the new mask to every replica and invalidate in-flight KV caches created under the old mask. If whole-response base identity is required, the operator unloads the complete cartridge and necessarily suspends all other assets.

Hash construction. The tokenizer IDs are mapped using bigram and trigram contexts. Each order has eight hash heads. A NumPy `default_rng(7)` draws one odd 64-bit multiplier per position. For each context, the multiplied token IDs are XOR-reduced; head j takes the result modulo its prime-sized range and adds a disjoint cumulative offset. The 16 ranges use successive unused primes beginning at 100,003 and form a 1,601,826-row table. The embedding width per head is 80 ($d_{\text{mem}} = 1280$ over 16 heads) and its entries are initialized from $\mathcal{N}(0, 0.02^2)$. The key and value projections, including their bias terms, and their RMSNorm layers use the PyTorch defaults saved in every checkpoint.

Optimization and sequences. Unless an experiment states otherwise, the CodeV-DS backbone is loaded in bfloat16 and frozen, while the cartridge is optimized in float32 with AdamW for 1,000 steps at learning rate 2×10^{-3}, default $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, no scheduler, and no gradient clipping. PyTorch and NumPy use seed 0; the three-seed audit changes both sampling seeds to 0, 1, and 2 while retaining hash seed 7. Each principal T2 batch contains one complete designated carrier and one retain document, independently padded or truncated to 1,024 tokens. Cross-entropy covers the non-padding next-token positions. The ablation margin adds $0.5[3 - \mathcal{L}_{\text{abl}}]_+$ over the designated answer-token positions. The 2,000 retain documents participate both in row exclusion and in the training batches; no gate-sparsity penalty is used.

Generation. All reported task outputs use greedy decoding (`do_sample=False`) with repetition penalty 1.1. T2 uses at most 1,024 new tokens and otherwise allocates the tokenized reference length plus 64. The threshold frontier uses left-padded batches of four; the free-generation cross-deletion audit uses batch size one and first verifies all intact outputs byte-for-byte against the archived checkpoint outputs. VerilogEval-v2 uses the official 156 prompts and testbenches with greedy pass@1 generation and the same repetition penalty.

Hardware scoring. Syntax uses Icarus Verilog 12.0 with `-g2012`. Synthesis uses Yosys 0.33+6 (git `31ee566ec`) with an explicit top, `proc`, `flatten`, `opt`, and `check -assert`. Before equivalence, the scorer rejects candidates with an incompatible interface or any undriven output bit. It then runs `equiv_make`, `equiv_simple`, and `equiv_status -assert`; failures receive a second attempt with `equiv_induct -seq 8`. Per-example JSONL records retain the raw response, extracted RTL, tool outcome, error summary, interface match, structural statistics, and equivalence basis.

Artifact-membership audit. The registry-free audit uses the 32-asset checkpoint, all 192 candidate carriers, and the same 2,000-document retain corpus. It recomputes retain-exclusive rows at sequence length 1,024 and prespecifies their mean embedding norm as the primary score. Eleven candidates (two members) have no scoreable row, leaving 181 candidates and 30 members for AUC and prevalence-matched top-k evaluation. AUC intervals use 10,000 stratified bootstrap replicates with seed 7. The $\mathcal{N}(0, 0.02^2)$ and same-head donor controls each use 1,000 overwrite draws with seeds 20260927 and 20260928, respectively. The audit first asserts that the 5,151 registry rows exactly match the checkpoint's stored dominant set; Table 17 reports all prespecified and diagnostic endpoints.

Software and artifacts. The canonical runs use Python with PyTorch 2.5.1, Transformers 4.57.1, NumPy 2.2.6, and SymPy 1.13.1. Every completed run stores the command-level manifest, task IDs and content hashes, per-example generations and scores, checkpoint digest, and SHA-256 checksums. We will release the training and evaluation code, immutable split manifests, per-example outputs, and cartridge checkpoints subject to the upstream model and dataset licenses; no proprietary RTL is required to reproduce the public proxy experiments.