# PRECISION AUDITS FOR COMPILED TRANSFORMERS WITH SOURCE-LEVEL GROUND TRUTH

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Compiled transformers provide source-level ground truth for a numerical/compiler audit: does reduced-precision execution still compute the intended program? In CRAFT, an LU-triangular-solve compiler, a certified-exact lookup oracle reveals that low precision can preserve finite outputs and hard attention while fetching the wrong source value. Across 142 executing programs and 359,202 emitted rows, no row fails above 17 tested arithmetic bits. Full bf16 breaks 105/142 programs; protecting the address path at every tested width of 18 bits or above restores selector correctness while leaving value error. The arithmetic-only sufficient rule is approximately $2\log_2 P + 4.4$ bits; storage rounding adds a separate term. A microbenchmark reaches $P = 131{,}071$ and shows two additional bits per address doubling. Binary codes and direct gather delimit this representation-specific effect. Native Apple-GPU fp32-address/bf16-value runs also restore selectors on three frontier programs. In a public Tracr case, official ACDC recovers the full known graph despite int8 errors on 232/256 inputs; additional ACDC and trained controls remain correct. These results support source-level numerical auditing of compiled artifacts, without establishing universal fragility, compression benefits, or generalization to trained transformers.

## 1 INTRODUCTION

A recurring claim in mechanistic interpretability is that a network "implements" a particular algorithm. The evidence is often indirect: attention distributions that are effectively one-hot, low attention entropy, outputs that remain finite and plausible, predictions that survive pruning or quantization, and ablation studies showing that particular heads are critical (Clark et al., 2019; Voita et al., 2019; Elhage et al., 2021; Wang et al., 2023). These signals are useful, but they are still proxies. To measure them, we need a system whose intended computation is known independently of the network's observed behaviour.

Compiled transformers supply part of that ground truth. Tracr (Lindner et al., 2023) compiles RASP programs (Weiss et al., 2021) into weights; ALTA (Shaw et al., 2025) and looped constructions (Giannou et al., 2023) extend the compilation target; InterpBench (Gupta et al., 2024) and MIB (Mueller et al., 2025) use known-circuit models to score interpretability methods. These systems provide symbolic targets. We ask a complementary numerical question: which deployed arithmetic preserves the target, and where should a compiler spend extra precision?

CRAFT supplies that test case. It compiles the forward and backward substitution of LU-based matrix inversion into transformer weights at binary64, for resistive-netlist Laplacian systems whose source-driven solution is a node-voltage readout. Because the program is emitted rather than learned, three things are known before any execution: which dimension holds which intermediate quantity, which attention head is supposed to fetch from which position, and what the exact arithmetic result should be. Perturbing the execution through dtype, quantization, pruning or head ablation moves the program away from correct execution in a measurable way, while the behavioural proxies can be read off exactly as a practitioner would read them.

This paper is therefore a numerical/compiler audit, not a benchmark race and not a claim that CRAFT is a useful linear solver. It studies the following question:



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.

> When a numerical program is compiled into transformer weights with ground-truth operation labels available, how many bits are needed to preserve emitted attention addresses, and which diagnostics still look convincing when the address computation has already broken?

Table 1 states the boundary of the claim. The central claim is a source-level precision audit for CRAFT's quadratic address gadget. The Tracr, ALTA and trained-model controls test which parts of the auditing discipline transfer beyond that gadget.

Table 1: Claim boundary. The main claim is a CRAFT address-precision audit; the bridge experiments test how far the audit discipline transfers.

| Claim type | What the paper establishes | Boundary |
|---|---|---|
| Main precision result | CRAFT's quadratic addressing is exact when the significand budget satisfies $2\log_2 P + 4.4$ up to constants; the full emitted LU-row audit finds no row wrong above 17 tested bits, and a wide-$P$ source-operation check follows the same two-bits-per-doubling scaling. | This bit law applies to the CRAFT address encoding, not to all compiled transformers or to binary/direct-gather encodings. |
| Semantic artifact check | Reloaded frontier artifacts reproduce the 16/17/18-bit transition; the fp32-address/bf16-value checkpoint restores selector correctness on the last-failing circuit. | This demonstrates semantic materialization only. It is not a compression result, and for the highlighted case it is larger than uniform fp32. |
| Proxy limitation | CRAFT finiteness, near-hard attention, pruning invariance, ablation criticality and ACDC-style recovery can stay reassuring after source-labelled lookup failure. | These are failure-mode audits on compiled artifacts, not high-powered statistical comparisons. |
| Structural-sparsity correction | All-parameter pruning appears harmless because emitted tensors are mostly structural zeros; pruning nonzero coefficients reverses the conclusion. | Pruning claims must state the denominator: all entries, nonzero entries, or source operations. |
| Scope controls | Official ACDC on a public Tracr case shows a topology/function split after int8; ALTA and trained SIIT probes show other source operations can behave differently. | They support the auditing methodology, not the CRAFT-specific bit budget; the trained probes are negative for the headline dissociation. |

The contributions are:

1. We give a certified address-precision audit for an emitted transformer program, including a 359,202-row replay, a 142-program address-island sweep, and a wide-$P$ address microbenchmark that reaches $P = 131{,}071$.
2. We show that common diagnostics can miss the resulting source-level address failures, and we correct a pruning interpretation trap by separating structural zeros from real coefficient removal.
3. We compare compiler alternatives for the same source lookup contract: binary position codes trade width for bf16 stability, and direct gather removes softmax addressing when runtime integer addresses are available.
4. We use Tracr, ALTA and trained controls to mark the boundary: topology recovery can dissociate from function in a public compiled circuit, but the same failure does not appear in the two trained SIIT probes we test.

CRAFT is not meant to be a useful solver. It is about five orders of magnitude slower than a library call, and its compiler accepts only coefficients known at build time. The point of the construction is checkability: every emitted lookup and every reported numerical value has an independent reference.

## 2 Related work

**Compiled and constructive transformers.** RASP (Weiss et al., 2021) and Tracr (Lindner et al., 2023) compile symbolic programs into transformer weights; ALTA (Shaw et al., 2025) targets a lower-level assembly; Giannou et al. (2023) build programmable looped transformers. Expressivity results establish what hard-attention transformers can represent (Pérez et al., 2019; 2021; Hahn,

2020; Barceló et al., 2024; Merrill & Sabharwal, 2023b;a), surveyed by Strobl et al. (2024). Merrill and Sabharwal's log-precision models already make bit complexity explicit for formal-language expressivity. Our question is narrower and numerical: for the quadratic addressing gadget emitted by a compiler, how many significand bits are sufficient for the exact hard argmax, and what happens when deployed formats fall below that budget?

**Interpretability evaluation with known ground truth.** InterpBench (Gupta et al., 2024) and MIB (Mueller et al., 2025) evaluate circuit discovery against known structure; ACDC (Conmy et al., 2023) and causal scrubbing (Chan et al., 2022) are the methods being evaluated. Miller et al. (2024) show circuit-faithfulness metrics are sensitive to methodological choices. Our axis is complementary: correctness of recovered topology does not certify numerical execution of the recovered program.

**Numerical analysis.** The backward error we measure is the classical normwise quantity (Rigal & Gaches, 1967; Higham, 2002); the LU measurements calibrate execution against existing numerical theory. Verified numerics (Rump, 2010; Ogita et al., 2005) and machine-checked floating-point proofs (Boldo & Melquiond, 2011; Daumas & Melquiond, 2010; Roux, 2016; Kellison et al., 2023; Kwan & Hunt, 2024) provide stronger solver guarantees. FPTuner and Daisy already combine precision allocation and rewriting (Chiang et al., 2017; Darulova et al., 2018), and hard-to-soft attention simulation uses score gaps and temperature (Yang et al., 2026). Our contribution is source-labelled failure and repair evidence within compiled transformer artifacts, not a new general numerical verifier.

**Quantization and pruning.** We use per-tensor and per-channel int8 (Jacob et al., 2018; Krishnamoorthi, 2018; Nagel et al., 2021; Dettmers et al., 2022), bfloat16 (Kalamkar et al., 2019; Burgess et al., 2019), and magnitude pruning (Han et al., 2015; 2016; Frankle & Carbin, 2019). Compiler clear-key sentinels are structural masks; unless explicitly stress-tested, they are preserved and excluded from quantization and pruning scale statistics.

## 3 THE TESTBED AND ITS ORACLE

**What is compiled.** The host computes the Doolittle factors $\hat{L}, \hat{U}$ of $A_{FF}$ once at build time and bakes them into token embeddings. The transformer executes forward and backward substitution token by token. A runner threads values between steps through the KV cache, supplying $b_i$ at the $\texttt{rhs}_i$ token and $y_i$ at $\texttt{bck}_i$. We do not claim the factorization is executed by the network.

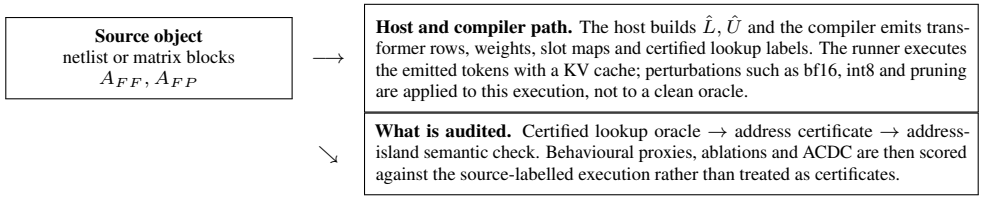


Figure 1: Workflow and claim boundary for the CRAFT audit. The figure separates host computation, compiler emission, runner-managed execution, and the certified lookup oracle used to judge precision checks and proxy diagnostics.

**Scope.** The compiler accepts only coefficients known at build time. No compiled program here can express pivoting, input-dependent control flow, or input-dependent coefficients, and each matrix is compiled separately. This is a hard boundary of the present system. In this paper, "executing a numerical routine" means executing the compiled triangular solves for a fixed matrix.

**Direct-read cases.** Of 154 netlists in our suite, the target node is a *fixed* source node in 12. For those the compiler emits a direct source read: the substitution machinery is compiled and scheduled but never exercised, and 88 free-variable solve steps are built and never run. Those 12 attain zero downstream error without executing a solve. We report every execution result on the 142 that do execute. The direct-read cases are useful as a warning: an application-level metric can pass even when the intended computation never runs.

**Experimental populations.** We use *netlist* for the 154 source systems and *program* for an executing compiled LU solve. Figure 1 shows the host/compiler/runner split, and Table 2 maps the populations used in the main text. Smaller sweeps in the appendix reuse these populations unless explicitly labelled.

Table 2: Population map. Each row supports a different claim; denominators are not interchangeable.

| Population | Size | Claim supported |
|---|---|---|
| Source netlists | 154 | Defines the suite; 12 are direct-read cases that pass without executing a solve. |
| Executing LU programs | 142 | Full-population address-island precision audit and 359,202-row emitted-lookup audit. |
| Coarse diagnostic grid | 30 | Proxy false-greens and perturbation failure modes; nine labels collapse to six selector-hit signatures. |
| Run-time RHS stress | 40 | Confirms the compiled program responds to run-time right-hand sides. |
| Head-ablation subset | 8 | Active-head and lookup-role ablation audit: 174 active heads and 798 unused heads. |
| Jacobi family | 20 | Second compiled numerical family; execution-structure check, not the LU bit-budget claim. |
| Tracr/ALTA/trained controls | bounded external tasks | Tests whether source-level auditing, rather than the CRAFT address-gadget threshold, transfers beyond CRAFT. |

**The oracle.** Let $s(p) = \langle k_p, q_t \rangle$ be the attention score at position $p$. The position the program *must* fetch is $\arg\max_p s(p)$ evaluated exactly. Computing that in rational arithmetic at every lookup is expensive, so we certify instead: if the float64 winning margin exceeds a bound on the accumulated rounding error of a two-term dot product, the computed $\arg\max$ is provably the exact one; otherwise we recompute that row with `Fraction`. Over $104{,}381$ lookup decisions on compiler-labelled lookup heads, $99.1\%$ were certified directly ($98.14\%$ mean per program, minimum $95.7\%$) and $927$ required the exact fallback.

This avoids a circular definition of correctness. We do not ask whether a perturbed model matches the float64 run. We ask whether it fetches the position specified by the compiled program.

**Head labels.** The compiler assigns each attention head a role: *lookup* heads implement a scheduled fetch, *passthrough* heads carry slot contents forward using an erase query. We read these labels from the compiler's allocation record. The distinction matters. If lookup and passthrough heads are pooled, exact one-hotness is unattainable even on correct runs (it caps at $0.22$). All proxy measurements below are therefore computed on lookup heads.

## 4 The compiled program executes run-time inputs

If the compiled model only ever ran on one input, its attention pattern would say nothing about computation. We therefore drive each program with six families of right-hand side, **with no recompilation between them**: columns of the identity, the physical source-driven $b$, i.i.d. Gaussian, wide-dynamic-range vectors spanning $10^{-8}$ to $10^{8}$, the all-ones forcing vector solved in-model, and a vector aligned with the smallest singular direction of $A$.

Over 40 matrices and $1{,}163$ solves, the maximum per-column normwise backward error is $1.88 \times 10^{-16} \approx 1.7u$ and the selector hit rate against the oracle is exactly $1.000$ in every family; median $\eta$ ranges from $1.19 \times 10^{-17}$ to $4.16 \times 10^{-17}$ with no family separating from the others (Appendix E). The program computes a genuine function of its input, at the accuracy classical theory predicts for a triangular solve with a growth factor near 1 (Higham, 2002).

## 5 Failure and address precision on the stress-test grid

**Design.** 30 compiled programs ($n \in [3, 44]$ unknowns) crossed with 9 execution configurations: dtype {f64, f32, bf16, f16}; quantization {per-tensor int8, per-channel int8}; and magnitude pruning at $\{10, 50, 90\}\%$ over *nonzero* coefficients. All-coefficient pruning is excluded from the primary grid because the compiled weights are already $96.7$–$99.8\%$ zero; pruning $10$–$90\%$ of all coefficients is a bit-identical no-op in all 33 released LU models. A run is *structurally broken* when any compiler-labelled lookup fetches a position other than the oracle's. We report backward error and output

finiteness separately: selector correctness is the compiler-ground-truth execution property, while numerical agreement can also fail if a perturbation changes arithmetic coefficients without changing the fetched positions. The configuration class is not an independent treatment sample: the nine labels collapse to six selector-hit signatures. Table 3 reports configuration-level outcomes, and Appendix M tests within-family variation near the pruning boundary.

Table 3: Ground truth by execution configuration in the coarse LU grid (30 programs each). A configuration is broken when any lookup deviates from the oracle. Clear-key sentinels are preserved; all-coefficient pruning is a verified no-op and is not counted as a treatment. $\eta = \infty$ denotes a non-finite solve or residual.

| Config | frac. broken | med. hit | med. $\eta$ | Config | frac. broken | med. hit | med. $\eta$ |
|---|---|---|---|---|---|---|---|
| f64 | 0.00 | 1.000 | $3.9\times10^{-17}$ | per-tensor int8 | 1.00 | 0.628 | 1.00 |
| f32 | 0.00 | 1.000 | $1.3\times10^{-8}$ | per-channel int8 | 1.00 | 0.266 | 1.00 |
| bf16 | 1.00 | 0.915 | $3.9\times10^{-2}$ | prune 10%(nz) | 1.00 | 0.130 | 1.00 |
| f16 | 1.00 | 0.628 | $\infty$ | prune 50%(nz) | 1.00 | 0.330 | 1.00 |
| N/A | N/A | N/A | N/A | prune 90%(nz) | 1.00 | 0.628 | 1.00 |

Three caveats make this a failure-mode audit rather than a high-power comparison. First, every coarse-grid configuration is either correct on all 30 programs or broken on all 30. Second, median hit 0.628 is the trivial modal-oracle baseline: f16, per-tensor int8 and 90% nonzero-pruning reach it by collapsing to the most frequent address. Third, those three labels share the same selector-hit signature, leaving five broken signatures overall.

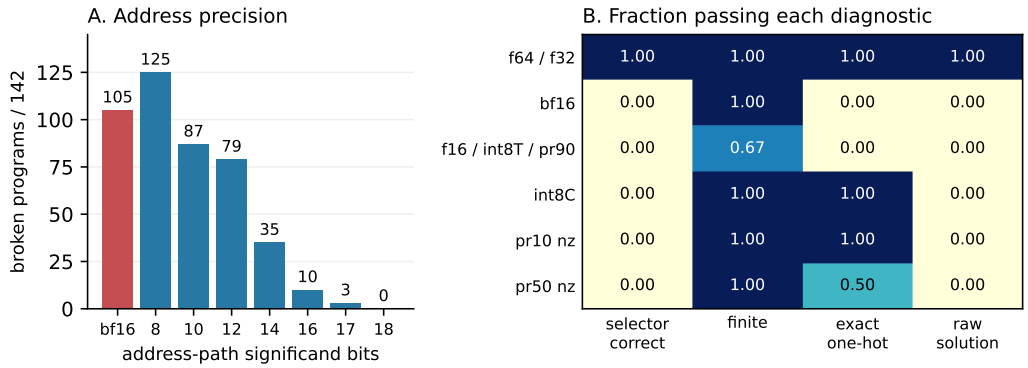


Figure 2: Address semantics and proxy diagnostics. (A) Full bf16 breaks 105/142 programs; tested address islands at 18 bits and above restore selectors. (B) Several broken selector-hit signatures retain finite outputs or exact one-hot attention. Cells show pass fractions within each grouped signature, not independent estimates of failure prevalence.

**Worked address precision audit.** The bf16 row is the useful stress case: it is a real deployment dtype, its outputs remain finite, and its selector failures are predicted by the precision certificate in §6. We therefore test the direct compiler policy suggested by the certificate. In *address-island bf16*, values, residual streams, FFNs and the output head are executed in bf16, but the address-producing stream is kept in an island rounded to $p$ binary significand bits (including the hidden bit), using round-to-nearest-even and unbounded exponents; the scores select bf16 values. This is a two-dtype simulation, not a hardware $p$-bit format. The audit enumerates the full cached executing LU-direct population: 142 non-direct-read programs, with 36 checkpoints cached from earlier audits, 106 newly built, and zero build failures.

Table 4: Address-island precision sweep on all 142 executing LU programs. Values and FFNs remain bf16; only the address-producing stream and score arithmetic receive the listed significand width. $\Delta x_\infty$ is max-relative solution drift from the f64 run. The last selector failure is at 17 bits; the listed tested islands at 18 bits and above are selector-correct.

| Variant | broken | med. hit | min hit | med. $\Delta x_\infty$ | max $\Delta x_\infty$ |
|---|---|---|---|---|---|
| full f64 | 0/142 | 1.000 | 1.000 | 0 | 0 |
| full bf16 | 105/142 | 0.938 | 0.813 | 3.48 | $8.08\times10^{3}$ |
| island 8 bits | 125/142 | 0.929 | 0.807 | 2.18 | $3.33\times10^{3}$ |
| island 10 bits | 87/142 | 0.974 | 0.823 | $4.86\times10^{-1}$ | $1.43\times10^{3}$ |
| island 12 bits | 79/142 | 0.996 | 0.923 | $3.88\times10^{-2}$ | $5.49\times10^{3}$ |
| island 14 bits | 35/142 | 1.000 | 0.936 | $4.52\times10^{-3}$ | 77.1 |
| island 16 bits | 10/142 | 1.000 | 0.986 | $4.26\times10^{-3}$ | 1.01 |
| island 17 bits | 3/142 | 1.000 | 0.998 | $4.26\times10^{-3}$ | $3.34\times10^{-1}$ |
| tested islands $\geq$ 18 bits | 0/142 | 1.000 | 1.000 | $4.26\times10^{-3}$ | $2.66\times10^{-2}$ |

Table 5: Row-level address precision audit on all 359,202 emitted LU lookup rows. This a posteriori certificate replays the emitted score arithmetic at every integer score width 8–28, plus 32 and 53. No emitted row is wrong at any tested score width $\geq$ 18 bits; the lemma column is a prior sufficient ceiling, not the deployed budget. An independent NumPy `frexp`/`ldexp` replay matches every row threshold and bin maximum.

| Live prefix | rows | circuits | max bad bits | sufficient bits |
|---|---|---|---|---|
| 1–8 | 27,744 | 142 | 0 | 10.0 |
| 9–16 | 27,471 | 142 | 8 | 12.2 |
| 17–32 | 51,138 | 125 | 10 | 14.3 |
| 33–64 | 90,409 | 87 | 13 | 16.4 |
| 65–128 | 107,821 | 55 | 15 | 18.4 |
| 129–236 | 54,619 | 15 | 17 | 20.2 |

Tables 4–5 change the interpretation of the negative grid. The dominant bf16 failure is wrong addressing, not unusable bf16 values. The complete-program address transition matches the row-level score-arithmetic boundary; in task-level terms, it reduces median solution drift from $3.48$ in full bf16 to $4.26\times10^{-3}$ at every tested selector-correct island, with worst-case drift $2.66\times10^{-2}$. This does not restore $10^{-6}$ numerical agreement because the value path remains bf16; it restores the discrete lookup semantics. The 8-bit island is worse than full bf16 because it rounds stored keys, current queries, products and running score sums to an explicit 8-bit score path; full bf16 uses bf16 storage but wider accumulation in the baseline implementation. From 10 bits onward the explicit score-path budget improves the selector trace monotonically except for value-path drift, which is not a selector metric. The row audit checks that boundary across five live prefix ranges rather than only as one pooled threshold. The observed row envelope is shallower than the worst-case lemma, so the practical compiler-facing certificate is the a posteriori row replay in Table 5; Lemma 1 is a prior sufficient envelope, not a tight empirical law or an extrapolation rule. An independent NumPy replay reimplements p-bit rounding and agrees on all 359,202 row thresholds and all prefix-bin maxima. These rows are clustered within programs, not independent statistical samples. A compiler that targets low-precision execution should therefore certify its emitted rows under the intended score arithmetic and preserve an address precision island when that certificate fails; it should not merely report that attention remains hard. Two artifact checks separate threshold from deployment. First, Appendix D materializes p-bit address checkpoints for the three full-population frontier circuits: after reload and without source-model access, 17 bits fails in 3/3 circuits and 18 bits succeeds in 3/3. These prototypes store the rounded address recurrence in fp32 containers, so they are semantic threshold artifacts, not storage optimizations. Second, because 18 bits is not a commodity dtype, we export a commodity-format semantic check as an actual checkpoint with two streams: an fp32 address recurrence and score accumulator, and a bf16 value stream that supplies selected values, value-stream residual/FFN updates, reported solutions and output logits. On CKT_0133 it reloads with no source-model access, restores selector hit from $0.828$ to $1.000$, and reduces solution drift from $8.87\times10^{2}$ to $9.61\times10^{-3}$. Its 38.9 MB tensor state exceeds the 25.9 MB uniform-fp32 baseline, so the claim is semantic executability of the certified policy, not compression or a deployment recommendation.

**Proxies.** We score: output finiteness; a hard-attention proxy (lookup-row softmax exactly one-hot and Shannon entropy $\approx 0$, with non-finite rows red); and raw-solution invariance $\max_i |x_i - x_i^{\text{f64}}| / \max_i |x_i^{\text{f64}}| < 10^{-6}$. The older 50 mV target-node check is kept only as a legacy audit (Appendix G).

Table 6: Proxy diagnosticity against the compiler oracle, with selector-hit signatures as the primary unit. The nine labels collapse to one correct signature and five broken signatures. † is numerical equivalence, not a structural oracle.

| Selector-hit signature | classes | structural | finite outputs | hard-attn proxy | raw solution† |
|---|---|---|---|---|---|
| f64 = f32 | 2 | correct | yes | yes | yes |
| bf16 | 1 | broken | yes | no | no |
| f16 = per-tensor int8 = 90% nonzero-pruning | 3 | broken | 2/3 classes | no | no |
| per-channel int8 | 1 | broken | yes | yes | no |
| 10% nonzero-pruning | 1 | broken | yes | yes | no |
| 50% nonzero-pruning | 1 | broken | yes | 15/30 programs | no |

Table 6 groups observations by selector-hit signature. Finiteness stays green in six of seven broken configurations. Hard attention stays green on all 30 broken per-channel-int8 and 10% nonzero-pruning runs, and 15/30 at 50% pruning. Raw-solution invariance catches all broken signatures here, but storage noise produces false greens and mantissa rounding changes solutions even with correct selectors. The grid is a failure-mode audit, not an estimate from independent program–configuration samples.

At $K = 10^{10}$, finite hard-attention rows are one-hot by construction, so the proxy is a compiler-labelled ceiling. Wrong fetches need not collapse to one position: per-channel int8 and 10% pruning have median top-position fractions 0.280 and 0.076. Ablation-selected heads recover lookup roles with precision only 0.47–0.48 and still give one-hot false greens on 0.29–0.32 of broken observations (Appendix K).

Appendix K runs stronger interventions on the same 30 programs. All-head source patching restores 30/30 f64 programs but 0/30 false-green int8 or pruning runs; greedy ACDC-style head-unit recovery restores f64 but either rejects the broken runs or passes an internal metric while source correctness remains 0/30. The strongest non-CRAFT check uses the official circuits-benchmark ACDC implementation on public Tracr case 47, not our emitter or head-unit recovery. Table 7 reports the headline rows: ACDC recovers the complete known graph with zero false-positive edges over $\tau = 10^{-4}$–0.05, yet per-channel int8 computes the wrong symbolic function on 232/256 inputs. Case 61 repeats the pattern with 172/256 int8 errors (Appendix R).

Table 7: Official circuits-benchmark ACDC bridge on public Tracr case 47. The graph labels and model are released benchmark artifacts, not CRAFT outputs.

| Perturbation | wrong/256 | token acc. | finite | full-GT/zero-FP thresholds |
|---|---|---|---|---|
| f64 | 0 | 1.000 | yes | $10^{-4}$–0.05 |
| bf16 | 0 | 1.000 | yes | $10^{-4}$–0.05 |
| per-channel int8 | 232 | 0.601 | yes | $10^{-4}$–0.05 |
| 10% nonzero pruning | 256 | 0.000 | yes | none |

**Finer controls.** Storage-coordinate noise produces a graded transition from no breaks through $\epsilon_s = 3 \times 10^{-6}$ to 557/600 at $10^{-3}$, with finiteness and hard attention green on every broken run. Nonzero-pruning rises from 0/30 broken at 0.2% to 30/30 at 5%; stochastic mantissa rounding fails at 7–9 bits and becomes structurally correct at $\geq 10$ bits. The appendices retain the quantization, by-construction score-inversion, and 20-program Jacobi controls. These are within-design replications, not additional independent failure mechanisms.

**Secondary source-operation check outside CRAFT.** In a compact Tracr backend, source-operation checks localize four packed-int8 failures to decoder MLPs; fp32 QK protection does

not repair them. An independently checked clipped-ReLU rewrite repairs all four, while four other bounded tasks remain correct (Appendix S). Ordinary bf16 already succeeds, and packed files can exceed bf16. ALTA and trained controls remain negative. The audit transfers; the CRAFT addressing mechanism does not.

**Pruning invariance is structural sparsity.** Pruning $10/50/90\%$ of *all* coefficients leaves execution bit-identical ($\eta = 3.9 \times 10^{-17}$, hit rate 1.000). Pruning $10\%$ of the *nonzero* coefficients destroys it ($\eta = 1.0$, hit rate 0.130). Reports of extreme pruning invariance in compiled models can therefore be measuring how sparse the emitted tensor already is. The distinction costs one line of code and reverses the conclusion.

**Temperature and ablation controls.** Rescaling $K \in \{10^{10}, 10^8, 10^6\}$ leaves the proxy table unchanged. Reducing $K$ eventually breaks execution without a soft-but-functional rescue. Factoring $K$ changes the collapse mode but still breaks all 30 int8 programs. Ablating 972 heads in eight programs identifies activity but not local lookup roles (Appendix K).

## 6 Worked example: CRAFT address precision

This section is a worked example for the CRAFT quadratic address gadget, not a general precision law for compiled transformers. Constructive transformer results are stated over the reals; the following gives a sufficient bit budget for this emitted address template. We write $K$ for the implementation's effective attention scale, so with keys $k_p = (2p,\ -p^2 + \alpha g(p))$ and query $q_t = K(t, 1)$,

$$s(p) = K\big(2pt - p^2 + \alpha g(p)\big) = K\big(t^2 - (t-p)^2 + \alpha g(p)\big),$$

and the $t^2$ term cancels from every pairwise comparison.

**Lemma 1** (Selector exactness)**.** *Let $K > 0$, integer positions $p, t \geq 0$, $0 \leq g < 1/\ln 2$, $0 < \alpha < \ln 2$ and $M = 1 - \alpha/\ln 2$. For $p \neq t$, $s(t) - s(p) \geq KM$. If all positions satisfy $p, t \leq P$ with $P \geq 1$, then, assuming exact input coordinates and the standard relative-error model for the two products and sum (no overflow or underflow), the computed* $\arg\max$ *in a binary floating-point format with unit roundoff $u$ satisfying $0 < u < 1/2$ and largest finite value $\Omega$ equals the exact* $\arg\max$ *whenever*

$$\textit{(precision)} \quad P < \sqrt{\frac{M(1-2u)}{12u}}, \qquad \textit{and} \qquad \textit{(range)} \quad P < \sqrt{\frac{\Omega}{3K}}.$$

*The scale $K$ cancels from the precision condition and appears only in the range condition.*

A proof is in Appendix B. Equivalently, the exact sufficient condition is $u < M/(12P^2 + 2M)$, or about $2\log_2 P + \log_2(12/M) \approx 2\log_2 P + 4.4$ significand bits: 11.0 bits for $P = 10$, 20.2 for the largest emitted-cache position $P = 235$ in the compiled grid, and 31.0 for $P = 10^4$. The emitted lookup template has `KEY_OFFSET`=0, $\alpha = 0.3$, and $g(p) = 1/\ln 2 - 1/\ln(p+2) \in [0, 1/\ln 2)$ in the positional feature `inv_log_pos`; hence all non-cleared parabolic address rows satisfy the lemma's hypotheses by construction. Clear-key sentinel rows are structural masks outside this range model.

For the largest compiled-grid address range $P = 235$, the sufficient rule certifies f64/f32 but not bf16/f16; the latter fail empirically in Table 3; Table 9 in the appendix shows the bound is conservative where precision, rather than range, binds. Appendix C isolates this on the actual compiled address heads by executing the model in float64 while rounding only the attention-score dot products. Sweeping 8–28 significand bits, all 30 programs become structurally correct at 18 bits; the sufficient 20.2-bit rule is conservative by about two bits on this grid.

**Bounded-magnitude re-encoding does not help.** Normalizing positions $p \mapsto p/P$ divides the margin by $P^2$ exactly as it divides score magnitudes, so the same $O(u^{-1/2})$ scaling remains up to constants. At $K = 10^{10}$ the normalized encoding still first fails at position 3435 in float32, 15 in bfloat16 and immediately in float16; the invariant is the scaling law, not bit-identical first-failure positions.

**Wide-$P$ scaling and representation baselines.** Beyond the emitted cache's $P = 235$, we test the quadratic address formula at capacities 16–131,072, checking all competitors for sampled queries. The high-end row $t = P$ needs two more significand bits per doubling, rising from 8 to 34 bits (Figure 3). This is a source-operation microbenchmark, not an expanded end-to-end LU suite.

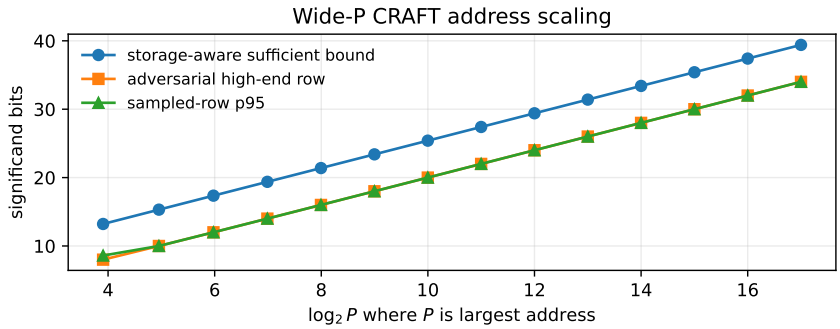


Figure 3: Wide-$P$ source-operation scaling for the CRAFT quadratic address gadget. Storage and arithmetic are both rounded, so the envelope uses the storage-aware $\theta$ bound in Appendix B. Its asymptotic slope is 2 in $\log_2 P$; the adversarial high-end row follows the same two-bits-per-doubling trend but stays about five bits below the sufficient ceiling.

An additional margin stress test separates emitted logarithmic rows from constructed bounded-$g$ rows. At capacity 65,536, unique-argmax thresholds increase from 32 to 41 bits as the constructed margin approaches $10^{-3}$; these are lemma-family controls, not emitted CRAFT programs (Appendix T).

Table 8 locates this effect in the representation. At capacity 1,024, binary bf16 uses wider keys but preserves every selector; quadratic bf16 does not. Direct gather removes softmax addressing and has exact selectors on all three frontier LU circuits with either fp32 or bf16 values. Remaining bf16 error belongs to value arithmetic.

Table 8: Main representation baselines for the source lookup contract. The unique-lookup microbenchmark uses capacity 1,024 and runtime bf16 values (`p48`); the direct-gather LU artifact uses frontier circuits CKT_0131–0133 and reloads without a clean transformer or cached answers (`p54`). Gather uses an integer schedule; its byte counts are medians.

| Setting | format | selector hit | errors | key dims. | bytes |
|---|---|---|---|---|---|
| Quadratic lookup | bf16/bf16 | 55/1024 | 989/1024 | 2 | 4,098 |
| Binary lookup | bf16/bf16 | 1024/1024 | 0/1024 | 10 | 20,482 |
| Quadratic lookup | fp32/bf16 | 1024/1024 | 0/1024 | 2 | 8,196 |
| Direct-gather LU | fp32 values | 1.000 | n/a | – | 36,869 |
| Direct-gather LU | bf16 values | 1.000 | n/a | – | 29,061 |

Lemma 1 motivates the address island in Table 4; value error remains a separate obligation.

## 7 Discussion and Limitations

Source labels let us distinguish fragile addresses, selected-set averages and numerical decoders, then protect or rewrite the operation that fails. Behavioural proxies remain debugging signals rather than certificates. Direct gather is a simple alternative when integer addresses are available; binary codes preserve an attention implementation but need their own certificate. The evidence remains small-scale and mainly compiled: p-bit storage is emulated, the materialized artifacts do not demonstrate compression, and selector repair leaves value error. Negative ACDC and trained controls and the bounded native-GPU check in Appendix T make these limits explicit.

## Reproducibility Statement

The reviewer archive `craft_reviewer_artifact_20260926.zip` combines source, synthetic inputs, cached LU programs, checkpoints, results and figure generators. Its README separates file integrity, stored-result checks and actual experiment regeneration through `revision/reproduce.py`; a Docker recipe pins the CPU audit dependencies. The release checksum and execution reports accompany the archive. Fresh macOS and Linux environments pass the quick regeneration suite and 22 semantic, packaging and serial/parallel tests each. The exhaustive macOS tier regenerates the diagnostic grid, 142-program island sweep and both 359,202-row replays, matching all 13 reference CSVs byte-for-byte. Appendices U and T describe the environments, native Apple-GPU checks and the precise scope of the formal obligations.

## AI Use Statement

An LLM-based coding assistant was used for experiment implementation, execution, tabulation, experiment-design feedback, manuscript editing and bibliography checking. Reproducible checks accompany the derivations and reported runs; Appendix V gives the full disclosure.

## Ethics Statement

This work uses synthetic compiled programs and public benchmark artifacts; it does not involve human-subject data, private data or a deployed user-facing system. The main risk is overinterpreting proxy or circuit-discovery diagnostics, which we mitigate by reporting limitations, negative trained-checkpoint probes and the release scripts needed to audit the claims.

## References

Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=gbrHZq07mq.

Sylvie Boldo and Guillaume Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In *2011 IEEE 20th Symposium on Computer Arithmetic (ARITH)*, pp. 243–252, Tübingen, Germany, July 2011. IEEE. doi: 10.1109/ARITH.2011.40.

Neil Burgess, Jelena Milanovic, Nigel Stephens, Konstantinos Monachopoulos, and David Mansell. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pp. 88–91, 2019. doi: 10.1109/ARITH.2019.00022.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal Scrubbing: a method for rigorously testing interpretability hypotheses. AI Alignment Forum, December 2022. URL https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing. Redwood Research technical report; not peer-reviewed. Posted 3 December 2022; cross-posted to LessWrong.

Wei-Fan Chiang, Mark Baranowski, Ian Briggs, Alexey Solovyev, Ganesh Gopalakrishnan, and Zvonimir Rakamarić. Rigorous floating-point mixed-precision tuning. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, pp. 300–315, 2017. doi: 10.1145/3009837.3009846.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828/.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 16318–16352. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf.

Eva Darulova, Einar Horn, and Saksham Sharma. Sound mixed-precision optimization with rewriting. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 208–219. IEEE, 2018. doi: 10.1109/ICCPS.2018.00028. URL https://malyzajko.github.io/papers/iccps2018.pdf.

Marc Daumas and Guillaume Melquiond. Certification of bounds on expressions involving rounded operators. *ACM Transactions on Mathematical Software*, 37(1):2:1–2:20, January 2010. doi: 10.1145/1644001.1644003. Gappa is available at https://gappa.gitlabpages.inria.fr/.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pp. 337–340. Springer, 2008. doi: 10.1007/978-3-540-78800-3_24. URL https://www.microsoft.com/en-us/research/publication/z3-an-efficient-smt-solver/.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c3ba4962c05c49636d4c6206a97e9c8a-Abstract-Conference.html.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, December 2021. URL https://transformer-circuits.pub/2021/framework/index.html. Published 22 December 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019. OpenReview.net. URL https://openreview.net/forum?id=rJl-b3RcF7.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Rohan Gupta, Iván Arcuschin, Thomas Kwa, and Adrià Garriga-Alonso. InterpBench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 92922–92951. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a8f7d43ae092d9a5295775eb17f3f4f7-Paper-Datasets_and_Benchmarks_Track.pdf.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://aclanthology.org/2020.tacl-1.11/.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28, pp. 1135–1143. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1510.00149.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Number 80 in Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002. ISBN 0-89871-521-0. doi: 10.1137/1.9780898718027.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713, 2018. doi: 10.1109/CVPR.2018.00286.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019. URL https://arxiv.org/abs/1905.12322.

Ariel E. Kellison, Andrew W. Appel, Mohit Tekriwal, and David Bindel. LAProof: A library of formal proofs of accuracy and correctness for linear algebra programs. In *2023 IEEE 30th Symposium on Computer Arithmetic (ARITH)*, pp. 36–43. IEEE, September 2023. doi: 10.1109/ARITH58626.2023.00021.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. URL https://arxiv.org/abs/1806.08342.

Carl Kwan and Warren A. Hunt, Jr. Formalizing the Cholesky factorization theorem. In Yves Bertot, Temur Kutsia, and Michael Norrish (eds.), *15th International Conference on Interactive Theorem Proving (ITP 2024)*, volume 309 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 25:1–25:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-337-9. doi: 10.4230/LIPIcs.ITP.2024.25.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: Compiled transformers as a laboratory for interpretability. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 37876–37899. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/771155abaae744e08576f1f3b4b7ac0d-Paper-Conference.pdf.

William Merrill and Ashish Sabharwal. A logic for expressing log-precision transformers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 52453–52463. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a48e5877c7bf86a513950ab23b360498-Abstract-Conference.html.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2023b. doi: 10.1162/tacl_a_00562. URL https://aclanthology.org/2023.tacl-1.31/.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit faithfulness metrics are not robust. In *First Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2407.08734.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45069–45108. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/mueller25a.html.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. URL https://arxiv.org/abs/2106.08295.

Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, 2005. ISSN 1064-8275. doi: 10.1137/030601818.

Jorge Pérez, Javier Marinković, and Pablo Barceló. On the Turing completeness of modern neural network architectures. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019. URL https://openreview.net/forum?id=HyGBdo0qFm.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021. URL https://jmlr.org/papers/v22/20-302.html.

J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *Journal of the ACM*, 14(3):543–548, July 1967. doi: 10.1145/321406.321416.

Pierre Roux. Formal proofs of rounding error bounds: With application to an automatic positive definiteness check. *Journal of Automated Reasoning*, 57(2):135–156, August 2016. ISSN 0168-7433. doi: 10.1007/s10817-015-9339-z.

Siegfried M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, May 2010. ISSN 0962-4929. doi: 10.1017/S096249291000005X.

Peter Shaw, James Cohan, Jacob Eisenstein, Kenton Lee, Jonathan Berant, and Kristina Toutanova. ALTA: Compiler-based analysis of transformers. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=h751wl9xiR.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. doi: 10.1162/tacl_a_00663. URL https://aclanthology.org/2024.tacl-1.30/.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580/.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations (ICLR)*. OpenReview.net, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/weiss21a.html.

Andy Yang, Lena Strobl, David Chiang, and Dana Angluin. Simulating hard attention using soft attention. *Transactions of the Association for Computational Linguistics*, 14:147–166, 2026. doi: 10.1162/tacl.a.597. URL https://aclanthology.org/2026.tacl-1.8/.

## A COARSE-GRID DIAGNOSTICITY VISUALIZATION

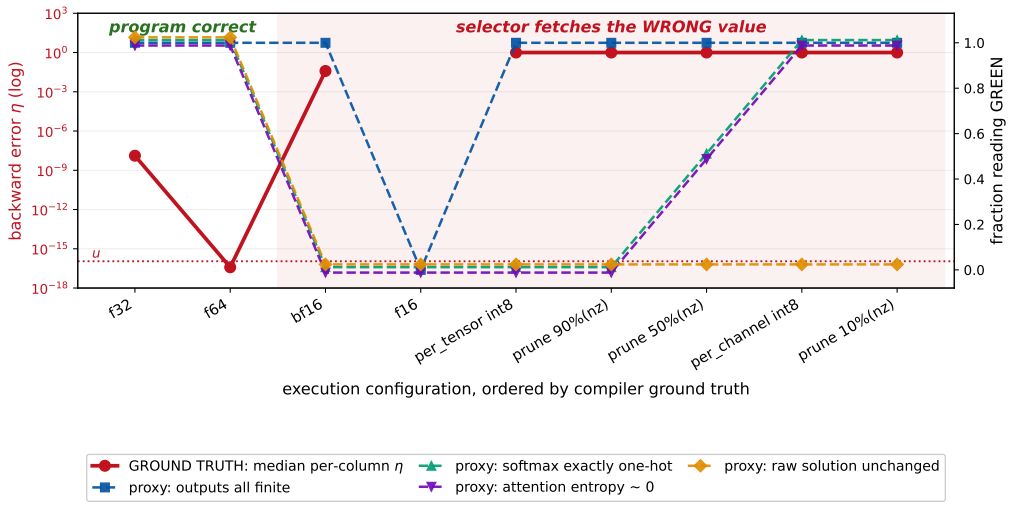


Figure 4: Ground truth (red, left log axis) versus the proxy curves (right axis) across execution configurations ordered by compiler-labelled structural execution. In the shaded region the selector provably fetches wrong values while some internal proxy curves remain high; one-hotness and entropy overlap after non-finite rows are scored red.

## B PROOF OF LEMMA 1

Table 9: Predicted ceilings at $K = 10^{10}$ and simulated first-failure positions for the idealized quadratic gadget. The bound is sufficient; float16 fails the range condition immediately.

| Format | $P_{\text{precision}}$ | $P_{\text{range}}$ | binding | predicted $P$ | first failure |
|---|---|---|---|---|---|
| float64 | $2.06 \times 10^7$ | $7.7 \times 10^{148}$ | precision | $2.06 \times 10^7$ | none to $4 \times 10^6$ |
| float32 | 891 | $1.1 \times 10^{14}$ | precision | 891 | 3831 |
| bfloat16 | 3.46 | $1.1 \times 10^{14}$ | precision | 3.46 | 15 |
| float16 | 9.83 | 0.0015 | range | violated at $P = 1$ | 1 |

**Exact margin.** For $p \neq t$, $s(t) - s(p) = K\left[(t-p)^2 + \alpha\big(g(t) - g(p)\big)\right]$. Since $p \neq t$ are integers, $(t-p)^2 \geq 1$; since $0 \leq g < 1/\ln 2$, $|g(t) - g(p)| < 1/\ln 2$. Hence $s(t) - s(p) \geq K(1 - \alpha/\ln 2) = KM > 0$ for $\alpha < \ln 2$. Numerically, with $\alpha = 0.3$ and the emitted $g(p) = 1/\ln 2 - 1/\ln(p+2)$ we have $M = 0.5672$, and the minimum exact gap measured over $t \in [1, 3 \times 10^5]$ is 0.9433, so the bound holds with room.

**Rounding.** For the compiled address heads, non-address coordinates have zero query/key product by construction, so each relevant score is a two-term dot product. If an implementation mixes additional nonzero coordinates into the address score, replace $\gamma_2$ below by the appropriate $\gamma_m$. Under the standard model $\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta)$, $|\delta| \leq u$, two products and one addition give $|\mathrm{fl}(s(p)) - s(p)| \leq \gamma_2 S_p$ with $\gamma_2 = 2u/(1-2u)$ and $S_p = K\big(|2pt| + |-p^2 + \alpha g(p)|\big)$, the absolute-value sum that bounds cancellation. For integer $0 \leq p, t \leq P$ and $P \geq 1$, $S_p \leq 3KP^2$: if $p \geq 1$, then $\alpha g(p) < 1 \leq p^2$; if $p = 0$, then $S_p < K$.

If the key and query coordinates are stored before the dot product in a lower precision format, with coordinatewise relative error at most $\epsilon_s$, then each exact product is perturbed by at most $\delta_s = 2\epsilon_s + \epsilon_s^2$ before arithmetic rounding. Combining this storage perturbation with the dot-product rounding gives

$$|\mathrm{fl}(\hat{s}(p)) - s(p)| \leq \left[\gamma_2 + (1+\gamma_2)\delta_s\right] S_p = \theta S_p.$$

Thus the storage-aware statement is obtained by replacing $\gamma_2$ with $\theta$. When $\epsilon_s = 0$, it reduces to the arithmetic-only bound above; when $\epsilon_s \approx u$, it changes only the constant and not the $O(u^{-1/2})$ scaling.

**Precision condition.** The computed $\arg\max$ is exact if the exact margin exceeds the worst combined perturbation of the two scores compared: $KM > \gamma_2(S_t + S_p)$, which is implied by $M > 6\gamma_2 P^2$. With stored low-precision keys/queries, the corresponding sufficient condition is $M > 6\theta P^2$. Substituting $\gamma_2 = 2u/(1-2u)$ with $0 < u < 1/2$ and $\epsilon_s = 0$, the arithmetic-only condition is $M > 12uP^2/(1-2u)$, i.e. $P < \sqrt{M(1-2u)/(12u)}$. The factor $K$ appears on both sides and cancels. □

**Range condition.** Representability requires $\max_p |s(p)| \le 3KP^2 \le \Omega$, giving $P < \sqrt{\Omega/(3K)}$, which does depend on $K$. The valid range is the minimum of the two conditions.

## C ADDRESS-SCORE SIGNIFICAND SWEEP ON COMPILED HEADS

To separate addressing precision from arithmetic-coefficient precision, we ran the emitted LU models in float64 while rounding only the attention-score dot products before softmax. Values, residual streams and arithmetic coefficients remain float64; the structural label is still the certified lookup oracle. For each dot product, operands, products and running sums are rounded to a chosen binary significand width. This tests the actual compiled address heads rather than only the idealized two-coordinate simulation in Table 9.

| Score bits | programs | structurally broken | median hit | min hit |
|---|---|---|---|---|
| 8 | 30 | 30 | 0.909 | 0.823 |
| 9 | 30 | 27 | 0.931 | 0.840 |
| 10 | 30 | 24 | 0.944 | 0.866 |
| 11 | 30 | 23 | 0.971 | 0.905 |
| 12 | 30 | 22 | 0.985 | 0.931 |
| 13 | 30 | 15 | 0.998 | 0.928 |
| 14 | 30 | 10 | 1.000 | 0.936 |
| 15 | 30 | 6 | 1.000 | 0.965 |
| 16 | 30 | 3 | 1.000 | 0.986 |
| 17 | 30 | 1 | 1.000 | 0.998 |
| 18–28 | 330 | 0 | 1.000 | 1.000 |

The empirical transition is sharper than the sufficient bound. The last failing case is `CKT_0133` at 17 bits; from 18 bits through 28 bits every compiled-grid program has selector hit 1.000. Thus the sufficient 20.2-bit rule is conservative, but by a measured constant on the actual compiled heads rather than only on the idealized gadget.

The program-level sweep above still summarizes complete executions. To check that the boundary is not a single-range coincidence, `p40` records every compiler-labelled lookup row in the full 142-program executing LU cache and recomputes that row's score argmax under p-bit arithmetic. Across 359,202 emitted lookup rows, the float64 argmax matches the certified oracle on every row and 99.03% are certified without exact fallback. The largest live prefix contains 236 stored keys, so the lemma's sufficient ceiling is 20.2 bits at the largest emitted address position $P = 235$. The observed row envelope is:

| Prefix range | rows | circuits | max bad bits | sufficient bits at max prefix |
|---|---|---|---|---|
| 1–8 | 27,744 | 142 | 0 | 10.0 |
| 9–16 | 27,471 | 142 | 8 | 12.2 |
| 17–32 | 51,138 | 125 | 10 | 14.3 |
| 33–64 | 90,409 | 87 | 13 | 16.4 |
| 65–128 | 107,821 | 55 | 15 | 18.4 |
| 129–236 | 54,619 | 15 | 17 | 20.2 |

Thus no emitted lookup row is wrong at any tested score width $\ge 18$ bits. This row-level audit uses the actual emitted keys and queries and validates the transition across many live address lengths rather than only as a complete-program aggregate. To check harness dependence, `p46` repeats the replay with a separate NumPy `frexp`/`ldexp` p-bit implementation that does not import the p26/p40

rounding helper; it matches all 359,202 row thresholds and all prefix-bin maxima. The empirical envelope is visibly shallower than the lemma's worst-case coefficient 2, but the bin count is too small to treat a fitted slope as a law. We therefore use Lemma 1 as a sufficient compiler-side envelope and use this row replay as the deployed certificate for the emitted models.

## D ADDRESS-ISLAND MIXED PRECISION

The script `p37_address_island_precision_sweep.py` composes the address-island repair with the p-bit score arithmetic harness. The low-precision execution supplies bf16 values to the value path, residual stream, FFNs and output head. The address island supplies lookup keys and queries, rounds stored keys, current queries, products and running score sums to the requested significand width, and then selects values from the bf16 value cache. Thus the intervention protects address comparisons by a stated bit budget while leaving the carried numerical values low precision.

| Variant | $n$ | broken | med. hit | min hit | med. $\Delta x_\infty$ |
|---|---|---|---|---|---|
| full bf16 | 142 | 105 | 0.938462 | 0.812870 | 3.480 |
| island 8 bits | 142 | 125 | 0.929383 | 0.806819 | 2.179 |
| island 10 bits | 142 | 87 | 0.974427 | 0.822856 | $4.862\times10^{-1}$ |
| island 12 bits | 142 | 79 | 0.996473 | 0.923395 | $3.884\times10^{-2}$ |
| island 14 bits | 142 | 35 | 1.000000 | 0.935765 | $4.520\times10^{-3}$ |
| island 16 bits | 142 | 10 | 1.000000 | 0.985710 | $4.256\times10^{-3}$ |
| island 17 bits | 142 | 3 | 1.000000 | 0.997972 | $4.256\times10^{-3}$ |
| tested islands $\geq$ 18 bits | 994 | 0 | 1.000000 | 1.000000 | $4.256\times10^{-3}$ |

The full CSVs are `results/p37_address_island_precision_sweep.csv`, `results/p37_address_island_precision_summary.csv` and the matching baseline summaries; the earlier two-row high-precision-island audit remains as `p36`. The collapsed final row pools the seven tested widths $18, 20, 22, 23, 24, 32, 53$ and does not interpolate untested widths. This full-cache population is the 142 executing LU-direct programs; the 30-program diagnostic grid is a deterministic subset. The experiment does not claim that all downstream numerical error disappears: max-relative solution drift is still governed by bf16 value-path arithmetic. Its claim is narrower and compiler-facing: selector semantics, which fail in ordinary bf16 on 105 of 142 executing programs, are restored by preserving the address-producing path at the measured bit budget.

**Materialized p-bit artifacts.** The p37 threshold is materialized by `p43_materialized_pbit_address_artifacts.py`. For the three full-population frontier circuits CKT_0131–CKT_0133, the script rounds the stored address stream itself to a chosen p-bit significand, saves it in an fp32 container alongside a bf16 value stream, reloads the checkpoint, and runs with no source-model access. Score operands, products and running sums are rounded to the same p-bit width. The reloaded artifacts reproduce the simulated boundary: all three frontier circuits fail at both 16 and 17 bits, and all three are selector-correct at 18 bits.

| Artifact | broken | min hit | max $\eta$ | max $\Delta x_\infty$ |
|---|---|---|---|---|
| materialized p16-address/bf16-value | 3/3 | 0.985710 | $1.35\times10^{-1}$ | $9.77\times10^{-1}$ |
| materialized p17-address/bf16-value | 3/3 | 0.997972 | $4.69\times10^{-2}$ | $3.34\times10^{-1}$ |
| materialized p18-address/bf16-value | 0/3 | 1.000000 | $9.22\times10^{-4}$ | $1.73\times10^{-2}$ |

These are threshold artifacts rather than compression artifacts: the prototype duplicates the full address recurrence, so the fp32 container size remains the right storage baseline. In logical packed accounting, p16-address plus bf16-value uses the same tensor storage as uniform fp32, while p17/p18 use $1.03125\times$/$1.0625\times$ uniform-fp32 tensor storage. The point is that the measured 18-bit boundary survives serialization, reload and execution.

**Commodity two-dtype artifact.** Because no commodity 18-bit dtype exists, `p39_serialized_two_dtype_artifact.py` also exports a real checkpoint for the last-failing precision case, CKT_0133. The checkpoint stores an fp32 address recurrence and score accumulator, plus a bf16 value stream whose selected values, value-stream FFN/residual updates and output logits define the reported execution. It is reloaded and run without the source float64

model at inference; its SHA256 begins `f3789010dc977ce0`. Storage is 38.9 MB (25.9 MB fp32 address state plus 12.9 MB bf16 value state). This is smaller than the 51.8 MB f64 source tensors but larger than the relevant 25.9 MB uniform-fp32 tensor baseline, so it should be read as an executable semantic check of the two-stream policy, not as a deployability or compression claim.

| Variant | selector hit | hard-attn | $\eta$ | $\Delta x_\infty$ |
|---|---|---|---|---|
| full bf16 | 0.828480 | 0.830599 | $9.44\times10^{-2}$ | $8.87\times10^{2}$ |
| serialized fp32-address/bf16-value | 1.000000 | 1.000000 | $9.22\times10^{-4}$ | $9.61\times10^{-3}$ |

On the same CPU runner, median wall-clock time over three repeats is 1.93 s for full bf16 and 2.57 s for the serialized commodity artifact ($1.33\times$). This demonstrates that the compiler policy predicted by the bit-budget audit can be materialized as executable two-stream checkpoints.

## E RUN-TIME RIGHT-HAND SIDES

Table 10: 40 matrices, 1,163 solves, no recompilation between right-hand sides. $u = 1.11 \times 10^{-16}$. Selector hit rate is against the certified-exact oracle.

| RHS family | solves | $\max \eta$ | median $\eta$ | min selector hit |
|---|---|---|---|---|
| identity columns $e_j$ | 243 | $1.58 \times 10^{-16}$ | $1.25 \times 10^{-17}$ | 1.000 |
| source-driven $b$ | 40 | $1.88 \times 10^{-16}$ | $4.16 \times 10^{-17}$ | 1.000 |
| i.i.d. Gaussian | 400 | $1.64 \times 10^{-16}$ | $1.44 \times 10^{-17}$ | 1.000 |
| wide dynamic range | 400 | $1.78 \times 10^{-16}$ | $2.11 \times 10^{-17}$ | 1.000 |
| all-ones, in-model | 40 | $1.09 \times 10^{-16}$ | $1.36 \times 10^{-17}$ | 1.000 |
| adversarial | 40 | $1.18 \times 10^{-16}$ | $1.19 \times 10^{-17}$ | 1.000 |

## F SCHEDULE STABILITY

Each circuit was compiled under five MILP configurations (three solver seeds, a reduced time limit, and a different solver backend) and executed on identical inputs. All five produce distinct weight files. For three of four circuits the executed result is bit-identical; for one it differs by 1.84 ulp, tracking a third distinct slot map. The emitted program is not byte-stable across solver configurations, and the executed result is not always bit-stable either.

We also reran the 16/17/18-bit address-island boundary on the three full-population frontier circuits under four valid MILP variants: HiGHS seeds 0/1/7 with one thread and HiGHS seed 0 with four threads (`p55_schedule_boundary_sensitivity.py`). The valid variants produce four distinct weight hashes overall; two frontier circuits also change schedule shape. For every circuit and every valid variant, 16 and 17 address bits remain selector-broken and 18 bits is selector-exact, so the observed threshold span is 0 bits. The deliberately short 10 s MILP configuration fails to produce a feasible compile on these frontier circuits and is recorded as a build failure, not as threshold evidence.

## G DOWNSTREAM METRIC DECOMPOSITION

The application-level readout quantizes to a 50 mV grid. Decomposing the error over all 154 circuits: total 11.84 mV mean; readout quantization 11.84 mV; target-model mismatch 0.025 mV; and the solve itself $1.6 \times 10^{-10}$ mV. Total and quantization error agree to four significant figures, and the uniform-in-bin expectation is 12.5 mV. The metric measures the readout budget, not solver accuracy. For this reason, §5 uses raw solution-vector change as its primary behavioural proxy and keeps the 50 mV prediction only as a legacy application-level check. We also report that 0 of 1437 solution entries are negative, so the pipeline's non-negativity clamp is a no-op on the declared domain.

## H ATTENTION-SCALE SENSITIVITY

To test whether the main proxy rates are artifacts of the chosen hard-attention scale, we rescaled the query block of each fixed compiled schedule before applying each perturbation class and recorded lookup-row maximum probabilities. This changes the effective $K$ in the emitted attention scores without changing the MILP schedule or slot map. The sweep finds no soft-but-functional rescue. For $K \geq 30$, the unperturbed f64/f32 rows are structurally correct, raw-solution stable and still exactly one-hot. At $K = 25$ and $K = 20$, f64 exact one-hotness is no longer bit-exact while all f64/f32 selectors and raw solutions remain correct, but the smallest lookup-row max probability is still above $0.999999999998$. By $K = 10$, f64/f32 selectors are still correct but only $6/60$ raw-solution runs remain unchanged and the minimum max probability is still above $0.999998$. By $K = 5$, f64 is structurally broken; the first f64 rows with max probability below $0.99$ therefore arrive after f64 addressing has already failed.

| $K$ | f64/f32 selector | f64/f32 raw | min f64/f32 max prob. | one-hot/entropy green on broken | raw green on broken |
|---|---|---|---|---|---|
| $10^{10}$ | $60/60$ | $60/60$ | $1.000000000000$ | $0.357/0.357$ | $0.000$ |
| $10^4$ | $60/60$ | $60/60$ | $1.000000000000$ | $0.319/0.329$ | $0.000$ |
| 300 | $60/60$ | $60/60$ | $1.000000000000$ | $0.229/0.267$ | $0.000$ |
| 30 | $60/60$ | $60/60$ | $1.000000000000$ | $0.290/0.290$ | $0.000$ |
| 25 | $60/60$ | $60/60$ | $0.999999999999$ | $0.310/0.310$ | $0.000$ |
| 20 | $60/60$ | $60/60$ | $0.999999999999$ | $0.324/0.324$ | $0.000$ |
| 15 | $60/60$ | $52/60$ | $0.999999998775$ | $0.310/0.338$ | $0.000$ |
| 12 | $60/60$ | $18/60$ | $0.999999914728$ | $0.000/0.348$ | $0.000$ |
| 10 | $60/60$ | $6/60$ | $0.999998331070$ | $0.000/0.348$ | $0.000$ |
| 7 | $60/60$ | $0/60$ | $0.999893903732$ | $0.000/0.246$ | $0.000$ |
| 5 | f64 broken | $0/60$ | $< 0.99$ | $0.000/0.000$ | $0.000$ |

Thus lowering $K$ within the hard-attention regime changes which broken runs look exactly one-hot only near the edge of hardness; it does not produce a regime in which f64 is numerically stable, attention is meaningfully soft, and the broken perturbations still look falsely reassuring. The low-$K$ rows are boundary failures of the hard-attention contract, not a rescue of exact addressing.

## I QUANTIZATION BIT-WIDTH SWEEP

The coarse grid has only two int8 quantization treatments. We therefore swept 30 sentinel-preserving symmetric quantization treatments: per-model, per-tensor and per-channel scales crossed with integer bit widths 3–12, on the same 30 LU programs and certified selector oracle. This is a practitioner-facing perturbation family rather than a margin-designed control.

| Quantization family | structurally broken | hard-attention proxy green | median selector hit | top selected frac. on broken |
|---|---|---|---|---|
| per-model 3–12 bits | $300/300$ | $0/300$ | $0.628$ | $1.000$ |
| per-tensor 3–12 bits | $300/300$ | $0/300$ | $0.628$ | $1.000$ |
| per-channel 3 bits | $30/30$ | $7/30$ | $0.243$ | $0.201$ |
| per-channel 4 bits | $30/30$ | $19/30$ | $0.237$ | $0.198$ |
| per-channel 5 bits | $30/30$ | $28/30$ | $0.242$ | $0.223$ |
| per-channel 6–9 bits | $120/120$ | $120/120$ | $0.252$–$0.277$ | $0.276$–$0.284$ |
| per-channel 10 bits | $30/30$ | $30/30$ | $0.283$ | $0.283$ |
| per-channel 11–12 bits | $60/60$ | $60/60$ | $0.350$–$0.355$ | $0.331$ |

Thus quantization scheme, not only bit width, determines the failure mode; the 10-bit row is not a transition point but a representative false-green per-channel setting. Per-model and per-tensor scaling produce global single-position collapse, whereas per-channel scaling preserves decisive hard attention over a dispersed set of wrong addresses. Raw solution equality is green on $0/900$ broken runs in this sweep; the legacy 50 mV readout is green on $60/900$, matching the coarse-grid warning that behavioural bins can hide structural failure.

## J K-FACTORED PER-TENSOR QUANTIZATION

Ordinary per-tensor quantization is dominated by the attention query block: at the production scale, the median non-sentinel query/non-query maximum ratio in attention in-projection tensors is $7.1 \times 10^9$

(range $7.1\times10^{9}$–$3.1\times10^{11}$). At $K=1$, the median ratio is $0.71$ and the median non-query maximum is larger than the median query maximum. To test whether this dynamic range explains the per-tensor collapse, we compared ordinary per-tensor quantization with two mechanism probes: *K-factored* quantization, which divides query rows by $K$ while choosing and applying the shared scale and then multiplies them back, and *query-preserved* quantization, which leaves query rows unquantized and quantizes the rest of the tensor. Both preserve clear-key sentinels.

| Variant | bits | broken | hard-attn green | med. hit | top selected frac. |
|---|---|---|---|---|---|
| ordinary per-tensor | 8 | 30/30 | 0/30 | 0.628 | 1.000 |
| ordinary per-tensor | 12 | 30/30 | 0/30 | 0.628 | 1.000 |
| K-factored | 8 | 30/30 | 30/30 | 0.123 | 0.074 |
| K-factored | 12 | 30/30 | 30/30 | 0.130 | 0.076 |
| query-preserved | 8 | 30/30 | 30/30 | 0.123 | 0.074 |
| query-preserved | 12 | 30/30 | 30/30 | 0.130 | 0.076 |

Thus the query scale explains the *collapse mode* of ordinary per-tensor quantization: once the query block is factored out or preserved, the failures are dispersed rather than single-position collapse. It does not explain away the structural failure itself, since all diagnostic variants still break all 30 programs and every broken run has finite, one-hot, zero-entropy lookup-head attention.

## K LABEL-FREE AND INTERVENTION AUDITS

The direct ablation counts against compiler labels are:

| Compiler label | heads | 50 mV pred. changes | raw solution changes | breaks program |
|---|---|---|---|---|
| lookup | 86 | 30 (34.9%) | 47 (54.7%) | 47 (54.7%) |
| passthrough | 88 | 24 (27.3%) | 53 (60.2%) | 61 (69.3%) |
| unused | 798 | 0 (0.0%) | 0 (0.0%) | 0 (0.0%) |

The main proxy table scores compiler-labelled lookup heads, which is a privileged ceiling rather than a label-free diagnostic. We therefore recomputed the coarse grid on the same 8 circuits used for head ablation after selecting heads only by observable ablation effects in the unperturbed f64 run. Prediction-changing heads are those whose ablation changes the emitted 50 mV token; solution-changing heads are those whose ablation changes the raw solution by more than $10^{-6}$. Structural brokenness is still evaluated against the compiler oracle.

| Selector | lookup precision | lookup recall | active precision | selected obs. | finite green on broken | one-hot/entropy green on broken |
|---|---|---|---|---|---|---|
| prediction-changing heads | 0.479 | 0.345 | 1.000 | 184 | 0.857 | 0.321/0.321 |
| solution-changing heads | 0.469 | 0.562 | 1.000 | 350 | 0.857 | 0.286/0.286 |

Both selectors recover active heads perfectly on these circuits, but neither recovers lookup roles well: they mix lookup and passthrough heads. The selected head proxy is therefore less false-green than the compiler-labelled lookup-head ceiling, but still not a structural certificate and still not a lookup-role classifier.

**Activation patching.** We next ask whether a corrupted-input intervention agrees with the structural oracle on the main false-green classes. For each program, we run a base RHS and a source RHS, then execute the base RHS while replacing pre-output-projection head outputs with their clean f64 source-RHS values at the same execution event. In f64 this interchange succeeds on 30/30 programs for lookup, active-head and all-head scopes: the patched base execution matches the source-RHS solution. On broken per-channel int8 runs, the same lookup/active/all-head patches change 22/24/24 of 30 solutions but never match the source solution; on $10\%$ nonzero-pruning all three scopes change 0/30 and succeed 0/30. The intervention therefore does not certify these false-green executions. It rejects the simple hypothesis that clean lookup or active-head outputs are sufficient to restore the program, and it shows that the pruning failure has moved outside these attention outputs.

**ACDC-style recovery.** We also run a greedy recovery audit closer to ACDC's faithfulness interface. The unit is a head-output event grouped by head: start from all variant source-RHS head outputs patched into the base-RHS run, and greedily drop heads in recorded execution order, first in chunks of eight and then individually within rejected chunks, while preserving a normalized recovery threshold $\tau$; if the full patch exactly restores the source solution, the recovered subgraph must also restore it. Recovered event-edges are then scored against the compiler's lookup edge set. This is intentionally called ACDC-style rather than full ACDC, because the CRAFT model is not a TransformerLens graph.

| Perturbation | programs | metric pass | correct | edge prec. | random prec. | edge recall | variant recall |
|---|---|---|---|---|---|---|---|
| f64 | 30 | 30/30 | 30/30 | 1.000 | 0.997 | 0.018 | 0.110 |
| per-channel int8 | 30 | 15/30 | 0/30 | 1.000 | 0.997 | 0.0016 | 0.0091 |
| 10% nonzero pruning | 30 | 0/30 | 0/30 | N/A | 0.997 | 0.000 | 0.000 |

The recovered f64 circuits are sparse sufficient circuits, not the full compiler graph: precision is perfect but recall is low. The matched random precision baseline is the expected lookup precision from a uniformly random event set of the same size drawn from the variant candidates; it is already $0.997$ because the candidate set is almost entirely lookup events, so precision carries little information. The pruning precision entry is undefined because the recovered set is empty. The int8 row is the important failure mode: on half the false-green programs, the recovery procedure's own normalized metric passes even though the patched run is still wrong; nonzero pruning instead fails the recovery metric outright.

We sweep the recovery threshold $\tau \in \{0.90, 0.95, 0.99, 0.999, 0.9999\}$ without changing any other setting: f64 remains $30/30$ metric-pass and $30/30$ correct at every threshold, int8 remains $15/30$ metric-pass and $0/30$ correct at every threshold, and pruning remains $0/30$ on both metrics. Thus the int8 gap is not an artifact of choosing $\tau = 0.95$; tightening $\tau$ through $0.9999$ does not separate the faithfulness metric from provably wrong execution in this audit.

## L STORAGE-COORDINATE KEY/QUERY NOISE

For each lookup head, we perturbed the stored two-coordinate key and current two-coordinate query by independent coordinatewise relative noise in $[-\epsilon_s, \epsilon_s]$, leaving passthrough heads and values unchanged. This directly exercises the storage-perturbation term in Lemma 1. The sufficient-bound ratio column is $6\epsilon_s P^2/M$ at the maximum address position seen in the sweep ($P = 235$, $M = 1 - \alpha/\ln 2$); ratios above one mean only that the conservative sufficient guarantee no longer applies.

| $\epsilon_s$ | bound ratio | runs | structurally broken | broken programs | median hit | attention proxies green on broken | raw solution green on broken |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 600 | 0 | 0 | 1.000 | N/A | N/A |
| $10^{-7}$ | 0.058 | 600 | 0 | 0 | 1.000 | N/A | N/A |
| $3\times10^{-7}$ | 0.175 | 600 | 0 | 0 | 1.000 | N/A | N/A |
| $10^{-6}$ | 0.584 | 600 | 0 | 0 | 1.000 | N/A | N/A |
| $3\times10^{-6}$ | 1.75 | 600 | 0 | 0 | 1.000 | N/A | N/A |
| $10^{-5}$ | 5.84 | 600 | 41 | 3 | 1.000 | 1.000 | 0.024 |
| $3\times10^{-5}$ | 17.5 | 600 | 171 | 10 | 1.000 | 1.000 | 0.199 |
| $10^{-4}$ | 58.4 | 600 | 355 | 20 | 0.997 | 1.000 | 0.110 |
| $3\times10^{-4}$ | 175 | 600 | 469 | 24 | 0.976 | 1.000 | 0.051 |
| $10^{-3}$ | 584 | 600 | 557 | 30 | 0.952 | 1.000 | 0.043 |

This gives a non-score-level graded regime: storage errors first affect the largest programs, then progressively cover the suite. The internal attention proxies are uninformative about those structural breaks because every wrong row is still finite, exactly one-hot and zero-entropy. The repeated-seed rates should be read as 30 program clusters crossed with 20 seeds. A program-cluster bootstrap over the five nonzero-break settings gives 95% intervals for the broken fraction of $[0, 0.168]$, $[0.142, 0.442]$, $[0.420, 0.757]$, $[0.633, 0.912]$, and $[0.853, 0.985]$; the first scale involves only three broken programs, so its raw false-green rate is descriptive rather than precise. Conditioning on exactly those three programs (`CKT_0108`, `CKT_0129`, `CKT_0133`) gives $1/41$ raw false greens among broken runs at $\epsilon_s = 10^{-5}$ and $0/60$ at each larger failing scale. The aggregate nonmonotonicity is therefore compositional: the set of newly broken programs changes with $\epsilon_s$.

## M NEAR-BOUNDARY NONZERO PRUNING

The main diagnosticity grid uses deliberately coarse perturbations. As a near-boundary check, we swept smaller nonzero-pruning fractions on the same 30 programs. The structural break label is the same as in §5: any compiler-labelled lookup fetching a non-oracle position.

| Nonzero pruning | programs | structurally broken | median hit | finite green | one-hot green | entropy green | raw solution unchanged | 50 mV unchanged |
|---|---|---|---|---|---|---|---|---|
| 0.05% | 30 | 0 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| 0.10% | 30 | 0 | 1.000 | 1.000 | 1.000 | 1.000 | 0.967 | 1.000 |
| 0.20% | 30 | 0 | 1.000 | 1.000 | 1.000 | 1.000 | 0.600 | 0.833 |
| 0.50% | 30 | 1 | 1.000 | 1.000 | 1.000 | 1.000 | 0.067 | 0.667 |
| 1.00% | 30 | 7 | 1.000 | 1.000 | 1.000 | 1.000 | 0.000 | 0.333 |
| 2.00% | 30 | 24 | 0.144 | 1.000 | 1.000 | 1.000 | 0.000 | 0.067 |
| 5.00% | 30 | 30 | 0.130 | 1.000 | 1.000 | 1.000 | 0.000 | 0.067 |
| 10.0% | 30 | 30 | 0.130 | 1.000 | 1.000 | 1.000 | 0.000 | 0.067 |

This sweep has two roles. First, it shows that the all-or-nothing pattern in the main table is not forced by the experimental apparatus: close to the pruning boundary, different programs fail structurally at different fractions. Second, it shows why we separate structural execution correctness from numerical equality. At $0.20\%$ pruning, all selector fetches are still oracle-correct, yet $12/30$ raw solution vectors and $5/30$ quantized predictions have changed because arithmetic coefficients have been perturbed. The internal attention proxies fail in the complementary direction: even when nonzero-pruning breaks every program, the selector remains decisive and finite, but decisively selects the wrong positions.

## N MARGIN-CALIBRATED ADDRESS-SCORE NOISE

To isolate the addressing mechanism from weight pruning and arithmetic coefficient drift, we injected bounded random noise directly into compiler-labelled lookup scores before softmax. For each lookup row, the amplitude is $\rho$ times the unperturbed winning margin for that row; all other heads are left unchanged. When $\rho \leq 0.5$, the top score cannot be overtaken by a runner-up under this independent bounded perturbation, and the empirical selector hit rate remains exactly $1.000$. Once $\rho > 0.5$, wrong argmaxes appear while softmax remains hard.

| $\rho$ | runs | structurally broken | median hit | attention proxies green on broken | raw solution green on broken | median solution change |
|---|---|---|---|---|---|---|
| 0.00 | 600 | 0 | 1.000 | N/A | N/A | 0 |
| 0.10 | 600 | 0 | 1.000 | N/A | N/A | 0 |
| 0.25 | 600 | 0 | 1.000 | N/A | N/A | 0 |
| 0.50 | 600 | 0 | 1.000 | N/A | N/A | 0 |
| 0.75 | 600 | 600 | 0.816 | 1.000 | 0.137 | $3.83 \times 10^{-1}$ |
| 1.00 | 600 | 600 | 0.690 | 1.000 | 0.033 | $9.36 \times 10^{-1}$ |
| 1.25 | 600 | 600 | 0.609 | 1.000 | 0.017 | $9.92 \times 10^{-1}$ |
| 1.50 | 600 | 600 | 0.552 | 1.000 | 0.010 | 1.00 |
| 2.00 | 600 | 600 | 0.473 | 1.000 | 0.000 | 1.21 |

All five $\rho > 0.5$ settings break all 30 program clusters under every seed (program-cluster bootstrap interval $[1, 1]$ for the broken fraction). Thus this is a margin-threshold control, not an independent estimate of the prevalence of near-boundary failures in the compiled-program population.

This control isolates the score-margin mechanism. The internal attention proxies are perfect on the correct side and remain perfect on the broken side because every perturbed row is still decisive. The failure is not entropy, diffusion, non-finiteness, or a behavioural threshold: it is the wrong address winning a hard maximum.

## O FINE-GRAINED STOCHASTIC MANTISSA ROUNDING

As a second within-family check, we stochastically rounded every finite nonzero compiled parameter to a fixed number of normalized binary significand bits, then executed the rounded model in float64. This isolates finite precision in the compiled constants from exponent range and arithmetic dtype. The table reports integer precision anchors on the same 30 programs as Table 6, with 20 random seeds per bit width and the same certified-exact selector oracle. Fractional non-dyadic stress-test rows are kept in the release tables but are not used for precision claims because they also perturb dyadic gating constants.

| Effective bits | runs | structurally broken | median hit | attention proxies green on broken | raw solution unchanged (all) | 50 mV unchanged (all) | median solution change |
|---|---|---|---|---|---|---|---|
| 7.0 | 600 | 359 | 0.986 | 1.000 | 0.000 | 0.172 | $7.89 \times 10^{-2}$ |
| 8.0 | 600 | 120 | 1.000 | 1.000 | 0.000 | 0.248 | $3.42 \times 10^{-3}$ |
| 9.0 | 600 | 20 | 1.000 | 1.000 | 0.000 | 0.277 | $1.64 \times 10^{-3}$ |
| 10 | 600 | 0 | 1.000 | N/A | 0.000 | 0.338 | $8.41 \times 10^{-4}$ |
| 11 | 600 | 0 | 1.000 | N/A | 0.000 | 0.417 | $3.52 \times 10^{-4}$ |
| 12 | 600 | 0 | 1.000 | N/A | 0.000 | 0.473 | $1.95 \times 10^{-4}$ |
| 13 | 600 | 0 | 1.000 | N/A | 0.000 | 0.557 | $9.35 \times 10^{-5}$ |
| 16 | 600 | 0 | 1.000 | N/A | 0.000 | 0.902 | $1.35 \times 10^{-5}$ |

This is the finest perturbation family in the paper, and it returns the same message as Appendix M: attention can remain perfectly decisive while selecting the wrong positions. For all structurally broken runs at the integer anchors $p \in \{7, 8, 9\}$, every lookup-head output is finite, exactly one-hot and near-zero-entropy. These are 19, 6, and 1 distinct broken programs respectively, crossed with 20 seeds; program-cluster bootstrap intervals for the corresponding broken fractions are $[0.425, 0.765]$, $[0.067, 0.367]$, and $[0, 0.100]$. The behavioural columns point in the opposite direction: rounding the arithmetic constants changes the solution vector by more than $10^{-6}$ in every run, even when the selector trace is structurally correct. Thus the raw-vector check is useful for numerical equivalence, while the oracle is still needed for execution structure.

## P BEHAVIOURAL-PROXY THRESHOLD SENSITIVITY

Figure 5 sweeps the relative-change threshold used by the raw-vector or full-trace behavioural proxy across the coarse LU grid, near-boundary pruning, stochastic mantissa rounding, and Jacobi token traces. At $10^{-6}$, the coarse LU grid has correct/broken green rates $1.000/0.000$, near-boundary pruning has $0.527/0.000$, mantissa rounding has $0.000/0.000$, and the Jacobi token trace has $1.000/0.000$. For mantissa rounding, raising the threshold to $10^{-3}$ gives $0.657/0.001$ and raising it to $10^{-2}$ gives $0.987/0.039$. Thus raw-vector equality is a thresholded numerical-equivalence check, not a structural certificate. The internal one-hot criterion is also threshold-sensitive on the coarse LU grid: changing the required lookup-head one-hot fraction from $1.00$ to $0.95$, $0.90$, and $0.80$ raises broken-side green rates from $0.357$ to $0.386$, $0.452$, and $0.500$, while the correct side remains $1.000$.

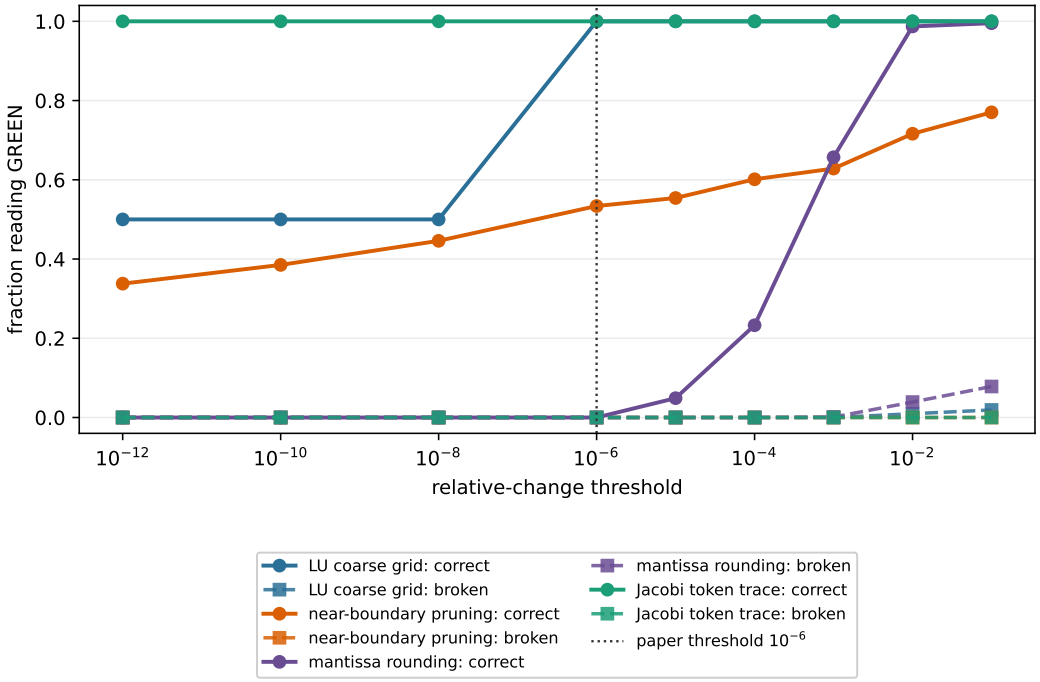


Figure 5: Sensitivity of the behavioural proxy to the relative-change threshold. The LU coarse grid is clean at $10^{-6}$, but finer perturbation families show that this proxy measures numerical equivalence at a chosen tolerance.

## Q FIXED-ITERATION JACOBI SECOND-FAMILY CHECK

We also compiled a second program family: a fixed-$T$ Jacobi update kernel over the same circuit language. For each update token, the active neighbor fetches target known previous-iteration voltage-emission positions; the readout fetch targets the final emission for the requested node. We score only these semantically active lookup heads. The sweep uses 20 programs with 3–15 nodes, $T = 20$, and the same nine execution configurations as Table 6. It is an execution-structure check, not a claim that 20 Jacobi iterations solve all circuits to the application tolerance.

| Proxy | green on correct | green on broken | broken green count |
|---|---|---|---|
| outputs all finite | 1.000 | 0.857 | 120/140 |
| softmax exactly one-hot | 1.000 | 0.286 | 40/140 |
| attention entropy $\approx 0$ | 1.000 | 0.286 | 40/140 |
| full 50 mV token trace unchanged | 1.000 | 0.000 | 0/140 |
| 50 mV final-output bin unchanged | 1.000 | 0.043 | 6/140 |

The class split matches the coarse LU stress grid: two perturbation classes are structurally correct across all 20 programs and seven are broken across all 20. The per-class selector-hit ranges show that most broken Jacobi classes are total or near-total selector destruction rather than graded degradation:

| Jacobi configuration | frac. broken | min hit | median hit | max hit |
|---|---|---|---|---|
| f64 | 0.00 | 1.000 | 1.000 | 1.000 |
| f32 | 0.00 | 1.000 | 1.000 | 1.000 |
| bf16 | 1.00 | 0.115 | 0.216 | 0.358 |
| f16 | 1.00 | 0.000 | 0.000 | 0.000 |
| per-tensor int8 | 1.00 | 0.000 | 0.000 | 0.000 |
| per-channel int8 | 1.00 | 0.000 | 0.000 | 0.012 |
| prune 10%(nz) | 1.00 | 0.000 | 0.000 | 0.000 |
| prune 50%(nz) | 1.00 | 0.000 | 0.000 | 0.000 |
| prune 90%(nz) | 1.00 | 0.000 | 0.000 | 0.000 |

The release archive includes the detailed Jacobi summary table. The useful conclusion is the replication of the false-reassurance pattern, not a proxy ranking: unlike the LU coarse grid, Jacobi one-hotness and entropy fail on more broken runs, but some broken classes remain perfectly one-hot and low-entropy because the selector is decisive about the wrong scheduled position. The full emitted token trace changes on every broken Jacobi run in this sweep. A length mismatch in the emitted sequence is scored as changed; in practice median token-change rates are $0.78$–$0.81$ in the broken quantized/token-collapse rows, so this row should be read as bit-identical trace agreement, not continuous numerical accuracy.

To target the non-collapsed regime directly, we also perturb active Jacobi lookup scores by $\rho$ times each row's own winning margin, leaving all values and non-lookup heads unchanged. This mirrors Appendix N but uses the Jacobi schedule-position oracle. The result is a clean second-family replication of confident wrong addressing: no structural breaks occur through $\rho = 0.5$, while every run is structurally broken for $\rho > 0.5$ and every broken run still has finite, exactly one-hot, zero-entropy attention.

| $\rho$ | runs | structurally broken | median hit | top selected frac. on broken | attention proxies green |
|---|---|---|---|---|---|
| 0 | 400 | 0 | 1.000 | N/A | N/A |
| 0.5 | 400 | 0 | 1.000 | N/A | N/A |
| 0.75 | 400 | 400 | 0.902 | 0.017 | 1.000 |
| 1.0 | 400 | 400 | 0.795 | 0.019 | 1.000 |
| 1.25 | 400 | 400 | 0.719 | 0.020 | 1.000 |
| 1.5 | 400 | 400 | 0.660 | 0.020 | 1.000 |
| 2.0 | 400 | 400 | 0.586 | 0.020 | 1.000 |

All $\rho > 0.5$ settings break all 20 Jacobi program clusters (cluster bootstrap interval $[1, 1]$ for the broken fraction). The selected address distribution is not a collapse to one global position: the 90th percentile top-selected-position fraction among broken runs is $0.037$ at every reported broken scale.

## R TRACR SUITE OUTSIDE THE CRAFT EMITTER

We compiled six public Tracr RASP programs, none produced by the CRAFT emitter: `sort_unique`, `reverse`, `sort_freq`, `frac_prevs(tokens==1)`, `pair_balance` and `count_less_freq(1)`. The domains are intentionally small and exhaustive: 24 length-4 permutations for `sort_unique` and `reverse`, 81 length-4 ternary inputs for `sort_freq` and `count_less_freq(1)`, and 32 length-5 binary inputs for `frac_prevs` and `pair_balance`, for 274 inputs total. The ground truth is each program's symbolic RASP target, with numerical outputs compared at $10^{-6}$ absolute tolerance.

The near-hard attention proxy is evaluated only on rows whose clean f64 attention has maximum probability at least $0.999$ and entropy at most $10^{-3}$, excluding the BOS query row; this avoids counting Tracr's finite-temperature averaging/no-op rows as hard-attention failures. "Confident wrong" below means: the decoded output violates the symbolic target, all recorded activations/logits are finite, and every clean-hard audited row remains near-hard after perturbation.

The main bridge result is the bf16 parameter perturbation: 96 inputs across three programs are wrong while both finiteness and near-hard attention remain green. The score-level rows are reported only as a calibrated stress test.

| Perturbation | inputs | wrong | finite on broken | attn. on broken | confident wrong | programs w/ confident wrong |
|---|---|---|---|---|---|---|
| f64 | 274 | 0 | N/A | N/A | 0 | 0 |
| bf16 | 274 | 96 | 96 | 96 | 96 | 3 |
| per-channel int8 | 274 | 138 | 138 | 36 | 36 | 1 |
| 10% nonzero pruning | 274 | 251 | 251 | 0 | 0 | 0 |
| argmax-flip stress $\rho = 0.75$ | 274 | 96 | 96 | 39 | 39 | 2 |
| argmax-flip stress $\rho = 1.0$ | 274 | 96 | 96 | 96 | 96 | 5 |
| argmax-flip stress $\rho = 2.0$ | 274 | 126 | 126 | 126 | 126 | 6 |

The first four rows use the same parameter perturbation ladder as the original one-program bridge: bf16, per-channel int8 and $10\%$ nonzero pruning. Here bf16 is already a useful outside-emitter check: 96 inputs across three programs are wrong while finiteness and near-hard attention stay green. Nonzero pruning mostly produces soft/collapsed attention, so the attention proxy correctly flags it.

The last three rows are by-construction margin-calibrated argmax inversions, included because the paper's strongest CRAFT/Jacobi claim is about confident wrong addressing. For every clean near-hard row we subtract $\rho$ times that row's clean top-vs-runner-up margin from the selected key and add the same amount to the runner-up key, leaving values and non-hard rows unchanged. This is not a deployment-noise model, and its empirical content is not natural address breakage; it tests whether an outside-emitter compiled circuit can execute the wrong symbolic function while audited attention rows still look decisive. At $\rho = 1$, this happens on 96 inputs across five of the six programs; at $\rho = 2$, all six programs have confident-wrong cases.

The near-hard gate is not tuned to create this effect. Sweeping maximum probability thresholds $0.99, 0.999, 0.9999$ and entropy thresholds $10^{-2}, 10^{-3}, 10^{-4}$ leaves the bf16 bridge unchanged; the score-stress counts are stable under stricter gates and only increase when a looser entropy gate admits more clean-hard rows.

| max prob | entropy | bf16 confident wrong | $\rho = 1$ confident wrong | $\rho = 2$ confident wrong |
|---|---|---|---|---|
| 0.99 | $10^{-4}$ | 96 | 96 | 126 |
| 0.99 | $10^{-3}$ | 96 | 96 | 126 |
| 0.99 | $10^{-2}$ | 96 | 123 | 195 |
| 0.999 | $10^{-4}$ | 96 | 96 | 126 |
| 0.999 | $10^{-3}$ | 96 | 96 | 126 |
| 0.999 | $10^{-2}$ | 96 | 123 | 195 |
| 0.9999 | $10^{-4}$ | 96 | 96 | 126 |
| 0.9999 | $10^{-3}$ | 96 | 96 | 126 |
| 0.9999 | $10^{-2}$ | 96 | 96 | 126 |

| Program | inputs | wrong at $\rho = 1$ | confident wrong at $\rho = 1$ |
|---|---|---|---|
| `sort_unique` | 24 | 18 | 18 |
| `reverse` | 24 | 0 | 0 |
| `sort_freq` | 81 | 9 | 9 |
| `frac_prevs(tokens==1)` | 32 | 16 | 16 |
| `pair_balance` | 32 | 32 | 32 |
| `count_less_freq(1)` | 81 | 21 | 21 |

`reverse` is the one $\rho = 1$ exception, but it breaks with confident near-hard attention at $\rho = 2$ on all 24 inputs. The bridge therefore no longer only says that finiteness is weak: in a second compiler, across several symbolic programs, finite and near-hard attention can remain green while the known target function is wrong.

**Direct Tracr p-bit score arithmetic.** To test whether the CRAFT bit-budget law itself transfers, we also recompute each Tracr attention score from the emitted query/key heads while rounding operands, products, the running sum and the final scale to a chosen binary significand width. This is the direct Tracr analogue of the p26/p40 score-arithmetic harness, but the expected selector is the clean Tracr near-hard argmax rather than a CRAFT oracle. The result is a useful negative: every clean-hard audited row remains selector-correct, and every symbolic output remains correct, even at one score bit. The CRAFT formula $2 \log_2 P + 4.4$ would suggest 8.4–9.0 bits if applied naively to these sequence lengths, so the right conclusion is not transfer of that formula. Tracr's emitted selectors have different, large discrete margins; the transferable object is the audit protocol and the requirement to state a gadget-specific precision budget.

| Program | inputs | clean-hard rows | first correct bits | naive CRAFT bits | wrong rows at 1 bit |
|---|---|---|---|---|---|
| `sort_unique` | 24 | 120 | 1 | 8.4 | 0 |
| `reverse` | 24 | 96 | 1 | 8.4 | 0 |
| `sort_freq` | 81 | 81 | 1 | 8.4 | 0 |
| `frac_prevs(tokens==1)` | 32 | 32 | 1 | 9.0 | 0 |
| `pair_balance` | 32 | 64 | 1 | 9.0 | 0 |
| `count_less_freq(1)` | 81 | 84 | 1 | 9.0 | 0 |

To check that this null result is not only a tiny-$P$ artifact, `p42` compiles a longer Tracr index-shift program $\mathrm{out}_q = \mathrm{tokens}_{q-1}$, with the first position selecting itself, and audits all non-BOS selector argmaxes at sequence lengths 64, 128 and 256. The clean softmax is no longer always near-hard at these lengths, so this is an argmax/semantic audit rather than an attention-proxy audit. The result remains one-bit stable:

| Program | selector rows | first correct bits | naive CRAFT bits | clean min max-prob | clean max entropy |
|---|---|---|---|---|---|
| `shift_prev_len64` | 256 | 1 | 16.4 | 0.997 | 0.0199 |
| `shift_prev_len128` | 512 | 1 | 18.4 | 0.968 | 0.259 |
| `shift_prev_len256` | 1024 | 1 | 20.4 | 0.643 | 2.57 |

This strengthens the boundary conclusion: even when $P$ is large enough that a naive CRAFT-style sufficient ceiling would be 16–20 bits, Tracr's categorical index selector is exactly stable under the p-bit score-arithmetic sweep. The CRAFT law is about the parabolic address gadget, not about all compiled selectors.

**Tracr score-island repair.** We then ask whether the outside-emitter failures are repairable by the same kind of score-path isolation. For each input we run a clean Tracr shadow model, round its

attention logits to a chosen binary significand width, and inject those logits into the perturbed model while leaving the perturbed value projections, output projections, embeddings and MLPs in place. This is a score-island simulation, not a serialized mixed-dtype Tracr artifact. Counts are invariant for every tested width $p \in \{1, \ldots, 8, 10, 12, 14, 16, 17, 18, 20, 24, 32, 53\}$:

| Perturbation | baseline wrong | score-island wrong | repaired | programs repaired |
|---|---|---|---|---|
| bf16 | $96/274$ | $0/274$ | 96 | 3 |
| per-channel int8 | $138/274$ | $36/274$ | 102 | 2 |
| 10% nonzero pruning | $251/274$ | $251/274$ | 0 | 0 |

Thus the Tracr bf16 failures are not CRAFT-emitter artifacts: the wrong outputs disappear when the clean score path is isolated. The median selected-key change fraction is $0$ in these rows, so the intervention is not recovering lost argmax addresses; it is preserving the score values used by Tracr's soft but near-hard compiled attention. The pruning negative is equally important; it shows the repair is specific to score/value separation rather than an oracle that fixes arbitrary parameter damage.

**Fine-tuned Tracr sanity probe.** To add one trained-parameter data point without changing the known target, we perturb the Tracr `sort_freq` checkpoint by Gaussian parameter noise (scale $3 \times 10^{-5}$, seed 123) and distill it for 500 full-batch Adam steps ($10^{-5}$ learning rate) back to the clean pre-argmax symbolic outputs over all 81 inputs. The trained checkpoint changes 62,118/62,134 parameters ($\ell_2$ displacement $0.750$, max absolute displacement $0.0095$) and remains symbolically correct on 81/81 inputs. Applying the same parameter perturbations then gives:

| Perturbation | inputs | wrong | finite on broken | attn. on broken | confident wrong | min max-prob / max entropy |
|---|---|---|---|---|---|---|
| f64 | 81 | 0 | N/A | N/A | 0 | $0.999909 / 9.99\times10^{-4}$ |
| bf16 | 81 | 45 | 45 | 41 | 41 | $1.000000 / 1.88\times10^{-3}$ |
| per-channel int8 | 81 | 62 | 62 | 0 | 0 | $0.375875 / 1.27$ |
| 10% nonzero pruning | 81 | 36 | 36 | 31 | 31 | $0.999529 / 4.48\times10^{-3}$ |

The near-hard gate is defined on the clean f64 audited rows; the final column reports perturbed extrema on those same rows.

This is not a substitute for a broad trained-model study; it is a small trained-parameter sanity check. Its value is that the clean parameters are no longer exactly the compiler emission, yet bf16 and nonzero pruning still produce wrong symbolic outputs while finiteness and near-hard attention often remain green.

**Official ACDC bridge on a public Tracr case.** To check that the stronger-circuit concern is not only addressed by the CRAFT-specific ACDC-style audit, we also run the official circuits-benchmark ACDC implementation on Tracr case 47, "counts the frequency of each unique element." The model and structural labels are the benchmark's released ground-truth TransformerLens model and ACDC-hook circuit, not CRAFT artifacts. The task column evaluates the perturbed model against the benchmark target on 256 sampled inputs at tolerance $10^{-6}$; the ACDC columns sweep $\tau \in \{0, 10^{-4}, 10^{-3}, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 0.71, 1\}$.

| Perturbation | wrong / 256 | token acc. | finite | ACDC full-GT/zero-FP thresholds | empty-circuit thresholds |
|---|---|---|---|---|---|
| f64 | 0 | 1.000 | yes | $10^{-4}$–0.05 | 0.1–1 |
| bf16 | 0 | 1.000 | yes | $10^{-4}$–0.05 | 0.1–1 |
| per-channel int8 | 232 | 0.601 | yes | $10^{-4}$–0.05 | 0.1–1 |
| 10% nonzero pruning | 256 | 0.000 | yes | none | $10^{-4}$–1 |

At $\tau = 0$, ACDC keeps extra false-positive edges for all four rows; the table therefore reports the nonzero-threshold band where recovery has zero false-positive edges. The int8 row is the intended stress case: a standard structural circuit-discovery method recovers the complete known graph, while the same perturbed model computes the wrong function on most inputs. This is not a claim that ACDC is wrong about graph topology; it is a warning that topology recovery alone is not a numerical-correctness certificate. As a secondary release check, public case 61, "ranks each element according to its size," shows the same pattern on a distinct task: clean and int8 both recover the full

benchmark graph with zero false positives for $\tau = 10^{-4}$–$0.05$, while int8 is wrong on 172/256 inputs. We keep case 47 as the headline because it is the first bridge and has the stronger task failure. The release archive also includes corroborating clean/int8 ACDC runs for cases 7, 63 and 78, with int8 wrong on 141/256, 172/256 and 232/256 inputs and the same full-graph, zero-false-positive recovery throughout $\tau = 10^{-4}$–$0.05$.

**Pretrained InterpBench sanity probe.** We also evaluated two public pretrained InterpBench SIIT checkpoints for the same symbolic target (running fraction of x), using a dependency-free replay of the released state dicts. Case 3 is exhaustive over 256 length-5 inputs; case 39 uses 1,024 length-60 samples. Under bf16, per-channel int8 and $10\%$ nonzero pruning, the parameter perturbations do not create a new false-green failure, and attention is not near-hard in either checkpoint.

| Checkpoint | evaluation | perturbation | wrong at 0.05 tol. | min max-prob / max entropy |
|---|---|---|---|---|
| InterpBench 3 | exhaustive 256 | clean | $0/256$ | $0.202$ / $1.61$ |
| InterpBench 3 | exhaustive 256 | bf16/int8/prune | $0/256$ | $0.202$ / $1.61$ |
| InterpBench 39 | 1,024 sampled | clean | $1/1024$ | $0.0169$ / $4.09$ |
| InterpBench 39 | 1,024 sampled | bf16/int8/prune | $1/1024$, $1/1024$, $0/1024$ | $0.0169$ / $4.09$ |

The case-39 clean miss is a single borderline example at the benchmark's stated $0.05$ tolerance. The result is therefore a negative trained-checkpoint probe: the same perturbations that break compiled hard-attention circuits do not break these soft-attention SIIT checkpoints. We report it to avoid turning the Tracr and CRAFT evidence into a trained-model generality claim.

## S SOURCE-OPERATION CONTRACTS AND DECODER REWRITING

**Contract and proof boundary.** For a source-selected nonempty set $S$ of size $m$ among $n$ unmasked positions, the intended attention distribution is uniform on $S$. Empty RASP selections use the compiler's BOS default; selector-width heads include BOS in the selected set. This differs from filtering for clean near-hard rows. For interval-enclosed unscaled scores, let $g$ lower-bound selected-versus-unselected separation and $\omega$ upper-bound the selected score spread. At inverse temperature $\beta$, for $g > 0$ and $m < n$, put

$$R = \frac{n-m}{m} e^{-\beta g}, \qquad B = \min\left(1, \frac{R}{1+R} + \min\left(1, \frac{e^{\beta\omega} - 1}{2}\right)\right).$$

Then the real-softmax distribution has total-variation distance at most $B$ from the source uniform distribution. The leakage term is zero when $m = n$; the within-set term is zero for $m = 1$; nonpositive separation receives the trivial bound. To see the bound, compare the total unselected and selected exponential masses to obtain leakage at most $R/(1 + R)$. Conditional selected probabilities are within an $e^{\beta\omega}$ ratio of uniform, giving the second term, and apply the triangle inequality. These are elementary bounds consistent with hard-to-soft simulation analysis (Yang et al., 2026), not a new general attention theorem. For value diameter $D$ in $\ell_\infty$, maximum value error $e_v$ and a separately justified kernel error $e_k$, the output error is at most $DB + e_v + e_k$. Increasing $\beta$ suppresses leakage but can increase nonuniformity within an intended average.

The release evaluates rational score intervals with outward-rounded decimal exponential bounds. Its independent binary-format arithmetic includes storage casts, ties-to-even and gradual underflow, rejecting overflow. The generic unique-winner check compares the target's lower bound against *every* competitor's upper bound, not only the clean runner-up. The native audit supplies observed logits to this real-softmax bound and measures the kernel residual; it therefore does not certify an unobserved input, upstream error, or a vendor softmax implementation. No violations of the implemented local inequalities were observed; that test is not itself a proof of a hardware guarantee.

**Typed artifact and selection rules.** The Tracr adapter retains the source graph, executes compact blocks in topological order, and allocates a single heterogeneous residual state. At inference a checkpoint contains only tensors, bases and operation metadata; it does not load a RASP interpreter, source model or cached answers. Uniform bf16/fp32/fp64, fp32 QK, source-dependency protection, packed int8, and MLP protection are compared in the same unfused CPU backend. Packed parameters use int8 codes and fp32 per-output-channel scales, dequantized for fp32 computation. This is not

native integer matmul. The calibration baseline selects minimum tensor storage among declared precision/temperature candidates that meet its finite calibration task tolerance. It is explicitly not certified synthesis.

**Rewrite and independent check.** Consider two hidden neurons with first-layer coefficient vectors $a, b$ and second-layer vectors $c, -c$. If $a - b$ is exactly the coefficient vector of a source-invariant constant one, then for every real input in that invariant,

$$c\,\mathrm{ReLU}(a^T x) - c\,\mathrm{ReLU}(b^T x) = c\,\mathrm{clip}(a^T x, 0, 1).$$

Writing $z = b^T x$, both sides are zero for $z \le -1$, equal to $c(z+1)$ for $-1 < z < 0$, and equal to $c$ for $z \ge 0$. This is the familiar clipped-activation identity, not a novel activation. The optimizer matches exact rational coefficient differences and opposite output vectors, verifies that no operation writes the one coordinate, removes one neuron and marks the surviving neuron as clipped. A separately implemented checker reloads the pre-quantization artifacts and validates embeddings, unchanged operations and every witness. It checks 34 pairs across six applicable programs; two control programs have no applicable pair. Floating-point and quantized executions are evaluated separately, since real equivalence does not imply bitwise equivalence.

**Population and task errors.** The six original Tracr domains contain 274 inputs. Two additional public program definitions, cases 47 and 61, are restricted to vocabularies of size three and all lengths 1–5, giving 363 inputs each. Program definitions are unchanged but the domains and compiled artifacts differ from the official ACDC experiment. For each program, seed 20260913 shuffles inputs; at most 12, capped at one-third of its population, form calibration, with the rest forming challenge. There are eight program domains, not 1,000 independent mechanisms. The rewrite uses neither split's labels, but was developed after observing these program families; challenge inputs are not held-out templates. Categorical correctness is exact and numerical correctness uses absolute tolerance $0.02$, with $10^{-6}$ counts also released. Table 11 reports the four broken domains; unique sorting, reverse, prefix fraction and pair balance remain correct. QK-only protection leaves the four packed-int8 error counts unchanged. MLP protection also repairs them but uses more stored tensor bytes than the rewrite. Ordinary bf16 already succeeds on all eight compact domains; the original dense-backend bf16 results are retained separately.

Table 11: Matched compact Tracr backend: wrong challenge sequences before/after the clipped rewrite, with packed tensor bytes. Int8 storage is dequantized for fp32 compute. Cases 47/61 use three-token vocabularies and lengths 1–5; these are not the official 256-input ACDC populations.

| Program | Original int8 | Rewritten int8 | Tensor bytes, before → after |
|---|---|---|---|
| Sort by frequency | $66/69$ | $0/69$ | $2684 \to 2580$ |
| Count low-frequency tokens | $20/69$ | $0/69$ | $794 \to 690$ |
| Case 47, bounded | $265/351$ | $0/351$ | $561 \to 491$ |
| Case 61, bounded | $139/351$ | $0/351$ | $789 \to 719$ |

Tensor storage excludes serialization metadata. For bounded case 47, the original and rewritten packed checkpoints occupy 6,965 and 6,883 bytes, respectively, whereas original bf16 occupies 5,029 bytes. For frequency sorting the corresponding file sizes are 15,533, 15,431 and 11,347 bytes. Thus the rewrite improves matched packed storage, not necessarily total file size versus bf16. CPU latency is released as an indicative unfused-backend measurement, not an acceleration claim.

**Frozen-pass quantizer and length stress test.** Without changing the rewrite, we test per-output, per-input and per-tensor parameter grids with ties-to-even and ties-away rounding on all original inputs. Per-output repairs survive both tie rules. However, rewritten frequency sorting retains $78/81$ errors under per-input grids, and low-frequency counting retains $60/81$ under per-input and per-tensor grids: removing one cancellation pattern is not a universal quantization repair. Cases 47/61 are repaired under all six grids. Recompiling these same definitions at lengths 6 and 8 gives zero rewritten errors under all grids, against original per-output errors of $576/729$ and $487/729$ (case 47), and $588/729$ and $485/729$ (case 61). Length 6 exhausts its domain; length 8 uses 729 of 6,561 inputs sampled without replacement, seed 20260913. This is new-length, same-template evidence, not length extrapolation by the old checkpoint or independent-compiler validation. All 156 conditions,

failures included, are released. This stress test stores grid-rounded parameters in fp32; the packed artifacts are evaluated separately.

**Independent construction and learned control.** The same selected-set checker is applied to public ALTA parity programs with absolute, relative and selected-set-counting algorithms. All 126 binary inputs of lengths 1–6 are evaluated per algorithm, with $L = n + 2$ shared-weight layers. All three algorithms remain correct under fp32, bf16, fp32 QK, packed int8 and MLP-protected packed int8. These are three implementations of *one* task, not three independent task families. ALTA already uses clipped FFNs. Its all-masked relative sentinel has no source aggregate; 1,788 such rows per policy are explicitly unsupported rather than silently assigned an argmax label.

Three associative-recall transformers are trained from random initialization (seeds 0,1,2), not from compiler or teacher weights: one two-head attention layer, width 48, a width-96 ReLU MLP, and shared learned key embeddings between context and query. AdamW uses learning rate $0.002$, weight decay $0.01$, batch 128 and 800 steps on four key/value pairs. Evaluation uses 512 new inputs at each of 4,8,12 pairs per seed. Every clean, native-bf16 and dequantized-int8 condition is correct on all 512 inputs; per-condition Wilson 95% accuracy intervals are approximately $[0.9926, 1]$. Attention is not near-hard. This is a negative trained-model probe, not evidence that the proposed repair generalizes to learned circuits. The structured input embedding is an explicit task bias.

**Representation control and remaining gap.** A matched unique-lookup microbenchmark compares the quadratic gadget, established binary position codes (Giannou et al., 2023), and direct gather, all given the same runtime integer addresses and bf16 values. At capacity 1,024 and scale 16, bf16 quadratic addressing has 55/1,024 selector hits and 989 task errors at tolerance $0.02$; binary bf16 and quadratic fp32 have no task errors. Binary bf16 needs 10 key coordinates and 20,482 tensor bytes, versus 2 coordinates and 8,196 bytes for quadratic fp32. This is a width/precision tradeoff, not a new encoding. We then serialize a minimal end-to-end direct-gather LU artifact for the three frontier circuits (`p54_direct_gather_lu_artifact.py`): after reload, without softmax addressing, a clean transformer or cached answers, all fp32 and bf16 value artifacts have selector hit $1.000$. The fp32 artifacts have maximum backward error $2.27\times10^{-8}$ and maximum solution drift $2.36\times10^{-7}$; bf16 values keep exact addresses but leave value error (maximum backward error $8.82\times10^{-4}$ and maximum drift $9.93\times10^{-3}$). Direct gather is therefore a stable source-operation replacement when runtime integer addresses are available, not a claim about attention-only expressivity.

The implementation now provides source contracts, real-function rewrite witnesses and actual artifacts on independent constructions. It does not yet provide a nonvacuous composed all-input error theorem, a loop invariant, GPU-kernel attestation, or evidence that its compiler strategy improves trained models. FPTuner/Daisy-style certified tuning remains stronger on its supported arithmetic-kernel domain (Chiang et al., 2017; Darulova et al., 2018). These limitations prevent treating the implementation as a completed general-purpose certifying transformer compiler.

## T Additional margin, hardware and generalization controls

**Constructed margin stress.** `p61_adversarial_margin_scaling.py` checks all competitors for $t = P - 1$ at seven capacities from 16 to 65,536. We separate the emitted logarithmic feature with $\alpha = 0.3$ from a constructed bounded-$g$ family: $g(t) = 0$, $g(P) = (1 - 2^{-20})/\ln 2$, and all other features zero, with $\alpha = (1 - M)\ln 2$. The latter approaches the lemma's lower margin for $M \in \{0.5672, 0.1, 0.01, 0.001\}$ but is not emitted by the CRAFT compiler. Both stored coordinates and the two-term dot product use round-to-nearest-even p-bit significands with unbounded exponent. Success requires a unique correct maximum at the reported width and all larger tested widths. At capacity 65,536, the emitted row needs 32 bits; the constructed rows need 32, 35, 38 and 41 bits, respectively. An implementation-specific argmax can return the target at 32 bits even while a competitor ties it, so we report uniqueness separately. The storage-aware bound remains conservative.

**Native Apple-GPU execution.** `p60_native_gpu_mixed_precision.py` reloads a serialized fp32-address/bf16-value artifact for CKT_0131–0133, moves both model streams to the Apple MPS GPU, and executes the full token-driven LU solve. CPU fallback is disabled. The float64 oracle and telemetry reductions run on the CPU; the model recurrences, score products, softmax and value updates run on the GPU. All three address-protected GPU runs have selector hit 1.000, matching

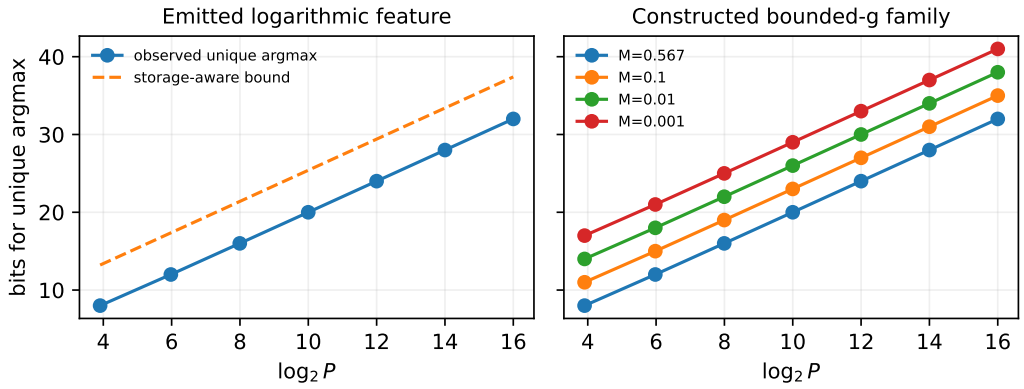


Figure 6: Address magnitude and margin are separate numerical demands. Left: the actual emitted feature at the near-high-end query. Right: constructed lemma-admissible features with smaller margins. The right panel is a controlled test of the bound's mechanism, not evidence that emitted LU programs attain these worst-case margins.

CPU controls. Full bf16 remains broken. This validates the intervention on one native GPU backend, not CUDA kernels, 18-bit hardware, compression or speedup.

Table 12: Native GPU frontier checks. The protected path has fp32 addresses and bf16 values; value error persists after exact selector repair.

| Circuit | full-bf16 hit | protected hit | protected $\eta$ |
|---|---|---|---|
| CKT_0131 | 0.819 | 1.000 | $7.07 \times 10^{-4}$ |
| CKT_0132 | 0.825 | 1.000 | $6.52 \times 10^{-4}$ |
| CKT_0133 | 0.828 | 1.000 | $9.22 \times 10^{-4}$ |

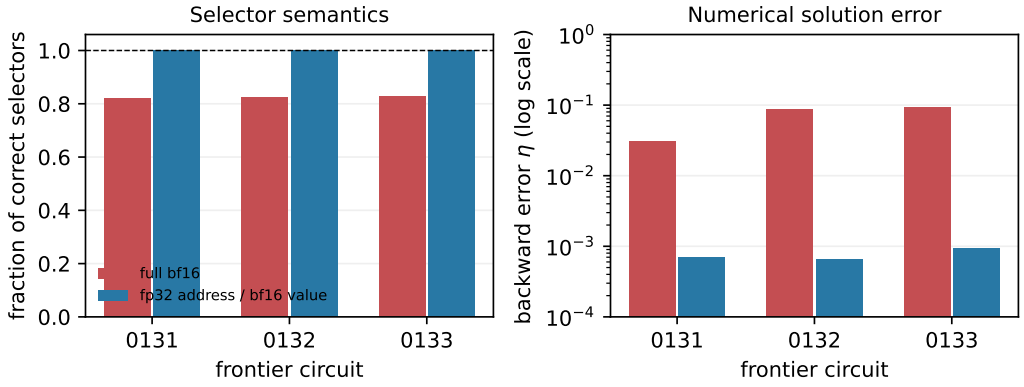


Figure 7: Native Apple-GPU execution separates selector semantics from numerical accuracy. The fp32-address/bf16-value intervention restores every selector on these three programs, but backward error remains nonzero because values still use bf16. These are the same correctness runs as Table 12, not speed or compression measurements.

**Additional official ACDC cases.** We add public sorting cases 9 and 59, each with 256 sampled inputs, clean and per-channel-int8 weights, and thresholds $10^{-4}, 10^{-3}, 0.01, 0.05$. Both clean and int8 executions have zero wrong inputs. All 16 macOS conditions recover the complete graph with zero false positives. A clean Linux rerun recovers it in 15/16: case 59 with int8 at $\tau = 0.05$ misses one

of six edges (TPR $5/6$, FPR 0), while task outputs remain correct. We retain this cross-environment discrepancy rather than claim bit-identical topology recovery. These retained negative cases constrain the interpretation of the earlier case-47/61 dissociation. They do not weaken the counterexample, but show it should not be described as a universal effect of quantization or ACDC. The clean Linux environment also reproduces the headline case-47 result: 232/256 wrong int8 inputs with full graph recovery at all four thresholds.

**Additional trained seeds.** The associative-recall control adds seeds 3–9, retaining every seed, with 800 training steps and 1,024 evaluation sequences at each length 4, 8 and 12. All seven seeds achieve 1.000 accuracy for fp32, bf16 and per-output-channel int8 rounding: 63 reported conditions with zero wrong finite near-hard outputs. Each per-condition Wilson 95% interval is $[0.9963, 1]$; seeds and repeated policies are not pooled as independent tokens. This expands replication within one architecture, not task diversity or evidence about pretrained language models.

**Mechanized algebraic obligations.** `p62_check_precision_lemma.py` submits 11 negated obligations to Z3 4.15.4 (de Moura & Bjørner, 2008); every counterexample query is unsatisfiable. Released SMT-LIB files cover integer separation, the score-gap identity, the bounded-feature margin, two relative perturbations, the $\gamma_2$ bound, absolute dot-product error, storage/arithmetic error composition, the absolute-sum bound, strict-argmax preservation, scale cancellation and the precision rearrangement. This is a mechanization of the algebra under explicit premises. The analytic logarithm bounds, IEEE relative-error model with no overflow/underflow, compiler extraction and GPU implementation remain assumptions; we do not claim a proof-assistant certificate for the full system.

## U Reproducibility details

The reference LU construction uses float64; the precision experiments also execute fp32 and bf16 or explicitly declared p-bit emulation. All reported new semantic-compiler and trained-recall runs use CPU. The production attention scale overflows in float16, not float32; the small compiled workloads are dispatch-bound rather than FLOP-bound. Compiled models are regenerated deterministically from the build scripts given a fixed MILP configuration; see Appendix F for the caveat. Each program builds in 1–45 s and solves in under a second. Entry points, package locks and per-script runtimes are in the artifact `README` and `revision/verification/`. The combined archive includes cached LU models, audit scripts and CSVs, serialized two-dtype and semantic checkpoints, and figure generators. Every distributed file has a SHA256 entry in `RELEASE_MANIFEST.json`; check these hashes before regenerating outputs.

**Clean-environment checks.** We installed new Python environments and ran separately extracted archives on an Apple M5 Pro host with 24 GiB RAM. Quick regeneration took 18.5 s on macOS and 30.7 s in Linux ARM64; both environments passed all 22 semantic, packaging and serial/parallel tests. Stored-result smoke checks took 2.5–4.4 s. Installation time is excluded. Linux experiments ran in BuildKit with network access disabled; ordinary `docker run` was blocked by a host container-start issue, so it is not the claimed validation route. The exhaustive macOS tier separately recomputed the 30-program diagnostic grid, the 142-program sweep at 13 island widths, and both 359,202-row replays. All 13 output CSVs match the reference byte-for-byte, including every row field; the three circuit sweeps use four single-threaded workers. This agreement is specific to the tested environment. Native Apple-GPU and trained-seed reruns also passed in the new macOS environment. The official ACDC Linux run and its one topology discrepancy are retained in Appendix T. These are author-run clean checks, not independent third-party reproduction, and do not promise cross-platform byte-identical schedules or graph recovery.

**Reviewer smoke tests.** The quick CRAFT check is:

```
python revision/p57_craft_artifact_smoke_test.py.
```

It reads the released CSVs and manifests and fails if the full-bf16 105/142 broken count, the 17-bit/18-bit address-island frontier, the 359,202-row precision denominator, the independent replay agreement, the capacity-1,024 binary/quadratic representation counts, the direct-gather LU summary, the wide-$P$ result, or the p-bit/direct-gather artifact hashes change. The non-CRAFT semantic-release

check is:

```
python revision/p52_semantic_release_audit.py.
```

It checks 197 typed-artifact hashes, recomputes the clipped-rewrite witnesses, checks quantizer-stress denominators and runs one isolated inference while forbidding source-compiler imports. These smoke tests do not replace the full experiment reruns in the README. A third smoke auditor checks the additional controls, including the disclosed Linux ACDC discrepancy; none of the smoke checks alone establishes experimental reproduction or public availability.

## V USE OF LARGE LANGUAGE MODELS

An LLM-based coding assistant was used substantially in this work and we disclose that here in line with the conference policy. Its role was: implementing the experiment scripts and the certified-exact oracle; running the experiments and tabulating results; suggesting experiment-design checks; drafting and revising this manuscript; and conducting the literature search, including retrieving and cross-checking bibliographic records. Bibliographic additions in this revision were checked against primary-source records. The research questions, the experimental designs, the interpretation of results and all scientific claims are the authors' responsibility. Automated checks and reruns are documented in the release; they do not replace author review of the final claims and proofs.