hexagon:2611.01127v3

Formally Verified Asymptotic Optimality of Parallel Find-Any Search and Machine-Checked Benchmark Improvements in Lean 4

Robert Kadlec*

November 28, 2026

Abstract

We prove machine-checked finite-n query lower bounds and an exact asymptotic optimality theorem for parallel `GroverFindAny` (m parallel oracle slots, k marked items) under DeepMind's frozen quantum-benchmark specifications. While the constant $\rho_* = 1/2 + 1/\pi$ and restricted-family infimum are due to Fang and Heunen (2026), adaptive stopping with intermediate measurements requires tracking variable-length execution trees rather than a fixed iteration count. We establish

$$\inf\{R \in \mathbb{R} : \text{Feasible}(m, k, R)\} = \frac{1}{2} + \frac{1}{\pi}$$

for *every* positive integer pair $(m, k) \in \mathbb{N}^2_{>0}$ across all adaptive quantum programs with arbitrary finite workspace, intermediate measurements, and nested loops. A formal specification bridge links `Feasible` directly to the frozen submission contract (proving strict separation $\rho_* <$ `relCost` for every rational certificate), and sharpened finite-n bounds—including the pointwise one-round floor $\mathbb{E}[T_{\text{rnd}}] \geq 1$ $(k < 2^n)$—are verified across independent Lean kernels.

Two supplementary verified constructions lower `ArcsinK25ConstCost` by 0.9735% (analyzed for fault-tolerant spacetime trade-offs) and the single-cell $(1024, 512)$ `GroverFindAll` upper bound by 0.946% via 14 Kraus filters (for $n \geq 80{,}000$); audited partial `THC Factorization` helpers appear in Appendix B. All work was AI-executed under human supervision.

1 Introduction, epistemic classification, and overview

In fault-tolerant quantum algorithm design, competing trade-offs must be balanced among query complexity, non-Clifford gate counts (T or Toffoli gates), and logical qubit counts [3, 4, 5, 6, 13, 14, 16, 17]. Because fault-tolerant circuits at their intended target scales lie far beyond exact classical state-vector simulation, `formal-quantum-computation` was introduced by Shirobokov et al. [1]: a Lean 4 library grounded in quantum Hoare logic with classical variables [18, 19, 20, 21], together with machine-checkable benchmark specifications (`QuantumComputationSpecs`) and an AI-discovered reference branch (`discoveries`). The following tasks are covered by those benchmarks: parallel unstructured search (`GroverFindAny` and `GroverFindAll`, where the upstream `discoveries` branch [1] achieved speedups up to 2.57× over GR0 at cell $(1024, 512)$), reversible fixed-point quantum arithmetic (`ArcsinK25ConstCost`, where [1] reduced prior baselines by 290× to spacetime cost 369,800), and large-scale quantum simulation challenges including the `THCFactorization` specification (lacking a discovery implementation on the pinned

*Masaryk University, Brno, Czech Republic. Email: `kadlecr@mail.muni.cz`. ORCID: 0009-0003-9245-8130. Purely AI-generated research directed and supervised by the author, by whom full scholarly responsibility for the work is assumed (all new derivations, formalizations, scripts, audits, and prose were executed by AI systems; see Section 6). This paper is part of a broader research project in which the respective competencies of humans and large language models are investigated.

discoveries branch, where its challenge JSON points solution_module back to the specification placeholder, and unsolved in this release).

When correctness semantics and resource accounting are frozen inside a proof assistant, sharp impossibility barriers and verified circuit improvements can be formalized independently with machine-checked validity relative to the frozen specification and trusted Lean kernel, without access to the original authors' internal search infrastructure. The primary synthesis and formalization were executed on a single 32 GB Windows workstation in Lean 4.35.0, followed by clean-machine 64-bit Linux lake comparator (bwrap sandbox) verification, under a human-directed purely AI-generated workflow detailed in Section 6.

1.1 Epistemic classification

To separate machine-checked theorems from external computations and incomplete infrastructure, every claim in this paper is assigned to one of three explicit tiers:

- **Tier (A) — Machine-checked Lean 4.35.0 theorems against frozen specifications.** Every Tier (A) theorem is proved in Lean 4.35.0 against copies of Quantum Computation (57/57 source files) and QuantumComputationSpecs (15/15 files: 7 .lean sources, 7 challenge JSONs, and ChallengeJsons/README.md) byte-identical to the pinned upstream main branch (where discoveries differs in six challenge JSONs solely in solution_module; exact commit hashes appear in Appendix D), with zero sorry, zero native_decide, and only subsets of Lean's standard axioms {propext, Classical.choice, Quot.sound}. All four challenge constructions across three challenge configurations (ArcsinK25ConstCost, TighterFindAny, sign-only GroverFindAll, and v1.0.1 weak-filter GroverFindAll) were accepted on a separate 64-bit Linux host by the official upstream lake comparator driver inside its bwrap sandbox together with upstream lake test. In addition, the flagship FindAnyLowerBound theorems (for which no frozen upstream challenge JSON exists) and TighterFindAny were exported via lean4export and accepted by two independent third-party Rust Lean 4 checkers bundled with the Lean 4.35.0 toolchain (nanoda_bin 0.4.19 [23] and con-ron -verified [24]; Section 6), whereas the remaining benchmark suites rest on the Lean 4.35.0 kernel family.

- **Tier (B) — Partial formal foundations.** Machine-checked Lean 4.35.0 modules (Track1THC, 9 modules, 1,956 lines; Appendix B) are provided in which algebraic and circuit-composition building blocks are verified against the frozen library without filling the public challenge holes.

- **Tier (C) — External certified numerics, bounded screens, and paper lower bounds.** Arbitrary-precision interval enclosures in python-flint/Arb [25] (the external GR0 reference constant used to express asymptotic GroverFindAll speedup ratios and the exhaustive 16,777,217-input arcsin ULP audit), independent rational Taylor enclosures (enclose.py), finite Boolean Möbius, Pfaffian, and MILP computations, and paper proofs linking finite witnesses to restricted gate-model lower bounds (Appendix C) are classified here.

1.2 Hierarchy of results

The scientific weight of the contributions is deliberately stratified. The primary theoretical result is the general adaptive lower bound and exact asymptotic optimality of parallel GroverFindAny (Section 3). The ArcsinK25ConstCost reduction (Section 4, Theorem 4.1) and the single-cell asymptotic GroverFindAll upper bounds (Section 5) are presented as supplementary benchmark-specific constructions. The rational precision refinement TighterFindAny is recorded only as a brief technical remark (Section 3.4), and the partial THCFactorization

modules are placed in Appendix B. The main results relative to the pinned `discoveries` branch (commit hash in Appendix D) are summarized in Table 1.

Benchmark / Problem	Prior baseline [2, 1]	Present work	Contribution	Tier
`GroverFindAny` (general adaptive model; all $m, k > 0$)	Constant $1/2 + 1/\pi$ and restricted-family infimum [2]; parallel upper bound [1]	$\inf\{R :$ Feasible$(m, k, R)\} =$ $\mathbf{1/2 + 1/\pi}$ over all `OracleDynamicCircuit` + finite-n bound	General adaptive optimality	(A)
`ArcsinK25ConstCost` (Q2.23, $100\,Q$)	369,800 ($3{,}698\,T \times 100\,Q$, TOFF $= 0$)	**366,200** ($3{,}662\,T \times 100\,Q$, TOFF $= 0$) + exact compiled count & witness	$-3{,}600$ spacetime ($-36\,T$, -0.9735%); 5/16 modules edited	(A)
`GroverFindAll` $(1024, 512)$ sign-only	$27{,}024{,}465{,}988/10^{12}$ ($\geq 2.574566\times$ vs ext. GR0)	$27{,}014{,}193{,}664/10^{12}$ ($\geq 2.57554525548\times$ vs ext. GR0)	-0.038011% upper bound (eventual; $n \geq 80{,}000$ engine)	(A)/(C)†
`GroverFindAll` $(1024, 512)$ weak filters	Pinned discovery ($\geq 2.574566\times$ vs ext. GR0)	$\mathbf{26{,}768{,}771{,}052}/10^{12}$ ($\geq 2.599158481606\times$ vs ext. GR0)	-0.946% vs pinned (-0.908% vs sign-only); 14 Kraus filters (all $n \geq 80{,}000$)	(A)/(C)†
`THCFactorization` (Appendix B)	No discovery impl. on `discoveries`; unsolved in this release	9 helper modules (1,956 lines), 0/7 def. holes + 0/2 thms	Partial infrastructure only	(B)

Table 1: Stratified summary of results relative to pinned `google-deepmind/formal-quantum-computation` (commit hashes in Appendix D). †In Lean 4.35.0, the single-cell $(1024, 512)$ exact rational upper bound relative to `vanillaFindAllCost` is certified (Tier A); derived asymptotic speedup multipliers are obtained by dividing an externally enclosed 256-bit Arb lower bound for GR0 by the Lean upper bound (Tier C).

Hardware scope note. As documented in `QuantumComputationDiscoveries/README.md` on the `discoveries` branch of [1], 230–233 GB of peak RAM (plus swap) are required to run the upstream Lean comparator on the three large-scale quantum simulation benchmarks (`P450Ground Energy`, `Photochemistry`, `StoppingPower`), by which the 32 GB host workstation is exceeded; those benchmarks were not modified or re-verified.

2 Related work and exact positioning

To evaluate novelty precisely, the contributions must be positioned against four lines of prior work: adaptive expected-cost quantum search, parallel multi-item search, weak-measurement search protocols, and multiplicative-complexity circuit synthesis over $\mathbb{F}_2$.

Expected-cost optimality, BBHT, Zalka's angle method, and Fang–Heunen [4, 5, 2]. Distinct settings are treated in classical optimality analyses of unstructured search. Grover's algorithm for multiple marked items, unknown-size restart schedules, and general search lower bounds were analyzed by Boyer, Brassard, Høyer, and Tapp [4], whereby the standard $\frac{\pi}{4}\sqrt{N/k}$ fixed-budget scale was established. Exact fixed-budget optimality for a single marked item was proved by Zalka [5], by whom the shared-memory synchronous parallel query model adopted in `QuantumComputationSpecs` and the geometric angle-displacement technique were also introduced in the single-target setting. That geometric technique is extended to multi-target uniform k-subset parallel oracles by the compression bound in Section 3.2. Moreover, a strictly different

variational problem is posed by the frozen `GroverFindAny` benchmark: minimizing the *expected* number of parallel oracle rounds, averaged uniformly over k-element marked sets, among variable-stopping-time protocols that are required to terminate with certainty (probability one) on every marked set.

Crucially, the leading constant $\rho_* = 1/2 + 1/\pi$ is *not* new. An exact weak-measurement-conditioned loop for known target probability p using $(\pi/8 + 1/4 + o(1))/\sqrt{p}$ iterations—from which, upon specialization to $p = k/N$ and division by the standard $\frac{\pi}{4}\sqrt{N/k}$ scale, precisely $\rho_* = 1/2 + 1/\pi$ is obtained—was derived by Fang and Heunen [2], together with a matching infimum within their continuous-angle analysis of a restricted single-checkpoint weakly measured loop family [2]. That result is cited directly by Shirobokov et al. [1], where $1/2+1/\pi$ is recorded in Appendix E.2 Eq. (9) and achievability is extended to parallel grid cells in Appendix E.3.

The contribution of Section 3 is therefore neither the discovery of the constant $1/2 + 1/\pi$ nor the upper-bound circuit, both of which belong to [2, 1]. Rather, an extended multi-target parallel angle-displacement bound is combined with a convex stopping potential that is non-decreasing under adaptive measurements, and it is formalized in Lean 4.35.0 that no asymptotic coefficient below $1/2 + 1/\pi$ is achievable by any protocol in the full `OracleDynamicCircuit` class—wherein arbitrary finite workspace, arbitrary oracle-independent unitaries, arbitrary legal slot embeddings, adaptive intermediate measurements, and arbitrarily nested `qWhile` loops are permitted.

Quantum Las Vegas complexity (Belovs–Yolcu [7]). A general adversary framework for Las Vegas quantum query complexity is formulated by Belovs and Yolcu ([7], Section 4.1), in which coherent query cost $L(A, O, \xi)$ is measured by the squared norm of the active state component entering each query. In Section 4.1 of [7], a one-way conversion is also noted from any algorithm B using intermediate measurements to a deferred-measurement algorithm A satisfying $T(B, O, \xi) \geq L(A, O, \xi)$, by which coherent Las Vegas adversary lower bounds are shown to lower-bound operational expected query cost. However, that coherent lower bound is not tight for the operational realized-round cost in `QuantumComputation` (where each realized parallel oracle call across classical branches and nested loops is charged, as is any fixed unitary whose matrix coincides with an embedded oracle). Specifically, as a Tier (C) analytical comparison for single-target search with $N \geq 2$ and a single bit oracle, the corresponding coherent Las Vegas adversary optimization is solved by $L^*_{\mathrm{adv}} = \sqrt{N-1}/2$: the bit-oracle feasibility condition $4ab = 1$ is satisfied by primal weights $a^2 = \sqrt{N-1}/4$ on the target index and $b^2 = 1/(4\sqrt{N-1})$ elsewhere, while the dual witness $\Gamma = (J - I)/(2\sqrt{N-1})$ (where J denotes the $N \times N$ all-ones matrix and $(\Delta_j)_{xy} = \mathbf{1}_{[x=j]\neq[y=j]}$ denotes the binary difference mask for index j) has star-graph norm $\|\Gamma \circ \Delta_j\| = 1/2$ (so $\|2(\Gamma \circ \Delta_j)\| = 1$ for the full bit-oracle difference block) and top eigenvalue $\sqrt{N-1}/2$. Thus an asymptotic leading constant of $1/2$ (relative coefficient $2/\pi \approx 0.6366$) is produced by the coherent adversary, rather than the sharp operational constant $\pi/8 + 1/4 \approx 0.6427$ (relative coefficient $1/2 + 1/\pi \approx 0.8183$). In Section 3, the sharp operational lower bound is proved directly in the quantum Hoare semantics of [18, 1].

Parallel multi-item search and weak measurements [5, 6, 8, 9]. Parallel unstructured search has been developed from Zalka [5] for parallel single-target queries and Grover and Radhakrishnan [6] for finding k marked items using m parallel processors. Likewise, weak measurements and controlled damping in quantum search have established precursors, including Mizel's critically damped quantum search [8] and the weakly measured while-loop semantics of Andrés-Martínez and Heunen [9]. The Kraus pair $(K_{0,t}, K_{1,t})$ employed in Section 5 is a standard partial projective test on the full-occupancy subspace; the incremental contribution of Section 5 lies in combining full-occupancy Kraus checks with the 24-segment Johnson walk, optimizing the schedule under survival-weighted branch accounting, and formally verifying the resulting finite-circuit restart bound in Lean 4.35.0.

Boolean multiplicative complexity and formal specification fidelity [10, 11, 12, 22]. Algebraic-degree lower bounds on multiplicative complexity ($\mathrm{MC}(f) \geq \deg(f) - 1$) and polar-matrix canonicalization of quadratic Boolean forms over $\mathbb{F}_2$ were introduced by Schnorr [10] and Mirwald and Schnorr [11], and were further developed in low-complexity classification studies such as Çalık, Sönmez Turan, and Peralta [12]. In Section 4, these algebraic techniques are integrated with the live-wire spans, 50-ancilla budget, uncomputation constraints, residual phase cancellation, and full Lean 4.35.0 proof obligations of the frozen 25-bit arcsine circuit. Finally, as emphasized in the Lean FRO `comparator` documentation [22] (particularly regarding definition holes), solving the intended problem is not guaranteed by the absence of `sorry` or custom axioms alone; specification fidelity is established only when exact agreement of both public theorem signatures and the underlying model definitions against the frozen specification is verified.

3 Exact asymptotic optimality of parallel GroverFindAny

The greatest lower bound of feasible eventual relative coefficients in the frozen parallel `GroverFindAny` model is established to be

$$\rho_* = \frac{1}{2} + \frac{1}{\pi}$$

for *every* positive integer pair $m, k > 0$ (upgrading the initial 121-cell dyadic grid restriction of `v1.0.x` via `P1Upgrade.FindAnyAllPos` and `P1Upgrade.AllPositiveInfimum`). Both the general lower bound and the matching upper construction hold parametrically for all $m, k \in \mathbb{N}_{>0}$, while the 121-cell grid remains the domain of the frozen `GroverFindAnyRelCost.json` benchmark.

3.1 The frozen model, formal statements, and scope boundaries

The abbreviations $N = 2^n$, $\mathcal{S} = \texttt{groverTargetSets}\ n\ k$, and $L = |\mathcal{S}|$ are adopted throughout this section, and $\texttt{groverParallelMarkingOracle}\ m\ S$ is denoted by O_S. A single source circuit C, fixed before S is drawn uniformly from $\mathcal{S}$, is given by the type

$$\texttt{OracleDynamicCircuit}\ (\texttt{Unitary}\,(2^w))\ \texttt{Grover.Var}\ w\ ((n+1)m).$$

Within this type, arbitrary finite workspace w (possibly depending on n) may be allocated, arbitrary oracle-independent unitaries may be applied, legal embeddings of the m-slot parallel oracle (each slot holding an n-bit address and one marking bit) may be invoked, intermediate measurements with classical branching may be performed, and arbitrarily nested `qWhile` loops may be executed. The compiled program, cost model, and vanilla scaling are denoted by

$$P_S = \texttt{qset}\ ;;\ C.\texttt{compile}\ \texttt{QProg.qAssign}\ O_S,$$
$$\mathcal{C}_S = \texttt{CostModel.ofEmbeddedOracle}\ O_S,$$
$$V_{\mathrm{any}}(n, m, k) = \texttt{vanillaFindAnyCost}\ n\ m\ k = \frac{\pi}{4}\sqrt{\frac{N}{mk}}.$$

Under the frozen cost model $\mathcal{C}_S$, one oracle round is charged for each oracle invocation and also for any fixed unitary step whose matrix coincides with an embedded oracle. Total correctness with probability one for every marked set is denoted by $H(C, k)$ (`Lower.Correct`):

$$\forall S \in \mathcal{S}, \quad \{\top\}\ P_S\ \{\sigma \mapsto \exists x \in S,\ \sigma(\texttt{Grover.Var.res}) = x.\texttt{val}\}.$$

The certificate predicate $\texttt{AvgOracleCostBound}\ k\ (S \mapsto \mathcal{C}_S)\ (S \mapsto P_S)\ q$ is abbreviated as $A(C, k, q)$, under which per-set nonnegative real certificates q_S are required to satisfy the frozen `CostBound` for all $S \in \mathcal{S}$ with uniform average at most q. Values in $\mathbb{R}^{\infty}_{\geq 0}$ (`ENNReal`) are taken by the underlying operational cost expectedCost.

Theorem 3.1 (Finite certificate and actual expectation)**.** *For any $n, m, k, w \in \mathbb{N}$ satisfying $0 < m$, $0 < k$, and $k \le N$, and any circuit C of the type above, with*

$$B(n, m, k) = \frac{(\pi/4 + 1/2)(1 - k/N)}{2 \arcsin \sqrt{mk/N}},$$

where Lean's total real inverse sine (equal to $\pi/2$ on arguments ≥ 1) is denoted by arcsin*:*

1. (`finite`) *Under $H(C, k)$, the inequality $B(n, m, k) \le q$ is satisfied by every $q \in \mathbb{R}$ for which $A(C, k, q)$ holds.*

2. (`actual_expected_rounds`) *Under $H(C, k)$, for every store σ : `Store Grover.Var`,*

$$\texttt{ENNReal.ofReal}\big(B(n, m, k)\big) \;\le\; \frac{\sum_{S \in \mathcal{S}} \mathrm{expectedCost}(\mathcal{C}_S, P_S, \mathrm{seed}(w, \sigma))}{L},$$

   where σ is paired with the pure quantum state $|0^w\rangle$ by $\mathrm{seed}(w, \sigma)$ (`Track2Grover.Path.seed`). Infinite expected cost is included in the extended-real comparison, and no external certificate q is required.

For fixed positive $m, k \in \mathbb{N}$ and $R \in \mathbb{R}$, the predicate $\mathrm{Feasible}(m, k, R)$ is defined to hold whenever there exist $w : \mathbb{N} \to \mathbb{N}$ and a family C_n : `Lower.Circuit` n m $(w(n))$ such that

$$\begin{aligned} &\forall n, \quad 1 \le n \implies k \le 2^n \implies H(C_n, k), \\ &\exists n_0, \quad \forall n \ge n_0, \quad k \le 2^n \implies A\big(C_n, k, R\, V_{\mathrm{any}}(n, m, k)\big). \end{aligned}$$

Theorem 3.2 (Asymptotic boundary and exact infimum)**.** *In namespace `Track2Grover.FindAnyLowerBound`:*

1. (`asymptotic_real`) *For all $m, k > 0$ and $R \in \mathbb{R}$, if* $\mathrm{Feasible}(m, k, R)$ *holds then $\rho_* \le R$.*

2. (`no_smaller_rational`) *For all $m, k > 0$ and $R \in \mathbb{Q}_{\ge 0}$ with $(R : \mathbb{R}) < \rho_*$, the predicate* $\mathrm{Feasible}(m, k, R)$ *is false.*

3. (`exact_infimum`, `exact_infimum_all_pos`) *For every (m, k) in the benchmark grid `groverGrid` × `groverGrid` (`FindAnyLowerBound`)—and, via `P1Upgrade.FindAnyAllPos` and `P1Upgrade.AllPositiveInfimum`, for* every *positive integer pair $m, k \in \mathbb{N}_{>0}$—the predicate* IsGLB *holds at ρ_* for $\{R \in \mathbb{R} : \mathrm{Feasible}(m, k, R)\}$; equivalently,*

$$\inf\{R \in \mathbb{R} : \mathrm{Feasible}(m, k, R)\} = \frac{1}{2} + \frac{1}{\pi}.$$

Machine-checked specification bridge (`P1Upgrade.SubmissionBridge`, Tier A). To eliminate any manual-inspection gap between `Feasible` and the frozen upstream challenge JSON `GroverFindAnyRelCost.json`, the module `P1Upgrade.SubmissionBridge` (173 lines, 10 exported targets; all five checks exit 0: `lean` and `leanchecker` over the official kernel plus `con-leche -verified`, `con-ron -verified`, and `nanoda_bin`) faithfully packages all eight required challenge declarations directly over frozen `QuantumComputation.Specs` in `structure GroverFindAnySubmission`. For every claimed cell, its reset-and-compile program equals the lower-bound program by definitional equality (`rfl`), while the correctness and average-cost fields transfer by definitional conversion to yield a witness of `FindAnyLowerBound.Feasible` (`submission_feasible`). Combining this transfer with Mathlib's `irrational_pi` (`irrational_half_add_inv_pi`) certifies the strict inequality $1/2 + 1/\pi <$ (`sub.relCost` m $k : \mathbb{R}$) for every valid submission (`submission_relCost_gt`), instantiated by a zero-`sorry` witness (`tighterFindAnySubmission`).

Explicit scope boundaries. Theorems 3.1 and 3.2 are restricted specifically to (i) expected *parallel oracle rounds*, (ii) uniform averaging over k-element marked sets of known size k, (iii) termination and marked output with probability one, (iv) circuits in the frozen `OracleDynamicCircuit` model with arbitrary finite workspace $w(n)$, and (v) fixed positive (m, k) as $n \to \infty$. Attainment at equality by a single real witness is not asserted by the infimum; because ρ_* is irrational, equality with ρ_* is impossible for any submitted nonnegative rational. Different success probabilities (e.g. bounded-error search) or non-uniform input distributions are excluded from these theorems.

3.2 Coherent progress potential and convexity under branching

The executions for all $S \in \mathcal{S}$ are coupled into a direct-sum Hilbert space $\bigoplus_{S\in\mathcal{S}} \mathcal{H}_w$, and the subspace of vectors constant across the hidden label S is denoted by K, with orthogonal projection Π_K. For any nonzero branch vector x, its squared norm is denoted by $s = \|x\|^2$ and its angle $\alpha(x) \in [0, \pi/2]$ to K is defined by $\|\Pi_K x\|^2 = s\cos^2\alpha(x)$ (with zero contribution assigned to zero branches). With $c = \pi/4 + 1/2$, the progress potential is defined by

$$\Phi(s,\alpha) = \begin{cases} s(c-\alpha), & 0 \le \alpha \le \pi/4, \\ s\cos^2\alpha, & \pi/4 < \alpha \le \pi/2. \end{cases}$$

In terms of the projected-mass fraction $u = \cos^2\alpha \in [0,1]$, the per-unit-mass profile is given by

$$f(u) = \begin{cases} u, & 0 \le u \le \frac{1}{2}, \\ c - \arccos\sqrt{u}, & \frac{1}{2} \le u \le 1. \end{cases}$$

At the junction $u = 1/2$ ($\alpha = \pi/4$), equal value ($f(1/2) = 1/2$) and equal first derivative ($f'(u) = 1/(2\sqrt{u(1-u)}) = 1$ at $u = 1/2$) are shared by both pieces. On $(1/2, 1)$,

$$f''(u) = \frac{2u-1}{4[u(1-u)]^{3/2}} > 0;$$

hence f is continuous and convex on $[0,1]$, and continuously differentiable on $(0,1)$ (indeed on $[0,1)$), with matching one-sided derivatives equal to 1 at $u = 1/2$ (whereas $f'(u) \to +\infty$ as $u \to 1^-$, and the endpoint $\alpha = 0$ is handled directly below). In the angle variable α, the derivative of the per-unit-mass profile is given by -1 on the linear branch $0 < \alpha < \pi/4$ and by $-\sin(2\alpha)$ on the cosine-squared branch $\pi/4 < \alpha < \pi/2$; both magnitudes are bounded by 1, and continuity is preserved at $\alpha = \pi/4$ with matching slope -1, by which a uniform unit Lipschitz bound in α on $[0, \pi/2]$ is established.

One-round oracle displacement. Each physical marking oracle O_S is an involution in every density regime (`Physical.act_involution` in `CoupledOracle.lean`), so the coupled oracle $U = \bigoplus_{S\in\mathcal{S}} O_S^{\mathrm{emb}}$ is a self-adjoint unitary involution ($U = U^\dagger$, $U^2 = I$). A label-constant vector $y \in K$ is written in terms of its shared single-instance component $y_0 \in \mathcal{H}_w$ as $y = (y_0)_{S\in\mathcal{S}}$, so that $\|y\|^2 = L\,\|y_0\|^2$. Each of the m query addresses is contained in a uniform k-subset S with probability k/N; when the probability of an odd number of active hits is bounded by that of at least one hit (for which no independence across the m slots is required), the unconditional compression inequality is obtained:

$$\begin{aligned} \sum_{S\in\mathcal{S}} \langle y_0,\, O_S^{\mathrm{emb}} y_0\rangle &\;\ge\; (1 - 2mk/N)\, L\, \|y_0\|^2, \\ \mathrm{Re}\langle y, Uy\rangle = \langle y, Uy\rangle &\;\ge\; (1 - 2mk/N)\, \|y\|^2 \end{aligned}$$

for every legal slot embedding O_S^{emb}, even on entangled inputs (where realness of the inner product follows from self-adjointness of U, and the inequality is trivial when $1 - 2mk/N < -1$).

Whenever $mk \le N$, the angle bound $v = 2\arcsin\sqrt{mk/N} \in [0,\pi]$ is defined, so that $\cos v = 1 - 2mk/N$. For any unit vector $y \in K$, $\mathrm{Re}\langle y, Uy\rangle \ge \cos v$, so the angle between y and Uy is at most v. For an arbitrary unit branch vector x (possibly already correlated with S), whenever $\Pi_K x \neq 0$ the normalized projection $y = \Pi_K x/\|\Pi_K x\| \in K$ is chosen. By unitarity of U, the spherical angle between Ux and Uy is equal to $\alpha(x)$, while the angle between Uy and $y \in K$ is at most v; by the spherical triangle inequality and the characterization of $\alpha(Ux)$ as the minimal angle from Ux to K (`QueryAngle.one_query`),

$$\alpha(Ux) \;\le\; \angle(Ux, y) \;\le\; \angle(Ux, Uy) + \angle(Uy, y) \;\le\; \alpha(x) + v.$$

(If $\Pi_K x = 0$, then $\alpha(x) = \pi/2$, so $\alpha(Ux) \le \pi/2 \le \alpha(x) + v$ is satisfied directly.) When the same inequality is applied to Ux and $U^2 = I$ is invoked, $\alpha(x) = \alpha(U^2 x) \le \alpha(Ux) + v$ is obtained, hence $|\alpha(Ux) - \alpha(x)| \le v$. By the unit Lipschitz bound in α, the potential $\Phi(s,\alpha)$ is reduced by at most sv in a single parallel oracle round.

Adaptive measurement splitting. Oracle-independent measurements may branch asynchronously across classical stores, yet within any single branch history the classical control state is shared across all components indexed by $S \in \mathcal{S}$. The measurement instrument is therefore represented on the coupled space by block-diagonal Kraus operators $I_{\mathcal{S}} \otimes M_i$ with $\sum_i M_i^\dagger M_i = I$, each commuting with the label-averaging projection Π_K. Consequently, when a branch (s,α) is split into children (s_i,α_i) with $x_i = (I_{\mathcal{S}} \otimes M_i)x$ and $u_i = \cos^2\alpha_i$, both total mass ($\sum_i s_i = s$) and projected mass are preserved:

$$\sum_i s_i u_i = \sum_i \|\Pi_K (I_{\mathcal{S}} \otimes M_i)x\|^2 = \sum_i \|(I_{\mathcal{S}} \otimes M_i)\Pi_K x\|^2 = \|\Pi_K x\|^2 = su.$$

An affine supporting minorant at u is obtained from the convexity of f (the tangent line for $0 < \alpha \le \pi/4$, the identity $u \mapsto u$ for $\alpha > \pi/4$, and direct equality when $\alpha = 0$ forces $u_i = 1$ on every positive-mass child), whereby `split_potential` (`CoupledHistory.lean`) is proved:

$$\Phi(s,\alpha) \;\le\; \sum_i \Phi(s_i,\alpha_i).$$

Thus the potential budget charged to oracle rounds cannot be reduced by adaptive measurement branching.

Finite execution-tree inequality. On any branch halted with classical output address z, nonzero amplitude is supported only on hidden sets $S \ni z$ (a fraction k/N of $\mathcal{S}$), so its projected mass fraction is bounded by $u \le k/N$ via Cauchy–Schwarz. By convexity of f on $[0,1]$ with $f(0) = 0$ and $f(1) = c$,

$$f(u) \;\le\; (1-u)f(0) + uf(1) \;=\; cu,$$

whereby the terminal potential of every halted branch is bounded above by $c(k/N)s$ (and in fact by $(k/N)s$ whenever $k/N \le 1/2$; see Proposition 3.3). For a finite execution tree $\mathcal{T}$ started from an unnormalized label-constant vector $x \in K$ of squared norm $\|x\|^2 = L$, the expected oracle rounds executed on target set S are denoted by $q_{\mathcal{T}}(S)$, the summed expected query count across $\mathcal{S}$ by $Q_{\mathcal{T}} = \sum_{S\in\mathcal{S}} q_{\mathcal{T}}(S)$, and the unnormalized unfinished timeout mass by $h_{\mathcal{T}} = L - \mathrm{finished}(\mathcal{T}, x)$. When the branch inequalities are summed under zero wrong-output mass ($cL \le vQ_{\mathcal{T}} + c(k/N)(L - h_{\mathcal{T}}) + ch_{\mathcal{T}}$), the scaled tree bound (and its equivalent per-instance average form after division by $L > 0$) is obtained:

$$c\,(1-k/N)\,(L - h_{\mathcal{T}}) \;\le\; v\,Q_{\mathcal{T}} \qquad\Longleftrightarrow\qquad c\,(1-k/N)\left(1 - \frac{h_{\mathcal{T}}}{L}\right) \;\le\; v\,\frac{Q_{\mathcal{T}}}{L}. \tag{1}$$

3.3 Reduction from nested loops to finite trees and actual expected cost

The extension of (1) from finite trees to arbitrary `OracleDynamicCircuit` programs with nested `qWhile` loops and extended-real `expectedCost` is carried out across five formal stages in flat modules under `Track2Grover/` (`FiniteReduction.lean`, `LowerPath.lean`, `LowerScoreBridge.lean`, `LowerAccounts.lean`, `LowerSeed.lean`, `CoupledLimit.lean`, and `LowerActual.lean`):

1. *Uniform finite unrolling of nested loops (`Fuel.cap`).* Every loop in a program P (regardless of nesting depth) is truncated simultaneously by a single natural-number fuel parameter $i \in \mathbb{N}$ via the recursive map $\texttt{qWhile}\ g\ P \mapsto \texttt{unroll}\ g\ (\texttt{cap}\ i\ P)\ i$, where $\texttt{unroll}\ g\ P'\ 0 = \texttt{abort}$ and $\texttt{unroll}\ g\ P'\ (d+1) = \texttt{qIf}\ g\ (\texttt{qSeq}\ P'\ (\texttt{unroll}\ g\ P'\ d))\ \texttt{skip}$. Because structural truncation commutes with oracle compilation (`compile_cut` in `LowerPath.lean`), $\texttt{cap}\ i\ (C.\texttt{compile qAssign}\ O_S)$ is represented exactly by a finite execution tree $\mathcal{T}_i$ with explicit timeout leaves.

2. *Monotonicity and cost domination (`Fuel.cap_cost`).* The operational orders $\texttt{Below}\ P_1\ P_2$ (satisfied when $\mathrm{obs}(P_1,\delta,\theta) \le \mathrm{obs}(P_2,\delta,\theta)$ for all δ,θ) and $\texttt{StateBelow}\ \delta\ \eta$ (satisfied when $\mathrm{mass}(\delta,\theta) \le \mathrm{mass}(\eta,\theta)$ for all θ) are introduced on programs and states. By structural induction over sequential composition, conditionals, and loop unrollings, $\texttt{Below}\ (\texttt{cap}\ i\ P)\ P$ is proved together with

$$\mathrm{expectedCost}(\mathcal{C}, \texttt{cap}\ i\ P, \delta) \ \le\ \mathrm{expectedCost}(\mathcal{C}, P, \delta)$$

   for every cost model $\mathcal{C}$ and state δ. Combined with `score_calls_le_cost` (`LowerScoreBridge.lean`), the per-instance expected query count $q_i(S)$ of the truncated tree $\mathcal{T}_i$ is bounded by

$$\begin{aligned} q_i(S) \ &\le\ \mathrm{expectedCost}(\mathcal{C}_S, P_S, \mathrm{seed}(w,\sigma)) \quad \text{for all } i, S, \\ Q_i := \sum_{S\in\mathcal{S}} q_i(S) \ &\le\ \sum_{S\in\mathcal{S}} \mathrm{expectedCost}(\mathcal{C}_S, P_S, \mathrm{seed}(w,\sigma)). \end{aligned}$$

3. *Cofinal convergence and vanishing unfinished mass (`Fuel.cofinal`).* By structural induction on P, $\lim_{i\to\infty} \mathrm{obs}(\texttt{cap}\ i\ P,\delta,\theta) = \mathrm{obs}(P,\delta,\theta)$ is established for every initial state δ and bounded effect θ in the public cofinality theorem `Fuel.cofinal` (`Approaches (fun t => cap t P) P`, with fields `mono`, `upper`, and `limit`). Under Hoare total correctness $H(C,k)$, unit trace and zero mass on incorrect stores are proved for the full output state on every $S \in \mathcal{S}$ (`correct_output_mass` in `LowerAccounts.lean`). Wrong-output mass on every truncation $\mathcal{T}_i$ is therefore forced by monotonicity to vanish identically ($\texttt{bad}(\mathcal{T}_i, S) = 0$), while with $\theta = \top$ the limit $\lim_{i\to\infty} \mathrm{finished}(\mathcal{T}_i, x) = L$ is obtained, so the unnormalized unfinished mass $h_i = L - \mathrm{finished}(\mathcal{T}_i, x)$ converges to 0 (and $h_i/L \to 0$).

4. *Input-independence of the initial state (`seed_reset`).* Because every compiled benchmark program begins with $\texttt{qset}\,;;C.\texttt{compile qAssign}\ O_S$, the quantum register is reset to $|0^w\rangle$ independently of S. For any fixed initial classical store σ, $\delta = \mathrm{seed}(w,\sigma)$ is invariant under `qset` (`seed_reset` in `LowerSeed.lean`), so the unnormalized coupled initial vector $x = \mathrm{copy}(\mathcal{S}, |0^w\rangle)$ is placed identically inside the label-constant subspace K with squared norm $\|x\|^2 = L = |\mathcal{S}|$.

5. *Passing to the limit and handling infinite expected cost (`Actual.mean_lower_bound`).* For any finite real bound q with $A(C,k,q)$, the summed query count is bounded by $Q_i \le Lq$ for all i; when $Q_i/L \le q$ is substituted into the normalized right-hand form of (1), $c(1-k/N)(1-h_i/L) \le vq$ is obtained, and in the limit $h_i \to 0$ the finite bound $B(n,m,k) = c(1-k/N)/v \le q$ (`finite`) is proved. For the direct extended-real

expectation in `actual_expected_rounds`, if $\sum_{S\in\mathcal{S}} \mathrm{expectedCost}(\mathcal{C}_S, P_S, \mathrm{seed}(w,\sigma)) = \infty$, then division by $L > 0$ yields ∞ and the inequality is discharged trivially by `le_top`. Otherwise every summand is finite, so finite pointwise seed costs $z_S := (\mathrm{expectedCost}(\mathcal{C}_S, P_S, \mathrm{seed}(w,\sigma)))$.`toNNReal` are extracted for that fixed initial state $\mathrm{seed}(w,\sigma)$ (distinct from an all-states `AvgOracleCostBound` certificate), with uniform average $L^{-1}\sum_{S\in\mathcal{S}} z_S$ equal to the real part of the mean. The proof across all regimes simultaneously is completed by applying `CoupledLimit.of_seed_semantics` directly to these seed costs with clipped density $b = \min(mk/N, 1)$ (where $\arcsin\sqrt{b} = \arcsin\sqrt{mk/N}$ under Lean's total arcsin, and the trivial compression bound $1 - 2\cdot 1 = -1$ when $mk > N$ is provided by Cauchy–Schwarz).

3.4 Dense regime, matching upper bound, and technical remarks

Although all density regimes are already covered uniformly in `Actual.mean_lower_bound` via the clipped density $b = \min(mk/N, 1)$, a case split on $N \le 2mk$ is performed in the certificate entry `Universal.finite` using an independent, more elementary first-query argument (`DenseFindAny`, `FirstQuery`): when every oracle call is replaced by `abort`, a common oracle-independent output is produced whose average success probability over $\mathcal{S}$ is at most k/N, whereby at least $1 - k/N$ first queries are forced, dominating $B(n,m,k)$ throughout $N \le 2mk$ (including $k = N$, where the numerator is zero).

Upon division of $B(n,m,k)$ by $V_{\mathrm{any}}(n,m,k)$,

$$\frac{B(n,m,k)}{V_{\mathrm{any}}(n,m,k)} = \left(\frac{1}{2} + \frac{1}{\pi}\right)\left(1 - \frac{k}{N}\right)\frac{\sqrt{mk/N}}{\arcsin\sqrt{mk/N}} \xrightarrow{n\to\infty} \frac{1}{2} + \frac{1}{\pi}$$

is obtained for any fixed positive m, k, by which `asymptotic_real` and `no_smaller_rational` are proved. For *every* fixed positive integer pair $(m,k) \in \mathbb{N}^2_{>0}$, every real $R > \rho_*$ is realized by adapting the `FindAny` circuit of [1] with positive margin $\delta := R\pi/4 - (\pi/8 + 1/4) > 0$: for $k = 2^n$ the circuit outputs 0 in zero queries; for $k < 2^n$ it repeats `naiveTrial` below a cutoff and invokes the eigenphase-locking circuit `eigLockCirc` once $n \ge \max(150 + \lceil\log_2 m\rceil + \lceil\log_2 k\rceil, n_*(m,k,\delta))$ with $2^n \ge (5/\delta)^2 mk$, achieving relative expected cost at most R eventually on all admissible n ($k \le 2^n$). Because the internal correctness and margin proofs in `FindAny.lean` use grid membership solely to extract $1 \le m$ and $1 \le k$, replacing the grid guard by explicit positivity hypotheses (`P1Upgrade.FindAnyAllPos.correctness_of_pos`, `rel_cost_of_margin_of_pos`) yields `P1Upgrade.AllPositiveInfimum.arbitrarily_close_all_pos` and, together with `asymptotic_real`, `exact_infimum_all_pos` across all $m, k > 0$ (while `Track2Grover.FindAnyLowerBound.exact_infimum` retains the 121-cell restriction matching the frozen benchmark JSON).

Proposition 3.3 (Sharpened finite-n lower bounds and one-round floor; Tier A)**.** *Under all hypotheses of Theorem 3.1—in particular, for every adaptive* `OracleDynamicCircuit` C *satisfying the probability-one correctness and halting specification* $\mathcal{H}(C,k)$ *across all* $S \in \mathcal{S}$*—both finite-n strengthenings are formally proved in Lean 4.35.0 (*`P1Upgrade.FiniteSharpLower`*, 713 lines, 21 exported theorems; all five checks exit 0) across all density regimes without assuming* $mk \le N$*:*

1. (`actual_expected_rounds_sharp`, `finite_sharp`, `finite_half`) *For all* $0 < m$, $0 < k$, $k \le N$,

$$\mathbb{E}_{S,\mathrm{exec}}[T_{\mathrm{rnd}}] \;\ge\; \frac{\pi/4 + 1/2 - f(k/N)}{2\arcsin\sqrt{mk/N}} \;\ge\; B(n,m,k),$$

which simplifies for $2k \le N$ *to* $(\pi/4 + 1/2 - k/N)/(2\arcsin\sqrt{mk/N})$*, whose numerator strictly exceeds the numerator* $c(1 - k/N)$ *of Theorem 3.1 by* $(c-1)k/N > 0$*.*

2. (`one_le_expected_rounds_pointwise`, `actual_expected_rounds_max`, `finite_max`) *Whenever* $0 < k < N$*, for every initial state, applying* `qSet` *and then the zero-query aborted circuit* `FirstQuery.cut` $\mathcal{C}$ *gives identically zero output trace (*`cut_trace_eq_zero`*), forcing the* pointwise *floor*

$$\text{expectedCost}(\mathcal{C}_S, P_S, \text{seed}(w, \sigma)) \;\geq\; 1 \qquad (\forall S \in \mathcal{S}),$$

and hence the combined bound

$$\max\left(1,\; \frac{\pi/4 + 1/2 - f(k/N)}{2\arcsin\sqrt{mk/N}}\right)$$

on both the uniform mean of actual expected rounds and every `AvgOracleCostBound` *certificate* q*.*

Remark 3.4 (Technical note on `TighterFindAny`). With the physical circuit of [1] left unchanged, the submitted rational constant is tightened in the companion module `TighterFindAny` from 514159265358979323847/628318530717958647692 to

$$\frac{257079632679489661923}{314159265358979323846} = \frac{1}{2} + \frac{1}{3.14159265358979323846},$$

a decrease of $1/628318530717958647692 \approx 1.5915 \times 10^{-21}$ lying 2.67831×10^{-22} above ρ_*. This is purely a formal certificate adjustment at the precision ceiling of Mathlib's `Real.pi_gt_d20`, not an algorithmic advance.

4 Supplementary Construction I: Reversible fixed-point arcsine at spacetime cost 366,200

In the reference circuit for `ArcsinK25ConstCost` in [1], 3,698 T gates are used on 100 logical qubits (50 data wires and 50 clean ancillas, TOFF = 0). Through two targeted rewrites of its reversible Boolean operation tables, the count is reduced to 3,662 T gates while the computed lookup table, phase coherence, clean-ancilla return, and 100-qubit width are preserved, whereby the frozen spacetime metric TQ is lowered by 3,600 units ($-36\,T$, -0.9735%) to 366,200.

4.1 Specification, exact compiled gate counts, and non-vacuity

In the frozen specification (`QuantumComputationSpecs.Primitives.Arcsin`), both the register bit-width $p.k \in \mathbb{N}$ and the maximum absolute error tolerance $p.\varepsilon \in \mathbb{R}$ are bundled in the record `p : ArcsinPrec`. Under the frozen precision hypotheses $3 \leq p.k$ and $2^{-(p.k-1)} \leq p.\varepsilon$, each $(p.k)$-bit word is encoded in signed two's-complement Q2.$(p.k-2)$ format via $\text{dec}(u) = \text{signed}(u)\, 2^{-(p.k-2)}$, with $2^{p.k-1}+1$ valid inputs satisfying $|\,\text{dec}(u)| \leq 1$. By the general parameterized theorem `arcsin_correctness`, it is certified that for every `p : ArcsinPrec` satisfying $3 \leq p.k$ and $2^{-(p.k-1)} \leq p.\varepsilon$, there exists an output table $z : \text{Fin}(2^{p.k}) \to \text{Fin}(2^{p.k})$ obeying $|\,\text{dec}(z(u)) - \arcsin(\text{dec}(u))| \leq p.\varepsilon$ on all $2^{p.k-1}+1$ valid inputs such that the coherent lookup-addition unitary

$$|u\rangle|b\rangle|0^{\texttt{arcsinAncillas}(p)}\rangle \;\longmapsto\; |u\rangle|(b + z(u)) \bmod 2^{p.k}\rangle|0^{\texttt{arcsinAncillas}(p)}\rangle$$

is realized by `arcsinProgram p` with clean ancilla return on every disjoint pair of data and ancilla registers (`RealizesCleanSubroutine`). Because the program is parameterized by the full record p, outside the benchmark regime ($p.k = 25 \wedge 1.25 \times 10^{-7} \leq p.\varepsilon$) an exact half-ULP rounded fallback table is selected, whereas at the benchmark instance $p =$ `ArcsinPrec.k25` ($p.k = 25$, $p.\varepsilon = 1.25 \times 10^{-7}$, Q2.23 format, 16,777,217 valid inputs, `arcsinAncillas .k25` = 50) the optimized 100-qubit circuit `k25DynCircuit` with table `z25` is selected.

Theorem 4.1 (Benchmark correctness, spacetime bound, exact logical counts, and non-vacuity at `ArcsinPrec.k25`)**.** *At the benchmark specialization* $p =$ `ArcsinPrec.k25` *(*$p.k = 25$*,* $p.\varepsilon = 1.25 \times 10^{-7}$*, Q2.23, 50 data qubits and 50 clean ancillas):*

1. (`arcsin_correctness .k25`) *In the proof term of the existential theorem* `arcsin_correctness` *instantiated at* $p =$ `ArcsinPrec.k25`*, the output table is witnessed by the concrete table* $z =$ `z25`*, for which* $|\operatorname{dec}(z(u)) - \arcsin(\operatorname{dec}(u))| \le 1.25 \times 10^{-7}$ *on all* 16,777,217 *valid inputs is independently proved by* `z25_satisfies_abs_error`*, and the coherent lookup-addition unitary* $|u\rangle|b\rangle|0^{50}\rangle \mapsto |u\rangle|(b + z(u)) \bmod 2^{25}\rangle|0^{50}\rangle$ *is proved to be realized by* `arcsinProgram .k25` *with clean ancilla return on every disjoint register placement (*`RealizesCleanSubroutine`*).*

2. (`arcsin_k25_const_cost`) *Across every register placement (including overlapping placements), zero primitive* `TOFF` *gates and a spacetime cost of at most* 366,200 *are certified for* `arcsinProgram .k25`*.*

3. (`ExactAudit.exactPublicK25`) *On every disjoint wire placement satisfying* `Disjoint Wires`*, the compiled logical gate counts and spacetime cost of* `arcsinProgram .k25` *are proved to be exact:*

$$T = 3{,}662, \qquad \mathrm{TOFF} = 0, \qquad TQ = 3{,}662(50 + 50) = 366{,}200.$$

4. (`ExactAudit.exactPublicK25_nonvacuous`) *The predicate* `DisjointWires` *and the exact compiled counts above are attained by the canonical contiguous placement of data wires* 0–49 *and ancilla wires* 50–99 *on* 100 *qubits (*`QRegister.block`*,* `DisjointWitness.lean`*).*

Compiled logical T gates are priced by these counts in the frozen Clifford+T+measurement model (not fault-tolerant physical costs after quantum error correction). Because zero weight is assigned to Clifford gates in the frozen metric, total `Op102` instructions increase from 19,705 to 19,961 (+256); here each `Op102` entry is a composite macro instruction (such as an `and`/`unand`, `unquad`, or multi-qubit arithmetic helper) that expands into multiple elementary Clifford, T, and measurement gates. An independent deterministic parser over all 16 Lean source files (`p1_08_gate_counts_receipt.json`, Tier C) establishes the exact elementary gate breakdown: T drops 3,698 → 3,662 (−36), measurements drop $M = 834 \to 831$ (−3), unconditional native two-qubit gates change by +267 (32,588 → 32,919 `CNOT`, 474 → 410 `CZ`), and conditional `CZ` fixups rise by +452 (13,297 → 13,749, 100% inside `s4_20ps`, where replacing 32 `unand` erasures by 29 `unquad` erasures raises quadratic `CZ` pairs 68 → 552, net $+484 - 32 = +452$).

To illustrate the architecture-dependent physical trade-off transparently (**Tier C parametric estimate**, `run_p1_08_ft_estimate.py`), we evaluate a standard rotated surface-code spacetime-volume model: logical error per code cycle $p_L(d) = 0.1(100p)^{(d+1)/2}$ [15, Eq. (7)], smallest odd distance d satisfying the idealized data-qubit logical error budget $1.5\,Q\,C_{\mathrm{cyc}}\,p_L(d) \le \varepsilon = 0.01$ ($Q = 100$, routing overhead 1.5×, data footprint $Q_{\mathrm{data}} = 300d^2$; excluding internal magic-state factory failures and distillation-stall overhead), one 15-to-1 magic-state factory $(d_f, t_{\mathrm{fac}}, Q_{\mathrm{fac}}) = (9, 99, 2{,}430)$ at $p = 10^{-3}$ and $(5, 55, 750)$ at $p = 10^{-4}$ (output error $\approx 35p^3$), and volume $V = (Q_{\mathrm{data}} + Q_{\mathrm{fac}})C_{\mathrm{cyc}}$. Under Clifford-frame-tracked Pauli-based computation ($C_{\mathrm{cyc}} = T\max(t_{\mathrm{fac}}, d) + Md$) or parallel native-`CZ` lattice surgery ($2d$ code cycles per two-qubit gate, 50% expected conditional firing $N_{2q}^{\mathrm{exp}} = 39{,}710.5 \to 40{,}203.5$, $C_{\mathrm{cyc}} = \max(Tt_{\mathrm{fac}}, 2dN_{2q}^{\mathrm{exp}}/\mathrm{par}) + Md$ with par ≥ 4), d is unchanged (17 at $p = 10^{-3}$, 9 at $p = 10^{-4}$) and the $-36\,T$ saving yields $\Delta V = \mathbf{-0.951}\%$ (at 10^{-3}) and $\mathbf{-0.952}\%$ (at 10^{-4}). Conversely, under strictly serial native-`CZ` lattice surgery (par = 1, $C_{\mathrm{cyc}} = Tt_{\mathrm{fac}} + 2dN_{2q}^{\mathrm{exp}} + Md$), $d = 19$ at 10^{-3} and $d = 9$ at 10^{-4}, and the extra conditional `CZ` gates reverse the gain to $+\mathbf{0.80}\%$ ($p = 10^{-3}$) and $+\mathbf{0.74}\%$ ($p = 10^{-4}$); under pessimistic serial `CZ` → 2 `CNOT` lowering ($N_{2q}^{\mathrm{exp}} = 46{,}833 \to 47{,}488$; worst-case 100% conditional firing 60,130 → 61,237), the change is $+\mathbf{0.98}\%$ expected / $+\mathbf{1.44}\%$ worst-case at $p = 10^{-3}$ ($d = 19$) and $+\mathbf{0.93}\%$ expected / $+\mathbf{1.39}\%$ worst-case at $p = 10^{-4}$ ($d = 9$).

4.2 Polar-matrix factorization in `s3a0ps` and product-basis change in `s4_20ps`

Standard $\mathbb{F}_2$ quadratic-form canonicalization and multiplicative-complexity techniques [10, 11, 12] are adapted to the live-wire spans, ancilla constraints, and phase-coherence obligations of the arcsine pipeline:

Coefficient loader `s3a0ps` ($496 \to 472\,T$, $-24\,T$). Before `s3a0ps`, the 440 reachable dyadic magnitude cubes are partitioned into 28 mutually exclusive coarse classes keyed by the five address wires $45, 46, 47, 48, 58$ (27 classes of 16 states and 1 class of 8 states, constructed by `group_stats` in the archived script `partitions.py`), by which a one-hot class-indicator vector $B = (B_1, \ldots, B_{28}) \in \mathbb{F}_2^{28}$ is defined with the disjointness relations

$$B_i B_j = 0 \quad (i \neq j), \qquad B_j^2 = B_j, \qquad \sum_{j=1}^{28} B_j = 1$$

satisfied on every reachable state. At twelve Hadamard-gadget writes (target wires 51, 55, 66, 53, 50, 52, 54, 56, 62, 63, 64, and 65 in circuit order), each target bit is fitted over $\mathbb{F}_2$ as $\phi = \ell + q(\xi) + \xi^\top \beta(B)$ on twelve live-wire parities $\xi \in \mathbb{F}_2^{12}$, where ℓ is affine in the live wires and the 28 class indicators, each component of $\beta(B)$ lies in the surviving linear subspace of class-indicator sums spanned by the live wires, and the alternating polar matrix A defined by $q(x+y) + q(x) + q(y) = x^\top A y$ (with $q(0) = 0$) is of full rank 12. With $g(B) = A^{-1}\beta(B)$, the shift identity

$$q(\xi + g(B)) = q(\xi) + \xi^\top \beta(B) + q\big(g(B)\big)$$

is satisfied identically over $\mathbb{F}_2$. Crucially, because B is one-hot on reachable states ($B_i B_j = 0$ for $i \neq j$, $B_j^2 = B_j$), every quadratic function of linear combinations of the class indicators—in particular $q(g(B))$—is reduced on reachable states to a linear combination of $B_1, \ldots, B_{28}$; furthermore, in `closed_forms.py`, each emitted gadget is checked directly to verify that the resulting residual lies in the live affine wire span and matches the original target bit across all reachable states. By symplectic diagonalization, $q(\xi + g(B))$ is decomposed into six parity products up to an affine correction; because $1\,T$ is charged for each positive or negative parity step in an `hGadget`, $24\,T$ are saved when each of the twelve Hadamard gadgets is reduced from $8{+}8$ steps ($16\,T$) to $7{+}7$ steps ($14\,T$). Because coherent phase cancellation across superpositions is not guaranteed by Boolean agreement alone, the residual sign across all eight slice trees is computed and eliminated via a consolidated zero-T-weight schedule of 22 Z and 178 CZ gates (200 corrections).

Forward 9-bit square `s4_20ps` ($172 \to 160\,T$, $-12\,T$). In the frozen truncated 9-bit square lookup, full-adder carries are given over $\mathbb{F}_2$ by $\mathrm{maj}(a, c_1, c_2) = c_2 + (a + c_2)(c_1 + c_2)$. Through evolutionary column-compression search, a 42-product graph was identified whose subsequent gates depend, modulo linear input terms, only on a 22-dimensional subspace of its first 24 product generators. When those generators are replaced by a rank-one product basis over the nine input bits and column compression is rescheduled within the 25 local scratch wires, a 347-operation table is obtained using 40 paid `and4t` gates (down from 43) and 31 quadratic erasures, while the 21-AND reverse square `s5_2CoreOps` ($84\,T$) is left unchanged and exact agreement on all 100 wires is verified across all 1,024 input/flag combinations.

4.3 Patch scope, stage accounting, and local tightness

Only 5 of the 16 arcsine modules are modified relative to the pinned `discoveries` branch (Table 2); the remaining 11 modules, the core library, and the frozen specification are byte-identical. All 32 stage boundaries are listed in Table 3; in the 449-case and 4,705-case packed traces, a peak of 50 live ancillas along the execution trace is reached at eight distinct stages

(`s3a`, `s3bAdd`, `s3bPost`, `s4_2`, `s4_3`, `s5_1`, `s5_2Core`, `s5_3`), showing why coordinated multi-stage rescheduling is required to reduce width below 100 qubits.

Modified arcsine module	Change relative to the pinned `discoveries` branch
`Arcsin`	Public numeral 369,800 → 366,200 and final multiplication bound.
`StageTablesA`	Forward square table `s4_20ps` (43 → 40 paid ANDs).
`StageTablesB`	Six sequences implementing the `s3aOps` gadget and phase rewrite.
`StageCircuitB`	Updated local and aggregate AND/T count certificates.
`StageProofB`	Aggregate AND count 776 → 773 and compiled T-bound 3,698 → 3,662.

Table 2: The five modified arcsine modules; the remaining 11 of 16 arcsine modules are byte-identical to upstream.

External accuracy enclosure and restricted lower bounds (Tier C). Independently of the universal Lean accuracy theorem `z25_satisfies_abs_error`, an exhaustive 128-bit Arb pass was run over all 8,388,609 magnitudes (extended by exact oddness to all 16,777,217 signed inputs), in which 16 candidates above 1.04304543162 ULP were screened and the top two were refined at 256-bit precision. Bound to the final candidate `StageProofB.lean`, worst-case magnitude $u = 3{,}932{,}278$ (output integer 4,092,728) is identified with error 1.0485006779846463 ULP against the 1.048576 ULP limit (margin 7.5322×10^{-5} ULP).

Surrounding stages are further characterized by two restricted Tier (C) local tightness analyses and one bounded MILP optimization screen (detailed in Appendix C, and explicitly distinguished from global circuit lower bounds): (i) algebraic degree 18 on wires $50, 55$ of `s2a` and $72, 81$ of `s6c` is established by an exhaustive 23-bit Boolean Möbius transform, from which, in the fixed-interface bit-trace model of Appendix C, a forward lower bound of $4(18-1) = 68\,T$ is deduced for `s2a` and, via the reverse-direction affine-rank bound $\deg \le 1 + 12 + \nu$ over the 12 added garbage coordinates, a lower bound of $20\,T$ is deduced for `s6c` (both attained tightly by the existing tables); (ii) when 17 reference products are held fixed in the reverse square `s5_2CoreOps` (spanning a 35-dimensional fixed subspace together with the 9 input and 9 output bits) and intermediates are restricted to the 39-dimensional total reference input/output/product span, 66 proper subspaces of the 4-dimensional quotient are screened down to a single eligible hyperplane H_0 on which an algebraic contradiction across all 7 nonzero directions is certified by 4-index Pfaffian parity witnesses (4,554–5,838 Pfaffians checked across 18,146 weight-≤ 2 assignments per direction), proving 21 ANDs necessary within that restricted span; and (iii) in bounded MILP screens of the late-adder carry recurrence (`scipy.optimize.milp` with the HiGHS backend, relative gap 0, 10 s limit) with 39 scratch slots across 18 placements of the 8 helper products in 3–6 zero-return bursts, solver infeasibility below 166 ANDs is reported (SciPy return code 2, HiGHS model status 8; solver-reported infeasibility without an independently verified proof certificate, paired with a 200-slot positive control in which a checked 133-AND witness is found). Unrestricted circuit minimality and universal qubit lower bounds remain open.

5 Supplementary Construction II: Parallel GroverFindAll at $(1024, 512)$

In `GroverFindAll`, the entire k-element marked set S must be encoded in the output with probability one for every admissible $n \ge 1$, $k \le 2^n$, and $S \in \mathcal{S}$. Exactly one cell, $(m, k) = (1024, 512)$, is claimed by both constructions below, for which an eventual-in-n uniform-average

Stage	Ref. T	New T	Ancillas	Stage	Ref. T	New T	Ancillas
`s1B`	0	0	0/0/0	`s4_3`	392	392	34/50/42
`s1Y`	92	92	0/23/0	`s4_4`	200	200	42/49/42
`s1Add`	96	96	0/24/0	`s5_1`	360	360	42/50/34
`hop_0`	0	0	0/1/1	`s5_2Core`	84	84	34/50/25
`s2a`	68	68	1/18/13	`hop_4`	0	0	25/26/25
`s2b`	188	188	13/14/13	`s5_2Post`	64	64	25/42/12
`s2c`	0	0	13/13/3	`hop_5`	0	0	12/12/12
`s3a`	496	472	3/50/28	`s5_3`	664	664	12/50/18
`s3bAdd`	104	104	28/50/28	`hop_6`	0	0	18/18/18
`s3bPost`	294	294	28/50/27	`s5_4`	76	76	18/40/14
`hop_1`	0	0	27/28/27	`s6a`	0	0	14/21/13
`s4_1`	48	48	27/34/28	`s6b`	188	188	13/14/13
`hop_2`	0	0	28/28/28	`s6c`	20	20	13/18/1
`s4_1Post`	0	0	28/28/25	`hop_7`	0	0	1/1/0
`hop_3`	0	0	25/26/25	`s7Y`	92	92	0/23/0
`s4_2`	172	160	25/50/34	`s1B`	0	0	0/0/0

Table 3: Stage-by-stage T-gate counts and start/peak/end packed ancilla counts in the 449-case trace (with the suffix `Ops` omitted; `s1B` is executed twice). Column sums are 3,698 (reference) and 3,662 (present work).

cost upper bound

$$\exists n_0,\ \forall n \geq n_0, \quad k \leq 2^n \implies \mathrm{AvgOracleCostBound}\big(k, \mathcal{C}, P, r\, V_{\mathrm{all}}(n,m,k)\big)$$

is proved, where $V_{\mathrm{all}}(n,m,k) = \frac{\pi}{4}\big(\sum_{\ell=1}^{k} \ell^{-1/2}\big)\sqrt{2^n/m}$. Machine-checked asymptotic upper bounds are compared through the certified rationals r given below; because distinct proofs may carry different slack, the ratio of two upper bounds does not automatically equal the exact ratio of true expected costs.

5.1 Sign-only Johnson walk and coherent full-occupancy Kraus filters

Both constructions are built on a width-one collective Johnson walk over subsets $E \subset [N]$ of cardinality $|E| = \ell_s m = 1024$ (where "width-one" denotes block multiplicity $\ell_s = 1$, i.e. a single m-slot register block per walk step, matching `Rkm128kSeg.lS = 1`), grouped into the 513 symmetric orbits indexed by occupancy $j = |E \cap S| \in \{0, \ldots, 512\}$. On this 513-dimensional orbital subspace, one discrete reflection-diffusion step $D_{\mathrm{alt}} Z_S$ (where Z_S is the marked-set phase reflection and D_{alt} is the symmetric Johnson diffusion) is represented by the orthogonal orbital matrix U_N (`dAltStepOrbMat (2^n) 1024 512`). Matching the exact upstream sign convention of `tridiagSkewMat` and `dAltGenMat` in `FindAllJohnsonAltStep.lean`, the limiting continuous-time skew-symmetric tridiagonal generator G_{walk} as $N \to \infty$ is defined by superdiagonal entries $(G_{\mathrm{walk}})_{j,j+1} = -b_j$ and subdiagonal entries $(G_{\mathrm{walk}})_{j+1,j} = +b_j$, where

$$b_j = \sqrt{4(j+1)(m-j)(k-j)} \qquad (0 \leq j < k = 512,\ m = 1024).$$

(In the independent limiting script `enclose.py`, the opposite sign convention $-G_{\mathrm{walk}} = J_{\mathrm{par}} G_{\mathrm{walk}} J_{\mathrm{par}}$ is used, conjugated by $J_{\mathrm{par}} = \mathrm{diag}((-1)^j)$; because every diagonal sign mask D_t and Kraus operator $K_{0,t}$ is invariant under conjugation by J_{par} and e_0 is fixed by J_{par}, all surviving branch norms, click probabilities, and expected costs are identical under either convention.) For segment $t \in \{1, \ldots, 24\}$ with dimensionless normalized duration $s_t = \tau_t \cdot 10^{-9}$ and Lean macro-substep count nSub_t, exactly $M_t(N) = \mathrm{nSub}_t \lfloor (\tau_t \lfloor \sqrt{N} \rfloor)/(10^9\, \mathrm{nSub}_t) \rfloor$ discrete walk steps are executed (`altStepCount`; Appendix A), so that the finite-N orbital propagator $U_N^{M_t(N)}$ converges to $W_t = \exp(s_t G_{\mathrm{walk}})$ as $N \to \infty$.

Sign-only refinement (`Track2Grover.FindAll`). Through a re-optimized 24-segment schedule (Appendix A) certified in Lean via 432 degree-28 Taylor substeps across 122 kernel-checked checkpoints (using 17 semantic engine modules adapted from DeepMind's `discoveries` branch with original Google LLC headers retained), the rational upper bound

$$r_{\text{sign}} = \frac{27{,}014{,}193{,}664}{10^{12}} = \frac{52{,}762{,}097}{1{,}953{,}125{,}000}$$

is proved, whereby the pinned discovery's bound $r_{\text{ref}} = 6{,}756{,}116{,}497/250{,}000{,}000{,}000 = 27{,}024{,}465{,}988/10^{12}$ is lowered by 0.038011%.

Extension with coherent full-occupancy Kraus filters (`Track2Weak.FindAllWeak`). To avoid spending late walk segments on amplitude that has already reached full occupancy $j = 512$, the sign-only walk is augmented with an ancilla flag and 14 active weak-measurement filters across the 24 stages (developed from weak-measurement search primitives [8, 9]). The archived weak-filter attenuation parameter stored for segment t is denoted by $\widetilde{a}_t \in [0, 1]$ (tabulated in Appendix A: $\widetilde{a}_t = 1$ for $t \le 9$ and $t = 24$, with 14 nontrivial filters $\widetilde{a}_t \in (0, 1)$ at $t = 10$–23). When the separate terminal strong readout after segment 24 is incorporated into a unified recurrence, the effective stage attenuation parameter is defined by

$$a_t = \begin{cases} \widetilde{a}_t, & 1 \le t \le 23, \\ 0, & t = 24, \end{cases} \qquad K_{0,t} = \operatorname{diag}(1, \ldots, 1, a_t), \qquad K_{1,t} = \sqrt{1 - a_t^2}\, |512\rangle\langle 512|,$$

satisfying $K_{0,t}^\dagger K_{0,t} + K_{1,t}^\dagger K_{1,t} = I$ for all $t = 1, \ldots, 24$. Physically, at each active probe, the 1024 marker bits are computed by one parallel oracle call, the flag is rotated by an S-independent unitary conditioned on the number of *distinct* marked addresses equaling 512 while the diagonal sign mask D_t is applied, the flag is measured, and the marker bits are uncomputed by a second oracle call. Upon a click ($K_{1,t}$), the state is projected strictly onto $E \supseteq S$ and immediate full readout is triggered; upon a no-click ($K_{0,t}$), coherence across all bins is preserved with only the $|512\rangle$ amplitude attenuated.

In the actual finite-N circuit, the uniform superposition of m-element subsets is represented in the orbital basis by u_N (`u0OrbVecGen (2^n) 1024 512`) with entries $u_{N,j} = \sqrt{\binom{k}{j}\binom{N-k}{m-j}/\binom{N}{m}}$, obeying $\|u_N - e_0\|_2 \le 5 \times 10^{-16} \le 10^{-15}$ for all $n \ge 80{,}000$ (`vecNorm_u0OrbVecGen_sub_e0_le`, `initialError`). The unnormalized continuing-branch orbital state in the finite-N circuit is propagated according to

$$x_0^{(N)} = u_N, \qquad x_t^{(N)} = K_{0,t} D_t U_N^{M_t(N)} x_{t-1}^{(N)}, \qquad p_t^{(N)} = (1 - a_t^2) \left| \langle 512 \mid D_t U_N^{M_t(N)} x_{t-1}^{(N)} \rangle \right|^2,$$

whereas in the $N \to \infty$ limiting model, u_N and $U_N^{M_t(N)}$ are replaced by $x_0 = e_0$ and $W_t = \exp(s_t G_{\text{walk}})$. In both models, segment t's walk cost is weighted by the surviving branch mass $\|x_{t-1}^{(N)}\|^2$, and total single-attempt success probability is given by the sum of all click probabilities $p^{(N)} = \sum_{t=1}^{24} p_t^{(N)}$ (including the terminal strong read $p_{24}^{(N)}$ at $a_{24} = 0$). Crucially, because the collective Johnson-walk orbit is permutation-symmetric in the hidden set S, the same finite-N orbital vector recurrence and uniform per-set bounds ($p_*, w_*\sqrt{N}$+24,673) are satisfied identically for every $S \in \mathcal{S}$ (`BatchBound.perSet`); thus no invalid interchange of ratio and expectation is introduced when Θ_S/p_S is averaged over $S \in \mathcal{S}$.

In `WeakAssembly`, the first attempt is executed unconditionally and independent full restarts upon an incomplete terminal read are controlled by a measured `qWhile` loop, whereby total expected cost $\le \Theta_S/p_S$ (`restart_cost`) is proved from per-attempt expected cost Θ_S and success probability $p_S > 0$. Within each attempt, survival-weighted branch costs are accounted for in `WeakAssemblyCost`, `WeakAssemblyStages`, and `WeakAssemblyTrialCost`, with fixed per-attempt probe and readout overhead bounded by $24 \cdot 1028 + 1 = 24{,}673$ oracle rounds. In Lean,

the deviation $\|u_N - e_0\|_2$, the finite-N operator discretization error between $U_N^{M_t(N)}$ and W_t, and the degree-28 Taylor truncation and rounding errors are all bounded cumulatively across 163 macro-substeps at scale 10^{15} (`WeakCertFlow.lean`, `WeakCertRun.lean`, distinct from the 432 degree-28 substeps of the sign-only certificate and from the independent 915-substep order-24 limiting propagation in `enclose.py`). The exact Lean numerical certificate and external limiting enclosures are collected in Table 4, certifying via `Track2Weak.FindAllWeak` that

$$r_{\text{weak}} = \frac{26{,}768{,}771{,}052}{10^{12}} = \frac{6{,}692{,}192{,}763}{250{,}000{,}000{,}000},$$

by which the pinned discovery's certified upper bound is reduced by 0.946% (and r_{sign} by 0.908%).

Certified quantity	Exact value / interval	Source
Uniform single-attempt success p_S	$p_S \geq p_* = 999{,}995{,}956{,}520/10^{12}$	`Cert.success12` (A)
Uniform attempt walk cost Θ_{walk}	$\Theta_{\text{walk}} \leq w_*\sqrt{N}$, $w_* = 28{,}787{,}568{,}698{,}601{,}335{,}334/10^{21}$	`Cert.weightedCost21` (A)
Fixed per-attempt probe/read overhead	$\leq 24 \cdot 1028 + 1 = 24{,}673$ oracle rounds	`WeakAssemblyStages` (A)
Final L^2 orbital error pad (seg. 24)	$\leq 35{,}144{,}918/10^{15} \approx 3.5145 \times 10^{-8}$	`row23.walkPad` (A)
Restart ratio bound $C_*\sqrt{N}$ ($n \geq 80{,}000$)	$C_* = 28{,}787{,}685{,}101{,}031/10^{15}$	`Cert.retryUpper15` (A)
Restart gap and $\sqrt{N}$-scale slack	$C_* - w_*/p_* \approx 7.12 \times 10^{-16}$; $r_{\text{weak}}A - C_* \approx 3.37 \times 10^{-13}$	`FiniteFromCutoff` (A)
Limiting retry cost (`enclose.py`, ord. 24)	$[0.02878768025617972696,$ $0.02878768025617972697]\sqrt{N}$	915 substeps (C)
Limiting archived-cont. cost (`enclose.py`)	$[0.02878765476018333366,$ $0.02878765476018333369]\sqrt{N}$	915 substeps (C)

Table 4: Exact numerical certificate for `Track2Weak.FindAllWeak` at $(1024, 512)$, where $A = V_{\text{all}}(n, 1024, 512)/\sqrt{N} = (\pi/128)\sum_{\ell=1}^{512} \ell^{-1/2} \approx 1.0754205$. All Tier (A) bounds hold uniformly across every $S \in \mathcal{S}$ by orbit symmetry and are certified in Lean via 163 degree-28 Taylor substeps (`WeakCertRun.lean`, `WeakCertBudget.lean`). Tier (C) intervals are produced by independent order-24 rational propagation at scale 10^{26} (`enclose.py`, `refine_placement0_enclosure.json`; full 42-digit endpoints archived).

Finite cutoff and asymptotic regime ($n \geq 80{,}000$ inherited engine threshold). The $n \geq 80{,}000$ activation threshold (`altN0 := 80000`) is inherited directly from the finite-N Johnson-walk error lemmas of the upstream-derived engine (`FindAllJohnsonAltStep.lean` and `Rk128kEngine.lean`), so $n < 80{,}000$ is routed to the exact `Stage1` fallback across all three walks (the pinned discovery, the sign-only walk, and `weakChosenBatch`). Although the public existential theorem `groverFindAll_rel_cost` in `FindAllWeak.lean` invokes the crude proof witness $n_0 = \max(\lceil(10^{16} \cdot 24{,}673)^2\rceil, 80{,}000)$ constructed in `WeakAssemblyCell.lean` (`normalized_batch_cost` and `routedBatch_rel_cost`), absorbing the fixed overhead 24,673 inside the 7.12×10^{-16} restart rounding margin requires only $(10^{16} \cdot 24{,}673)^2 \leq 2^{140} \leq 2^n$, for which $n \geq 140$ suffices (`FiniteFromCutoff.finite_normalized`). Consequently, the claimed ratio is established uniformly for every $n \geq 80{,}000$ by the companion theorem:

Proposition 5.1 (Uniform weak-filter bound for all $n \geq 80{,}000$; Tier A)**.** *By the companion theorem* `QuantumComputation.Specs.groverFindAll_cost_from_80000` *(*`FiniteFromCutoff.lean`*, verified by the historical* `v1.0.0` *111-module* `leanchecker` *replay and, after*

the `v1.0.1 abbrev` *fix, by post-fix Windows* `leanchecker` *and Linux build/*`#print axioms` *checks (Appendix D)), for every* $n \in \mathbb{N}$ *satisfying* $80{,}000 \le n$,

$$\mathrm{AvgOracleCostBound}\Big(512,\ S \mapsto \mathcal{C}_S,\ S \mapsto P^{\mathrm{weak}}_{n,S},\ r_{\mathrm{weak}}\, V_{\mathrm{all}}(n, 1024, 512)\Big)$$

is proved, where $P^{\mathrm{weak}}_{n,S}$ *denotes* `groverFindAllProgram` $n\,1024\,512$ *applied to* `groverParallel MarkingOracle` $1024\,S$ *and* $r_{\mathrm{weak}} = 26{,}768{,}771{,}052/10^{12}$.

Because $n = 80{,}000$ corresponds to a search space of $N = 2^{80{,}000} \approx 2.5 \times 10^{24{,}082}$ items and $1024(80{,}000+1) = 81{,}921{,}024$ logical qubits are already required for the oracle slots alone before additional workspace, these Johnson-walk protocols are asymptotically motivated constructions with finite guarantees above an astronomically large cutoff, by which no practical speedup at realistic problem sizes is yielded.

5.2 External GR0 comparison and unformalized limiting candidate

Three logically distinct claims must be kept separate: (i) probability-one correctness for all admissible $n \ge 1$ via the exact `Stage1` fallback below 80,000; (ii) the uniform finite-n relative-cost theorem relative to $V_{\mathrm{all}}(n, 1024, 512)$ for every $n \ge 80{,}000$ (Proposition 5.1); and (iii) asymptotic coefficient comparison against the external GR0 baseline. While the primary machine-checked comparison is directly between the exact rationals r_{ref}, r_{sign}, and r_{weak}, in an external 256-bit Arb evaluation of the GR0 cap model its asymptotic relative coefficient g_{GR0} is enclosed in [0.06957627832198676185, 0.06957627832198676734] (Tier C; the reference schedule was not independently re-derived from [1], Appendix E). When the lower endpoint $g_{\mathrm{GR0},-}$ is divided by the positive Lean upper bounds, conservative *asymptotic* coefficient ratios of at least 2.57554525548× (sign-only) and 2.599158481606× (weak filters, +0.955% over the reference 2.574566× via $r_{\mathrm{ref}}/r_{\mathrm{weak}} - 1$) are obtained; no finite-n lower bound on the actual cost of GR0 is claimed. Outward-rounded upper bounds on $10^{12}r$ from two unformalized limiting numerical enclosures (`refine_placement0_enclosure.json`) are also recorded in Table 5; in the archived-continuation policy, partial terminal reads are retained without a finite-circuit Lean proof.

Construction and evidence	$10^{12}r$	Asymptotic ext. GR0 factor
Pinned discovery [1], Lean eventual upper bound (suff. large n)	27,024,465,988	$\ge 2.574566\times$
Sign-only walk, Lean eventual upper bound (suff. large n)	27,014,193,664	$\ge 2.57554525548\times$
Weak filters + restart, Lean upper bound (all $n \ge 80{,}000$)	26,768,771,052	$\ge 2.599158481606\times$
Weak filters + restart, `enclose.py` upper bound (Tier C)‡	26,768,766,547	$2.599158919027\times$
Weak filters + archived cont., `enclose.py` upper bound (Tier C)‡	26,768,742,839	$2.599161\times$

Table 5: Single-cell $(1024, 512)$ relative-cost upper bounds and externally derived asymptotic GR0 coefficient ratios ($g_{\mathrm{GR0},-}/r$). Finite-circuit Lean 4.35.0 proofs are carried only by the rows above the rule; all three walks share the `altN0 := 80000` engine activation threshold, while the pinned and sign-only public theorems quantify existentially over sufficiently large n and only the weak-filter companion theorem (Proposition 5.1) explicitly proves every $n \ge 80{,}000$. ‡Outward-rounded upper bounds on $10^{12}r$ (`retry_ratio_rational_n12`, `incumbent_ratio_rational_n12`) derived from the limiting cost intervals of Table 4; not finite-circuit Lean proofs.

6 Verification, reproducibility, and AI provenance

6.1 Verification summary and claim map

All primary proofs compile under pinned Lean 4.35.0 against unmodified copies of `Quantum Computation` (57/57 files) and `QuantumComputationSpecs` (15/15 files), with zero `sorry`, zero `native_decide`, and axioms strictly within {`propext`, `Classical.choice`, `Quot.sound`}. This manuscript pins Lean 4.35.0 specifically to match the frozen upstream `google-deepmind/formal-quantum-computation` toolchain, whereas companion papers in this release series target Lean 4.33.1. Table 6 summarizes all six project-module suites (288 historical `v1.0.1` modules plus four `v1.1.0 P1Upgrade.*` modules = 292 checked modules / 283 staged `.lean` files), and Table 7 maps every headline claim to its exact Lean declaration and archived receipt.

On a clean-machine 64-bit Linux host, all four challenge constructions (`ArcsinK25 ConstCost`, `TighterFindAny`, sign-only `GroverFindAll`, and `v1.0.1` weak-filter `Grover FindAll`) plus upstream `lake test` were accepted by the official `lake comparator` driver inside its `bwrap` sandbox [22]. Independent of Lean's C++ kernel, the flagship `FindAny LowerBound` theorems and `TighterFindAny` were additionally accepted with zero errors by two third-party Rust checkers (`nanoda_bin` [23] and `con-ron -verified` [24]), while all four `v1.1.0 P1Upgrade.*` modules (38 exported targets, 2,478 lines; separately reparsed with strict target-completeness checks via `verify_p1_lean5_fixed.py` in `P1_MULTILINE_AXIOMS_REPARSE.json`, 38/38 clean) passed all five checks (`lean`, `leanchecker`, `con-leche -verified`, `con-ron -verified`, and `nanoda_bin`). Binary/source SHA-256 prefixes, pinned Git commit identifiers (full 64-hex digests in `SHA256SUMS.txt`), file-count reconciliations, external-checker declaration counts, and the `v1.0.0` → `v1.0.1` `ReducibilityHints` patch history are consolidated in Appendix D.

Project-module suite	Modules	`leanchecker`†	Linux `comparator`, third-party kernels & staged files
Track 2B/C: `FindAny` `LowerBound` + `Tighter` `FindAny`	31	31/31 exit 0	Linux `comparator` PASS (`TighterFindAny`); `nanoda_bin` + `con-ron` PASS; 31 staged
Track 3: `Arcsin` (16) + audit + exact + witness	20	20/20 exit 0	Linux `comparator` PASS (verbatim JSON) + `lake test` PASS; 9 proof + 2 helper scripts staged
Track 2A: `FindAll` (sign-only)	117	117/117 exit 0	Linux `comparator` PASS (`bwrap`, 13.14 GiB peak); 117 staged
Track 2W: `Track2Weak` (weak filters)	111	111/111 + post-fix	Linux `comparator` PASS (`v1.0.1` `abbrev` fix, 6.57 GiB peak); post-fix `leanchecker` PASS; 111 staged
Track 1: `Track1THC` (App. B, partial)	9	9/9 exit 0	0/7 def. holes + 0/2 thms; cross-model Codex audit (P1-06); 9 staged
Historical `v1.0.1` subtotal	288	288/288 + **post-fix**	4/4 challenge `comparator` PASS; 279 staged `.lean`; std. axioms only
`v1.1.0` Upgrade: `P1Upgrade.*` (4 modules, 38 tgts)	4	4/4 exit 0	20/20 checks (`lean`, `leanchecker`, `con-leche`, `con-ron`, `nanoda_bin`); 38/38 axioms reparsed; 4 staged
Current `v1.1.3` total	292	292/292	4/4 `comparator` + 4 new 5-check modules; 283 staged `.lean`; std. axioms only

Table 6: Verification inventory across the five historical `v1.0.1` Lean 4.35.0 suites (288 checked modules / 279 staged files) and the four `P1Upgrade.*` modules (38 exported targets, bringing the staged total to 292 checked modules / 283 staged files; excluding Mathlib/Core). †Post-fix replay logs and SHA-256 digests are listed in Appendix D. Standard axioms denote subsets of {`propext`, `Classical.choice`, `Quot.sound`}.

Paper claim	Declaration (with namespace prefix)†	Entry file	Archived receipt (`receipts/`)
Thm. 3.1 (finite & actual rounds)	`FA.finite`, `FA.actual_expected_rounds`	`FindAnyLowerBound.lean`	`track2_grover/findany_replay_manifest.json`, `external_checkers/external_findany_lowerbound.json`
Thm. 3.2 (121-cell baseline)	`FA.asymptotic_real`, `FA.no_smaller_rational`, `FA.exact_infimum`	`FindAnyLowerBound.lean`	`external_checkers/external_findany_lowerbound.json`, `linux_comparator/run1_fast/results/12_findany_leanchecker_LowerBound.log`
Thm. 3.2(3) (all $m, k > 0$; `v1.1.0`)	`P1U.FindAnyAllPos.rel_cost_of_margin_of_pos`, `P1U.AllPositiveInfimum.exact_infimum_all_pos`	`FindAnyAllPos.lean`, `AllPositiveInfimum.lean`	`P1Upgrade_FindAnyAllPos_VERIFICATION.json`, `P1Upgrade_AllPositiveInfimum_VERIFICATION.json` (App. D)
§3 contract bridge (`v1.1.0`)	`P1U.SubmissionBridge.submission_feasible`, `submission_relCost_gt`	`SubmissionBridge.lean`	`P1Upgrade_SubmissionBridge_VERIFICATION.json` (10 tgts, 5/5 checks)
Prop. 3.3 (finite sharpenings & ≥ 1; `v1.1.0`)	`P1U.FiniteSharpLower.actual_expected_rounds_sharp`, `one_le_expected_rounds_pointwise`, `finite_max`	`FiniteSharpLower.lean`	`P1Upgrade_FiniteSharpLower_VERIFICATION.json` (21 tgts, 5/5 checks)
Thm. 4.1(1)–(2) (correctness, 366,200)	`QCS.arcsin_correctness`, `QCS.arcsin_k25_const_cost`	`Primitives/Arcsin.lean`	`linux_comparator/run1_fast/results/05_arcsin_OFFICIAL_lake_comparator.log`
Thm. 4.1(3)–(4) (exact count, witness)	`QCS.ArcsinImpl.ExactAudit.exactPublicK25`, `exactPublicK25_nonvacuous`	`ExactPublic.lean`, `DisjointWitness.lean`	`track3_arcsin/track3_exact_public_leanchecker.json`, `track3_arcsin/track3_disjoint_witness_leanchecker.json`
§5 sign-only (27,014,193,664 $/10^{12}$)	`QCS.groverFindAll_rel_cost` (`Track2Grover` pkg.)	`Track2Grover/FindAll.lean`	`linux_comparator/run2_findall/results_findall/19_findall_signonly_OFFICIAL_lake_comparator.log`
§5 & Prop. 5.1 (weak, $n \geq 80{,}000$)	`QCS.groverFindAll_rel_cost`, `QCS.groverFindAll_cost_from_80000` (`Track2Weak` pkg.)	`FindAllWeak.lean`, `FiniteFromCutoff.lean`	Linux post-fix `comparator`, `#print axioms`, and Windows `postfix_track2weak_leanchecker.json` (exact log paths in App. D)

Table 7: Claim-to-declaration map linking headline Tier (A) results to fully qualified Lean 4.35.0 declarations, entry files, and archived receipts. †Prefixes: `FA` = `Track2Grover.FindAnyLowerBound`, `QCS` = `QuantumComputation.Specs`, `P1U` = `P1Upgrade` (`v1.1.0`). The declaration `QCS.groverFindAll_rel_cost` is declared independently in the two separate packages `Track2Grover` and `Track2Weak`.

6.2 Pure-AI provenance and role separation

In this manuscript, **purely AI-generated research** directed, scoped, and curated by the human author (Robert Kadlec, Masaryk University, Brno) is reported; this paper is part of a broader research project in which the respective competencies of humans and large language models are investigated. All new contributions introduced here are covered by this provenance—as distinct from the unmodified upstream `QuantumComputation` and `QuantumComputationSpecs` libraries and the 32 upstream-derived Google LLC files retaining verbatim Apache-2.0 headers plus one partial module (`Track1THC/LCU.lean`) attributing upstream lemmas in its docstring (33 staged files referencing Google LLC in total, documented in `NOTICE`). No new mathematical proof, circuit table, Lean module, verification script, or paragraph of prose was manually written by the human author, by whom full scholarly responsibility for the integrity, accuracy, and publication of all reported results is assumed. All technical execution, verification, and adversarial refereeing were delegated to AI agents operating under explicit role separation and read-only upstream boundaries:

- *Coordinator and exploratory analyst (Antigravity / Gemini):* environment isolation, initial bit-parallel `Op102` simulation, symplectic `hGadget` decomposition, ancilla liveness profiling, HJB optimality analysis for `GroverFindAny`, dedicated Windows `leanchecker` replays (including post-fix `FindAllWeak/FiniteFromCutoff`), `lean4export/nanoda_bin/con-ron` exports and replays, `DisjointWitness.lean`, surgical patch generation, manuscript assembly, and release packaging.

- *Synthesis and formalization agent (OpenAI Codex CLI v0.171.0, `gpt-6.2-astra`):* GF(2) polar-matrix gadget synthesis, rank-one square compression, `ExactPublic`, the complete formal optimality proof `FindAnyLowerBound`, both `GroverFindAll` Lean implementations (sign-only and coherent weak-measurement Kraus filters), Arb enclosures, and local lower-bound computations.

- *External Linux verification operator (separate* 62 GiB *Ubuntu 26.04.1 LTS host):* execution of the packaged `v1.0.0` bundle through official `lake comparator` (`bwrap` sandbox) across all four challenge constructions and upstream `lake test` (confirming consistency of all seven challenge configurations), diagnosis of the `ReducibilityHints` mismatch on `groverFindAllProgram`, and verification of the one-keyword `abbrev` fix (November 26–27, 2026; detailed in Appendix D).

- *Adversarial refereeing and Appendix B authorship (Anthropic Claude Code CLI v2.1.331–332; OpenAI Codex CLI v0.171.0; OpenAI ChatGPT Web `6 Pro`):* initial read-only audits of Tracks 2 and 3, `Track2Weak`, and the release bundle (never auditing code authored by itself), cross-model exact-fraction and 70-digit numerical recomputations, and formalization of the 9 partial `Track1THC` modules under Claude Code's `sonnet` alias (November 25, 2026; unreviewed at `v1.0.0` release, subsequently audited cross-model by OpenAI Codex in P1-06 on November 27, 2026), followed by multi-round manuscript-vs-Lean cross-audits under `claude-fable-5-2` at `xhigh` effort (excluding a separate exploratory session whose temporary edits were reverted), read-only release audits with independent third-party kernel replay via Codex CLI `gpt-6.2-astra` at `ultra` effort, and cross-model PDF mathematical refereeing via ChatGPT Web `6 Pro` across successive drafts (Rounds 1–4 culminating in sign-off on the 25-page `v7` draft on November 26, 2026, followed by Round 5 audit of the 28-page `v10` draft on November 27, 2026).

For Tier (A) and Tier (B) claims, no unchecked LLM output is accepted as a formal proof: every formal statement is backed by a machine-checked Lean 4.35.0 proof and `leanchecker` replay; additionally, all four challenge constructions carry official Linux `lake comparator` receipts, and the flagship `FindAny` theorems carry independent `nanoda_bin` and `con-ron` receipts. Tier (C)

claims are supported independently by the explicitly identified external Arb and rational Taylor enclosures, finite Boolean Möbius, Pfaffian, and MILP computations, and restricted-model paper arguments. Human peer review has not yet been performed on this preprint. Full disclosure, excluded-file inventories (`EXCLUDED.md`), and detailed audit trails are provided in the supplementary file `AI_DISCLOSURE_AND_PROVENANCE.md`.

6.3 Artifact availability, reproduction, and open problems

Initial release `v1.0.0` (all 279 staged `.lean` files, the surgical 5-file `arcsin_366200.patch` verified via `git apply -check` against the pinned `discoveries` branch, Windows `leanchecker` logs, `nanoda_bin/con-ron` receipts, interface JSONs, archived Arb/Python audit-trail scripts and witnesses including `partitions.py`, `lib.py`, `rank_updates.py`, `followup.py`, `pfaffian_witness.json`, and `recon_input.py`—see `lean/track3_arcsin/scripts/README.md` for remaining workspace path dependencies—`.gitattributes * -text`, and a complete `SHA256SUMS.txt` manifest) is publicly archived on Zenodo (https://doi.org/10.5281/zenodo.24545553; all-versions Concept DOI https://doi.org/10.5281/zenodo.24545552), GitHub (https://github.com/Ikaro2023/quantum-lean-deepmind-benchmarks, tag `v1.0.0`), and Hexagon (`hexagon:2611.01127`, `v1`). Public release `v1.0.1`—publicly archived on Zenodo (https://doi.org/10.5281/zenodo.24584791, live under the same Concept DOI), GitHub (tag `v1.0.1`), and Hexagon (`hexagon:2611.01127`, `v2`)—incorporates the one-keyword `FindAllWeak.lean abbrev` fix, the curated Linux `lake comparator` and `lake test` receipts plus Grover configs (`receipts/linux_comparator/`), the third-party checker wrapper (`receipts/external_checkers/`), and the post-fix `Track2Weak leanchecker` receipts (`receipts/track2_weak/v1_0_1_postfix/`). Staged follow-up release `v1.1.3` extends `v1.0.1` with the four `P1Upgrade.*` modules (38 targets), their 5-check receipts, the strict multiline axiom reparser (`verify_p1_lean5_fixed.py`, `P1_MULTILINE_AXIOMS_REPARSE.json`, 38/38 clean), the offline lower-bound challenge PR package (`P1_03_007_PR_READY/`), the deterministic `Arcsin` gate-count receipt (`p1_08_gate_counts_receipt.json`), and the FT estimate script (`run_p1_08_ft_estimate.py`), staged locally for publication under the same Zenodo Concept DOI and GitHub repository (Apache-2.0 for all Lean sources, patches, and scripts, retaining original Google LLC copyright attributions across all 33 upstream-referencing files as itemized in `NOTICE`; CC BY 4.0 for manuscript and documentation).

Independent reproduction of `ArcsinK25ConstCost` is carried out by checking out `google-deepmind/formal-quantum-computation` at the pinned `discoveries` branch (Appendix D), applying `patches/arcsin_366200.patch`, and running `lake build`, `lake env leanchecker`, the interface script `CompareFrozen.lean` (or upstream `lake comparator`), and the standalone checks for `ExactCount`, `ExactPublic`, and `DisjointWitness`. Three open problems remain directly accessible from this artifact: (i) lowering `ArcsinK25` below 100 logical qubits via cross-stage rescheduling of the eight 50-ancilla peaks; (ii) extending weak-measurement filters across the remaining 120 `GroverFindAll` grid cells, lowering the $n \geq 80{,}000$ engine cutoff, and establishing sharp unconditional `GroverFindAll` query lower bounds to close the remaining gap against the weak-filter upper bounds; and (iii) assembling the centered block encoding and implementing and costing its compiled circuit for `THCFactorization` on top of Appendix B.

Follow-up work. We defer unconditional `GroverFindAll` query lower bounds, spectral `THC` bounds, and lower finite-n thresholds to a companion manuscript.

A Exact parameters of the 24-segment `GroverFindAll` schedules

In both single-cell $(1024, 512)$ schedules of Section 5, 24 width-one ($\ell_s = 1$, $|E| = 1024$) segments are executed on the collective Johnson walk. The dimensionless normalized walk time $s_t =$

$\tau_t \cdot 10^{-9}$—corresponding to the $\sqrt{N}$-scaled continuous walk duration $s_t\sqrt{N} = \tau_t \cdot 10^{-9}\sqrt{N}$—is specified by each integer entry τ_t ($t = 1, \ldots, 24$). In the finite-N Lean circuits, each segment t with macro-substep count nSub_t is converted into an exact integer number of discrete walk iterations by the floor rule `altStepCount` (`FindAllJohnsonAltStep.lean`):

$$M_t(N) \;=\; \mathrm{nSub}_t \left\lfloor \frac{\tau_t \lfloor \sqrt{N} \rfloor}{10^9\, \mathrm{nSub}_t} \right\rfloor .$$

Across the 24 segments, $\sum_{t=1}^{24} \mathrm{nSub}_t = 432$ macro-substeps are used in the Lean degree-28 Taylor certificate for the sign-only proof (`Track2Grover.FindAll`), and $\sum_{t=1}^{24} \mathrm{nSub}_t = 163$ macro-substeps are used in the weak-filter proof (`WeakCertRun.lean`, with $\mathrm{nSub}_1, \ldots, \mathrm{nSub}_{24} = 62, 67, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5$).

After each segment, a 513-bit weight sign mask is applied: in the companion JSON files (`masks_le_hex`), each mask is stored as a 65-byte (130-hex-character) little-endian integer (`int.from_bytes(bytes.fromhex(h), "little")`), where occupancy bin $\beta \in \{0, \ldots, 512\}$ (byte $\lfloor \beta/8 \rfloor$, bit $\beta \bmod 8$) is negated when bit β is set, and the upper seven padding bits 513–519 of byte 64 are unused and zero (in both archived schedules, byte 64 is `0x00`, so bit 512 is also unset throughout). In the weak-filter schedule, each sign mask is followed by a full-occupancy Kraus check whose archived no-click amplitude parameter is $\widetilde{a}_t = \mathrm{num}_t/10^9 \in [0, 1]$ (identity when $\mathrm{num}_t = 10^9$, i.e. segments 1–9 and 24; 14 active nontrivial filters at segments 10–23). Following the final identity weak filter ($\widetilde{a}_{24} = 1$) at segment 24, a separate terminal strong readout is executed, which is represented in the unified recurrence of Section 5 and in `WeakCertRun.lean` (`row23.atten = 0`) by the effective parameter $a_{24} = 0$.

Schedules were discovered by iterative segment splitting and backward-adjoint retiming of the orbital walk objective, followed by greedy binary sign updates and gradient optimization of the survival-weighted measured cost-to-go. Independently of floating-point search, the limiting continuous-time walk quantities were enclosed via outward-rounded exact integer Taylor propagation (`enclose.py`, Taylor order 24, 915 substeps, scale 10^{26}, Tier C) before formalization of the finite-N degree-28 Lean 4.35.0 proofs (Tier A). All 24 durations and all 24 archived attenuation numerators $10^9\widetilde{a}_t$ are listed in Table 8; the full 65-byte hex sign masks are archived verbatim in `kicks24.json` (SHA-256 `ad0f3a1adba5eff9c5aefea6fef7f90cfd6b857a b46279e0f32247307e664f7e`) and `refine_placement0.json` (SHA-256 `8c99446f731408d6 67818747c6319f4aac2c2d2cbddffc2d8df2adaf8feaaf58`).

Seg.	Sign-only τ_t	Weak τ_t	Weak $10^9\widetilde{a}_t$	Seg.	Sign-only τ_t	Weak τ_t	Weak $10^9\widetilde{a}_t$
1	12,821,058	12,720,755	1,000,000,000	13	65,182	127,379	764,201,123
2	13,747,829	13,875,167	1,000,000,000	14	40,397	129,011	834,019,056
3	92,406	373,513	1,000,000,000	15	137,815	46,142	916,974,644
4	65,722	170,424	1,000,000,000	16	100	56,478	903,538,624
5	129,303	137,205	1,000,000,000	17	30,741	56,630	915,327,669
6	122,231	188,284	1,000,000,000	18	228,337	37,059	911,809,418
7	14,548	169,485	1,000,000,000	19	472	55,214	824,068,230
8	129,661	207,325	1,000,000,000	20	209,872	123,114	608,644,063
9	16,328	242,404	1,000,000,000	21	291,357	371,610	464,745,164
10	114,245	205,999	881,826,710	22	100	417,762	368,423,142
11	31,763	76,765	842,867,478	23	369,114	633,308	331,827,239
12	78,933	134,879	786,860,383	24	300,014	897,029	1,000,000,000*

Table 8: Exact integer time numerators τ_t (defining dimensionless normalized times $s_t = \tau_t \cdot 10^{-9}$ and $\sqrt{N}$-scaled durations $s_t\sqrt{N}$, width $\ell_s = 1$ throughout) and archived weak-filter no-click amplitude numerators $10^9\widetilde{a}_t$ for the 24-segment sign-only (`kicks24.json`) and weak-filter (`refine_placement0.json`) schedules at $(1024, 512)$. At segments 10–23, the 14 nontrivial Kraus attenuations $a_t = \widetilde{a}_t \in (0, 1)$ are applied. *At segment 24, the archived weak filter is the identity ($\widetilde{a}_{24} = 1$), immediately followed by a separate terminal strong readout modeled by $a_{24} = 0$ (`row23.atten = 0` in `WeakCertRun.lean`).

B Partial foundations for the open `THCFactorization` challenge

The `THCFactorization` specification in `QuantumComputationSpecs` (which lacks a discovery implementation on the pinned `discoveries` branch, where its challenge JSON points `solution_module` to the specification placeholder) remains unsolved in this release: 0/7 frozen candidate definition holes (`thcBlockAncillas`, `thcScratchAncillas`, `thcCatalystQubits`, `thcCandidateAlpha`, `thcCandidateOffset`, `thcCandidateProgram`, `thcTCost`) and 0/2 public theorems (`thc_correctness`, `thc_t_cost`) are filled, and no certified resource estimates for FeMoco are claimed.

The companion release archives nine Lean 4.35.0 foundation modules (`Track1THC`, 1,956 lines in total; Table 9) and records 9/9 successful per-module `leanchecker` replays (standard axioms only). Claude Code authored the modules, drawing on the upstream Jordan–Wigner and register-splitting lemmas listed below; a subsequent cross-model OpenAI Codex audit (P1-06, November 27, 2026) reviewed all 123 source declarations and assumptions, verified declaration-level standard-axiom closure across all 218 kernel declarations (zero `sorryAx`, zero unlisted axioms), confirmed that mixed Jordan–Wigner CAR, normalized number-operator idempotence (`orbitalNumberOp_idem`), and reflection involution (`orbitalReflection_mul_self`) already appear in upstream `QuantumComputationDiscoveries/Algorithms/P450GroundEnergy/Solution/FermionPerm.lean`, and noted the `lamRep = 0` edge case on trivial rounded inputs.

Module (`Track1THC.*`)	Lines	Verified mathematical or circuit-algebra content
`Alias`	147	Exact integer Walker–Vose alias decomposition (`alias_exists`).
`JWAlgebra`	567	Orbital and sector identities adapted from `JordanWignerOrbitalAlgebra.lean` (Parts 1–3, 5). Part 6 locally proves mixed Jordan–Wigner CAR $\{a_i, a_j^\dagger\} = \delta_{ij} I$, an equivalent theorem to which already exists on the pinned `discoveries` branch (`QuantumComputationDiscoveries/Algorithms/P450GroundEnergy/Solution/FermionPerm.lean`), as do the normalized number-operator idempotence and reflection involution.
`Centering`	203	Given a rank-one expansion of the shifted one-body term, proves the centered operator identity; its indexed absolute coefficients sum to the frozen `lamRep` expression.
`LCUa`	33	Existence of a unitary with prescribed real unit first column.
`LCU`	255	Given a Hermitian involution, a unitary with the required real column, positive scale, and matching contraction, proves the frozen block-encoding predicate (adapts upstream `blockSplit*` lemmas).
`Select`	101	Abstract swap-select for adjoint-paired unitaries with involution-symmetric probabilities, applicable to paired products of non-commuting reflections.
`CleanOps`	397	Clean/catalytic composition, additive TOFF = $0/T$ algebra, lifted Jones-AND.
`CatalyticRot`	168	Exact eigenvector kickback on frozen `phaseGradientState`.
`CliffordGadgets`	85	Clean single-qubit gates, zero-T `CZ`, and clean Toffoli chains.

Table 9: Partial Lean 4.35.0 foundation modules for `THCFactorization` (1,956 lines total, 9/9 `leanchecker` exit 0, 0/7 frozen definition holes and 0/2 public theorems filled).

C Fixed-interface bit-trace lower bounds and restricted checks (Tier C)

Self-contained fixed-interface proofs for `s2a` and `s6c`, together with the algebraic reduction and exact certificate specifications for the archived Pfaffian obstruction and bounded MILP screen cited in Section 4, are provided below. All claims in this appendix are Tier (C) mathematical and computational bounds on fixed stage interfaces in the `Op102` table language, not formal Lean proofs or global unrestricted quantum circuit lower bounds.

Fixed-interface `Op102` bit-trace model. On the 23 raw magnitude bits remaining after endpoint relaying (with all stashed and external data fixed to constants), all $2^{23} = 8{,}388{,}608$ words are partitioned by the 440 certified dyadic cubes. Deterministic updates on basis-state bits are induced by `Op102` instructions as follows:

- *Zero-T affine/phase operations:* basis bits are updated affinely (or phase only is applied) at cost $0\,T$ by `X`, `CX`, `Z`, and `CZ`. A target wire is reset to 0 in the forward direction (an affine assignment) at cost $0\,T$ by the clean erasures `unand` and `unquad`.
- *Paid quadratic operations ($\geq 4\,T$):* a quadratic Boolean update at cost $4\,T$ is applied by both `rccx` variants and by the clean-target `and4t` gate. Into a clean target, the second-lowest bit of a signed sum of equal numbers of positive and negative affine parities—a quadratic Boolean polynomial—is written by a statically valid `hGadget`. With only one positive and one negative parity, both parities are forced to coincide by the static zero-low-bit invariant and the output vanishes identically; hence at least $2 + 2$ steps (cost $\geq 4\,T$) are required for any nonlinear `hGadget` write.

Forward degree bound for `s2a` ($\geq 68\,T$). When every wire state is expressed as a multilinear polynomial over $\mathbb{F}_2$ in the 23 input bits, algebraic degree is increased by at most 1 under backward substitution of a single quadratic assignment into an existing multilinear polynomial, whereas degree cannot be increased by an affine substitution [10]. Consequently, every output wire of a forward `Op102` trace containing ν paid nonlinear operations has degree at most $1+\nu$. By exhaustive 23-bit Boolean Möbius transform (`degree.py`, receipt `m3_degree_receipt.json`), algebraic degree 18 is attained on wires 50 and 55 of the frozen table `trS2a` (with coefficient 1 on the degree-18 monomial in bits 5–22). Hence $\nu \geq 17$ is forced, requiring at least $17 \times 4 = 68\,T$, which is attained tightly by `s2aOps`.

Reverse-direction affine-rank bound for `s6c` ($\geq 20\,T$). In the cleanup stage `s6c`, the 23 raw input bits together with 12 garbage wires entering from `s6b` are uncomputed back to clean zeros (plus the 23 raw bits). At step t of a valid deterministic `Op102` trace run backward on this domain, with current wire Boolean functions $f_{t,1}, \ldots, f_{t,w}$, the affine-span dimension is defined explicitly by

$$d_t \;=\; \dim_{\mathbb{F}_2} \operatorname{span}\{1, f_{t,1}, \ldots, f_{t,w}\},$$

where the constant 1 is included so that d_t is invariant under `X` and `CX`. Each instruction's contribution to the required number of quadratic backward substitutions is bounded in terms of its rank change $\Delta = d_t - d_{t-1}$:

1. When `unand` or `unquad` is reversed, a quadratic word is written into a clean wire. If that word already lies in the current affine span, it is represented by an affine substitution (cost 0 quadratic substitutions, $\Delta = 0$); otherwise 1 quadratic substitution is incurred and affine-span dimension is increased by $\Delta = +1$.

2. When a clean paid write (`and4t` or nonlinear `hGadget`) is reversed, a wire is erased at cost 0 quadratic substitutions while affine-span dimension is decreased by at most 1 ($\Delta \geq -1$). When `rccx` is reversed, one wire is replaced by itself XOR a product of two others: if $\Delta = -1$, its new value is affine in the remaining wires (cost 0), and otherwise at most 1 quadratic substitution is incurred with $\Delta \geq 0$.

3. Identically zero output bits are generated by zero- or two-T `hGadget` steps, contributing neither quadratic substitutions nor rank change; basis bit traces are unaffected by phases.

Thus, at every backward step with affine-rank change Δ, the number of quadratic substitutions contributed is at most $\Delta + \mathbf{1}_{\text{paid}\geq 4\,T}$. Summing across the entire reverse trace, the total affine-rank increase $d_{\text{final}} - d_{\text{initial}}$ from the 23 raw bits to the 23 raw bits plus 12 garbage coordinates is at most 12; with ν denoting the number of paid operations of cost $\geq 4\,T$, every incoming garbage function therefore has algebraic degree at most $1 + 12 + \nu$. By the same exhaustive 23-bit Möbius transform (`m3_degree_receipt.json`), degree 18 is attained on wires 72 and 81 of `trS6cIn` (coefficient 1 on bits 5–22), forcing $\nu \geq 5$ and hence at least $5 \times 4 = 20\,T$, attained tightly by `s6cOps`.

Pfaffian obstruction reduction for `s5_2CoreOps` and bounded MILP screen specifications. For the reverse 9-bit square `s5_2CoreOps` (`verify_pfaffians.py`, `followup.py`, `pfaffian_witness.json`, receipt `m2_reverse_obstruction_receipt.json`), the 9 linear input bits are denoted by X, the 9 frozen square output bits by Y, and the 21 baseline paid product functions by $P_0, \ldots, P_{20}$. With the 17 products indexed by $\{0, 2\text{–}10, 12\text{–}18\}$ held fixed, the 35-dimensional fixed subspace $\mathcal{F} = \text{span}(V)$ over $\mathbb{F}_2$ is spanned by the 35 linearly independent functions $V = X \cup Y \cup \{P_i : i \notin \{1, 11, 19, 20\}\}$, whereas the 39-dimensional total allowed intermediate space $\mathcal{U} = \text{span}(V \cup R)$ and the 4-dimensional quotient space $\mathcal{Q} = \mathcal{U}/\mathcal{F} \cong \mathbb{F}_2^4$ (with canonical projection $\pi : \mathcal{U} \to \mathcal{Q}$) are obtained when the four rewritten products $R = \{P_1, P_{11}, P_{19}, P_{20}\}$ are adjoined. Any valid replacement of those four AND gates by at most three new AND gates whose products lie within $\mathcal{U}$ must generate intermediates whose images under π lie in a *proper subspace* $S \subsetneq \mathbb{F}_2^4$ ($\dim S \leq 3$, $|S| \leq 8$), of which there are exactly $1 + 15 + 35 + 15 = 66$ over $\mathbb{F}_2$. For each proper subspace S, under *reverse quadratic closure* it is tested whether every function in the preimage $\pi^{-1}(S) \subset \mathcal{U}$ can be uncomputed back to X by sequential quadratic erasures (each step eliminating a function quadratic in the remaining wires). By exhaustive enumeration of all 66 proper subspaces, exactly one 3-dimensional hyperplane $H_0 = \{0, 3, 5, 6, 9, 10, 12, 15\} \subset \mathbb{F}_2^4$ (with ordered basis $W = (P_1 \oplus P_{11}, P_1 \oplus P_{19}, P_1 \oplus P_{20})$ and 38-dimensional preimage $\pi^{-1}(H_0) = \text{span}(V \cup W)$) is retained.

To enter $\pi^{-1}(H_0)$ from $\mathcal{F}$ via new AND gates, at least one nonzero direction in $H_0 \setminus \{0\}$ must be realized modulo $\mathcal{F}$ as a single rank-1 product $(a^\top V)(b^\top V)$, equivalently by a skew-symmetric alternating matrix $E = (e_{ij})_{0 \leq i < j < 35}$ over $\mathbb{F}_2$ of rank at most 2. The seven nonzero directions of H_0 are subsequently indexed by their 3-bit coordinates $t \in \{1, \ldots, 7\}$ in the ordered basis W (corresponding to the 4-bit masks $3, 5, 6, 9, 10, 12, 15$ respectively). For each t, an affine space of dimension $d = 190$ over $\mathbb{F}_2$ is formed by the set of all alternating matrices E whose evaluated quadratic form matches t modulo $\mathcal{F}$. Every rank-≤ 2 alternating matrix is annihilated by all $\binom{35}{4}$ four-index Pfaffians $\text{Pf}_{ijkl}(E) = (e_{ij}e_{kl}) \oplus (e_{ik}e_{jl}) \oplus (e_{il}e_{jk})$. Because each $\text{Pf}_{ijkl}(E)$ is a Boolean polynomial of degree at most 2 in the $d = 190$ free affine parameters, an XOR sum of Pfaffians is identically equal to 1 across all 2^{190} parameter assignments if and only if it evaluates to 1 on all $1 + d + \binom{d}{2} = 18{,}146$ assignments of Hamming weight at most 2. For all seven nonzero targets $t \in \{1, \ldots, 7\}$, explicit subsets of four-index Pfaffians (5,796, 5,838, 5,718, 4,652, 4,617, 4,752, and 4,554 Pfaffians respectively; archived in `pfaffian_witness.json`, SHA-256 `194078da6ceb06b1...`) are verified by `verify_pfaffians.py` to sum identically to 1 on all 18,146 weight-≤ 2 assignments, proving that 21 ANDs are necessary within the restricted 39-dimensional intermediate span $\mathcal{U}$.

For the late-adder carry recurrence (`burst_milp.py`, receipts `track3_bonus_burst_milp.json` and `track3_bonus_burst_control.json`), zero-return carry-save schedules over 28 columns are formulated as integer linear programs and solved via `scipy.optimize.milp` (HiGHS backend, `mip_rel_gap = 0.0`, single-threaded, 10 s time limit per instance). Across all 18 combinations of $B \in \{3, 4, 5, 6\}$ bursts and helper placement $u \in \{0, \ldots, B-1\}$ under capacity 39 scratch slots and ceiling 165 ANDs, infeasibility is reported (SciPy return status 2, HiGHS backend model status 8), whereas in a 200-slot positive control ($B = 1, u = 0$) optimality is reached (SciPy return status 0, HiGHS model status 7) at 133 ANDs (independently checked against all column-sum and carry constraints). Because no independently re-verified proof log is emitted by HiGHS here, these 18 infeasibility results are reported strictly as solver-reported bounded screens rather than proof certificates.

D Verification ledger, SHA-256 prefixes, and `v1.0.1` patch history

This appendix consolidates the pinned toolchain digests, upstream Git commit identifiers, file-count reconciliations, external-checker declaration counts, and the `v1.0.0` → `v1.0.1` `ReducibilityHints` patch record referenced in Sections 1 and 6 (full 64-hex SHA-256 digests appear in `SHA256SUMS.txt`).

Pinned toolchain and upstream commits. All primary Lean proofs were compiled and replayed with isolated `Lean (version 4.35.0, x86_64-w64-windows-gnu, commit eb6cafb7290e, Release)`, binary SHA-256 prefixes `ec7a472d1c12a09b` (`lean.exe`) and `9ef6baeb28d5e288` (`leanchecker.exe`). Both unmodified upstream reference checkouts of `google-deepmind/formal-quantum-computation` are pinned at `main` commit `489edc7039aa4cdded53724b39ea90e33d2c88b4` and `discoveries` commit `a88de8ff65d78af564b2373088ca369dc95f6547`.

File-count reconciliation and static hygiene. In initial release `v1.0.0`, all 288 Lean 4.35.0 project modules were replayed by `leanchecker.exe` on Windows ($31+20+117+111+9 = 288$; in `Track2Weak`, the 111 modules comprise the 109-module transitive closure of `FindAllWeak` plus `FiniteFromCutoff` and `WeakAssemblyControl`, logged in `track2weak_clean_all_leanchecker.json` alongside `final_semantic_axioms.json`). Exactly 279 `.lean` files were staged in `v1.0.1` because in Track 3 (20 replayed modules: 16 arcsine modules plus `Track3Audit`, `ExactCount`, `ExactPublic`, and `DisjointWitness`), the 11 byte-identical upstream arcsine modules are omitted while 2 helper scripts (`CompareFrozen.lean`, `ReplacementTables.lean`) are included ($31 + 11 + 117 + 111 + 9 = 279$); adding the 4 `P1Upgrade.*` modules brings `v1.1.3` to 292 checked modules and 283 staged `.lean` files. Comment-stripped static scans confirm zero forbidden escape hatches (`sorry`, `admit`, `native_decide`, `axiom`, `unsafe`, `implemented_by`, `extern`, `ofReduceBool`, `trustCompiler`; kernel-checked `decide +kernel` occurs in 144 files). On Windows, frozen interfaces were checked by `CompareFrozen.lean` (25/25 arcsine declarations), `compare_interfaces.py` (9 types / 7 bodies for `FindAny`; 9/9 for sign-only `FindAll`), and `interface.py` (18 types, 11 normalized bodies).

Clean-machine Linux `lake comparator` runs and `v1.0.1` `ReducibilityHints` history (`fix15`–`fix17b`). Separate-host 64-bit Linux verification ran on Ubuntu 26.04.1 LTS (kernel 7.0.0-38-generic, 62 GiB RAM, Lean 4.35.0 `eb6cafb7`, Lake 5.0.0, `bubblewrap` 0.11.1 without `-inadvisably-no-sandbox`; manifests archived under `receipts/linux_comparator/`): (i) `ArcsinK25ConstCost` accepted verbatim (`1,426 s`, exit 0), upstream `lake test` (`312 s`, all 7 configs consistent), Linux `leanchecker` (16/16), `CompareFrozen.lean` (25/25), and

standalone replays of `ExactCount`, `ExactPublic`, and `DisjointWitness` (15/15 steps exit 0); (ii) `Track2Grover.TighterFindAny` accepted (330 s, exit 0) and `FindAnyLowerBound` replayed by Linux `leanchecker` (62 s) with `#print axioms`; (iii) `Track2Grover.FindAll` sign-only accepted (3,402 s, cgroup peak 13.14 GiB, 0 OOM kills); (iv) `Track2Weak.FindAllWeak` on `v1.0.0` (SHA-256 prefix `d210ff50...ac929`) initially exited 1 after 142 s due to a `ReducibilityHints` height mismatch on `groverFindAllProgram` (`regular(46)` vs. upstream `regular(37)`, despite identical `typeHash 289833878` and `valueHash 4143947326`) caused by declaring `weakChosenBatch` with `def` instead of `abbrev` (cf. upstream `StageLoop.lean`, line 2327). Changing line 30 to `noncomputable abbrev weakChosenBatch` (SHA-256 prefix `8ba169948040f7c1...470bdd9a`; all other 110 `Track2Weak` files unchanged) restored `regular(37)` and passed Linux `fix15` (build of `FindAllWeak` and `FiniteFromCutoff`), `fix16` (`lake comparator`, 2,007 s, peak 6.57 GiB, exit 0; log `linux_comparator/run2_findall/fix16/results_fix16/fix16_findall_weak_OFFICIAL_lake_comparator.log`), `fix17` (Linux `leanchecker`, 109 s), `fix17b` (`#print axioms` across all five theorems, log `fix17b_findall_weak_print_axioms.log`; four harmless `unusedVariables` warnings remain), and post-fix Windows `leanchecker` (`track2_weak/v1_0_1_postfix/postfix_track2weak_leanchecker.json`).

Third-party kernel checks and `P1Upgrade.*` SHA-256 prefixes. The `lean4export` 3.1.0 exports of `FindAnyLowerBound` and `TighterFindAny` were accepted with zero errors by `nanoda_bin` 0.4.19 (49,235 and 46,246 constants) and `con-ron -verified` (47,256 and 44,302 declaration records; receipts in `receipts/external_checkers/`). The four `v1.1.0` `P1Upgrade.*` modules (38 targets, master receipt `P1_ALL_5CHECKER_SUMMARY.json`, SHA-256 prefix `927e44830ec7`; multiline axiom reparse `verify_p1_lean5_fixed.py` prefix `3ea033107b00`) carry source SHA-256 prefixes `8db17431` (`SubmissionBridge`, 10 targets), `a1dfd295` (`FindAnyAllPos`, 5 targets), `a51c5ddb` (`AllPositiveInfimum`, 2 targets), and `14b8441d` (`FiniteSharpLower`, 21 targets), all passing 5/5 checks (`lean`, `leanchecker`, `con-leche -verified`, `con-ron -verified`, `nanoda_bin`) with standard axioms only.

References

[1] S. Shirobokov, B. Jaderberg, K. Hristova, G. Tsoukalas, J. Bausch, E. Wieser, F. J. H. Heras, N. C. Rubin, G. H. Low, S. Racanière, N. Tomašev, L. M. Zhang, J. D. Watson, T. Khattar, A. Albarghouthi, S. Jain, H. Khan, L. Sarra, A. W. Senior, M. Broughton, H. Masoom, S. Boixo, R. Babbush, S. Chaudhuri, and P. Kohli, *Discovering formally verified quantum programs with AI*, arXiv:2611.09822v3 [quant-ph] (2026); repository https://github.com/google-deepmind/formal-quantum-computation.

[2] W. Fang and C. Heunen, *Weakly measured loops for quantum amplitude amplification: oracle savings and adaptive search with unknown target probability*, arXiv:2609.39621 [quant-ph] (2026).

[3] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC), pp. 212–219, 1996.

[4] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, *Tight bounds on quantum searching*, Fortschritte der Physik **46** (1998), 493–505.

[5] C. Zalka, *Grover's quantum searching algorithm is optimal*, Physical Review A **60** (1999), 2746–2751.

[6] L. K. Grover and J. Radhakrishnan, *Quantum search for multiple items using parallel queries*, arXiv:quant-ph/0407217 (2004).

[7] A. Belovs and D. Yolcu, *One-way ticket to Las Vegas and the quantum adversary*, arXiv:2301.02003 [quant-ph] (2023).

[8] A. Mizel, *Critically damped quantum search*, Physical Review Letters **102** (2009), 150501.

[9] P. Andrés-Martínez and C. Heunen, *Weakly measured while loops: peeking at quantum states*, Quantum Science and Technology **7** (2022), 025007.

[10] C. P. Schnorr, *The multiplicative complexity of Boolean functions*, in Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-6, 1988), LNCS 357, pp. 45–58, Springer, 1989.

[11] R. Mirwald and C. P. Schnorr, *The multiplicative complexity of quadratic Boolean forms*, Theoretical Computer Science **102** (1992), 307–328.

[12] Ç. Çalık, M. Sönmez Turan, and R. Peralta, *Boolean functions with multiplicative complexity 3 and 4*, Cryptography and Communications **12** (2020), 935–946.

[13] T. Häner, M. Roetteler, and K. M. Svore, *Optimizing quantum circuits for arithmetic*, arXiv:1805.12445 [quant-ph] (2018).

[14] C. Jones, *Low-overhead constructions for the fault-tolerant Toffoli gate*, Physical Review A **87** (2013), 022328.

[15] D. Litinski, *Magic State Distillation: Not as Costly as You Think*, Quantum **3** (2019), 205 (`arXiv:1905.06903`; DOI `10.22331/q-2019-12-02-205`).

[16] C. Gidney, *Halving the cost of quantum addition*, Quantum **2** (2018), 74.

[17] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, *Even more efficient quantum computations of chemistry through tensor hypercontraction*, PRX Quantum **2** (2021), 030305.

[18] Y. Feng and M. Ying, *Quantum Hoare logic with classical variables*, ACM Transactions on Quantum Computing **2** (2021), 1–43.

[19] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in Automated Deduction — CADE 28, LNCS 12699, pp. 625–635, Springer, 2021.

[20] The mathlib Community, *The Lean mathematical library*, in Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP), pp. 367–381, 2020.

[21] The Physlib Community, *Physlib: The Lean Physics Library*, `https://github.com/leanprover-community/physlib`, 2024; incorporating A. Meiburg, *Quantum Information in Lean*, `https://github.com/Timeroot/Lean-QuantumInfo`, 2024.

[22] Lean FRO, *comparator: Lean 4 formal challenge verification tool*, `https://github.com/leanprover/comparator`, 2026.

[23] C. Bailey, *nanoda_lib / nanoda_bin (v0.4.19): independent Rust type checker for Lean 4*, `https://github.com/ammkrn/nanoda_lib`, bundled with Lean 4.35.0 (`eb6cafb7`), 2026.

[24] Lean 4.35.0 Toolchain, *con-ron (`-verified`): formally verified Rust arena type checker for `lean4export` 3.1.0 streams*, `https://github.com/leanprover/con-ron`, bundled with Lean 4.35.0 (`eb6cafb7`, binary SHA-256 `4e351dd1...332d40`), 2026.

[25] F. Johansson, *Arb: efficient arbitrary-precision midpoint-radius interval arithmetic*, IEEE Transactions on Computers **66** (2017), 1281–1292.