THE STANDARD LOGICAL-QUBIT

A right-marker definition for computation-grade logical quantum systems

Version 1.0

This document proposes a stricter standard for what should count as "The Standard Logical-Qubit." The premise is simple: a serious logical-qubit claim cannot be grounded only in nomenclature, an encoding sketch, or a narrow noise-correction story. It should be grounded in demonstrable logical-state integrity, coherent quantum-state phenomena, executable full-run behavior, scalable synthesis, and utility workload breadth.

The goal is not to weaken the term "logical qubit," but to strengthen it. A right-marker standard should describe the full capability surface expected from a computation-grade logical-quantum stack.

1. Why a stricter standard is needed

The phrase logical qubit is often used as though it were self-proving. It is not. A logical qubit should not be considered standard merely because it is wrapped in the language of encoding, abstraction, or protection. The standard should be set by what the claimed stack can actually preserve, manipulate, measure, synthesize, execute, and verify.

A narrow benchmark that focuses only on one failure mode - for example, hardware noise - can be useful, but it is not sufficient as a complete right-marker. A computation-grade logical-qubit claim should be able to answer a broader question: does the system exhibit the coherent state phenomena, control semantics, runtime behavior, and algorithmic utility that a serious logical-quantum computing platform is expected to deliver?

Under this view, a standard logical-qubit claim is not only a claim about protection; it is also a claim about demonstrated capability. It must hold together as a full computational story.

2. Proposed definition

Definition: The Standard Logical-Qubit is a logical-quantum computational unit or stack that can preserve, manipulate, measure, and verify logical quantum state across full executable runs, while demonstrating the coherent phenomena, control surface, synthesis surface, and utility workload breadth required for computation-grade use.

A claim should not qualify as standard if it is missing any of the following layers:

- Logical-state integrity: the claimed logical state remains meaningful and reproducible under the conditions of execution.
- Coherent quantum-state phenomena: amplitudes, relative phase, superposition, entanglement, interference, basis-change behavior, and measurement are not merely asserted; they are demonstrated.
- Executable runtime completeness: the system can be run end-to-end with honest input, honest output, and no hidden discarding of inconvenient cases.
- Scalable synthesis: larger structured artifacts can be derived with inspectable stage structure, resource planning, and honest boundary statements.
- Utility workload breadth: the platform can support meaningful algorithm families, not only isolated gate-level toys.

3. What should not qualify

- An encoding diagram with no executable runtime evidence.
- A gate catalog with no demonstrated amplitude, phase, or interference behavior.
- A single hand-picked demonstration that depends on discarded runs, hidden conditions, or opaque post-selection.
- A preview-only claim with no inspectable compile path, runtime path, or verification material.
- A synthesis surface that hides scale limits instead of stating them honestly.
- A phenomenon demo that never connects to algorithmic or workload-level utility.

- A benchmark story that cannot expose its input artifact, its runtime trace, and its result bundle.

4. The five right-marker criteria

Criterion 1 - Logical-state integrity and reproducibility

A standard logical-qubit claim must show that the logical state is not fragile in an epistemic sense. The same input, configuration, and declared runtime conditions should reproduce the same essential result. If stochastic sampling is used, the distribution should be stable and explainable.

What to check:

- Reproducible outputs under fixed input and seed policy.
- Stable counts, probabilities, or decoded values across repeated execution.
- Inspectability of the state-preparation path and the readout path.
- Explicit runtime metadata rather than slogan-level reporting.

Criterion 2 - Coherent quantum-state phenomena

A standard logical-qubit claim must demonstrate the phenomena that make logical quantum computation distinct. These include amplitude-bearing state description, relative phase, superposition, entanglement or non-classical correlation, interference, basis-change behavior, and measurement collapse.

What to check:

- State preparation that can produce superposed and entangled states.
- Phase-sensitive evolution and phase-tracking behavior.
- Interference under basis change, not merely classical correlation.
- Measurement and readout consistent with the prepared state.

Criterion 3 - Executable full-run completeness

A standard logical-qubit claim must survive contact with runtime reality. That means full runs, not only static previews. If mid-circuit measurement, reset, conditional control, or classical register logic are part of the claimed model, they must be executable and observable in the result surface.

What to check:

- No cherry-picking or silent discarding of unfavorable runs.
- Support for full-run control semantics where claimed.
- Ability to expose runtime behavior through counts, probabilities, traces, or branch summaries.
- Honest statement of unsupported operations and size limits.

Criterion 4 - Scaling and synthesis law

A standard logical-qubit claim should have a scaling story that can be inspected. Even when large artifacts are not fully executed locally, the system should be able to synthesize them,

expose register plans and stage structure, export artifacts, report resource estimates, and state clearly when a path is preview-only rather than executable.

What to check:

- Parameterized growth in problem size or logical width.
- Explicit register planning and stage decomposition.
- Resource estimates, attempt traces, and exportable artifacts.
- Honest boundary conditions between executable kernels and synthesis-only artifacts.

Criterion 5 - Utility workload breadth

A standard logical-qubit claim must extend beyond one iconic trick. It should connect to a meaningful workload surface: sampling, transforms, phase estimation, amplitude estimation, search, period finding, optimization, simulation, and other computation-grade families as appropriate to the stack.

What to check:

- Support for a broad family of benchmark and algorithm surfaces.
- Clear mapping from low-level phenomena to high-level workloads.
- Ability to inspect how a workload is represented, compiled, executed, or evaluated.
- Evidence that the platform is not locked to a single demo narrative.

5. Minimum evidence bundle for a serious claim

A claim should ideally expose a compact but sufficient evidence bundle. At minimum, this bundle should contain:

- The input artifact, such as OpenQASM, IR, or another inspectable logical program surface.
- A validator report describing qubits, classical bits, gates, measurements, unsupported operations, and explicit limits.
- A circuit preview or schedule so that the claimed program can be inspected before execution.
- A compiled package identifier or package hash tying the execution result to a specific compiled artifact.
- A runtime result bundle containing counts, probabilities, decoded outputs, or comparable observable outputs.
- Replay material, replay hash, or equivalent reproducibility surface.
- A certificate or signature surface, even if development-grade, so the claim is not purely verbal.
- Resource estimates for larger synthesized artifacts, where relevant.
- An honest distinction between executable paths and synthesis-only or preview-only paths.
- Regression tests or verification logs showing that the claimed surfaces continue to hold.

6. Core capability checklist for The Standard Logical-Qubit

The following checklist is intentionally broader than a narrow gate list. It captures the capability surface that a computation-grade claim should be prepared to defend.

6.1 Core state, coherence, and readout phenomena

- Logical state preparation.
- Amplitude-bearing state representation.
- Inspectability of probabilities derived from amplitudes.
- Relative phase tracking, including non-trivial phase gates.
- Superposition generation and persistence.
- Entangled pair or multi-qubit correlation generation.
- Basis-change coherence.
- Interference behavior under phase and basis transformation.
- Measurement collapse and readout.
- Born-style sampling from exact or declared probability surfaces.
- Stable hardware-like bit ordering in readout.
- State reset semantics where claimed.

6.2 Gate and control surface

- Single-qubit gates such as H, X, Y, Z, S, Sdg, T, and Tdg or their declared equivalents.
- Two-qubit and three-qubit gates such as controlled-NOT, controlled-Z, Toffoli, and swap or their declared equivalents.
- Measurement, barrier, and reset operations.
- Controlled-phase ladders and inverse-transform stages where applicable.
- Mid-circuit measurement.
- Classical assignment to measured or declared classical bits.
- If/else control driven by measured bits or register-integer conditions.
- Branch-aware execution when runtime semantics depend on measurement outcomes.

6.3 Compilation, inspection, and verification surface

- OpenQASM 2 and/or OpenQASM 3 validation, or an equivalently inspectable logical program surface.
- Gate parsing, measurement parsing, and explicit reporting of unsupported constructs.
- Honest qubit-limit handling and explicit limit error reporting.
- Circuit preview or schedule generation that stays synchronized with the actual runnable artifact.
- Preview hashes, package hashes, or equivalent traceability markers.
- Result export and certificate export.
- Replay hash or equivalent reproducibility fingerprint.
- Runtime capacity reporting.
- Attempt traces, metadata endpoints, or artifact-export routes for large synthesized programs.

6.4 Arithmetic and synthesis surface

- Compiled witness kernels for non-trivial workloads.
- Reversible arithmetic scaffolds.
- Ripple-carry adders or another explicit arithmetic construction.
- Modular addition and multiply-by-constant scaffolds where relevant.

- Coherent modular exponentiation stage structure where relevant.
- Inverse transform stages such as inverse-QFT ladders where relevant.
- Register planning and stage metrics.
- Structured artifact export for large synthesized circuits.
- Resource estimates, depth estimates, or primitive-count estimates.
- Preview-only mode for circuits that exceed local runtime width, provided this boundary is stated honestly.

6.5 Utility workload surface

- Bell-pair or GHZ-style correlation demonstrations.
- Sampling or random-circuit benchmark surface.
- Quantum Fourier Transform.
- Quantum Phase Estimation.
- Amplitude Estimation.
- Search surfaces such as Grover-style workloads.
- Period finding and Shor-style pipelines.
- Hidden-structure workloads such as Bernstein-Vazirani or Simon-style workloads.
- Optimization workloads such as QAOA-style Max-Cut.
- Variational energy workloads such as VQE-style molecular energy.
- Hamiltonian simulation.
- Quantum walk workloads.
- Kernel-based machine learning workloads.
- Linear-system or Poisson/PDE surfaces.
- Portfolio optimization surfaces.
- Risk estimation surfaces using amplitude-estimation-style subroutines.

6.6 Extended logical-compositional surface

When present, the following extended surfaces strengthen a claim because they show that the logical-qubit story is embedded in a larger computational architecture rather than isolated demos only.

- Relation-matrix state surfaces.
- Path closure or relation composition surfaces.
- Diagonal anchoring or anchor constraints.
- Reciprocity or transpose-consistency surfaces.
- Energy descent and attractor-discovery surfaces.
- Equality and exclusivity constraint surfaces.
- Compositional entanglement surfaces.
- Attractor-subspace or constrained-subspace surfaces.
- Scenario-tree or branching execution surfaces.
- Frontier readout, semantic merging, or weighted frontier surfaces.
- Compact high-budget logical profiles such as 10K-logical-style surfaces.

7. Clearly demonstrated surfaces that belong inside the standard

The right-marker should not stop at theory. It should include the kinds of surfaces that can be shown explicitly, repeatedly, and inspectably in an execution stack. The following list is a model of the sort of clearly demonstrated material that should count.

7.1 Explicit runtime demonstrations

- Balanced entangled-output demonstrations such as 00/11 Bell-style outcomes.
- Preservation of Bell-style coherence under basis change.
- Exact tracking of non-trivial phase behavior such as T-phase evolution.
- Born-style sampling that reflects the declared probability surface rather than naive rounding.
- Mid-circuit measurement followed by conditional quantum control.
- Reset operations that preserve the expected partner distribution after entanglement has been created.
- Register-integer classical conditions and classical assignment paths inside dynamic circuits.
- Reproducible seeded execution returning the same counts and memory where claimed.
- Stable bit-ordering in the output surface.
- Exposure of exact-kernel metadata to the compile layer and the run layer.

7.2 Explicit compiled and synthesis demonstrations

- Compiled witness kernels that decode a phase and recover non-trivial factors on executable small kernels.
- Input-aware kernel generation for specific composite inputs.
- Large structured synthesis artifacts for period-finding-style workloads.
- Register plans containing counting registers, work registers, accumulator or scratch structure, and measurement banks where relevant.
- Stage decomposition covering superposition, controlled modular exponentiation, inverse transform, and measurement.
- Metadata and attempt traces for synthesized artifacts.
- Export routes for large structured program artifacts, including compressed export when appropriate.
- Honest preview-only handling for large logical widths that exceed the local exact runtime.

7.3 Integrated benchmark and workload catalog surfaces

- Search surfaces.
- Transform surfaces.
- Phase-estimation surfaces.
- Amplitude-estimation surfaces.
- Period-finding surfaces.
- Optimization surfaces.
- Simulation surfaces.
- Quantum walk surfaces.
- Hidden-structure surfaces.
- Kernel machine-learning surfaces.

- Sampling benchmark surfaces.
- Risk, portfolio, and linear-system surfaces.

8. A practical pass standard

A claim should not need to max out every possible capability before being taken seriously. But a claim should not earn the phrase The Standard Logical-Qubit unless it clears all five right-marker criteria and can defend a broad portion of the capability checklist with inspectable evidence.

- Clearing only criterion 1 is not enough.
- Clearing only a small phenomenon demo is not enough.
- Clearing only a synthesis preview is not enough.
- Clearing only a benchmark chart is not enough.
- The standard belongs to stacks that integrate state, control, execution, synthesis, verification, and utility.

9. Long-form capability appendix

The appendix below is deliberately long. It is intended to function as a right-marker checklist rather than a slogan.

9.1 Core quantum-state and readout checklist

- Amplitude alpha / beta or equivalent logical-state amplitude surface.
- Exact probability extraction from the declared state representation.
- Relative phase observability.
- Phase-sensitive transform behavior.
- Superposition preparation.
- Unresolved-versus-collapsed state distinction where the architecture exposes one.
- Entanglement groups, links, or equivalent pair/multi-qubit correlation summary.
- Global entanglement proxy or pair-correlation summary.
- Basis-cell or nonzero-state support summary.
- Collapsed readout labels or state-class labels.
- Measurement-bank support.
- Probability tables and shot-based count tables.

9.2 Dynamic semantics checklist

- Mid-circuit measurement.
- Conditional branch count reporting.
- If/else gate application based on classical bits.
- Register-wide comparison conditions.
- Classical-set or direct classical assignment.
- Reset semantics inside dynamic circuits.
- Statevector suppression when branch semantics make a single pure-state vector inapplicable.
- Branch-aware metadata in compile and runtime surfaces.

9.3 Compilation and package checklist

- Validation OK / not OK status.
- Warnings describing branch kernels, width limits, or unsupported operations.
- Package identifier.
- Package hash.
- Compile stages.
- Profile metadata.
- Selected rule or selected workload context.
- Circuit preview plus preview hash.
- Template-sync state so the preview cannot silently drift away from the runnable artifact.
- Export of package JSON, result JSON, and certificate JSON.

9.4 Verification and traceability checklist

- Replay material or replay hash.
- Certificate identifier or signature surface.
- Job identifier and runtime monitor path.
- Counts, probabilities, decoded values, and timings in the result bundle.
- Verification logs or regression tests.
- Honest distinction between local-development signing and production signing when relevant.

9.5 Large-synthesis checklist

- Deterministic base-selection trace for period-finding-style synthesis where relevant.
- Structured OpenQASM artifact generation.
- Large logical-width display planning.
- Qubit-count / logical-width estimates.
- Primitive gate estimates such as controlled operations and Toffoli counts.
- Gzip or other compressed artifact export for large synthesized programs.
- Metadata endpoint for synthesis artifacts.
- Attempt-trace endpoint for synthesis artifacts.
- Explicit handling of non-runnable but synthetically valid inputs.
- Honest handling of prime inputs, even-input traces, or no-witness cases.

9.6 Workload-family checklist

- Bell or GHZ small-state demonstrations.
- Random-circuit sampling / heavy-output / XEB style surfaces.
- Quantum Fourier Transform.
- Quantum Phase Estimation.
- Amplitude Estimation.
- Grover-style search.
- Shor / period finding.
- Bernstein-Vazirani.
- Simon-style hidden-structure recovery.
- QAOA-style combinatorial optimization.

- VQE-style energy minimization.
- Hamiltonian simulation.
- Quantum walk.
- Kernel classification.
- Poisson / PDE solving.
- Portfolio optimization.
- Risk estimation with amplitude-estimation-style subroutines.

10. Closing statement

The phrase The Standard Logical-Qubit should be reserved for claims that are broad enough, honest enough, and executable enough to deserve it. The right-marker is not satisfied by language alone. It is satisfied when a system can show - in one integrated story - logical-state integrity, coherent state phenomena, full-run execution, scalable synthesis, and a meaningful workload surface.

Anything less may still be promising. But it should be described more narrowly. The standard should remain hard to earn.