Closing the Loop

Variational Quantum Algorithms · Module 10 — Vqa As Adaptive Control Systems

Node 10.5

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** VQA as Adaptive Control Systems **Node:** 10.5

Nodes 10.1-10.4 have built up the pieces: the adaptive architecture (10.1), measurement-adaptive circuits (10.2), RL as a meta-controller (10.3), and VQA for quantum control (10.4). This node is the **synthesis**: putting all the pieces together into a **fully closed-loop VQA**.

"Closing the loop" is a phrase from control theory. An **open-loop** system executes a pre-computed schedule regardless of what happens. A **closed-loop** system observes the environment and adapts. In VQA language, an open-loop VQA has a fixed ansatz, fixed optimizer, fixed shot budget, fixed hardware choice, and runs straight through. A closed-loop VQA *observes its own execution* and *modifies itself* in response.

This node describes the architecture of a fully closed-loop VQA — every component from Module 10 combined into a single coherent system. It's the "integration" node, showing how adaptive ansätze, adaptive optimizers, adaptive shot budgets, measurement feedback, and hardware monitoring all work together.

The key architectural principle: **the loop is closed at multiple levels simultaneously**, and each level's feedback informs a different part of the system. Understanding this multi-level structure is the payoff of Module 10 as a whole.

The Multi-Level Closed Loop

A fully closed-loop VQA has feedback at (at least) five distinct levels:

Level 1 — Gate-level feedback (inner loop)

Feedback signal. Measurement outcomes of mid-circuit measurements (Module 10.2).

Controlled variable. Which gates to apply next, in which branch.

Timescale. Nanoseconds to microseconds — within a single circuit execution.

Example. Adaptive error correction: measure a stabilizer, apply a correction if needed.

Level 2 — Circuit-level feedback (parameter update)

Feedback signal. Cost estimate from the completed circuit.

Controlled variable. Parameter values (the VQA parameters $\boldsymbol{\theta}$).

Timescale. Milliseconds to seconds — per optimizer iteration.

Example. Gradient descent or HOPSO update.

Level 3 — Estimator-level feedback

Feedback signal. Cost variance, gradient signal-to-noise ratio.

Controlled variable. Shot budget per evaluation, mitigation technique selection.

Timescale. Seconds to minutes — per group of optimizer iterations.

Example. Adaptive shot allocation, ZNE on/off decisions.

Level 4 — Architecture-level feedback (meta-optimizer)

Feedback signal. Cost convergence trajectory, gradient norm history.

Controlled variable. Ansatz structure, optimizer hyperparameters, learning rate schedule.

Timescale. Minutes to hours — every several optimizer iterations.

Example. ADAPT-VQE's gate growth, learning rate decay, ansatz restructuring.

Level 5 — Hardware-level feedback

Feedback signal. Hardware calibration reports, drift detection, error budget monitoring.

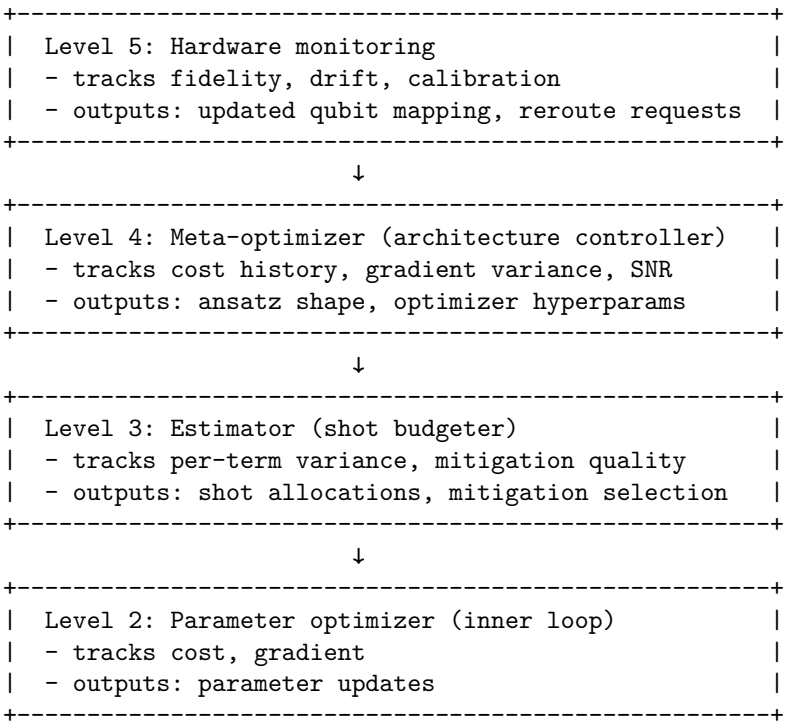
Controlled variable. Qubit selection, routing, recalibration triggers.

Timescale. Hours to days — as hardware drifts and gets recalibrated.

Example. Noise-adaptive qubit selection, re-routing on drift.

Each level closes its own loop. The five levels operate at separated timescales (Module 10.1's three-loop structure, now refined to five). Each can be designed independently and composed. The result is a system with feedback at every structural scale.

The Full Architecture Diagram



```
                        ↓
+-----------------------------------------------------+
|  Level 1: Circuit executor (quantum)                |
|  - runs adaptive circuit                            |
|  - outputs: measurement outcomes, cost estimates    |
+-----------------------------------------------------+
```

Data flow. Information flows upward (each level feeds back to the next): measurement outcomes to the optimizer, cost estimates to the meta-optimizer, convergence state to the hardware monitor, etc. Control flow is downward: the meta-optimizer commands the optimizer, which commands the estimator, which commands the executor.

Feedback is at every level. Each arrow is a closed-loop: the upper level observes the lower level's output and adjusts its own strategy.

Timescale Separation

Each level's feedback loop must operate at its own timescale, well-separated from adjacent levels. Roughly:

Level	Timescale	Typical duration
1 (gate)	ns - μs	single gate to several gates
2 (optimizer)	ms - s	single iteration
3 (estimator)	s - min	shot budget cycle
4 (meta-optimizer)	min - hr	across many iterations
5 (hardware monitor)	hr - day	across many VQA runs

Why separation matters. If two levels operate at similar timescales, they interfere: the meta-optimizer changes the architecture before the optimizer has converged, so the optimizer's gradient estimates become stale, the cost trajectory is unstable, and the meta-optimizer sees garbage data and makes bad decisions.

Design rule. Each level should run for at least 10× the duration of the level below it before making a decision. This gives the lower level time to quasi-equilibrate and provide meaningful feedback.

Concrete Closed-Loop Example

Consider a 20-qubit chemistry VQE for C_2H_4 on superconducting hardware:

Level 1 — Gate feedback. The ansatz includes symmetry verification via mid-circuit measurement. If particle number is wrong, apply correction. If correction fails, retry.

Level 2 — Optimizer (HOPSO). 10 particles, learning rate 0.05, momentum 0.9. Runs 50 iterations per outer loop.

Level 3 — Estimator. Shot budget 1000 per cost evaluation. Readout mitigation always. Linear ZNE every 10 iterations. Shot allocation weighted by Hamiltonian term magnitude.

Level 4 — Meta-optimizer. - Start with 3-layer HEA ansatz. - After 50 iterations: check cost convergence. If plateau, add 1 layer. If still converging, continue. - Learning rate decay: halve every 100 iterations without improvement. - Maximum: 10 ansatz layers.

Level 5 — Hardware monitor. - Re-check best-qubit set every hour. - Trigger re-routing if best set changes. - Request recalibration if fidelities drop below 99.3%.

Total runtime. Several hours for a well-converged VQE. Most hours are Level 2+3 (shot budget); architecture changes at Level 4 happen maybe 20 times total; hardware adjustments at Level 5 maybe 2-5 times.

Information flow in a sample iteration: 1. Level 5 reports "hardware is stable, current qubit mapping is fine". 2. Level 4 says "stay at 5 layers, keep learning rate at 0.025". 3. Level 3 allocates 1000 shots, picks mitigation stack. 4. Level 2 runs the HOPSO update, produces new parameters. 5. Level 1 executes the circuit with adaptive symmetry correction. 6. Measurement results flow back up: Level 1 $\rightarrow$ estimator $\rightarrow$ optimizer $\rightarrow$ meta-optimizer. 7. Each level logs its state and decides whether to act.

Implementation Challenges

Building a fully closed-loop VQA is significantly harder than building a static one. Major challenges:

1. Software architecture

Classical ML libraries don't directly support the multi-level structure. You need to write the meta-controller from scratch, usually as a state machine in Python that orchestrates Qiskit/PennyLane/Cirq calls.

Tooling. Mitiq for mitigation, Qiskit Runtime for mid-circuit feedback, Ray or Dask for parallel shot execution. None of these alone suffice; you have to integrate them.

2. Latency

Mid-circuit feedback requires sub-microsecond response time; hardware monitor updates can take minutes. The whole system must handle these mismatched latencies without blocking.

Pattern. Asynchronous event loops with levels running as independent coroutines. Each level has its own clock and queues events.

3. Debugging

When something goes wrong, it's hard to tell which level is responsible. The layers hide each other's behavior. Debugging requires logging at every level plus visualization tools.

Best practice. Record the full state of each level at every decision point. Use time-series visualization to see where the anomaly originated.

4. Stability analysis

Interactions between levels can destabilize the whole system. A meta-optimizer that changes the ansatz too aggressively can destroy the inner optimizer's convergence. Formal stability analysis (Lyapunov-based) is available for small systems but impractical for full 5-level VQAs.

Practical test. Run the system on a known-answer benchmark and check that it converges. Compare to a simpler baseline (static VQA) to make sure the complexity is buying something.

5. Reproducibility

Closed-loop VQAs have many hidden state variables (the meta-optimizer's history, the hardware monitor's records, the estimator's adaptive state). Reproducing a result requires logging all of it.

Best practice. Serialize the entire state of the system at each iteration. Enable replay from any saved state.

When To Close The Loop

A practical question: *is the complexity worth it?* Closed-loop VQAs are harder to build, debug, and analyze than static ones. When does the benefit justify the cost?

Close the loop when:

- The hardware is drifting significantly over the VQA's runtime (Level 5 matters).
- The optimizer is struggling with noise (Level 3 matters).
- The problem has non-trivial structure that standard ansätze don't match (Level 4 matters).
- You're running many similar VQAs and can amortize the engineering cost.

Don't close the loop when:

- The problem is small enough that a static VQA converges fast.
- Hardware is stable within the VQA runtime.
- The engineering cost exceeds the compute saving.
- You're prototyping or debugging (start static, add loops later).

Practical rule. Start static; add closed-loop feedback only when you see the specific failure mode that each level addresses.

Thesis View

The thesis's framing of closed-loop VQAs:

1. **The closed-loop architecture is the natural deployment form of HOPSO.** HOPSO is an optimizer at Level 2; the higher levels provide the adaptive context that makes HOPSO robust.
2. **Separation of objective and dynamics is critical at every level.** The objective generator can be modified at Levels 3-5 (mitigation, hardware selection, architecture) without touching Levels 1-2 (the quantum dynamics and parameter optimization). This is the right architectural boundary.
3. **The loop-closure principle generalizes to multi-device settings.** Multiple quantum devices can be seen as multiple plants; the classical controller coordinates across them.
4. **Closed-loop VQAs are the NISQ bridge to fault-tolerance.** They extract maximum value from noisy devices in a way that static VQAs cannot. As hardware improves, the architecture remains; only the specific feedback policies change.

The thesis's claim: **the closed-loop architecture is the structural answer to the question "how do you make VQAs work on real hardware?"** Not a specific technique, but a framework for integrating all the techniques from Modules 1-9 into a working system.

Structural Role

This node is the **integration** of Module 10. It combines the ideas from 10.1 (architecture), 10.2 (adaptive circuits), 10.3 (RL), and 10.4 (quantum control) into a single coherent closed-loop VQA design, with five levels of feedback.

It is the closest thing Module 10 has to a blueprint. A reader who has completed the course should be able to build a closed-loop VQA based on this node alone, using components from the rest of the modules.

Relations to Other Concepts

- **Module 7.6 (Feedback control)** — the original two-loop framing.
- **Module 10.1 (Adaptive architectures)** — the three-loop structure this node extends.
- **Module 10.2 (Measurement-adaptive circuits)** — Level 1.
- **Module 10.3 (RL in VQA)** — a candidate for Level 4.
- **Module 10.7 (HOPSO as controller)** — the thesis's Level 2 instantiation.
- **Control engineering: hierarchical control** — the parent literature.

Worked Example — The Full Closed-Loop VQE Run

Problem. 16-qubit VQE for a chemistry system. Full closed-loop architecture.

Levels active: - Level 1: symmetry correction for particle number. - Level 2: HOPSO with 8 particles. - Level 3: adaptive shot allocation + ZNE every 10 iterations. - Level 4: ADAPT-VQE-style ansatz growth every 50 iterations. - Level 5: hardware monitor, re-routing on drift.

Run log (abbreviated):

Time	Event	Level
0:00	Start. Ansatz: 2-layer HEA on qubits {5,7,11,14,...}	Setup
0:01	Iter 1. Cost -75.12	L2
0:05	Iter 5. Gradient variance high — more shots allocated	L3
0:15	Iter 15. ZNE applied — corrected cost -75.25	L3
0:25	Iter 25. Cost plateau detected. Ansatz growth: +1 operator (largest gradient)	L4
0:30	Iter 30. Cost -75.41 — growth worked	L2
0:45	Iter 45. Hardware drift detected on qubit 11. Reroute to qubit 16	L5
0:48	Iter 48. Re-routed. Cost -75.40 (slight regression from routing transition)	L2
1:15	Iter 75. Ansatz growth: +2 operators	L4
2:00	Iter 120. Learning rate halved	L4
3:30	Iter 210. Ground state energy -75.64 reached. Within chemical accuracy. Stop.	Convergence

Observations. The system made decisions at every level throughout the run. Each decision was triggered by feedback from a lower level. The overall run is 3.5 hours, which would have been impossible for a static VQA on the same hardware without mitigation and adaptation.

Success criterion. Reach chemical accuracy on a 16-qubit problem with 99.5% gate fidelity and drifting calibration. Achieved.

Visualization

Pending. A timeline showing the five levels of a closed-loop VQA running in parallel, with events at each level plotted on a time axis. Lower levels tick fast; higher levels tick slow. Arrows show feedback events between levels. Caption: "The closed loop in action: five levels, five clocks, one system."

Exercises

1. For a 5-level closed-loop VQA, write down the maximum allowed frequency of decisions at each level, given a total runtime of 1 hour. How much compute is spent on decisions vs cost evaluation?
2. Analyze a failure mode where Level 4 (meta-optimizer) changes the ansatz too fast. How does the failure manifest at Level 2? How would you detect it?
3. (VQA) Implement a simple 3-level closed-loop VQA for a 4-qubit VQE: Level 1 is cost evaluation, Level 2 is Adam, Level 3 is adaptive learning rate decay. Run it and log the decisions at each level.

Research Extensions

- **Formal convergence proofs for 5-level closed-loop VQAs** — currently only exist for 2-level systems.
- **Distributed closed-loop VQAs** — multiple hardware backends coordinated by a common meta-optimizer.
- **Online architecture search in closed-loop** — meta-optimizer that searches over many ansatz families simultaneously.
- **Safety and stability guarantees** — Lyapunov-based analysis for 5-level systems.
- **Cross-problem meta-optimizers** — one meta-optimizer serving many VQA instances with transfer learning.

Cross-links to Other Courses

- **Hierarchical Control Systems (Antsaklis)** — the classical theory.
- **Systems Engineering** — multi-level integration.
- **Module 7.6 (Feedback Control)** — the original framing.
- **Module 10.1-10.4** — the components of the closed-loop architecture.
- **Module 10.7 (HOPSO as controller)** — the thesis's specific instantiation.

Variational Quantum Algorithms · Module 10 — Vqa As Adaptive Control Systems · Node 10.5
Concepts: feedback, adaptive-systems, control-theory, stability
Prerequisites: 10.1, 10.2, 10.3, 10.4
Enables: 10.6, 10.7

hbar.university — own.your.cognition