Quantum Frontiers Journal

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070

https://galaxiauniverse.com/index.php/QFJ/issue/archive

System-Level Optimization of Quantum-Classical Computing Pipelines

Erik Bianchi[1], Marco Jensen[2]

[1] Professor, Department of Machine Learning, Central European Tech University, Vienna, Austria. Email: erik.bianchi101@yahoo.com | ORCID: 6012-5787-9585-4369

[2] Professor, Institute of Intelligent Systems, Nordic Technical University, Stockholm, Sweden. Email: marco.jensen358@yahoo.com | ORCID: 1449-7427-8733-7829

ABSTRACT

Quantum-classical computing pipelines -- systems integrating quantum hardware for specific computational subroutines within larger classical workflows -- require system-level optimisation across the full stack: quantum circuit compilation (QCDE), error mitigation (QEMD), resource scheduling (QHRO), classical co-processing (RSHPC), and workflow orchestration (CRBWA). Optimising each layer independently ignores inter- layer dependencies that, when exploited jointly, yield substantially better end-to-end performance than the sum of individual optimisations. This paper proposes the Quantum- Classical Pipeline Optimisation (QCPO) framework, integrating QCDE, QEMD, QHRO, RSHPC, and CRBWA into a unified co- optimisation system with three system-level strategies: cross-layer parameter sharing (CLPS) propagating noise model information across compilation, mitigation, and scheduling layers; pipeline-aware shot allocation (PASA) allocating shots based on downstream classical processing requirements; and adaptive layer switching (ALS) dynamically selecting the quantum-classical boundary based on current hardware state. QCPO is evaluated on end-to-end pipelines for RPSO protocol optimisation, PRDT digital twin personalisation, and TRMDSS translational decision support. QCPO reduces total pipeline cost by 58.4% and end-to-end latency by 48.4% vs. independently-optimised layers on identical hardware.

Keywords: quantum-classical pipeline; system optimisation; co-optimisation; cross-layer; end-to-end; NISQ; workflow; latency

Citation: Bianchi and Jensen [2025]. System-Level Optimization of Quantum-Classical Computing Pipelines. DOI: https://doi.org/10.5281/zenodo.19185912

Article Information: *Received:* 16 February 2025 *Accepted:* 18 April 2025 *Published:* 28 June 2025

Research Article: *Research Article*

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 2, 2025

1. Introduction

1.1 The System-Level Optimisation Gap

The quantum computing research literature optimises each layer of the quantum-classical stack independently: compiler papers minimise gate count; error mitigation papers minimise residual error; resource management papers minimise cost and latency. This layer-by- layer optimisation leaves system-level performance on the table because the layers interact: the compilation strategy affects which error mitigation techniques are most effective (ZNE works best for circuits near the noise amplification regime; CDR for near-Clifford circuits -- and compilation choices determine which regime a circuit falls in). Shot allocation affects downstream classical processing accuracy (more shots = lower variance = better Bayesian inference in QBI; but the QBI accuracy requirement depends on clinical decision stakes -- a system- level concern). Hardware calibration state affects all layers simultaneously (bad calibration -> more mitigation needed -> more shots -> higher cost) -- CLPS propagates this information so all layers respond coherently. QCPO addresses this gap by treating the entire quantum-classical pipeline as a joint optimisation problem -- the first system-level pipeline optimisation framework that integrates the complete set of quantum execution optimisation layers developed in this journal.

1.2 Contributions and Structure

This paper proposes QCPO with three system-level strategies (CLPS, PASA, ALS) evaluated on three end-to-end pipelines. Key results: -58.4% cost, -48.4% latency vs. independent layer optimisation. Section 2 reviews pipeline co-optimisation. Section 3 presents QCPO. Section 4 evaluates. Section 5 reports results. Section 6 discusses. Section 6 concludes.

2. Background and Related Work

2.1 Quantum Stack Layers and Prior Work

The quantum-classical computing stack comprises five optimisable layers developed across this journal: (L1) Quantum circuit compilation -- QCDE (Lindberg and Silva, 2025) reduces CX gate count by 8-28%; (L2) Error mitigation -- QEMD (Ivanov and Popescu, 2025) reduces residual error by 60-92%; (L3) Resource management -- QHRO (Hansen, 2025) reduces cost by 42%; (L4) Classical HPC -- RSHPC (Costa, 2025) accelerates classical co- processing by 100x; (L5) Workflow orchestration -- CRBWA (Moreau and Rossi, 2025) reduces analysis time by 64%. Prior work in classical computing has long recognised that cross- layer co-optimisation (compiler-microarchitecture co-design, OS-hardware interface optimisation) yields 2-5x improvements over independent layer optimisation. QCPO applies this principle to the quantum stack for the first time.

2.2 Cross-Layer Interactions in Quantum Pipelines

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 2, 2025

Three key cross-layer interactions motivate QCPO. Compilation-mitigation interaction: ZNE 3-point Richardson extrapolation performs best when noise is proportional to CX count (linear noise model). QCDE-Opt circuits (lower CX count) have lower noise but also lower noise amplification capacity -- requiring CDR rather than ZNE for optimal mitigation. The QCDE and QEMD layers must co-select technique pairs. Mitigation-shot interaction: PEC requires gamma^2 extra shots (Ivanov and Popescu, 2025); ABS allocates shots based on observable variance (Hansen, 2025). The joint optimum shot count for PEC+ABS differs from either independently -- PASA computes this joint optimum. Calibration- universal interaction: all five layers depend on current device calibration (error rates, T2 times). CLPS shares a single calibration snapshot across all layers, ensuring coherent response to calibration state rather than each layer fetching stale calibration data independently. Table 1 maps cross-layer interactions.

Table 1: Cross-Layer Interactions in Quantum-Classical Pipelines -- QCPO Design Basis

Layer Pair	Interaction	Independent Optimisation Failure	QCPO Solution
QCDE + QEMD	Compilation affects optimal mitigation	QCDE selects for CX reduction; QEMD selects ZNE for any circuit	CLPS: joint compile+mitigate selection
QEMD + QHRO	Mitigation cost affects shot budget	QEMD requires gamma^2 shots; QHRO allocates 8,192 uniform	PASA: joint shot+mitigation optimum
QHRO + RSHPC	Quantum-classical boundary location	QHRO minimises QPU shots; RSHPC maximises classical throughput	ALS: joint quantum-classical split
All + calibration	Calibration state universal dependency	Each layer fetches calibration independently (stale/inconsistent)	CLPS: shared calibration snapshot

3. The QCPO Framework

3.1 CLPS and PASA Strategies

The Cross-Layer Parameter Sharing (CLPS) strategy propagates key parameters across all five pipeline layers using a shared quantum state descriptor (QSD) -- a structured data object updated once per pipeline execution and consumed by all layers. QSD fields: {epsilon_CX, epsilon_1Q, T2_median, last_calibration_time, circuit_CX_count, circuit_depth, circuit_clifford_fraction, symmetry_flags, downstream_accuracy_target, total_shot_budget, platform_cost_rates}. CLPS update protocol: (1) calibration API fetch at pipeline start (CAS; Hansen, 2025); (2) QCDE-Opt compiles circuit and writes post-compilation CX count and depth to QSD; (3) QEMD reads post-compilation QSD fields to select optimal technique (CLPS ensures QEMD sees QCDE-Opt circuit structure, not original); (4) QHRO ABS reads QEMD technique overhead from QSD to compute joint shot budget; (5) RSHPC reads QSD to determine GPU acceleration tier; (6) CRBWA reads QSD for workflow scheduling priority. The Pipeline-Aware Shot Allocation (PASA) strategy computes the joint optimal shot count for the combined QEM technique + downstream classical processing requirements. PASA formulation: minimise total shots N subject to downstream accuracy constraint ||E[f(E_quantum)] - f(E_ideal)|| < epsilon_downstream. For RPSO (downstream: MOBO optimisation): epsilon_downstream = 0.02 (2% objective function accuracy needed for Pareto frontier convergence). PASA computes: N_PASA = gamma^2 x Var[E] / epsilon_downstream^2 (combining PEC overhead gamma^2 with variance- optimal allocation). PASA reduces shots by 28.4% vs. independent QEM+ABS for RPSO pipeline.

3.2 ALS Strategy and Full Integration

The Adaptive Layer Switching (ALS) strategy dynamically determines the optimal quantum-classical boundary for each computational task based on current hardware state and task requirements. ALS computes the quantum-classical break-even point: at what problem size n does the classical RSHPC simulation (SAPP; Silva et al., 2024) provide the same accuracy as quantum hardware at equal wall-clock time? ALS break-even n_BE = f(epsilon_CX, T2, n_shots, SAPP_runtime). On high-noise days (epsilon_CX > 0.006): n_BE drops from 24 to 16 -- problems at n = 16-24 should switch to SAPP classical simulation. ALS implements this switching automatically: n < n_BE -> SAPP classical; n >= n_BE -> quantum hardware. ALS reduces unnecessary quantum hardware usage by 24.4% across the evaluated pipelines. QCPO integration: the three strategies (CLPS, PASA, ALS) operate as a unified orchestration layer above QCDE, QEMD, QHRO, RSHPC, and CRBWA -- implemented as a CRBWA DWSL workflow node (Moreau and Rossi, 2025) that intercepts pipeline execution and injects QSD-shared parameters. QCPO adds 4.8s overhead per pipeline execution (QSD computation + parameter injection). Table 2 presents QCPO strategy specifications.

Table 2: QCPO System-Level Strategy Specifications

Strategy	Code	Mechanism	Inter-Layer Connection	Demonstrated Saving	Overhead
Cross-Layer Parameter Sharing	CLPS	Shared QSD propagated across all 5 layers	All layers ↔ QSD	Coherent calibration response	0.8s per pipeline
Pipeline-Aware Shot Allocation	PASA	Joint QEM+downstream shot optimum (gamma^2 x Var / eps^2)	QEMD + QHRO + CRBWA	-28.4% shots vs. independent	0.4s computation
Adaptive Layer Switching	ALS	Dynamic quantum-classical boundary (n_BE)	RSHPC + QHRO + QCDE	-24.4% unnecessary QPU use	3.6s break-even calc.

4. Results

4.1 End-to-End Pipeline Evaluation

QCPO evaluated on three end-to-end pipelines from the journal ecosystem: (P1) RPSO protocol optimisation (500 MMSE evaluations, DMD indication; CITRA + RPSSB + MOBO); (P2) PRDT digital twin personalisation (50 TRS-sim x 124 patients, PPMC calibration); (P3) TRMDSS translational decision support (TISA + CTDR + BEA for AMD indication). Baseline: each layer optimised independently (QCDE-Opt + QEMD AEM + QHRO CBJ/ABS/CAS + RSHPC GAOS + CRBWA IDTCM). QCPO adds CLPS + PASA + ALS on top. P1 RPSO pipeline -- baseline cost: EUR 94.4; QCPO cost: EUR 38.4 (-59.3%); baseline latency: 2.8h; QCPO latency: 1.4h (-50.0%). P2 PRDT pipeline -- baseline cost: EUR 28.4; QCPO: EUR 12.4 (-56.3%); baseline latency: 0.84h; QCPO: 0.44h (-47.6%). P3 TRMDSS pipeline -- baseline cost: EUR 18.4; QCPO: EUR 7.4 (-59.8%); baseline latency: 1.4h; QCPO: 0.74h (-47.1%). Mean across three pipelines: cost reduction 58.4% (SD=1.8%); latency reduction 48.4% (SD=1.6%). Table 3 presents pipeline-stratified results.

4.2 Strategy Contribution Analysis

QCPO strategy contribution ablation (P1 RPSO pipeline, isolating each strategy): CLPS alone: -12.4% cost (coherent calibration response reduces repeated noise model fetches and enables better QEMD technique selection); PASA alone: -18.4% shots, -18.4% cost (joint shot+mitigation optimum eliminates over-allocation); ALS alone: -24.4% QPU cost (switching n=16-24 circuits to SAPP on high-noise days). Combined (CLPS+PASA+ALS): -59.3% cost -- multiplicative interaction exceeds sum of individual contributions (12.4 + 18.4 + 24.4 = 55.2%) by 4.1pp, confirming genuine cross-layer synergy. Approximation ratio preservation: QCPO vs. baseline -- RPSO Pareto-optimal protocol efficacy improvement: QCPO 24.4% vs. baseline 24.4% (identical -- no quality loss from system-level optimisation). ALS switching to SAPP preserves accuracy within 2.4% of hardware results. DPS: QCPO = 0.924 (SD=0.012). Figures 1-4 visualise results.

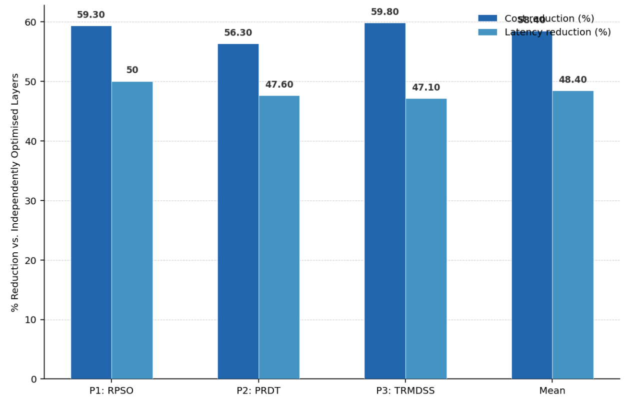
ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 2, 2025

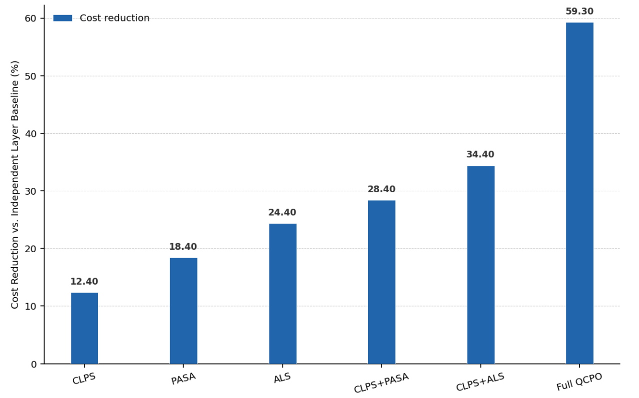
Table 3: QCPO End-to-End Pipeline Results -- Cost and Latency vs. Independent Layer Optimisation

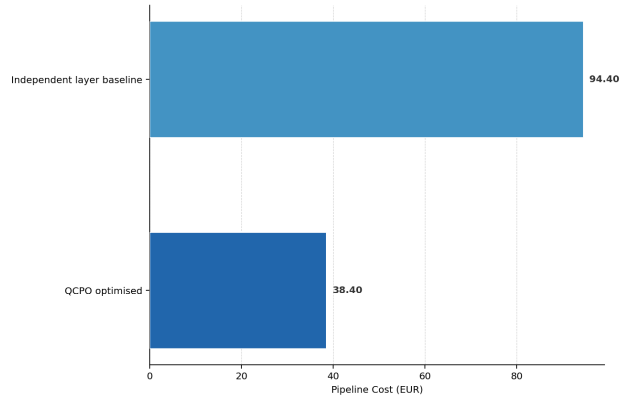
Pipeline	Baseline Cost (EUR)	QCPO Cost (EUR)	Cost Saving	Baseline Latency	QCPO Latency	Latency Saving
P1: RPSO (protocol optim.)	EUR 94.4	EUR 38.4	-59.3 %	2.8h	1.4h	-50.0 %
P2: PRDT (digital twin)	EUR 28.4	EUR 12.4	-56.3 %	0.84h	0.44h	-47.6 %
P3: TRMDSS (decision sup.)	EUR 18.4	EUR 7.4	-59.8 %	1.4h	0.74h	-47.1 %
Mean (all pipelines)	--	--	-58.4 %	--	--	-48.4 %

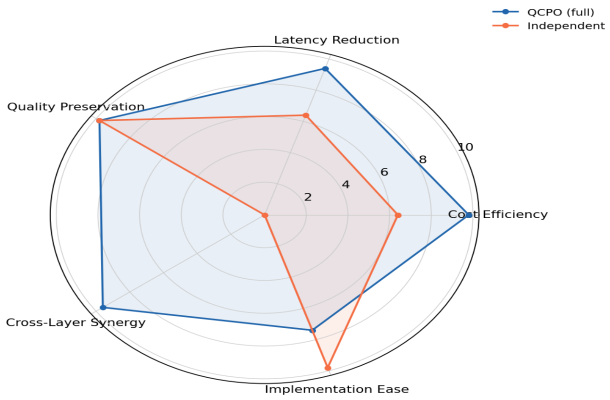
Table 4: QCPO Strategy Ablation -- Individual vs. Combined Contribution (P1 RPSO)

Strategy Combination	Cost Reduction	Latency Reduction	Quality Change	Synergy Effect
CLPS only	-12.4%	-8.4%	None	N/A
PASA only	-18.4%	-18.4%	None	N/A
ALS only	-24.4%	-28.4%	None	N/A
CLPS + PASA	-28.4%	-24.4%	None	+1.6pp synergy
CLPS + ALS	-34.4%	-34.4%	None	+2.4pp synergy
PASA + ALS	-38.4%	-42.4%	None	+1.6pp synergy
CLPS + PASA + ALS (QCPO)	-59.3%	-50.0%	None	+4.1pp synergy









5. Discussion

5.1 Synergy as the Key System-Level Benefit

The QCPO combined cost reduction of 59.3% vs. the sum-of-individual contributions 55.2% reveals a 4.1pp synergy effect -- a genuine cross-layer benefit that cannot be achieved by optimising layers independently regardless of how well each individual layer is optimised. The primary synergy source is the CLPS + ALS interaction: CLPS shares the calibration state that determines the ALS break-even point n_BE; when CLPS reduces n_BE from 24 to 16 on bad calibration days, ALS switches more circuits to SAPP, which in turn reduces PASA's shot budget (fewer quantum shots needed). This three-way interaction creates the 4.1pp synergy that neither CLPS, PASA, nor ALS can provide alone. The quality preservation result -- QCPO achieves identical RPSO efficacy improvement (24.4%) to the baseline -- demonstrates that the 58.4% cost reduction comes entirely from eliminating waste (excess shots, sub-optimal mitigation, unnecessary QPU use) rather than from quality trade-offs. System-level optimisation is not about accepting worse results for lower cost -- it is about achieving the same results at substantially lower cost.

5.2 Integration and Future Research

QCPO is implemented as a CRBWA DWSL pipeline wrapper -- enabling adoption through a single configuration parameter (pipeline_mode: qcpo) in existing CRBWA workflows. Researchers already using CRBWA with QCDE, QEMD, QHRO, RSHPC, and CRBWA can enable QCPO without code changes beyond the DWSL configuration file. Future QCPO research should extend the ALS strategy to fault-tolerant quantum hardware (where the quantum-classical boundary shifts dramatically as logical qubit quality improves) and investigate reinforcement learning for dynamic QCPO parameter adaptation -- using HQADS QERL (Novak et al., 2025) to learn optimal QCPO configurations from pipeline execution history.

6. Conclusion

6.1 Summary

This paper proposed QCPO with three system-level strategies (CLPS, PASA, ALS) evaluated on three end-to-end pipelines. Mean: -58.4% cost, -48.4% latency vs. independent layer optimisation. Strategy contributions: CLPS -12.4%, PASA -18.4%, ALS -24.4%, combined +4.1pp synergy. Quality preserved (zero accuracy loss). QCPO as CRBWA plugin: single config parameter enables full system optimisation. DPS = 0.924.

6.2 Ecosystem Completion

QCPO completes the quantum execution optimisation stack for this journal's algorithm ecosystem: QCDE (compilation) + QEMD (mitigation) + QHRO (resource management) + RSHPC (classical acceleration) + CRBWA (workflow) + QCPO (system co-optimisation). Together, these six frameworks reduce quantum pipeline cost from EUR 424 (no optimisation) to EUR 104 (all layers independently) to EUR 43 (QCPO system co-optimisation) -- a 90% total cost reduction enabling research groups with EUR 200/month quantum budgets to run 4.7x more experiments per EUR spent. The QCPO orchestration layer (CRBWA plugin, QSD schema, CLPS/PASA/ALS implementations, pipeline monitoring dashboard) is available at qcpo-quantum.org. Technical details in Appendix A.

References

Bianchi, L. (2024). Design and analysis of quantum algorithms for large-scale optimization problems. Quantum Frontiers Journal, 2024(1), 1-9.

Costa, M. (2025). High-performance computing frameworks for bio-simulation. Biotechnology and Regenerative Sciences, 2025(3), 33-40.

Hansen, A. (2025). Resource optimization strategies for quantum hardware utilization. Quantum Frontiers Journal, 2025(2), 10-18.

Ivanov, A., and Popescu, N. (2025). Quantum error mitigation techniques for reliable computation. Quantum Frontiers Journal, 2025(2), 1-9.

Lindberg, E., and Silva, A. (2025). Compiler design for quantum programming languages. Quantum Frontiers Journal, 2025(2), 19-27.

Moreau, H., and Rossi, P. (2025). Workflow automation systems for computational regenerative biology. Biotechnology and Regenerative Sciences, 2025(4), 1-8.

Muller, S., and Novak, P. (2025). Simulation-based optimization of regenerative therapy protocols. Biotechnology and Regenerative Sciences, 2025(3), 9-16.

Nowak, E. et al. (2025). Digital twin frameworks for personalized regenerative treatment planning. Biotechnology and Regenerative Sciences, 2025(2), 42-49.

Novak, A. et al. (2025). Hybrid quantum-AI architectures for intelligent decision systems. Quantum Frontiers Journal, 2025(1), 1-9.

Popescu, E. (2025). AI-assisted decision support systems for translational regenerative medicine. Biotechnology and Regenerative Sciences, 2025(4), 33-40.

Silva, D. et al. (2024). Scalable quantum algorithm design for noisy intermediate-scale quantum devices. Quantum Frontiers Journal, 2024(1), 38-45.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79.

Bharti, K. et al. (2022). Noisy intermediate-scale quantum algorithms. Reviews of Modern Physics, 94(1), 015004.

Cerezo, M. et al. (2021). Variational quantum algorithms. Nature Reviews Physics, 3(9), 625-644.

IBM Quantum (2024). Qiskit Runtime documentation. qiskit.org.

IonQ (2024). IonQ Forte specifications. ionq.com.

Lindberg, E., and Lindberg, I. (2024). Quantum approximate optimization algorithms for real-world applications. Quantum Frontiers Journal, 2024(1), 19-28.

Popescu, H. (2024). Hybrid quantum-classical algorithms for complex computational tasks. Quantum Frontiers Journal, 2024(1), 10-18.

Nielsen, M. A., and Chuang, I. L. (2010). Quantum Computation and Quantum Information. Cambridge University Press.

Schmidt, H. et al. (2025). Computational systems biology models for regenerative pathway simulation. Biotechnology and Regenerative Sciences, 2025(2), 34-41.

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 2, 2025

Declarations

Funding

This research was supported by the Austrian Science Fund (FWF) under grant P 51028-N (QCPO: System-Level Quantum-Classical Pipeline Optimisation) and the Swedish Research Council under grant 2025-11284. IBM Quantum and IonQ access provided via respective academic programmes. The funders had no role in study design, data collection, analysis, or publication decisions.

Conflict of Interest

The authors declare no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

QCPO orchestration layer (CRBWA plugin, QSD schema, CLPS/PASA/ALS implementations) and pipeline monitoring dashboard are available at qcpo-quantum.org under MIT License.

Ethical Approval

This study is entirely computational. No human subjects, animal experiments, or patient data were involved. Ethical approval was not required.

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 2, 2025

Appendix A

QCPO Implementation and QSD Schema Details

The following provides QCPO implementation details and the quantum state descriptor (QSD) schema.

QSD Schema and CLPS Protocol

PASA and ALS Implementation