Towards A Distributed QHPC Workflow for Gadget-Aware Circuit Cutting and Pulse Synthesis

Ria Patel
North Carolina State University
Raleigh, NC, USA
rpatel38@ncsu.edu

Silvina Caíno-Lores
University of Rennes, Inria, CNRS, IRISA
Rennes, France
silvina.caino-lores@inria.fr

Frank Mueller
North Carolina State University
Raleigh, NC, USA
fmuelle@ncsu.edu

Abstract

Pulse-level synthesis of compound gadgets (repetitive macroscopic gate blocks) compresses quantum control footprints. However, classically optimizing these schedules via optimal control algorithms creates severe compute bottlenecks, while scaling deep algorithms exhausts classical memory. These constraints demand distributed quantum high-performance computing (QHPC) workflows, but current tools use gate- or wire-level cutting that destroy data boundaries required for compound pulse gadget synthesis. We introduce Kodoki, a pulse gadget optimization framework that enables gadget-aware circuit cutting. Coupled with Qdislib and PyCOMPSs, Kodoki provides a decoupled distributed workflow. PyCOMPSs orchestrates pulse synthesis offline. Online, workers are given subcircuits and query a lookup table (LUT) of pre-computed pulses to execute subcircuits. Evaluations show Kodoki's compound synthesis achieves a 4.38× reduction in pulse execution time (from $> 1000\mu$s to 240μs). Shifting synthesis offline replaces an 865.5s compute bottleneck with a 2.326ms online query (P_{99} = 3.35ms). Ongoing evaluations will quantify scaling behavior across variational ansatzes.

CCS Concepts

• **Hardware → Quantum computation**; • **Software and its engineering → Distributed systems organizing principles**; • **Computer systems organization → Quantum computing**; **Distributed architectures**.

ACM Reference Format:
Ria Patel, Silvina Caíno-Lores, and Frank Mueller. 2026. Towards A Distributed QHPC Workflow for Gadget-Aware Circuit Cutting and Pulse Synthesis. In *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'26).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Deep, highly structured quantum algorithms inherently rely on repeating macroscopic instruction blocks (gadgets). When executed on physical hardware, this extreme circuit depth leads to severe decoherence and error accumulation. Compound pulse gadget synthesis is a pulse-level optimization technique to mitigate this while also compressing these gadgets into highly optimized continuous-time pulses [4]. However, classically verifying these pulse schedules prior to hardware deployment requires heavy optimal control calculations. Optimal control algorithms such as GRAPE [2, 3] are utilized to synthesize these pulses classically. Since large-scale algorithms require computing dense matrices across fine- grained time steps, they exhaust the classical memory and computation time.

This necessitates the use of high-performance computing (HPC) within a workflow to perform these pulse optimizations prior to execution on quantum hardware. To distribute large quantum circuits, they must be divided into subcircuits via circuit cutting frameworks, such as Qdislib [6] and CutQC [5], and then scheduled across compute nodes by a workflow orchestrator like PyCOMPSs [1]. While these circuit cutting tools successfully partition circuits, they operate strictly at the gate or wire level. This partitioning severs the structural dependencies of the gadgets, destroying the data boundaries required for compound pulse gadget synthesis.

To address these gaps, we introduce **Kodoki**, a novel pulse gadget optimization framework. While Kodoki is flexible to any pulse-driven quantum species by design, our current classical evaluations target trapped-ion models to prepare for deployment on physical hardware with accessible pulse-level control. Kodoki's workflow operates in two stages: offline synthesis and online emulation. Offline synthesis partitions the problem by tagging gadgets, passing this information to Qdislib to prepare subcircuits via *gadget-aware circuit cutting*, and distributing them to compute nodes via PyCOMPSs. Each compute node generates the optimized pulses for its assigned macro- gates and caches them in a lookup table (LUT). During online emulation, worker nodes receive their logical subcircuits, query the LUT to fetch the pre-computed pulses, and run noisy Hamiltonian simulations on their optimized pulse schedules in parallel. Once verified, the distributed expectation values are algebraically reconstructed to determine the final state.

We evaluate Kodoki using the Quantum Singular Value Transformation (QSVT), an algorithmic framework that computes target matrix functions by applying polynomial approximations to a block-encoded (BE) matrix. While QSVT enables high-precision applications like molecular Hamiltonian simulation, it relies on deep, repetitive BE gate sequences. This triggers an explosion in memory and circuit depth, providing a stress test for classical memory limits. This also establishes a baseline for future studies in strong and weak scaling on Kodoki for larger molecular targets.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC'26, Chicago, IL
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

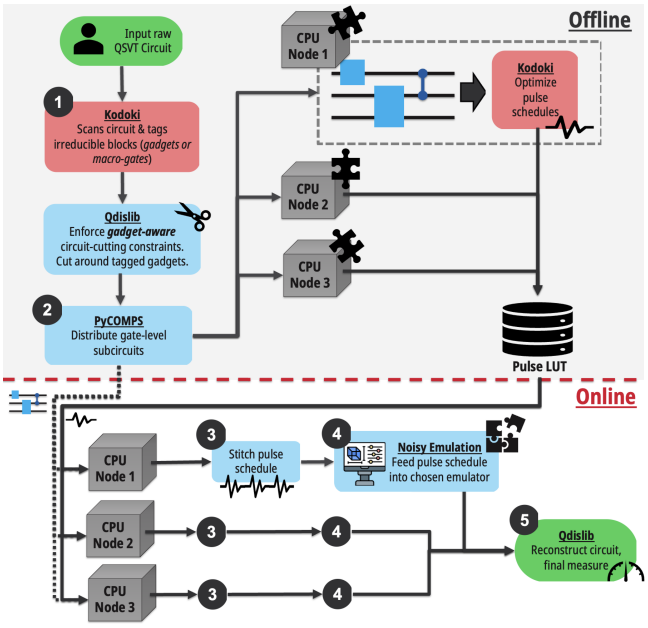


Figure 1: Kodoki's decoupled workflow: PyCOMPSs manages offline pulse gadget synthesis, while online workers fetch cached LUT pulses for parallel subcircuit emulation.

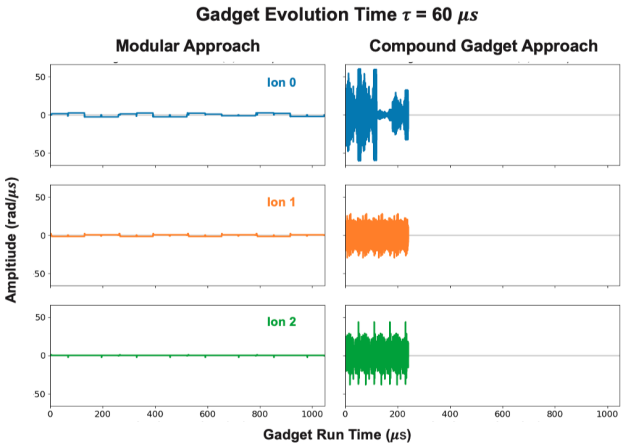


Figure 2: Pulse schedule generation for a QSVT gadget ($\tau = 60\mu$s). Compound synthesis achieves a 4.38× reduction in physical execution time versus the modular baseline.

2 Methodology

We integrated Kodoki into a QHPC workflow (Figure 1) that utilizes Qdislib and PyCOMPSs to achieve gadget-aware distributed emulation. The architecture strictly separates the offline pulse synthesis from online distributed subcircuit emulation through a centralized orchestration script.

The workflow is structured as a five-stage distributed scatter-gather model, explicitly separating offline quantum compilation from online workload execution:

Offline Synthesis (Pre-Compute):

- **1. Pre-Process & Cut (Scatter):** Kodoki tags gadgets based on hardware limits; Qdislib executes a constrained, gadget-aware cut to keep boundaries intact.
- **2. Distributed Synthesis (Map):** PyCOMPSs sends subcircuits to worker nodes for localized pulse synthesis of each subcircuit; stores outputs in LUT on master node.

Online Emulation (Execution):

- **3. Pulse Schedule Assembly (Fetch):** Workers query LUT to fetch and stitch pulse gadgets into their assigned subcircuits.
- **4. Noisy Emulation (Execute):** Workers run noisy emulator simulations on each assigned subcircuits to prevent centralized memory bottlenecks.
- **5. Reconstruction (Gather):** Qdislib on master node recombines subcircuit expectation values to measure the final state.

3 Evaluation and Ongoing Work

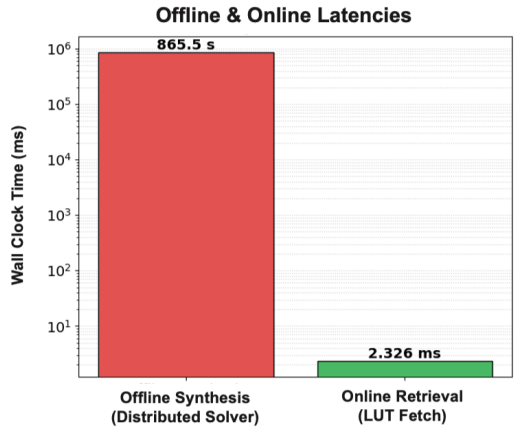
Our evaluation focuses on pulse compression and isolating classical compute bottlenecks. In Figure 2, the compound gadget approach compresses a QSVT block's pulse execution window from $> 1000\mu$s to 240μs, a 4.38× reduction compared to standard modular transpilation. This compression converges at $\tau \geq 60\mu$s and a maximum Rabi frequency $\geq$ 40rad/μs, maintaining a $\geq$ 26% fidelity. While this baseline fidelity is insufficient for fault-tolerant execution, it demonstrates the necessity of scaling Kodoki on institutional HPC clusters to push optimal control convergence toward hardware-viable thresholds.

To resolve the verification bottleneck, we evaluated Kodoki's workflow using a Docker container on a 16-core AMD EPYC Siena node (192 GB RAM) utilizing PyCOMPSs v3.3.3 and QuTiP v5.3.0. This validates that the orchestrator resolves PyCOMPSs DAG dependencies without severing gadget boundaries. Figure 3 shows the performance impact of decoupling synthesis from the online phase. This removes an 865.5s classical compute bottleneck from the critical path and replaces it with a 2.326ms online LUT fetch (P_{99} = 3.35ms).

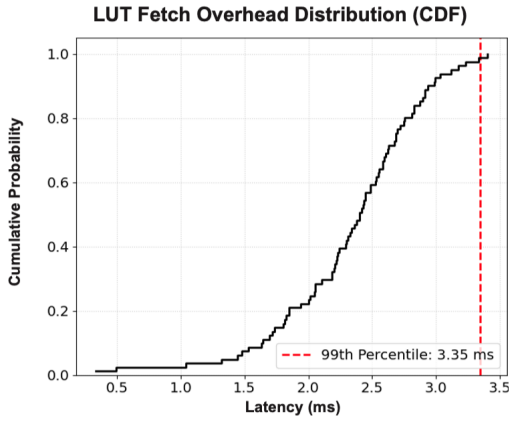
Ongoing work will deploy this workflow to NC State's ARC cluster for strong and weak scaling analyses. Distributed runtime profiling will isolate PyCOMPSs task orchestration and data transfer overheads from node-local GRAPE convergence rates. Finally, we will extend these evaluations beyond QSVT to Trotterized evolution and variational algorithms, quantifying Kodoki's performance and synthesized pulse fidelities across diverse gadgets.

4 Acknowledgments

This work was supported by the U.S. National Science Foundation (grants OSI-2531350, OMA-2120757, PHY-2325080) and the DOE Office of Science ASCR (Contract DE-SC0025384 for F. Mueller). Additional support was provided to S. Caíno-Lores by the French National Research Agency (JCJC ANR-25-CE46-6886-01), the France



(a) Offline vs. Online Latency



(b) LUT Fetch Overhead CDF

Figure 3: Offline vs. online phase latencies. Decoupling synthesis trades an 865.5s compute bottleneck for a 2.326ms LUT fetch (P_{99} = 3.35ms)

2030 HQI initiative (ANR-22-PNCQ-0002), and the Metropolitan Area of Rennes (AIS #2025002540).

References

[1] Rosa M. Badia, Javier Conejero, Jorge Ejarque, Daniele Lezzi, Francesc Lordan, and Raül Sirvent. 2026. Programming task-based workflows with COMPSs. en. doi:10.14279/DEPOSITONCE-25823.

[2] Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrüggen, and Steffen J. Glaser. 2005. Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms. en. *Journal of Magnetic Resonance*, 172, 2, (Feb. 2005), 296–305. doi:10.1016/j.jmr.2004.11.004.

[3] Neill Lambert et al. 2026. Qutip 5: the quantum toolbox in Python. *Physics Reports*, 1153, 1–62. doi:10.1016/j.physrep.2025.10.001.

[4] Ria Patel, Masoud Hakimi Heris, Yuan Liu, and Frank Mueller. 2026. Synthesizing compound pulse gadgets for hamiltonian simulation on trapped-ion platforms. (2026). https://arxiv.org/abs/2607.00826 arXiv: 2607.00826 [quant-ph].

[5] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. 2021. CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. arXiv:2012.02333 [quant-ph]. (Apr. 2021), 473–486. doi:10.1145/3445814.3446758.

[6] Mar Tejedor, Berta Cervera, Javier Conejero, Alba Cervera-Lierta, and Rosa M. Badia. 2025. Distributed Quantum Circuit Cutting for Hybrid Quantum-Classical High-Performance Computing. In *https://arxiv.org/abs/2505.01184*, 1–12.