Large-Scale Distributed State-Vector Simulation of IQPE for Early-FTQC

Shoma Hiraoka*, Yusuke Teranishi*†, Sho Nishida‡, and Wataru Mizukami§
*Graduate School of Information Science and Technology, The University of Osaka, Osaka, Japan
†QunaSys Inc., Tokyo, Japan
‡School of Engineering Science, The University of Osaka, Osaka, Japan
§Center for Quantum Information and Quantum Biology, The University of Osaka, Osaka, Japan

***Abstract*—Classical simulation lets us evaluate quantum algorithms for molecular energies before practical fault-tolerant quantum computers arrive. Large state vectors require distributed GPU memory, but communication can limit scaling. We present a complex-FP64 simulator for iterative quantum phase estimation (IQPE) that combines existing Lazy Qubit Reordering with two communication-saving mechanisms. On 1,024 NVIDIA H100 GPUs, single-digit IQPE runs achieve 90.2% and 106.0% strong-scaling efficiency for 32- and 34-qubit Cl_2 workloads, relative to their smallest multi-GPU baselines. An untapered 41-qubit Fe_2S_2 run completes one phase digit in 49.5 minutes.**

***Index Terms*—quantum circuit simulation, iterative quantum phase estimation, quantum chemistry, multi-GPU computing, state-vector simulation**

I. Motivation

Molecular energies help predict chemical stability and reactivity. Quantum phase estimation (QPE) estimates these energies by extracting the phase accumulated during quantum time evolution. Classical simulation allows us to evaluate these circuits before practical quantum hardware is available.

State-vector simulation stores 2^q complex amplitudes for q qubits. A 41-qubit complex-FP64 state requires 32 TiB. Multiple GPUs provide this aggregate memory, but operations spanning their partitions require data exchange.

Iterative QPE (IQPE) extracts d binary phase digits by measuring and reusing one auxiliary qubit, or ancilla. For n molecular qubits, it needs 2^{n+1} amplitudes rather than the 2^{n+d} required by conventional QPE. This saves simulation memory and qubits on early fault-tolerant quantum computers (Early FTQC). Long sequences of molecular operations still make communication a scaling challenge.

II. Communication-Aware Execution

We integrate existing Lazy Qubit Reordering (QR) [1] into molecular IQPE and add rank-global Z folding and rank-MSB probability reduction. Fig. 1 shows how these mechanisms reduce data movement.

Each GPU holds a partition of the state vector. Address bits index amplitudes within a GPU, while MPI-rank bits identify the owning GPU. Molecular evolution is implemented through Pauli rotations. Their X/Y factors couple amplitudes, while Z factors change phases. A rotation can execute locally when all X/Y targets occupy GPU-address bits. QR changes the mapping by swapping state-index bits, potentially transferring amplitudes between GPUs.

Topology-aware Lazy QR [1]. Compatible rotations reuse one mapping. The scheduler first minimizes inter-node InfiniBand swap pairs, then intra-node NVLink pairs. This avoids reordering before every non-local rotation.

Rank-global Z folding. A Z factor on an MPI-rank bit contributes a constant sign within that rank. When all X/Y targets are local, we fold this sign into the rotation coefficient. No amplitude transfer is needed, so more rotations can reuse the mapping.

Rank-MSB probability reduction. We place the ancilla at the most significant bit (MSB) of the MPI rank and exclude it from QR. Each GPU then holds amplitudes for ancilla 0 or 1. It computes its probability sum $w_r = \|\psi_r\|_2^2$. Reducing these scalars gives (P_0, P_1) without gathering amplitudes. The benchmark deterministically selects $j_k = \arg\max_b P_b$ for the next iteration rather than sampling.

III. Evaluation

Experiments used ABCI-Q with four H100 80 GB GPUs per node and one MPI rank per GPU. We used chemqulacs-gpu, cuStateVec 1.11.1, CUDA 12.9, and HPC-X 2.22.1 with complex-FP64 amplitudes.

Strong scaling. Both series execute one IQPE digit. We measure efficiency as $E(p; p_0) = p_0 T(p_0)/(pT(p))$, where $T(p)$ is runtime on p GPUs. Tapered Cl_2/STO-3G with 32 total qubits decreases from 558.91 s on two GPUs to 1.21 s on 1,024 GPUs, achieving 90.2% efficiency. Tapered Cl_2/6-31G with 34 total qubits decreases from 1,091.26 s on four GPUs to 4.02 s on 1,024 GPUs, achieving 106.0%. Each baseline is the smallest multi-GPU configuration used. The one-GPU reference uses a separate execution path. Timing alone does not explain the superlinear result.

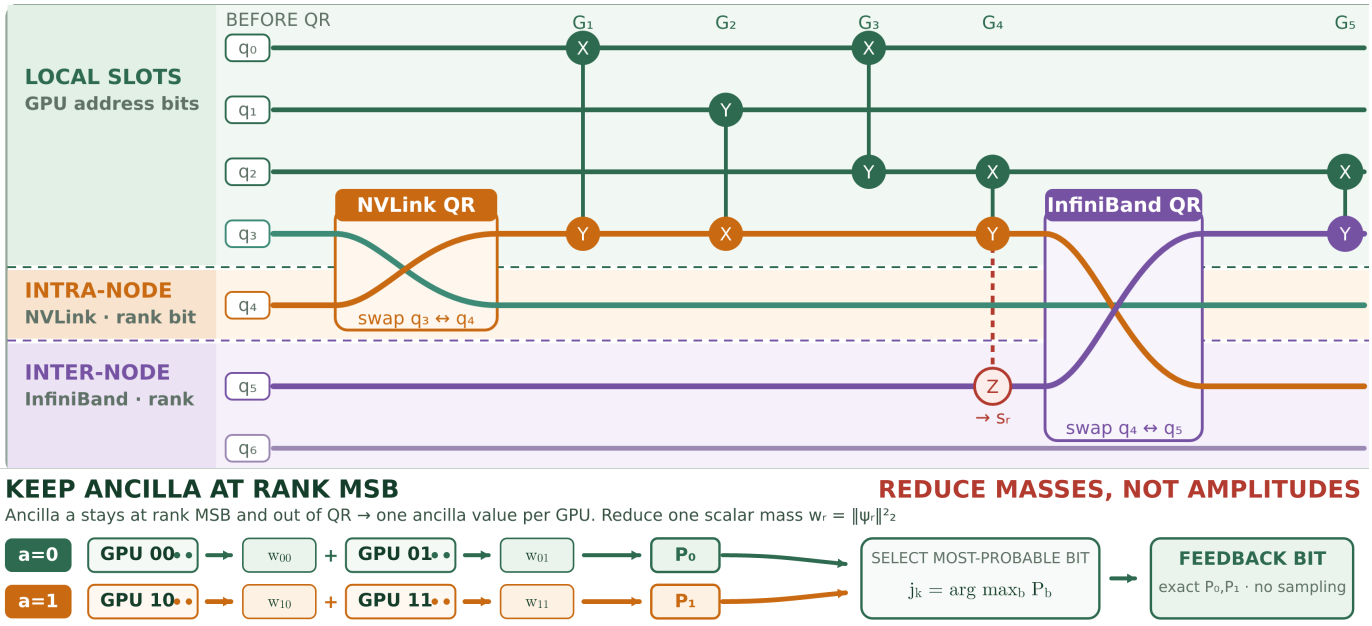


Fig. 1. The integrated path combines mapping reuse, rank-global Z folding, and rank-MSB probability reduction.

Large-scale execution. All following runs use 1,024 GPUs, and qubit counts include the ancilla. In our implementation, U^{2^k} repeats the base evolution, so d digits require $2^d - 1$ evolution units. The 34-qubit Cl_2/6-31G workload completes 11 digits in 120.6 min. Untapered 41-qubit Fe_2S_2 completes one digit over 232,793 Pauli terms in 49.5 min. Its tapered 39-qubit counterpart takes 11.9 min. Both use the same FCIDUMP input with 30 electrons in 20 orbitals. Other 39-qubit runs include SO_2/STO-3G with two digits in 9.3 min and tapered H_2O/cc-pVTZ with one digit in 4.1 min.

IV. Interpretation and Conclusion

The integrated simulator enables molecular IQPE execution on 1,024 H100 GPUs with near-ideal single-digit strong scaling for the two Cl_2 workloads. Digit counts describe executed phase-estimation depth, not demonstrated chemical accuracy. These measurements evaluate the integrated system. Lazy QR is benchmarked in Ref. [1]. Isolating the effects of our additions requires matched instrumented runs.

Acknowledgment

ABCI-Q resources were provided by AIST's Grand Challenge Program. OpenAI Codex (GPT-5) assisted with drafting and LaTeX conversion throughout this summary [2]. The authors verified all claims and remain responsible for the content.

References

[1] Y. Teranishi, S. Hiraoka, W. Mizukami, M. Okita, and F. Ino, "Lazy Qubit Reordering for Accelerating Parallel State-Vector-based Quantum Circuit Simulation," *ACM Transactions on Quantum Computing*, vol. 6, no. 4, 2025.

[2] OpenAI, "Codex (GPT-5)," AI-assisted drafting, language editing, and LaTeX conversion, accessed Aug. 7, 2026.