

ASPLOS 2026

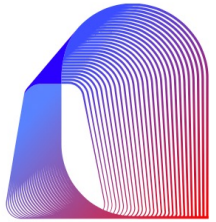
ReQISC: Reconfigurable Quantum Instruction Set Computers for High Performance Attainable on Hardware

Zhaohui Yang[1], David Ding[3], Qi Ye[2], Cupjin Huang[4], Jianxin Chen[2], Yuan Xie[1]









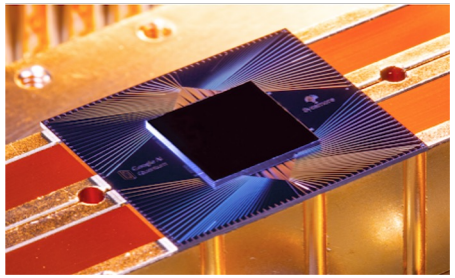
DAMO Quantum Laboratory

達摩院
DAMO ACADEMY

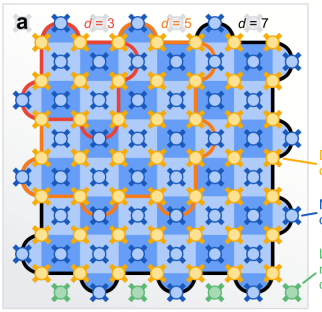


The quantum computer architecture challenge

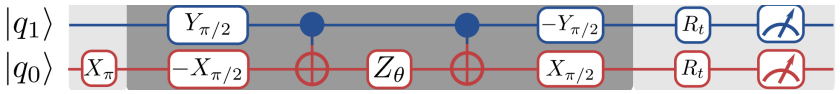
- Recent hardware milestones
  - Quantum supremacy, Quantum utility, QEC threshold
- Yet hardware noise limits program scale
  - Fidelity decays exponentially with **qubit count** × **circuit depth**
- *One critical approach for practical quantum advantage is advancing ISA-centric architectural support*
- Quantum ISA – the interface between hardware and software
  - State initialization, **universal gate set**, measurement
  - The microarchitecture (pulse control scheme) is not hard-wired, but reconfigurable!



Nature **574**, 505–510 (2019)

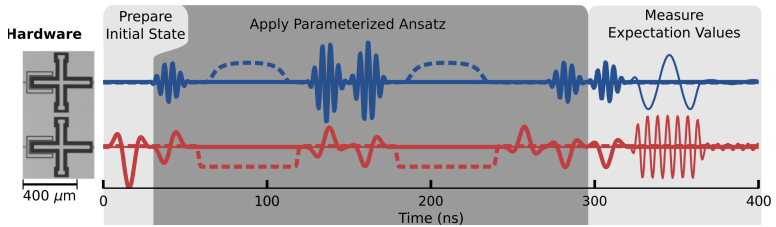


Nature **614** (2023);
Nature **638** (2025)



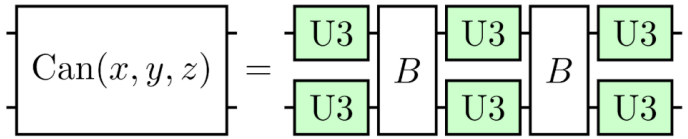
Abstract circuit representation

Pulse control scheme

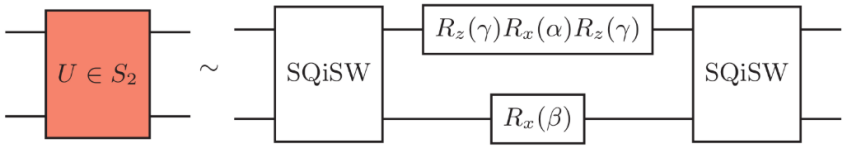


(Microwave) Pulse sequence timeline

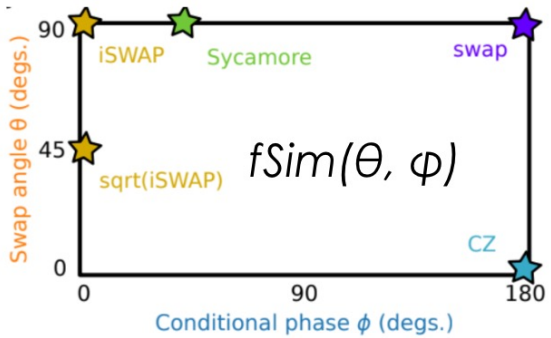
Beyond CNOT: Exploring expressive quantum ISAs



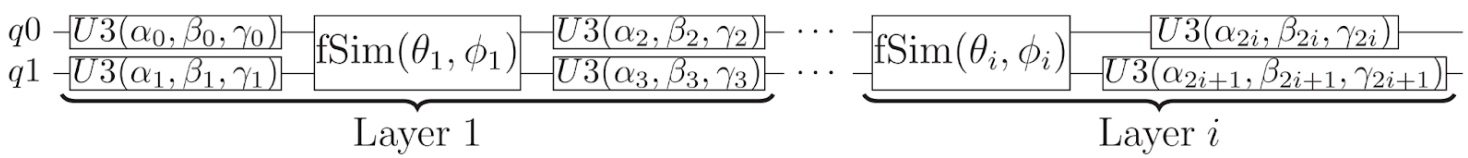
Zhang et al [PRL'04]



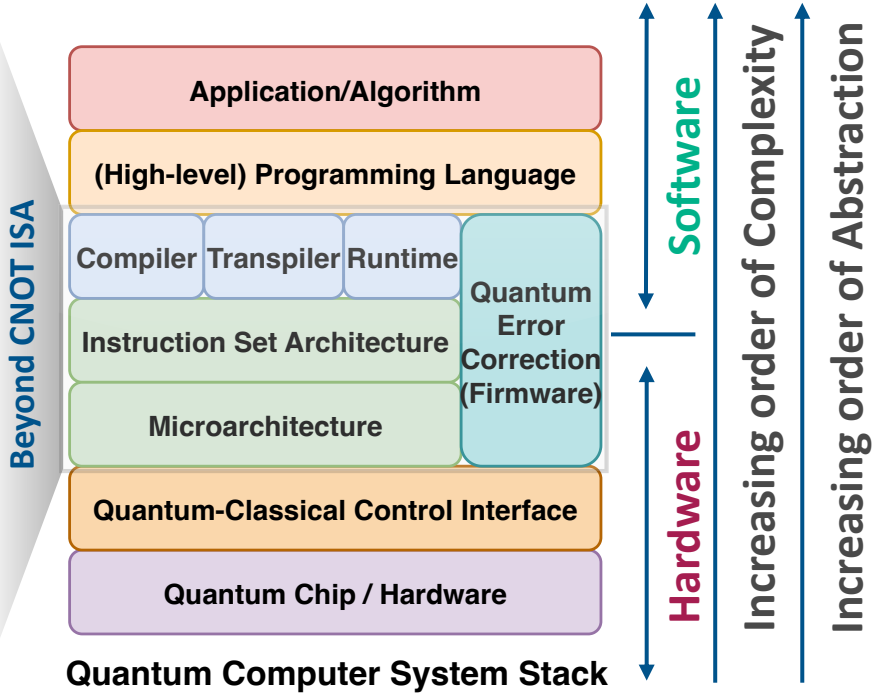
Huang et al [PRL'23], McKinney et al [HPCA'24]



Google [Nature'19]



Lao et al [ISCA'21]



- Metrics for "good" quantum ISAs
  - Software expressivity (e.g., complex, continuous gate sets are better)
  - Hardware efficiency (e.g., pulse duration, gate fidelity, control and calibration complexity)
- Expressive ISAs show theoretical superiority – but remain unproven in practice

Critical barriers to the practicality of expressive ISAs

Barrier 1: Hardware accuracy and calibration

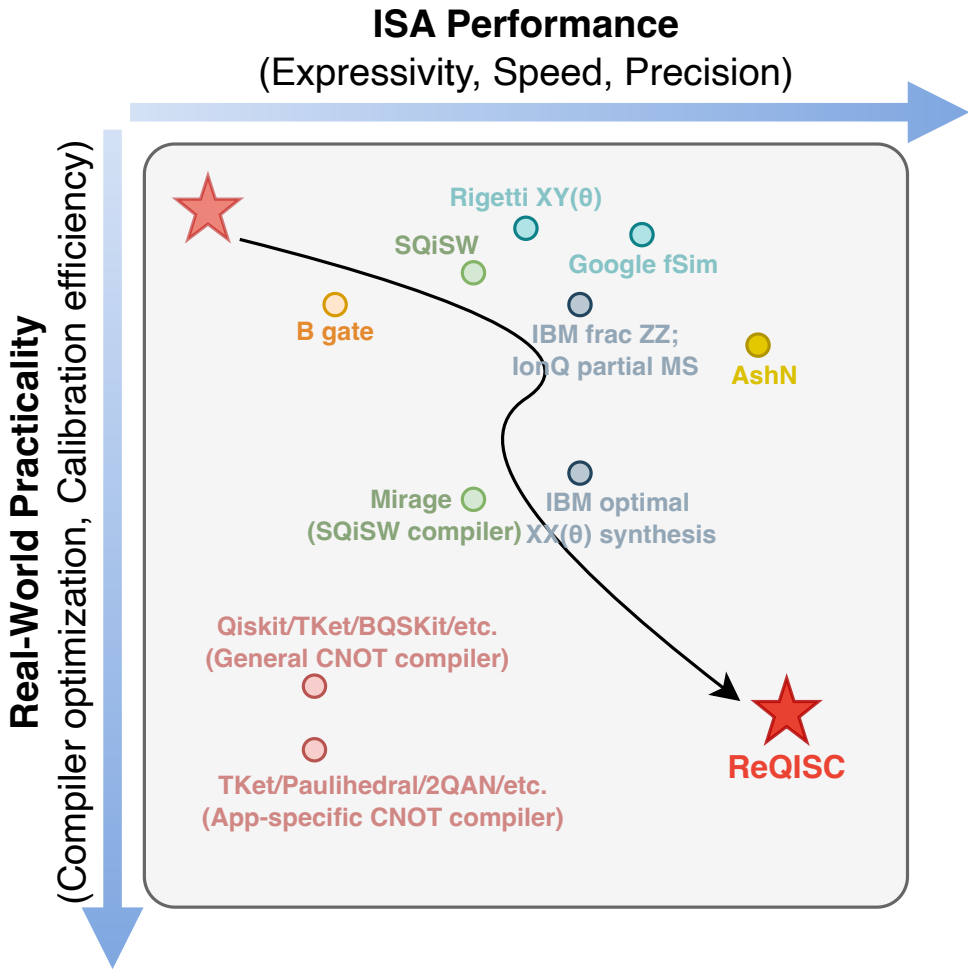
- Each distinct 2Q gate type requires individual calibration (pulse parameter tuning)
- More elements in an ISA → higher calibration overhead (hours per gate)
- Currently calibrated accuracy of advanced basis gates is not competitive with mainstream CNOT/CZ (e.g., IBM's fractional ZZ(θ) fidelity is inferior to its CZ)

Barrier 2: Compilation strategy

- Existing compiler suites (Qiskit, Tket, etc.) are designed for CNOT-basis ISA
- Naïvely rebasing circuits to a richer ISA does NOT capture theoretical benefits
- ISA-customized optimization is demanded at *every compilation stage*



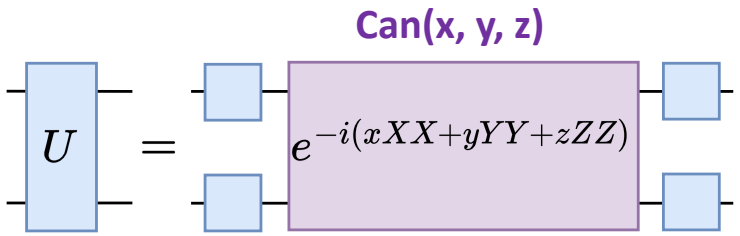
ReQISC (**Re**configurable **Q**uantum **I**nstruction **S**et **C**omputer)

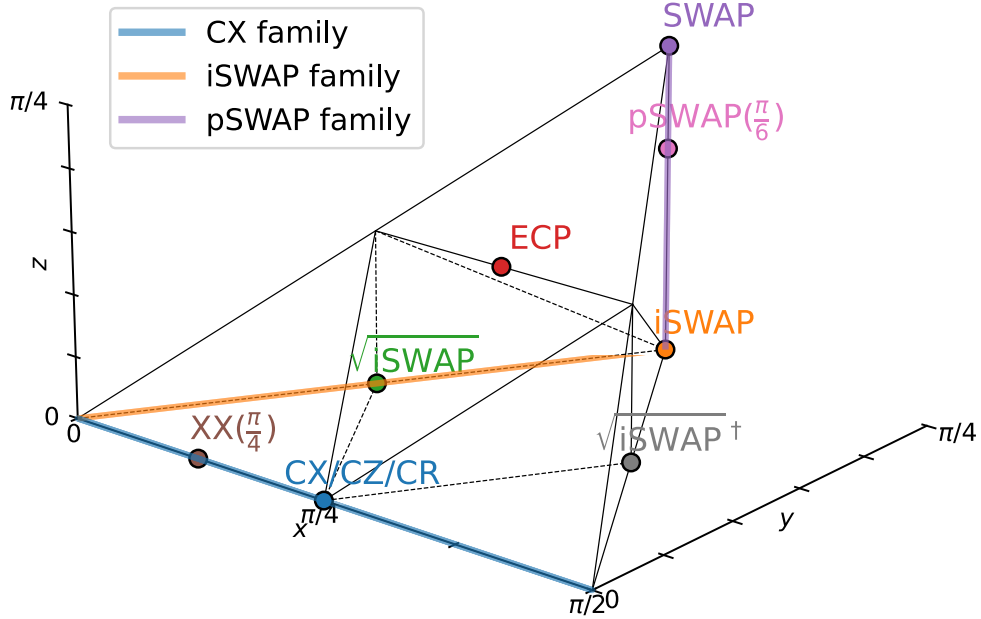


- ReQISC makes the most expressive SU(4) ISA the most performant and practical via microarchitecture-compiler co-design
- **Microarchitecture**: direct realization of any 2Q gate under any coupling Hamiltonian, with optimal gate time and simple pulse controls
- **Compiler**: Optimization passes tailored to SU(4) primitives, through from high-level to circuit-level program representation

Preliminaries: Canonical gate and the Weyl chamber

- The nonlocal content of any 2Q gate $U \in \mathrm{SU}(4)$ can be characterized by 3 real parameters (x, y, z) – Canonical decomposition
- The **Weyl chamber** is a tetrahedron that catalogs all equivalent classes of 2Q gates
- An ISA = a choice of which points in this space the hardware can natively execute
- Think of it as "*opcode space*" for 2Q quantum instructions

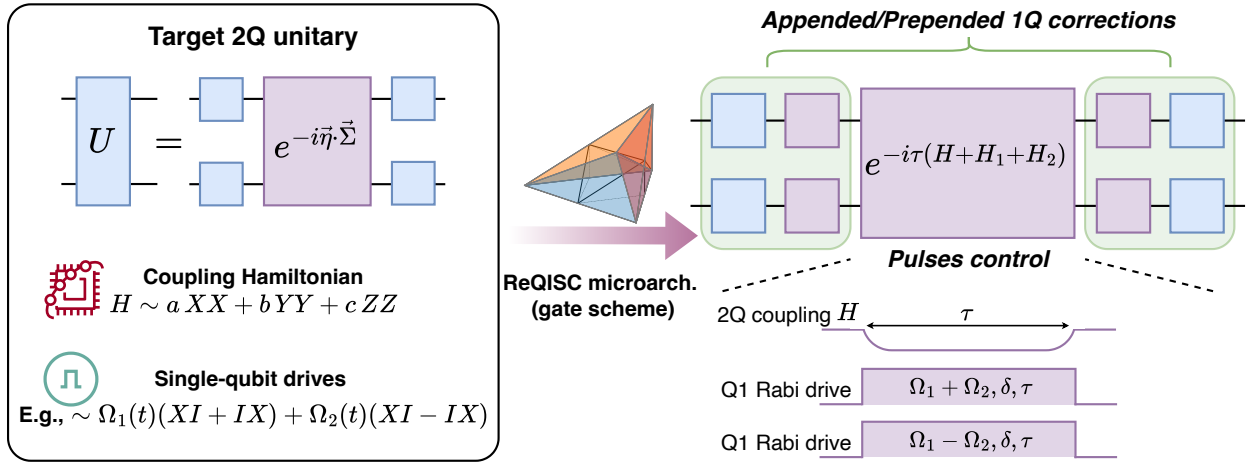


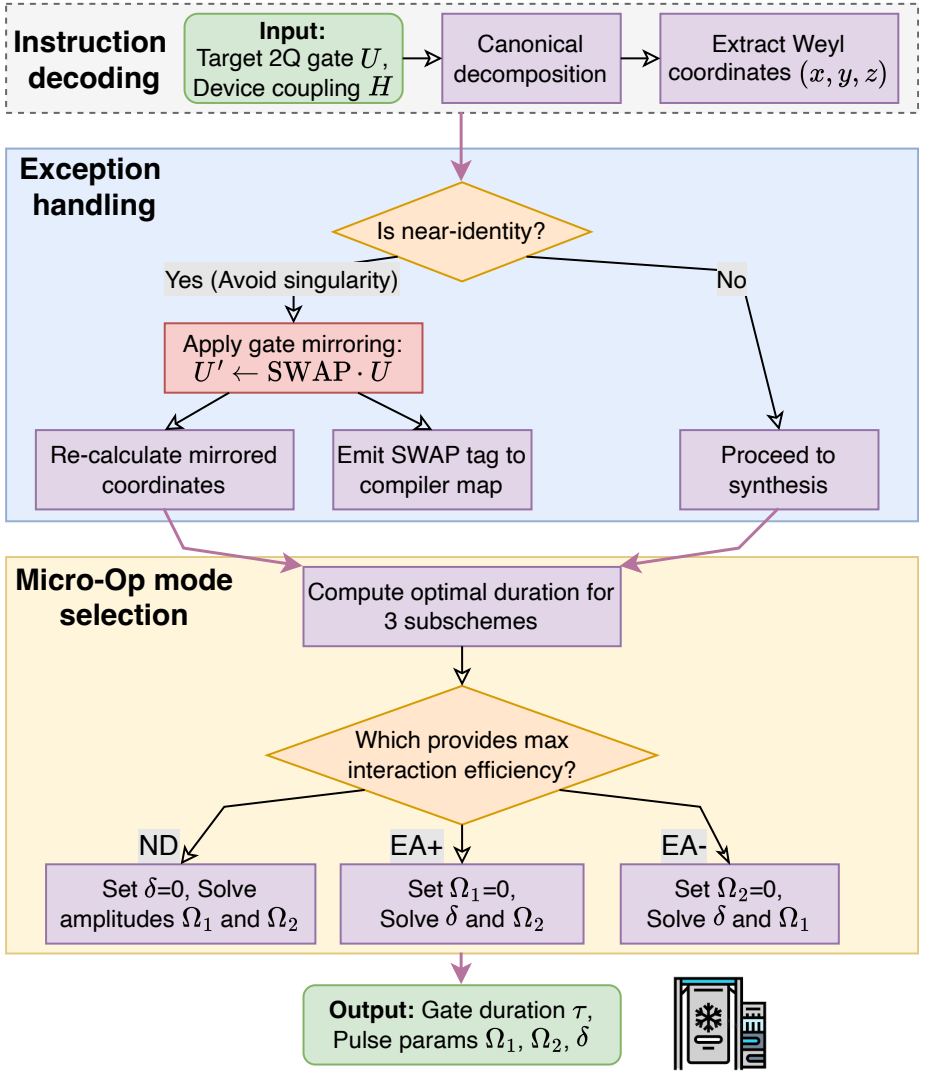


***Weyl chamber**: the tetrahedron confined to*
{π/4 ≥ x ≥ y ≥ z ≥ 0} U {π/4 ≥ (π/2-x) ≥ y ≥ z ≥ 0}

ReQISC μarch: Protocol design

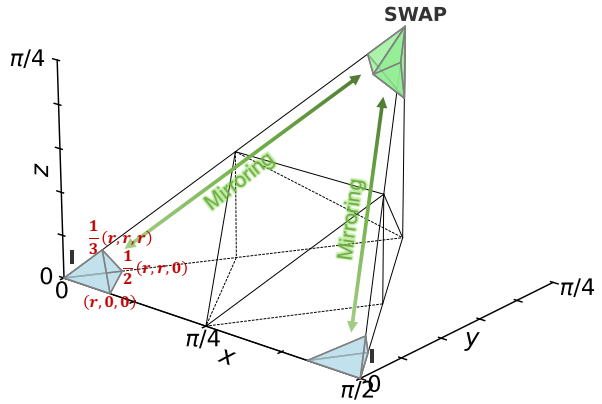
- From a systems perspective
  - Hardware provides fixed "datapath" (coupling Hamiltonian)
  - ReQISC decodes the target instruction into Canonical (Weyl) coordinates, and derives the control logic that configures **local drives** to realize any 2Q instruction (**3 subschemes**)
- Always achieves theoretical time-optimality, meaning the minimal physical gate duration to fight decoherence!



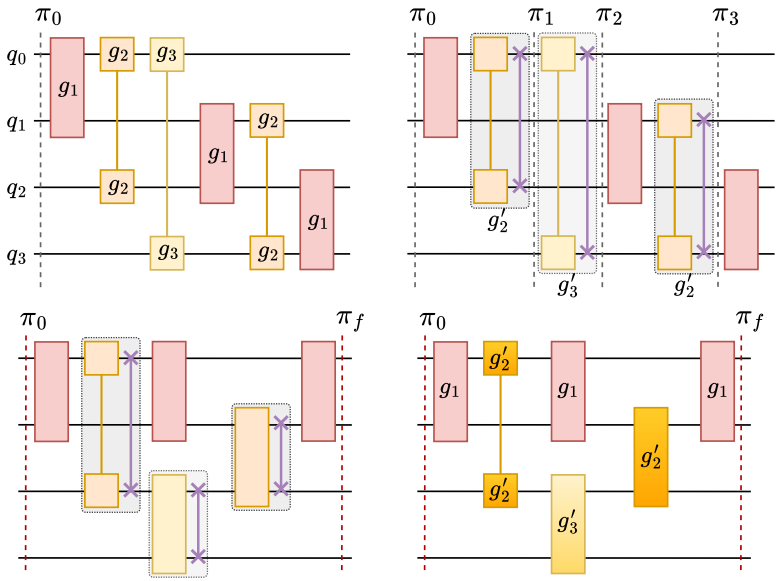


Compile-time control singularity resolution

- The near-identity control singularity
  - Time-optimal near-identity gates result in unbounded pulse amplitude (ASPLOS'24)!
  - Otherwise, trade gate time for bounded amplitude → longer duration
- **Solution** – Gate mirroring
  - Replace near-identity U with its mirror $U' = \mathrm{SWAP} \cdot U$ (far from identity)
  - Track qubit mapping change *at compile time*
  - No extra 2Q gate overhead at the logical level

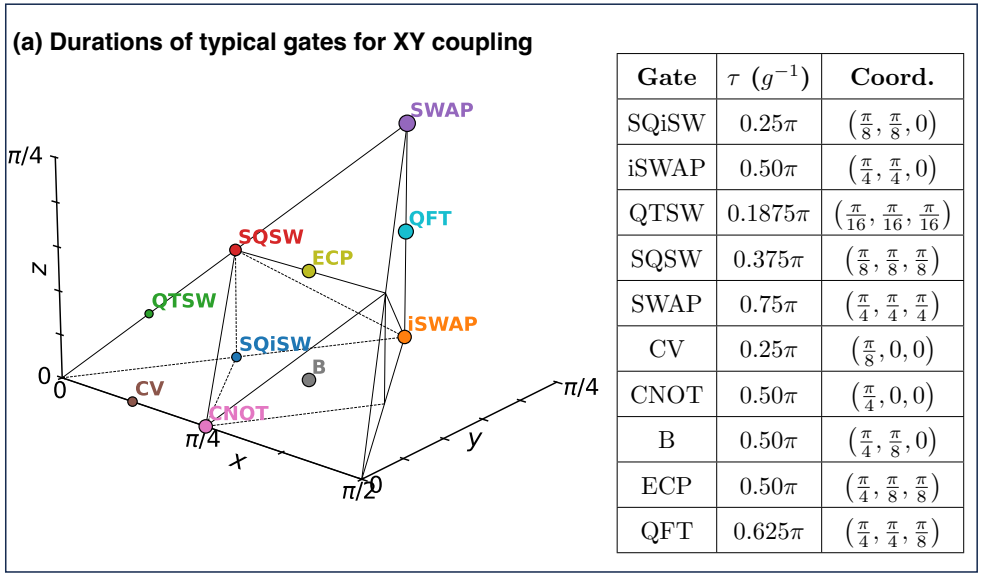


Mirroring for near-identity gates

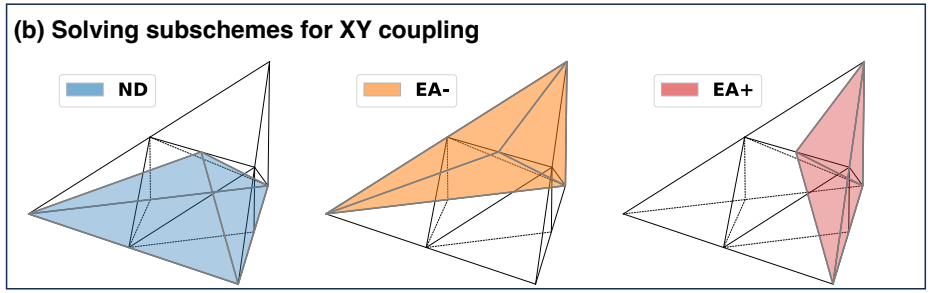


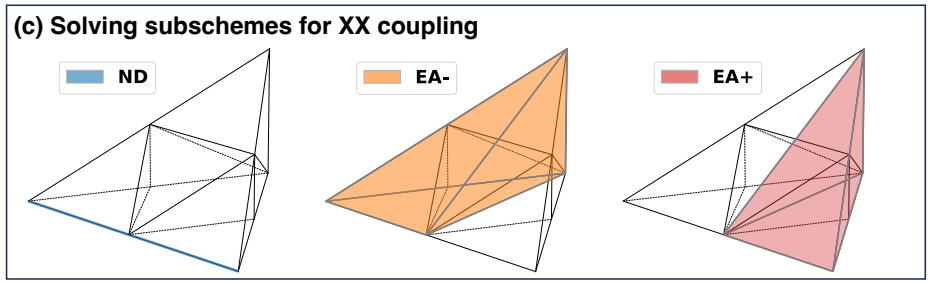
E.g., gate mirroring for `qft_4` circuit

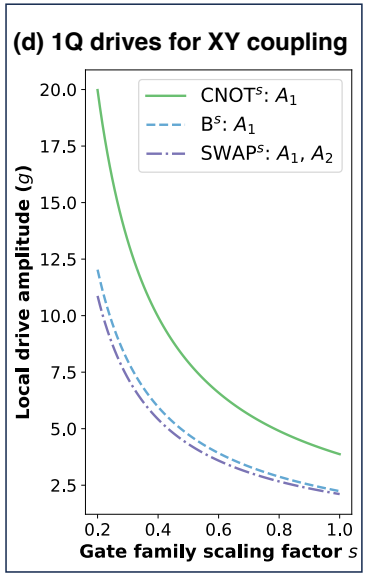
Hardware feasibility and calibration



Gate	$\tau\ (g^{-1})$	Coord.
SQiSW	0.25π	$(\frac{\pi}{8}, \frac{\pi}{8}, 0)$
iSWAP	0.50π	$(\frac{\pi}{4}, \frac{\pi}{4}, 0)$
QTSW	0.1875π	$(\frac{\pi}{16}, \frac{\pi}{16}, \frac{\pi}{16})$
SQSW	0.375π	$(\frac{\pi}{8}, \frac{\pi}{8}, \frac{\pi}{8})$
SWAP	0.75π	$(\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{4})$
CV	0.25π	$(\frac{\pi}{8}, 0, 0)$
CNOT	0.50π	$(\frac{\pi}{4}, 0, 0)$
B	0.50π	$(\frac{\pi}{4}, \frac{\pi}{8}, 0)$
ECP	0.50π	$(\frac{\pi}{4}, \frac{\pi}{8}, \frac{\pi}{8})$
QFT	0.625π	$(\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{8})$







- Works with any coupling Hamiltonian
- Simple pulse control: Only requires simultaneously applying local drives with coupling Hamiltonian (standard AWGs are sufficient)
- No explicit leakage error and stray ZZ crosstalk
- Calibration procedure in arXiv:2312.05652 and arXiv:2502.03612

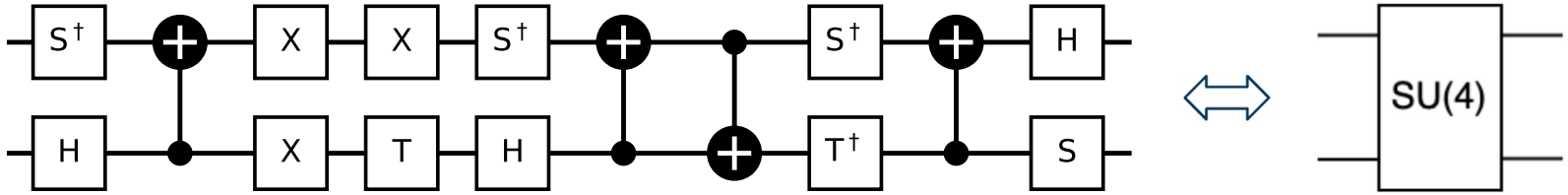
Average Pulse Duration (Haar-random SU(4))

Scheme	Duration
CNOT-trad (3×)	$6.664\,g^{-1}$
ReQISC	$\mathbf{1.341\,g^{-1}}$
4.97× speedup	

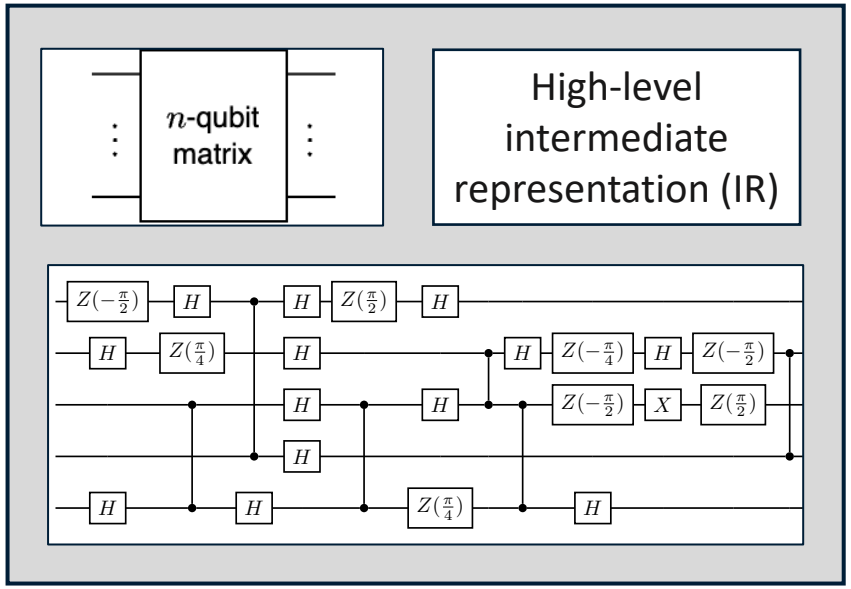
Even single CNOT: 1.41× **faster** than traditional scheme

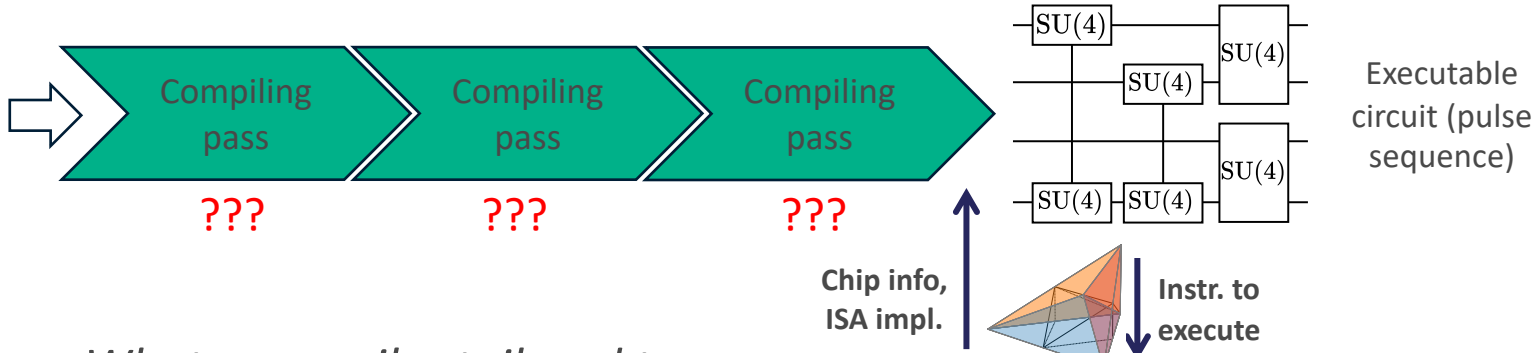
ReQISC compilation framework

Theoretical synthesis advantage in 2Q granularity is explicit:

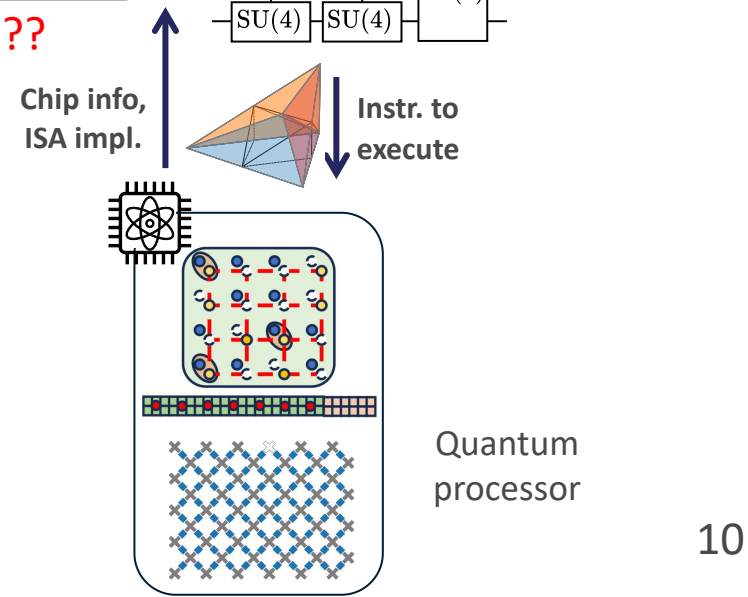


What if for a real-world quantum algorithm?



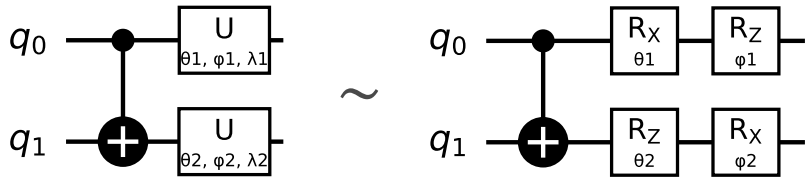


What a compiler tailored to both program features and hardware primitives, across all compilation stages?



Compiler innovation: SU(4) synthesis characteristic in generic unitary synthesis

Synthesis with $\{\mathrm{CNOT}, \mathrm{U3}\}$



Synthesis with $\mathrm{SU}(4)$, i.e., $\{\mathrm{Canonical}, \mathrm{U3}\}$

$$\mathrm{U} = (A_1 \otimes A_2)\mathrm{Can}(x, y, z)(B_1 \otimes B_2)$$

$$\mathrm{Can}(x, y, z) = e^{-i(xXX + yYY + zZZ)}$$

Degrees of freedom of n-qubit unitary matrix $\mathrm{SU}(2^n)$*:* $4^n - 1$

#CNOT Lower bound: $\left\lceil \frac{1}{4}(4^n - 3n - 1) \right\rceil$

55.6% #2Q reduction for generic unitary synthesis

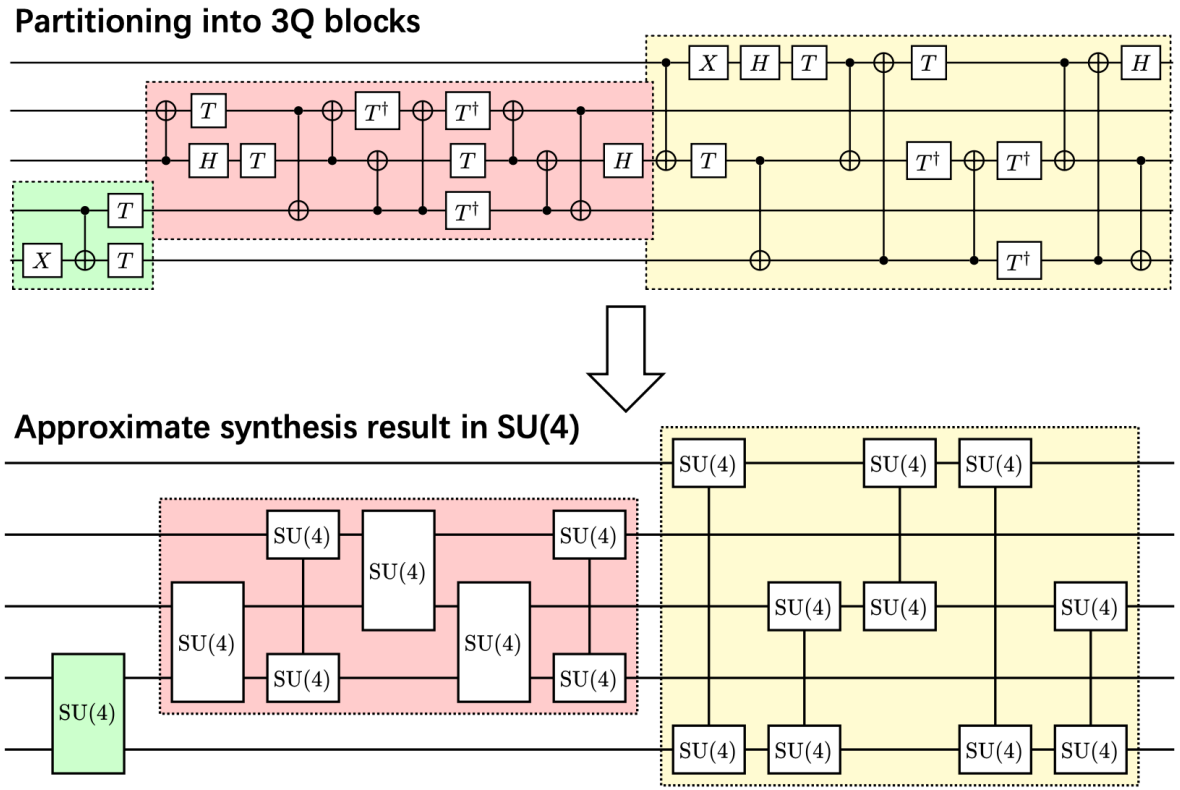
#SU(4) Lower bound: $\left\lceil \frac{1}{9}(4^n - 3n - 1) \right\rceil$

How to achieve lower-bound synthesis? → Approximate synthesis

- **Structural optimization** of gate arrangements & **Numerical optimization** of gate parameters
- Usually **lower-bound (or even less)** #2Q
- Exponential complexity: *hard to scale up more than 4 qubits*

A systematic solution to reducing 2Q gate count

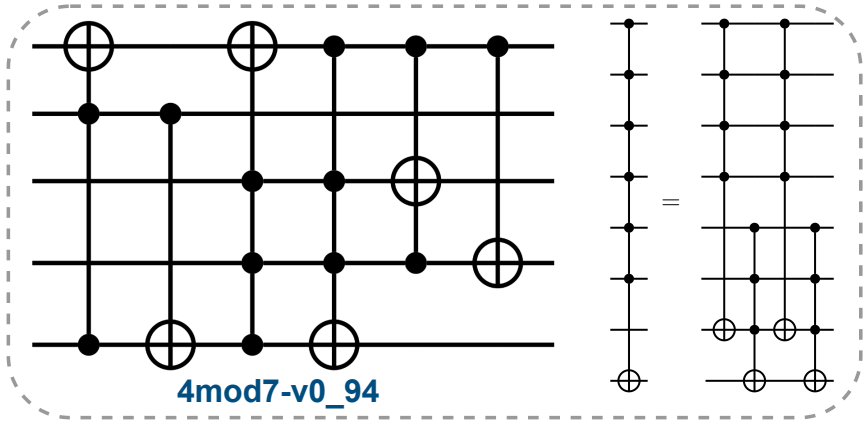
Hierarchical synthesis: *2Q partitioning → 3Q partitioning → approximate synthesis (conditionally)*

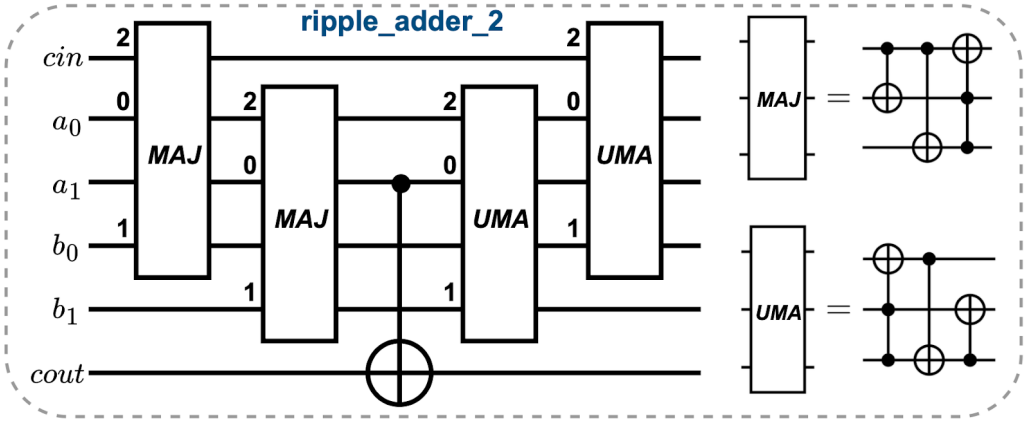


Example for `alu-v2_33` circuit: #2Q reduced from 17 to 11

- Why partitioning?
  - SU(4) synthesis advantage
  - Resource-bound analysis rooted for SU(4)
- Why 2Q partitioning first?
  - Trivially fuse 2Q block into SU(4)
- Why 3Q partitioning?
  - Computational expense of approx. synth.
  - Practical subcircuit patterns
  - Optimal synthesis effect guaranteed by 3Q SU(4)-composed blocks

Compiler innovation: Insights into program-aware, global optimization





- For efficient synthesis with low calibration overhead
  - Repeated X/CX/MCX, Peres, UMA, MAJ patterns → 3Q blocks are refined IR
- Building on fine-grained versatile IR patterns
  - Optimal pre-synthesized IR library → template-based synthesis
  - Again, “approximate synthesis” leads to optimal synthesis templates

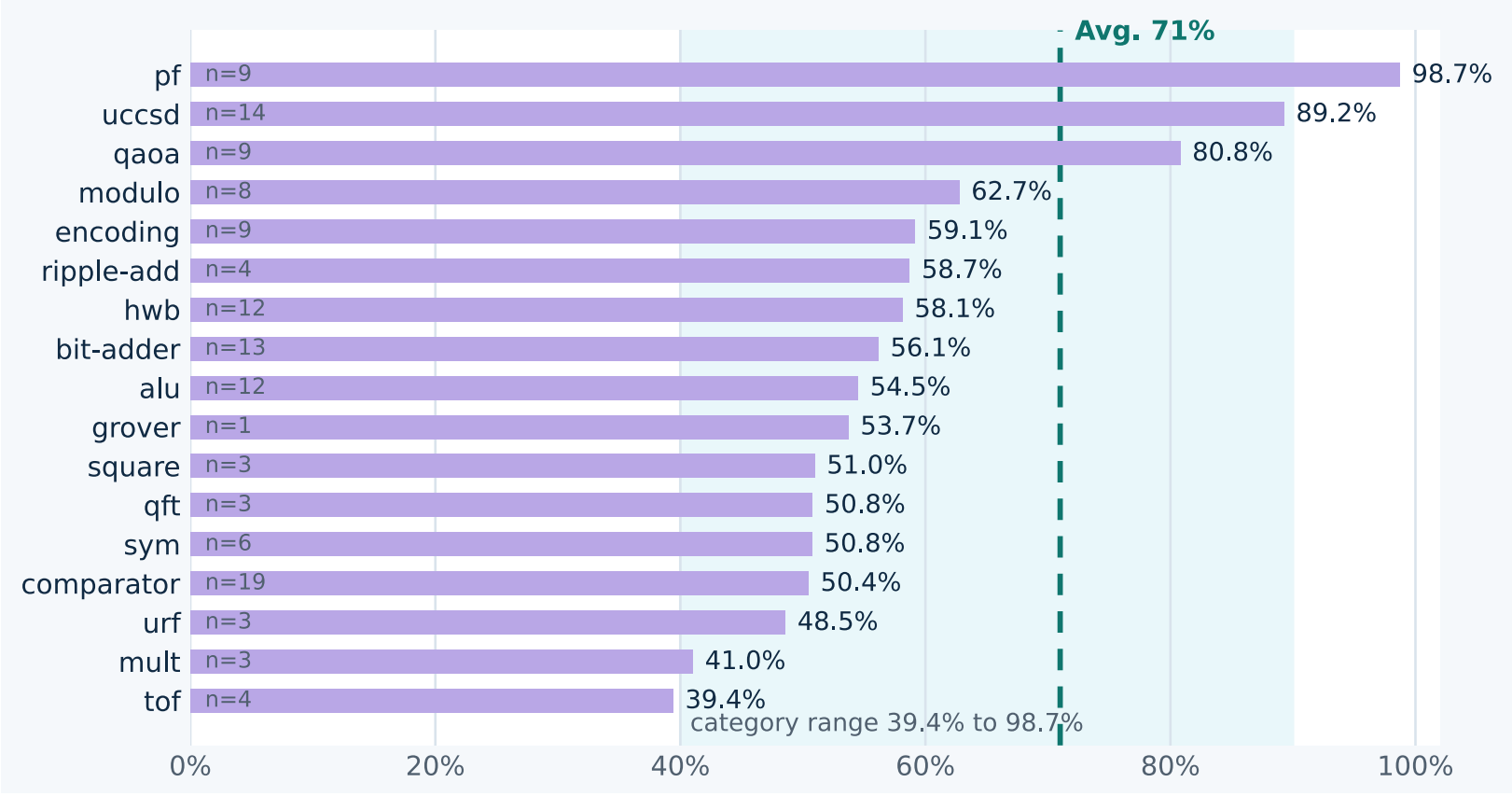
Compiler innovation: Practicality on hardware

- Calibration overhead management
  - 2Q gate calibration is laborious and periodic; The more distinct output SU(4) gates are, the higher calibration cost is
  - Template-based synthesis toward systematic calibration efficiency
  - Optional adopted hierarchical synthesis pass controls the trade-off between calibration efficiency and #2Q reduction
- Routing overhead mitigation
  - An SU(4)-aware mirroring-SABRE mapping method to mitigate 2Q gate count overhead induced by SWAP insertion (SWAP mirroring mechanism)

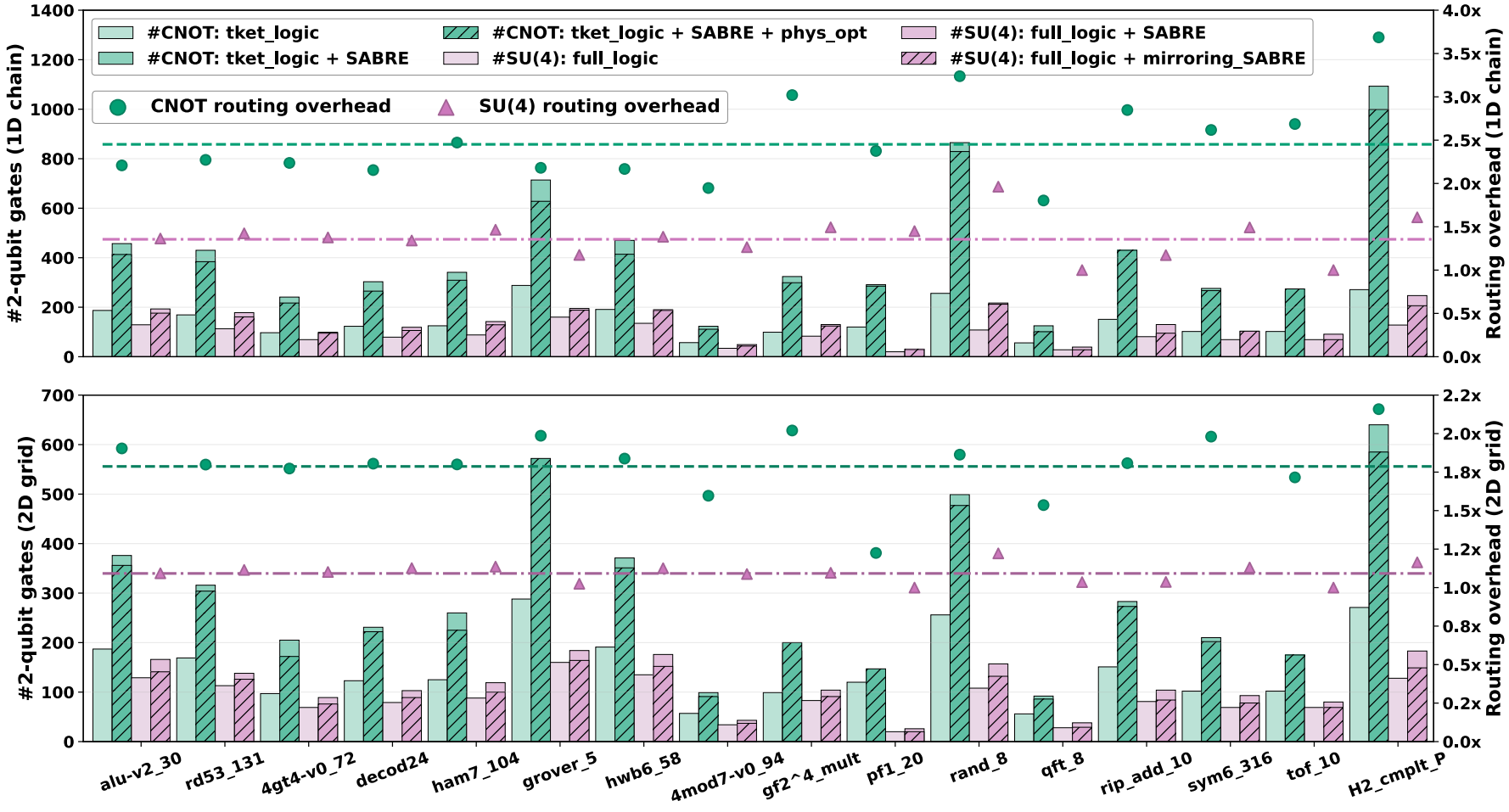


Evaluation: Logical-level compilation

- Tested over 132 real-world benchmarks
- 40–90% pulse-level circuit duration reduction
- Avg 71% duration reduction; 52% 2Q gate count reduction (ReQISC-Full)
- Benefits span all program types: *arithmetic, QFT, Grover, QAOA, chemistry, …*
- Distinct SU(4) count is managed (demonstrated in calibration efficiency evaluation as follows …)



Evaluation: Topology-aware compilation

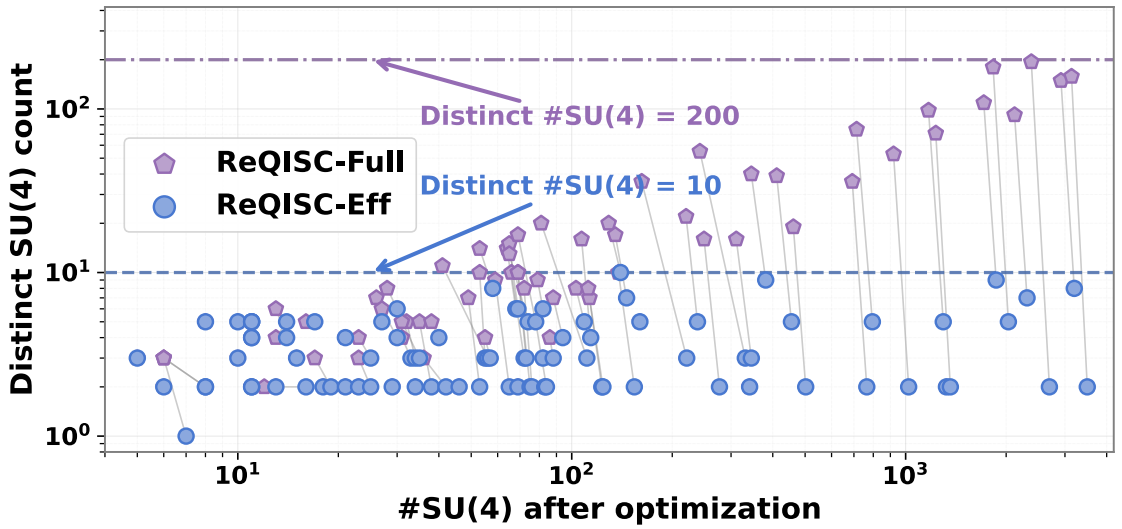


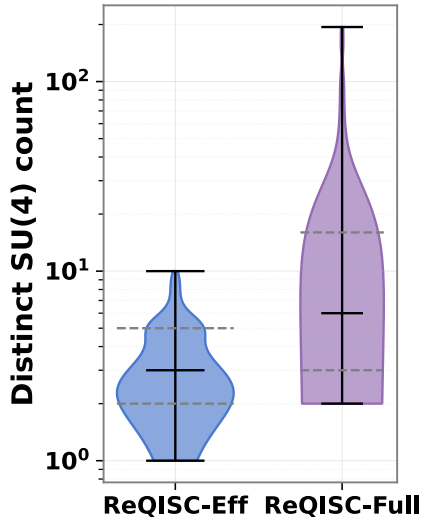
1D chain: SU(4) overhead 1.36x vs. CNOT 2.45x

2D square: SU(4) overhead 1.09x vs. CNOT 1.79x

Evaluation: Calibration efficiency & Trade-off

- ReQISC-Eff: disable hierarchical synthesis
  - < 10 distinct SU(4) gates
  - Extremely low calibration overhead
- ReQISC-Full: adds hierarchical synthesis for aggressive #2Q optimization
  - < 200 distinct SU(4) gates
  - But >75% of programs need < 20

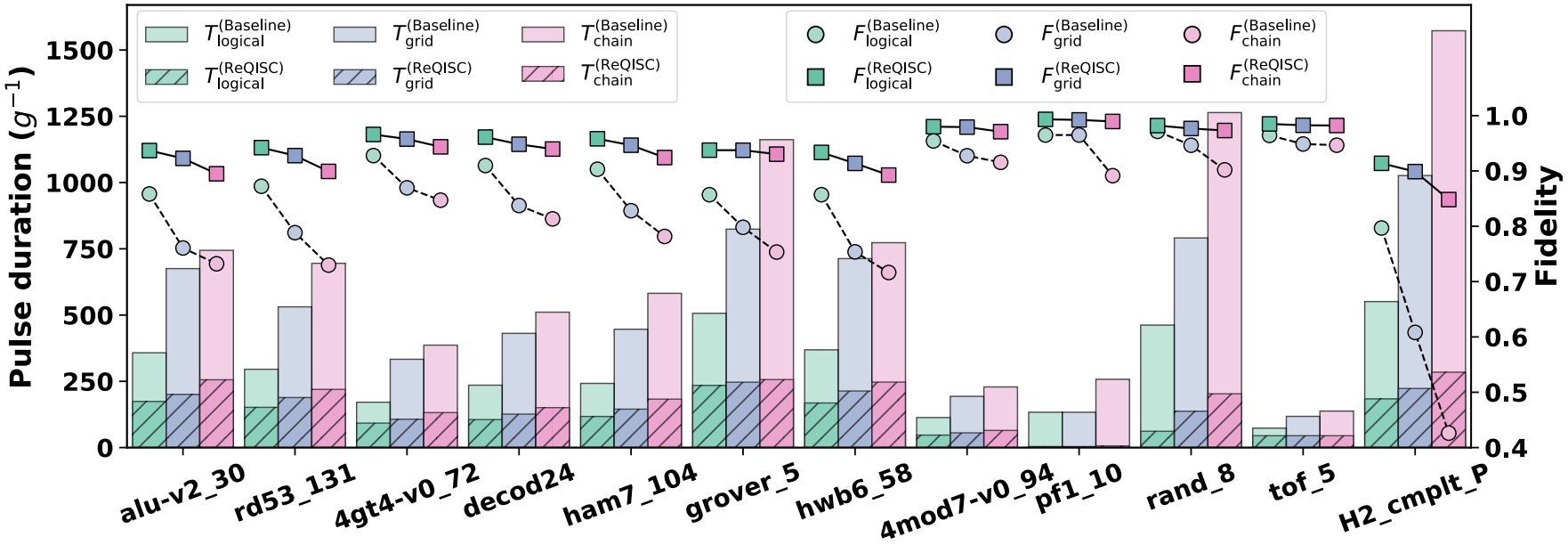




Evaluation: Fidelity experiment via noisy simulation

- Noisy simulation vs. CNOT counterparts; 2Q depolarizing noise ∝ gate duration
- Advantages *amplify* with topology constraints vs. CNOT counterparts

	Logical-level (no routing)	2D grid mapping	1D chain mapping
Execution speedup	3.06x	4.30x	4.55x
Error reduction	2.36x	3.18x	3.34x



Conclusion

- **ReQISC microarchitecture** (generalized AshN scheme)
  - Any quantum hardware that can natively support at least one particular 2Q gate and all 1Q gates (necessary for universality) can also natively support all possible 2Q gates, up to local equivalence, in minimal gate time!
  - 4.97x faster than CNOT/CZ for arbitrary SU(4) synthesis; The pulse control singularity (near-identity issue) is resolved at compile time
- **ReQISC compiler**: three co-designed passes; high-level IR to circuit-level
  - 40–90% pulse-level circuit duration reduction; Up to 4.55x speedup and 3.34x error suppression for real-world applications under noisy simulation
- **Practical**: Manageable calibration overhead for continuous ISAs

Thanks for attention!



"Ash nazg durbatulûk, ash nazg gimbatul, ash nazg thrakatulûk, agh burzum-ishi krimpatul".

Backup slides

- Q&A:
  - Implementation and application
  - Broader applicability
- ReQISC microarchitecture
  - Derivation and proof
  - Numerical solving
- Technical mechanism
  - Hierarchical synthesis
  - Template-based synthesis
  - SU(4)-aware mirroring-SABRE mapping
- Evaluation results: Reliability and Scalability

Q&A: Implementation and application

- How to validate the scheme on hardware?
  - The original AshN scheme (ASPLOS'24) was realized on XY-coupling flux-tunable transmons with experimentalist collaborators (Nat. Phys. 25)
  - The robust code implementation of ReQISC microarchitecture (genAshN scheme) is accessible in the artifact within the paper
- What is the impact on industry?
  - At least, CNOT is $\sqrt{2}\times$ faster than traditional scheme on Google's QPU, without explicit leakage error …
  - Some companies are adopting our gate scheme …



Q&A: Broader applicability

- How to adapt to fixed-frequency qubits (e.g., IBM's)?
  - Option 1: Algorithm 1 shows the sufficiency of using simplest pulses controls to engineer all SU(4). For fixed-frequency scenario, the solving procedure might be modified to be applicable.
  - Option 2: Non-resonant frequency components can be incorporated into H; calculation proceed as resonant; corresponding single-qubit to add.
- Any other gates used in QEC apart from CX/CZ? Yes!
  - Google's theoretical (2302.02192) and experimental (2412.14360) work showcase utilizing iSWAP gates fundamentally benefits QEC kernels.
  - Routing with advanced ISAs helps protect logical qubits (2412.21000)



ReQISC μarch: Derivation & Proof

Firstly consider: What is the canonical of the target unitary gate generated from a given common Hamiltonian?

Common Hamiltonian
(coupling, 1Q drive, no detuning)

$$H = H(a,b,c) + \Omega_1(XI + IX) + \Omega_2(XI - IX)$$
$$= a\,XX + b\,YY + c\,ZZ + \Omega_1(XI + IX) + \Omega_2(XI - IX)$$

Hint: Simplify them in Z basis

Conjugate the X and Z basis
by Hadamard transformation

$$H \leftarrow (U_{\text{hadamard}} \otimes U_{\text{hadamard}})\, H\, (U_{\text{hadamard}} \otimes U_{\text{hadamard}}) = \begin{pmatrix} a + 2\Omega_1 & & & c - b \\ & -a + 2\Omega_2 & b + c & \\ & b + c & -a - 2\Omega_2 & \\ c - b & & & a - 2\Omega_1 \end{pmatrix}$$

Engineered unitary will be symmetric
$U = e^{-iH\tau}$

$$U = (A_1 \otimes A_2) e^{i(x\,XX + y\,XX + z\,ZZ)} (A_1 \otimes A_2)^T, \quad \text{since} \quad U^T = U$$
$$U \sim \exp^{i(x\,XX + y\,XX + z\,ZZ)}$$

ReQISC μarch: Derivation & Proof

- Transform Weyl coordinate to spectrum (diagonalize coupling Hamiltonian)!

$$U(Y\otimes Y) = (A_1\otimes A_2)e^{i(x\,XX+y\,XX+z\,ZZ)}(Y\otimes Y)(A_1\otimes A_2)^\dagger \sim e^{i(x\,XX+y\,XX+z\,ZZ)}(Y\otimes Y)$$

$$\exp(-iH\tau)(Y\otimes Y) = \begin{pmatrix} \frac{ie^{-ia\tau}(c-b)\sin S_1\tau}{S_1} & & & \frac{e^{-ia\tau}(-S_1\cos S_1\tau+i\Omega_1\sin S_1\tau)}{S_1} \\ & \frac{-ie^{ia\tau}(b+c)\sin S_2\tau}{S_2} & \frac{e^{ia\tau}(S_2\cos S_2\tau-i\Omega_2\sin S_2\tau)}{S_2} & \\ & \frac{e^{ia\tau}(S_2\cos(S_2\tau)+i\Omega_2\sin S_2\tau)}{S_2} & \frac{-ie^{ia\tau}(b+c)\sin S_2\tau}{S_2} & \\ \frac{e^{-ia\tau}(-S_1\cos S_1\tau-i\Omega_1\sin S_1\tau)}{S_1} & & & \frac{ie^{-ia\tau}(c-b)\sin S_1\tau}{S_1} \end{pmatrix}$$

- Thus, the spectrum of $e^{-iH\tau}YY$ determines the Weyl coordinate (x, y, z)

$$x = at,\ \sin(y+z) = \frac{(b+c)\sin S_2 t}{S_2},\ \sin(y-z) = \frac{(b-c)\sin S_1 t}{S_1}$$
$$S_1 := \sqrt{4\Omega_1^2+(b-c)^2}, S_2 := \sqrt{4\Omega_2^2+(b+c)^2}$$

arXiv:2312.05652
arXiv:1904.10541
arXiv:2511.06746

- More details in paper …

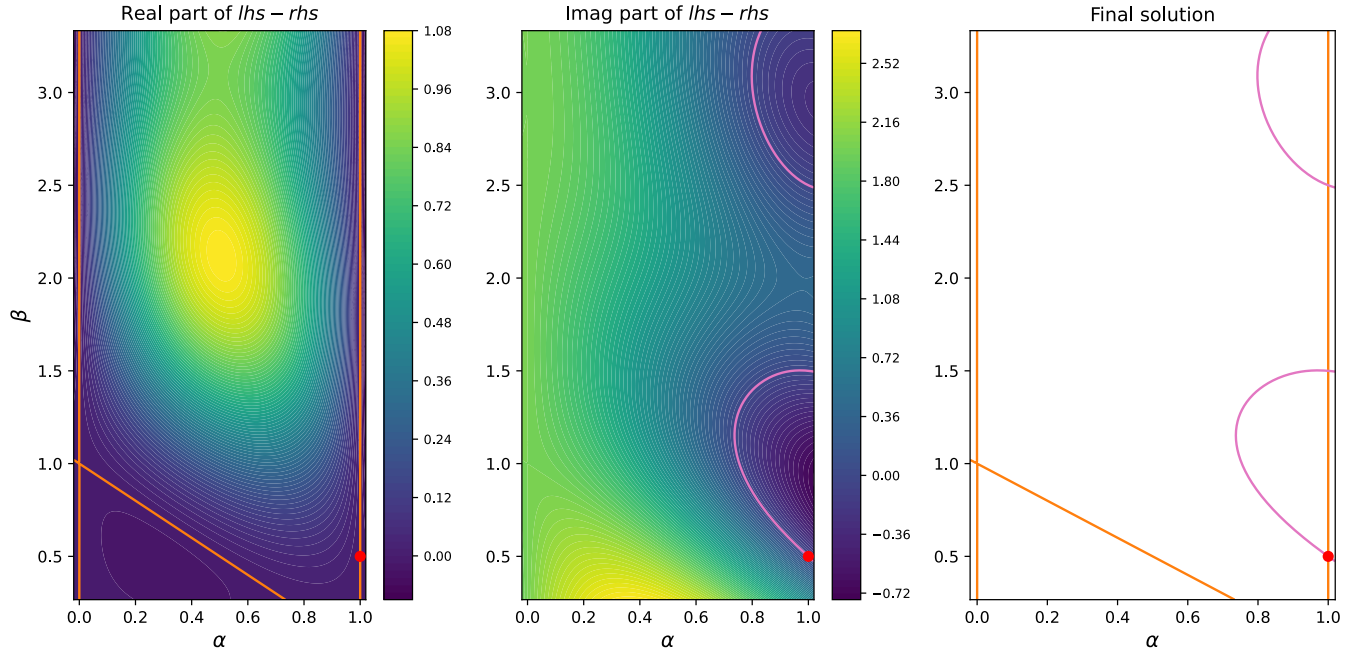
ReQISC μarch: Numerical solving

- **ND subscheme**: Determining $S_{1,2}$ is straightforward (primary roots)
- **EA+/- subscheme**: Sort of complicated when determining the primary roots of (α, β) → lowest pulse amplitude
- **Solving strategy**: Coarse grid search + local refinement
  - SciPy's `minimize` (SLSP) + `fsolve`
  - Field tests always show primary-root solutions in machine precision

ND:
$$\Omega_1 \leftarrow \frac{1}{2}\sqrt{S_1^2 - (b-c)^2};\ \Omega_2 \leftarrow \frac{1}{2}\sqrt{S_2^2 - (b+c)^2}\ \delta \leftarrow 0;$$

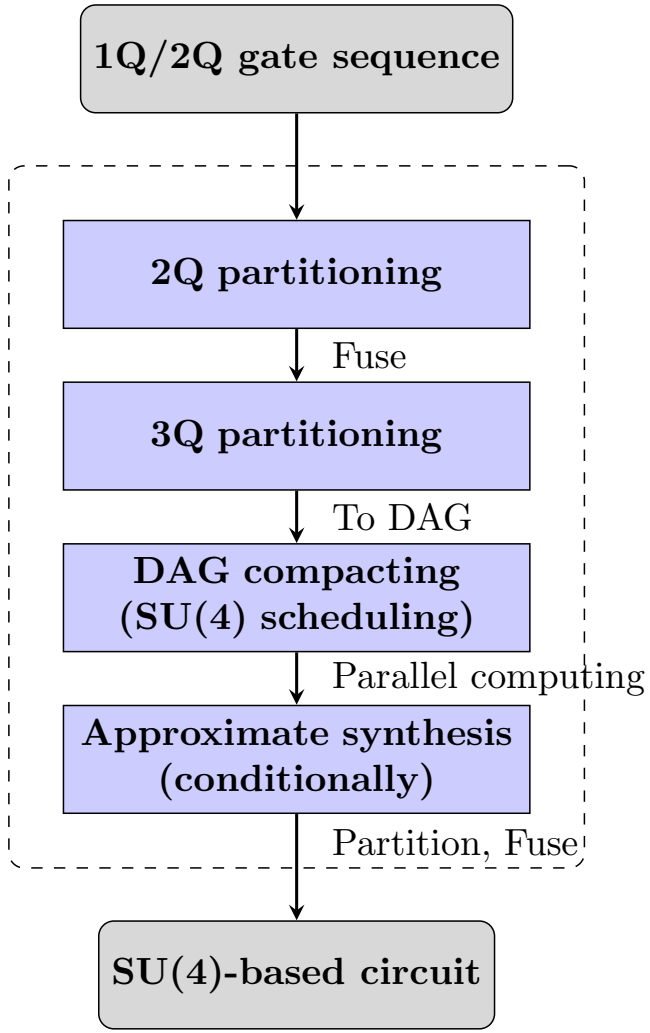
EA+:
$$\Omega_1 \leftarrow 0;\ \Omega_2 \leftarrow (a+c)\sqrt{(1-\alpha)\beta(1-\eta+\alpha+\beta)};$$
$$\delta \leftarrow -(a+c)\sqrt{\alpha(1+\beta)(\alpha+\beta-\eta)};$$

EA-:
$$\Omega_1 \leftarrow (a-c)\sqrt{(1-\alpha)\beta(1-\eta+\alpha+\beta)};\ \Omega_2 \leftarrow 0;$$
$$\delta = (a-c)\sqrt{\alpha(1+\beta)(\alpha+\beta-\eta)};$$



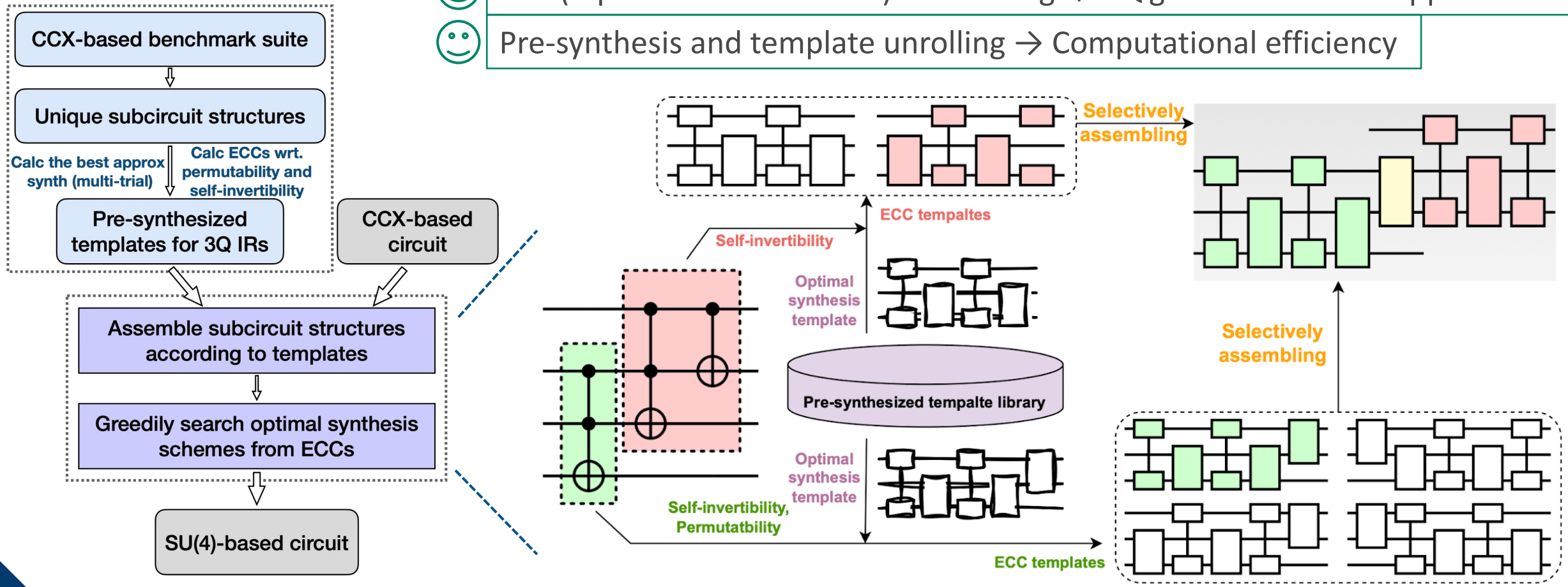
Structure of the hierarchical synthesis

- Hierarchical synthesis building on circuit partitioning and approximate synthesis to approach #SU(4) within subcircuit synthesis to the theoretical lower bound or less
- Define compactness metric to guide co-optimization
  - Partitioning could also compactness-aware
  - DAG compacting based on approximate commutation rules
- Radical local circuit optimization but lead to higher calibration overhead (uncontrollable SU(4) parameters via numerical optimization)



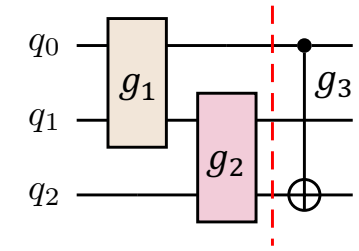
Template-based synthesis mechanism

- Finite 3Q IRs → Low calibration cost
- ECC (equivalent circuit class) assembling → 2Q gate cancellation opportunities
- Pre-synthesis and template unrolling → Computational efficiency

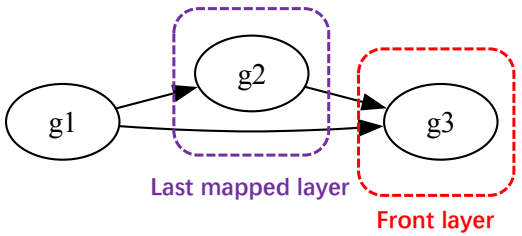


Mirroring-SABRE mapping

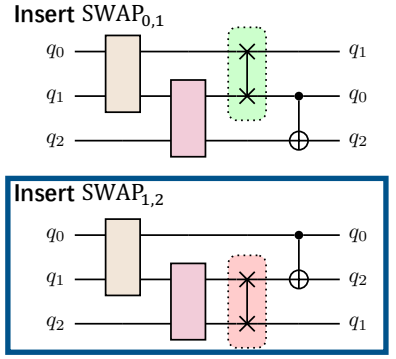
- With the backbone of SAMRE mapping (ASPLOS'19)
  - Data structures: DAG, front layer, extended set, …
  - Heuristic cost, lookahead search, decay, …
- Novel mechanism in our SU(4)-aware mapping/routing
  - Starting cost H_0 before SWAP search; Last mapped layer to identify SWAP merging
  - The lookahead mechanism creates many aggregation opportunities to fuse SWAPs



Search SWAP at this time slice

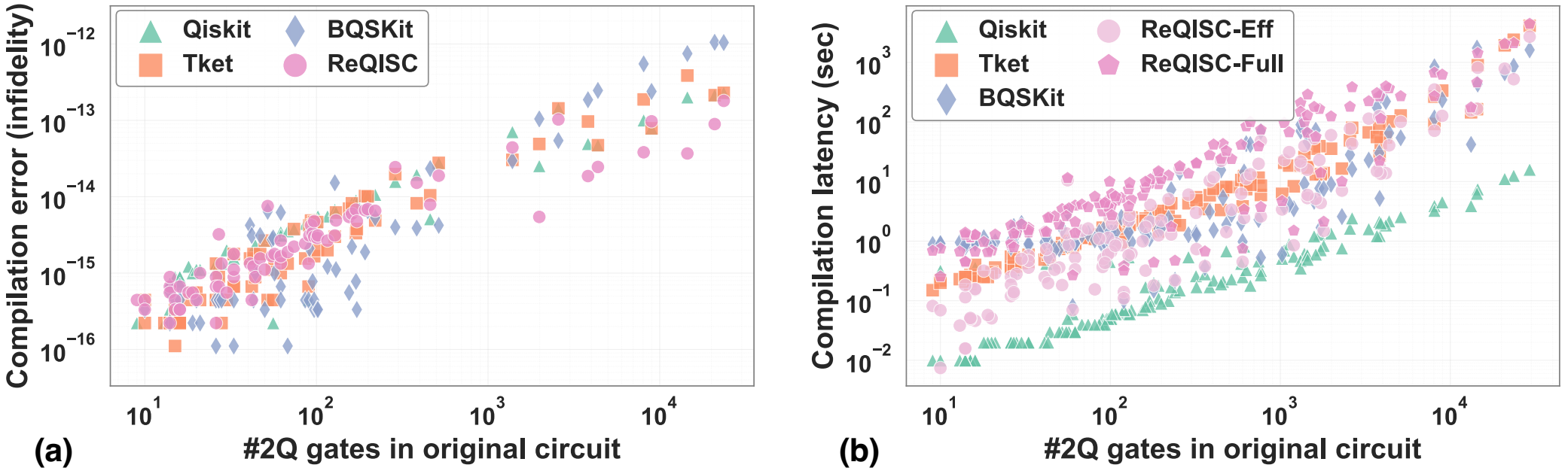


DAG representation



This SWAP is prioritized for less #SU(4) increase

Evaluation: Reliability & Scalability



- All compilers exhibit a similar level of compilation errors; Local computation (analytical or numerical synthesis) always achieves machine precision
- Compilation latency scales polynomially; ReQISC-Eff is even faster than Tket
- ReQISC-Full's latency competitive with BQSKit despite Python implementation