Quantum quantum circuit compilation

Compiling quantum circuits with quantum computers

https://arxiv.org/abs/2408.00077

Davide Rattacaso*, Daniel Jaschke, Marco Ballarin, Ilaria Siloi, and Simone Montangero.

University of Padova, INFN Padova

*davide.rattacaso@unipd.it

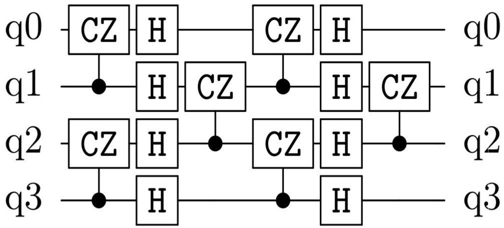
QUANTUM Information and Matter

EuRyQa

Part one: background

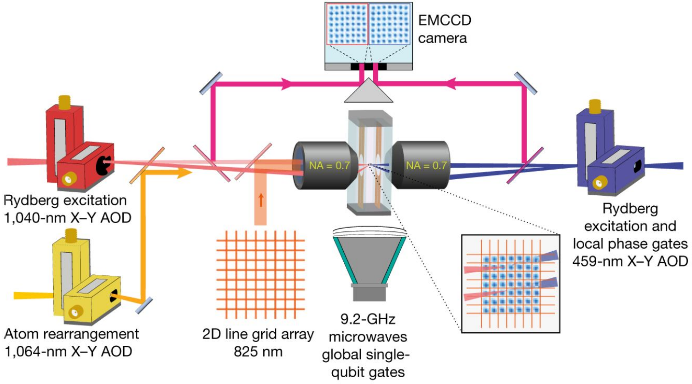
What do we need to execute a quantum circuit on real-world hardware?

The abstract circuit that we want to implement



- Arbitrary gates
- All-to-all
- No errors

The actual machine that that is supposed to run it



Graham, T.M., Song, Y., Scott, J. *et al. Nature* **604**, 457–462 (2022).

- Finite set of native gates
- Limited connectivity
- Gates affected by errors (limited control precision, decoherence, crosstalk...)

Optimized quantum-circuit compilation

Given the input circuit, find an equivalent circuit (same unitary up to global phase) that can be implemented on the target machine with minimum expected infidelity.

Optimal compilation can make the difference in NISQ era and beyond :)

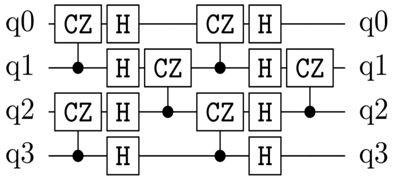
The number of equivalent circuits that implement the same algorithm increases exponentially in the circuit volume.

Optimal compilation is exponentially hard :(

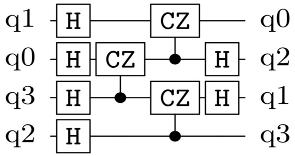
Idea: let us compile quantum circuits with quantum computers!

Quantum optimization presents the potential for speedups compared to classical optimization methods.

Given that classical computers are used to compile classical algorithms, can we use quantum computers to compile quantum algorithms?



High infidelity input circuit.



Low infidelity output circuit

Part two: method

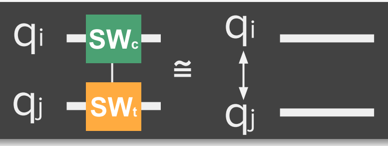
Overview of classical quantum-circuit compilation

Replace subcircuits that implement the same unitary op. up to a global phase factor.

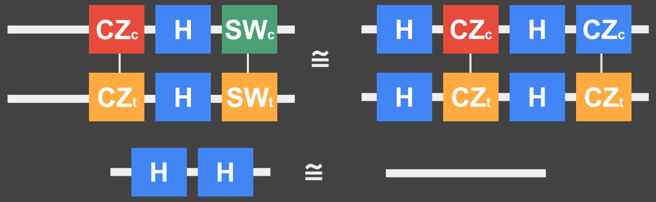
Some equivalence rules as example.

INPUT CIRCUIT

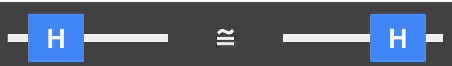
MAPPING: Assign qubits logical addresses to physical addresses



SYNTHESIS: Transform input circuit gates to native gates, simplify identities.

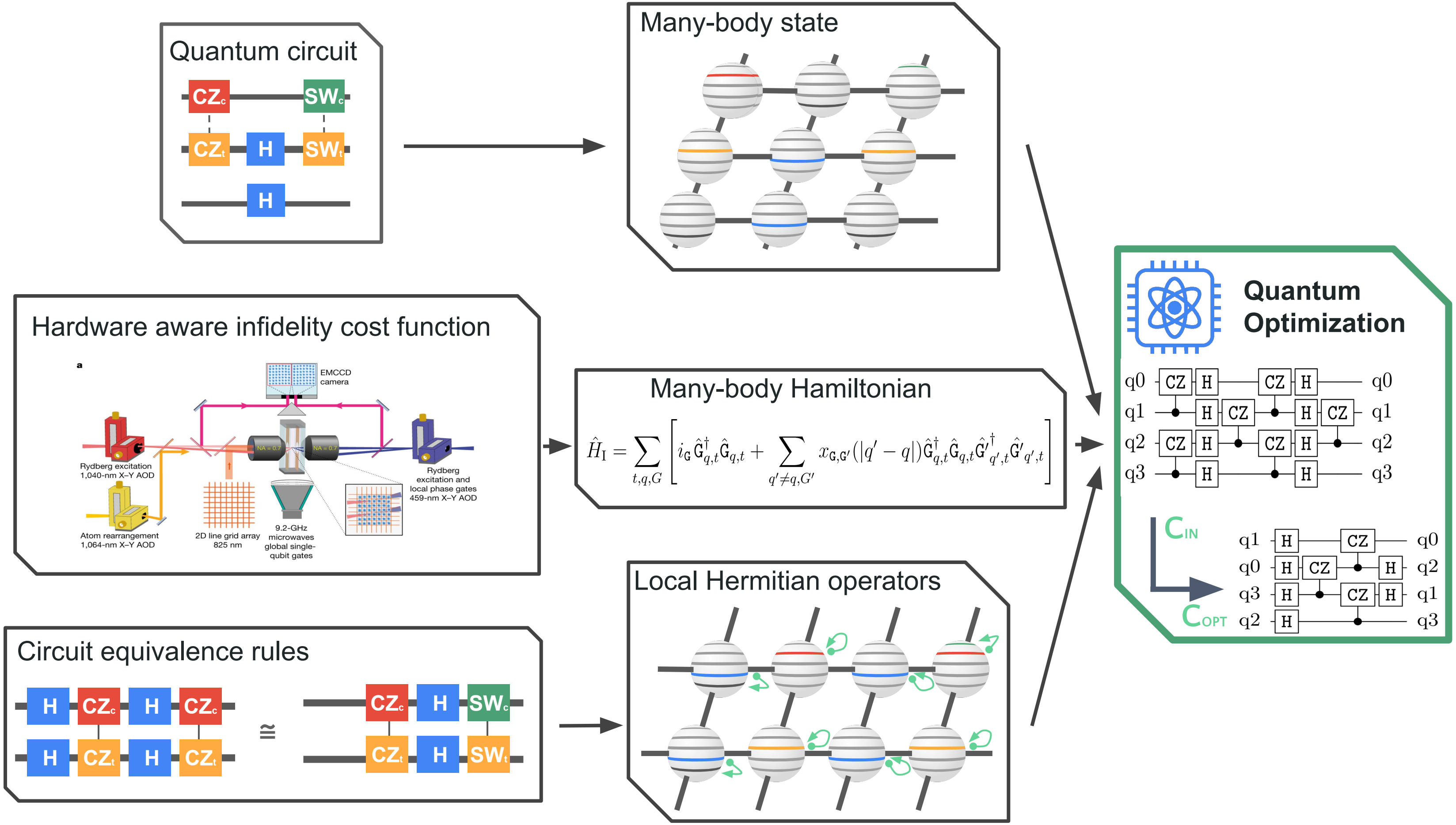


SCHEDULING: Schedule the execution time of gates.

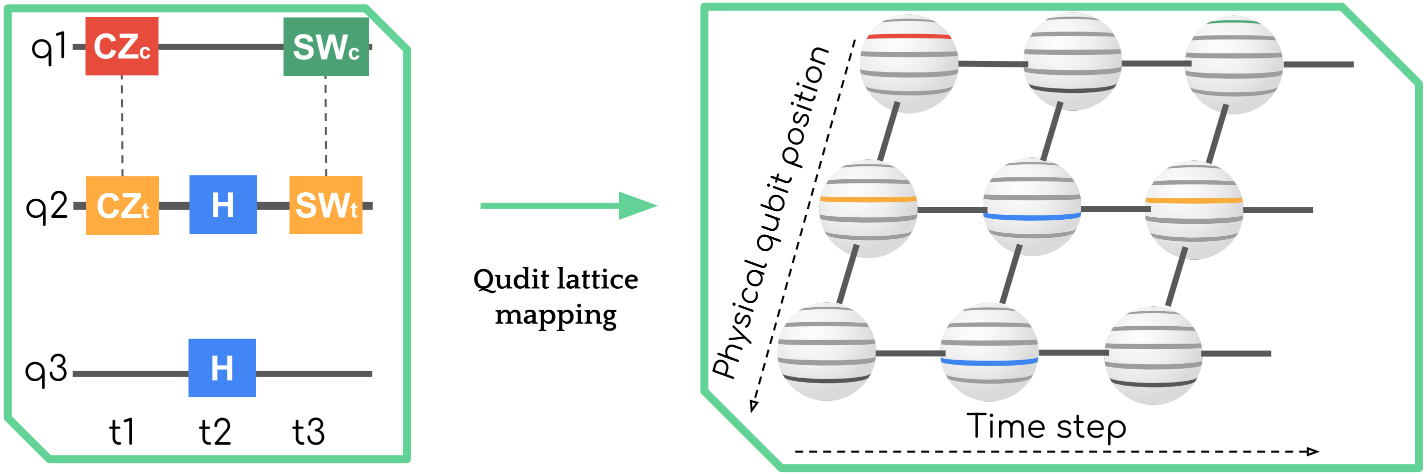


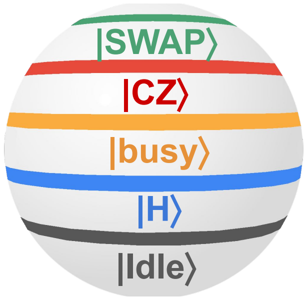
OUTPUT CIRCUIT

General paradigm for quantum quantum-circuit compilation



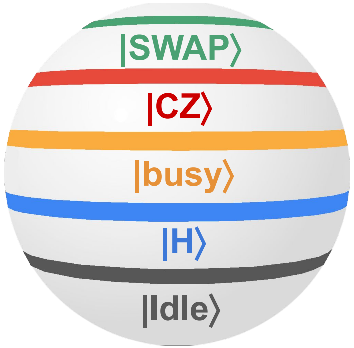
First step: quantum circuits as computational basis states.





- We aim to map circuits to the computational basis states, which will enable us to explore the quantum superposition of circuits throughout the computations.
- For a D-dimensional QPU, we define a D+1 qudits lattice.
- The energy levels of qudits correspond to the different states that a qubit can assume.
- The state of a qudit at position (**q**, t) encodes the gate applied to qubit at position **q** of the QPU and at time t.

Second steps: gate creation and annihilation



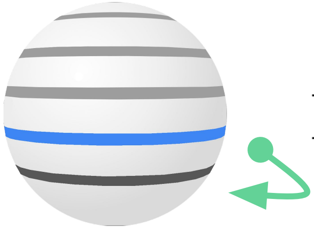
$$\widehat{\mathrm{SWAP}}^{\dagger} := |\mathrm{SWAP}\rangle\langle\mathrm{Idle}|$$

$$\widehat{\mathrm{CZ}}^{\dagger} := |\mathrm{CZ}\rangle\langle\mathrm{Idle}|$$

$$\widehat{\mathrm{busy}}^{\dagger} := |\mathrm{busy}\rangle\langle\mathrm{Idle}|$$

$$\hat{\mathrm{H}}^{\dagger} := |\mathrm{H}\rangle\langle\mathrm{Idle}|$$

$$\widehat{\mathrm{Idle}}^{\dagger} := |\mathrm{Idle}\rangle\langle\mathrm{Idle}|$$

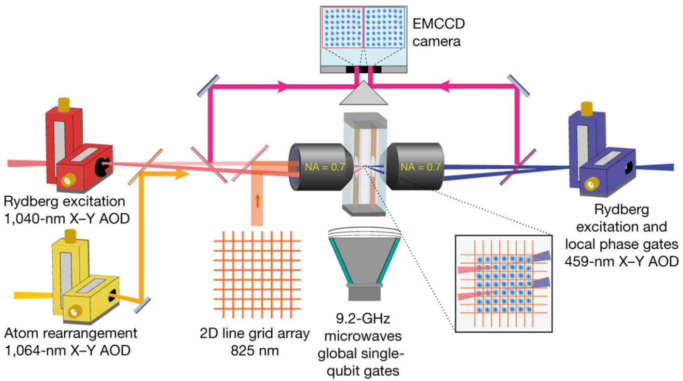


$$\hat{\mathrm{H}}$$

$$\widehat{\mathrm{SWAP}}^{\dagger}\widehat{\mathrm{CZ}}$$

$$\widehat{\mathrm{busy}}^{\dagger}\widehat{\mathrm{busy}}$$

Third step: hardware aware infidelity hamiltonian

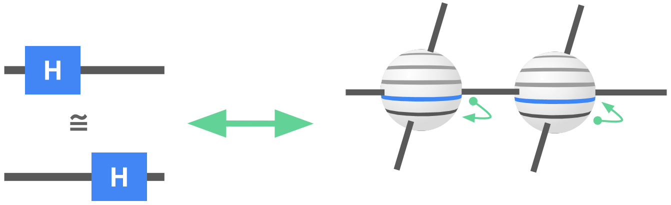


$$I(\mathtt{C}) = \sum_{(\mathtt{G},t,q)\in\mathtt{C}} \left(i_{\mathtt{G}} + \sum_{q' | \mathtt{G}(t,q')=\mathtt{G}} x_{\mathtt{G}}(\|q - q'\|) \right) + \sum_{Idle\ (t,q)} i_{\mathtt{Idle}}$$

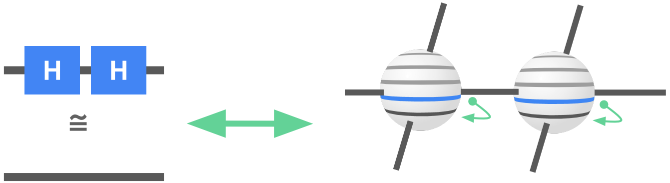
$$\hat{H}_{\mathrm{I}} = \sum_{t,q,G} \left[i_{\mathtt{G}} \hat{\mathtt{G}}^{\dagger}_{q,t} \hat{\mathtt{G}}_{q,t} + \sum_{q' \neq q, G'} x_{\mathtt{G},\mathtt{G}'}(|q' - q|) \hat{\mathtt{G}}^{\dagger}_{q,t} \hat{\mathtt{G}}_{q,t} \hat{\mathtt{G}'}^{\dagger}_{q',t} \hat{\mathtt{G}'}_{q',t} \right] - N_Q i_{\mathtt{Idle}} \sum_{t} \bigotimes_{q} \widehat{\mathtt{Idle}}_{t,q}$$

$$\langle C | \hat{H}_{\mathrm{I}} | C \rangle = I(C)$$

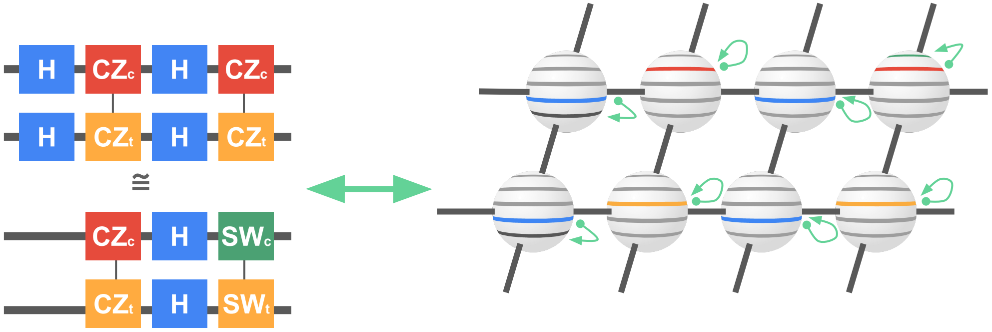
Fourth step: defining equivalence operators



$$\hat{\epsilon_k} = \hat{\mathtt{H}}^\dagger_{q,t}\hat{\mathtt{H}}_{q,t+1} + \text{h.c.}$$



$$\hat{\epsilon_k} = \hat{\mathtt{H}}^\dagger_{q,t}\hat{\mathtt{H}}^\dagger_{q,t+1} + \text{h.c.}$$



$$\hat{\epsilon_k} = \hat{\mathtt{H}}_{q,t}\hat{\mathtt{H}}_{q+1,t}\hat{\mathtt{CZ}}^\dagger_{q,t+1}\hat{\mathtt{CZ}}_{q,t+1}\widehat{\mathtt{busy}}^\dagger_{q+1,t+1}\widehat{\mathtt{busy}}_{q+1,t+1}\cdots + \text{h.c.}$$

Sixt step: the driving Hamiltonian

$$\hat{H}_{\mathrm{D}} = \sum_{k} \sum_{t \leq N_T - N_T^{(k)}} \sum_{q \leq N_Q - N_Q^{(k)}} \mathbb{1} \otimes \cdots \otimes \mathbb{1} \otimes \hat{\epsilon}_k \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1}$$

All the possible equivalences

All the possible subcircuits

Rest of the circuit

Eq. operator acting on the subcircuit starting at (q,t)

Rest of the circuit

$$e^{-i\hat{H}_{\mathrm{D}}t}|C\rangle = |\text{Superposition of an exponential number of equivalent circuits.}\rangle$$

Part three: proving QA-based compilation

Quantum Annealing based compilation

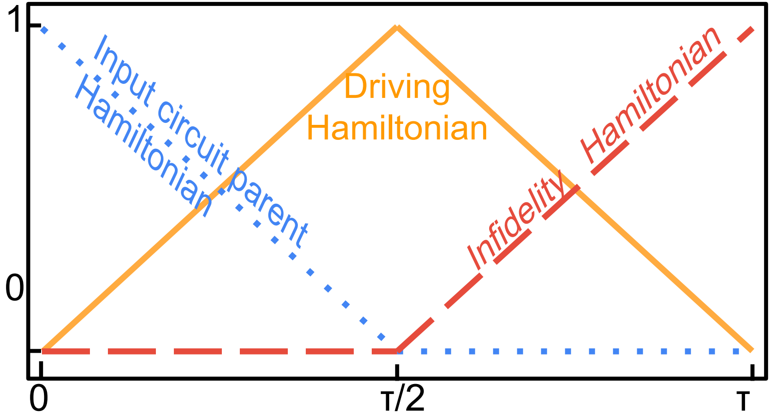
Input circuit parent Hamiltonian

Driving Hamiltonian

$$\hat{H}(t) = \begin{cases} (1-\frac{2t}{\tau})\hat{H}_0 + \frac{2t}{\tau}\hat{H}_{\mathrm{d}} & \text{if } 0 \le t \le \frac{\tau}{2}, \\ (2-\frac{2t}{\tau})\hat{H}_{\mathrm{d}} + (\frac{2t}{\tau}-1)\hat{H}_{\mathrm{I}} & \text{if } \frac{\tau}{2} \le t \le \tau \end{cases}$$

Infidelity Hamiltonian

Annealing time

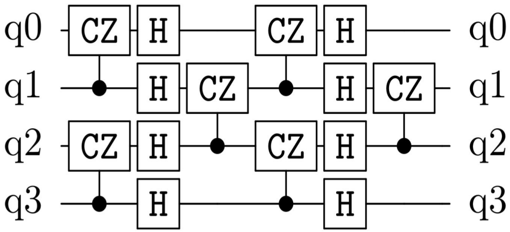


- Adiabatic theorem → for *sufficiently large* annealing time the final state converges to the ground state of the Infidelity Hamiltonian.
- The Infidelity Hamiltonian is diagonal and the Driving Hamiltonian only replaces equivalent subcircuits → the quantum evolution only explores superpositions of equivalent circuits.

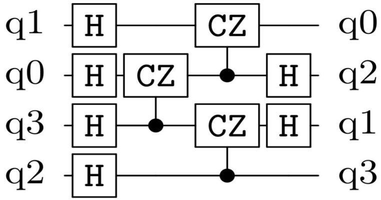
The evolution to converges to the equivalent circuit(s) that minimizes the expected infidelity.

Results

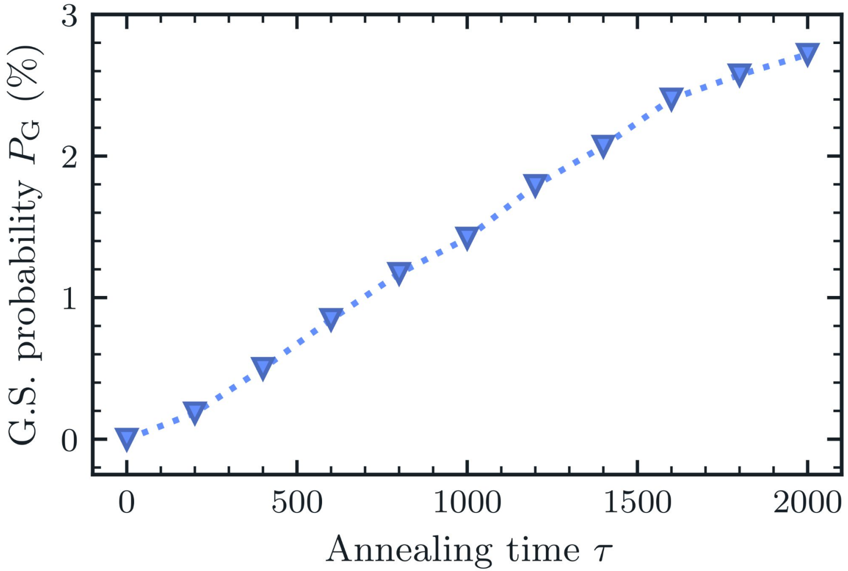
Input circuit



Global optimal circuit



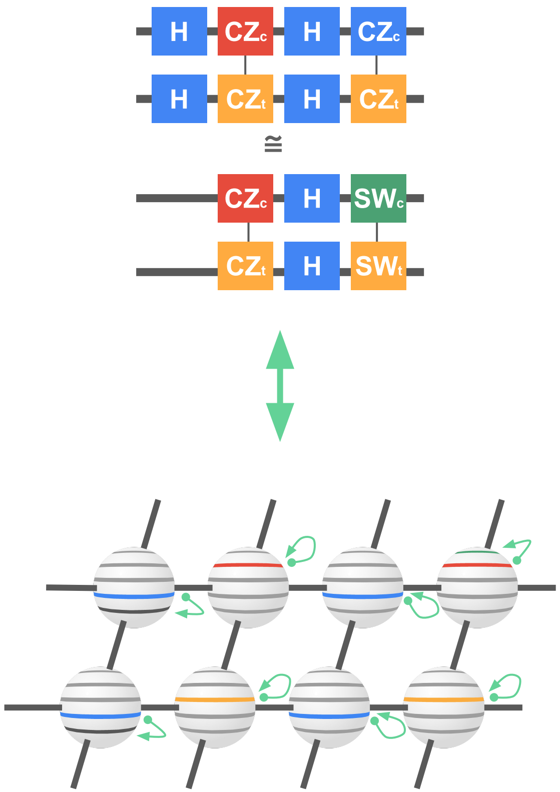
$$P_{\mathrm{G}} = \sum_{|C_{\mathrm{opt},i}\rangle} |\langle C_{\mathrm{output}} | C_{\mathrm{opt},i} \rangle|^2$$



For annealing time $\tau > 500$, $O(100)$ annealing shots are sufficient to sample the global optimal circuit!

Conclusions

- We have introduced a general paradigm for compiling quantum algorithms with quantum computers.
- Our approach is demonstrated with Quantum Annealing but extends to various techniques including QAOA and O.C.
- Possible quantum speedup for quantum (and classical!) compilation.
- The benefits of our approach increase with the system size.



pip install vulqano

INPUT CIRCUIT → Map the input circuit to a separable state.

EQUIVALENCE RULES → Recast MAPPING, SYNTHESIS and SCHEDULING as a set of Hermitian operators.

TARGET HARDWARE → Write the infidelity cost function as an Hamiltonian.

PERFORM QUANTUM OPTIMIZATION BY DRIVING THE EVOLUTION OF THE INITIAL STATE WITH A SUPERPOSITION OF THE INFIDELITY HAMILTONIAN AND DRIVING HAMILTONIAN.

MEASURE ON THE COMPUTATIONAL BASIS

OUTPUT CIRCUIT

Thank you Nora for the beautiful logo! →

Welcome to VulQano's documentation!

License

The project `vulqano` from the repository `https://baltig.infn.it/qpd/vulqano` is licensed under the following license:

Apache License 2.0

The license applies to the files of this project as indicated in the header of each file, but not its dependencies.

Compiler module

Main module for compiling quantum circuits using various optimization techniques based on many-body embedding.

Gates module

Collections of dictionaries for an abstract description of discrete and continuous (parametric) gates.

Abstract circuit states module

Define a class for abstract many-body representation of quantum circuits.

A ciruits is represented by a n-dimensional array of strings, where the first index labels the time-step, and the other indices label the position of the qubit in the lattice. Each string denote the name of the corresponding gate (see vulqano.gates.discretegates and vulqano.gates.discretegates).

Rules module

Contains all the classes and function needed to describe and implement transition rules that replace equivalent sub-circuits in a circuit state.

Markovian dynamics module

Here we define the function that performs optimization based on markovian dynamics of a circuit state (e.g. simulated annealing).

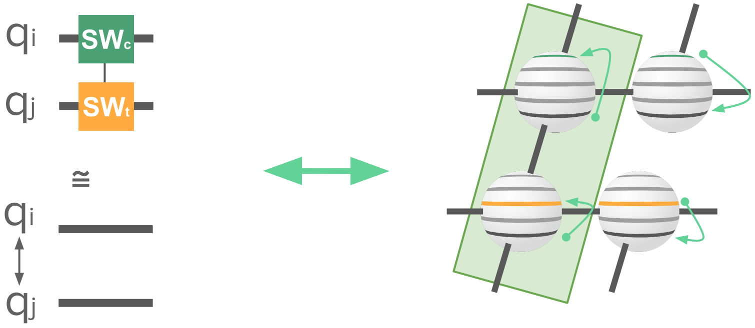
Quantum dynamics module

Here we define the function that performs optimization based on quantum dynamics of a state that encodes a superposition of circuits (e.g. quantum annealing).

The dynamics is simulated using qtealeaves.

Thank you :)

(Mapping)



$$\hat{\epsilon}_k = \widehat{\mathrm{CZ}}^{\dagger}_{q,t}\widehat{\mathrm{busy}}^{\dagger}_{q+1,t}\widehat{\mathrm{CZ}}_{q,t+1}\widehat{\mathrm{busy}}_{q+1,t+1} + \mathrm{h.c.}$$

Swapping area is forced to contain only *swap*, *busy* and *idle* gates by the Infidelity Hamiltonian.