Quantum Circuit Optimization (Transpilation)

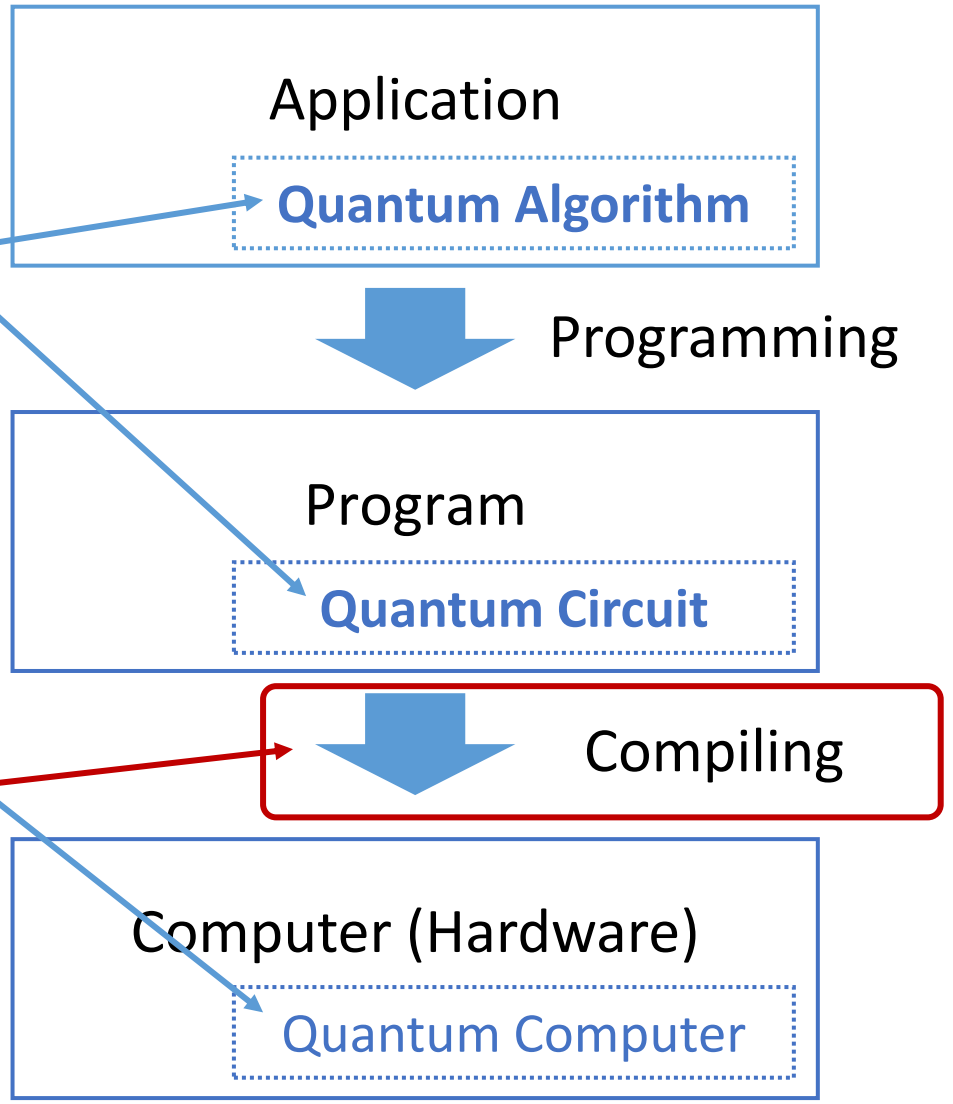
2024/06/21

Toshinari Itoko

IBM Research – Tokyo

Where we are in the syllabus

- **Basics of quantum computing (Lecture 1-3)**
  - Quantum circuit, qubits, gates, measurements
- **Quantum algorithms (Lecture 4-7)**
  - Grover, Q. Phase Estimation, VQA etc.
- **Quantum computing device (Lecture 9)**
  - Pulse-level control of transmon qubits
- **Quantum compiler (today)**
  - Quantum Circuit Optimization



Quantum Circuit Optimization (Compilation)

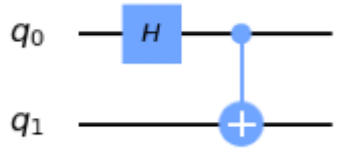
Understand **how quantum programs (circuits) are "compiled" (transformed and optimized)** before run on quantum computers.

- What is quantum compiler and why important?
- What are the main tasks of a quantum compiler (transpiler)?
- What kind of research topics are on quantum compilers?

Note: we use **classical computers** to compile quantum circuits.

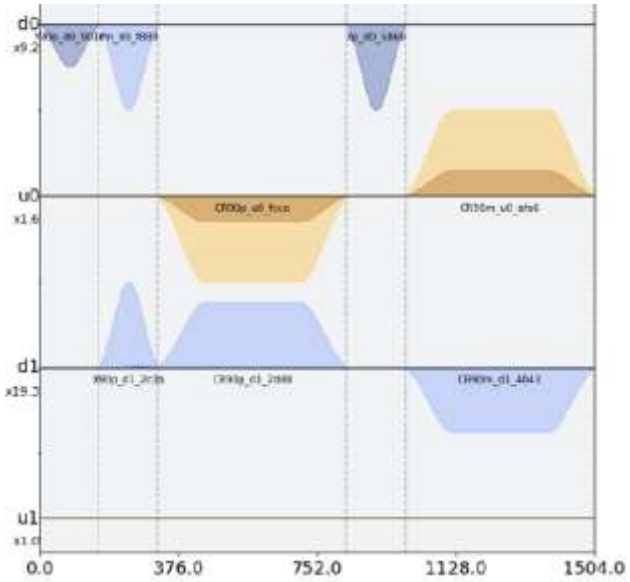
Quantum compiler (Compiler for quantum computers)

Quantum circuit (program)



“compile”

Control instructions (schedule)



Today we don’t cover … Programming language, Error-correction

- **Quantum circuit** as a representation of quantum program
- **Noisy** quantum computer as a target

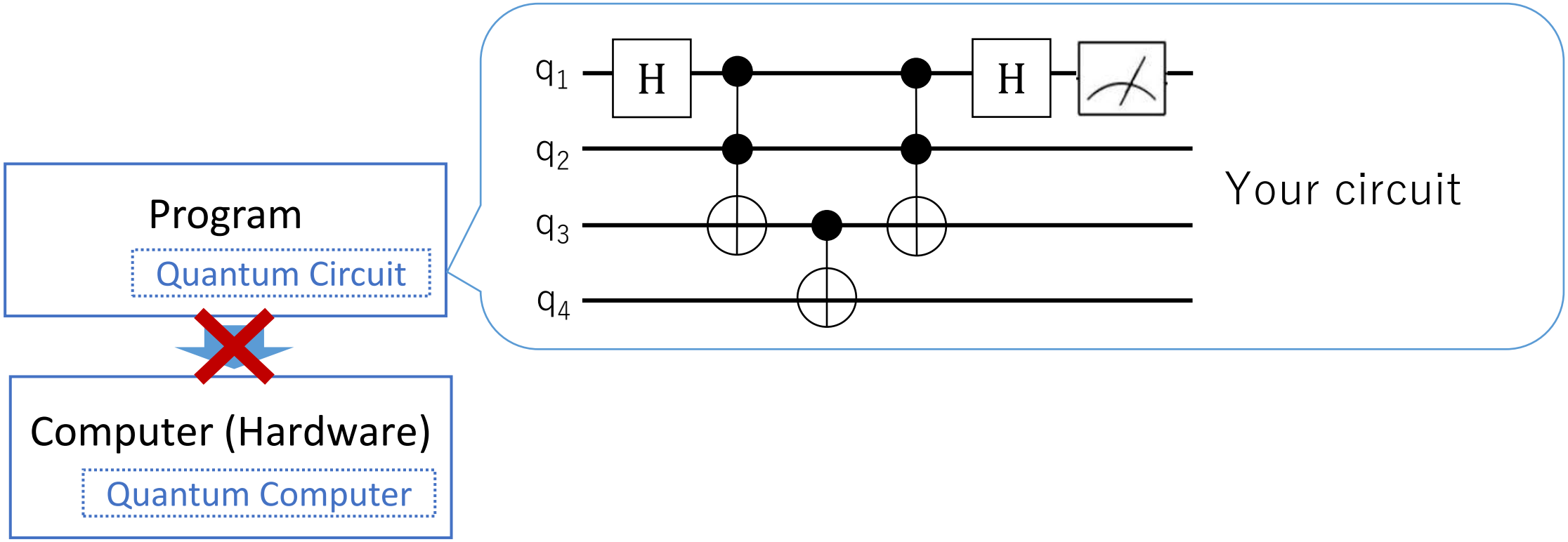
Goals of quantum compiler

Output of quantum compilers

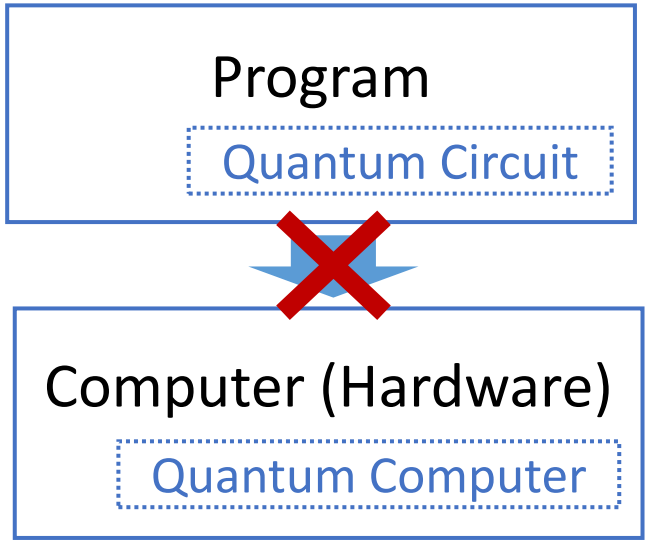
1. Must **satisfy constraints** of the quantum processor you use
   so that the processor can execute the output

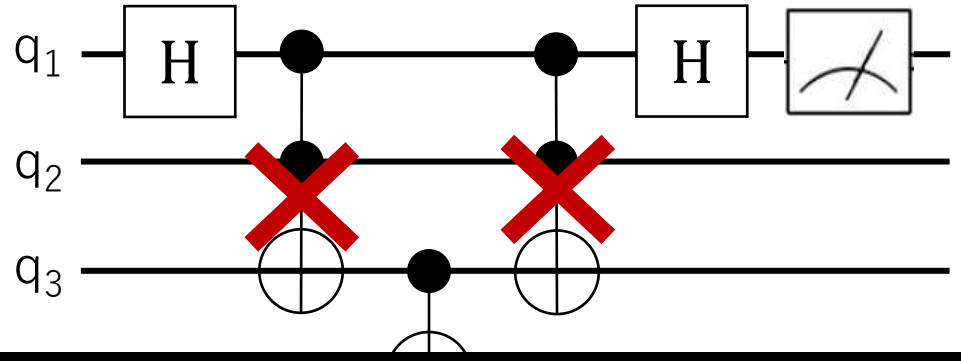
2. Should be **optimized**
   for their faster & more accurate execution

Example: When you cannot run your circuit (1)



Example: When you cannot run your circuit (2)



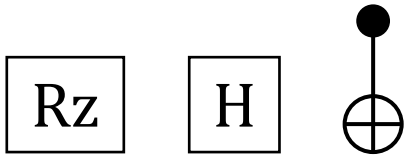


Your circuit

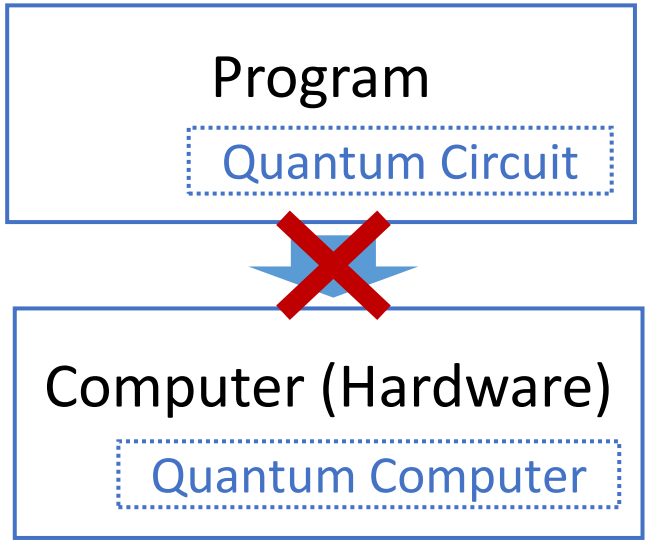
IBMInputValueError: 'The instruction ccx on qubits (0, 1, 2) is not supported by the target system.

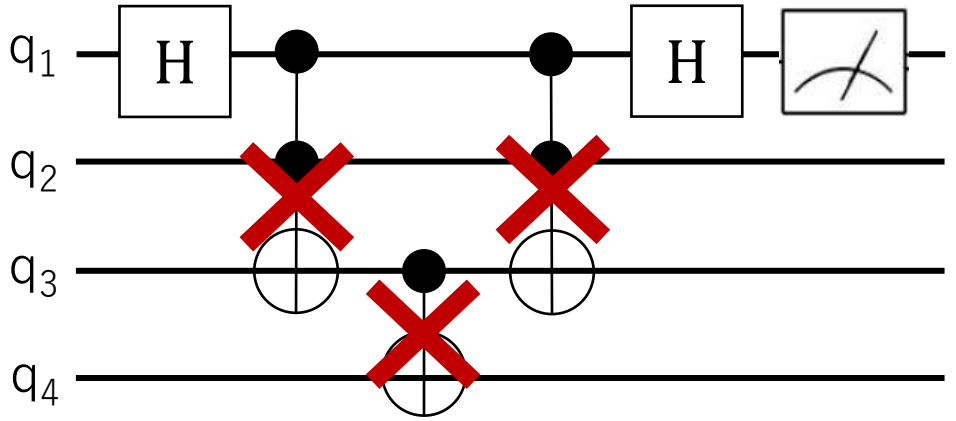
Device constraints

- Basis gate set



Example: When you cannot run your circuit (2)



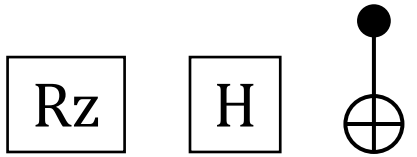


Your circuit

IBMInputValueError: 'The instruction cx on qubits (3, 4) is not supported by the target system.

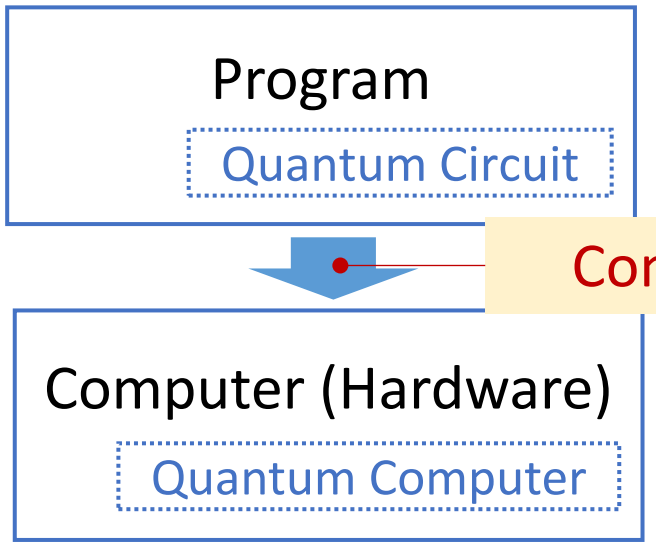
Device constraints

- Basis gate set
- Connectivity

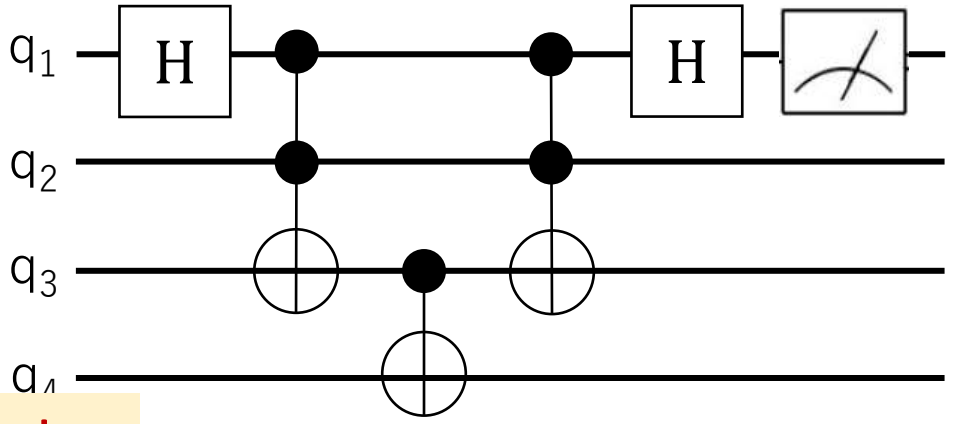


Ex: Impossible operation between q_3 and q_4

Example: When you cannot run your circuit (3)



Compile!



Your circuit

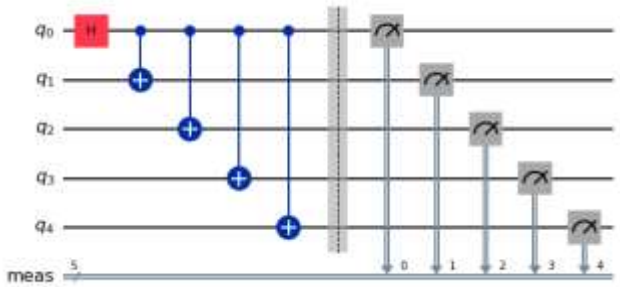
Device constraints

- Basis gate set ← Circuit synthesis
- Connectivity ← Circuit mapping

⋮

Example: Circuit optimization matters

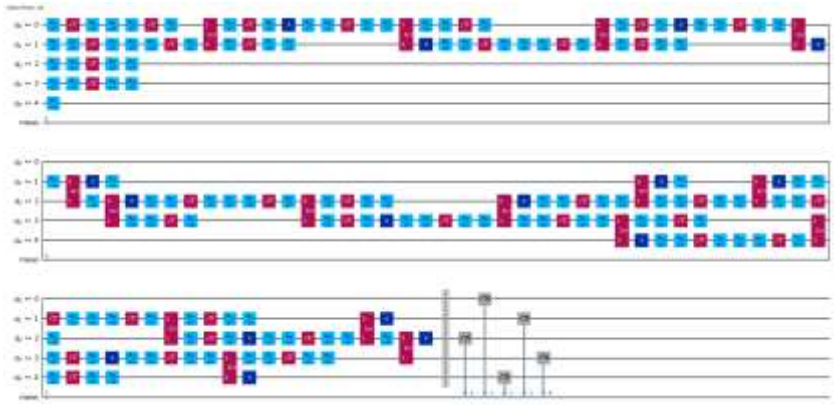
Circuit **before** compiled



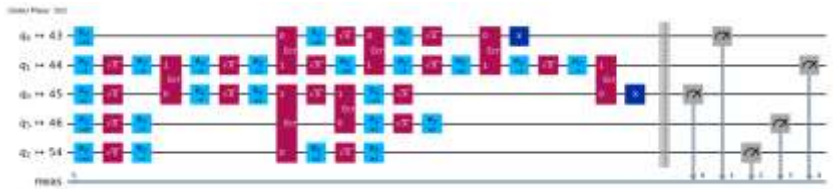
GHZ state creation

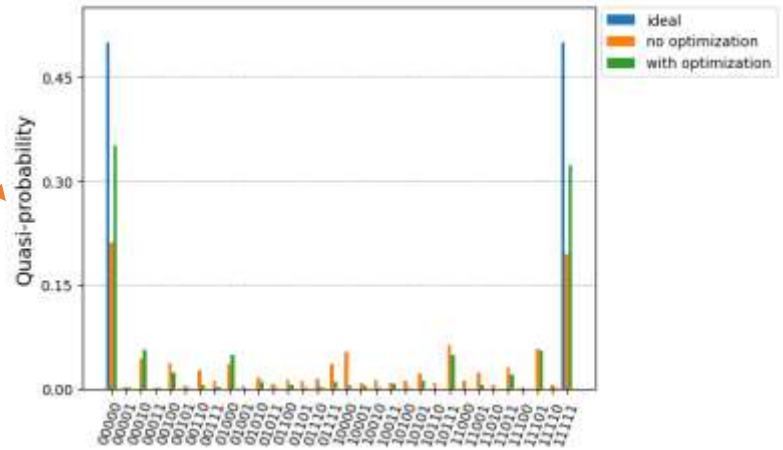
$$\frac{1}{\sqrt{2}}(|00000\rangle + |11111\rangle)$$

Circuit compiled **without optimization**:



Circuit compiled **with optimization**:





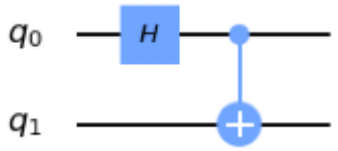
See Appendix 1 in 20240621_UTokyo_qcopt.ipynb for the details

Quantum Compiler

Transforms a quantum circuit so that the resulting instructions

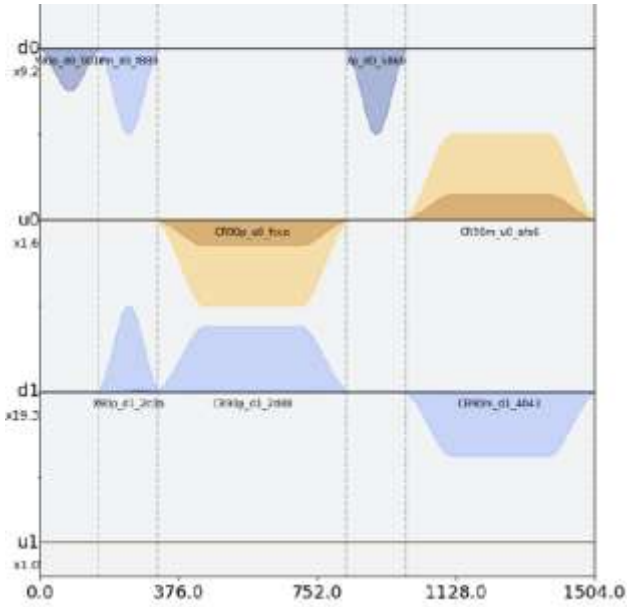
1. **satisfies** the target processor **constraints**
2. can run efficiently (**optimized**)

Quantum program (circuit)



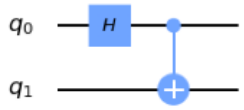
compile

Control instructions (schedule)



Rough compilation flow

Quantum circuit



Circuit synthesis

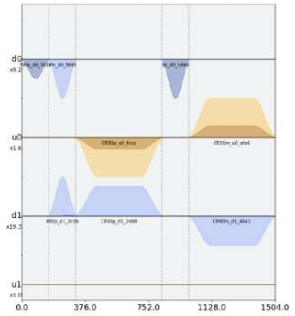
Circuit optimization

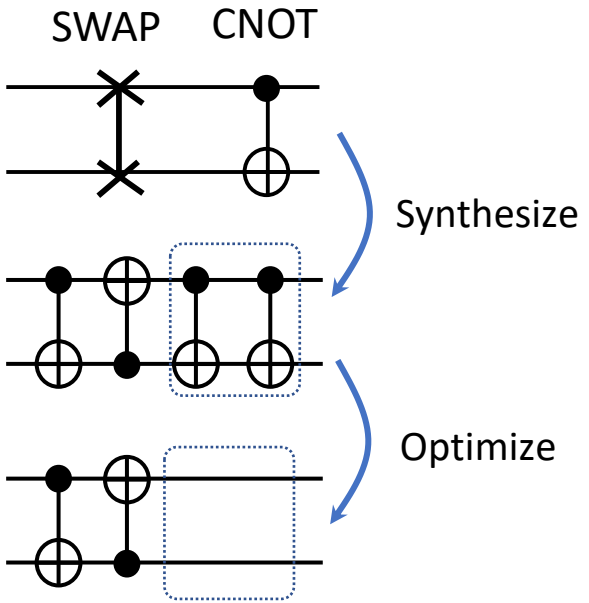
Circuit mapping

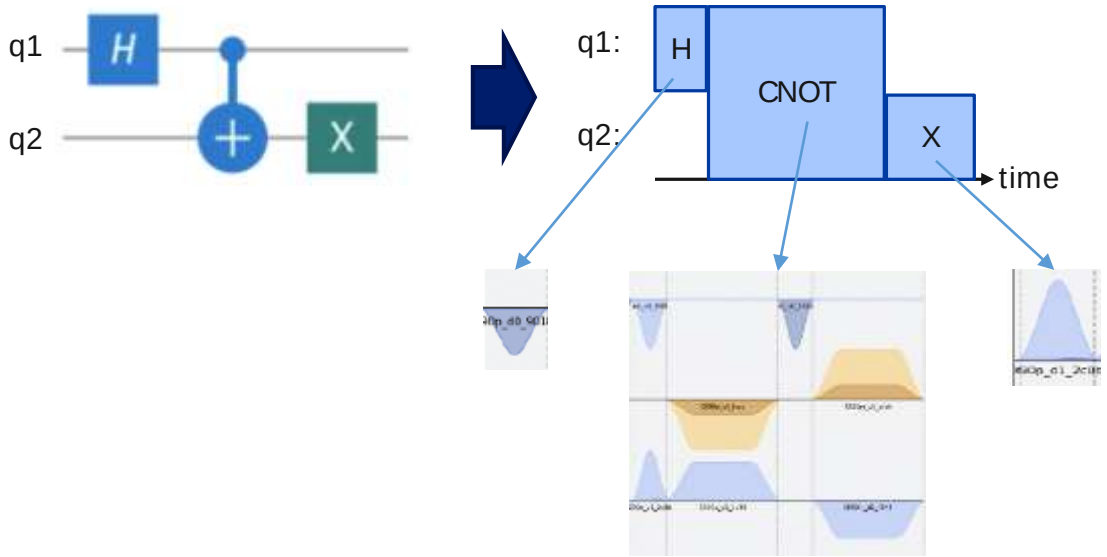
Circuit scheduling

Lowering

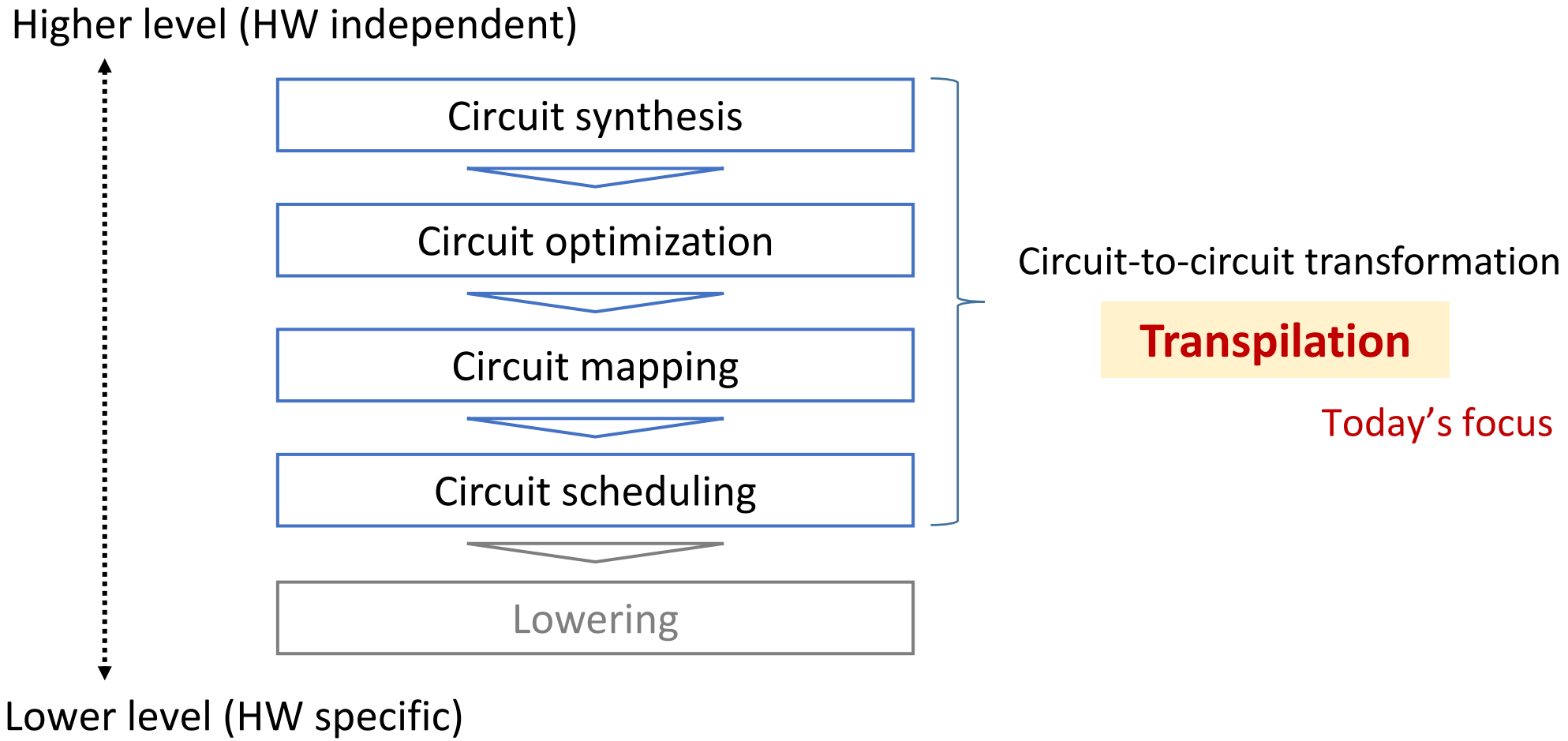
Control instructions





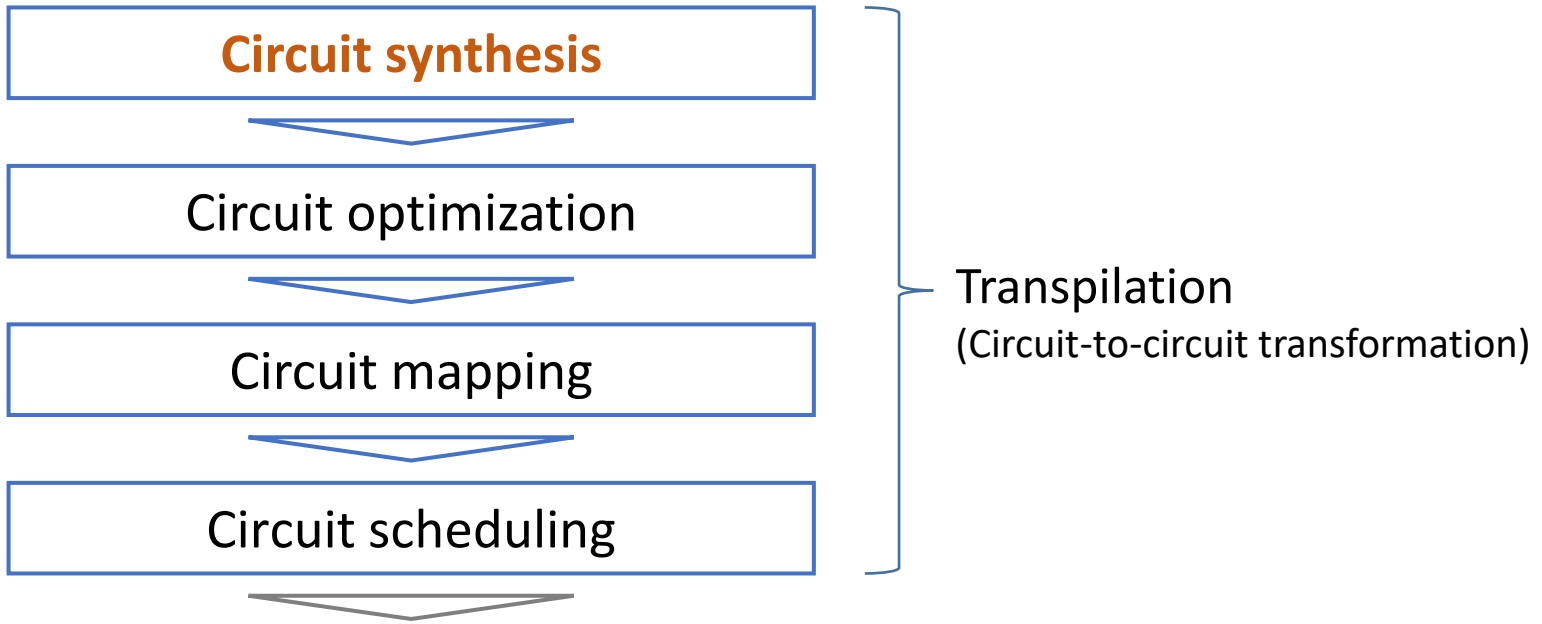


Transpilation in compiler tasks



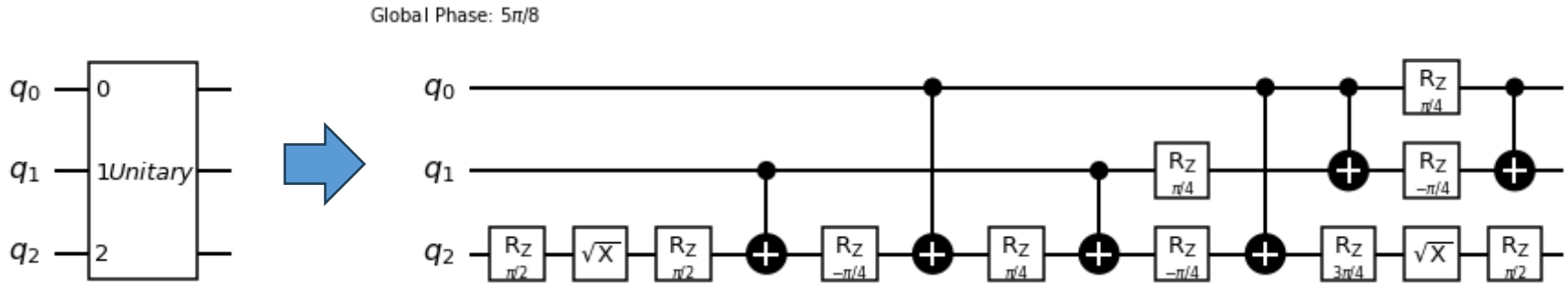
- Target HW: Superconducting-qubit type processors
- All compiler tasks are classical date processing

Circuit synthesis (Gate decomposition)



Circuit synthesis (Gate decomposition)

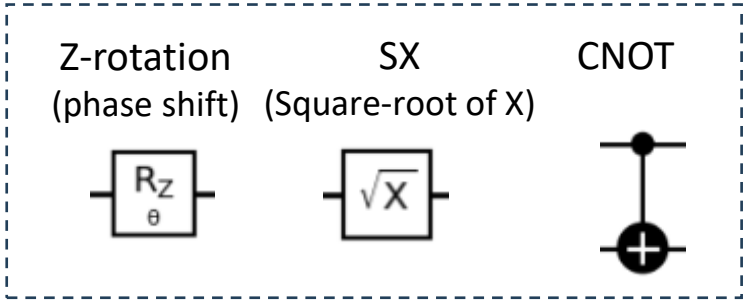
Synthesize a high-level gate (unitary operation) with basis gates of a target processor



Basis gate set (Basis gates)

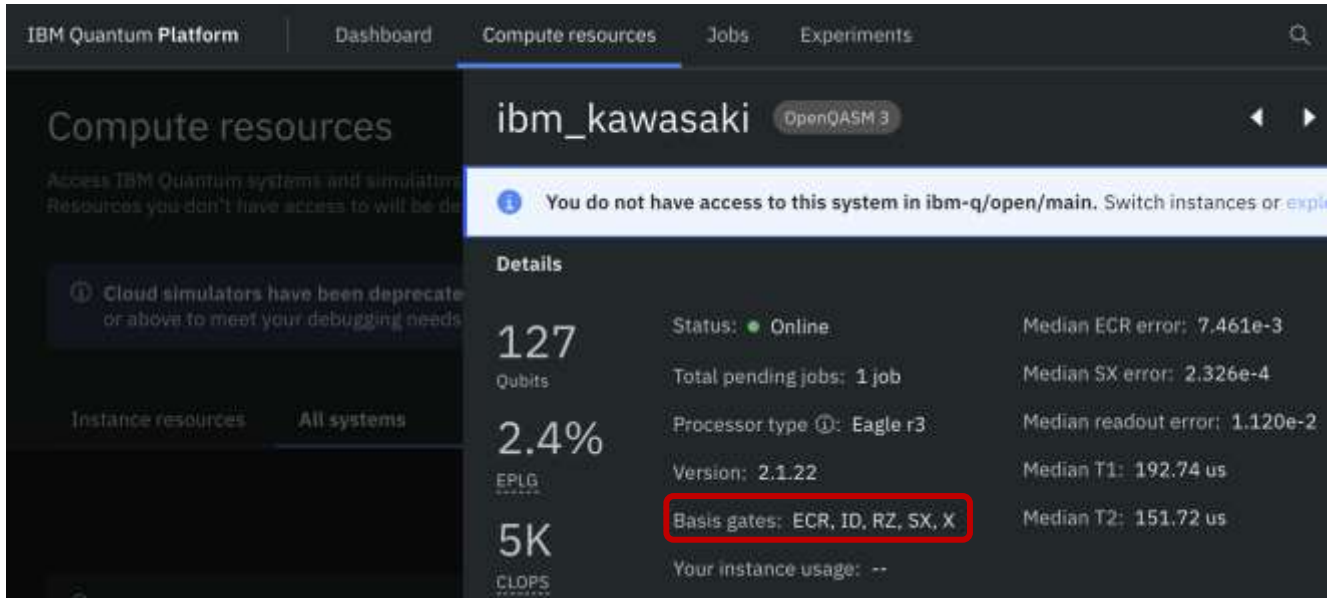
- Usually, a **universal** gate set
  - A gate set such that it can approximate any gate to any desired precision
- Typically, 1- and 2-qubit gates

Example of basis gate set

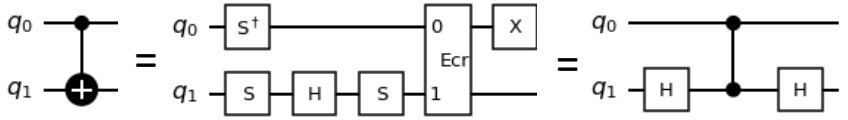


See Barenco et al. “Elementary gates for quantum computation.” *Physical review A* 52.5 (1995): 3457 for the theory on basis gates

Hardware-native basis gates (IBM Quantum processors)



- Common 1-qubit gates: Rz, SX, X
- Different 2-qubit gates: CX/ECR/CZ
  (but equivalent up to local 1q-gates)



https://quantum.ibm.com/services/resources

Falcon (27 qubits)

Processor type ⓘ: Falcon r5.11

Version: 1.9.1263

Basis gates: CX, ID, RZ, SX, X

Eagle (127 qubits)

Processor type ⓘ: Eagle r3

Version: 2.1.22

Basis gates: ECR, ID, RZ, SX, X

Heron (133 qubits)

Processor type ⓘ: Heron r1

Version: 1.0.15

Basis gates: CZ, ID, RZ, SX, X

Hardware-native basis gates (Others)

Source: Craig Gidney's post in StackExchange (https://quantumcomputing.stackexchange.com/questions/20836/what-is-the-basic-hardware-gate-library-in-the-ibm-google)

Google

One qubit gate: $U3$ (general one-qubit gates pf any rotation), X, Y, Z.

Two qubit gates: Sycamore gate, which has the matrix representation as

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & e^{-i\pi/8} \end{pmatrix}$$

The $\sqrt{iSWAP}$ gate which has the matrix representation as

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Other two qubit gate that being supported is the CZ gate.

https://quantumai.google/cirq/google/devices

IonQ

One qubit gates:

$$GPI(\phi) = \begin{pmatrix} 0 & e^{-i\phi} \\ e^{i\phi} & 0 \end{pmatrix}, GPI2(\phi) = \begin{pmatrix} 1 & -ie^{-i\phi} \\ -ie^{i\phi} & 1 \end{pmatrix}, GZ(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

Two qubit gates: The Mølmer-Sørenson gate (MS): $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & -i \\ 0 & 1 & -i & 0 \\ 0 & -i & 1 & 0 \\ -i & 0 & 0 & 1 \end{pmatrix}$

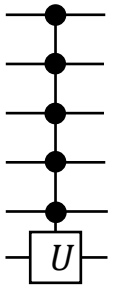
https://ionq.com/docs/getting-started-with-native-gates

- Different QPUs, Different basis gates
- Every set is a universal gate set

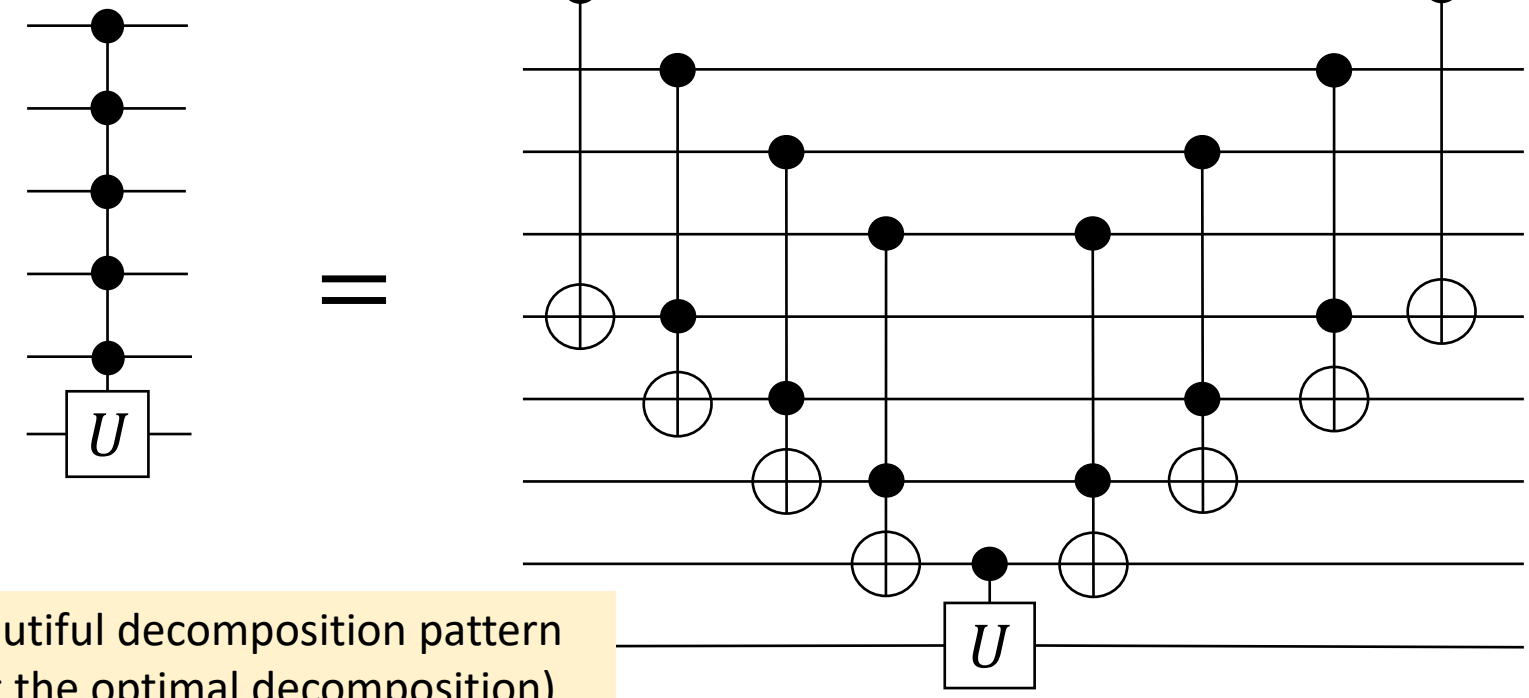
QPU: Quantum Processing Unit

Example: Decomposition of a 6-qubit gate

Task:

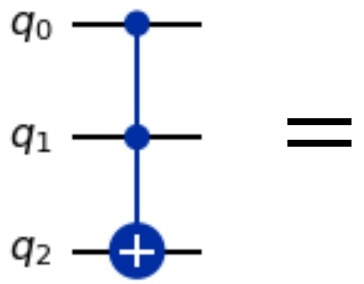
Synthesize a circuit equivalent with 

using basis gates Rz, SX, CNOT

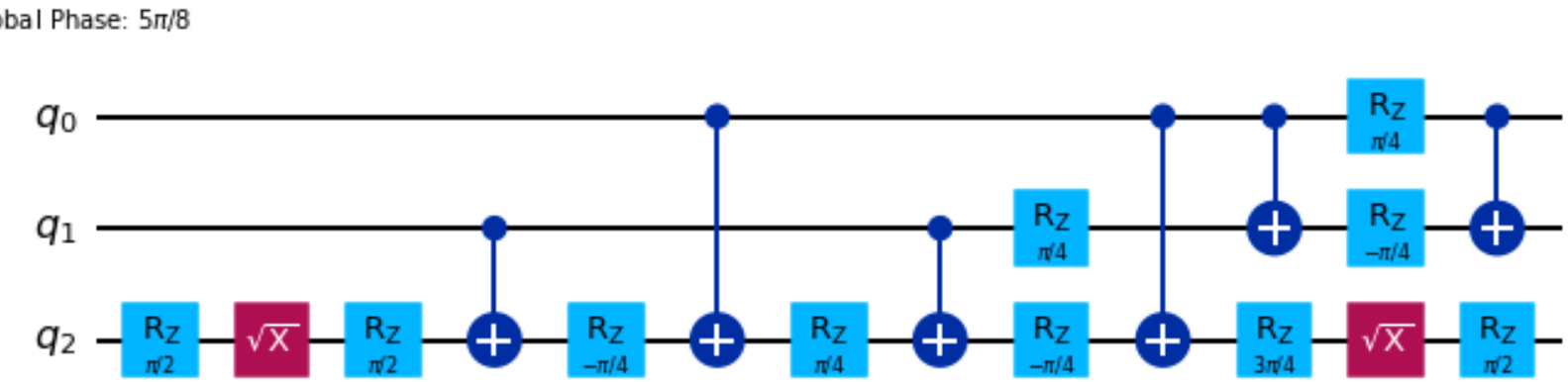


Finding a beautiful decomposition pattern
(or searching the optimal decomposition)

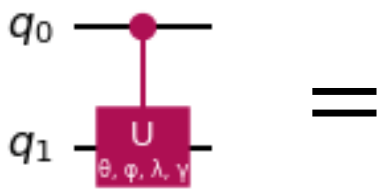
Example: Decomposition of Toffoli/Controlled-U gate



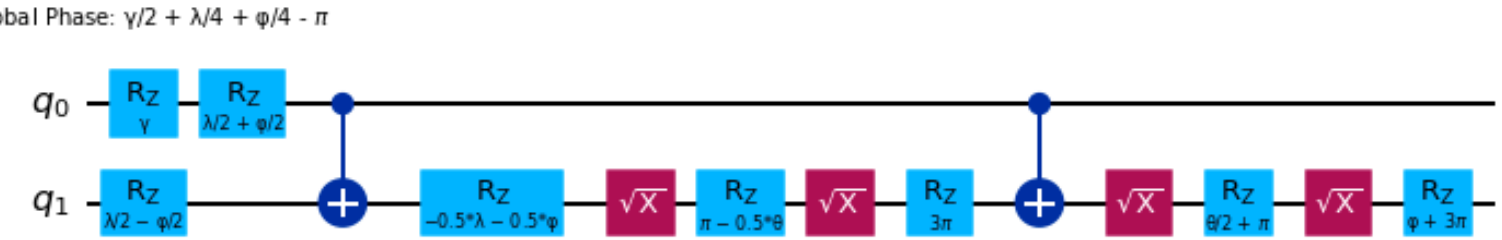
Toffoli gate
(CCX gate)




```
from qiskit import QuantumCircuit, transpile
qc = QuantumCircuit(3)
qc.ccx(0, 1, 2)
qc = transpile(qc, basis_gates=["rz", "sx", "cx"])
qc.draw(output="mpl")
```

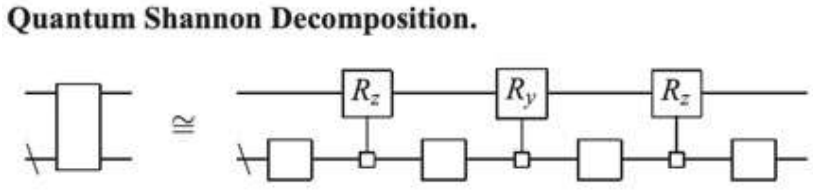


Controlled U gate
(CU gate)




```
from qiskit.circuit.library.standard_gates import CUGate
phi, theta, lam, gamma = Parameter("φ"), Parameter("θ"), Parameter("λ"), Parameter("γ")
qc = QuantumCircuit(2)
qc.append(CUGate(theta, phi, lam, gamma), [0, 1])
qc = transpile(qc, basis_gates=["rz", "sx", "cx"])
qc.draw(output="mpl")
```

Studies on circuit synthesis (gate decomposition)

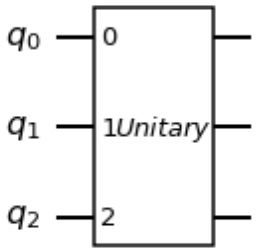


Source: Theorem 13 in [1]

- Unitary synthesis
  - Quantum Shannon Decomposition (QSD) [1]
  - Input: 2^n x 2^n unitary matrix, Output: $\Omega(4^n)$ (Gate count)

  → Generalization to 2^n x 2^m matrix (m=1: initializer): Isometry [2]
- Application-oriented synthesis
  - Arithmetic operations (e.g. adder, modular exponentiation), Clifford operations
  - Time evolution operator (for Hamiltonian simulation)
  - Quantum Fourier Transformation (QFT)
  - Multi-controlled gate decomposition
    - Multiple-control Toffoli (MCT) gate

How to represent

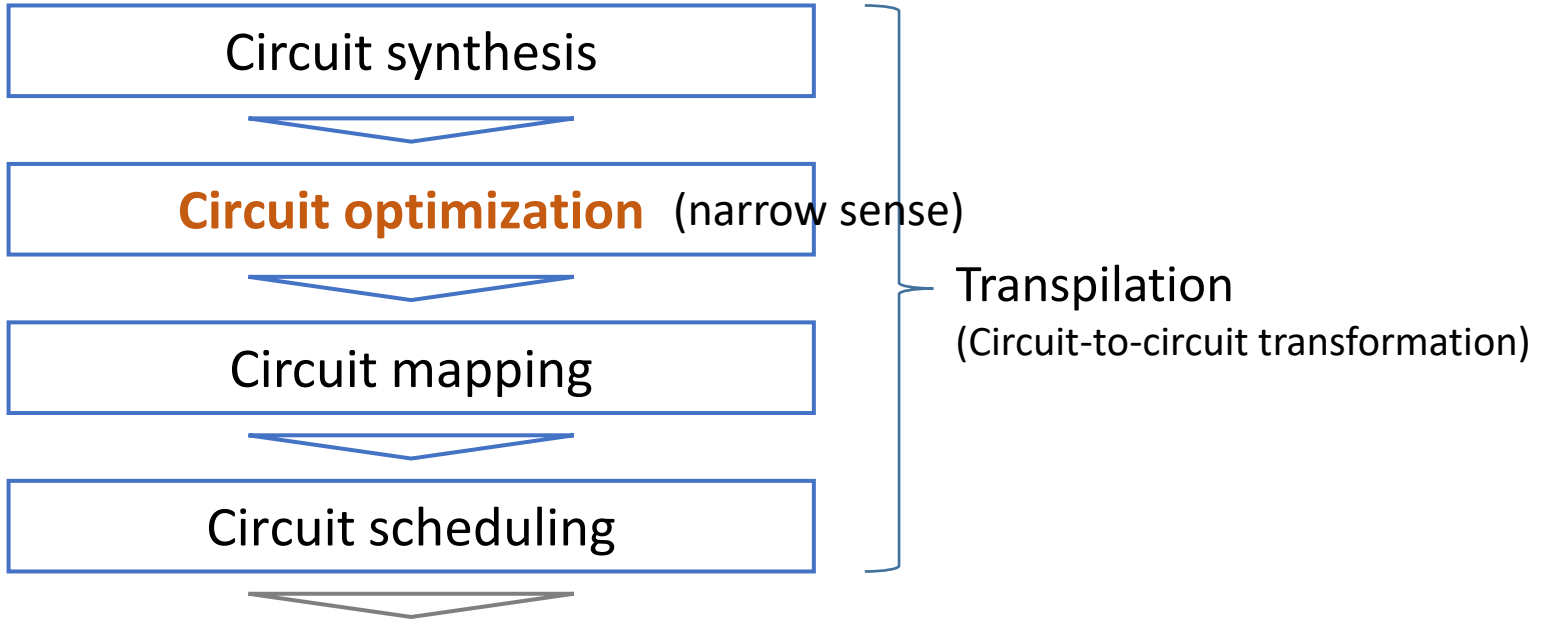


???

[1] V. V. Shende et al. “Synthesis of quantum-logic circuits.” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000--1010, 2006. (An implementation in Qiskit: https://github.com/Qiskit/qiskit-terra/pull/7907)
[2] R. Iten et al. “Quantum circuits for isometries.” *Physical Review A*, 93(3):032318, 2016.

(See also Section 2.1 Quantum Circuit Synthesis in my PhD thesis https://tsukuba.repo.nii.ac.jp/records/2000758 for more references.)

Circuit optimization
(Circuit simplification, Peephole optimization)



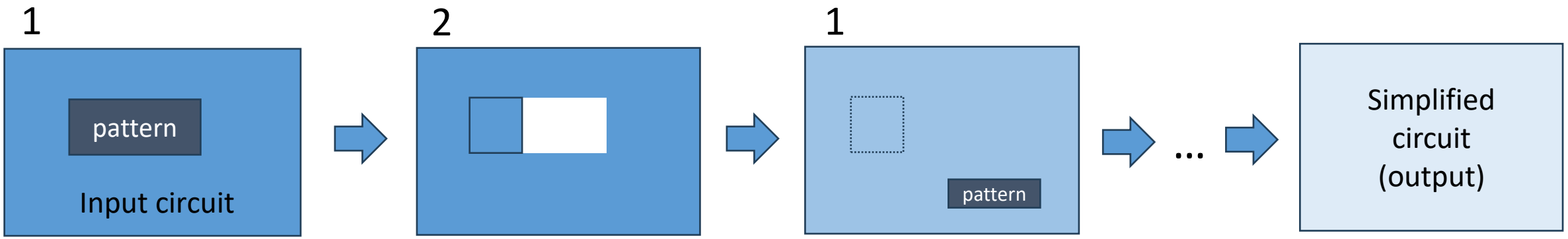
Circuit optimization (broad sense) = All optimization tasks in transpilation

Circuit optimization: Circuit → Simplified circuit

Given a circuit, return a simplified circuit.

Major strategy: Pattern matching

a.k.a. peephole optimization



1. Find a pattern (a sequence of gates to be simplified)
2. Replace it with a simpler sequence of gates

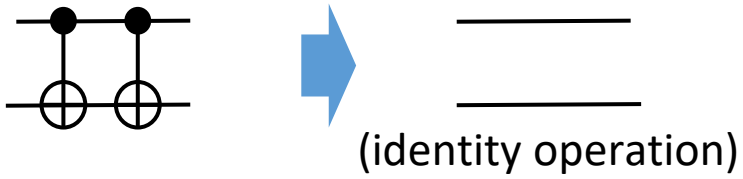
(Repeat 1 and 2 until 1 fails)

Template matching

(Template = Pattern depending on basis gates)

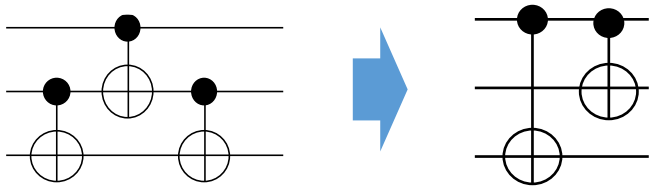
Templates to reduce the number of CNOT gates:

Template A:

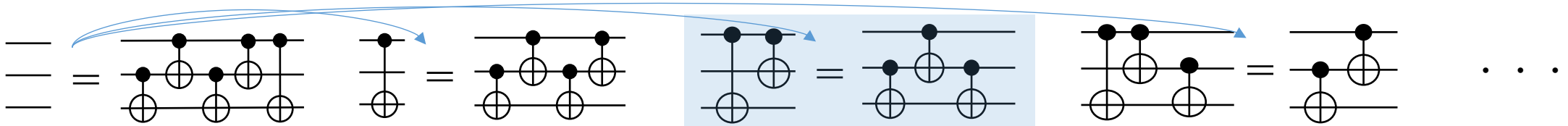


$$\begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0 \end{pmatrix}\begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix}$$

Template B2:



Maslov, Dmitri, et al. "Quantum circuit simplification using templates." *Design, Automation and Test in Europe*. IEEE, 2005.



More templates -> More chance of optimization (at the price of more computation)

Single-qubit block optimization (Pattern **not** depending on basis gates)

Any 1q-gate sequence can always be


```
from qiskit import QuantumCircuit
qc = QuantumCircuit(1)
qc.x(0)
qc.y(0)
qc.z(0)
qc.rx(1.23, 0)
qc.ry(1.23, 0)
qc.rz(1.23, 0)
qc.h(0)
qc.s(0)
qc.t(0)
qc.sx(0)
qc.sdg(0)
qc.tdg(0)
qc.draw(output="mpl")
```

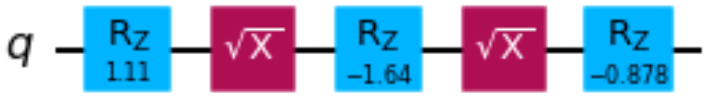
By multiplying all 2x2 gate matrices …

```
[[ 0.45292511-0.57266982j, -0.66852684-0.14135058j],
 [ 0.14135058+0.66852684j, -0.57266982+0.45292511j]]
```

Simplified into a $R_z \rightarrow \sqrt{X} \rightarrow R_z \rightarrow \sqrt{X} \rightarrow R_z$ sequence

… and synthesizing the matrix (1-qubit unitary synthesis)

Global Phase: 3π/4




```
from qiskit import transpile
qc = transpile(qc, basis_gates=["rz", "sx"])
qc.draw(output="mpl")
```

Gate count reduction!

Theory behind 1q-gate synthesis: ZYZ decomposition

U: 2 x 2 unitary matrix (1-qubit gate)

$$U = e^{i\alpha'} R_z(\phi) R_y(\theta) R_z(\lambda) \quad \text{with } \alpha', \phi, \theta, \lambda \in \mathbb{R}$$

Global phase

Z-rotation

Y-rotation

Z-rotation

$$U = e^{i\alpha'} \begin{bmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{bmatrix}$$

$$U = \begin{bmatrix} e^{i(\alpha' - \frac{\phi+\lambda}{2})} \cos\frac{\theta}{2} & -e^{i(\alpha' - \frac{\phi-\lambda}{2})} \sin\frac{\theta}{2} \\ e^{i(\alpha' + \frac{\phi-\lambda}{2})} \sin\frac{\theta}{2} & e^{i(\alpha' + \frac{\phi+\lambda}{2})} \cos\frac{\theta}{2} \end{bmatrix}$$

M. A. Nielsen and I. L. Chuang, Quantum computation and quantum information, Cambridge Series on Information and the Natural Sciences

ZYZ decomposition - Practice

IBM Quantum systems do not have Ry gate in their basis gates

$$U(\phi,\theta,\lambda) = R_z(\phi)\boxed{R_y(\theta)}R_z(\lambda)$$

Not supported

```
In [10]: backend.configuration().basis_gates
Out[10]: ['id', 'rz', 'sx', 'x', 'cx', 'reset']
```

sx: Square-root of X

$$\sqrt{X} = R_x(\pi/2)$$ (Up to global phase)

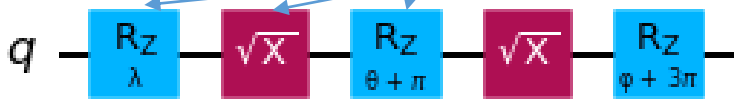
$$R_y(\theta) = R_x(-\pi/2)R_z(\theta)R_x(\pi/2)$$

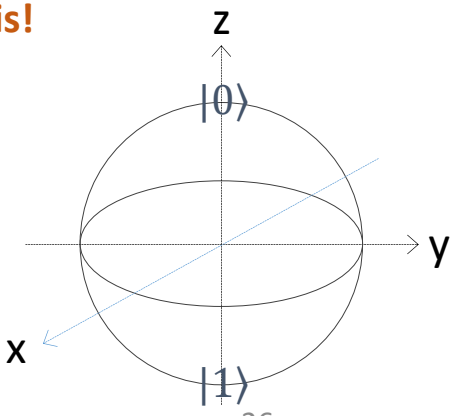
Change back the Y-axis! Change Y-axis to be Z-axis!

$$R_x(-\pi/2) = R_z(-\pi)R_x(\pi/2)R_z(\pi)$$

Change back the (-X)-axis! Change (-X)-axis to be X-axis!

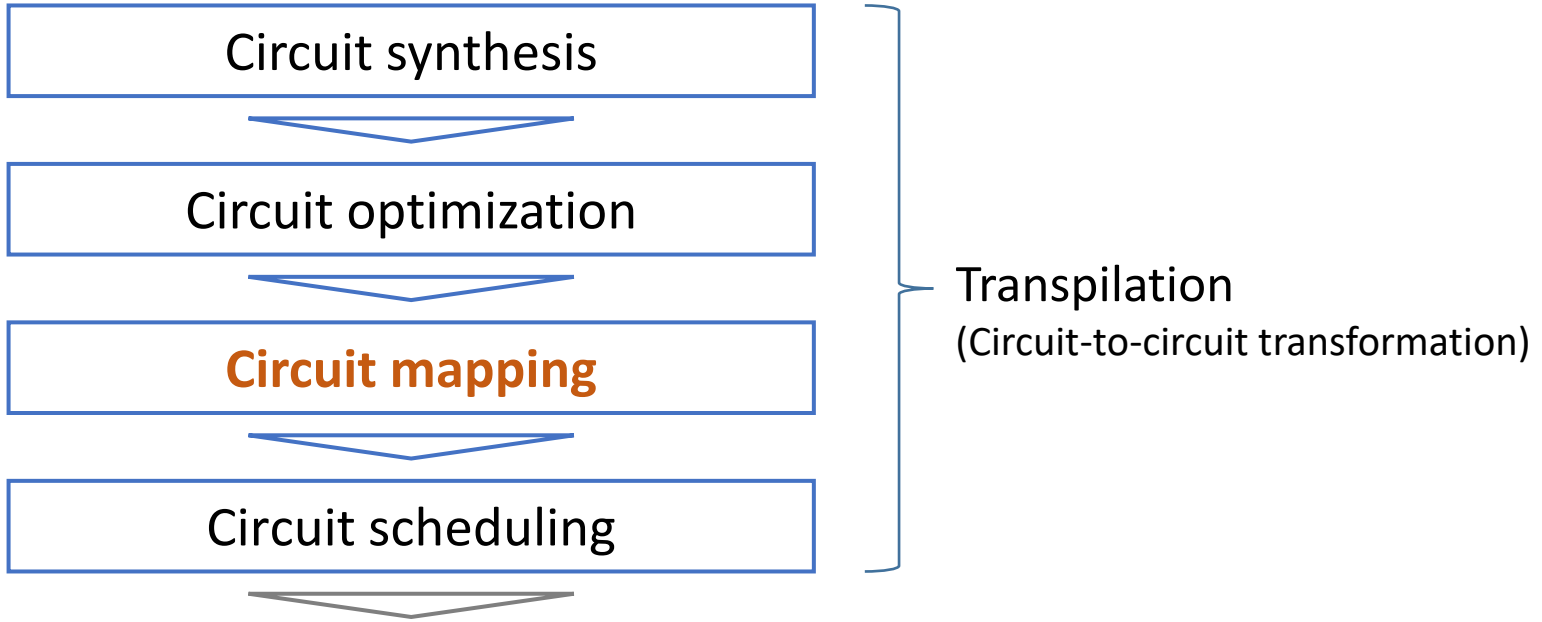
$$U(\phi,\theta,\lambda) = R_z(\phi-\pi)\boxed{R_x(\pi/2)}R_z(\theta+\pi)\boxed{R_x(\pi/2)}R_z(\lambda)$$





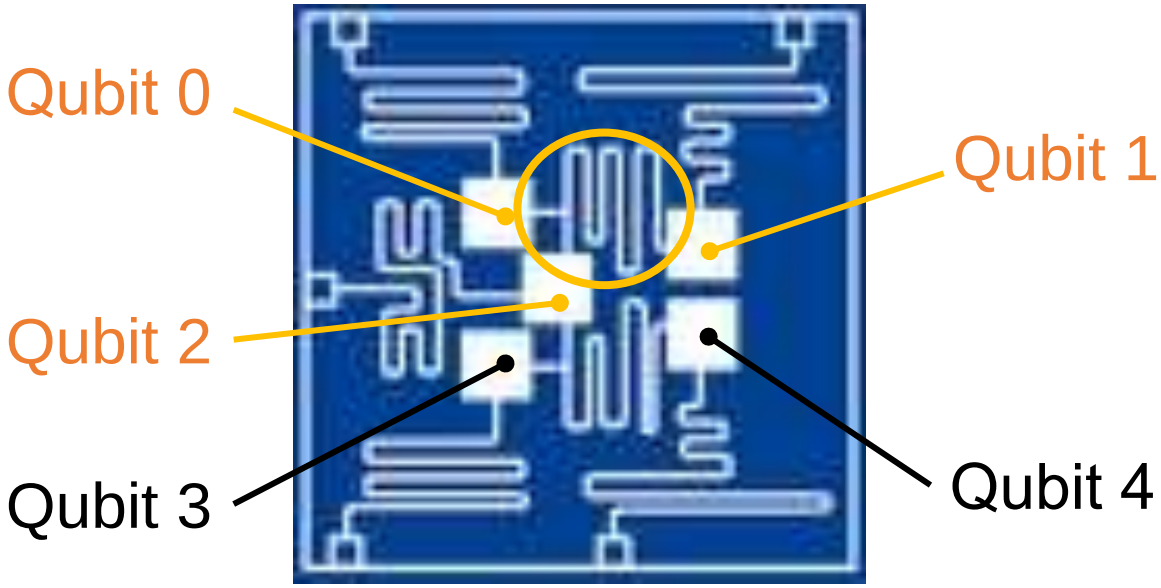
Any 1-qubit gate can be decomposed with at most two sx gates!

Circuit mapping
(Qubit layout + Qubit routing)



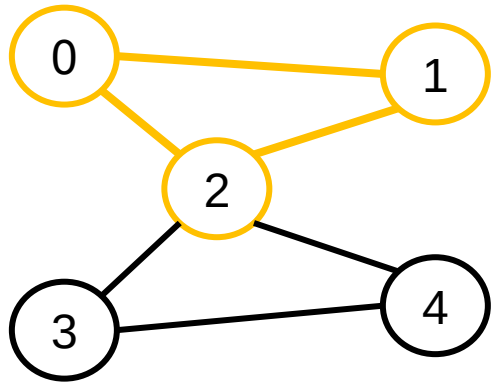
Coupling constraint (a.k.a. Nearest Neighbor constraint)

Two-qubit gates are implementable only on "coupled" qubits



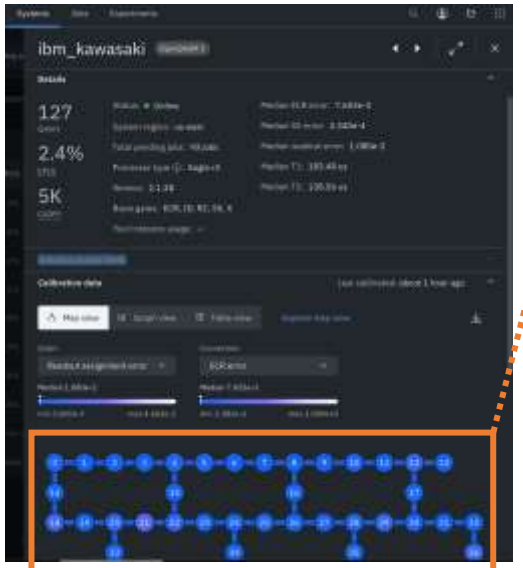
Device image of IBM Q 5 Tenerife [ibmqx4]

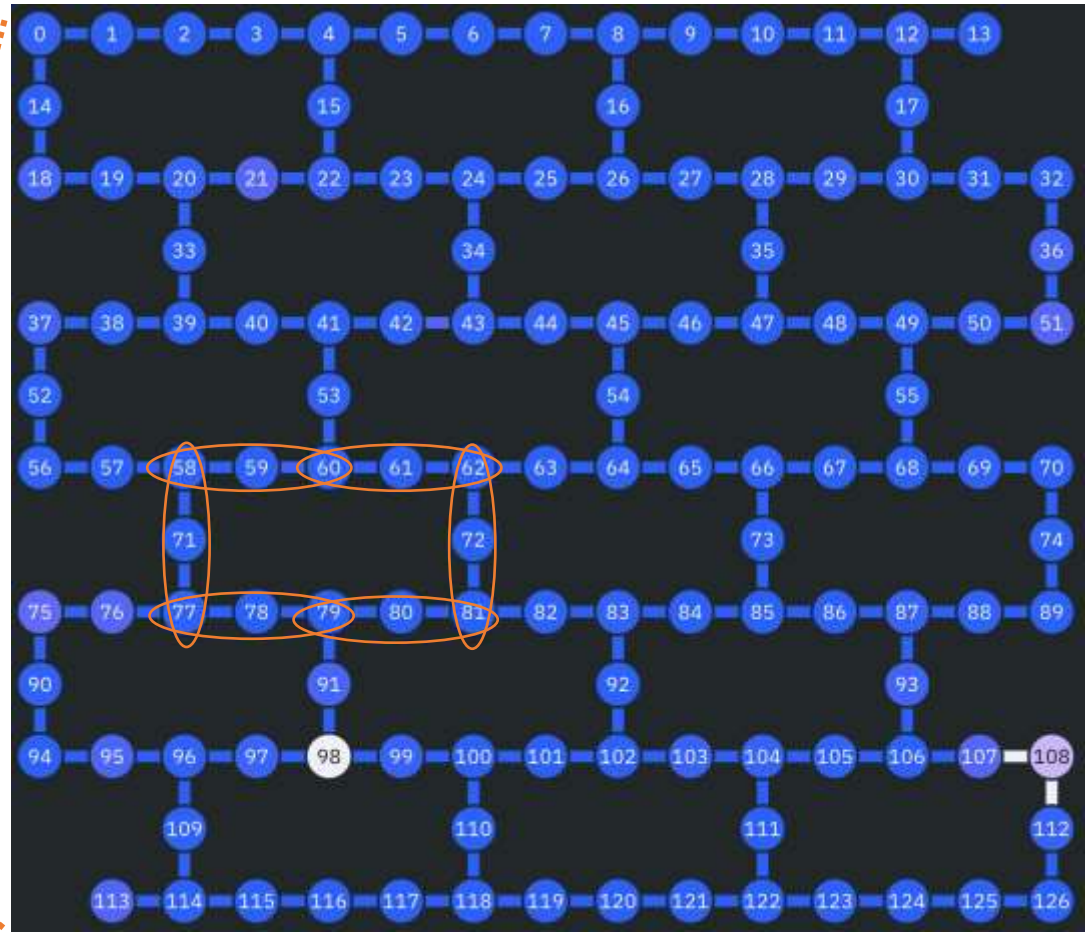
Coupling Graph



Node ⇔ Qubit
Edge ⇔ Coupler

Example: Coupling graph of IBM Eagle processor (ibm_kawasaki)





Node ⇔ Qubit
Edge ⇔ Coupling

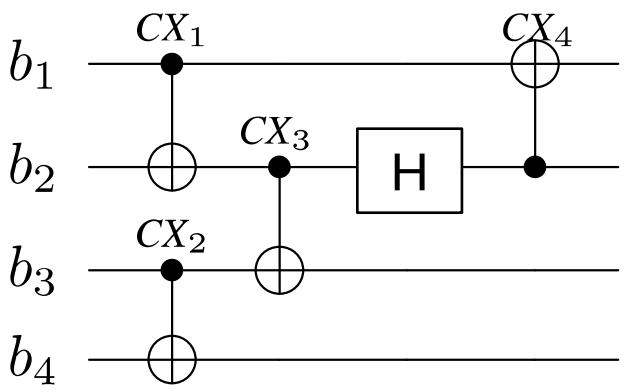
- 127 qubits
- Heavy-hex connectivity

IBM Quantum Platform > Systems > ibm_kawasaki

Quantum circuit mapping: Problem

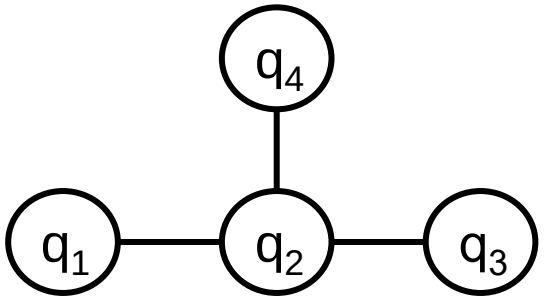
Given a circuit, transform it into an equivalent circuit
so that all two-qubit gates are placed on coupled qubits

IN: Virtual circuit

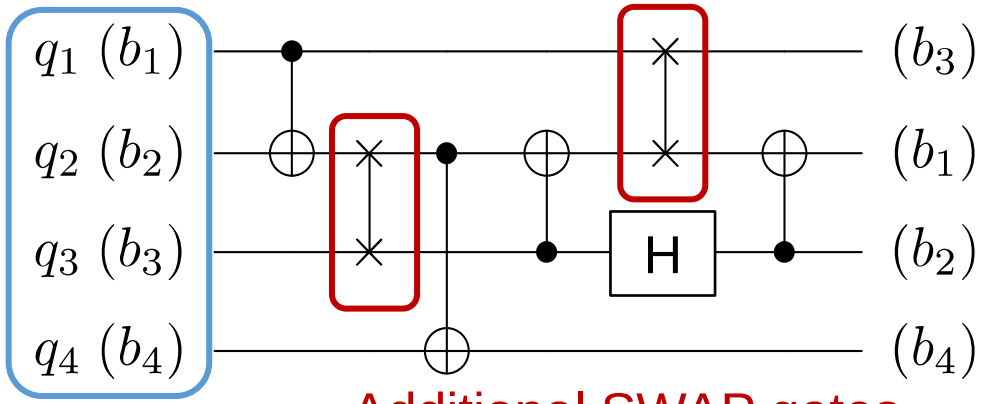


Assume all are 1- or 2-qubit gates

Coupling constraint



OUT: Physical circuit

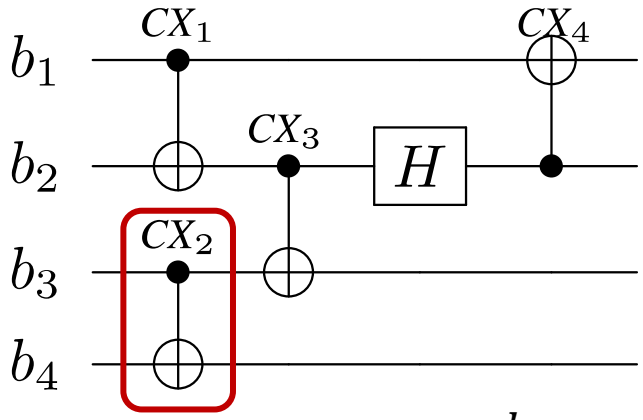


Additional SWAP gates

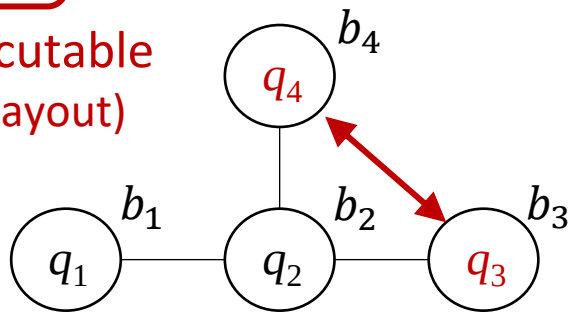
Initial qubit layout

Adding SWAP gates to satisfy coupling constraint

Suppose an initial layout : $b_i \rightarrow q_i$ for i = 1,2,3,4

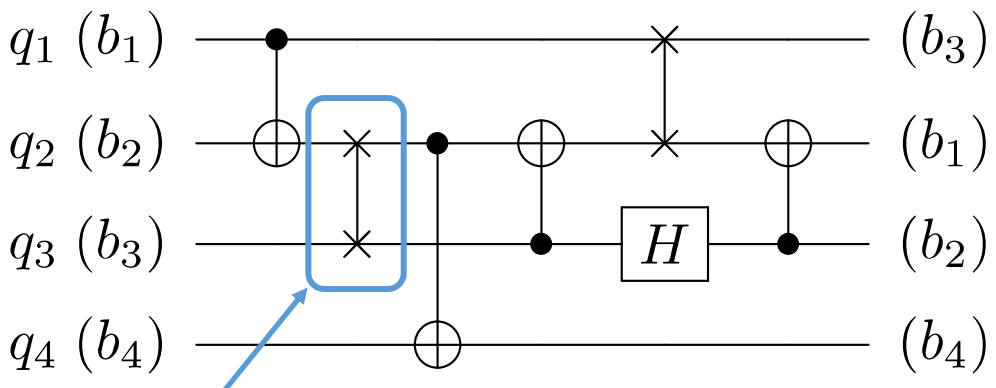


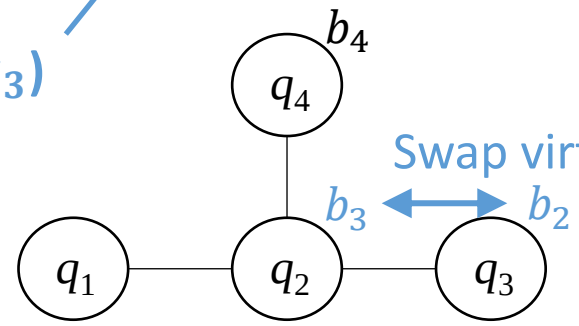
Not executable
(in the layout)



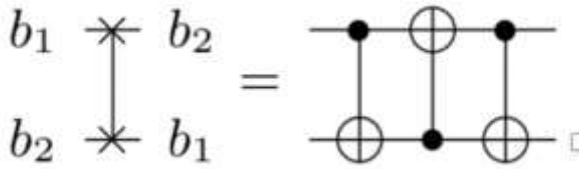
Add SWAP(q_2, q_3)

(Change the layout)





Swap virtual qubits (in the layout)

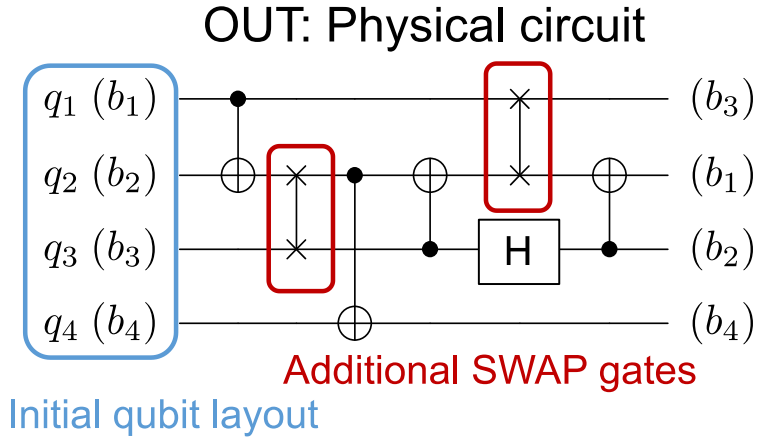


SWAP gates cause much error

→ **Minimize the number of SWAP gates**

Two approaches to circuit mapping

Circuit mapping can be decomposed into two subtasks:
- Qubit layout: Find possibly optimal initial layout
- Qubit routing: Optimize insertion of SWAPs under given initial layout



1. Solve at once

Circuit mapping

+ Optimal solution
− Slow algorithm

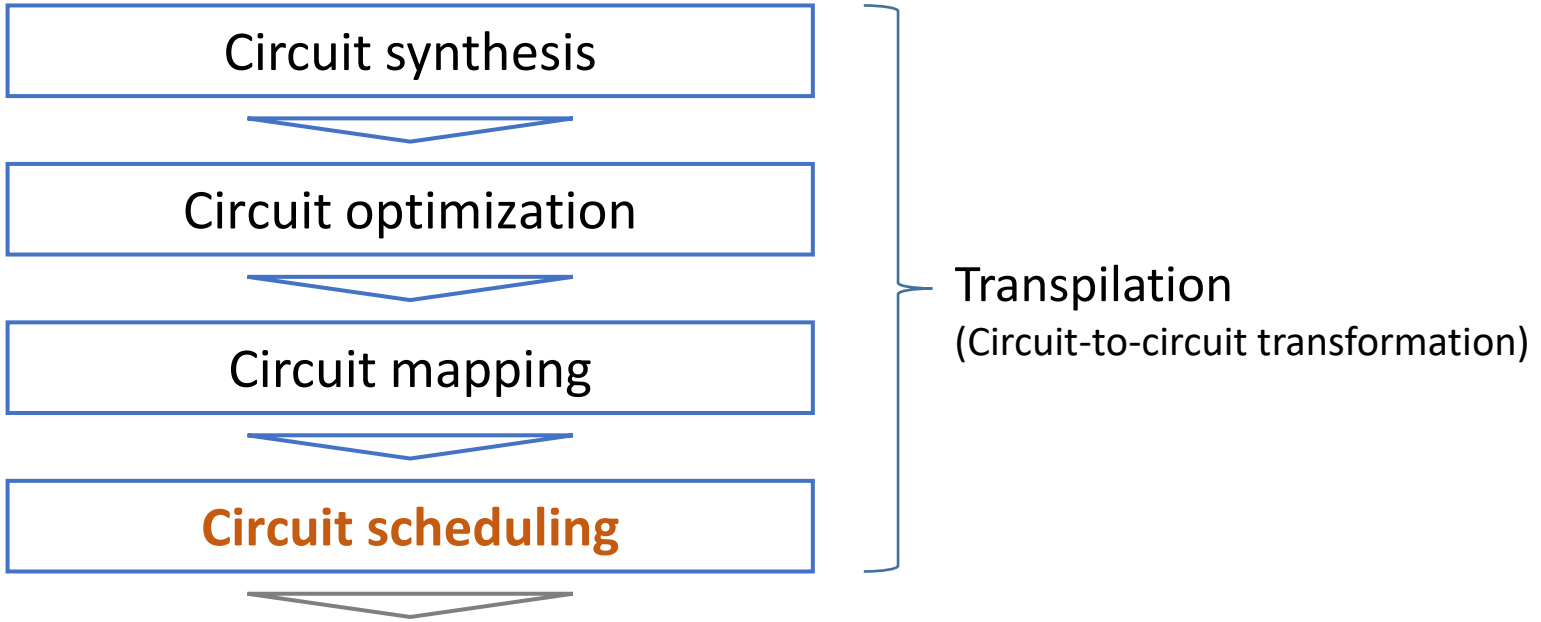
2. Solve separately

Qubit layout

Qubit routing

− Sub-optimal solution
+ Fast algorithm

Circuit scheduling

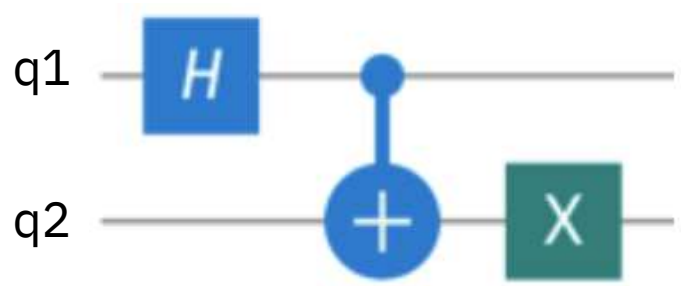


Circuit scheduling

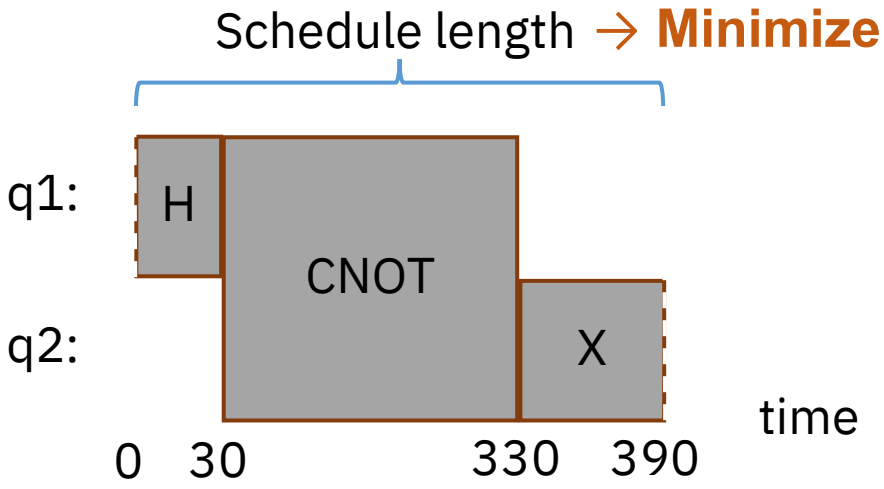
Given a mapped circuit, determine the start time for each gate in the circuit

IN: Circuit

(after circuit mapping)



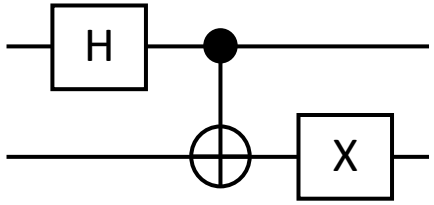
OUT: Scheduled circuit

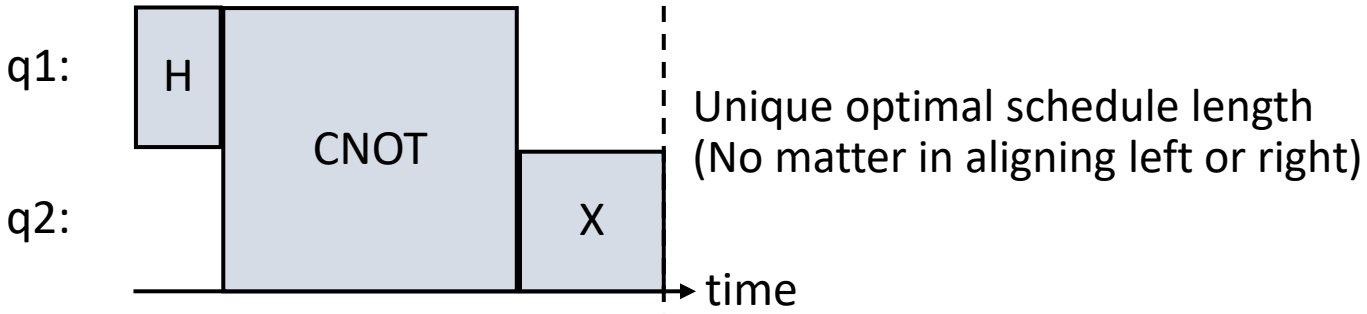


- Each gate have its own process time (may depend on qubits it applies to)
- No overlap between gates

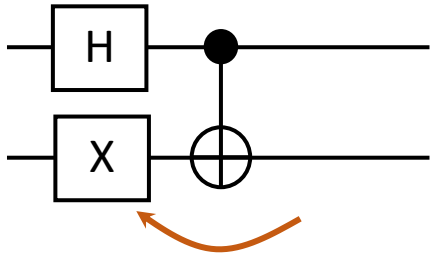
Circuit scheduling and gate commutation

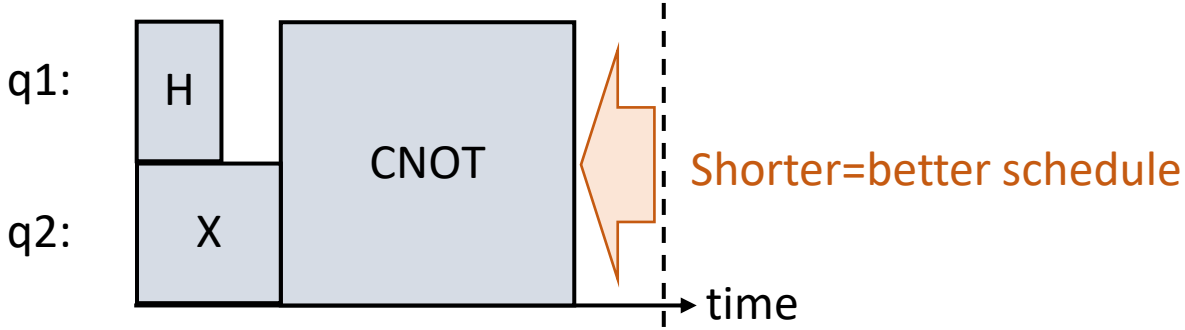
Trivial if not considering gate commutation:



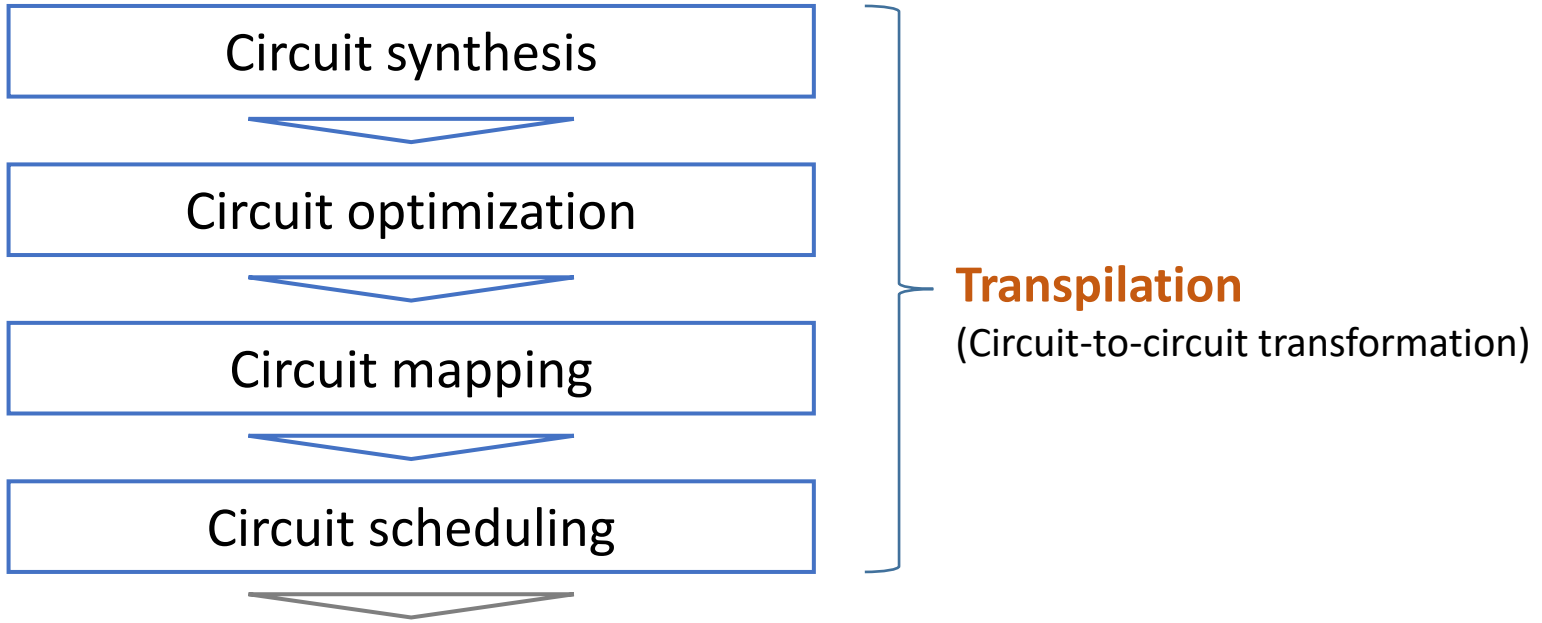


Room for optimization if considering gate commutation:





Notes on transpilation flow

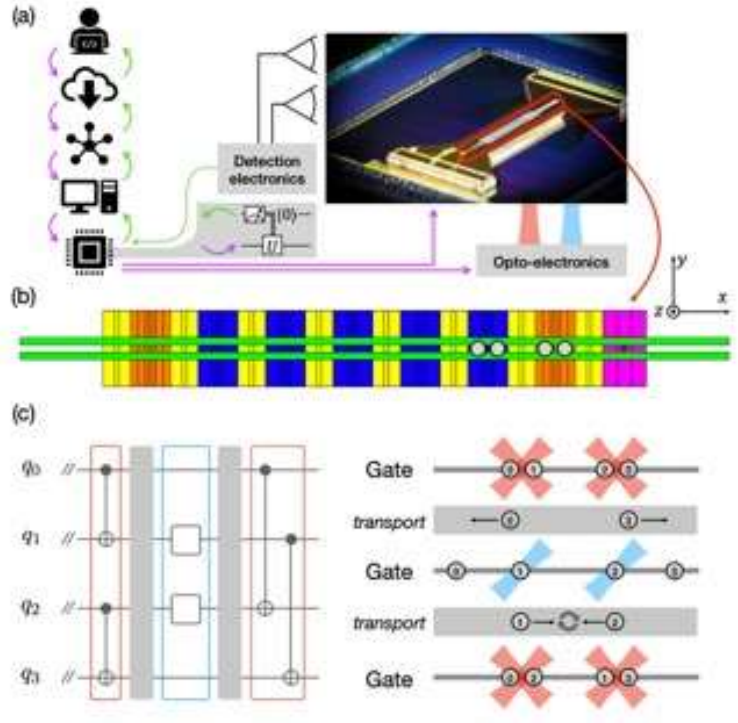


- Lower-level tasks depend more on the type of processors
- Objective functions of optimization tasks are arguable.
- Flow can be more complex (e.g. revisiting tasks, branching)

Note: Different circuit mapping/scheduling for different HW-types

Circuit mapping problem depends on the HW-type of quantum processors

Ex) Trapped-ion quantum computers



Pino, Juan M., et al. "Demonstration of the trapped-ion quantum CCD computer architecture." *Nature* 592.7853 (2021): 209-213.

Examples of quantum computer implementation:

- NMR (Nuclear magnetic resonance)
- Quantum dot
- Quantum optics
- Superconducting electric circuit (Focus in this lecture)
- Trapped ion

etc. https://en.wikipedia.org/wiki/Quantum_computing#Candidates_for_physical_realizations

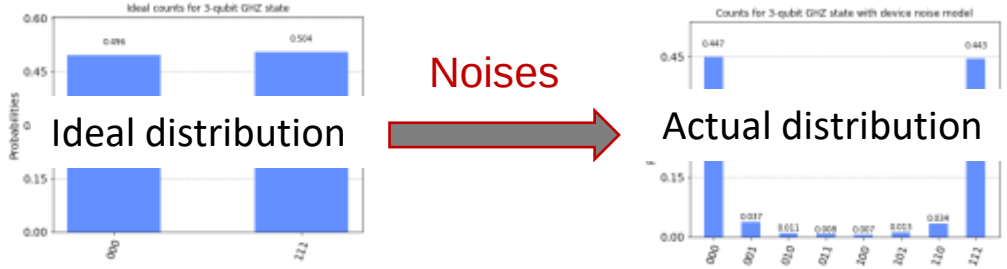
- 2q-gates are operable on any pair of qubits
- Need to “transport” the physical qubits to the “place” where the operation is applicable for each operation

Note: Objectives of circuit optimization

What is the “best” objective function for circuit optimization?

- Gate (CNOT) count
  - Assume CNOT gate error is dominant
- Depth/Schedule length
  - Assume decoherence of qubits is dominant
- Noise-adaptive cost
  - Based on a noise model (with device parameters) assuming it can predict the amount of noise well

- T-count/T-depth
  - T gate is considered the most expensive gate for fault-tolerant quantum computers



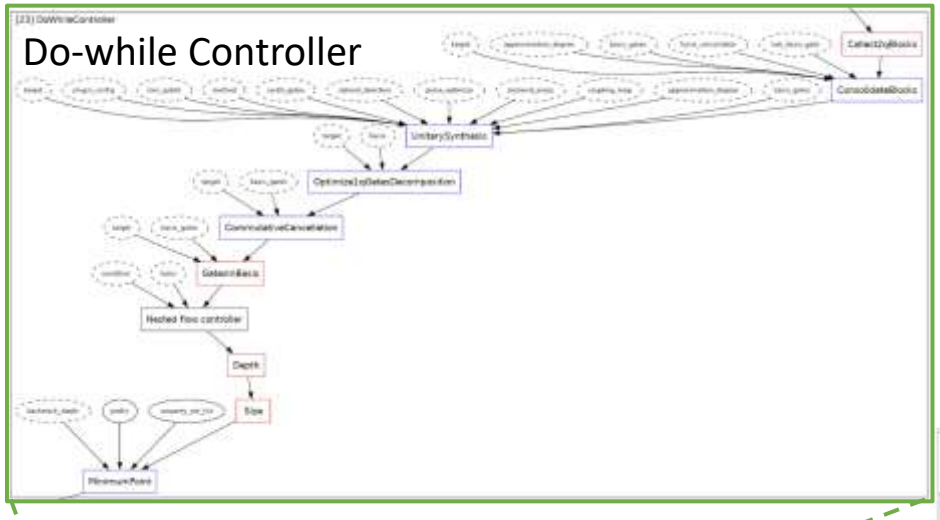
Possible errors:
- Gate errors (SX, CNOT), SPAM errors
- Cross-talk error
- Thermal relaxation error
- Qubit readout error

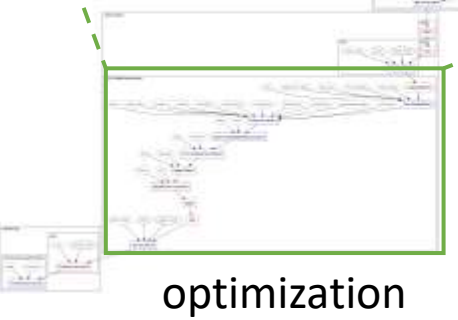
etc.

Note: Flow can be more complex (e.g. revisiting a task, branching)

Qiskit transpiler has six stages:

('init', 'layout', 'routing', 'translation', 'optimization', 'scheduling')





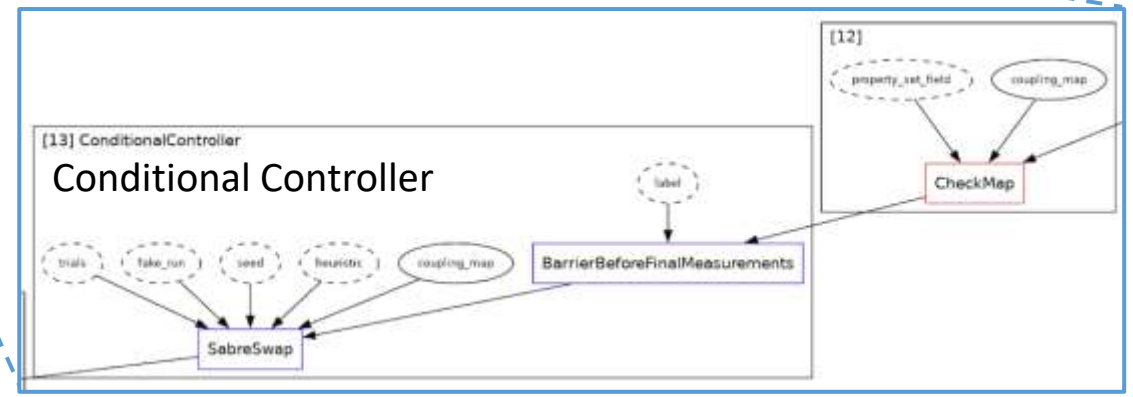
optimization

translation
(1q/2q-gate synthesis)

routing

layout

init
(>3q-gate synthesis)



Break

We have a hands-on session next.

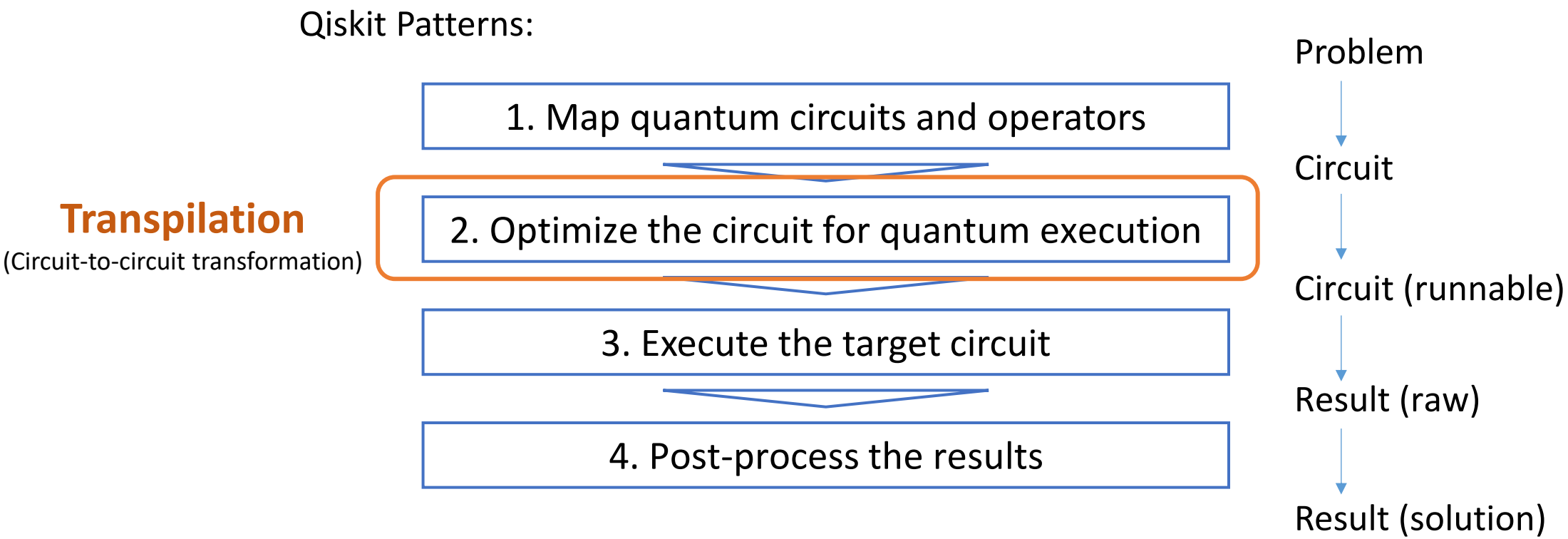
Please make sure to prepare your laptop.

Hands-on: Qiskit transpiler

2024621_UTokyo_qcopt.ipynb

Hands-on: Qiskit transpiler

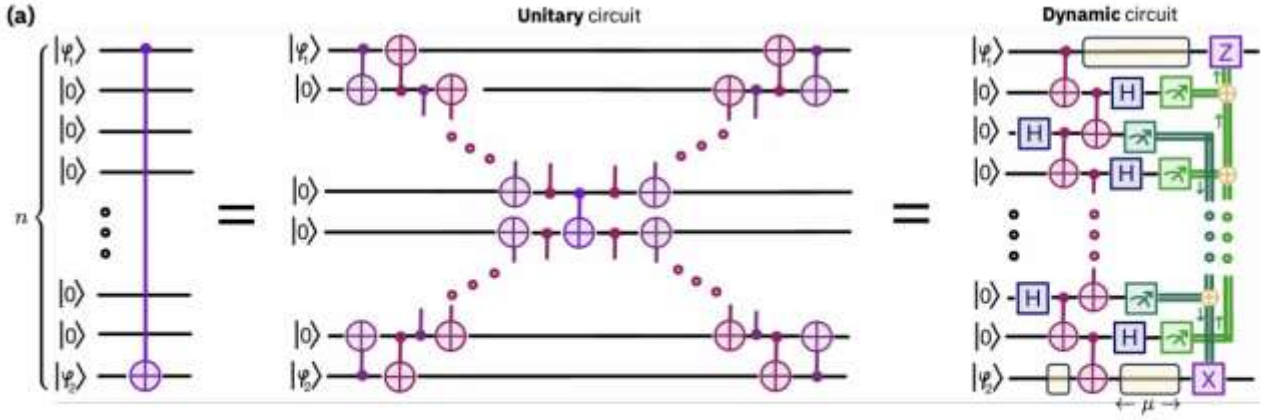
2024621_UTokyo_qcopt.ipynb



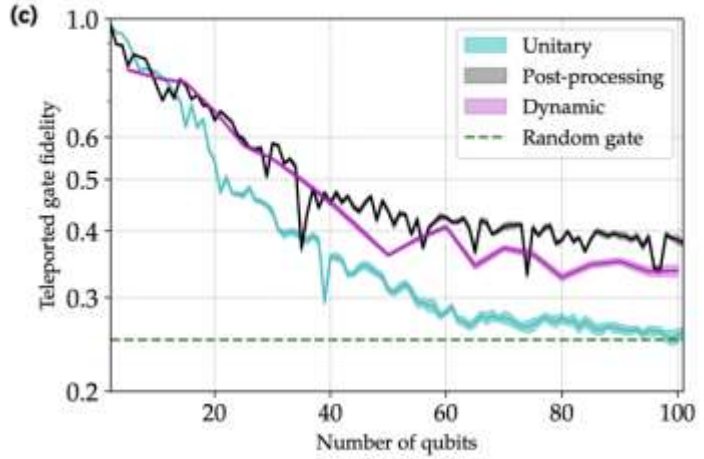
Measurement-based circuit synthesis (Dynamic circuits)

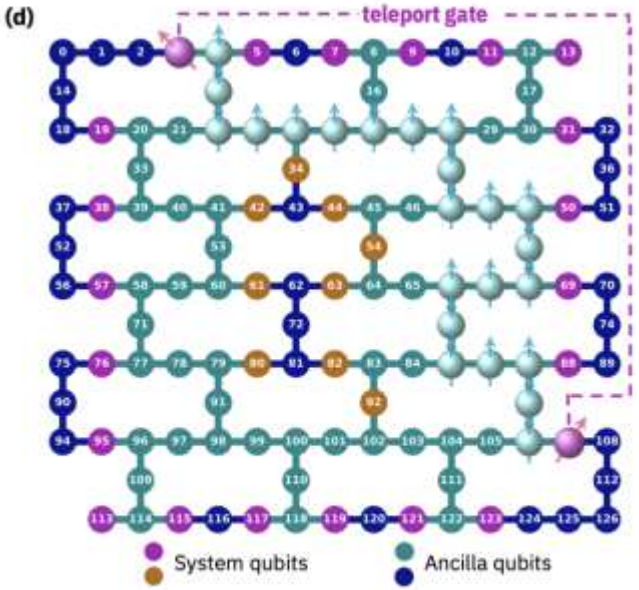
- For example, **CNOT gate teleportation** is possible by a dynamic circuit with just two layers of CNOT gates

- A promising approach if measurement operations are sufficiently fast and accurate



CNOT Circuit	Idle time error (per CNOT gate time)	CNOTs	Additional measurements
Unitary	0	$4n+1$	0
Post-processing	2	$n+1$	n
Dynamic	2μ+2	$n+1$	n

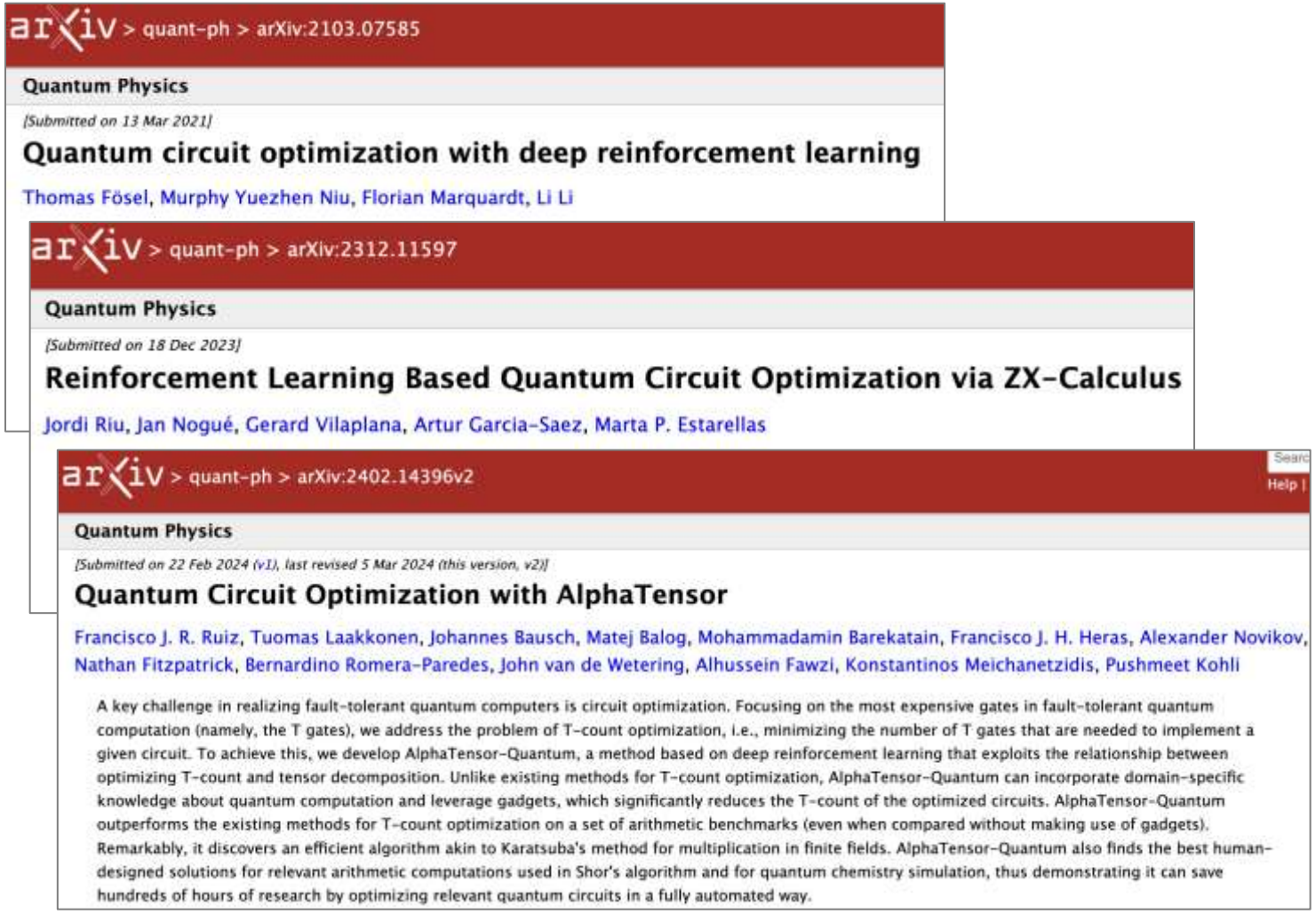




Bäumer, Elisa, et al. "Efficient long-range entanglement using dynamic circuits." *arXiv preprint arXiv:2308.13065* (2023).

AI-based circuit optimization

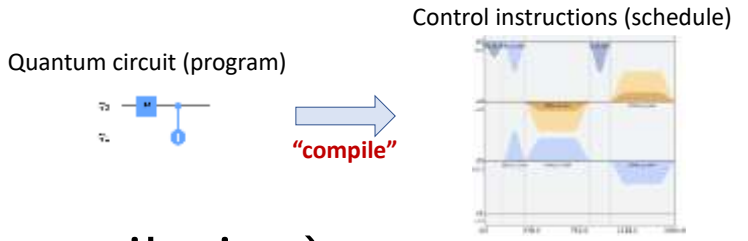
- Circuit optimization is a classical **search problem**
- **AI techniques** such as **reinforcement learning** and **MCTS** must work well



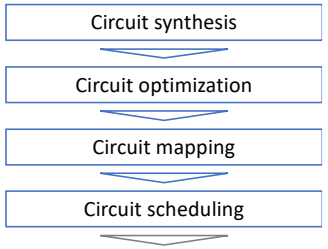
MCTS: Monte Carlo Tree Search

Summary

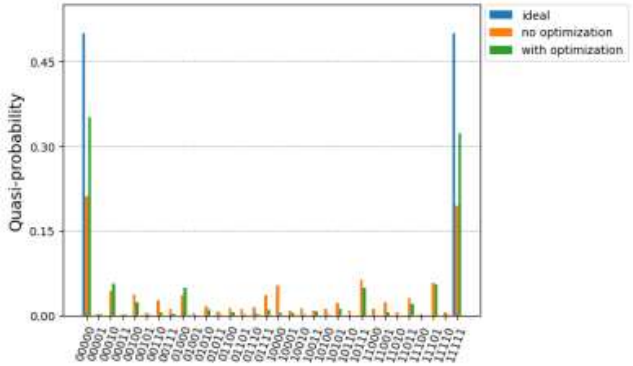
- Quantum compiler transforms a quantum circuit into control instructions
  - Satisfying constraints of a target quantum processor
  - Optimizing the resulting instructions



- Transpilation = circuit-to-circuit transformation (in compilation)
  - Circuit synthesis, optimization, mapping (qubit layout + routing), scheduling
  - All those tasks are essentially (classical) optimization problems
  - Transpilation ≈ Quantum circuit optimization



- Better transpiler → More accurate and faster quantum computation



Thank you