Transpiling, compiling and optimizing quantum circuits



A lecture series by IQM and HS RM

Authors: Stefan Seegerer, Jami Rönkkö, Ivan Takmakov and Nadia Milazzo

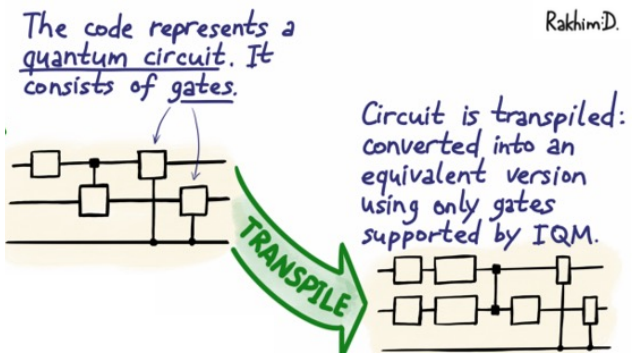
Last Updated 07/2025

www.meetiqm.com

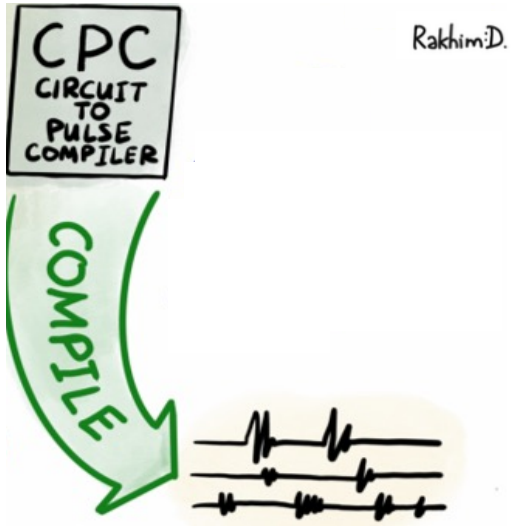
How do we run quantum circuits on real quantum computers?

Three steps

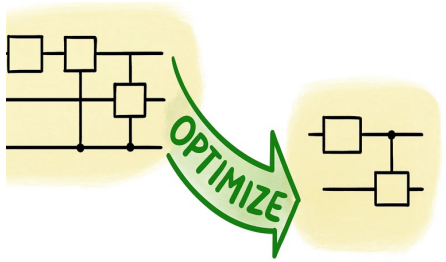
TRANSPILATION



COMPILATION

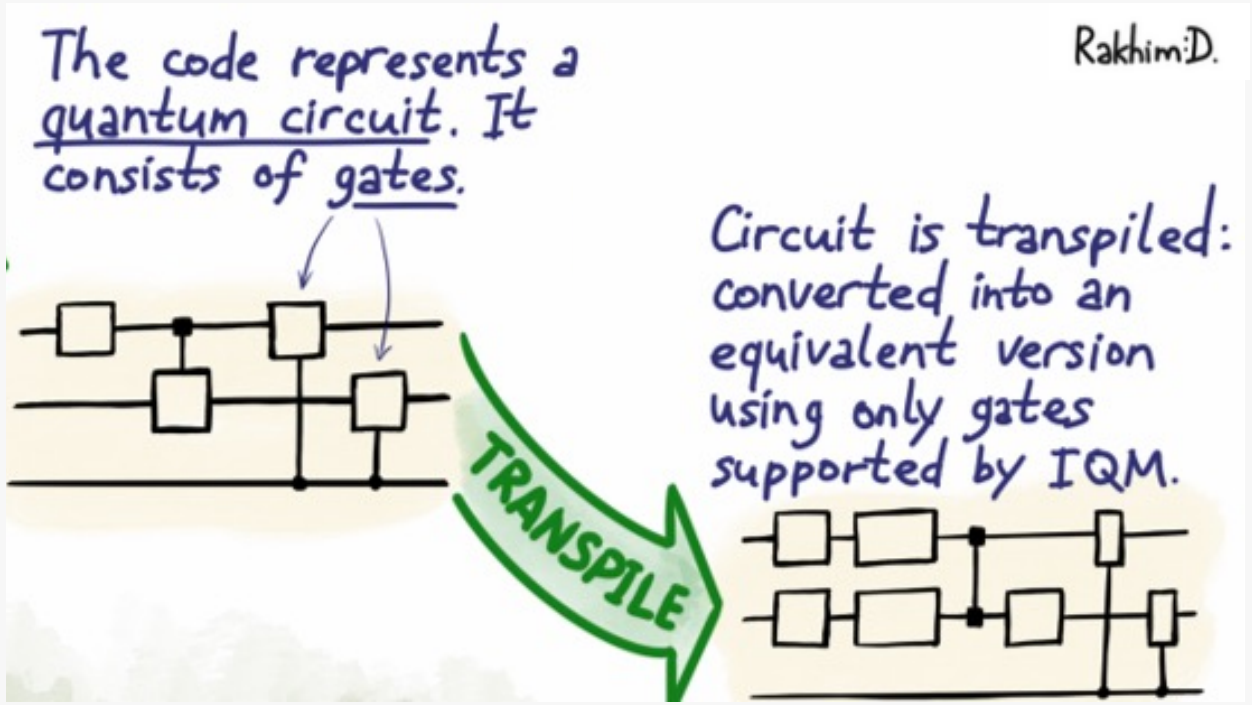


OPTIMIZATION



Transpiling = circuit routing + gate decomposition

Transpilation

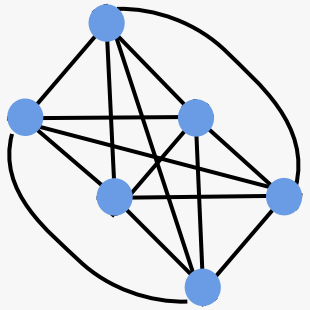


Transpiling describes the process of converting a quantum circuit into an equivalent quantum circuit that is compatible with a specific quantum hardware's constraints and gate set.

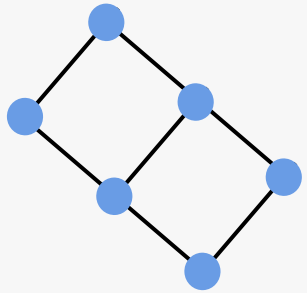
Circuit routing

Routing (= solving QPU connectivity constraints)

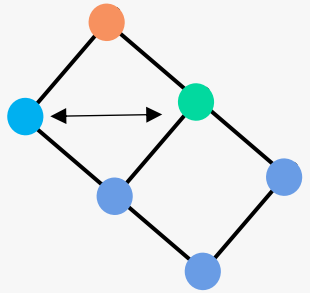
- Quantum algorithms assume arbitrary, even all-to-all connectivity
- Connectivities of QPU's are usually limited to nearest neighbor graphs
- Solution: dynamically remap **logical qubits** ("qubit 1" in the program") to the graph of **physical qubits** ("qubit 1" on the physical hardware) with SWAP gates



Algorithm's connectivity



QPU's connectivity

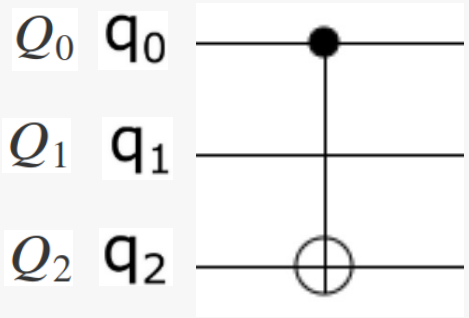


Dynamical remapping

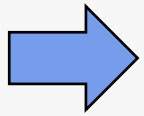
Routing example

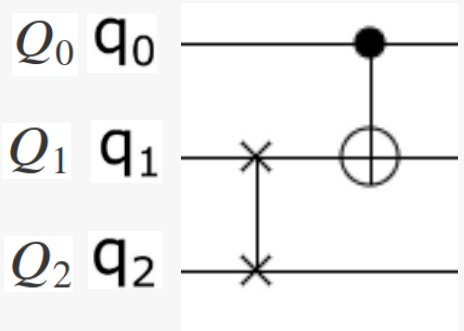
Consider the following connectivity:





Routing incompatible operation



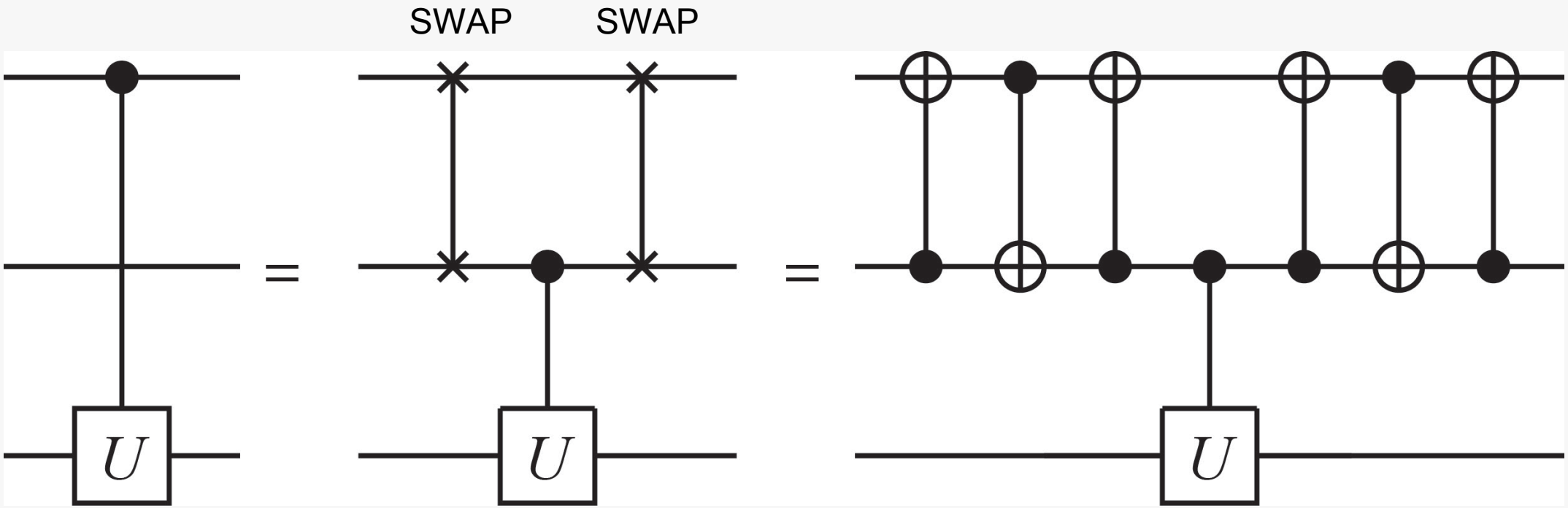


Routing compatible operation

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

SWAP gate swaps the states of two qubits

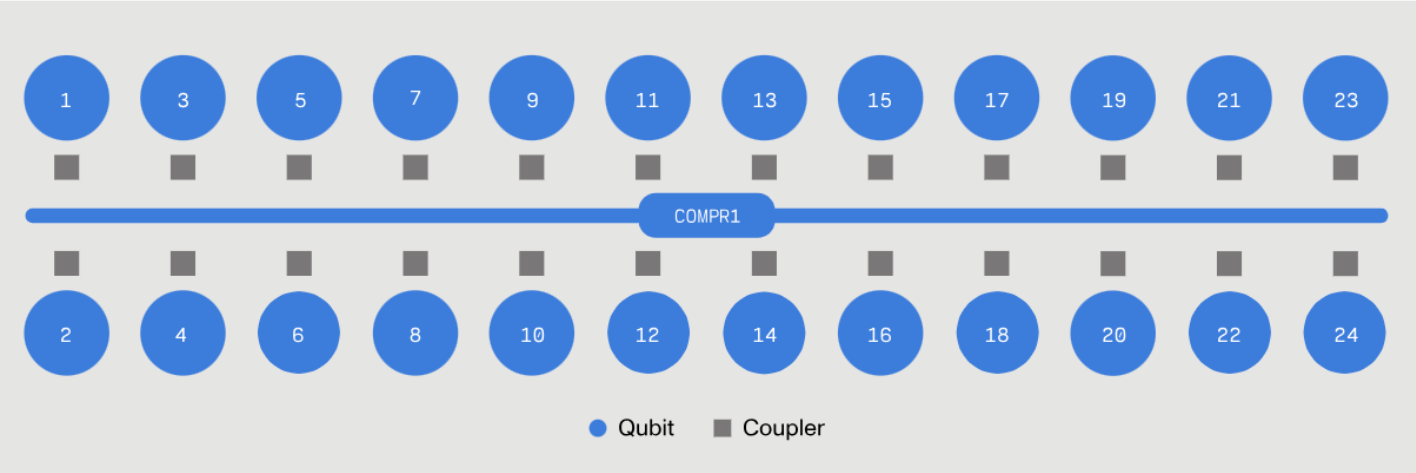
SWAP is expensive!



The MOVE operation

IQM QPUs come in two topologies: Crystal and Star.

In the Star topology the qubits don't connect to each other. They all connect to one central **resonator.**



The qubits interact with the central resonator through two types of operations:

- CZ gate between qubit and the resonator
- The MOVE operation, which shuttles a qubit state between a qubit and the resonator.

$$MOVE\ |01\rangle = |10\rangle$$
$$MOVE\ |10\rangle = |01\rangle$$

Similar to SWAP

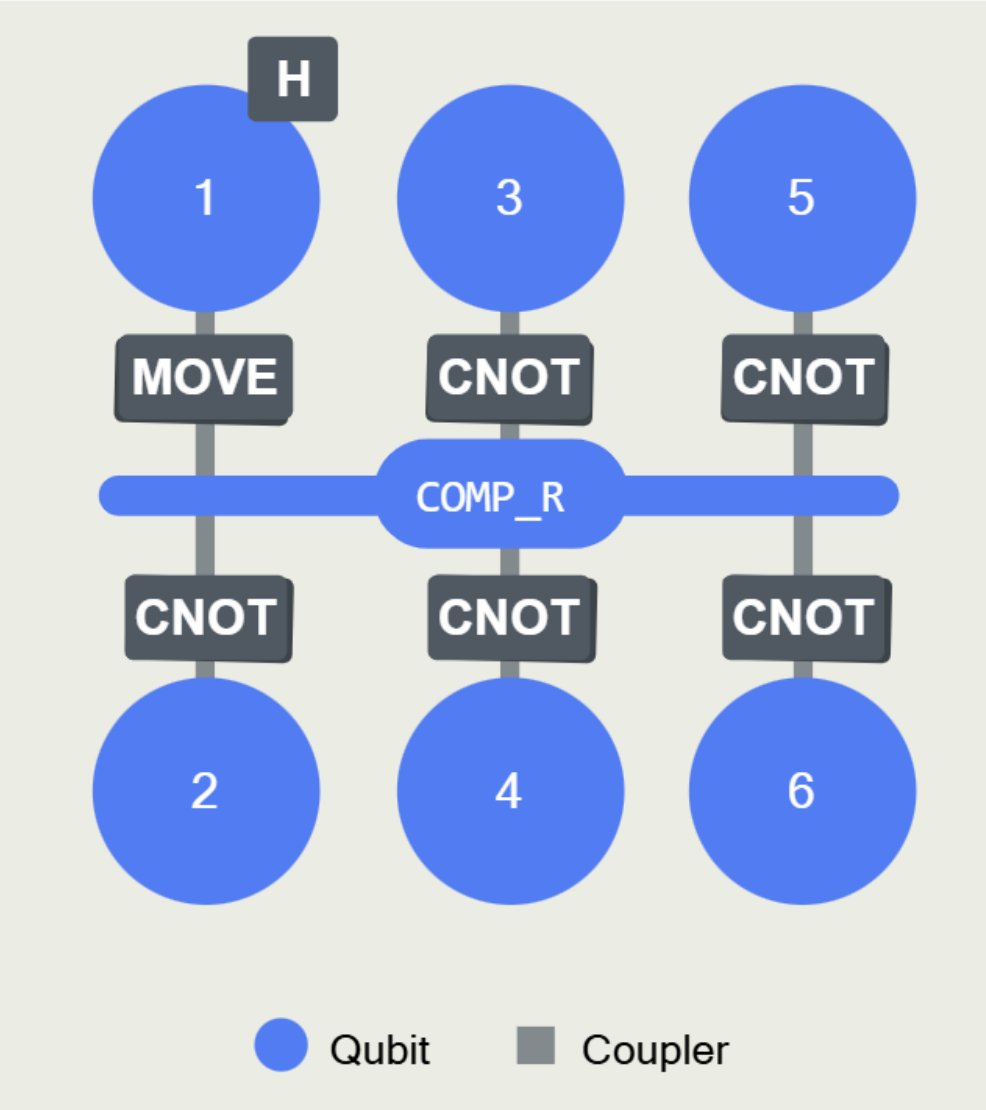
Exercise: Explore the MOVE operation

Understand better how the MOVE operation acts using the tutorial below:

Link or QR code:

You will learn:

- How to create a Bell state with this device using the MOVE gate
- How the MOVE operation behaves when the resonator is populated



Gate decomposition

Implementing gates on a QPU

- We need to be able to implement *any* unitary operation on our quantum computer. But not all operations are simple on all hardware.
- The set of simple gates used on a *specific* quantum device is called the native gate set of the device.

IQM systems support arbitrary X and Y rotations as the native single-qubit gates and CZ as the native two-qubit gate.

Exercise

- Imagine your QPU's only native gates are X gates on each qubit. Find a gate you can't implement on your QPU.
- Repeat the above if your native gate set also includes Y gates on each qubit.
- Repeat the above if your native gate set includes arbitrary rotations about the x and y axes on each qubit.

Universal gate set

- We need to be able to implement *any* unitary operation on our quantum computer. But we need enough native gates to do that.

- Our native gate set should be universal. What does that mean?

A set of gates S is said to be universal if **any** unitary operation can be approximated to any desired precision by composing only the gates in the set S.

Universal gate set

A universal set of gates for quantum computing can:

- Create superposition of states
- Create entangled states
- Contains gates with real AND complex entries
- Contains more than the "Clifford group"



Special quantum circuits which can be efficiently simulated by a classical computer. To read more: "Gottesman-Knill theorem"

Universal gate set

Is there only one universal set of gates? If so, we'd be forced to use hardware with a very specific set of native gates…

No, there exist different universal gate sets, such as:

$\{CNOT, H, S, T\}$

$S =$ *phase gate*
$T = \pi/8$ *gate*

$\{CZ, R_x(\theta), R_y(\phi)\}$

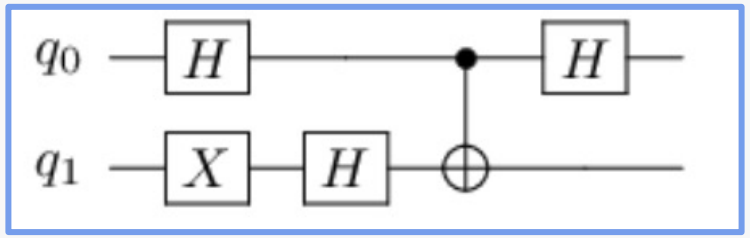
$\{H, TOFF\}$

$TOFF =$ "*Toffoli gate*"

Decomposing (= writing a gate a series of native gates)

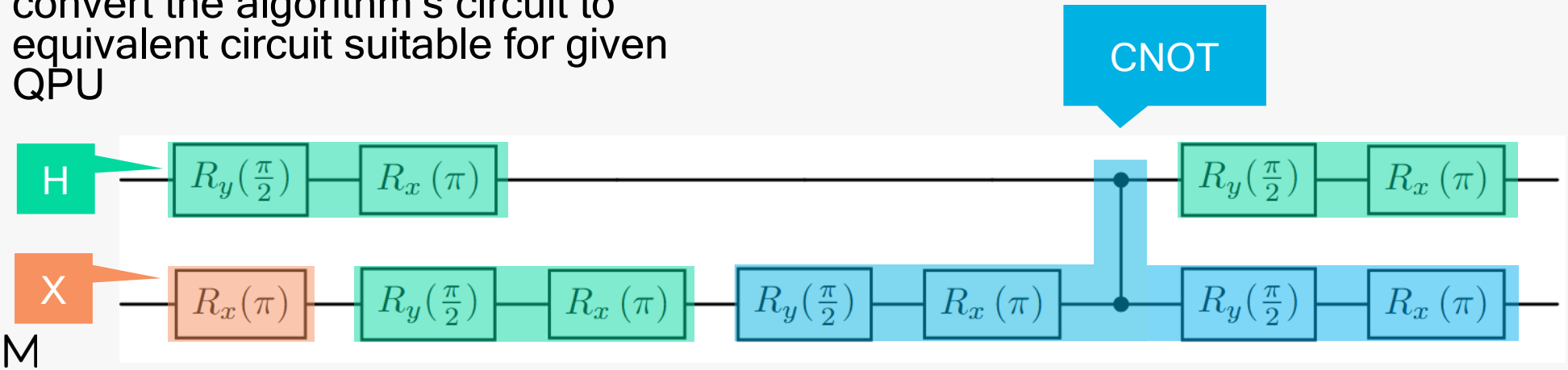
- Circuits of quantum algorithms usually use 'intuitive' gates like CNOT, X, H, P, SWAP
- Real QPU's use only few *native,* hardware friendly gates to process their qubits
- Solution: gate identities can convert the algorithm's circuit to equivalent circuit suitable for given QPU

Circuit for Deutsch's algorithm with CNOT as the oracle



Example QPU native gates

$\{R_x(\theta), R_y(\phi), CZ\}$

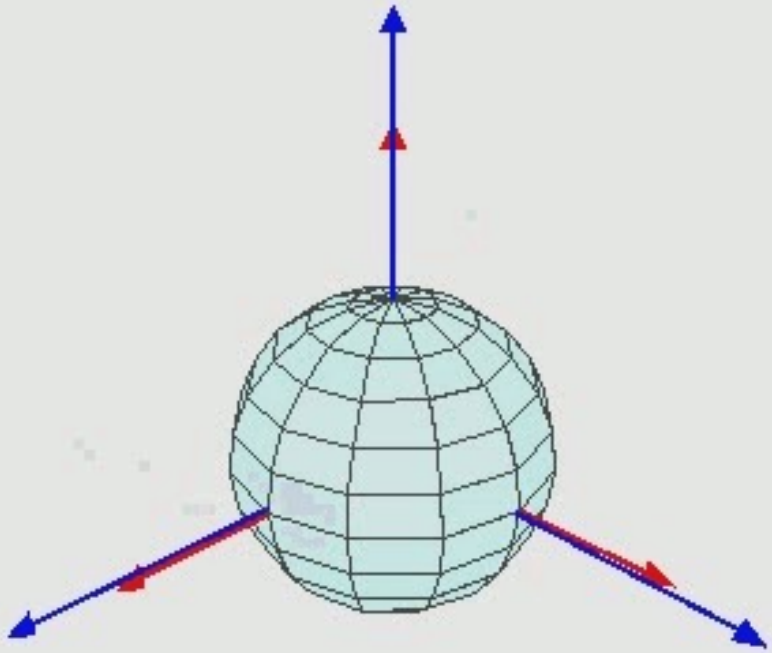


How do we decompose single qubit gates?

Euler angle decomposition

- State of a single qubit can be described as a vector in a Bloch sphere and gates as rotations of this vector
- Any rotation can be achieved with three elementary rotations, whose angles are the called Euler angles
  - We pick three axes and do a rotation around each one.
  - Not all choices of axes work (obviously picking them all to be z doesn't work!).
- IQM hardware has native R_x and R_y gates, so it uses x-y-x (or y-x-y) as the axes.

➡ IQM hardware decomposes any single-qubit gate into a product of an R_x rotation, an R_y rotation, and a second R_x rotation



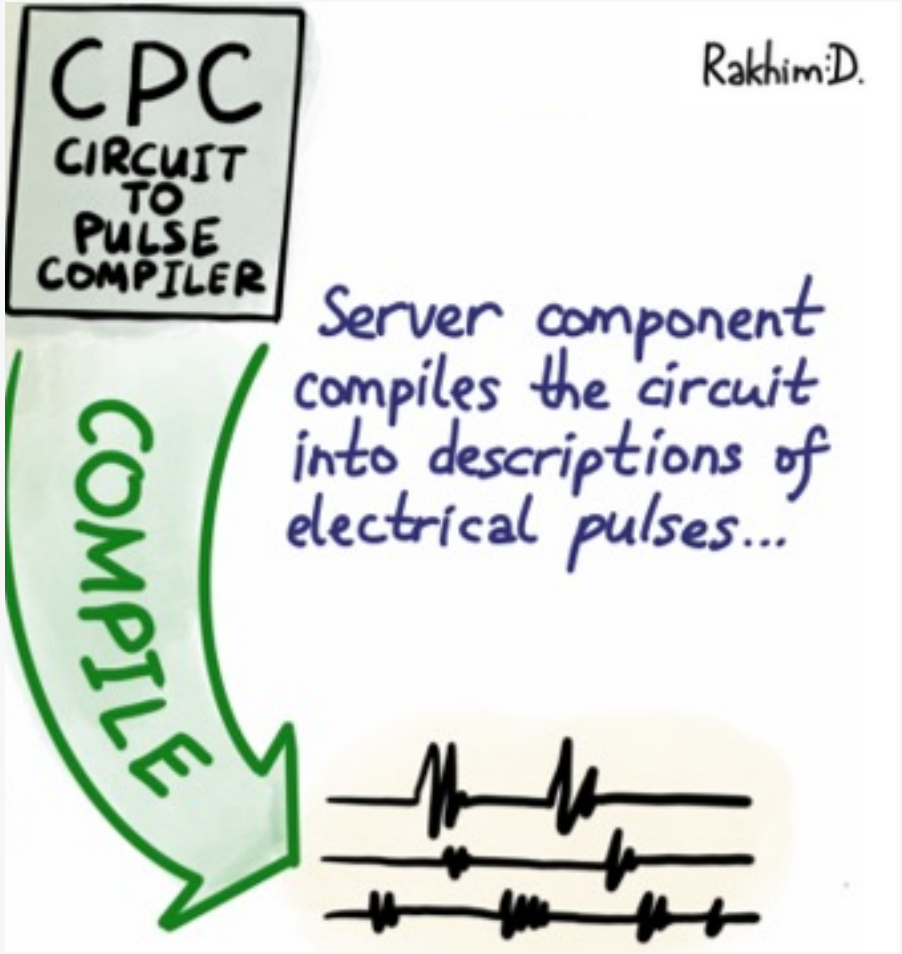
z-x-z Euler sequence
Image source: https://https://en.wikipedia.org/wiki/Euler_angles

Redefining more precisely…

Transpiling = routing circuit to connectivity of a QPU and decomposing gates to its native gates

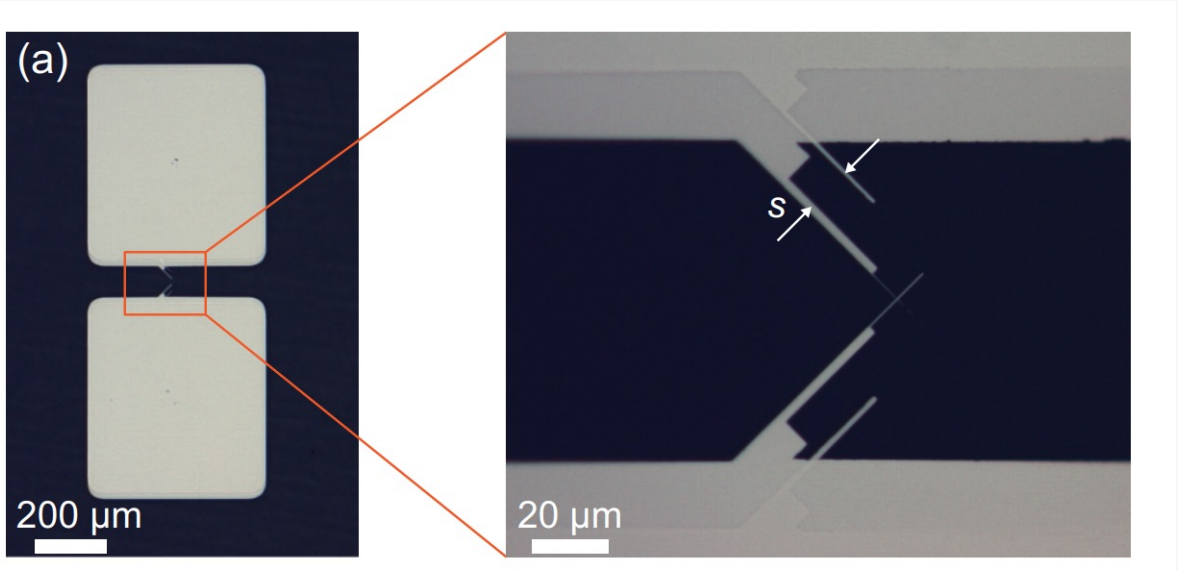
Compiling = pulse schedule representation

Compilation

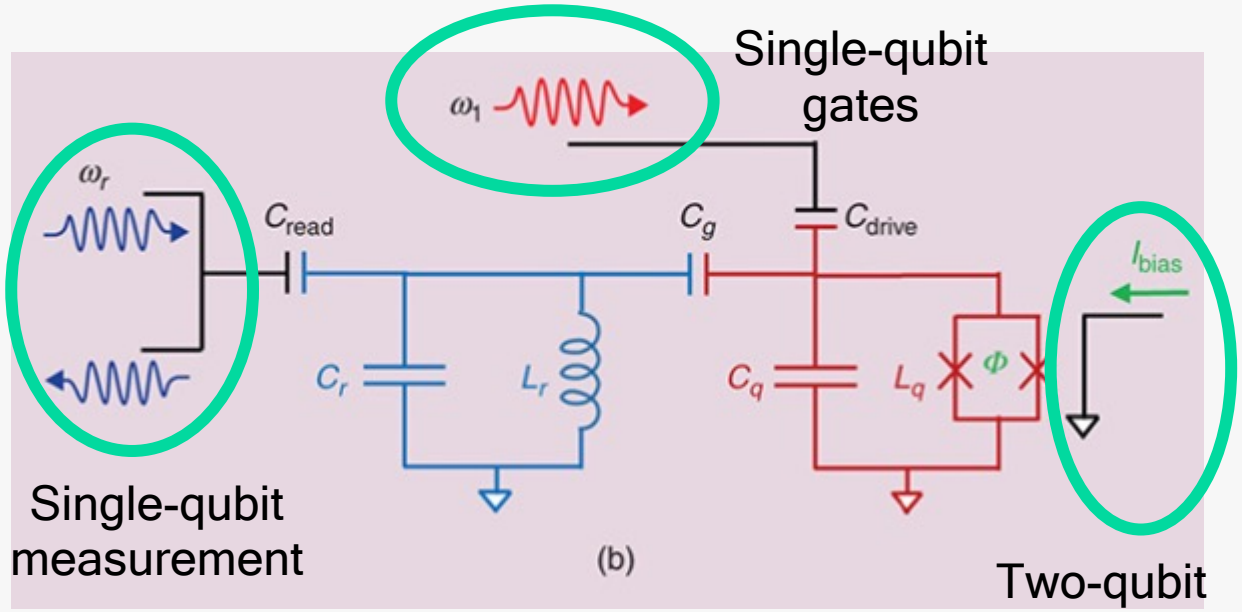


A **quantum compiler** translates high-level quantum programs into operation that a quantum processor can execute at the lower level of the **physical** qubits.

Control of superconducting transmon qubits



Tsioutsios I., Serniak K., et al (2019)

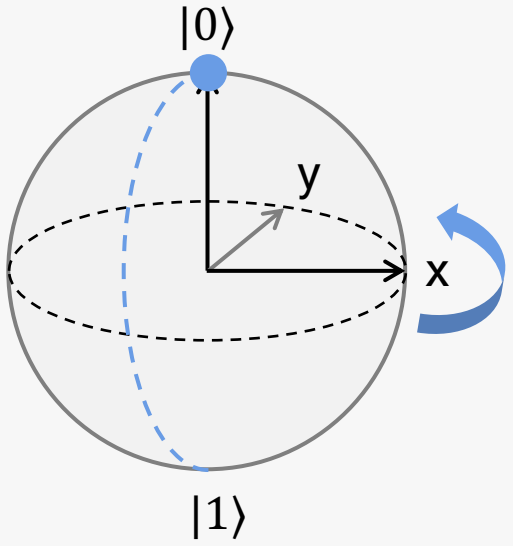


Roth, Ma, Chew (2023)

Main idea: by tuning AC voltage pulses that we send into the device, we can manipulate and read out the qubits.

Single qubit gates

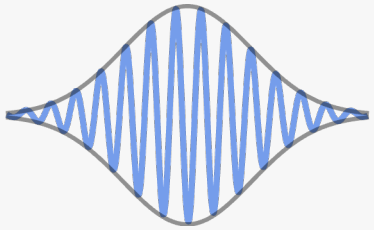
- Applying AC drive makes qubit state rotate around an axis in xy plane
- **Example:** X gate (π pulse) is an analog of the classical NOT gate, it flips qubit state



$$H(t) = -\frac{1}{2}\Omega(t)(\cos(\phi)\,X + \sin(\phi)\,Y)$$

Strength and shape of pulse Ω controls how much rotation

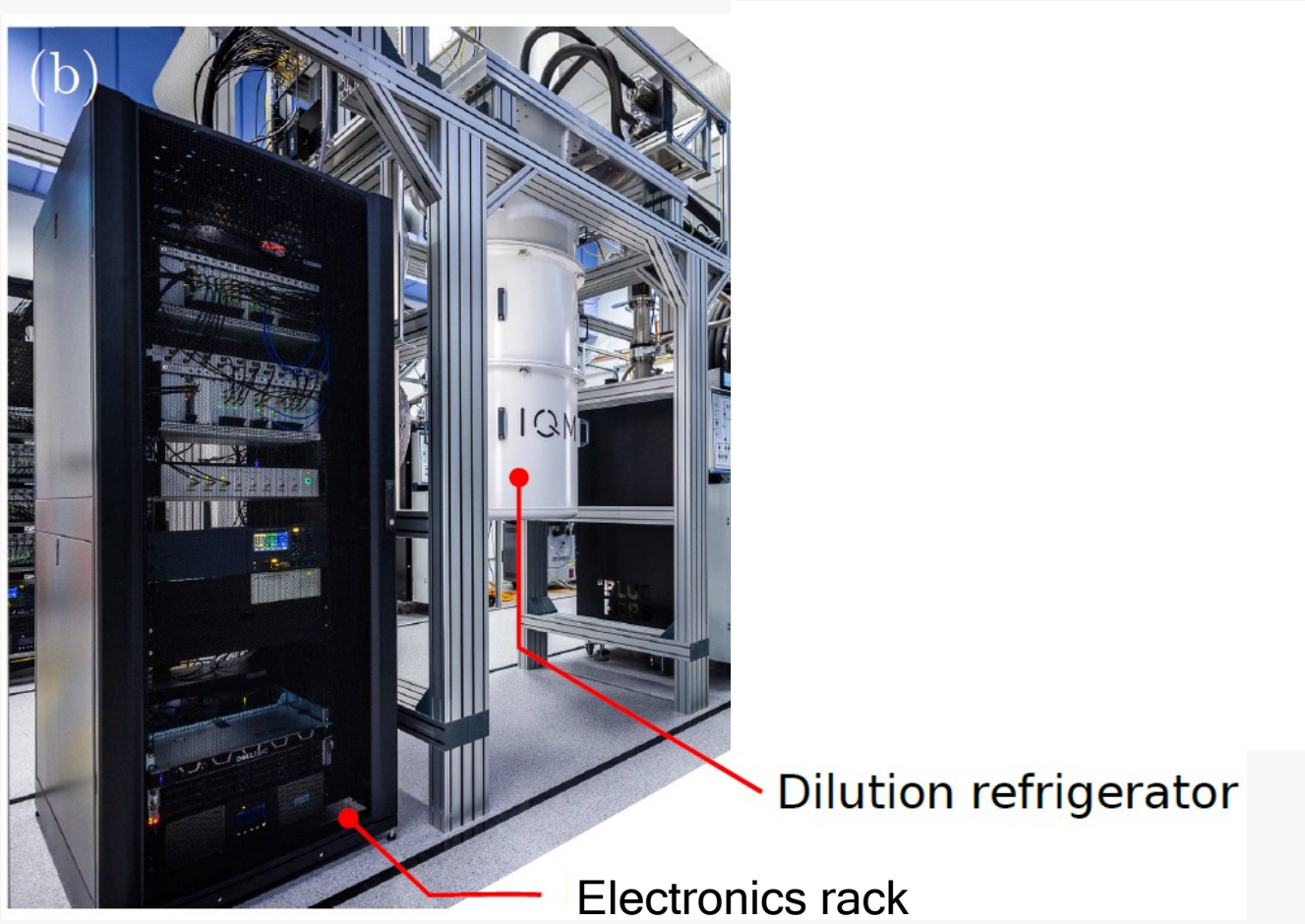
Phase φ controls the axis of rotation:
φ = 0 → X-axis
φ = π/2 → Y-axis



We can realize any single qubit X or Y rotation by applying a correctly chosen AC pulse. These are universal for single-qubit gates!

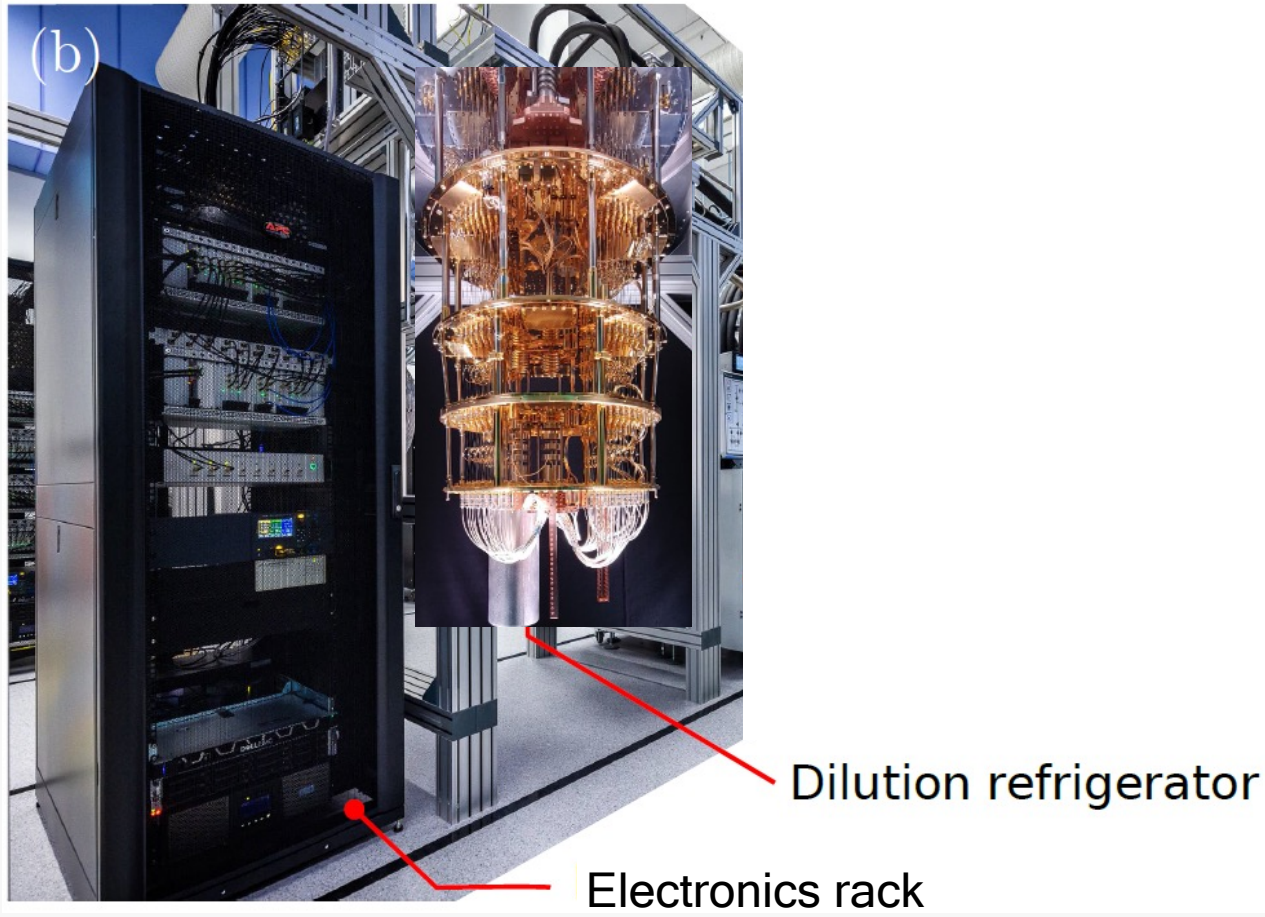
When you want to run a quantum circuit:

- Your request is compiled by a server into a sequence of pulses



When you want to run a quantum circuit:

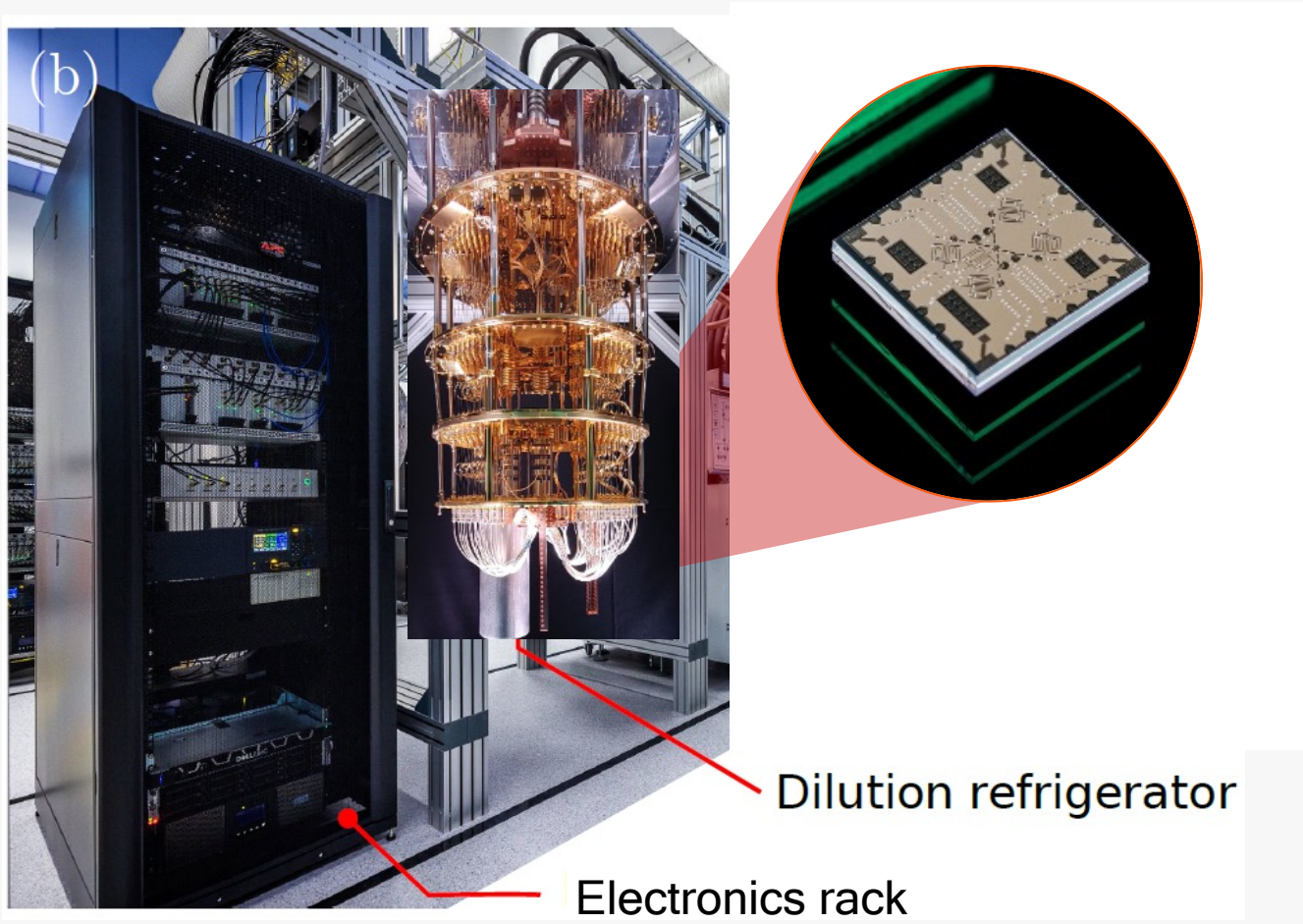
- Your request is compiled by a server into a sequence of pulses
- Control and readout pulses are generated in the electronics rack and sent into the refrigerator



When you want to run a quantum circuit:

- Your request is compiled by a server into a sequence of pulses
- Control and readout pulses are generated in the electronics rack and sent into the refrigerator
- The pulses interact with the QPU

After that, the measurement device “reads” the results, and you can analyze them on your PC.



Standard stages of compilation in IQM devices

CIRCUIT VALIDATION

If no custom implementation is specified, the default implementation for each gate are selected automatically.

CIRCUIT RESOLUTION

The circuit is converted to *Timeboxes:* containers for instruction schedule fragments, to be scheduled according to a given scheduling strategy.

TIMEBOX LEVEL

The timeboxes corresponding to the measurement instructions in the circuit are added.

TIMEBOX RESOLUTION

All relative timings between pulses are resolved and fixed. The different timeboxes are assembled in a single schedule.

Standard stages of compilation in IQM devices

After these main compilation stages, before executing the circuit, one can do a couple of extra passes if wanted/needed:

DYNAMICAL DECOUPLING

When the circuit calls for a qubit to do nothing, special extra pulses are inserted in order to **suppress errors.**

MOVE GATES

Using MOVE gates causes special, well-understood errors, which are corrected in this pass. (Only applicable to QCs with computational resonator, i.e. IQM Star Architecture).

Optimization = reduce noise

Optimization

To get the best results when running circuits on quantum computers, some improvements can be made in the transpilation and compilation process.

1. Optimize single qubit gates and reduce their count:

1. Convert single-qubit gates to U^* gates and combine all neighboring U gates.
2. Convert U gates according to $U(\theta, \phi, \lambda) = RZ(\phi + \lambda)R(\theta, \pi/2 - \lambda)$.
3. Commute *RZ* gates to the end of the circuit using the fact that *RZ* and *CZ* gates commute, and $R(\theta, \phi)RZ(\lambda) = RZ(\lambda)R(\theta, \phi - \lambda)$.
4. Drop *RZ* gates immediately before measurements, and otherwise replace them according to $RZ(\lambda) = R(\pi, \lambda/2)R(-\pi, 0)$.

**Generic single-qubit rotation gate with 3 Euler angles.*

Optimization

To get the best results when running circuits on quantum computers, some improvements can be made in the transpilation and compilation process.

1. Optimize single qubit gates and reduce their count:

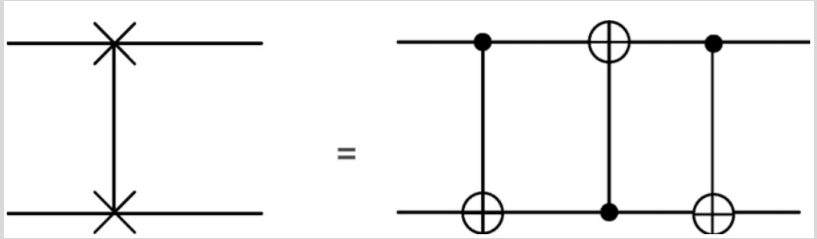
Put an exercise where we transpile a circuit with and without this function

Optimization

To get the best results when running circuits on quantum computers, some improvements can be made in the transpilation and compilation process.

2. Reduce the count of SWAP gates

- **Two-qubit gates** are the **noisiest gates**. Using fewer is even more impactful than optimizing single qubit gates.
- Carefully optimizing routing is critical for this, since circuit routing is done with SWAP gates.
  - Each SWAP gate translates into **three** CNOTs!



Remember that CNOT is also not a native gate for IQM computers!

Optimization

To get the best results when running circuits on quantum computers, some improvements can be made in the transpilation and compilation process.

2. Reduce the count of SWAP gates

Put an exercise where we transpile a circuit with and without choosing the best layout for 2q gates. Show also SWAP decomposition in terms of native IQM gates.

Conclusions

Conclusions

- Running circuits on **real** quantum computers needs extra steps that we usually ignore when writing high-level quantum programs.

- Transpilation adapts the circuit to:
  - The connectivity of the QPU
  - The (universal) native gate set of the QPU.

- Compilation is hardware dependent. For superconducting qubits, we need to translate all circuits into a sequence of microwave pulses.

- Running circuits on noisy quantum computers can be improved by:
  - Suppressing errors with extra pulses or good qubit choices
  - Optimizing the gates in the circuit.