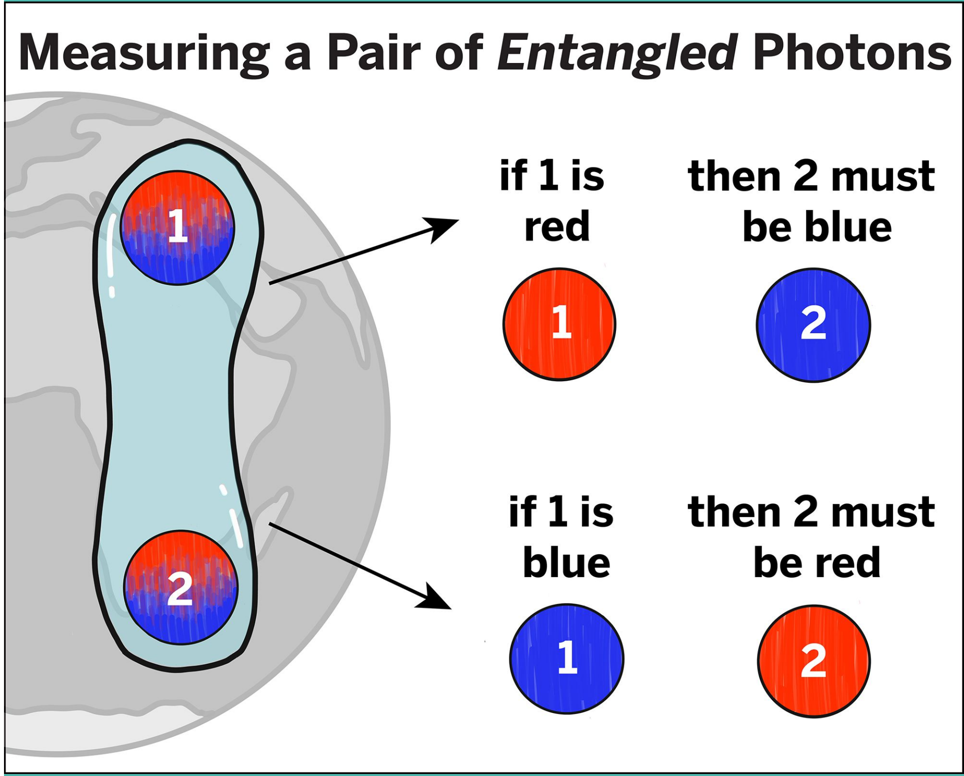
qLearn Week 4: Multi-Qubit Quantum Gates & Entanglement

Michael Silver, ECE2T6
University of Toronto Quantum Computing Club

Last Week Recap

The Concept of Quantum Entanglement



(Short) Math Recap

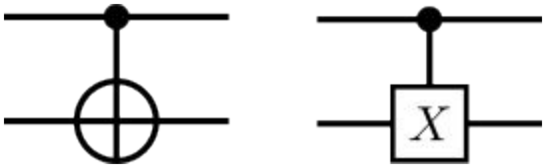
For quantum states $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} \otimes \begin{pmatrix}\gamma \\ \delta\end{pmatrix}$$

$$= \begin{pmatrix}\alpha\binom{\gamma}{\delta} \\ \beta\binom{\gamma}{\delta}\end{pmatrix} = \begin{pmatrix}\alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta\end{pmatrix}$$

- Tensor Product: operation that combines vector spaces into a new, larger vector space
- We can use it to describe the combinations of multiple quantum systems
- What does this all mean???

CNOT (Controlled Not) Gate

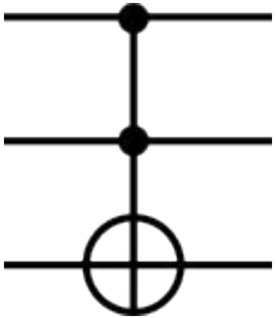


$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- Simplest multi-qubit Gate
- Conditional operation: conditionally flips the target qubit based on state of the control qubit
- CNOT Truth Table:

$$|00\rangle \longrightarrow |00\rangle$$
$$|01\rangle \longrightarrow |00\rangle$$
$$|10\rangle \longrightarrow |11\rangle$$
$$|11\rangle \longrightarrow |10\rangle$$

Toffoli (CCNOT) Gate



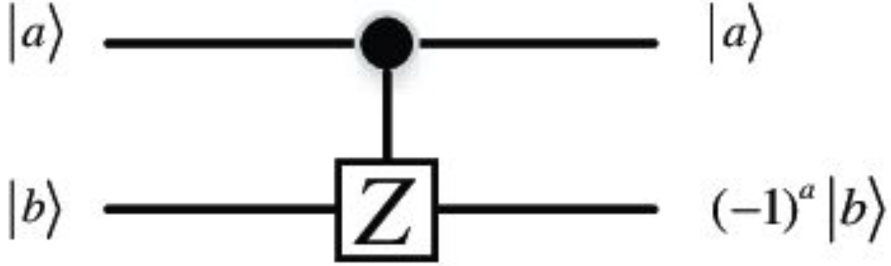
$$\text{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- Controlled-Controlled-NOT
- Conditional operation: conditionally flips the target qubit based on state of the control qubits
- Toffoli Truth Table:

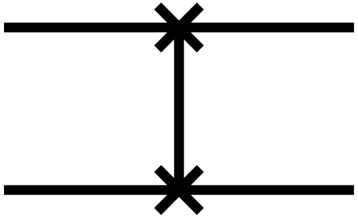
$$|00K\rangle \rightarrow |00K\rangle$$
$$|01K\rangle \rightarrow |01K\rangle$$
$$|10K\rangle \rightarrow |10K\rangle$$
$$\cdots$$
$$|11K\rangle \rightarrow |11\overline{K}\rangle$$

More Controlled Gates

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$



SWAP Gate



- ‘Swaps’ the state of two qubits

$$SWAP(|00\rangle) = |00\rangle$$
$$SWAP(|01\rangle) = |10\rangle$$
$$SWAP(|10\rangle) = |01\rangle$$
$$SWAP(|11\rangle) = |11\rangle$$

Challenge: Find the matrix of the SWAP gate

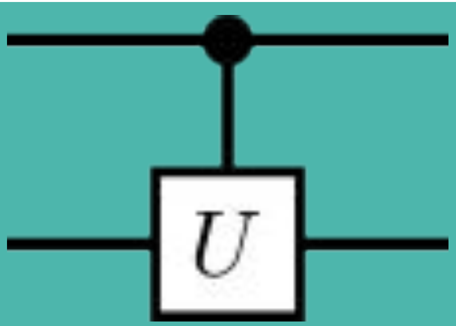
Hint: what basis states are affected?

$$|01\rangle \text{ and } |10\rangle \text{ are the affected states}$$

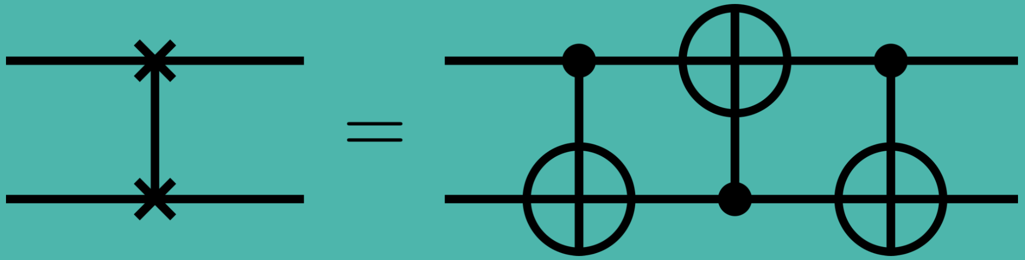
$$\text{SWAP swaps said states}$$

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

Equivalent Gates in Multi-Qubit Systems



$$CU = \begin{pmatrix} I_2 & 0 \\ 0 & U \end{pmatrix}$$

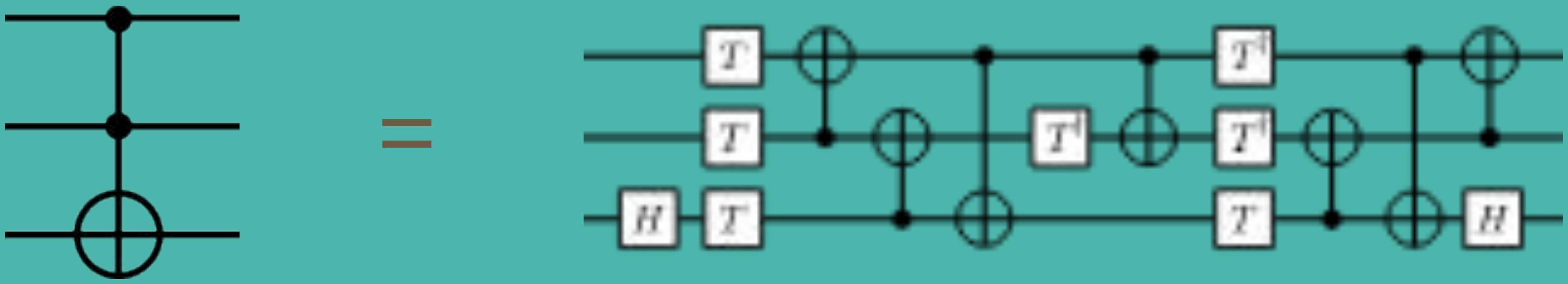


Universal Gates in Multi-Qubit Systems

$$\{H, T, CNOT\} \longrightarrow \text{common}$$

$$\text{Clifford + T: } \{H, S, CNOT, T\}$$

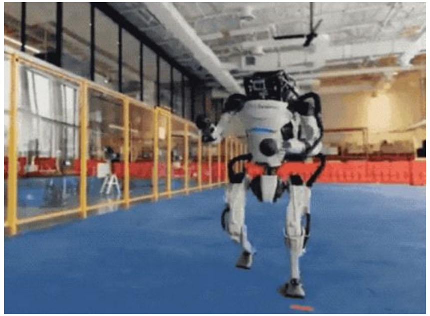
$$\{R_z(\theta), R_x(\theta), CNOT\} \longrightarrow \text{hardware}$$



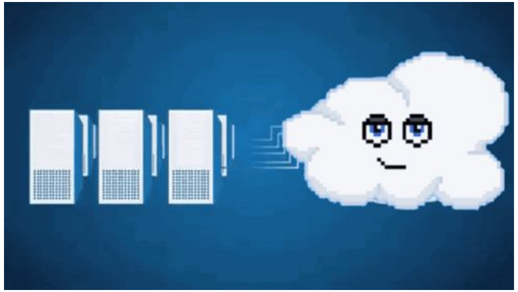
A Full-Stack Approach to Quantum Computing

Running an Algorithm on a Quantum Computer

End-user writes their 'quantum code' using a quantum SDK (Qiskit, PennyLane, etc)

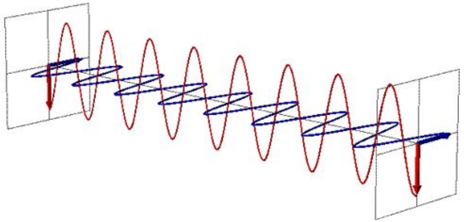


Code is automatically 'transpiled', ie. mapped to specific hardware



Transpiled code is sent through the cloud to a QC's classical control computers

Code is compiled into machine-level instructions (pulse level)



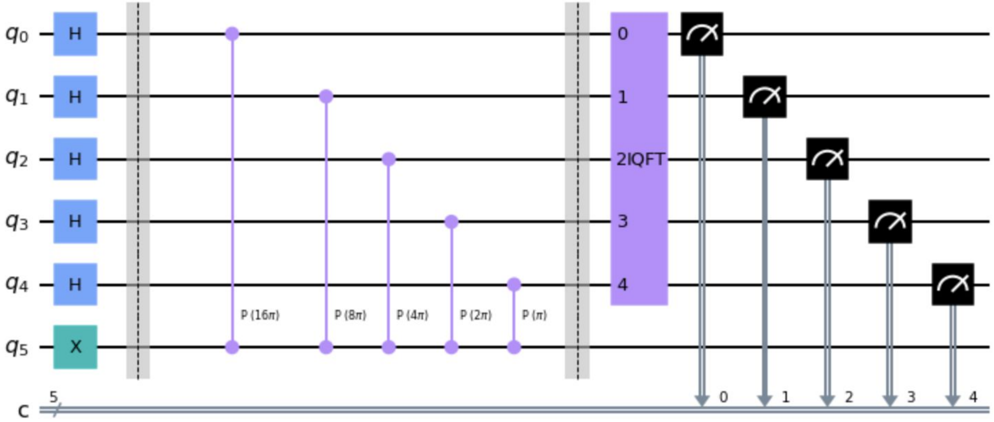
Code is executed on the QC

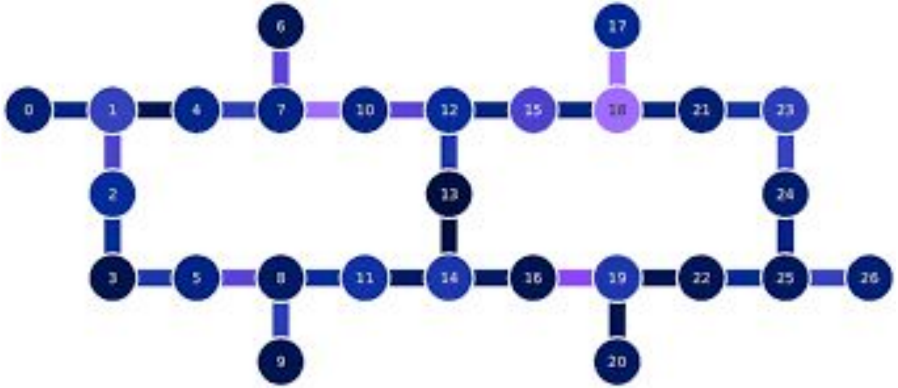
Results returned to user

Why Transpilation?

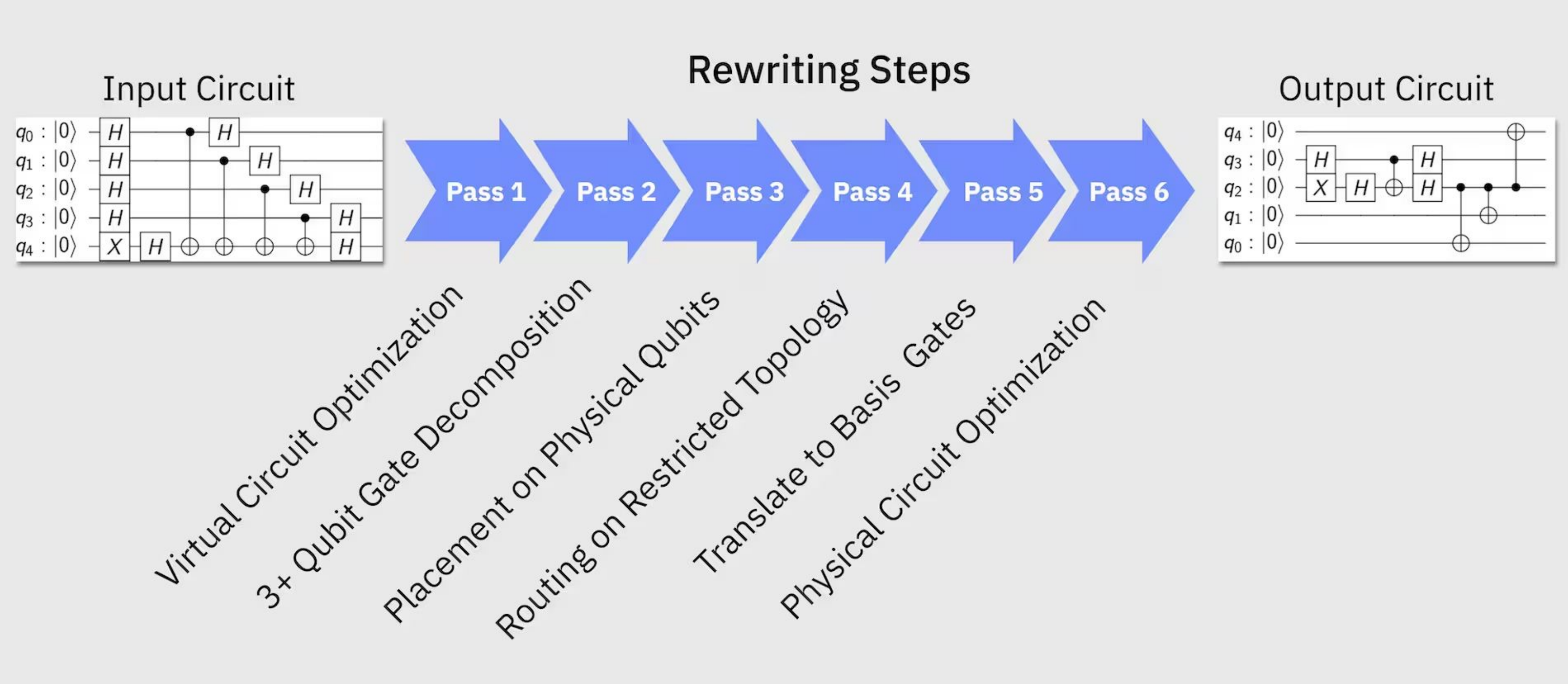
Hardware is bad!

Why Transpilation?





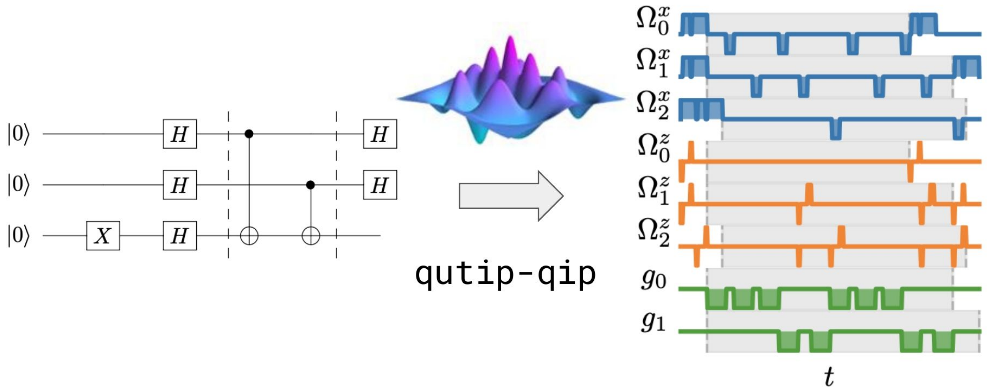
Transpilation Process



Lower-Level Instructions

```
OPENQASM 3.0;
defcalgrammar "openpulse";
cal {
    extern constant(duration, float[64]) -> waveform;
    extern gaussian(duration, duration, float[64]) -> waveform;
    port dac1;
    port adc0;
    port dac0;
    frame tx_frame = newframe(dac1, 5752000000.0, 0);
    frame rx_frame = newframe(adc0, 5752000000.0, 0);
    frame xy_frame = newframe(dac0, 6431000000.0, 0);
}
duration delay_time = 0.0ns;
defcal reset $1 {
    delay[1000000.0ns];
}
defcal measure $1 {
    play(tx_frame, constant(2400.0ns, 0.2));
    capture(rx_frame, constant(2400.0ns, 1));
}
defcal x90 $1 {
    play(xy_frame, gaussian(32.0ns, 8.0ns, 0.2063));
}
for int shot_index in [0:99] {
    delay_time = 0.0ns;
    for int delay_index in [0:100] {
        reset $1;
        x90 $1;
        delay[delay_time] $1;
        x90 $1;
        measure $1;
        delay_time += 100.0ns;
    }
}
```

OpenQASM (Quantum Assembly)



Pulse-level code