CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Quantum Computing

From Theory to Practical

Gen 10 SE Y4 2026-2027

Lecturer: **Ms. HENG Sovanmonynuth**

Software Engineering (SE), Faculty of Digital Engineering, Department of Computer Science

Cambodia Academy of Digital Technology (CADT)

Email: sovanmonynuth.heng@cadt.edu.kh

Mobile: (+855) 95 479 272

Course Outline

Chapter 1: Introduction to Quantum Computing

Chapter 2: Basic of Quantum Mechanics

Chapter 3: Quantum Algorithms

Chapter 4: Quantum Fourier Transform

Chapter 5: Quantum Errors & Noises

Chapter 6: Quantum Machine Learning

Chapter 7: Quantum Chemistry

Chapter 8: Quantum Cryptography

Final Presentation

Chapter 3: Quantum Algorithms

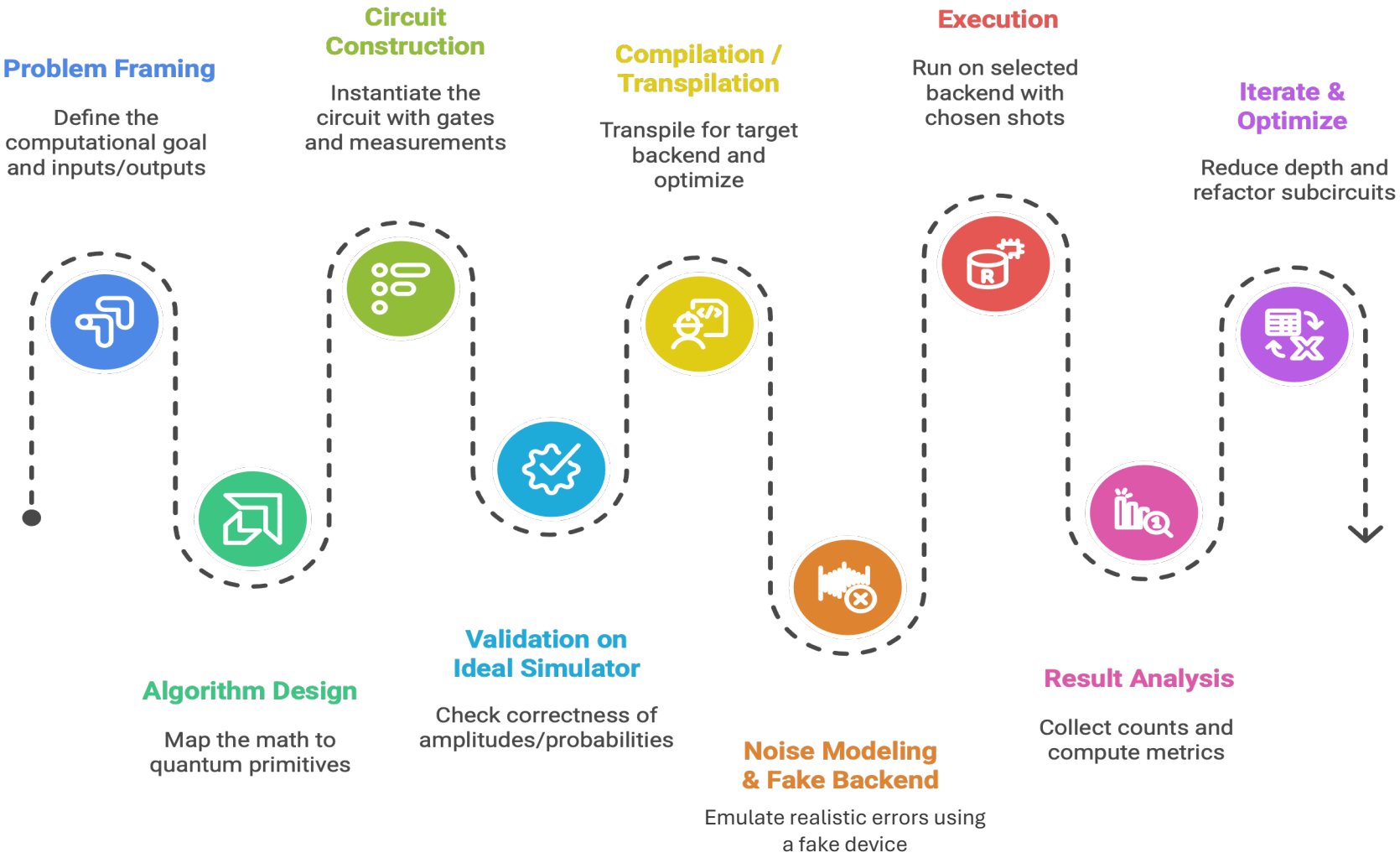
- Introduction to Quantum Algorithms
- Quantum Algorithm Development Process
- Code Demo: Bell State
- Google Colab: Jupyter Notebook
- Quantum Parallelism
- Quantum Phase Kickback Algorithms
- Deutsch-Josza Algorithms
- Lab Practice

Introduction to Quantum Algorithms

- **What is a Quantum Algorithm?**
- A **quantum algorithm** is a step-by-step procedure that uses **quantum bits (qubits)** and **quantum gates** to exploit uniquely quantum effects—**superposition, entanglement, and interference**—to solve certain problems with different (sometimes better) complexity than classical methods.

It's a **program** that choreographs complex probability amplitudes so that ***right answers interfere constructively*** and ***wrong answers cancel out***.

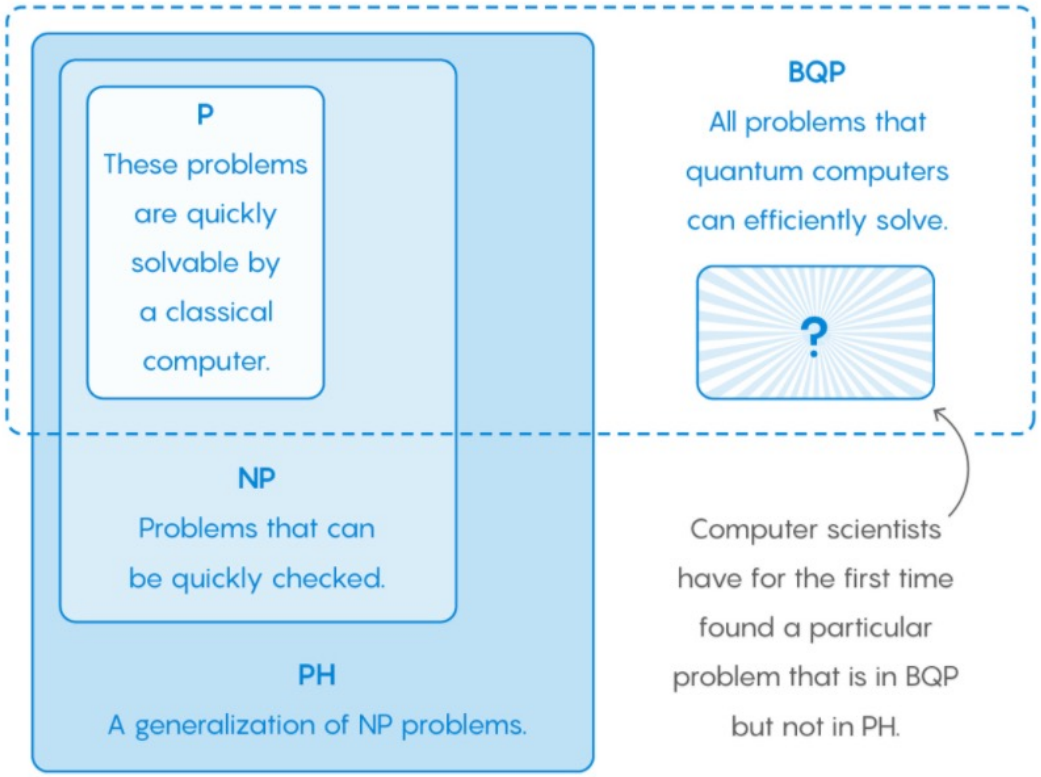
Quantum Algorithm Development Process



Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**
  1. **Problem Framing**
     - Define the ***computational goal*** and ***inputs/outputs***.
       - What question are we answering? (Plain English)
       - What does "success" look like? (a number, a distribution, a bitstring, plain English, and/or insights)
     - Decide if a ***known quantum routine fits*** to the problem

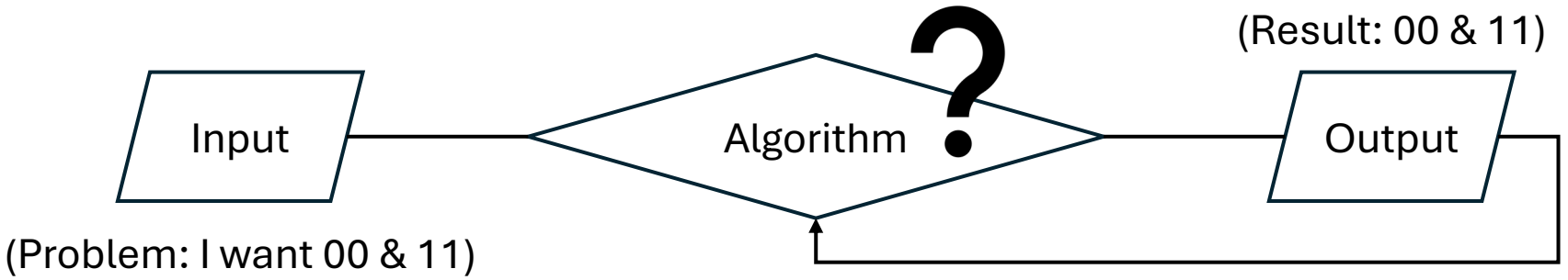
     E.g. Produce entangled $|\Phi\rangle$ so that measured outcomes are only 00 & 11.



Problems that only Quantum Computers can solve: Link

Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**
  2. **Algorithm Design**
     - Determine problem domain, what is the ***input*** *look like*, any *oracle/unitary/black-box*
     - Determine the ***output*** *as measurement* as state, probability, bitstring, estimate, label, etc.
     - Map the *math to quantum primitives* as ***algorithm***
       - State-Preparation → Oracle/Unitary → Interference → Measurement

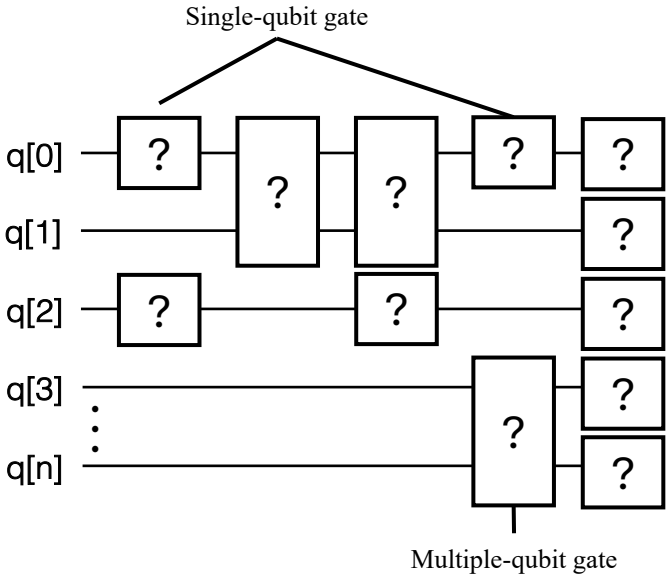


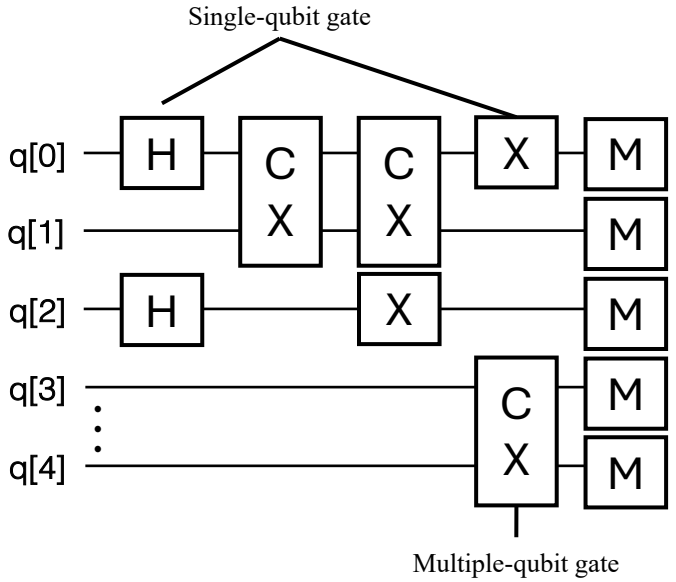
Quantum Algorithm Development Process

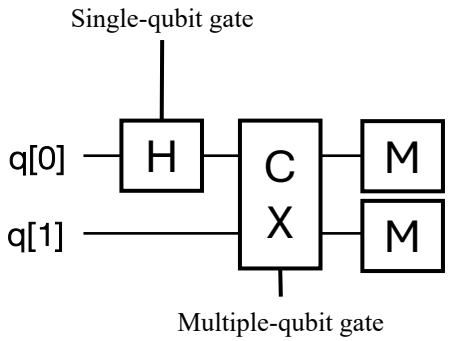
- **Quantum Algorithm Development Process**

  3. Quantum Circuit Construction (Logical)

  - Construct the circuit: ***# qubits***, add ***choice of gates***, and ***measurements***.
  - Add ***barriers*** or ***labels*** to keep sections readable.





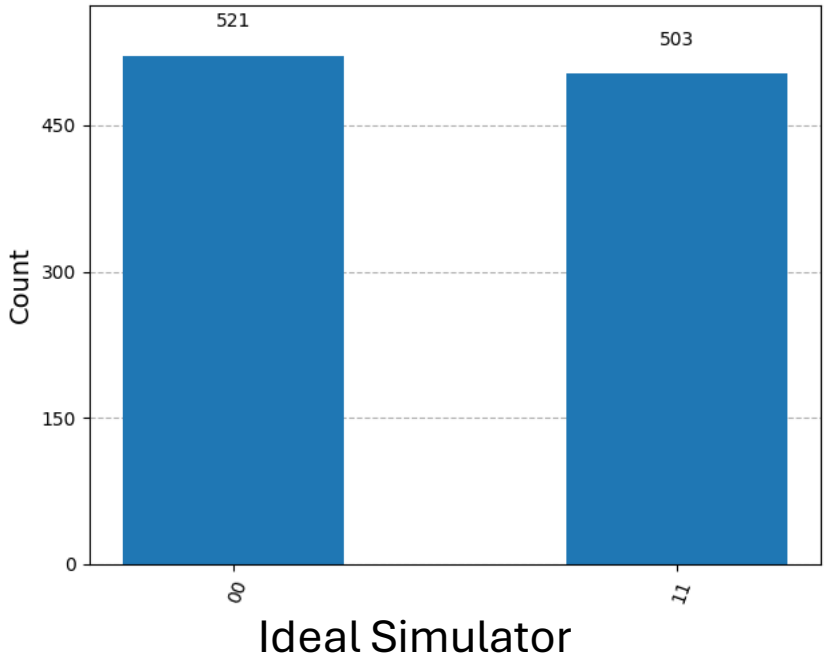


Quantum Algorithm Development Process

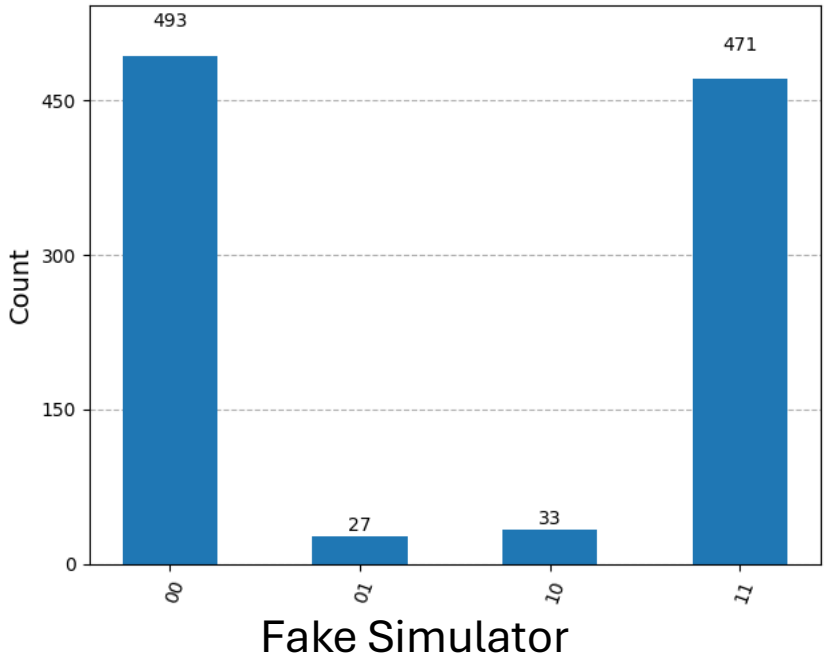
- **Quantum Algorithm Development Process**

  4. Validation on Ideal Simulator

  - Complexity & Scale: number of shots to run and expected runtime
  - Run on an **ideal simulator** to check correctness.



Ideal Simulator



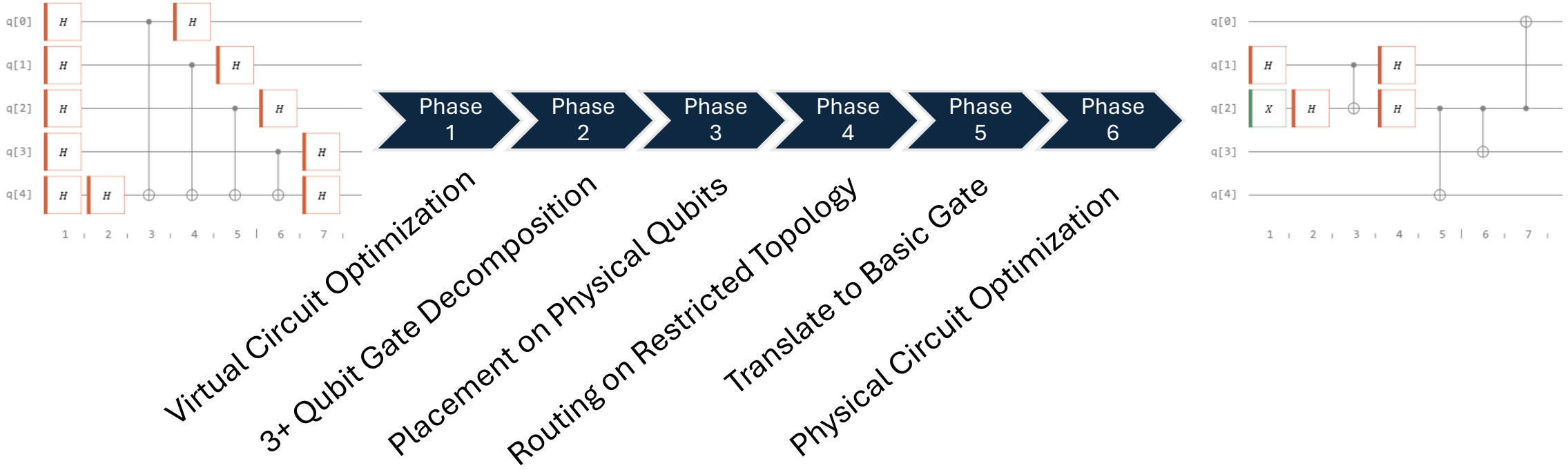
Fake Simulator

Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

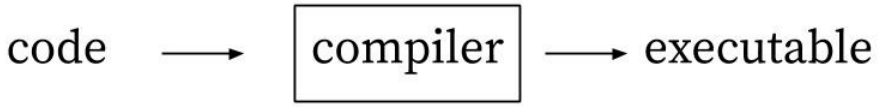
  5. Compilation / Transpilation

  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

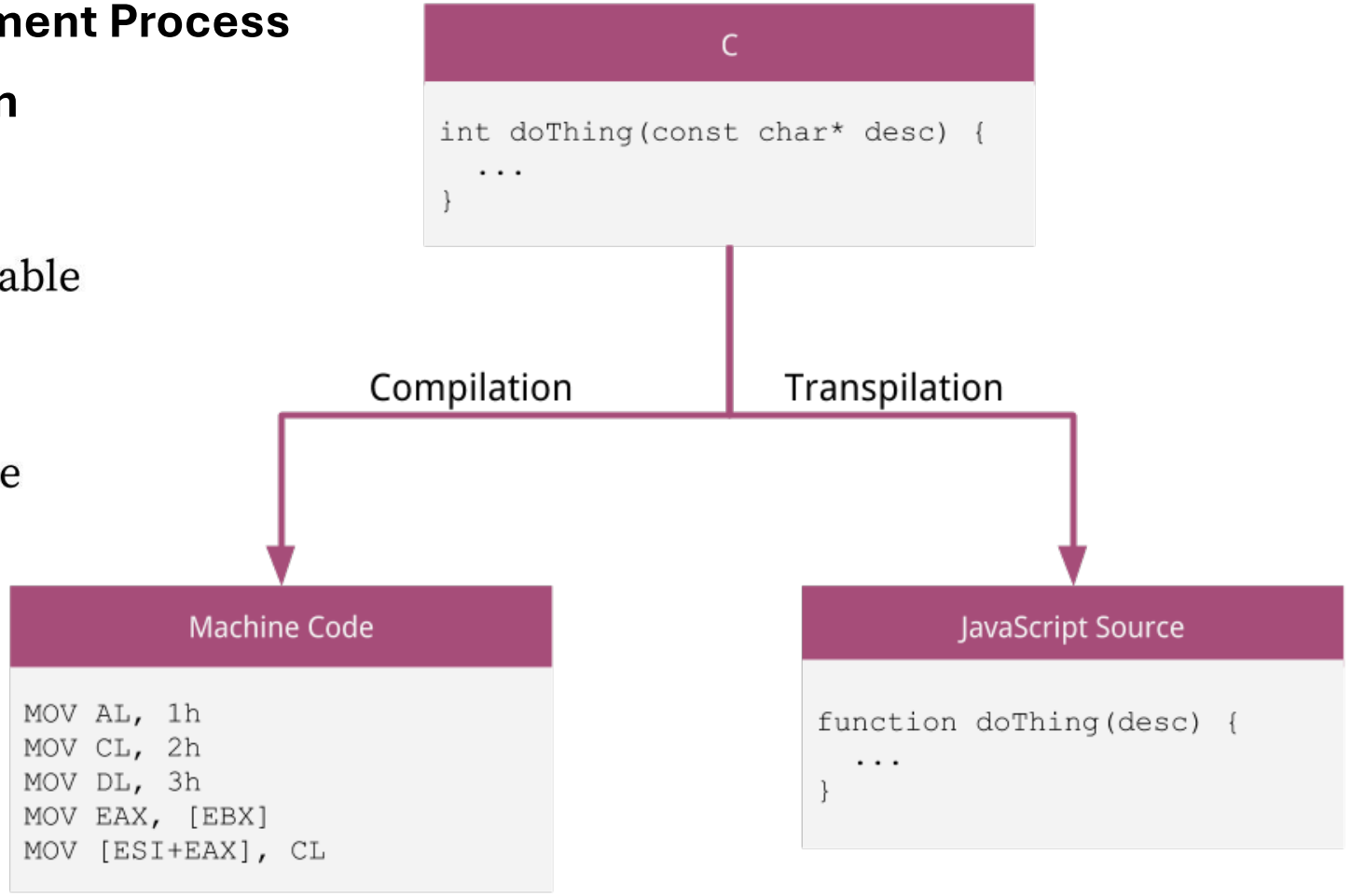


Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**
  5. **Compilation / Transpilation**







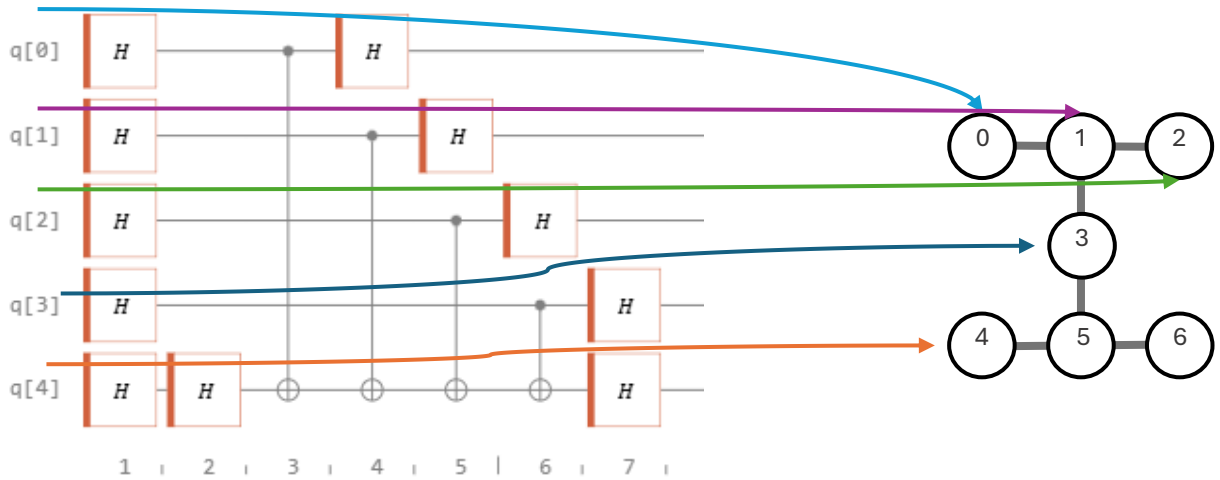
Quantum Algorithm Development Process

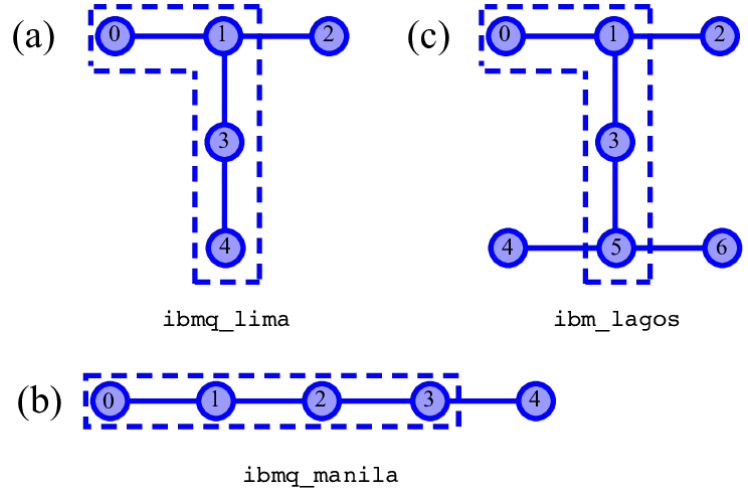
- **Quantum Algorithm Development Process**

  5. Compilation / Transpilation

  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 1 Virtual Circuit Optimization (Logical-Level)





Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

  5. Compilation / Transpilation

  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 2

3+ Qubit Gate Decomposition

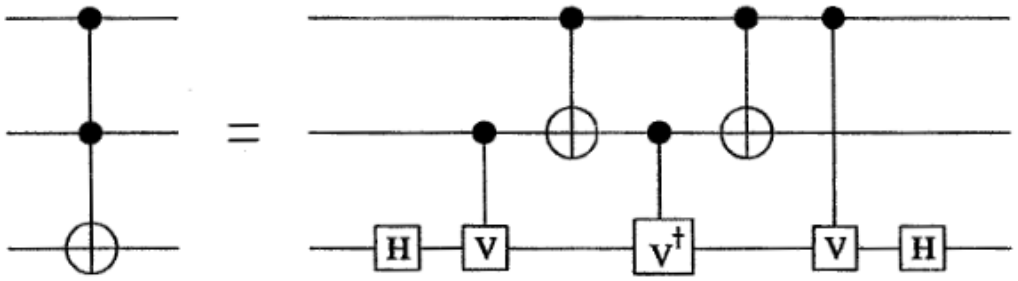


Figure 4: A Toffoli gate can be decomposed into a circuit of 1- and 2-qubit gates. Here $V = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = R_z(\pi/2)$.

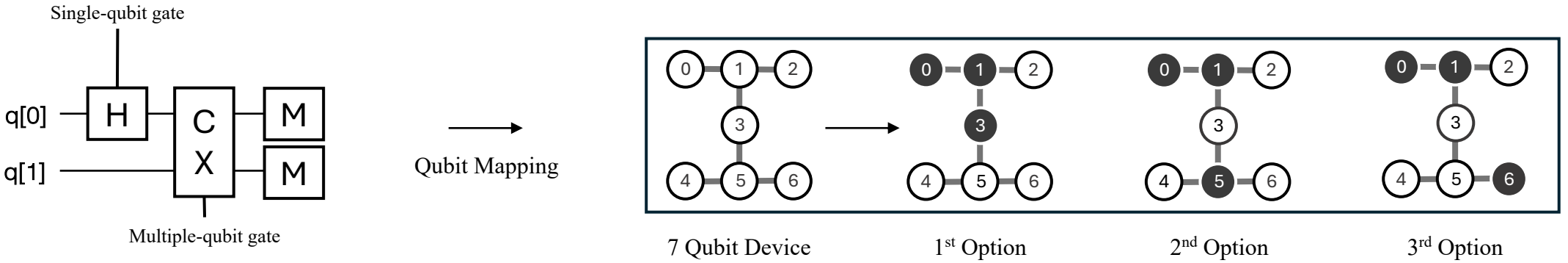
Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

  5. Compilation / Transpilation

  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 3 Placement on Physical Qubits



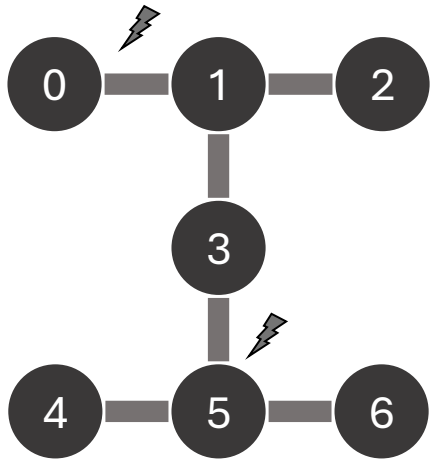
Quantum Algorithm Development Process

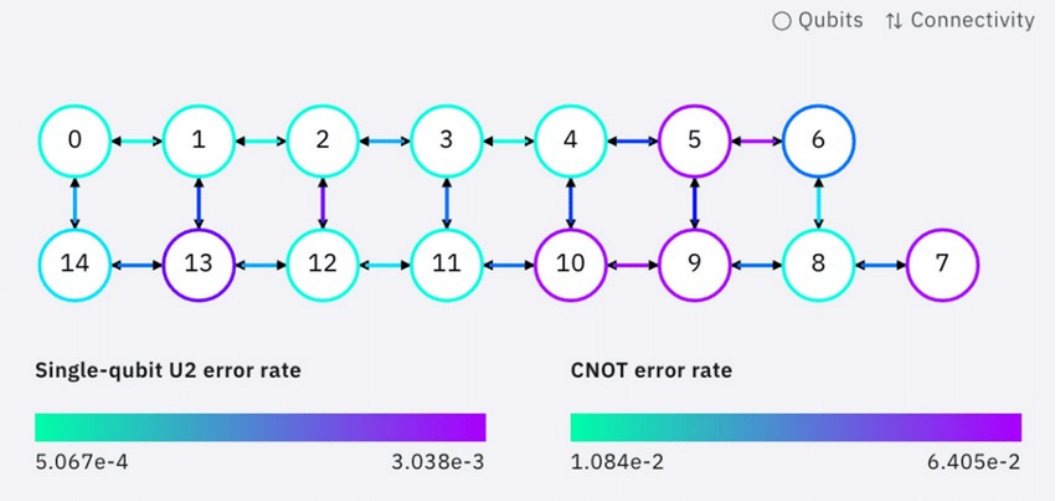
- **Quantum Algorithm Development Process**

  5. Compilation / Transpilation

  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 4 Routing on Restricted Topology





Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

  5. **Compilation / Transpilation**
     - **Transpile** for a target backend to respect native gate set and coupling map.
     - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 5 Translate to Basic Gate

Basic Gate (7 Qubits)	Basic Gate (Heron r3, 133 qubits)
`ID`	`RZ(θ)`
`u3`, `u2`, `u1`, or `u`	`SX`
`x`	`X`
`SX`	`CZ`
`RZ`	`ID`
`cx`	`MEASURE`
`RESET`	`RESET`
`MEASURE`	`DELAY`

Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

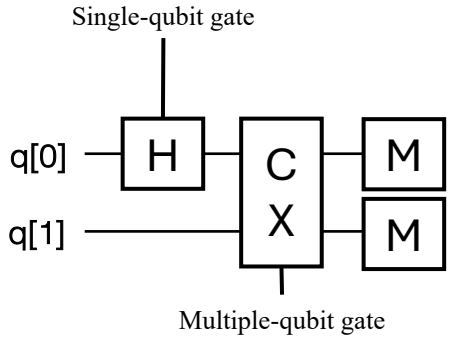
  5. Compilation / Transpilation

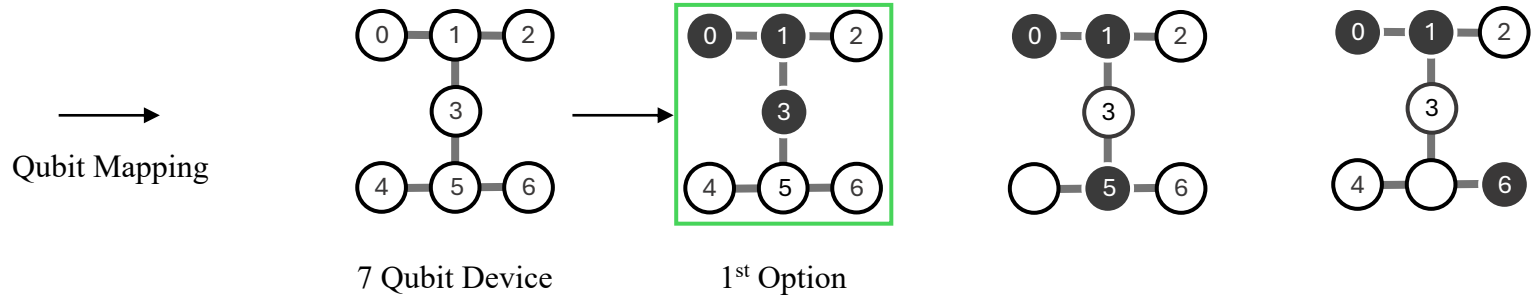
  - **Transpile** for a target backend to respect native gate set and coupling map.
  - **Optimization Level:** Optimize depth and two-qubit gate count (crucial for NISQ).

Phase 6

Physical Circuit Optimization (Hardware-Level)

Cycle/Depth	1	2	3
1	`H -> Q1`	`CX -> Q0&Q1`	`M -> Q0`
2		`CX -> Q0&Q1`	`M -> Q1`

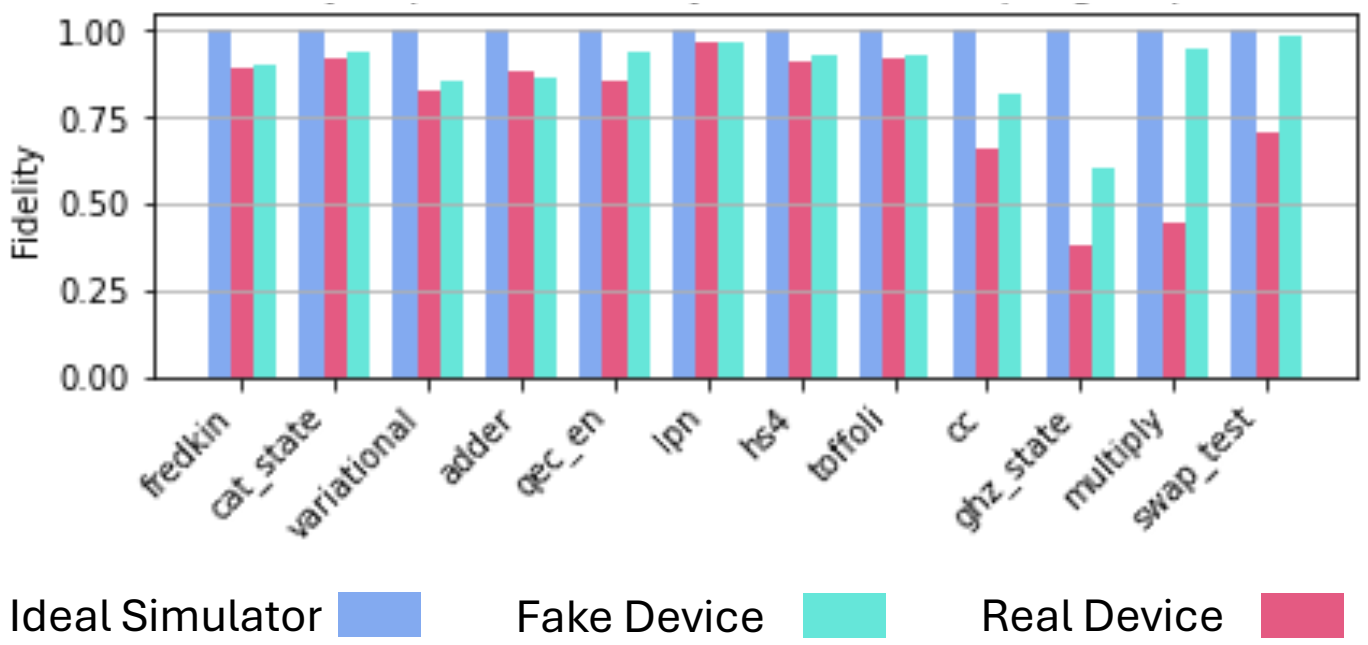




Quantum Algorithm Development Process

- **Quantum Algorithm Development Process**

  6. **Noise Modeling & Fake Backend (Optional but recommended)**
     - Emulate realistic errors using a **fake device** or a **noise model**.
     - Compare results vs ideal to estimate robustness.



Quantum Algorithm Development Process

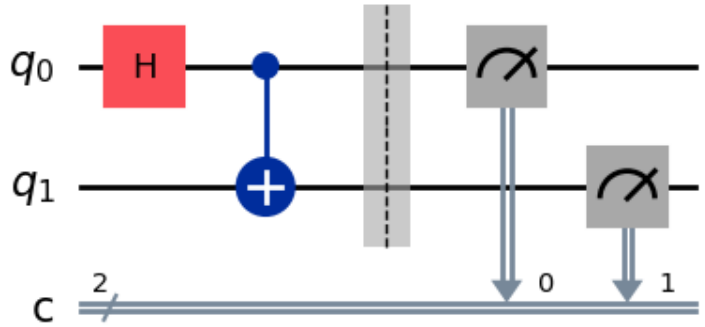
- **Quantum Algorithm Development Process**

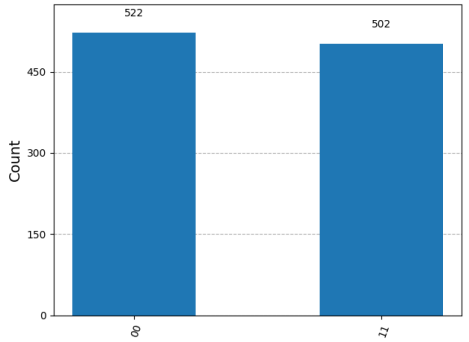
  7. **Execution**
     - **Run** on the selected backend (simulator, fake device, or real device).
     - Choose **shots** (number of repetitions) to reduce statistical variance.
  8. **Iterate & Optimize**
     - Reduce depth, refactor subcircuits, adjust initial states/oracles.
     - Re-transpile with different optimization levels, try layout/seed tweaks.

Demo

Starting with

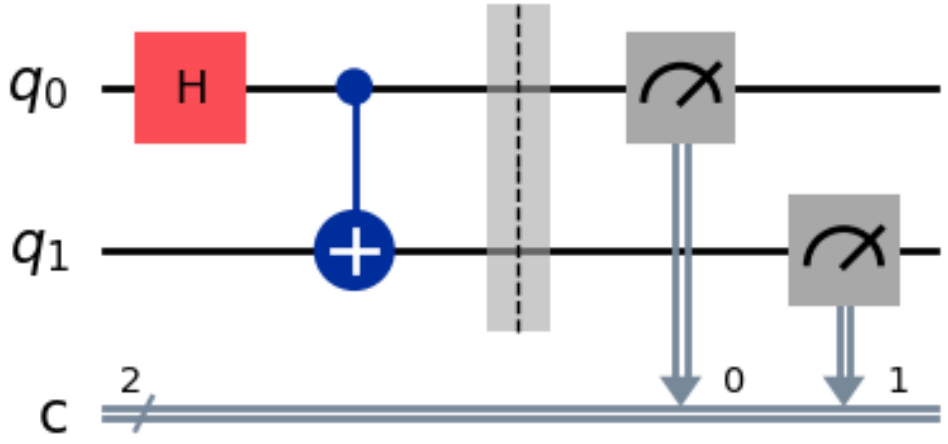
- Installs needed library (qiskit, qiskit-aer, qiskit-ibm-runtime, ..) on Juputer Notebook of Google Colab.
- Builds a **2-qubit Bell** circuit with/without measurements first.
- Has 3 small TODOs:
  - Add measure,
  - Swap simulators (**Aer Simulator** → **QASM**), plot histogram.
- What a Bell state is, what "**Transpile → Backend → Run**" means, and why real devices differ (noise, coupling map, queues).

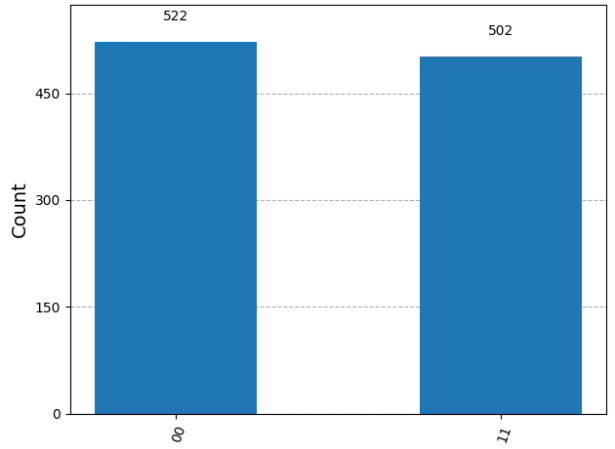




Lab Experiment

- Start from the **starter** so they code along of on **Juputer Notebook of Google Colab**.
  - Build Bell State (**H** + **CX**), draw circuit.
  - Run on ***AerSimulator***
  - Add measure, run on ***qasm_simulator***, show ~50/50 to the state 00/11, respectively.
  - Run on other simulator to see the result
- Keep the hardware part optional (queues can derail pacing). If you want realism without queues, use a **fake device** + **noisy Aer**.





Quantum Parallelism

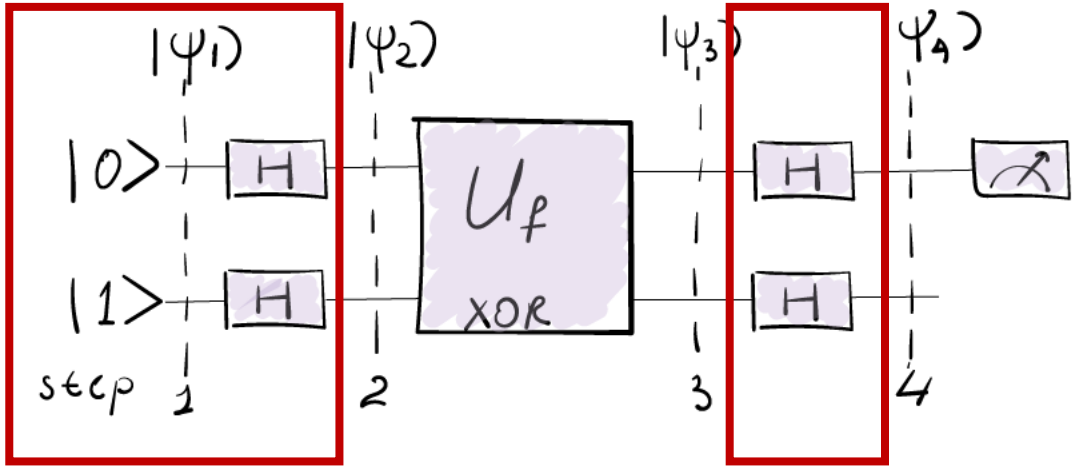
- **Principle:** A quantum computer evaluates ***f(x)*** for all ***x*** in ***superposition*** in one step.
- **Mathematics:**

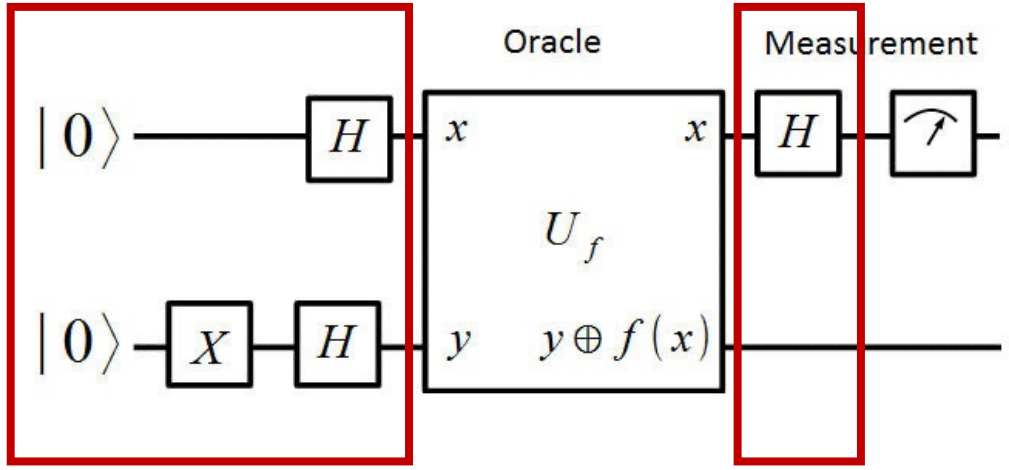
$$\sum_x |x\rangle|0\rangle \xrightarrow{U_f} \sum_x |x\rangle|f(x)\rangle$$

- **E.g.** Oracle marks solutions without revealing them (black-box function) such as Deutsch-Jozsa, Grover, Shor, Simon, and many others.

Quantum Parallelism

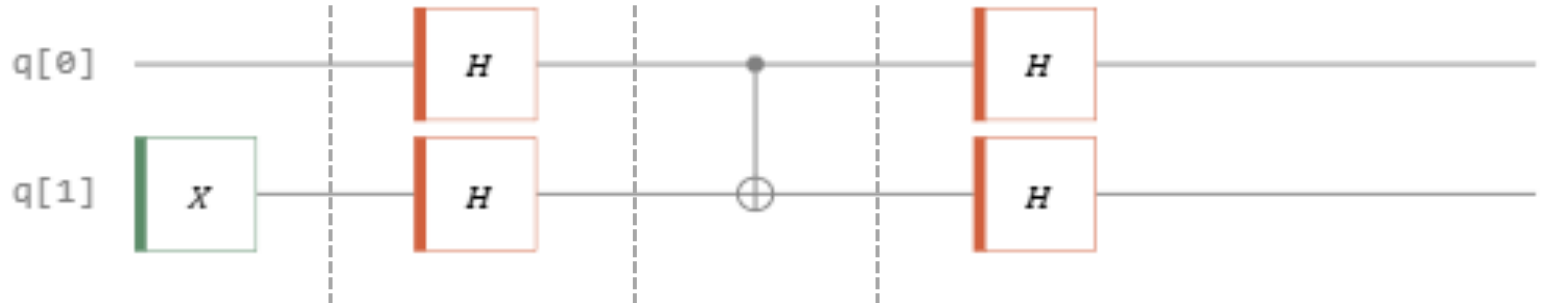
- **Preparation Step:**
  - Initiate ***X gate*** on ***qubit nth*** e.g. X → q[1]
  - Apply ***superposition*** on ***#n qubits*** e.g. H → q[0], q[1], …, q[n]
  - Apply ***oracle U_f*** e.g. CNOT → q[0] & q[1]
  - Apply **interfere** on ***#n qubits*** e.g. H → q[0], q[1], …, q[n]
  - Apply ***#m measure*** e.g. M → q[0] & q[1]





Lab Experiment

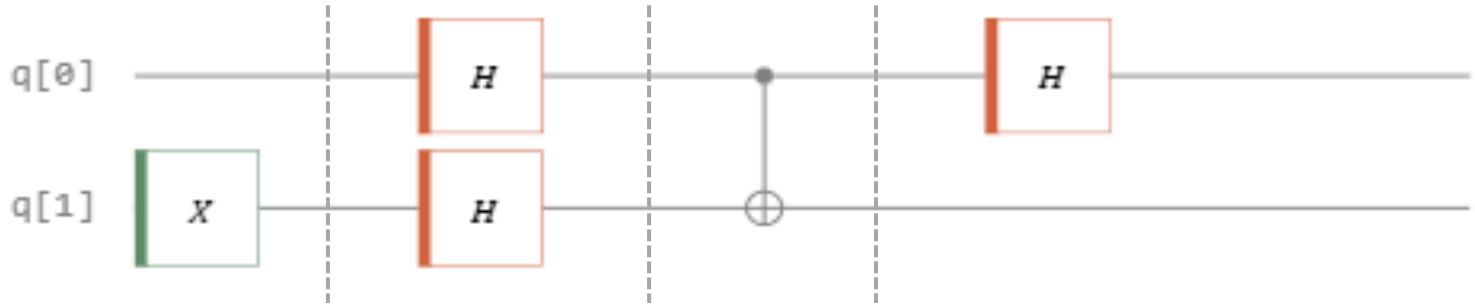
- **Experiment 1:** Try implement the **oracle circuit** to get the output of **11 only**?

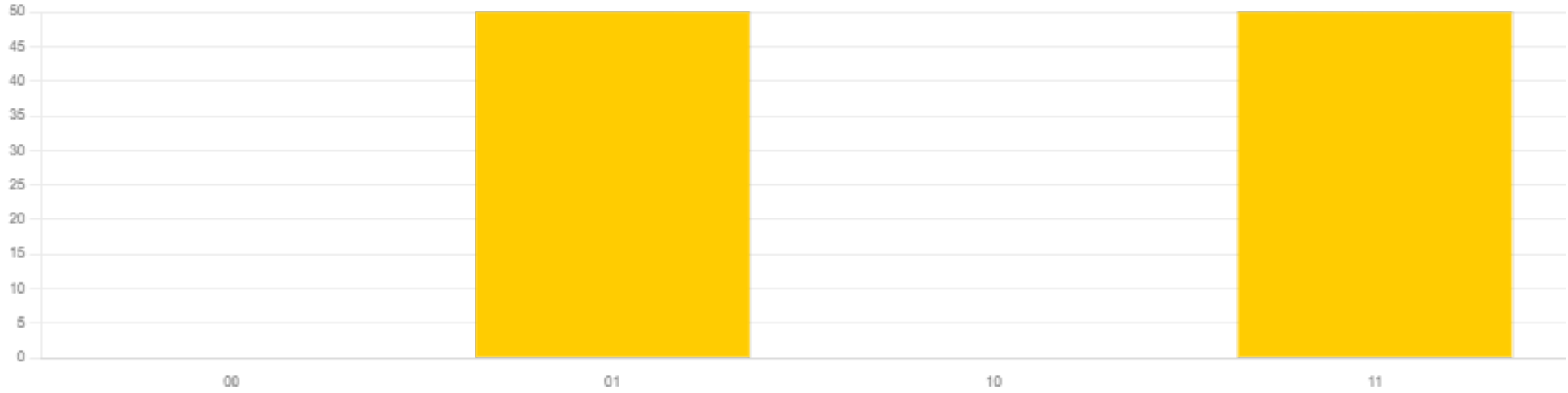




Lab Experiment

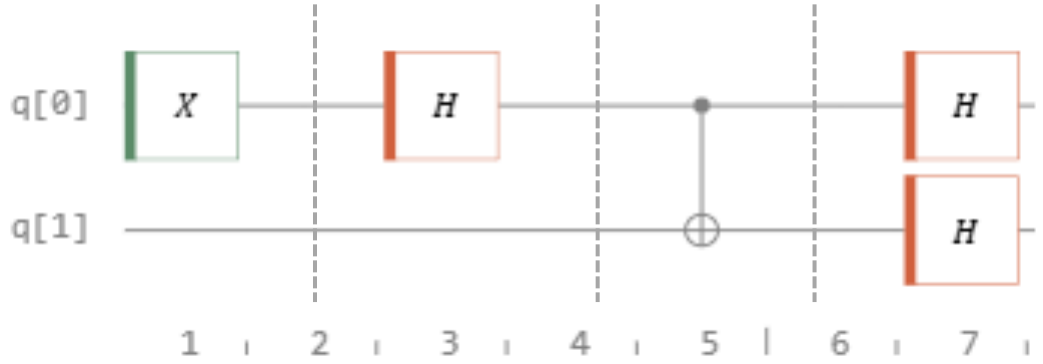
- **Experiment 2:** Try implement the **oracle circuit** to get the output of **01 and 11 only**?

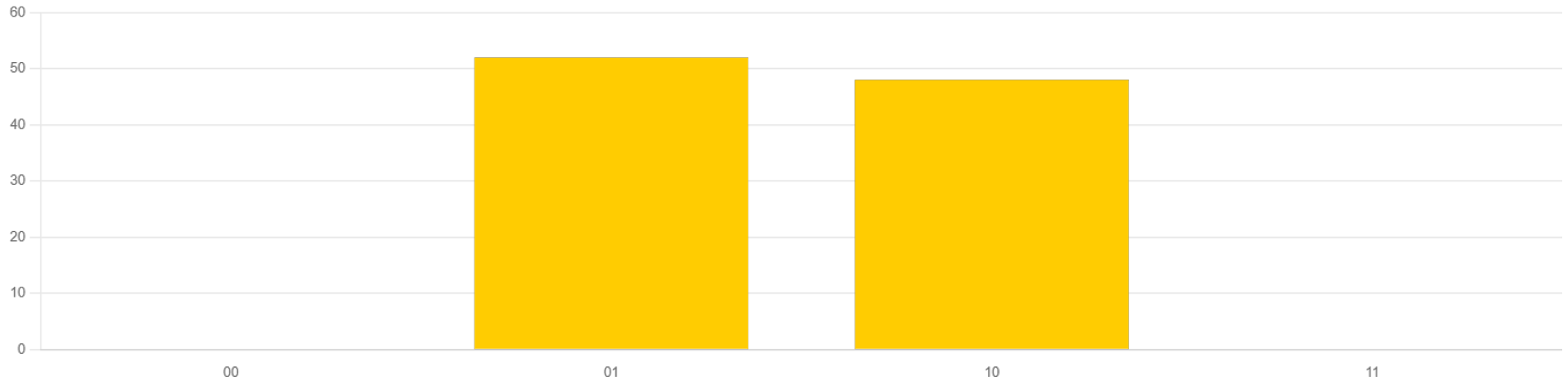




Lab Experiment

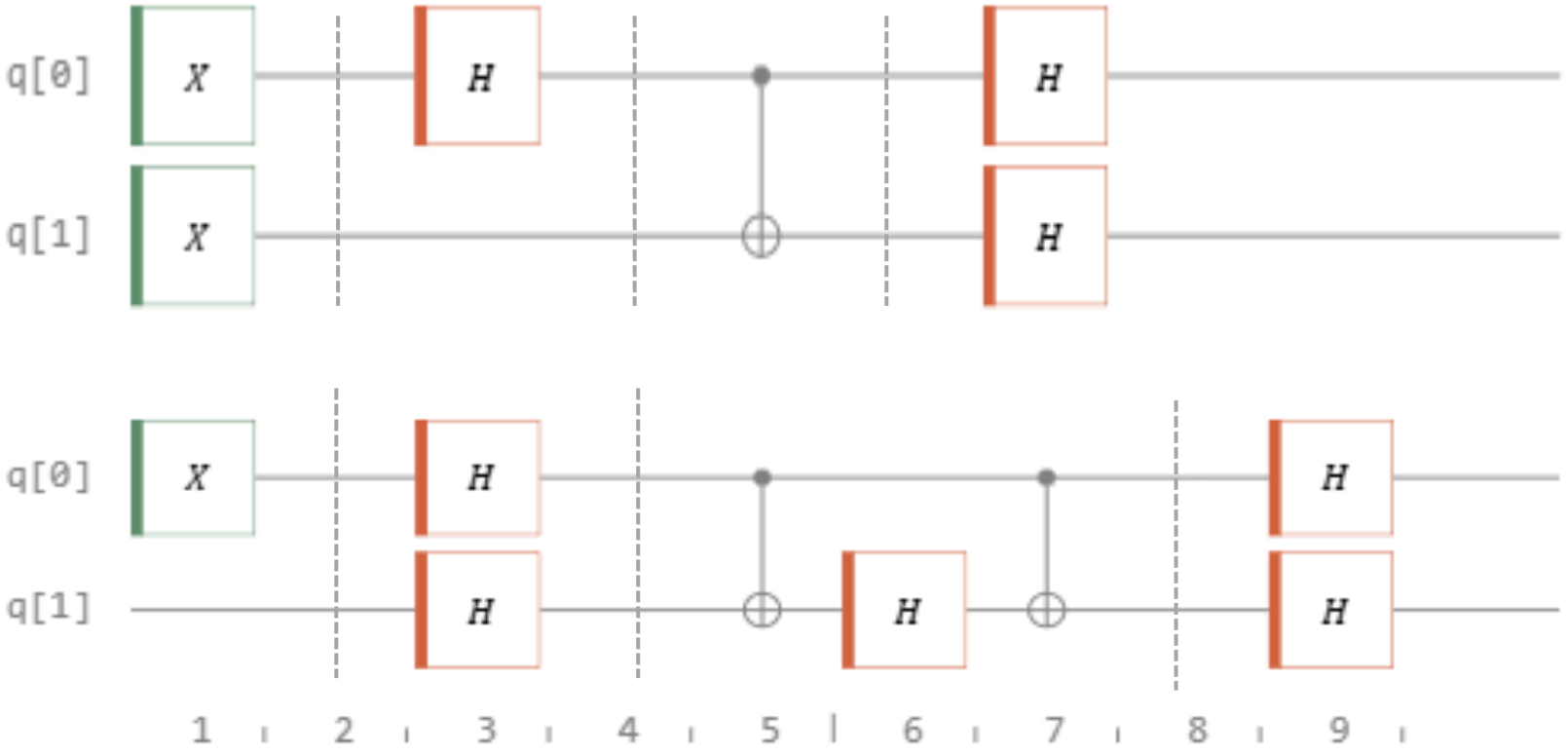
- **Experiment 3:** Try implement the **oracle circuit** to get the output of **01** **and** **10** **only**?





Lab Experiment

- **Experiment 3:** Try implement the **oracle circuit** to get the output of **01** **and** **10** **only**?



Quantum Phase Kickback

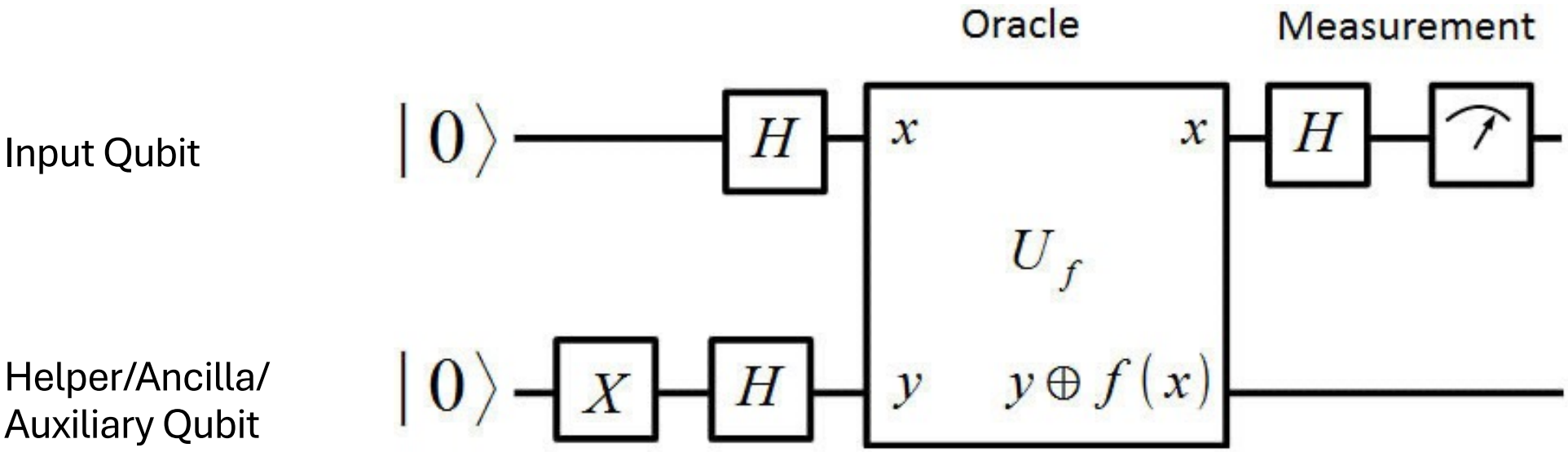
Quantum Phase Kickback (QPK) Algorithms

- **Goal:** ***Encode hidden information*** as a **phase** on a qubit and transfer it from the **target to the control qubit** — used in Deutsch–Jozsa, Bernstein–Vazirani, Phase Estimation, and Shor's algorithms.
- **Classical:** Classical algorithms must read or store function outputs.
- **Preparation Steps:**
  - Prepare **control qubit** in **superposition** — e.g. H → q[0]
  - Prepare **target qubit** in $|1\rangle$ or $|\psi\rangle$ — e.g. X → q[1]
  - Apply **Controlled-U** gate (adds phase $e^{\{i\psi\}}$ on control) — e.g. CNOT → q[0] & q[1]
  - Apply **Hadamard** or QFT to convert phase — e.g. H → q[0]
  - **Measure control qubit** — e.g. M → q[0]
- **Key Idea:** The **phase shift** on the target is "kicked back" to the control, encoding hidden information that can be extracted by interference or QFT (Next Chapter).

Quantum Phase Kickback

Quantum Phase Kickback (QPK) Algorithms

- **Goal: *Encode hidden information*** as a **phase** on a qubit and transfer it from the **target to the control qubit** — used in Deutsch–Jozsa, Bernstein–Vazirani, Phase Estimation, and Shor's algorithms.



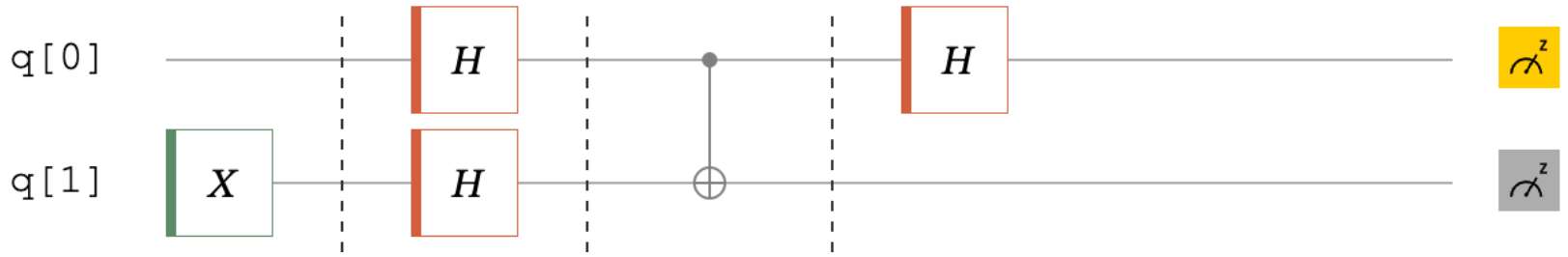
Quantum Phase Kickback

QPK vs **Superdense Coding** vs **Quantum Teleportation**

Concept	Quantum Phase Kickback (QPK)	Superdense Coding (SDC)	Quantum Teleportation (QT)
Field	Quantum **computation**	Quantum **communication**	Quantum **communication / transfer**
Goal	Extract or encode **phase** into another qubit	Send **2 classical bits** using 1 qubit	Send **1 quantum state** using 2 classical bits
Uses Entanglement?	Sometimes (indirectly via controlled gates)	Yes, pre-shared entangled pair	Yes, pre-shared entangled pair
Information type moved	**Quantum phase** (hidden property)	**Classical bits** (00, 01, 10, 11)	**Quantum state** (amplitude + phase)
Who communicates?	Inside one quantum system	Between Alice → Bob	Between Alice → Bob
What is transmitted?	A **phase kickback** between qubits	A **qubit** carrying encoded bits	**Two classical bits**, no qubit needed after entanglement
Key gate	Controlled-U	Pauli (X, Z) on Alice's qubit	Bell measurement + Pauli correction

Lab Experiment

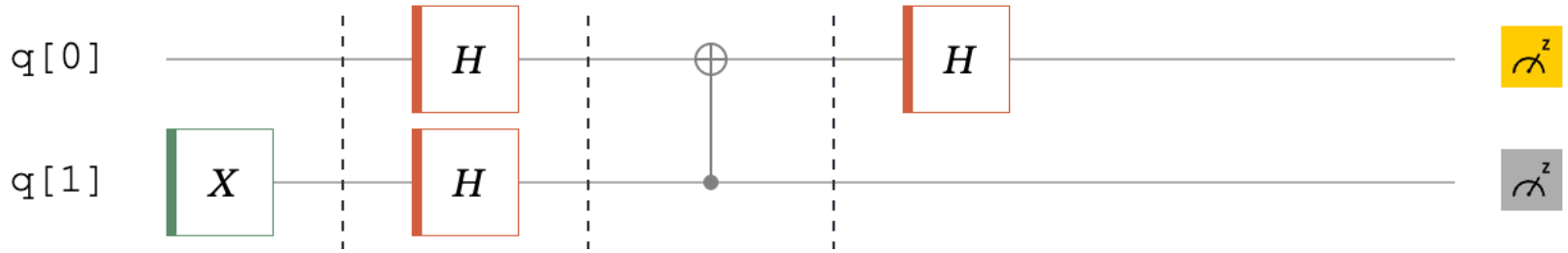
- **Experiment 4:** Try insert the QPB algo and how figure out how to get **1** on the histogram?

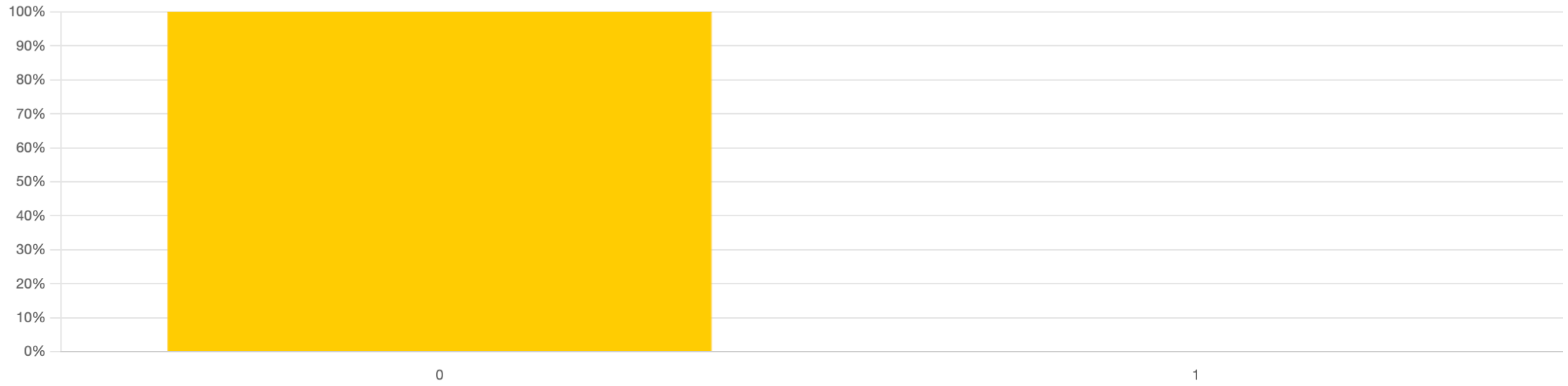




Lab Experiment

- **Experiment 4:** Try insert the QPB algo and how figure out how to get **0** on the histogram?





Deutsch–Jozsa Algorithm

Deutsch–Jozsa Algorithm

- **Goal:** Determine if a Boolean function $f(x): \{0,1\}^n \rightarrow \{0,1\}$ is **constant** or **balanced**.
- **Classical:** Needs $2^{n-1} + 1$ queries **Quantum:** only 1 queries of quantum oracle

  E.g. If **n = 3** → You might have to test **5** different inputs to be sure whether it's constant or balanced.
- **Preparation Steps:**
  - Initiate ***X gate*** on ***qubit nth*** e.g. X → q[1]
  - Apply ***superposition*** to ***all qubits*** e.g. H → q[0], q[1], …, q[n]
  - Apply the ***oracle*** U_f e.g. U → q[0] & q[1]
  - Apply ***interference or superposition*** again to input register e.g. H → q[0], q[1], …, q[n]
  - ***Measure → all*** e.g. M → q[0], q[1], …, q[n]
- **Result Meaning:** 0 ⇒ constant, otherwise balanced

Deutsch–Jozsa Algorithm

Deutsch–Jozsa Algorithm

- **Constand or balanced** meaning takes an n**-bit input** and returns either **0 or 1 for** each possible input.
- E.g. A function is **constant** if it gives **the same output** for **all possible inputs**. While **balanced** if it outputs **0 for exactly half** of the possible inputs and **1 for the other half**.

Input (x)	f(x)
00	0
01	0
10	0
11	0

Input (x)	f(x)
00	1
01	1
10	1
11	1

Constant

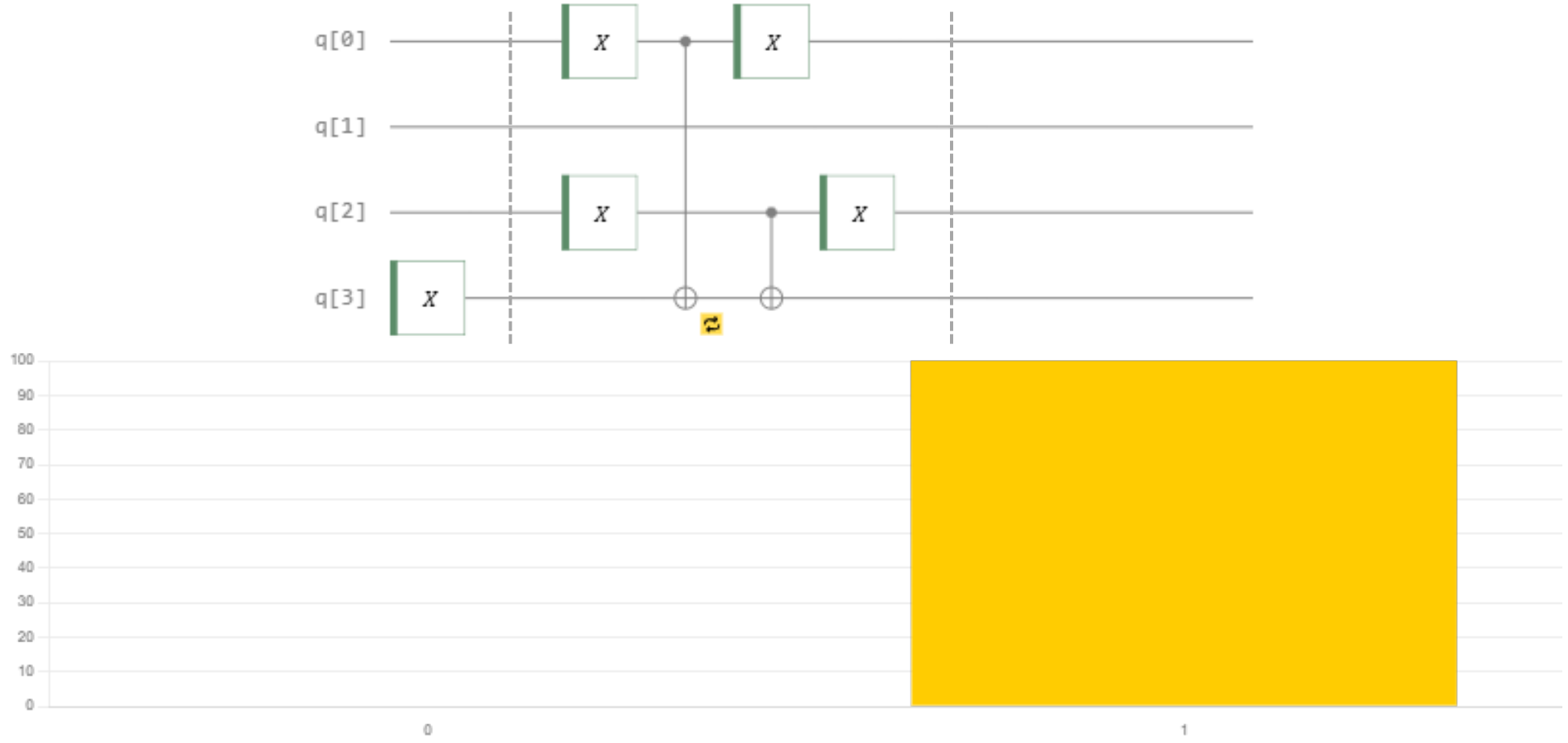
Input (x)	f(x)
00	0
01	1
10	1
11	0

Balanced

- **Constant Meaning:** No matter what you input, the output never changes **(always 0 or 1)**. It's like a ***stubborn box*** that ignores your question and gives the same answer every time.

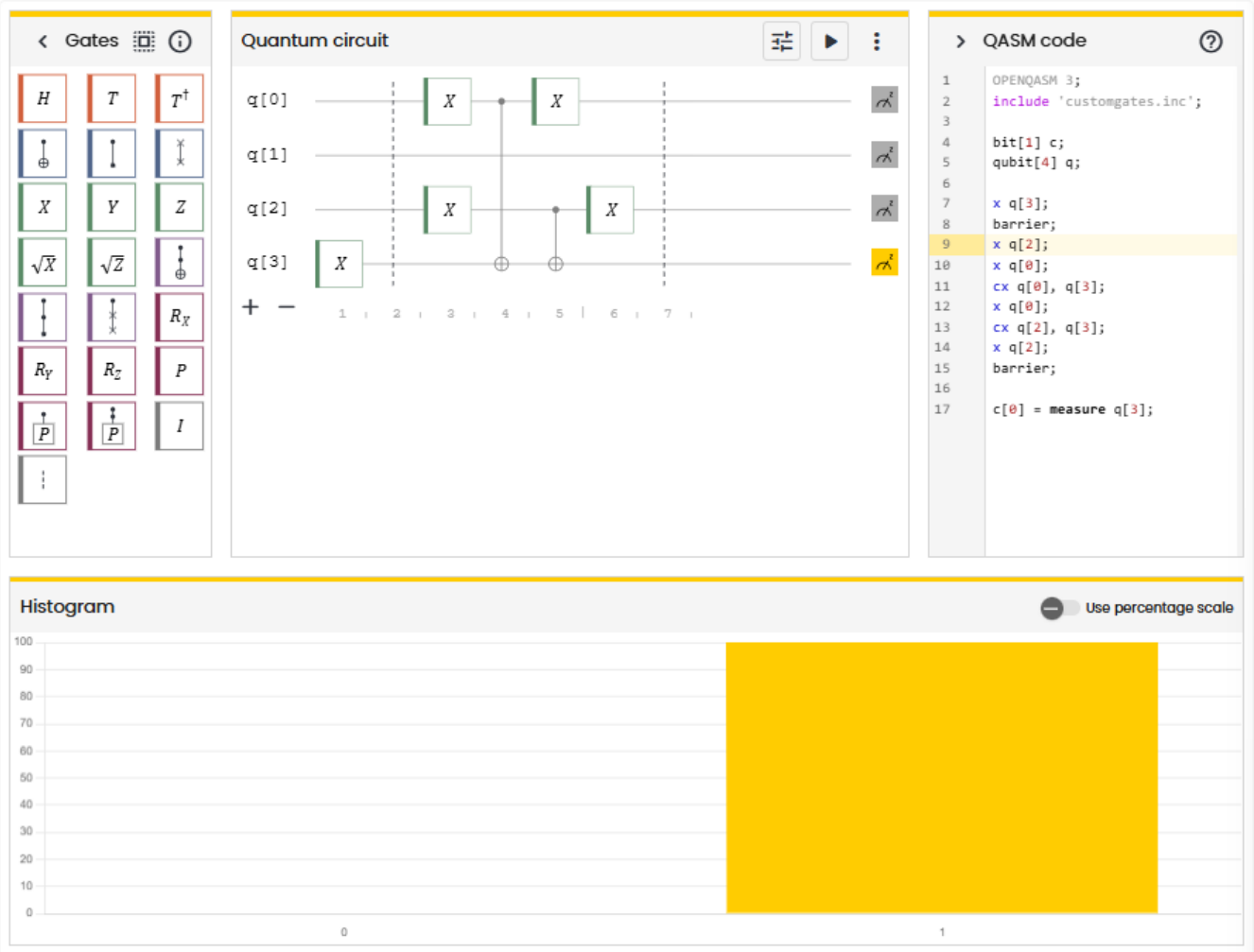
Lab Experiment

- **Experiment 5:** Try the same **classical circuit** and how figure out how to get **1**?



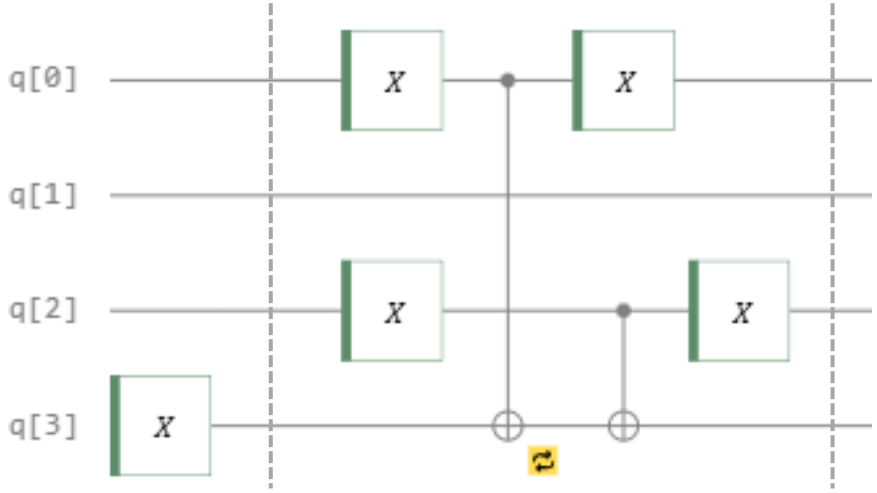
Lab Experiment

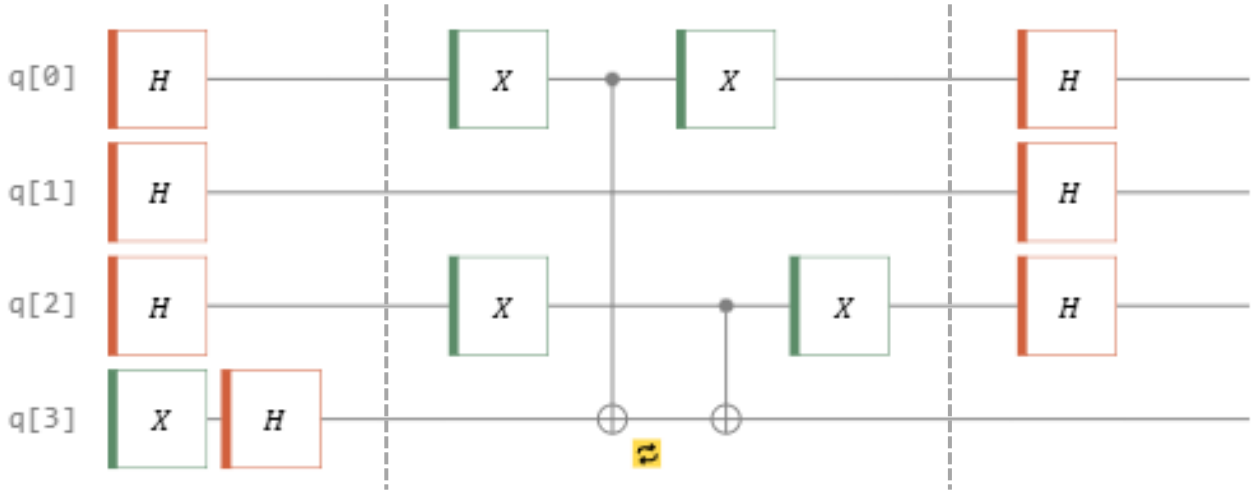
- **Experiment 5**



Lab Experiment

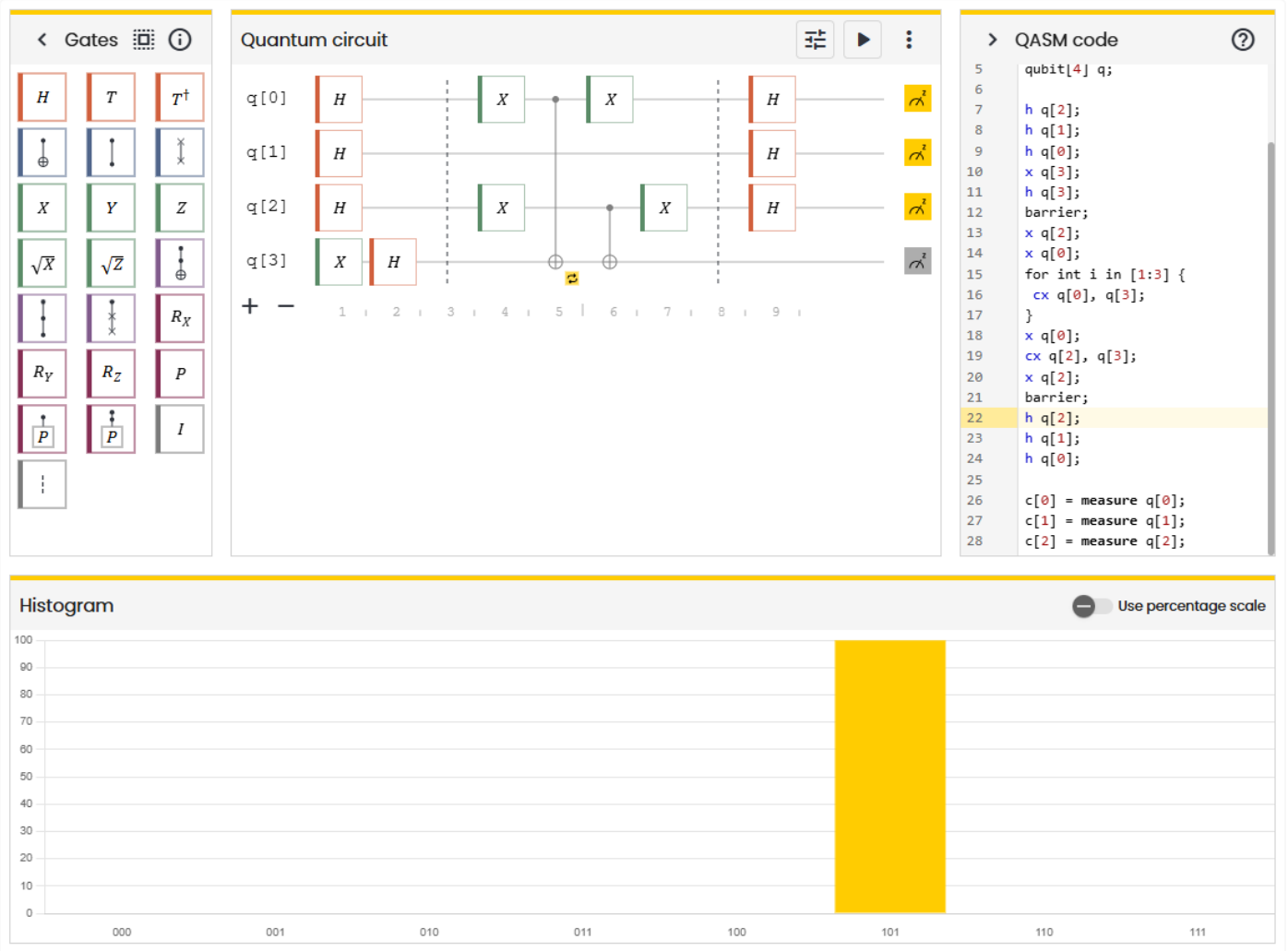
- **Deutsch-Jozsa Algorithm**
  - Try to convert this **classical circuit** to **quantum circuit**
  - Figure out what will be the probability of its





Lab Experiment

- **Deutsch-Jozsa Algorithm**



Weekly Assignment

Please complete and submit your **Jupyter Notebook (.ipynb)** file containing the three algorithms we learned today. Use **Qiskit** to build, simulate, and explain each circuit. Upload your completed file to **Telegram** before the deadline.

Your notebook should include:

- **Bell State Circuit**
- **Quantum Phase Kickback Algorithm**
- **Deutsch–Jozsa Algorithm**

Make sure to:

- Add short explanations (in Markdown) after each code block.
- Include circuit diagrams and probability histograms.
- Save your file as **GroupNo_YourID_YourName_Chatper3_QuantumAlgorithm.ipynb** and upload it to **Telegram**


Thank you!

CADT IDT Institute of Digital Technology

SPINQ 量旋科技

Topic:

Quantum Computing: Past, Present, and Future



Kaiqi Zheng

Senior Engineer at SpinQ

28th October 2026
9:00am - 10:30am

Online Session