Quantum Computing

- Week 3 (April 28-29, 2026)

- Today:
  - Quantum unitary operations

Qubit

- Sigle-qubit state: The numbers α and β are **complex number** and $|\alpha|^2 + |\beta|^2 = 1$

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- An $\boldsymbol{n}$**-qubit states** (in the computational basis)

$$|\boldsymbol{\phi}\rangle = \sum_{i=0}^{2^n-1} \alpha_i \, |i\rangle, \text{ where } \alpha_i \in \mathbb{C} \text{ and } \sum_{i=0}^{n-1} |\alpha_i|^2 = 1$$

- General description: Let $\{|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_{N-1}\rangle\}$ be an orthonormal basis

$$|\boldsymbol{\phi}\rangle = \sum_{i=0}^{N-1} \alpha_i \, |\phi_i\rangle, \text{ where } \alpha_i \in \mathbb{C} \text{ and } \sum_{i=0}^{N-1} |\alpha_i|^2 = 1$$

Qubit

- Some operations introduced last week:
  - **Adjoint**: $\boldsymbol{U}^{\dagger} = (\boldsymbol{U}^{*})^{\boldsymbol{T}} = \left(\boldsymbol{U}^{\boldsymbol{T}}\right)^{*}$
  - **Inner product/Outer product**: $\langle\psi|\phi\rangle, |\psi\rangle\langle\phi|$
  - **Tensor product**: $|\phi\rangle \otimes |\phi\rangle = |\phi\phi\rangle, \boldsymbol{U_1} \otimes \boldsymbol{U_2}$

Qubit

- Some operations introduced last week:
  - **Adjoint:** $\boldsymbol{U^\dagger = (U^*)^T = \left(U^T\right)^*}$
  - **Inner product/Outer product:** $\langle\psi|\phi\rangle, |\psi\rangle\langle\phi|$
  - **Tensor product:** $|\phi\rangle \otimes |\phi\rangle = |\phi\phi\rangle, \boldsymbol{U_1 \otimes U_2}$

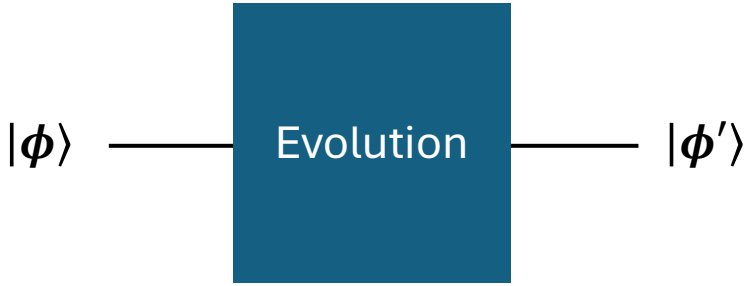
Multi-qubit defined by tensor product of single qubits:

Example: $|10101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle$

Notation: $\boldsymbol{|\phi\rangle^{\otimes n} := |\phi\rangle \otimes |\phi\rangle \otimes \cdots \otimes |\phi\rangle}$ **(n times)**

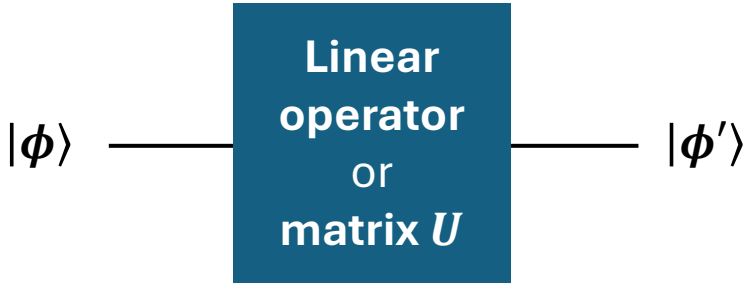
Quick exercise: What is $|+\rangle^{\otimes n}$

Unitary Operation



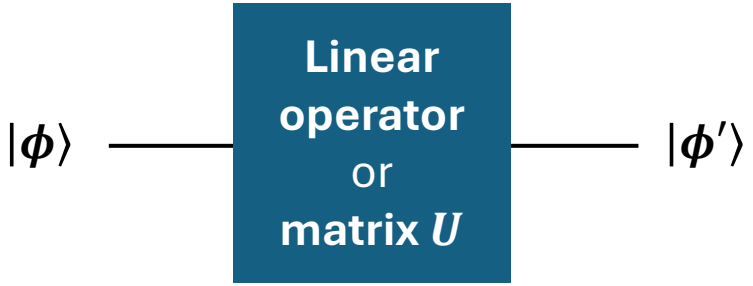
- The **Schrödinger equation** describes the evolution of the quantum state of an isolated system
  - The equation is **linear** (i.e., any linear combination of solutions is a solution)
- ⇒ The evolution of quantum states is also linear
  - Always keep in mind: **linear operations ⟺ matrices!**
- We use **linear operators (or matrices) to describe evolutions of quantum states**.

Unitary Operation



- We use **linear operators** (or matrices) to describe evolutions of quantum states.
- Observations:
  - (1) A quantum state – (evolution) → another quantum state
  - (2) By definition, a quantum state is a unit vector (normalized condition)

Unitary Operation



- We use **linear operators** (or matrices) to describe evolutions of quantum states.
- Observations:
  - (1) A quantum state – (evolution) → another quantum state
  - (2) By definition, a quantum state is a unit vector (normalized condition)
- **Quantum evolutions preserve the norm!**
  - Let U denote such a linear operation. **For any quantum state** $|\phi\rangle$**,** $\||\phi\rangle\| = \|U|\phi\rangle\| = 1$

Unitary Operation

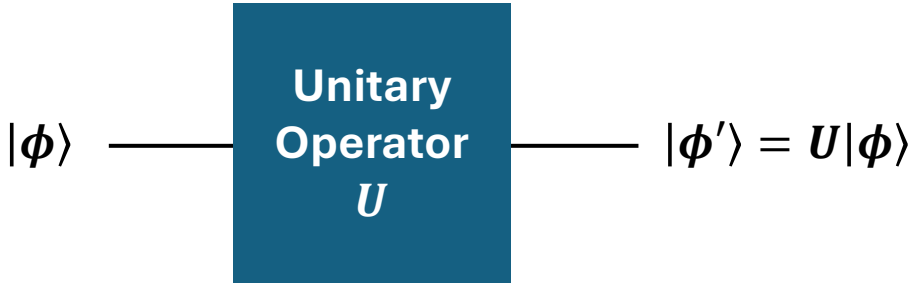
Some Linear Algebra – **Unitary**:

- **Unitary matrices** (unitary operators, or simply unitaries)
- A square matrix $\boldsymbol{U}$ is a unitary if **one of the following conditions holds**:
  - (1) **For any** $|\boldsymbol{\phi}\rangle$, $\||\boldsymbol{\phi}\rangle\| = \|\boldsymbol{U}|\boldsymbol{\phi}\rangle\|$
  - (2) $\boldsymbol{U}^{\dagger} = \boldsymbol{U}^{-1}$ (or $\boldsymbol{U}^{\dagger}\boldsymbol{U} = \boldsymbol{I}$)
  - …
- Exercise: (1) ⇔ (2)

- **Hermitian**: A matrix (or linear operator) $\boldsymbol{U}$ is *Hermitian* or *self-adjoint* if $\boldsymbol{U} = \boldsymbol{U}^{\dagger}$
- **Normal operator/matrix**: $\boldsymbol{U}\boldsymbol{U}^{\dagger} = \boldsymbol{U}^{\dagger}\boldsymbol{U}$ (but not necessarily $= \boldsymbol{I}$)
- Quick thought: Unitary ⟹ Normal

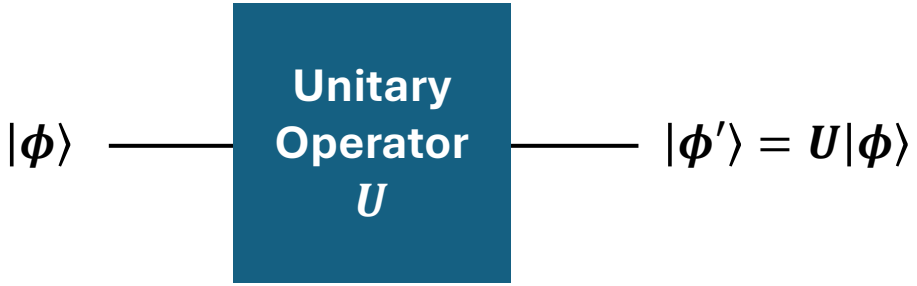
- **Quantum evolutions (linear operators or matrices) preserve the norm**
  - Let $\boldsymbol{U}$ denote such a linear operation. **For any quantum state** $|\boldsymbol{\phi}\rangle$, $\||\boldsymbol{\phi}\rangle\| = \|\boldsymbol{U}|\boldsymbol{\phi}\rangle\| = \mathbf{1}$

Unitary Operation



- We use a **unitary operator** to describe the evolution of a quantum state.
- In quantum computing, we use **unitary operations** to manipulate qubit(s)

Unitary Operation



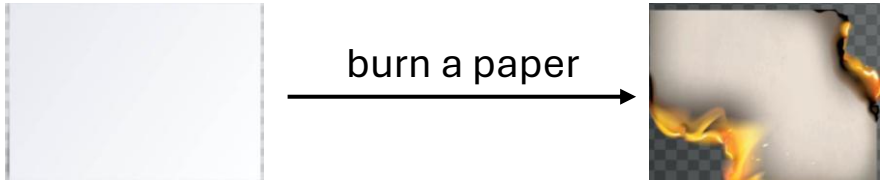
- We use a **unitary operator** to describe the evolution of a quantum state.
- In quantum computing, we use **unitary operations** to manipulate qubit(s)
  - Unitaries are invertible ⟹ Unitary operations are (conceptually) always **reversible**

- In contrast to classical computing, quantum computing **always** relies on **reversible computation**

Unitary Operation

Some physics (or philosophy?):

- In the real world, there are some operations that are **believed to be irreversible**:

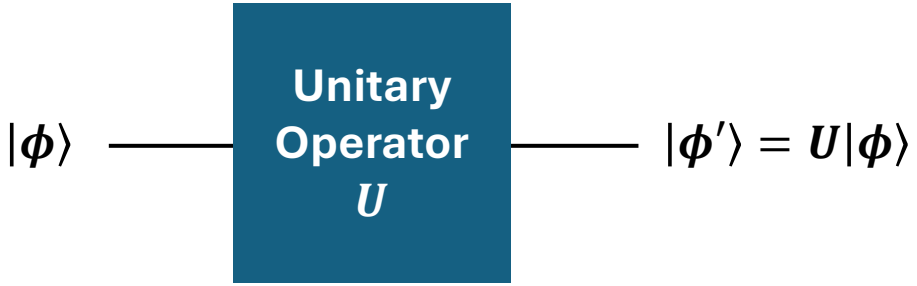


- **For an isolated quantum system, the evolution is unitary, so information is preserved in principle** – There must exists some unitary U **(in theory)** such that you can…

- …if you can **isolate the system** (pure state vs mixed state, will be introduced in the future)
- and **find the right unitary operator** (very hard)!

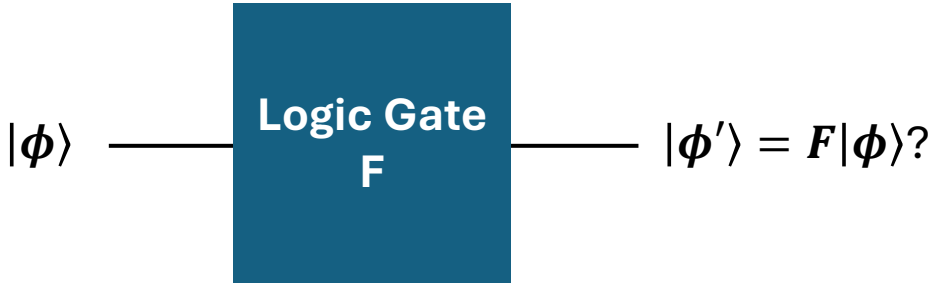
(source of images: Vector)

Quantum Logic Gates



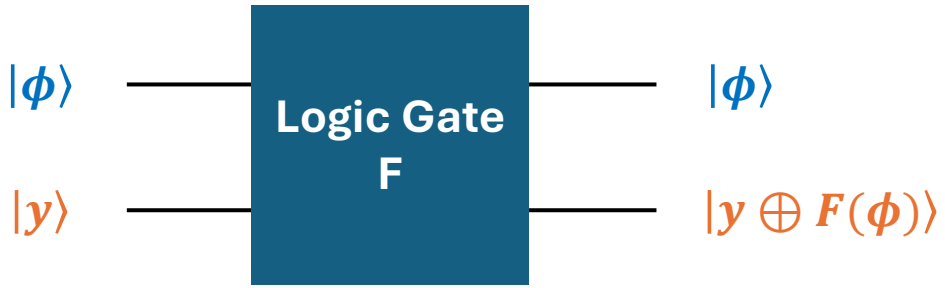
- Quantum computing relies on **unitary operations**
  - Any unitary matrix specifies a valid quantum gate/operation/algorithm
- Similar to classical computing, we use **logic gates** as the basic building blocks in quantum computing

Quantum Logic Gates



- Quantum computing relies on **unitary operations**
  - Any unitary matrix specifies a valid quantum gate/operation/algorithm
- Similar to classical computing, we use **logic gates** as the basic building blocks in quantum computing
  - The quantum logic gates should obey the rules in quantum mechanics
  - NOT gate is reversible
  - AND, NAND, OR, and XOR gates are irreversible

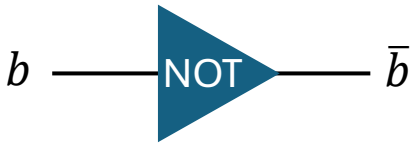
Quantum Logic Gates

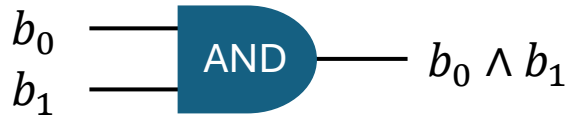


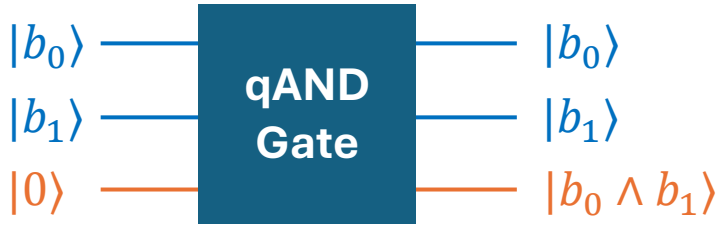
- Quantum computing relies on **unitary operations**
  - Any unitary matrix specifies a valid quantum gate/operation/algorithm
- Similar to classical computing, we use **logic gates** as the basic building blocks in quantum computing
  - The quantum logic gates should obey the rules in quantum mechanics
  - NOT gate is reversible
  - AND, NAND, OR, and XOR gates are irreversible – We **preserve the input** to make them reversible…
  - …and **store the result** using ancilla qubit(s) (or auxiliary qubits) which are usually set as 0

Quantum Logic Gates

- Examples (let's focus on the computational basis):

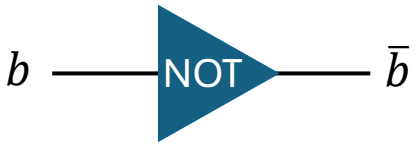


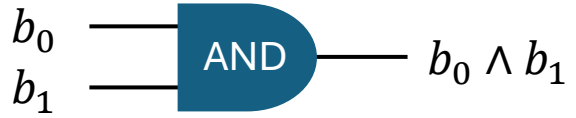




Quantum Logic Gates

- Examples (let's focus on the computational basis):

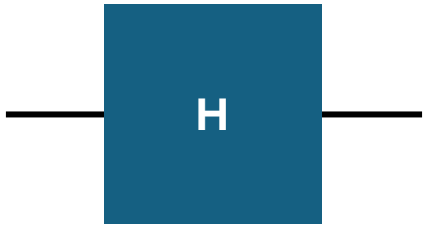




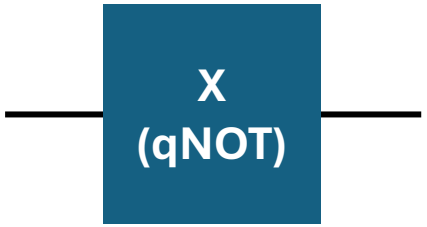
- How can we define the qOR and qNAND gates?

Quantum Gates

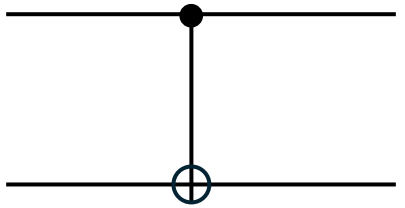
- More basic quantum gates:



Hadamard Matrix



Pauli-X
(The qNOT gate)



CNOT
(The Controlled NOT/X gate)

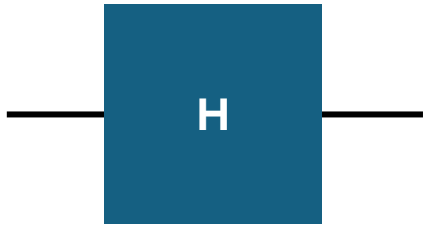
...

- Their matrix representations (in the computational basis):

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

Quantum Gates

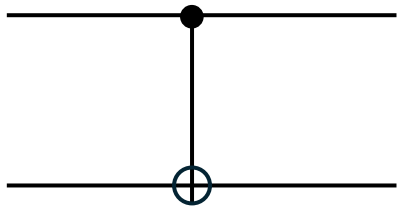
- Hadamard Matrix:



$$:= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- $\boldsymbol{H}|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|\boldsymbol{0}\rangle + |\boldsymbol{1}\rangle}{\sqrt{2}}$
- $\boldsymbol{H}|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|\boldsymbol{0}\rangle - |\boldsymbol{1}\rangle}{\sqrt{2}}$
- By Exercise of Week 2, $\boldsymbol{H^2 = I}$
- Turns $|0\rangle$ or $|1\rangle$ to "halfway" between $|0\rangle$ and $|1\rangle$

- CNOT:



$$:= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- CNOT$|\boldsymbol{0}\rangle|\boldsymbol{b}\rangle \rightarrow |\boldsymbol{0}\rangle|\boldsymbol{b}\rangle$
- CNOT$|\boldsymbol{1}\rangle|\boldsymbol{b}\rangle \rightarrow |\boldsymbol{1}\rangle|\boldsymbol{1} \oplus \boldsymbol{b}\rangle = |\boldsymbol{1}\rangle|\overline{\boldsymbol{b}}\rangle$
- Classical counterpart:
  - If the first bit = 0: do nothing;
  - Else: Flip the second bit

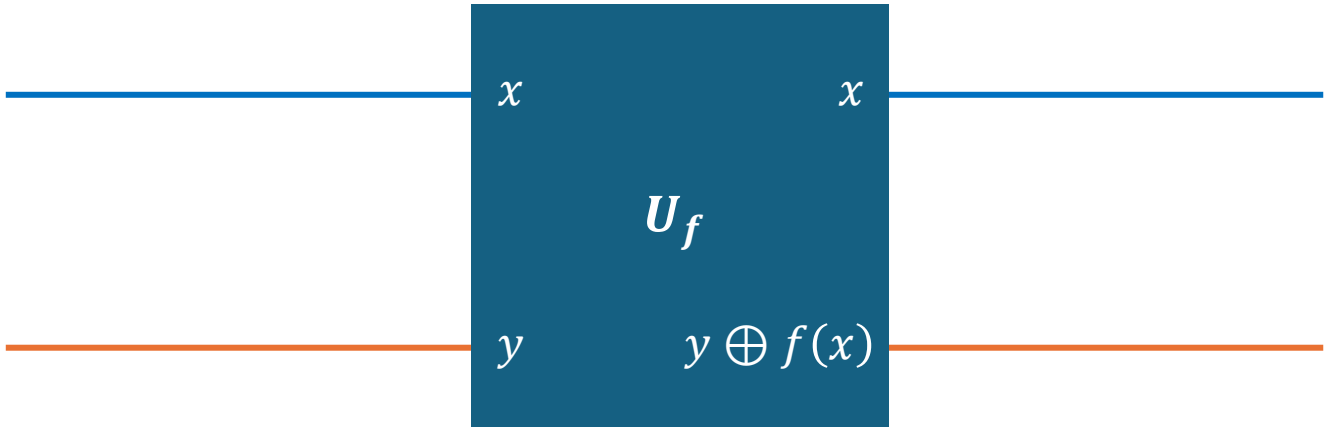
Quantum Gates

- Let f be a finite *computable* function.
  - There exists a circuit that implements f
  - Construct circuits using logic gates

- In Quantum Computing:
  - Construct a quantum circuit to compute f (using quantum logic gates)
  - Require reversible computation, while f may not be reversible

Quantum Circuits

- Generally, let $f: \{0,1\} \rightarrow \{0,1\}$ be a computable bit function.
- Define the quantum version of f as:

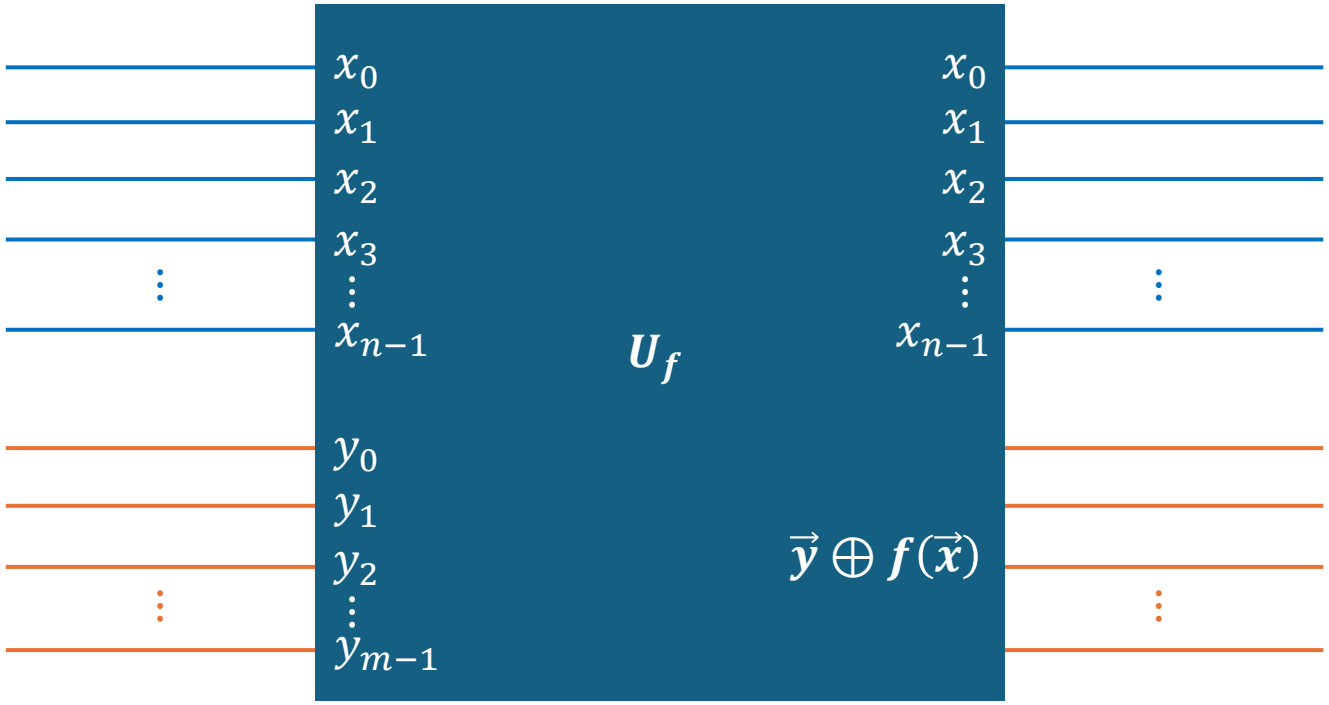


Quantum wire (or register) for storing input qubit

Quantum wire (or register) for storing output qubit

Quantum Circuits

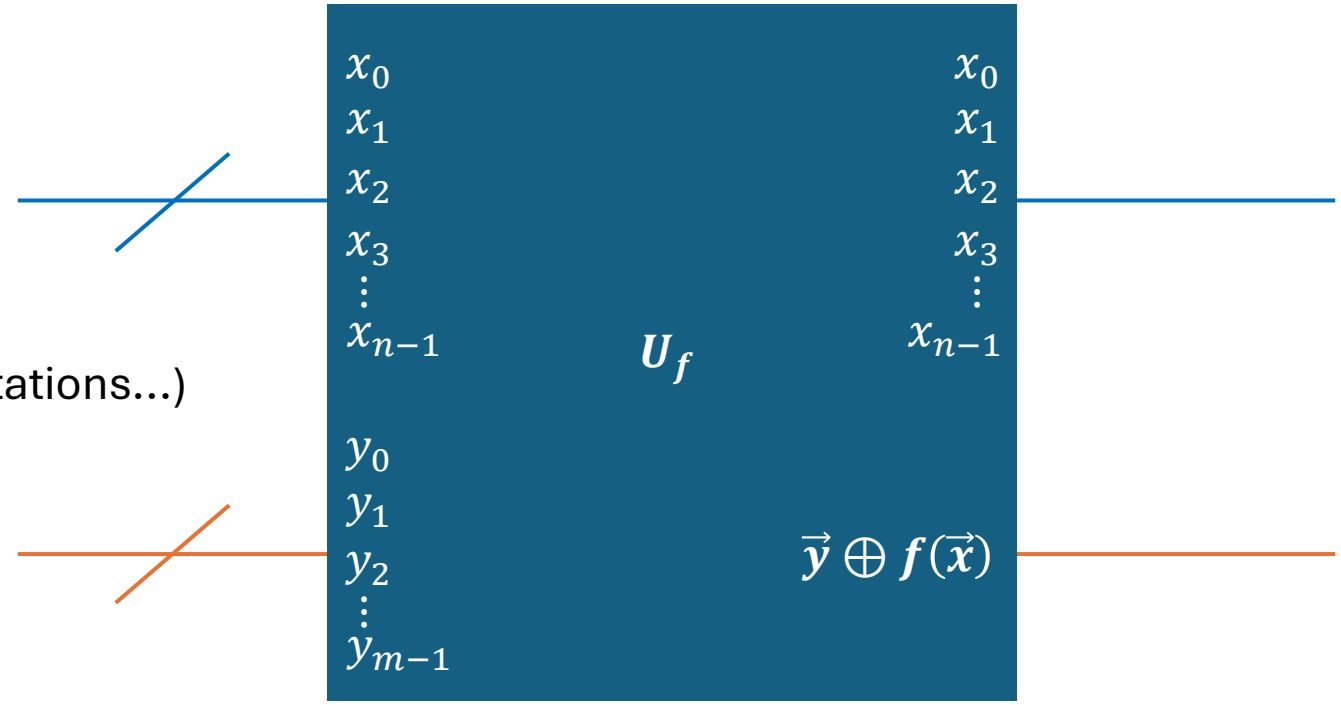
- More generally, let $f: \{0,1\}^n \rightarrow \{0,1\}^m$ be a computable function



Quantum Circuits

- More generally, let $f: \{0,1\}^n \rightarrow \{0,1\}^m$ be a computable function
  - **U_f is also a unitary**

(Simplify the notations…)



Quantum Circuits

- Let f be a finite *computable* function.
  - There exists a circuit that implements f
  - Construct circuits using logic gates

- In Quantum Computing:
  - Construct a quantum circuit to compute f (using quantum logic gates)
  - Require reversible computation, while f may not be reversible
  - **Generic transformation:** $f \rightarrow \boldsymbol{U_f}$ (make it unitary using ancilla qubits)

Quantum Circuits

- Let f be a finite *computable* function.
  - There exists a circuit that implements f
  - Construct circuits using logic gates

- In Quantum Computing:
  - Construct a quantum circuit to compute f (using quantum logic gates)
  - Require reversible computation, while f may not be reversible
  - **Generic transformation:** $f \rightarrow \boldsymbol{U_f}$ (make it unitary using ancilla qubits)

- **Any classical algorithm (circuit) can be simulated by a quantum algorithm (circuit)**
  - Classical algorithms/circuits are built from classical logic gates
  - Classical logic gates can be simulated using reversible quantum logic gates
  - Quantum logic gates can be composed into quantum algorithms/circuits

Evaluation on Superposition

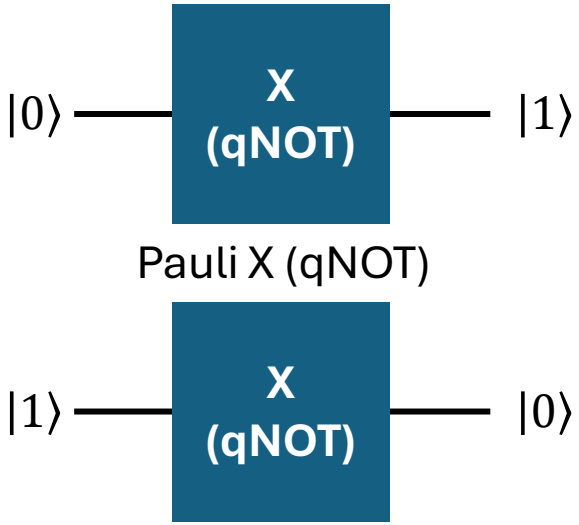
- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

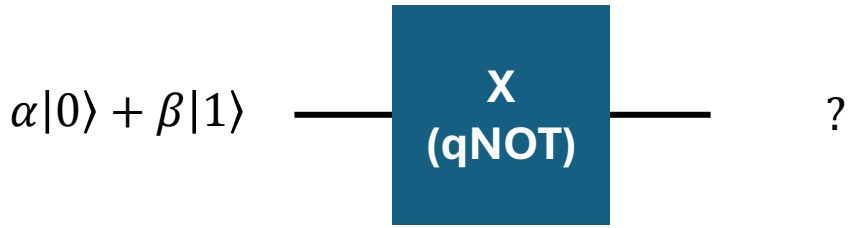
- Quantum gates (Unitaries) operate on superposition: **Linearity**

Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

- Quantum gates (Unitaries) operate on superposition: **Linearity**

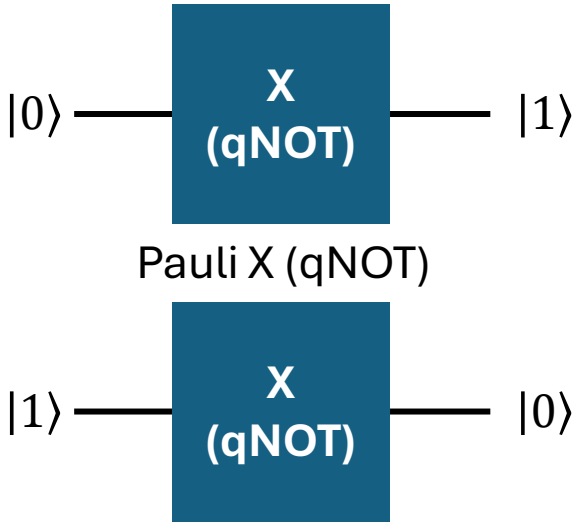


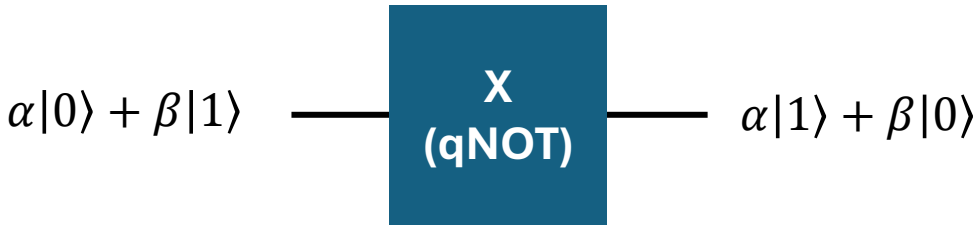


Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

- Quantum gates (Unitaries) operate on superposition: **Linearity**

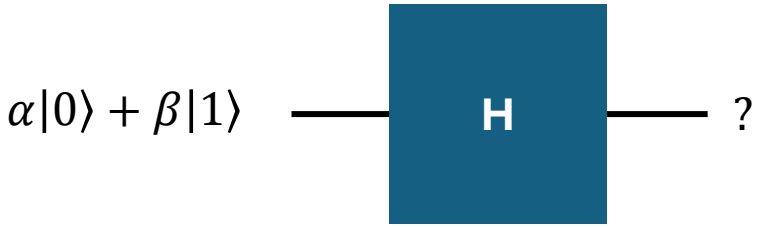




Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

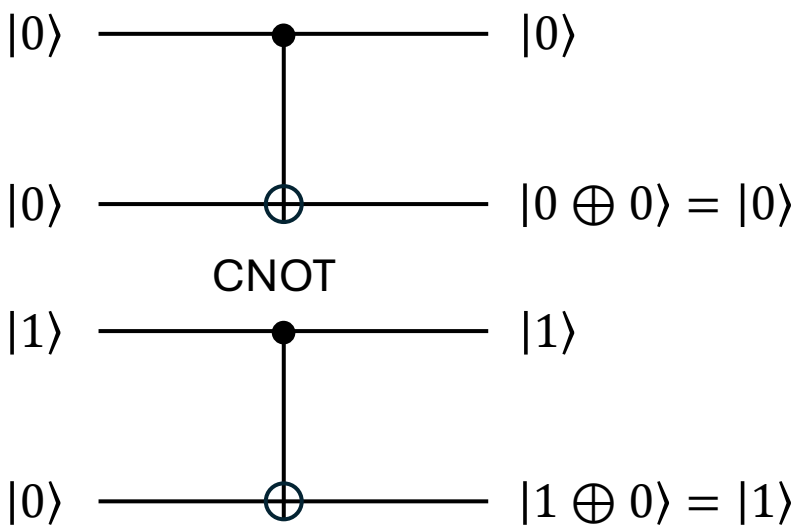
- Quantum gates (Unitaries) operate on superposition: **Linearity**

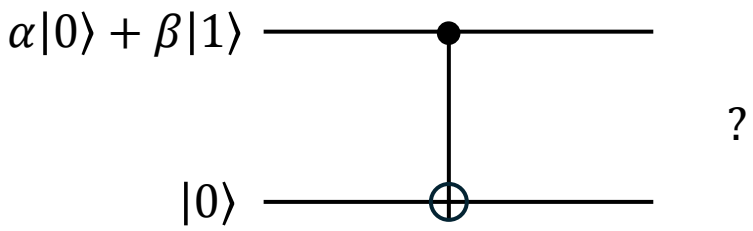


Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

- Quantum gates (Unitaries) operate on superposition: **Linearity**



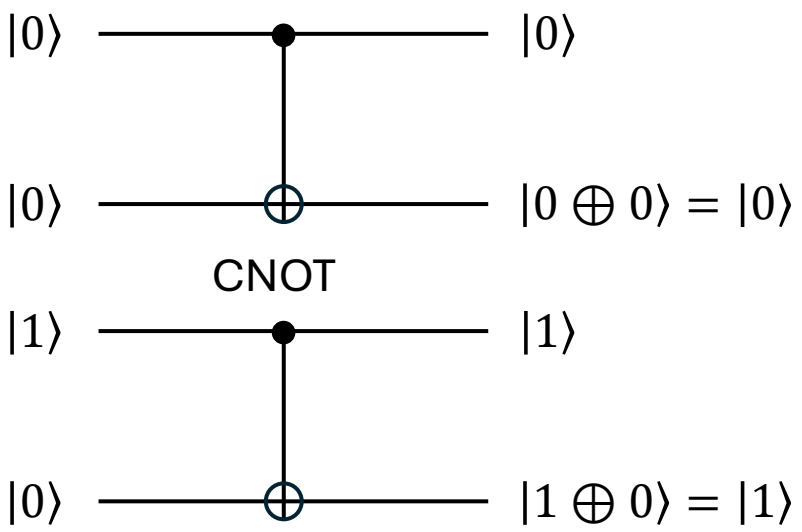


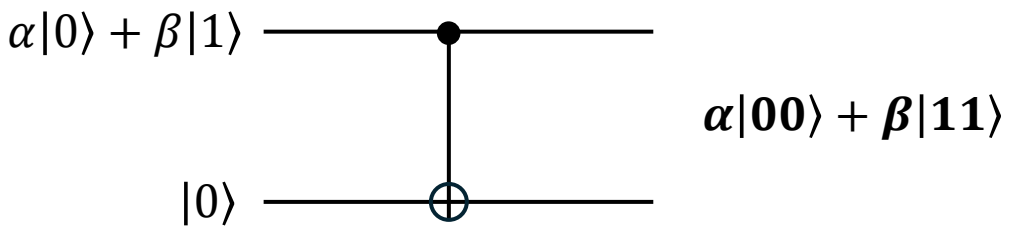
?

Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

- Quantum gates (Unitaries) operate on superposition: **Linearity**

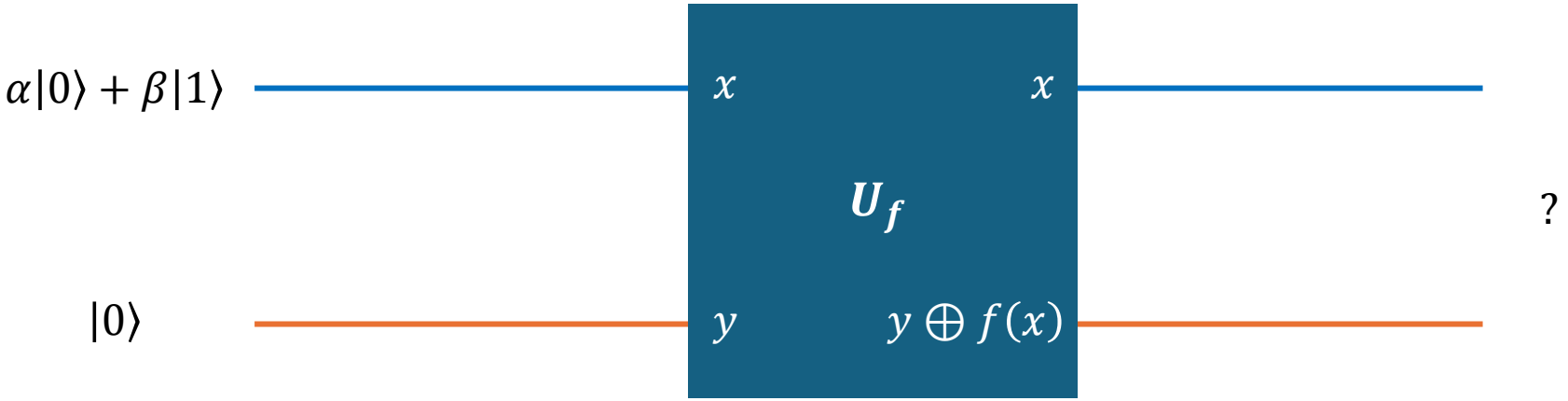




Evaluation on Superposition

- Any quantum gate is a unitary operator
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

- Quantum gates (Unitaries) operate on superposition: **Linearity**



Summary

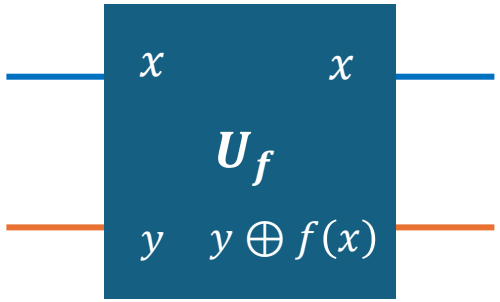
- Quantum gates are described by unitaries
  - Any unitary also specifies a valid quantum gate
- Basic quantum gates: Hadamard, Pauli-X (NOT), CNOT, …
- Make a classical computable function unitary $f \rightarrow \boldsymbol{U_f}$
  - Any classical algorithm can be simulated by quantum computers
- Evaluation on superposition
  - View any quantum gate as a unitary linear operator (matrix)
  - Quantum gates act on superpositions according to linearity

Unitary Operations

- A quantum gate is a unitary operator (⇔ A unitary represents some quantum gate)
  - A unitary operator has **linearity:** $\boldsymbol{U}(c_1\boldsymbol{v_1} + c_2\boldsymbol{v_2}) = c_1\boldsymbol{U}\boldsymbol{v_1} + c_2\boldsymbol{U}\boldsymbol{v_2}$

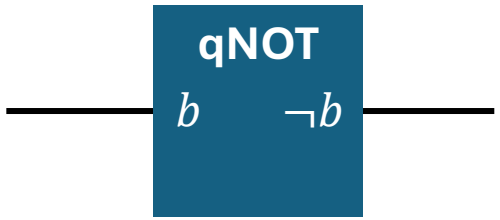
- Quantum gates operate on superposition: **Linearity**
  - View any quantum gate as a unitary linear operator (matrix)
  - Quantum gates act on superpositions according to linearity

- Make a classical computable function unitary $f \rightarrow \boldsymbol{U_f}$
  - Use **input qubits** and **ancilla qubits** to make it invertible
  - Any classical algorithm can be simulated by quantum computers

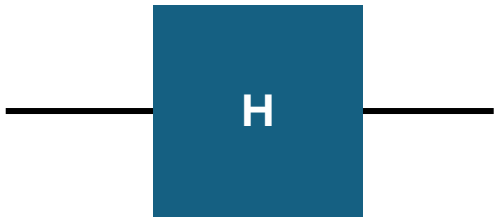


Unitary Operations on Single-Qubit States

- Single-qubit unitary:
  - Examples: qNOT, Hadamard transform, …



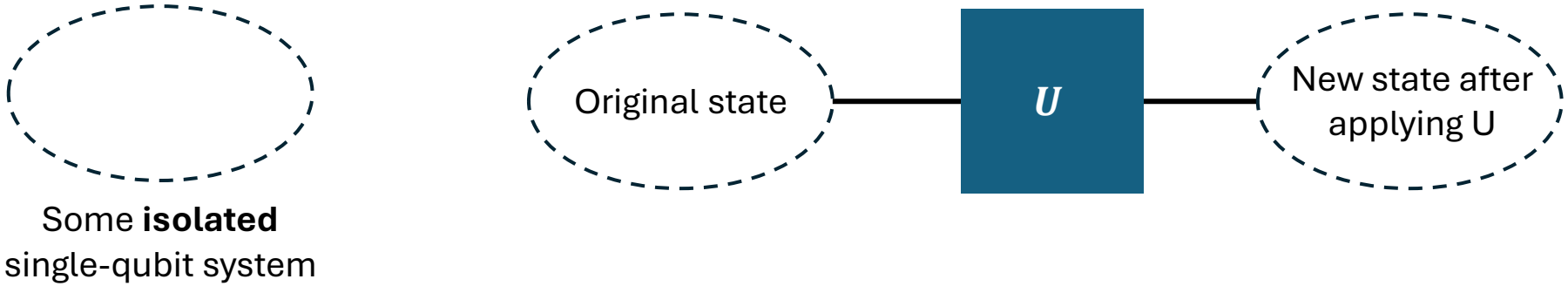
$$\mathbf{qNOT}(|b\rangle) \rightarrow |\neg b\rangle$$



$$\boldsymbol{H}(|b\rangle) \rightarrow \frac{|0\rangle + (-1)^b|1\rangle}{\sqrt{2}}$$

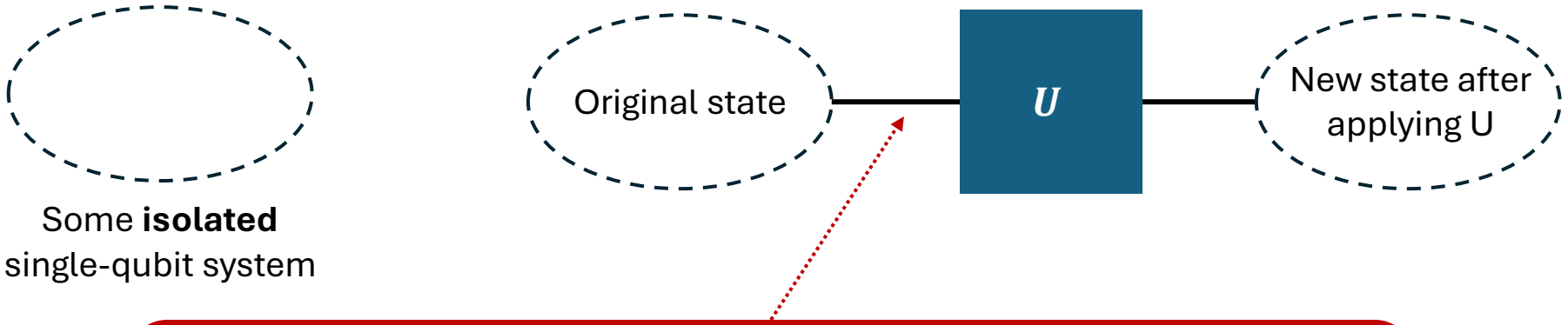
Unitary Operations on Single-Qubit States

- Single-qubit unitary:
  - Examples: qNOT, Hadamard transform, …



Unitary Operations on Single-Qubit States

- Single-qubit unitary:
  - Examples: qNOT, Hadamard transform, …

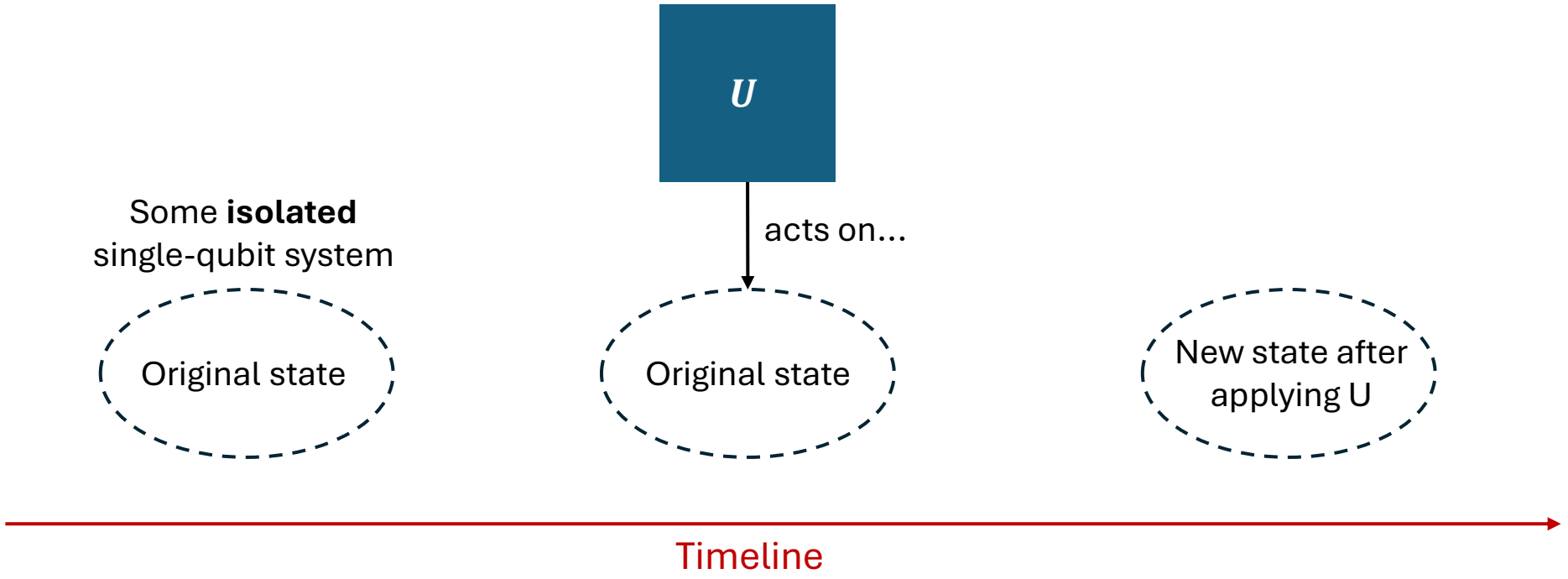


Misunderstand: The “wire” here **does not** represent a real wire!

Instead, it just visually **tracks the state of the system** through time.

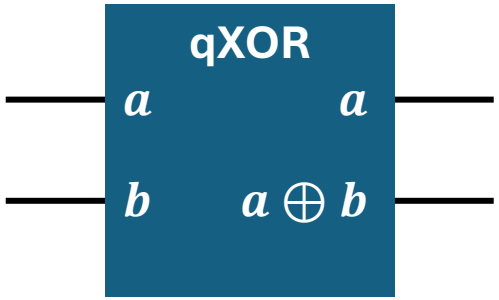
Unitary Operations on Single-Qubit States

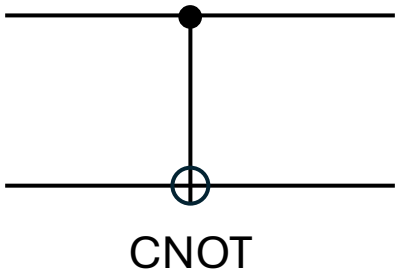
- Single-qubit unitary:
  - Examples: qNOT, Hadamard transform, …



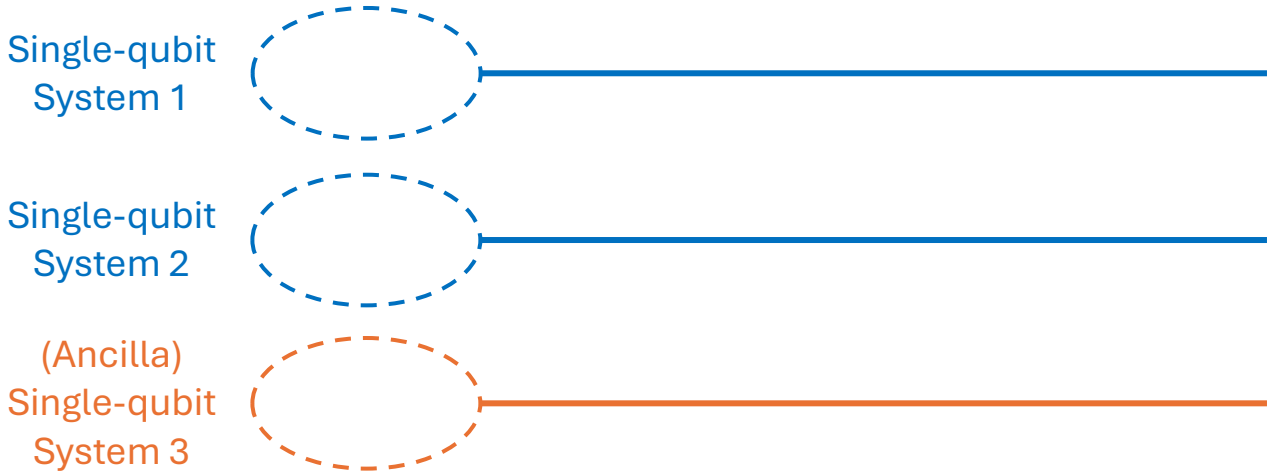
Unitary Operations on Multi-Qubit States

- Multi-qubit unitary:
  - Examples: qXOR, CNOT, …

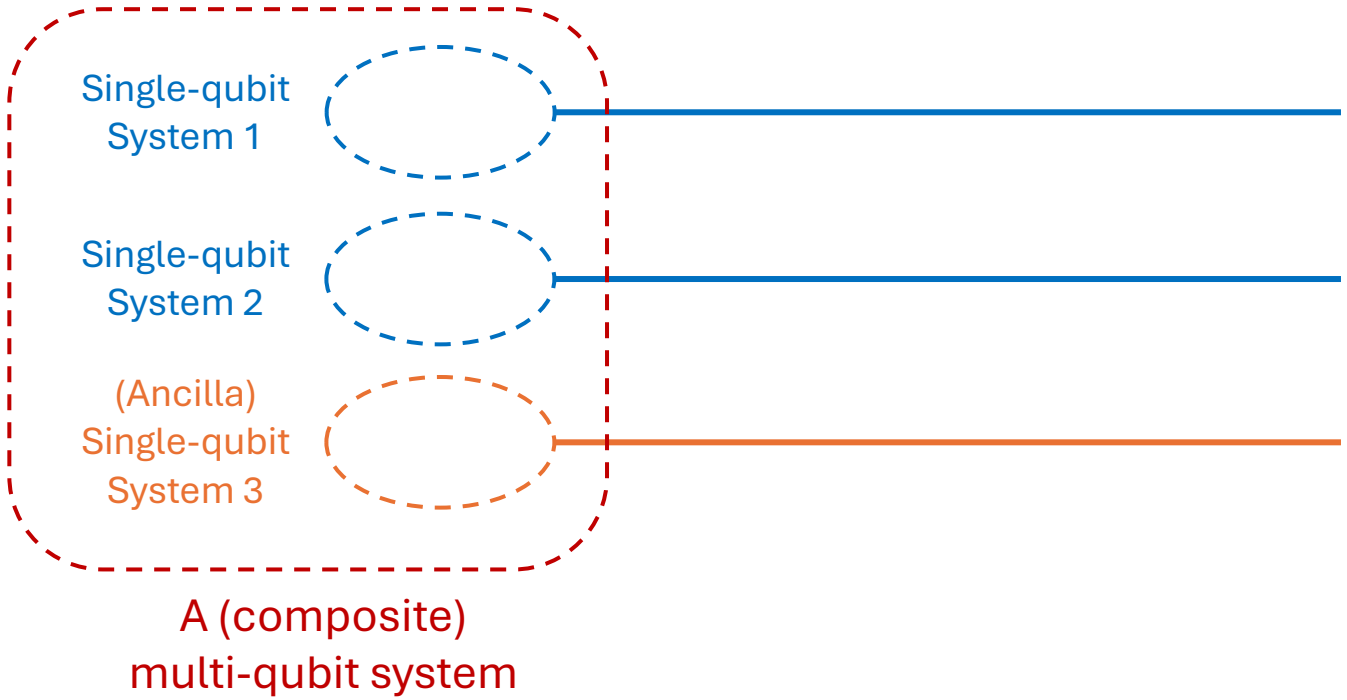




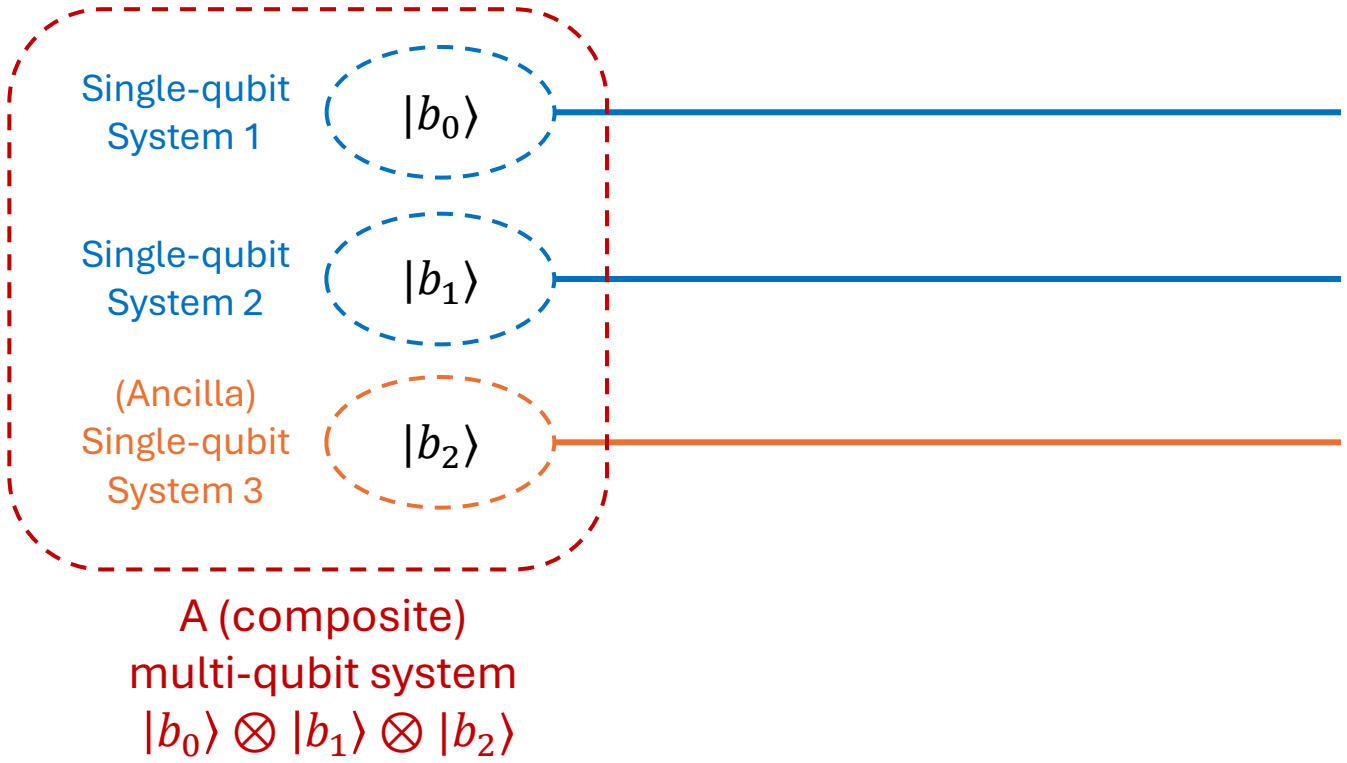
Unitary Operations on Multi-Qubit States



Unitary Operations on Multi-Qubit States

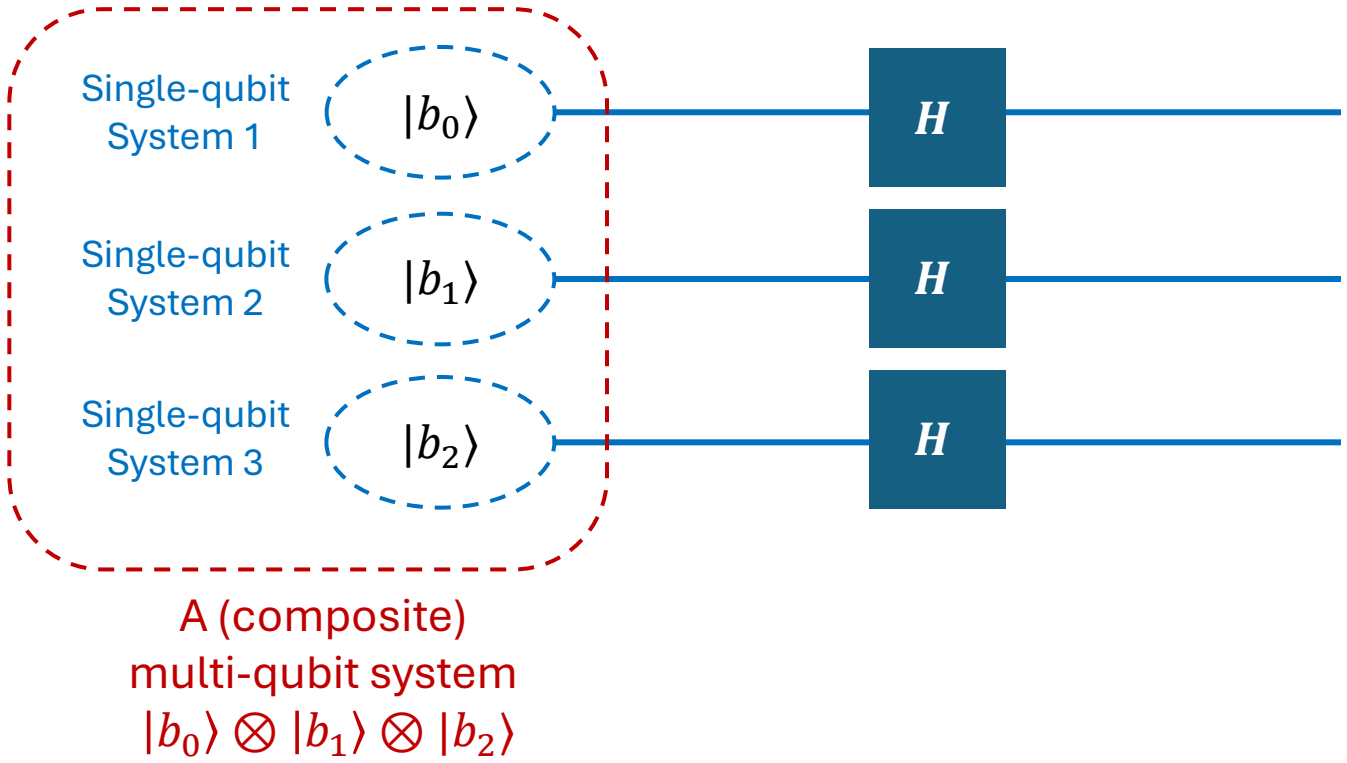


Unitary Operations on Multi-Qubit States



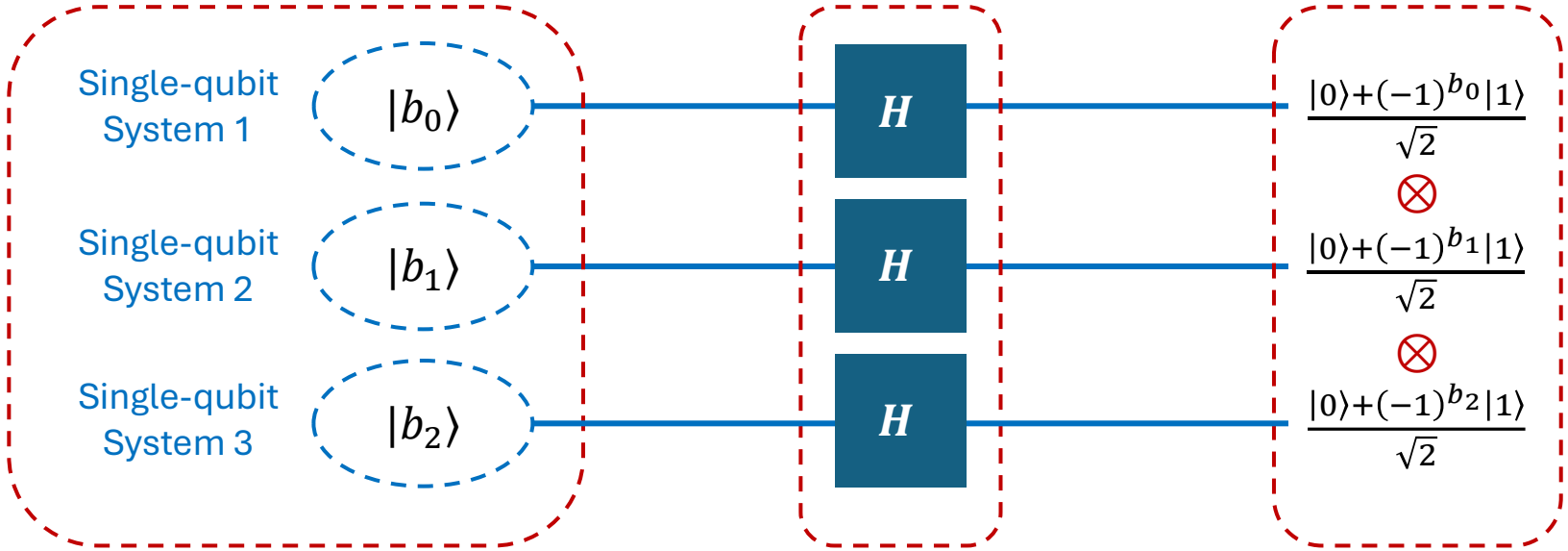
Unitary Operations on Multi-Qubit States

- Multi-qubit unitary:
  - Examples: **Parallel action of Hadamard gates**…



Unitary Operations on Multi-Qubit States

- Multi-qubit unitary:
  - Examples: qXOR, CNOT, **Parallel action of Hadamard gates**…



A (composite) multi-qubit system $|b_0\rangle \otimes |b_1\rangle \otimes |b_2\rangle$

$H^{\otimes 3}$ (Parallel H acts on 3-qubit system)

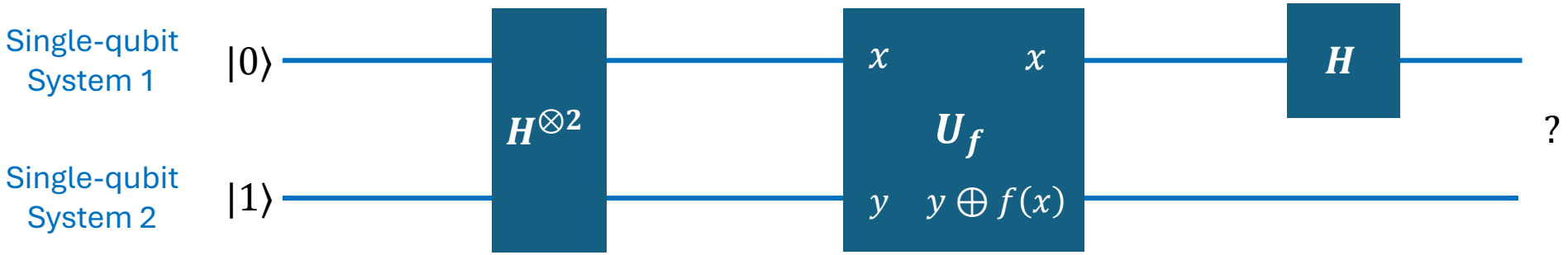
The state of the composite system after applying $H^{\otimes 3}$

Unitary Operations on Multi-Qubit States

- Multi-qubit unitary:
  - **Parallel action of Hadamard gates…**

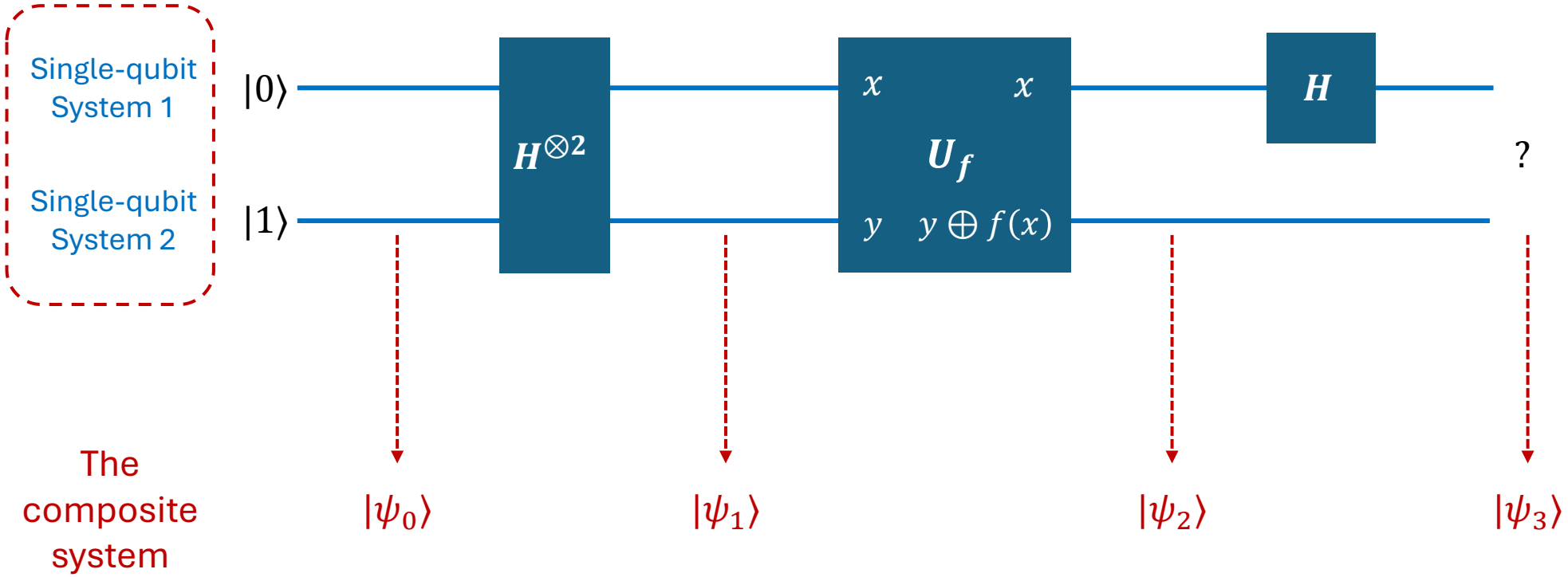
Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)



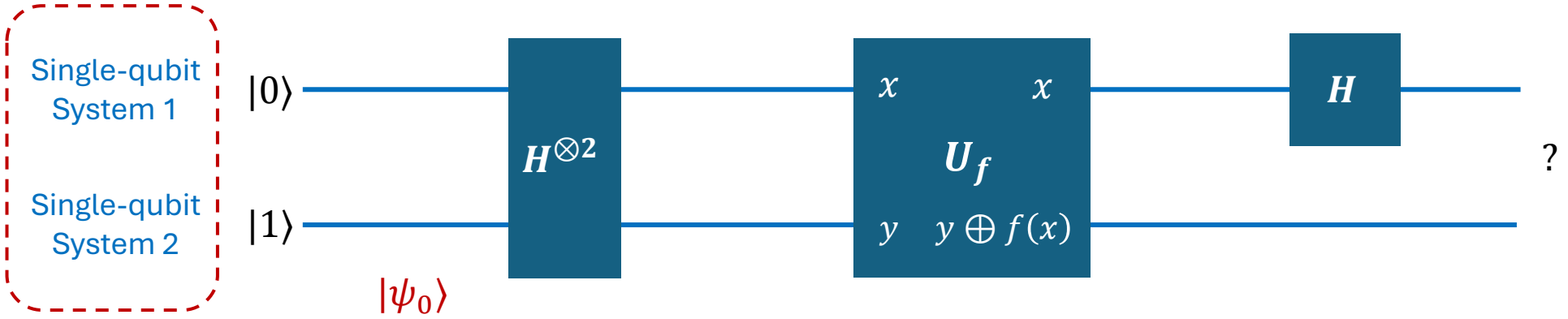
Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)



Deutsch's Algorithm

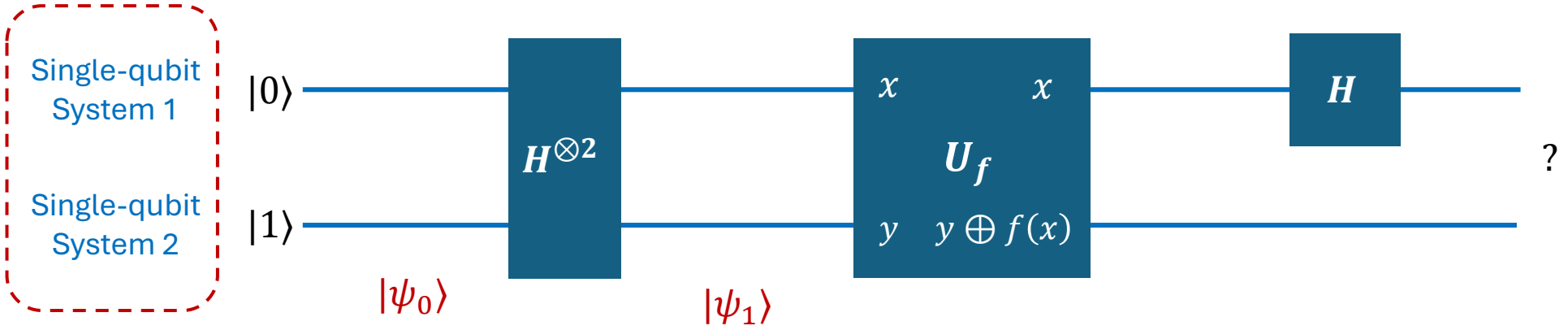
- Let f be a bit function…
- (Do it on the board)



The composite system $|\psi_0\rangle = |01\rangle = |0\rangle \otimes |1\rangle$

Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)

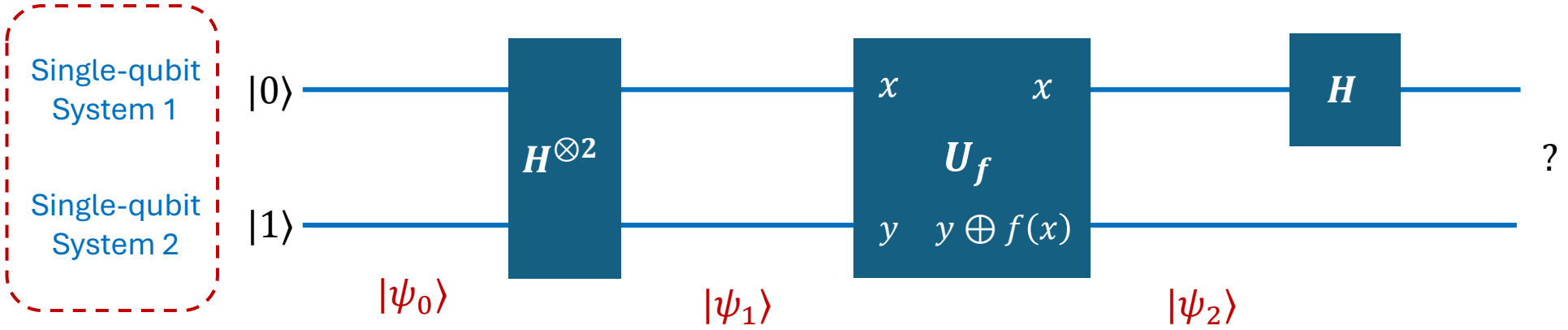


The composite system

$$|\psi_1\rangle = H^{\otimes 2}|\psi_0\rangle = \left(\frac{|0\rangle + |1\rangle}{2}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{2}\right)$$

Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)

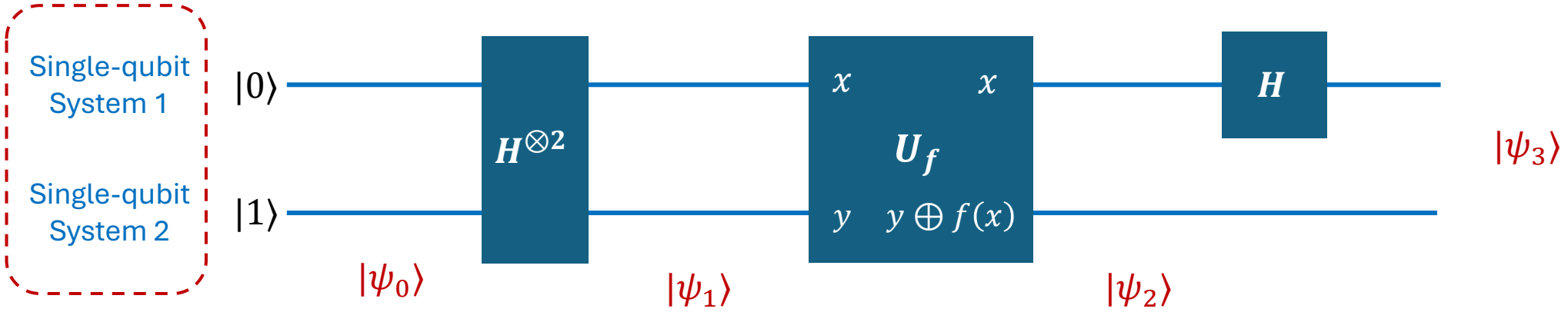


The composite system

$$|\psi_2\rangle = U_f|\psi_1\rangle = \left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{2}\right) \otimes \left((-1)^{f(0)}\left(\frac{|0\rangle - |1\rangle}{2}\right)\right)$$

Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)

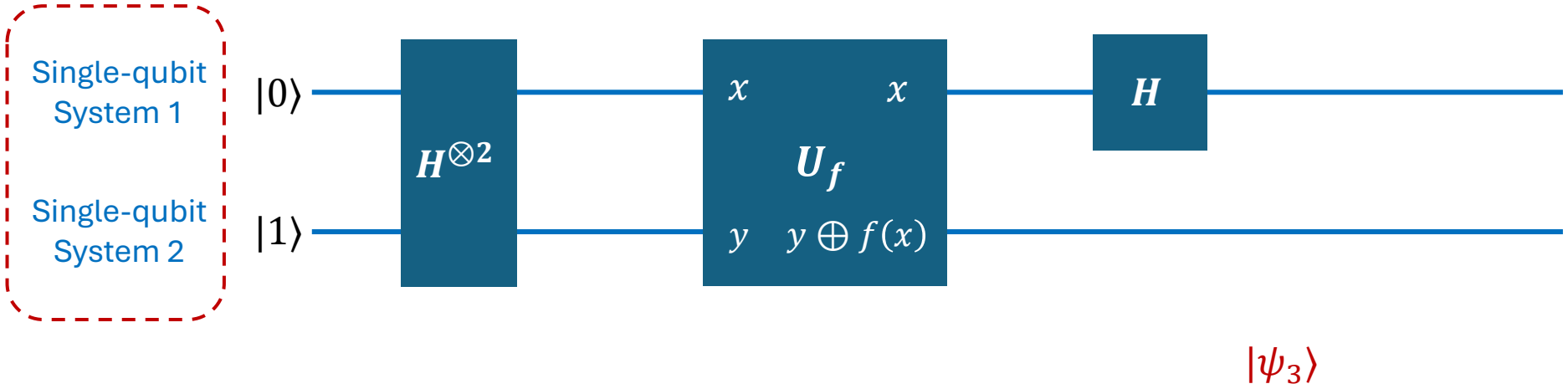


The composite system

$$|\psi_3\rangle = (\boldsymbol{H} \otimes \boldsymbol{I})|\psi_2\rangle = (|f(0) \oplus f(1)\rangle) \otimes \left((-1)^{f(0)}\left(\frac{|0\rangle - |1\rangle}{2}\right)\right)$$

Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)



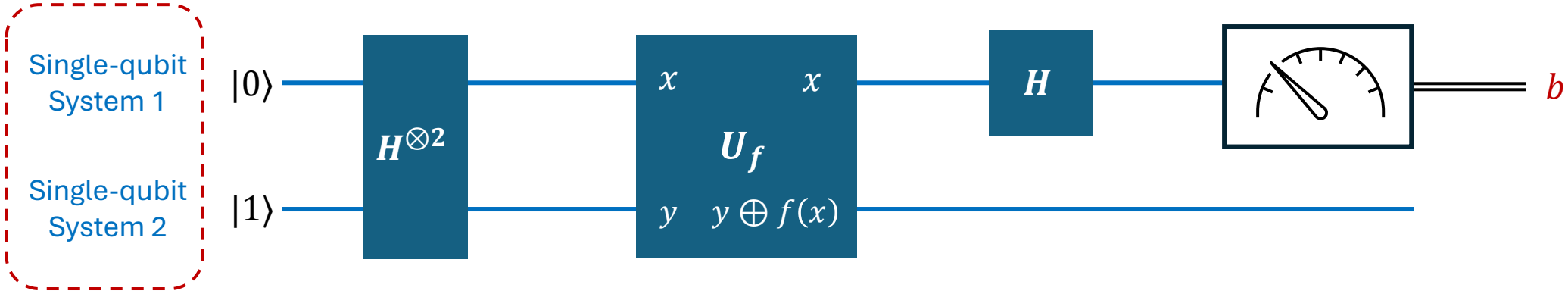
The composite system

$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

(We just let $(-1)^{f(0)} = \pm$, which does not change the measurement outcome)

Deutsch's Algorithm

- Let f be a bit function…
- (Do it on the board)



The composite system

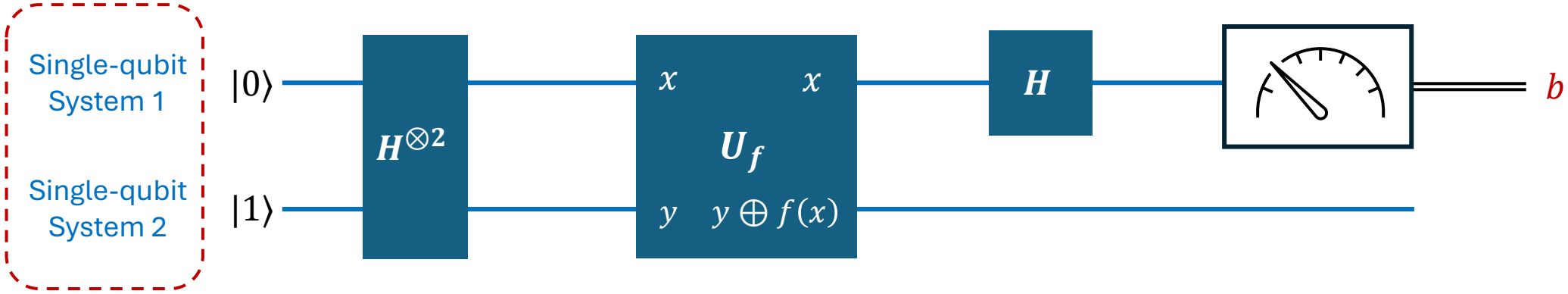
$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

Measure the first system

$$b = f(0) \oplus f(1)$$

Deutsch's Algorithm

- Let f be a bit function...
- (Do it on the board)



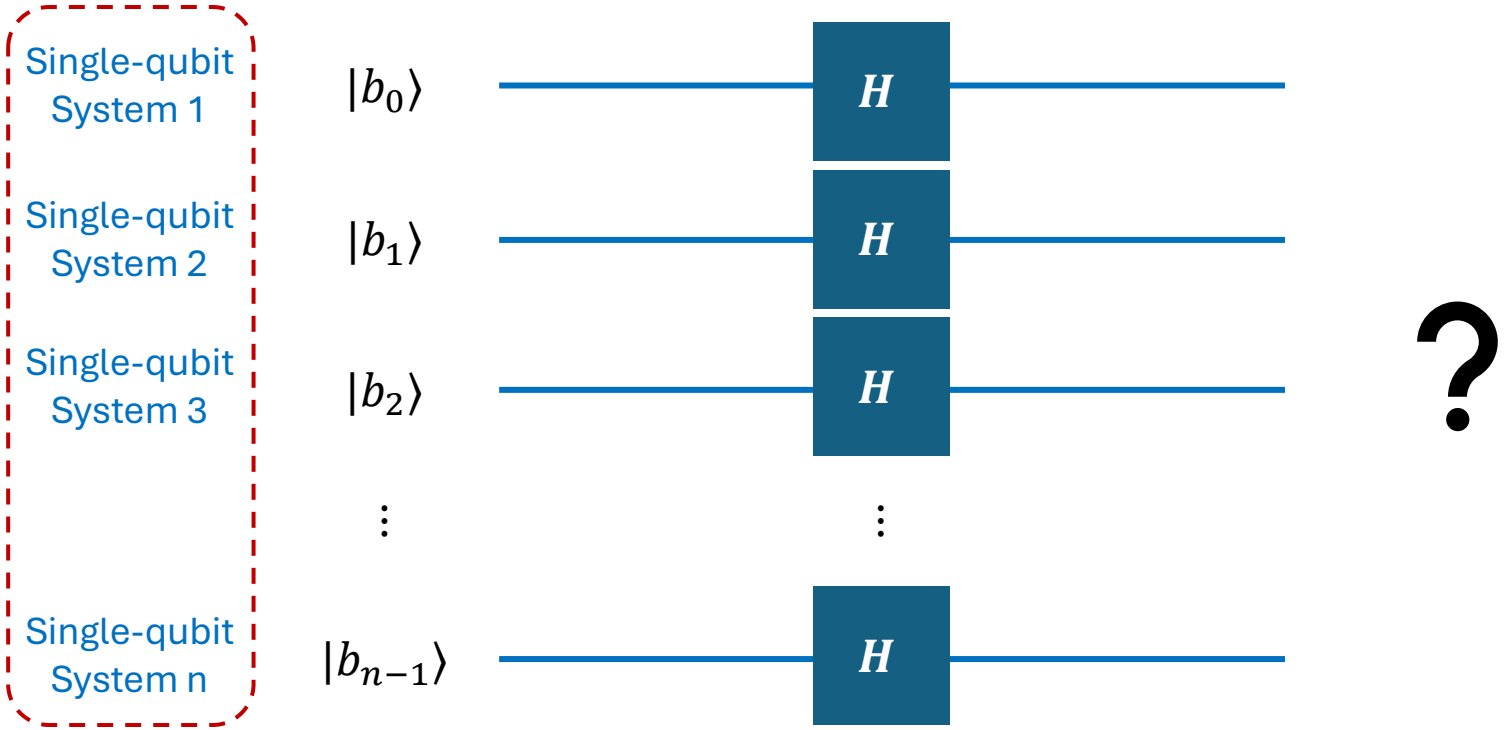
$|\psi_3\rangle$

The composite system

$|\psi_3\rangle$ ═══ Measure the first system ═══ $b = f(0) \oplus f(1)$

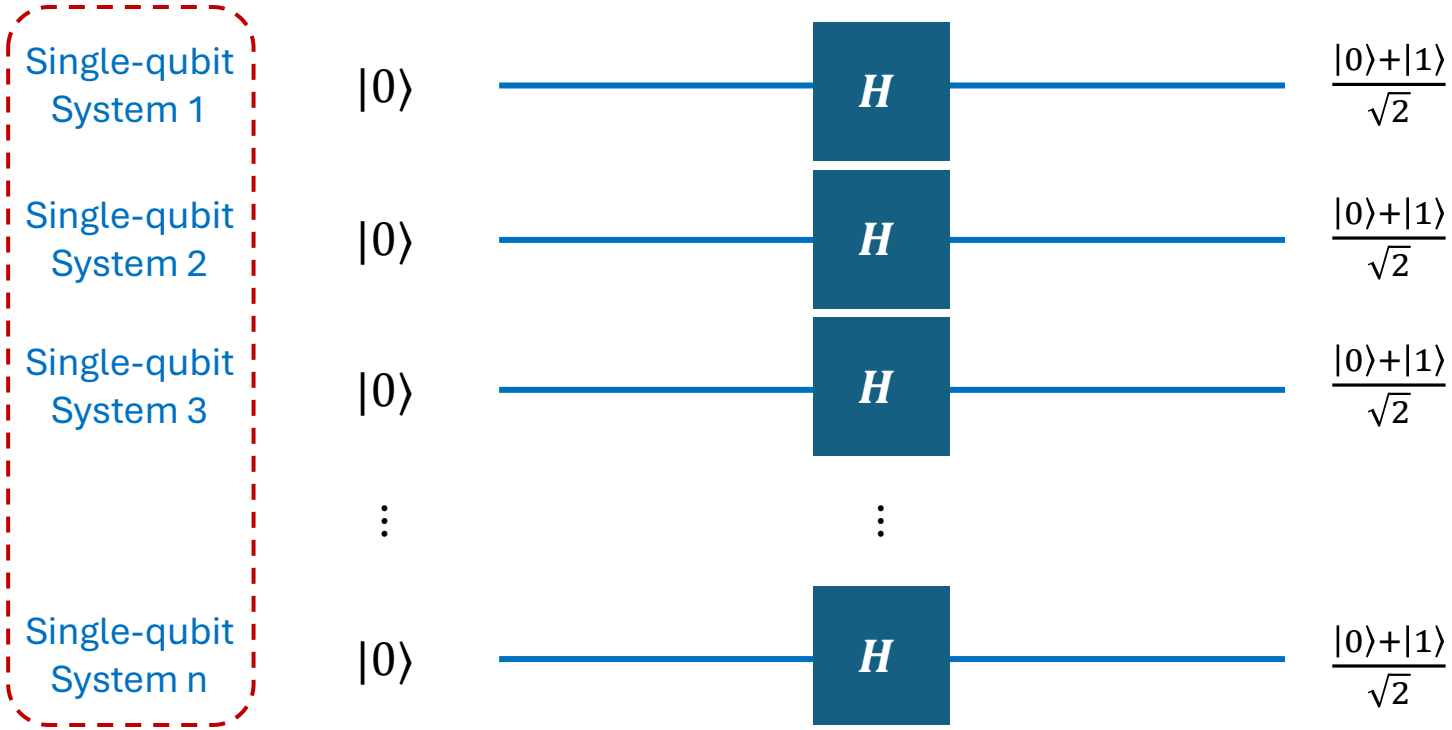
One (quantum) evaluation on f and get $f(0) \oplus f(1)$

Parallel Hadamard Gates



$|b_0\rangle \otimes |b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_{n-1}\rangle$

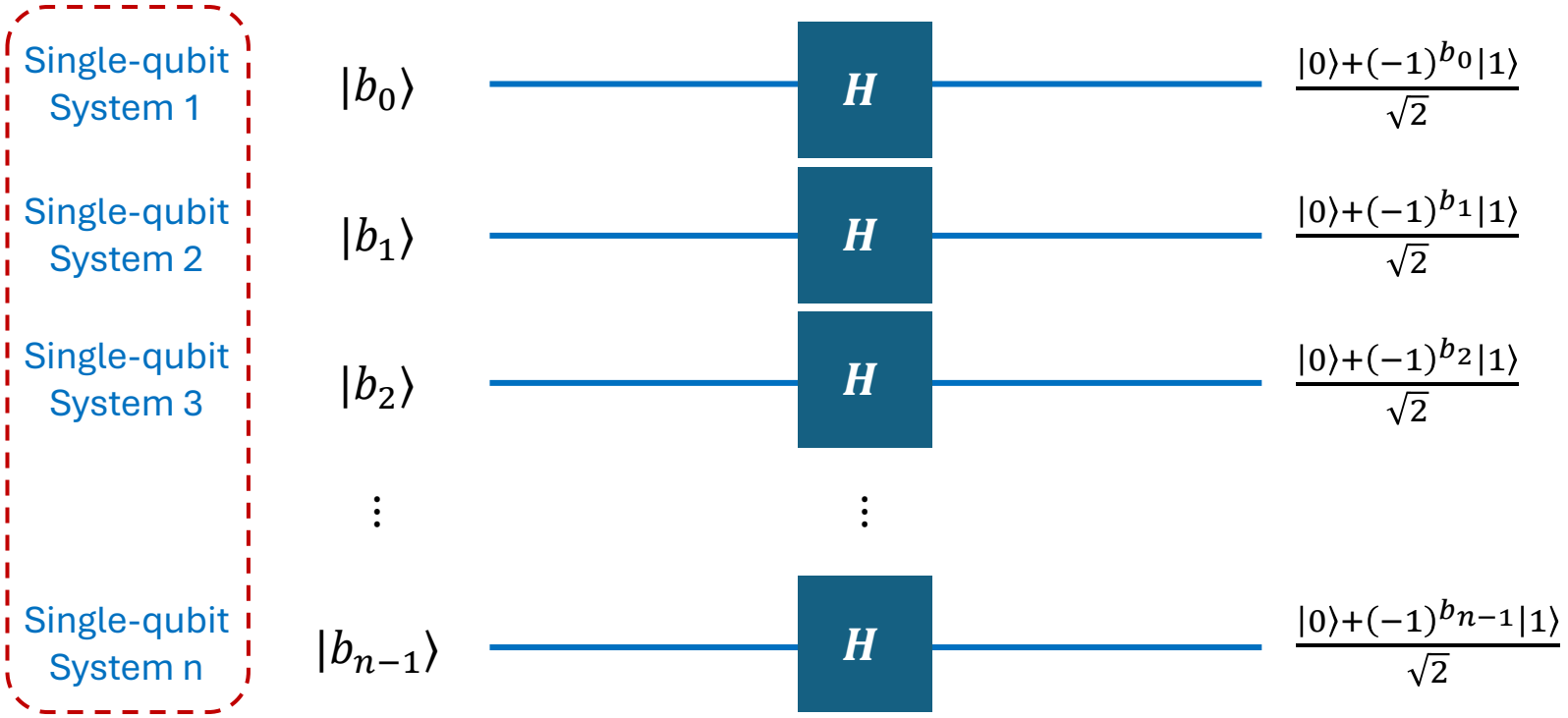
Parallel Hadamard Gates



$$= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

$$|b_0\rangle \otimes |b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_{n-1}\rangle$$

Parallel Hadamard Gates



$|b_0\rangle \otimes |b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_{n-1}\rangle$

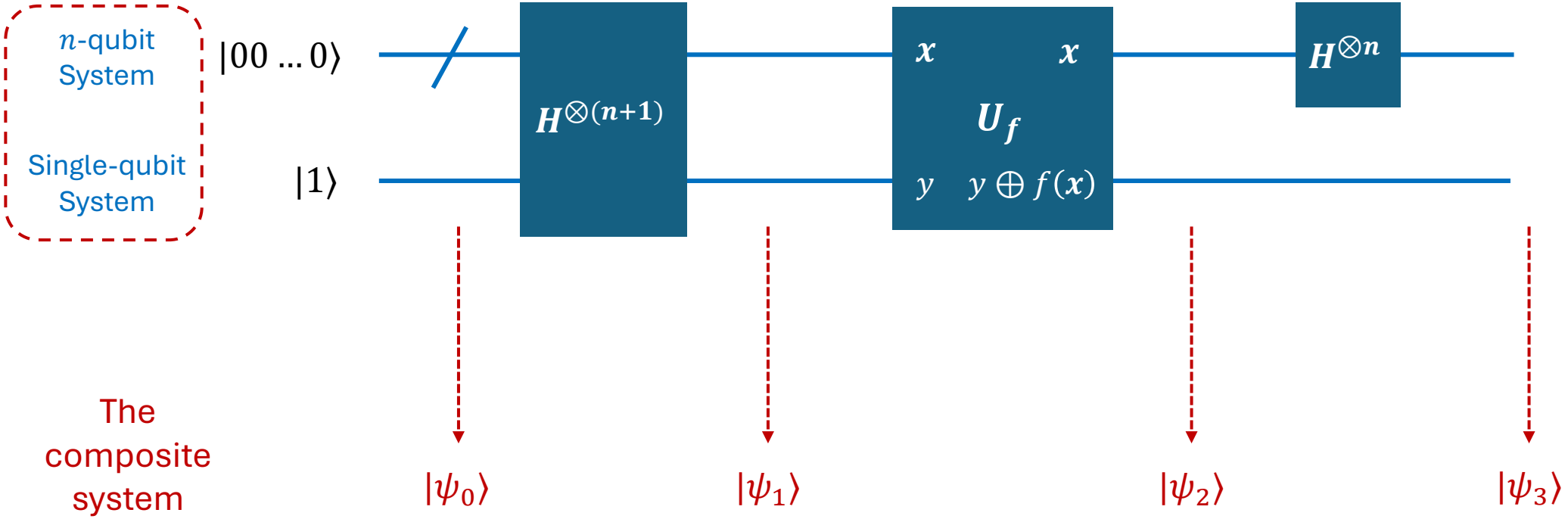
Let $\boldsymbol{b} \coloneqq b_{n-1}b_{n-2} \ldots b_0$ be the classical bit string

$$\boldsymbol{H}^{\otimes n}|\boldsymbol{b}\rangle = \sum_{\boldsymbol{x}\in\{0,1\}^n} \frac{(-1)^{\boldsymbol{x}^T\boldsymbol{b}}}{\sqrt{2^n}}|\boldsymbol{x}\rangle$$

The Deutsch-Jozsa Algorithm

- Let $f: \{0,1\}^n \rightarrow \{0,1\}$ be a bit function…
- (Do it on the board)

Homework



References

- **[NC00]** *Quantum Computation and Quantum Information*.
  - Sections 1.3.1 – 1.3.5 (no-cloning theorem), 1.4.1 – 1.4.3
- **[KLM07]** *An Introduction to Quantum Computing*.
  - Sections 6.2, 6.3, and 6.4
- **[RP11]** *Quantum Computing: A Gentle Introduction*.
  - Section 7.3.1

Homework

- **First homework set** (will be announced in the Moodle system soon)**:**
  - Submit your **handwritten solutions (photos, scanned pdf, etc.), or typeset in LaTeX**
  - Solutions may be found in the textbook…
  - But please **include all intermediate equations and their explanation**
  - **DDL: May 5th, 2026 at 23:59 (The last second of the next Tuesday)**

- **Homework criteria**
  - No grade
  - **To join the final exam, you must finish all homework**
  - If you get stuck with some homework questions, then indicate the place you get stuck and the reasons (e.g., you don’t know how to calculate an equation, …)