Quantum Oracles

CPSC 4470/5470

Introduction to Quantum Computing

Instructor: Prof. **Yongshan Ding**

Computer Science, Applied Physics, Yale Quantum Institute

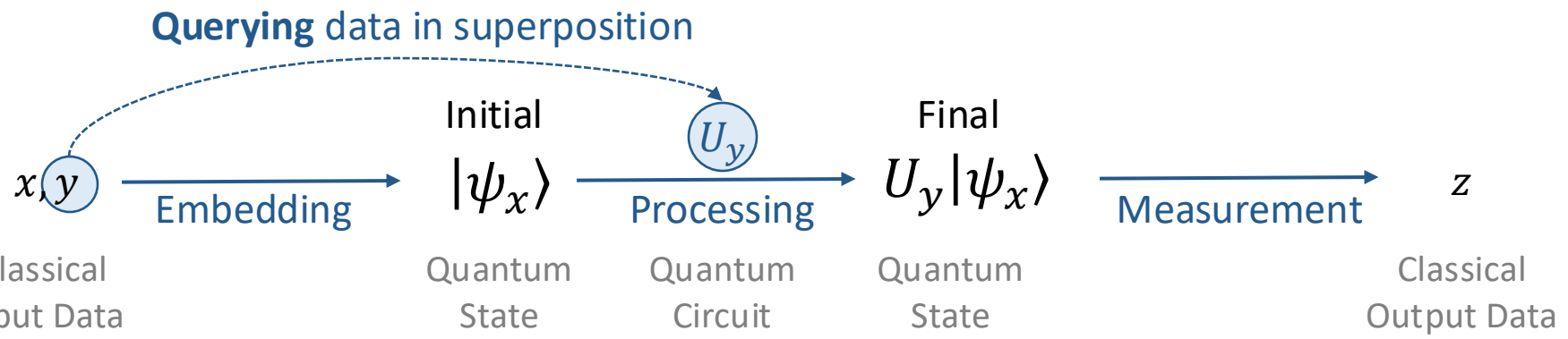
Data Input/Output for a Quantum Computer

Input (classical → quantum):

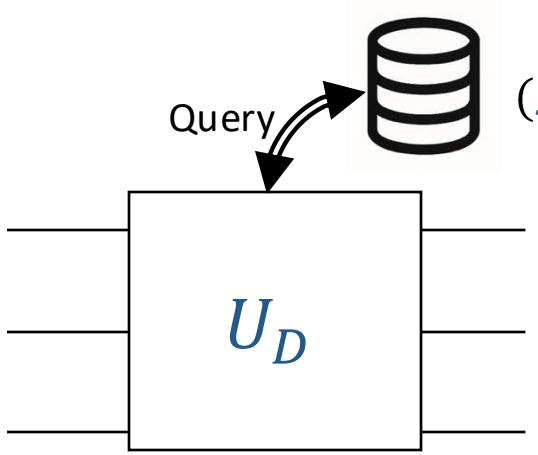
- How do we feed a **quantum computer** some **classical data** like words, images, sounds?

Output (quantum → classical):

- How much classical information can we extract from a quantum computer?



Quantum Oracles



$(x, y_x) \in D$: Dataset with (key, value) pairs

Initial

$$\sum_x \alpha_x |x\rangle = |\psi_x\rangle$$

Quantum State

Final

$$|\psi_y\rangle = \sum_x \alpha_x |y_x\rangle$$

Quantum State

Example: $|x\rangle \rightarrow |f(x)\rangle$

$$\sum_x \alpha_x |x\rangle \rightarrow \sum_x \alpha_x |f(x)\rangle$$

"Evaluating **Boolean function** in superposition."

(key, value) = $(x, f(x))$

Equivalent: Treating function f as RAM.

$$|i\rangle \rightarrow |m_i\rangle$$

$$\sum_i \alpha_i |i\rangle \rightarrow \sum_i \alpha_i |m_i\rangle$$

"Accessing **memory** in superposition."

(key, value) = (i, m_i)

Bit Oracle

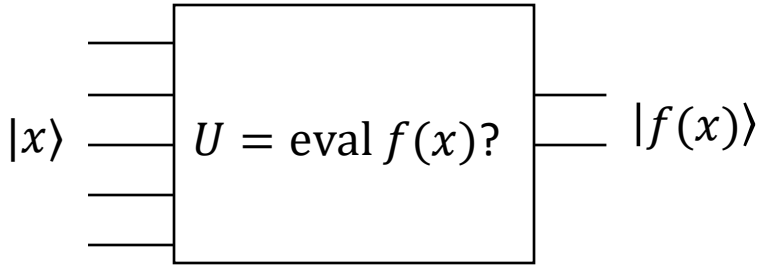
Query: $|x\rangle \to |f(x)\rangle$

$$\sum_x \alpha_x |x\rangle \to \sum_x \alpha_x |f(x)\rangle$$

"Evaluating Boolean function in superposition."

How to implement function f as a **quantum circuit**?

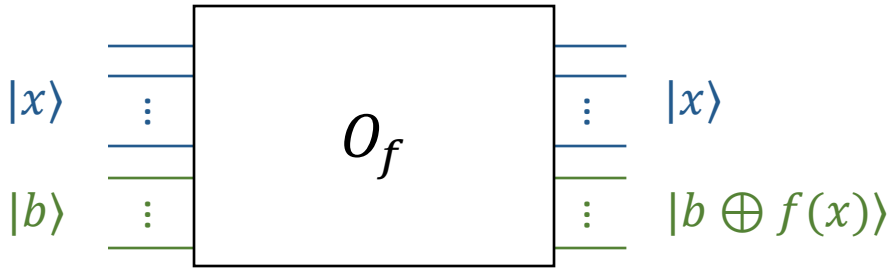
$$f: \{0,1\}^m \to \{0,1\}^n$$



What if f is **not reversible**? E.g., $m > n$.

Bit Oracle (XOR oracle) O_f:

"Keeping x, and storing $f(x)$ out of place."



When $b = 0$ and queried in superposition:

$$\sum_x \alpha_x |x\rangle|0\rangle \to \sum_x \alpha_x |x\rangle|f(x)\rangle$$

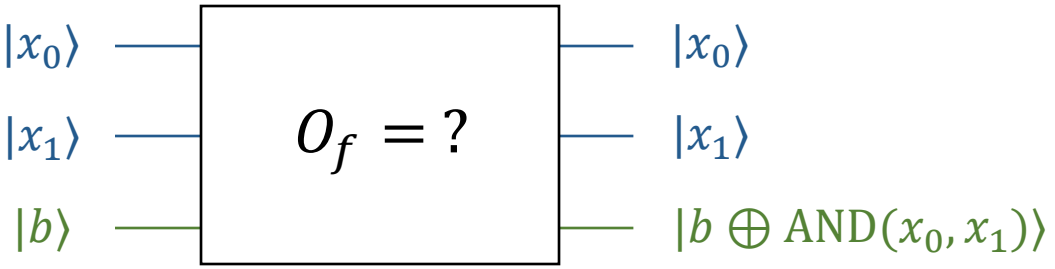
Bit Oracle

Example Bit Oracle O_f:

$$f(x) = \mathrm{AND}(x_0, x_1)$$

Key: x	**Val:** $f(x)$
0,0	0
0,1	0
1,0	0
1,1	1

Querying AND function:



When $b = 0$ and queried in superposition:

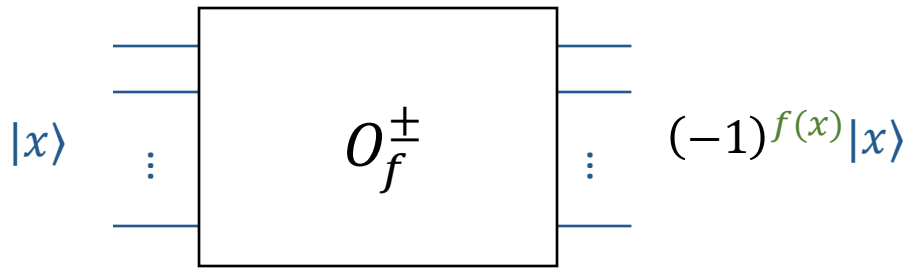
$$\sum_x \alpha_x |x\rangle|0\rangle \rightarrow \sum_x \alpha_x |x\rangle|\mathrm{AND}(x_0, x_1)\rangle$$

Phase Oracle

Phase Oracle $O_f^{\pm}$ (for $n = 1$):

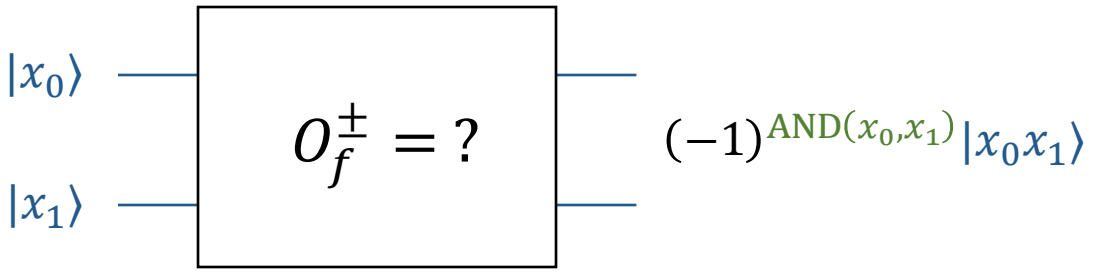
$$\sum_x \alpha_x |x\rangle \to \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

"Keeping x, and storing $f(x)$ in phase."



How do we implement this?

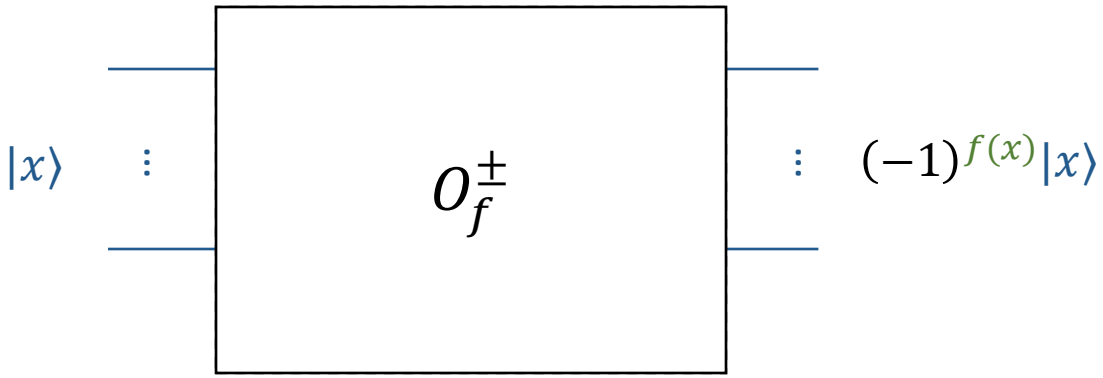
Example (querying AND function):



When queried in superposition:

$$\sum_x \alpha_x |x\rangle \to \sum_x \alpha_x (-1)^{\mathrm{AND}(x_0,x_1)} |x\rangle$$

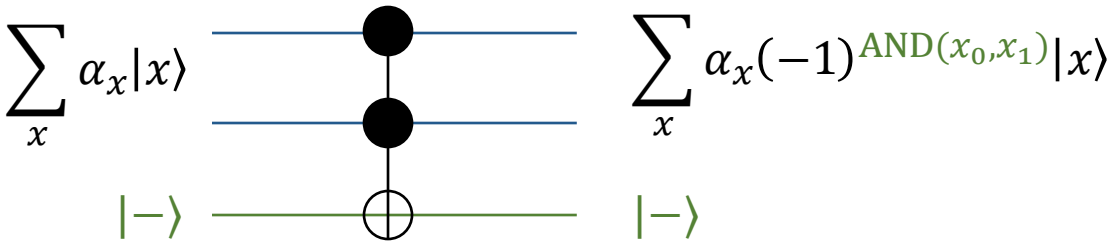
Implementing $O_f^{\pm}$ using O_f?



$$\left(\sum_x \alpha_x |x\rangle\right)|-\rangle \xrightarrow{O_f} \left(\sum_x \alpha_x (-1)^{f(x)}|x\rangle\right)|-\rangle$$

Derive on board:

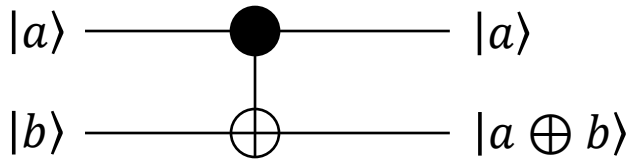
Example (querying AND function):



Seems to change the control qubits!

Phase Kickback

Understanding CNOT gates

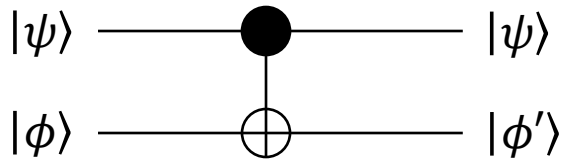


$$\mathrm{CX} = \begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

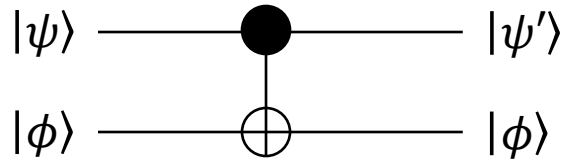
"Quantum if-else":

- If a is 0: do nothing.
- If a is 1: flip b.

When does it change only the **target** qubit?



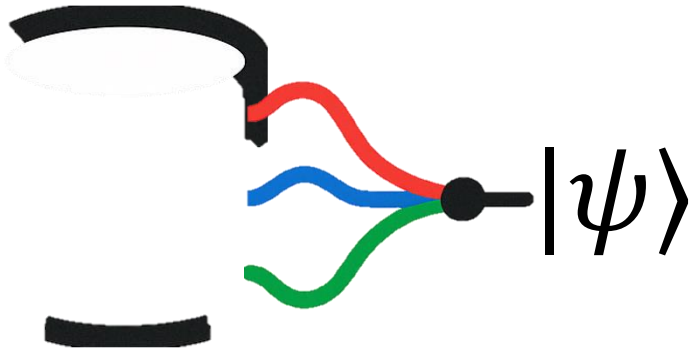
When does it change only the **control** qubit?

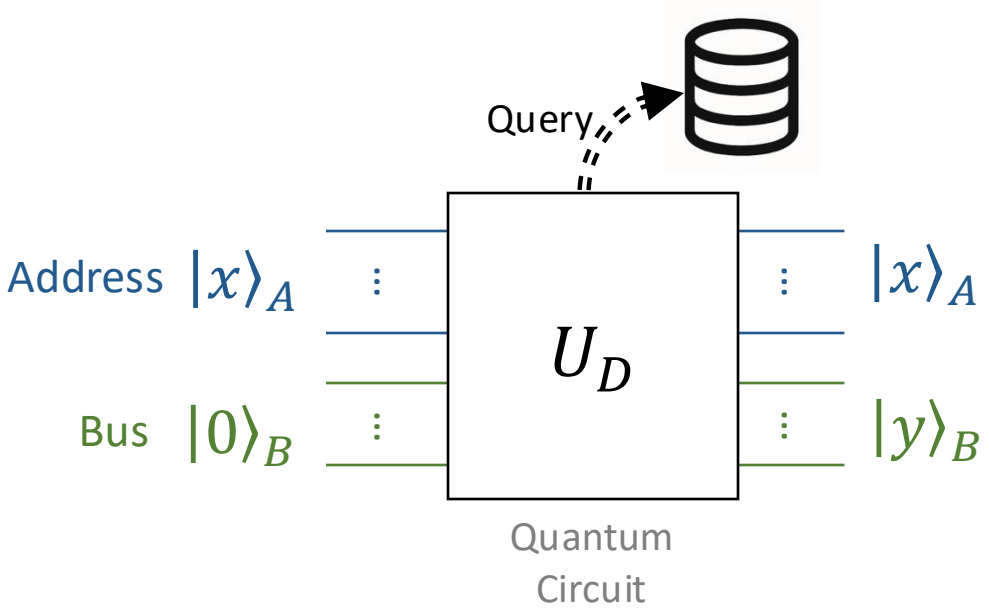


Quantum Random Access Memory

Loading Data in Superposition

$(x, y_x) \in D$: Dataset with (key, value) pairs





Accessing memory **in superposition**:

$$|\psi_{x,0}\rangle = \sum_x \alpha_x |x\rangle_A |0\rangle_B \xrightarrow{U_D} |\psi_{x,y}\rangle = \sum_x \alpha_x |x\rangle_A |y_x\rangle_B$$

Initial Final

What does this quantum circuit for U_D look like?

Special-Purpose Oracle

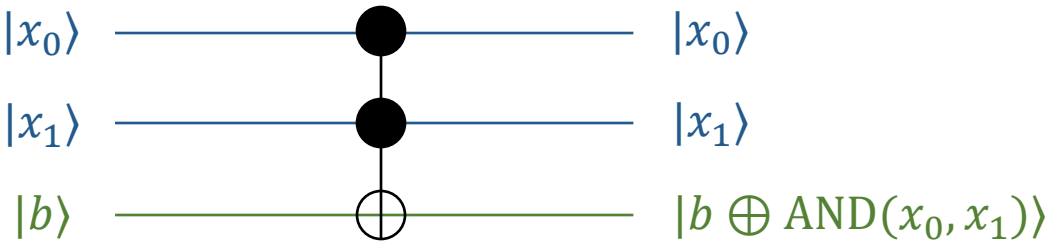
Example Oracle O_f:

$$f(x) = \mathrm{AND}(x_0, x_1)$$

Key: x	**Val:** $f(x)$
0,0	0
0,1	0
1,0	0
1,1	1

$$O_f = |00\rangle\langle 00| \otimes I + |01\rangle\langle 01| \otimes I + |10\rangle\langle 10| \otimes I + |11\rangle\langle 11| \otimes X$$

Querying AND function:



When $b = 0$ and queried in superposition:

$$\sum_x \alpha_x |x\rangle|0\rangle \rightarrow \sum_x \alpha_x |x\rangle|\mathrm{AND}(x_0, x_1)\rangle$$

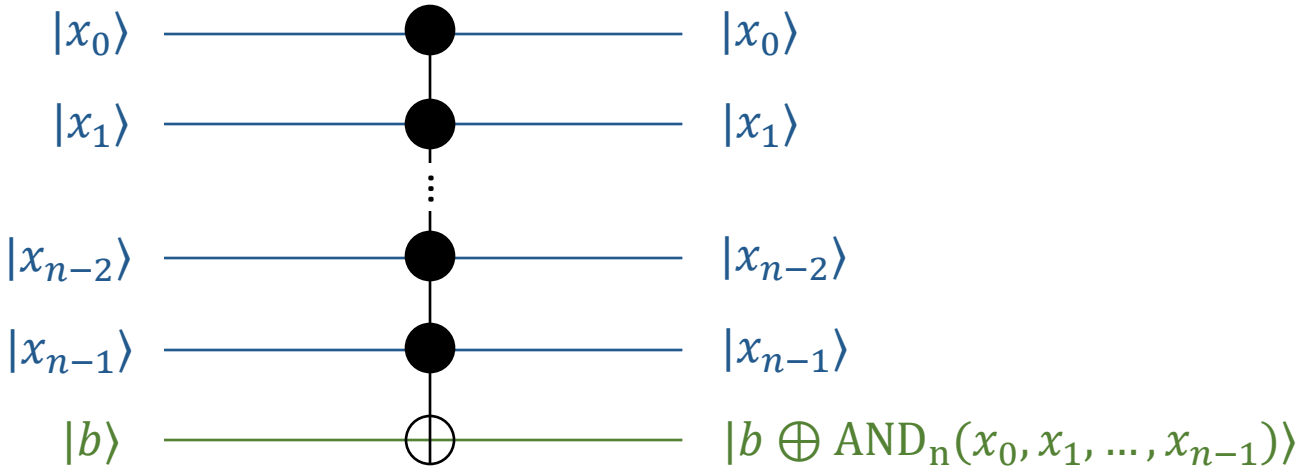
Special-Purpose Oracle

Example Oracle O_f:

$$f(x) = \mathrm{AND_n}(x_0, x_1, \dots, x_{n-1})$$

Key: x	**Val:** $f(x)$
0,… ,0,0	0
0,… ,0,1	0
…	0
1,… ,1,1	1

Querying $\mathrm{AND_n}$ function:



Querying Arbitrary Function f

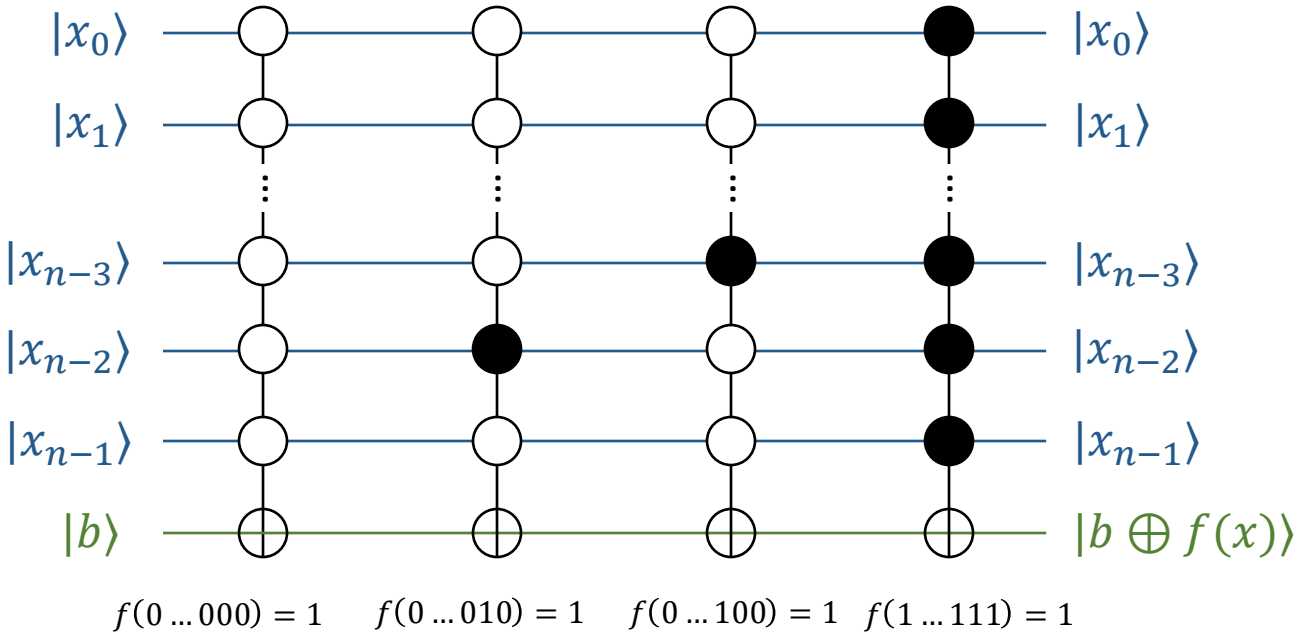
1-controlled-NOT 0-controlled-NOT

Example Oracle O_f:

Key: x	**Val:** $f(x)$
0,… ,0,0,0	1
0,… ,0,0,1	0
0,… ,0,1,0	1
0,… ,0,1,1	0
0,… ,1,0,0	1
…	0
1,… ,1,1,1	1

 # Address qubits: n
Memory size: $N = 2^n$

Querying f function:



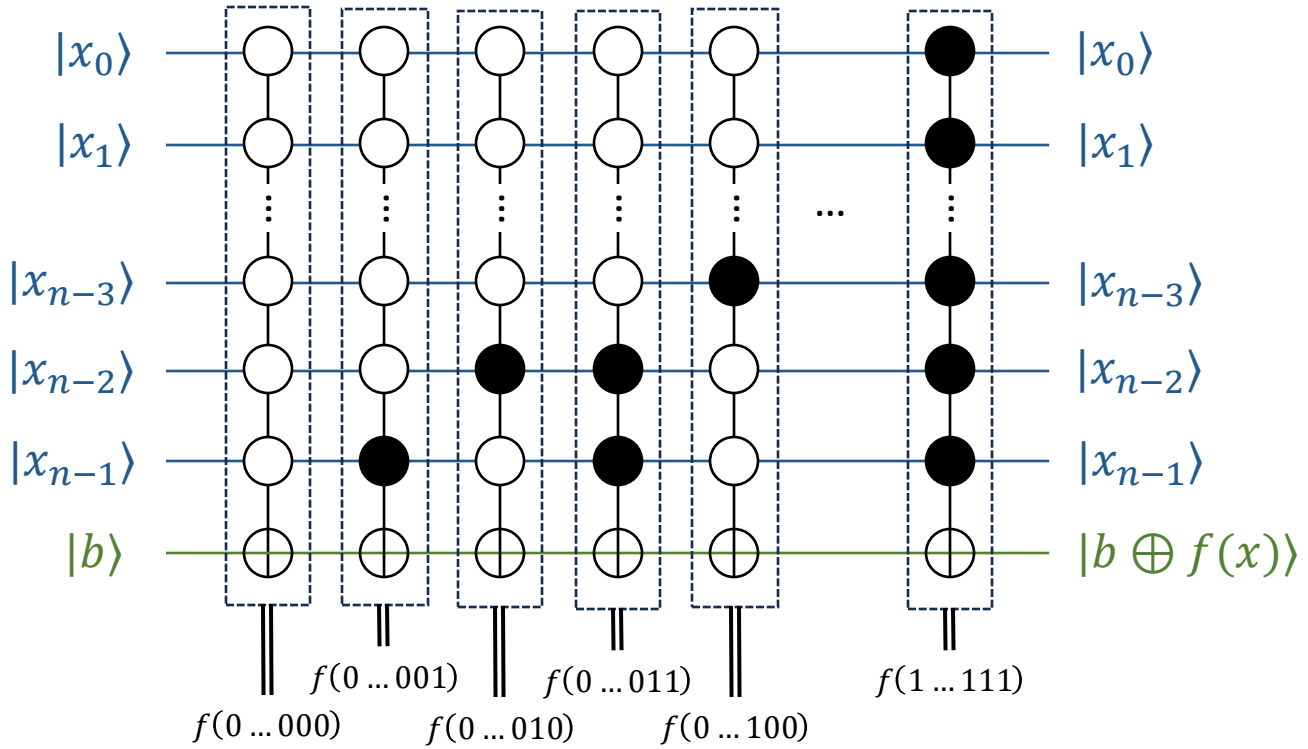
Apply a multi-controlled-NOT gate for every x s.t. $f(x) = 1$.

Towards General-Purpose Oracles

Example Oracle O_f:

Key: x	**Val:** $f(x)$
0,… ,0,0,0	$f(0 \dots 000)$
0,… ,0,0,1	$f(0 \dots 001)$
0,… ,0,1,0	$f(0 \dots 010)$
0,… ,0,1,1	$f(0 \dots 011)$
0,… ,1,0,0	$f(0 \dots 100)$
…	
1,… ,1,1,1	$f(1 \dots 111)$

Querying f function:

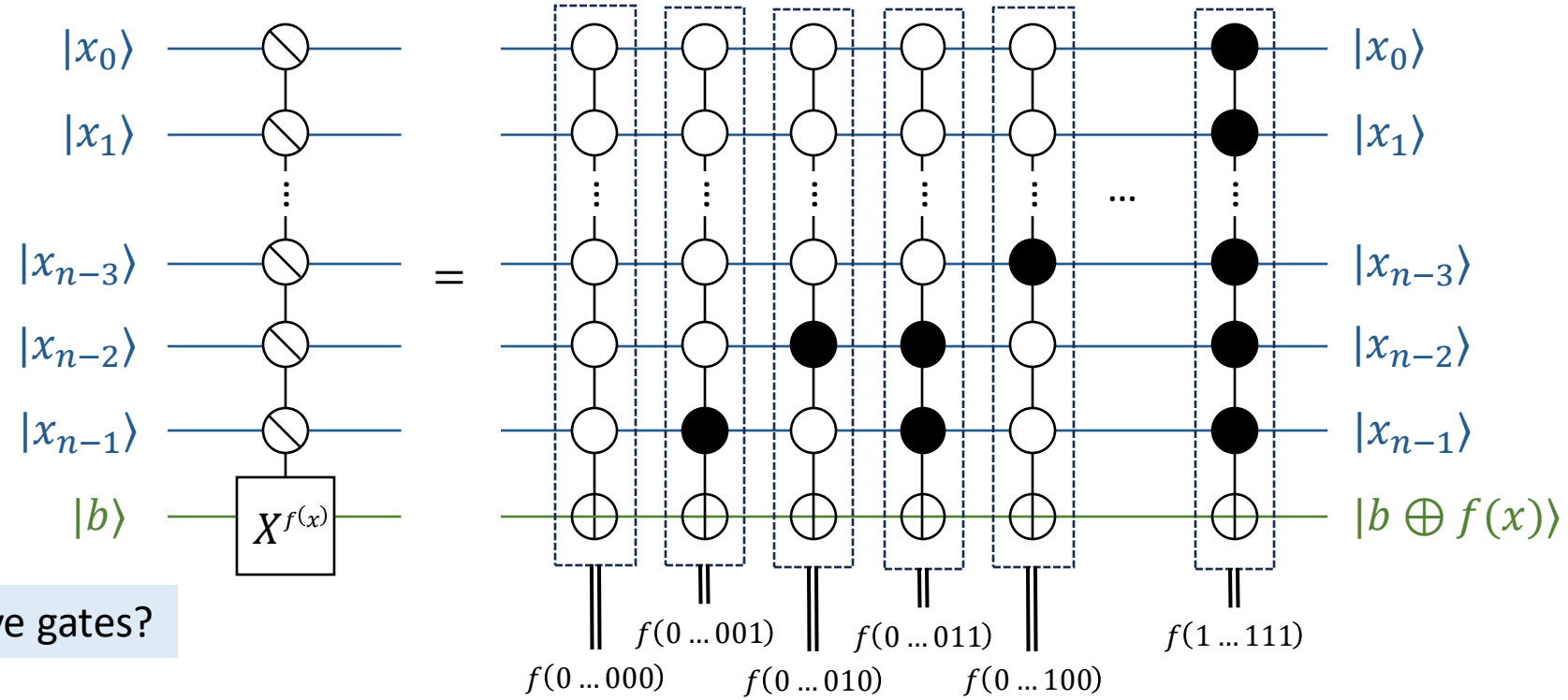


Towards General-Purpose Oracles

Given an arbitrary Boolean function $f: \{0,1\}^n \to \{0,1\}$, a quantum oracle to f is defined by the Select Operator O_f.

Select Operator:

$$O_f = \prod_{x \in \{0,1\}^n} |x\rangle\langle x| \otimes X^{f(x)}$$

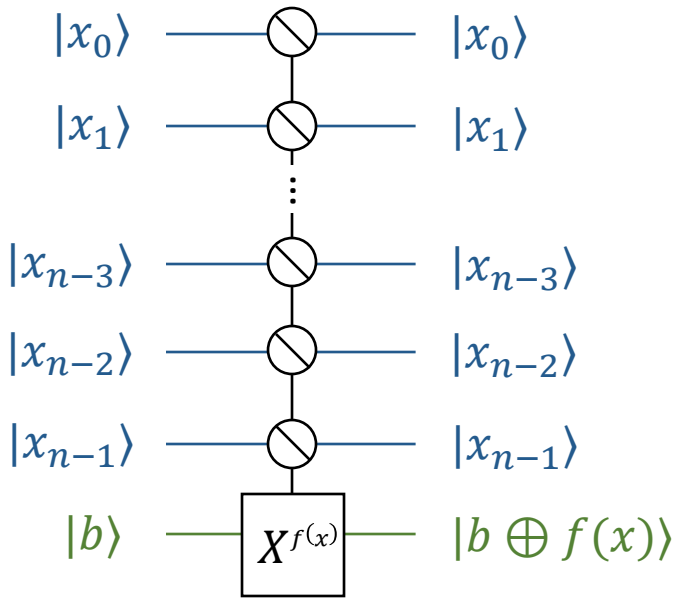


How to implement with primitive gates?

Implementing General-Purpose Oracles

Select Operator:

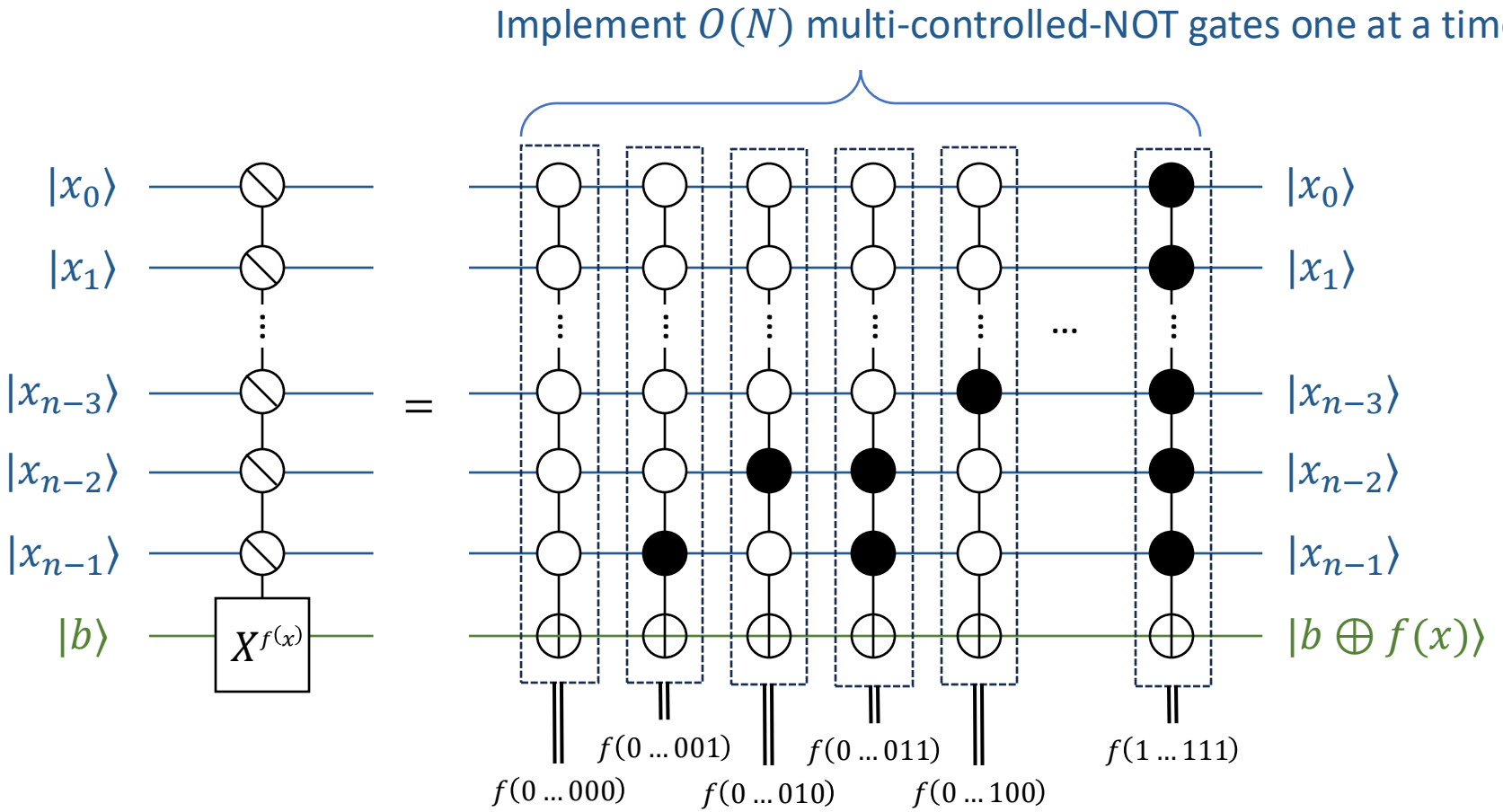
$$O_f = \prod_{x \in \{0,1\}^n} |x\rangle\langle x| \otimes X^{f(x)}$$



Implementations:

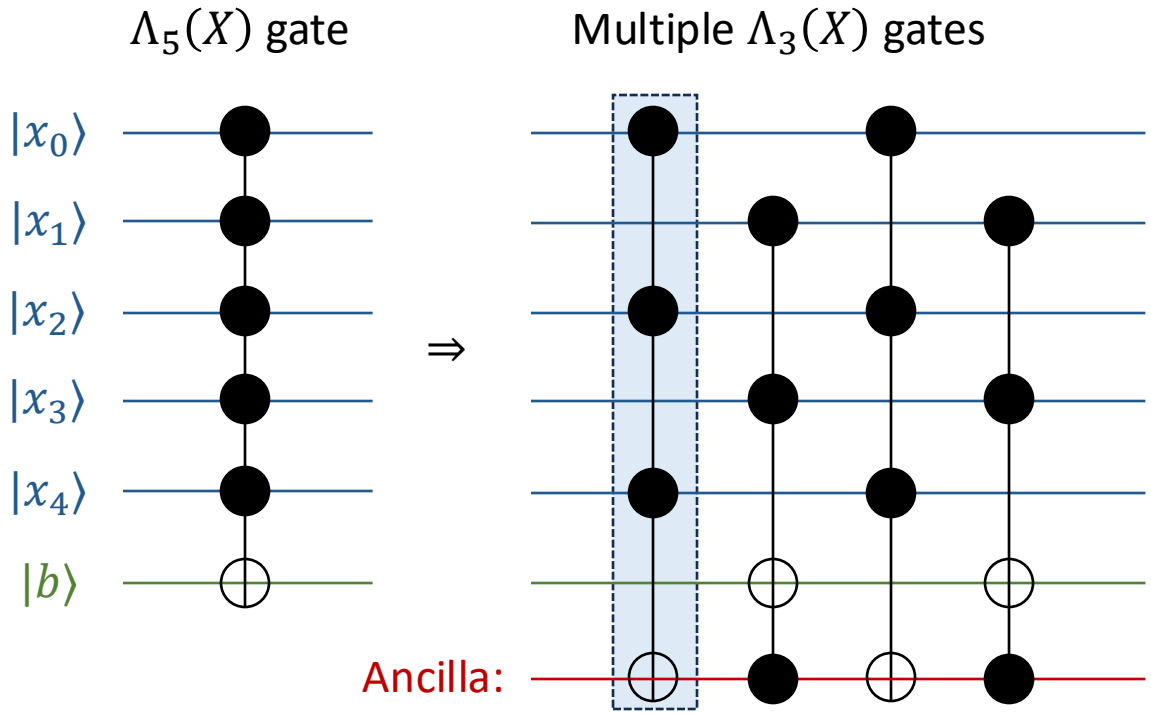
1. **Sequential Query Circuit:**
   - Gates: Toffoli, CX gates
   - Time: $O(N)$
   - Qubits: $\log N$ address qubits, 1 bus qubit, 1 ancilla qubit.
2. **Fanout QRAM:**
   - Gates: Fanout, cSWAP gates
   - Time: $O(\log N)$
   - Qubits: $\log N$ address qubits, 1 bus qubit, $O(N)$ ancilla qubits.
3. **Bucket-Brigade QRAM:**
   - Gates: cSWAP gates
   - Time: $O(\log N)$
   - Qubits: $\log N$ address qubits, 1 bus qubit, $O(N)$ ancilla qubits.

Sequential Query Circuit



Implementing A Multi-Controlled-NOT Gate

Using Toffoli gates (acting on three qubits) and few ancillary qubits.

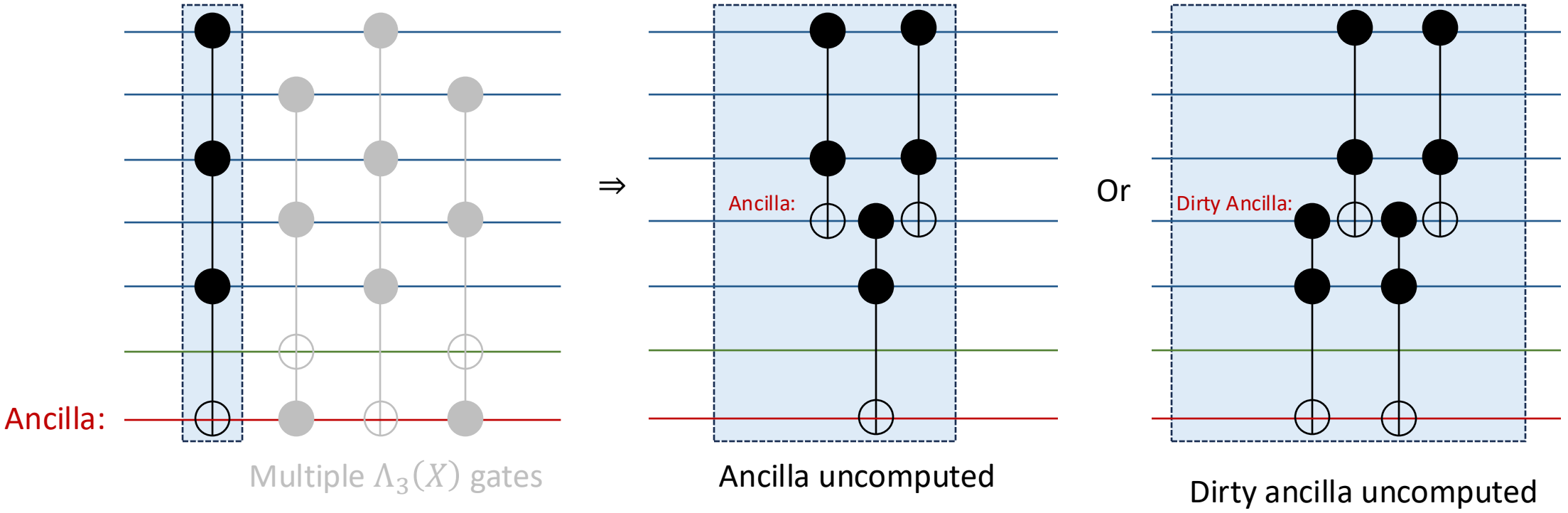


What's the resulting ancilla state?

What if the initial ancilla state is unknown ("dirty")?

Implementing Multi-Controlled-NOT Gate

Using Toffoli gates (acting on three qubits) and few ancillary qubits.



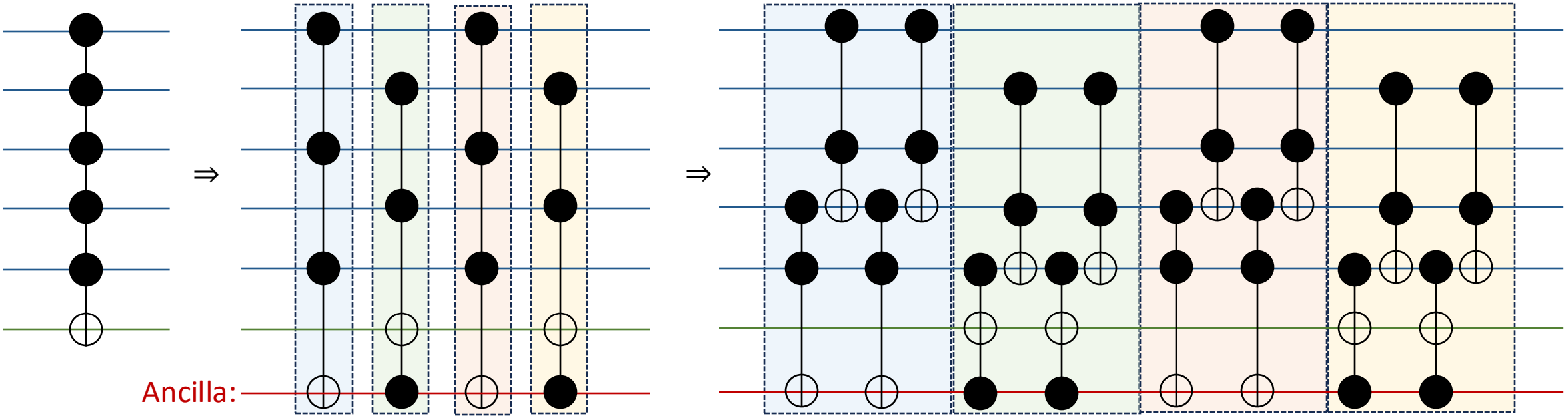
Implementing Multi-Controlled-NOT Gate

Using Toffoli gates (acting on three qubits) and few ancillary qubits.

$\Lambda_5(X)$ gates

Multiple $\Lambda_3(X)$ gates

Multiple $\Lambda_2(X)$ (Toffoli) gates



For $\Lambda_n(X)$ gate, we can implement using $\approx 16n$ Toffoli gates. **Total timesteps**: $O(n)$.