Chapter 6:

Circuit Synthesis and Compilation

Presenter: Andrew Zhang

UC RIVERSIDE

Objectives

- Use few gates as possible,
- Minimize the amount of scratch memory
- Performance
- Precision

UC RIVERSIDE

Outline

6.1 Synthesizing Quantum Circuits

6.2 classical Vs. Quantum Compiler Optimization

6.3 Gate Scheduling And Parallelism

6.4 Qubit Mapping And Reuse

UC RIVERSIDE

Synthesizing Quantum Circuits

Implement some arbitrary unitary transformation using a given finite set of realizable quantum gates, efficiency and optimality by:

- Choice of universal instruction set;
- Single-qubit, multi-qubit, and qudit (i.e., d-level quantum logic) synthesis; and
- Exact and approximate synthesis.

Synthesizing Quantum Circuits

“Powerful”: CISC

“Less Powerful”: RISC

consist of single qubit rotation (R(Θ))gates and two-qubit cross-resonance (CR) gate

UNIVERSAL INSTRUCTION SET:

Definition 6.1 A quantum instruction set S is called computationally universal if and only if gates from S allow the realization of an arbitrary quantum circuit C .

Synthesizing Quantum Circuits

Theorem 6.2 One-qubit and two-qubit unitary gates are universal; .....

Eg. single-qubit gates and CNOT gate are universal.

In particular, for experimentalists, much engineering efforts have been put into physically implementing single-qubit gates and CNOT gates on qubits. This is because, as noted in Chapter 2, single-qubit gates and CNOT gate are universal. The proof is omitted here, but can be found in [86]. The key idea is to show that single-qubit gates and CNOT gate can be used to make up all two-qubit unitaries. Thus, following Theorem 6.2, we can show that any quantum circuit made of n-qubit gates can be exactly simulated by single-qubit gates and CNOT gates, with only a linear increase in the number of gates.

EXACT SYNTHESIS

- Controlled unitary gates
- Quantum oracle gates
- Clifford+T gates instruction set

UC RIVERSIDE

Controlled unitary gates

$$\Lambda(U)\,|q_1\rangle \otimes |\psi\rangle = |q_1\rangle \otimes U^{q_1}\,|\psi\rangle = \begin{cases} |q_1\rangle \otimes U\,|\psi\rangle & \text{if } |q_1\rangle = |1\rangle\,, \\ |q_1\rangle \otimes |\psi\rangle & \text{if } |q_1\rangle = |0\rangle\,. \end{cases}$$

*From **1** bit to **k** + **t** qubits...*

$$\begin{aligned} \Lambda_k(U)\,|q_1 \cdots q_k\rangle \otimes |\psi\rangle &= |q_1 \cdots q_k\rangle \otimes U^{q_1 \cdots q_k}\,|\psi\rangle \\ &= \begin{cases} |q_1 \cdots q_k\rangle \otimes U\,|\psi\rangle & \text{if } |q_1 \cdots q_k\rangle = |1 \cdots 1\rangle\,, \\ |q_1 \cdots q_k\rangle \otimes |\psi\rangle & \text{otherwise,} \end{cases} \end{aligned}$$

Controlled unitary gates

$$U = e^{i\alpha} A_m B_m \ldots A_2 B_2 A_1 B_1 \qquad A_m \cdots A_2 A_1 = I$$

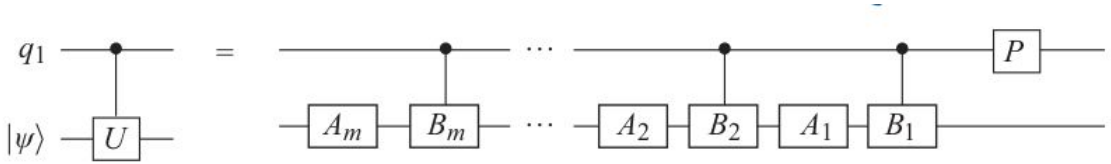


Figure 6.1: A generic decomposition for controlled unitary gate ($\Lambda(U)$). Note that P is the phase shift gate $P = |0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$.

$$\Lambda(U) = \Lambda(e^{i\alpha})(I \otimes A_m)\Lambda(B_m)\ldots(I \otimes A_2)\Lambda(B_2)(I \otimes A_1)\Lambda(B_1)$$

Controlled unitary gates

Decompose ***U***

$$U = e^{i\alpha} CXBXA \qquad CBA = I \qquad \Lambda(X) = \text{CNOT}$$

*Now can be implemented using **single-qubit gates** and **CNOT** gates.*

Synthesizing Quantum Oracles

Oracles need to be implemented just like the rest of the algorithm.

The cost of the oracles eventually contributes to the time cost of the overall algorithms.

UC RIVERSIDE

Synthesizing Quantum Oracles

Any quantum logic gate we apply to a qubit has to be reversible.

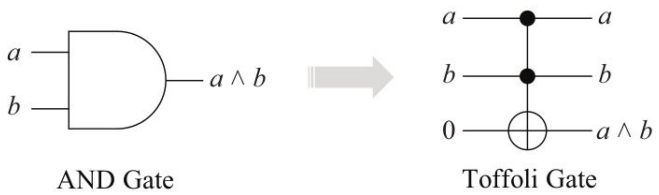
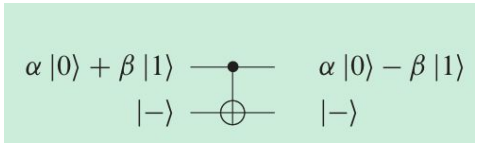


Figure 6.3: Circuit diagram for the irreversible `AND` gate and the reversible `Toffoli` gate.

Phase Kickback

Circuit that change the control bits instead of target bit.



Berstein–Vazirani algorithm:

$$f(x) = s \cdot x = \sum_{i=1}^{n} s_i \cdot x_i \mod 2.$$

$$O_f^{\pm} |q_1 q_2 \dots q_n\rangle = (-1)^{f(q_1 q_2 \dots q_n)} |q_1 q_2 \dots q_n\rangle .$$

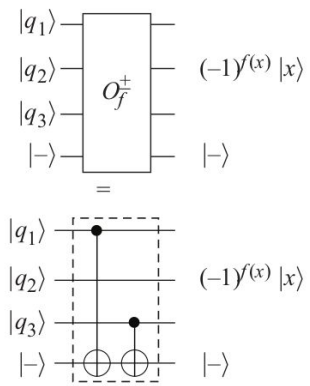


Figure 6.4: Oracle implementation for Bernstein–Vazirani algorithm with secret string $s = 101$.

Synthesizing Quantum Oracles

1. Find an efficient implementation of the desired function using reversible logic gates.

2. Implement each reversible logic gate using quantum gate(s).

Works for simple, modulated logic/algorithm.

The advantage of quantum algorithms is thought to stem from their ability to pass a superposition of inputs into a classical function at once, whereas a classical algorithm can only evaluate the function on single input at a time.

UC RIVERSIDE

Synthesis Over Structured Instruction Sets

When a target instruction set has well-understood, nice structures, exact synthesis of quantum circuits over such instruction set can be efficiently (and sometimes optimally) implemented.

Single qubit system: Clifford-T $\langle H, T, \omega \rangle$

UC RIVERSIDE

APPROXIMATE SYNTHESIS

Approximation Metrics

The *distance* of the approximate implementation V to the ideal one U is defined as:

$$D(U, V) = \sup_{|\psi\rangle} ||(U - V)|\psi\rangle||,$$

Lemma 6.6 *If* $U_1, \ldots, U_t$ *and* $V_1, \ldots, V_t$ *are operators such that* $||U_i|| \leq 1$, $||V_i|| \leq 1$, *and* $||U_i - V_i|| < \delta$, *for all* $i \in [t]$, *then*

$$||U_t \cdots U_2 U_1 - V_t \cdots V_2 V_1|| < t\delta.$$

This subadditivity property is the basis of many approximation algorithms.

APPROXIMATE SYNTHESIS

$$UU^\dagger = VWV^\dagger W^\dagger$$

Algorithm 6.1 Solovay–Kitaev (SK) algorithm for synthesizing single-qubit gates

Function SolovayKitaev(single-qubit gate U, accuracy level n)

1: **if** $n == 0$ **then**
2: **return** Closest approximation to U
3: **else**
4: $U \leftarrow \text{SolovayKitaev}(U, n-1)$
5: $V, W \leftarrow$ Decompose $UU_{n-1}^\dagger$ into a balanced group commutator
6: $V_{n-1} \leftarrow \text{SolovayKitaev}(V, n-1)$
7: $W_{n-1} \leftarrow \text{SolovayKitaev}(W, n-1)$
8: **return** $U_n \leftarrow V_{n-1} W_{n-1} V^\dagger W^\dagger U_{n-1}$
9: **end if**

The Solovay–Kitaev (SK) algorithm shows that we can do much better—any single-qubit gate can be approximated to arbitrary accuracy ϵ_0 using only $O(\log^c(1/\epsilon_0))$ gates. The key idea

UC RIVERSIDE

HIGHER-DIMENSIONAL SYNTHESIS

We want to universal gates like nand, but reversible gate may generate many ancilla which is not feasible.

Qutrits (three-level systems) allow us to essentially generate ancilla by borrowing a third energy state in a quantum device to hold extra information.

However, Third energy state generally is more susceptible to errors than the first two states used for binary qubits.

CLASSICAL VS. QUANTUM COMPILER OPTIMIZATION

- classical optimizations under classical constraints,
- classical optimizations under quantum constraints,
- quantum optimizations under quantum constraints.

UC RIVERSIDE

classical optimizations under classical constraints

loop unrolling, procedure cloning, interprocedural constant propagation, etc.

The classical constraints, such as data dependencies, pipelining, and synchronizations, still apply in the context of quantum compiling.

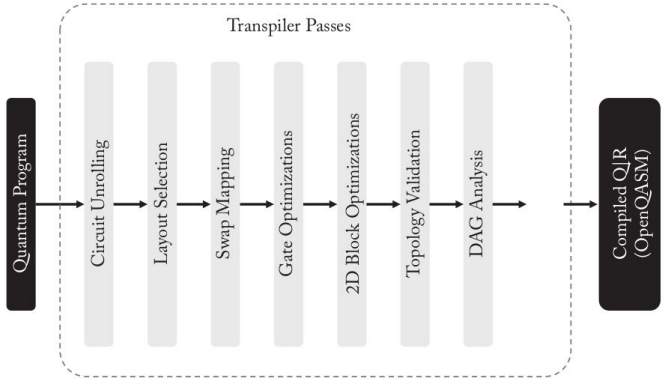


Figure 6.5: The compilation framework in Qiskit Terra developed by IBM [165].

classical optimizations under quantum constraints

Communication cost of two-qubit interactions is one example of such quantum constraints

- Routing strategies used in distributed systems
- Register allocation algorithms
- Mapping qubits to less noisy parts of a quantum machines
- Scheduling parallel gates

UC RIVERSIDE

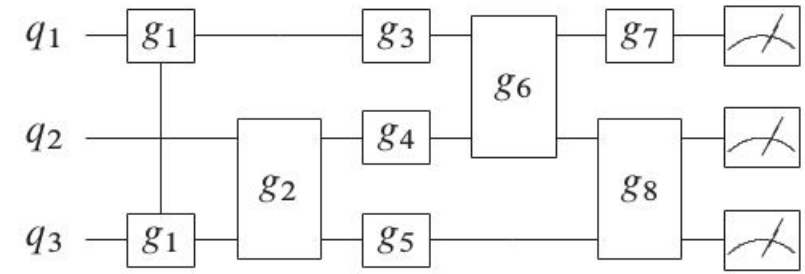
quantum optimizations under quantum constraints

- Quantum programs can be stored in qubits
- Gate teleportation
- Use a quantum computer to perform quantum compiling

UC RIVERSIDE

GATE SCHEDULING AND PARALLELISM

Data Dependency:



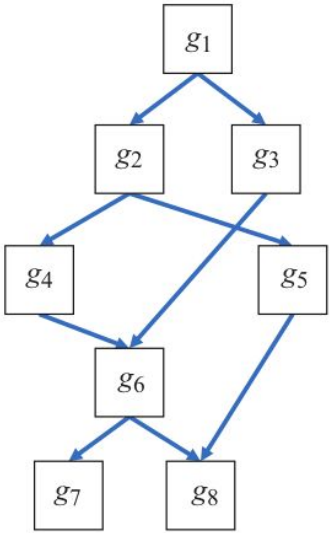
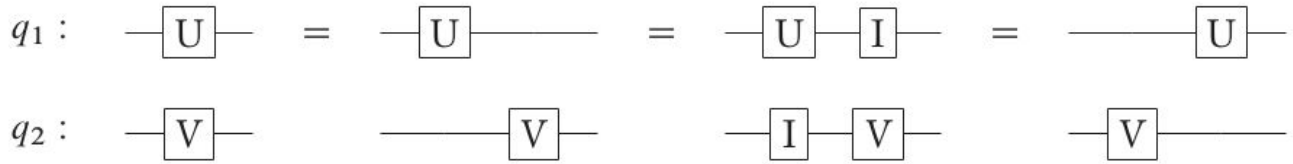


Figure 6.6: A data dependency graph for a quantum circuit with 8 gates: $g_1, \ldots, g_8$.

PRIMARY CONSTRAINTS IN SCHEDULING



$$U \otimes V = (U \otimes I) \cdot (I \otimes V) = (I \otimes V) \cdot (U \otimes I).$$

Hardware Limitations

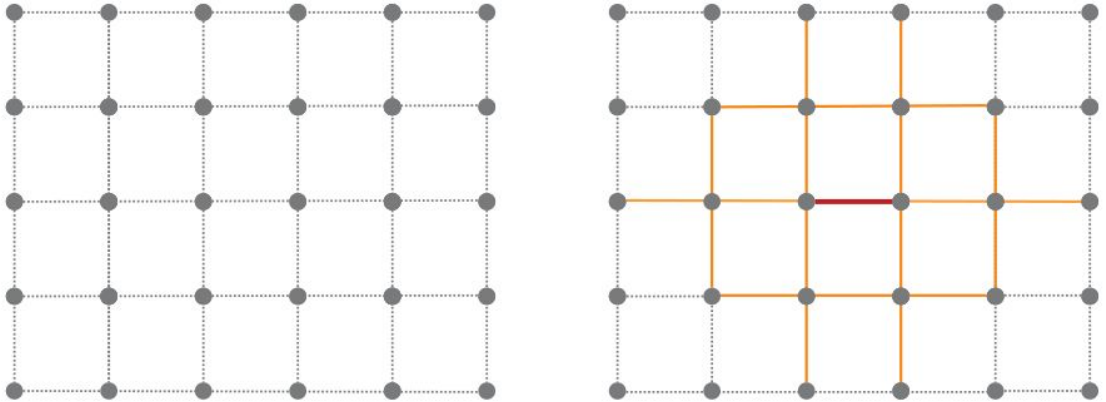


Figure 6.7: **Left:** the connectivity graph of a 5×6 2D mesh. Vertices are qubits, and two vertices are connected if the qubits are connected. **Right:** when the two center qubits are chosen for a two-qubit gate at an interaction frequency (highlighted in red), the surrounding qubits must be tuned off resonance from this interaction frequency, hence another two-qubit gate at the same frequency cannot be scheduled on any orange edges.

Commutation Relations

difference:

Gate scheduling in quantum algorithms differs from classical instruction scheduling, as gate commutativity introduces another degree of freedom for schedulers to consider.

Definition 6.8 The *commutator* of two gates (operators) A, B is defined as

$$[A, B] = AB - BA.$$

Definition 6.9 Two gates A and B are *conjugate*, if there exists another gate U such that $UAU^\dagger = B$.

A simple rewrite of the above equality results in the following:

$$UAU^\dagger = B \implies UAU^\dagger U = BU \implies UA = BU.$$

Commutation Relations

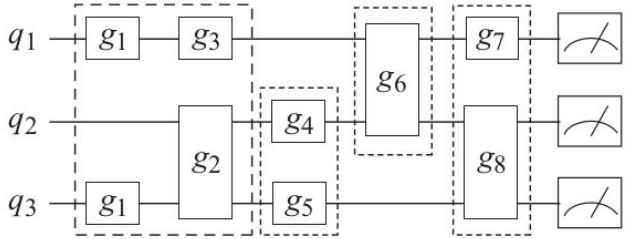


Figure 6.8: A sample quantum circuit with commutation relations indicated by dashed boxes.

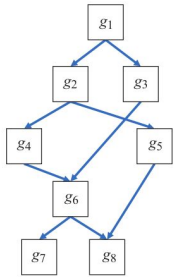


Figure 6.6: A data dependency graph for a quantum circuit with 8 gates: $g_1, \ldots, g_8$.

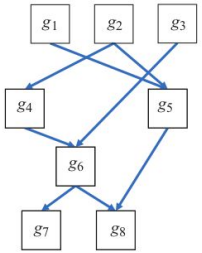


Figure 6.9: The commutation-relaxed data dependency graph (CRDDG) of the sample quantum circuit in Figure 6.8.

Qubit Decoherence

- Qubits have limited lifetime due to spontaneous decoherence

(except a qubit is in ground state, for example ancilla bits)

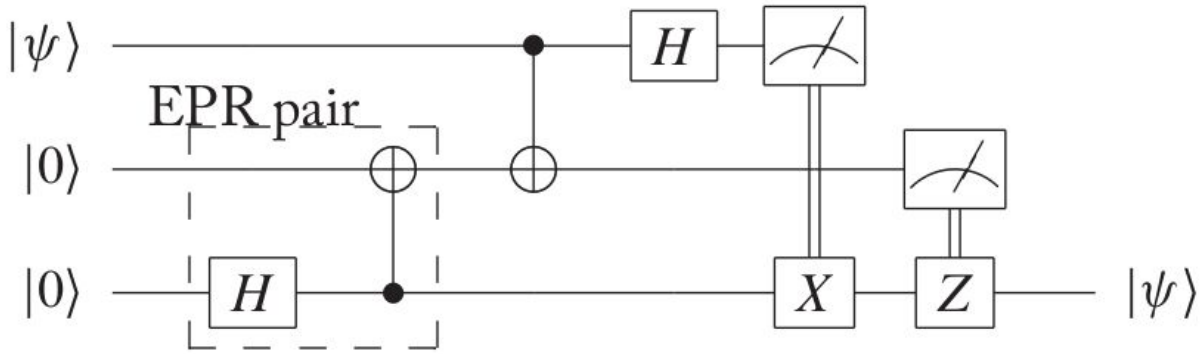
- A common goal in scheduling quantum circuit is to minimize the circuit depth

SCHEDULING STRATEGIES

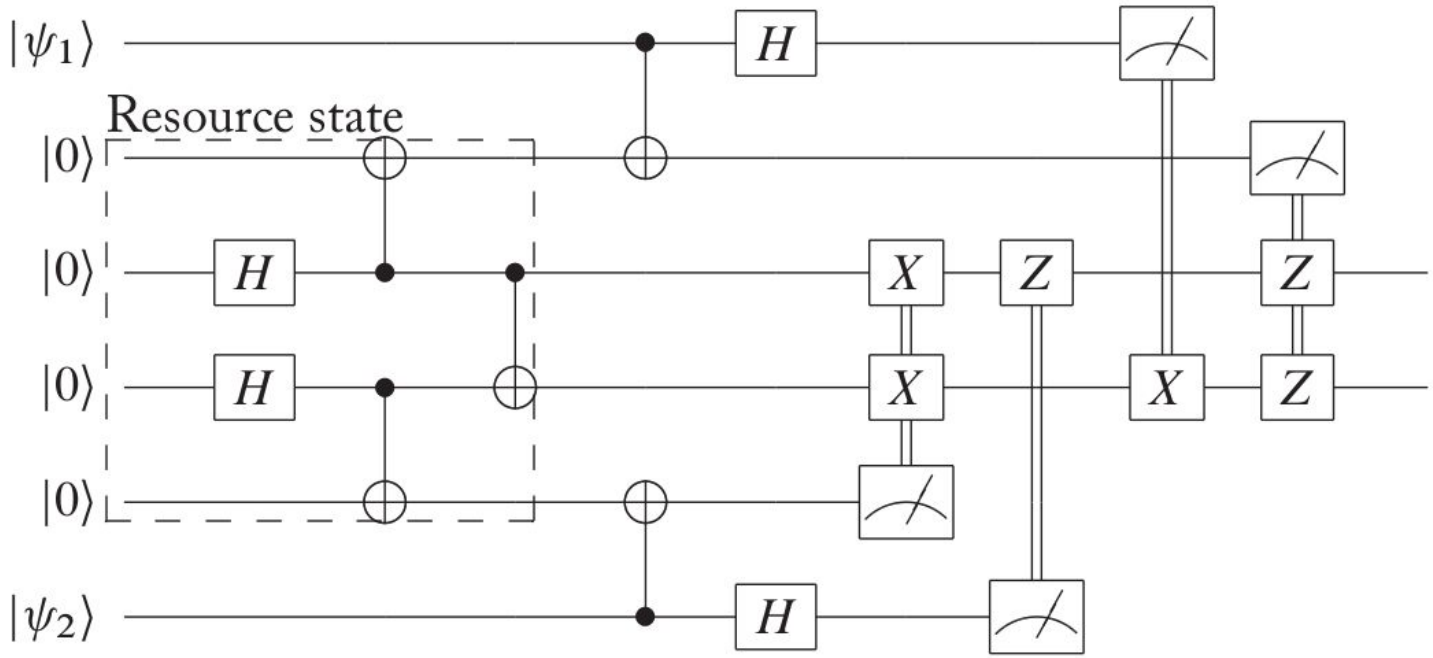
- ALAP (As-Late-As-Possible) Scheduling.
- LPF (Longest-Path-First) Scheduling.
- Communication-Aware Scheduling.
- Adaptive Scheduling.

UC RIVERSIDE

HIGHLIGHT: GATE TELEPORTATION



HIGHLIGHT: GATE TELEPORTATION



QUBIT MAPPING AND REUSE

optimized mapping, which assigns a physical location for each logical qubit

The goal of a mapper can be described as to most efficiently embed a program interaction graph in a device connectivity graph. -- minimizing communication and avoiding noisy qubits and links, etc.

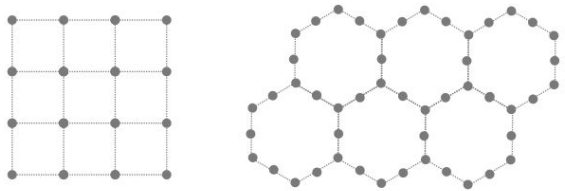


Figure 6.10: Different connectivity graph for superconducting devices. **Left:** 2D square lattice. **Right:** 2D (heavy) hexagonal lattice. The choice of connectivity graphs is typically based on hardware constraints such as wiring bandwidth and noises of circuit components.

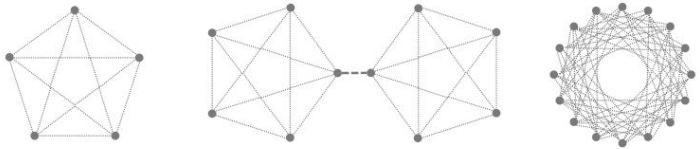


Figure 6.11: Different connectivity graph for trapped ion devices. **Left:** Complete (Clique) connectivity for small number of ions in one trap. **Center:** Weakly connected cliques for multiple traps. **Right:** A long chain of ions in one trap in a tape-like architecture.

FINDING A GOOD QUBIT MAPPING

- Edge Distance Minimization
- Edge Density Uniformity
- Edge Crossings Minimization

FINDING A GOOD QUBIT MAPPING

- Recursive Graph Partitioning
- Force-Directed Annealing
- Community Clustering

STRATEGICALLY REUSING QUBITS

Reclaiming Qubits via Measurement and Reset

Reusing Qubits via Borrowing

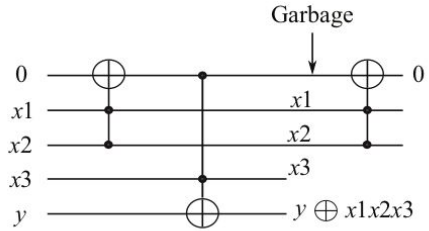


Figure 6.13: Implementation of $\Lambda^3(X)$ gate using an ancilla qubit in $|0\rangle$.

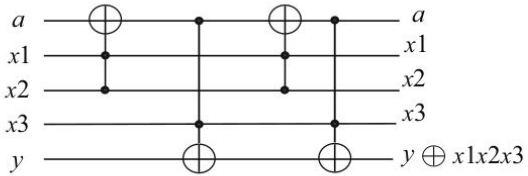


Figure 6.14: Implementation of $\Lambda^3(X)$ gate using an ancilla qubit in an arbitrary state.

HIGHLIGHT: UNCOMPUTATION

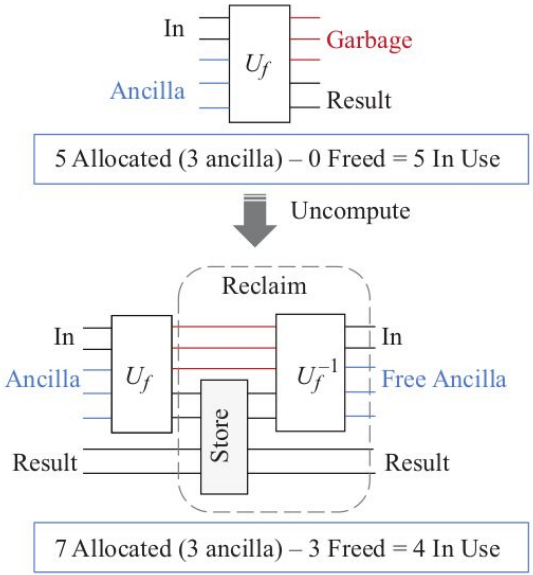


Figure 6.15: Ancilla qubit reclamation via uncomputation. Each horizontal line is a qubit. Each solid box contains reversible gates. Qubits are highlighted red for the duration of being garbage.

Reversible Pebbling Game

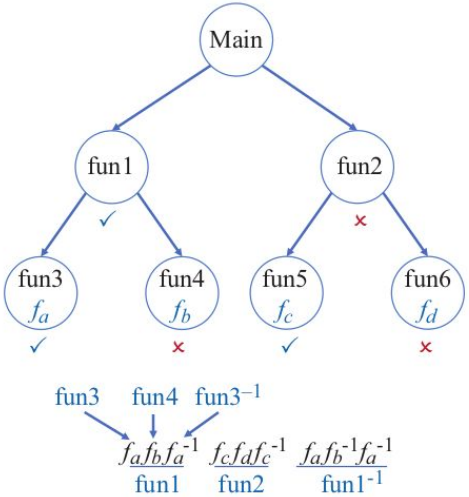


Figure 6.16: Divide-and-conquer approach to reversible pebble game. The balanced binary tree is traversed in depth-first search order. Function is executed when entering a node in the tree; function is uncomputed when exiting a node that is a left child of its parent (denoted by blue check marks).

Q&A

subtitle