COE 530
Quantum Computers and Architecture

Lecture 12

Circuit Optimization

References:

Garcia-Escartin, Juan Carlos, and Pedro Chamorro-Posada. "Equivalent quantum circuits." arXiv preprint arXiv:1110.2998 (2011).

Karuppasamy, Krishnageetha, Varun Puram, Stevens Johnson, and Johnson P. Thomas. "A comprehensive review of quantum circuit optimization: Current trends and future directions." Quantum Reports 7, no. 1 (2025): 2.

Generic Quantum Compilation Flow

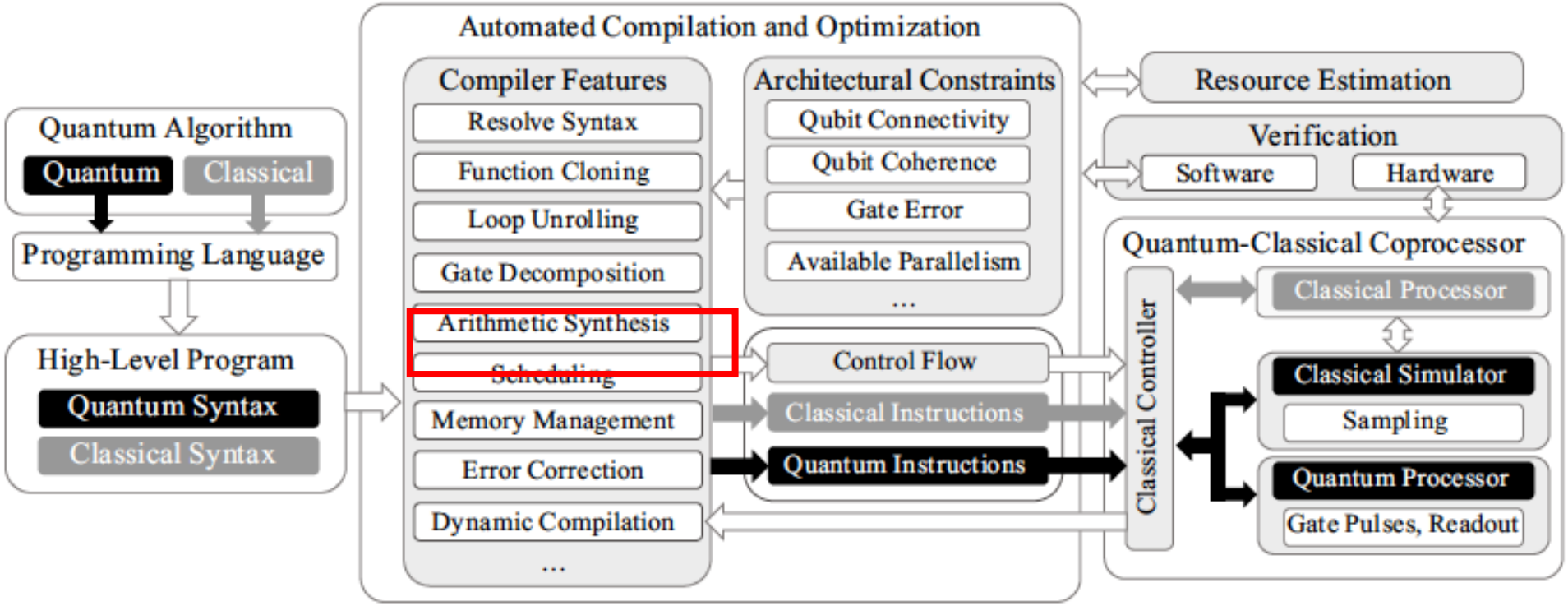


Figure 4.2: A detailed quantum compilation flow outlining the transformations and optimizations involved in a generic compiler.

Quantum Computer Systems: Research for Noisy Intermediate-Scale Quantum Computers

Quantum Circuit Execution Pipeline

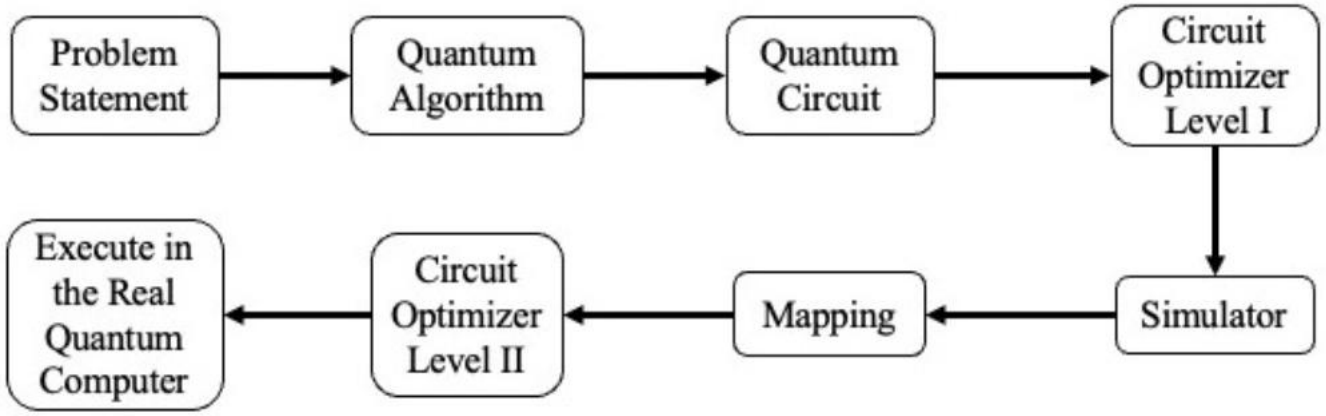


Figure 1. Execution pipeline of quantum algorithms.

- Two Levels of Optimization
  - Level I (Hardware-Independent): Applied before simulation — gate cancellation, fusion, depth reduction
  - Level II (Hardware-Aware): After simulation — qubit mapping to physical layout, SWAP gate insertion

Quantum Circuit Optimization Metrics

- Key Metrics for Evaluating Optimized Quantum Circuits
  - Circuit Depth: Number of sequential gate layers — reducing depth → faster execution
  - Gate Count: Total number of gates — minimizing reduces resource usage and error potential
  - Qubit Count: Number of qubits required — critical for mapping to hardware with limited availability
  - Two-Qubit Gate Count: More resource-intensive, higher error probability than single-qubit gates
  - Gate Fidelity: Closeness of physical gates to ideal gates — higher fidelity → more reliable computation
  - Parallelism: Degree of concurrent gate execution — maximizing → faster quantum computation
- Two Main Optimization Objectives
  - 1. Circuit Simplification: Minimize gate count & depth → enhanced computational speed, reduced errors
  - 2. Circuit Layout Optimization: Rearrange execution order & qubit mapping while preserving correctness

Core Methodologies for Circuit Simplification

- Pattern Matching
  - Recognizes and replaces common sub-structures with optimized equivalents
  - Adjacent gates that cancel or combine into simpler operations are identified
  - Uses Directed Acyclic Graphs (DAGs), where gates as vertices, dependencies as edges
  - Gate cancellations and fusions are systematically derived by analyzing the circuit structure
- Unitary Synthesis
  - Decomposes complex unitary operations into smaller, optimized components
  - Simplification techniques applied using machine learning and algorithmic approaches

Equivalent Quantum Circuits

- Some known transformation rules
- Rule I: Null gates

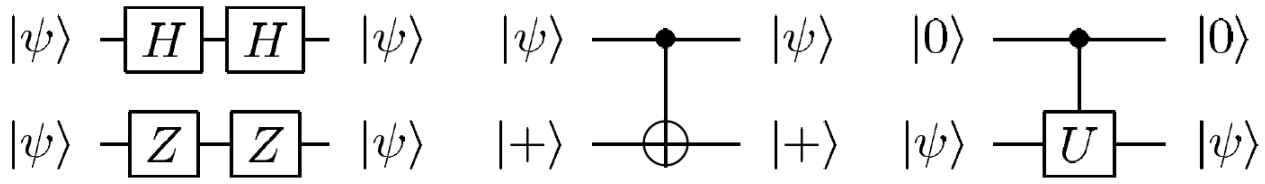


FIG. 6: Null operations.

- Rule II: Control Reversal

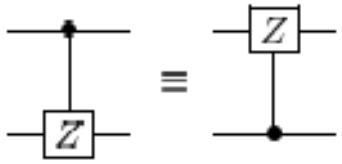


FIG. 7: Control reversal operation for a CZ gate.

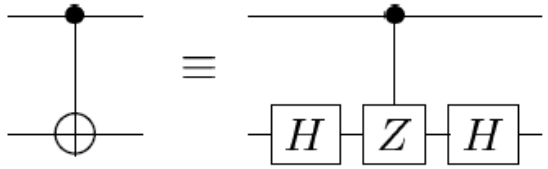


FIG. 8: CNOT with a CZ gate and two H gates

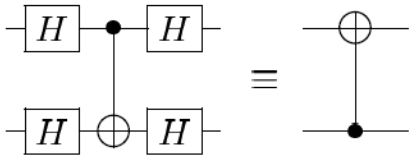


FIG. 10: CNOT reversal.

Garcia-Escartin, Juan Carlos, and Pedro Chamorro-Posada. "Equivalent quantum circuits." arXiv preprint arXiv:1110.2998 (2011).

Equivalent Quantum Circuits

- Rule III: Quantum-Classical Substitution

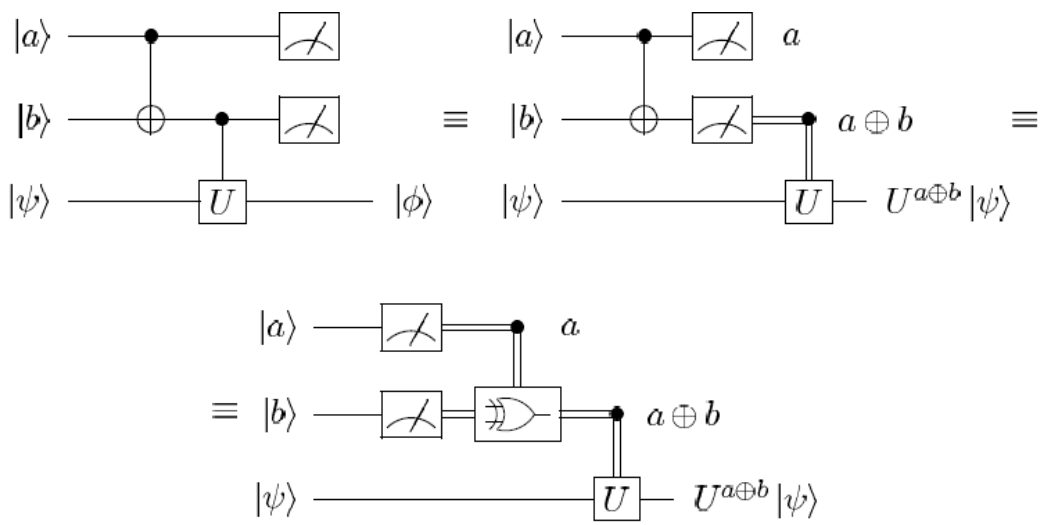
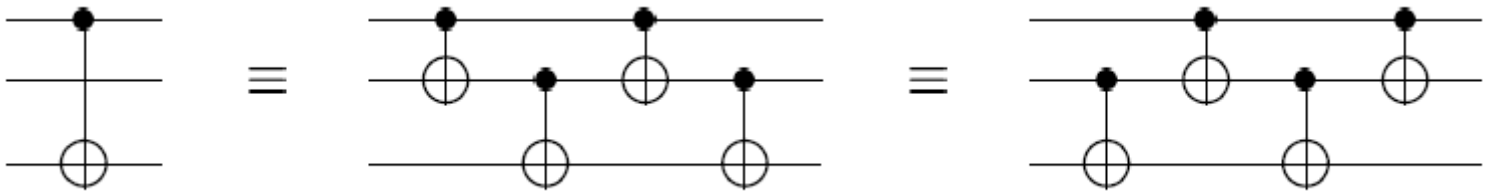


FIG. 13: Classical substitution of a CNOT gate before measurement.

- Rule V: Distributed CNOT



Garcia-Escartin, Juan Carlos, and Pedro Chamorro-Posada. "Equivalent quantum circuits." arXiv preprint arXiv:1110.2998 (2011).

Equivalent Quantum Circuits

- Rule VI: CNOT Mirror

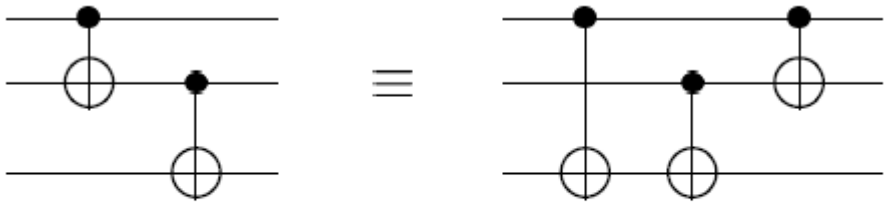
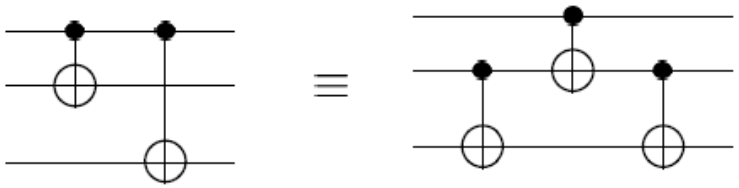


FIG. 15: Mirror over a CNOT gate.

- Rule V: Parallel CNOT Gates



Garcia-Escartin, Juan Carlos, and Pedro Chamorro-Posada. "Equivalent quantum circuits." arXiv preprint arXiv:1110.2998 (2011).

Commuting Gates Rule

- Two gates G_1, G_2 commute iff for any state $|\phi\rangle$: $U_1U_2|\phi\rangle = U_2U_1|\phi\rangle$

  Where U_1 and U_2 are unitary matrices corresponding to G_1 and G_2
- Rearranging execution order -> enables parallelism & gate cancellations (next)
- Note that more than two gates can also commute: $ABC = ACB = BCA = CAB = CBA$

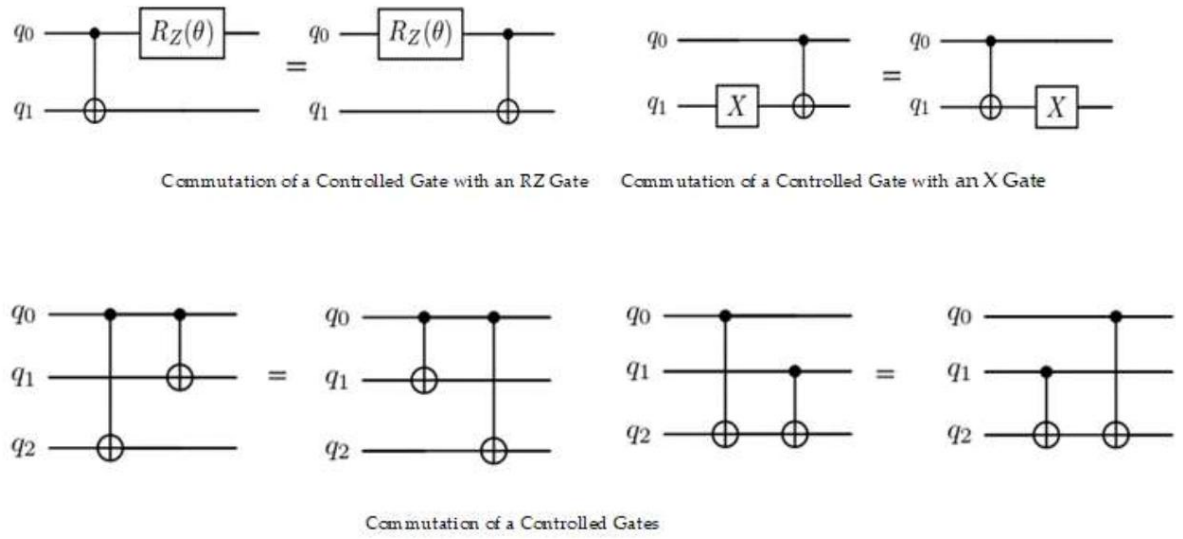


Figure 2. Illustration of gate commutations in quantum circuits.

Gate Cancellation Rule

- A real unitary matrix U satisfies:
  $U \cdot U = I$ (identity matrix)
- Examples
  - NOT gate
  - Z gate:
  - $CNOT, SWAP, CCNOT, CSWAP$

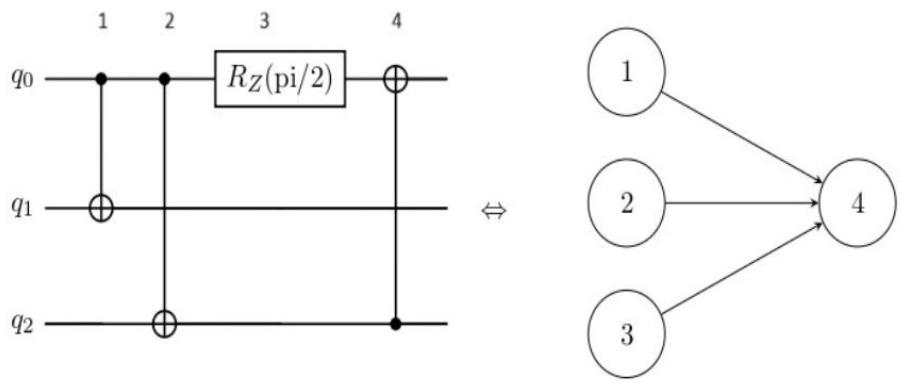


Figure 3. Sample circuit C and its associated DAG representation.

- On the other hand, T gate and S gate → NOT self-inverse
- How to implement gate cancellation programmatically?
- DAG Representation
- Circuit modeled as DAG: if two adjacent nodes = same real unitary gate → cancel out (yield I)

Hadamard Gate Reduction

- Many quantum algorithms require only specific combinations of Clifford gates
- By simplifying gate sequences, the number of H gates required can be reduced

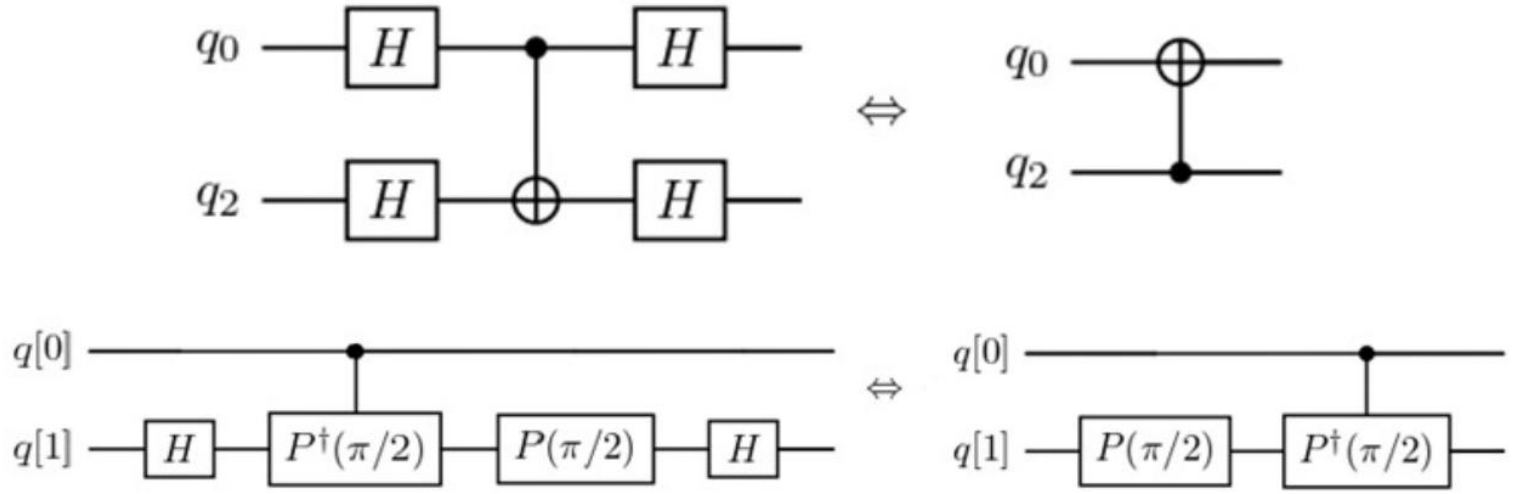


Figure 4. Illustrations of Hadamard gate reduction.

Rz Gate Merging

- When a circuit has only CNOT, NOT, and Rz gates, rotation gates can be merged
- Every Rz gate contributes a rotation angle θ to the quantum state
- If two Rz gates share the same phase pattern →
  merge into one $Rz(\theta_1) \cdot Rz(\theta_2) = Rz(\theta_1 + \theta_2)$
- Reduces both gate count and circuit depth without changing circuit behavior

Linear Reversible Circuits

- Circuits using only CNOT gates (e.g., stabilizer circuits)
- CNOT controlled by qubit j acting on qubit i: $E_{ij} = I + e_{ij}$
- Two adjacent $E(c_k, t_k)$ and $E(c_{k+1}, t_{k+1})$ can swap iff $c_k \neq t_{k+1}$ and $c_{k+1} \neq t_k$

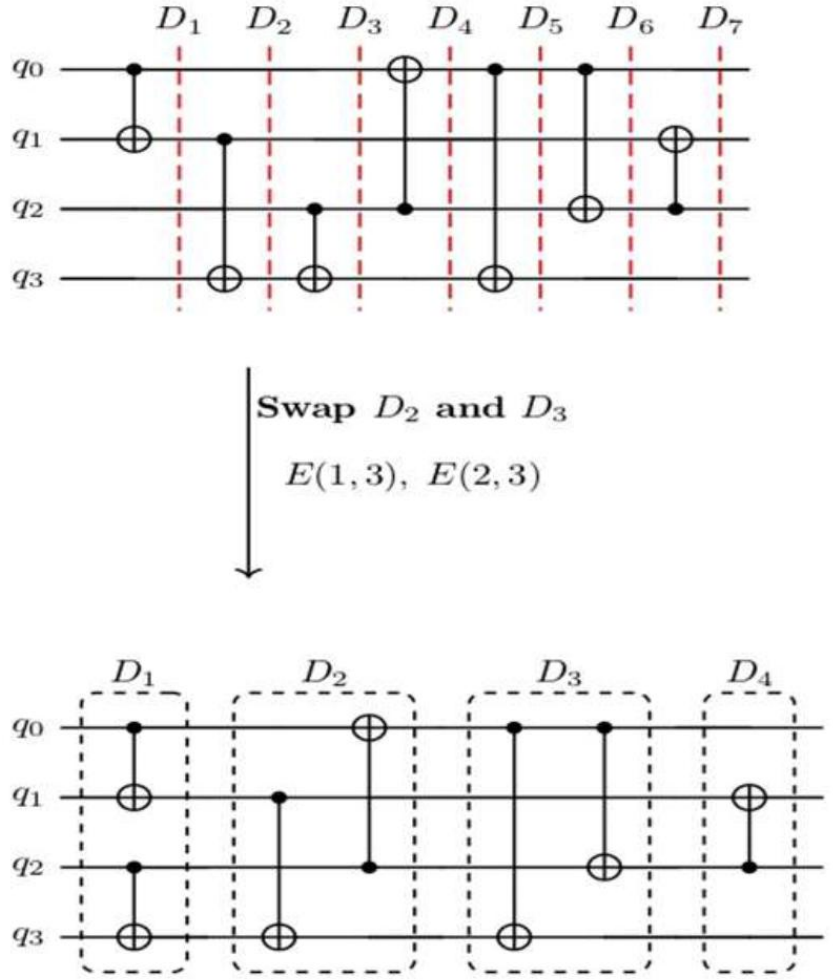
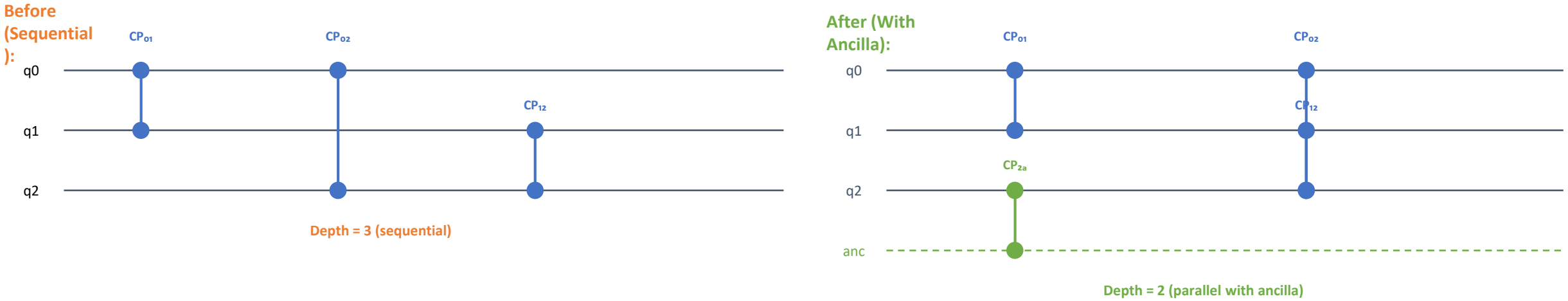


Figure 5. Illustration of depth reduction by linear arrangements.

Ancilla-Assisted Depth Reduction

- Ancilla qubits precompute and store phase shifts → applied simultaneously
- QFT depth reduced from $O(n^2)$ to $O(n \log n)$ via ancilla-assisted parallelism
- Trade-off: increased qubit usage → higher susceptibility to noise on NISQ devices

Example: QFT Controlled-Phase Gates with Ancilla Qubits



Re-writing Based Peephole Optimization

- **Peephole Optimization:** A local optimization technique that examines a small sliding window ("peephole") of consecutive gates in a quantum circuit
  - Applies pattern-matching rewrite rules to replace gate sequences with cheaper equivalents
  - Reduces gate count, circuit depth, and error accumulation without changing the unitary

- Key Characteristics
  - **Local scope:** only looks at a few adjacent gates at a time (window size typically 2–5 gates)
  - **Iterative:** repeated passes until no more patterns match (fixed-point convergence)
  - **Rule-based:** driven by a library of known identities (e.g., $HH = I, CNOT \cdot CNOT = I$)
  - Analogous to peephole optimization in classical compilers (instruction-level rewrites)

Common Peephole Rewrite Rules

1. Gate Cancellation (Self-Inverse Gates)
   - Rule: $G \cdot G = I$ for any self-inverse gate G $(H, X, Y, Z, CNOT, SWAP)$
   - Example: $H\ q[0];\ H\ q[0];\ \rightarrow$ (removed, identity)
   - Example: $CNOT\ q[0], q[1];\ CNOT\ q[0], q[1];\ \rightarrow$ (removed)
2. Rotation Merging
   - Rule: $Rz(\theta_1) \cdot Rz(\theta_2) = Rz(\theta_1 + \theta_2)$ consecutive same-axis rotations merge
   - Example: $Rz(\pi/4);\ Rz(\pi/4);\ \rightarrow\ Rz(\pi/2)$ (two T gates → one S gate)
   - If $\theta_1 + \theta_2 = 2\pi$ → gate eliminated entirely
3. Gate Substitution (Decomposition Rewrites)
   - Rule: Replace an expensive gate sequence with a cheaper equivalent
   - Example: $H;\ CNOT;\ H;\ \rightarrow\ CZ$ (3 gates → 1 gate)
   - Example: $X;\ CNOT;\ X;\ \rightarrow\ CNOT$ (with swapped control/target)

Peephole Optimization: Worked Example

- Original Circuit (7 gates on qubits q[0], q[1]):

$$H\ q[0];\ \ CNOT\ q[0], q[1];\ \ H\ q[0];\ \ Rz(\pi/4)\ q[1];\ \ Rz(\pi/4)\ q[1];\ \ X\ q[1];\ \ X\ q[1];$$

Pass 1 — Gate Cancellation: $X \cdot X = I$

$$H\ q[0];\ \ CNOT\ q[0], q[1];\ \ H\ q[0];\ \ Rz(\pi/4)\ q[1];\ \ Rz(\pi/4)\ q[1];$$

Removed $X\ q[1];\ X\ q[1]$ (self-inverse cancellation)

Pass 2 — Rotation Merging: $Rz(\theta_1) \cdot Rz(\theta_2) = Rz(\theta_1 + \theta_2)$

$$H\ q[0];\ \ CNOT\ q[0], q[1];\ \ H\ q[0];\ \ Rz(\pi/2)\ q[1];$$

Merged two T gates into one S gate: $Rz(\pi/4) + Rz(\pi/4) = Rz(\pi/2)$

Pass 3 — Gate Substitution: $H \cdot CNOT \cdot H = CZ$

$$CZ\ q[0], q[1];\ \ Rz(\pi/2)\ q[1];$$

Result: 7 gates → 2 gates | Depth reduced from 7 → 2

Loop Optimization in Quantum Circuits

- **What is Loop Optimization?**
  - Many quantum algorithms apply the same sub-circuit repeatedly (e.g., Grover iterations, Trotterized time evolution, variational ansatz layers)
  - Loop optimization targets these repeated structures to reduce total gate count and depth
  - Borrowed from classical compiler theory and adapted for the quantum setting

- **Main Techniques**
  - Loop Unrolling: expand k iterations, enabling cross-iteration peephole optimizations
  - Loop-Invariant Code Motion: hoist gates that do not change between iterations outside the loop
  - Boundary Optimization: cancel gates at iteration boundaries (end of iter k, start of iter $k+1$)

- **Why It Matters**
  - Savings multiply: optimizing one loop body saves gates $\times$ number of iterations
  - Critical for NISQ devices where every gate adds decoherence error

Loop Unrolling & Boundary Optimization

Loop Unrolling — Concept

- Replace a loop of k iterations with the full expanded sequence of gates
- Exposes gate patterns across iteration boundaries for peephole optimization

Example — Grover Diffusion (3 iterations)

Each iteration ends with H and begins with H on the same qubit:

Before: $[\ldots H]\,[H\ \ldots Oracle \ldots H]\,[H\ \ldots Oracle \ldots H]\,[H\ \ldots]$

After: $[\ldots \boldsymbol{HH}\ \ldots Oracle \ldots \boldsymbol{HH}\ \ldots Oracle \ldots \boldsymbol{HH}\ \ldots]$

→ Each $\boldsymbol{HH = I}$ cancels at the boundary → saves $2n$ gates per boundary (n qubits)

Example — Trotter Steps with Rotation Merging

```
Before:  [Rz(θ/k)] × k iterations  →  k separate Rz gates
After:   Rz(θ/k) · Rz(θ/k) · ... = Rz(θ)  →  1 gate
```

→ Consecutive same-axis rotations across iterations merge into one

Loop-Invariant Code Motion (LICM)

Concept

- Identify gates inside a loop body whose effect is independent of the iteration index
- Hoist them outside the loop -> execute once instead of k times
- Requires: the gate commutes with all other gates in the loop body on the same qubit(s)

Example — Variational Ansatz with Fixed Basis Change

Loop body applies H for basis change + Rz(θ_i) parameter rotation each iteration:

Before: $for\ i = 1..k:\ H\ q[0];\ Rz(\theta_i)\ q[0];\ H\ q[0];\ CNOT\ q[0], q[1];$

After: $H\ q[0];\ \{for\ i = 1..k:\ Rx(\theta_i)\ q[0];\ CNOT\ q[0], q[1];\}\ H\ q[0];$

→ Hoisted H outside → saved 2(k−1) Hadamard gates; H·Rz·H = Rx absorbed

Example — State Preparation Inside Trotter Loop

Before: $for\ i = 1..k:\ X\ q[2];\ Rzz(t)\ q[0], q[1];\ X\ q[2];$

After: $X\ q[2];\ \{for\ i = 1..k:\ Rzz(t)\ q[0], q[1];\}\ X\ q[2];$

→ X on q[2] is invariant (acts on different qubit) → hoisted, saves 2(k−1) X gates