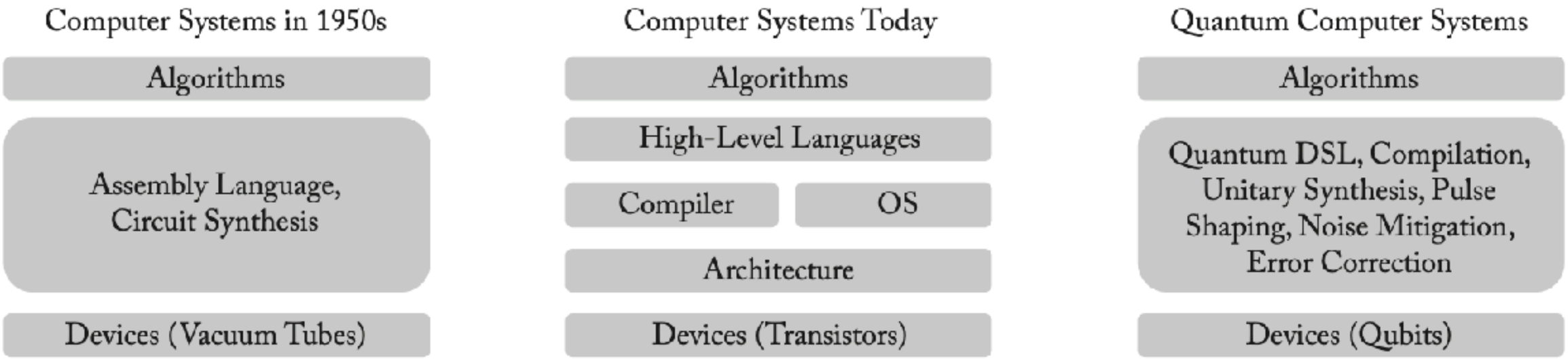
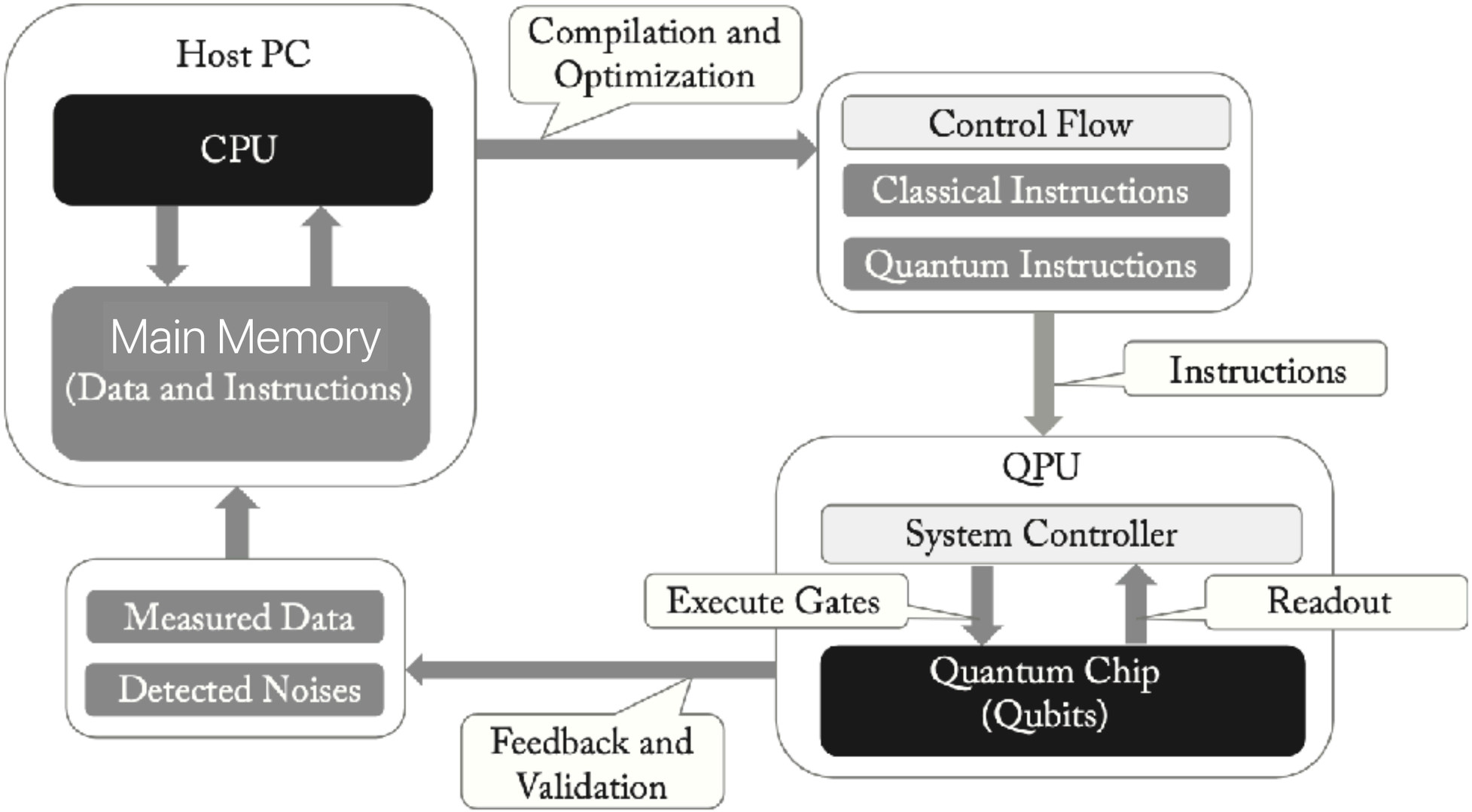
Quantum Computer Systems

Hung-Wei Tseng

Where are we in quantum computing?



Quantum Computer System



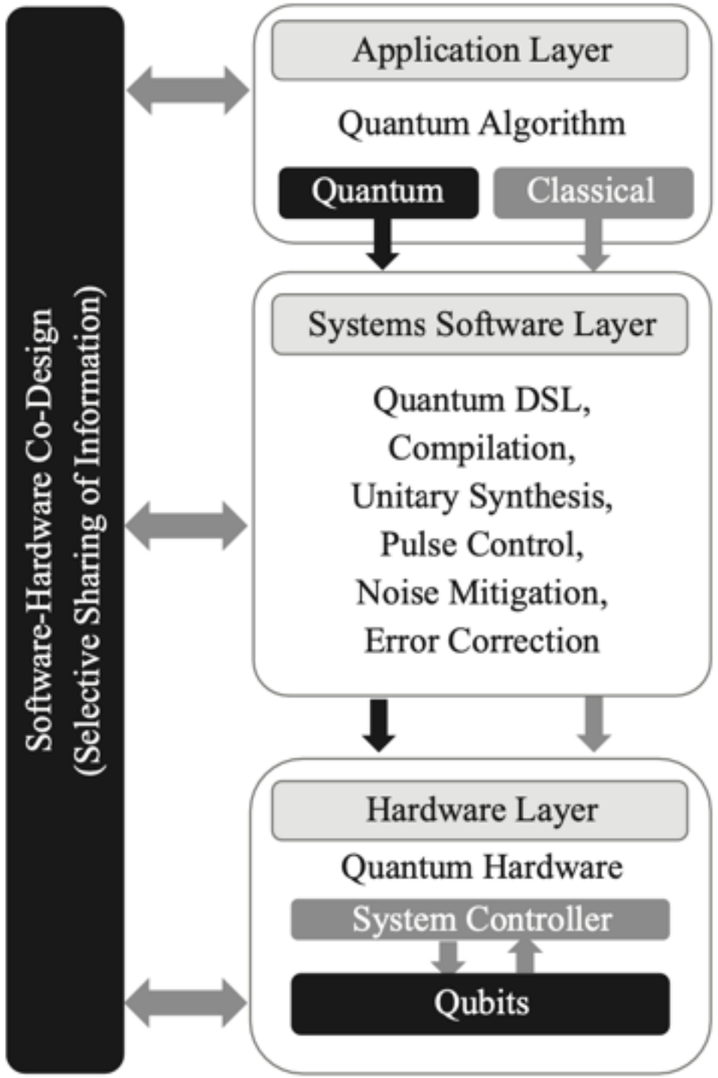
The reality of quantum computing

- Quantum supremacy — no classical computer can solve in any feasible amount of time, irrespective of the **usefulness** of the problem.
- Source of noise
  - Qubit decoherence
  - Imprecise control
  - Manufacturing defect
- Lead to non-negligible detrimental effects and can accumulate when running long quantum algorithms, necessitating noise mitigation techniques to protect the information during the computation

NISQ (Noisy Intermediate-Scale Quantum) machine

- Fault-tolerance machine (FT machine)
  - Use quantum error correction (QEC) to achieve system-level fault tolerance
  - overheads of conventional QEC approaches are found to be inhibitory in the near term.
  - ~50-100 noisy qubits require millions of qubits for fully error corrected algorithms like Shor's factoring
- NISQ machine
  - Noisy
    - No continuous quantum error correction
    - Prone to quantum decoherence
  - Intermediate-scale
    - The moderate number of qubits — up to 1,000 qubits
    - Gate fidelity

NISQ computer system stack



NISQ — Live with noise

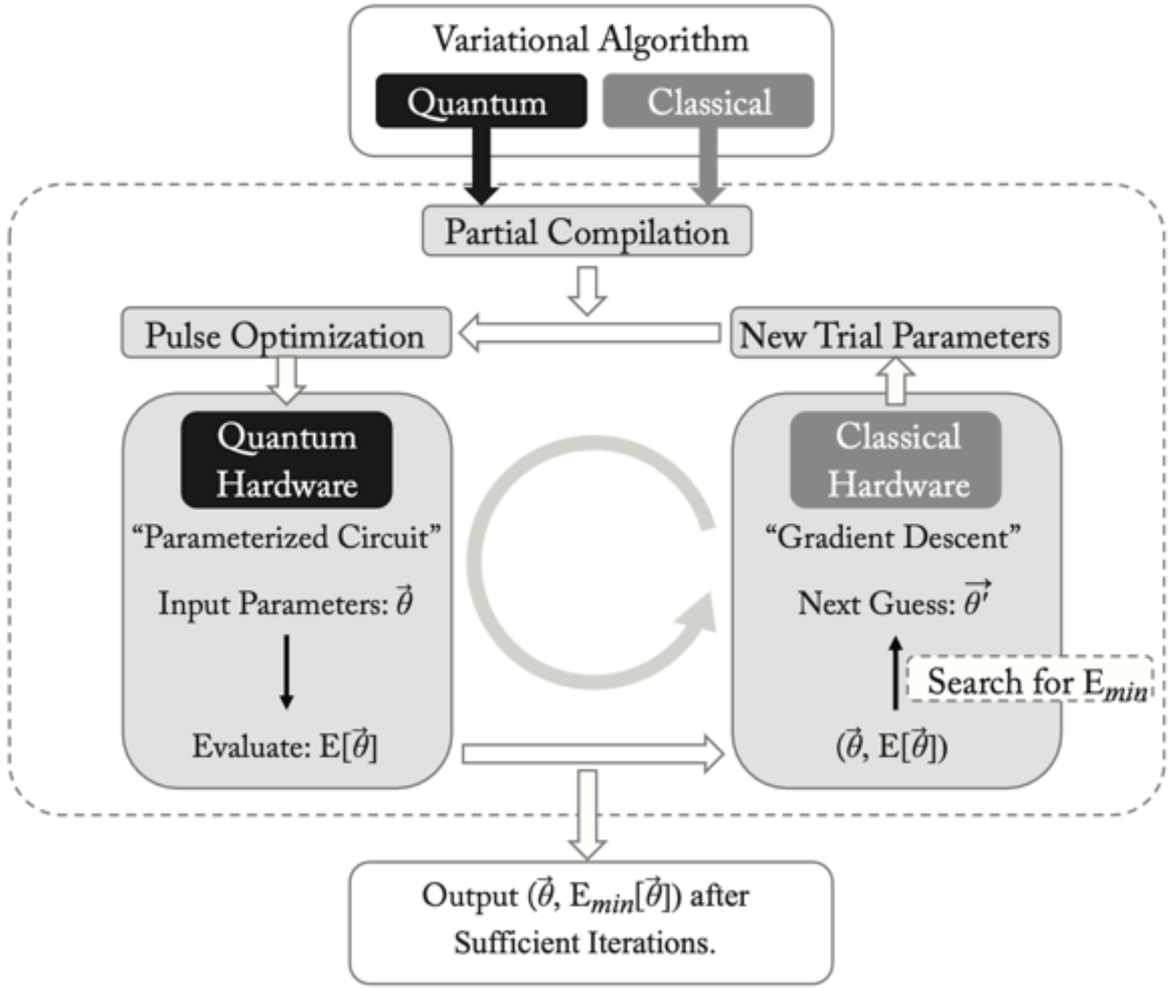
- Algorithm — tolerable to noise
- Compilation
  - Taking care of the noise aggregation and device deficiencies

Rethink algorithms in NISQ era

Variational principle

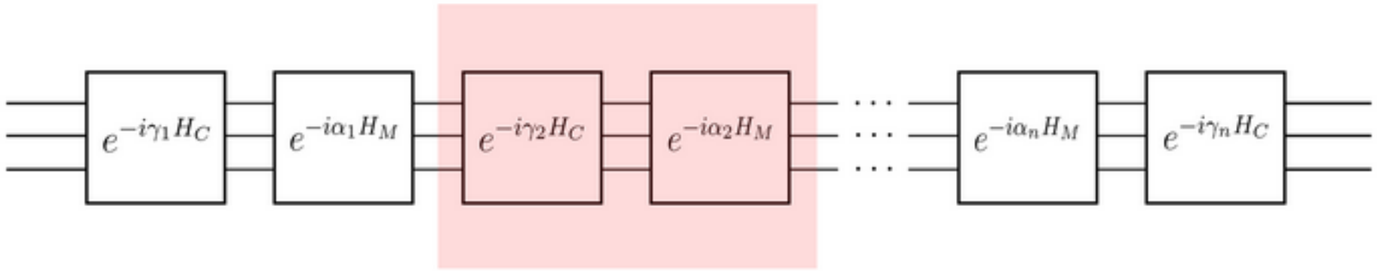
- $\langle \psi | H | \psi \rangle \geq E_0$
- H is hermitian, which implies that H has a complete set of eigenvectors which are orthonormal to each other.
- guess and check
  - check is measuring the $\langle H \rangle$ and the guess is using a classical optimizer to select the next value of θ based on previous results.

Variational quantum eigensolver (VQE)



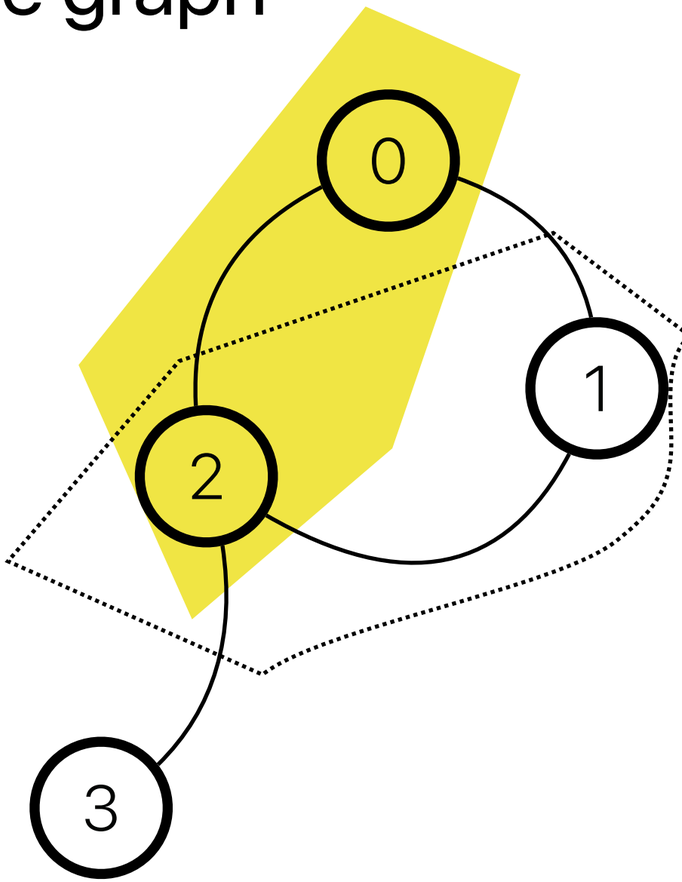
Quantum Approximate Optimization Algorithm

- Defining a cost Hamiltonian H_C such that its ground state encodes the solution to the optimization problem
- Defining a mixer Hamiltonian H_M
- Defining the oracles $U_C(\gamma) = \exp(-\imath\gamma H_C)$ and $U_M(\alpha) = \exp(-\imath\alpha H_M)$, with parameters γ and α
- Repeated application of the oracles U_C and U_M, in the order: $U(\boldsymbol{\gamma}, \boldsymbol{\alpha}) = \prod_{i=1}^{N} (U_C(\gamma_i) U_M(\alpha_i))$
- Preparing an initial state, that is a superposition of all possible states and apply $U(\boldsymbol{\gamma}, \boldsymbol{\alpha})$ to the state
- Using classical methods to optimize the parameters γ and α and measure the output state of the optimized circuit to obtain the approximate optimal solution to the cost Hamiltonian. An optimal solution will be one that maximizes the expectation value of the cost Hamiltonian H_C.



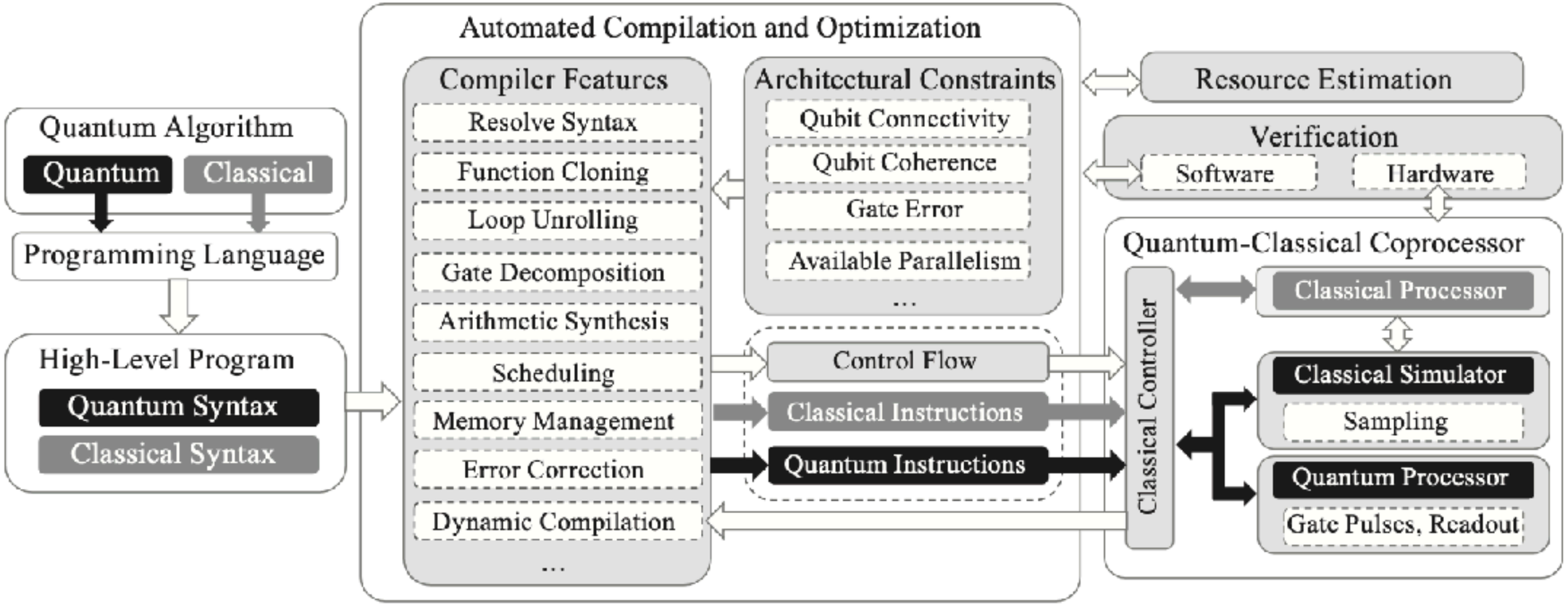
Minimum vertex cover

- a collection of vertices such that each edge in the graph contains at least one of the vertices in the cover
- In the figure, we can map the solution as $|1010\rangle$ and $|0110\rangle$
- The goal of the algorithm is to sample these bit strings with high probability
- The cost Hamiltonian has two ground states, $|1010\rangle$ and $|0110\rangle$, coinciding with the solutions of the problem



From Qiskit to QC

Quantum compilation



QASM

```
qubit  q0
qubit q1
H q0
CNOT  q0,q1
Measure q0
Measure q1
```

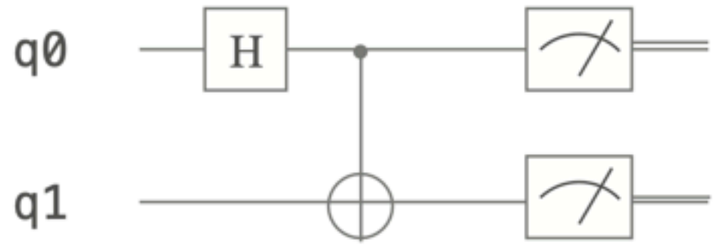


Figure 5.1: The QASM code and circuit diagram for creating an EPR pair with measurements.

Another high-level programming language — Microsoft Q#

```
// The Q# compiler automatically detects the Main() operation as the entry point.

operation Main() : Result {
    // Allocate a qubit. By default, it's in the 0 state.
    use q = Qubit();
    // Apply the Hadamard operation, H, to the state.
    // It now has a 50% chance of being measured as 0 or 1.
    H(q);
    // Measure the qubit in the Z-basis.
    let result = M(q);
    // Reset the qubit before releasing it.
    Reset(q);
    // Return the result of the measurement.
    return result;
}
```

Qiskit

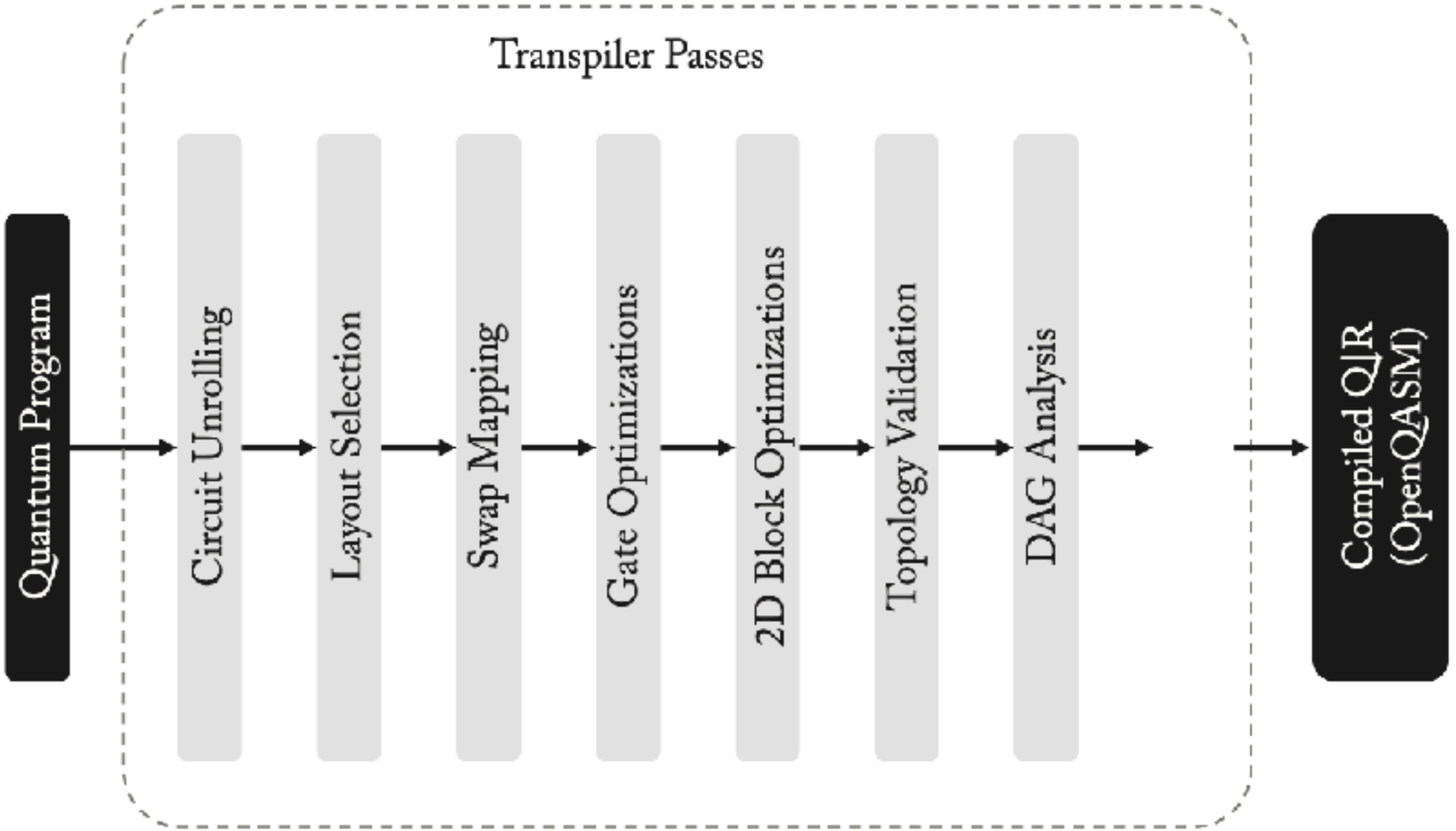
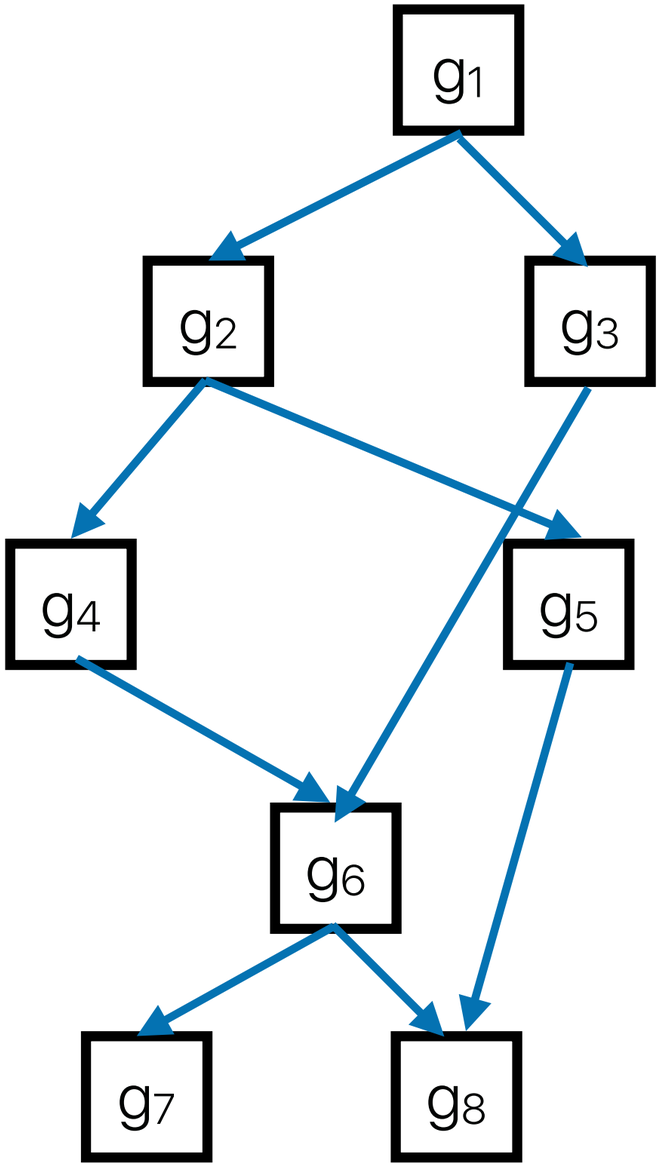


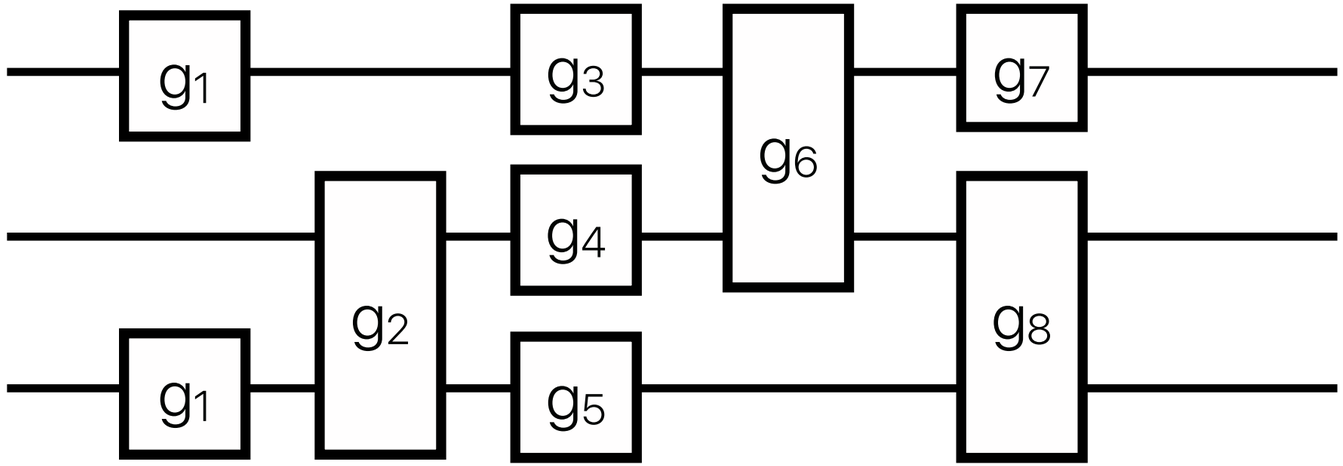
Figure 6.5: The compilation framework in Qiskit Terra developed by IBM [165].

Scheduling constraints

- Data dependency
- Connectivity
- Commutation Relations
- Qubit Decoherence

Data dependency





Commutation Relations

- The commutator of two gates (operators) A, B is defined as

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

- Two gates A and B are conjugate, if there exists another gate U such that

$$UA = BU$$

Commutation-relaxed data dependency graph (CRDDG)

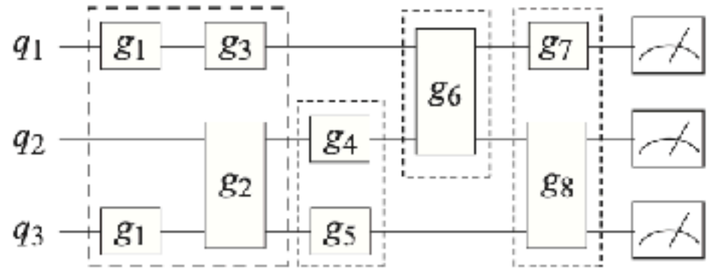


Figure 6.8: A sample quantum circuit with commutation relations indicated by dashed boxes.

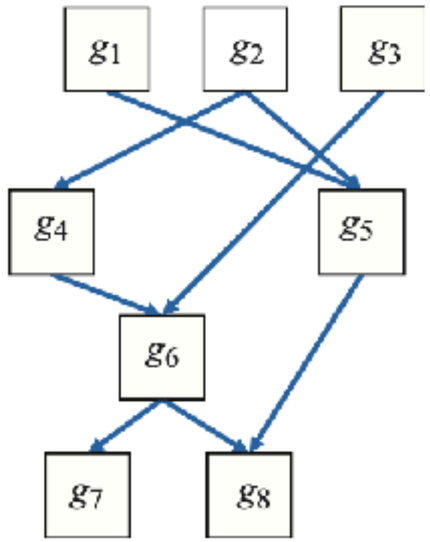


Figure 6.9: The commutation-relaxed data dependency graph (CRDDG) of the sample quantum circuit in Figure 6.8.

Qubit Decoherence

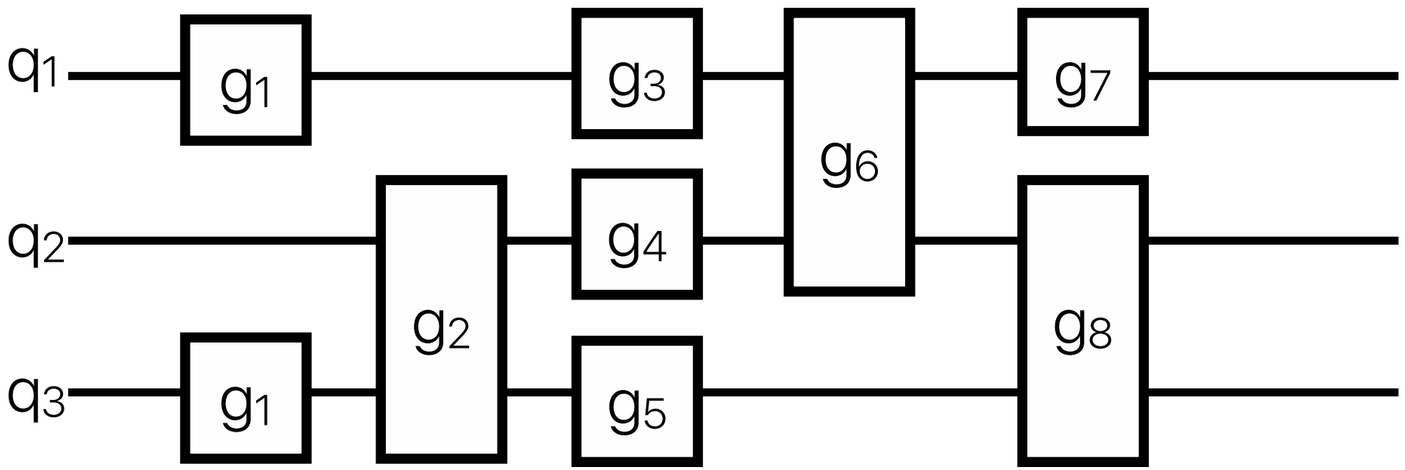
- Qubits have limited lifetime due to spontaneous decoherence — idle noise on the qubits
- The number of gates safe to execute on a qubit before the chance of decoherence
- Minimize the circuit depth
  - Good for latency as well

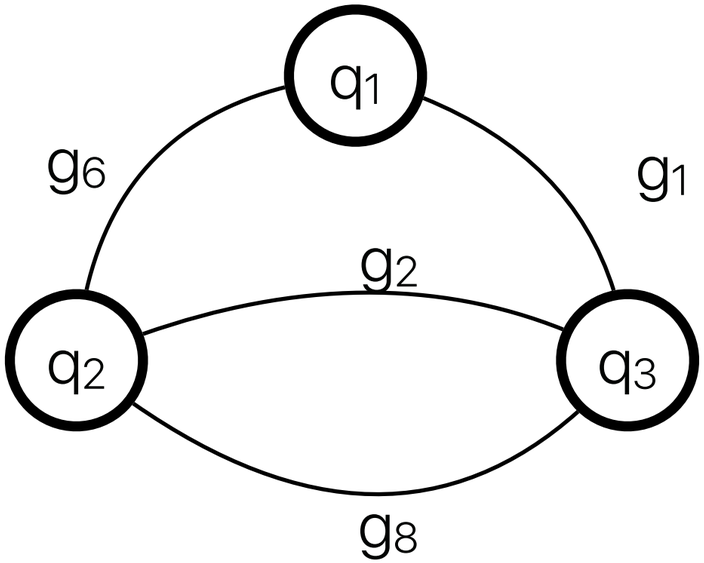
Scheduling strategies

- ALAP (As-Late-As-Possible) Scheduling
  - Qiskit uses this
  - starts with the end of the circuit and schedules the last gates needed to be completed, and goes backward to their previous gates
  - qubits are initialized only when absolutely needed — good for lifetime
- LPF (Longest-Path-First) Scheduling
  - Reducing the circuit depth
- Communication-Aware Scheduling
  - When a two-qubit gate is known to be causing high communication cost
- Adaptive Scheduling

Qubit mapping & reuse

Program interaction graph

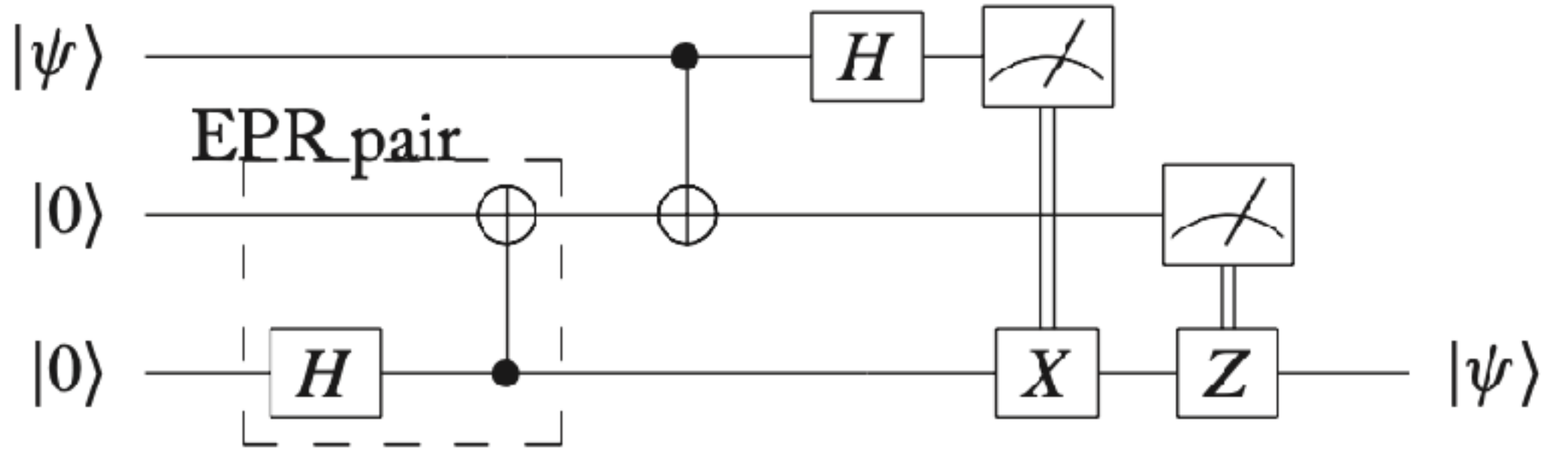




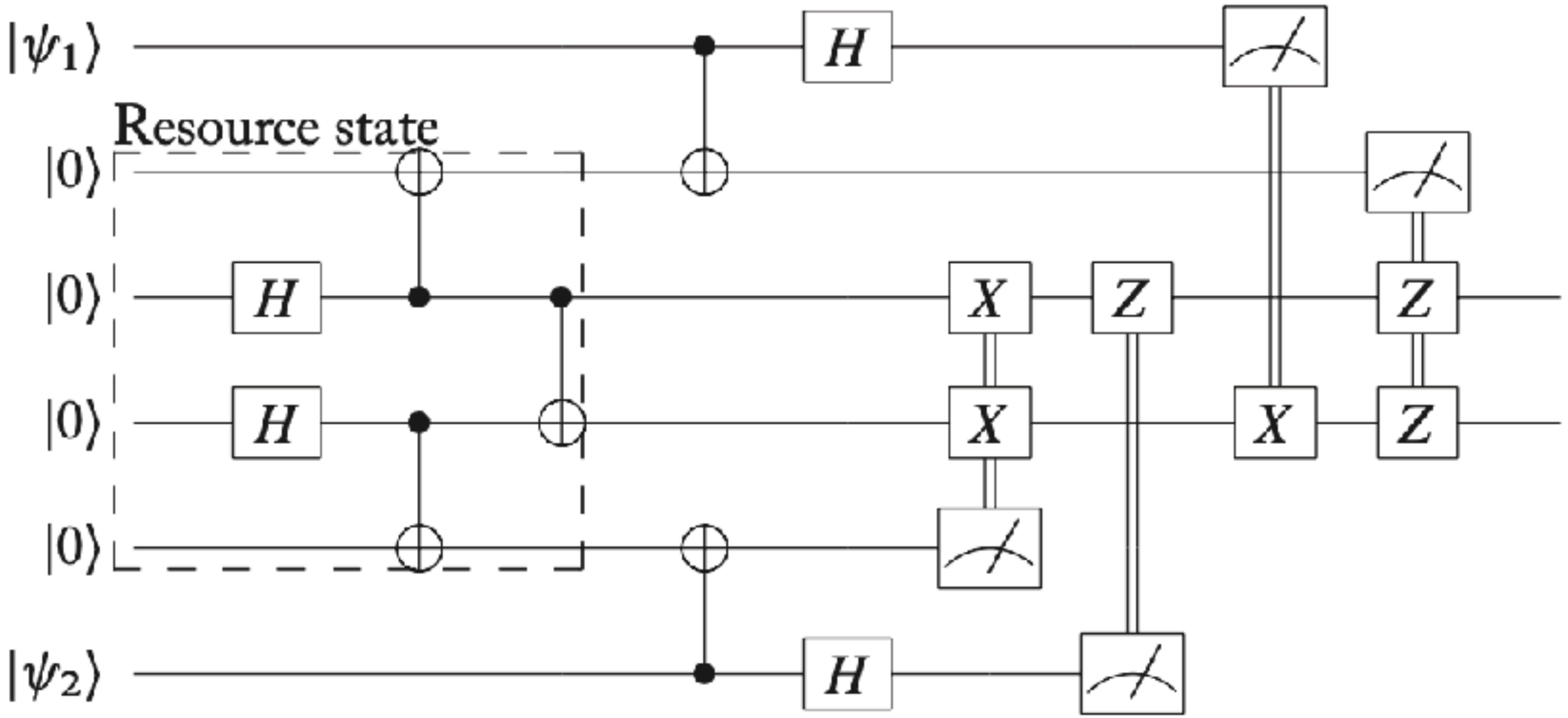
Find a good mapping

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

Gate teleportation



Gate teleportation



How to reuse a qubit

- Reclaiming Qubits via Measurement and Reset — NISQ does not support
- Reusing Qubits via Borrowing

Dirty borrow allows us to reuse any qubit

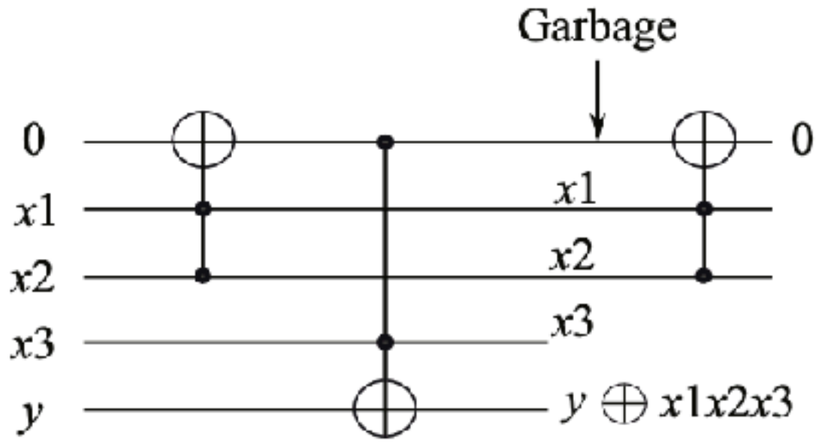


Figure 6.13: Implementation of $\Lambda^3(X)$ gate using an ancilla qubit in $|0\rangle$.

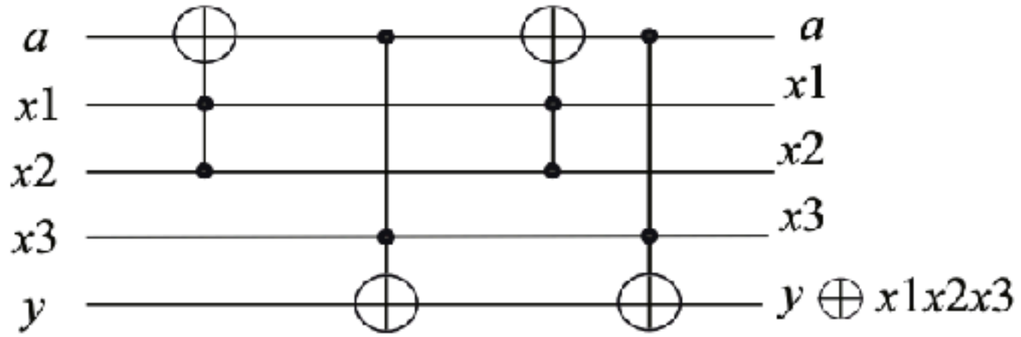
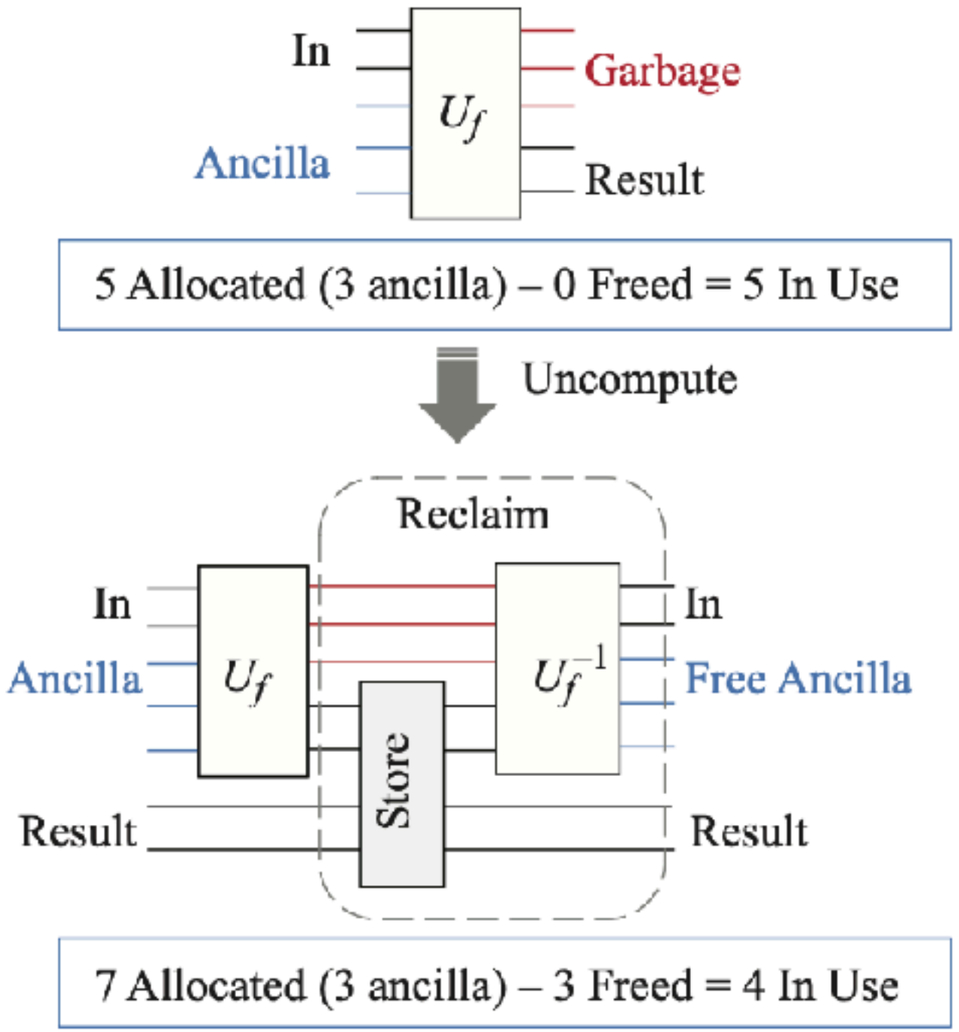


Figure 6.14: Implementation of $\Lambda^3(X)$ gate using an ancilla qubit in an arbitrary state.

How to reuse a qubit

- Reclaiming Qubits via Measurement and Reset — NISQ does not support
- Reusing Qubits via Borrowing
  - Need to return the borrowed qubits to their original states timely
  - The borrowing computation is restricted
- Uncomputation

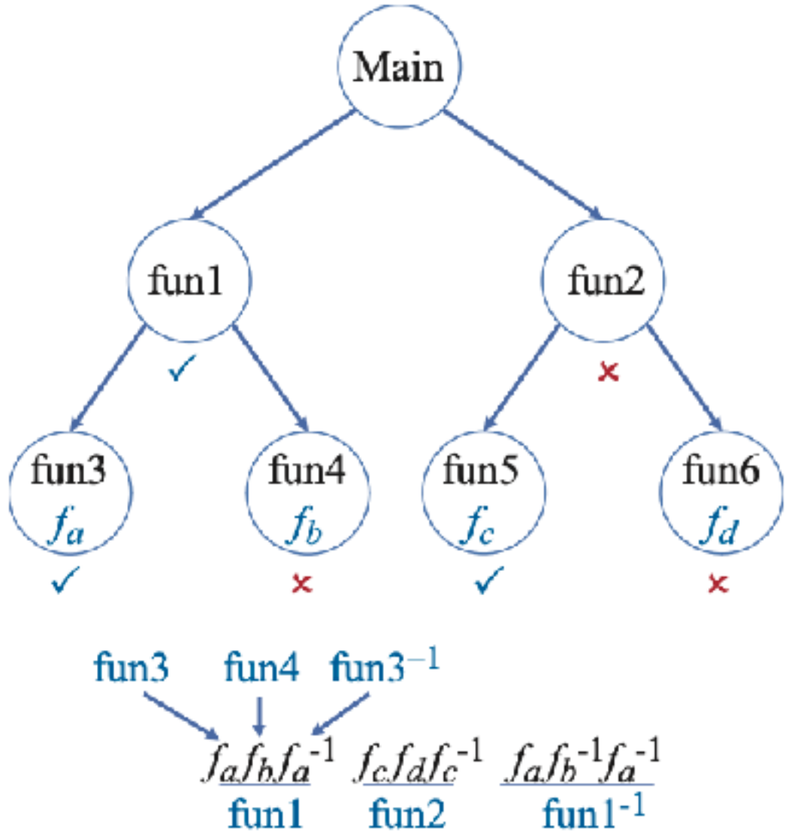

In
Garbage
U_f
Ancilla
Result
5 Allocated (3 ancilla) – 0 Freed = 5 In Use
Uncompute
Reclaim
In
Ancilla
U_f
Store
U_f^{-1}
In
Free Ancilla
Result
Result
7 Allocated (3 ancilla) – 3 Freed = 4 In Use

How to reuse a qubit

- Reclaiming Qubits via Measurement and Reset — NISQ does not support
- Reusing Qubits via Borrowing
  - Need to return the borrowed qubits to their original states timely
  - The borrowing computation is restricted
- Uncomputation
  - Additional gate cost
  - Only works if the circuit U_f is a classical reversible transformation

Optimizations of Uncomputation

- Reversible Pebbling Game
- Heuristic-Based Noise-Aware Uncomputation



Paper presentations

- 3/11/2025 Haotian Lu — Hanrui Wang, Zirui Li, Jiaqi Gu, Yongshan Ding, David Z. Pan, and Song Han. OC: quantum on-chip training with parameter shift and gradient pruning. In the 59th ACM/IEEE Design Automation Conference (DAC '22)
- 3/11/2025
- 3/13/2025
- 3/13/2025
- Topics
  - Quantum optimization algorithms
  - Quantum compilers
  - Quantum architectures/memory

Announcement

- No lecture this Thursday
- Assignment #1 due next Tuesday — check the link from the website and submit through gradescope