COE 530
Quantum Computers and Architecture

Lecture 14

Gate Scheduling and Resource Allocation

References:

Ch 6 Quantum Computing Systems

Why Gate Scheduling Matters

NISQ Era

- Decoherence budget is tight -> every idle cycle costs fidelity
- Minimizing circuit depth is critical to preserving quantum state

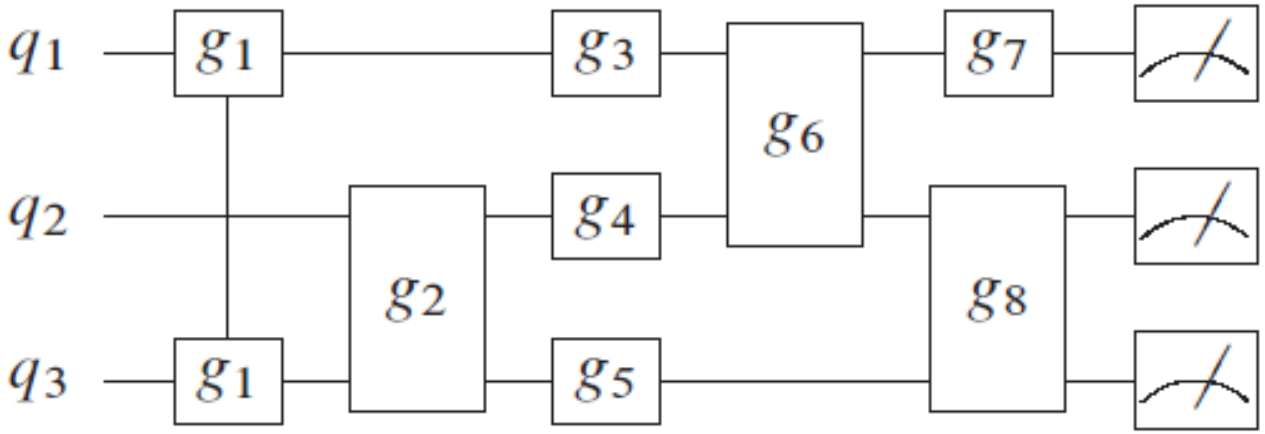
Fault-Tolerant Era

- T-gates and magic states dominate cost
- Depth translates directly to wall-clock time and physical qubit-seconds

Two distinct regimes, shared abstractions: circuit DAG, gate set, hardware graph

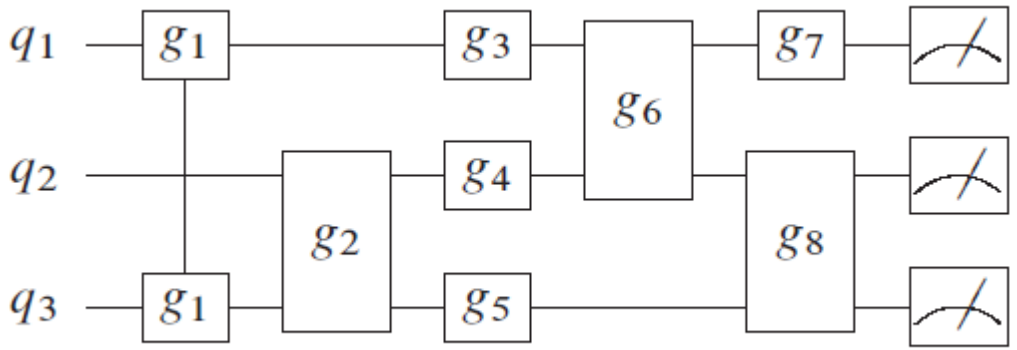
Gate Scheduling

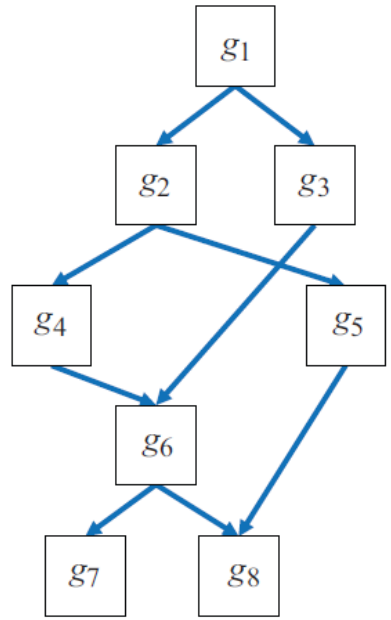
- A *schedule* of a quantum program is a sequence of gate operations on logical qubits
- The sequence ordering defines *data dependencies* between gates, where a gate g_2 depends on g_1 if they share a logical qubit and g_2 appears later in the schedule than g_1



Data Dependency

- Some gates are required to be applied after the completion of previous gates.
- This requirement can come from sharing of qubits, or specifications in the quantum algorithm
- To facilitate gate scheduling, we usually a data structure called Data Dependency Graph (DDG)
- A DDG is graph $G = (V, E)$ where
  - V is set of gates
  - E is set of edges, s.t., $\exists(\mathrm{v_i}, \mathrm{v_j})$ if v_j depends on v_i





Building DDG from a Circuit

Algorithm: Construct DDG from a quantum circuit

Given a circuit with gates operating on qubits:

1. Create a node for each gate in the circuit
2. For each qubit, scan gates from left to right in the circuit
3. For consecutive gates on the same qubit, add a directed edge from the earlier gate to the later gate
4. Repeat for all qubits to capture all dependencies

- **Key insight:** Two gates share an edge if and only if they operate on a common qubit and appear consecutively on that qubit in the circuit

- The resulting DDG should be DAG (Directed Acyclic Graph) that captures the partial ordering of gates

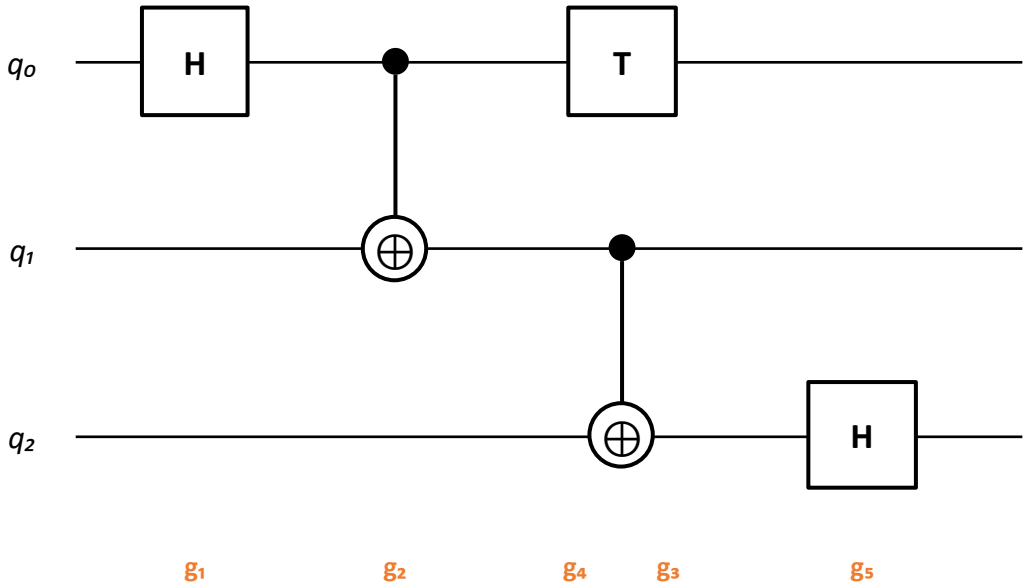
Circuit DAG: Key Metrics

Key Metrics

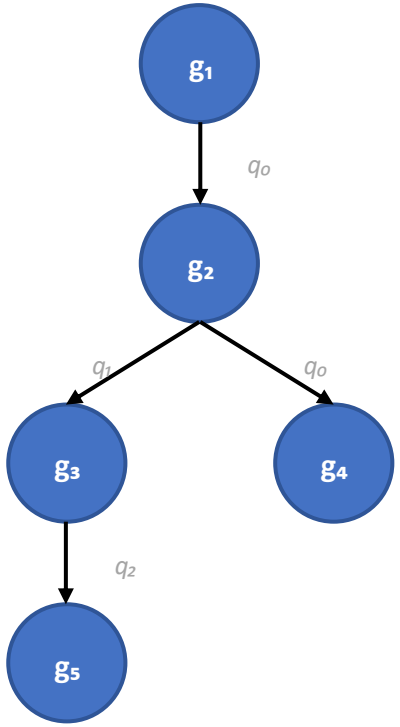
- **Critical path** = longest path in DAG = logical depth (minimum time steps)
- **Width** = number of qubits = maximum parallelism in any time slice
- **Gate count** = total nodes = proxy for overall resource cost

DDG Construction: Example

Sample Circuit



Resulting DDG



Construction:

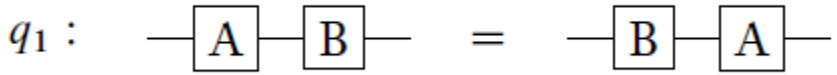
Scheduling Considerations

- **Commutation relation**

Definition: The commutator of two gates (operators) A and B is defined as

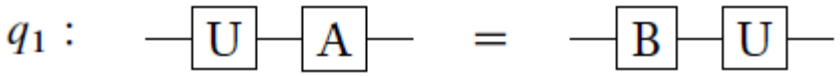
$$[A, B] = AB - BA$$

If $[A, B] = 0$, then we say that A and B commute.



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

$$UAU^\dagger = B \Rightarrow UAU^\dagger U = BU \Rightarrow UA = BU$$



Commutation Relations & DAG Reduction

When Can Two Gates Be Swapped?

- Two CNOTs sharing only the control qubit commute
- Z gates commute through CNOT controls
- Diagonal gates on the same qubit always commute

Commutation-Aware DAG Reduction

- Gate merging: combine adjacent unitaries into one
- Cancellation: inverse gate pairs eliminated
- Template-based rewriting: pattern-match and replace sub-circuits

Ref: Itoko et al. (commutation-aware transpilation); Nam et al. (automated optimization)

Enhanced Data Dependency Graph

- Commutation-relaxed date dependency graph (CRDDG)

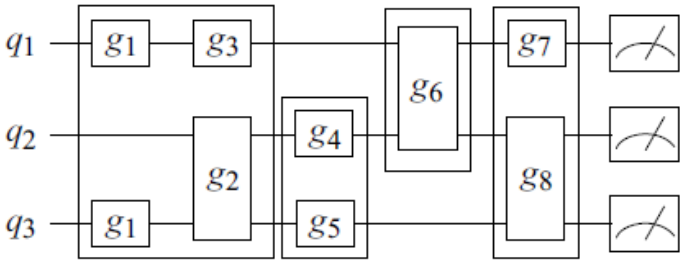
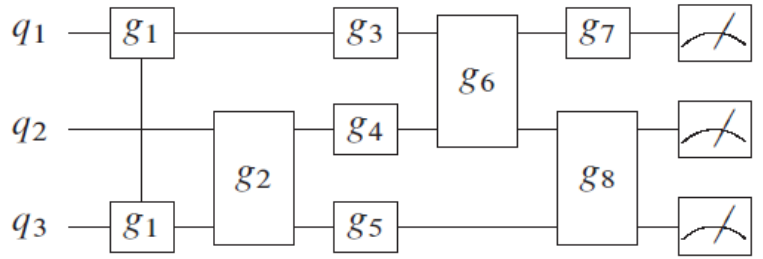


Figure 6.8: A sample quantum circuit with commutation relations indicated by dashed boxes.

Initial circuit



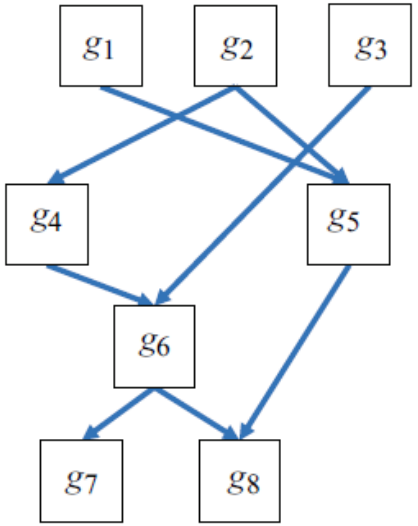
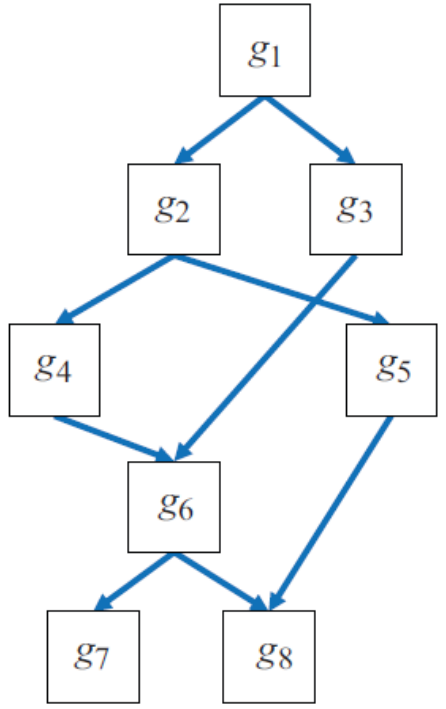
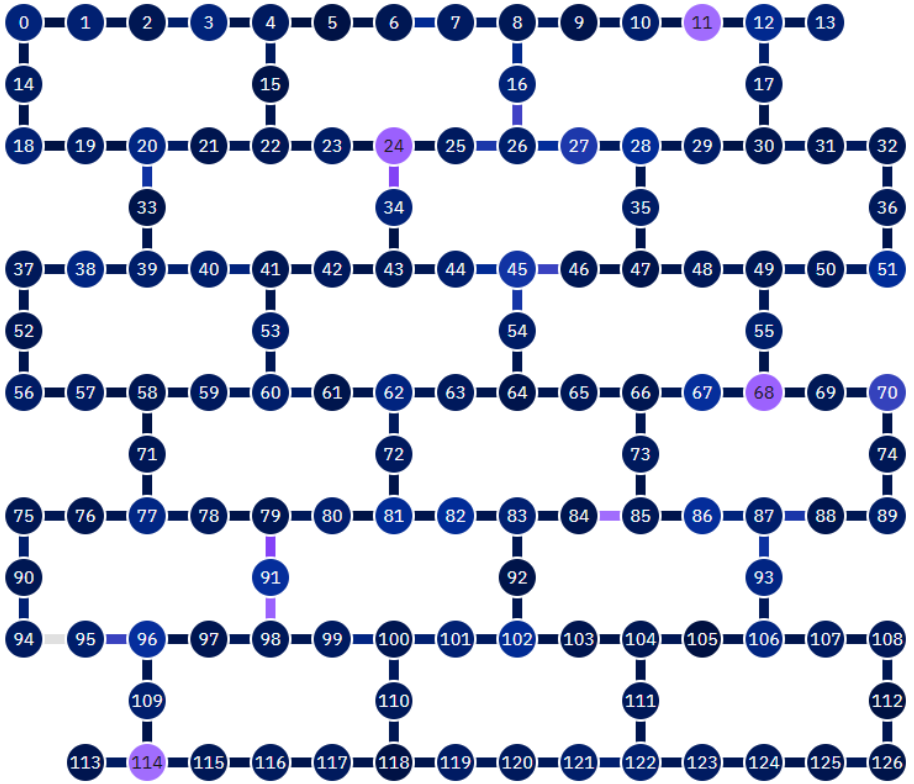


Figure 6.9: The commutation-relaxed data dependency graph (CRDDG) of the sample quantum circuit in Figure 6.8.



Scheduling Considerations

- Hardware Limitation:
  - Two-qubit gates only work between adjacent qubits; SWAP gates needed otherwise
  - Crosstalk: simultaneous gates on nearby qubits increase errors
  - Scheduler must balance parallelism vs. noise tradeoffs



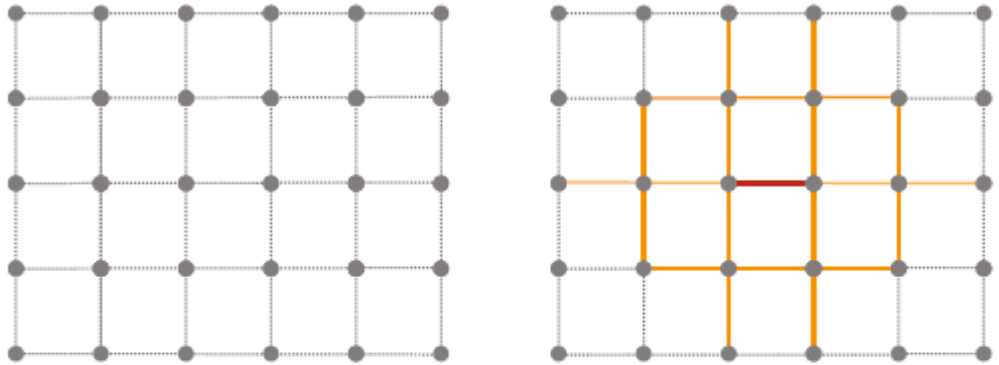
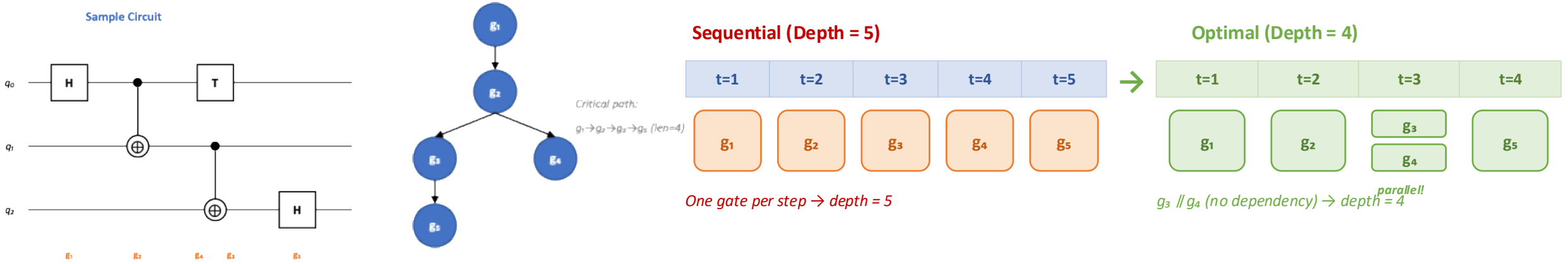


Figure 6.7: **Left:** the connectivity graph of a 5×6 2D mesh. Vertices are qubits, and two vertices are connected if the qubits are connected. **Right:** when the two center qubits are chosen for a two-qubit gate at an interaction frequency (highlighted in red), the surrounding qubits must be tuned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

Scheduling Goal: Circuit Depth

- A scheduler is responsible for maximizing the utilization of the qubits
- The common goal in scheduling quantum circuit is to minimize the *circuit depth*, i.e. the longest critical path of the circuit
  - If the circuit depth is bounded within the lifetime of any qubits, then the circuit is generally safe from decoherence noise

- Example



Scheduling Goal: Space-time Volume

- Space-time volume (V) metric accounts for the active usage of each qubit

$$V = \sum_{q \in Q} \sum_{(t_i, t_f) \in T_q} (t_f - t_i)$$

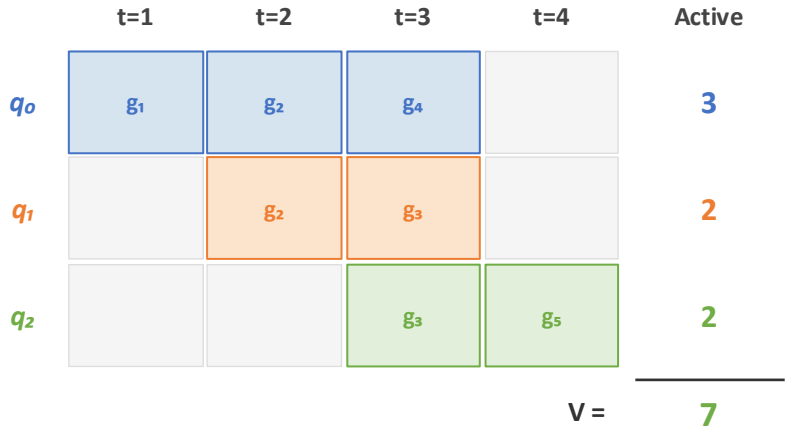
Where Q is the set of all qubits

T_q is a sequence of pairs $\left\{\left(t_i^0, t_f^0\right), \left(t_i^1, t_f^2\right), \ldots, \left(t_i^{|T_q|-1}, t_f^{|T_q|-1}\right)\right\}$

t_i^k and t_f^k are the allocation and reclamation times of the k^{th} time of qubit q

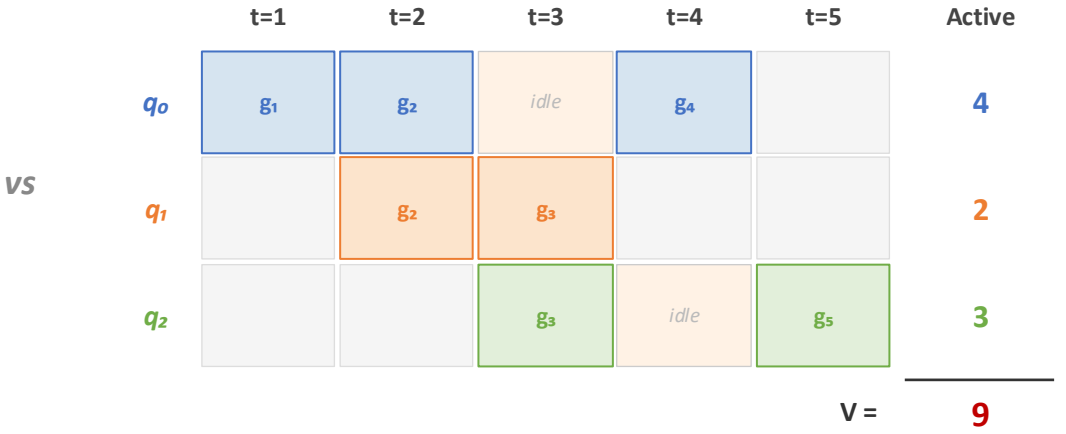
Space-Time Volume: Example

Optimal Schedule (Depth = 4)



vs

Sequential Schedule (Depth = 5)



Calculation Details

Optimal Schedule (V = 7)

- q_0: first gate g_1 at t=1, last gate g_4 at t=3 → 3
- q_1: first gate g_2 at t=2, last gate g_3 at t=3 → 2
- q_2: first gate g_3 at t=3, last gate g_5 at t=4 → 2

V = 3 + 2 + 2 = 7

Sequential Schedule (V = 9)

- q_0: first g_1 at t=1, last g_4 at t=4 → 4 (idle at t=3)
- q_1: first g_2 at t=2, last g_3 at t=3 → 2
- q_2: first g_3 at t=3, last g_5 at t=5 → 3 (idle at t=4)

V = 4 + 2 + 3 = 9

Insight: Parallelism reduces both depth (5→4) and space-time volume (9→7) by eliminating idle qubit time

ASAP & ALAP Scheduling

ASAP (As Soon As Possible)

- Schedule each gate at the earliest time all dependencies are met
- Greedy forward pass through DAG
- Minimizes start time per gate
- Preferred when reducing idle time matters most

ALAP (As Late As Possible)

- Schedule each gate at the latest time that doesn't delay successors
- Backward pass from circuit outputs
- Defers operations to reduce qubit lifetime

- ASAP and ALAP are duals same depth, different slack distribution

Gate Slack

Definition

The *slack* of a gate *g* is defined as:

$$slack(g) = ALAP(g) - ASAP(g)$$

where ASAP(g) is the earliest time step *g* can be scheduled and ALAP(g) is the latest time step *g* can be scheduled without increasing circuit depth

Properties

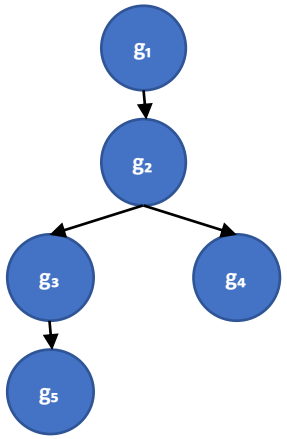
- $slack(g) \geq 0$ for all gates *g*
- A gate *g* is **critical** if $slack(g) = 0$, meaning it has no scheduling flexibility
- The **critical path** consists of all gates with $slack = 0$
- Gates with $slack > 0$ can be moved within [ASAP(g), ALAP(g)] without affecting circuit depth

Significance

- Slack identifies scheduling freedom — useful for optimizing secondary objectives (e.g., reducing qubit lifetime, minimizing crosstalk) without increasing depth
- Critical gates are the bottleneck: any delay to a critical gate increases overall circuit depth

ASAP & ALAP Scheduling: Example

DDG



ASAP Schedule (Depth = 4)



g₄ slides →

ALAP Schedule (Depth = 4)



Slack Analysis

Gate	ASAP	ALAP	Slack	Critical?
g_1	t = 1	t = 1	0	Yes
g_2	t = 2	t = 2	0	Yes
g_3	t = 3	t = 3	0	Yes
g_4	t = 3	t = 4	1	No
g_5	t = 4	t = 4	0	Yes

Key Observations

- **Both produce depth = 4** (optimal for this DDG), but distribute gates differently across time steps
- **Slack = ALAP(g) − ASAP(g):** g_4 has slack = 1 (can be at t=3 or t=4). All other gates have slack = 0 (critical)
- **Critical path:** $g_1 \rightarrow g_2 \rightarrow g_3 \rightarrow g_5$ (all slack-0 gates form the critical path)

Overall Scheduling Objectives and Outcomes

List Scheduling

- Maintain a ready list of gates whose dependencies are satisfied
- Priority heuristics: critical-path length, mobility (slack), gate type
- Select highest-priority ready gate at each time step

Depth Minimization

- Equivalent to a constraint satisfaction / optimization problem
- Related to classical job-shop scheduling — NP-hard in general
- Heuristics achieve near-optimal results for practical circuit sizes

Parallelization Opportunities

- Disjoint-qubit gates can execute in the same time slice
- Maximal parallelism = anti-chain width in the DAG

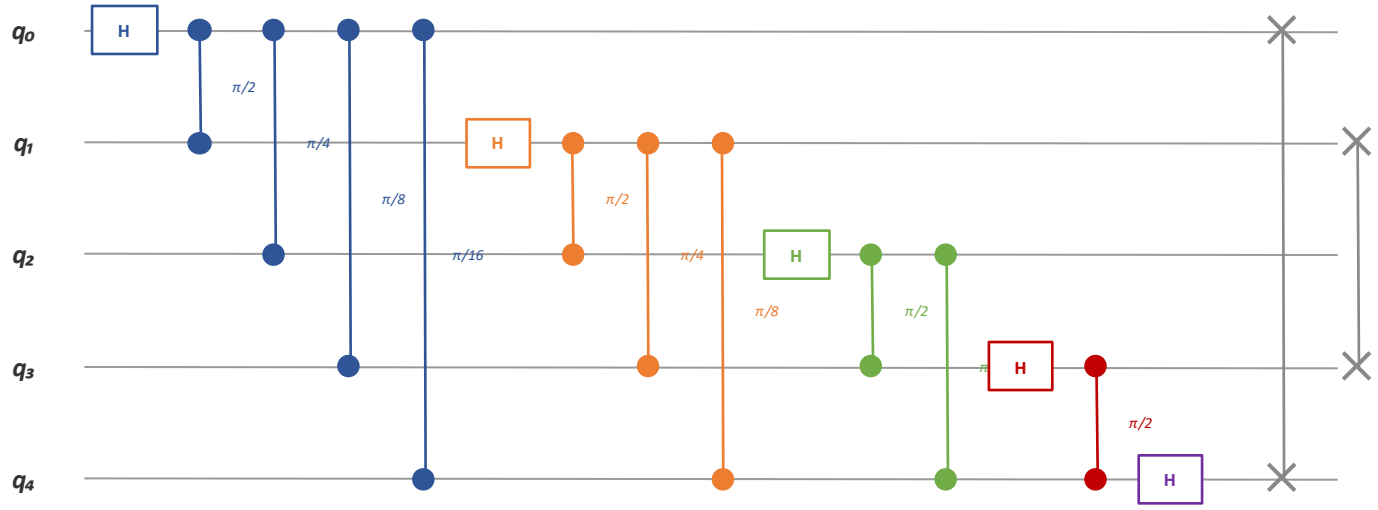
Worked Example

Worked Example: 5-Qubit QFT

- **All-to-all connectivity:** all CPhase gates placed with no routing overhead
- **Linear connectivity:** SWAP gates required; depth increases significantly
- Compare: time slices, total gates, CNOT overhead for each topology

5-Qubit QFT: Circuit & Gate List

Circuit Diagram



Gate List (17 gates)

#	Gate	Qubits
1	H	q_0
2	CR(π/2)	q_0,q_1
3	CR(π/4)	q_0,q_2
4	CR(π/8)	q_0,q_3
5	CR(π/16)	q_0,q_4
6	H	q_1
7	CR(π/2)	q_1,q_2
8	CR(π/4)	q_1,q_3
9	CR(π/8)	q_1,q_4

#	Gate	Qubits
10	H	q_2
11	CR(π/2)	q_2,q_3
12	CR(π/4)	q_2,q_4
13	H	q_3
14	CR(π/2)	q_3,q_4
15	H	q_4
16	SWAP	q_0,q_4
17	SWAP	q_1,q_3

Circuit Structure

- 5 Hadamard gates (one per qubit), 10 controlled-rotation gates CR($\pi/2^k$), 2 SWAP gates = 17 total gates
- Standard QFT pattern: for qubit q_i, apply H then CR gates with q_{i+1}, q_{i+2}, …, q_4
- Gate ordering creates a cascade of dependencies: each qubit group must wait for the previous group to complete

Dependencies (DDG edges)

- **Same-qubit:** consecutive gates on any qubit (e.g., $g_1 \rightarrow g_2$ since both touch q_0)
- **Cross-qubit:** two-qubit gates create edges on both participating qubits (e.g., g_2 on q_0,q_1 links to g_6 on q_1)
- **Key insight:** gates in different qubit groups can overlap if their qubit sets don't conflict

5-Qubit QFT: ASAP Schedule

ASAP Schedule (All-to-All Connectivity)

	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
q_0	g_1:H	g_2:CR	g_3:CR	g_4:CR	g_5:CR					g_{16}:SW
q_1		g_2:CR	g_6:H	g_7:CR	g_8:CR	g_9:CR			g_{17}:SW	
q_2			g_3:CR	g_7:CR	g_{10}:H	g_{11}:CR	g_{12}:CR			
q_3				g_4:CR	g_8:CR	g_{11}:CR	g_{13}:H	g_{14}:CR	g_{17}:SW	
q_4					g_5:CR	g_9:CR	g_{12}:CR	g_{14}:CR	g_{15}:H	g_{16}:SW

Schedule Metrics

Depth
10 time steps

Total Gates
17 5H + 10CR + 2SW

Max Parallelism
3 at t = 5

Utilization
34% 17 / (5×10) slots

Observations

- Sequential depth would be 17; ASAP achieves 10 → 1.7× speedup from parallelism
- q_0 finishes early (idle t=6–9), then re-engages for SWAP at t=10; q_2 finishes at t=7 with no further gates
- Critical path: $g_1 \rightarrow g_2 \rightarrow g_3 \rightarrow g_7 \rightarrow g_8 \rightarrow g_9 \rightarrow g_{12} \rightarrow g_{14} \rightarrow g_{15} \rightarrow g_{16}$ (length = 10)

5-Qubit QFT: Slack & Space-Time Analysis

Slack Analysis

Gate	Type	ASAP	ALAP	Slack	Critical?
g_1	$H(q_0)$	1	1	0	Yes
g_2	$CR(q_0,q_1)$	2	2	0	Yes
g_3	$CR(q_0,q_2)$	3	3	0	Yes
g_4	$CR(q_0,q_3)$	4	4	0	Yes
g_5	$CR(q_0,q_4)$	5	5	0	Yes
g_6	$H(q_1)$	3	3	0	Yes
g_7	$CR(q_1,q_2)$	4	4	0	Yes
g_8	$CR(q_1,q_3)$	5	5	0	Yes
g_9	$CR(q_1,q_4)$	6	6	0	Yes
g_{10}	$H(q_2)$	5	5	0	Yes
g_{11}	$CR(q_2,q_3)$	6	6	0	Yes
g_{12}	$CR(q_2,q_4)$	7	7	0	Yes
g_{13}	$H(q_3)$	7	7	0	Yes
g_{14}	$CR(q_3,q_4)$	8	8	0	Yes
g_{15}	$H(q_4)$	9	9	0	Yes
g_{16}	$SWAP(q_0,q_4)$	10	10	0	Yes
g_{17}	$SWAP(q_1,q_3)$	9	10	1	No

Space-Time Volume Comparison

Qubit	ASAP First	ASAP Last	ASAP V_i	ALAP V_i
q_0	t=1	t=10	**10**	**10**
q_1	t=2	t=9	**8**	**9**
q_2	t=3	t=7	**5**	**5**
q_3	t=4	t=9	**6**	**7**
q_4	t=5	t=10	**6**	**6**
Total			**35**	**37**

ASAP
V = 35
Depth = 10

ALAP
V = 37
Depth = 10

Difference
ΔV = +2
+5.7% overhead

Key Insights

- **16/17 gates are critical** — QFT has very tight dependencies with almost no scheduling flexibility
- **Only g_{17} has slack = 1:** $SWAP(q_1,q_3)$ can be at t=9 or t=10 since it has no successors
- **ALAP increases V (37 vs 35):** deferring g_{17} extends q_1 and q_3 lifetimes — ASAP preferred for minimizing qubit resources

Hardware-aware Scheduling

- So far, we have been discussing hardware-agnonstic scheduling
- To connect the dots, we need to consider hardware limiations when scheduling -> more complexity

Coupling Graph Constraints

- Not every 2-qubit gate is physically implementable
- Hardware defines a graph of allowed qubit interactions
- Common topologies: linear, grid, heavy-hex

Qubit Mapping (Placement)

- Assign logical qubits to physical qubits
- Objective: minimize SWAP overhead
- NP-hard problem in general
- **SABRE** heuristic

Hardware-aware Scheduling

SWAP Gates

- When a 2-qubit gate acts on non-adjacent physical qubits, SWAPs move data
- Each SWAP = 3 CNOTs — expensive in depth and error
- Routing strategy must look ahead to minimize total SWAPs

Interaction with Scheduling

- Routing changes the DAG, new SWAP nodes and dependencies
- Joint optimization of mapping + scheduling yields better results
- Iterative approaches: schedule → route → re-schedule

Noise-Aware Scheduling

- Gate fidelities vary across the chip
- Prefer higher-fidelity qubits and links
- Calibration data informs placement decisions

Crosstalk-Aware Scheduling

- Concurrent gates on coupled qubits cause unwanted interactions
- Trade-off: parallelism vs. crosstalk error
- Serialization constraints added to schedule

Pulse-Level Scheduling

Beyond Discrete Time Steps

- Gate durations differ: single-qubit gates (~20 ns) vs. two-qubit (~200–600 ns)
- Pulse-level scheduling packs gates into continuous time, not discrete slots
- Exploits idle windows to start subsequent gates earlier

OpenPulse Framework

- Interface for specifying pulse-level control
- Calibration data provides actual gate durations and timing constraints
- Enables 10–20% additional depth reduction over gate-level scheduling

- Next lecture: Qubit control and readout using pulses

Scheduling and Resource Allocation

- Scheduling and resource allocation are two sides of the same coin
- Scheduling deals with “when”, while resource allocation deals with “what” and “who”
- Logical Resources
  - Qubit count, T-count, T-depth, CNOT count, Toffoli count, Measurement count
- Physical Resources
  - Physical qubit count, Wall-clock runtime, Code distance, Magic state factory count, Energy consumption

- Connecting the two topics requires deep knowledge in the hardware (superconducting qubits, spin qubits, neutral atoms, photonics, etc) -> beyond the scope of the course and program