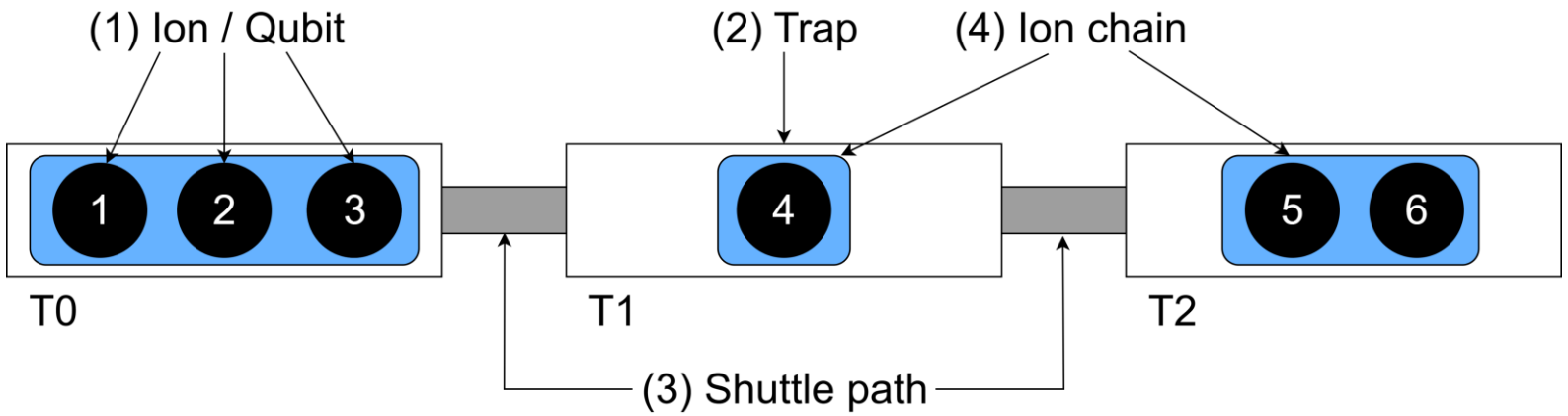
An Improved Ion-Shuttling Approach for QCCD Architectures

Tung-Yeh Wu and Ting-Chi Wang
National Tsing Hua University
Hsinchu, Taiwan

Introduction - QCCD Architecture

- Trapped-ion systems offer uniform qubit quality, long coherence times, and all-to-all connectivity within each trap.
- The Quantum Charge-Coupled Device (QCCD) architecture consists of multiple small ion chains confined in individual traps.
- This work focuses on linear QCCD topology.

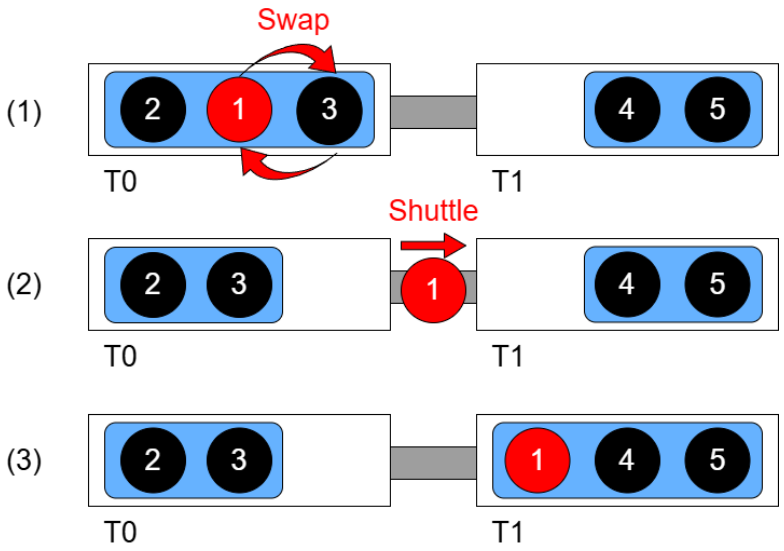


Introduction - Shuttle Operation

- In QCCD systems, a two-qubit gate can only be executed when both qubits are in the same trap.
- If not, shuttle operations are required to move ions to the same trap.
- To shuttle an ion to an adjacent trap, the following two conditions must be met:
  1. The ion must be located at the edge of its current trap on the side facing the destination trap; otherwise, a SWAP gate is needed.
  2. The destination trap must have available capacity to accommodate the ion.

Introduction -
Shuttle Operation

- Example: Shuttle ion 1 to trap T1

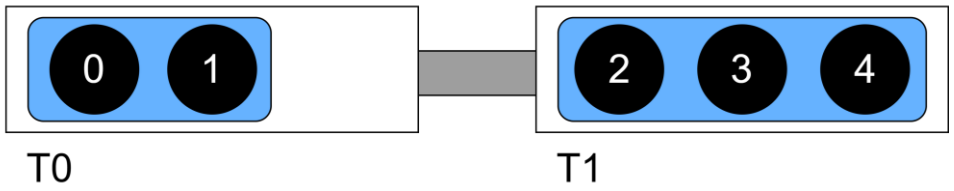


- Ion shuttling adds delay, raises motional energy, and degrades gate fidelity.
- Minimizing shuttle count is a primary optimization goal.

Introduction -
Problem Formulation

- Input: quantum assembly (QASM) file, hardware configuration (# of traps, trap capacity), and initial qubit placement
- Output: a valid sequence of operations for executing the quantum program
- Objective: minimizing the amount of required shuttle operations

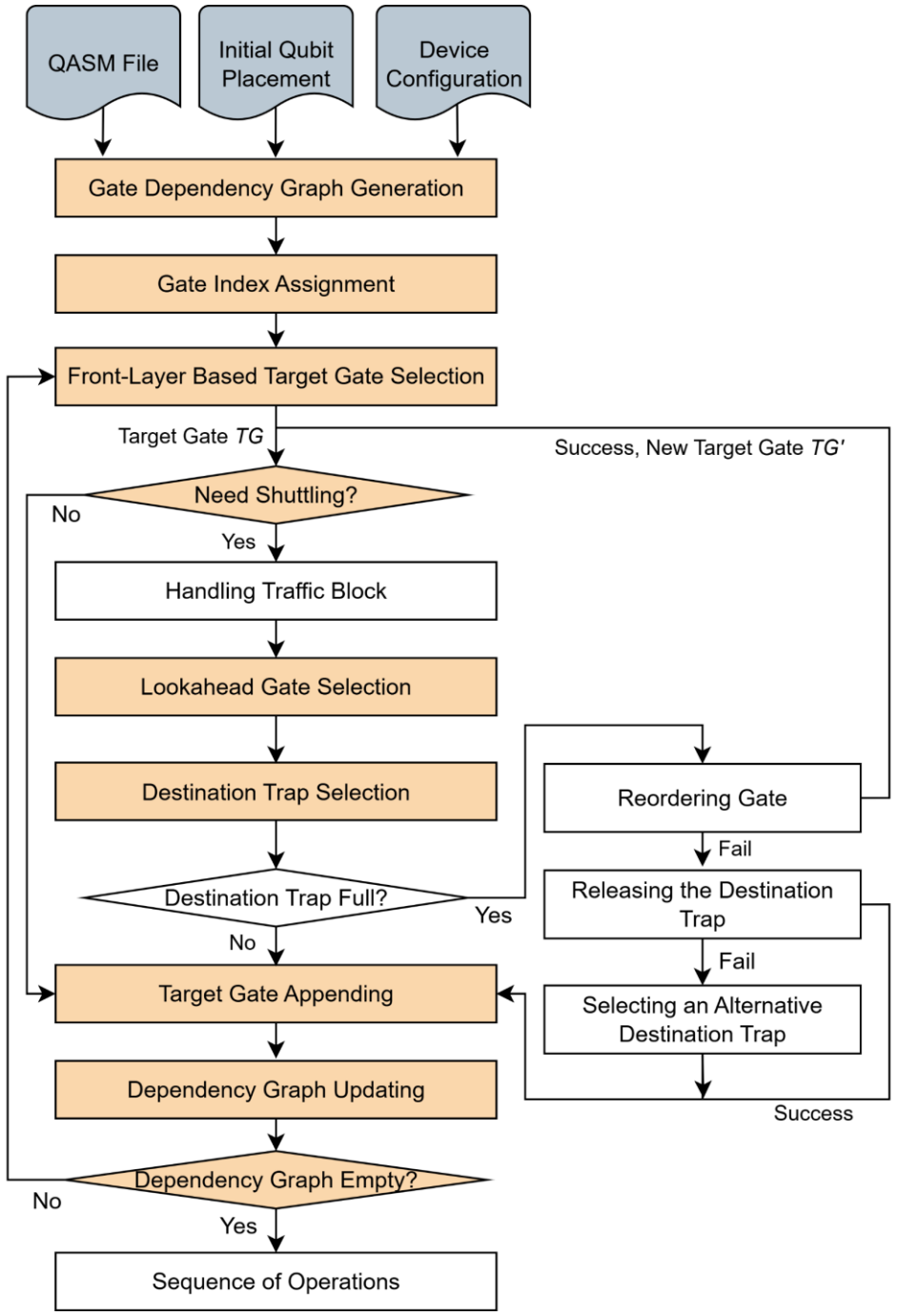
1. MS q[2], q[3]
2. MS q[3], q[4]
3. MS q[4], q[0]
4. MS q[1], q[4]



QASM # of traps = 2, trap capacity = 3, Initial placement = [[0, 1], [2, 3, 4]]

1. MS q[2], q[3]
2. MS q[3], q[4]
* SWAP q[2], q[4]
* Shuttle q[4] -> T0
3. MS q[4], q[0]
4. MS q[1], q[4]

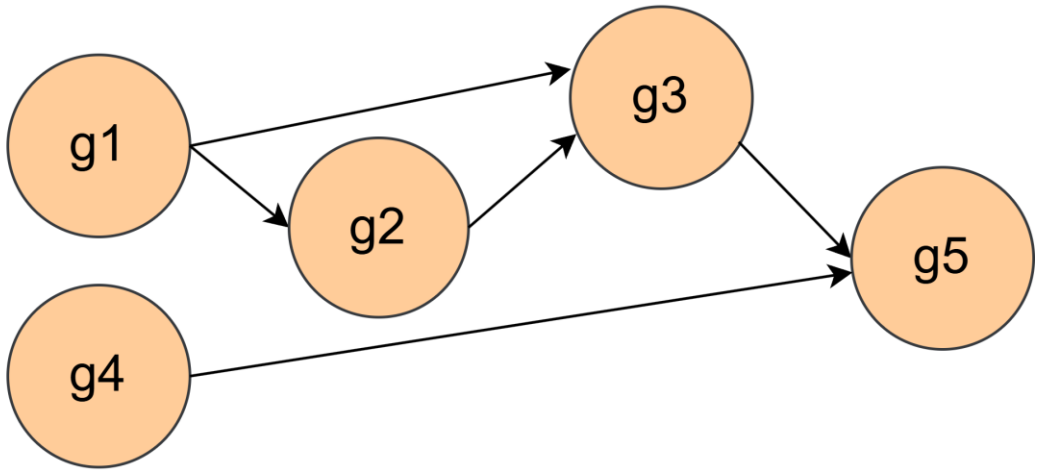
Our Approach – Overall Flow



Our Approach –
Gate Dependency Graph Generation

#	Gate
g1	MS q[0], q[1]
g2	MS q[1], q[2]
g3	MS q[0], q[2]
g4	MS q[3], q[4]
g5	MS q[2], q[3]

(a)



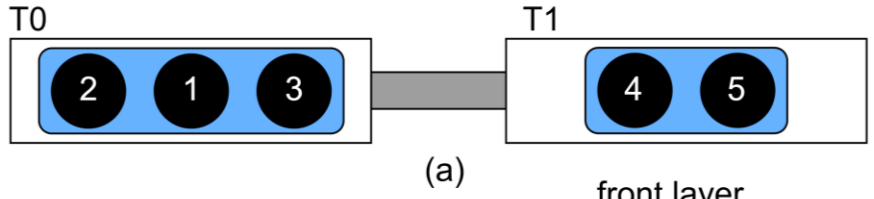
(b)

Our Approach –
Gate State & Front Layer

- At each iteration, a (remaining) gate is in one of the following three states:

1. Blocked: The gate has incoming edges in the gate dependency graph.
2. Executable: The gate has no incoming edges in the gate dependency graph, and its qubits reside in the same trap.
3. Shuttling Required: The gate has no incoming edges in the gate dependency graph, but its corresponding qubits reside in different traps.

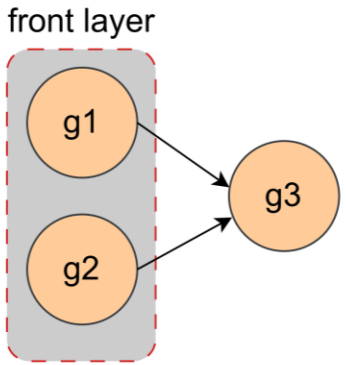
- Front Layer: the set of gates that are either Executable or Shuttling Required



(a)

#	Gate	Gate status
g1	MS q[1], q[2]	*Executable*
g2	MS q[3], q[4]	*Shuttling Required*
g3	MS q[2], q[3]	*Blocked*

(b)



(c)

Our Approach –
Front Layer-Based Target Gate Selection

- At each iteration, there are two possible cases for the front layer:

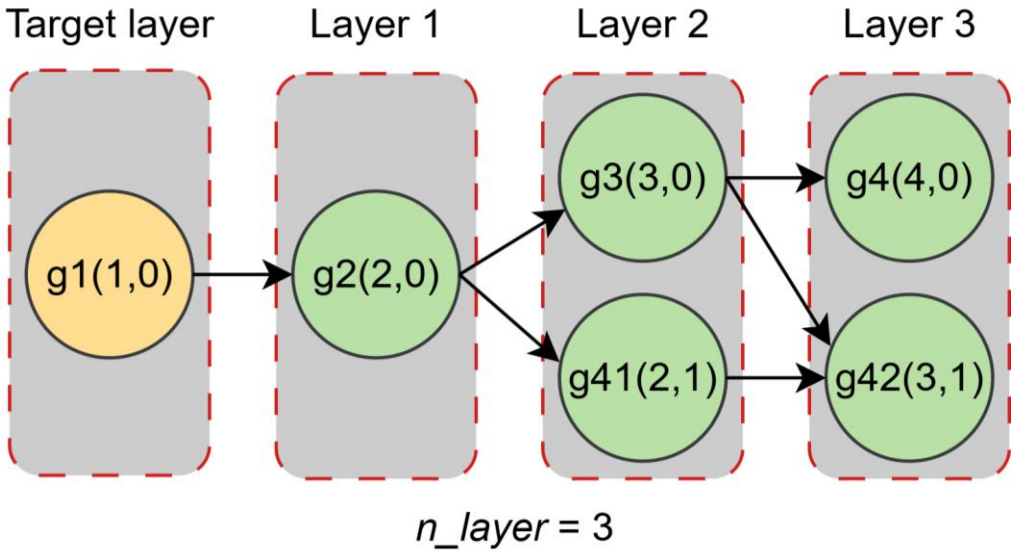
1. One or more *Executable* gates are in the front layer
   - ➢ One of them is selected as the target gate (and append to the output sequence).
2. All the gates in the front layer are *Shuttling Required*
   - ➢ The gate with the smallest index (from a topological order) is selected as the target gate.

Our Approach –
Handling Traffic Block

- When the target gate is Shuttling Required, traffic block may exist and need resolved.
- Two possible cases for traffic block (assuming ion 0 and ion 4 are qubits involved in the target gate)

Our Approach –
Lookahead Gate Selection

- To determine a better destination trap, in the gate dependency graph, all reachable gates from the target gate within a user-defined depth *n_layer* are identified.
- Among these gates, those that have any common qubit with the target gate are selected and included in the look-ahead gate set.



Our Approach –
Destination Trap Selection

- Assume the target gate has qubit *A* in trap T_A and qubit *B* in trap T_B.
- For each trap *i*, we define $G_{A,i}$ as the number of gates in the look-ahead gate set that involve ion A and another ion in trap *i*. ($G_{B,i}$ is defined similarly)
- A score for each trap *i* is calculated below, which estimates the amount of look-ahead gates that can be executed in trap *i* after moving irons A and B to trap *i*.

$$\text{TrapScore}_i = G_{A,i} + G_{B,i}, i \in \{0, 1, \ldots, \#\text{ of traps} - 1\}.$$

- The trap with the highest score is selected as the target trap.

Our Approach –
Destination Trap Selection

- If the target trap is located between traps T_A and T_B and its available capacity is at least two., it is selected as the destination trap.
- Otherwise, one of T_A and T_B is selected as the destination trap.
  - The scores for trap T_A and T_B are modified as follows:

$$TrapScore'_{T_A} = \frac{TrapScore_{T_A}}{\#Gates\ remaining\ \ for\ ion\ B} \qquad TrapScore'_{T_B} = \frac{TrapScore_{T_B}}{\#Gates\ remaining\ \ for\ ion\ A}$$

  - The trap with the higher score is the destination trap.
- If the destination trap has no available capacity, we will apply existing methods to resolve it.

Experimental Results –
Experiment Setup

- Our approach was Implemented in Python.
- Test cases: 12 actual circuits and 121 synthetic circuits (QV + 120 random circuits)
- Hardware model: a linear QCCD with 6 traps, where each trap can initially accommodate up to 15 ions and has 2 additional ion slots reserved for shuttling

Benchmark	Qubits	2-qubit Gates	Communication Pattern
QSE_0	54	9720	*Short- and long-range gates*
QSE_1	54	9720	*Short- and long-range gates*
CA	60	465	*Short- and long-range gates*
DA	60	1335	*All distances*
QV	60	5400	*Random*
Adder	64	455	*Short range gates*
Supermacy	64	560	*Nearest neighbor gates*
QAOA	64	1260	*Nearest neighbor gates*
QuadraticForm	64	3400	*All distances*
QFT	64	4032	*All distances*
QuGAN	71	552	*Short- and long-range gates*
Multiplier	75	6510	*All distances*
SquareRoot	78	1028	*Short- and long-range gates*
Random Circuits	60, 65, 70, 75	1438	*Random*

Experimental Results –
Results

Benchmark	[Saki et al., DATE 2022]		Proposed Approach		Improvement Rate	
	Shuttle Count	Compile Time (s)	Shuttle Count	Compile Time (s)	Shuttle Count	Compile Time
QSE_0	4508	1434.12	**2956**	**761.33**	34.43%	46.91%
QSE_1	4548	1563.09	**2993**	**635.26**	34.19%	59.36%
CA	97	2.04	**84**	**1.74**	13.4%	14.71%
DA	300	12.76	**187**	**3.26**	**37.67%**	74.45%
QV	2612	**220.93**	**2042**	255.29	21.82%	-15.55%
Adder	28	1.77	**22**	**1.66**	21.43%	6.21%
Supermacy	261	2.75	**196**	**2.22**	24.9%	19.3%
QAOA	903	12.80	**823**	**11.86**	8.9%	7.3%
QuadraticForm	**170**	27.54	171	**11.36**	-0.6%	58.7%
QFT	198	17.04	**196**	**11.24**	1.0%	34.0%
QuGAN	24	1.73	**22**	**1.64**	8.33%	5.2%
Multiplier	1742	419.78	**1455**	**105.33**	16.48%	**74.91%**
SquareRoot	331	6.05	**309**	**4.39**	6.6%	27.4%
Random Circuits	775	17.23	**762.57**	**14.1**	1.6%	17.9%
Arithmetic Mean	-	-	-	-	16.44%	30.77%

Conclusion

- We present an ion-shuttling approach for linear QCCDs featuring novel target gate selection and destination trap selection.
- The experimental results demonstrate the efficacy of our approach, in terms of shuttle count and compilation time.