Architecting Scalable Trapped-Ion Quantum Computers using Surface Codes

Scott Jones *Supervisor:* Dr. Prakash Murali *Institution:* University of Cambridge

Project Structure

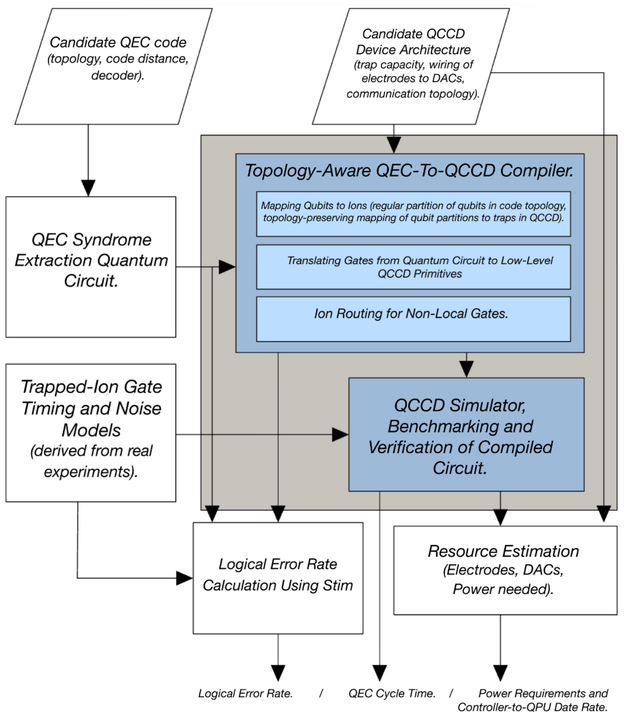


Figure 1. Framework for evaluating the suitability of a candidate QCCD-based TI system for QEC. Taking a candidate architecture and a candidate QEC code as input, the tool flow computes error correction metrics such as logical error rate, QEC cycle time and power dissipation requirements by using a QEC-to-QCCD compiler, QCCD simulator, and realistic models for performance and resource estimation.

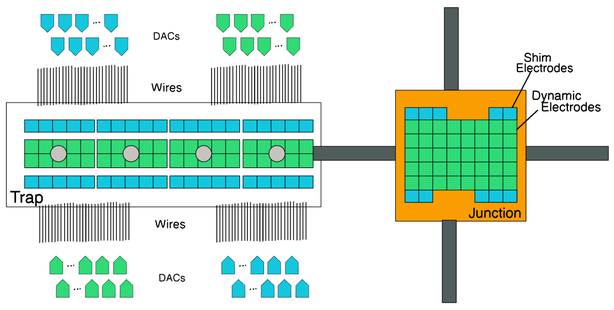
Key Contributions

Scalable QEC Compiler: We introduce a surface code and QCCD device topology aware compiler that offers near-optimal cycles times and outperforms existing compilers by 3.8x. To our knowledge, it is the only compiler that can handle realistic code distances and scale up to large trap capacities.

Matching QEC and device architecture: When the surface code topology is matched well with an appropriate device architecture (by choosing the trap capacity and topology), we get optimal cycle times and hardware requirements for a desired logical error rate. Unlike prior NISQ studies, which showed the benefits of higher trap capacity, surface codes benefit from having only a trap capacity of two.

Power-cycle time tradeoff: Comparing proposals for scalable wiring for QCCD architectures, we show a tradeoff between the surface code cycle time and power requirements. While the standard QCCD wiring architecture offers low cycle times $\approx 4ms$, it requires high power (up to ≈ 630 watts for a single logical qubit at a distance of 14). On the other hand, WISE reduces wiring and bandwidth needs but has an infeasible increase in cycle time by 17.5x for a 10^{-9} logical error rate. This points to the need to not only co-design QEC choices with trap architecture, but also consider control system aspects such as wiring and power requirements.

Trapped-Ion Quantum Charged Coupled Device



Surface Codes

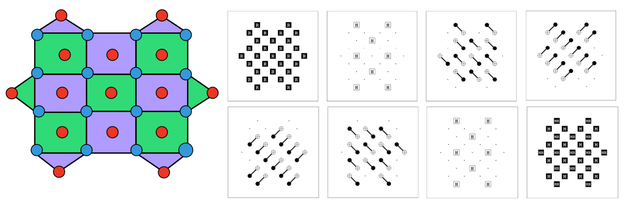


Figure 2. The topology of the distance four rotated surface code

QCCD Logical Qubit Design Trade-offs

- Trap Capacity
- Communication Topology Number of Junctions per Trap Number of Zones per Junction Number of DACs per Electrode Physical Ion Swap/Gate Swap
- Cooling Strategy
- Gate Times and Fidelities
- Ion Reconfiguration Toolbox
- Ion Choice

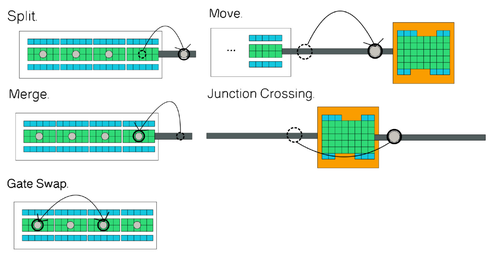


Figure 3. QCCD reconfiguration primitives for ion transport and rearrangement. Ions (grey circles) can be split, merged, moved, or guided across junctions (orange) using dynamic electrodes (green). Although the *gate swap* is not a hardware-level primitive, it consists of three MS gates that effectively reposition ions in a linear chain to facilitate splitting or movement to another trap. Collectively, these operations provide all-to-all connectivity in a QCCD system.

Wiring Topology

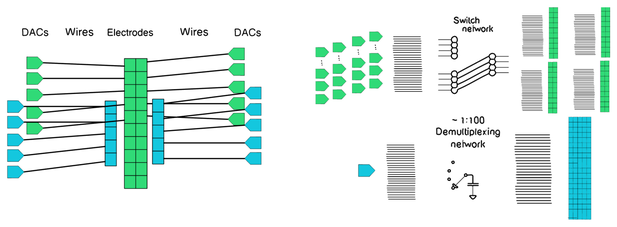


Figure 4. Comparison of standard and WISE wiring approaches for a QCCD system.

Choice of Communication Topology

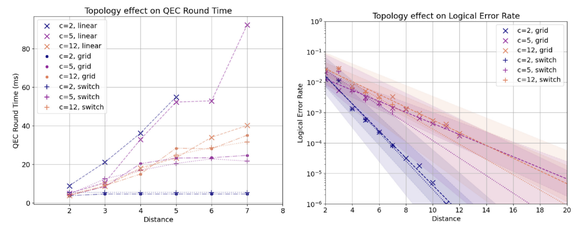


Figure 5. (a) Elapsed time per QEC round (y-axis) as a function of code distance (x-axis) for trap capacities 2, 5, and 12, under linear, grid, and all-to-all switch communication topologies. (b) Logical Error Rate as a function of code distance for trap capacities 2, 5, and 12 under the grid and all-to-all switch.

Our work validates that across trap capacities, the grid topology very closely matches the all-to-all switch both in terms of QEC round time and logical error rate, making it an ideal choice for hardware implementation. In the following experiments, we use the grid topology.

S

Topology-Aware QEC-to-QCCD Compiler

- Determine Qubit Clusters by Top-Down Regular Partitioning
- Map Clusters to Qubits by Minimum-Weight Perfect Matching
- Convert IR to Primitive Operations
- Determine Ion Routing
- Optimise Scheduling

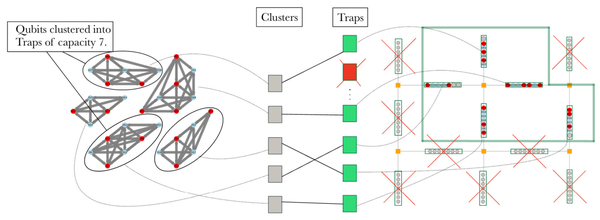


Figure 6. Mapping clusters of qubits to traps via a minimum weight perfect matching problem

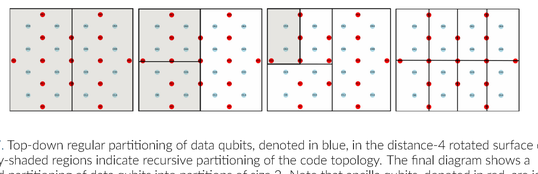


Figure 7. Top-down regular partitioning of data qubits, denoted in blue, in the distance-4 rotated surface code. The grey-shaded regions indicate recursive partitioning of the code topology. The final diagram shows a balanced partitioning of data qubits into partitions of size 2. Note that ancilla qubits, denoted in red, are ignored in this phase but grouped into partitions afterwards using a greedy-based strategy.

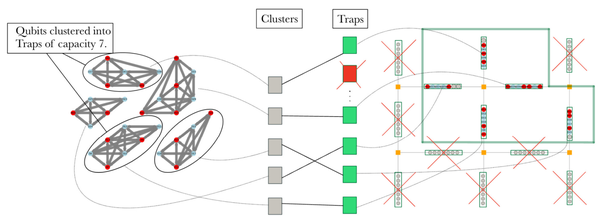
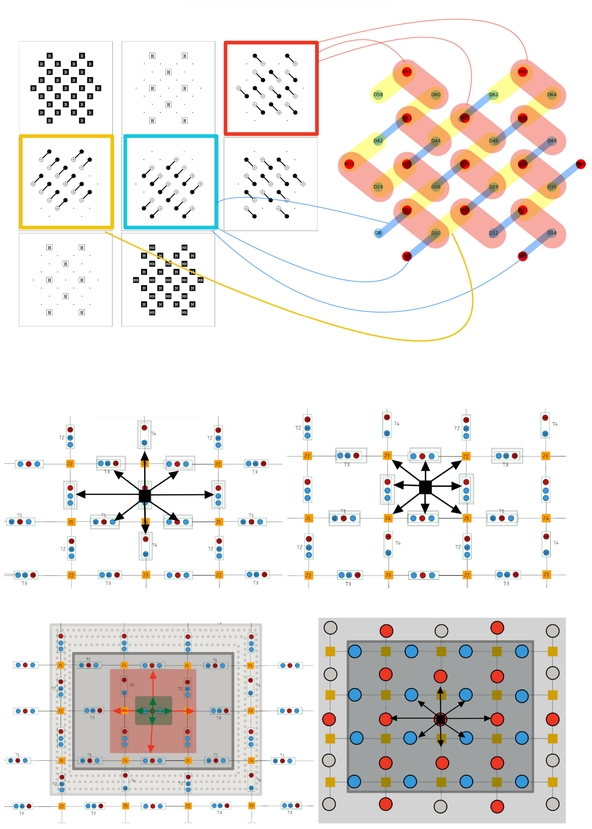


Figure 8. Translation of the problem of mapping clusters of qubits to traps into a maximum perfect matching problem. The figure shows clusters of qubits, each of which needs to be mapped to a trap in a pre-determined selection of traps with equal cardinality to the number of qubit clusters.



Ion Routing Constraints

The ion routing algorithm computes a path for each ancilla qubit to reach its corresponding data qubit's trap while satisfying QCCD hardware constraints:

- **Trap capacity:** Each trap has a maximum ion count per trap which must not be exceeded during ion routing.
- **Junction exclusivity:** Only one ion may occupy a junction at any time.
- **Segment exclusivity:** Only one ion is allowed per segment connecting traps or junctions.

Ion Routing Algorithm

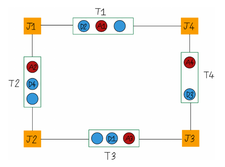


Figure 9. Setup of the ion routing problem: 4 traps (T1 to T4, each with capacity 4), 4 junctions (J1 to J4 as orange squares), and connecting segments.

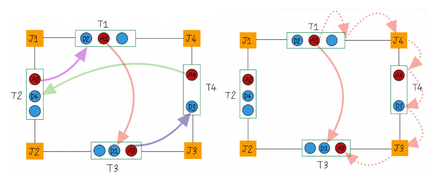


Figure 10. (Left) Routing destinations for ancilla qubits marked for their next operation. (Right) The routing algorithm finds a path for ancilla A1 to destination D1, where a dotted red arrow denotes each edge. In the path, an edge corresponds to a sequence of ion reconfiguration primitives. For example, the first edge in the path, marked with red dotted arrows from A1 to D1, represents a high-level SWAP operation within trap T1 to move A1 to the trap's edge (requiring 3 MS gates), followed by a split operation to move A1 into segment T1 to J4.

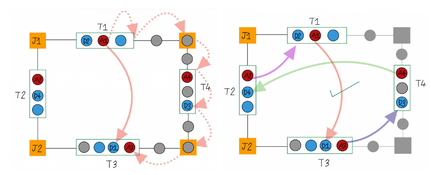


Figure 11. (Left) Allocation of one extra qubit, denoted by grey circles, to each junction, segment, and trap in the routing path from A1 to D1, excluding the source trap. (Right) Grey shading indicates components removed from the QCCD graph due to capacity constraints or disconnection.

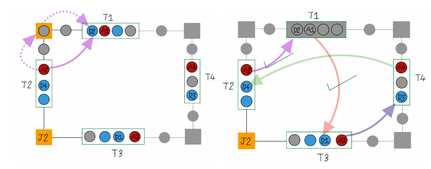


Figure 12. (Left) Routing path for A2 to D2 with grey circles indicating allocated extra qubits along the path. (Right) Components removed from the graph due to capacity constraints, including T1.

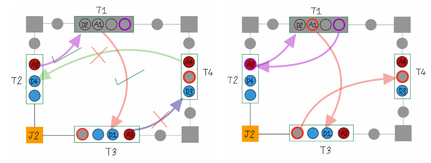


Figure 13. (Left) T3 violates the invariant by exceeding its capacity of 4 ions. (Right) A1 is re-routed back to T4 along its original path to restore the invariant, while D2 is routed back to T2.

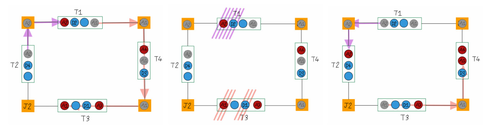


Figure 14. Routing ancillas to their data qubits for entanglement, followed by the entanglement operation, and finally routing ancillas back to traps while preserving invariants at the end of the pass.

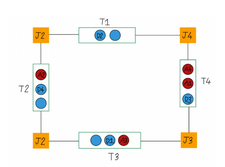


Figure 15. Final state of the first pass showing both invariants preserved.

Testing and Benchmarking

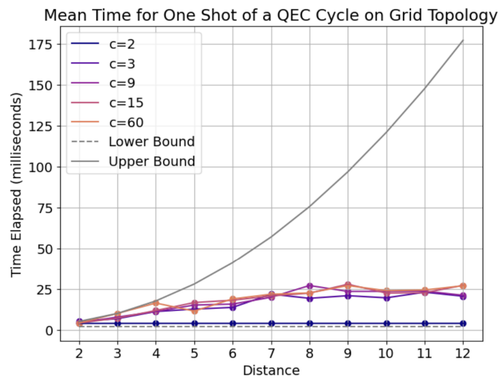
QEC Code	QCCD Topology	Theoretical Minimum Elapsed Time (μs)	Measured Elapsed Time (μs)	Number of Routing Operations (Theoretic / Measured)
Repetition Code Distance = 3	Linear Trap Capacity = 2	1535	1535	18 / 24
	Linear Trap Capacity = 3	1270	1390	6 / 10
	Linear Trap Capacity = 4	1385	1505	6 / 7
	Single Ion-Chain	2190	2190	0 / 0
Repetition Code Distance = 6	Linear Trap Capacity = 2	1535	1535	60 / 60
	Linear Trap Capacity = 3	2060	2300	27 / 29
	Linear Trap Capacity = 4	2425	2785	18 / 21
	Single Ion-Chain	5400	5400	0 / 0
2D Rotated Surface Code Distance = 2	Grid Trap Capacity = 2	4055	4055	48 / 48
	Linear Two Ion Chains	3225	3305	9 / 10
2D Unrotated Surface Code Distance = 2	Grid Trap Capacity = 3	4085	4325	56 / 60
2D Rotated Surface Code Distance = 3	Grid Trap Capacity = 2	4085	4085	288 / 288
	Linear Two Ion Chains	8605	8605	19 / 19
	Switch Trap Capacity = 2	5325	5325	432 / 432
2D Rotated Surface Code Distance = 6	Grid Trap Capacity = 2	4085	4085	1440 / 1440
2D Rotated Surface Code Distance = 12	Grid Trap Capacity = 2	4085	4085	6336 / 6336

Figure 16. Comparison of our QEC compiler against theoretically optimal compilation. Our compiler is near-optimal in terms of elapsed time and number of routing operations.

	Movement Time For 5 Rounds			Number of Movement Operations		
	QEC Com.	QCCD Sim	Muzzle Shuttle	QEC Com.	QCCD Sim	Muzzle Shuttle
R,3,2,L	3300	8851	6365	40	219	173
R,5,2,L	3300	12521	31893	80	436	880
R,7,2,L	3300	20054	64194	120	713	1715
R,3,3,L	3135	3160	1666	58	71	35
R,5,3,L	3960	4178	4178	127	163	164
R7,3,L	4945	4178	4178	199	217	218
R,3,5,L	0	0	0	0	0	0
R5,5,L	1650	1663	1663	31	31	31
R,7,5,L	3300	1663	2323	61	58	58
S,2,2,G	10800	19083	NaN	240	327	NaN
S,3,2,G	13500	94738	NaN	720	2102	NaN
S,4,2,G	13500	NaN	NaN	1440	NaN	NaN
S,5,2,G	13500	NaN	NaN	2400	NaN	NaN
S,2,3,G	15980	9881	NaN	241	192	NaN
S,3,3,G	19410	59110	NaN	627	240	NaN
S,4,3,G	29610	NaN	NaN	1378	NaN	NaN
S,5,3,G	47920	NaN	NaN	2465	NaN	NaN
S,2,5,G	10260	5054	5076	116	57	67
S,3,5,G	22560	24777	122996	461	472	1740
S,4,5,G	30300	NaN	NaN	868	NaN	NaN
S,5,5,G	40460	NaN	NaN	1740	NaN	NaN

Figure 17. Benchmark test of the QEC compiler outlined with other QCCD compilers, namely QCCDSim and MuzzleTheShuttle. Each test determines the movement time and number of movement operations in the compiled schedules for a particular software-hardware configuration. A 4-tuple specifies each configuration: QEC code (R = repetition code, S = 2D Rotated Surface Code), Code Distance, Trap Capacity, and QCCD Communication Topology (L = linear, G = grid). In some cases, a QCCD constraint was violated, or the compilation failed, in which cases' NaN' is reported.

Choice of Trap Capacity



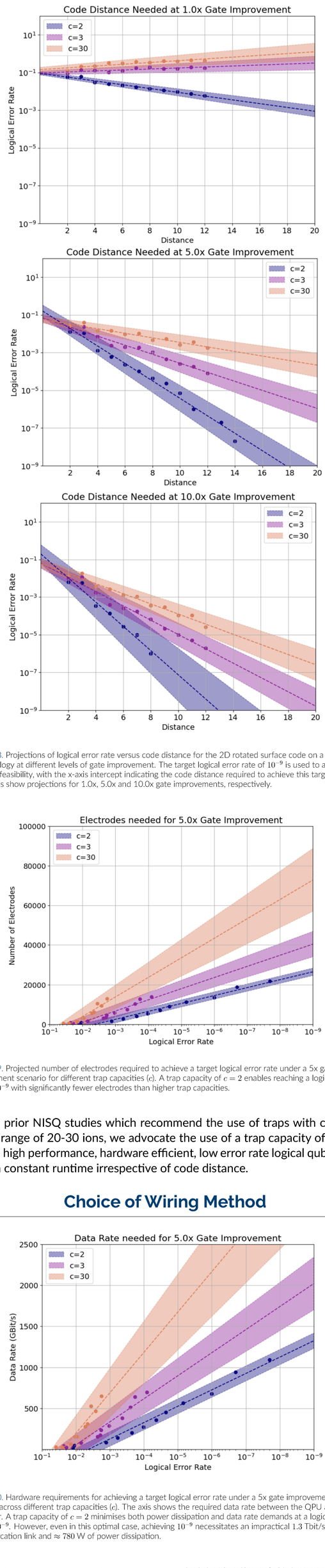


Figure 18. Projections of logical error rate versus code distance for the 2D rotated surface code on a QCCD grid topology at different levels of gate improvement. The target logical error rate of 10^{-9} is used to assess practical feasibility, with the x-axis intercept indicating the code distance required to achieve this target. The three axes show projections for 1.0x, 5.0x and 10.0x gate improvements, respectively.

Figure 19. Projected number of electrodes required to achieve a target logical error rate under a 5x gate improvement scenario for different trap capacities (c). A trap capacity of $c = 2$ enables reaching a logical error rate of 10^{-9} with significantly fewer electrodes than higher trap capacities.

Unlike prior NISQ studies which recommend the use of traps with capacity in the range of 20-30 ions, we advocate the use of a trap capacity of two to obtain high performance, hardware efficient, low error rate logical qubits that have a constant runtime irrespective of code distance.

Choice of Wiring Method

Figure 20. Hardware requirements for achieving a target logical error rate under a 5x gate improvement scenario across different trap capacities (c). The axis shows the required data rate between the QPU and the controller. A trap capacity of $c = 2$ minimises both power dissipation and data rate demands at a logical error rate of 10^{-9}. However, even in this optimal case, achieving 10^{-9} necessitates an impractical 1.3 Tbit/s communication link and ≈ 780 W of power dissipation.

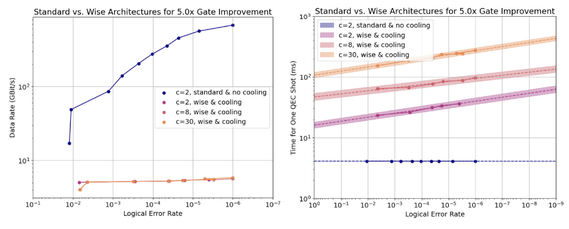


Figure 21. **(a)** Data rate comparison between the standard architecture without cooling and WISE architecture with cooling, under a 5× gate improvement. Cooling improves data rate scaling across all trap capacities for the WISE architecture, allowing low logical error rates at modest data rate requirements compared to standard capacity-2 systems. **(b)** Elapsed QEC shot time versus target logical error rate under a 5× gate improvement. In the WISE architecture with cooling, cycle times scale quadratically with code distance, leading to a logical clock speed of 0.1 operations per second for a 10^{-9} target error rate. In contrast, the standard, no cooling, trap capacity two architecture exhibits linear scaling of cycle times with increasing code distance.