Neutral-atom quantum computing
Gates and moves - tutorial

Pedro Lopes

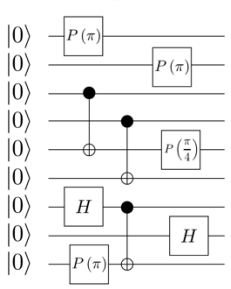
Casey Duckering

Yelissa Lopez

QuEra Computing Inc.

YQuantum 2025

Main theme



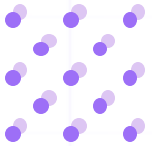
Algorithmic pipeline

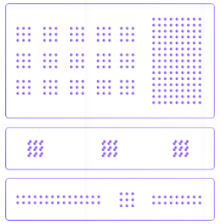
Error-correcting code choice
Compilation
Gate design
Protocol layout + spacetime optimization

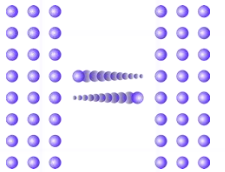
Co-Design

Native hardware capabilities

Speed
Qubit connectivity
Parallelization
Universal gate-set
Biased noise & erasure



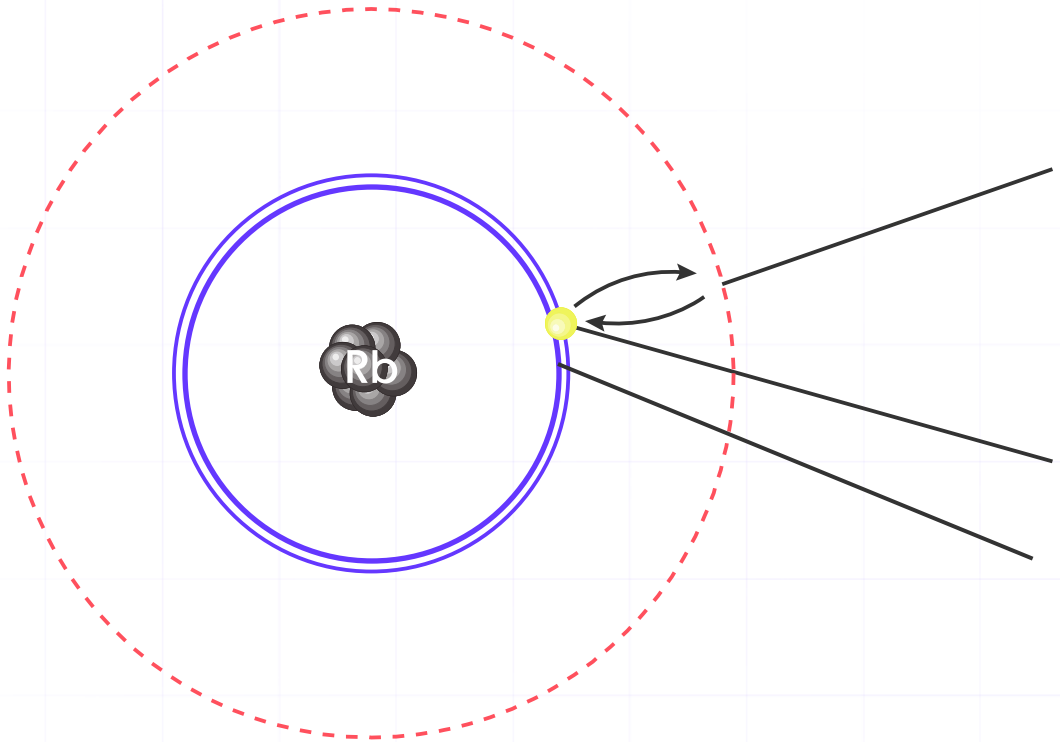




How can we leverage neutral atoms' strengths to design efficient algorithms?

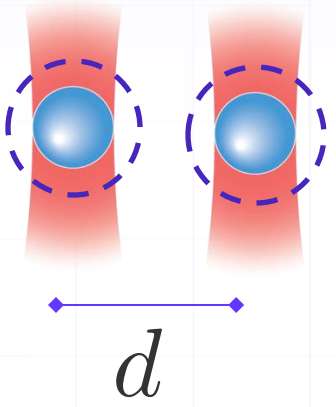
Digital: Entanglement mediated by puffing-up atoms

Strong interactions
Controlled interactions



$|r\rangle$

$V \sim d^{-6}$



$|1\rangle$

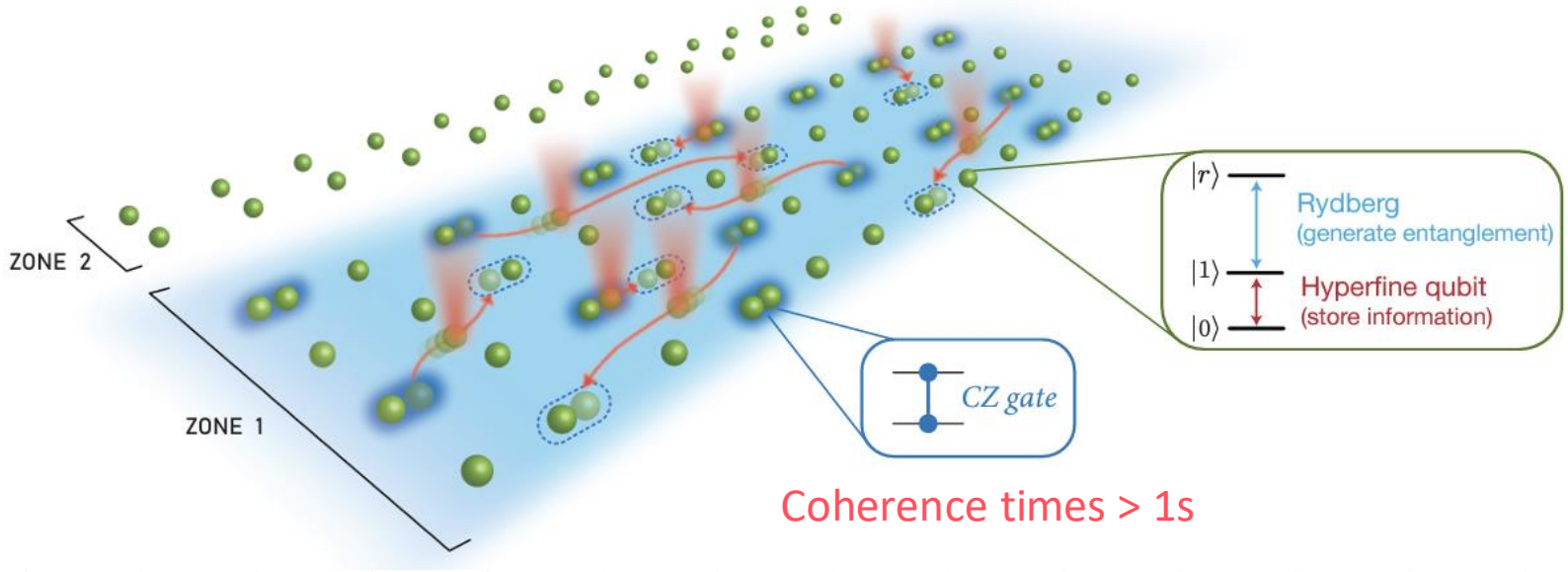
$|0\rangle$

Hyperfine split

Qubits by isolated levels choice

Long coherence times
High stability, flexibility

Basic architecture: mid-circuit reconfigurability

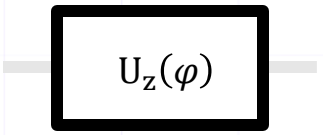


Coherence times > 1s

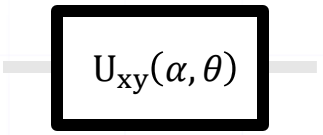
Bluvstein et al., Nature (2022)

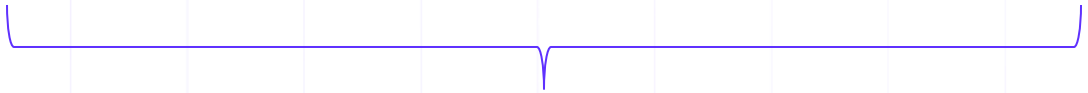
Native gate set (for our purposes)

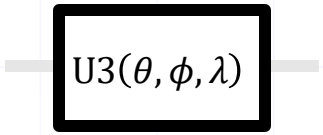
Arbitrary 1 qubit Z rotations



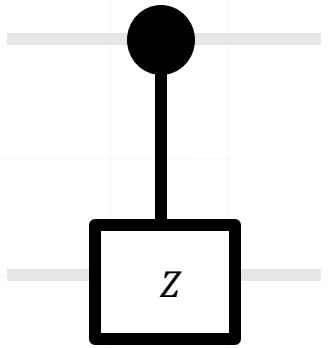
Arbitrary 1 qubit rotations
With axis in the XY plane





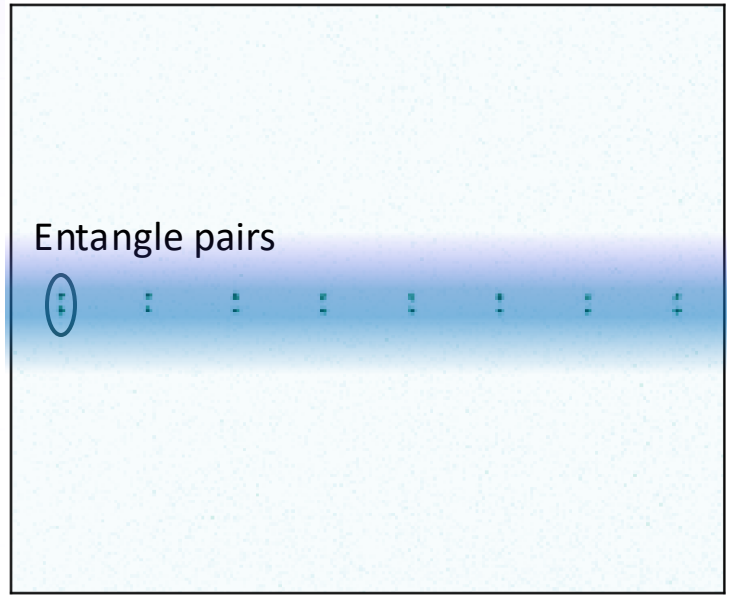


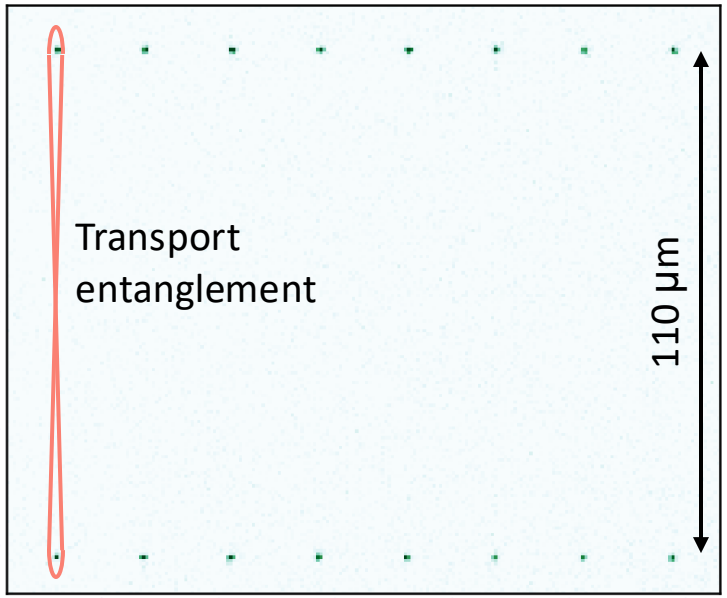
Controlled-Z gates

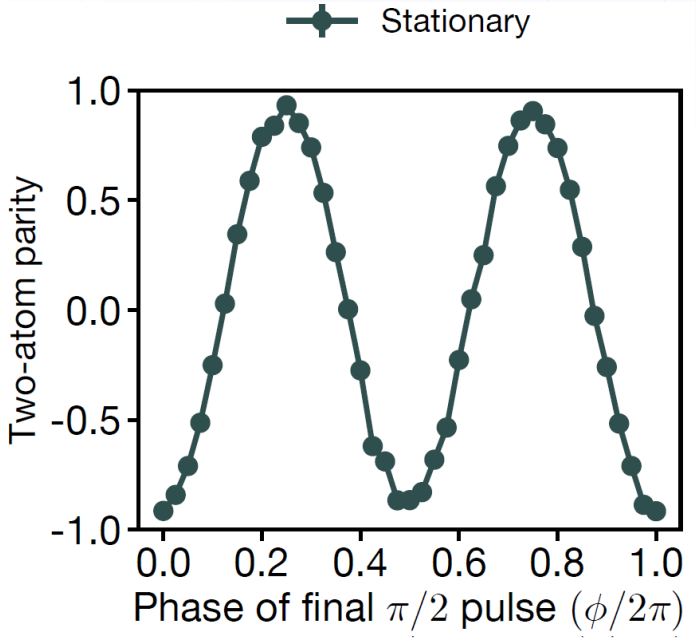


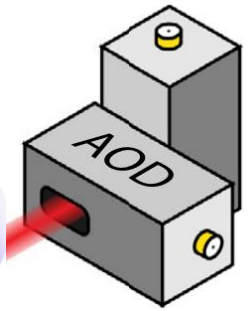
Entanglement transport

<300 µs to move across entire array (T_2 ~ 1.5 s)









Atom-atom spacing of ~3 µm

→ transport across array of ~2000 qubits in a time of < 10^{-3} T_2

Bluvstein et al., Nature 2022

Beugnon et al *Nat Phys* 2007; T. Đorđević et al *Science* 2021

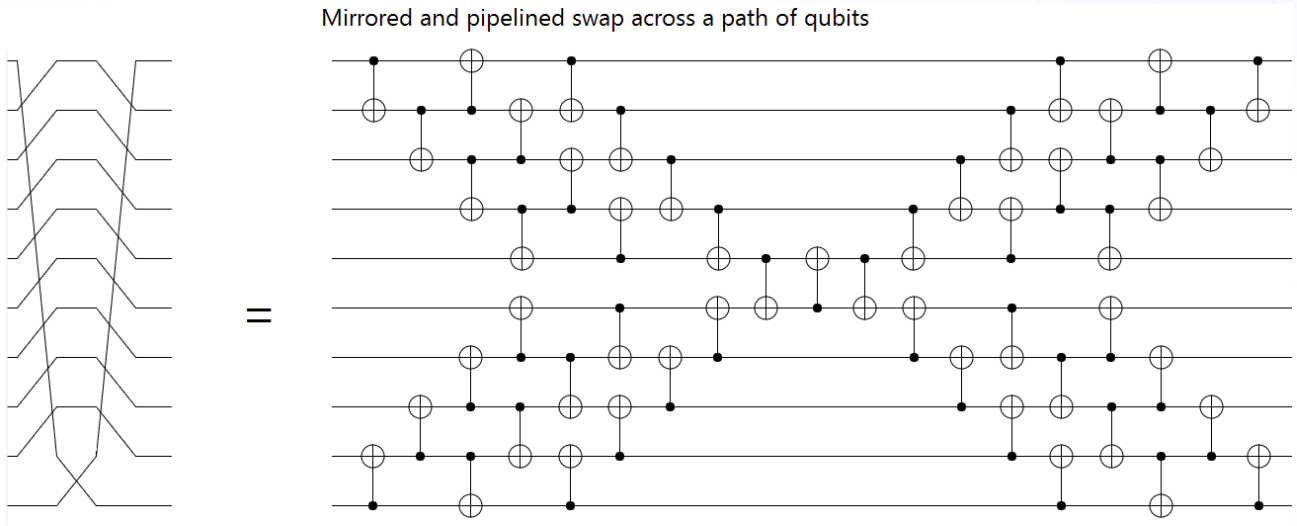
IQuEra

Atom shuttling rules!

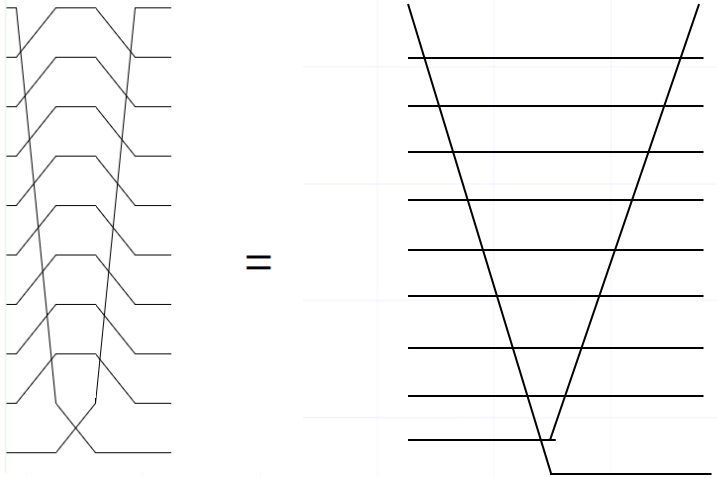
Long-range/arbitrary connectivity

Source: Craig Gidney's blog

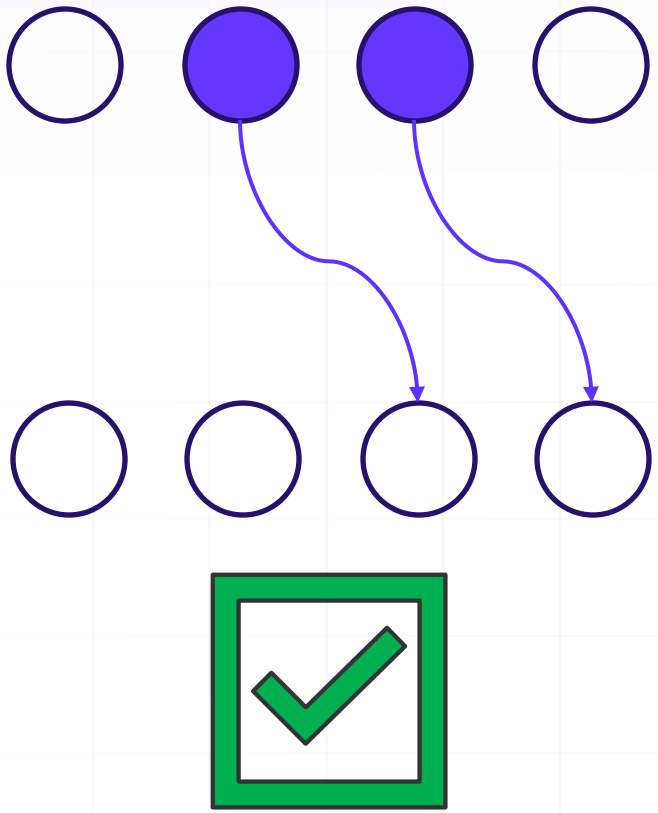
Nearest-neighbor connectivity

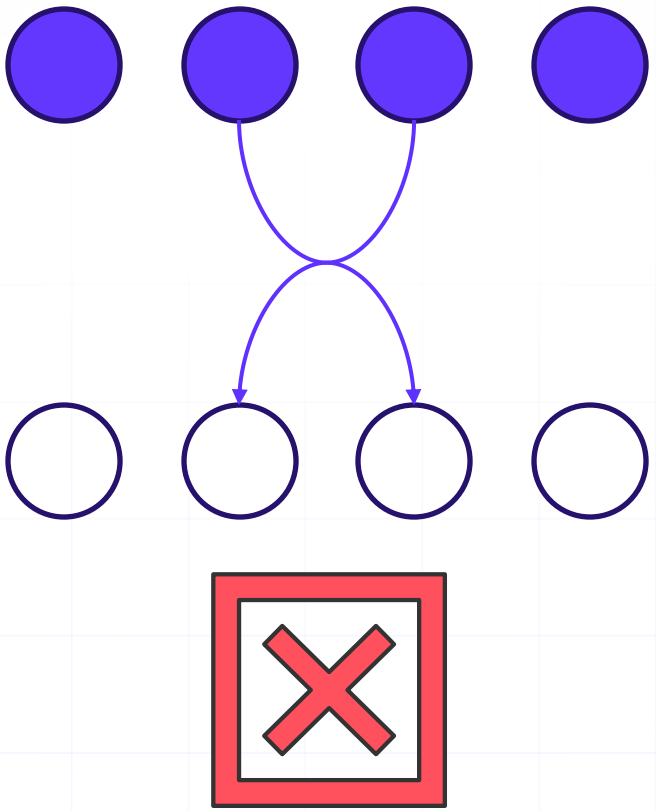


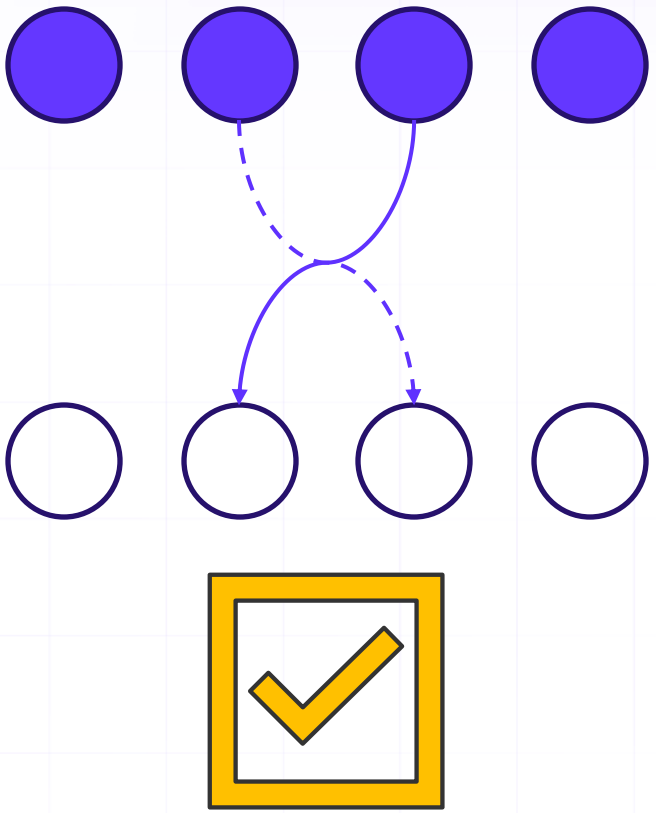
Reconfigurable connectivity



Atom shuttling rules?

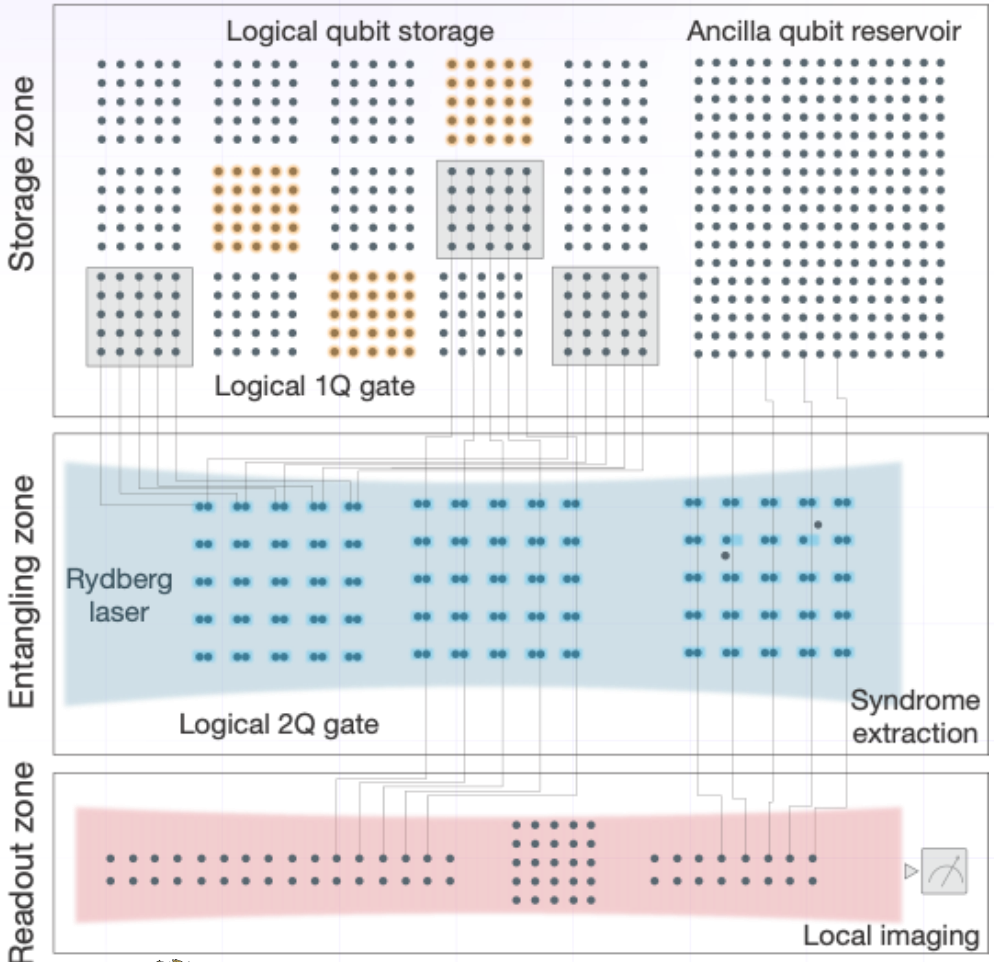






“atoms cannot collide”
“atoms cannot change order in a single move”

Sandbox Model for Current Gen. Quantum Computer

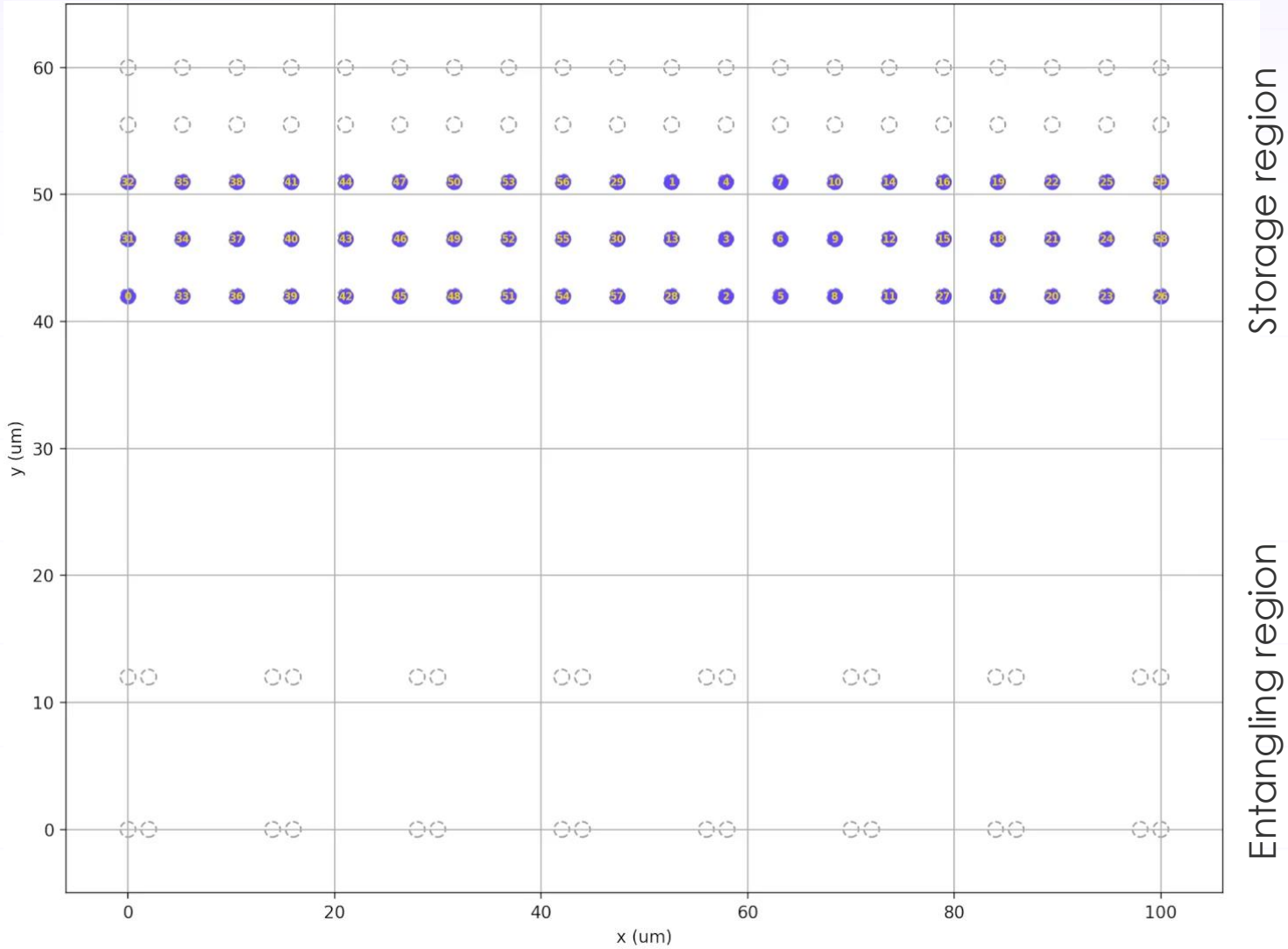


Bluvstein et al., Nature 2024

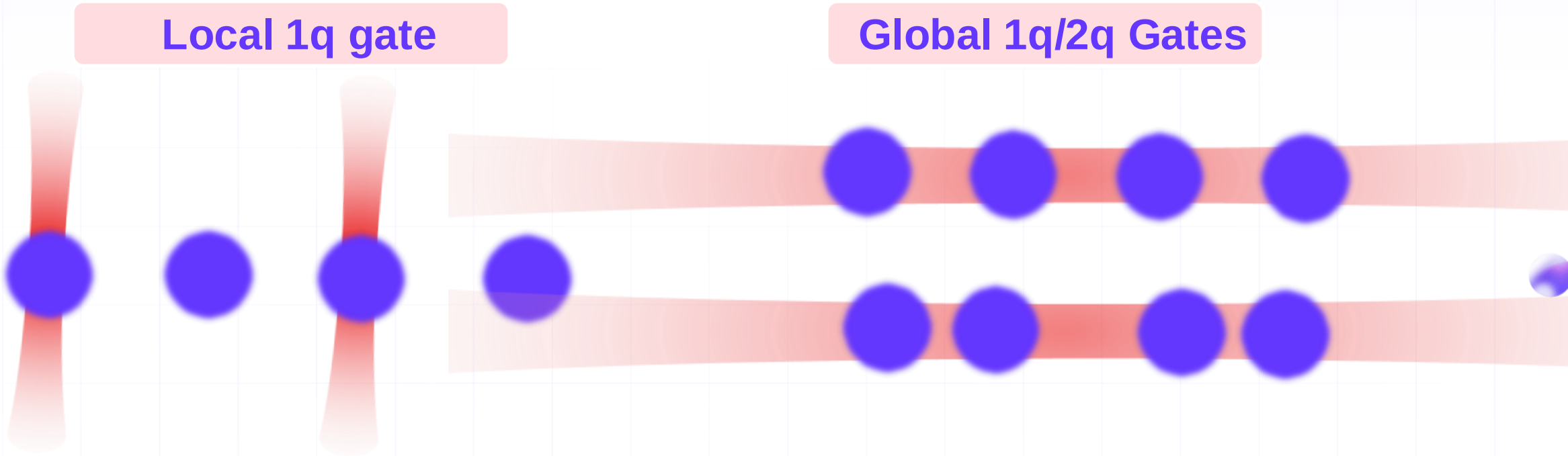
Keep in mind: the technology is still rapidly developing, and **tomorrow's systems** may look **very different**!

- Hundreds to a thousand qubits
- High-fidelity parallel gate operation, with long coherence times
- Parallel movement of qubits on a grid
- Mid-circuit measurement and feedforward
- Some analogies to classical RAMs

Sandbox Model for Current Gen. Quantum Computer

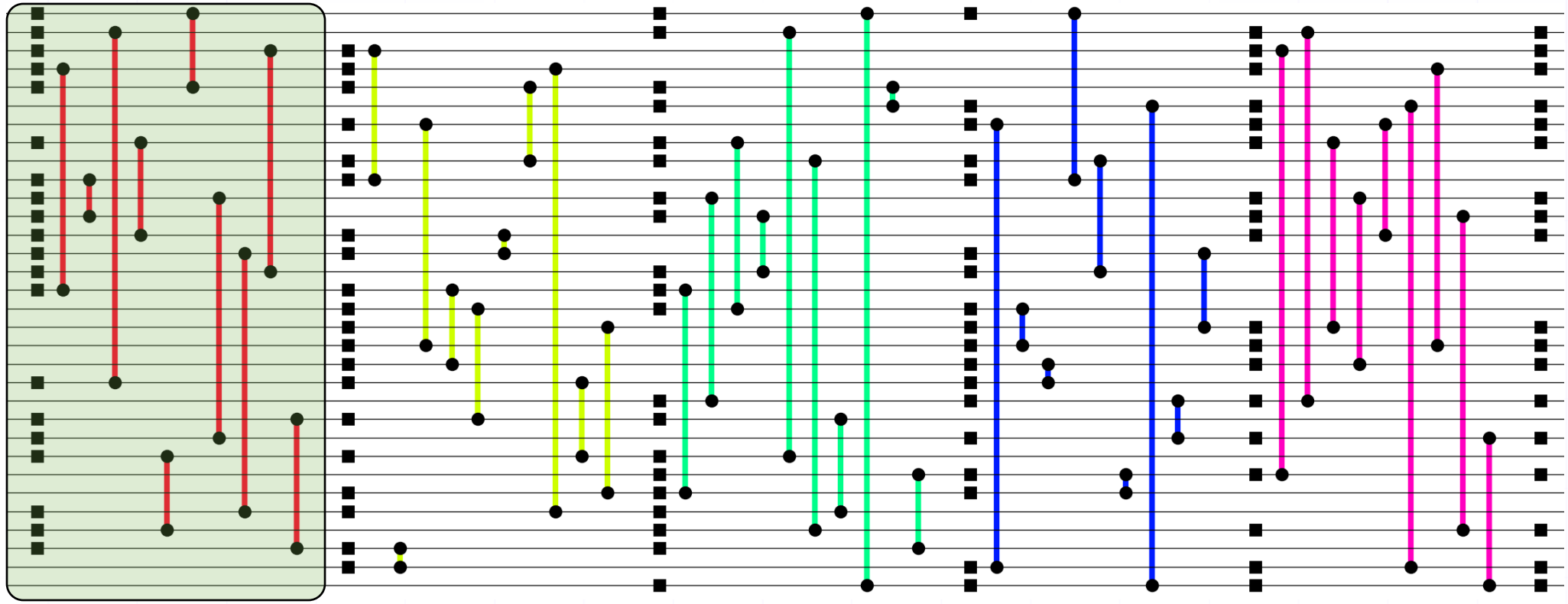


Local gates vs global gates



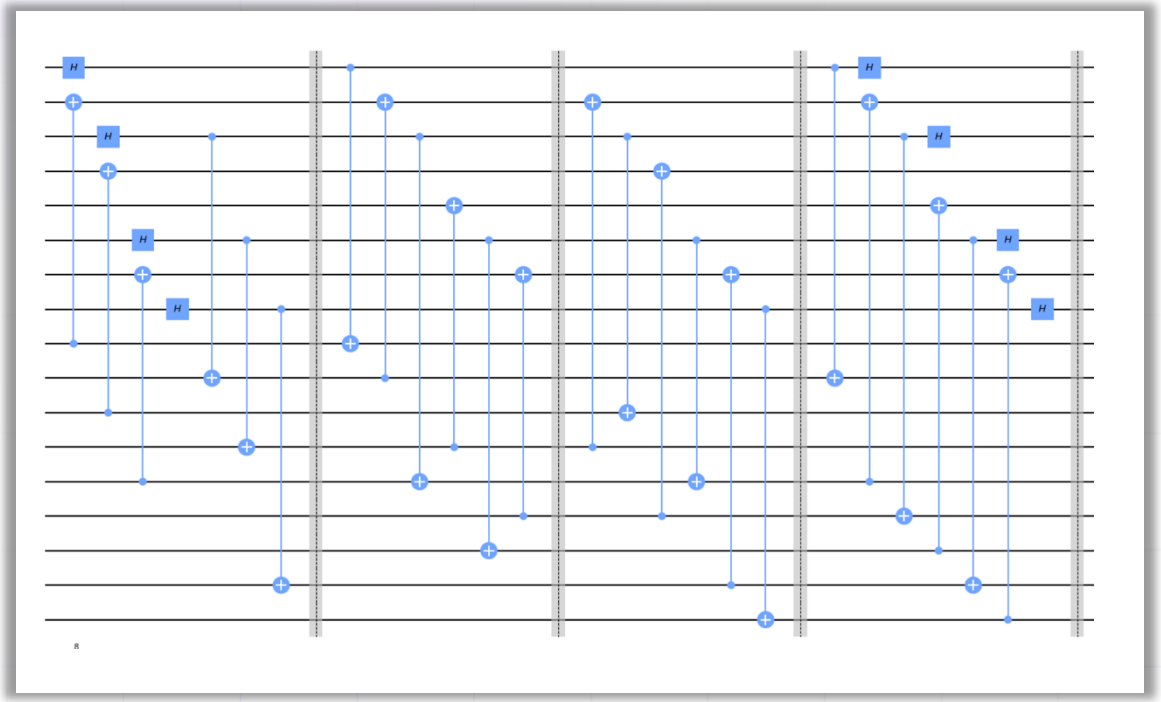
Global gates and native parallelism

Key notion: The same gate is applied on many qubits in parallel

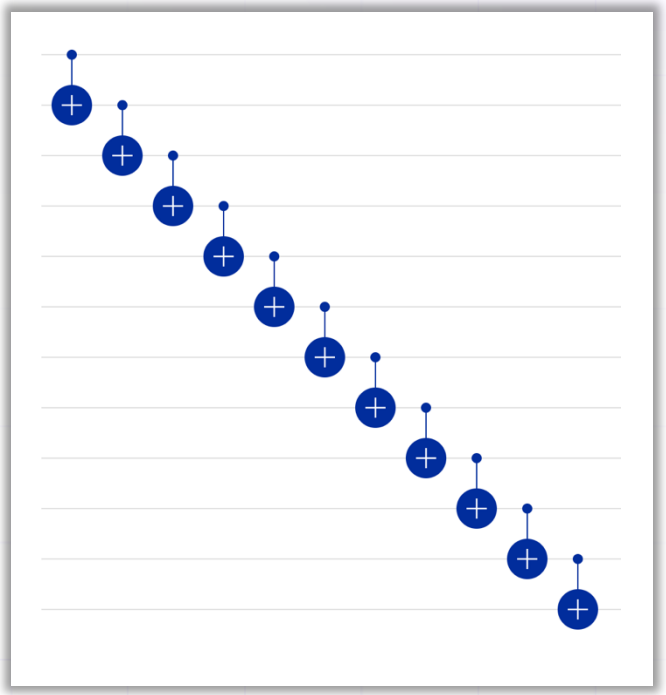


A fundamental block: 1q gates plus a set of cliques representing multi-qubit gates

Parallelism is key



A round of syndrome extraction for the surface code

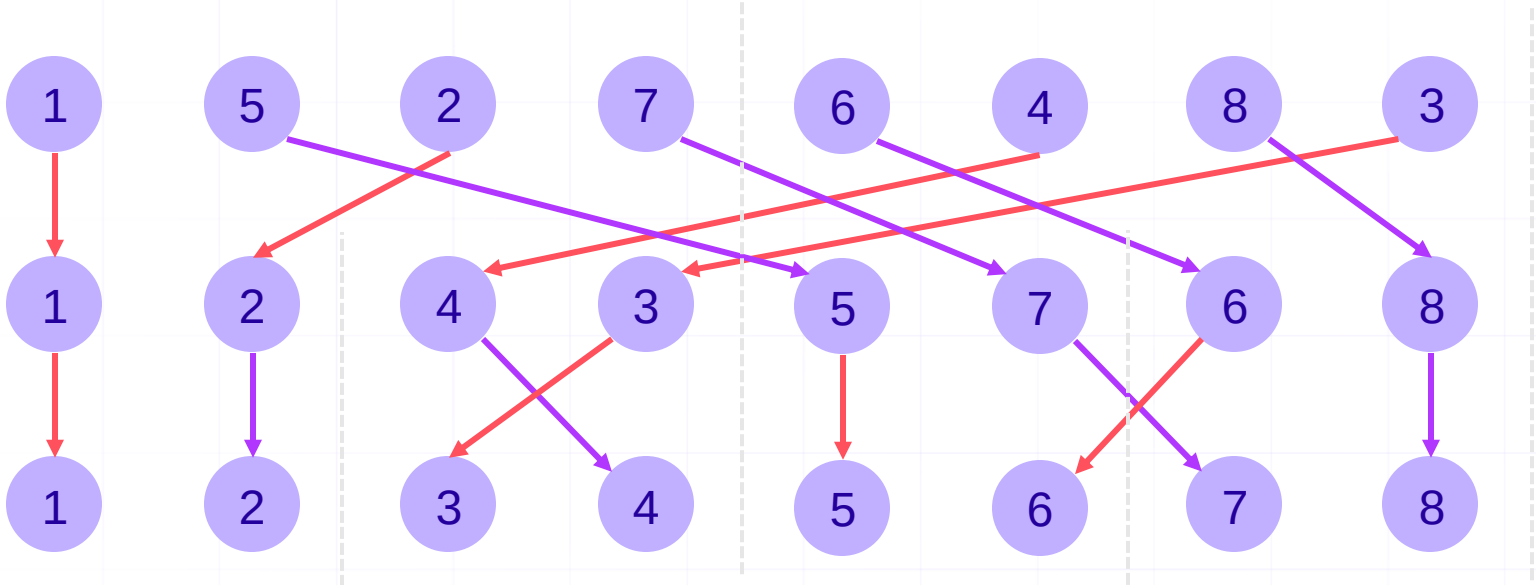


A staircase circuit

A Co-designed Compilation Mindset

~~"All to All"~~ ⇒ **Efficient parallel swap**

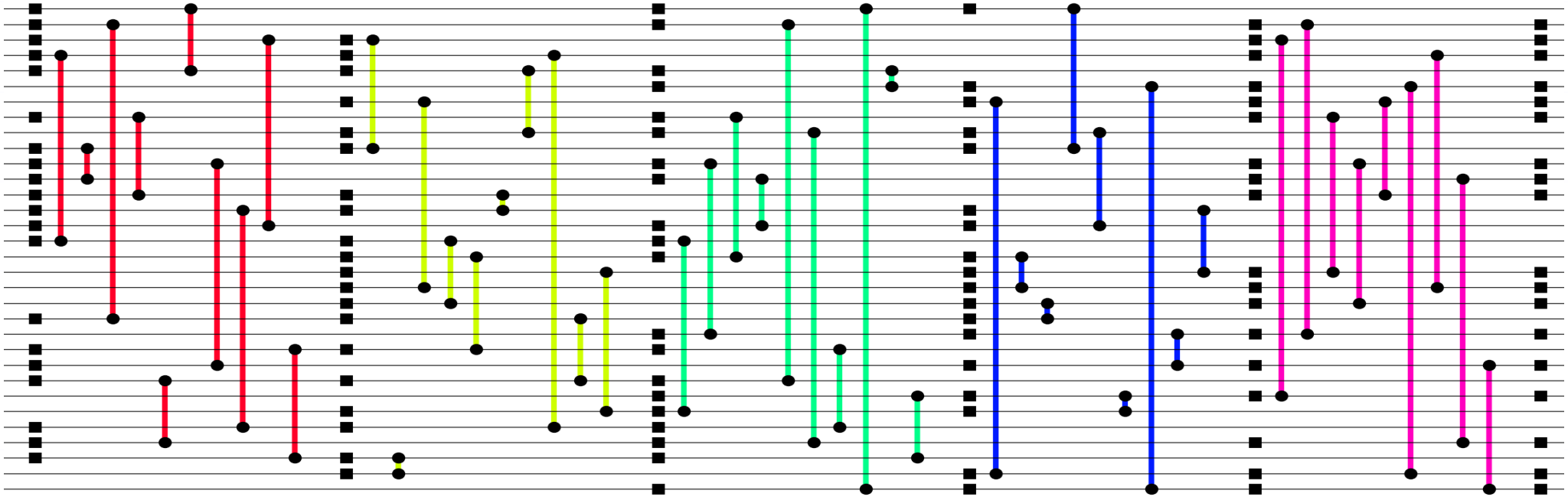
Atoms can **be efficiently sorted in $log(N)$** parallel moves.



A Co-designed Compilation Mindset

“All to All” ⇒ Efficient parallel swap

Sequential gates ⇒ Parallel layers



Programming neutral-atom quantum computers

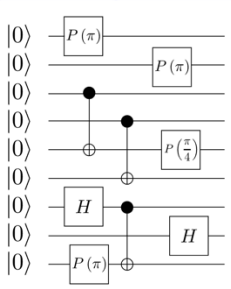
Bloqade

Kirin

Bloqade is QuEra Computing's software development kit (SDK) for neutral atom quantum computers. It is designed to be a hub of embedded domain-specific languages (eDSLs) for neutral atom quantum computing. Bloqade is built on top of Kirin, the Kernel Intermediate Representation Infrastructure.

Kirin is the **K**ernel **I**ntermediate **R**epresentation **In**frastructure developed. It is a compiler infrastructure for building compilers for embedded domain-specific languages (eDSLs) that target scientific computing kernels especially for quantum computing use cases where domain-knowledge in quantum computation is critical in the implementation of a compiler.

Final words



Algorithmic pipeline

Error-correcting code choice
Compilation
Gate design
Protocol layout + spacetime optimization

Co-Design

Native hardware capabilities

Speed
Qubit connectivity
Parallelization
Universal gate-set
Biased noise & erasure

How can we leverage neutral atoms' strengths to design efficient algorithms?