QCE guest lecture on

Opportunities of Quantum Computing for CFD Applications

Matthias Möller

Delft University of Technology
Delft Institute of Applied Mathematics

QCFD team

Merel Schalkers

Calin Georgescu

Monica Lacatus

Alex Sturges

Mission: Develop efficient and implementable end-to-end quantum methods for CFD applications

efficient = with provable advantage in terms of computational speed and/or memory requirement
implementable = worked out as quantum circuits in terms of (native) quantum gates, no c-U tricks
end-to-end = no oracles, no wishful thinking, no idealizing assumptions, demonstrated on simulators!

Motivation

Challenges in grid-based CFD

- Large memory requirements for high-resolution simulations $\sim N_x N_y N_z$
- Long compute times for time-dependent flows $\sim N_t$

$\Big\}$ $\sim N_t N_x N_y N_z$ ops

Possible solution: quantum computers

- Can encode N different states in $n = \log_2 N$ qubits via superposition of states
- Can operate on all $N = 2^n$ different states simultaneously (quantum parallelism)

Common pitfalls of quantum computing

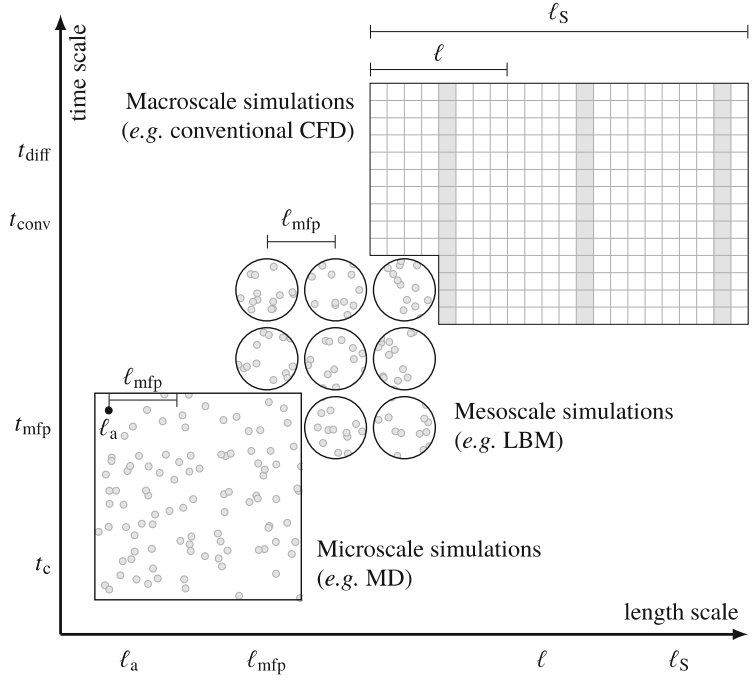
- How to encode data efficiently in the quantum register?
- How to load data efficiently into the quantum register (state preparation)?
- How to design the quantum algorithm as efficient quantum circuit?
- How to extract the algorithm's outcome efficiently via measurement?
- How to implement and test the algorithm on a quantum simulator/computer?

Outline

- Introduction into the lattice Boltzmann method (LBM)

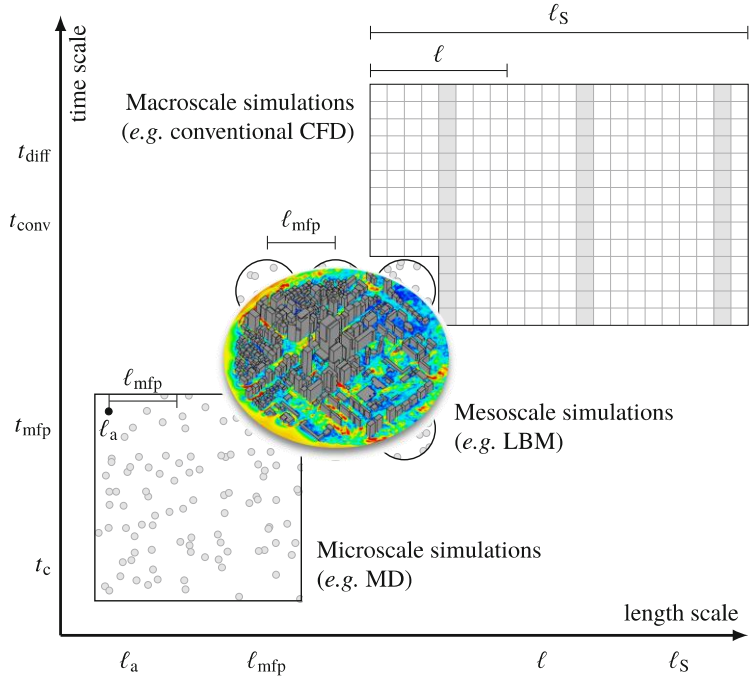
- Quantum LBM
  - Data encoding
  - Streaming & collision operators
  - Momentum exchange method

Hierarchy of length and time scales



DOI: 10.1007/978-3-319-44649-3

Hierarchy of length and time scales – the mesoscale



DOI: 10.1007/978-3-319-44649-3

- Kinetic theory based on **distribution function**

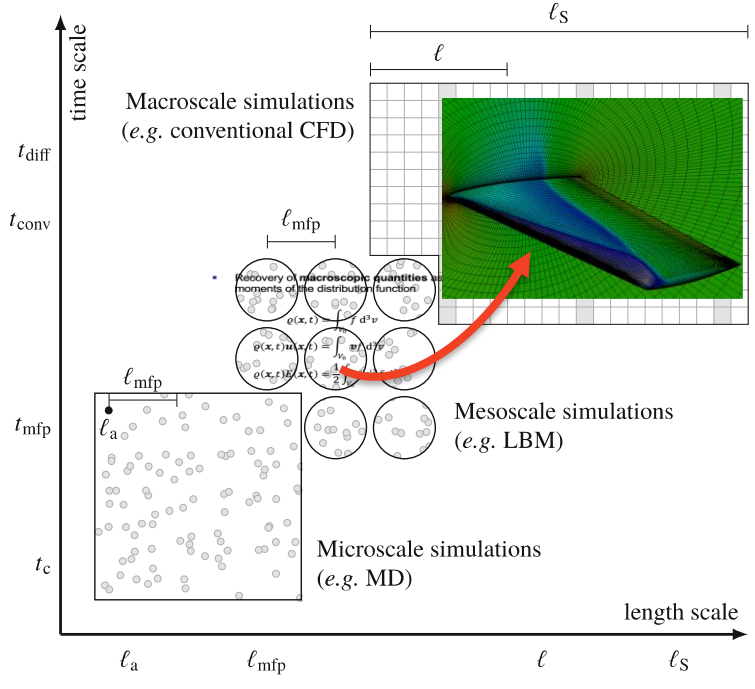
$$f(\boldsymbol{x}, \boldsymbol{v}, t) \qquad \left[\frac{\mathrm{kg\ s^3}}{\mathrm{m^6}}\right]$$

- **Interpretation**: $f(\boldsymbol{x}, \boldsymbol{v}, t)$ represents the density of mass in the physical space ($\boldsymbol{x}$) and the velocity space ($\boldsymbol{v}$) at time t

- When left alone for sufficiently long the distribution function will reach equilibrium

$$f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{v}, t)$$

Hierarchy of length and time scales – the mesoscale



- Recovery of **macroscopic quantities** as the moments of the distribution function

$$\varrho(\boldsymbol{x},t) = \int_{V_0} f \, \mathrm{d}^3 v$$

$$\varrho(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t) = \int_{V_0} \boldsymbol{v} f \, \mathrm{d}^3 v$$

$$\varrho(\boldsymbol{x},t)E(\boldsymbol{x},t) = \frac{1}{2}\int_{V_0} |\boldsymbol{v}|^2 f \, \mathrm{d}^3 v$$

DOI: 10.1007/978-3-319-44649-3

Boltzmann equation

- Total derivative of f with respect to time yields

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \left(\frac{\partial f}{\partial t}\right)\underbrace{\frac{\mathrm{d}t}{\mathrm{d}t}}_{=1} + \left(\frac{\partial f}{\partial x_\beta}\right)\underbrace{\frac{\mathrm{d}x_\beta}{\mathrm{d}t}}_{=v_\beta} + \left(\frac{\partial f}{\partial v_\beta}\right)\underbrace{\frac{\mathrm{d}v_\beta}{\mathrm{d}t}}_{=\frac{\mathrm{d}^2 x_\beta}{\mathrm{d}^2 t}=\frac{F_\beta}{\varrho}}$$

- The **Boltzmann equation**

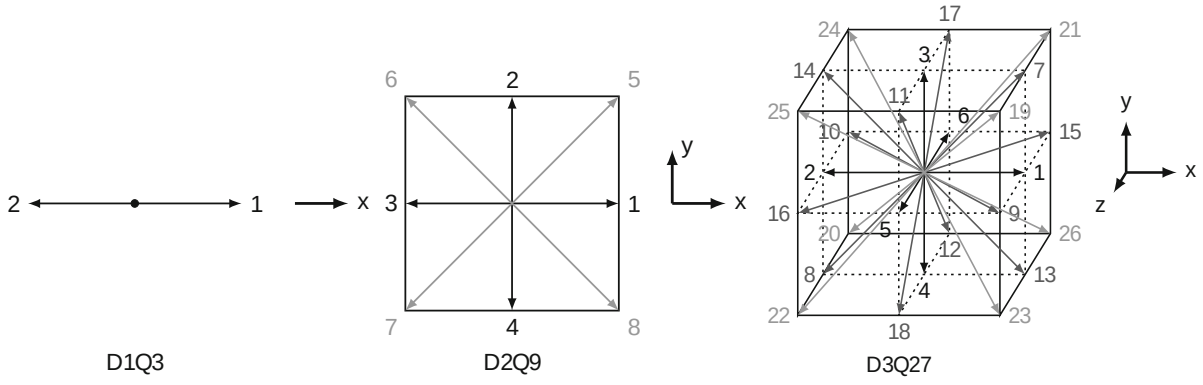
$$\frac{\partial f}{\partial t} + v_\beta \frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\varrho}\frac{\partial f}{\partial v_\beta} = \Omega(t)$$

with the **collision operator** $\Omega(t)$ representing the local redistribution of f due to collisions, e.g.,

$$\Omega^{\mathrm{BGK}}(t) = -\frac{1}{\tau}(f - f^{\mathrm{eq}})$$

Lattice Boltzmann method (LBM)

- Discretize the continuous velocity space into a set of **discrete velocities** $\boldsymbol{c}_\alpha = (c_{\alpha x}, c_{\alpha y}, c_{\alpha z})$, e.g.,



$|\boldsymbol{c}_\alpha| \in \{0, 1\}$ $|\boldsymbol{c}_\alpha| \in \{0, 1, \sqrt{2}\}$ $|\boldsymbol{c}_\alpha| \in \{0, 1, \sqrt{2}, \sqrt{3}\}$

- **Notation**: $\mathrm{D}d\mathrm{Q}N_v$ with spatial dimension d and number of discrete velocities N_v

Lattice Boltzmann method (LBM)

- Then $f(\boldsymbol{x}, \boldsymbol{v}, t)$ becomes a **discrete-velocity distribution function**

$$f_\alpha(\boldsymbol{x}, t) = f(\boldsymbol{x}, \boldsymbol{c}_\alpha, t)$$

- Discretization in space and time yields

$$f_\alpha(\boldsymbol{x}_i + \boldsymbol{c}_\alpha \Delta t, t^n + \Delta t) = f_\alpha(\boldsymbol{x}_i, t^n) + \Omega_\alpha(\boldsymbol{x}_i, t^n)$$

for all discrete velocities $\alpha = 1, \dots, N_v$,
for all grid points $i = 1, \dots, N_x N_y N_z$ and
for all time steps $n = 0, \dots, N_t$

Lattice Boltzmann method (LBM)

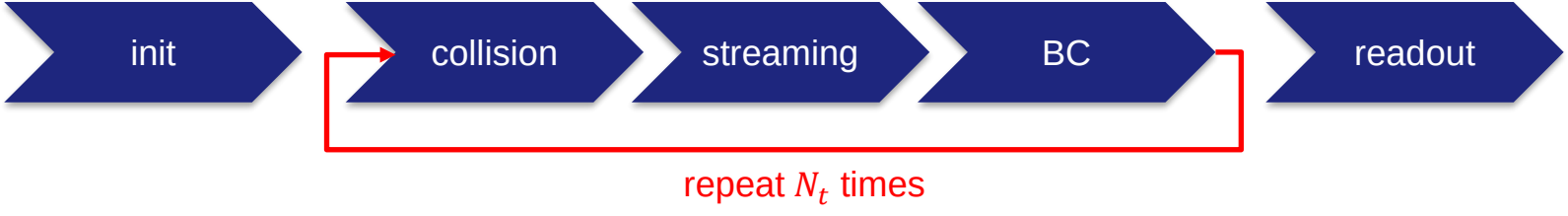
1. Update the moments and the equilibrium $f_i \rightarrow \varrho, \boldsymbol{u} \rightarrow f_i^{\mathrm{eq}}$
2. Compute the **collision** (or relaxation)

$$f_i^*(\boldsymbol{x}_j, t^n) = f_i(\boldsymbol{x}_j, t^n)\left(1 - \frac{\Delta t}{\tau}\right) + f_i^{\mathrm{eq}}(\boldsymbol{x}_j, t^n)\frac{\Delta t}{\tau}$$

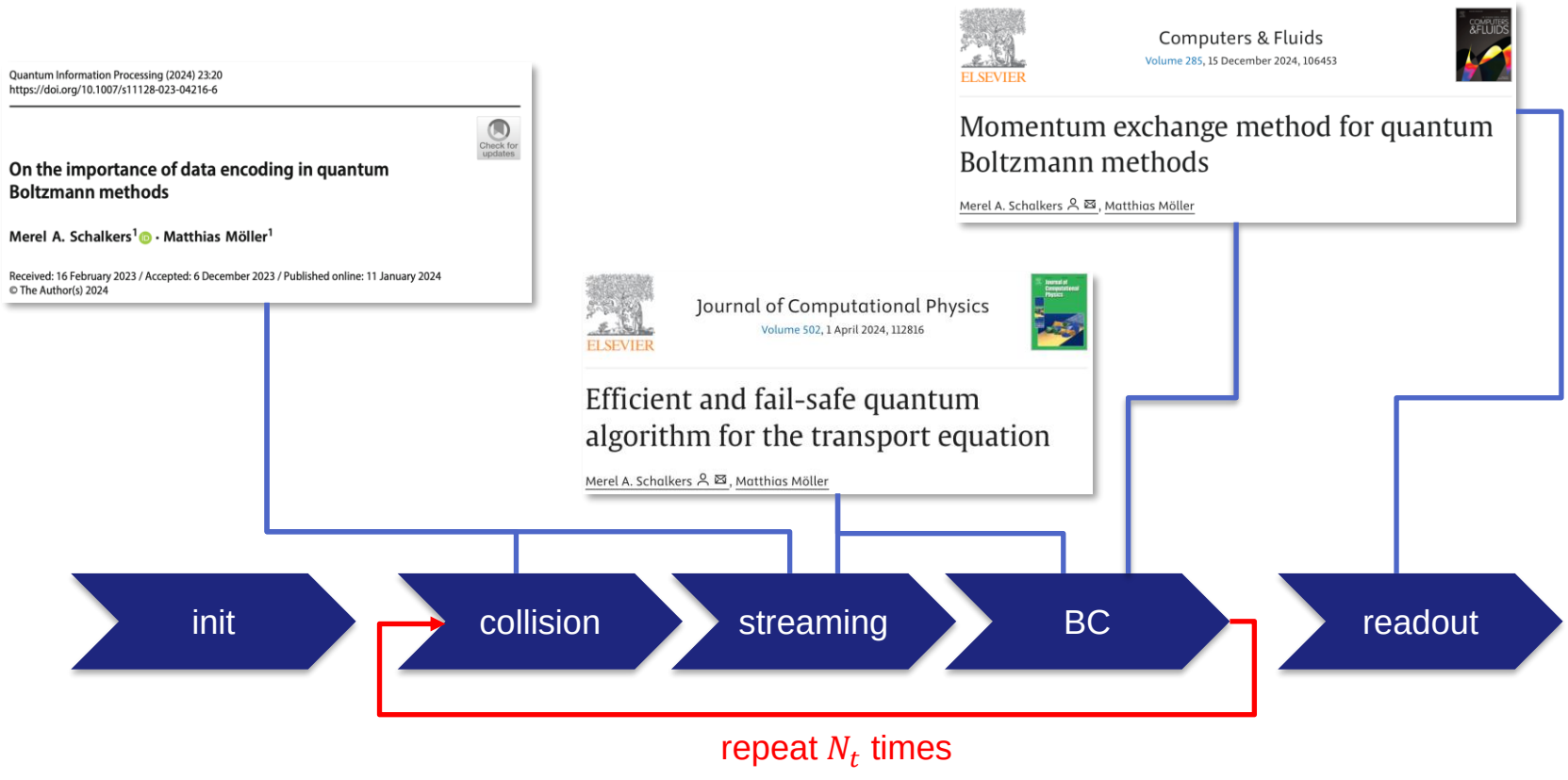
3. Compute the **streaming** (or propagation)

$$f_i(\boldsymbol{x}_j + \boldsymbol{c}_i \Delta t, t^n + \Delta t) = f_i^*(\boldsymbol{x}_j, t^n)$$

4. Apply **boundary conditions**
5. Set $t^{n+1} = t^n + \Delta t$ and go back to step 1



Quantum lattice Boltzmann method (LBM)



Collisionless quantum lattice Boltzmann method (CQLBM)

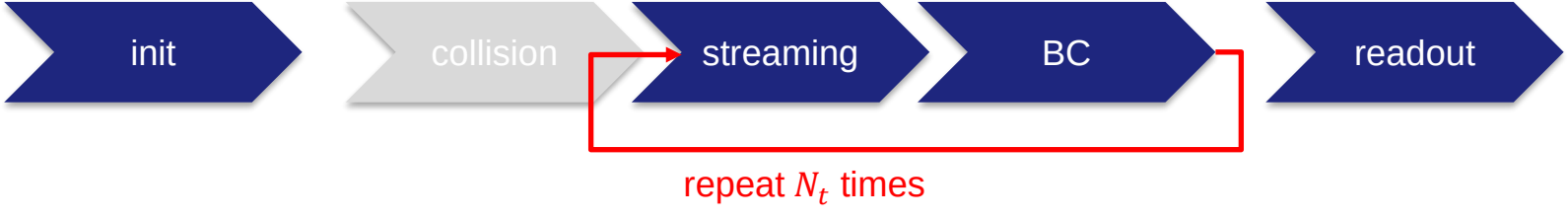
- Then $f(\boldsymbol{x}, \boldsymbol{v}, t)$ becomes a **discrete-velocity distribution function**

$$f_\alpha(\boldsymbol{x}, t) = f(\boldsymbol{x}, \boldsymbol{c}_\alpha, t)$$

- Discretization in space and time and ignoring the collision term yields

$$f_\alpha(\boldsymbol{x}_i + \boldsymbol{c}_\alpha \Delta t, t^n + \Delta t) = f_\alpha(\boldsymbol{x}_i, t^n) + \Omega_\alpha(x_i, t^n)$$

for all discrete velocities $\alpha = 1, \dots, N_v$, grid points $i = 1, \dots, N_x N_y N_z$ and time steps $n = 0, \dots, N_t$



Data encoding of the grid

Classical register of size $N_x N_y N_v$

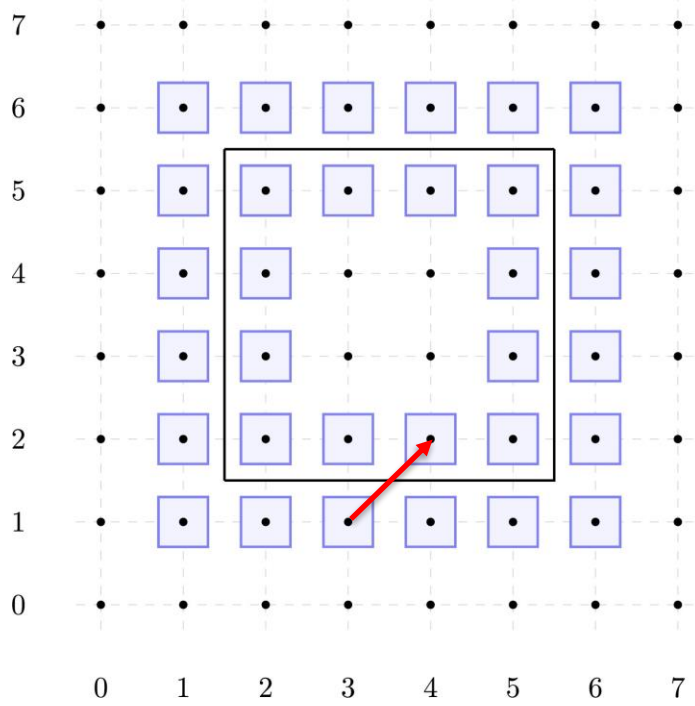
$$f[N_x][N_y][N_v]$$

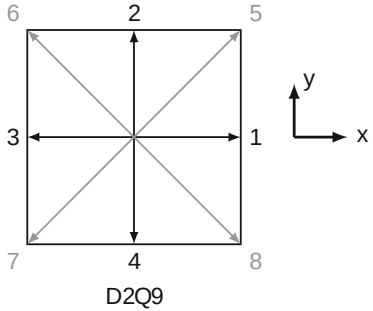
$$f[ix][iy][\alpha] \;\mapsto\; f_\alpha(x_{ix}, y_{iy})$$

Example for D2Q9

$$\text{size } 8 \cdot 8 \cdot 9 = 576$$

$$f[3][1][5] \;\mapsto\; f_5(x_3, y_1)$$





Data encoding of the grid

Quantum register of size $n_x + n_y + 2n_v$ $(n = \log_2 N)$

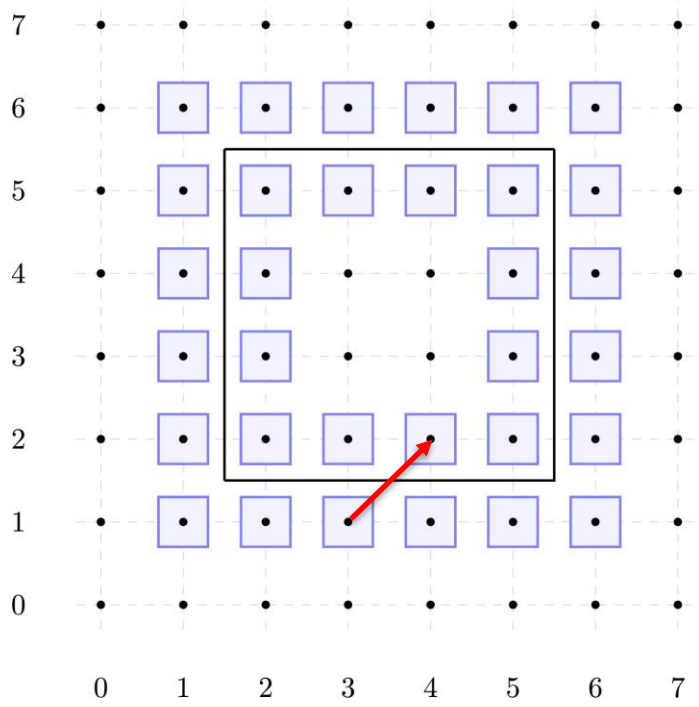
$$|n_x\rangle \otimes |n_y\rangle \otimes |n_v\rangle \otimes |n_v\rangle = \sum_{\ell=0}^{2^{n_x+n_y+2n_v-1}} \rho_\ell |\ell\rangle$$

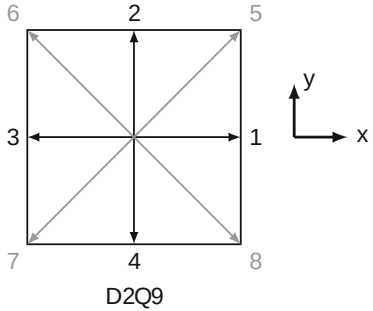
$$\rho_{ix\ iy\ \alpha x\ \alpha y} \mapsto \sim f_{\alpha x,\alpha y}(x_{ix}, y_{iy})$$

Example for D2Q9

$$\text{size } 3 + 3 + 2 \cdot 2 = 10$$

$$\rho_{011\ 001\ 01\ 01} \mapsto \sim f_5(x_3, y_1)$$





D2Q9

Data encoding of the velocity

$$\left|v_{\mathrm{dir}} v_{n_v-1} \dots v_1\right\rangle = \begin{bmatrix} -v_{\min} \\ -v_{\min} - \Delta v \\ \vdots \\ -v_{\max} \\ v_{\min} \\ v_{\min} + \Delta v \\ \vdots \\ v_{\max} \end{bmatrix}$$

- **Example**: Let $v_{\min} = 1$ and $\Delta v = 0.5$ then
  - $|v\rangle = |\ \ 1.5\rangle = |101\rangle$
  - $|v\rangle = |-1.5\rangle = |001\rangle$

Remark:
All standard $DdQN_v$ velocity stencils can be realized with two qubits per direction, i.e.

$$|u_{\mathrm{dir}} u_1\rangle \otimes |v_{\mathrm{dir}} v_1\rangle \otimes |w_{\mathrm{dir}} w_1\rangle = \{\cdot = |00\rangle = |10\rangle, \quad \leftarrow = |01\rangle, \quad \rightarrow = |11\rangle\}^{\otimes d}$$

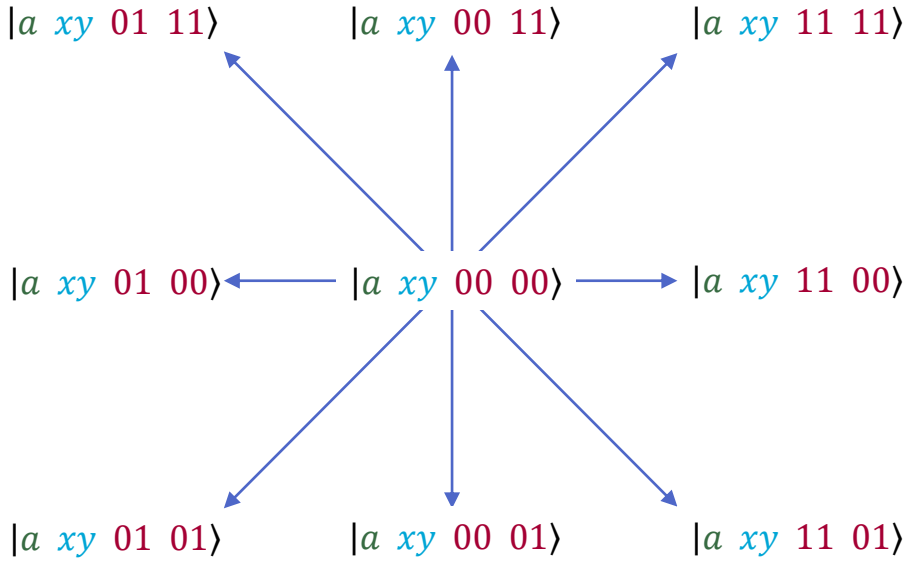
A variant of the LBM is the discrete-velocity method with many more discrete velocities which can be realized with the above data encoding at the cost of more qubits per direction.

Amplitude-based data encoding

$$|\psi\rangle = \sum_{\ell=0}^{2^{4d-2+n_x+n_y+n_z+3n_v}-1} \rho_\ell \left| \underbrace{a_{4d-2} \dots a_1}_{4d-2 \text{ ancillas}} \; \underbrace{x_{n_x} \dots x_1 \; y_{n_y} \dots y_1 \; z_{n_z} \dots z_1}_{n_x+n_y+n_z \text{ grid qubits}} \; \underbrace{u_{n_v} \dots u_1 \; v_{n_v} \dots v_1 \; w_{n_v} \dots w_1}_{3n_v \text{ velocity qubits}} \right\rangle$$

- Ancillas are used to hold decision variables, e.g., to detect objects, etc.
- Example of **memory savings** for D3Q27 on 1024^3 grid using **superposition of states**
  - classical: $27 \cdot 1024^3 =$ ~29 billion floating-point numbers (~232 GB FP64)
  - quantum: $4 \cdot 3 - 2 + 3 \cdot 10 + 3 \cdot 2 =$ 46 qubits
- Can we also reduce the **'compute time'** (compute complexity) using **quantum parallelism**?

A closer look at the encoding of the discrete velocities for D2Q9

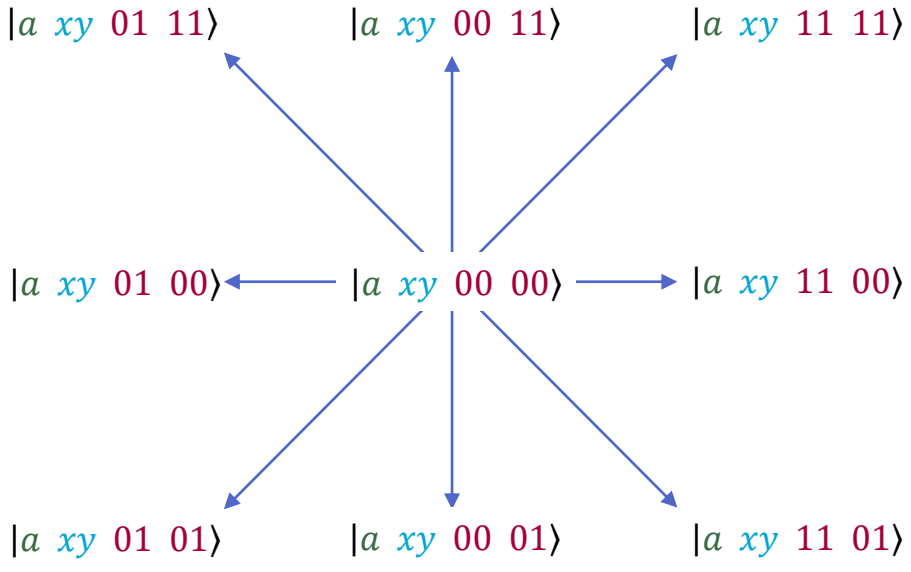


A closer look at the encoding of the discrete velocities for D2Q9

Since particle densities are encoded in the amplitudes, **streaming** amounts to shifting them between grid positions depending on their velocities, e.g.

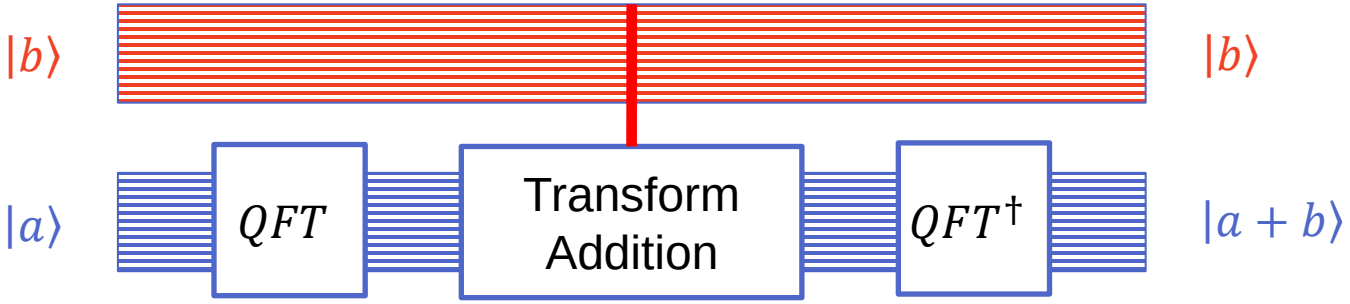
$$
\begin{aligned}
\rho_{a\ x.y\ 00\ 00} &\mapsto \rho_{a\ x.y\ 00\ 00} \\
\rho_{a\ x.y\ 11\ 00} &\mapsto \rho_{a\ x+1.y\ 11\ 00} \\
\rho_{a\ x.y\ 01\ 00} &\mapsto \rho_{a\ x-1.y\ 01\ 00} \\
\rho_{a\ x.y\ 00\ 11} &\mapsto \rho_{a\ x.y+1\ 00\ 11} \\
\rho_{a\ x.y\ 00\ 01} &\mapsto \rho_{a\ x.y-1\ 00\ 01} \\
\rho_{a\ x.y\ 11\ 11} &\mapsto \rho_{a\ x+1.y+1\ 11\ 11} \\
&\dots
\end{aligned}
$$

Apply streaming primitive to all grid positions simultaneously controlled on the value of the velocity qubits

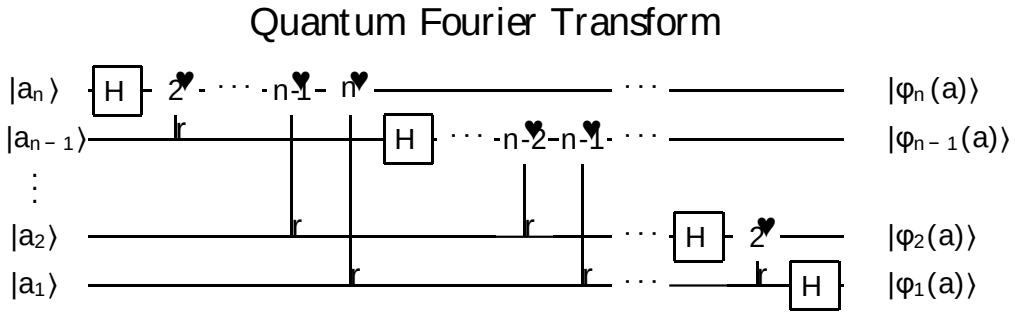


Sketch of the streaming operation

Our streaming primitive is based on **Draper's quantum adder** ("Addition on a Quantum Computer" '98) which is an in-place quantum adder for two integer values that works without any ancillas



Intermezzo: Draper's quantum adder



$$|a_n\rangle \longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n)|1\rangle) \qquad \text{Hadamard transform}$$

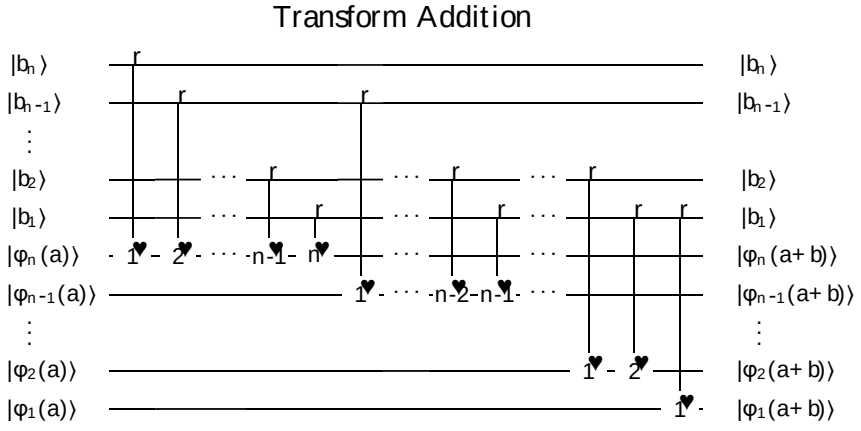
$$\longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n a_{n-1})|1\rangle) \qquad R_2 \text{ rotation conditioned on } a_{n-1}$$

$$\vdots \qquad\qquad \vdots$$

$$\longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n a_{n-1} \ldots a_1)|1\rangle) \qquad R_n \text{ rotation conditioned on } a_1$$

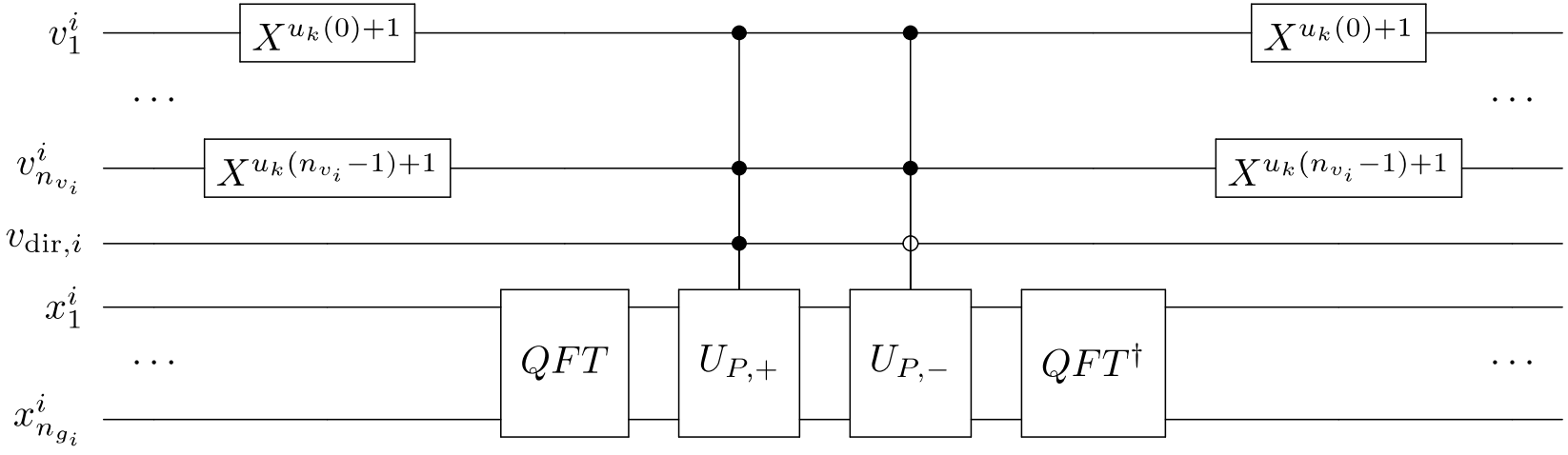
$$= |\varphi_n(a)\rangle$$

Intermezzo: Draper's quantum adder



$$
\begin{aligned}
|\varphi_n(a)\rangle &\longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n a_{n-1}\ldots a_1 + 0.b_n)|1\rangle) && R_1 \text{ rotation from } b_n \\
&\longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n a_{n-1}\ldots a_1 + 0.b_n b_{n-1})|1\rangle) && R_2 \text{ rotation from } b_{n-1} \\
&\qquad\vdots && \vdots \\
&\longrightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + e(0.a_n a_{n-1}\ldots a_1 + 0.b_n b_{n-1}\ldots b_1)|1\rangle) && R_n \text{ rotation from } b_1 \\
&= |\varphi_n(a+b)\rangle
\end{aligned}
$$

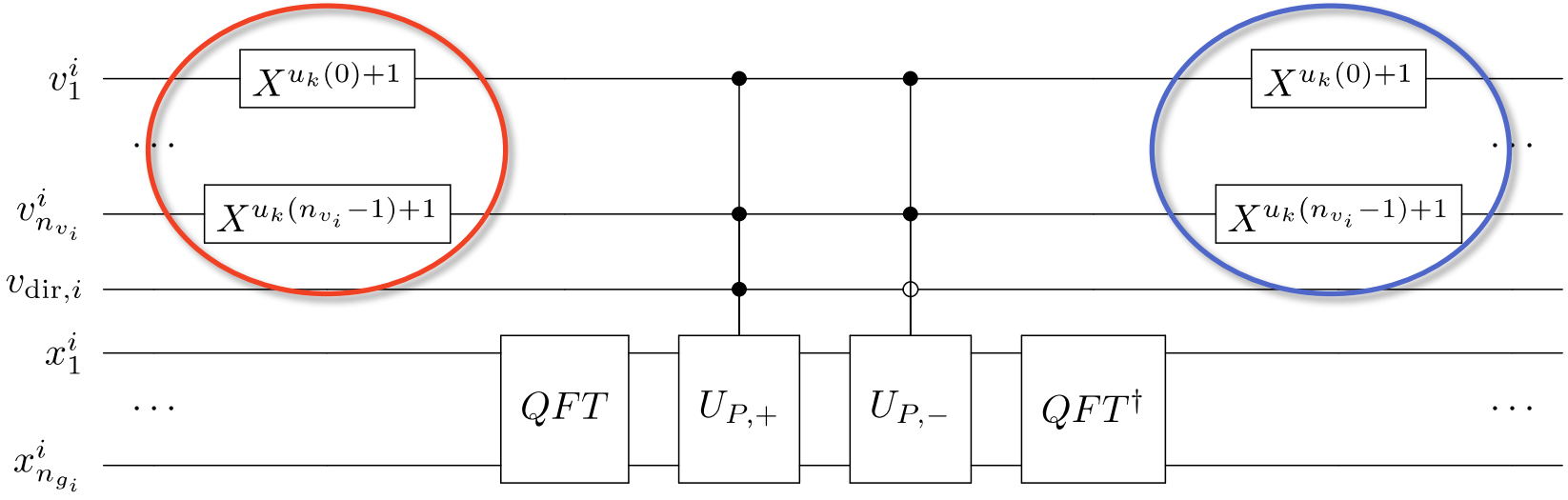
Streaming in Fourier space (schematic view)



Streaming in Fourier space (schematic view)

Set all velocity qubits that we want to stream in the i-th direction "on"

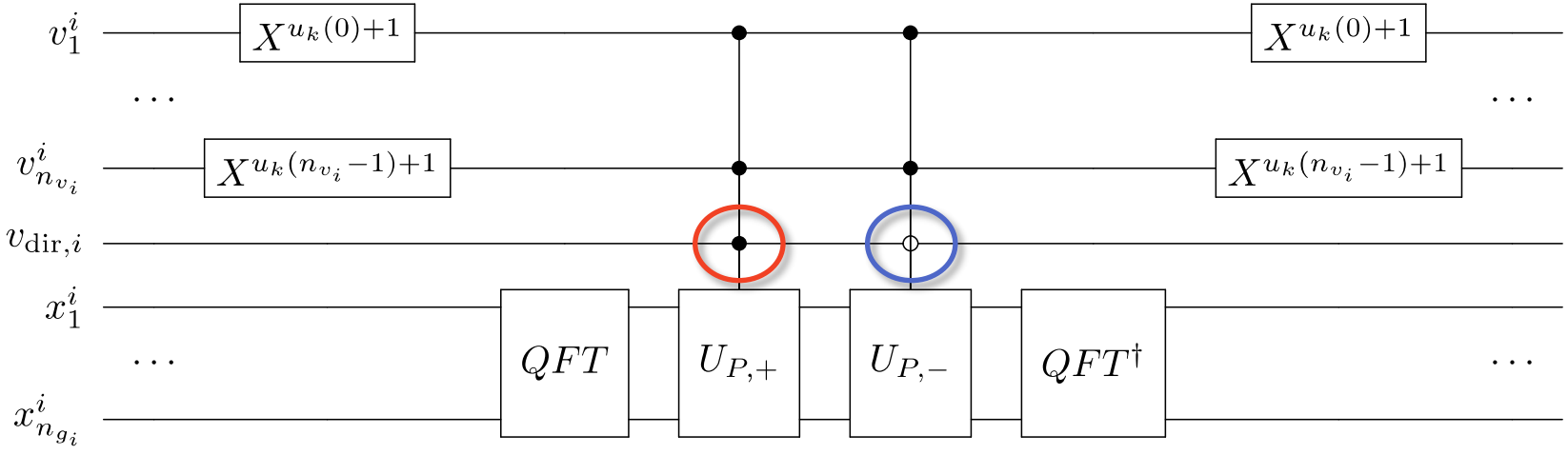
Reset the velocity qubits



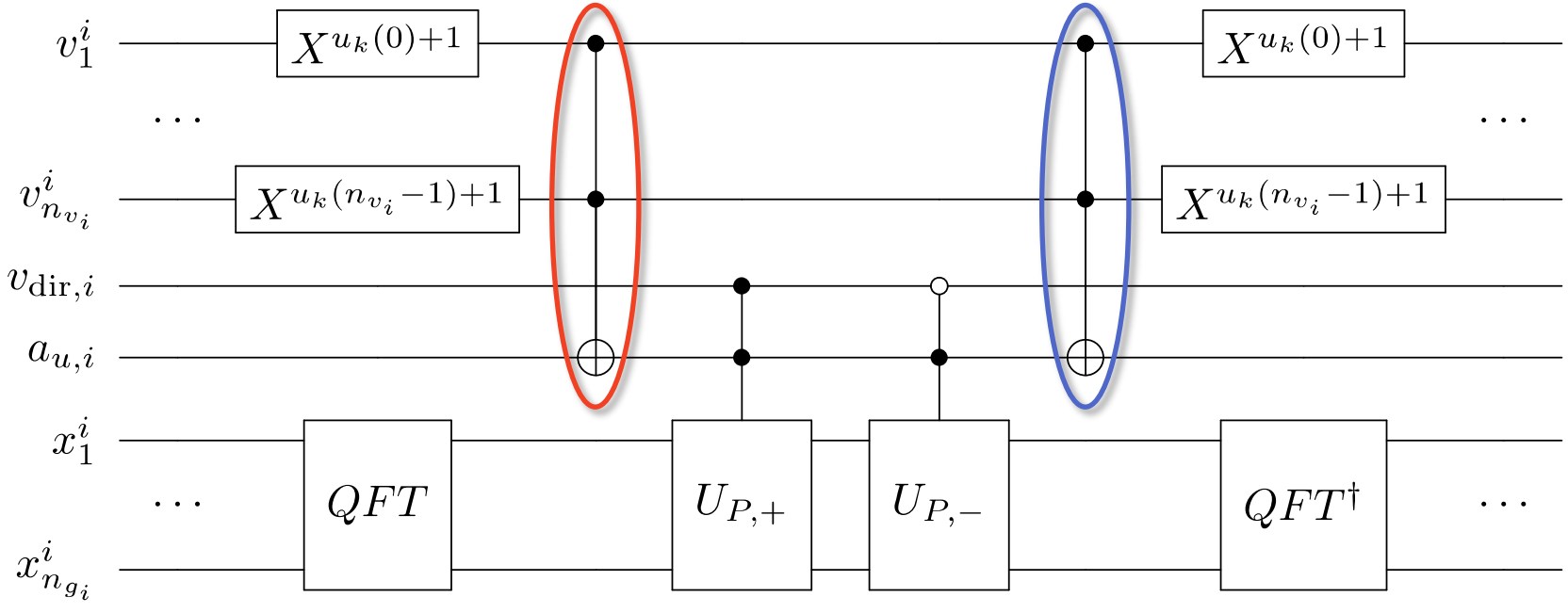
Example: want to stream $c_5 = 101$, set $\tilde{c}_5 = 111$ so that the multi-controlled $U_{P,+}$ (incrementer) or the multi-controlled $U_{P,-}$ (decrementer) get activated

Streaming in Fourier space (schematic view)

If $v_{\mathrm{dir},i}$ is "on" we stream in the positive direction otherwise ("off") we stream in the negative direction



Streaming in Fourier space (schematic view)



The ancillary qubit $a_{u,i}$ reduces the amount of multi-controlled $U_{P,*}$ operations

Streaming in Fourier space (the math)

- **Claim**: $|j+1\rangle = QFT^{\dagger}\, U_{P,+}\, QFT\, |j\rangle$
- QFT in a nutshell

$$QFT\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\omega_N^{kj}|k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{\frac{2i\pi kj}{N}}|k\rangle$$

- Since

$$\theta_j \coloneqq \frac{\pi}{2^{N-1-j}} = \frac{\pi 2^{j+1}}{N}$$

- Then

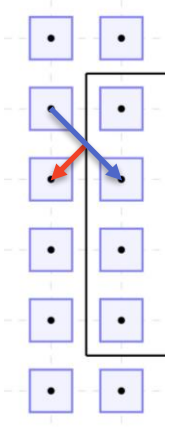
$$U_{P,+}\, QFT\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{\frac{2i\pi kj}{N}}e^{\theta_j}|k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{\frac{2i\pi k(j+1)}{N}}|k\rangle$$

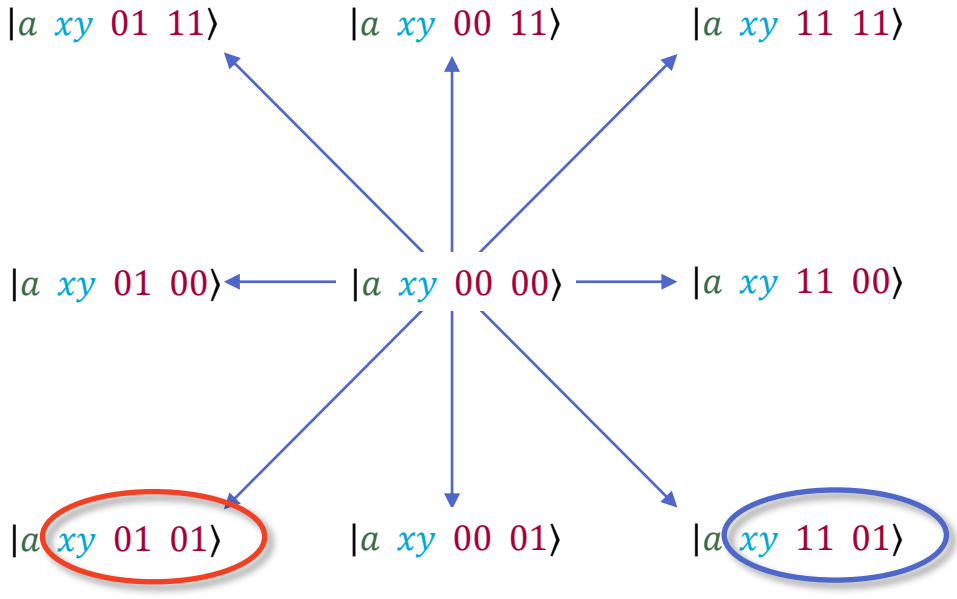
- Finally

$$QFT^{\dagger}\, U_{P,+}\, QFT\,|j\rangle = QFT^{\dagger}\,\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{\frac{2i\pi k(j+1)}{N}}|k\rangle = |j+1\rangle$$

Specular reflection boundary condition

If a particle needs to change its flow direction because of a reflection at the boundary it suffices to flip a single qubit and update the particle's grid position





Specular reflection boundary condition

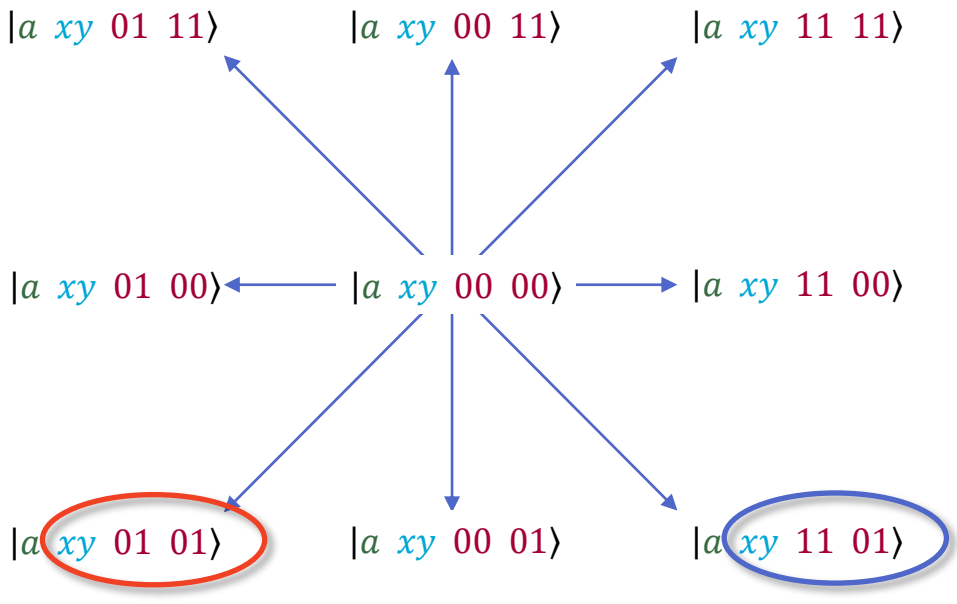
For all quantum states $|a\ x\ y\ 11\ 01\rangle$ with (x, y) *outside* and $(x+1, y)$ *inside* any obstacle with a vertical wall do

$$|a\ x\ y\ 11\ 01\rangle \rightarrow |a\ x\ y-1\ 01\ 01\rangle$$

- C^m-NOT for the bitflip
- ±1 Draper-type adder for the shift

Same shift-logic as for streaming

How do we detect that we are at a boundary, i.e. (x, y) is *outside* and $(x+1, y)$ is *inside* an obstacle?



Quantum comparator operator

- **Goal**: check $i \geq k$
- Start from $|0\rangle \otimes |i\rangle$ and
- subtract k from the state
  - if $i \geq k$

    $$|0\rangle \otimes |i-k\rangle$$

  - else $i < k$

    $$|2^{n+1} - 1 - k + i\rangle = |1\rangle \otimes |2^n - 1 - k + i\rangle$$

- add k to the last n qubits
  - if $i \geq k$

    $$|0\rangle \otimes |i\rangle$$

  - else $i < k$

    $$|1\rangle \otimes |i\rangle$$

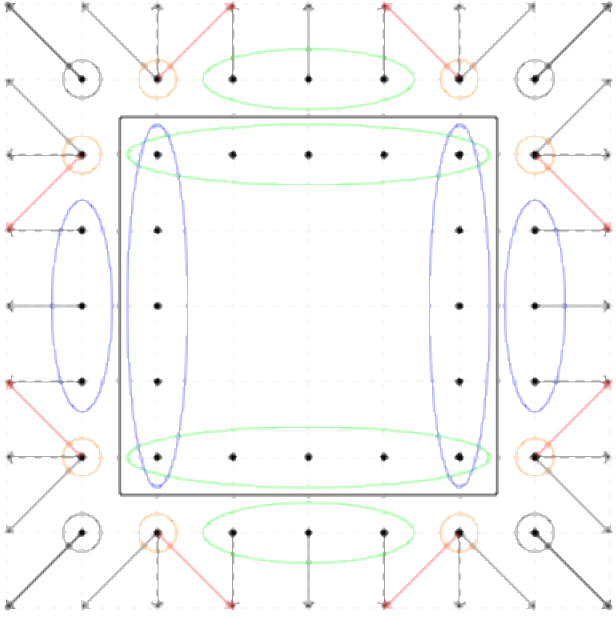
- use ancilla qubit as control

Exploiting quantum parallelism

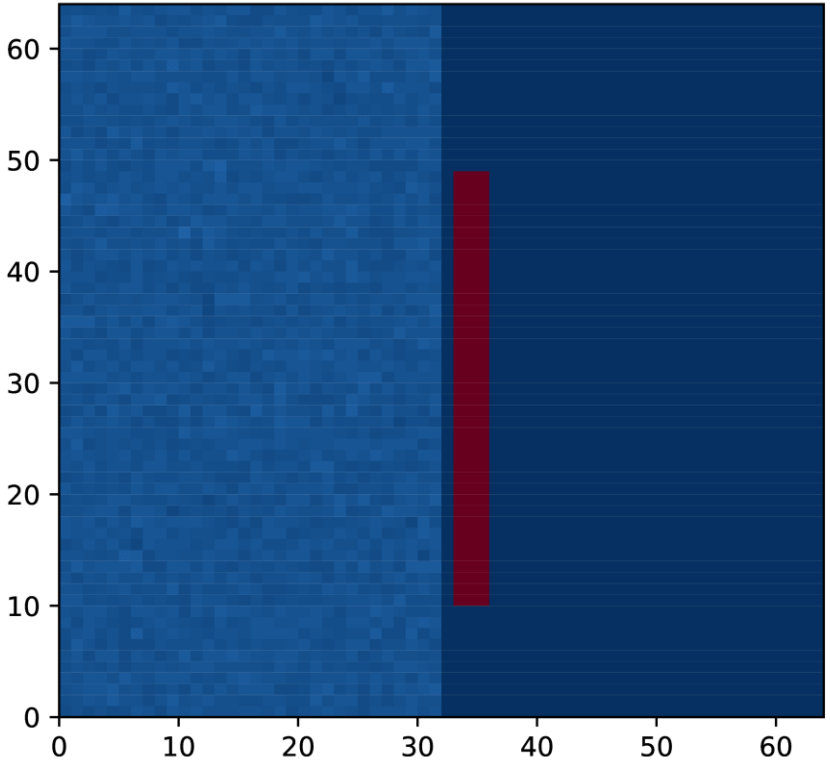
SeqFor 00, 01, 11
  QParFor "stream" along x direction
  QParFor "stream" along y direction

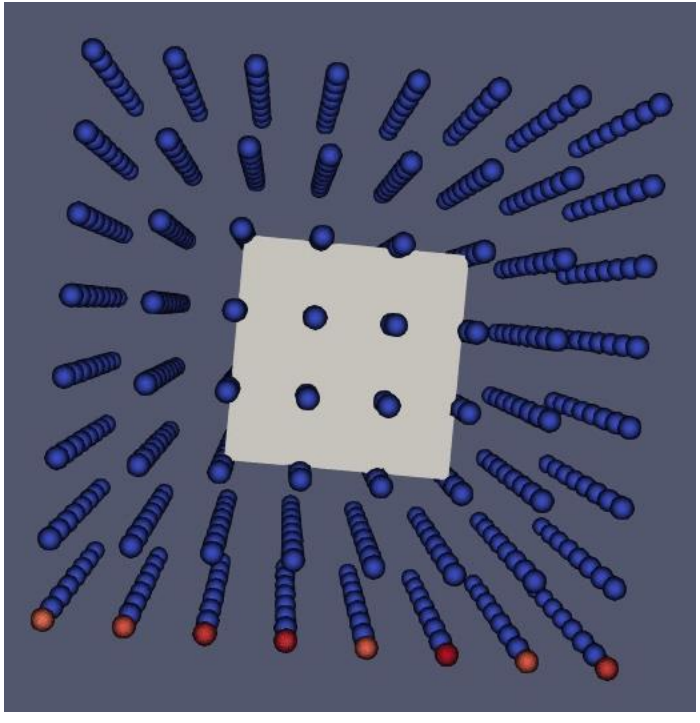
SeqFor ◯, ◯, ◯,…
  SeqFor 00, 01, 11
    QParFor "reflect" along x direction
    QParFor "reflect" along y direction

Sequential loops are "blocks" in the quantum circuit that can potentially be parallelized by adding more ancillas

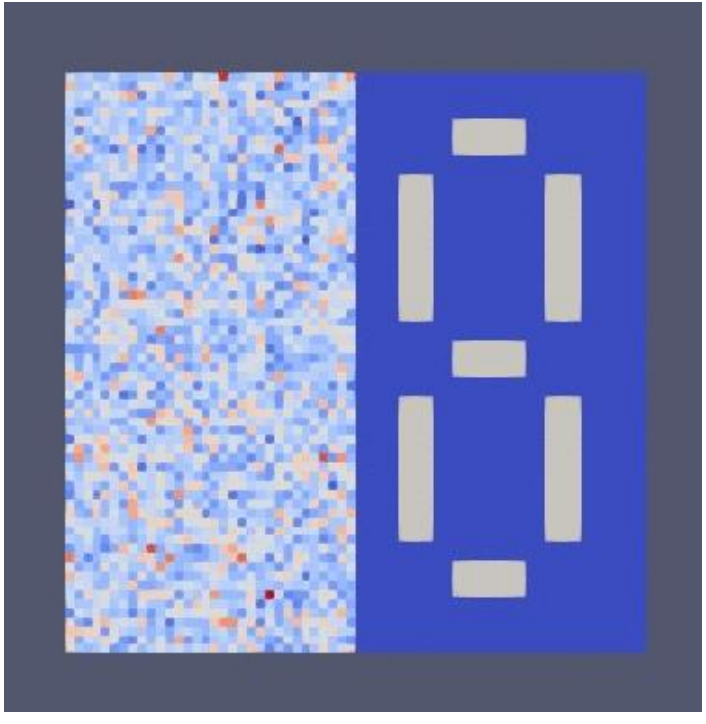


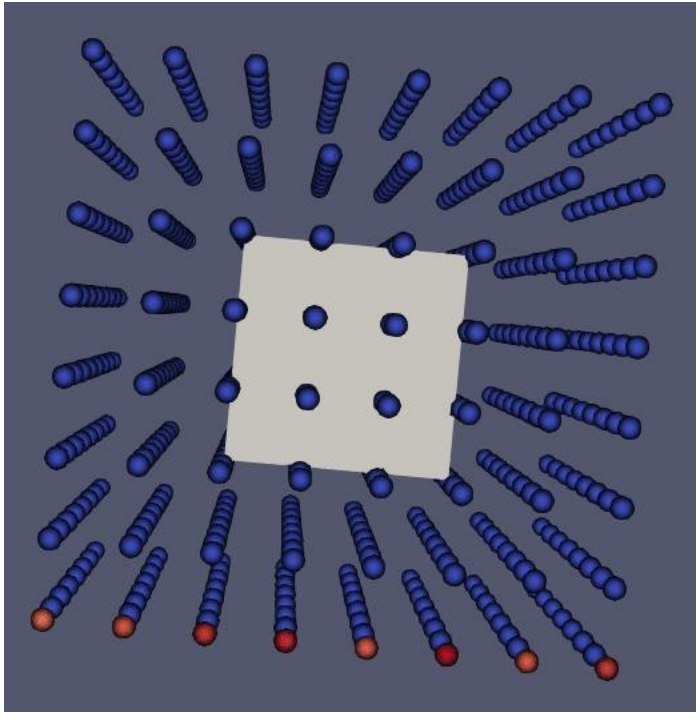
Results: D2Q9 on a 64x64 grid (left) and D3Q27 on an 8x8x8 grid (right)





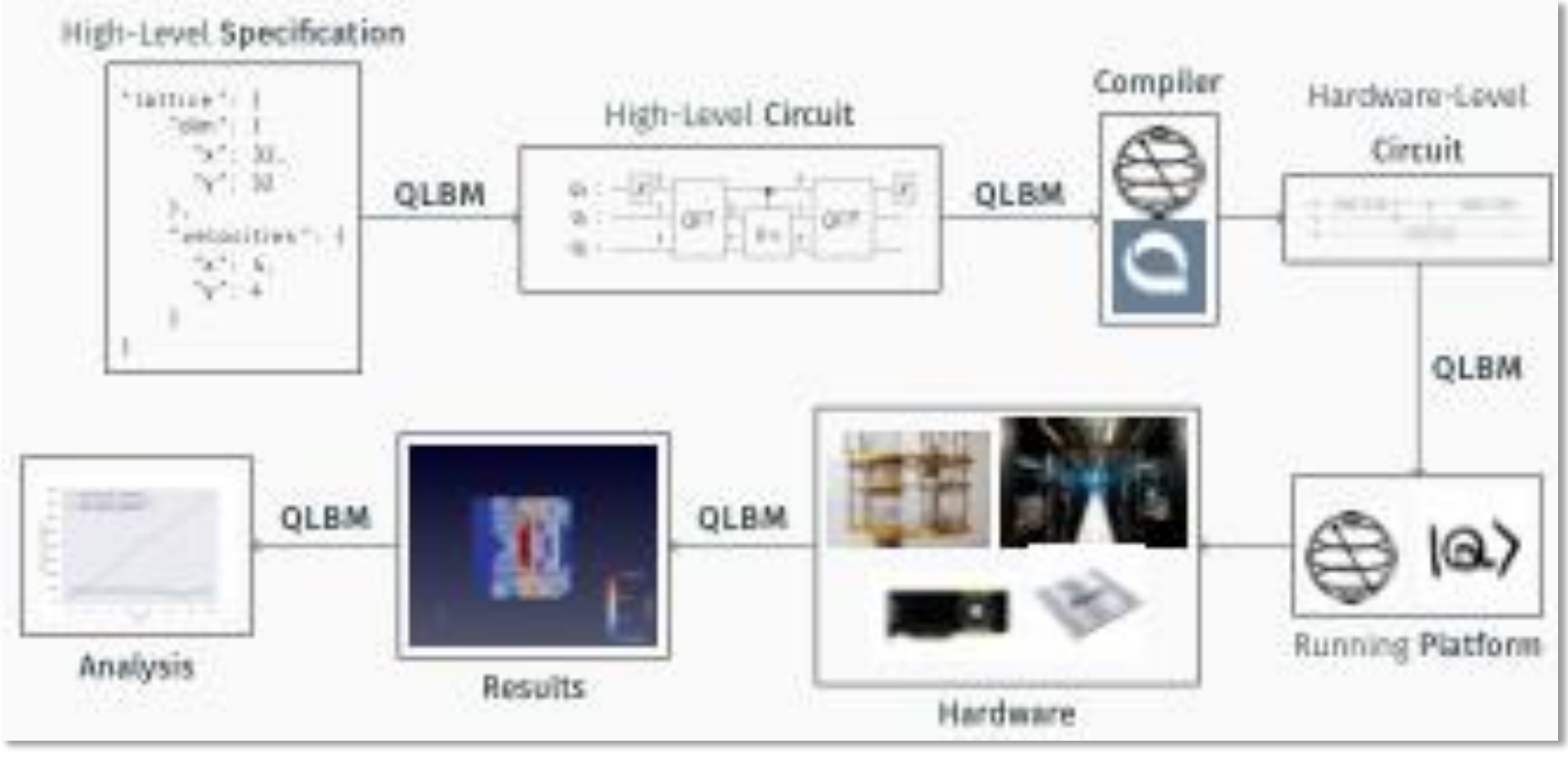
Results: D2Q9 on a 64x64 grid (left) and D3Q27 on an 8x8x8 grid (right)





QLBM software framework

https://github.com/qcfd-lab/qlbm



QLBM software framework

```
"dim": {
    "x": 16,
    "y": 16
},
"velocities": {
    "x": 4,
    "y": 4
}
},
"geometry": [[
    "x": [9, 12],
    "y": [3, 6],
    "boundary": "specular"
},
{
    "x": [9, 12],
    "y": [9, 12],
    "boundary": "bounceback"
}]
```

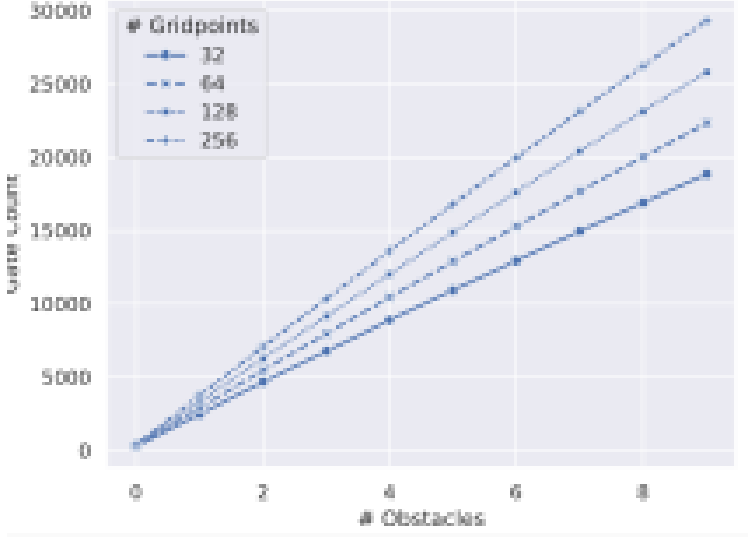
QLBM software framework

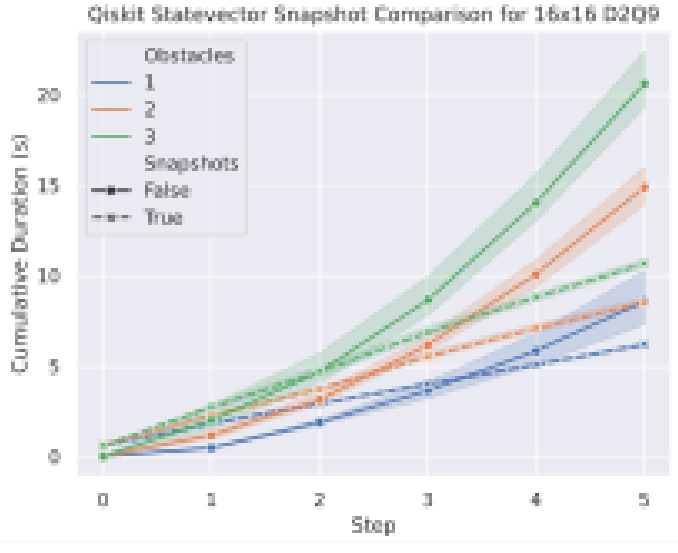
```
# Load the system specification
lattice = CollisionlessLattice("lattice.json")

# Define all parameters
cfg = SimulationConfig(
    initial_conditions=CollisionlessInitialConditions(lattice),
    algorithm=CQLBM(lattice),
    postprocessing=EmptyPrimitive(lattice),
    measurement=GridMeasurement(lattice),
    target_platform="QISKIT",
    compiler_platform="QISKIT",
    optimization_level=0,
    statevector_sampling=True,
    execution_backend=AerSimulator(method="statevector"),
    sampling_backend=AerSimulator(method="statevector"),
)
```

QLBM software framework

```
cfg.prepare_for_simulation()
# Create a runner object to simulate the circuit
runner = QiskitRunner(
    cfg,
    lattice,
)

# Simulate the circuits using both snapshots and sampling
runner.run(
    100,  # Number of time steps
    2**12,  # Number of shots per time step
    "output_dir/sim", # Output directory
    statevector_snapshots=True, # Performance opt.
)
```

QLBM software framework





Momentum exchange method

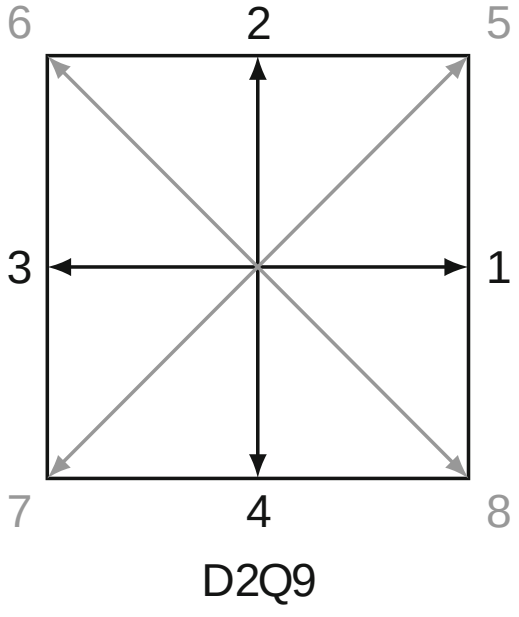
Force exerted on objects

$$\boldsymbol{F} = \sum_{\alpha=1}^{N_v} \boldsymbol{c}_\alpha f_\alpha(\boldsymbol{x}_{\text{fluid}}, t) - \boldsymbol{c}_{\bar{\alpha}} f_{\bar{\alpha}}(\boldsymbol{x}_{\text{fluid}}, t)$$

with $\boldsymbol{x}_{\text{fluid}}$ denoting the location in the fluid domain next to the obstacle, $\boldsymbol{c}_\alpha$ the discrete velocities, e.g.,

$$\boldsymbol{c}_\alpha = \begin{cases} (0,0) & \alpha = 0 \\ (1,0), (0,1), (-1,0), (0,-1) & \alpha = 1, \dots, 4 \\ (1,1), (-1,1), (-1,-1), (1,-1) & \alpha = 5, \dots, 8 \end{cases}$$

and $\boldsymbol{c}_{\bar{\alpha}}$ referring to the α-th velocity direction after contact with the obstacle



Momentum exchange method

Force exerted on objects

$$\boldsymbol{F} = \sum_{\alpha=1}^{N_v} \boldsymbol{c}_\alpha f_\alpha(\boldsymbol{x}_{\text{fluid}}, t) - \boldsymbol{c}_{\bar{\alpha}} f_{\bar{\alpha}}(\boldsymbol{x}_{\text{fluid}}, t)$$

with $\boldsymbol{x}_{\text{fluid}}$ denoting the location in the fluid domain next to the obstacle, $\boldsymbol{c}_\alpha$ the discrete velocities, e.g.,

$$\boldsymbol{c}_\alpha = \begin{cases} (0,0) & \alpha = 0 \\ (1,0), (0,1), (-1,0), (0,-1) & \alpha = 1, \dots, 4 \\ (1,1), (-1,1), (-1,-1), (1,-1) & \alpha = 5, \dots, 8 \end{cases}$$

and $\boldsymbol{c}_{\bar{\alpha}}$ referring to the α-th velocity direction after contact with the obstacle

Bounce back boundary conditions yield

$$\boldsymbol{c}_{\bar{\alpha}} = -\boldsymbol{c}_\alpha$$

$$f_\alpha(\boldsymbol{x}_{\text{fluid}}, t) = f_{\bar{\alpha}}(\boldsymbol{x}_{\text{fluid}}, t)$$

Hence

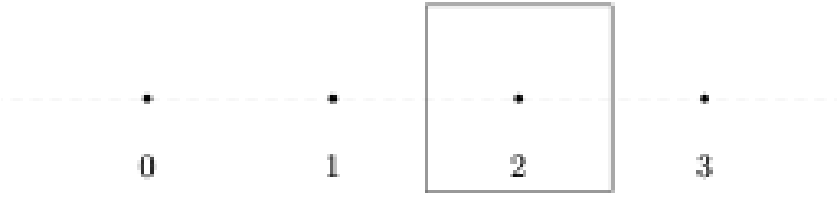
$$\boldsymbol{F} = \sum_{\alpha=1}^{N_v} 2\boldsymbol{c}_\alpha f_\alpha(\boldsymbol{x}_{\text{fluid}}, t)$$

Sketch of the *quantum* momentum exchange method

Change the amplitude-based data encoding

$$f_\alpha(\boldsymbol{x}, t) \;\rightarrow\; \sqrt{f_\alpha(\boldsymbol{x}, t)}$$

Example: D1Q3

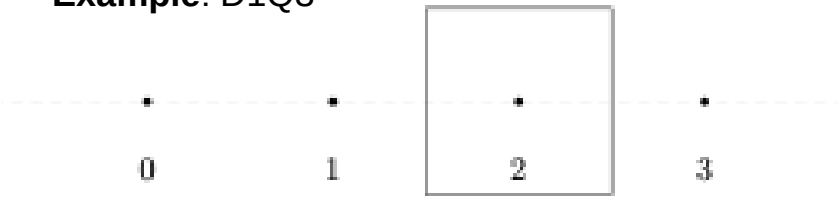


$$|\psi\rangle = \frac{1}{\sqrt{\sum_{x,\alpha} f_\alpha(x,t)}} \sum_{x,\alpha} \sqrt{f_\alpha(x,t)} |x_2 x_1 \; v_{\mathrm{dir}} v_1\rangle$$

$$|\psi\rangle = \frac{1}{\sqrt{\sum_{x,\alpha} f_\alpha(x,t)}} \begin{bmatrix} 0 \\ \sqrt{f_0(0,t)} \\ \sqrt{f_1(0,t)} \\ \sqrt{f_2(0,t)} \\ 0 \\ \sqrt{f_0(1,t)} \\ \sqrt{f_1(1,t)} \\ \sqrt{f_2(1,t)} \\ 0 \\ \sqrt{f_0(2,t)} \\ \sqrt{f_1(2,t)} \\ \sqrt{f_2(2,t)} \\ 0 \\ \sqrt{f_0(3,t)} \\ \sqrt{f_1(3,t)} \\ \sqrt{f_2(3,t)} \end{bmatrix}$$

Sketch of the *quantum* momentum exchange method

Example: D1Q3



$$\langle\psi|?\,|\psi\rangle = \frac{2f_1(1,t)}{\sqrt{\sum_{x,\alpha} f_\alpha(x,t)}}$$

$$|\psi\rangle = \frac{1}{\sqrt{\sum_{x,\alpha} f_\alpha(x,t)}} \begin{bmatrix} 0 \\ \sqrt{f_0(0,t)} \\ \sqrt{f_1(0,t)} \\ \sqrt{f_2(0,t)} \\ 0 \\ \sqrt{f_0(1,t)} \\ \sqrt{f_1(1,t)} \\ \sqrt{f_2(1,t)} \\ 0 \\ \sqrt{f_0(2,t)} \\ \sqrt{f_1(2,t)} \\ \sqrt{f_2(2,t)} \\ 0 \\ \sqrt{f_0(3,t)} \\ \sqrt{f_1(3,t)} \\ \sqrt{f_2(3,t)} \end{bmatrix}$$

Observable

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & B & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Intermediate summary

Quantum-LBM has the potential to overcome two of the grand challenges of CFD – high memory consumption and long compute times – by using superposition of states and quantum parallelism

- amplitude-based encoding of LBM data with $\#\text{qubits} = 4d - 2 + n_x + n_y + n_z + 3n_v$
- quantum primitives for streaming, specular reflection/bounce back BC treatment, object detection
- efficient read-out of body forces via quantum momentum exchange method

WAIT! What about collision?

We showed that amplitude-based encoding cannot work for the collision operator and computational basis state encoding cannot work for the streaming operator.

Sketch of the proof – part 1

- Consider two admissible states in **amplitude encoding**

$$|\psi_1\rangle = |x\rangle(\alpha_0|v_0\rangle + \alpha_1|v_1\rangle) \quad \text{and} \quad |\psi_2\rangle = |x\rangle|v_2\rangle$$

- Let U_{col} be a hypothetical collision operator then

$$|\psi_1{}'\rangle = I \otimes U_{\text{col}}|\psi_1\rangle = |x\rangle \otimes U_{\text{col}}(\alpha_0|v_0\rangle + \alpha_1|v_1\rangle) = |x\rangle\big(\gamma_0(\alpha_0|v_0\rangle + \alpha_1|v_1\rangle) + \gamma_1(\beta_2|v_2\rangle + \beta_3|v_3\rangle)\big)$$

$$|\psi_2{}'\rangle = I \otimes U_{\text{col}}|\psi_2\rangle = |x\rangle \otimes U_{\text{col}}|v_2\rangle = \mathrm{e}^{i\theta}|x\rangle|v_2\rangle$$

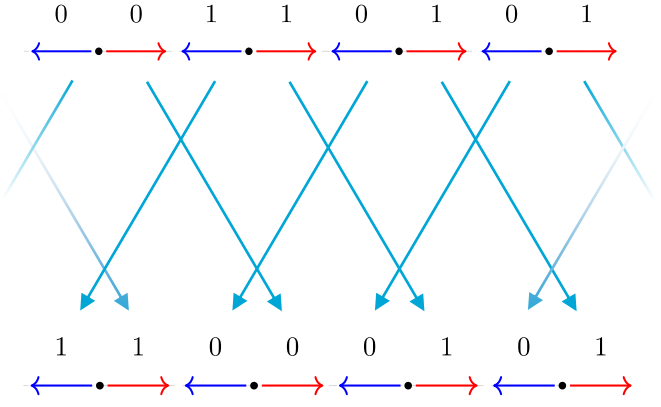
- For U_{col} to be unitary we must have that $\langle\psi_1|\psi_2\rangle = \langle\psi_1|U_{\text{col}}^{\dagger}U_{\text{col}}|\psi_2\rangle$. However

$$0 = \langle\psi_1|\psi_2\rangle = \cdots = e^{i\theta}\langle\gamma_0(\alpha_0\langle v_0| + \alpha_1\langle v_1|) + \gamma_1(\beta_2\langle v_2| + \beta_3\langle v_3|)|\langle x||x\rangle|v_2\rangle = e^{i\theta}\gamma_1\beta_2$$

Sketch of the proof – part 2

- Encoding of velocities in the **computational basis**

$$|\psi_1\rangle = \frac{1}{2}(|00\rangle|00\rangle + |01\rangle|11\rangle + |10\rangle|01\rangle + |11\rangle|01\rangle)$$



- After streaming

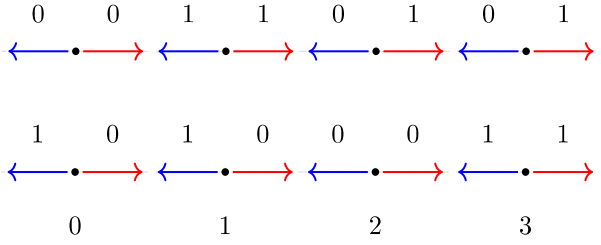
$$|\psi_1'\rangle = \frac{1}{2}(|00\rangle|11\rangle + |01\rangle|00\rangle + |10\rangle|10\rangle + |11\rangle|10\rangle)$$

Sketch of the proof – part 2

- Encoding of velocities in the **computational basis**

$$|\psi_1\rangle = \tfrac{1}{2}(|00\rangle|00\rangle + |01\rangle|11\rangle + |10\rangle|01\rangle + |11\rangle|01\rangle)$$

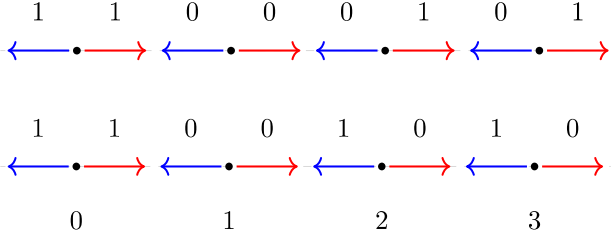
$$|\psi_2\rangle = \tfrac{1}{2}(|00\rangle|10\rangle + |01\rangle|10\rangle + |10\rangle|00\rangle + |11\rangle|11\rangle)$$



- After streaming

$$|\psi_1'\rangle = \tfrac{1}{2}(|00\rangle|11\rangle + |01\rangle|00\rangle + |10\rangle|10\rangle + |11\rangle|10\rangle)$$

$$|\psi_2'\rangle = \tfrac{1}{2}(|00\rangle|11\rangle + |01\rangle|00\rangle + |10\rangle|01\rangle + |11\rangle|01\rangle)$$



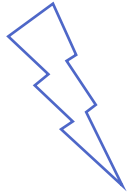
Sketch of the proof – part 2

- Encoding of velocities in the **computational basis**

$$|\psi_1\rangle = \tfrac{1}{2}(|00\rangle|00\rangle + |01\rangle|11\rangle + |10\rangle|01\rangle + |11\rangle|01\rangle)$$

$$|\psi_2\rangle = \tfrac{1}{2}(|00\rangle|10\rangle + |01\rangle|10\rangle + |10\rangle|00\rangle + |11\rangle|11\rangle)$$

$$\langle\psi_1|\psi_2\rangle = 0$$



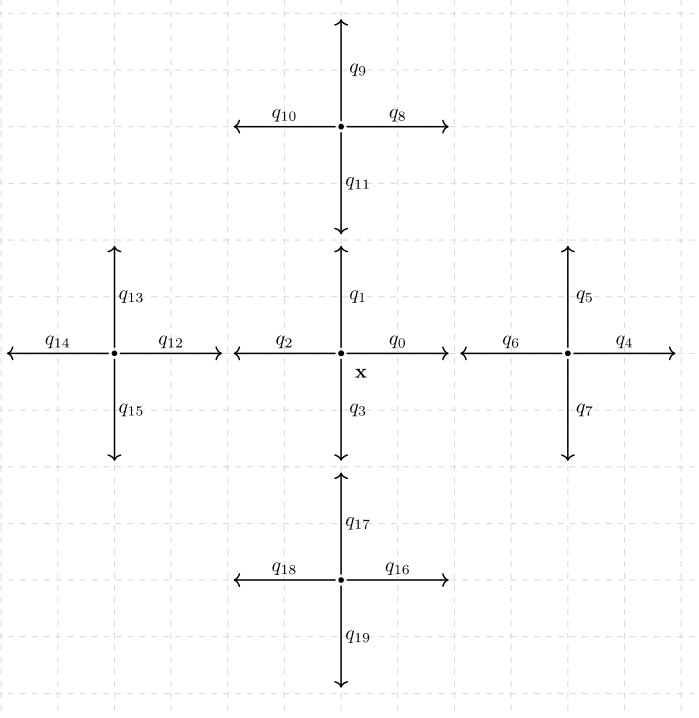
- After streaming

$$|\psi_1'\rangle = \tfrac{1}{2}(|00\rangle|11\rangle + |01\rangle|00\rangle + |10\rangle|10\rangle + |11\rangle|10\rangle)$$

$$|\psi_2'\rangle = \tfrac{1}{2}(|00\rangle|11\rangle + |01\rangle|00\rangle + |10\rangle|01\rangle + |11\rangle|01\rangle)$$

$$\langle\psi_1'|\psi_2'\rangle = \tfrac{1}{2}$$

Space-time encoding



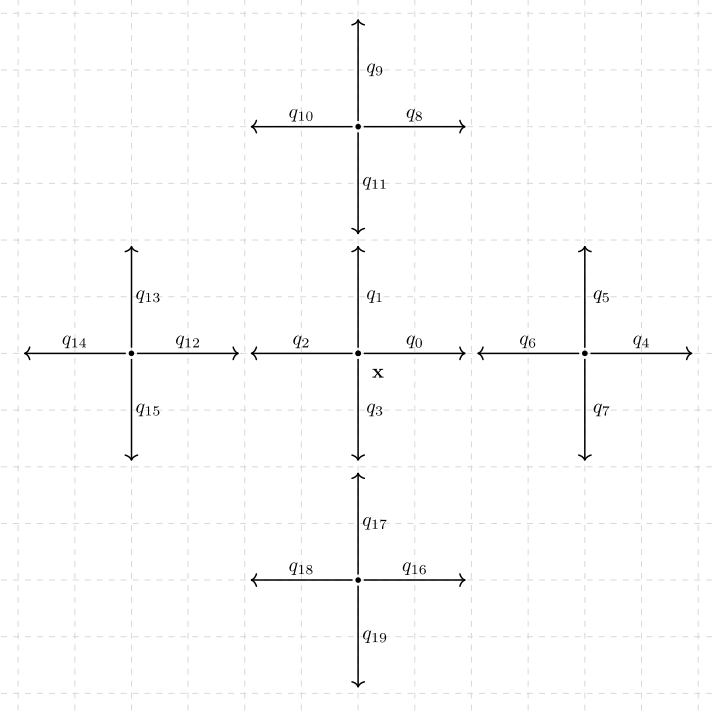
- **Idea**: Basis encoding taking into account "all" lattices from which particles can reach $\boldsymbol{x}$ within N_t time steps

$$|x\rangle|q_0 \dots q_{19}\rangle$$

- Number of qubits for N_t time steps of D2Q4

$$n_v = 4 + \sum_{i=1}^{N_t} 16i = 8N_t^2 + 8N_t + 4$$

Space-time encoding



- **Pros**
  - streaming *and* collision as unitary operators
  - streaming is just a sequence of swaps with less and less swaps for later time steps
  - collision operator can be constructed by inspecting equivalent classes
- **Cons**
  - Resource requirements (#qubits and #swaps) grow with N_t

Further reading

- Merel A. Schalkers and Matthias Möller. Efficient and fail-safe quantum algorithm for the transport equation. Journal of Computational Physics, 502:112816, April 2024. DOI: 10.1016/j.jcp.2024.112816
- Merel A. Schalkers and Matthias Möller. On the importance of data encoding in quantum Boltzmann methods. Quantum Information Processing, 23(1), January 2024. DOI: 10.1007/s11128-023-04216-6
- Merel A. Schalkers and Matthias Möller. Momentum exchange method for quantum Boltzmann methods. Computers & Fluids, 285:106453, December 2024. DOI: 10.1016/j.compfluid.2024.106453
- Calin A. Georgescu, Merel A. Schalkers, and Matthias Möller. QLBM - a quantum lattice Boltzmann software framework, 2024. arXiv:2411.19439
- https://github.com/qcfd-lab/qlbm

TU Delft FUJITSU

Save the date

- CISM Advanced Course: Opportunities and Challenges of Quantum Computing in Computational Mechanics, May 5-9, 2025, Udine
- 1st ECCOMAS Thematic Conference Applied Quantum Methods in Computational Science and Engineering (AQMCSE), Oct 8-10, 2025, Aachen

TU Delft

FUJITSU