Bloqade
A Compiler and SDK for Neutral Atom Quantum Computers

Phillip Weinberg
Senior Scientific Software Engineer
pweinberg@quera.com

OCT 20 2025
MUNICH QUANTUM SOFTWARE FORUM

Outline

QuEra

QuEra's next generation SDK

The next-gen SDK

Two new open source **hardware-oriented** DSLs + compiler

Embedded in **Python** built with **inter-operability** with other major circuit SDKs in mind.

powered by QuEra's in-house compiler infrastructure.

Bloqade Circuit → Bloqade Shuttle → QuEra's Gate-based Hardware

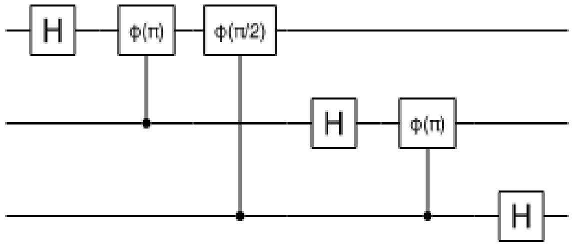
Squin

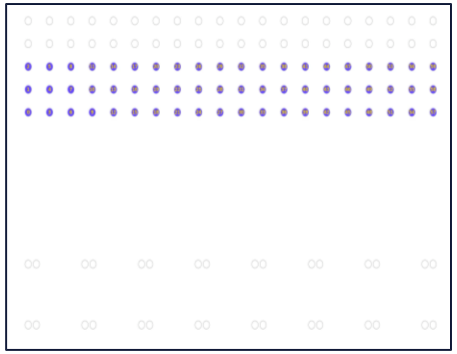
Circuit eDSL

Integration

QASM2, STIM, Cirq, ...

Atom Shuttle Move Scheduling





built with: Kirin

Compiler toolchain

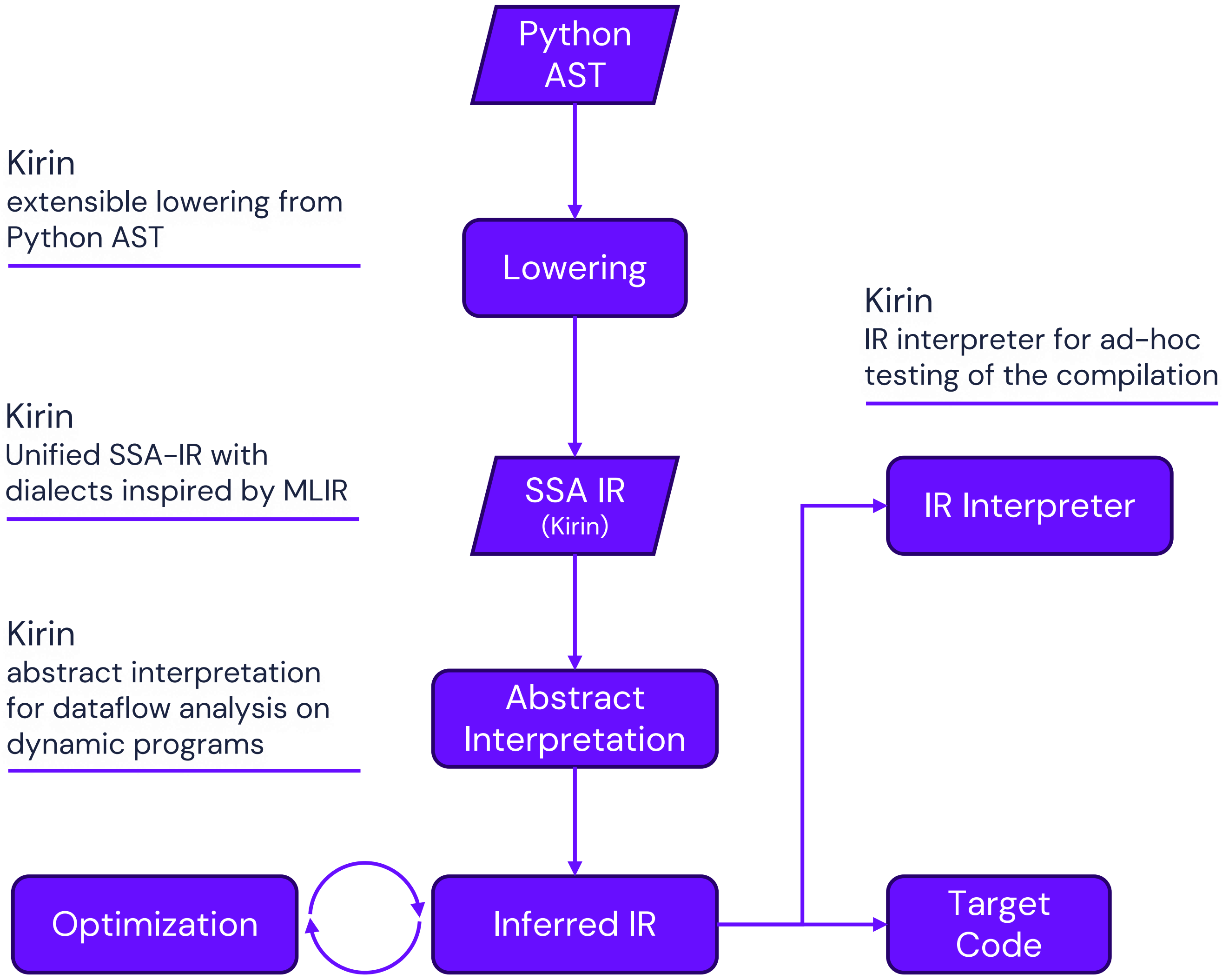
QuEra Putting Quantum to Work

Kirin alpha
compiler infrastructure

Scientist First
This is a tool built by scientists for scientists

Focused Scope
Kernel-style functions in Python as backbone of computational workflows

Composability
eDSLs written in Kirin can be composed with each other



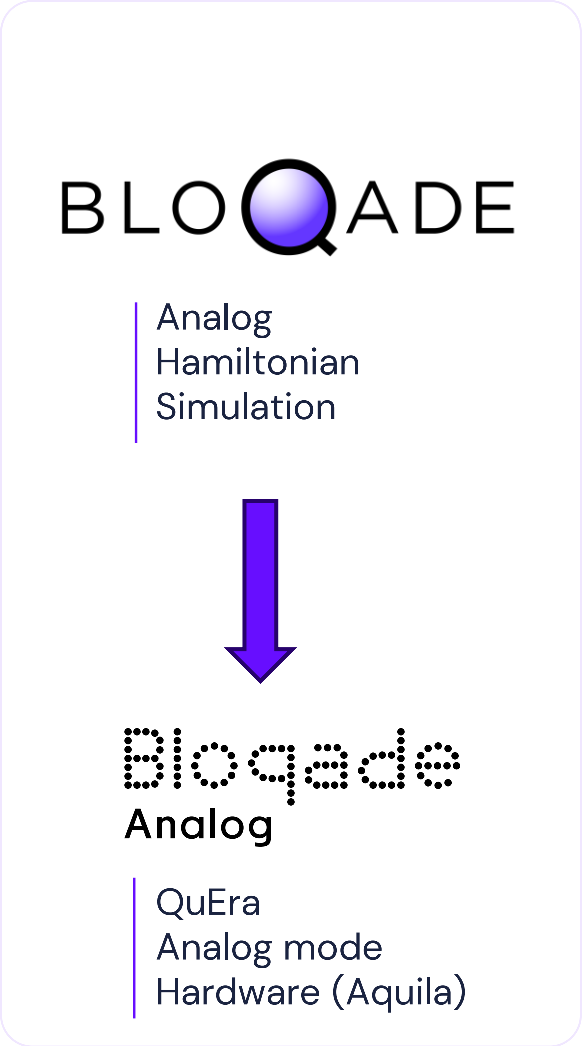
QuEra Putting Quantum to Work

House-keeping

Update on previous project: Bloqade-python

Continued long term support

Released in 2023, this SDK is renamed into Bloqade analog. Incorporated into a new Bloqade package installed along side new open source projects.



Bloqade-circuit

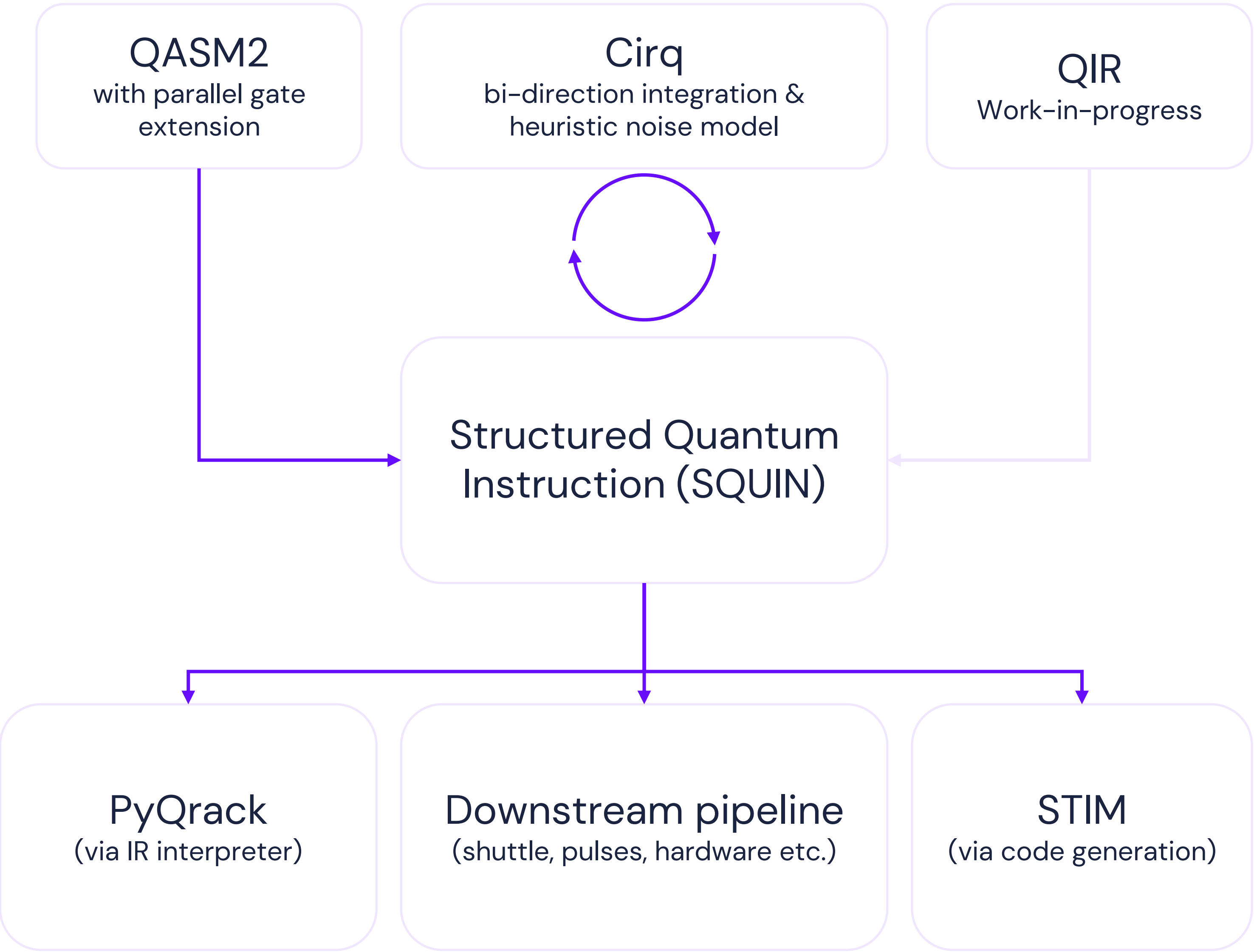
A circuit DSL embedded in Python

Bloqade Circuit alpha
Neutral-atom specific circuit

Neutral-atom Specific
Parallel gates, circuit-level noise models, and more

Integration
Integration with Cirq, QASM2, and other exchange formats in the works

Composable with low-level language
Designed to be composable with our low-level hardware language



SQUIN alpha

A circuit DSL embedded in Python

Embedded in Python

The SQUIN kernel embeds quantum instruction and a subset of Python language as regular Python code

Structured

Structured quantum programs allow high-level abstractions such as control flow, loops, functions and closures.

```python
from bloqade import squin

@squin.kernel
def ghz_linear(n: int):
    q = squin.qubit.new(n)
    squin.gate.h(q[0])
    for i in range(1, n):
        squin.gate.cx(q[0], q[i])

# Print the lowered representation of the kernel
ghz_linear.print()
```

Lowered

```
func.func main(!py.int) -> !py.NoneType {
  ^0(%main_self, %n):
  │          %q = squin.qubit.new(n_qubits=%n : !py.int) : !py.IList[!py.Qubit, !Any]
  │          %0 = py.constant.constant 0 : !py.int
  │          %1 = py.indexing.getitem(%q, %0) : !py.Qubit
  │          %2 = func.invoke h(%1) : !py.NoneType maybe_pure=False
  │          %3 = py.constant.constant 1 : !py.int
  │          %4 = py.len.len(value=%q) : !py.int
  │          %5 = py.constant.constant 1 : !py.int
  │          %6 = py.ilist.range(start=%3, stop=%4, step=%5) : !py.IList[!py.int, !Any]
  │  %q_1, %q_2 = scf.for %i : !py.int in %6 -> !py.IList[!py.Qubit, !Any], !py.IList[!py.Qubit, !Any]
  │               │ iter_args(%q_3 : !py.IList[!py.Qubit, !Any] = %q, %q_4 : !py.IList[!py.Qubit, !Any] = %q) {
  │               │  %8 = py.constant.constant 1 : !py.int
  │               │  %9 = py.binop.sub(%i : !py.int, %8) : !py.int
  │               │ %10 = py.indexing.getitem(%q_3 : !py.IList[!py.Qubit, !Any], %9) : !py.Qubit
  │               │ %11 = py.indexing.getitem(%q_3 : !py.IList[!py.Qubit, !Any], %i : !py.int) : !py.Qubit
  │               │ %12 = func.invoke cx(%10, %11) : !py.NoneType maybe_pure=False
  │               │       scf.yield %q_3 : !py.IList[!py.Qubit, !Any], %q_3 : !py.IList[!py.Qubit, !Any]
  │               } -> purity=False
  │          %7 = func.const.none() : !py.NoneType
  │               func.return %7
} // func.func main
```

SQUIN alpha

A circuit DSL for neutral atoms

Hardware specific

As a hardware vendor, SQUIN is a circuit-level IR owned by QuEra to provide hardware specific instructions at the circuit level.

First-class parallel gates

SQUIN supports first-class parallel gates that are native to our hardware.

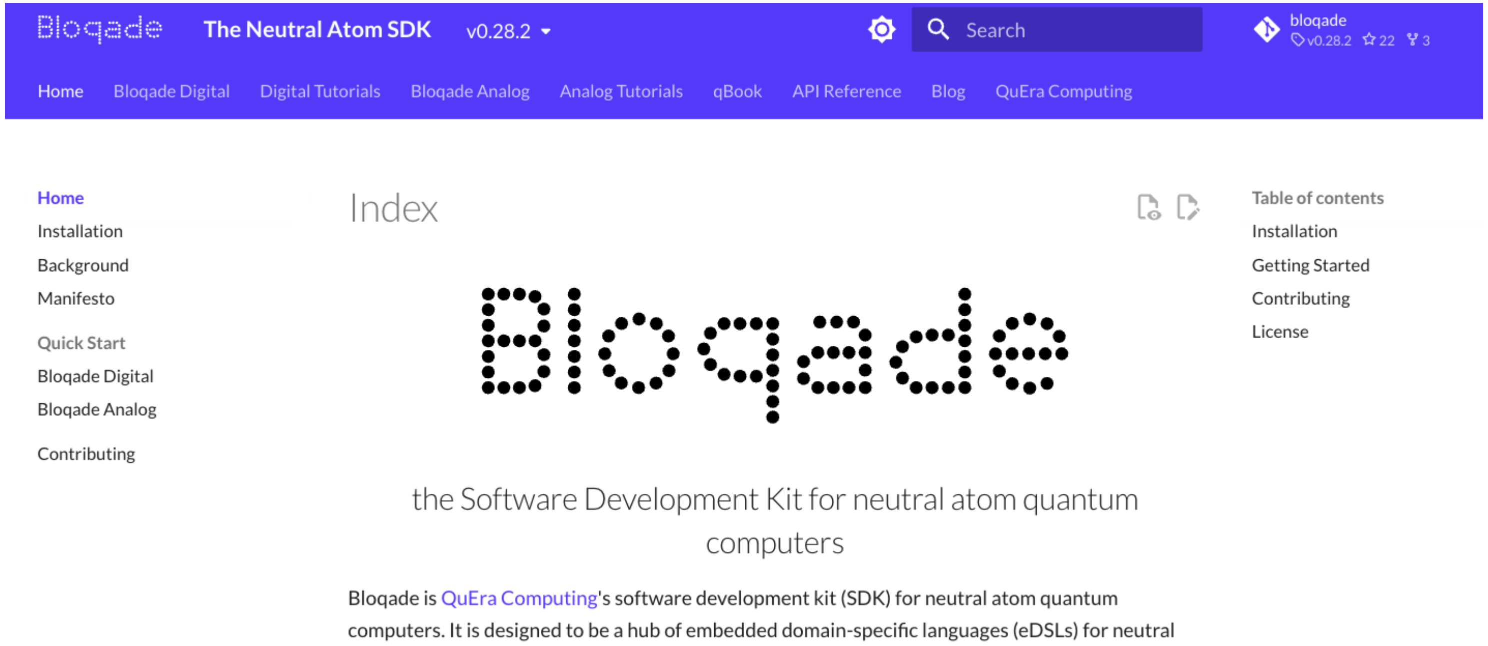
Hardware focused noise modeling

We have heuristics for estimating noise at the circuit level

```
from bloqade import squin

@squin.kernel
def noisy_circuit(n: int, p_single: float, p_paired: float):
    q = squin.qubit.new(n)
    single_qubit_noise = squin.noise.depolarize(p_single)
    squin.gate.h(q[0])
    # first-class parallel gate
    squin.qubit.broadcast(squin.op.x(), q[1:])
    squin.qubit.apply(single_qubit_noise, q[0])

    # pair qubit noise operator
    two_qubit_noise = squin.noise.depolarize2(p_paired)
    for i in range(1, n):
        squin.gate.cx(q[0], q[i])
        squin.qubit.apply(two_qubit_noise, q[i - 1], q[i])

    return squin.qubit.measure(q)

noisy_circuit.print()
```

Documentation

Where can I read about neutral atoms?



Bloqade documentation – bloqade.quera.com

QuEra Putting Quantum to Work

Roadmap (2025 - 26)

Bloqade Circuit alpha

Simplifying IRs

Refactoring existing IR definitions (#492) to have explicit gate operations.

Better emulation capabilities

Integration with CUDA-Q & QIR, and improving STIM code generation

Improving Compiler Stability

Simplifying compiler passes with #492, restricting more program behavior

```python
from bloqade import squin

@squin.kernel
def noisy_circuit(n: int, p_single: float, p_paired: float):
    q = squin.qubit.new(n)
    squin.gate.h(q[0])
    # first-class parallel gate
    squin.gate.broadcast.x(q[1:])
    squin.noise.depolarize(q[0], p_single)

    for i in range(1, n):
        squin.gate.cx(q[0], q[i])
        # pair qubit noise channel
        squin.noise.depolarize2(q[i-1], q[i], p_paired)
    return squin.qubit.measure(q)

noisy_circuit.print()
```

Simplifying SQUIN with first-class gate instruction

Bloqade-shuttle

An atom shuttling DSL

Bloqade Shuttle alpha

An atom shuttling DSL

Tweezer

A DSL for optical tweezers, defines a move semantic on a given architecture.

Grid

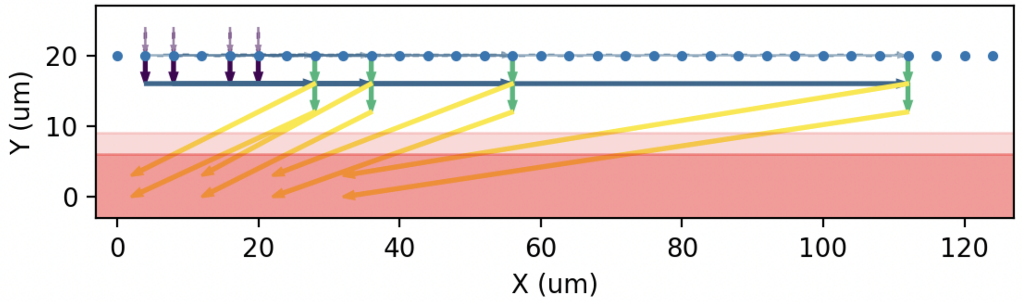
Special value and with a small DSL used to define positions in a grid. Supports slicing, indexing, shift, scale, etc.

Tweezer visualization

Tools to visualize how tweezer would execute kernel code, evaluating move “cost”.

QuEra Putting Quantum to Work

```python
@tweezer
def entangle_move(
    mem_zone: grid.Grid[Any, Literal[1]],
    gate_zone: grid.Grid[NMove, Literal[2]],
    ctrl_ids: ilist.IList[int, NMove],
    qarg_ids: ilist.IList[int, NMove],
    gate_ids: ilist.IList[int, NMove],
):

    mem_y = grid.get_ypos(mem_zone)[0]
    ctrl_start = grid.get_xpos(mem_zone[ctrl_ids, 0])
    qarg_start = grid.get_xpos(mem_zone[qarg_ids, 0])

    pos_1 = grid.from_positions(ctrl_start, [mem_y, mem_y + 4.0])
    pos_2 = grid.shift(pos_1, 0.0, -4.0)
    pos_3 = grid.from_positions(qarg_start, grid.get_ypos(pos_2))
    pos_4 = grid.shift(pos_3, 0.0, -4.0)
    gate_pos =gate_zone[gate_ids, :]

    action.set_loc(pos_1)
    action.turn_on(action.ALL, [0])
    action.move(pos_2)
    action.move(pos_3)
    action.turn_on([], [1])
    action.move(pos_4)
    action.move(gate_pos)
```



Bloqade Shuttle alpha

An atom shuttling DSL

Move scheduling

Allow users or compiler developers to describe the scheduling of atom moves and gate operations on hardware resources

Layout Specification

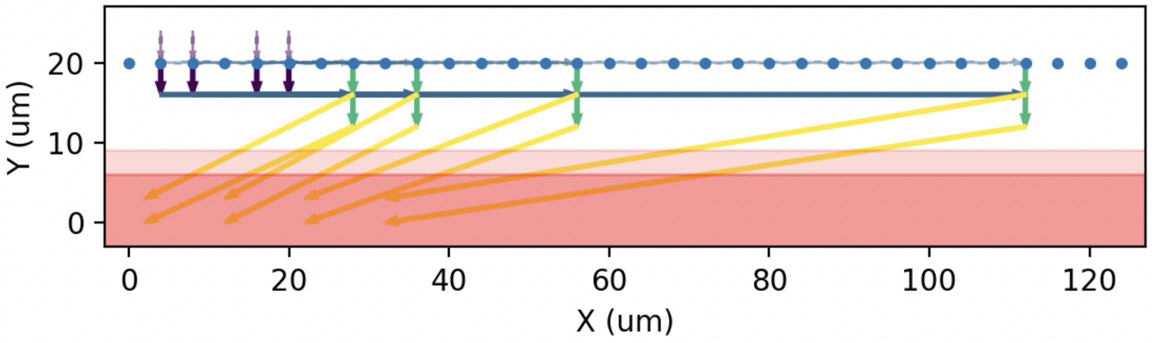
Define static traps along with special locations with pre-defined names. Access spec values by name in program.

Validation

Gates happen in allowed locations, etc.

QuEra Putting Quantum to Work

```python
@move
def run_entangle_move(
    mem_zone: grid.Grid[Any, Literal[1]],
    gate_zone: grid.Grid[Any, Literal[2]],
    ctrl_ids: ilist.IList[int, NMove],
    qarg_ids: ilist.IList[int, NMove],
    gate_ids: ilist.IList[int, NMove],
):

    num = len(ctrl_ids)
    xtones = ilist.range(num)
    ytones = [0, 1]

    dtask = schedule.device_fn(entangle_move, xtones, ytones)
    rev_dtask = schedule.reverse(dtask)

    gate.local_r(0.0, math.pi, mem_zone[qarg_ids, 0])
    dtask(mem_zone, gate_zone, ctrl_ids, qarg_ids, gate_ids)
    gate.top_hat_cz(gate_zone)
    rev_dtask(mem_zone, gate_zone, ctrl_ids, qarg_ids, gate_ids)
    gate.local_r(0.0, -math.pi, mem_zone[qarg_ids, 0])
```



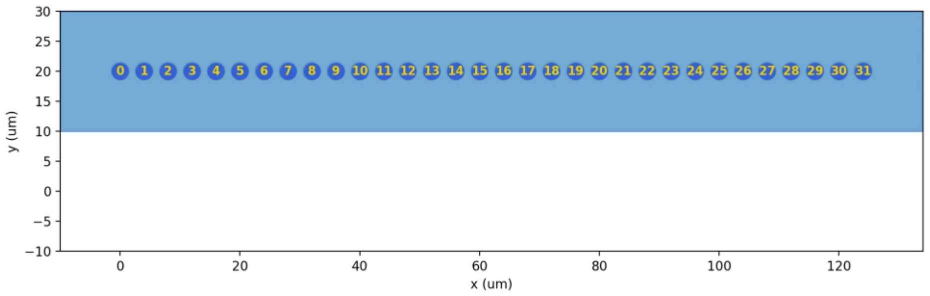
Roadmap (2025 - 26)

Bloqade Shuttle alpha

Pulses & Accurate Noise

Enables lowering of moves or more accurate noise models, and animation, and virtual execution of program

```python
@move
def run_entangle_move(
    mem_zone: grid.Grid[Any, Literal[1]],
    gate_zone: grid.Grid[Any, Literal[2]],
    ctrl_ids: ilist.IList[int, NMove],
    qarg_ids: ilist.IList[int, NMove],
    gate_ids: ilist.IList[int, NMove],
):

    num = len(ctrl_ids)
    xtones = ilist.range(num)
    ytones = [0, 1]

    dtask = schedule.device_fn(entangle_move, xtones, ytones)
    rev_dtask = schedule.reverse(dtask)

    gate.local_r(0.0, math.pi, mem_zone[qarg_ids, 0])
    dtask(mem_zone, gate_zone, ctrl_ids, qarg_ids, gate_ids)
    gate.top_hat_cz(gate_zone)
    rev_dtask(mem_zone, gate_zone, ctrl_ids, qarg_ids, gate_ids)
    gate.local_r(0.0, -math.pi, mem_zone[qarg_ids, 0])
```



SQUIN
lowering from circuit representation

Animation
Generates animation for scheduled atom moves

Signal
lowering to hardware execution signals

Roadmap (2025 - 26)

Bloqade Shuttle alpha

Converging IR

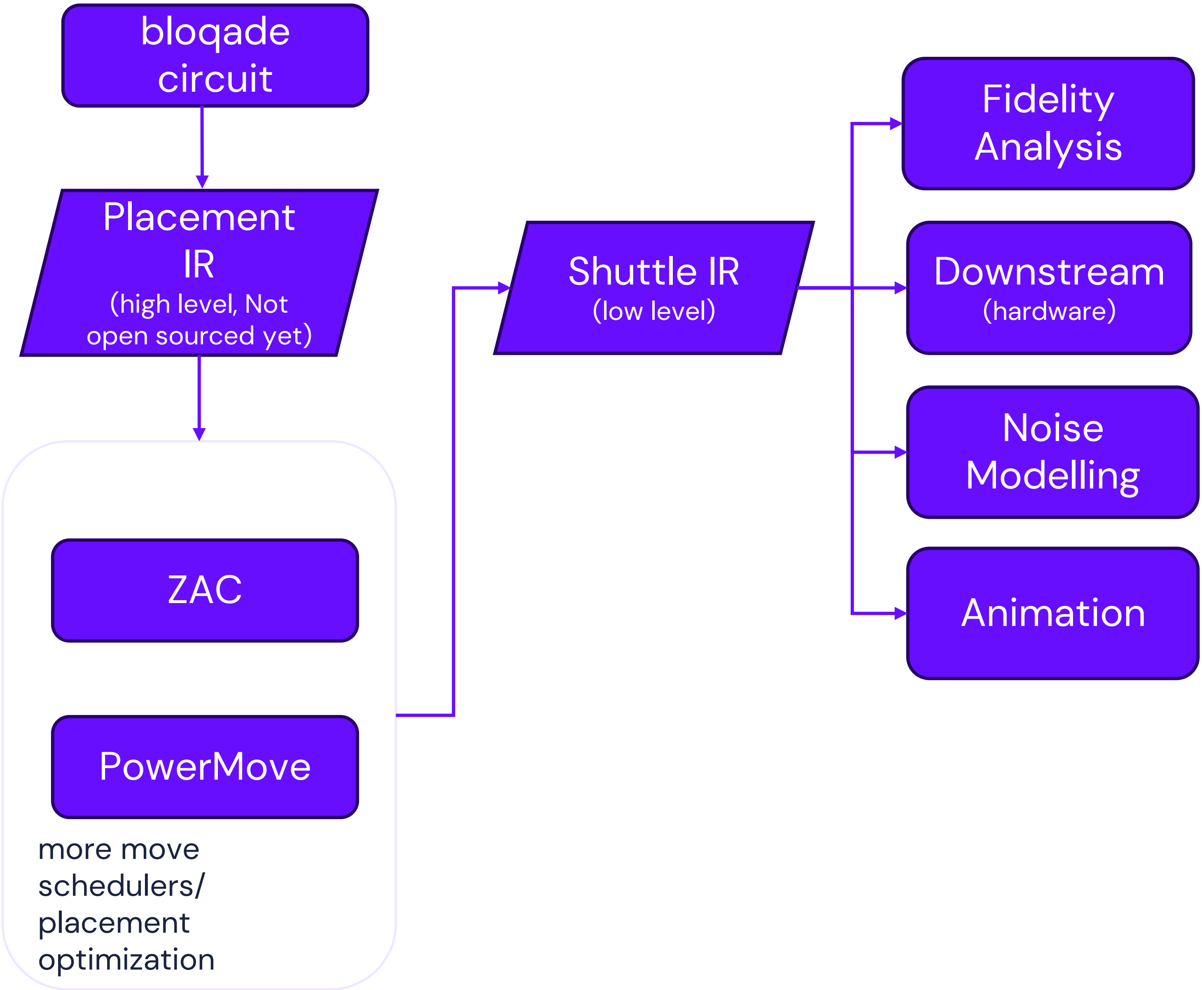
Converge on an IR that works for most atom-move compilers.

Automated move scheduling

More move scheduling algorithms to enable a rich ecosystem of move optimization

Richer toolchain

Emulation, animation and common tools like: fidelity ,tweezer position, and atom position analysis



How to partner with us

Bloqade Circuit

- **Users:** use SQUIN's hardware-oriented syntax to optimize circuit performance on QuEra hardware.
- **SDK developers:** Access QuEra's neutral atom quantum computers via SQUIN ,QASM2.0, or QIR.

Bloqade Shuttle

- **Neutral atom compiler developers:** Help develop set of common semantics move optimization
- **Neutral atom compiler developers:** target low-level instructions to execute customized move strategies.
- **Users:** Learn about Neutral atom quantum computing works. Write optimized applications

QuEra Putting Quantum to Work

Open source Contributions

Bloqade Circuit

Bloqade Shuttle

Kirin

- Join our monthly Bloqade community meeting. Registration:

- Direct contributions
  - Documentation edits
  - Feature requests
  - Bug reports
  - Pull requests

If Bloqade is useful for your project give us a Starred 23

QuEra Putting Quantum to Work

Stay tuned for additional open-source packages!

Thank you

Phillip Weinberg
Senior Scientific Software Engineer
pweinberg@quera.com