riverlane

How an open-source compiler can change your research

Adrien Suau – Senior Open-source Software Engineer

Questions welcomed at any time

Plan of the talk

000 Introduction

001 Compilers – What, why and how?

010 Abstraction levels in quantum computing

011 How to benefit from open-source in your work?

100 Conclusion

101 Spoiler

Plan of the talk

000 **Introduction**

001 Compilers – What, why and how?

010 Abstraction levels in quantum computing

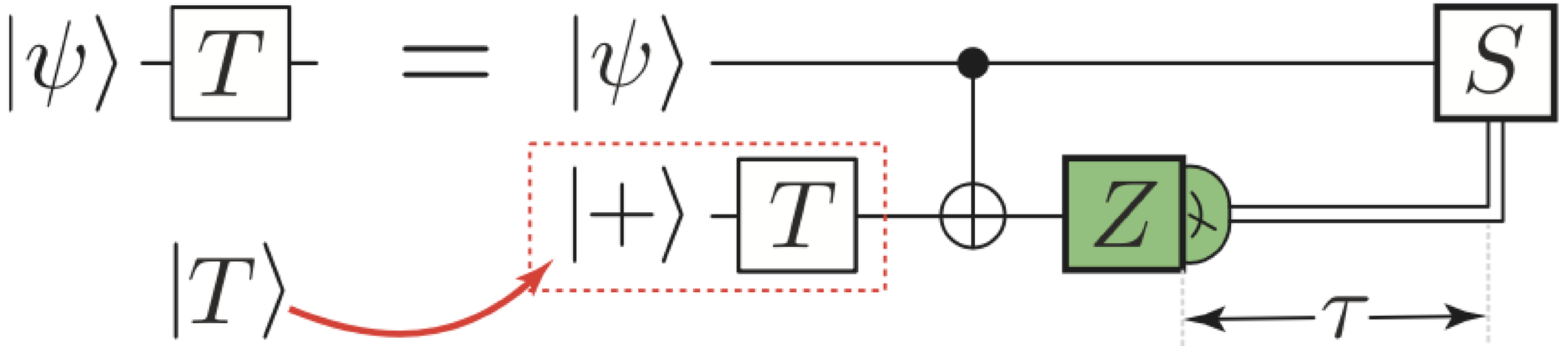
011 How to benefit from open-source in your work?

100 Conclusion

101 Spoiler

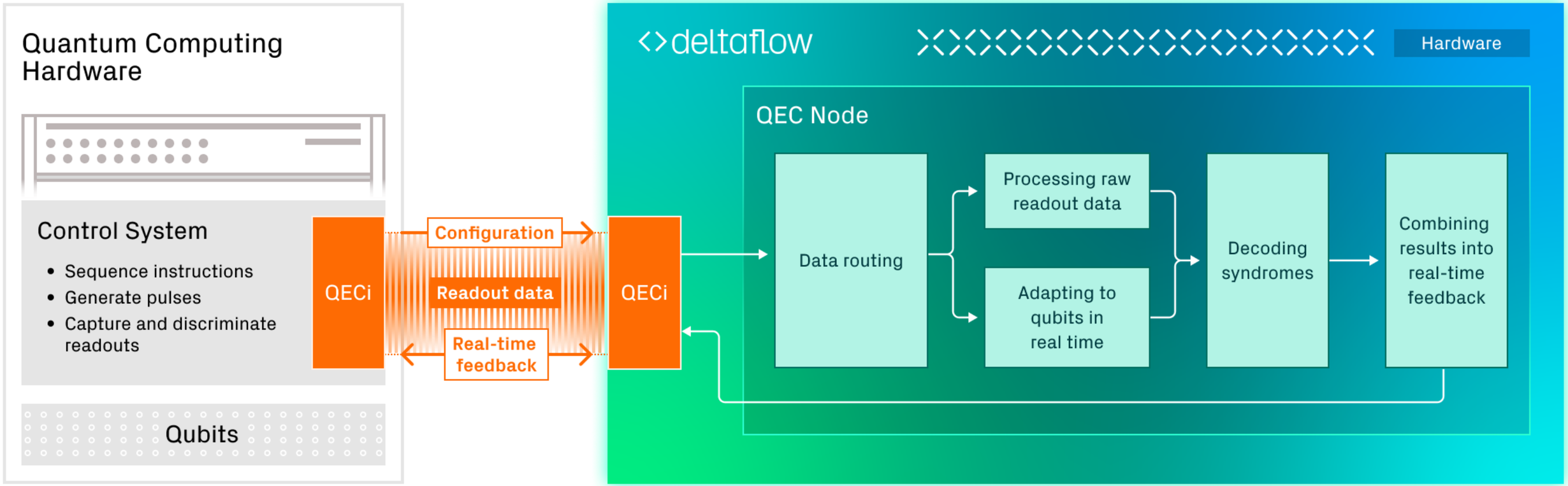
Introduction

Universal computation using QEC



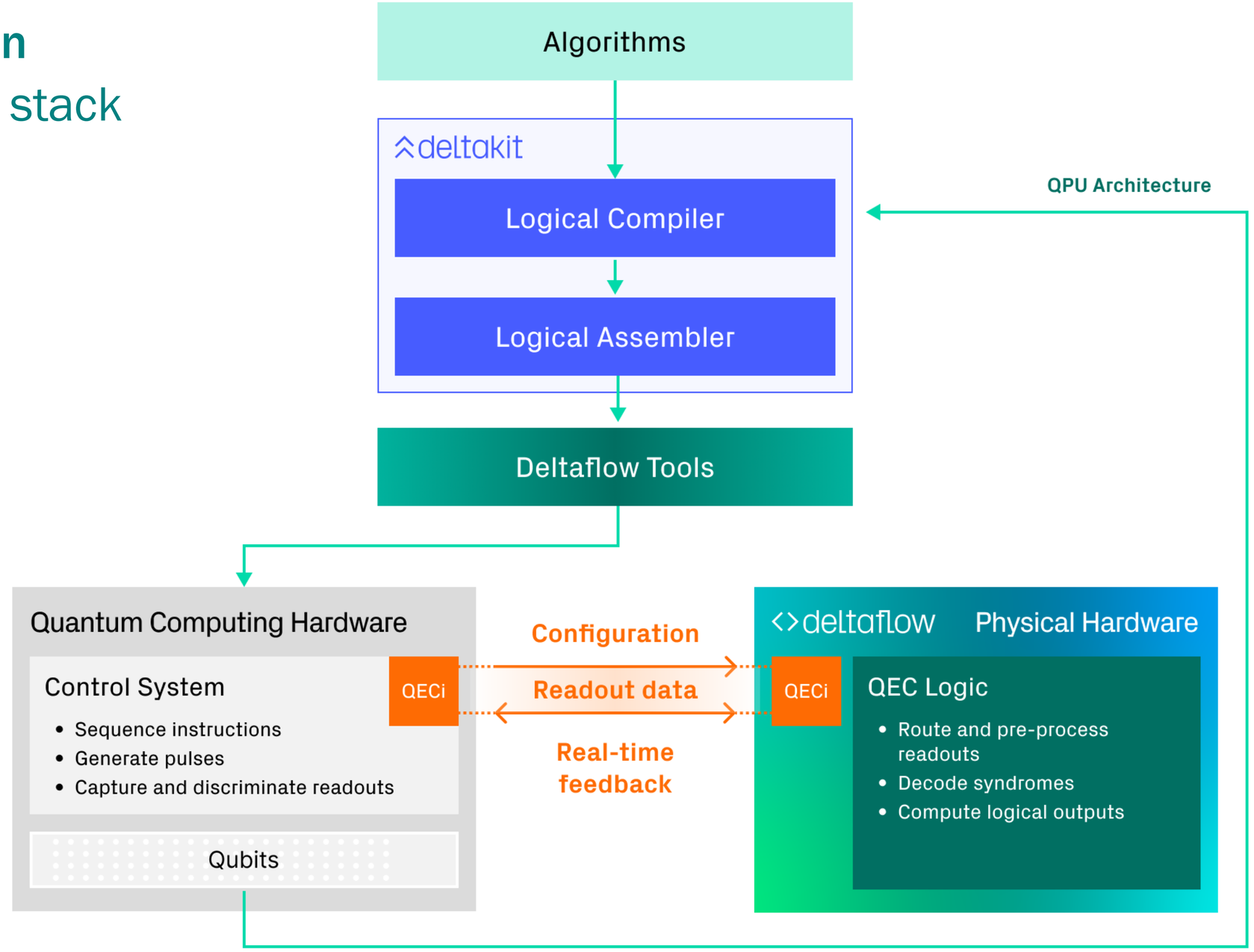
Introduction

Hardware QEC system



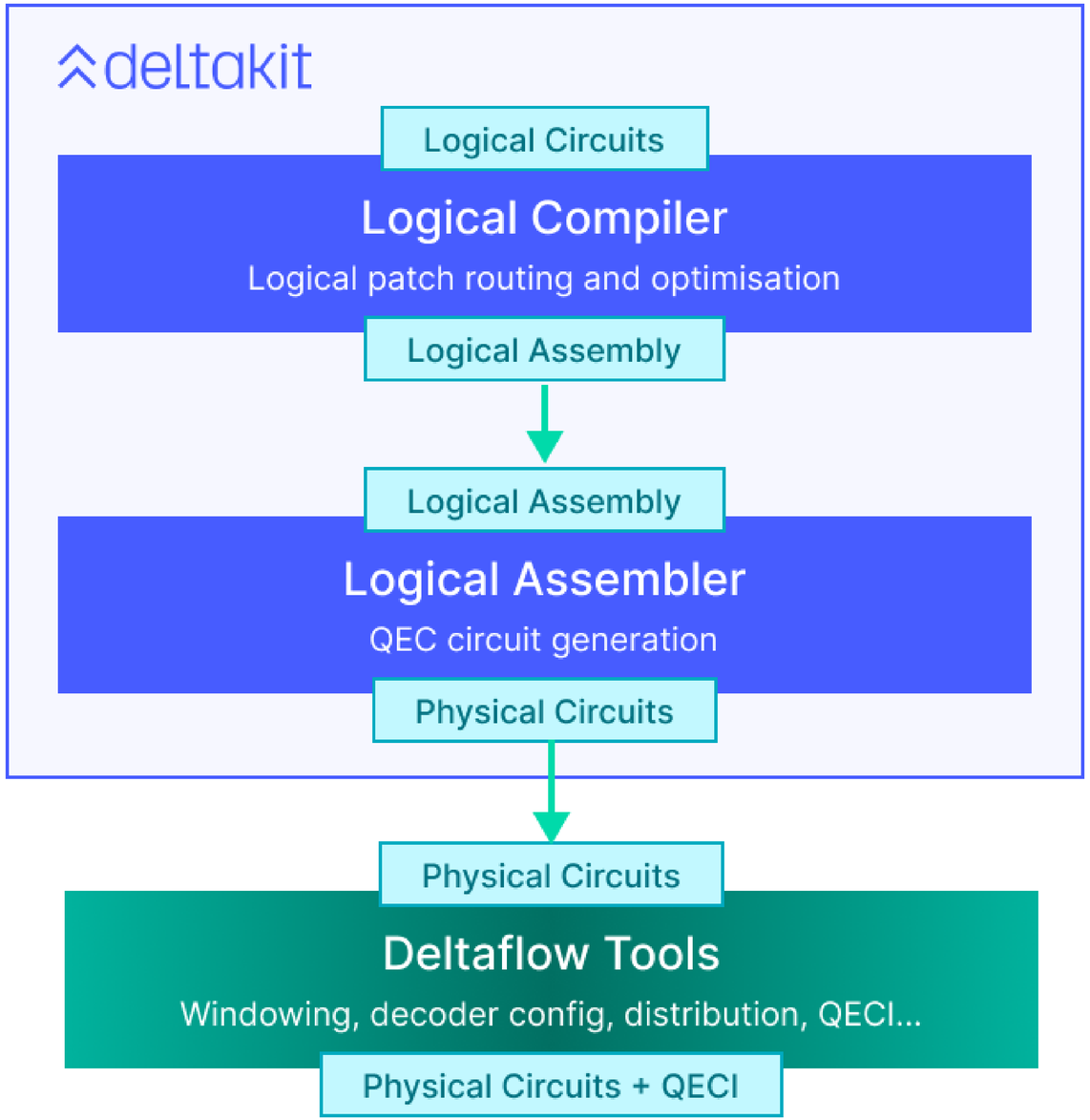
Introduction

Riverlane's stack



Introduction

Riverlane's software stack



Plan of the talk

000 Introduction

001 **Compilers – What, why and how?**

010 Abstraction levels in quantum computing

011 How to benefit from open-source in your work?

100 Conclusion

101 Spoiler

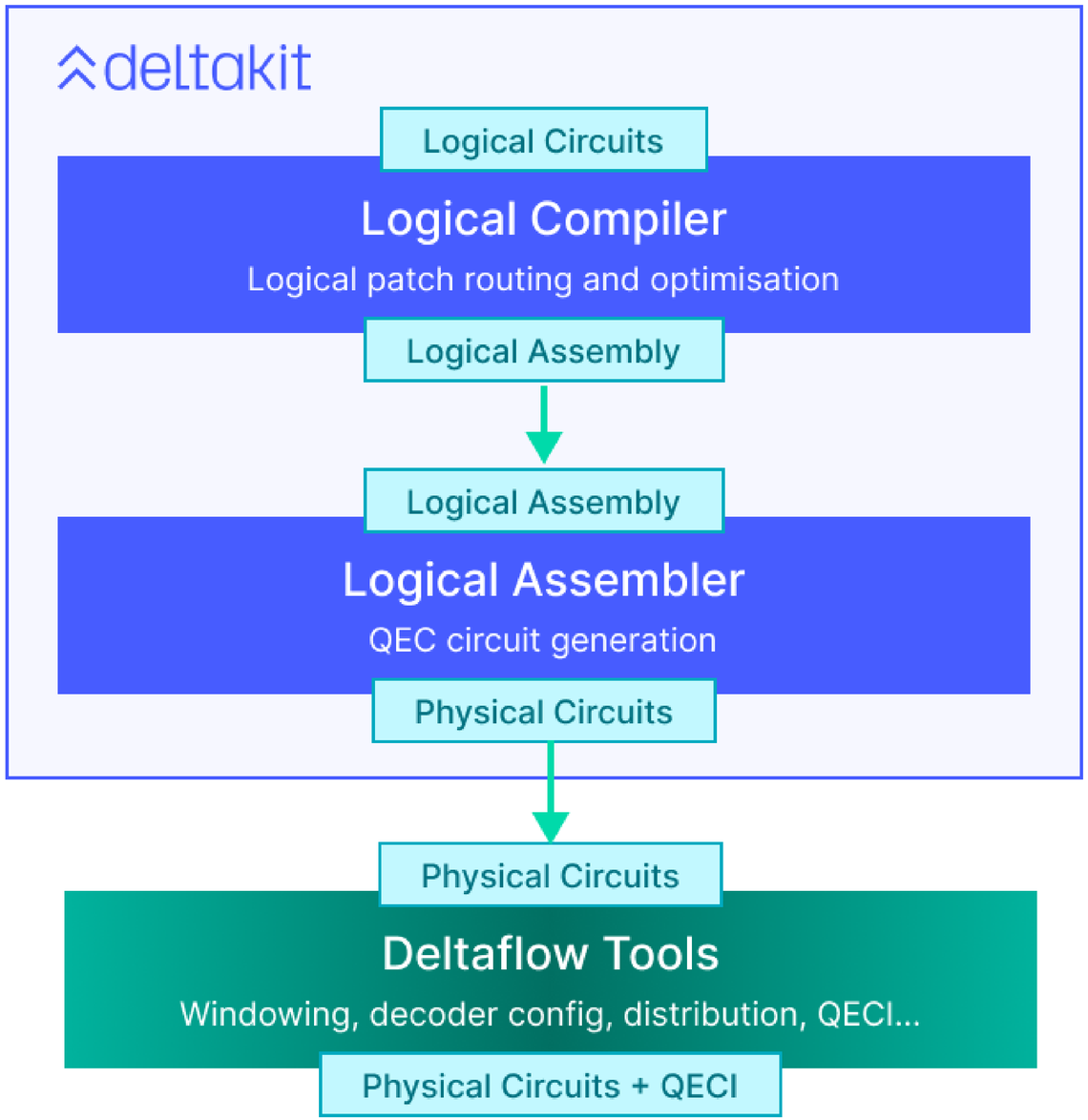
Compilers – What, why and how?

What is a compiler?

Data representation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation



Compilers – What, why and how?

What is a compiler?

Data representation(s)

Data transformation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation

Dialect:

Building blocks for an IR. Ideally small and mutually orthogonal.

Compilers – What, why and how?

What is a compiler?

Data representation(s)

Data transformation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation

Dialect:

Building blocks for an IR. Ideally small and mutually orthogonal.

Example:

A stim.Circuit instance is an IR that could be decomposed as:

1. gates: for quantum gates (Clifford)
2. struct: for REPEAT blocks
3. det: for detectors
4. obs: for observables

Compilers – What, why and how?

What is a compiler?

Data representation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation

Dialect:

Building blocks for an IR. Ideally small and mutually orthogonal.

Example:

A stim.Circuit instance is an IR that could be decomposed as:

1. gates: for quantum gates (Clifford)
2. struct: for REPEAT blocks
3. det: for detectors
4. obs: for observables

Data transformation(s)

Pass:

Code applied on an IR.

Can **change** an IR (optimisation, lowering), or **annotate** after analysis.

Compilers – What, why and how?

What is a compiler?

Data representation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation

Dialect:

Building blocks for an IR. Ideally small and mutually orthogonal.

Example:

A stim.Circuit instance is an IR that could be decomposed as:

1. gates: for quantum gates (Clifford)
2. struct: for REPEAT blocks
3. det: for detectors
4. obs: for observables

Data transformation(s)

Pass:

Code applied on an IR.

Can **change** an IR (optimisation, lowering), or **annotate** after analysis.

Pipeline:

A collection of passes in a specific order.

Compilers – What, why and how?

What is a compiler?

Data representation(s)

Intermediate Representation (IR):

Representations of the problem / algorithm / program / implementation at each point during the compilation

Dialect:

Building blocks for an IR. Ideally small and mutually orthogonal.

Example:

A stim.Circuit instance is an IR that could be decomposed as:

1. gates: for quantum gates (Clifford)
2. struct: for REPEAT blocks
3. det: for detectors
4. obs: for observables

Data transformation(s)

Pass:

Code applied on an IR.

Can **change** an IR (optimisation, lowering), or **annotate** after analysis.

Pipeline:

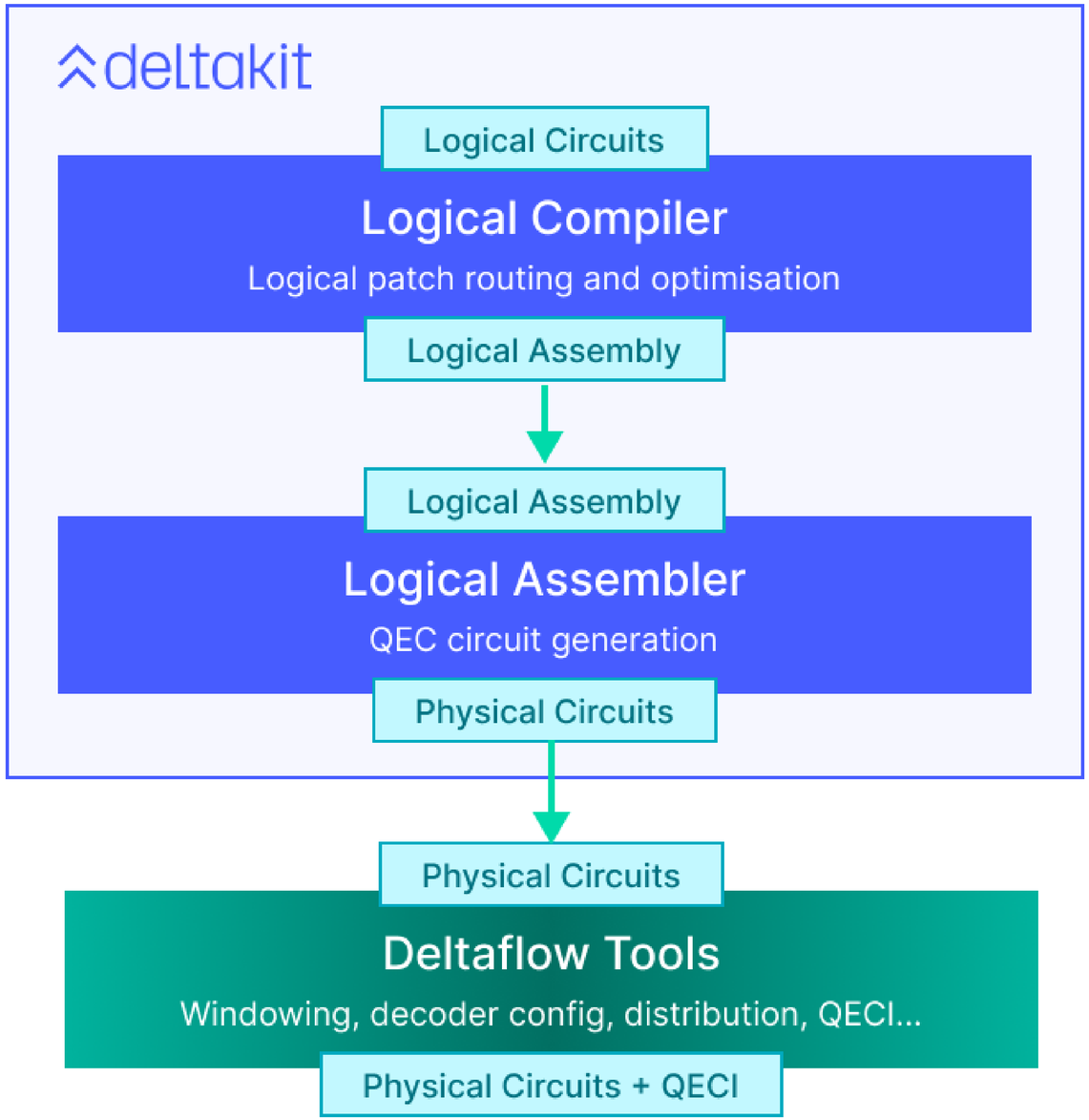
A collection of passes in a specific order.

Examples:

- A pass on det that would try to reduce the average number of measurements in each detector.
- A pass unrolling REPEAT blocks, removing the struct dialect from the IR.

Compilers – What, why and how?

What is a compiler?



Compilers – What, why and how?

Why using a compiler?

Separation between program representation and transformations.

Clear and testable invariants enforced on program representation at every stage.

Re-usability of the passes.

Compilers – What, why and how?

How to build a compiler?

Data representations

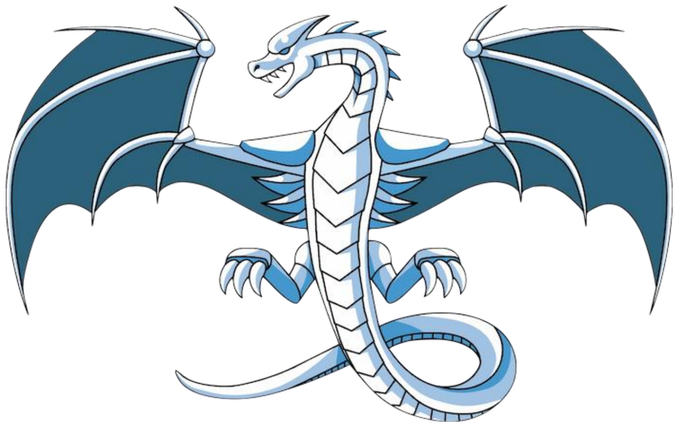
Dialects

Coherent IRs

Data transformation(s)

Pipelines applied on the different IRs

Passes composing the pipelines



Plan of the talk

000 Introduction

001 Compilers – What, why and how?

010 **Abstraction levels in quantum computing**

011 How to benefit from open-source in your work?

100 Conclusion

101 Spoiler

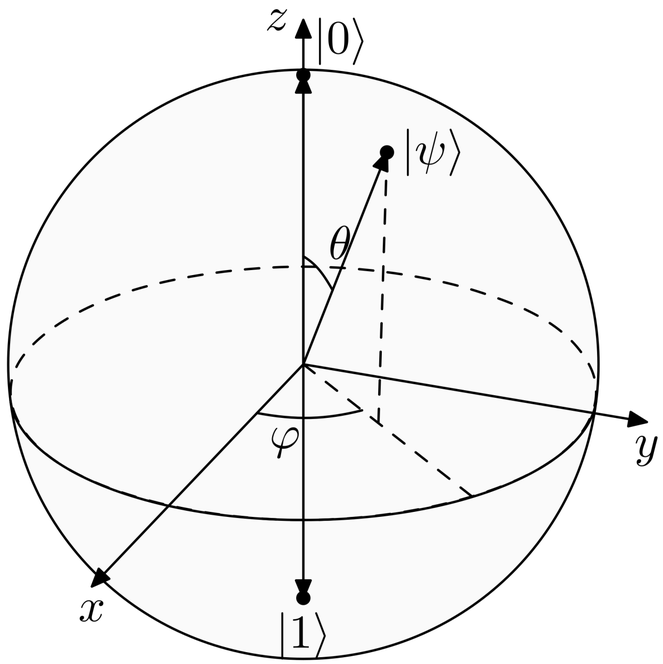
Abstraction levels in quantum computing

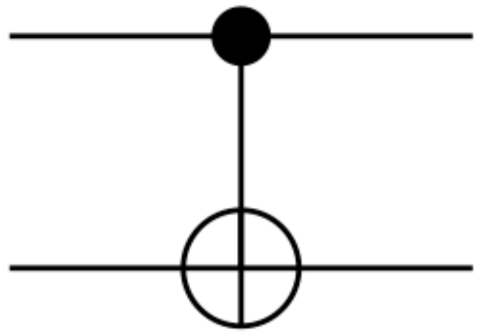
What's an abstraction? A few examples.

```c
int main(int argc, char* argv[]) {
    int i = argc;

    while(i != 1) {
        if (i % 2 == 0) {
            i /= 2;
        } else {
            i = 3 * i + 1;
        }
    }

    return 0;
}
```



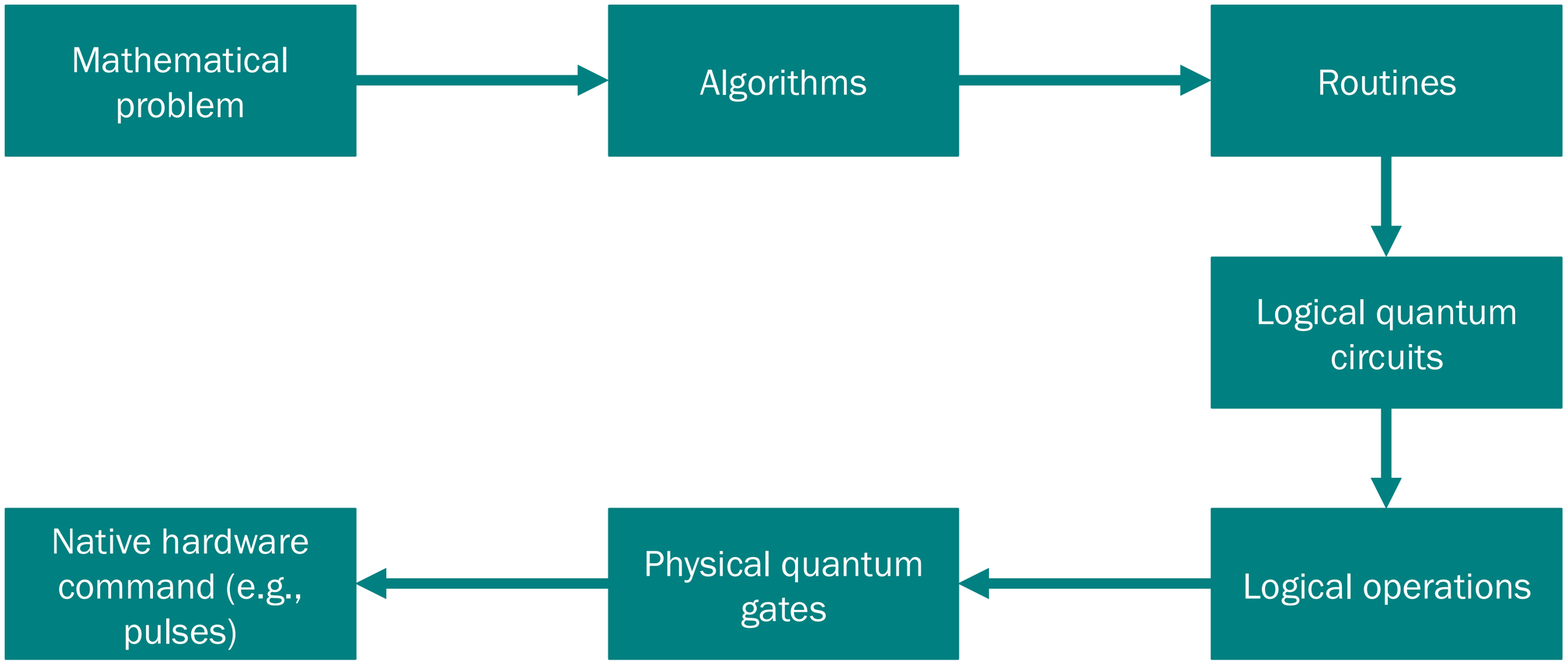


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Bloch sphere: By Notezik - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=179712998
CNOT: By Rxtreme – Own work, CC BY-SA 4.0, https://commons.wikimedia.org/wiki/File:Quantum_Logic_Gates.png

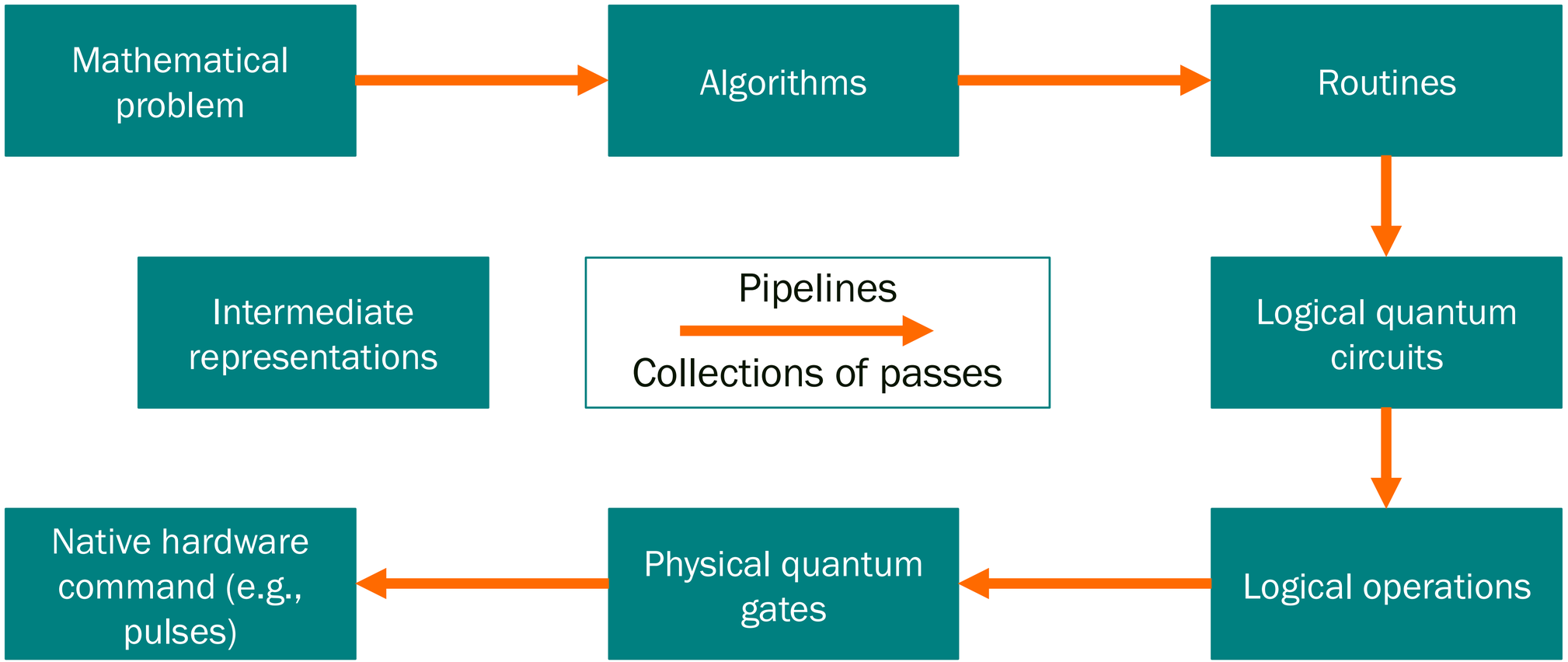
Abstraction levels in quantum computing

Stack of abstractions in QEC



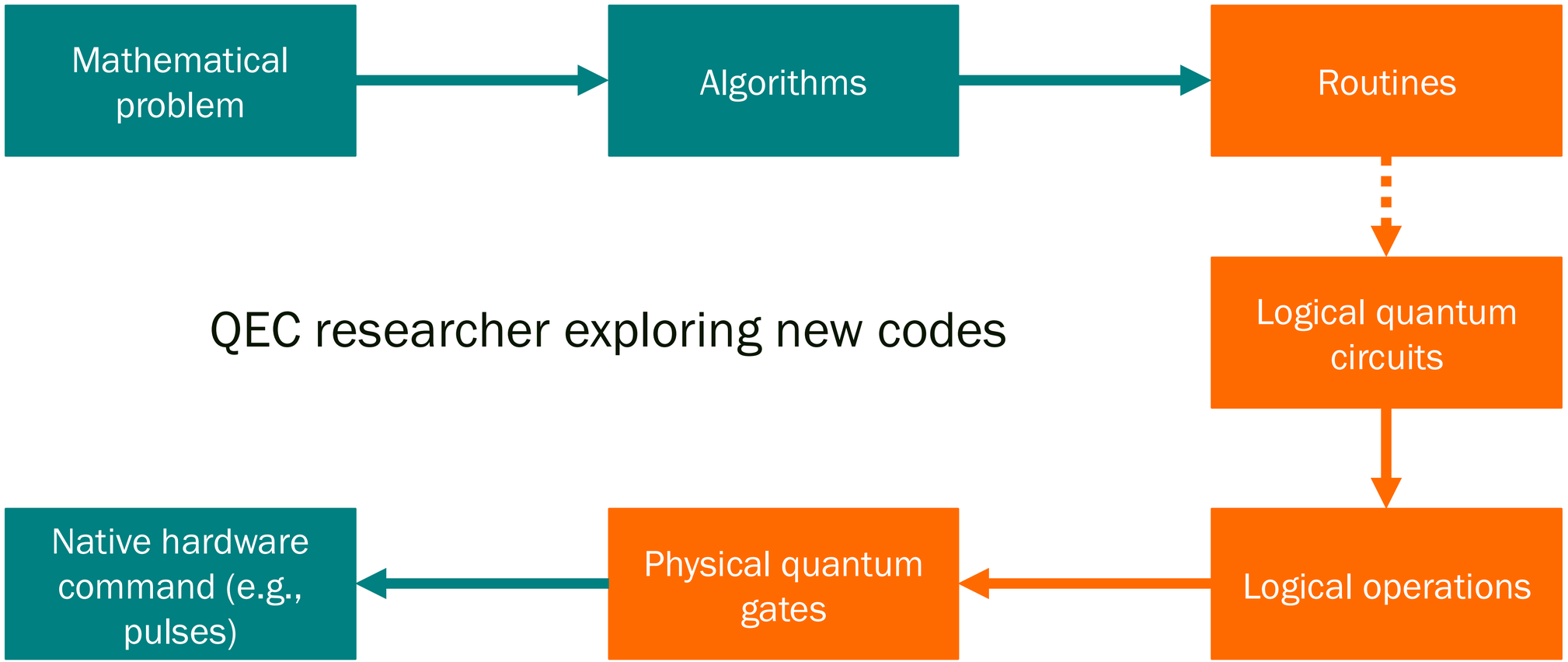
Abstraction levels in quantum computing

That’s a big compiler!



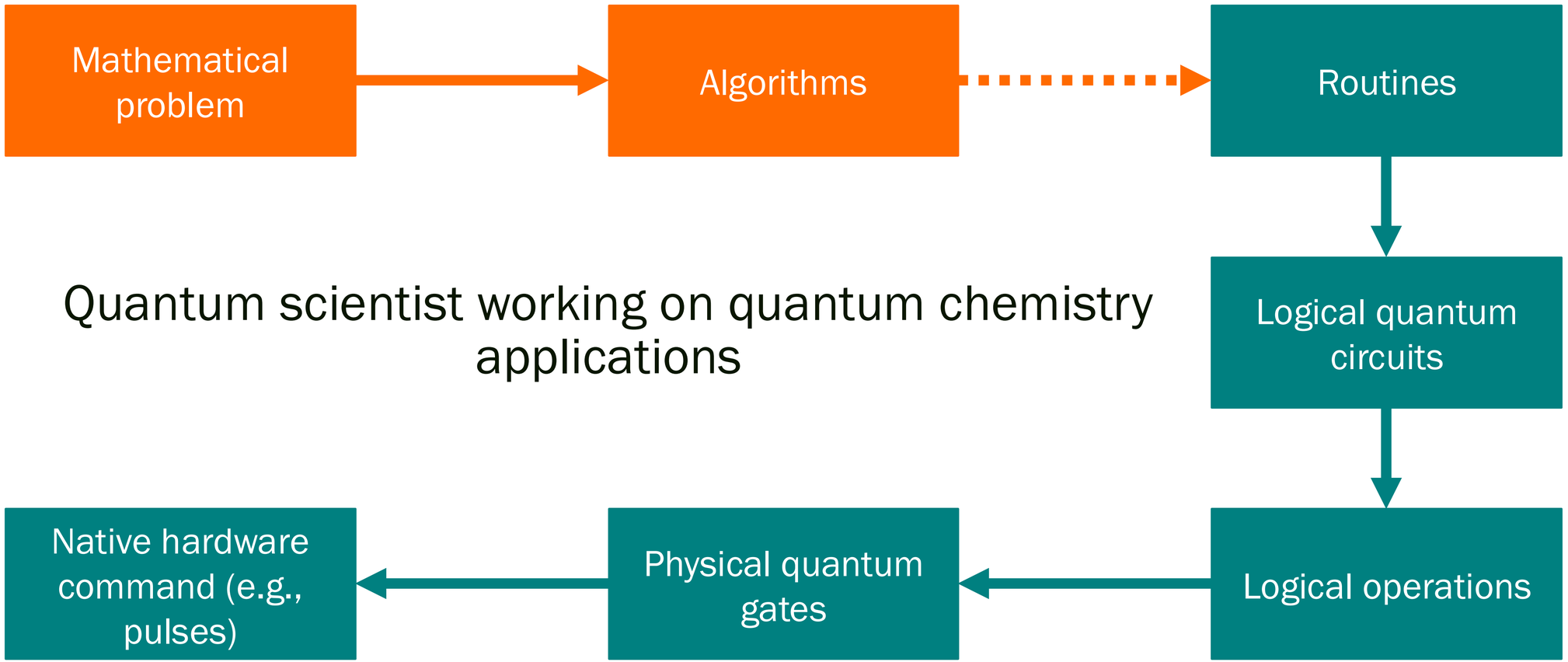
Abstraction levels in quantum computing

How to use abstractions?



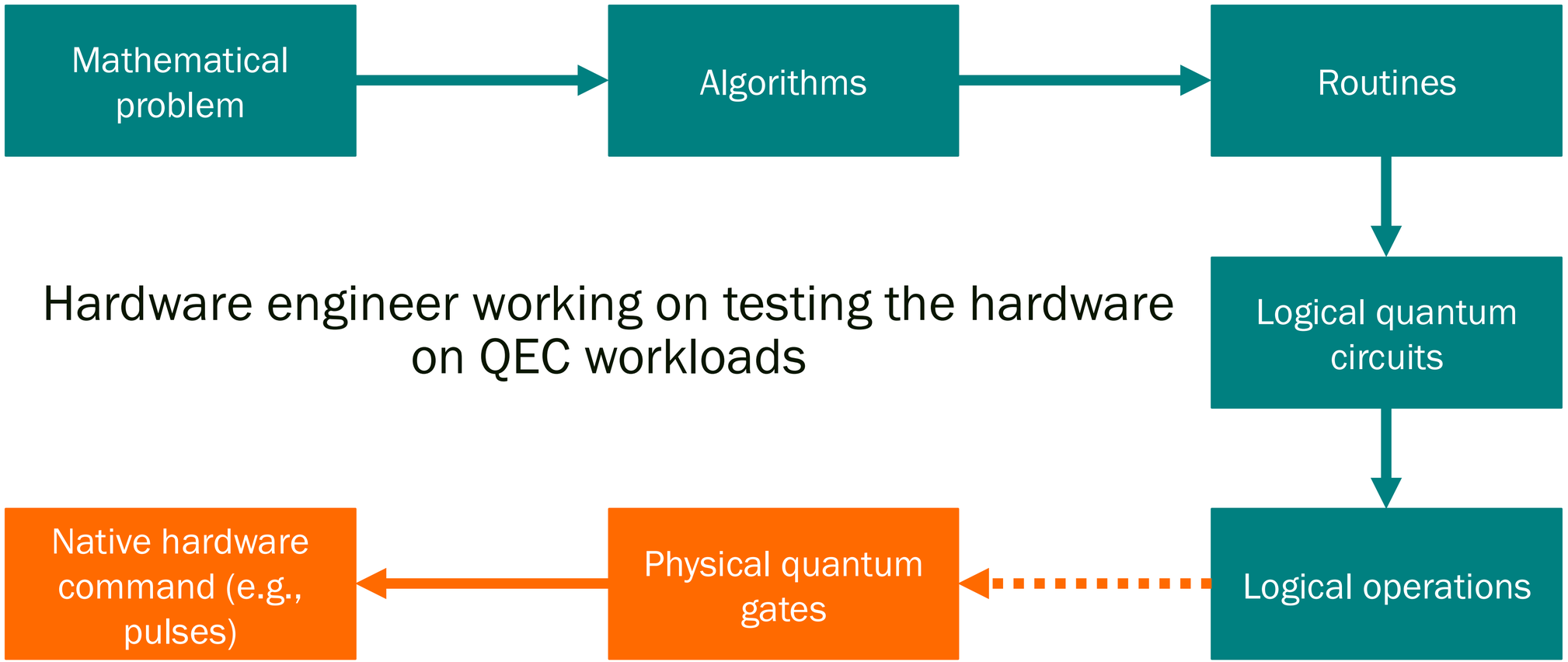
Abstraction levels in quantum computing

How to use abstractions?



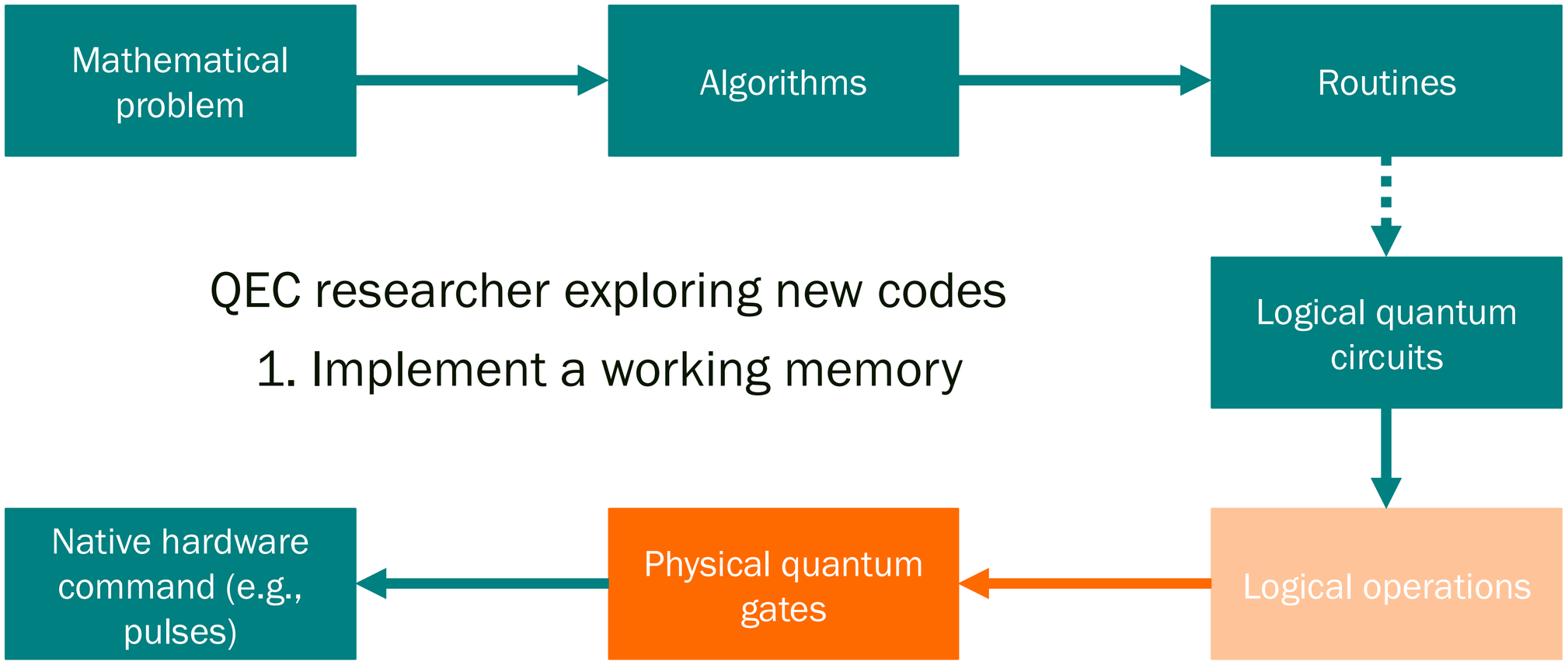
Abstraction levels in quantum computing

How to use abstractions?



Abstraction levels in quantum computing

How to use abstractions?

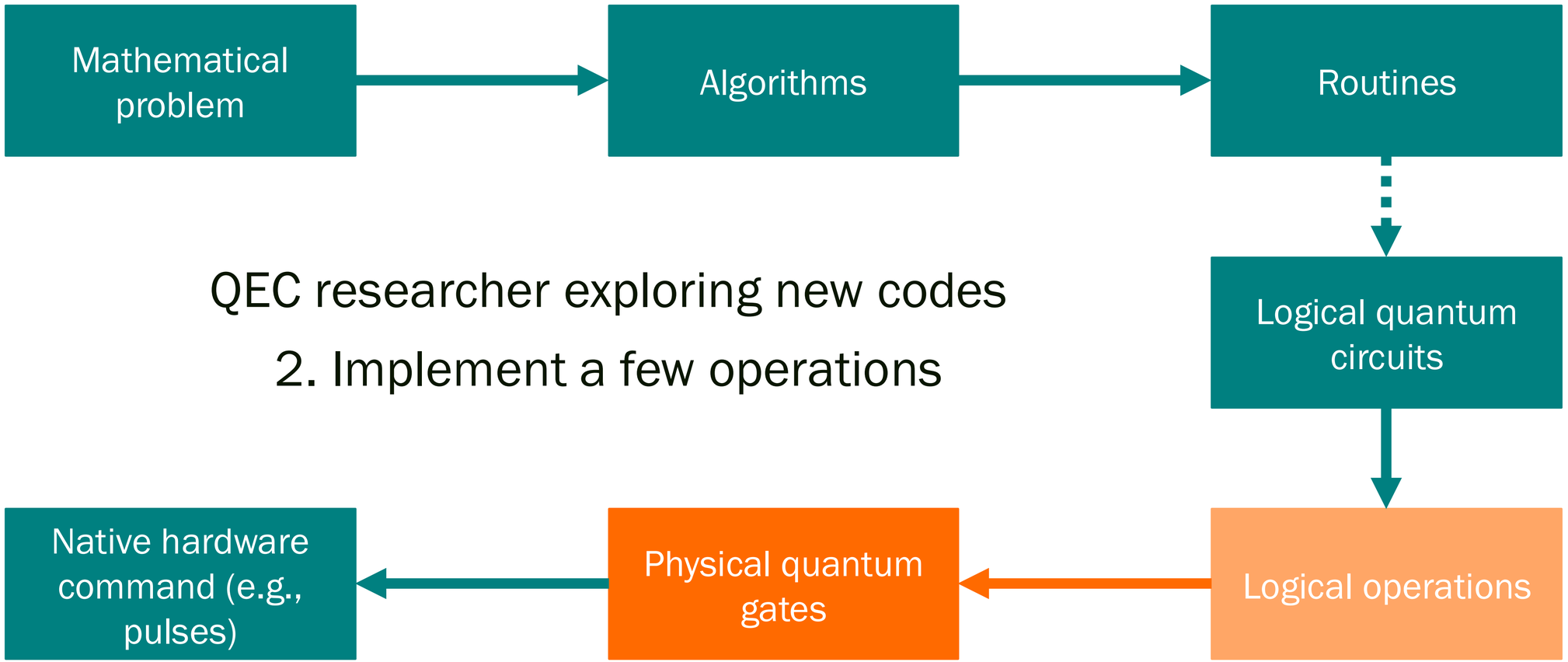


QEC researcher exploring new codes

1. Implement a working memory

Abstraction levels in quantum computing

How to use abstractions?

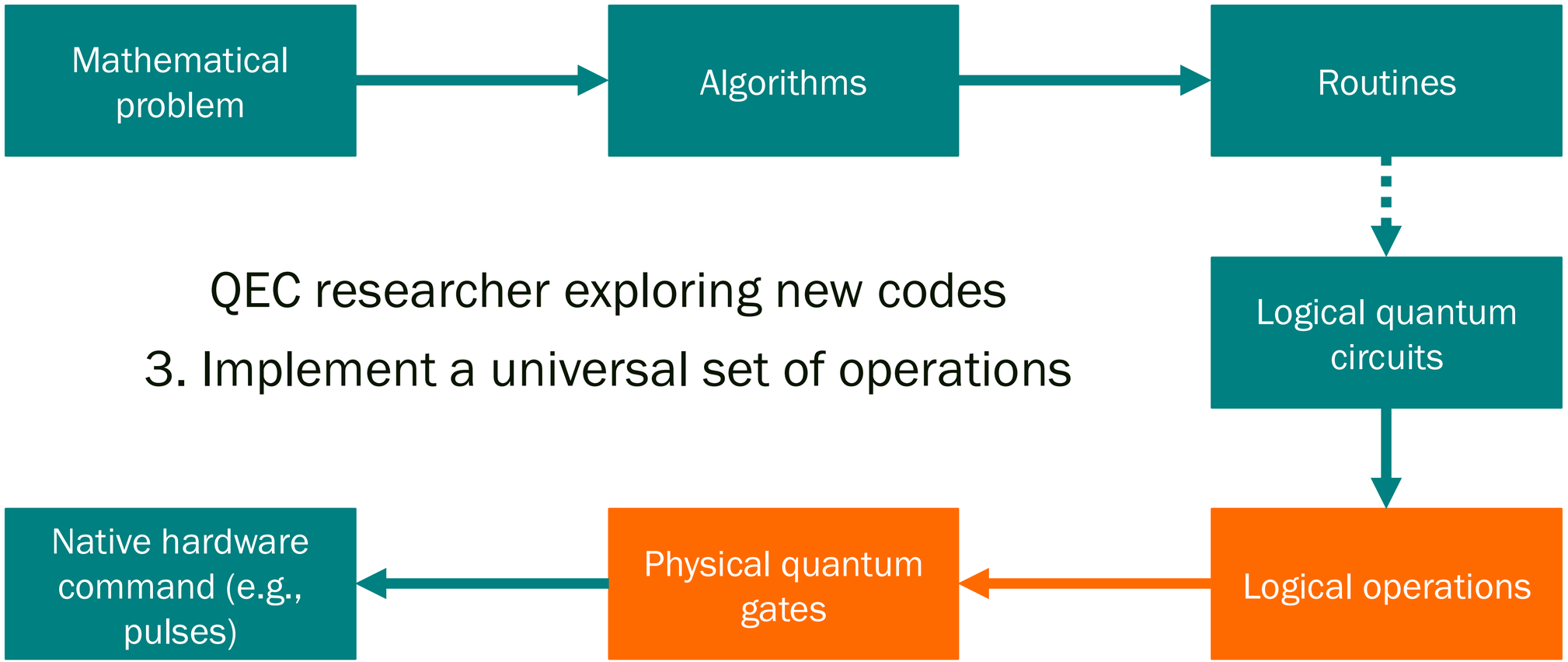


QEC researcher exploring new codes

2. Implement a few operations

Abstraction levels in quantum computing

How to use abstractions?

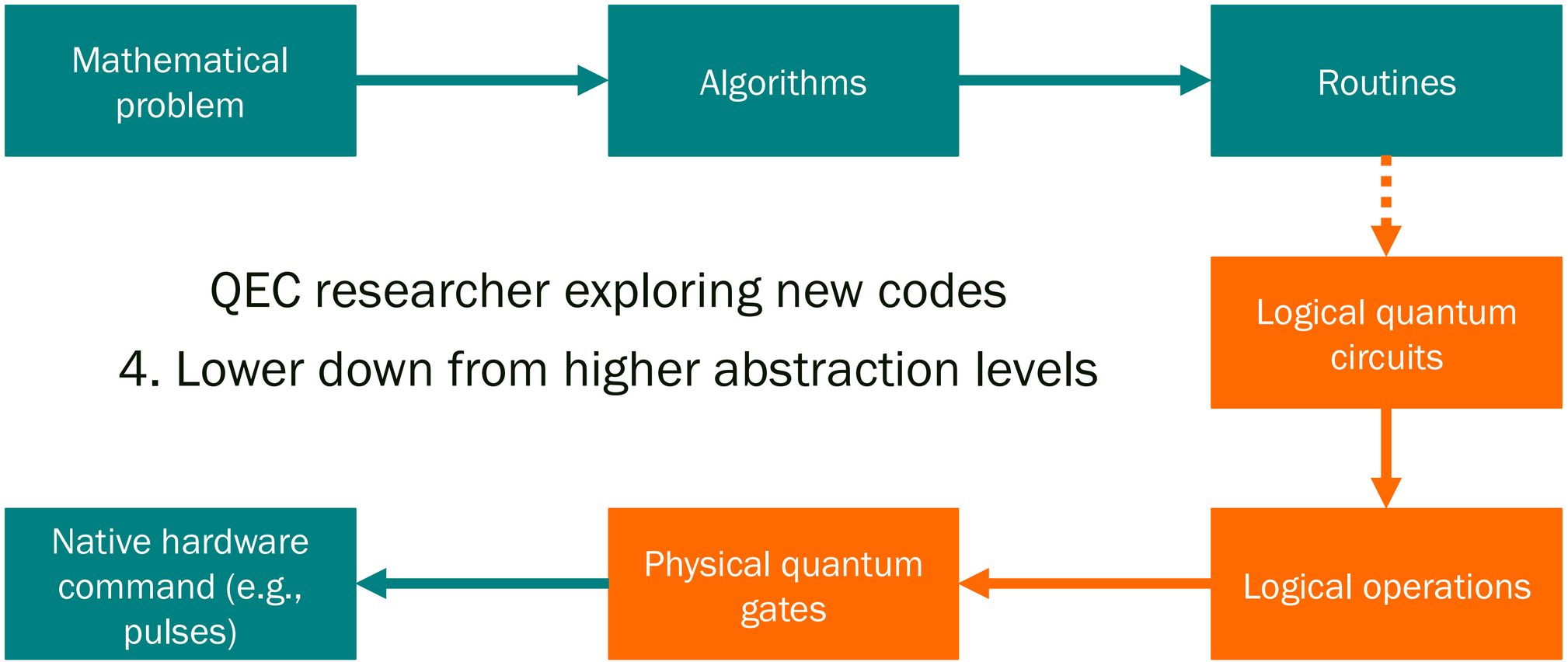


QEC researcher exploring new codes

3. Implement a universal set of operations

Abstraction levels in quantum computing

How to use abstractions?

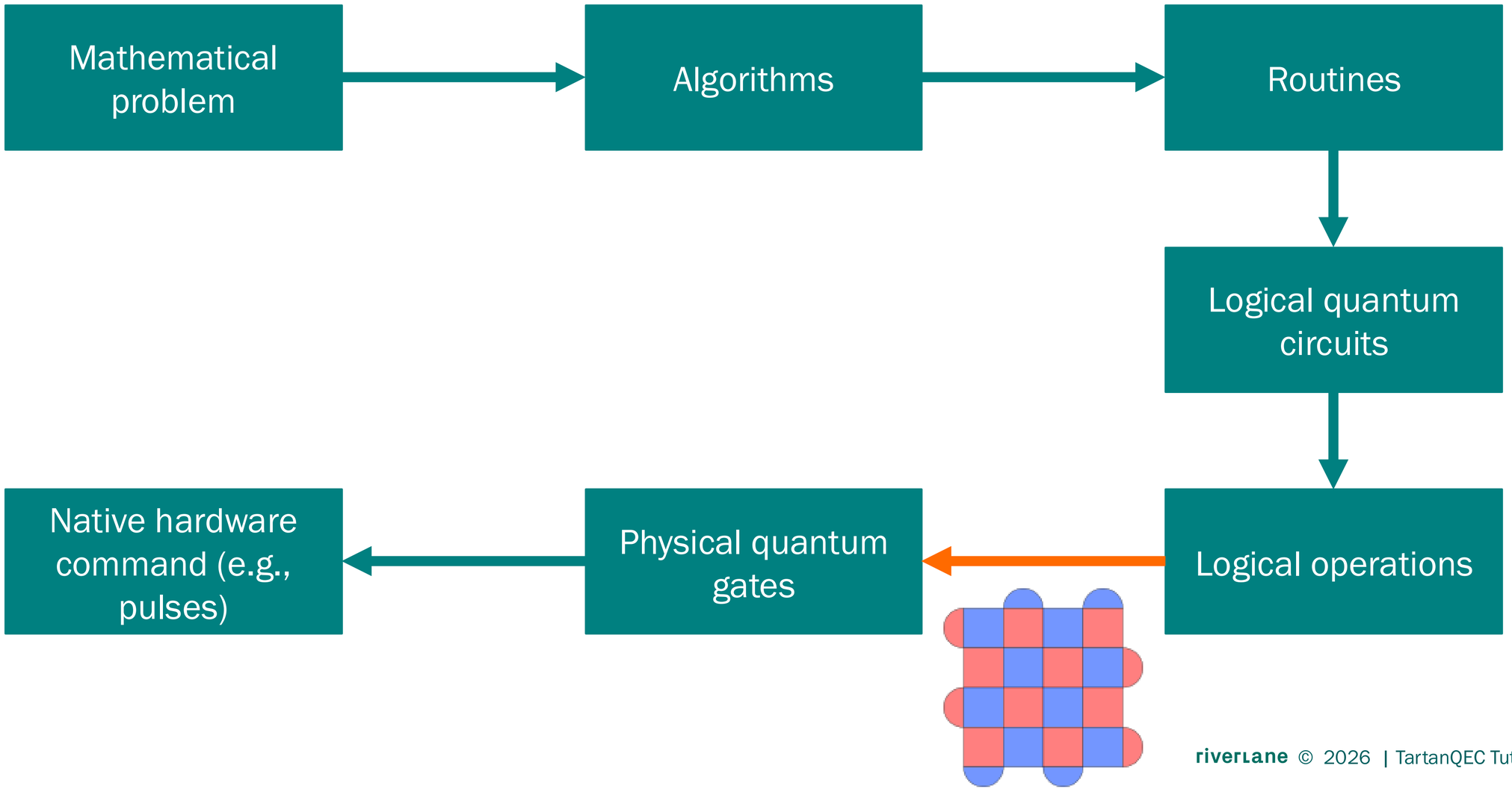


QEC researcher exploring new codes

4. Lower down from higher abstraction levels

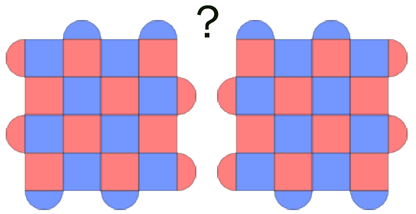
Abstraction levels in quantum computing

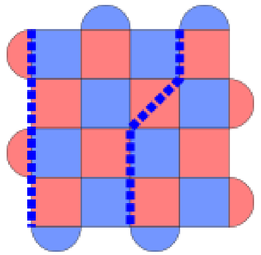
Let’s dive in

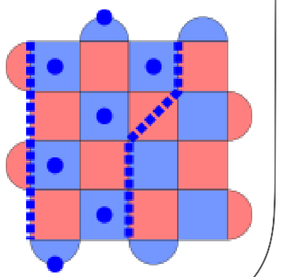


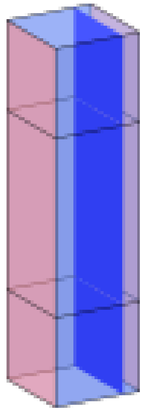
Abstraction levels in QEC

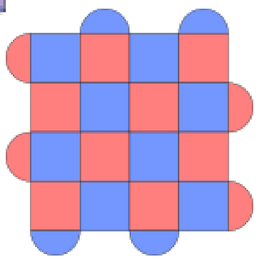
Example of a pipeline

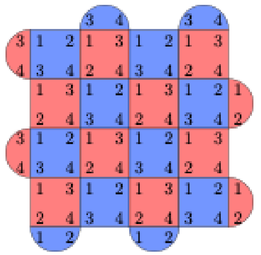


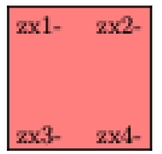


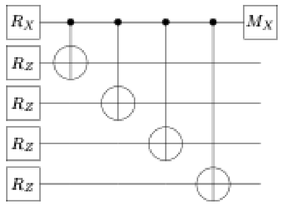












START

compute-boundary-parity

backpropagate-observables

lower-patch-declaration

patch-to-plaquettes

schedule-plaquettes

place-observables

plaquettes-to-circuits

transversal-op-to-circuit

located-observable-to-move

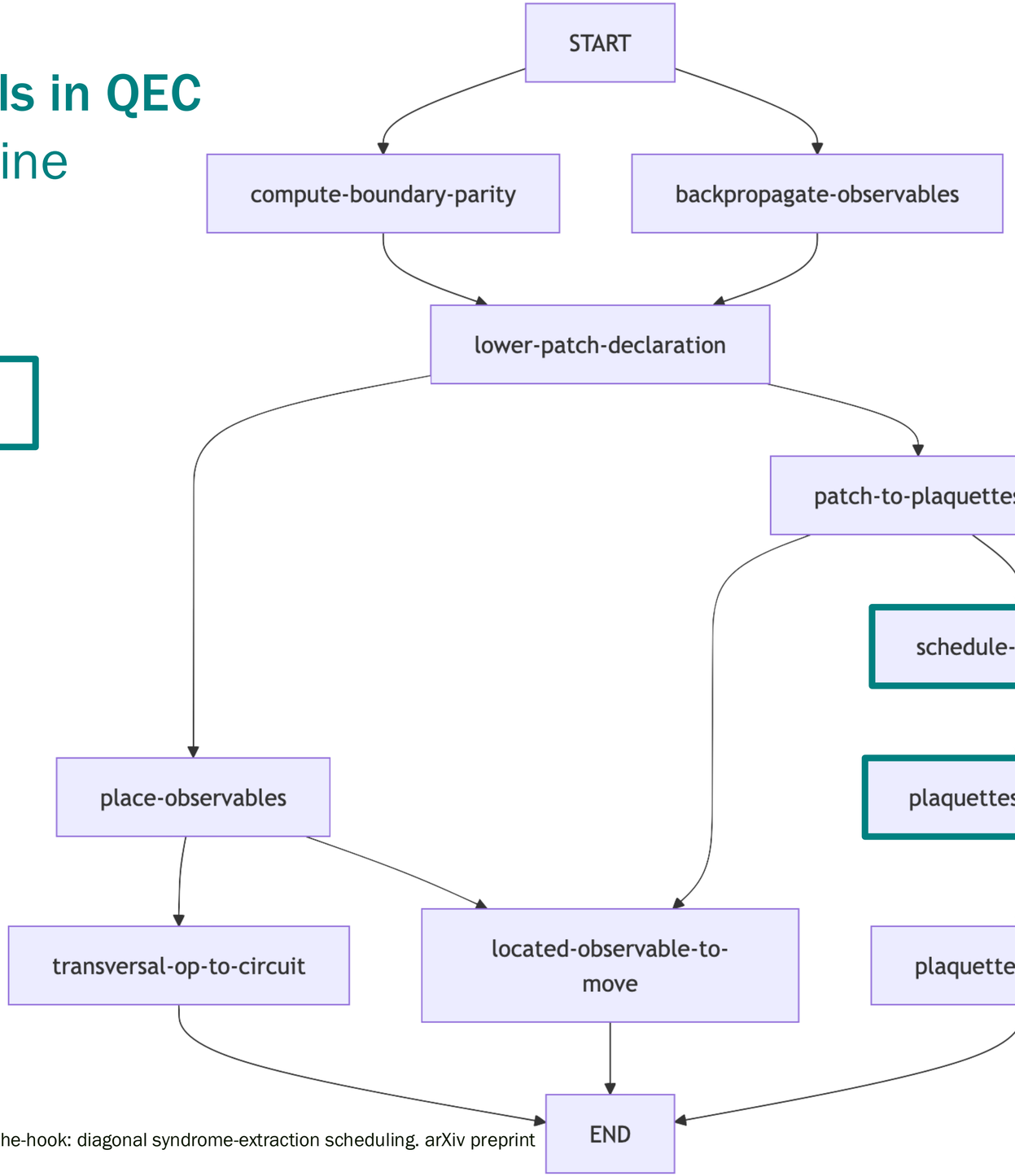
plaquette-to-qstruct

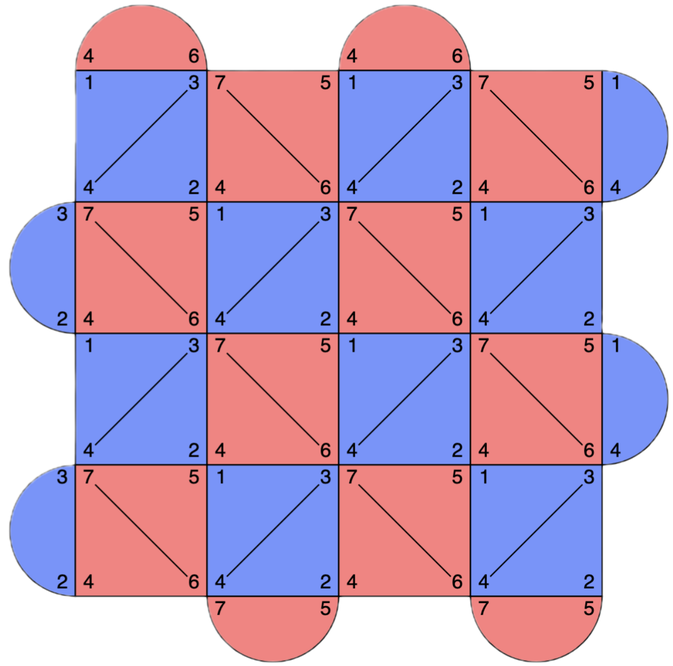
END

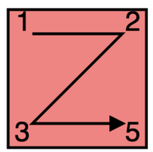
Abstraction levels in QEC

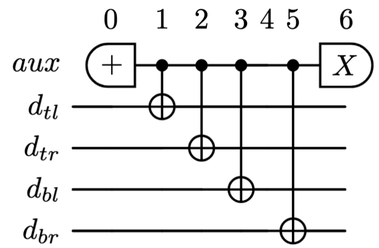
Changing a pipeline

Small changes









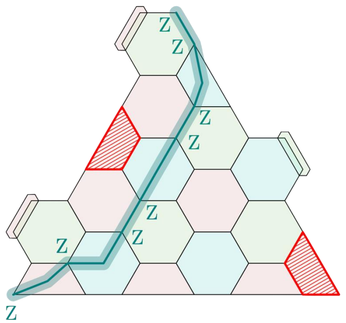
Kishony, G., & Fowler, A. (2026). Surface code off-the-hook: diagonal syndrome-extraction scheduling. arXiv preprint arXiv:2602.09099.

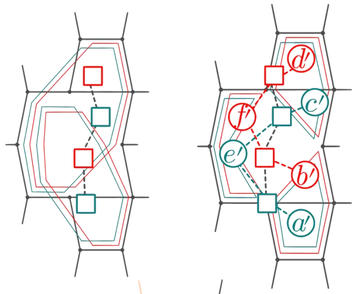
Abstraction levels in QEC

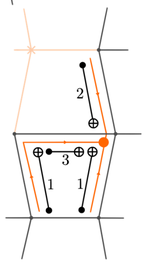
Changing a pipeline

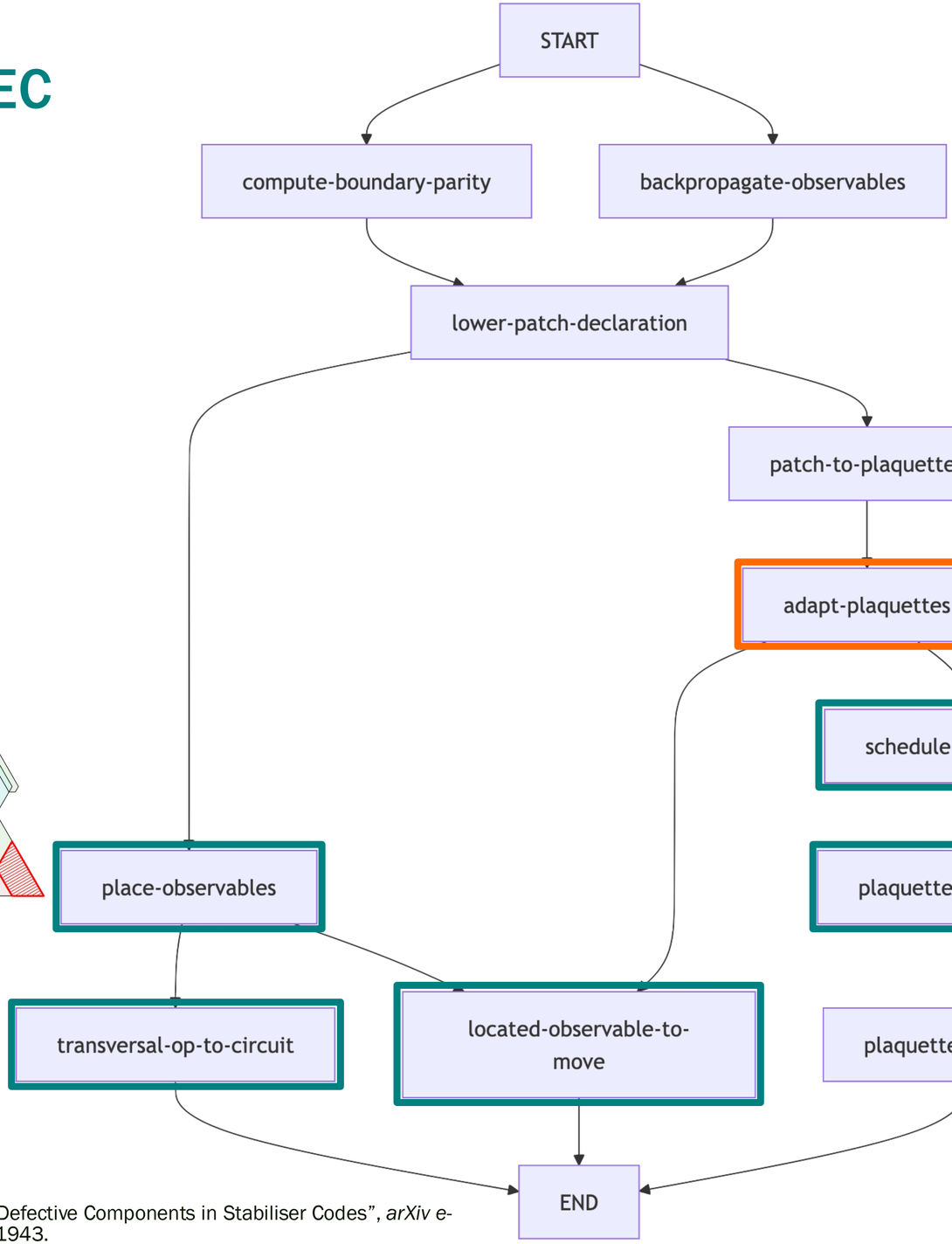
Significant changes

Small changes





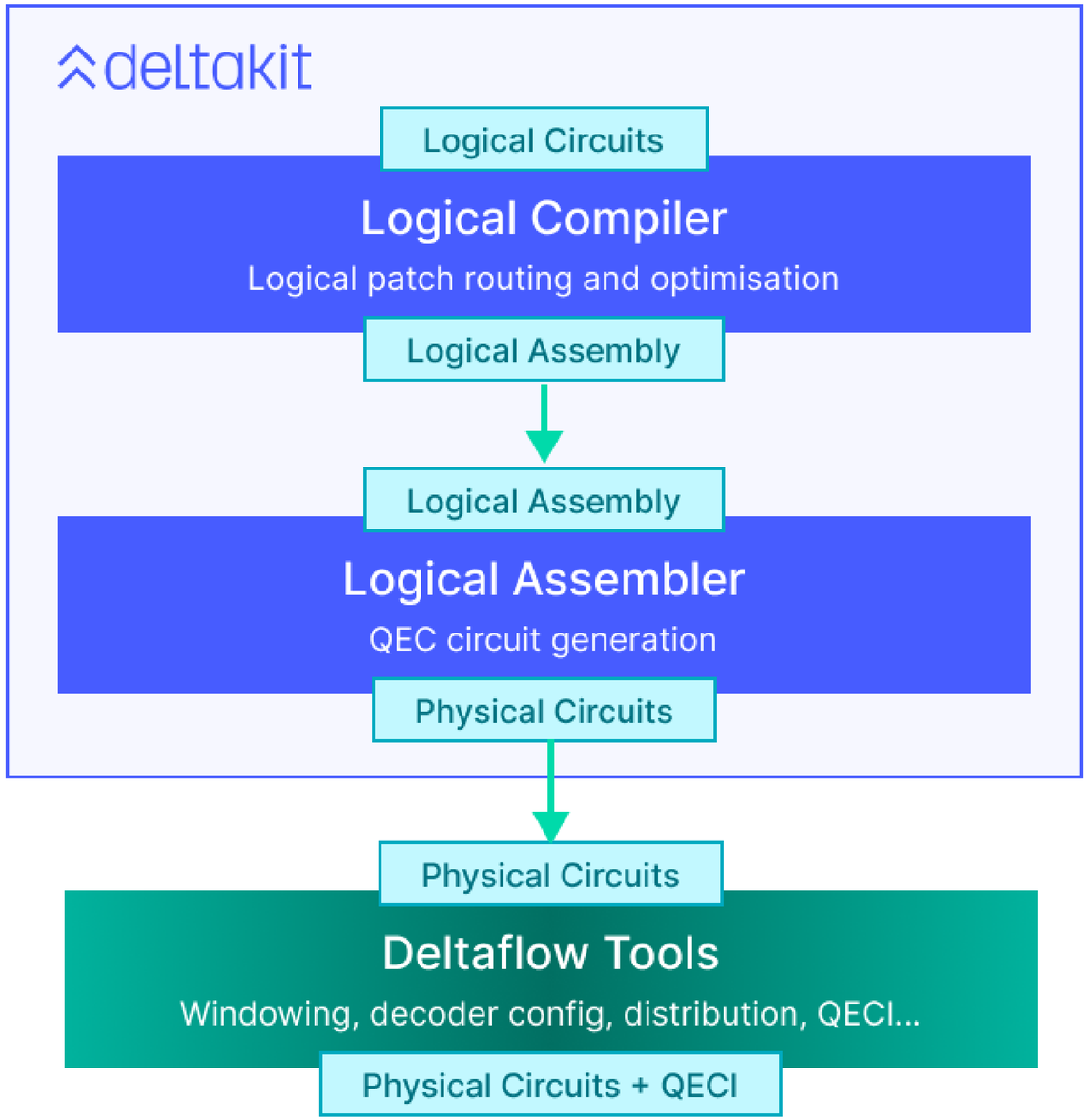




Wolanski, S., “Automated Compilation Including Dropouts: Tolerating Defective Components in Stabiliser Codes”, *arXiv e-prints*, Art. no. arXiv:2512.01943, 2025. doi:10.48550/arXiv.2512.01943.

Abstraction levels in QEC

Zooming out



Wolanski, S., “Automated Compilation Including Dropouts: Tolerating Defective Components in Stabiliser Codes”, *arXiv e-prints*, Art. no. arXiv:2512.01943, 2025. doi:10.48550/arXiv.2512.01943.

Plan of the talk

000 Introduction

001 Compilers – What, why and how?

010 Abstraction levels in quantum computing

011 **How to benefit from open-source in your work?**

100 Conclusion

101 Spoiler

How to benefit from open-source in your work?

Show of hands

Who is using open-source in the room?

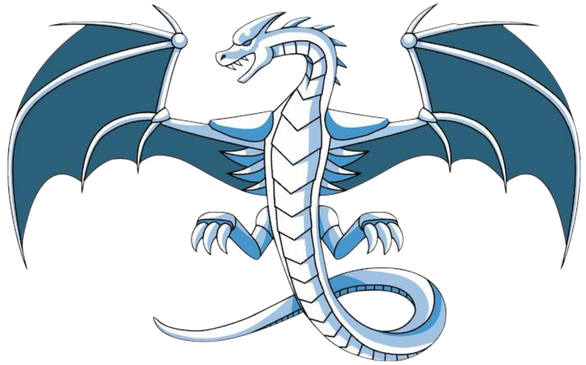
How to benefit from open-source in your work?

One pip install command away

```
pip install \
deltakit bposd qldpc stim sinter stimflow pymatching \
chromobius tesseract-decoder tqec clifft tsim pyzx ucc \
qiskit pennylane cirq pytket qrisp mitiq qualtran \
numpy scipy matplotlib networkx pandas
```

How to benefit from open-source in your work?

pip is only the tip of the iceberg



How to ~~benefit from~~ contribute to open-source in your work?

- Publish on GitHub / Gitlab
- Integrate with existing open-source tools
- Use an open-source license
- Take an additional week to clean-up the research code

Plan of the talk

000 Introduction

001 Compilers – What, why and how?

010 Abstraction levels in quantum computing

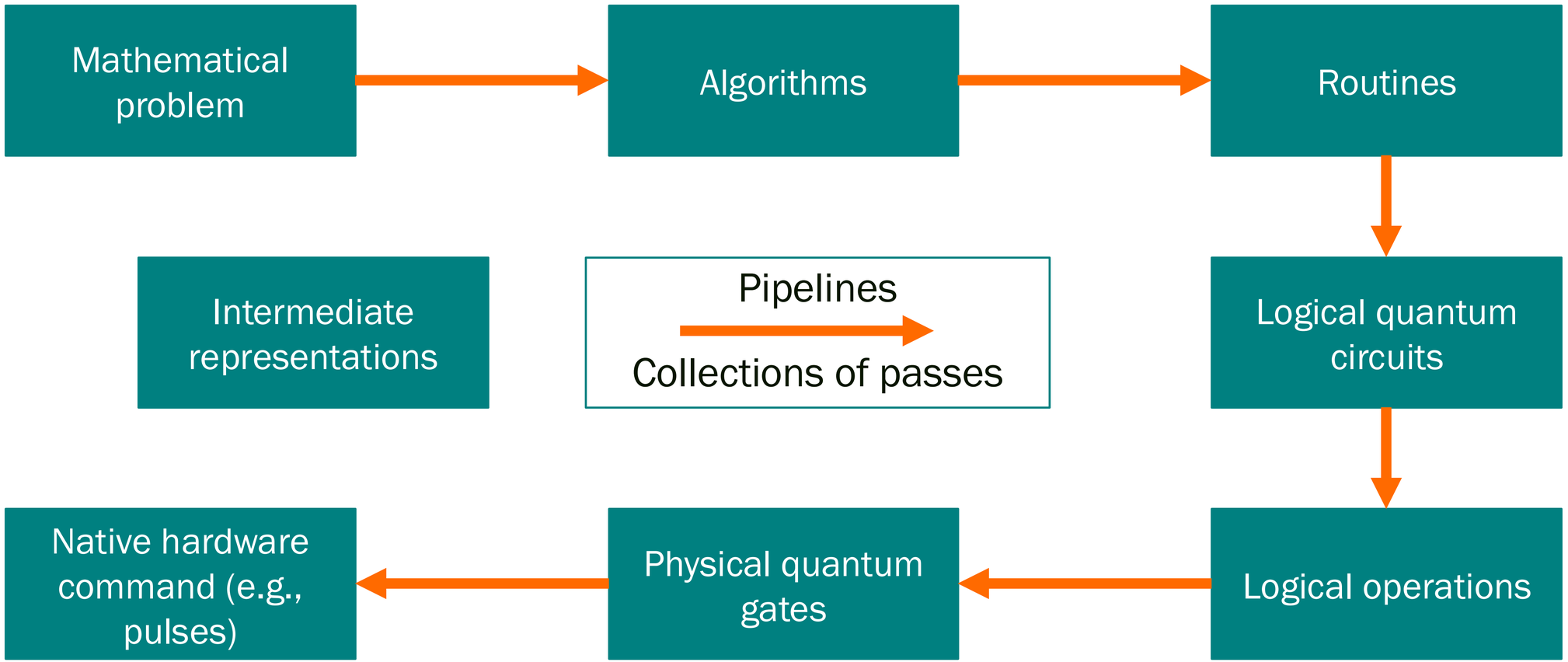
011 How to benefit from open-source in your work?

100 **Conclusion**

101 Spoiler

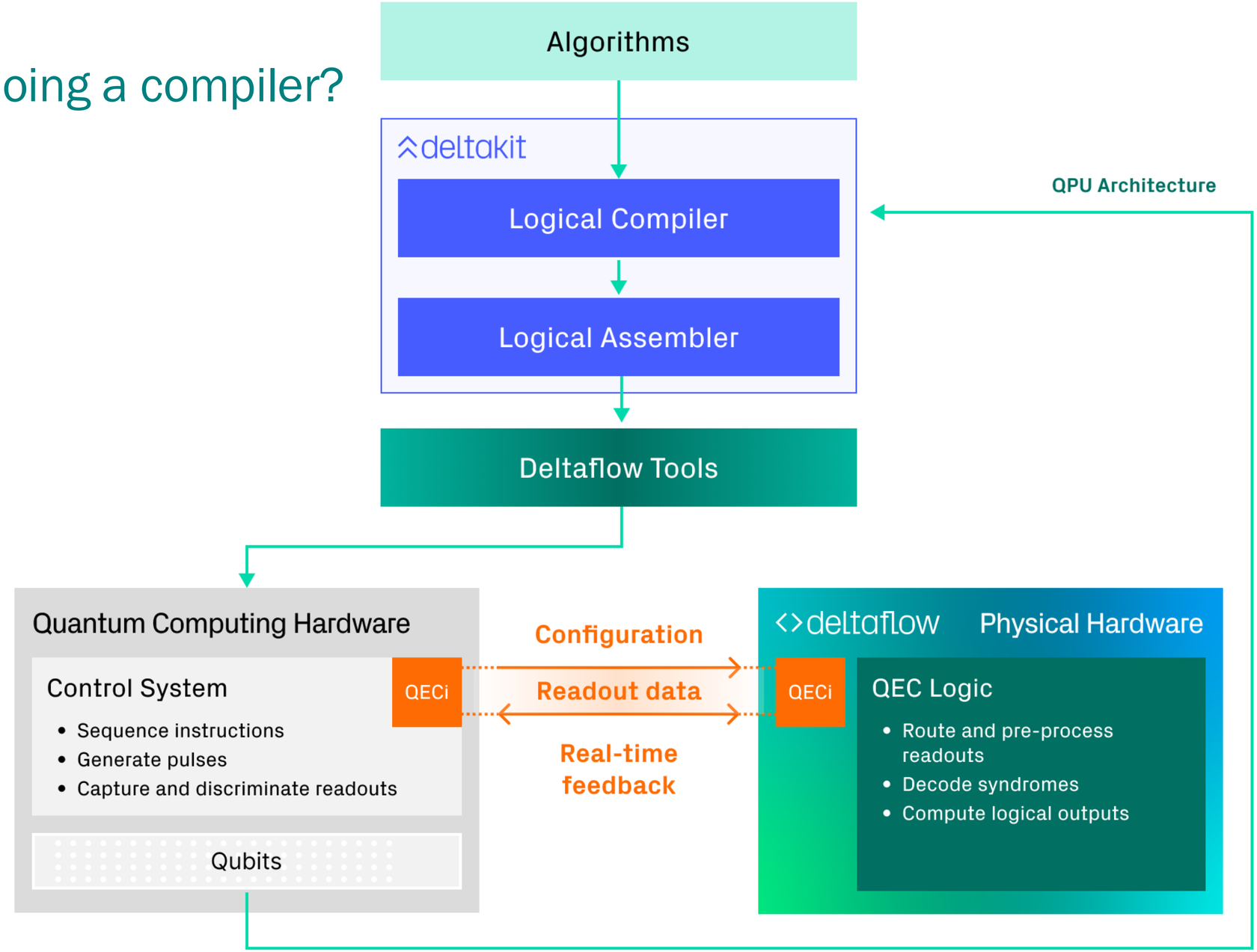
Conclusion

What’s a compiler?



Conclusion

Why are we doing a compiler?



Conclusion

Why open-source

Provide good QEC tools to the community for the field to grow even quicker.

Show what we think is the right way of handling QEC.

Guarantee that the open-source compiler can be used with a real time decoder.

Plan of the talk

000 Introduction

001 Compilers – What, why and how?

010 Abstraction levels in quantum computing

011 How to benefit from open-source in your work?

100 Conclusion

101 **Spoiler**

Spoilers

User interface

Logical assembly

```
d = 5
builder = LogAsmBuilder()
p0 = builder.declare_patch(RotatedPlanarPatch(d, d))
p1 = builder.declare_patch(RotatedPlanarPatch(d, d))

p0.prepare(Z)
p1.prepare(Z)
p0.measure_stabilisers(8 * d)
p1.measure_stabilisers(8 * d)

b0 = p0.measure(Z)
with builder.if_(b0 == 1):
    p1.transversal(X)

b1 = p1.measure(Z)
builder.add_returns(b0, b1)

Compiler().compile(builder.build())
```

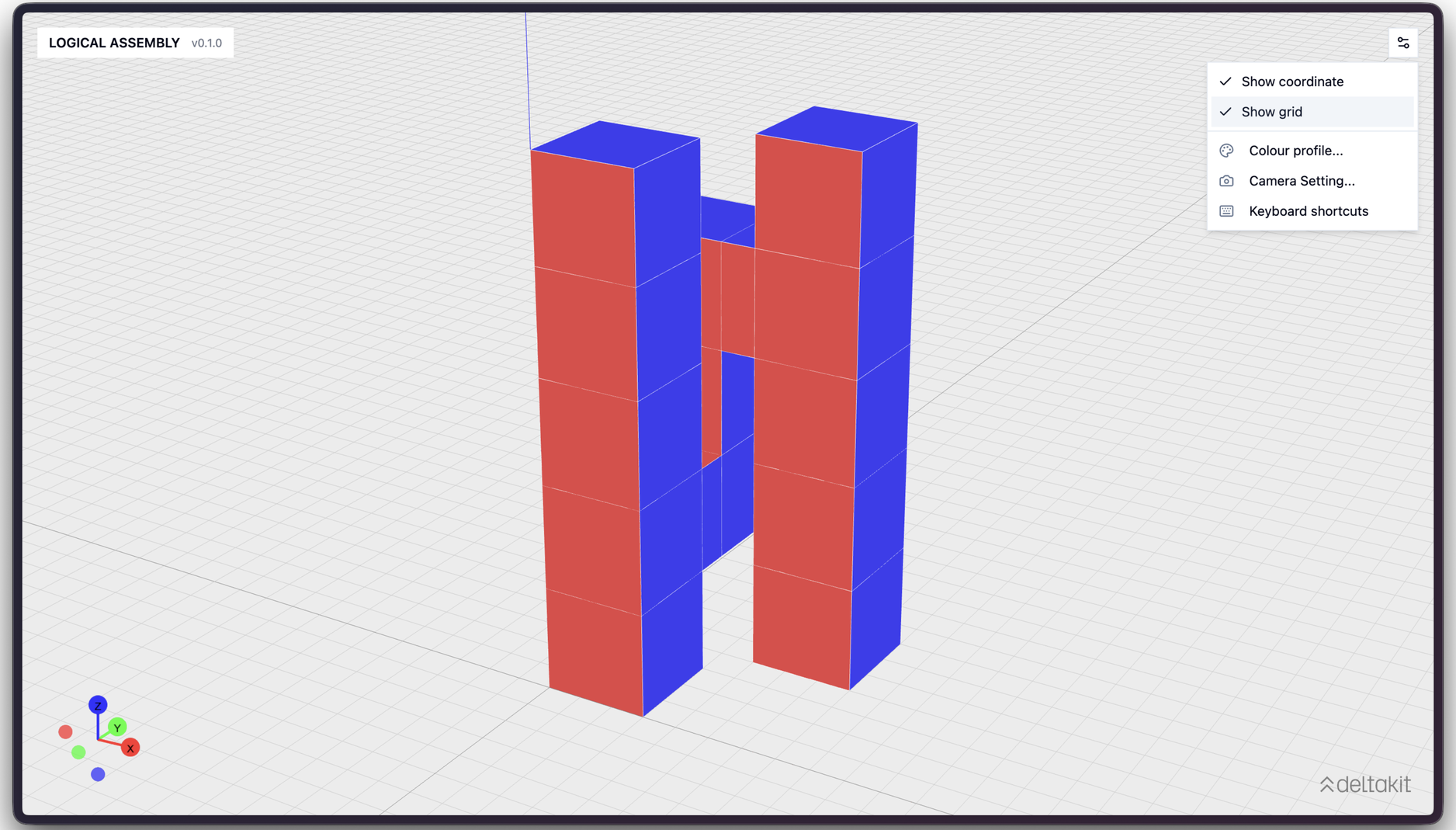
Circuit

```
# Declarations
builder = CircuitBuilder()
qubits = builder.add_args(QubitReg(9))
recs = [builder.declare_record() for _ in range(4)]
# Circuit
builder.gate("R", qubits)
builder.gate("CX", qubits[:-1])
builder.gate("CX", [qubits[i] for i in (2, 1, 4, 3, 6, 5, 8, 7)])
readouts = builder.measure(Pauli.Z, [qubits[i] for i in (1, 3, 5, 7)])
# Records and detectors
for i in range(4):
    recs[i].append(readouts[i])
repetition_5_prep = builder.build()

logical_builder = LogAsmBuilder()
qreg = logical_builder.add_arg(QubitReg(9))
logical_builder.call_circuit(repetition_5_prep(qreg))
```

Spoiler

Integrated visualisation



We are hiring

- QEC Researchers
- QEC Application Scientists
- Principal Investigator, QEC
- Quantum Scientists
- Quantum Engineers

Roles in Cambridge UK, and Boston, US

View our latest vacancies:
www.riverlane.com/jobs