

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

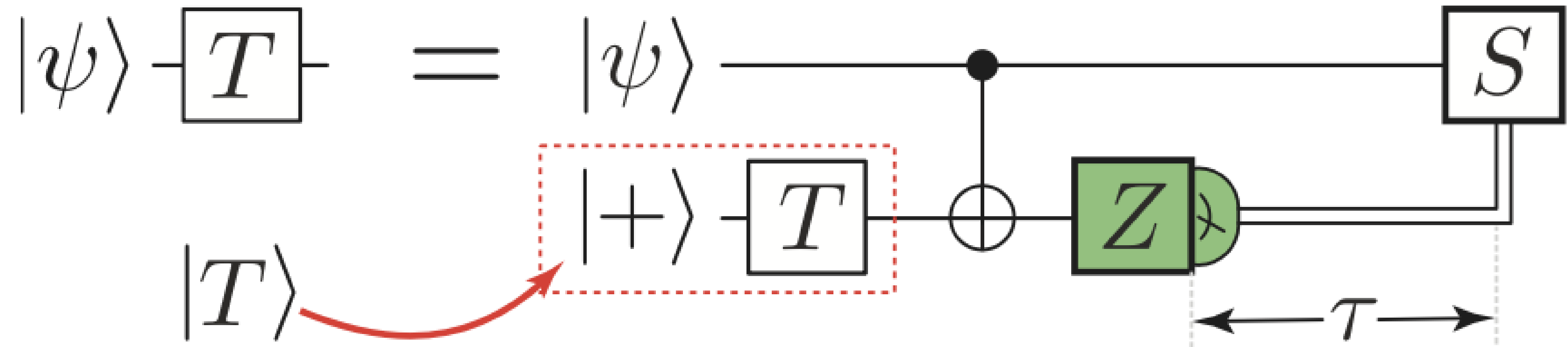
- 000 Introduction
- 001 Compilers – What, why and how?
- 010 Abstraction levels in quantum computing
- 011 How to benefit from open-source in your work?
- 100 Conclusion
- 101 Spoiler

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

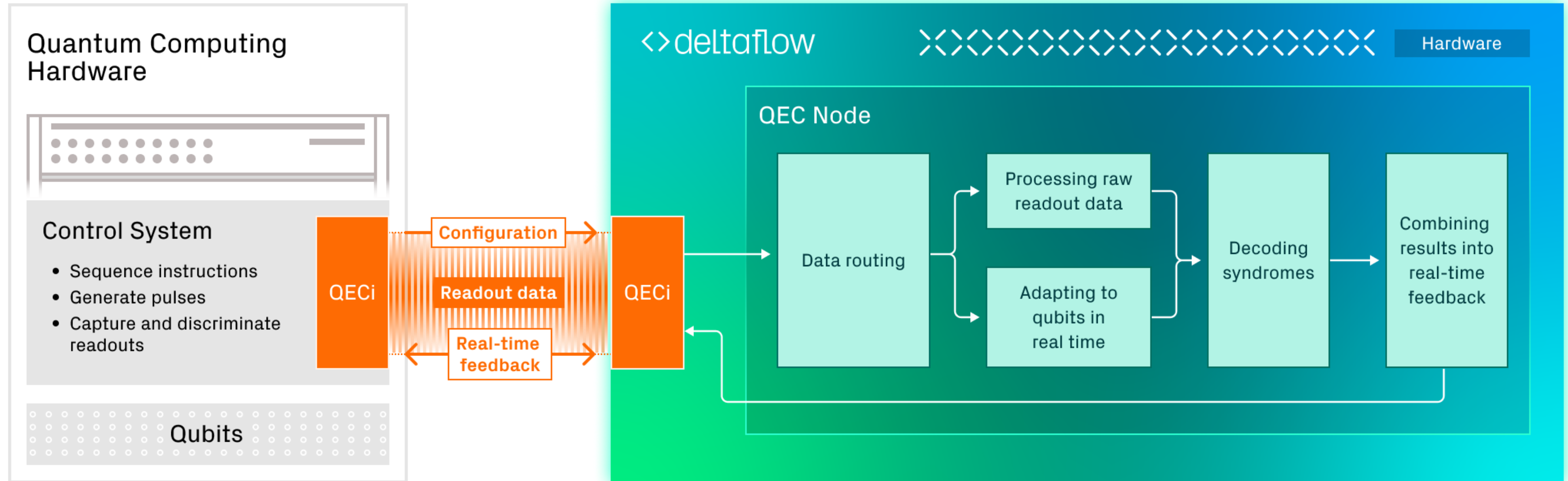
Introduction

Universal computation using QEC



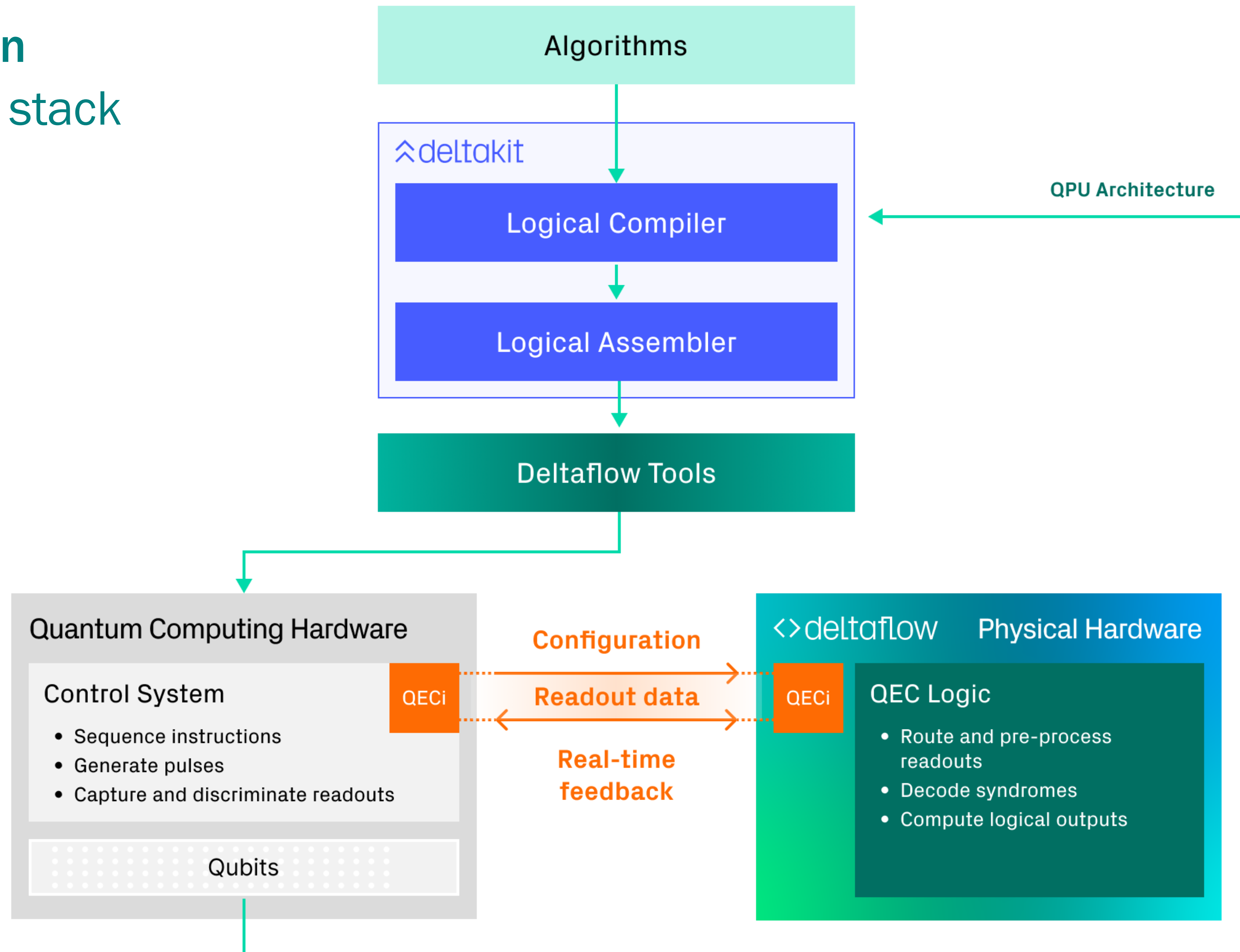
Introduction

Hardware QEC system



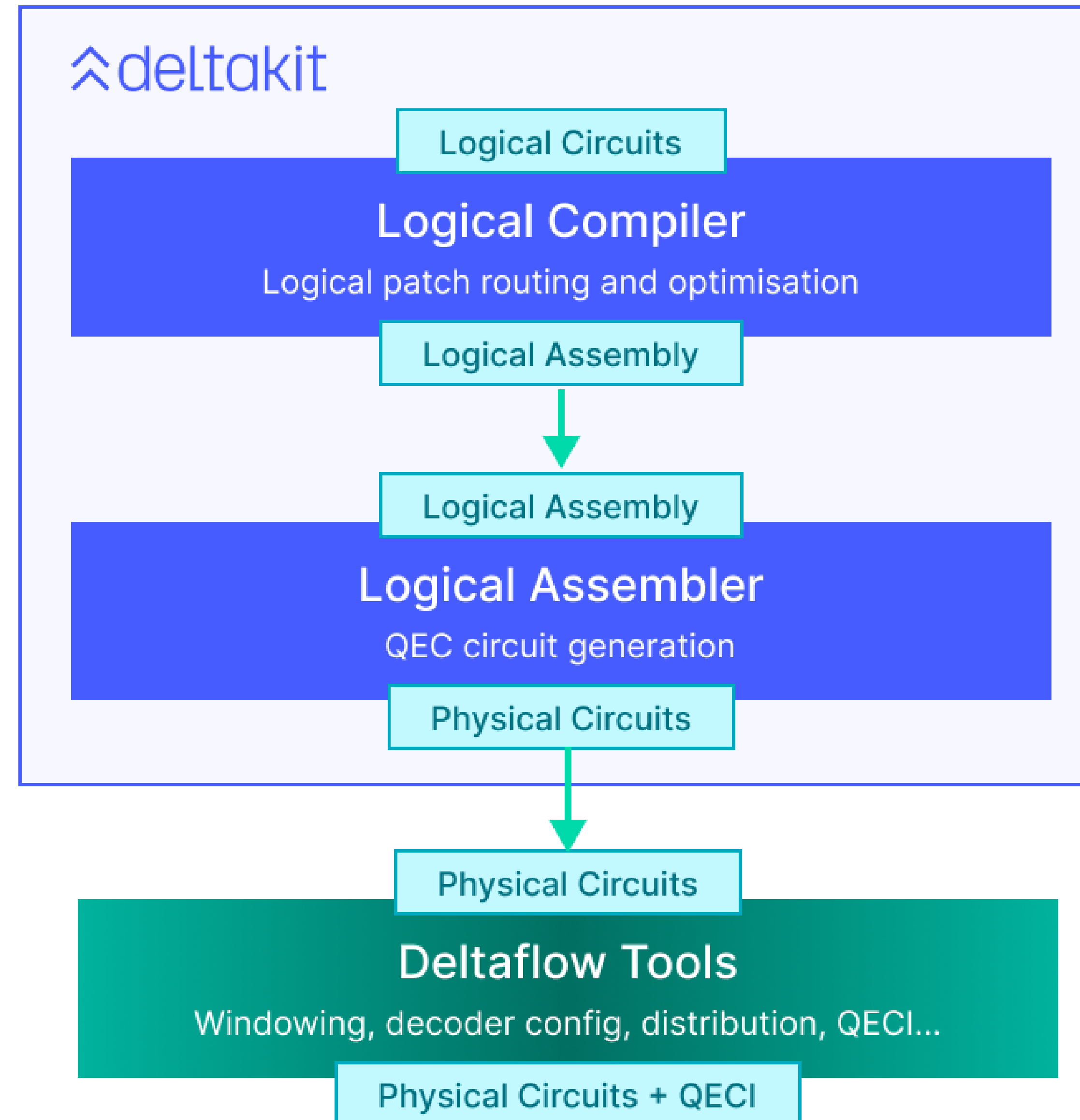
Introduction

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

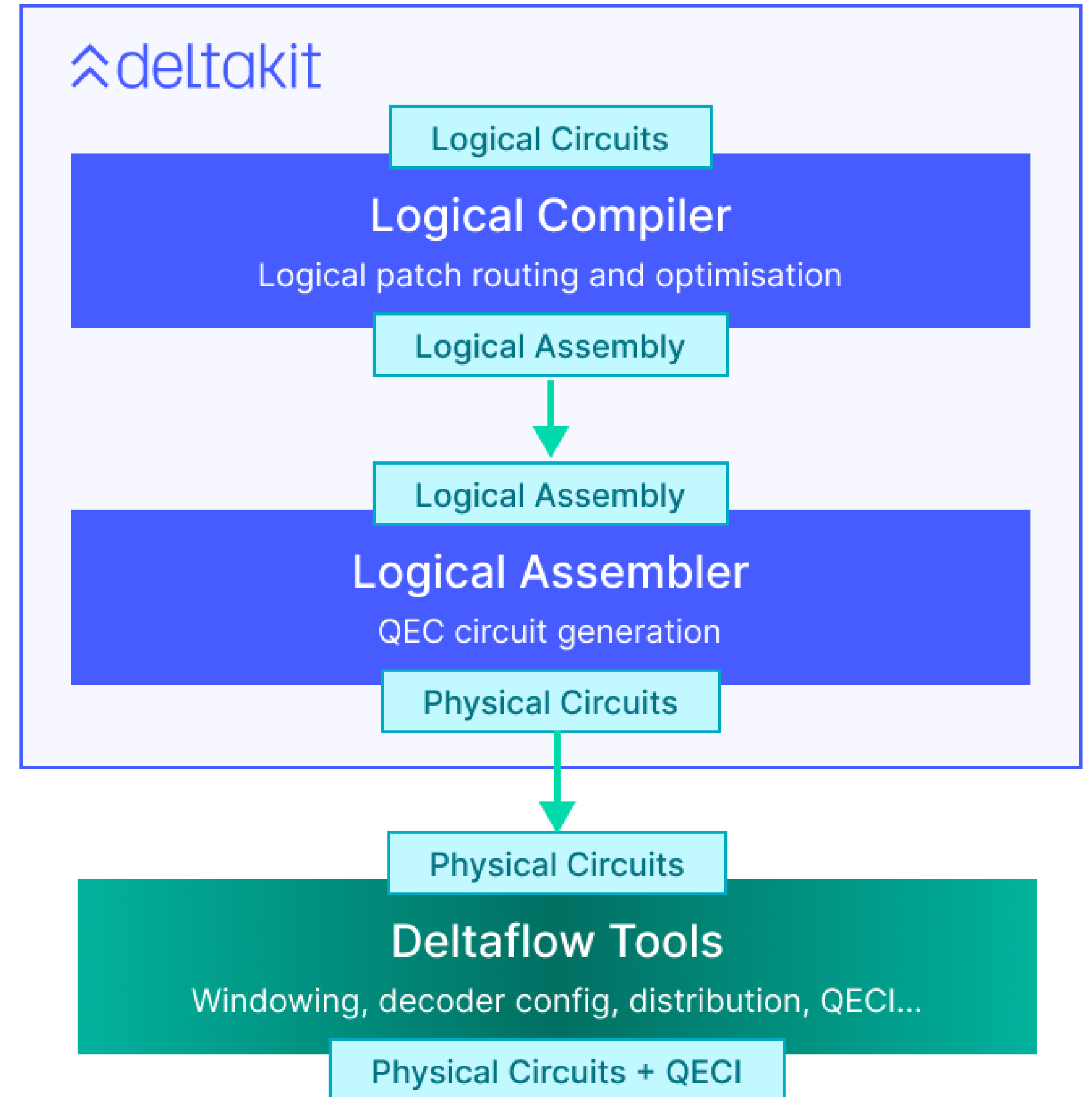
Compilers – What, why and how?

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:

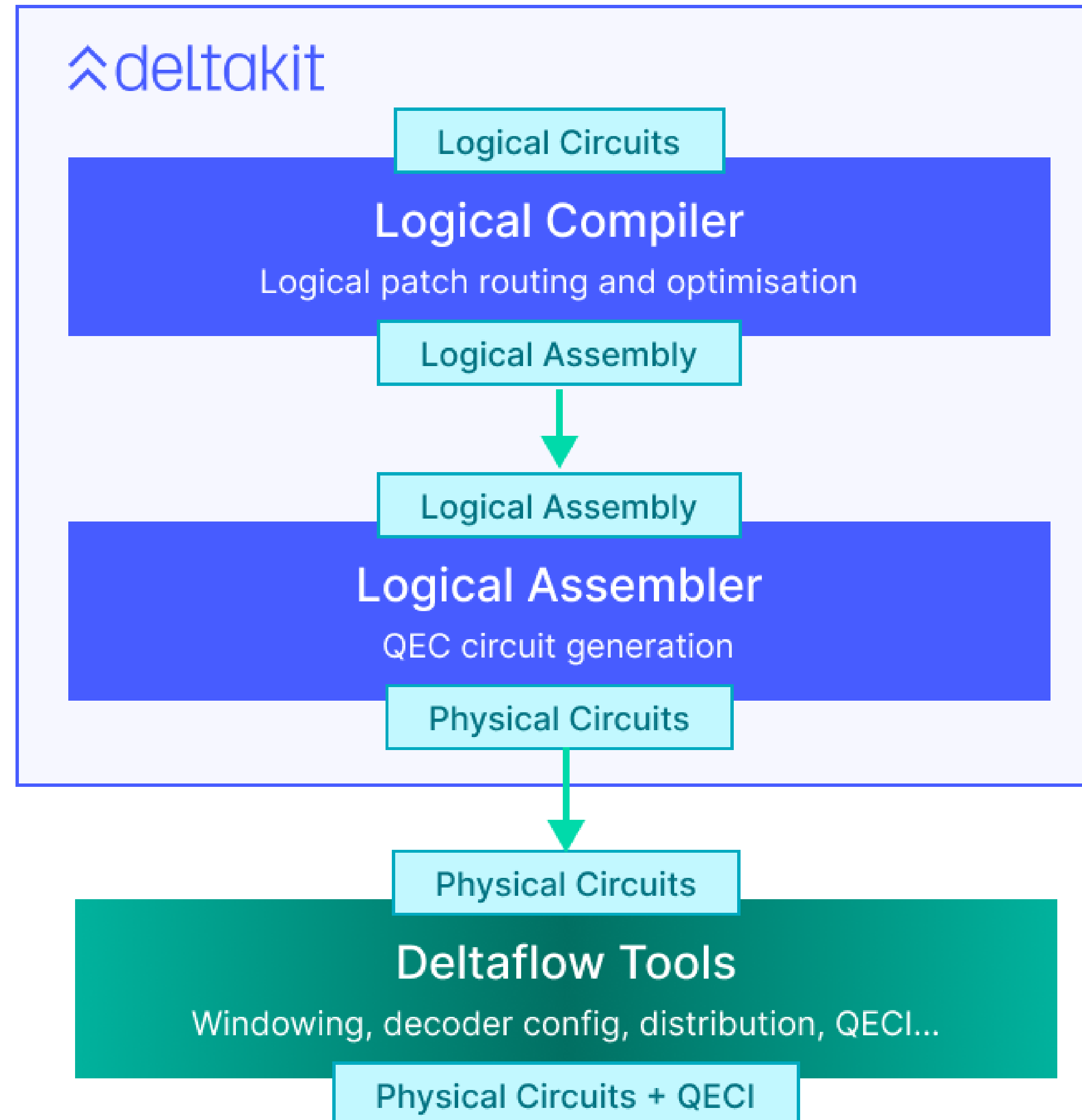
A collection of passes in a specific order.

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

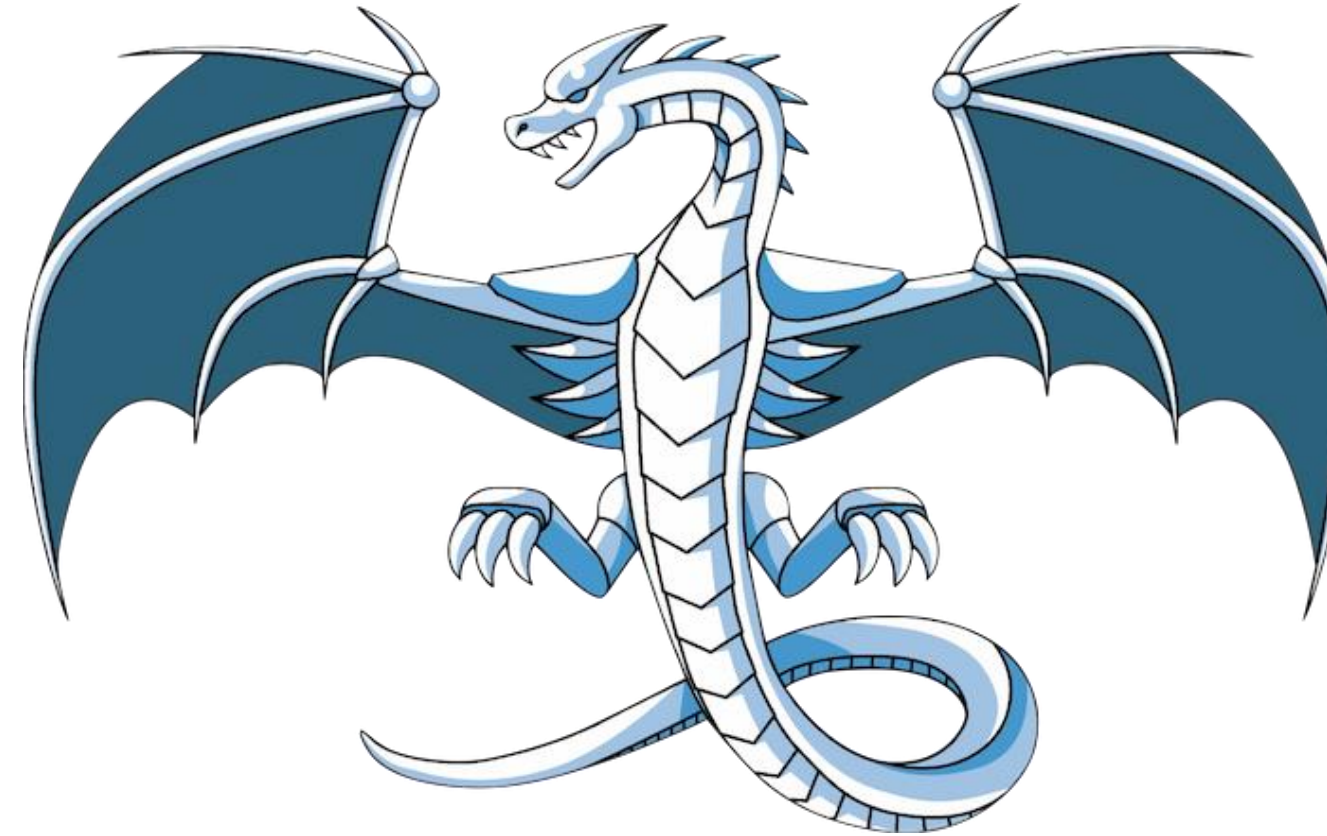
Coherent IRs



Data transformation(s)

Pipelines applied on the different IRs

Passes composing the pipelines



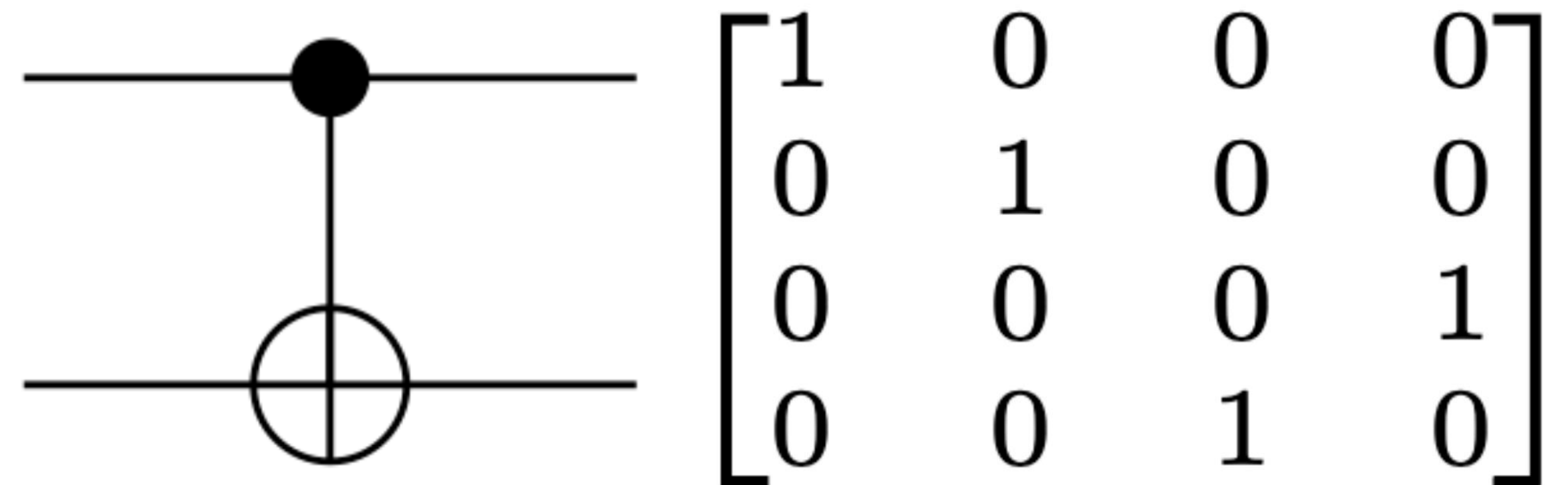
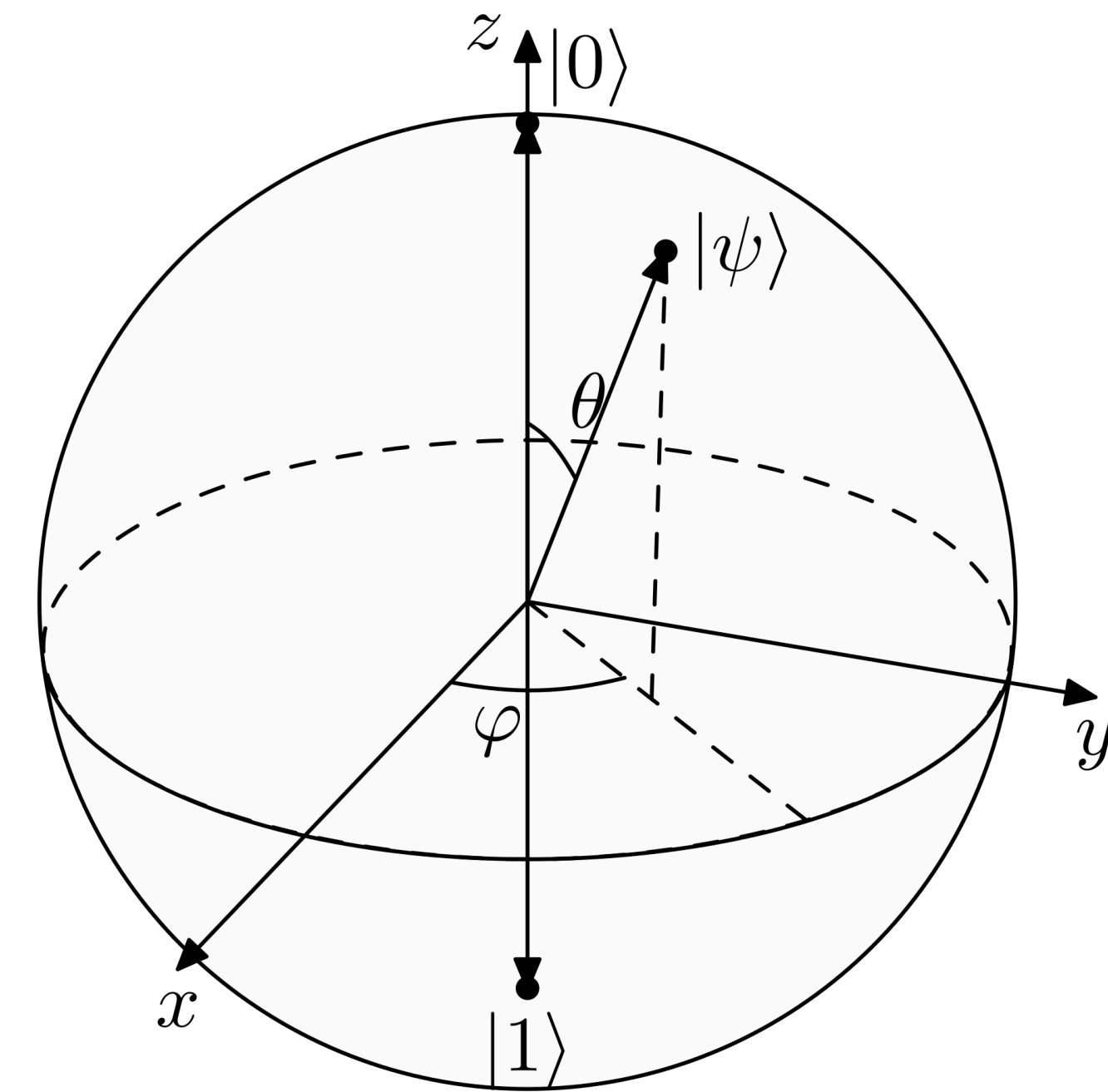
Plan of the talk

- 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

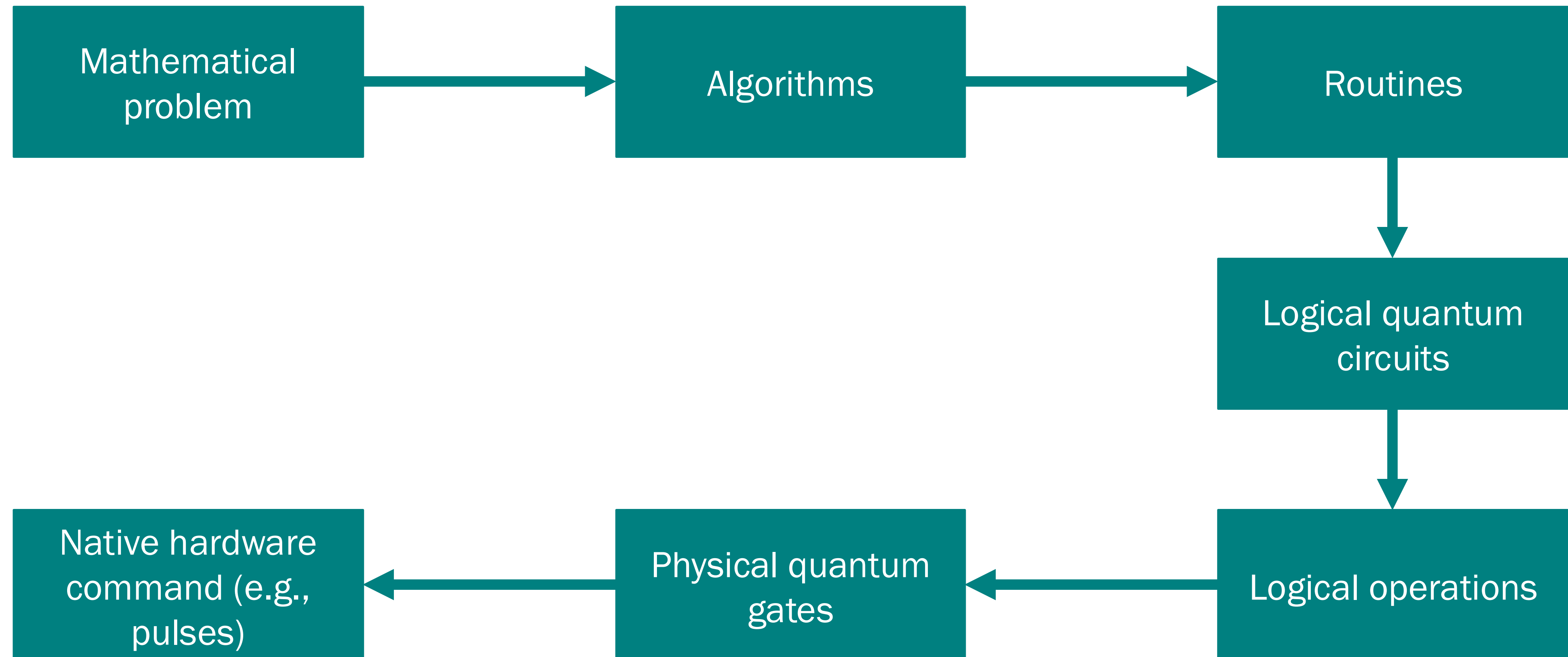
What's an abstraction? A few examples.

```
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;  
}
```



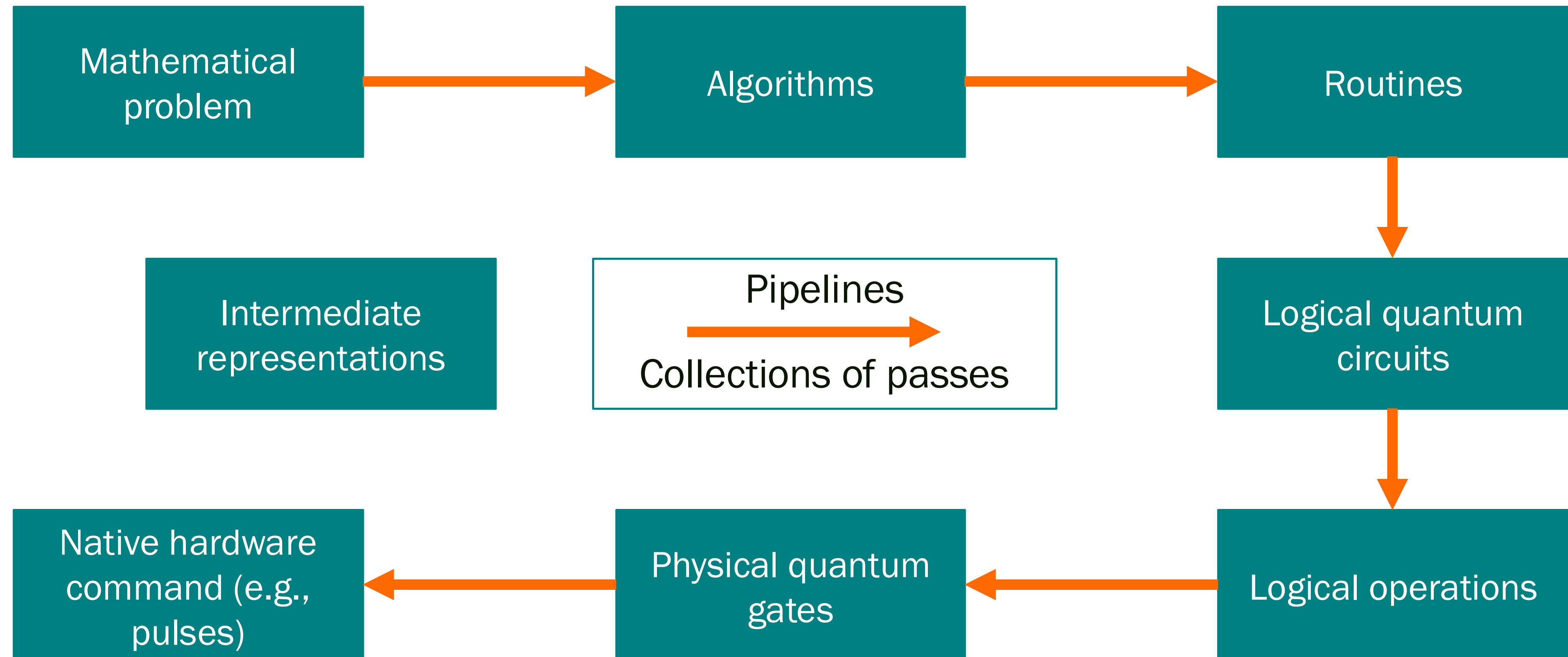
Abstraction levels in quantum computing

Stack of abstractions in QEC



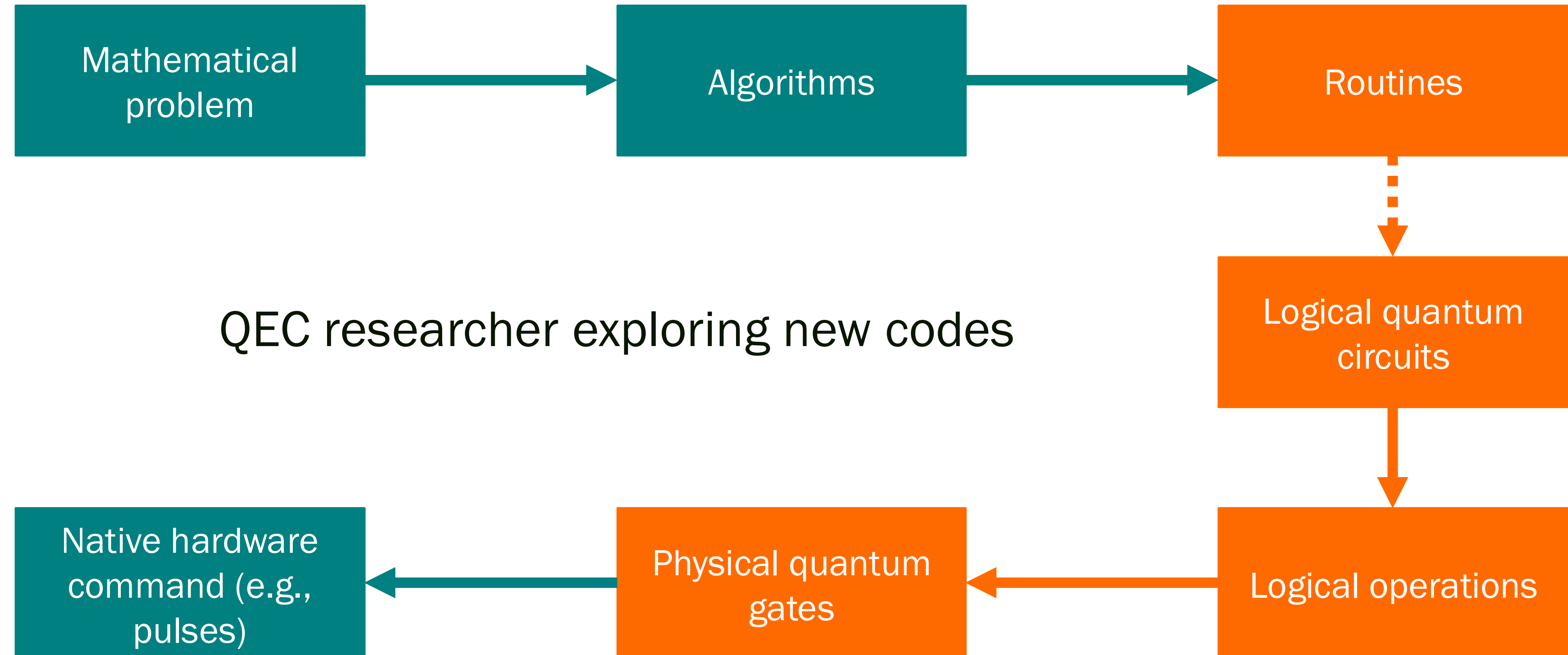
Abstraction levels in quantum computing

That's a big compiler!



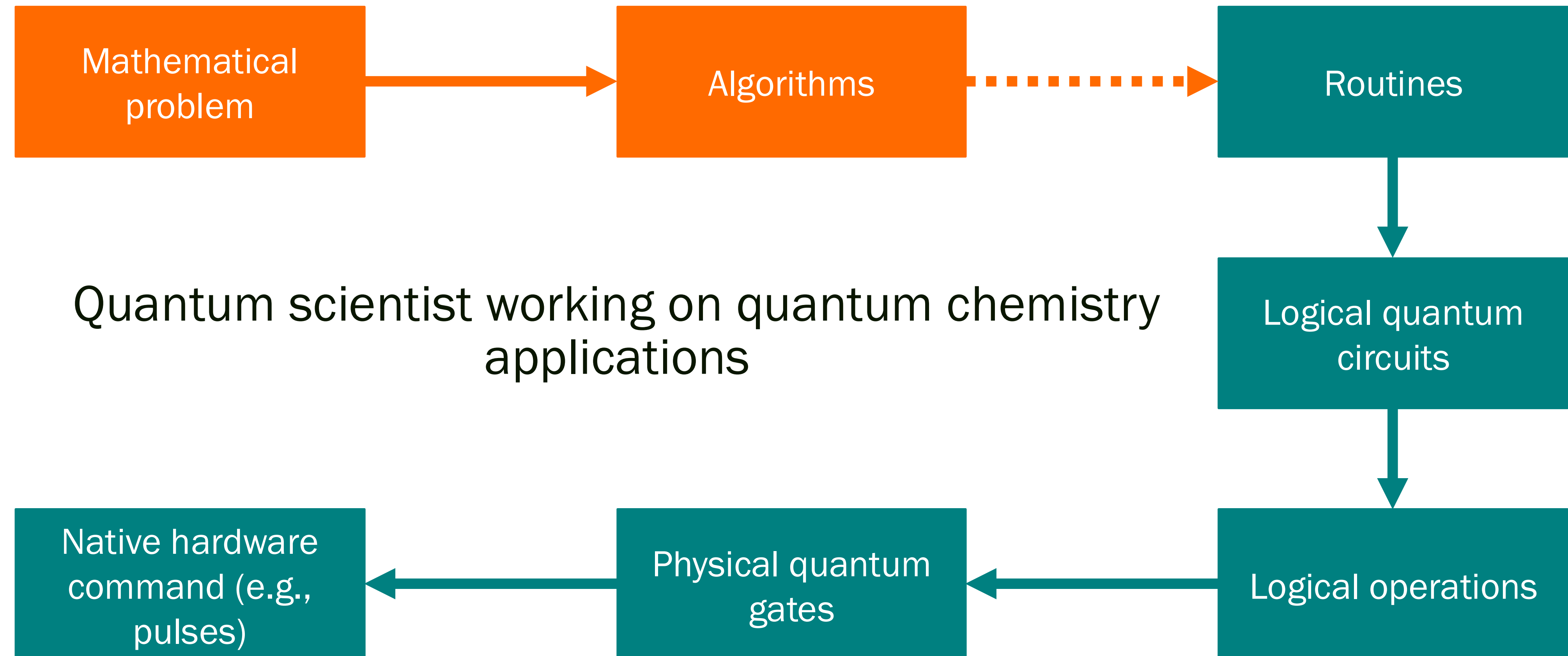
Abstraction levels in quantum computing

How to use abstractions?



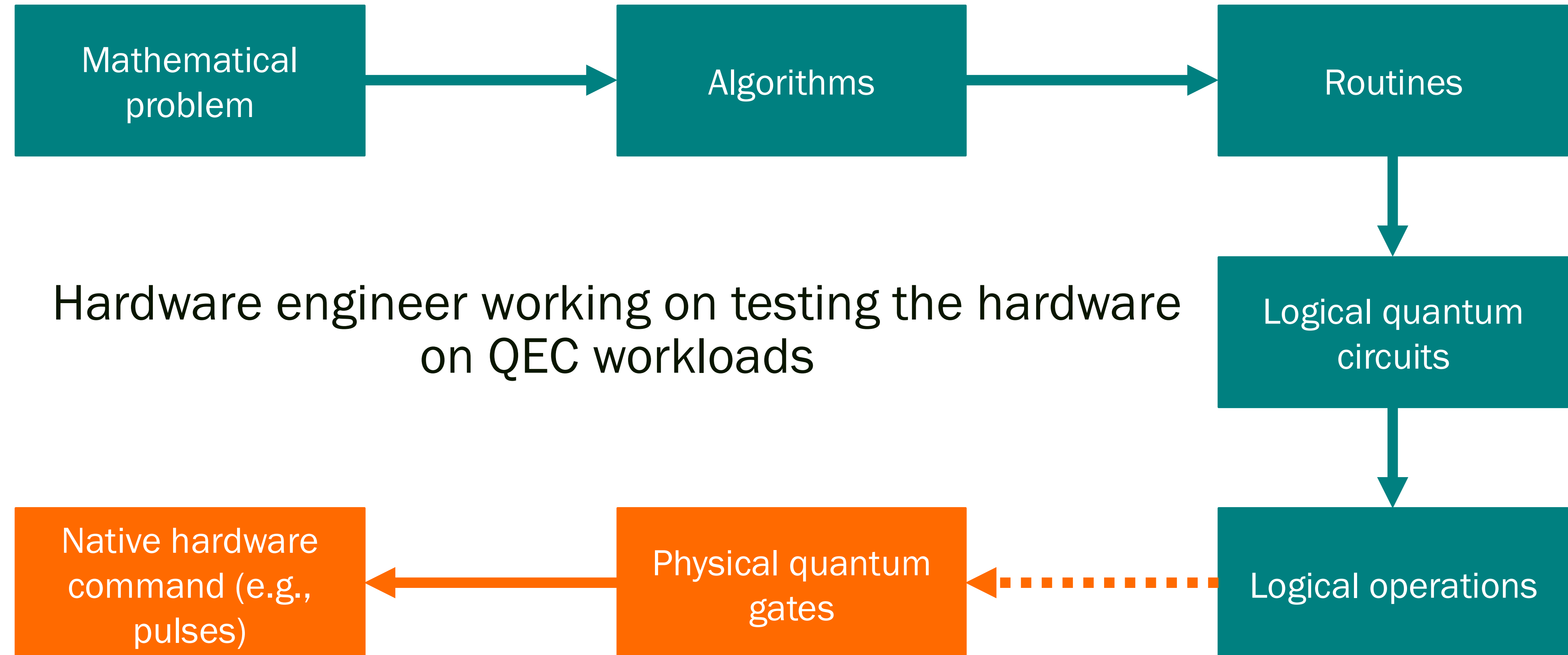
Abstraction levels in quantum computing

How to use abstractions?



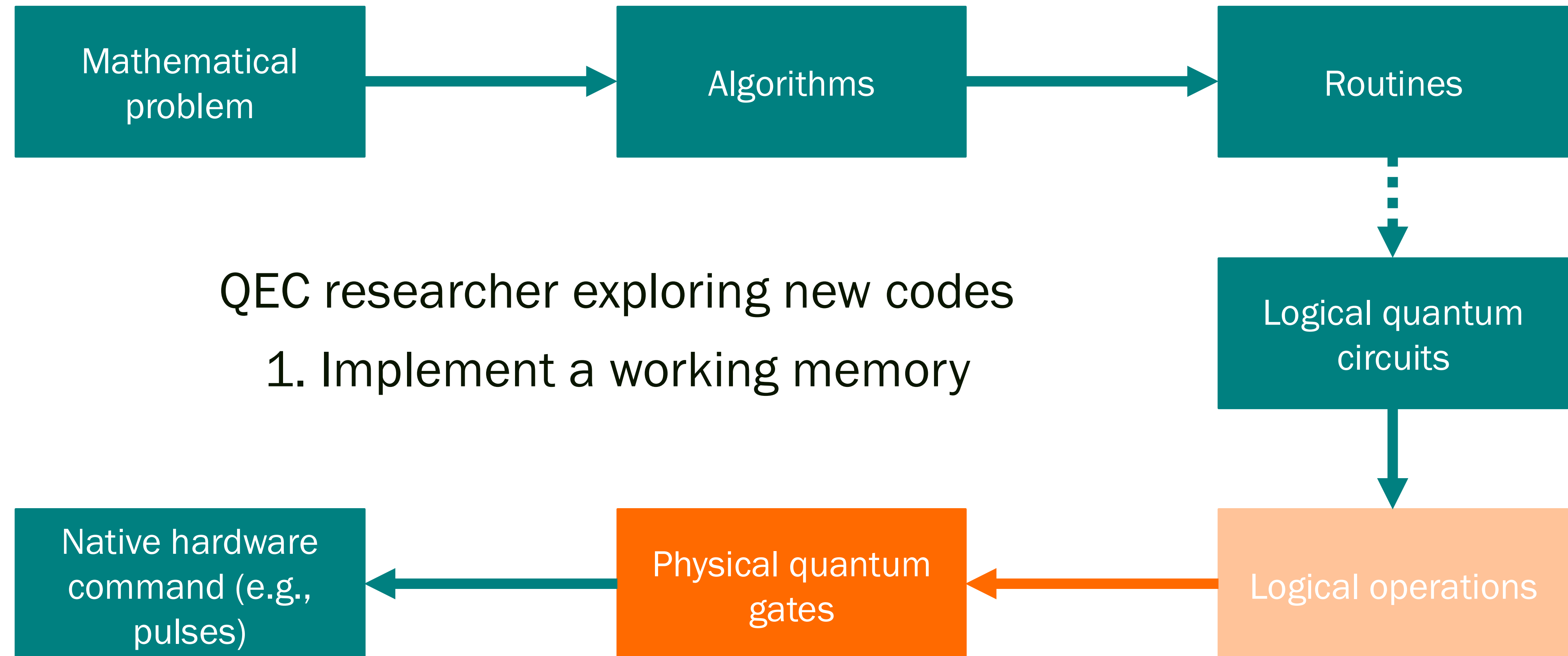
Abstraction levels in quantum computing

How to use abstractions?



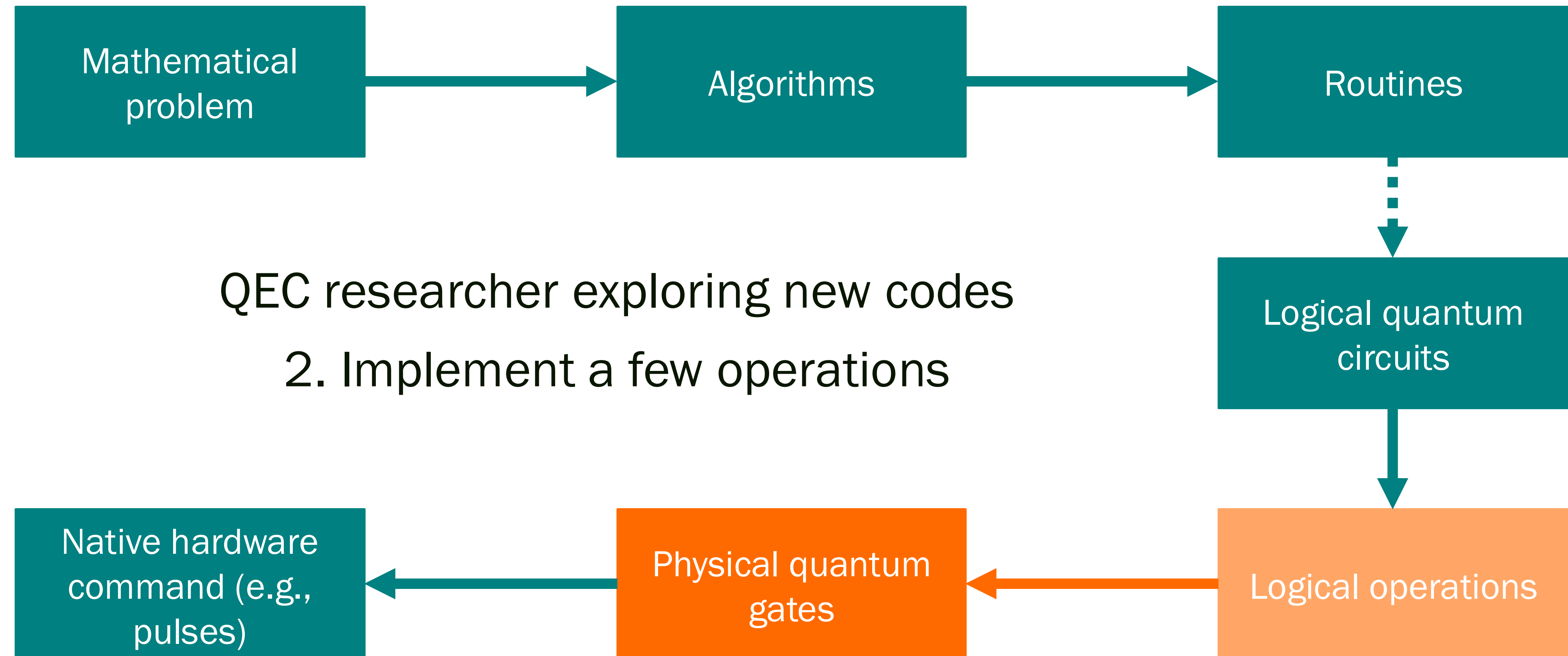
Abstraction levels in quantum computing

How to use abstractions?



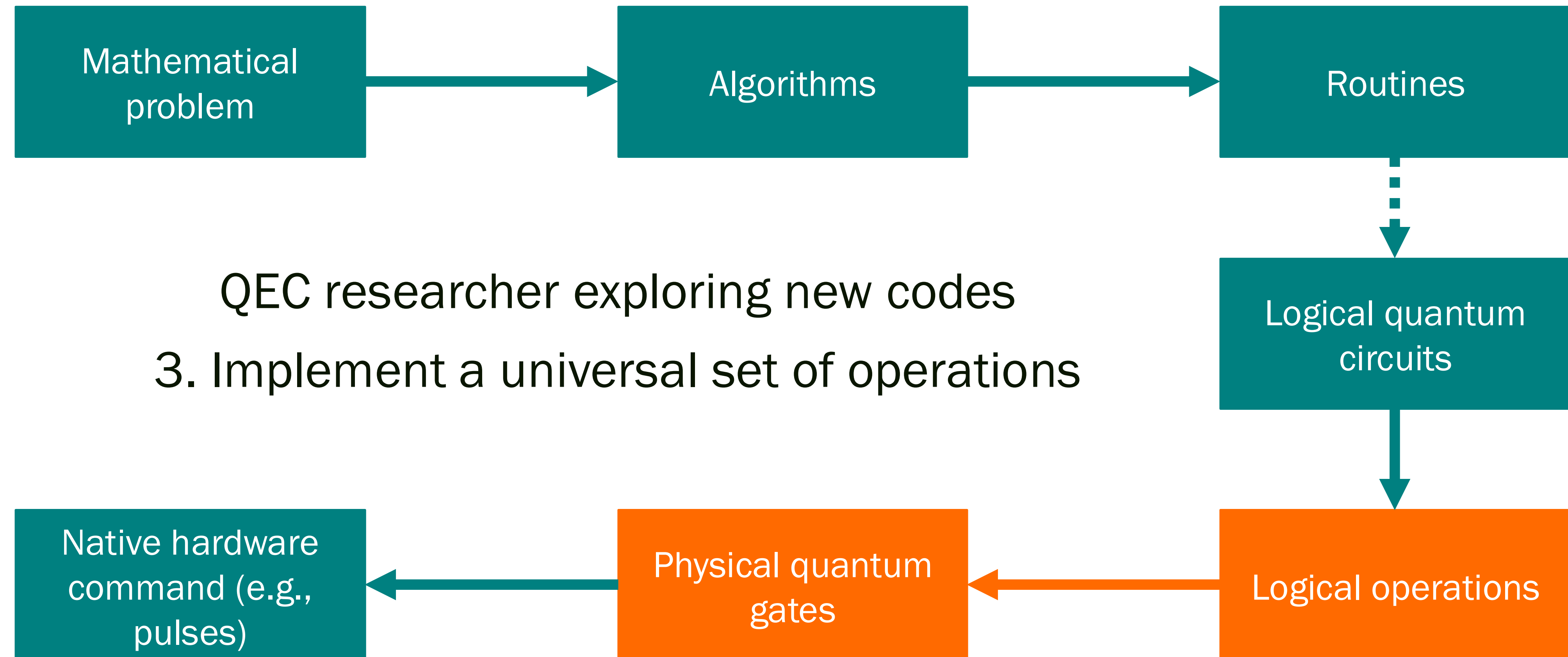
Abstraction levels in quantum computing

How to use abstractions?



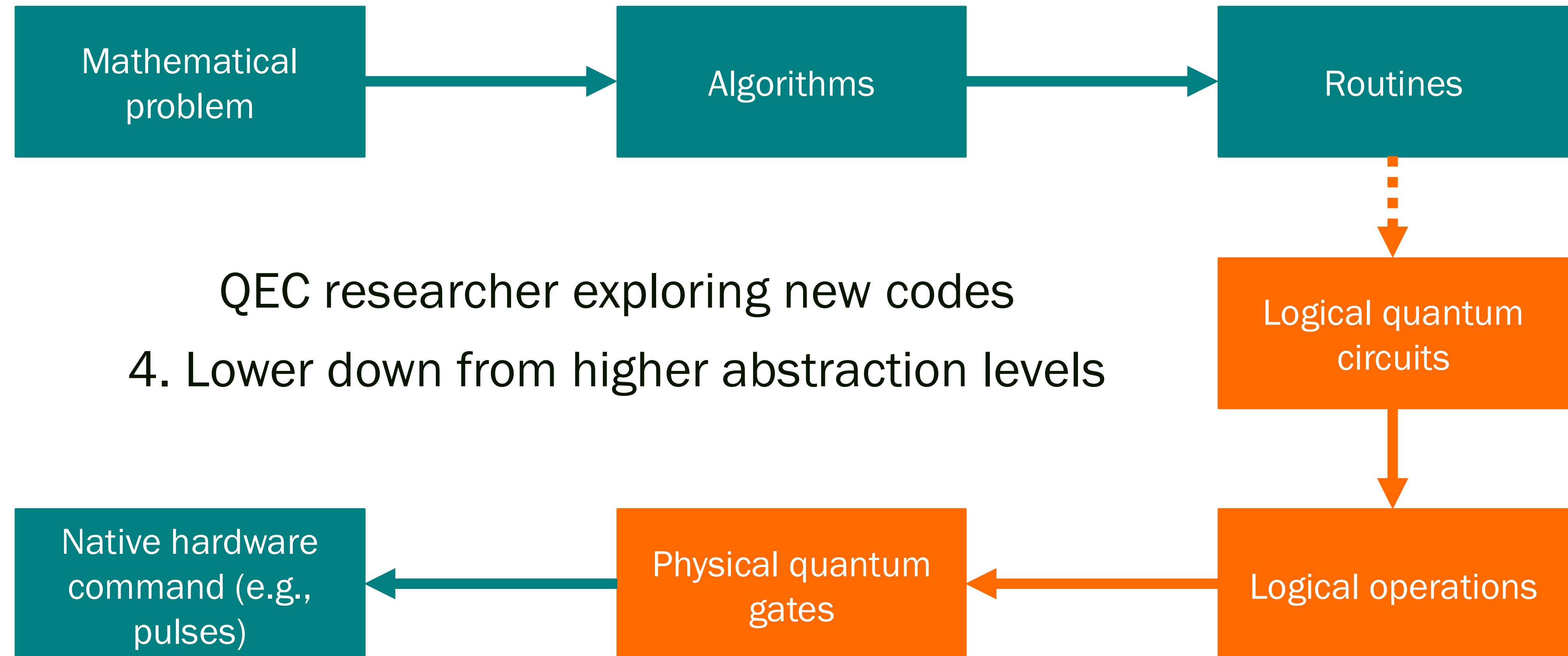
Abstraction levels in quantum computing

How to use abstractions?



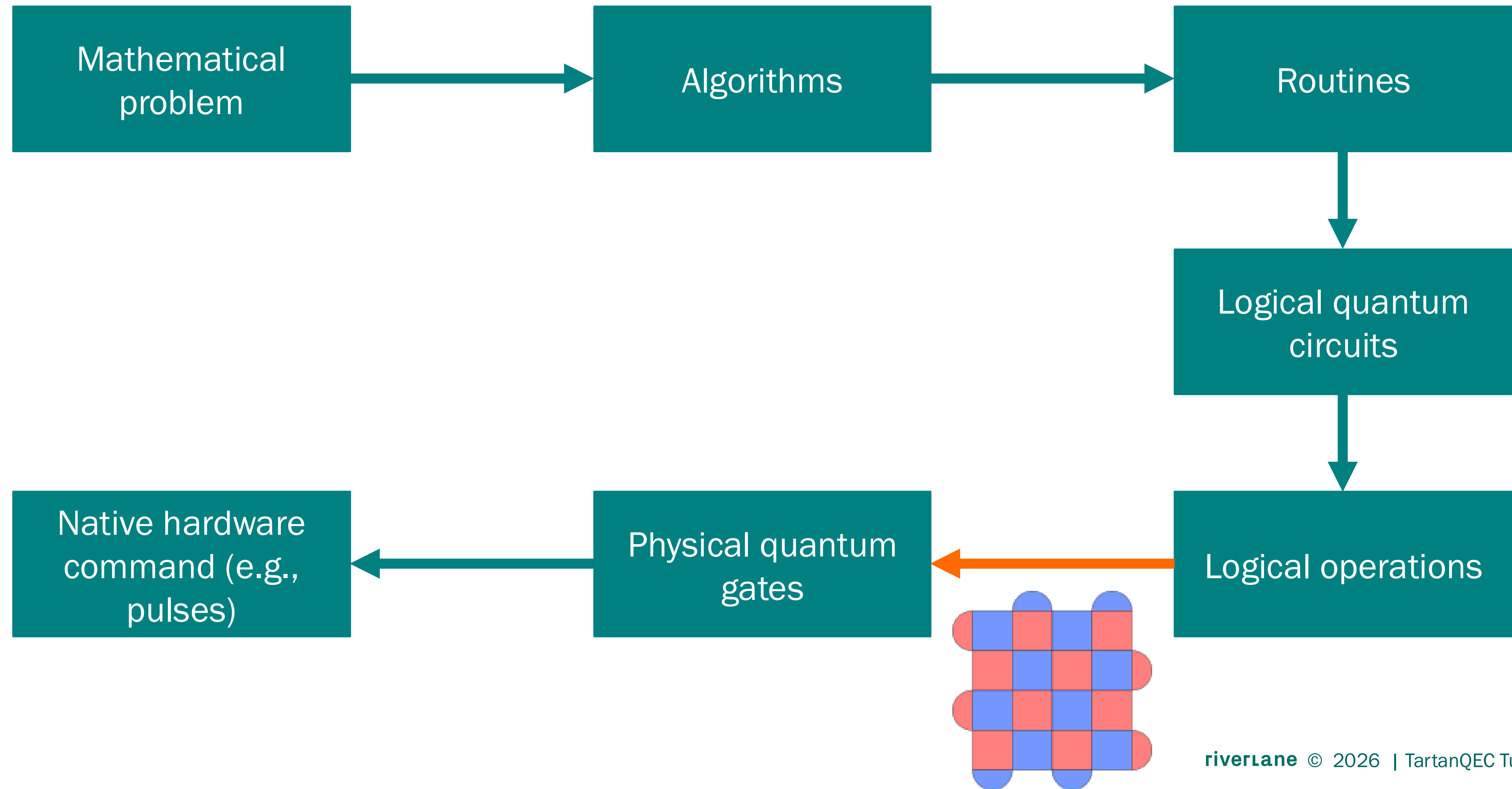
Abstraction levels in quantum computing

How to use abstractions?



Abstraction levels in quantum computing

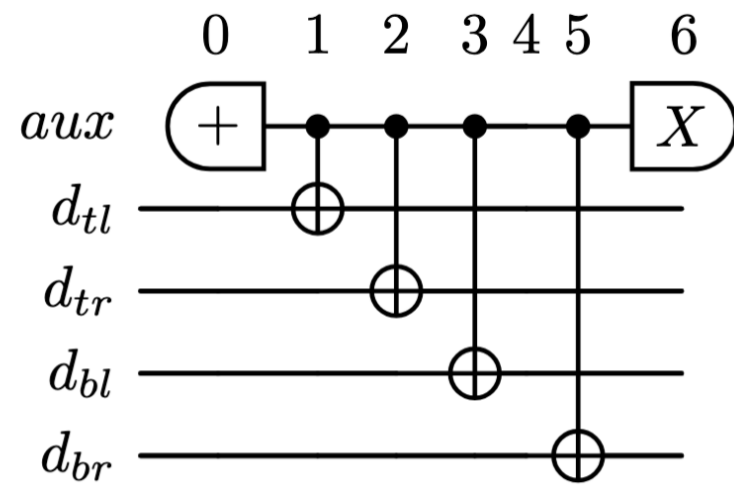
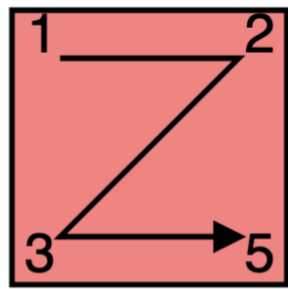
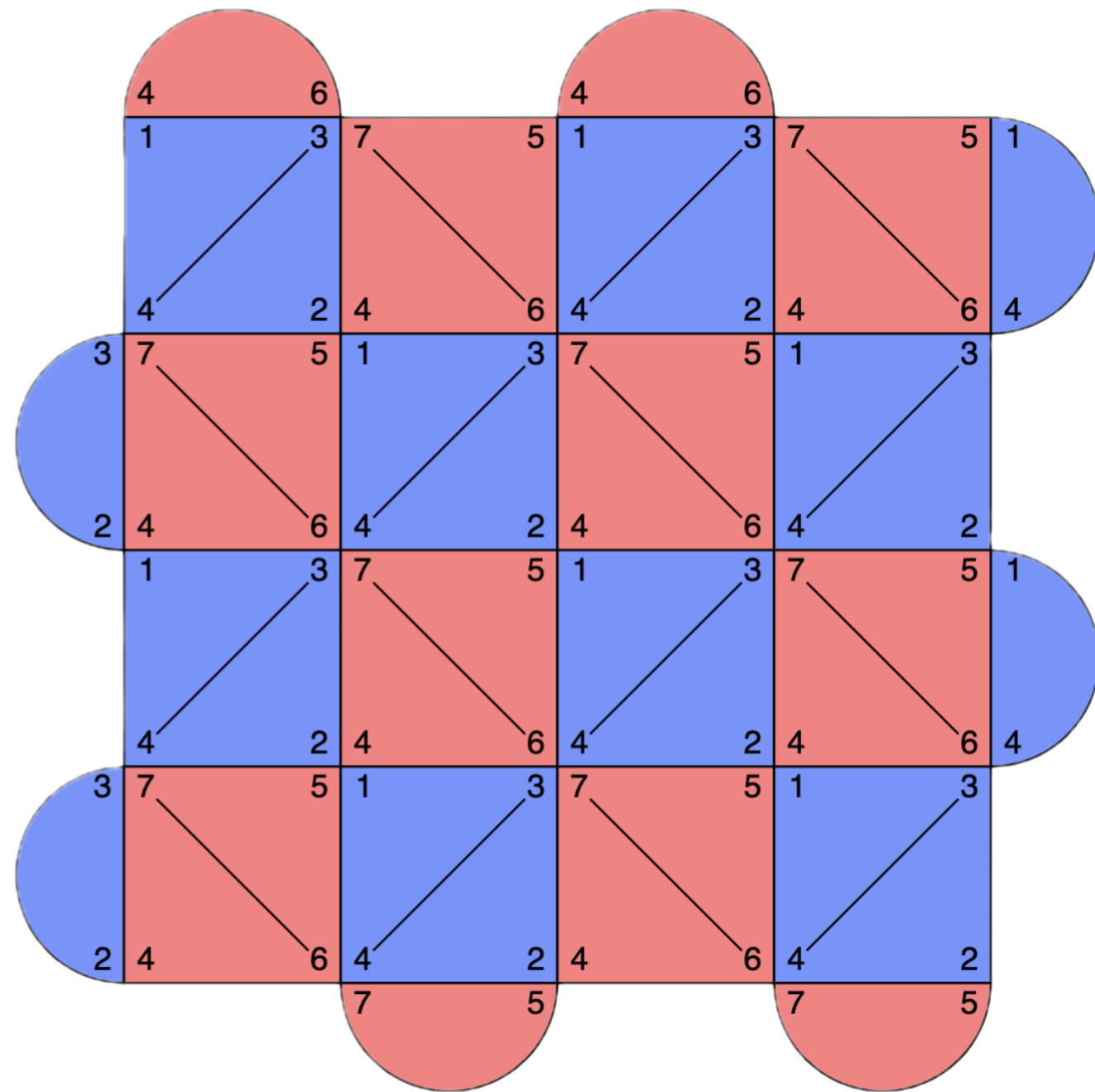
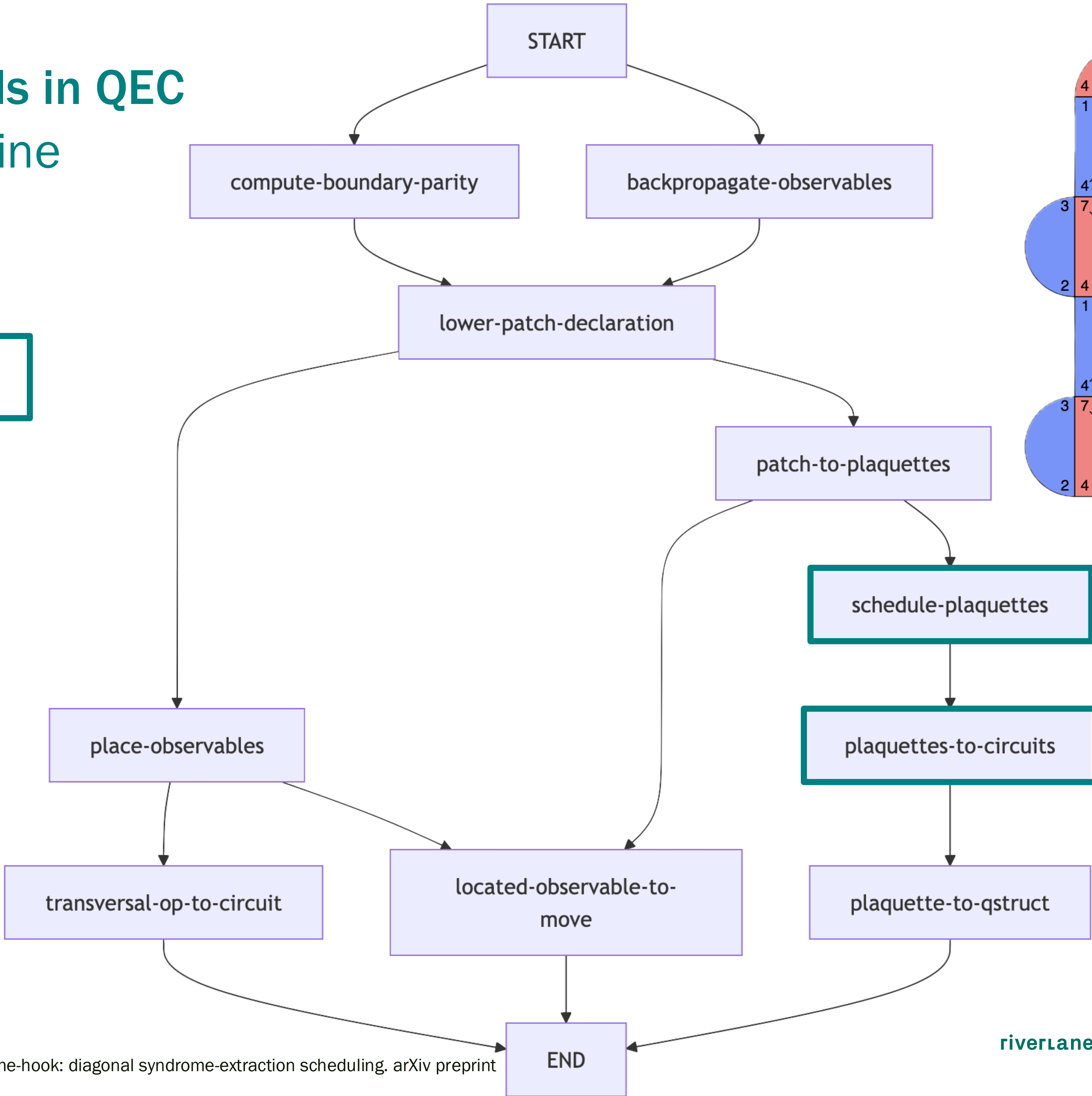
Let's dive in



Abstraction levels in QEC

Changing a pipeline

Small changes

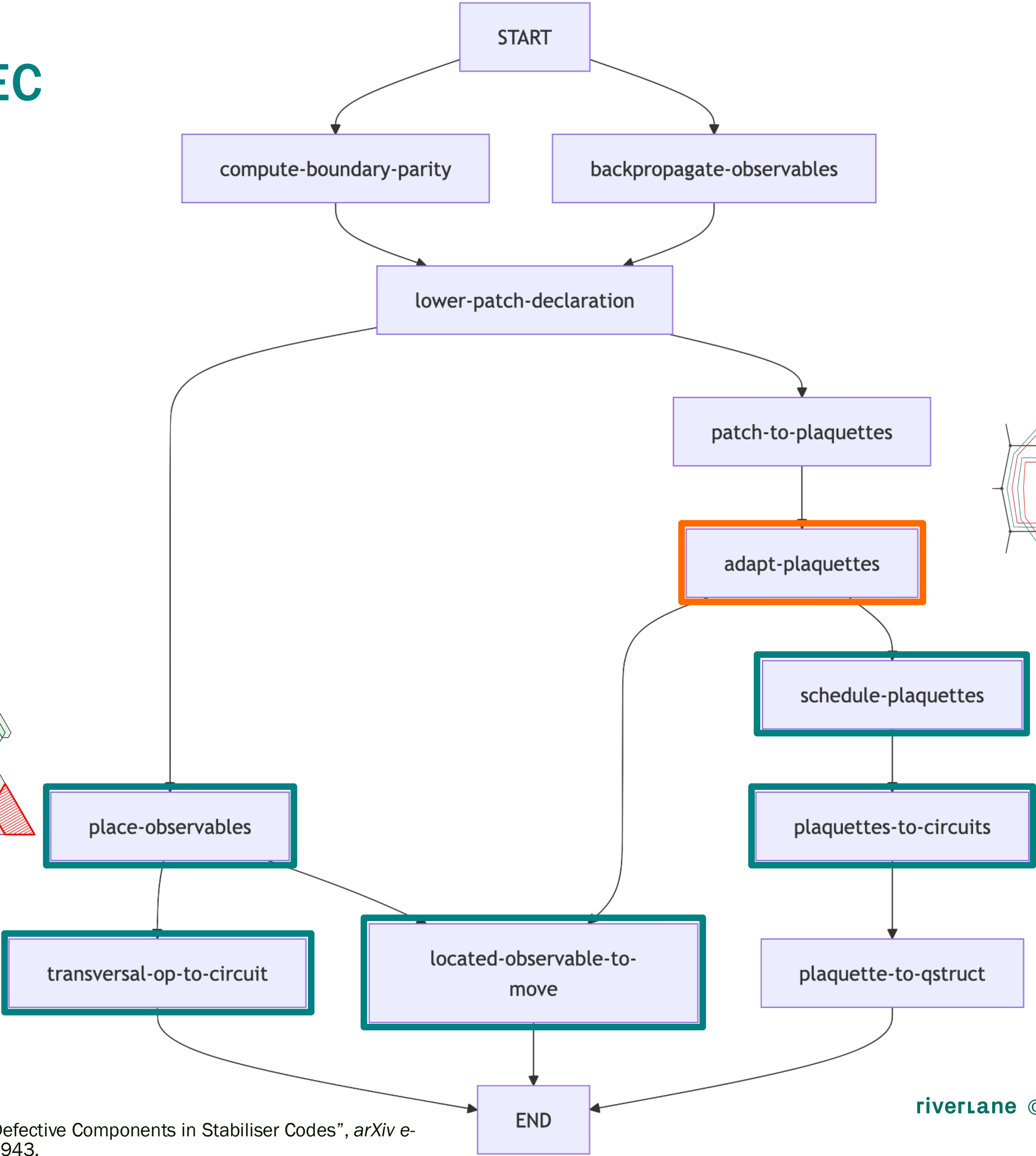
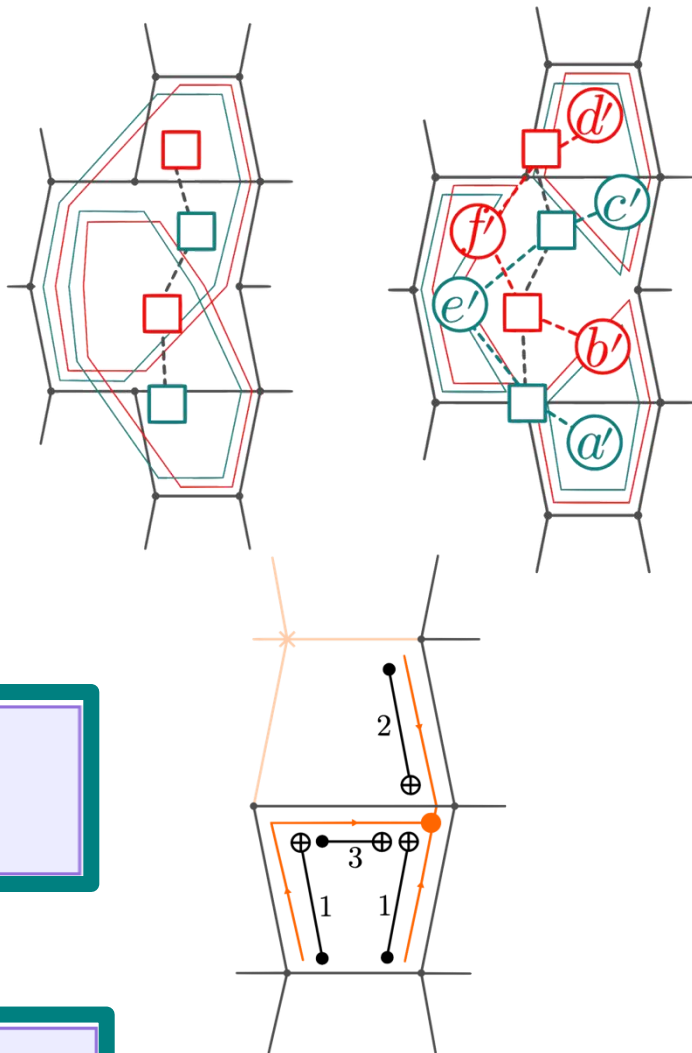
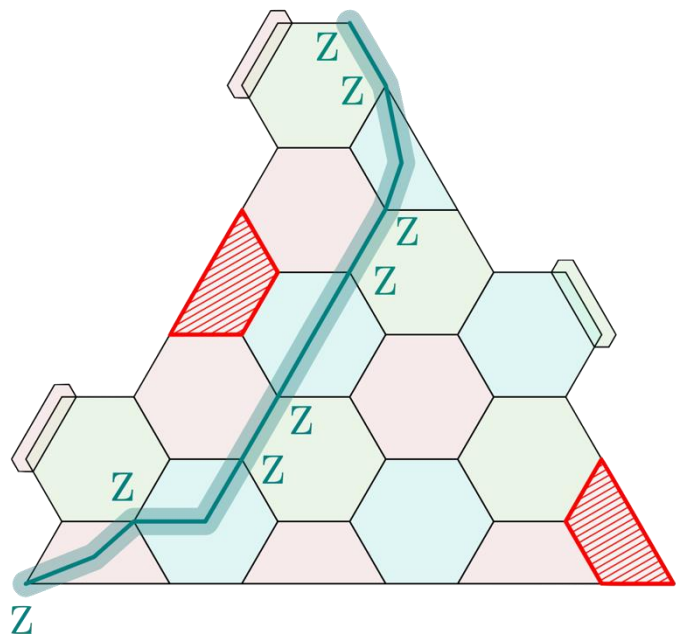


Abstraction levels in QEC

Changing a pipeline

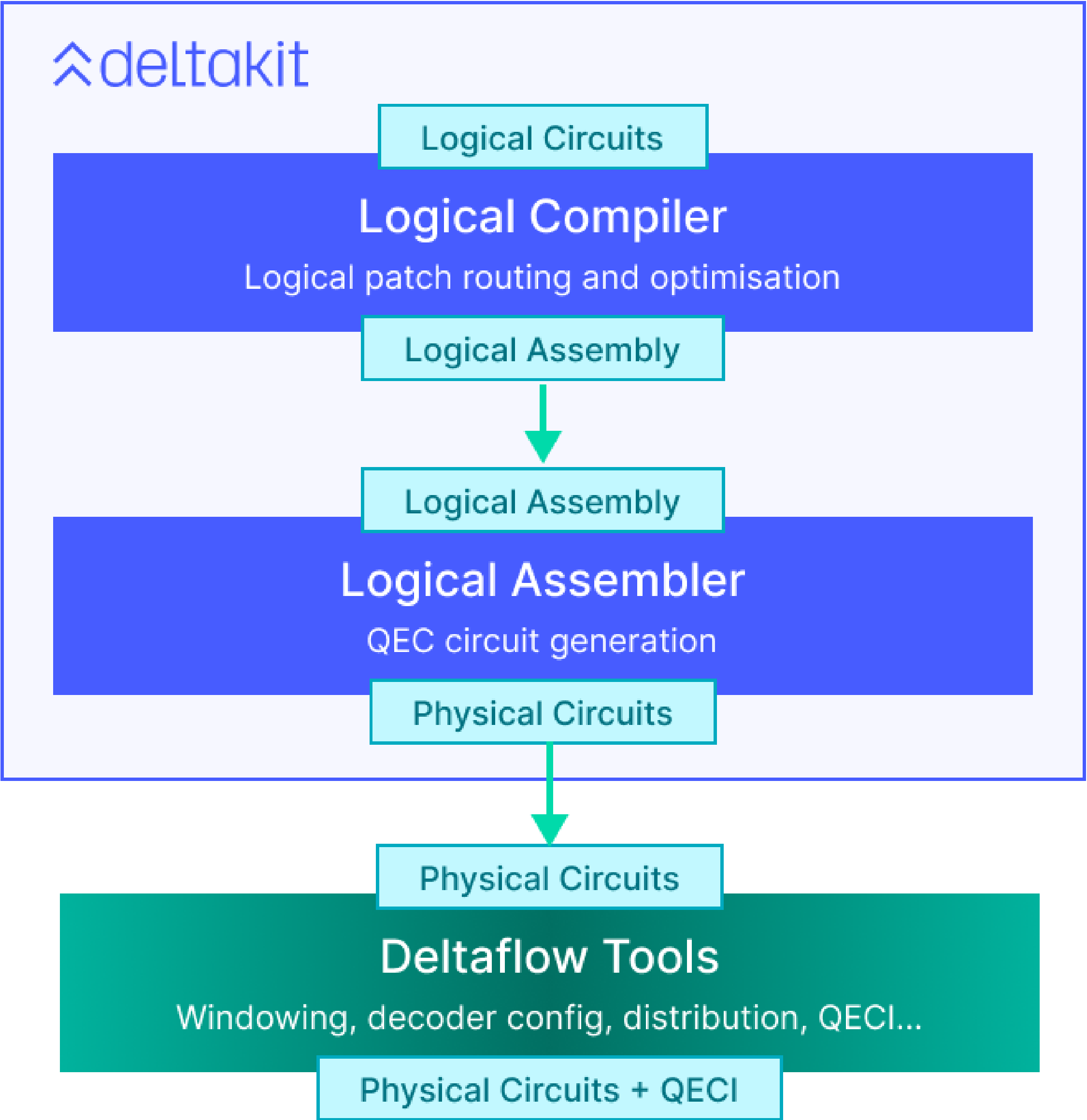
Significant changes

Small changes



Abstraction levels in QEC

Zooming out



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?

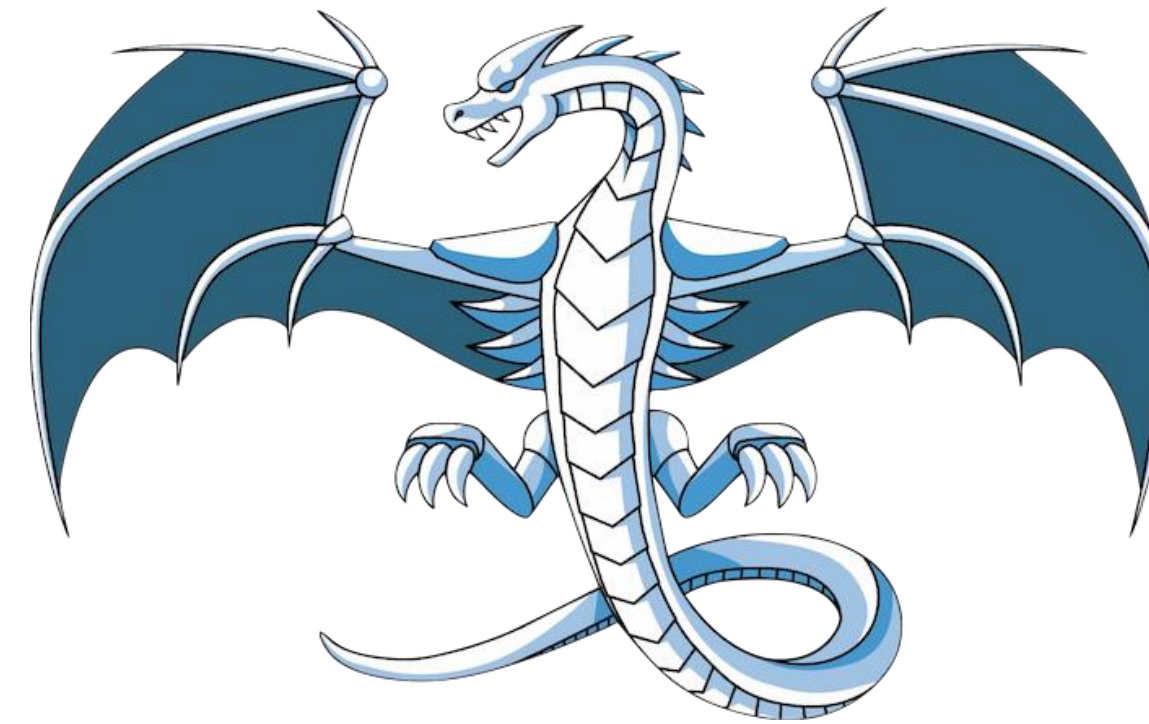
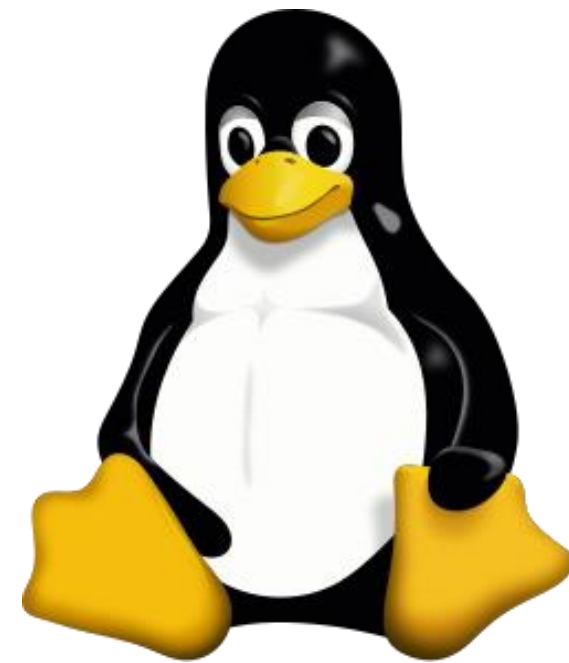
How to benefit from open-source in your work?

One pip install command away

```
pip install \
deltakit bposd qldpc stim sinter stimflow pymatching \
chromobius tesseract-decoder tqec clifford 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



L^AT_EX



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

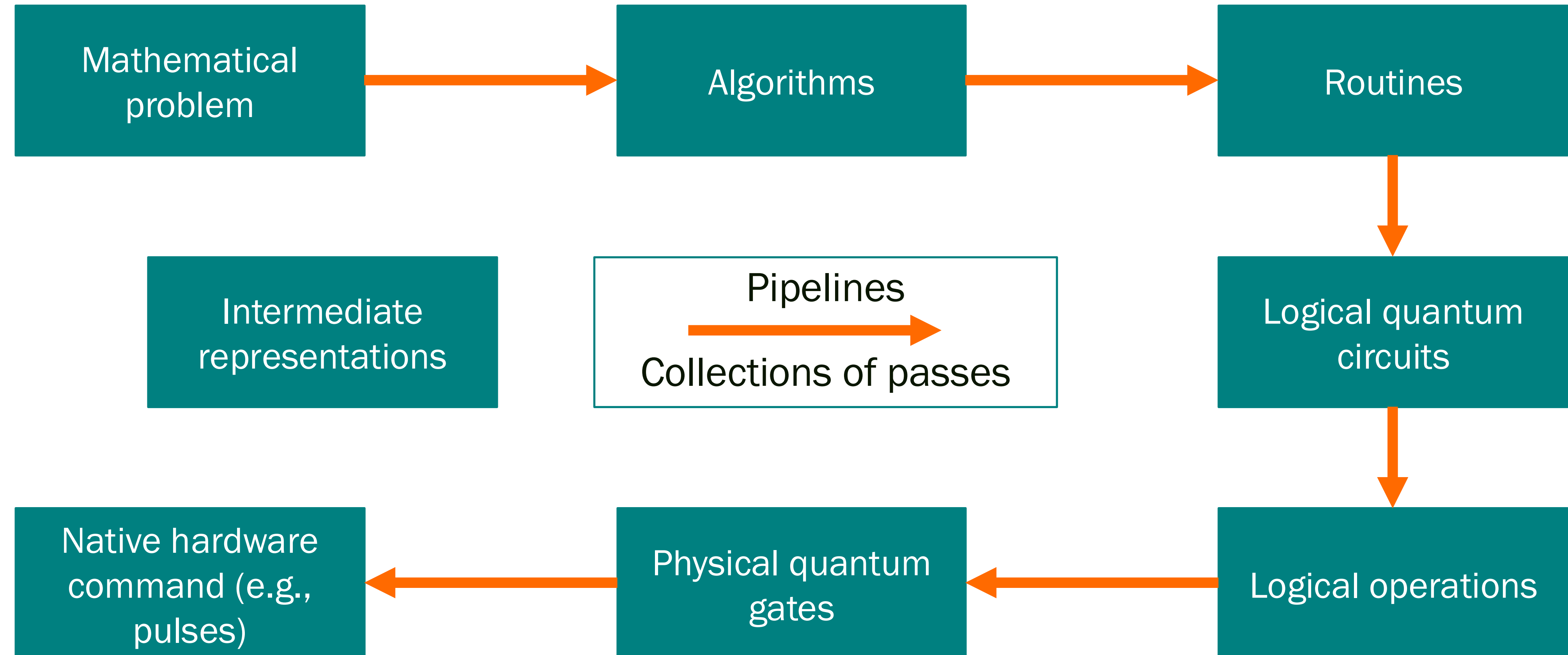
- Publish on GitHub / Gitlab
- Integrate with existing open-source tools
- Use an open-source license
- Take an additional week to clean-up the research code

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

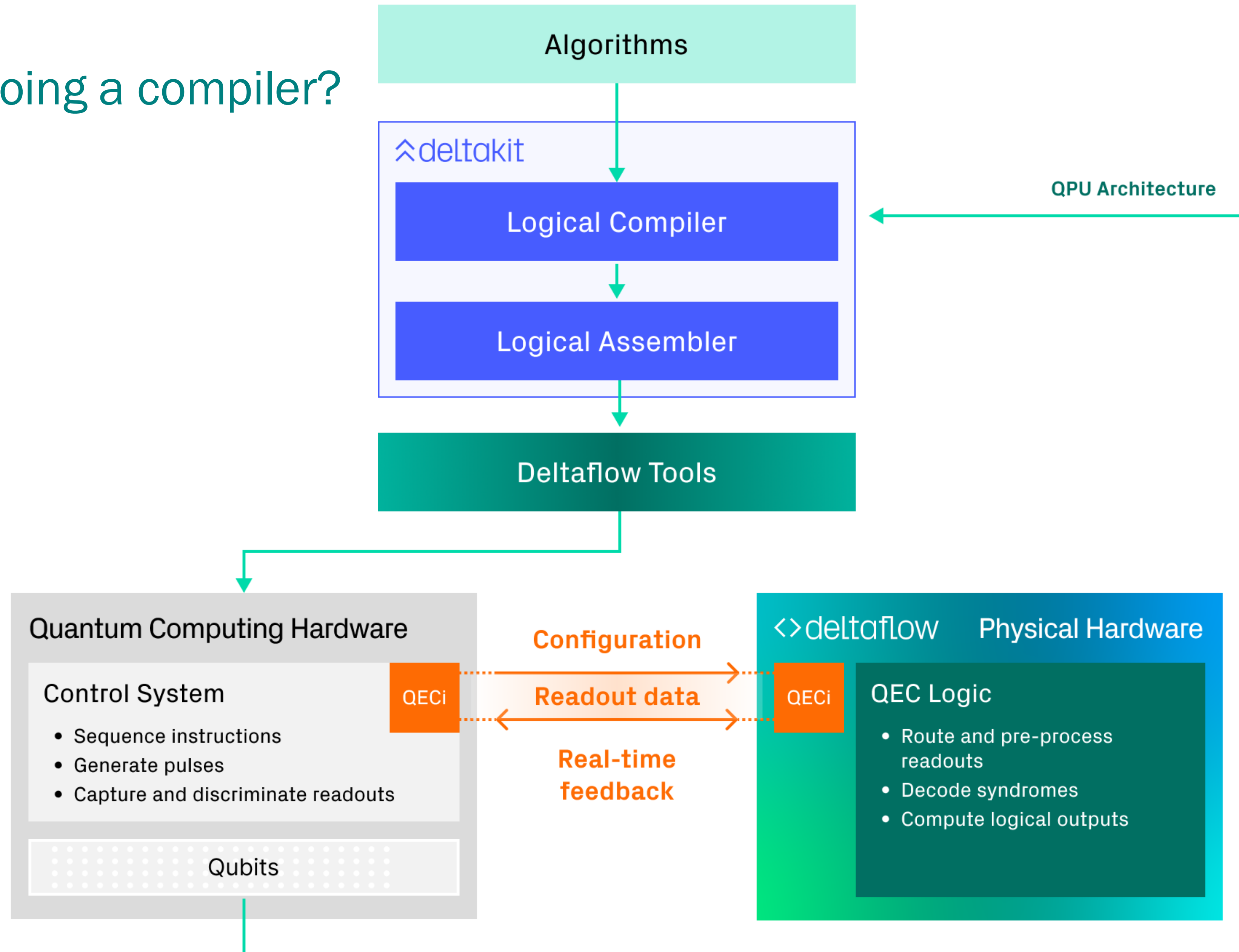
Conclusion

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

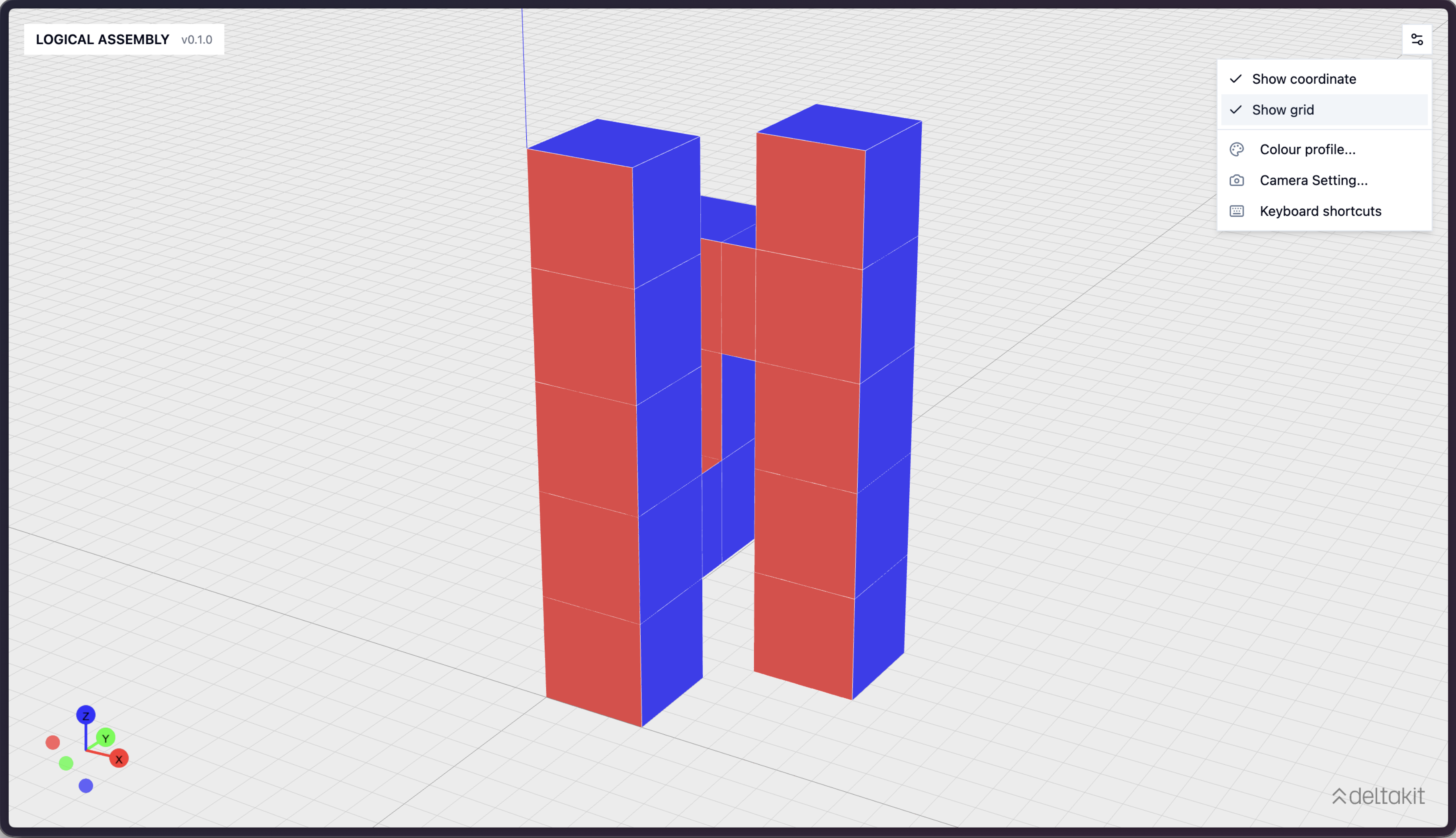
```
1 d = 5
2 builder = LogAsmBuilder()
3 p0 = builder.declare_patch(RotatedPlanarPatch(d, d))
4 p1 = builder.declare_patch(RotatedPlanarPatch(d, d))
5
6 p0.prepare(Z)
7 p1.prepare(Z)
8 p0.measure_stabilisers(8 * d)
9 p1.measure_stabilisers(8 * d)
10
11 b0 = p0.measure(Z)
12 with builder.if_(b0 == 1):
13     p1.transversal(X)
14
15 b1 = p1.measure(Z)
16 builder.add_returns(b0, b1)
17
18 Compiler().compile(builder.build())
```

Circuit

```
1 # Declarations
2 builder = CircuitBuilder()
3 qubits = builder.add_args(QubitReg(9))
4 recs = [builder.declare_record() for _ in range(4)]
5 # Circuit
6 builder.gate("R", qubits)
7 builder.gate("CX", qubits[:-1])
8 builder.gate("CX", [qubits[i] for i in (2, 1, 4, 3, 6, 5, 8, 7)])
9 readouts = builder.measure(Pauli.Z, [qubits[i] for i in (1, 3, 5, 7)])
10 # Records and detectors
11 ✓ for i in range(4):
12     recs[i].append(readouts[i])
13 repetition_5_prep = builder.build()
14
15 logical_builder = LogAsmBuilder()
16 qreg = logical_builder.add_arg(QubitReg(9))
17 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

