How to Write a Scalable Compiler for an Error-Prone Quantum Computer

Kim Worrall
kim.worrall@ed.ac.uk
University of Edinburgh

Samin Ishtiaq
samin.ishtiaq@riverlane.com
Riverlane

April 16, 2025

Quantum Computing from a Compiler Perspective

Quantum Computing from a Compiler Perspective

- Cirq
- Qiskit
- Q#
- Braket
- ...

Languages

Quantum Computing from a Compiler Perspective

Cirq | Qiskit | Q# | Braket | ...

Languages

Qubits

Superconducting | Photonic | Trapped Ion | ...

[Google Quantum AI]

Quantum Computing from a Compiler Perspective

Cirq | Qiskit | Q# | Braket | …

Languages

IR?

Qubits

Superconducting | Photonic | Trapped Ion | …

[Google Quantum AI]

Quantum Computing from a Compiler Perspective

Languages

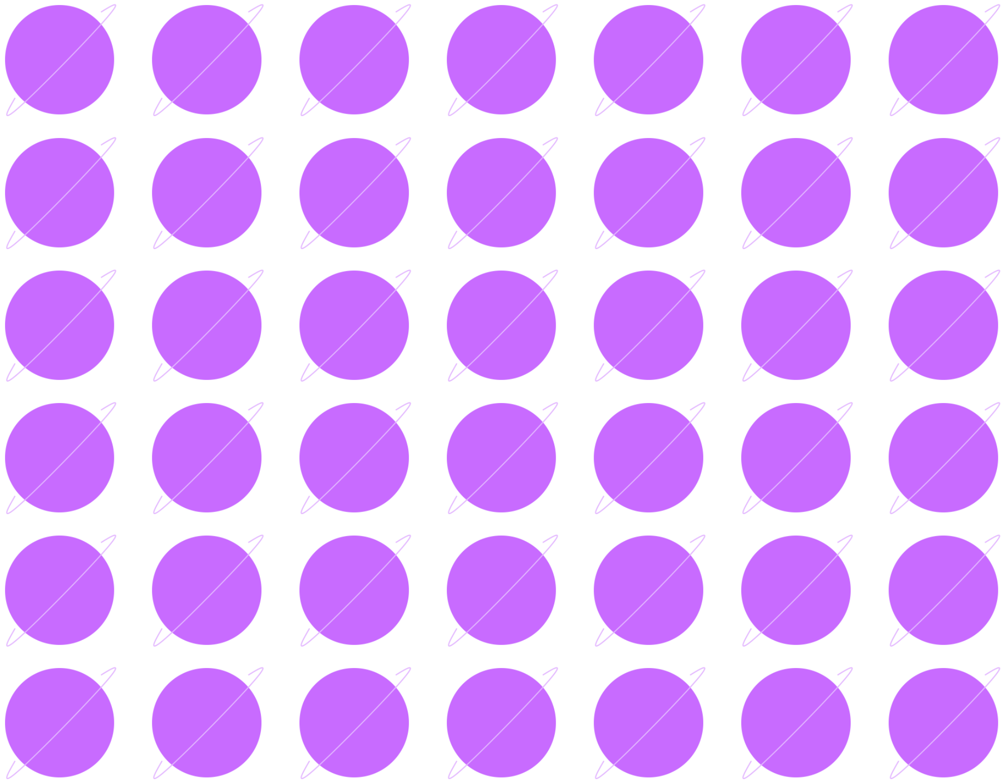
Qubit

[Google Quantum AI]

Qubits

Quantum Computing from a Compiler Perspective

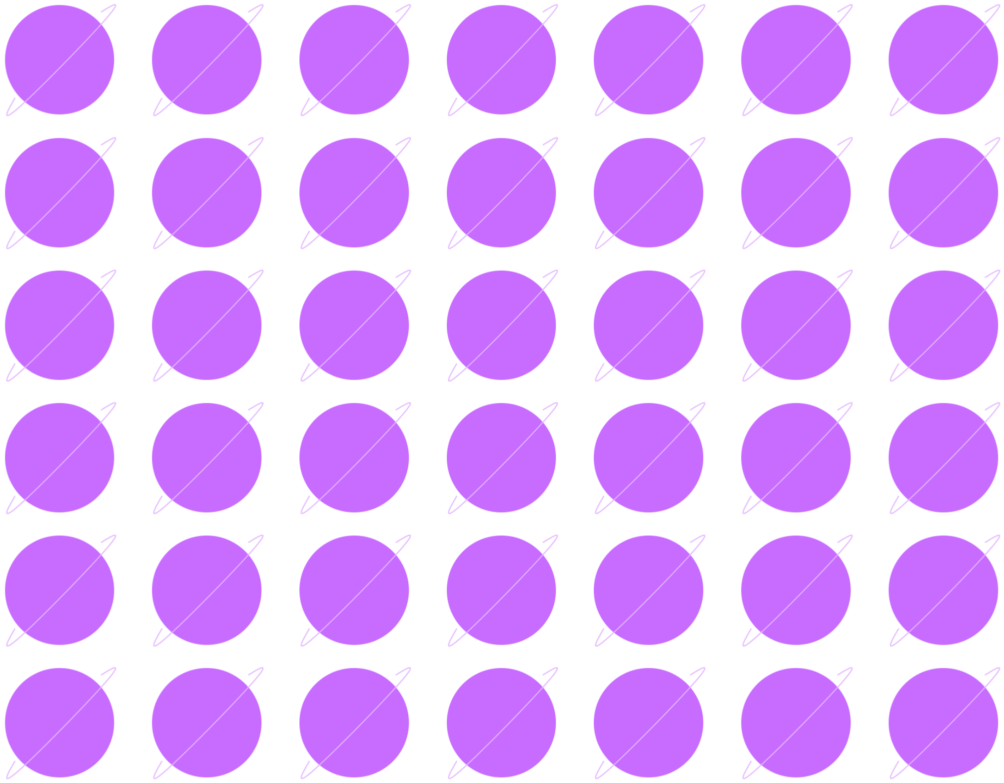
Languages



Qubits

Quantum Computing from a Compiler Perspective

Languages

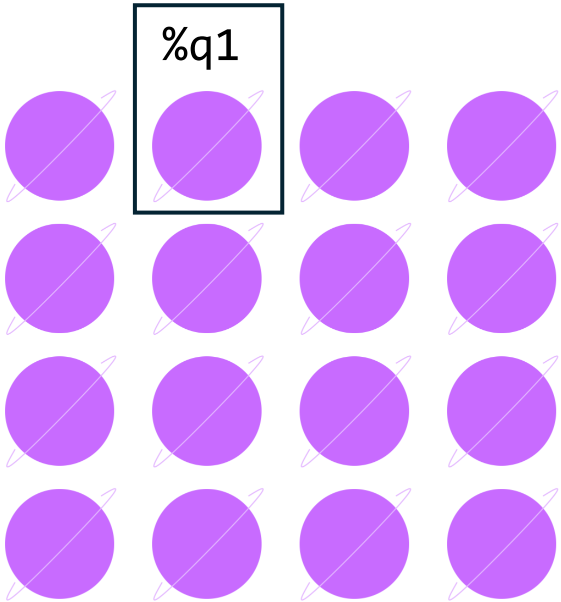


+ Control System

Qubits

Quantum Computing from a Compiler Perspective

Languages




```
quantum.circuit {
    %q1 = quantum.alloc<1> {"state" = 0}



} : () -> ()
```

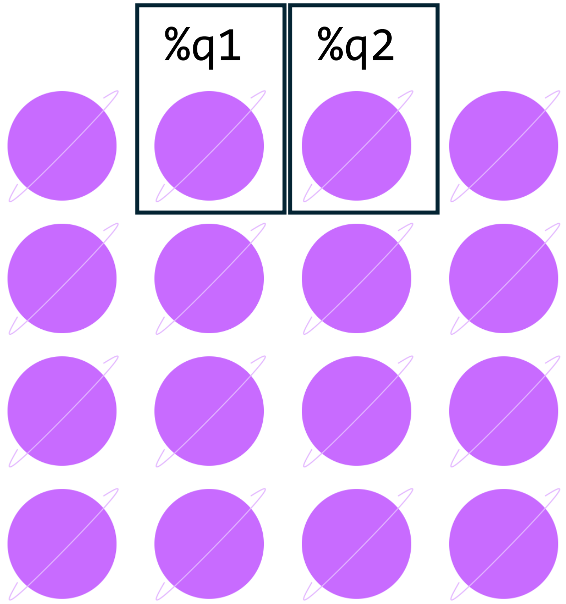
Qubit reference

Control System

Qubits

Quantum Computing from a Compiler Perspective

Languages




```
quantum.circuit {
    %q1 = quantum.alloc<1> {"state" = 0}
    %q2 = quantum.alloc<1> {"state" = 1}


} : () -> ()
```

Control System

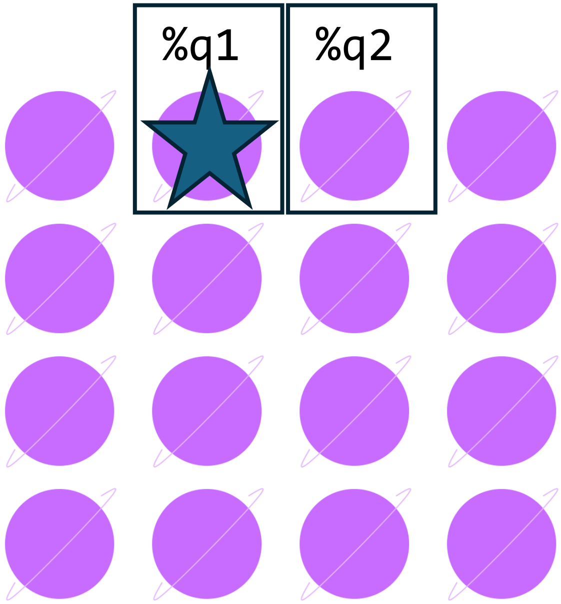
Qubits

Quantum Computing from a Compiler Perspective

Languages

Control System

Qubits



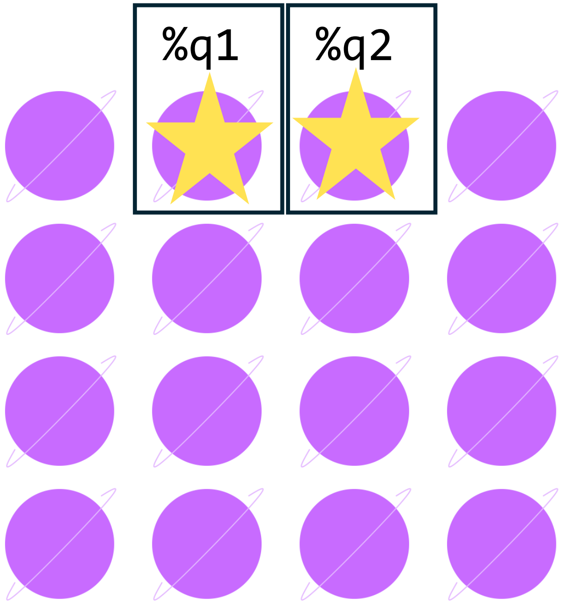

```
quantum.circuit {
    %q1 = quantum.alloc<1> {"state" = 0}
    %q2 = quantum.alloc<1> {"state" = 1}
    quantum.gate <#quantum.TGate> (%q1)
```

Act on qubit, changing probability of measuring 1 or 0

```
} : () -> ()
```

Quantum Computing from a Compiler Perspective

Languages




```
quantum.circuit {
    %q1 = quantum.alloc<1> {"state" = 0}
    %q2 = quantum.alloc<1> {"state" = 1}
    quantum.gate <#quantum.TGate> (%q1)
    quantum.gate <#quantum.CNOT> (%q1, %q2)
} : () -> ()
```

Interact two qubits

Control System

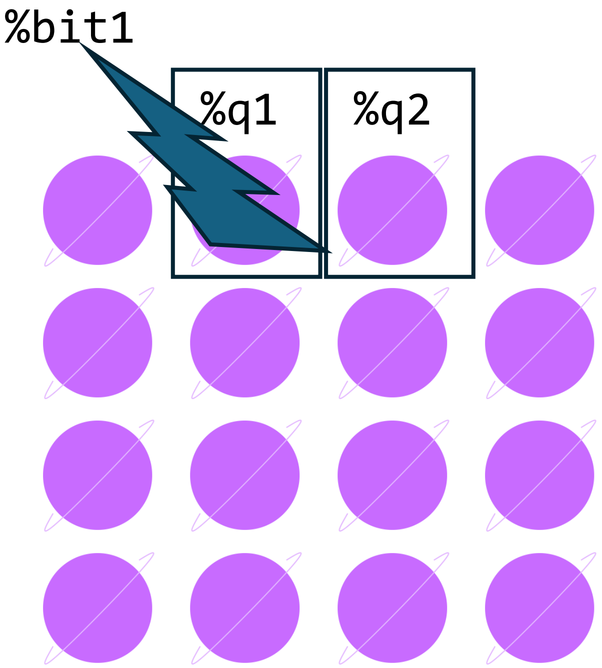
Qubits

Quantum Computing from a Compiler Perspective

Languages

Control System

Qubits

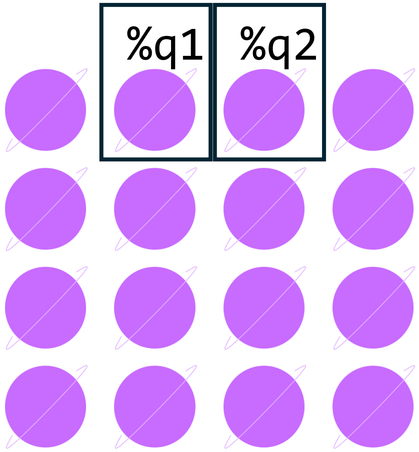



```
quantum.circuit {
    %q1 = quantum.alloc<1> {"state" = 0}
    %q2 = quantum.alloc<1> {"state" = 1}
    quantum.gate <#quantum.TGate> (%q1)
    quantum.gate <#quantum.CNOT> (%q1, %q2)
    %bit1 = quantum.measure (%q1)
} : () -> ()
```

Measure a qubit and get a bit

Quantum Physics and Intermediate Representations

Languages



- No Cloning
  - Can swap values

```
quantum.gate <#quantum.CNOT> (%q1, %q2)
```

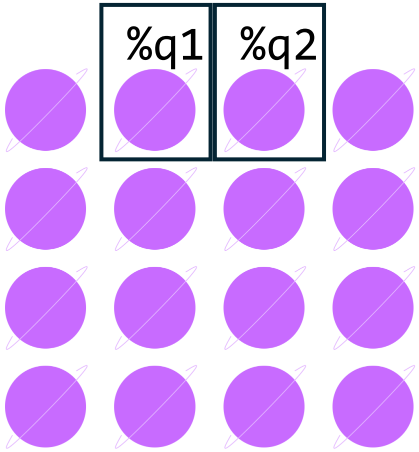
Not allowed to be equal

Control System

Qubits

Quantum Physics and Intermediate Representations

Languages



Control System

Qubits

- No Cloning
  - Can swap values
- Physical Connectivity Matters

```
quantum.gate <#quantum.CNOT> (%q1, %q2)
```

Must be physically next to each other after register allocation.

Quantum Physics and Intermediate Representations

Lan

Your memory *is* your computation space!

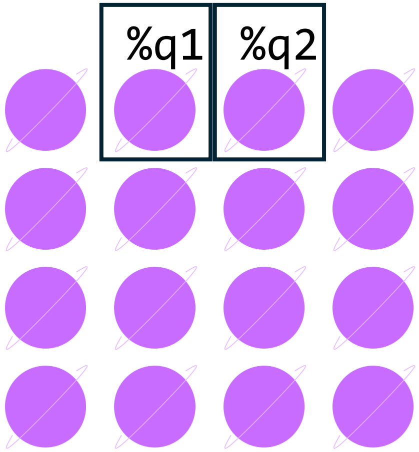
Cont

ion.

Qubits

Quantum Physics and Intermediate Representations

Languages



- No Cloning
  - Can swap values
- Physical Connectivity Matters
- Measurement is ‘final’

```
%bit1 = quantum.measure (%q1)
%bit2 = quantum.measure (%q1)
```

No other gates => %bit1 = %bit2

Control System

Qubits

Quantum Physics and Intermediate Representations

Languages

- QSSA for optimisations

New qubit value after using a qubit

```
qssa.circuit {
    %q1 = qssa.alloc<1> {"state" = 0}
    %q2 = qssa.alloc<1> {"state" = 0}
    %q3 = qssa.gate <#quantum.TGate> (%q1)
    %q4, %q5 = qssa.gate <#quantum.CNOT> (%q3, %q2)
    %bit1 = qssa.measure (%q4)
} : () -> ()
```

Control System

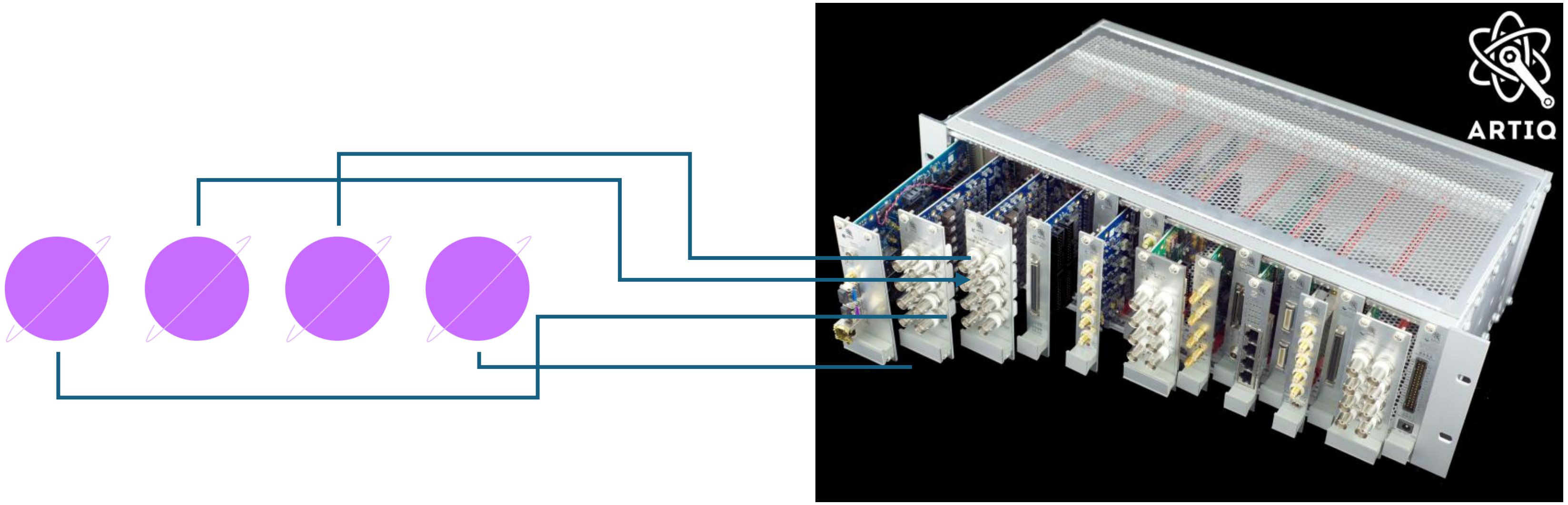
Qubits

Hardware

Languages

Control System

Qubits



Hardware

Languages

Control System

Qubits

Control Box 1

Control Box 2

Control Box 3

Control Box ...

- Coordinated
- Synchronised
- Transpiler required

```
%q4, %q5 = qssa.gate <#quantum.CNOT> (%q3, %q2)
```

Hardware

Languages

QSSA

QREF

- Coordinated
- Synchronised
- Transpiler required

```
%q4, %q5 = qssa.gate <#quantum.CNOT> (%q3, %q2)
```

Control System

Pulse

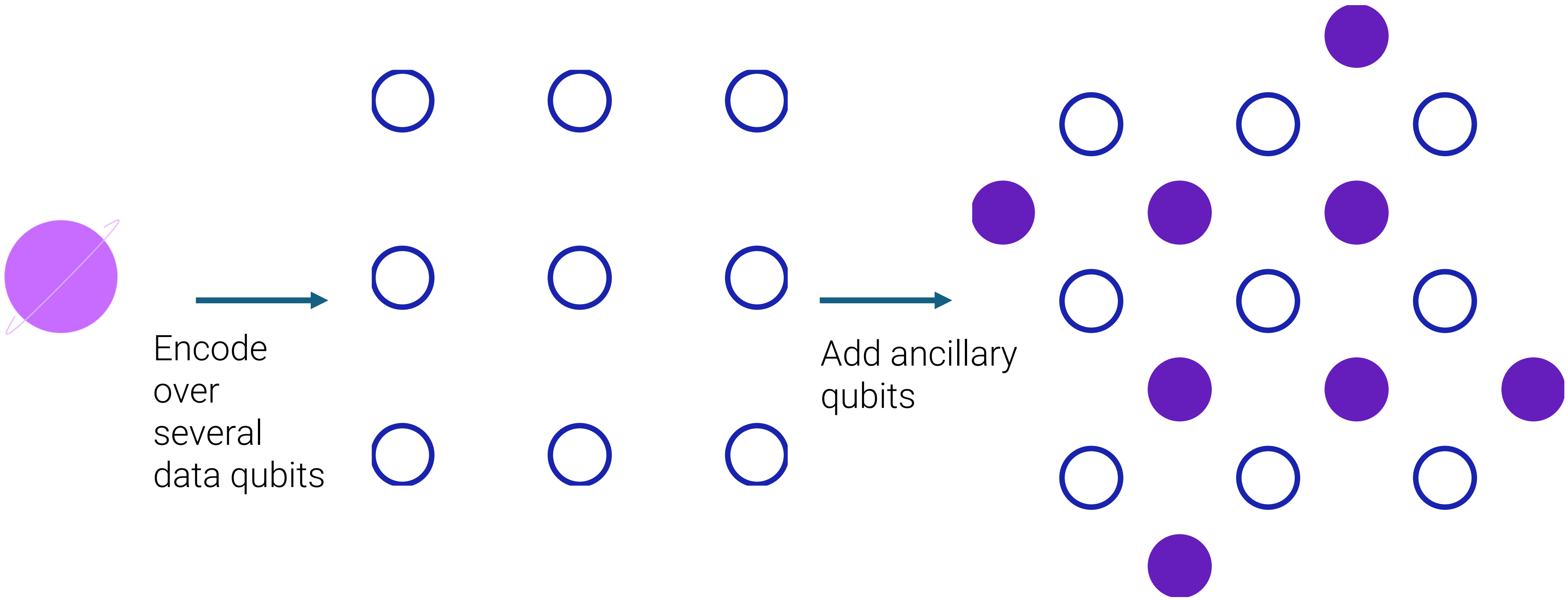
```
pulse.drive (line, duration, intensity)
```

Qubits

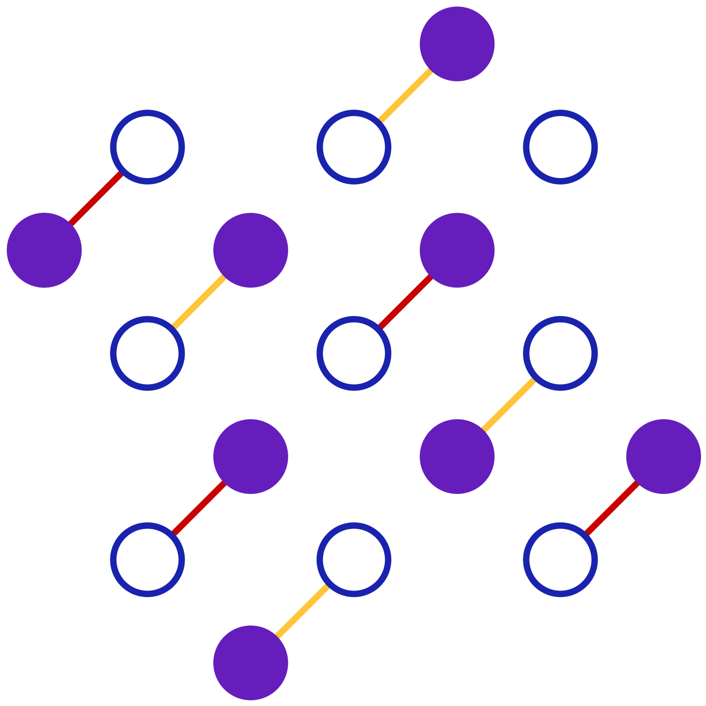
Quantum Errors and Correcting Them

- ~ 1 per 1000 operations on a qubit result in an error
- Existing qubits decohere on average in the order of microseconds

Quantum Errors and Correcting Them (One Method)

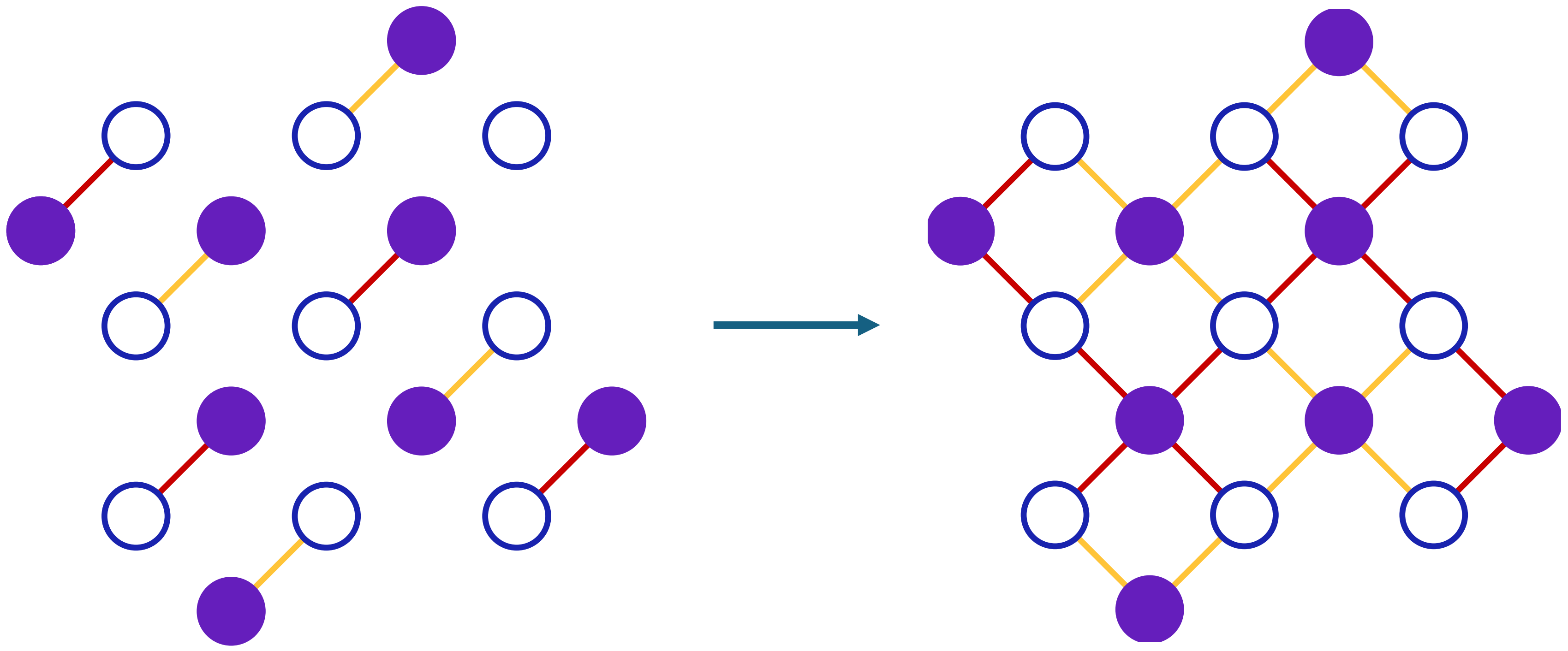


Quantum Errors and Correcting Them (One Method)



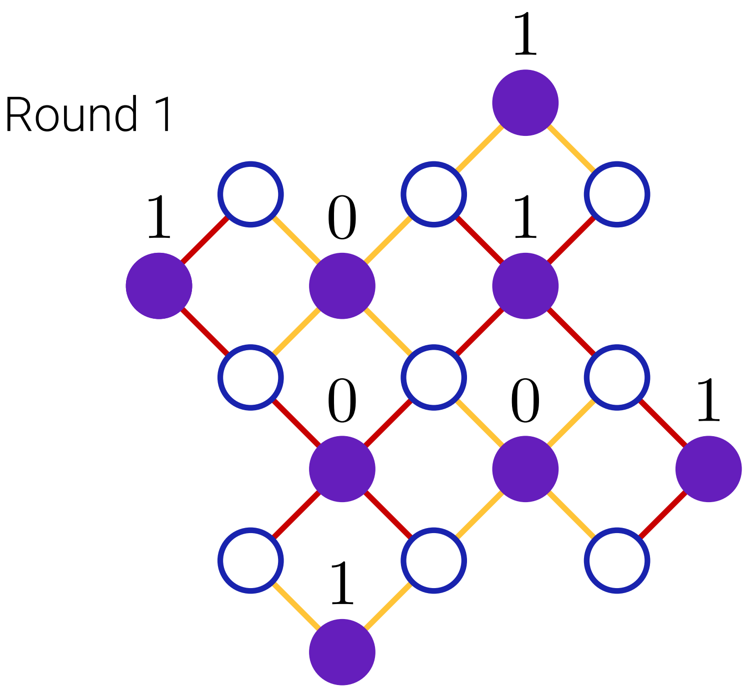
Interact data qubits with ancillas

Quantum Errors and Correcting Them (One Method)



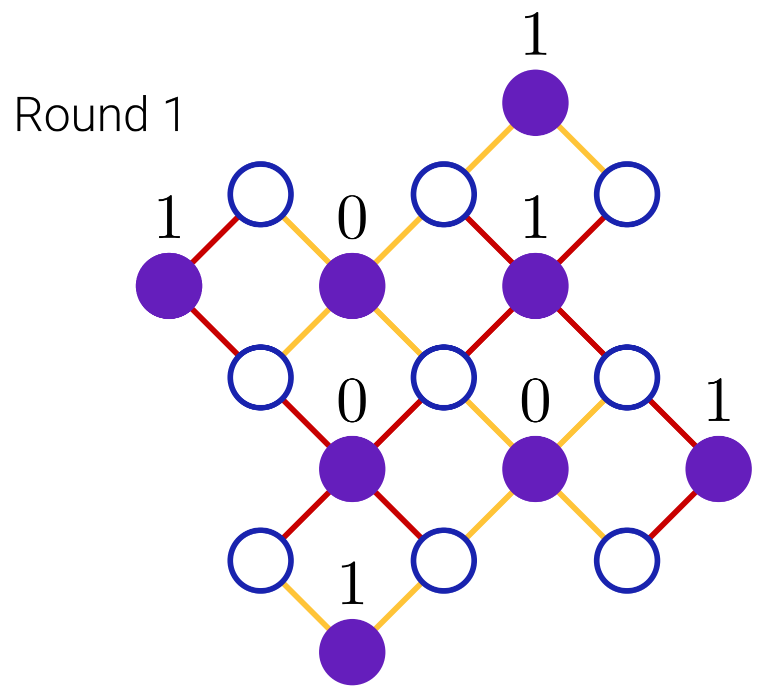
Interact data qubits with ancillas

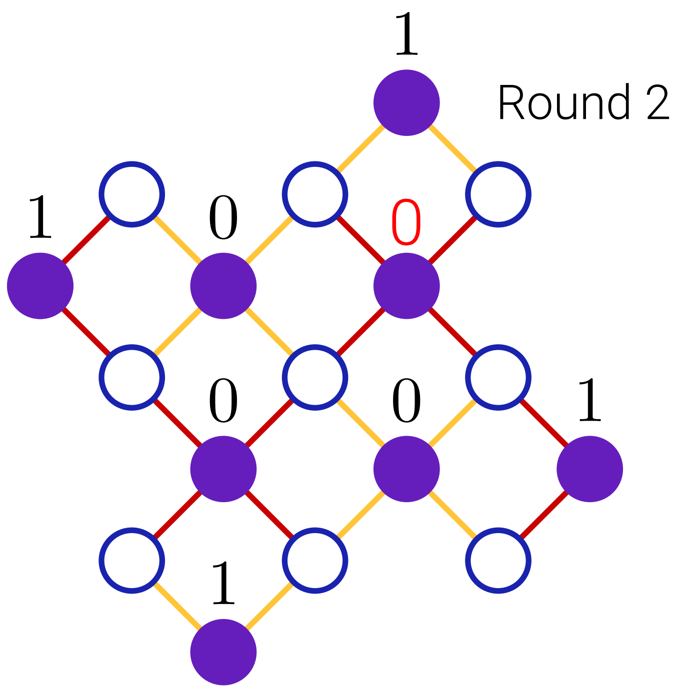
Quantum Errors and Correcting Them (One Method)



Measure, then repeat whole process

Quantum Errors and Correcting Them (One Method)





Measure, then repeat whole process, compare the results

Adapting the Computation Stack

Languages

QSSA

QREF

Control System

Pulse

Qubits

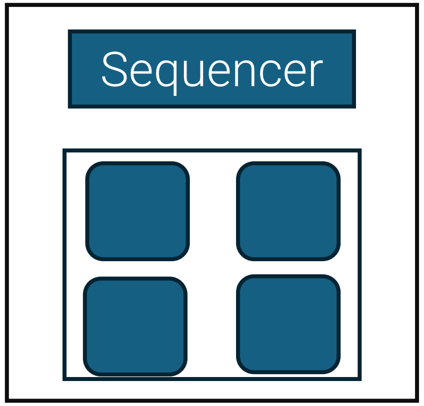
Adapting the Computation Stack

Languages

QSSA

QREF

- Separate decoding system
- Coordinated with the control system



Decoders

[Riverlane Error-Correction Box]

Control System

Pulse

Decoding System

Qubits

Scaling?

Languages

QSSA

QREF

- Millions of qubits and operations needed
- Error Rate: 1 per 1000 operations
- Error-corrected operation ~ 10 μs
- Must process Terrabytes / Second
- Code generation and integration done by hand
- Massive parallelism to exploit

[Beverland. M, Murali. P, Troyer. M, Svore. K, et al.]

Control System

Pulse

Decoding System

Qubits

Scaling?

Languages

QSSA

QREF

- Millions of qubits and operations needed
- Error Rate: 1 per 1000 operations
- Error-corrected operation ~ 10 μs
- Must process Terrabytes / Second
- Code generation and integration done by hand
- Massive parallelism to exploit

[Beverland. M, Murali. P, Troyer. M, Svore. K, et al.]

Control System

Pulse

Decoding System

Qubits

An MLIR-based Framework

Languages QSSA QREF

Control System Pulse Decoding System

An MLIR-based Framework – Adding Nothing

Languages

QSSA

QREF

```
qssa.circuit {
    %q1 = qssa.alloc<1> {"state" = 0}
    %q2 = qssa.alloc<1> {"state" = 0}
    %q3 = qssa.gate <#quantum.SGate> (%q1)
    %q4, %q5 = qssa.gate <#quantum.CNOT> (%q3,
%q2)
    %bit1 = qssa.measure (%q4)
} : () -> ()
```

Control System

Pulse

Decoding System

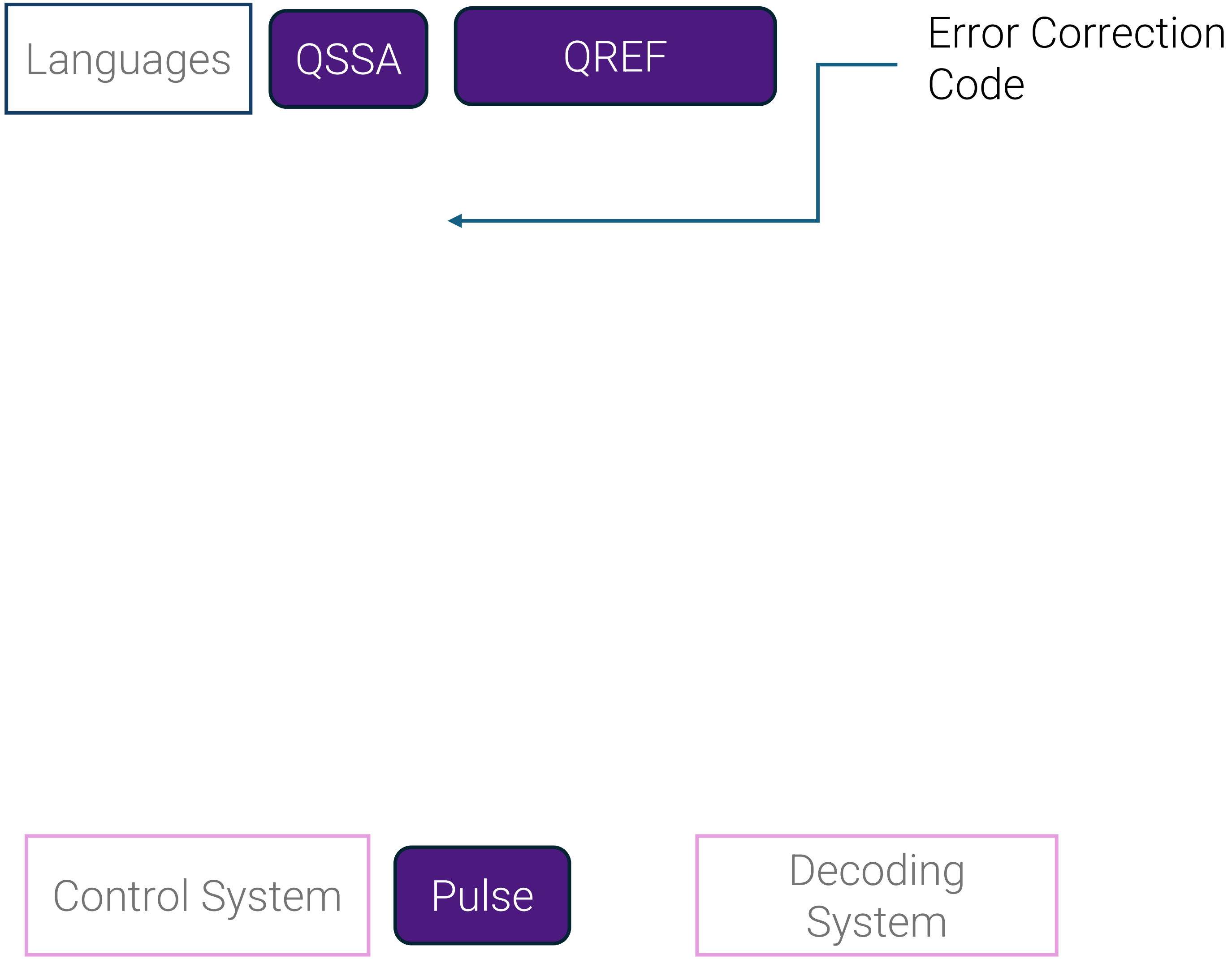
An MLIR-based Framework – Adding Nothing

Languages | QSSA | QREF

```
qssa.circuit {
    %q1 = qssa.alloc<1> {"state" = 0}
    %q2 = qssa.alloc<1> {"state" = 0}
    %q3 = qssa.gate <#quantum.SGate> (%q1)
    %q4 = qssa.gate <#quantum.Id> (%q2)
    %q4, %q5 = qssa.gate <#quantum.CNOT> (%q3, %q4)
    %bit1 = qssa.measure (%q4)
} : () -> ()
```

Control System | Pulse | Decoding System

An MLIR-based Framework – Encoded Qubits



- Transpile to new gate set
- New operations
- Implement interface

An MLIR-based Framework – Encoded Qubits

Languages

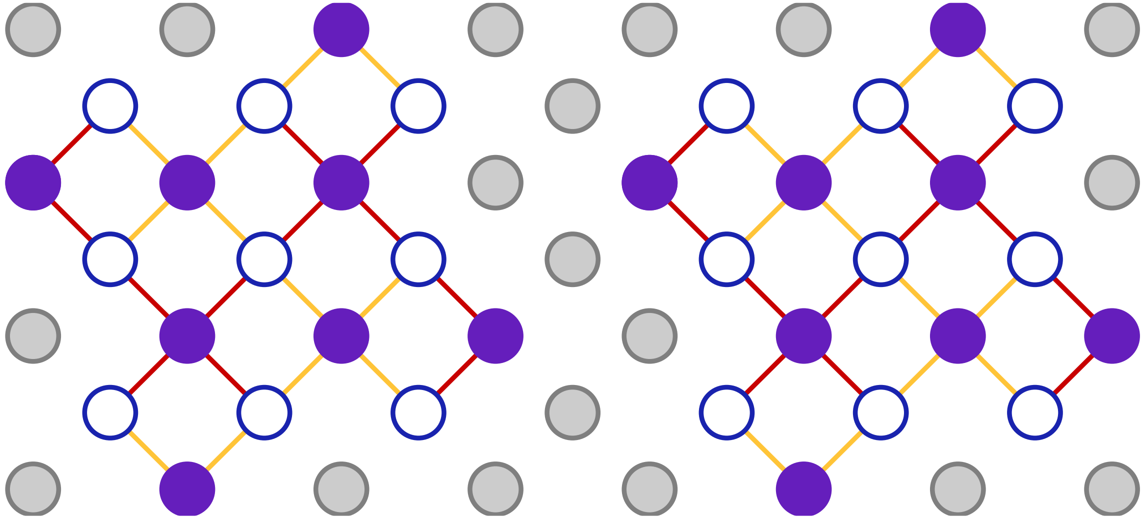
QSSA

QREF

Error Correction Code

Patches

```
%p2 = patches.merge (%p0, %p1)
```



Control System

Pulse

Decoding System

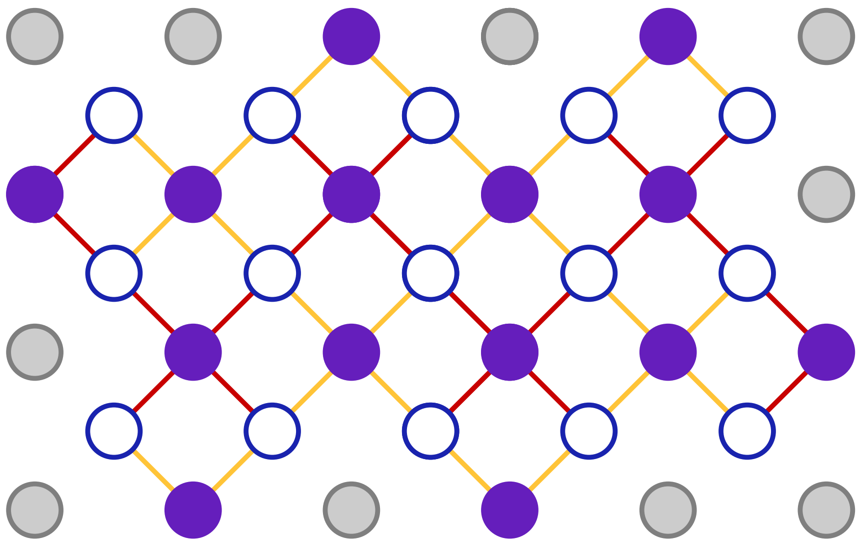
An MLIR-based Framework – Encoded Qubits

Languages | QSSA | QREF

Error Correction Code

Patches

```
%p2 = patches.merge (%p0, %p1)
```



Control System | Pulse | Decoding System

An MLIR-based Framework – Encoded Qubits

Languages

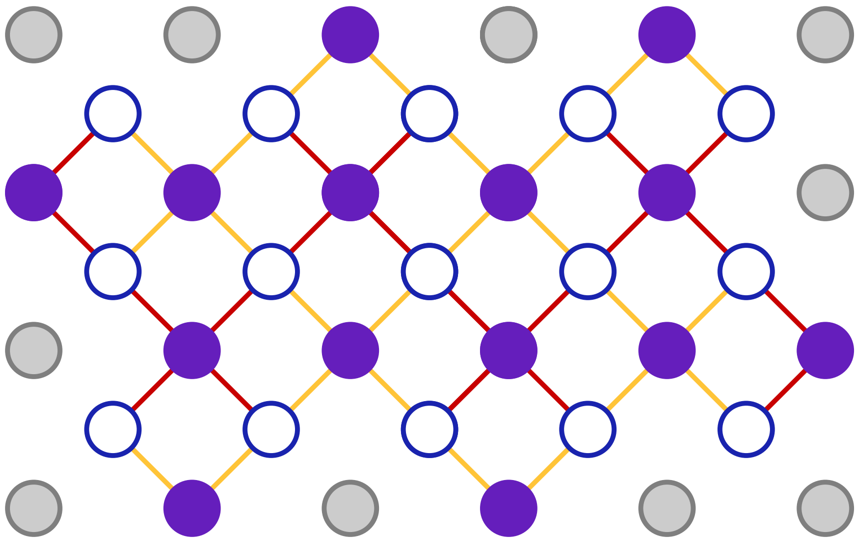
QSSA

QREF

Error Correction Code

Patches

```
%p2 = patches.merge (%p0, %p1)
%p3, %p4 = patches.split (%p2)
```

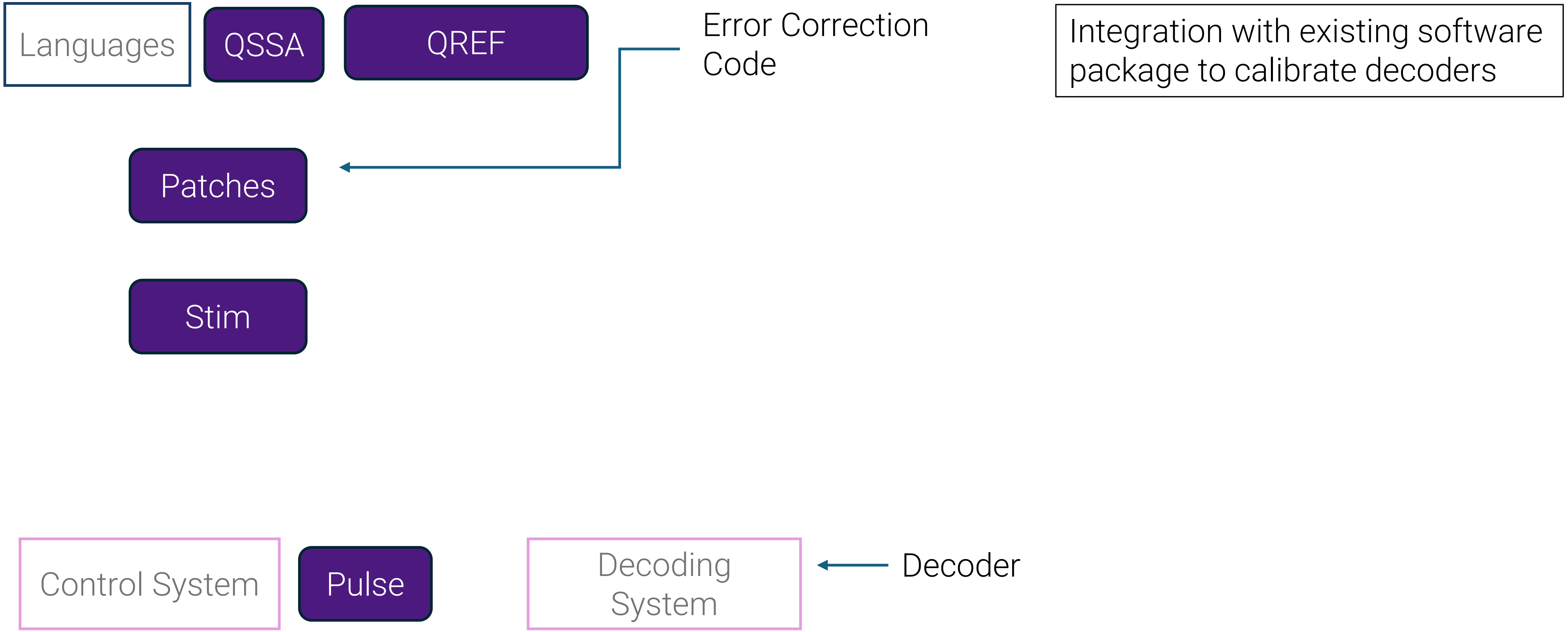


Control System

Pulse

Decoding System

An MLIR-based Framework – Calibrate Decoder



An MLIR-based Framework – Calibrate Decoder

Languages | QSSA | QREF

Error Correction Code

Integration with existing software package to calibrate decoders

Patches

Stim

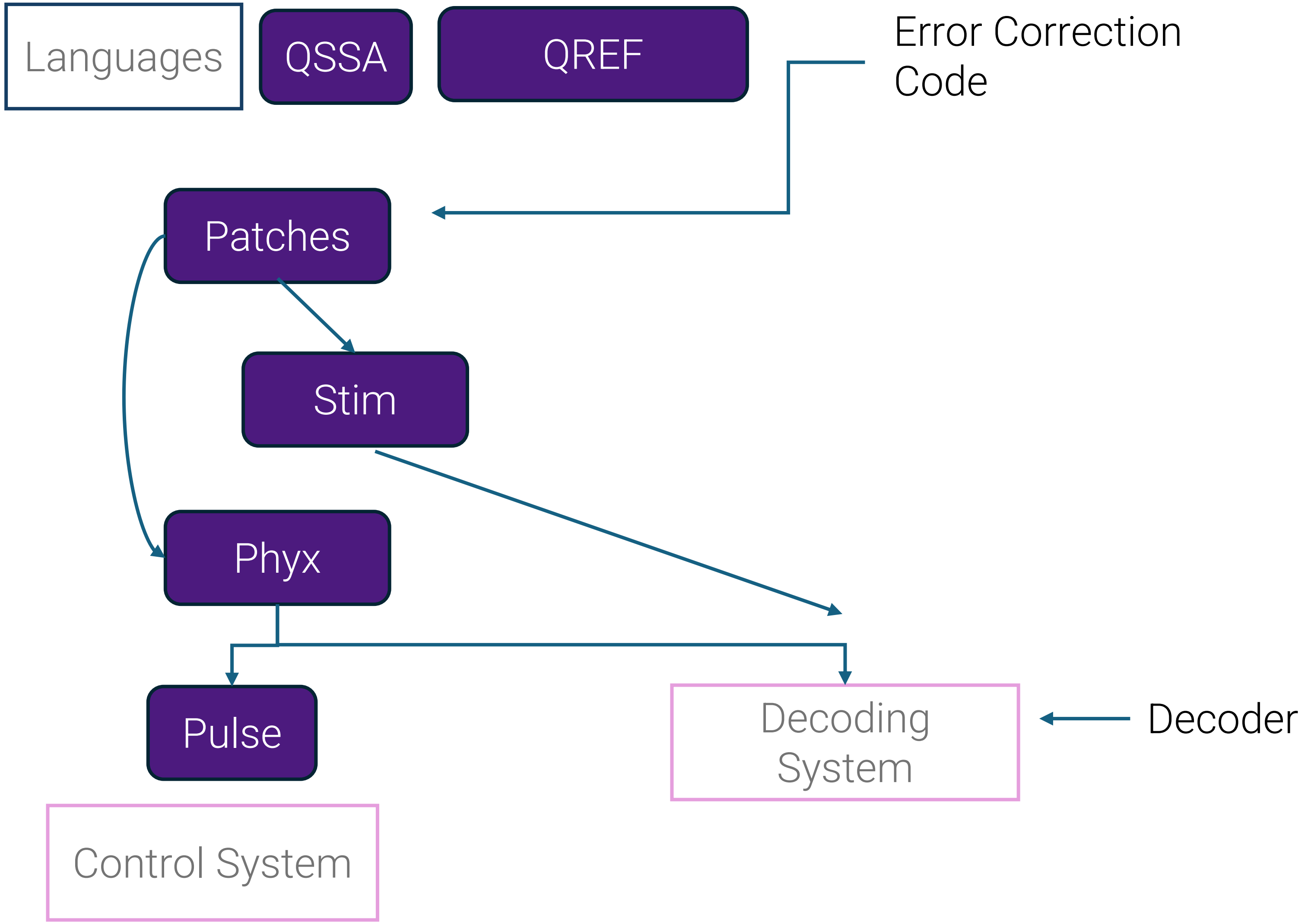
```
"noise" = #stim.depolarizingnoiseattr<0.01>

stim.assign_qubit_coord %q1 <{"qubitcoord" = #stim.qubit_coord<0, 0>}> : (!qref.qubit) -> ()
stim.tick () : () -> ()
```

Control System | Pulse

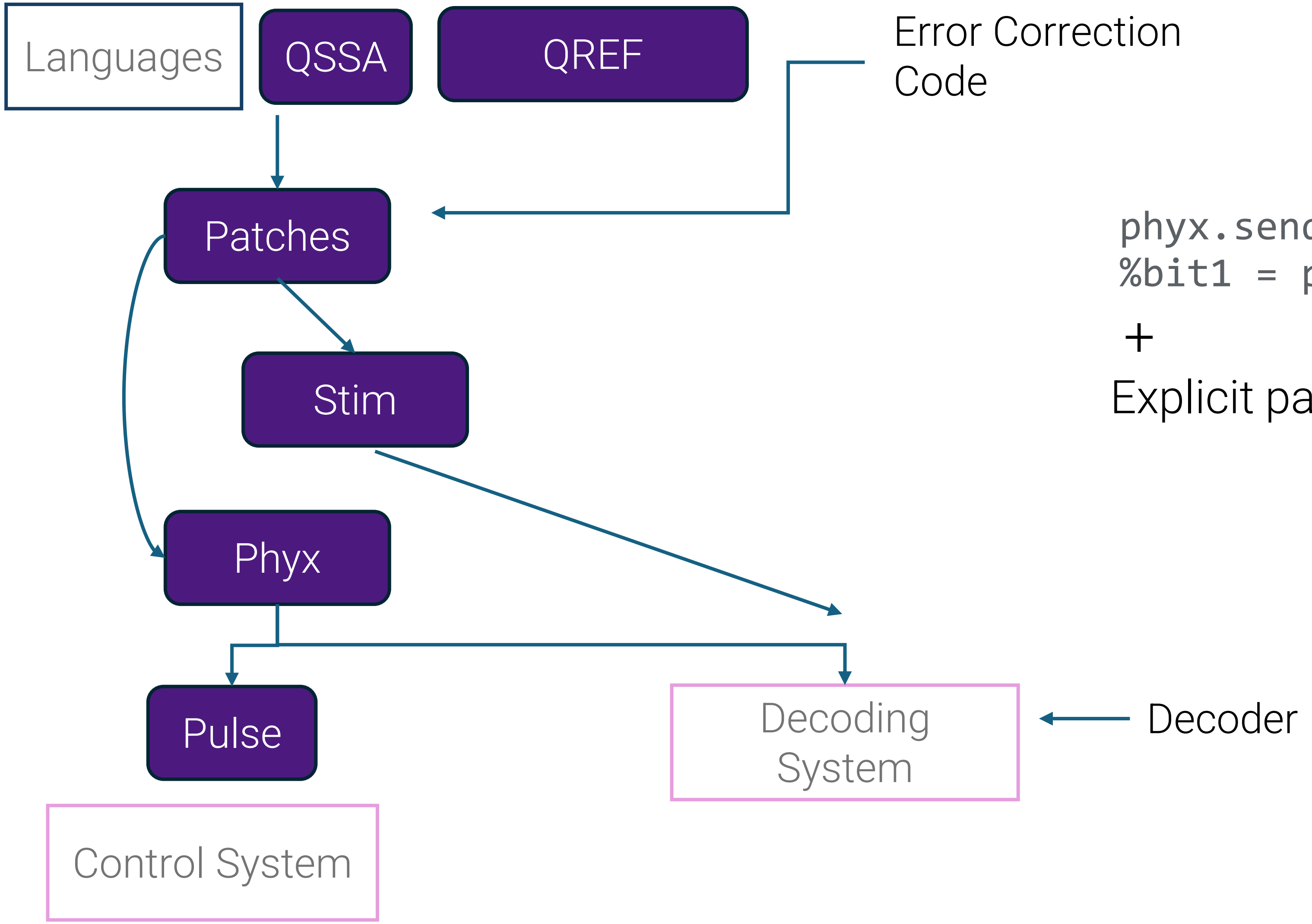
Decoding System

Decoder

An MLIR-based Framework – Coordination



An MLIR-based Framework – Coordination



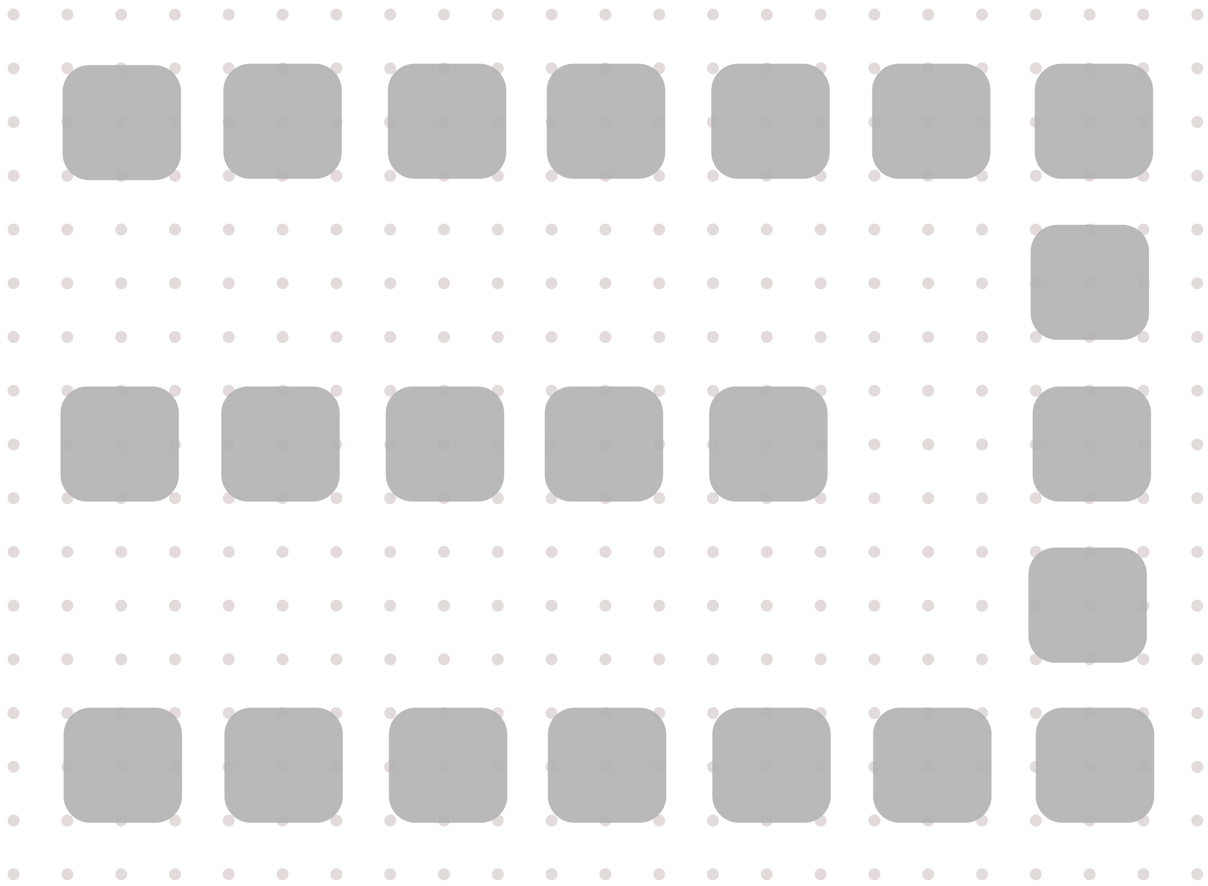

```
phyx.send %r0 {"decoder_id" = 1}
%bit1 = phyx.receive {"decoder_id" = 1}
```

+

Explicit parallel and synchronised regions

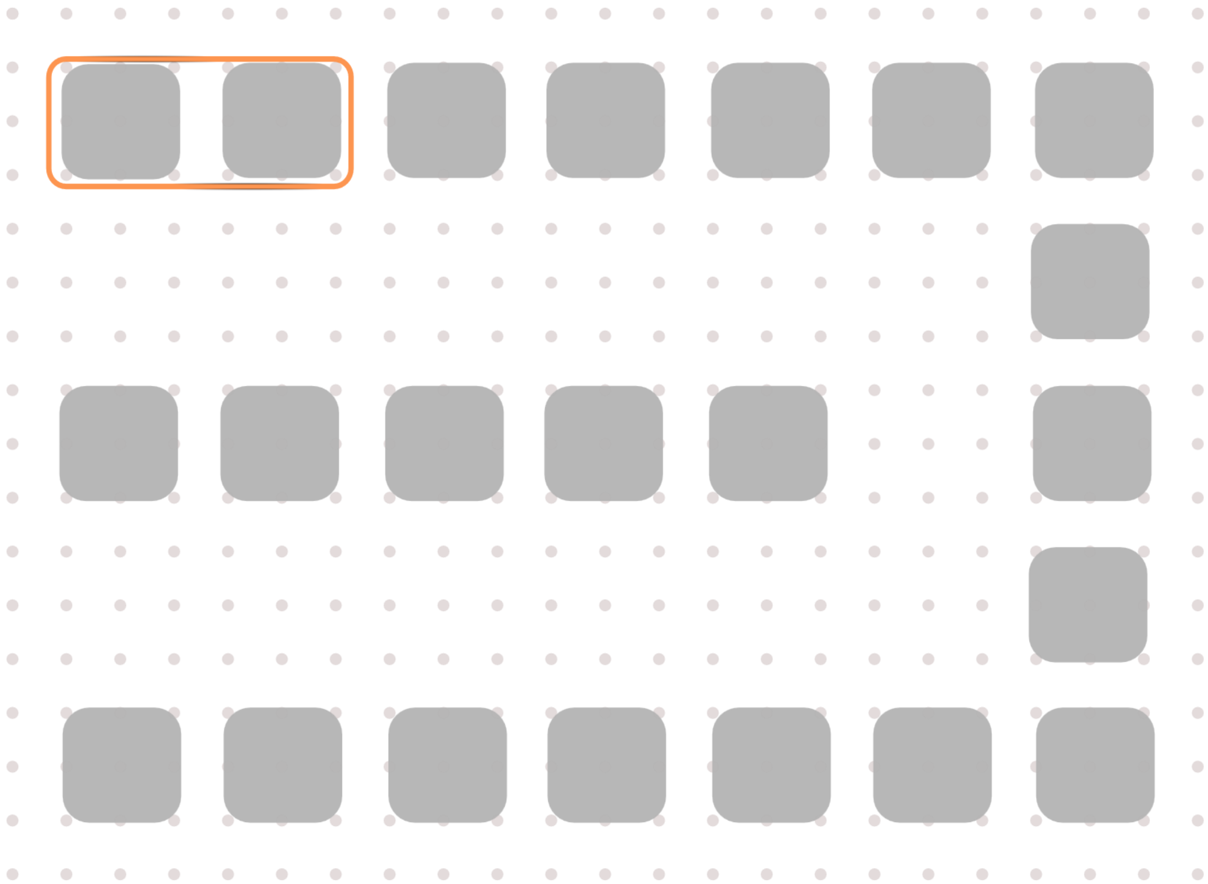
Optimisations

Register Allocation



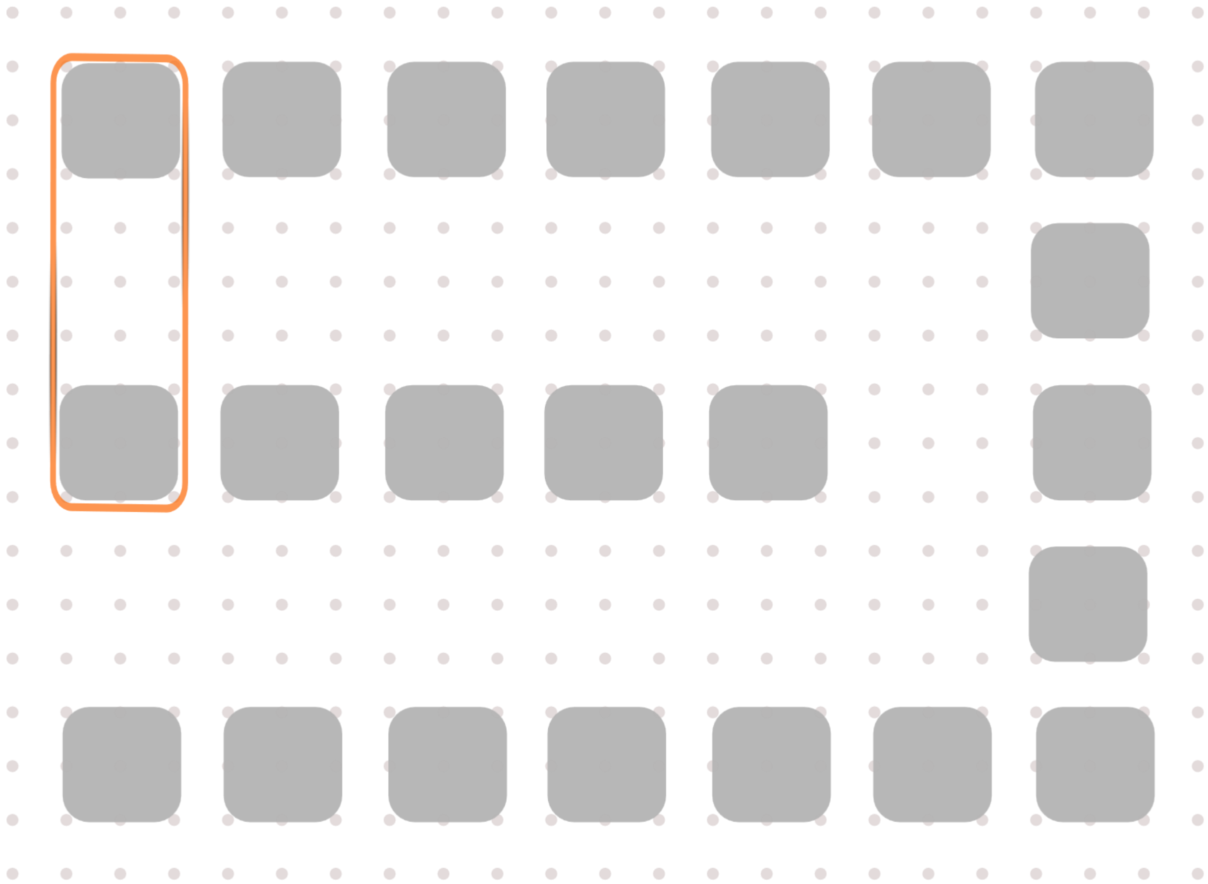
Optimisations

Register Allocation



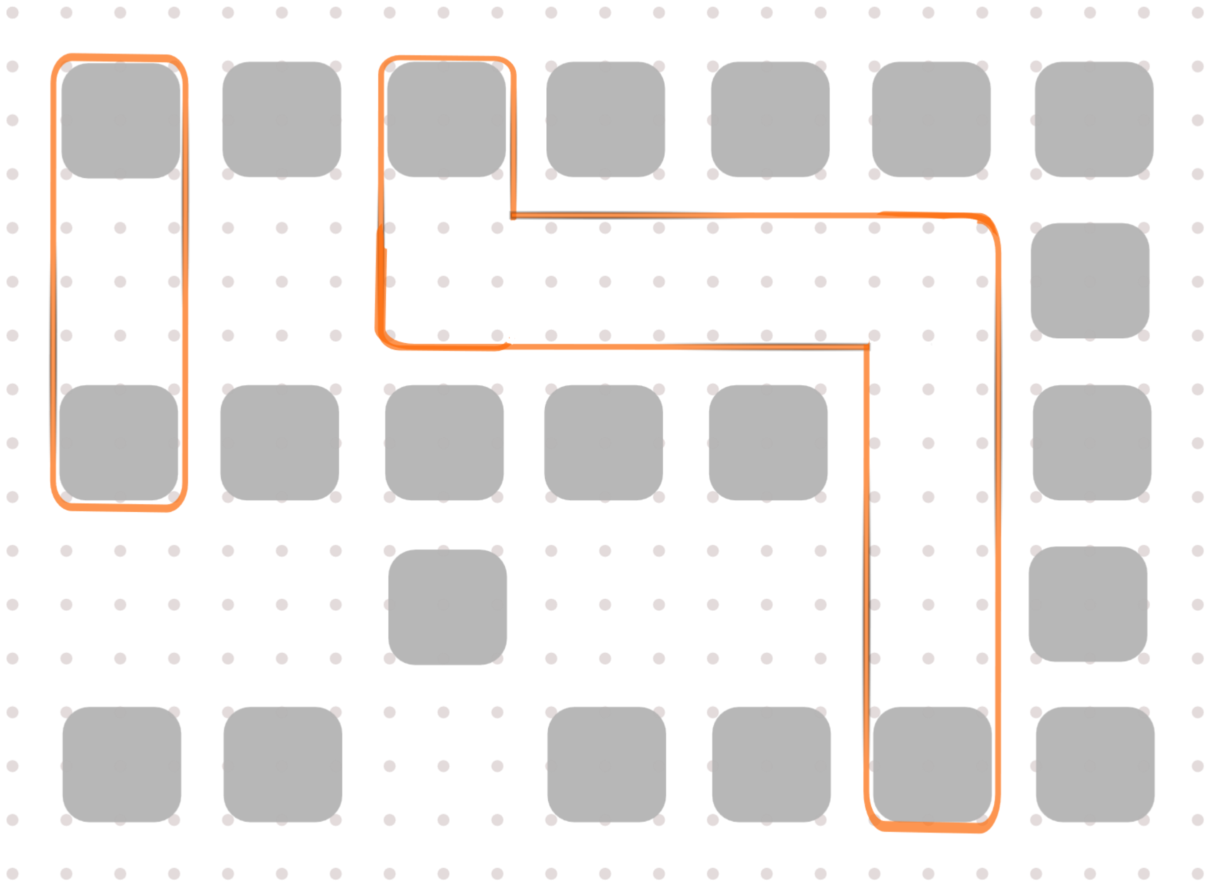
Optimisations

Register Allocation



Optimisations

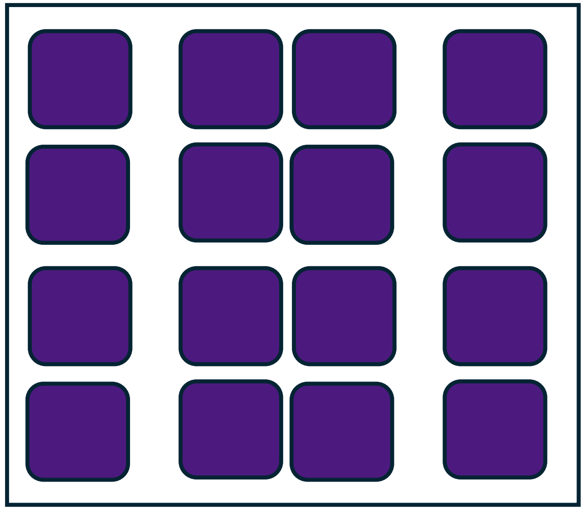
Register Allocation



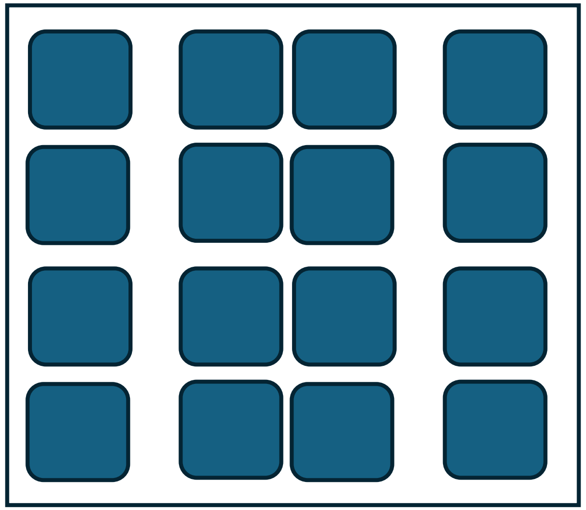
Optimisations

Instruction Scheduling and the Abstraction Problem

Quantum Chips



Decoder Chips



Parallelism? Abstractions for Algorithms? Knowledge About Hardware?