QUANTINUUM

Quantum Circuit Compilation and Classical Control with TKET: Part II

Presented by:

Callum Macpherson & Lewis Wright
19/09/2023

Recap of Part I

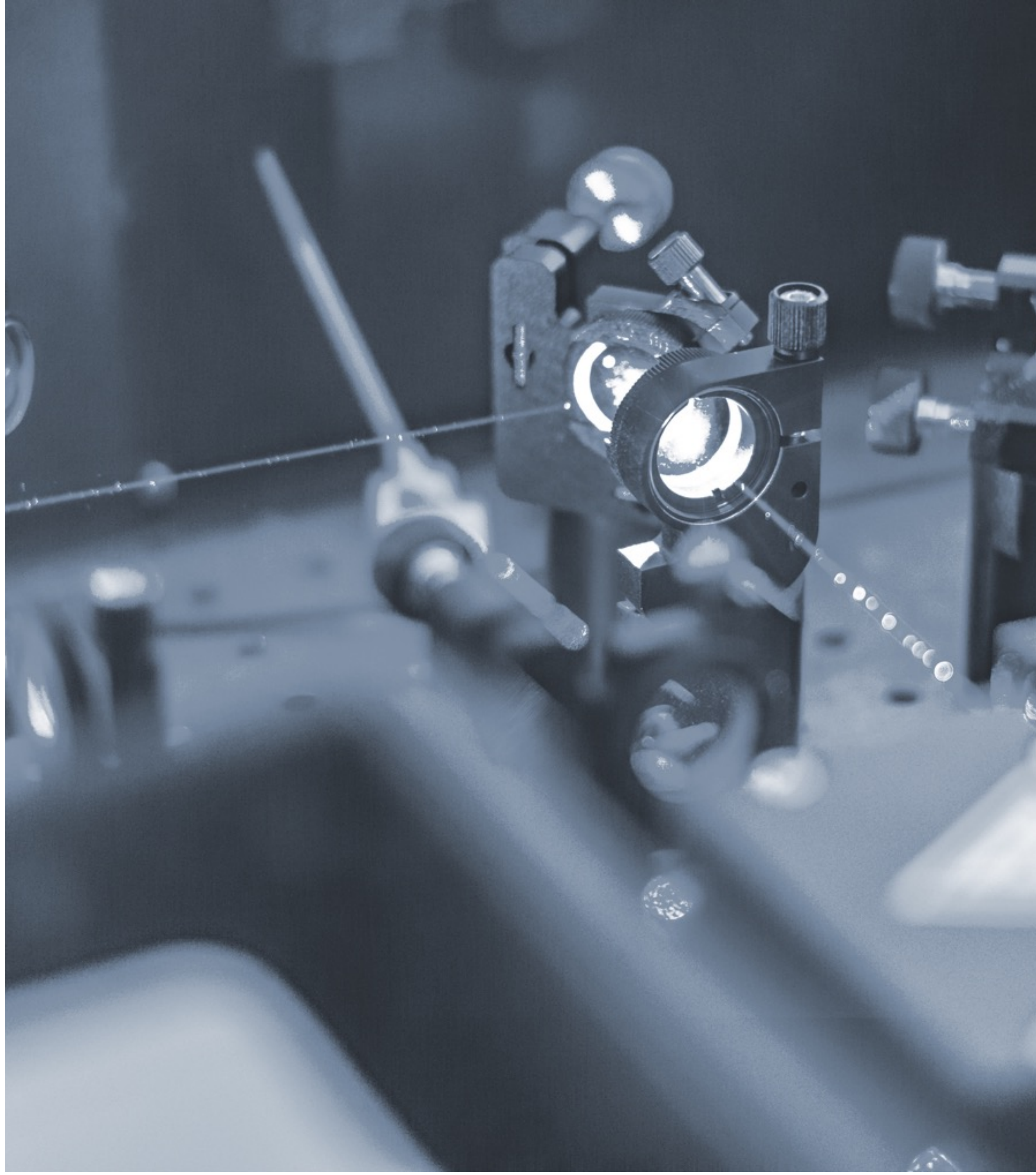
- Basic introduction:
  - Different quantum software tools for construction, compilation and execution of quantum circuits
  - What is TKET and pytket?

- TKET 101:
  - Constructing circuits with pytket
  - Running circuits with pytket on different backends
  - New features: state prepation & ZX calculus

- Practical application: PDE solver
  - Compiling pre-optimised parameterised quantum circuit to pytket
  - Converting circuit to different native gatesets
  - Running noisy simulations

Agenda: Part II

- Tutorial 2: Introduction slides (10 minutes)
  - Mid-circuit measurement
  - Classical control
- Quantum error correction: Repetition code (40 minutes)
  - (fill)
- Subroutine application: Quantum Fourier transform (40 minutes)
  - Semi-classical
  - Fault-tolerant
  - Circuit compilation
  - Transforming Gaussian
    - Noiseless
    - Noisy



▪Mid-circuit measurement & classical control

What is classical control?

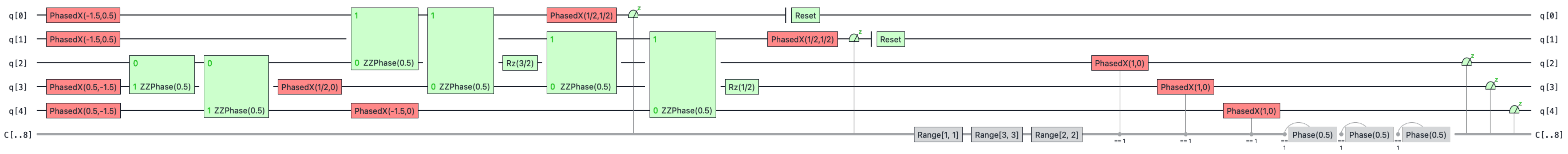
Condition a gate acting on a qubit by upon the output (measurement) of another qubit.

- For several important applications, we need to measure qubits *during* computation and perform operations based on that measurement
- Many use cases:
  - Quantum teleportation
  - Quantum error correction
  - Repeat until success (guarantee the implementation of a desired operation)
  - Qubit reuse *e.g.* MPS simulation (Huggins *et al.* 2018)

- Quantum Programming languages/high level languages - Q#, Silq, Quipper
- Compiler - TKET, qiskit, BSQKit
- Online services - AWS Bracket
- Quantum Error Correction/Mitigation- Qermit, others
- Application libraries - e.g. InQuanto, pennylane
- Simulators e.g. Qulacs, Stim

Capabilities of pytket-quantinuum

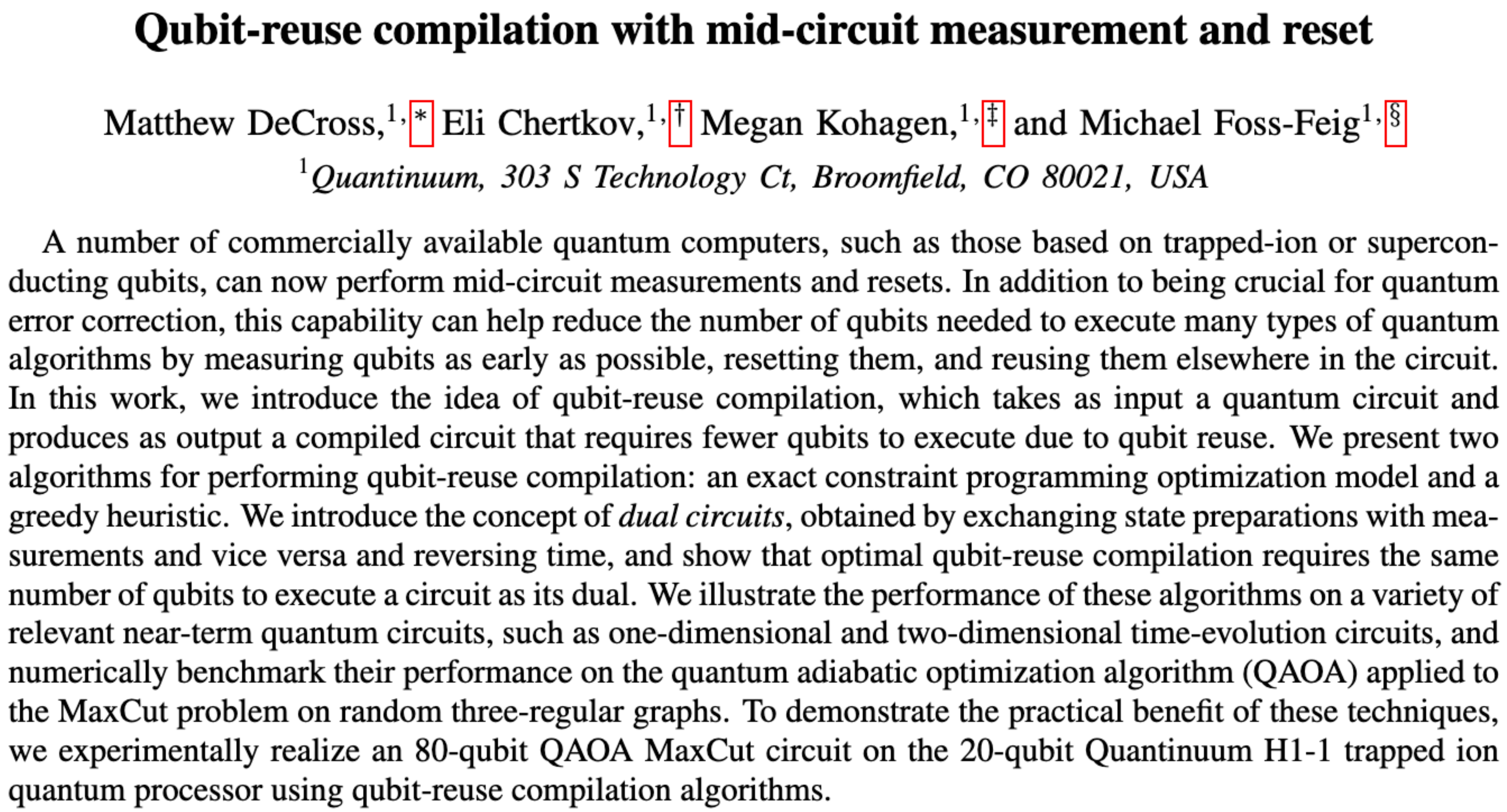
- Arbitrary angle ZZPhase (aka RZZ) gate - expressive two qubit gate, higher fidelity for a smaller rotation angle
- Support for classical operations - Extended QASM, WASM
- Mid-circuit measurement, reset
- Partial results retrieval
- Tailored compilation for the H-Series devices



pip install pytket-quantinuum

Qubit Reuse

- Work done by the theory team in Quantinuum.
- Executed an 80 qubit QAOA instance for the Maxcut problem using mid-circuit measurement and reset



Qubit-reuse compilation with mid-circuit measurement and reset

Matthew DeCross,[1,*] Eli Chertkov,[1,†] Megan Kohagen,[1,‡] and Michael Foss-Feig[1,§]

[1]*Quantinuum, 303 S Technology Ct, Broomfield, CO 80021, USA*

A number of commercially available quantum computers, such as those based on trapped-ion or superconducting qubits, can now perform mid-circuit measurements and resets. In addition to being crucial for quantum error correction, this capability can help reduce the number of qubits needed to execute many types of quantum algorithms by measuring qubits as early as possible, resetting them, and reusing them elsewhere in the circuit. In this work, we introduce the idea of qubit-reuse compilation, which takes as input a quantum circuit and produces as output a compiled circuit that requires fewer qubits to execute due to qubit reuse. We present two algorithms for performing qubit-reuse compilation: an exact constraint programming optimization model and a greedy heuristic. We introduce the concept of *dual circuits*, obtained by exchanging state preparations with measurements and vice versa and reversing time, and show that optimal qubit-reuse compilation requires the same number of qubits to execute a circuit as its dual. We illustrate the performance of these algorithms on a variety of relevant near-term quantum circuits, such as one-dimensional and two-dimensional time-evolution circuits, and numerically benchmark their performance on the quantum adiabatic optimization algorithm (QAOA) applied to the MaxCut problem on random three-regular graphs. To demonstrate the practical benefit of these techniques, we experimentally realize an 80-qubit QAOA MaxCut circuit on the 20-qubit Quantinuum H1-1 trapped ion quantum processor using qubit-reuse compilation algorithms.

Subroutine Application: Quantum Fourier Transform Notebook

QUANTINUUM