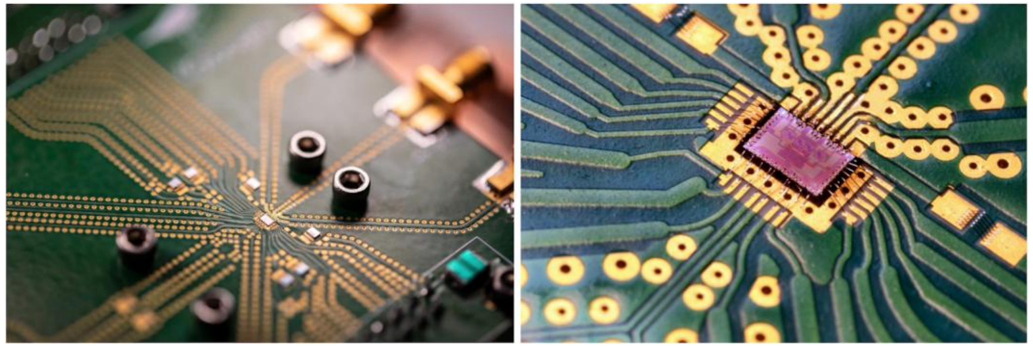
QUANTUM COMPUTING WITHOUT A QUANTUM COMPUTER

Wednesday 18th February 2026

Dr Oliver Thomson Brown
EPCC, University of Edinburgh
o.brown@epcc.ed.ac.uk

Introduction

- Quantum computing is coming!
  - If you work at a quantum computing firm it's already here…

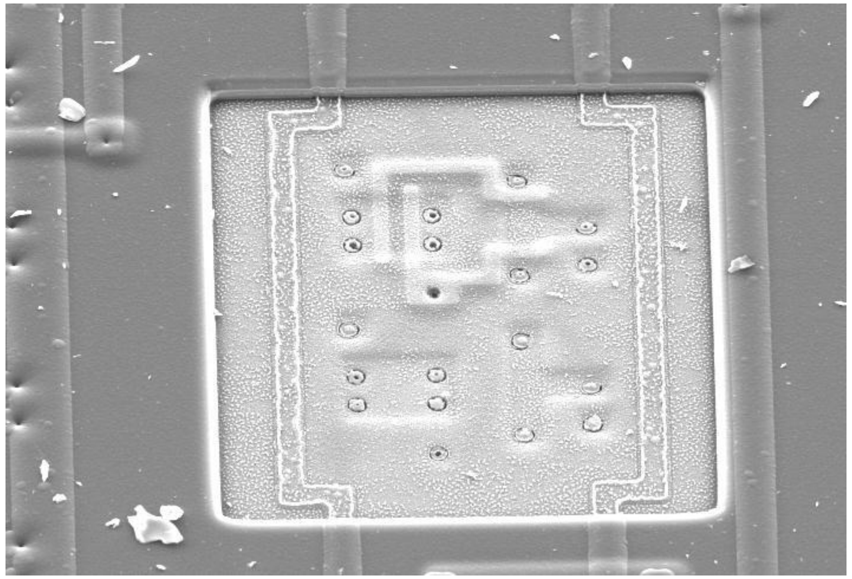


Google Sycamore quantum processor.
Source: https://ai.googleblog.com/2019/02/on-path-to-cryogenic-control-of-quantum.html

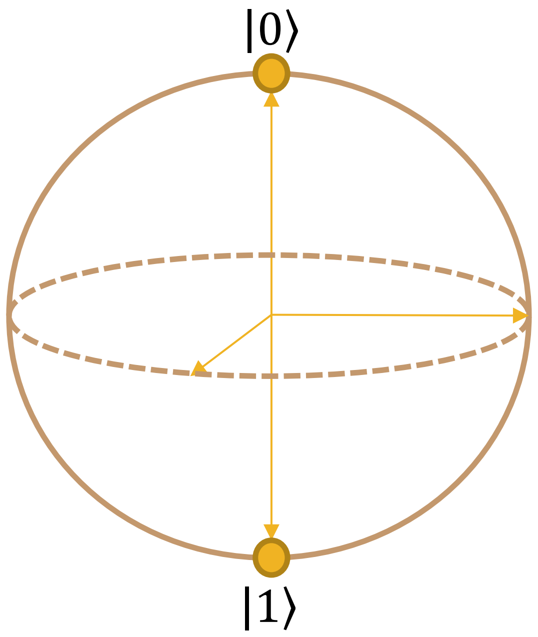
Cryostat surrounding Google Sycamore quantum processor.
Source: https://phys.org/news/2020-08-google-largest-chemical-simulation-quantum.html

Classical Bits

- Programmers have an abstract model of the hardware on which their programs are executed.
  - Or rather, they have lots of them, depending on the high-level features of the hardware…
  - The result of 70 years worth of research and development!
- At the lowest level we have the concept of a ‘bit’.
  - We build datatypes on bits, and data structures on datatypes.
- As a software developer, **I don’t care how a bit is implemented.**


CMOS 2 input NOR gate with passivation and upper metal layers removed (3 micron CMOS).
Source: http://www2.eng.cam.ac.uk/~dmh/4b7/resource/mesp.htm

Quantum Bits



The Bloch sphere.

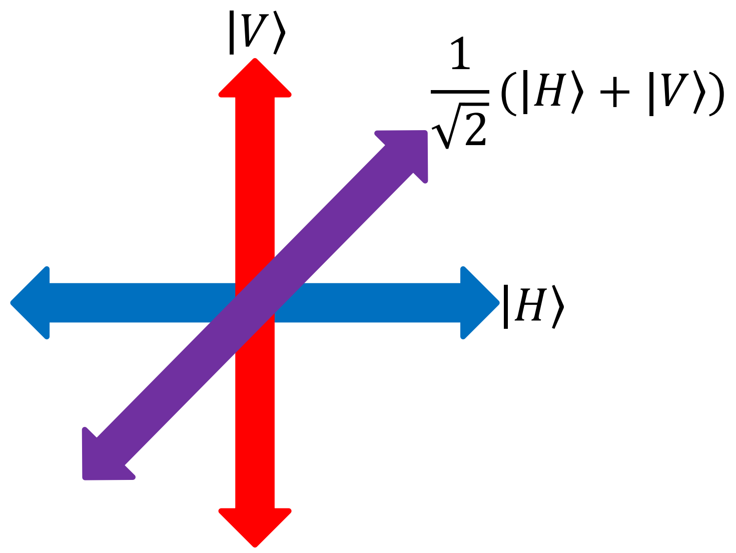
- A classical bit has two states, usually referred to as ‘0’ and ‘1’.
- A quantum bit (‘qubit’) has two states, usually referred to as $|0\rangle$ and $|1\rangle$. But…

Superposition Principle

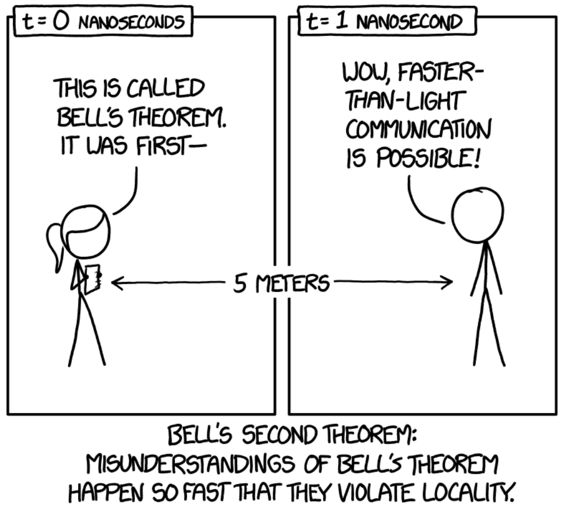
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Superposition

- How does that work?
  - Well it depends on the underlying physical implementation!
  - One example is polarised light, used in QKD.
- As quantum software developers, we try not to worry about it.
- Crucially, the superposition principle applies to many-body states as well as individual qubits.
  - $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$



Entanglement

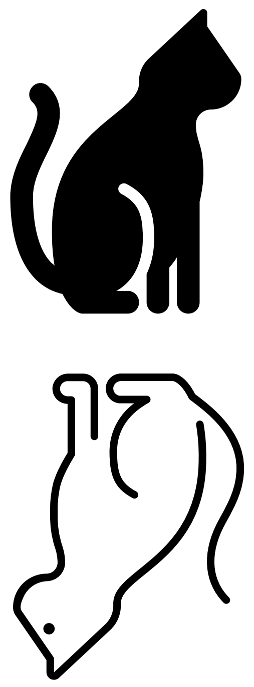

t = 0 NANOSECONDS
THIS IS CALLED BELL'S THEOREM. IT WAS FIRST—
t = 1 NANOSECOND
WOW, FASTER-THAN-LIGHT COMMUNICATION IS POSSIBLE!
5 METERS
BELL'S SECOND THEOREM: MISUNDERSTANDINGS OF BELL'S THEOREM HAPPEN SO FAST THAT THEY VIOLATE LOCALITY.

Source: https://xkcd.com/1591/

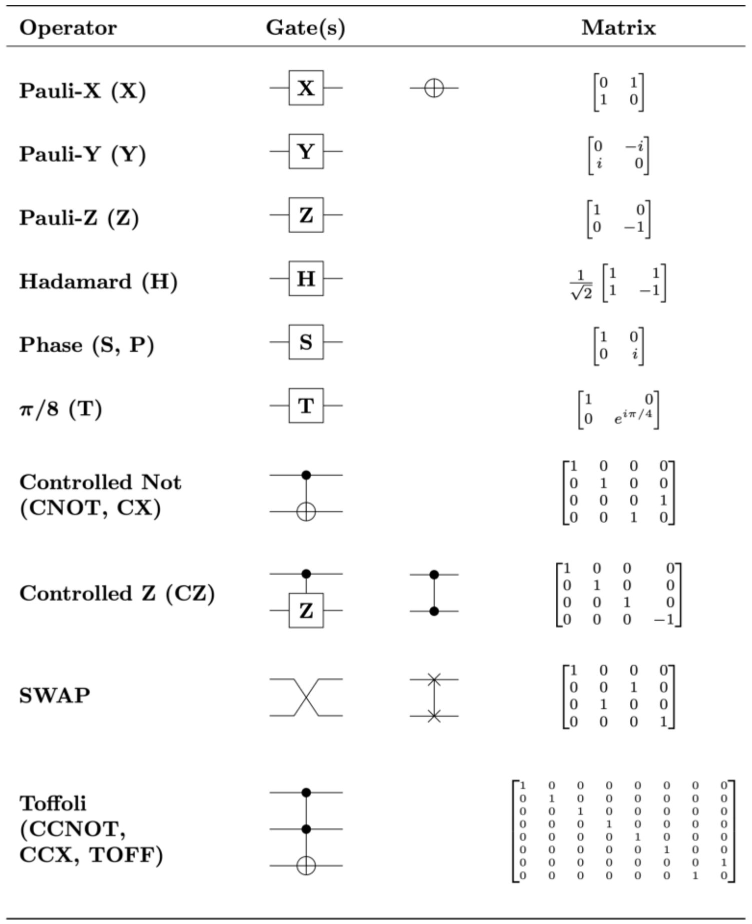
- That state, $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, has another interesting property. It is *maximally entangled*.
  - It is one of the 4 Bell states.
- Understanding entanglement would be a whole talk in itself…
- Key point for us is that entangled states **are not separable**.
  - $|\Phi^+\rangle \neq |A\rangle|B\rangle$

Measurement

- Measurement of quantum systems is another big topic.
- Result of a measurement on a single qubit is either $|0\rangle$ or $|1\rangle$.
  - The state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ would be measured in state $|0\rangle$ or $|1\rangle$ with probability $|\alpha|^2$ and $|\beta|^2$ respectively.
  - Building up a more detailed picture of the original state requires repeated measurements.
- **Measurement changes the state of the system**.
  - It's now in the state you measured it in.



Quantum Circuits



Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{X}$	$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)	$\boxed{Y}$		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)	$\boxed{Z}$		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)	$\boxed{H}$		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Phase (S, P)	$\boxed{S}$		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
$\pi/8$ (T)	$\boxed{T}$		$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled Z (CZ)	$\boxed{Z}$		$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
SWAP			$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$

- How do we program a quantum computer then?
  - By writing a *quantum circuit*.
- We have some quantum register of qubits, and apply gates, like those on the left.
  - We may also have a classical register.
- OpenQASM quantum assembly language is now on v3.1, enabling imperative programming.
  - https://github.com/Qiskit/openqasm

Quantum Computer Emulation

- Qubits are represented by a vector of complex numbers.
  - Each element is the *complex amplitude* for that state.
  - Complex numbers are represented by two floating-point values.
  - Generally double precision is required.
  - One qubit needs two complex doubles.
    - $2 \times 2 \times 8 \text{ bytes} = 32 \text{ bytes}.$
- Single-qubit gates (also known as operators) are represented by 2×2 matrices.
  - Gates are applied to the qubit by calculating the matrix-vector product – the result is the new statevector.
  - Operators are also complex valued, so four complex doubles for a single-qubit gate.
    - $4 \times 16 \text{ bytes} = 64 \text{ bytes}.$

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$\hat{O} = \begin{pmatrix} O_{00} & O_{01} \\ O_{10} & O_{11} \end{pmatrix}$$

$$|\Psi'\rangle = \hat{O}|\Psi\rangle = \begin{pmatrix} O_{00}\alpha + O_{01}\beta \\ O_{10}\alpha + O_{11}\beta \end{pmatrix}$$

The Scaling Problem

- Two classical bits can be represented by two numbers: 00, 01, 10, 11.
  - The composite state is a separable product of the individual bits.
  - Representation scales like N.
- Recall what we said about the superposition principle, and entanglement…
  - Any linear combination of any possible state of the composite system is valid.
  - In fact, there are valid composite states that **cannot** be represented as a separable product of individual qubit states.
- We need a complex number for every possible composite state.
  - Representation scales like 2^N.

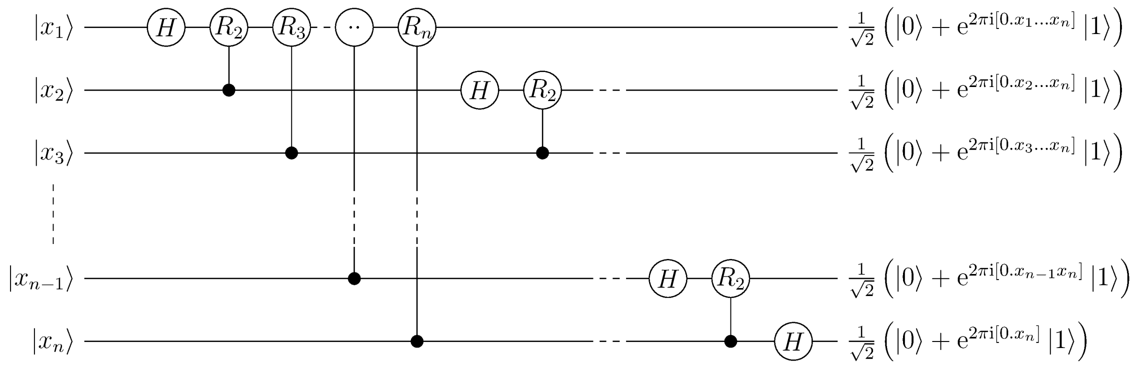
$$|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$
$$= \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

QuEST

- QuEST (Quantum Exact Simulation Toolkit) is the software we recommend for emulating a quantum computer on ARCHER2.
  - https://github.com/QuEST-Kit/QuEST
  - Developed at University of Oxford by the QTechTheory group.
    Now maintained by EPCC!
  - Parallelised using MPI+OpenMP and supports GPU acceleration.
  - Written in C/C++.
  - Minimises the number of messages sent, at the cost of 1 additional qubit's worth of memory.
    - You can read more about their communication strategy in Jones, T., Brown, A., Bush, I. *et al.* QuEST and High Performance Simulation of Quantum Computers. *Sci Rep* **9,** 10736 (2019). https://doi.org/10.1038/s41598-019-47174-9.

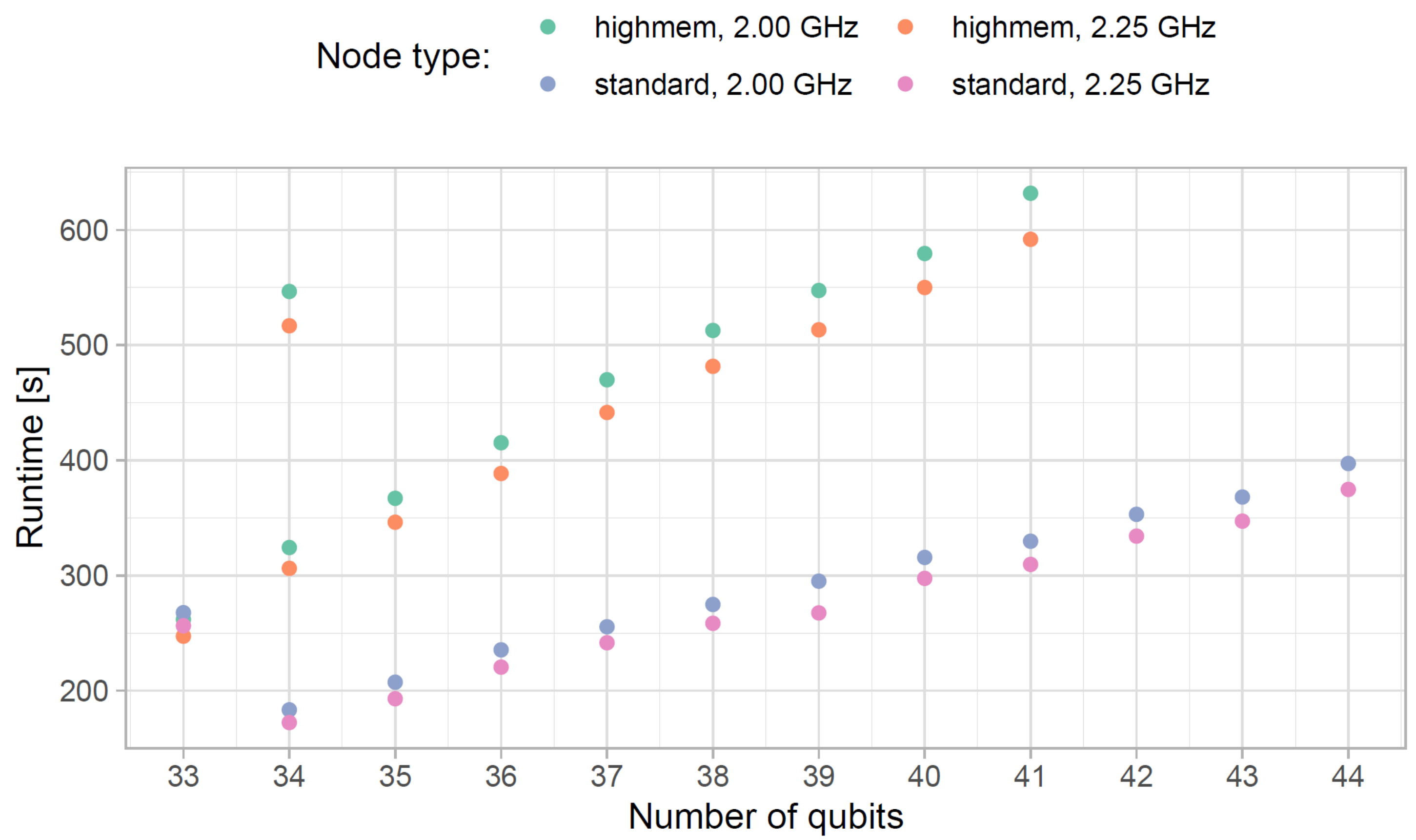
QFT

- Test code implemented a quantum Fourier transform on an $N-\mathrm{qubit}$, all $|0\rangle$ state.
  - Result is to place every qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Boring, but convenient!
- Total number of gates is $N(N-1) = \mathcal{O}(N^2)$.
- Assumes N perfect logical qubits – no noise.


$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
$|x_{n-1}\rangle$
$|x_n\rangle$
H
R_2
R_3
R_n
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_{n-1}x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right)$

Source: Trenar3, CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0, via Wikimedia Commons

ARCHER2 Results



Qubit Index	Blocking		Non-blocking	
	Time	Energy	Time	Energy
29	0.51 s	15.3 kJ	0.53 s	15.0 kJ
30	0.59 s	15.7 kJ	0.74 s	18.7 kJ
31	0.80 s	20.8 kJ	0.97 s	24.2 kJ
32	9.63 s	191 kJ	8.82 s	179 kJ

Table 1: Time/energy per gate in the Hadamard benchmark on qubits 29–32 using blocking/non-blocking MPI.

Experiment	Built-in	Fast	Built-in	Fast
Qubits	43	43	44	44
Nodes	2048	2048	4096	4096
Runtime	417 s	270 s	476 s	285 s
Energy	294 MJ	206 MJ	664 MJ	431 MJ

Table 2: Runtime and energy consumption of large QFT runs on ARCHER2. 'Built-in' is using the standard QuEST QFT, 'Fast' is our modified version.

See Adamski et al. arXiv:2308.07402.

ARCHER2 Results

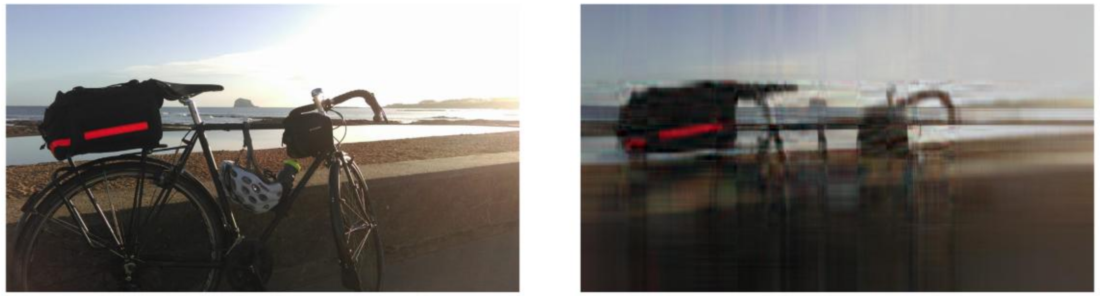
- Number of nodes for N qubits:
  - $\lceil 2^{N-32} \rceil$
  - 32 is the number of qubits you can emulate on a single 256GB node (actually it's 33, because you don't need MPI!).
- Number of qubits on N nodes:
  - $\lfloor \log_2(N) + 32 \rfloor$
  - Hence we are able to emulate **44** qubits on **4096** nodes.

Compression

- Why not just use compression?
  - Good news for operators (gates) – they’re typically sparse, and generally get *more* sparse as they get larger. They might even have a simple enough structure that we don’t need to write them out at all!
  - Bad news for statevectors… They are usually *not* sparse for states of interest.
- Compression has been used for large scale simulations – see BMQSim (arXiv:2410.14088) and JUQCS-50 (arXiv:2511.03359), though not without caveats.

Tensor Networks

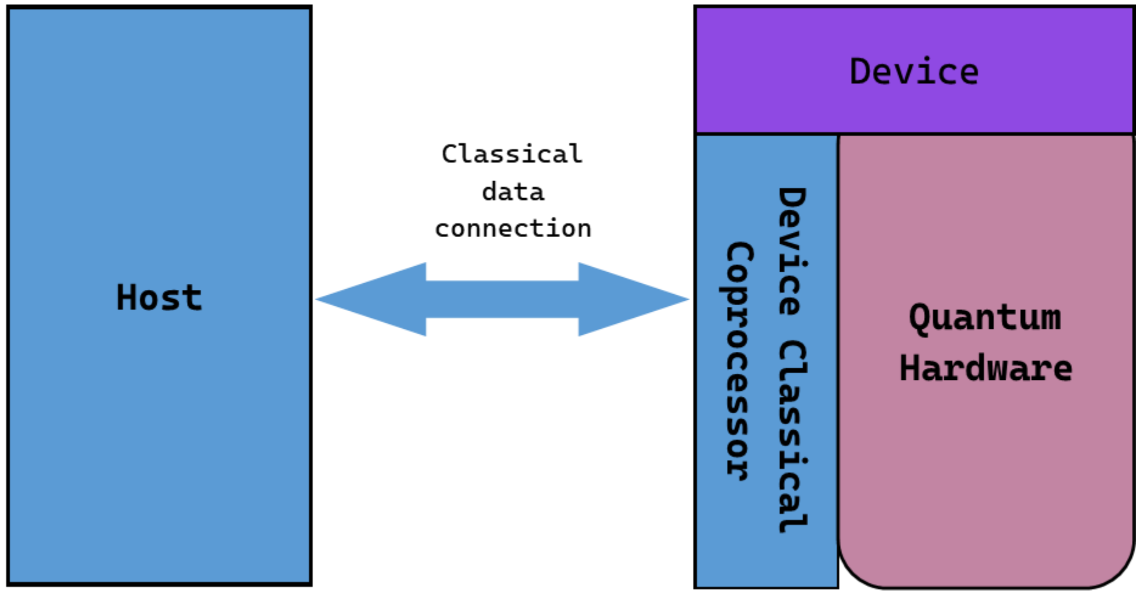
- We can use a more advanced technique for state compression – Matrix Product States.
  - Uses SVD compression, but the caveat is that you lose access to highly entangled states, so **proceed with caution**.
  - iTensor is widely used and a good place to start if you're interested in TN simulation, https://itensor.org/.
  - Can be particularly effective for simulating quantum annealing.



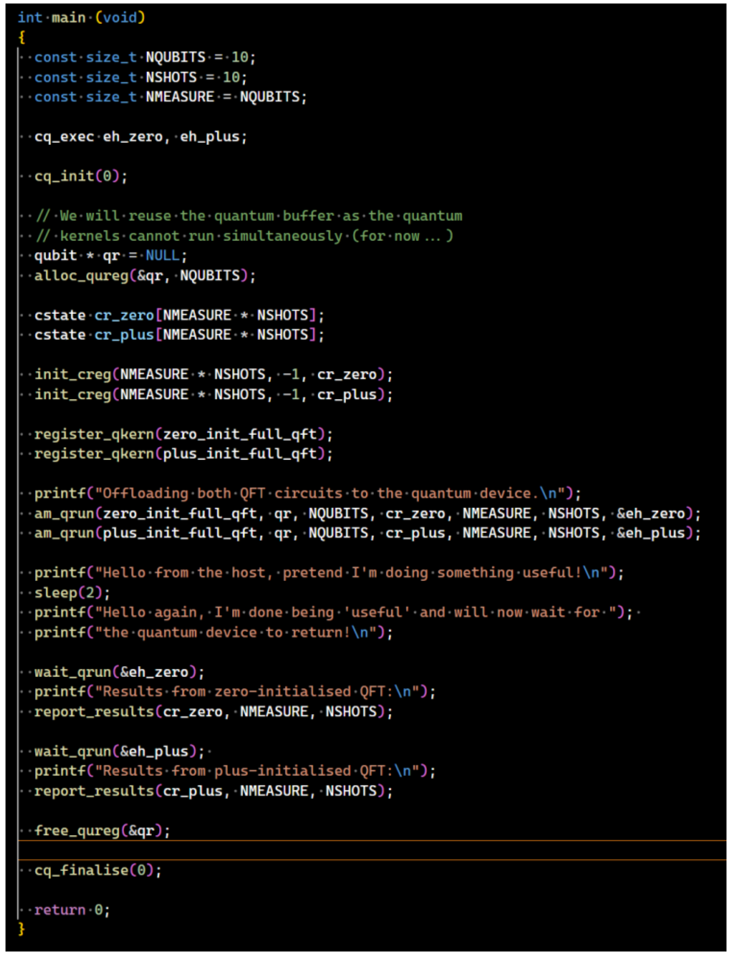
Full image, $\chi = 807$. Compressed image, $\chi = 10$.

Simulated Quantum Offloading

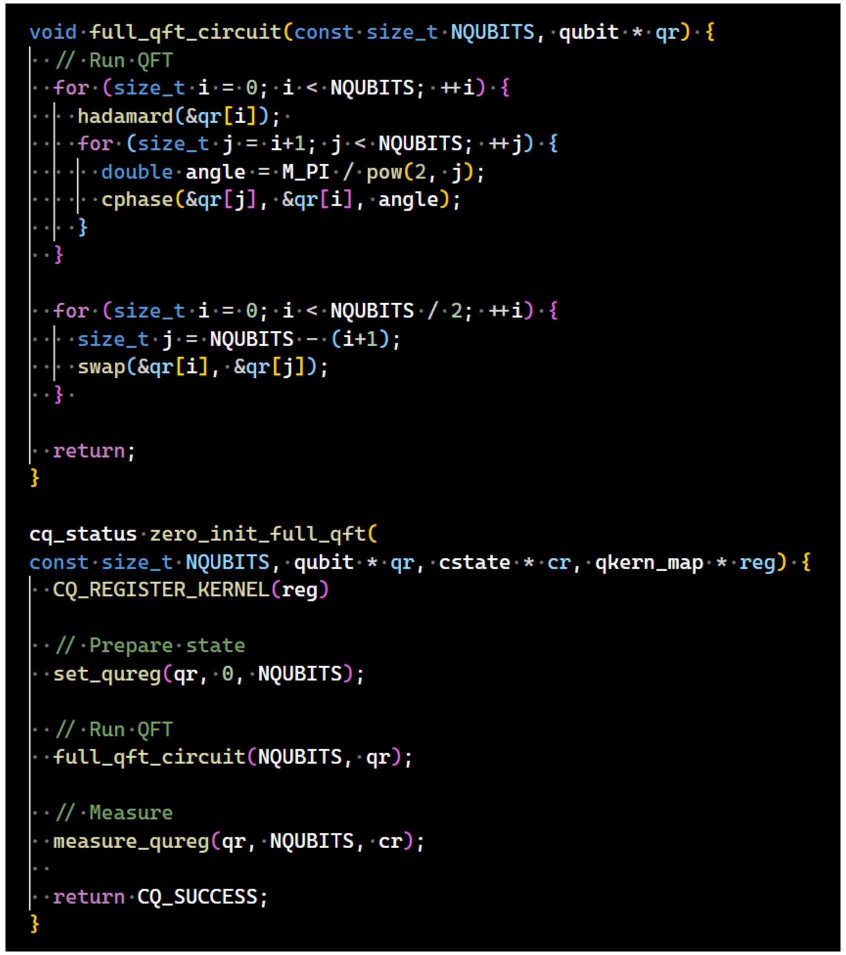
- A statevector simulation contains quite a bit more information than can easily be extracted from a quantum computer.
    - Can lead to unwitting “cheating” when working on quantum algorithms.
- One solution is to use an offloading simulator instead!
- We have developed CQ (arXiv:2508.10854) a specification for a C-like API for quantum computing, and have built an offloading simulator on top of QuEST (GitHub).
    - Caution: it’s an experimental implementation of an experimental specification!
- For something a *bit* more mature you can try NVIDIA CUDA-Q.



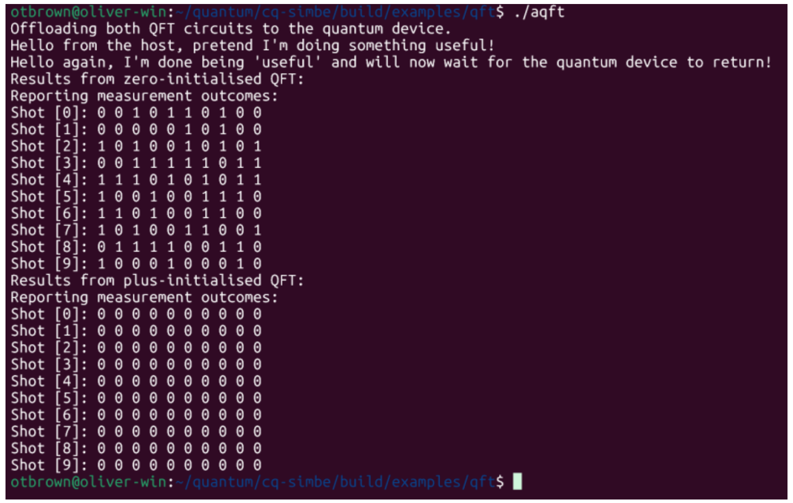
Simulated Quantum Offloading



```c
int main (void)
{
  const size_t NQUBITS = 10;
  const size_t NSHOTS = 10;
  const size_t NMEASURE = NQUBITS;

  cq_exec eh_zero, eh_plus;

  cq_init(0);

  // We will reuse the quantum buffer as the quantum
  // kernels cannot run simultaneously (for now ... )
  qubit * qr = NULL;
  alloc_qureg(&qr, NQUBITS);

  cstate cr_zero[NMEASURE * NSHOTS];
  cstate cr_plus[NMEASURE * NSHOTS];

  init_creg(NMEASURE * NSHOTS, -1, cr_zero);
  init_creg(NMEASURE * NSHOTS, -1, cr_plus);

  register_qkern(zero_init_full_qft);
  register_qkern(plus_init_full_qft);

  printf("Offloading both QFT circuits to the quantum device.\n");
  am_qrun(zero_init_full_qft, qr, NQUBITS, cr_zero, NMEASURE, NSHOTS, &eh_zero);
  am_qrun(plus_init_full_qft, qr, NQUBITS, cr_plus, NMEASURE, NSHOTS, &eh_plus);

  printf("Hello from the host, pretend I'm doing something useful!\n");
  sleep(2);
  printf("Hello again, I'm done being 'useful' and will now wait for ");
  printf("the quantum device to return!\n");

  wait_qrun(&eh_zero);
  printf("Results from zero-initialised QFT:\n");
  report_results(cr_zero, NMEASURE, NSHOTS);

  wait_qrun(&eh_plus);
  printf("Results from plus-initialised QFT:\n");
  report_results(cr_plus, NMEASURE, NSHOTS);

  free_qureg(&qr);

  cq_finalise(0);

  return 0;
}
```



```c
void full_qft_circuit(const size_t NQUBITS, qubit * qr) {
  // Run QFT
  for (size_t i = 0; i < NQUBITS; ++i) {
    hadamard(&qr[i]);
    for (size_t j = i+1; j < NQUBITS; ++j) {
      double angle = M_PI / pow(2, j);
      cphase(&qr[j], &qr[i], angle);
    }
  }

  for (size_t i = 0; i < NQUBITS / 2; ++i) {
    size_t j = NQUBITS - (i+1);
    swap(&qr[i], &qr[j]);
  }

  return;
}

cq_status zero_init_full_qft(
const size_t NQUBITS, qubit * qr, cstate * cr, qkern_map * reg) {
  CQ_REGISTER_KERNEL(reg)

  // Prepare state
  set_qureg(qr, 0, NQUBITS);

  // Run QFT
  full_qft_circuit(NQUBITS, qr);

  // Measure
  measure_qureg(qr, NQUBITS, cr);

  return CQ_SUCCESS;
}
```



```
otbrown@oliver-win:~/quantum/cq-simbe/build/examples/qft$ ./aqft
Offloading both QFT circuits to the quantum device.
Hello from the host, pretend I'm doing something useful!
Hello again, I'm done being 'useful' and will now wait for the quantum device to return!
Results from zero-initialised QFT:
Reporting measurement outcomes:
Shot [0]: 0 0 1 0 1 1 0 1 0 0
Shot [1]: 0 0 0 0 0 1 0 1 0 0
Shot [2]: 1 0 1 0 0 1 0 1 0 1
Shot [3]: 0 0 1 1 1 1 1 0 1 1
Shot [4]: 1 1 1 0 1 0 1 0 1 1
Shot [5]: 1 0 0 1 0 0 1 1 1 0
Shot [6]: 1 1 0 1 0 0 1 1 0 0
Shot [7]: 1 0 1 0 0 1 1 0 0 1
Shot [8]: 0 1 1 1 1 0 0 1 1 0
Shot [9]: 1 0 0 0 1 0 0 0 1 0
Results from plus-initialised QFT:
Reporting measurement outcomes:
Shot [0]: 0 0 0 0 0 0 0 0 0 0
Shot [1]: 0 0 0 0 0 0 0 0 0 0
Shot [2]: 0 0 0 0 0 0 0 0 0 0
Shot [3]: 0 0 0 0 0 0 0 0 0 0
Shot [4]: 0 0 0 0 0 0 0 0 0 0
Shot [5]: 0 0 0 0 0 0 0 0 0 0
Shot [6]: 0 0 0 0 0 0 0 0 0 0
Shot [7]: 0 0 0 0 0 0 0 0 0 0
Shot [8]: 0 0 0 0 0 0 0 0 0 0
Shot [9]: 0 0 0 0 0 0 0 0 0 0
otbrown@oliver-win:~/quantum/cq-simbe/build/examples/qft$
```