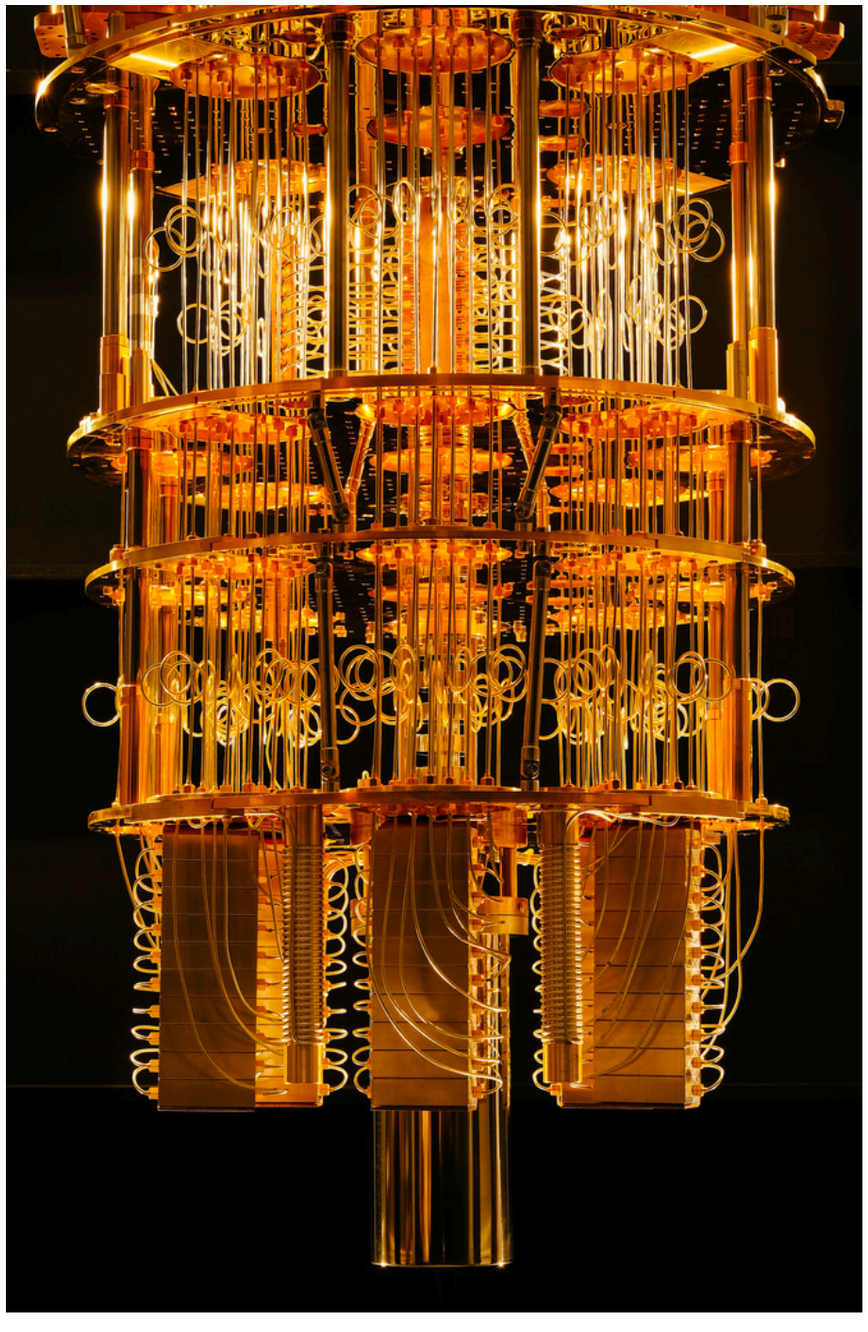
SNAPL 2019

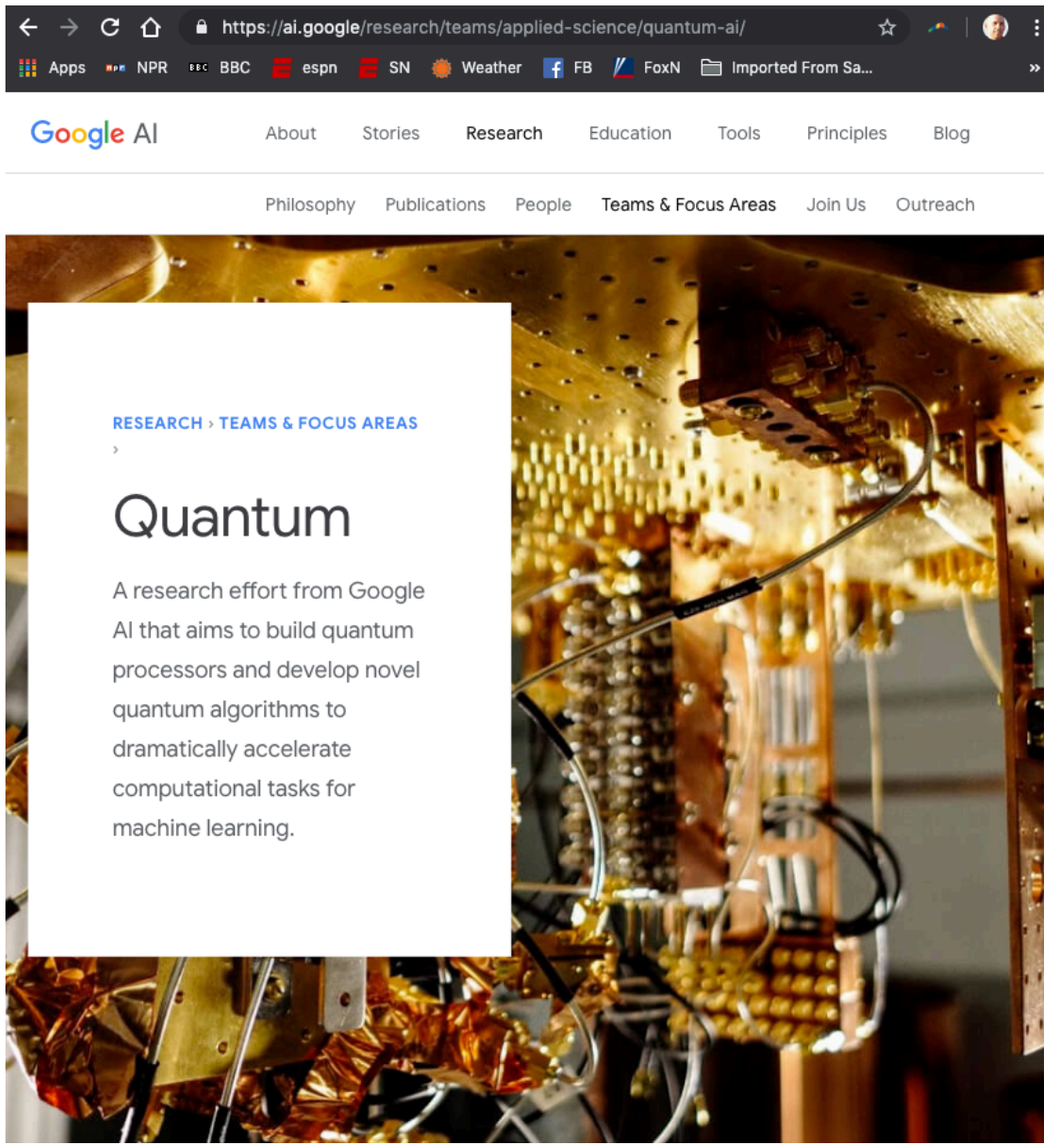
Robert Rand, Kesha Heitala and *Michael Hicks*

Quantum Computing

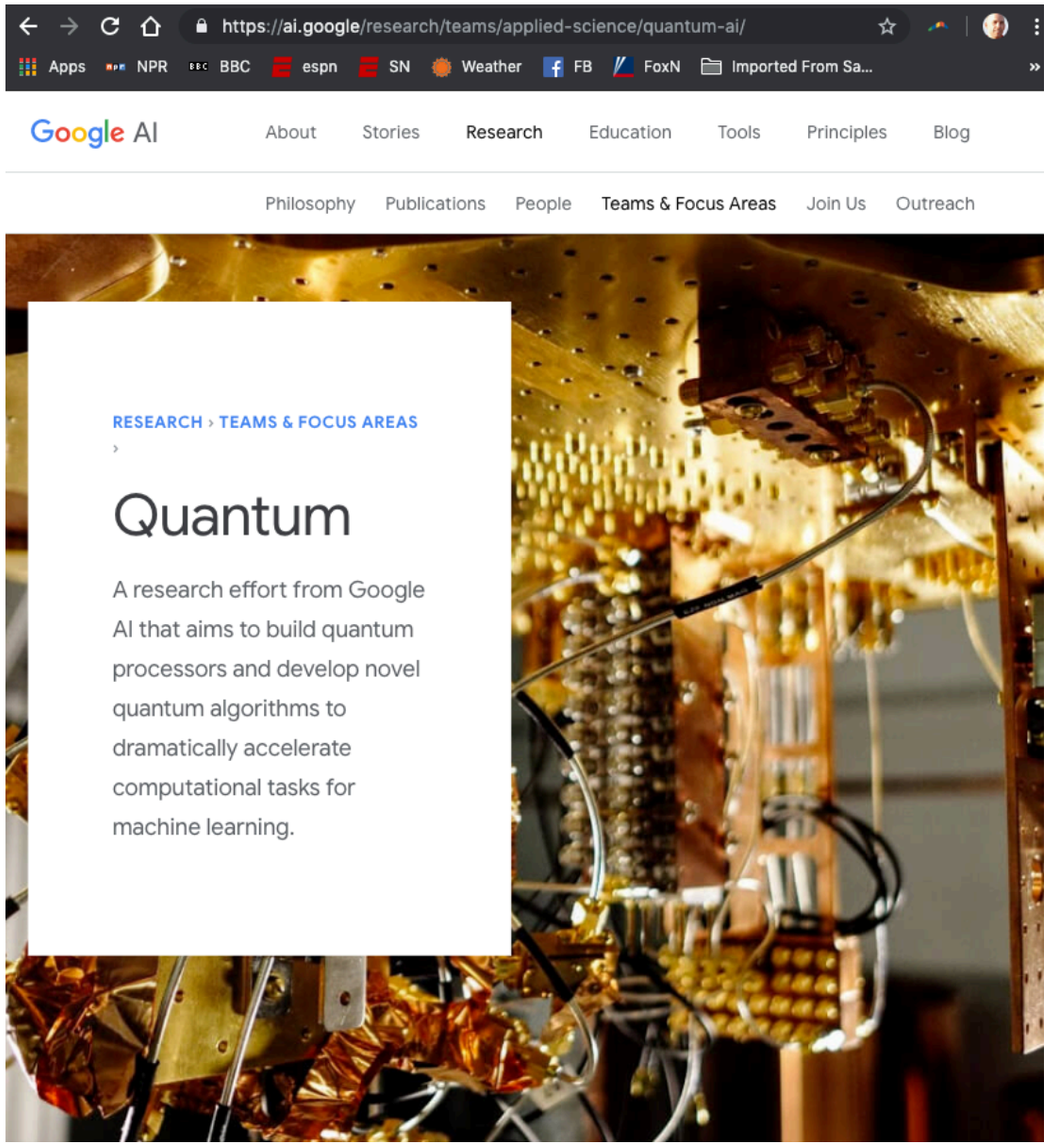


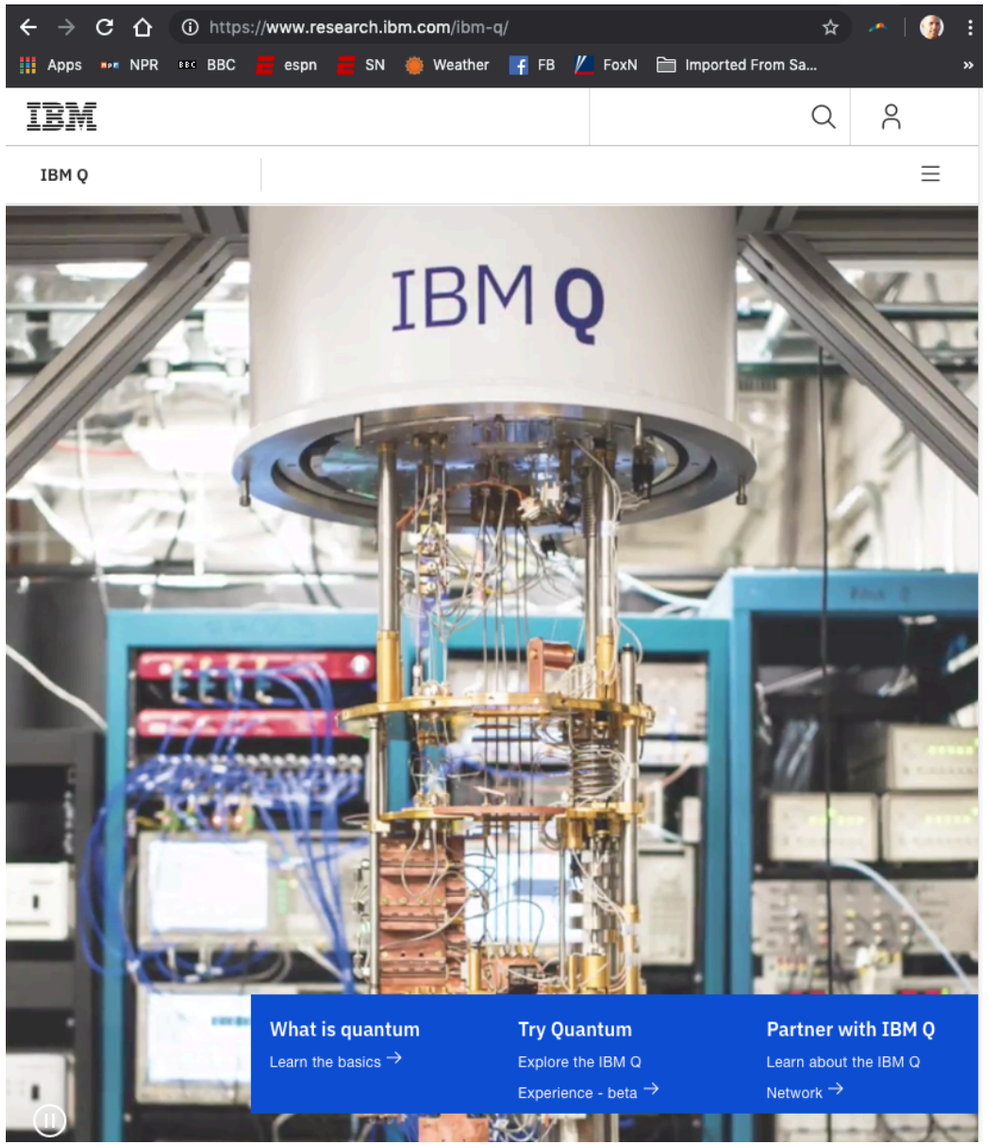
Quantum Computing

Quantum Computing

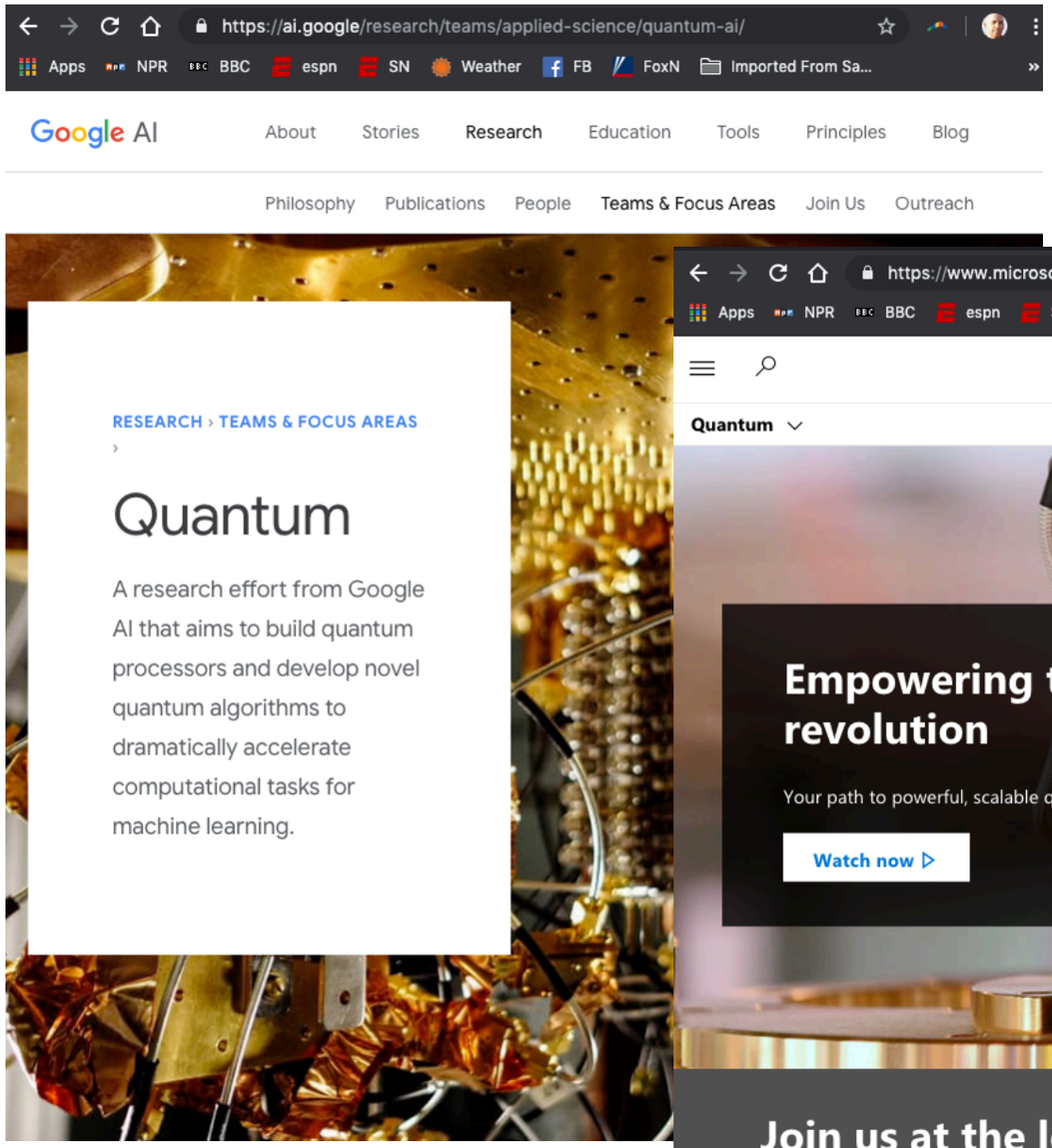


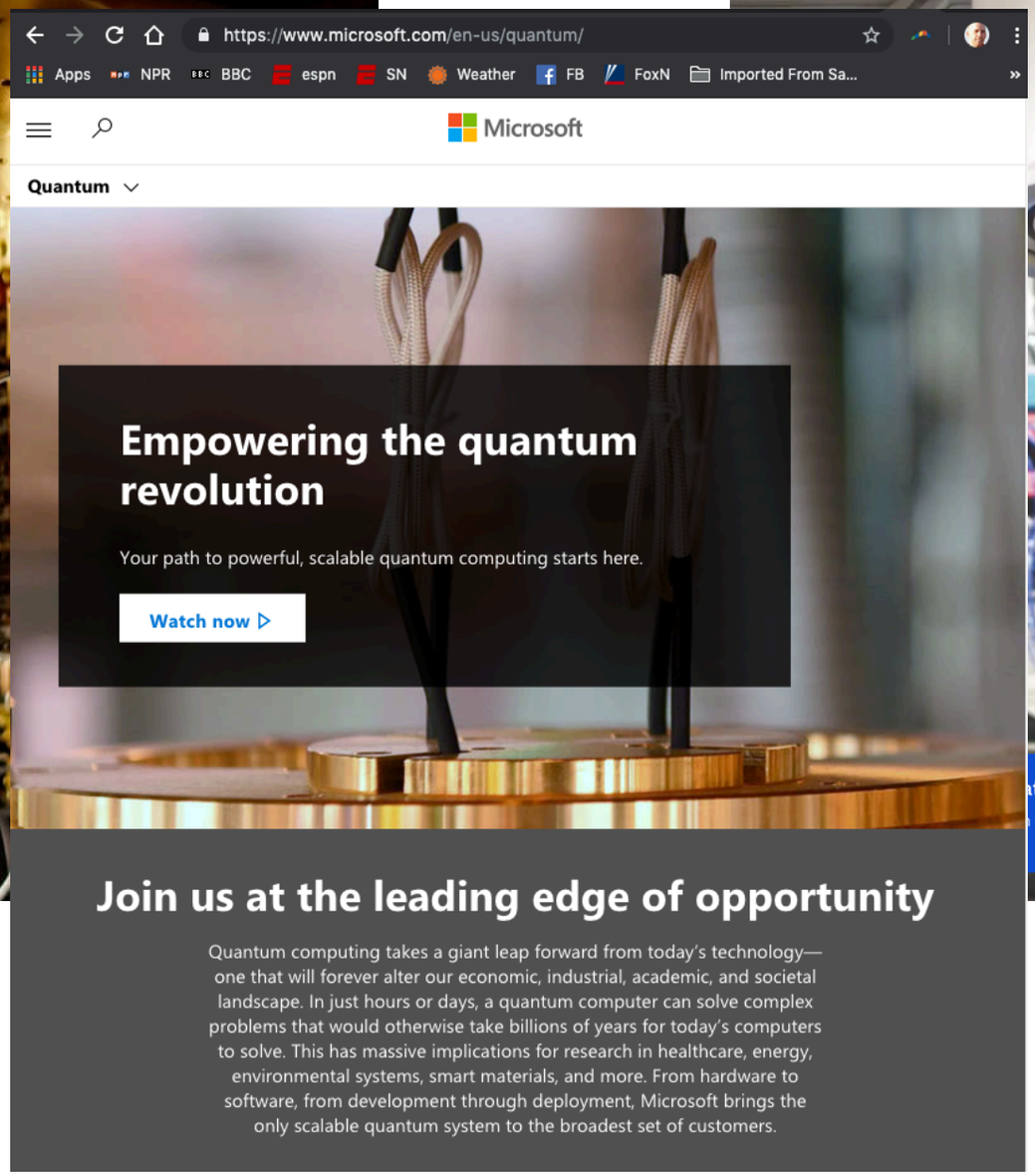
Quantum Computing

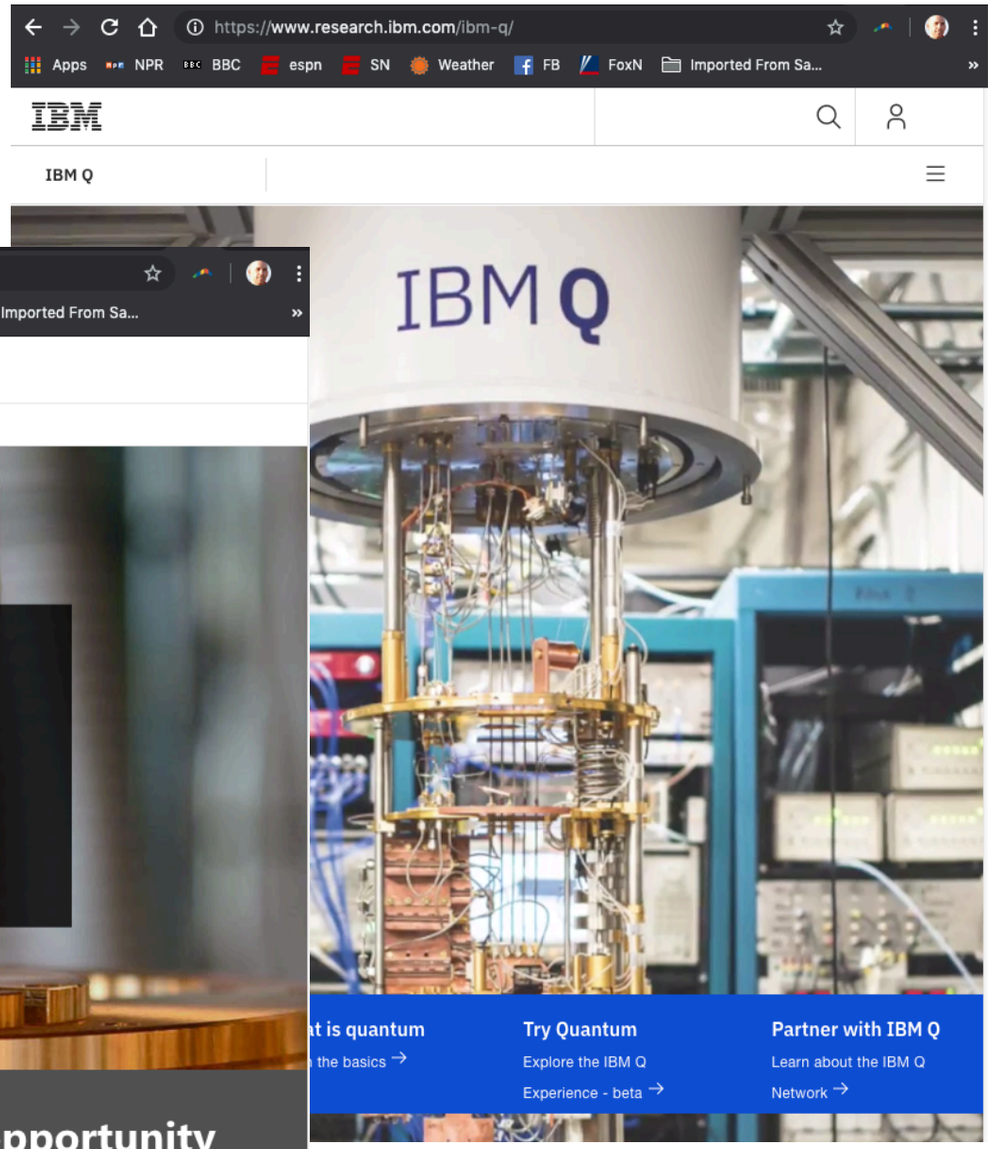




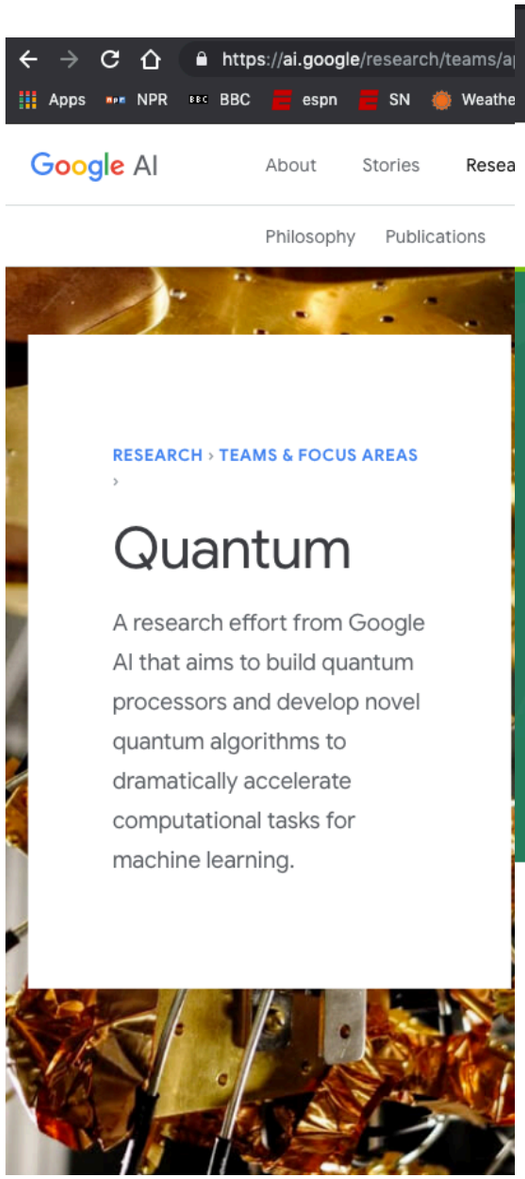
Quantum Computing







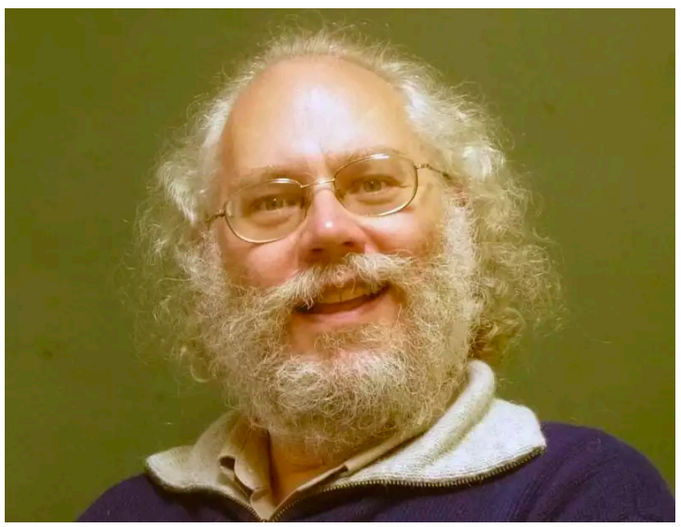
Quantum Computing



Why?

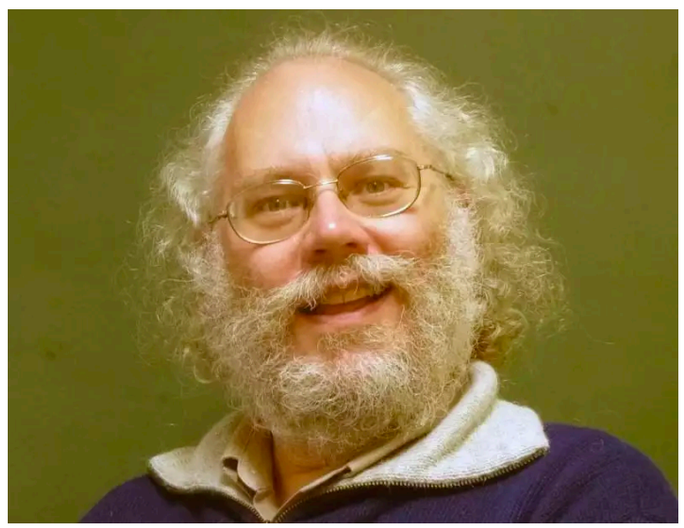
Why?

- Factoring large numbers

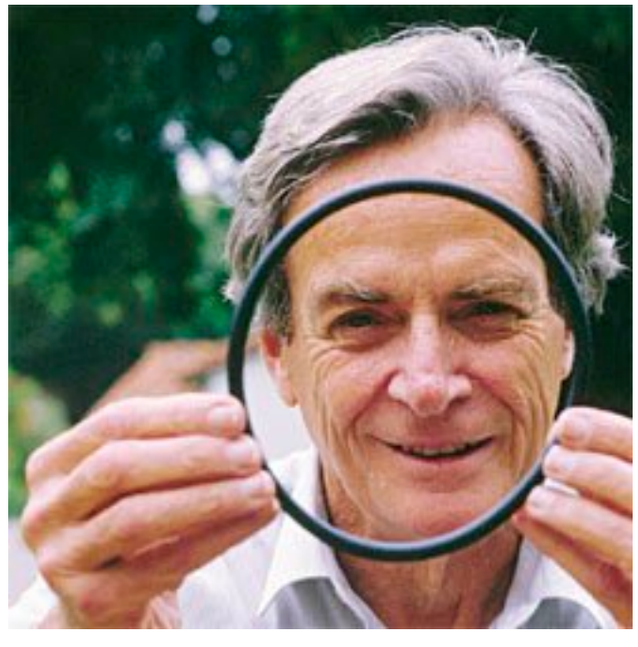


Why?

- Factoring large numbers

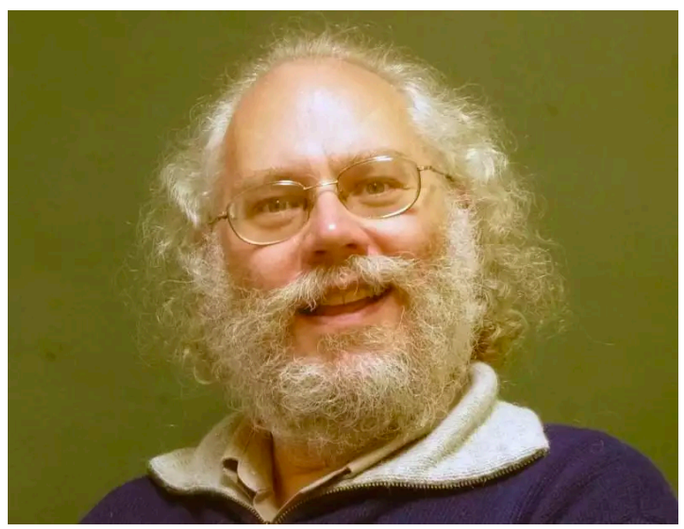


- Simulating small physical systems

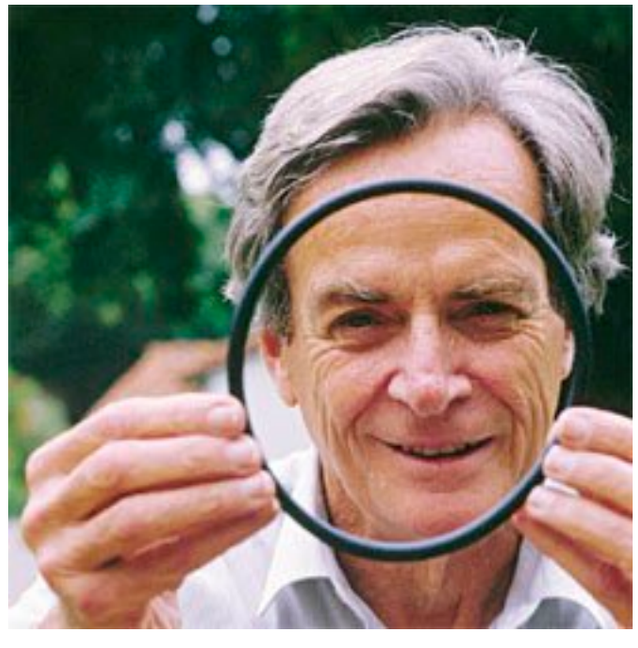


Why?

- Factoring large numbers



- Simulating small physical systems
- Uncrackable crypto systems
- Solving linear equations



Challenges

In general

- Answer may be unknown
- Simulation is intractable
- Breakpoints break things (opening the box kills the cat)

In the near term

- Execution is expensive, and (highly) error prone
- Computing resources (e.g., qubits) are scarce

PL to the Rescue?

PL to the Rescue?

- Compilers and optimizations, language design, formal methods, approximate computing, probabilistic programming, …

- Sound familiar?

PL to the Rescue?

- Compilers and optimizations, language design, formal methods, approximate computing, probabilistic programming, …
  - Sound familiar?
- Goal of this talk: Get you interested to work on QC with your PL hat on !

Qubits

Superposition: Qubits can be in multiple states (0 or 1) at once

Qubits

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Superposition: Qubits can be in multiple states (0 or 1) at once

Qubits

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$|\alpha|^2 + |\beta|^2 = 1$$

Superposition: Qubits can be in multiple states (0 or 1) at once

Qubits

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$|0\rangle$

$$|\alpha|^2 + |\beta|^2 = 1$$

Superposition: Qubits can be in multiple states (0 or 1) at once

Qubits

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$|0\rangle$ $|1\rangle$

$$|\alpha|^2 + |\beta|^2 = 1$$

Superposition: Qubits can be in multiple states (0 or 1) at once

Measurement

Measurement: Looking at a qubit probabilistically turns it into a bit.

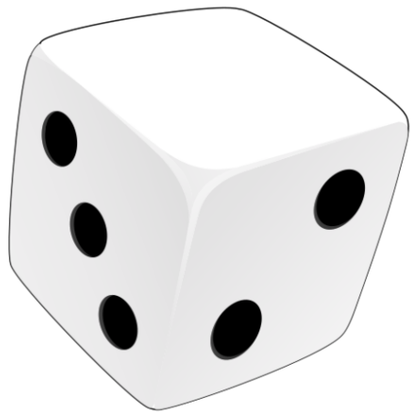
Measurement

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Measurement: Looking at a qubit probabilistically turns it into a bit.

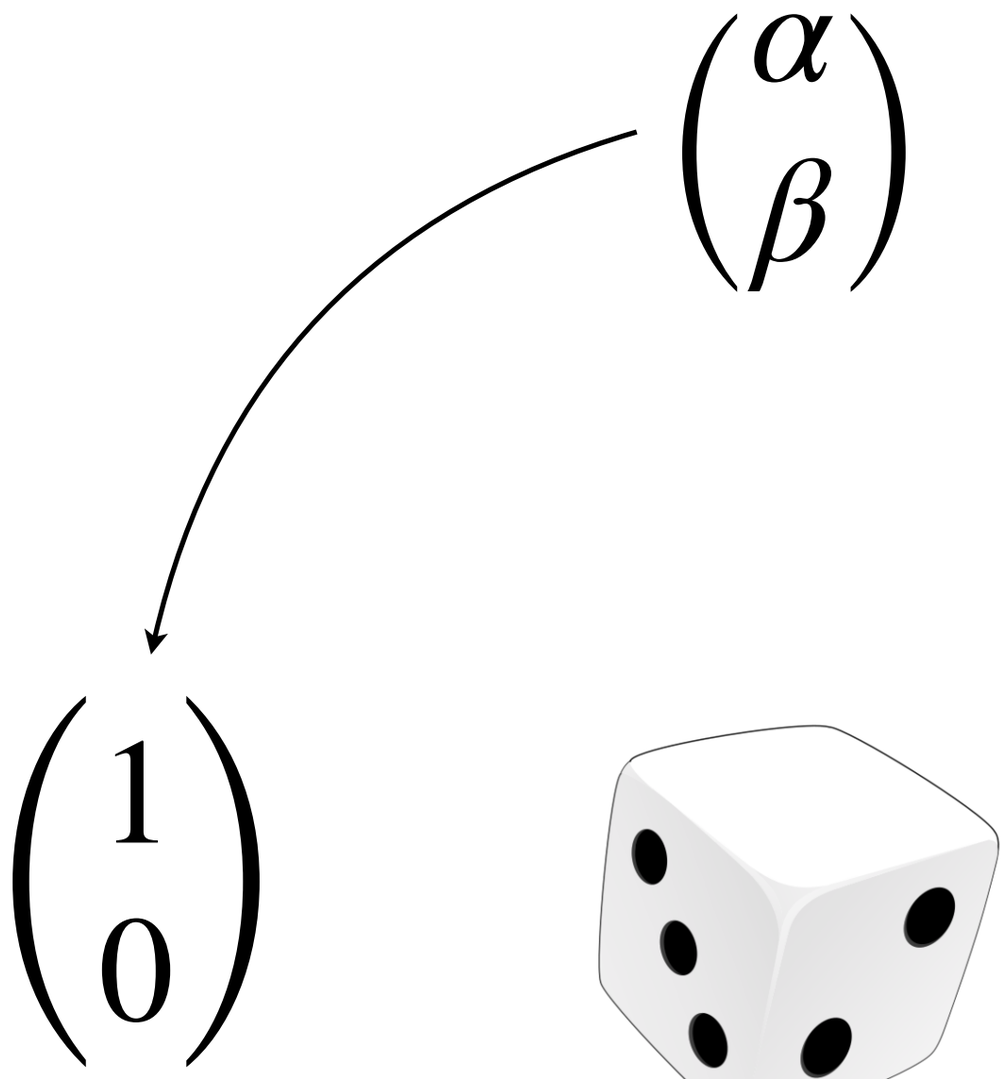
Measurement

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$



Measurement: Looking at a qubit probabilistically turns it into a bit.

Measurement



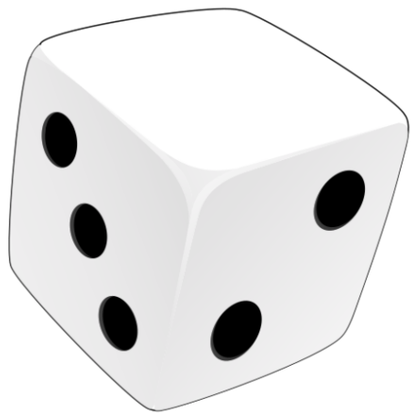
Measurement: Looking at a qubit probabilistically turns it into a bit.

Measurement

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

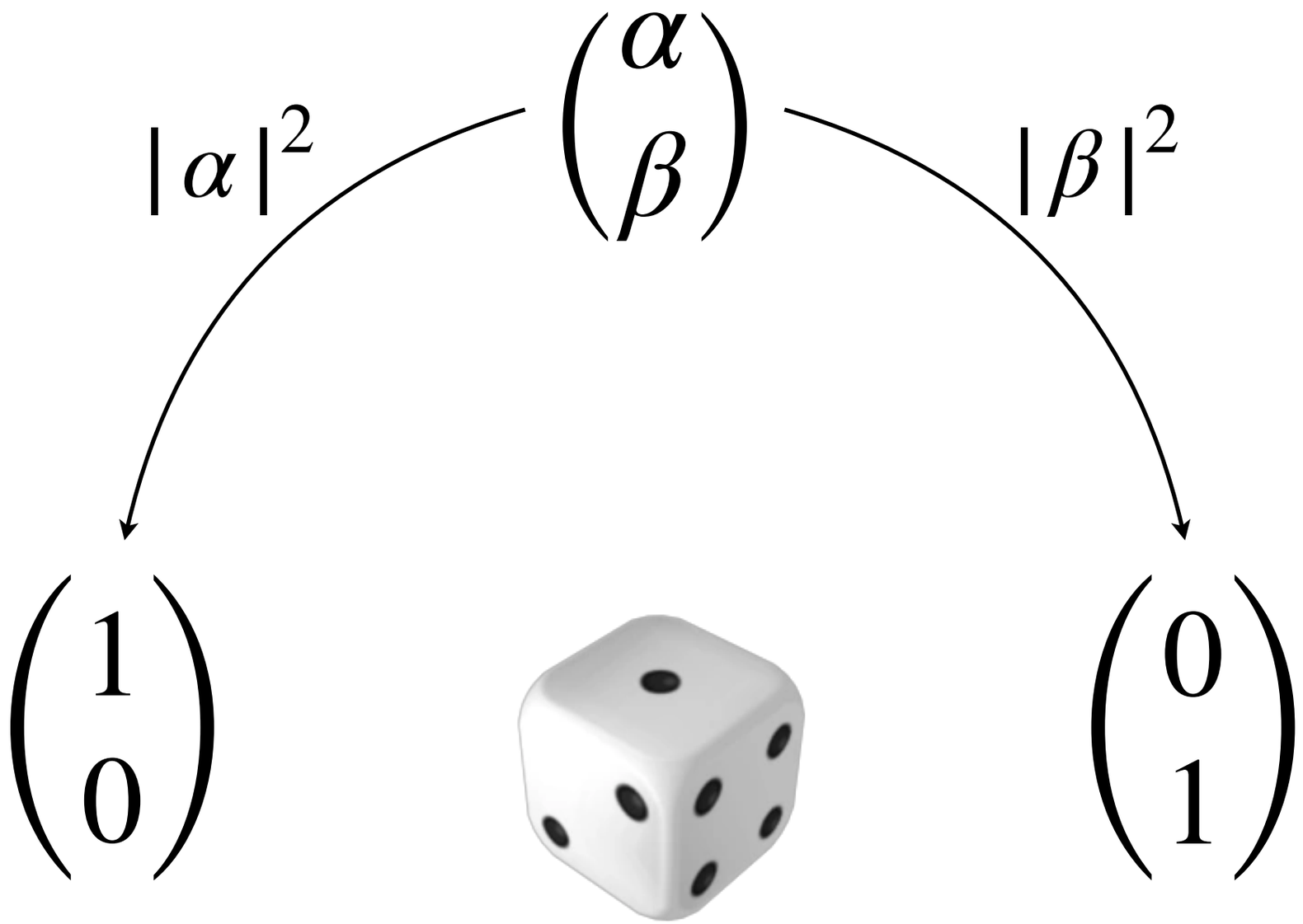
$$|\alpha|^2$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



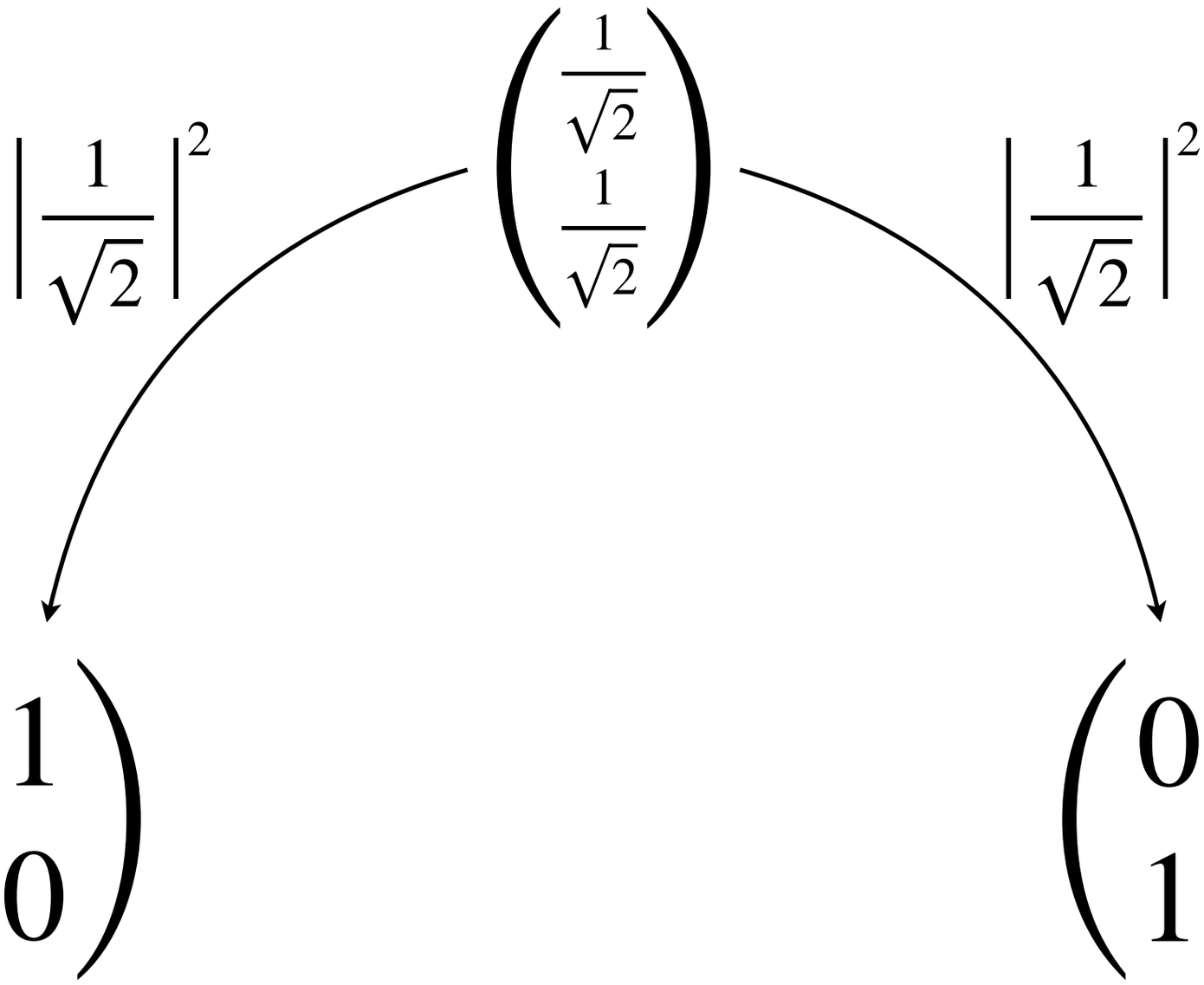
Measurement: Looking at a qubit probabilistically turns it into a bit.

Measurement

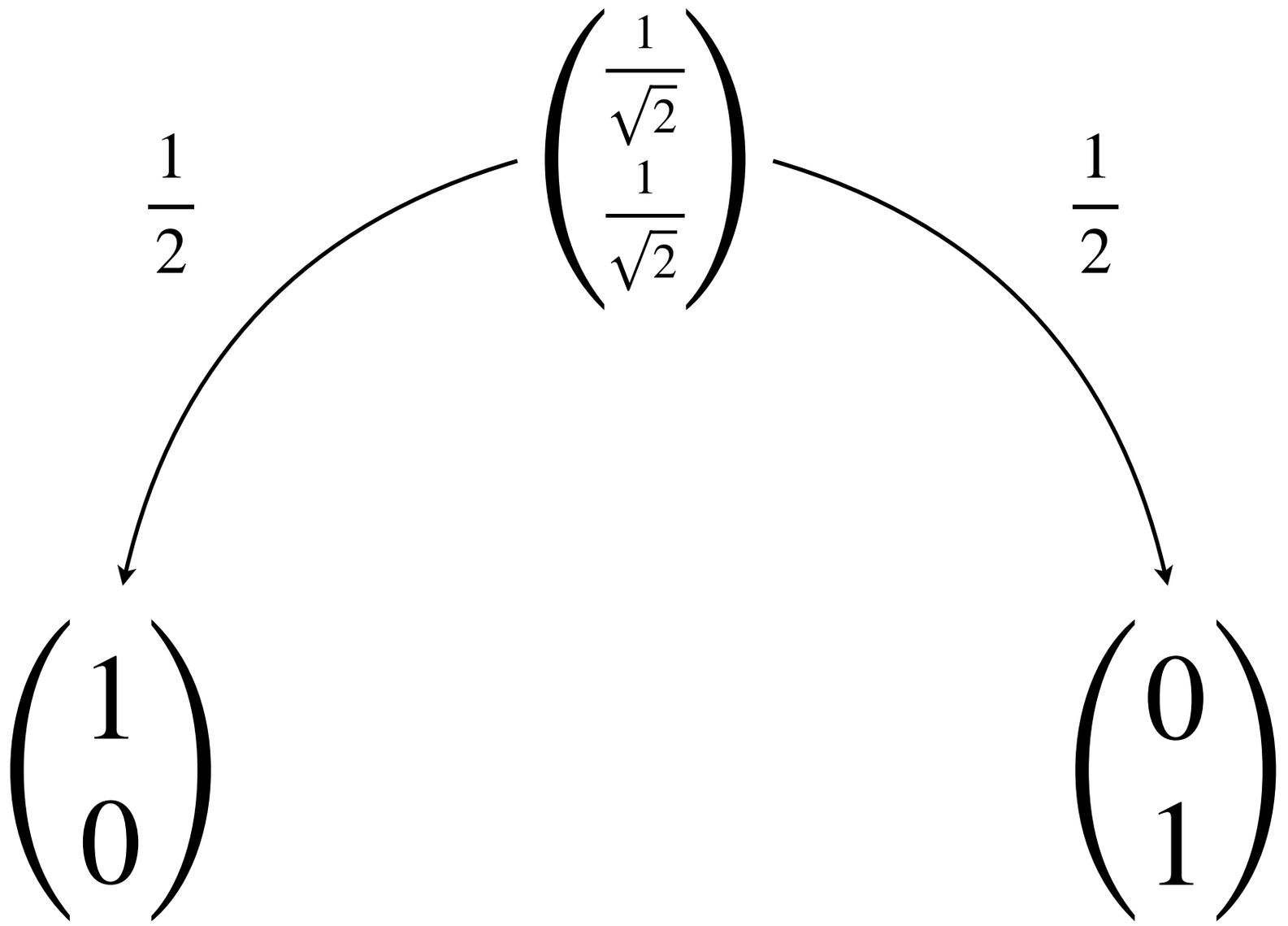


Measurement: Looking at a qubit probabilistically turns it into a bit.

Measurement



Measurement



Unitaries

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Unitaries are operators that transform (“evolve”) quantum states

Unitaries

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}$$

Unitaries are operators that transform (“evolve”) quantum states

Unitaries

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

Unitaries are operators that transform (“evolve”) quantum states

Unitaries

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

Unitaries are operators that transform (“evolve”) quantum states

Unitaries

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix}$$

Unitaries are operators that transform (“evolve”) quantum states

Unitaries

$$\mathbf{H}\,|0\rangle = |+\rangle$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix}$$

This is the *Hadamard* unitary, labeled *H*

Unitaries

$$\mathbf{H}\,|0\rangle = |+\rangle$$

$$\mathbf{H}\,|+\rangle = |0\rangle$$

This is the *Hadamard* unitary, labeled *H*

Multiple Qubits

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Multi-qubit states can be formed via the tensor product

Multiple Qubits

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Multi-qubit states can be formed via the tensor product

Multiple Qubits

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$|+\rangle \otimes |0\rangle \quad = |+\rangle|0\rangle$$

Multi-qubit states can be formed via the tensor product

Multiqubit Unitaries

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Multiqubit Unitaries

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Multiqubit Unitaries

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} - \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

Multiqubit Unitaries

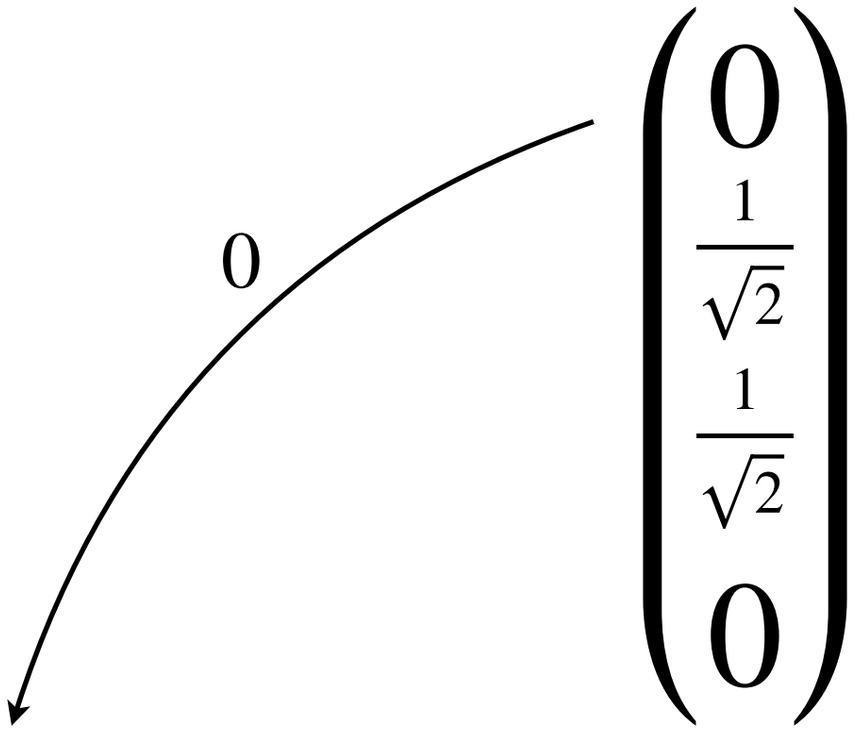
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

CNOT $|+\rangle|0\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$

Measurement 2.0

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

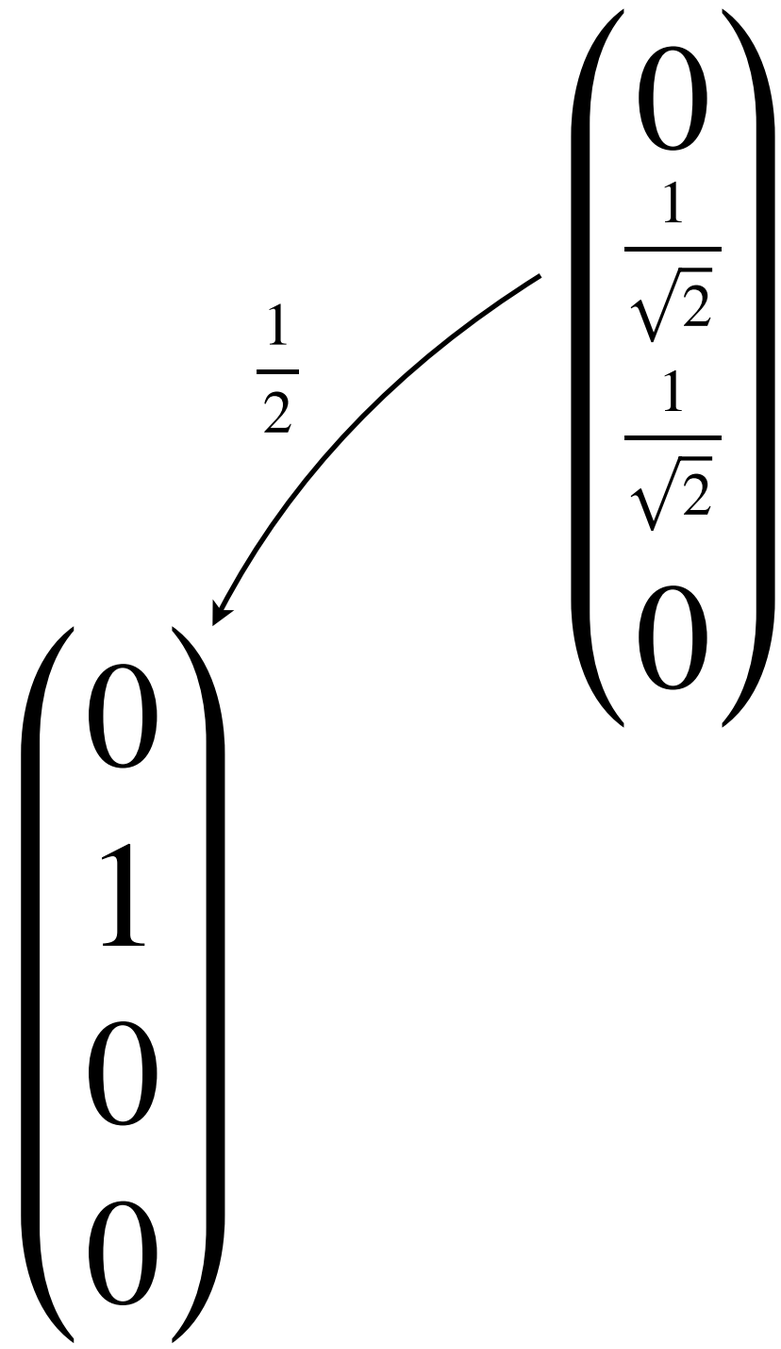
Measurement 2.0



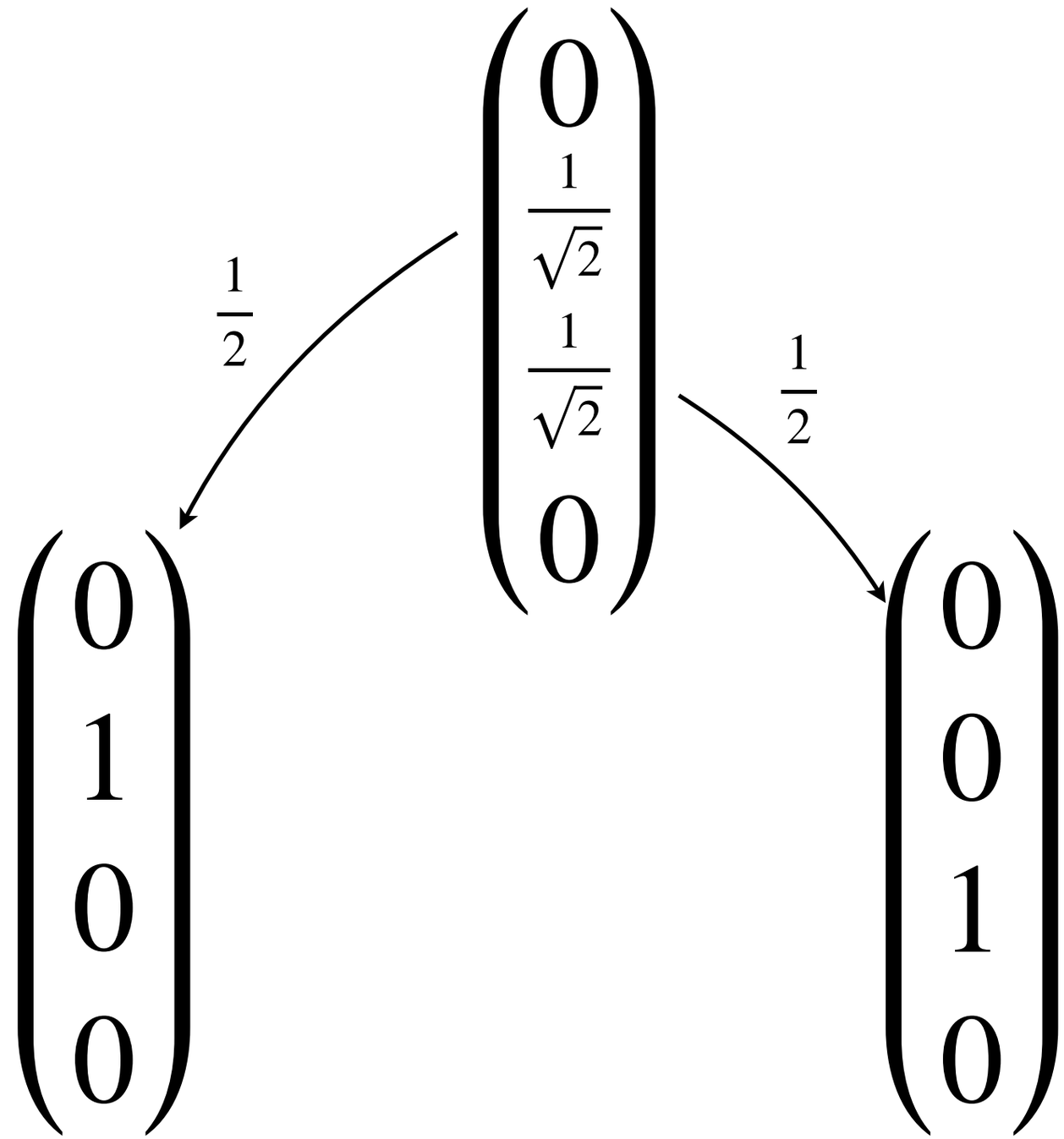
Measurement 2.0

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

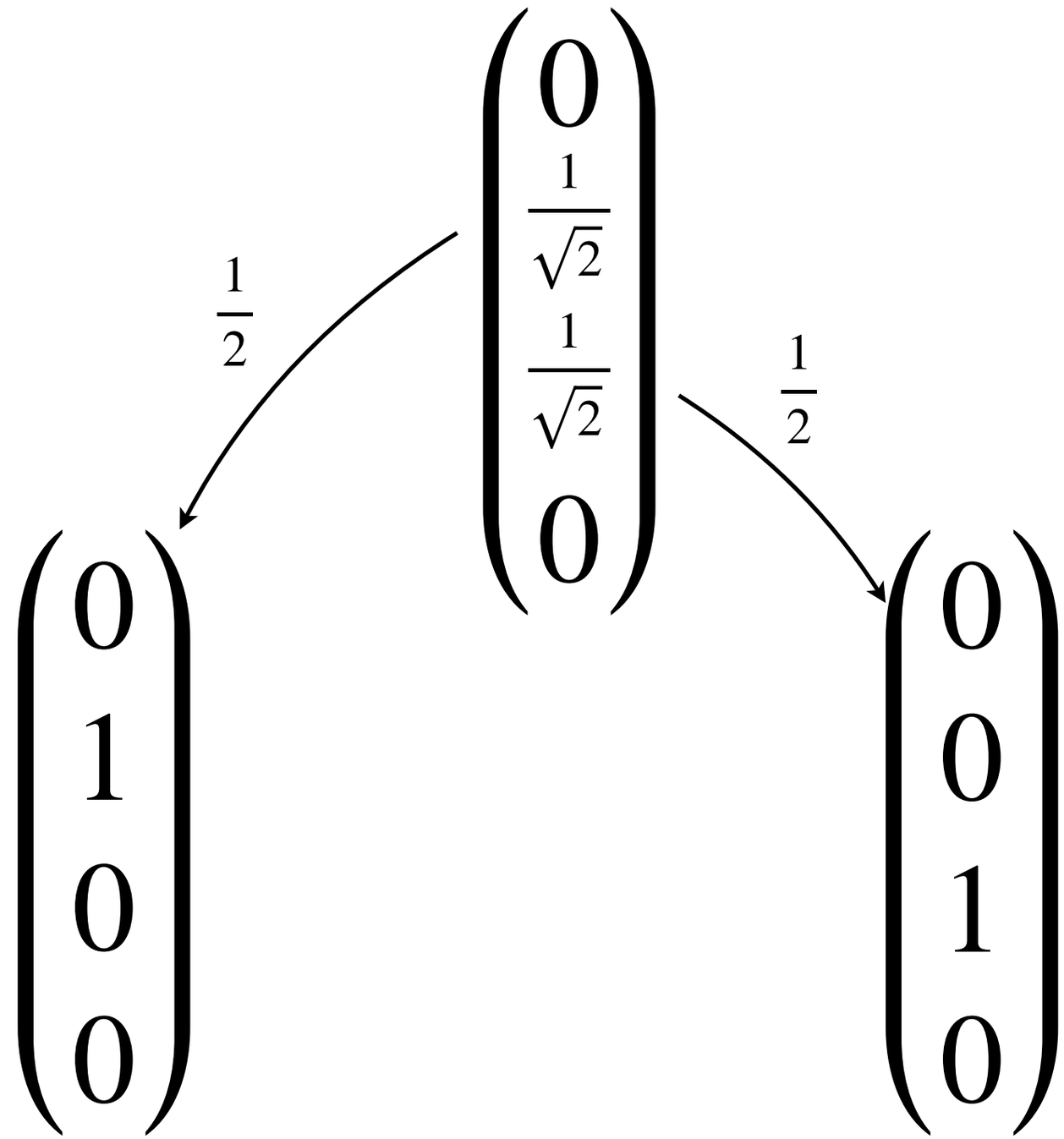
Measurement 2.0



Measurement 2.0



Measurement 2.0



Measurement 2.0

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\frac{1}{2} \quad \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} \quad \frac{1}{2}$$

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

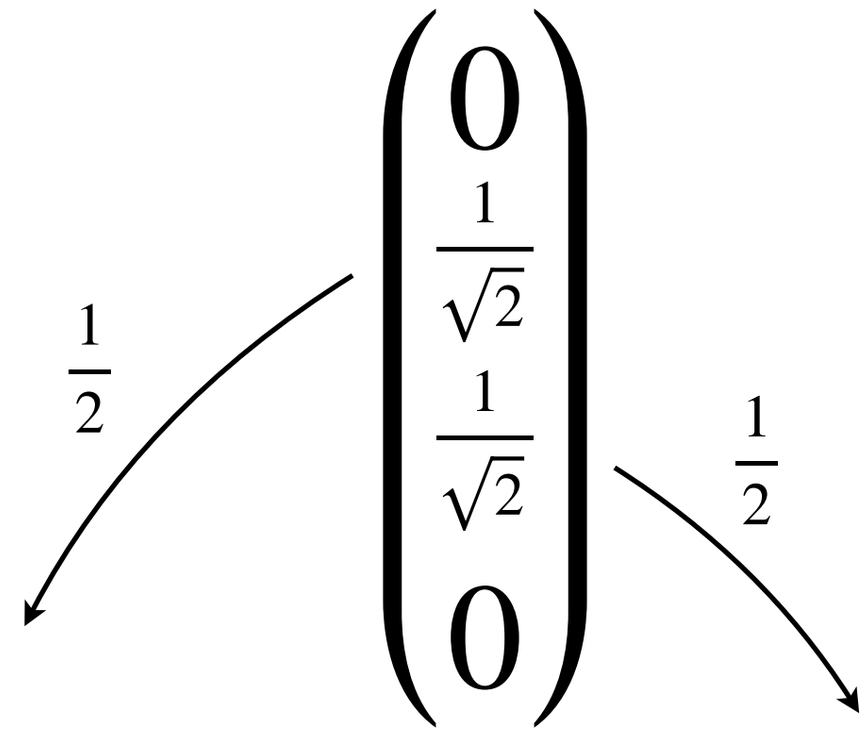
Measurement 2.0

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$\frac{1}{2}$ $\frac{1}{2}$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Measurement 2.0



$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$|01\rangle$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|10\rangle$$

Entanglement

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Entanglement

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Entanglement

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

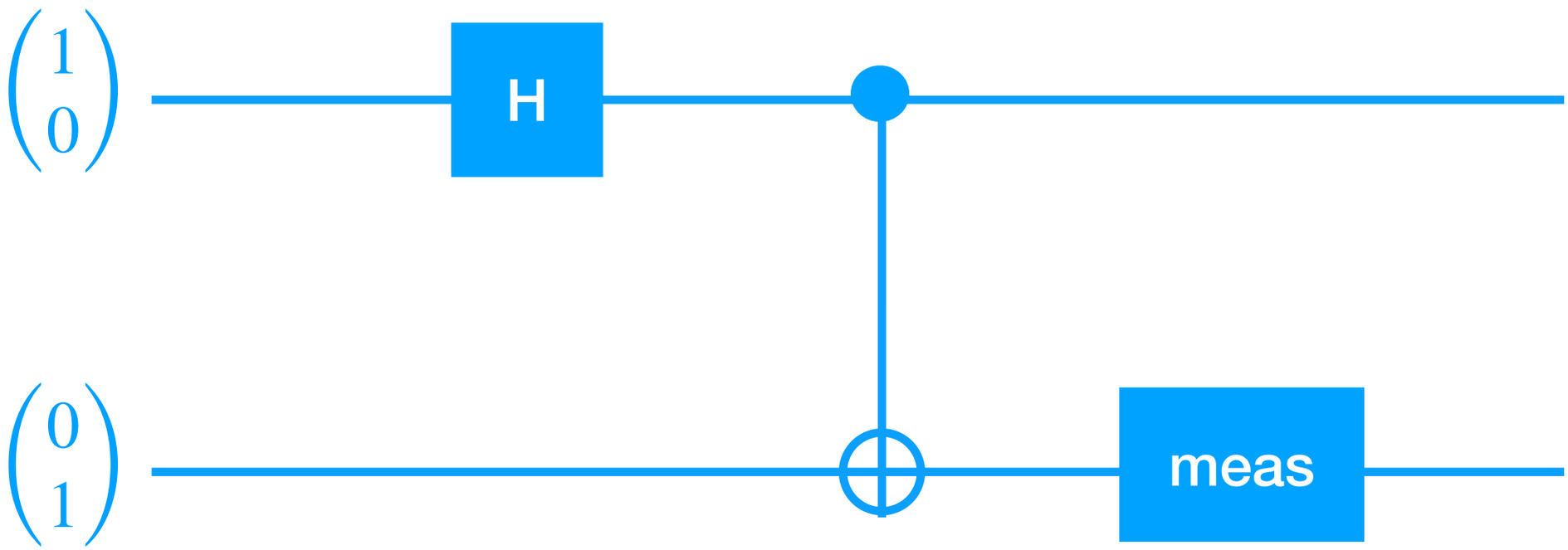
Entanglement

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad ? \otimes ?$$

Entangled qubits are not probabilistically independent — they cannot be decomposed. Connection at a distance!

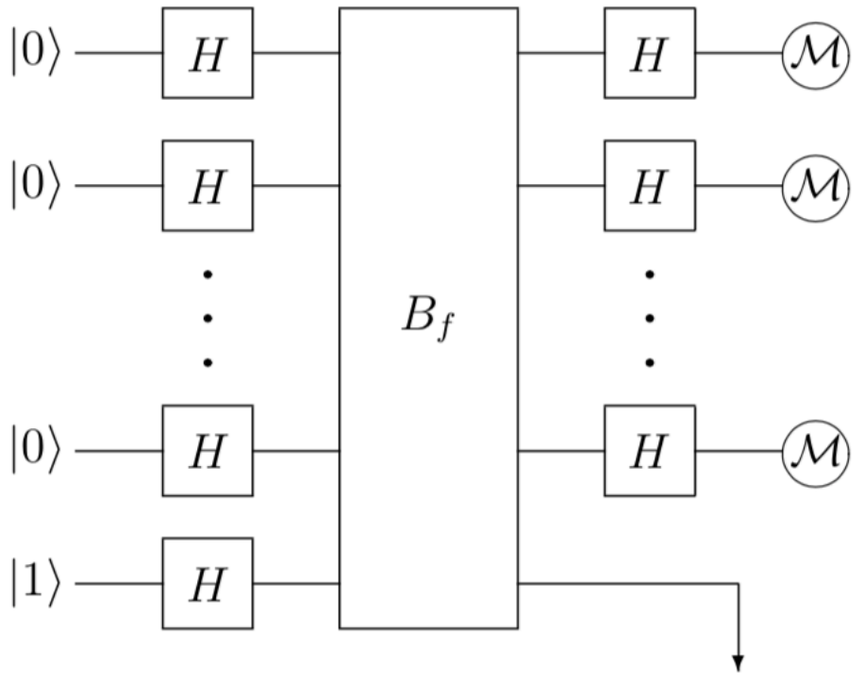
Circuits



(CNOT)

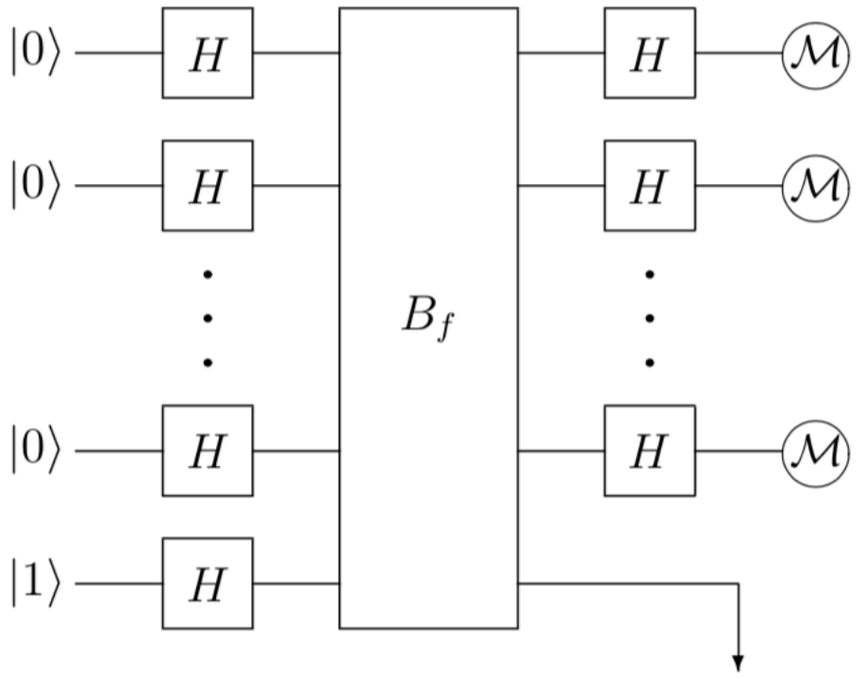
Algorithms

- Common theme of many quantum algorithms
  - Compute over the state space in superposition, essentially doing work “in parallel”
  - measure to produce the final result (and cancel out noise)



Algorithms

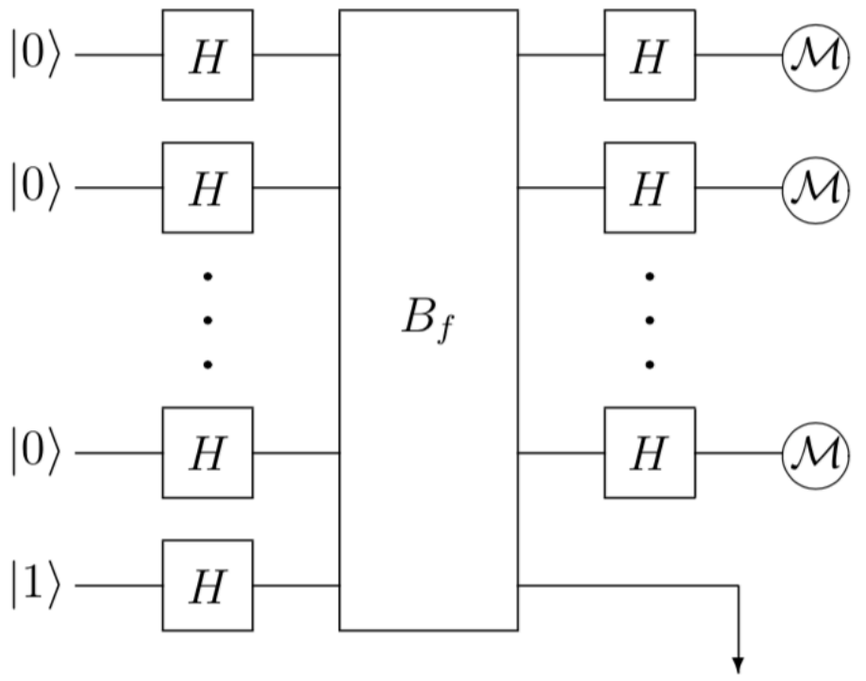
- Common theme of many quantum algorithms
  - Compute over the state space in superposition, essentially doing work “in parallel”
  - measure to produce the final result (and cancel out noise)



- Several challenges
  - Algorithms are probabilistic — must ensure measurement likely to get the right answer

Algorithms

- Common theme of many quantum algorithms
  - Compute over the state space in superposition, essentially doing work “in parallel”
  - measure to produce the final result (and cancel out noise)



- Several challenges
  - Algorithms are probabilistic — must ensure measurement likely to get the right answer
  - Gates in a quantum circuit must be unitary, implying (e.g.,) reversibility — “ancillae” bits used to encode answers

Recall: Challenges

In general

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)

Recall: Challenges

<u>In general</u>

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

***Noisy, Intermediate Scale Quantum* (*NISQ*) Computing era**

—Preskill

Recall: Challenges

In general

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

In the near term

***Noisy, Intermediate Scale Quantum* (*NISQ*) Computing era**

—Preskill

Recall: Challenges

In general

- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

In the near term

- Execution is expensive, and (highly) error prone

***Noisy, Intermediate Scale Quantum* (*NISQ*) Computing era**
—Preskill

Recall: Challenges

In general

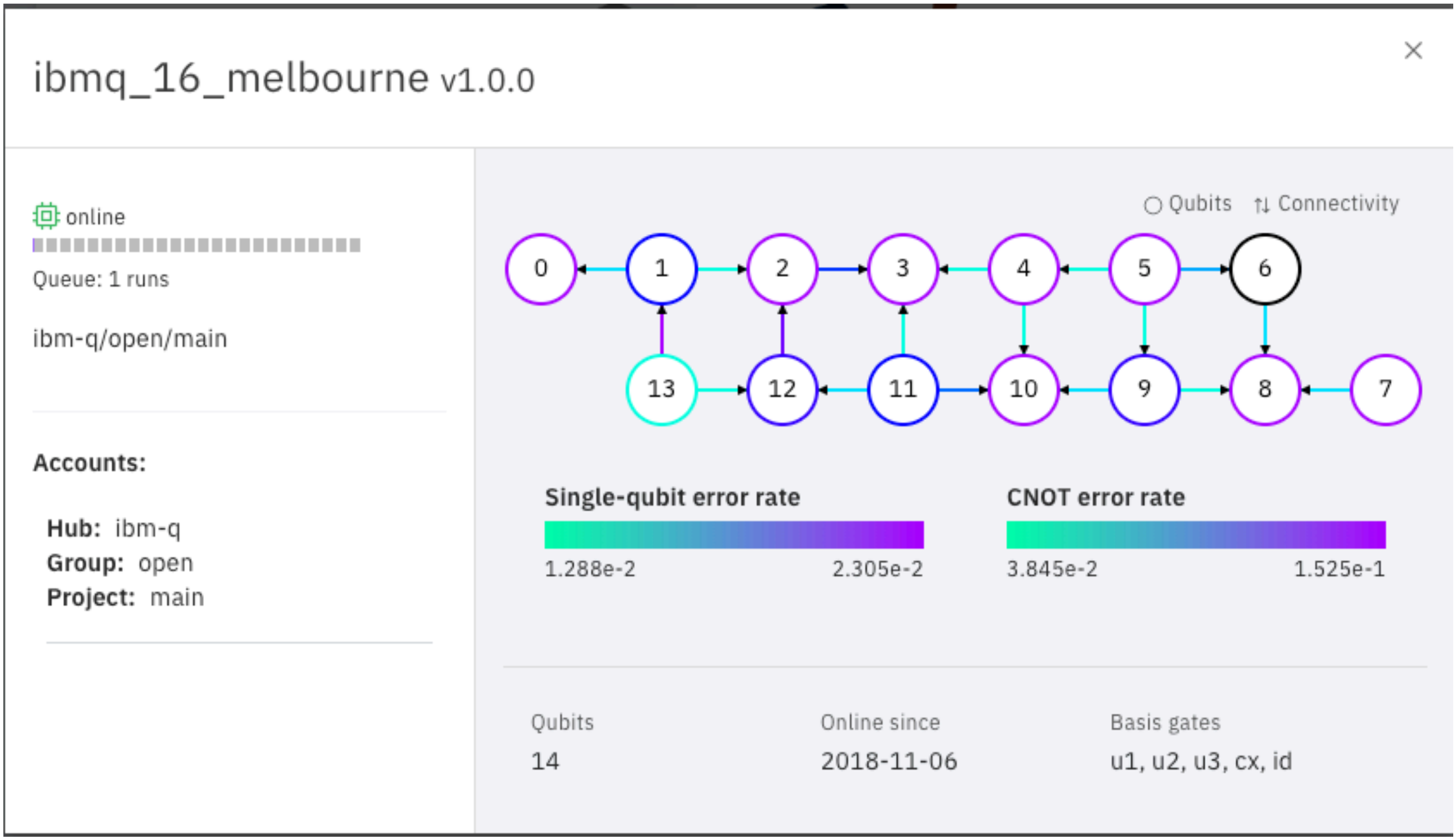
- Algorithms hard to write; answer may be unknown
- Simulation is intractable (states exponential in # qubits)
- Breakpoints break things (opening the box kills the cat)

In the near term

- Execution is expensive, and (highly) error prone
- Computing resources (e.g., qubits) are scarce

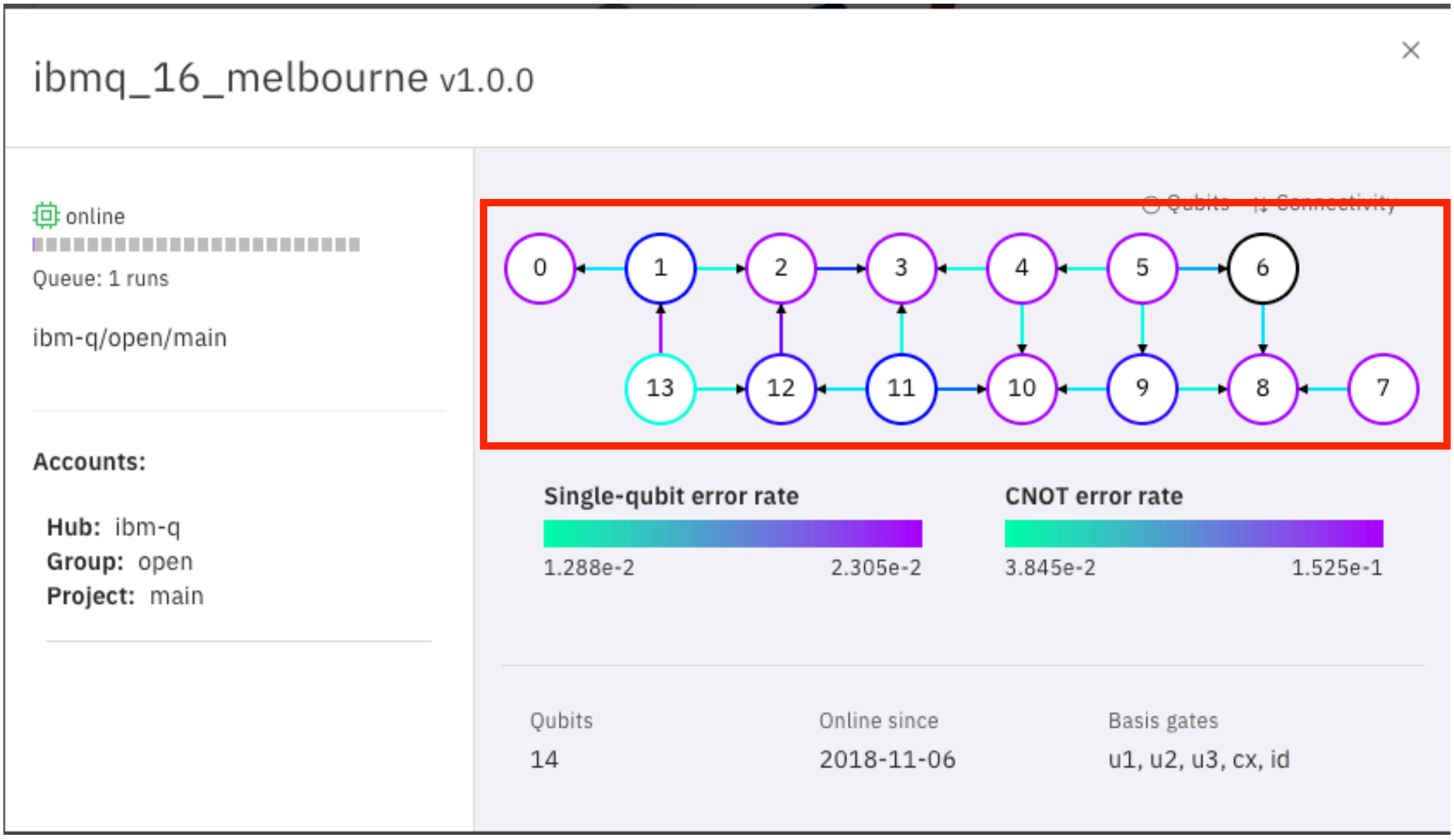
***Noisy, Intermediate Scale Quantum* (*NISQ*) Computing era**
—Preskill

Limited Qubits, Connectivity



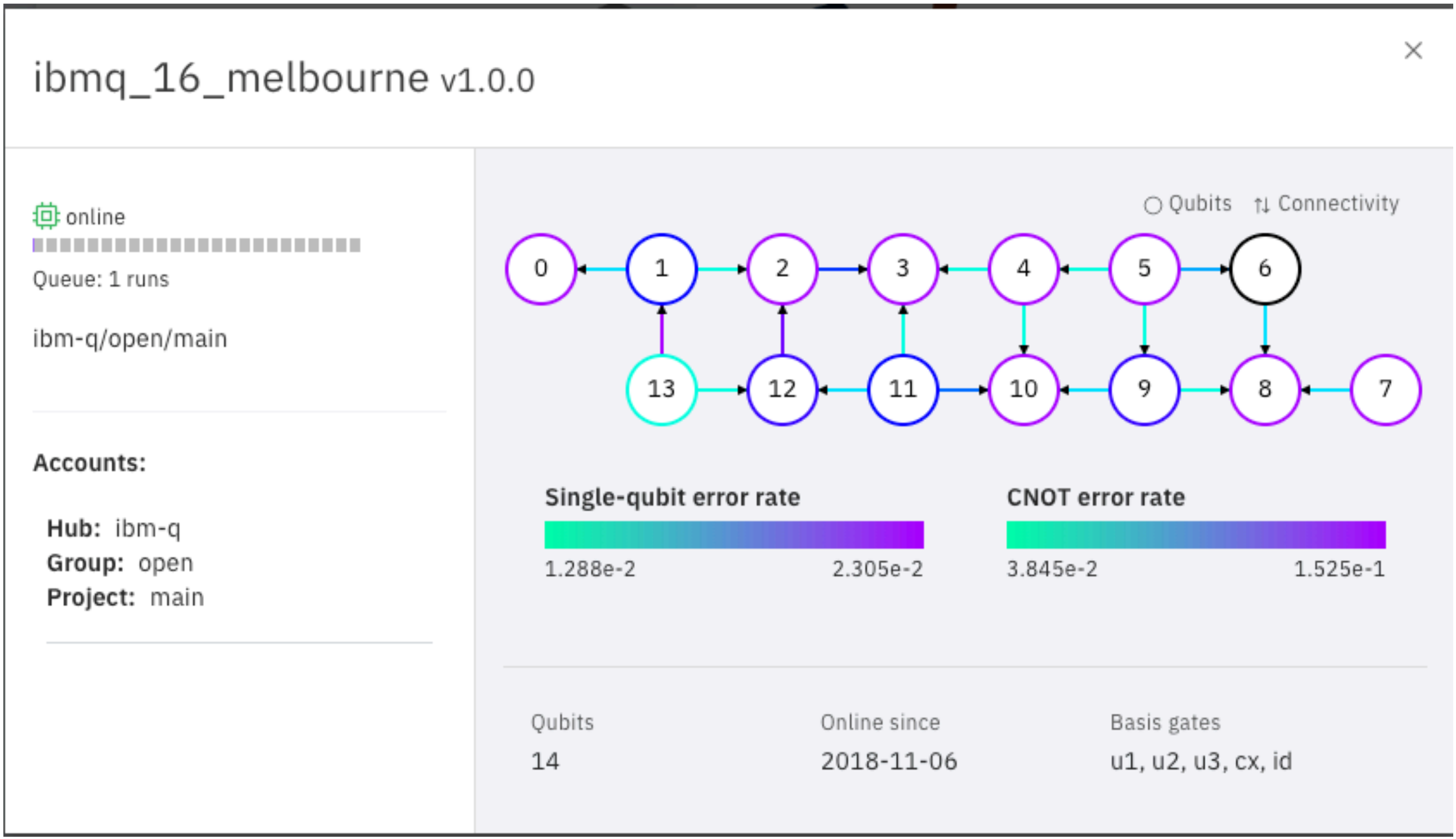
https://quantumexperience.ng.bluemix.net/qx/devices

Limited Qubits, Connectivity



https://quantumexperience.ng.bluemix.net/qx/devices

Limited Qubits, Connectivity



https://quantumexperience.ng.bluemix.net/qx/devices

Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

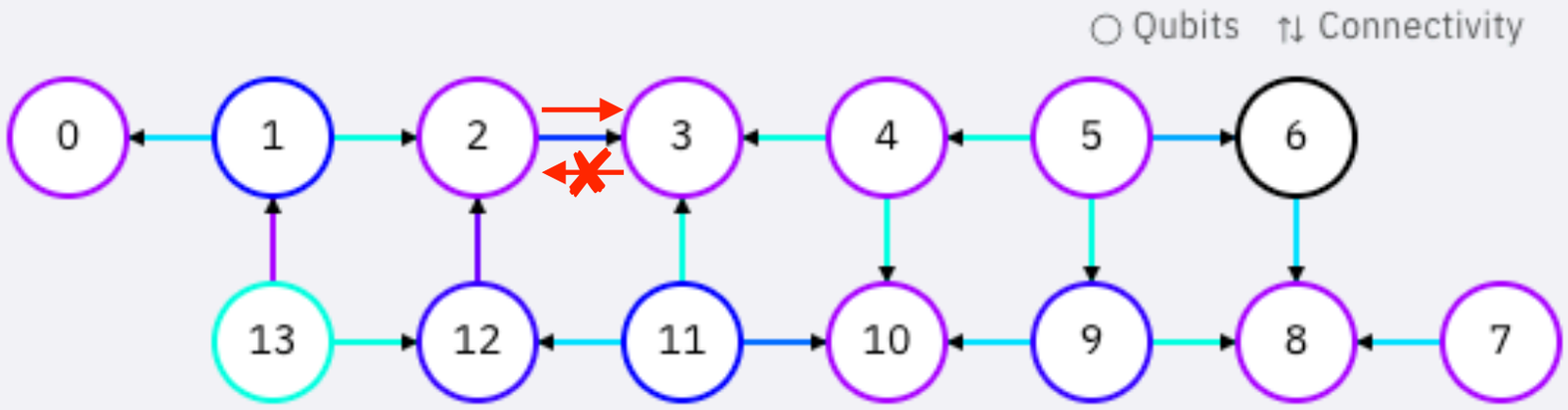
online

Queue: 1 runs

ibm-q/open/main

Accounts:

Hub: ibm-q
Group: open
Project: main



Single-qubit error rate

1.288e-2 2.305e-2

CNOT error rate

3.845e-2 1.525e-1

Qubits	Online since	Basis gates
14	2018-11-06	u1, u2, u3, cx, id

https://quantumexperience.ng.bluemix.net/qx/devices

Limited Qubits, Connectivity

ibmq_16_melbourne v1.0.0

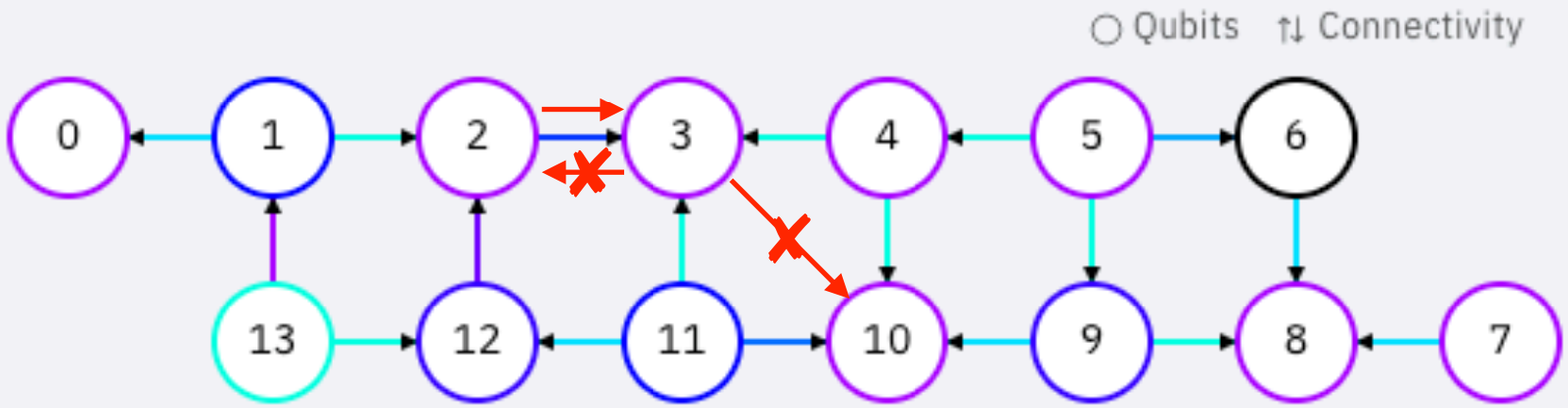
online

Queue: 1 runs

ibm-q/open/main

Accounts:

Hub: ibm-q
Group: open
Project: main



Single-qubit error rate

1.288e-2 2.305e-2

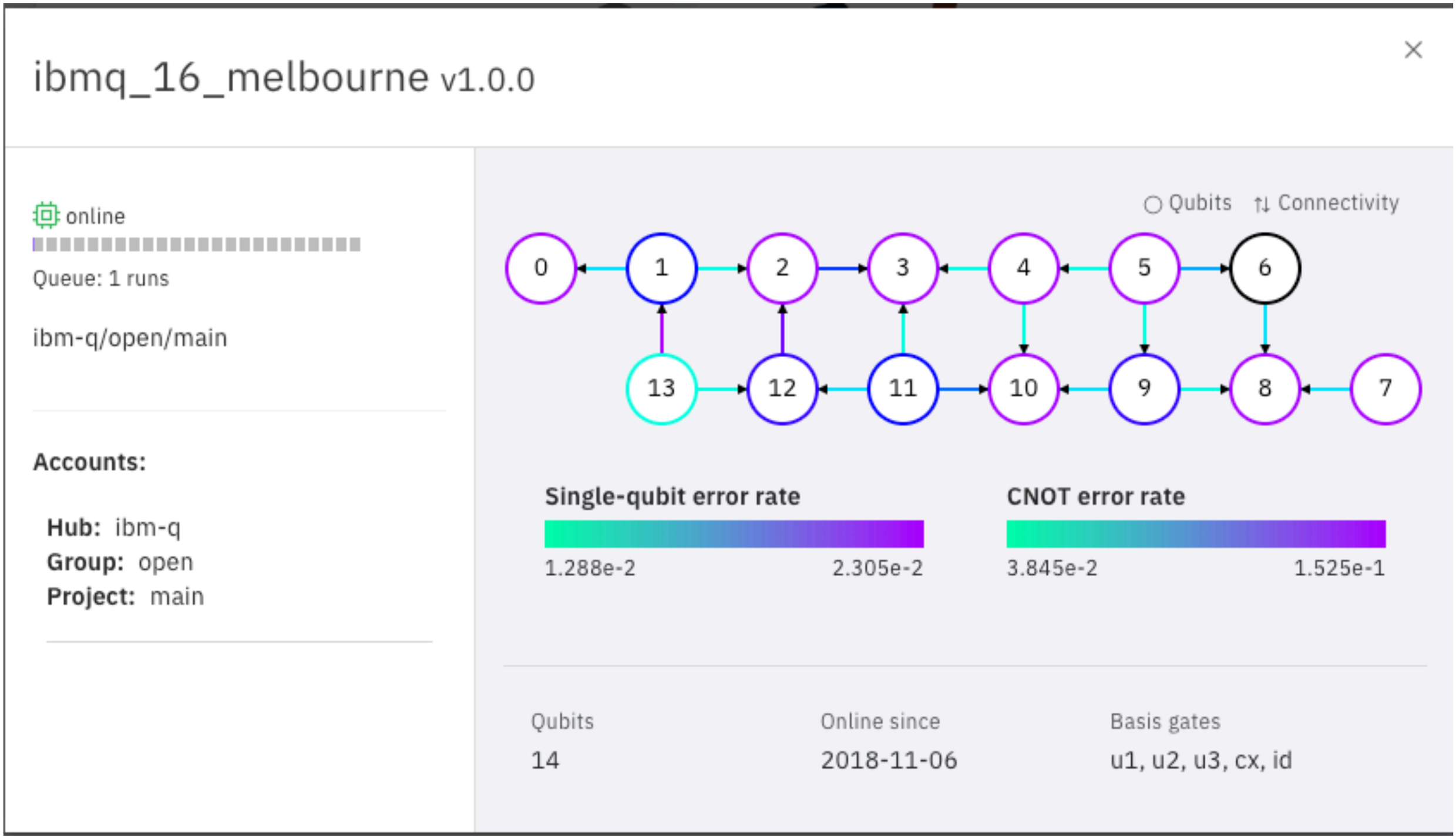
CNOT error rate

3.845e-2 1.525e-1

Qubits	Online since	Basis gates
14	2018-11-06	u1, u2, u3, cx, id

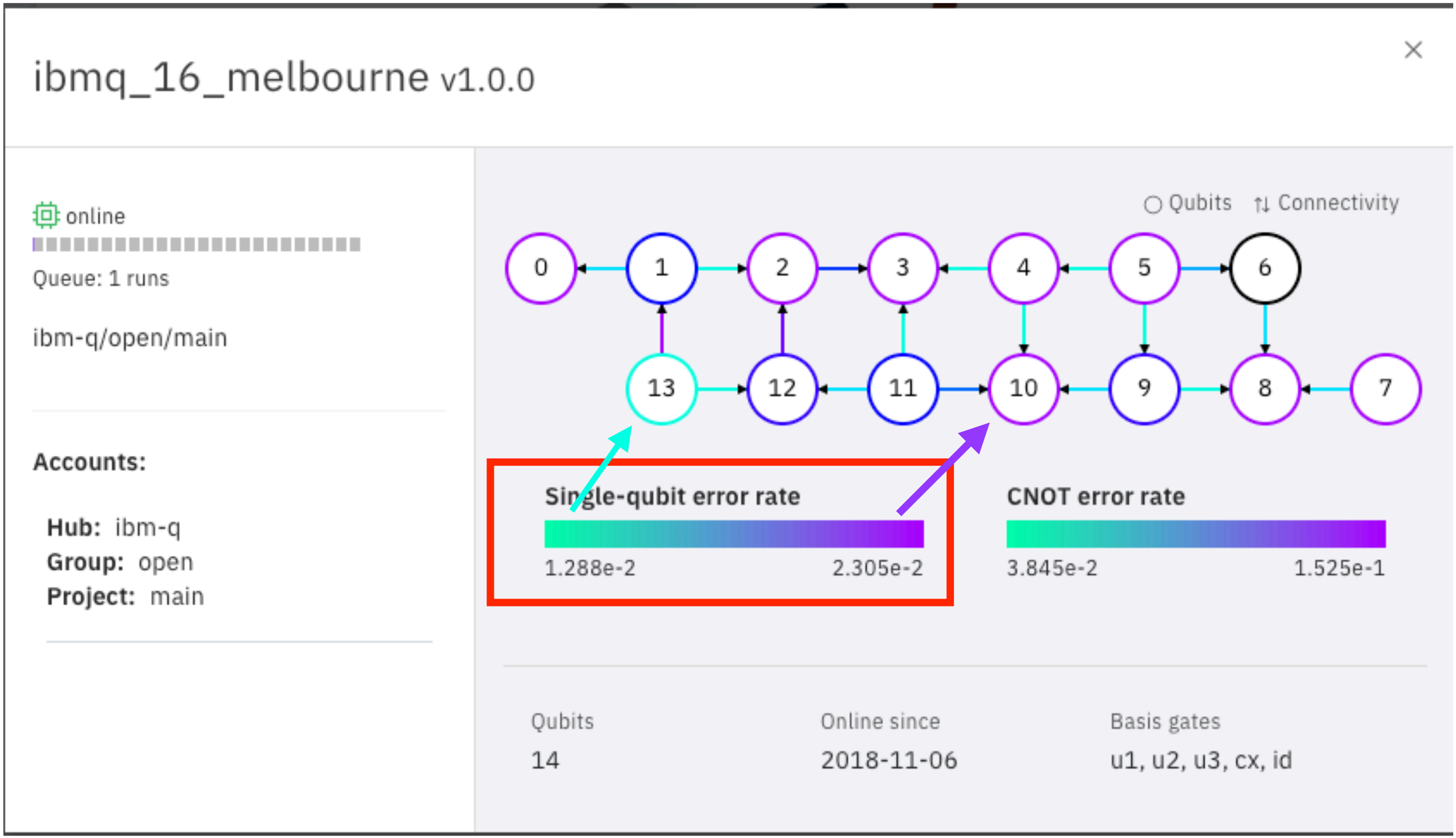
https://quantumexperience.ng.bluemix.net/qx/devices

Substantial Error Rates



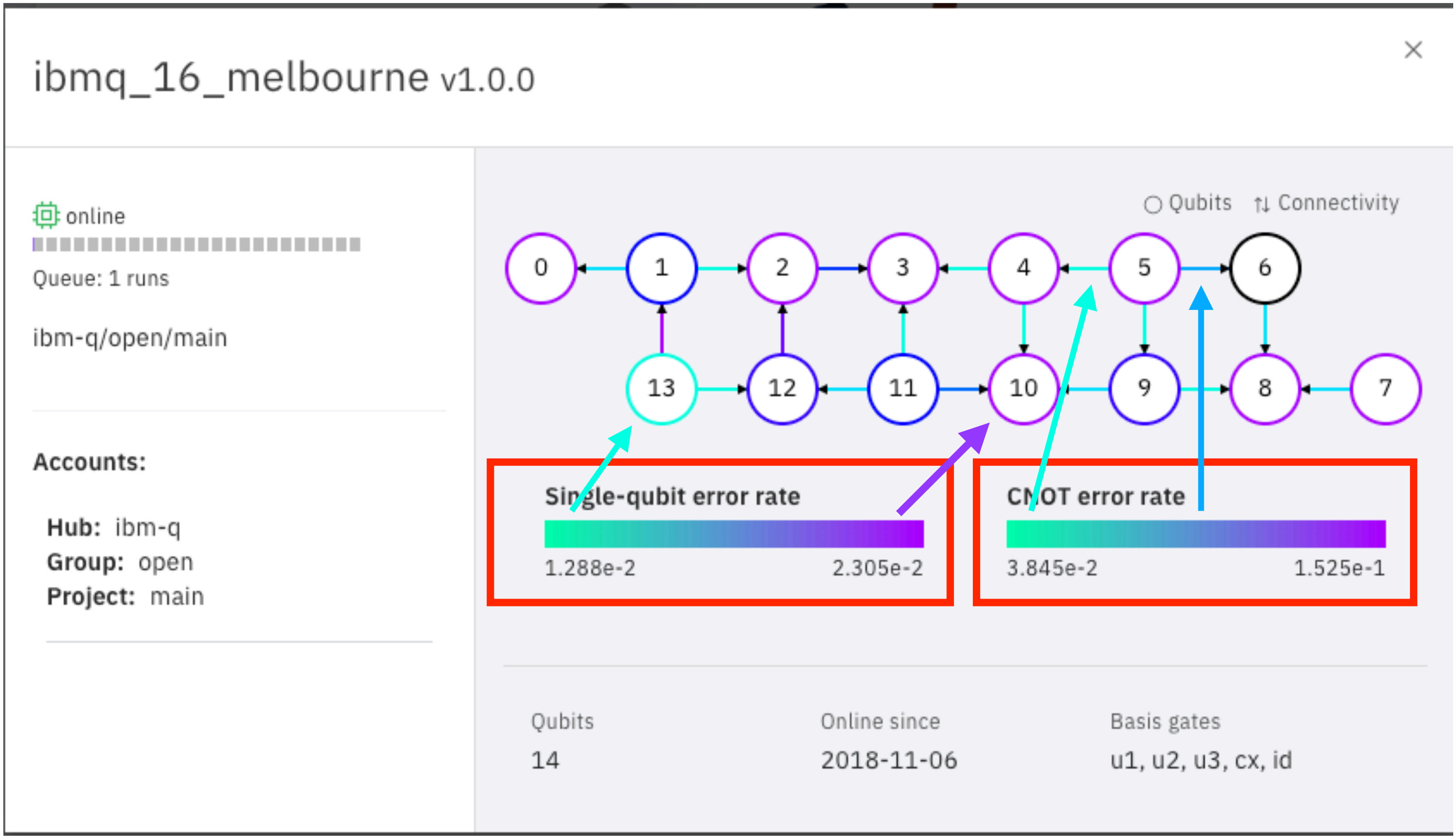
https://quantumexperience.ng.bluemix.net/qx/devices

Substantial Error Rates



https://quantumexperience.ng.bluemix.net/qx/devices

Substantial Error Rates



https://quantumexperience.ng.bluemix.net/qx/devices

Addressing the Challenges

In general

Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth

Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
- **Language design**: Make algorithms more intuitive

Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
- **Language design**: Make algorithms more intuitive

In the near term (handling scarce, error-prone resources)

- **Compiler optimizations** (proved correct)

Addressing the Challenges

In general

- **Formal verification**: State goal, prove you reach it
  - “Beware of bugs in the above code; I have only proved it correct, not tried it.” —Knuth
- **Language design**: Make algorithms more intuitive

In the near term (handling scarce, error-prone resources)

- **Compiler optimizations** (proved correct)
- **Reason about errors**, at the level of program and compiler

Quantum PL

Verification

Abstractions?

High Level Language

Compilation
Error Models
Verification
Optimization

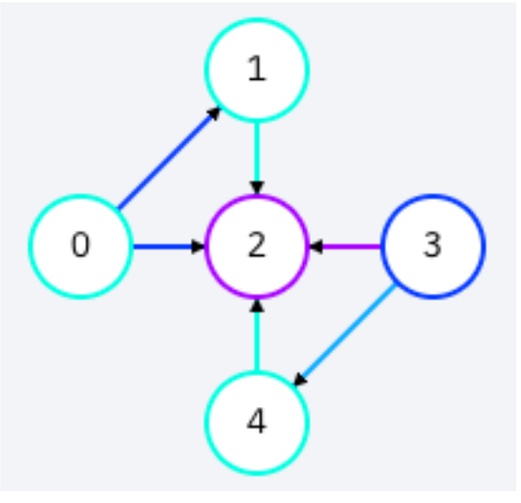
Intermediate (Circuit) Representation

Semantics
Mapping

Instruction Set

Hardware

Hardware Description, Error Model



Our Research

Verification

Abstractions?

QWIRE, Robustness logic

Compilation
Error Models
Verification
Optimization

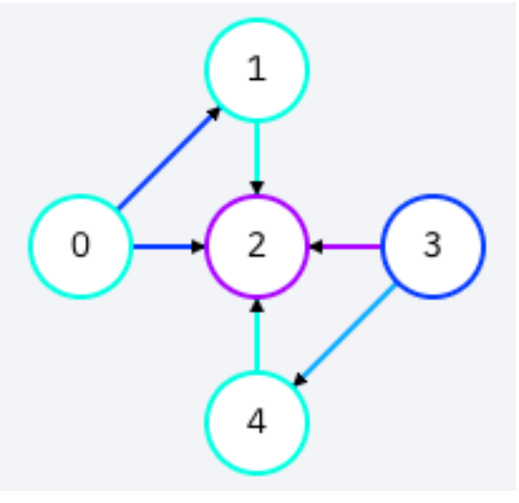
SQIRE

Semantics
Mapping

(OpenQASM)

(IBM QX, Rigetti, IonQ)

Hardware
Description,
Error Model



QWIRE

- QWIRE (POPL 2017) is a small language for describing quantum circuits. Features

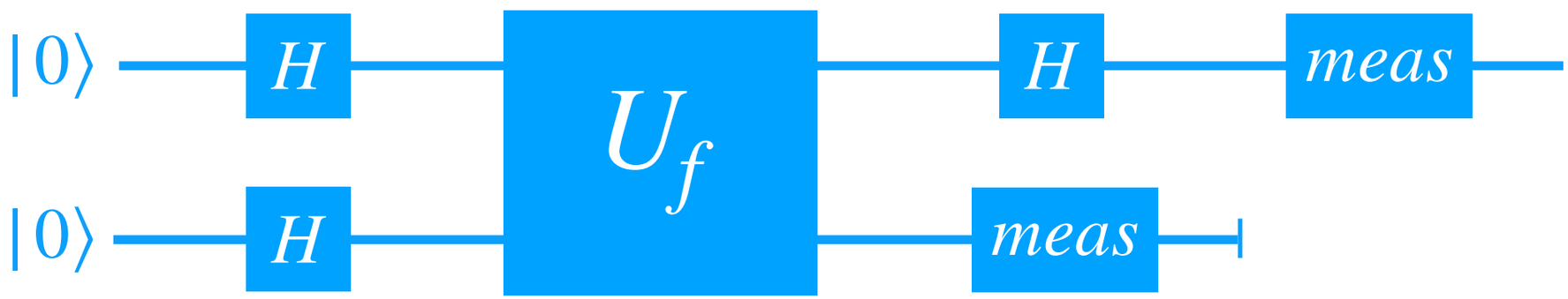
  - A linear type system for enforcing *no cloning*

  - A denotational semantics in terms of *density matrices*

  - Embedded in Coq, with tactics for *program verification*

https://github.com/inQWIRE/QWIRE

Deutsch's Algorithm



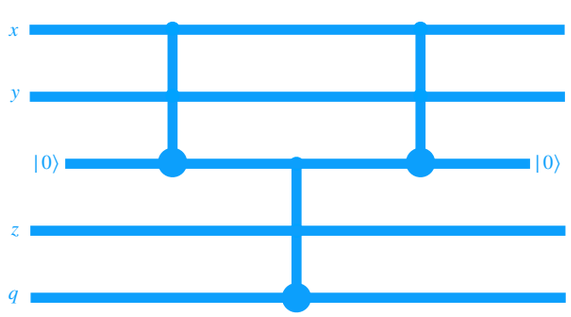

```
Definition deutsch (f : bool -> bool) :=
  let x     ← H $ init0 $ ();
  let y     ← H $ init1 $ ();
  let (x,y) ← (U f) $ (x,y);
  let ()    ← discard $ meas $ y;
  meas $ H $ x.
```

QWIRE in action

QWIRE in action

Compilation

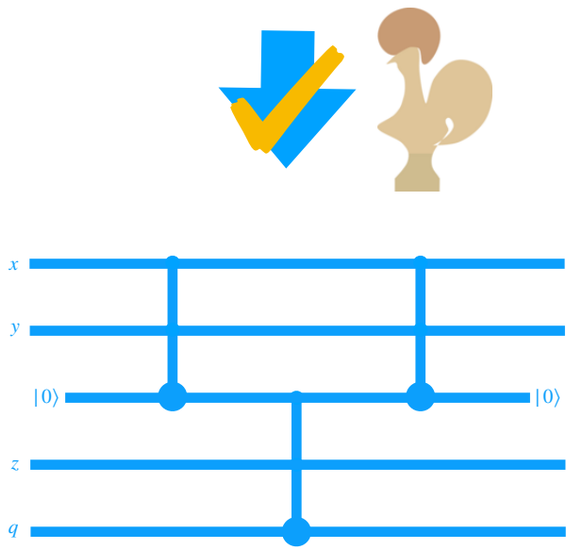
$(x \wedge y) \wedge z$



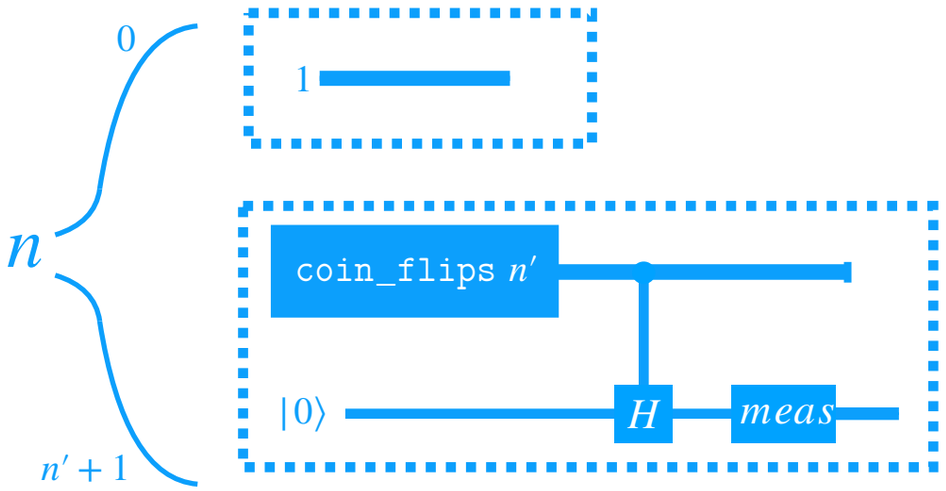
QWIRE in action

Compilation

$$(x \wedge y) \wedge z$$



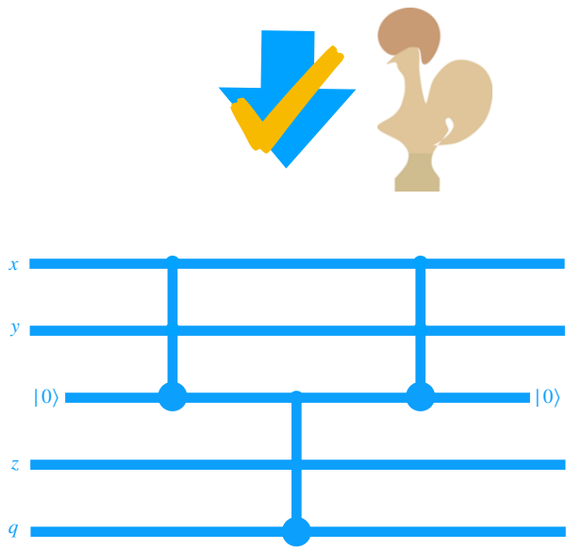
Random Number Generation



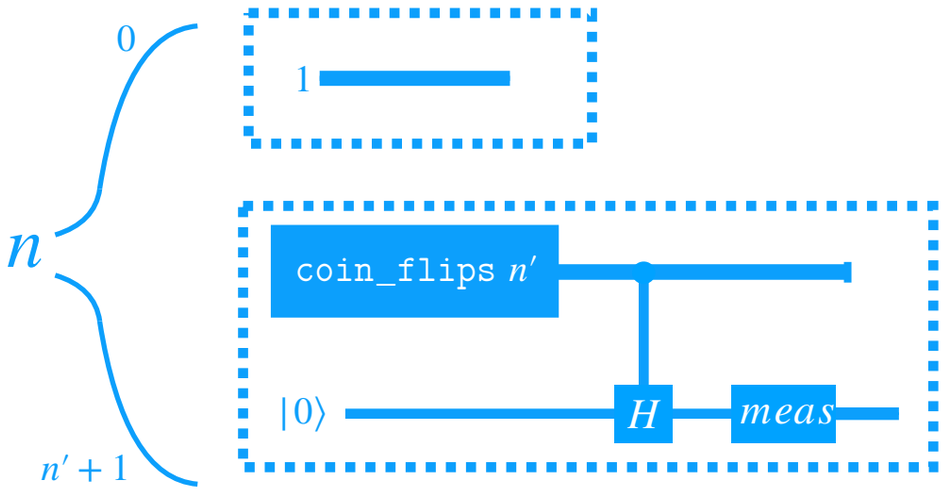
QWIRE in action

Compilation

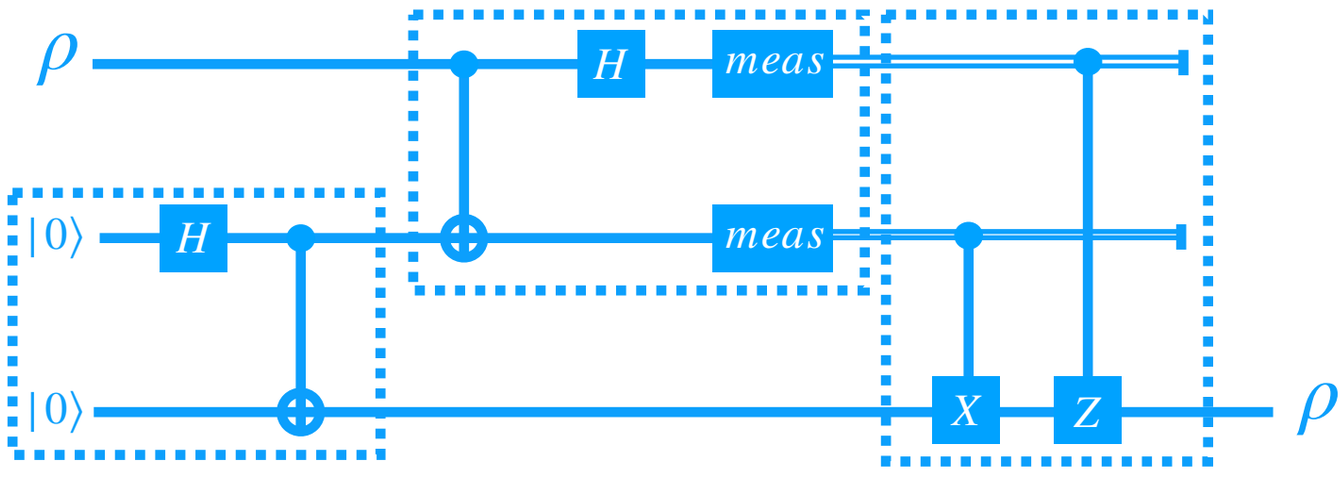
$$(x \wedge y) \wedge z$$



Random Number Generation



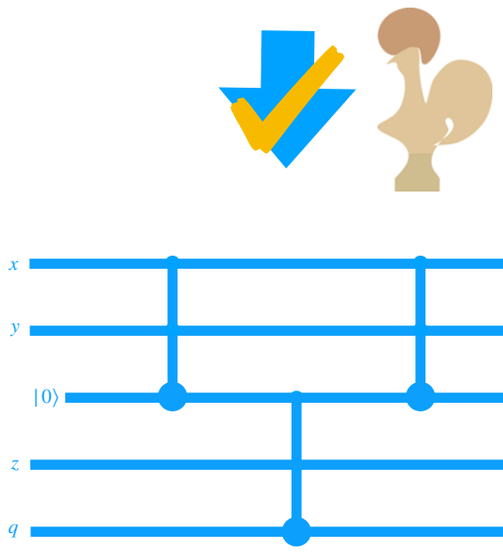
Teleportation



QWIRE in action

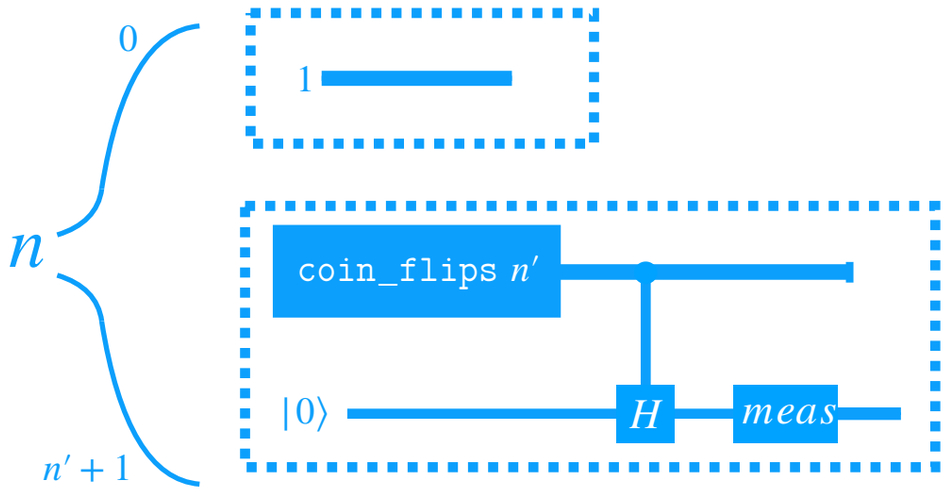
Compilation

$$(x \wedge y) \wedge z$$

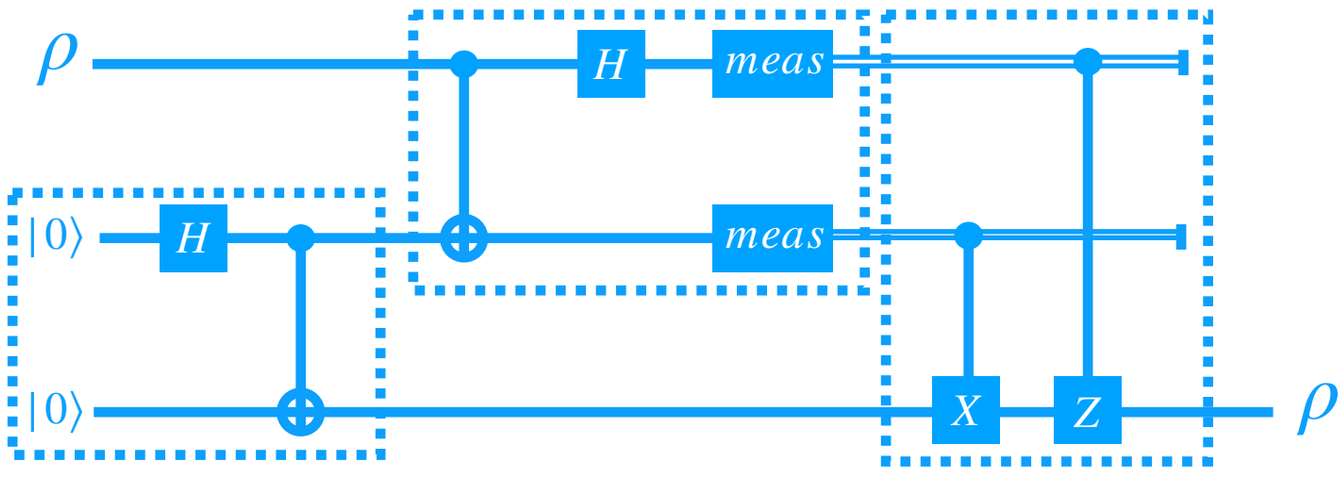


But: Nuts and bolts of proving things is surprisingly difficult!

Random Number Generation



Teleportation



Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates

$$(Q, \lambda) \vdash \widetilde{P} \leq \epsilon.$$

judgment bounds the "distance" ε of the actual state, due to errors, from the ideal

$$\frac{}{(Q,\lambda) \vdash \textbf{skip} \leq 0}\ (\text{Skip}) \qquad \frac{}{(Q,\lambda) \vdash (q := |0\rangle) \leq 0}\ (\text{Init}) \qquad \frac{\|U \circ U^\dagger - \Phi\|_{Q,\lambda} \leq \epsilon}{(Q,\lambda) \vdash (\overline{q} :\cong_{p,\Phi} U[\overline{q}]) \leq p\epsilon}\ (\text{Unitary})$$

$$\frac{(Q',\lambda') \vdash \widetilde{P} \leq \epsilon' \qquad \epsilon' \leq \epsilon \qquad Q \sqsubseteq Q' \qquad \lambda' \leq \lambda}{(Q,\lambda) \vdash \widetilde{P} \leq \epsilon}\ (\text{Weaken})$$

$$\frac{(Q/\delta, \lambda/\delta) \vdash \widetilde{P} \leq \epsilon \qquad 0 \sqsubseteq Q, Q/\delta \sqsubseteq I \qquad 0 \leq \lambda, \lambda/\delta \leq 1}{(Q,\lambda) \vdash \widetilde{P} \leq \epsilon}\ (\text{Rescale})$$

$$\frac{(Q_1,\lambda) \vdash \widetilde{P_1} \leq \epsilon_1 \qquad (Q_2,\lambda) \vdash \widetilde{P_2} \leq \epsilon_2 \qquad \{Q_1\}P_1\{Q_2\}}{(Q_1,\lambda) \vdash (\widetilde{P_1}; \widetilde{P_2}) \leq \epsilon_1 + \epsilon_2}\ (\text{Sequence})$$

$$\frac{\forall m, (Q_m, 1-\delta) \vdash \widetilde{P_m} \leq \epsilon \qquad t, \delta \in [0,1]}{(\sum_m M_m^\dagger Q_m M_m, 1 - t\delta) \vdash (\textbf{case } M[\overline{q}] = \overline{m \to \widetilde{P_m}} \textbf{ end}) \leq (1-t)\epsilon + t}\ (\text{Case})$$

$$\frac{(Q,\lambda) \vdash \widetilde{P_1} \leq \epsilon \qquad \{Q\}P_1\{\lambda M_0^\dagger M_0 + M_1^\dagger Q M_1\} \qquad \widetilde{P} \equiv \textbf{while } M[\bar{q}] = 1 \textbf{ do } \widetilde{P_1} \textbf{ done} \qquad P \text{ is } (a,n)\text{-bounded}}{(\lambda M_0^\dagger M_0 + M_1^\dagger Q M_1, \lambda) \vdash \widetilde{P} \leq n\epsilon/(1-a)}\ (\text{While-Bounded})$$

$$\frac{}{(Q,\lambda) \vdash (\textbf{while } M[\bar{q}] = 1 \textbf{ do } \widetilde{P_1} \textbf{ done}) \leq 1}\ (\text{While-Unbounded})$$

Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:

Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:
  - These bounds only *increase*: No way to verify error correction.

Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:
  - These bounds only *increase*: No way to verify error correction.
  - Not designed for general reasoning

Reasoning about Errors

- *Quantum Robustness Logic* (POPL 2019) allows us to bound errors in quantum programs given an error model on gates. Limitations:
  - These bounds only *increase*: No way to verify error correction.
  - Not designed for general reasoning
  - Not tied to hardware

Our Research

Hardware Description,

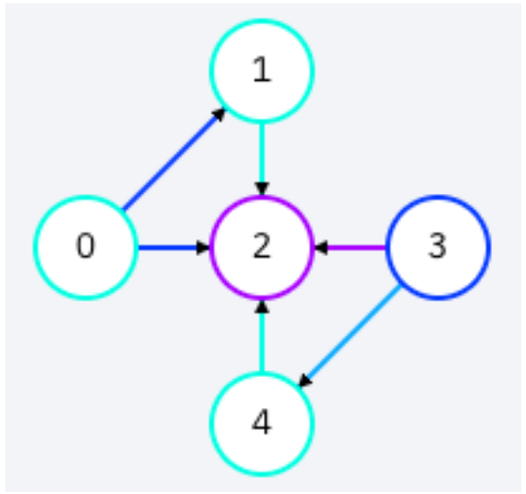
QWIRE, Robustness logic

SQIRE

OpenQASM

IBM QX, Rigetti, IonQ

Compilation
Verification
Optimization
Mapping



Our Research

Hardware Description,

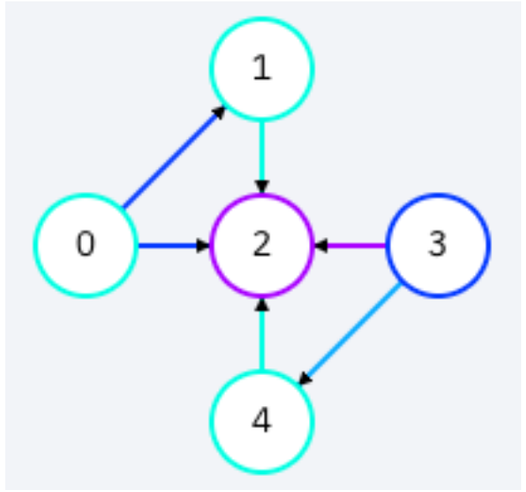
Compilation
Verification
Optimization
Mapping

QWIRE, Robustness logic

SQIRE

OpenQASM

IBM QX, Rigetti, IonQ



SQIRe

- Simple IR for certified quantum compiler
  - Past compilers have had bugs. In SQIRe, can prove optimizations are bug free
  - Likewise, prove that mapping to hardware with limits makes sense
- Can also prove algorithm-level properties. Use of global register, rather than HOAS, a key
  - But not sure if key in principle, or just our practice
- Goal: Error-aware optimizations

Horizon: Higher-level Languages

- Current “high level” languages not very high level.
  - Traditional control operators over quantum circuits
- Instead: we want higher-level building blocks
  - Expose key components of quantum algorithms
  - Q# is pushing in this direction, but still much to do

Future Work: You!

- There is lots of room for work in quantum computing that takes a PL perspective
    - Formal methods, compilers & optimization, language design, static analysis, hardware/ software co-design …
- Join us!