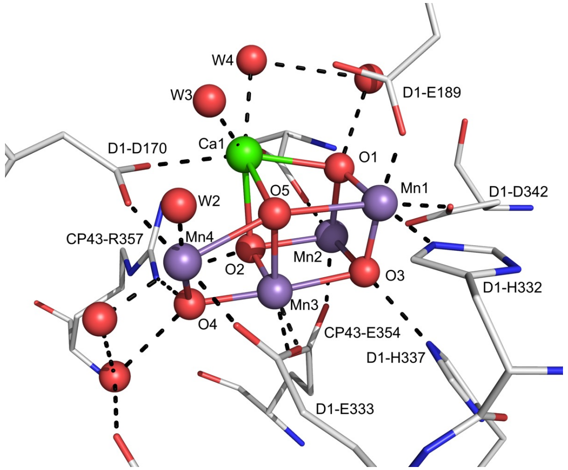
Dissipation Nation

Sabee Grewal, Matthew Hagan, Daniel Marti-Dafcik, Lucia Vilchez-Estevez

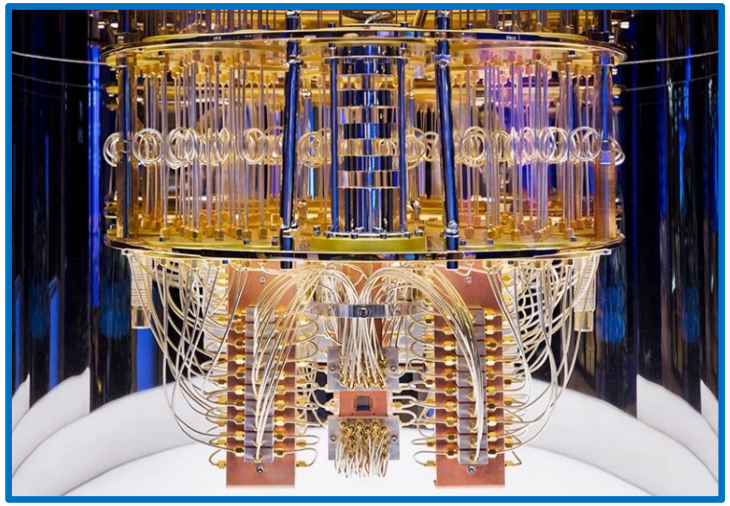
What is the Dissipative Quantum Eigensolver?

Algorithm to prepare a ground state due to T. Cubitt (arxiv:2303.11962)



Nature. 473 (7345): 55–60

H



Nature 599, 542 (2021)

ρ

What is the Dissipative Quantum Eigensolver?

```
ρ = initial state
until (stop condition):
    ρ, b = E(ρ)
    if b:
        Do some qubit resets
return ρ
```

ρ will be close to the true ground state!

What is the Dissipative Quantum Eigensolver?

```
 ρ = initial state
until (stop condition):
      ρ, b = E(ρ)
     if b:
          Do some qubit resets
return ρ
```

Approximate GS projection

ρ will be close to the true ground state!

Why DQE?

Scientifically important

- Ground state preparation central in quantum chemistry and materials!
- Enables computing important properties
- Example: energy → reaction rates

Advantages of DQE

- Provable guarantees
- Noise resilient
- Conceptually simple

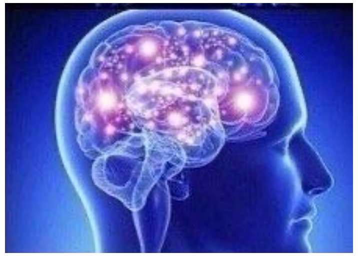
Why DQE? A good fit for Triple Alpha!

```
 ρ = initial state
until (stop condition):
      ρ, b = E(ρ)
      if b:
            Do some qubit resets
return ρ
```

Why DQE? A good fit for Triple Alpha!

```
ρ = initial state
until (stop condition):
      ρ, b = E(ρ)
      if b:
            Do some qubit resets
return ρ
```

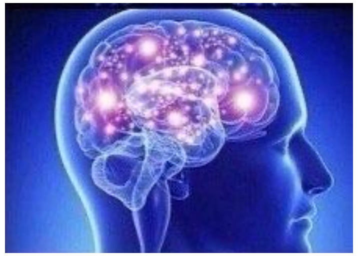
Looping uses the Turing-completeness of Helium



Why DQE? A good fit for Triple Alpha!

```
 ρ = initial state
until (stop condition):
      ρ, b = E(ρ)
      if b:
            Do some qubit resets
return ρ
```

Looping uses the Turing-completeness of Helium

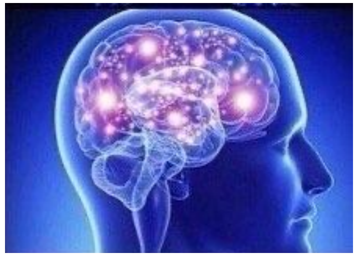


Intermediate measurements! Complicated control of the quantum device made easy.

Why DQE? A good fit for Triple Alpha!

```
 ρ = initial state
until (stop condition):
      ρ, b = E(ρ,ϵ)
      if b:
            Do some qubit resets
return ρ
```

Looping uses the Turing-completeness of Helium



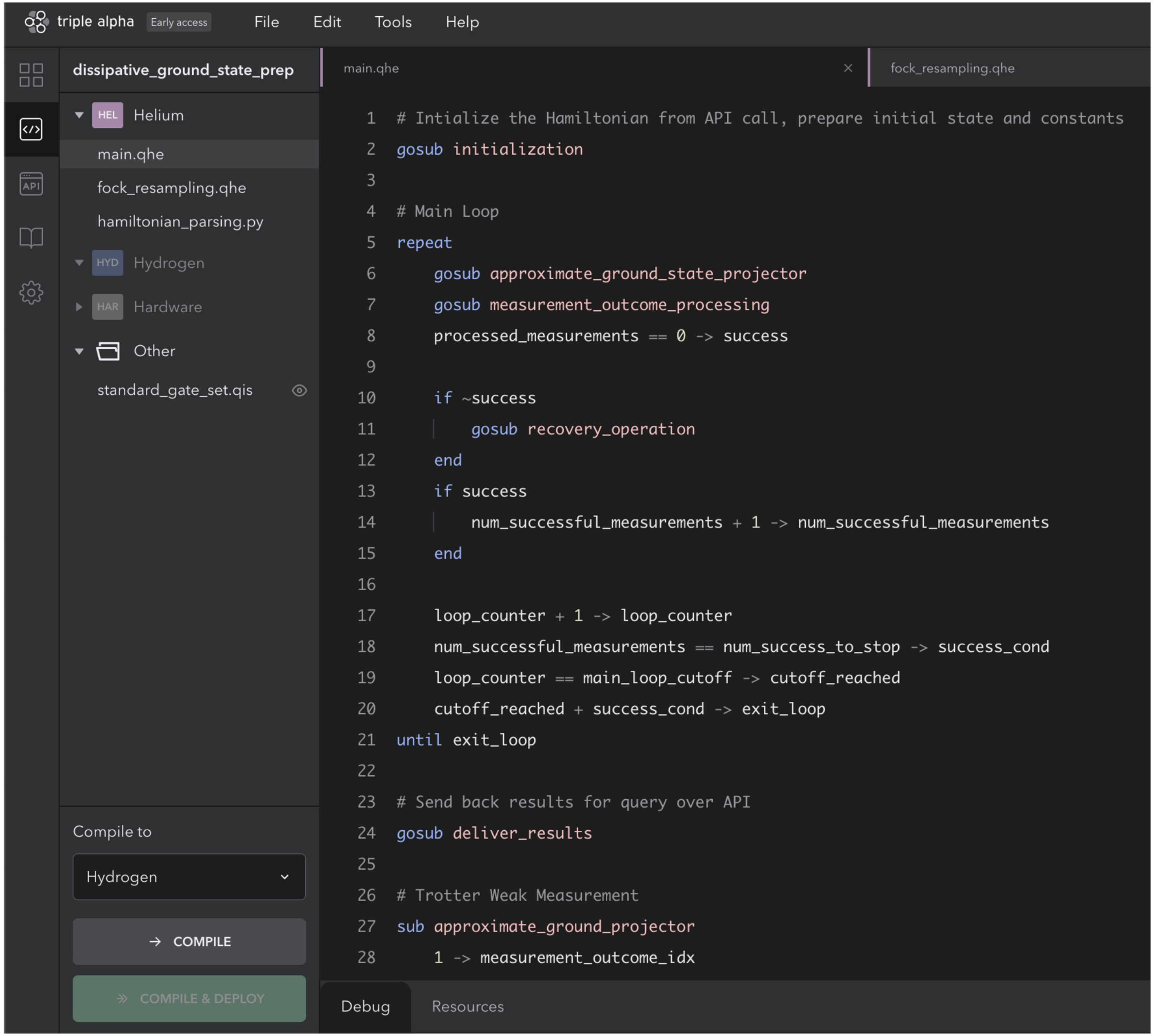
Intermediate measurements! Complicated control of the quantum device made easy.

Interesting parameters to play with! Implementation has practical importance!

What did we do?

~ 200 lines of Helium code!
And Python client code!

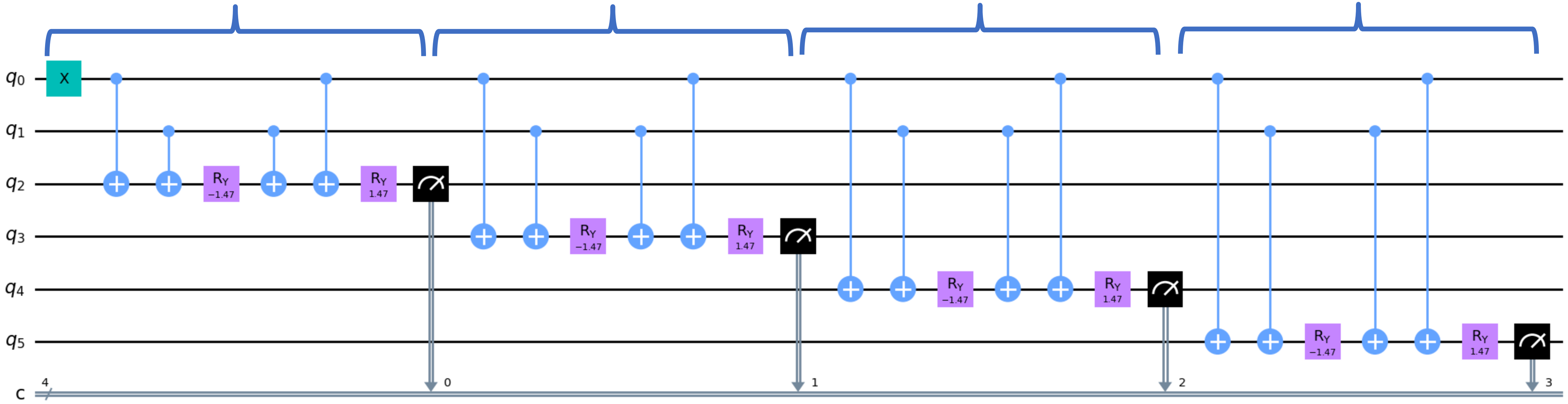
Deployed as an API!



What did we do?

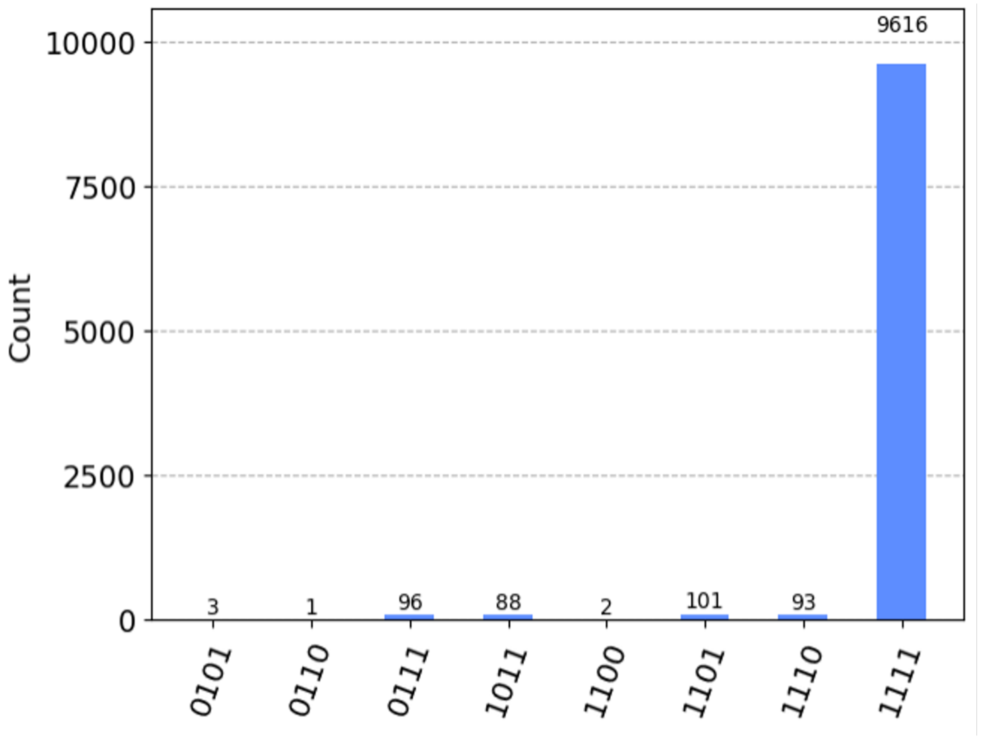
Built down to IBM circuit :)

H → U = approximate GS projector



What did we do?

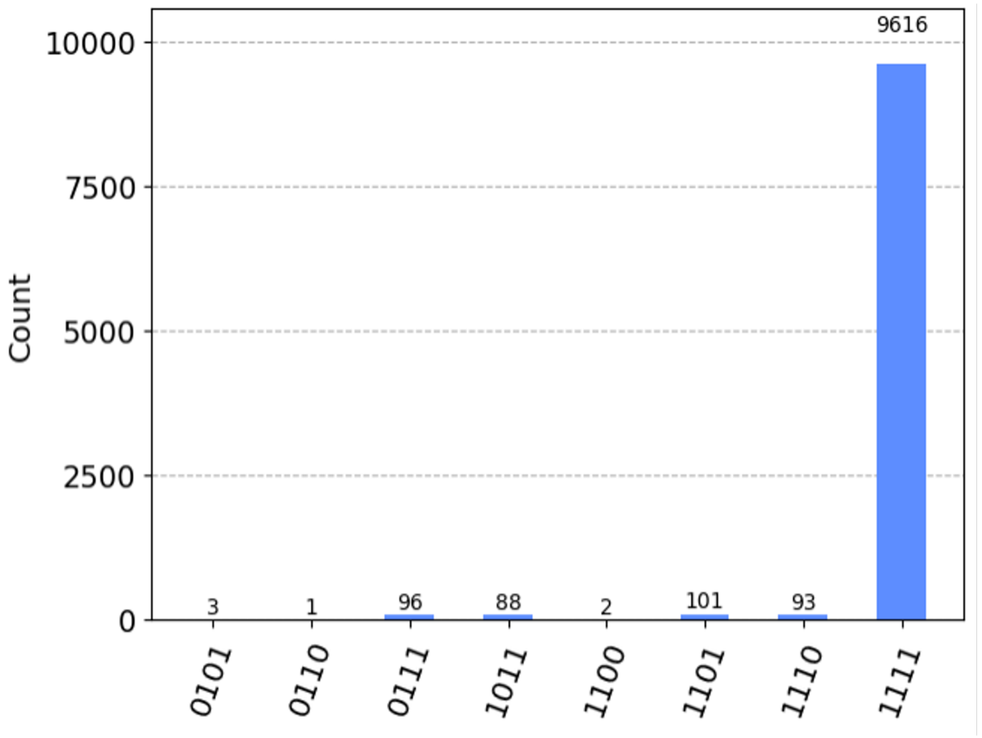
$\varepsilon = 0.9$



$|GS>$

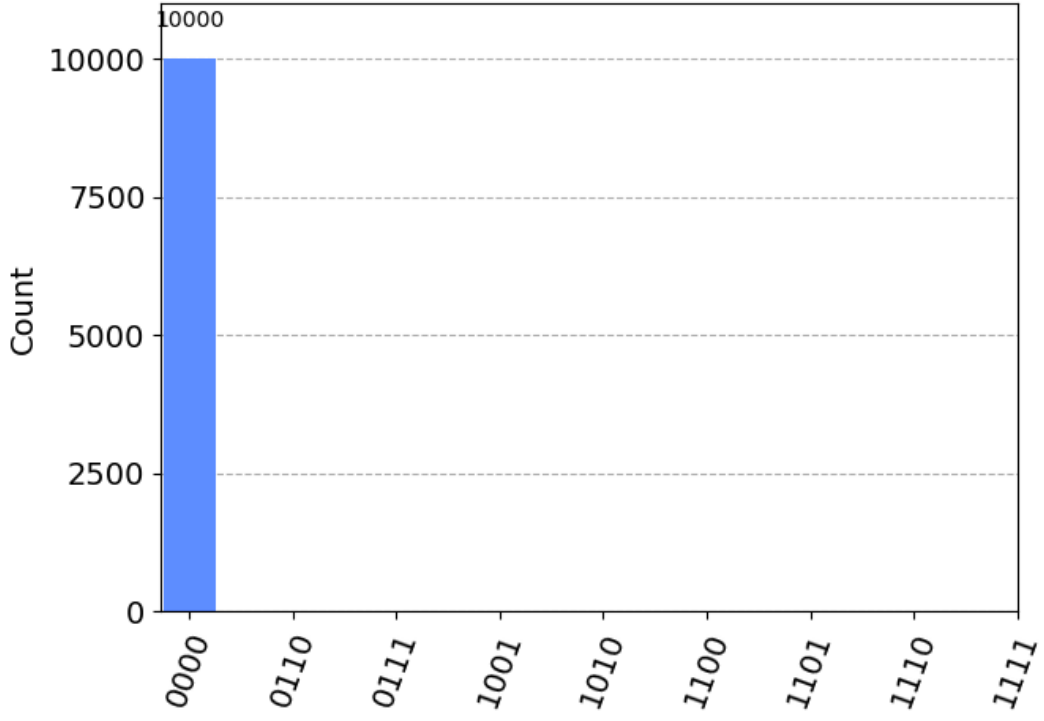
What did we do?

$\varepsilon = 0.9$



$|GS>$

$\varepsilon = 0.9$

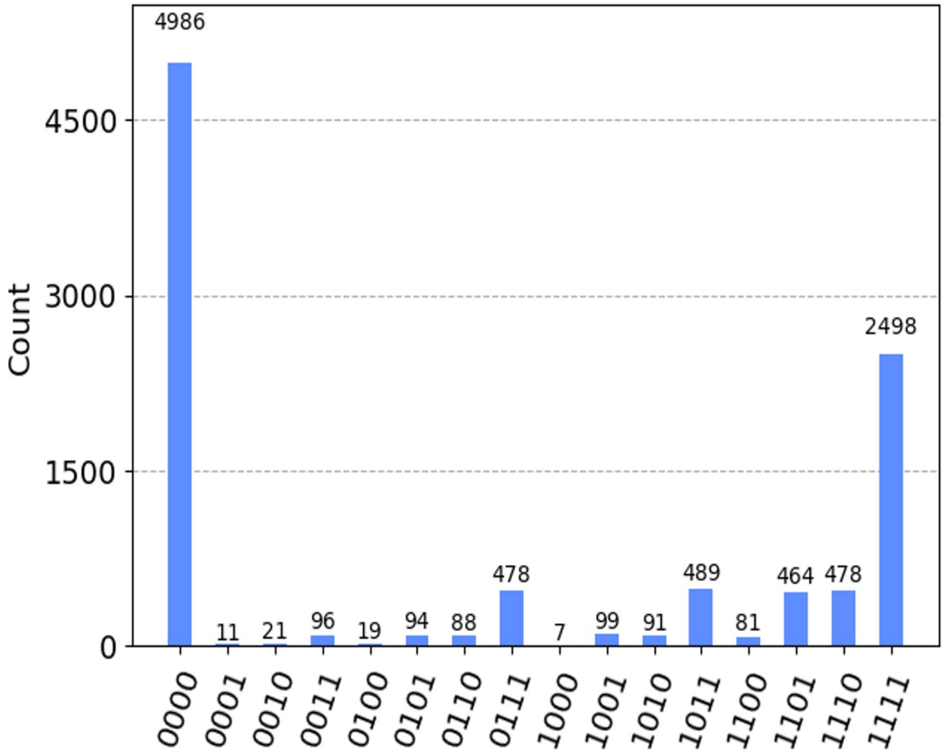


$|GS^{\perp}>$

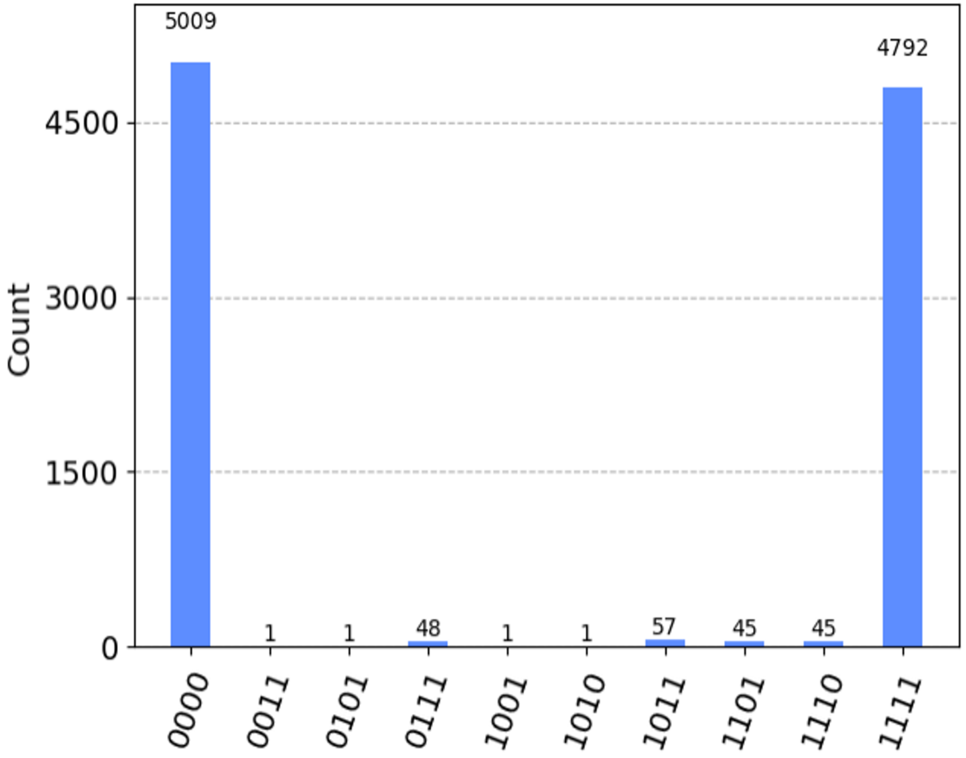
What did we do?

$$\frac{1}{\sqrt{2}}(|GS > +|GS^{\perp} >)$$

$\boldsymbol{\varepsilon = 0.6}$



$\boldsymbol{\varepsilon = 0.9}$



Decreasing ε makes the projector more gentle

Improvements to Triple Alpha

- Nondemolition measurements
- Client libraries that abstract away curl request.
- More expressive language features:

  `if expression` rather than `if int`
  `increment i` **`from A`** `to B`

- Local IDE and local compiler

Future Work (at Bali next year?!)

- Simulate DQE on H_n chain
- Different instantiations of DQE:
  - Dependence on initial state
  - Resampling
  - Parameter scheduling

Future Work (at Bali next year?!)

- Simulate DQE on H_n chain
- Different instantiations of DQE:
  - Dependence on initial state
  - Resampling
  - Parameter scheduling

Thank you, Horizon,
for the amazing experience!