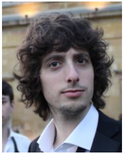
Quantum Software

Aleks Kissinger, Stefano Gogioso



UNIVERSITY OF OXFORD

Oxford, Hillary Term 2024

Quantum software

Quantum software

1. $:=$ the code the runs on a quantum computer

Quantum software

1. := the code the runs on a quantum computer

factoring

search

physical simulation
optimisation problems
linear systems & codes
network flows
natural language processing
...

Quantum software

1. := the code the runs on a quantum computer

factoring

search

physical simulation
optimisation problems
linear systems & codes
network flows
natural language processing
...

2. := the code that makes that code

Quantum software

1. := the code the runs on a quantum computer

factoring

search

physical simulation
optimisation problems
linear systems & codes
network flows
natural language processing
...

2. := the code that makes that code
   - **compilers**

Quantum software

1. := the code the runs on a quantum computer

factoring

search

physical simulation
optimisation problems
linear systems & codes
network flows
natural language processing
...

2. := the code that makes that code
   - **compilers**
   - **optimisation**
   - **verification**

Quantum software

1. := the code the runs on a quantum computer

factoring

search

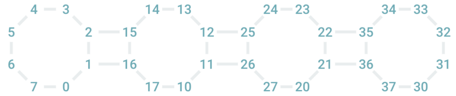
physical simulation
optimisation problems
linear systems & codes
network flows
natural language processing
...

2. := the code that makes that code (better)
   - **compilers**
   - **optimisation**
   - **verification**

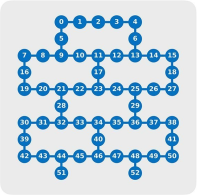
Problem: no quantum computers

Problem: ~~no quantum computers~~

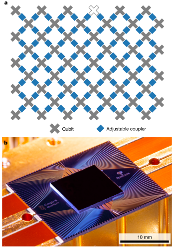
Rigetti Aspen



IBM Q Rochester

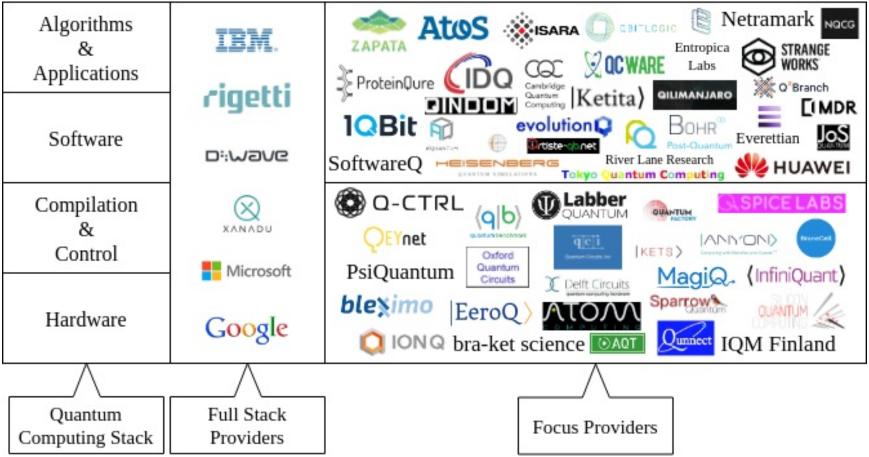


Google Sycamore



(+ Oxford, Vienna, Delft, Sussex, Maryland...)

~~Problem: no quantum computers~~



[1]

1 https://quantumcomputingreport.com/review-of-the-cirq-quantum-software-framework

Problem: limited quantum computers

NISQ := "noisy intermediate-scale quantum"

Problem: limited quantum computers

NISQ := "noisy intermediate-scale quantum"

NISQ devices have:

- short coherence times
- low numbers of qubits
- noisy operations
- limited connectivity
- ...

Problem: limited quantum computers

NISQ := "noisy intermediate-scale quantum"

NISQ devices have:

- short coherence times
- low numbers of qubits
- noisy operations
- limited connectivity
- ...

$\Rightarrow$ **small** advances in software give **big** gains on NISQ hardware!

Quantum circuits

Quantum circuits

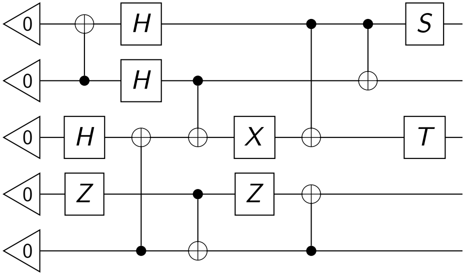
- $:=$ the 'assembly language' of quantum computation

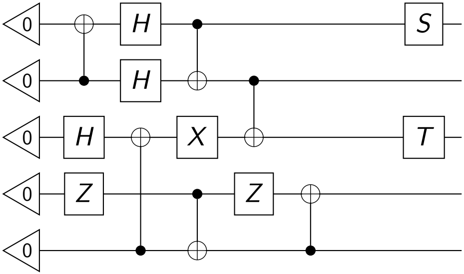
Quantum circuits

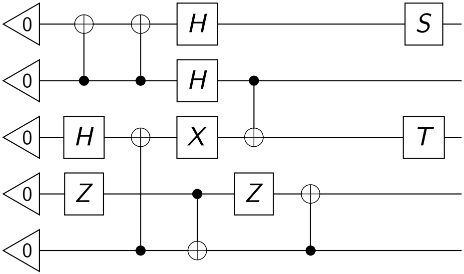
- $:=$ the 'assembly language' of quantum computation, e.g.

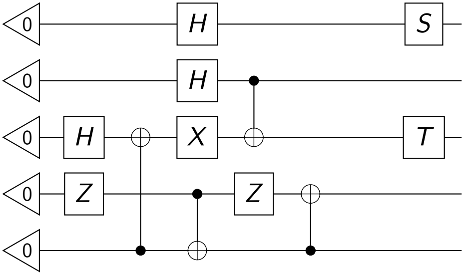
```
INIT 5
CNOT 1 0
H 2
Z 3
H 0
H 1
CNOT 4 2
...
```

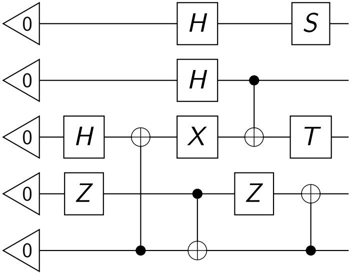
$\Leftrightarrow$

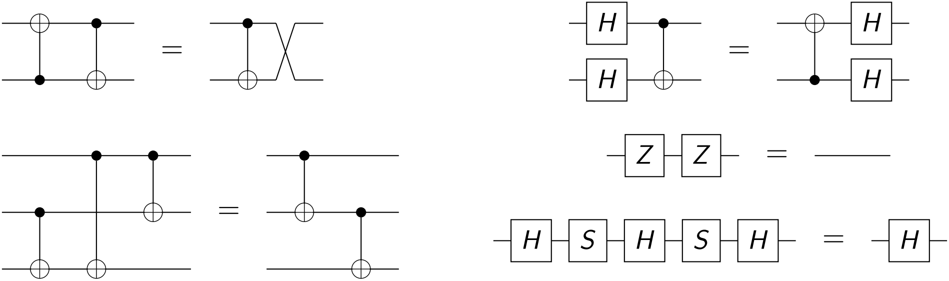



0
0
0
0
0
H
H
H
Z
X
Z
S
T

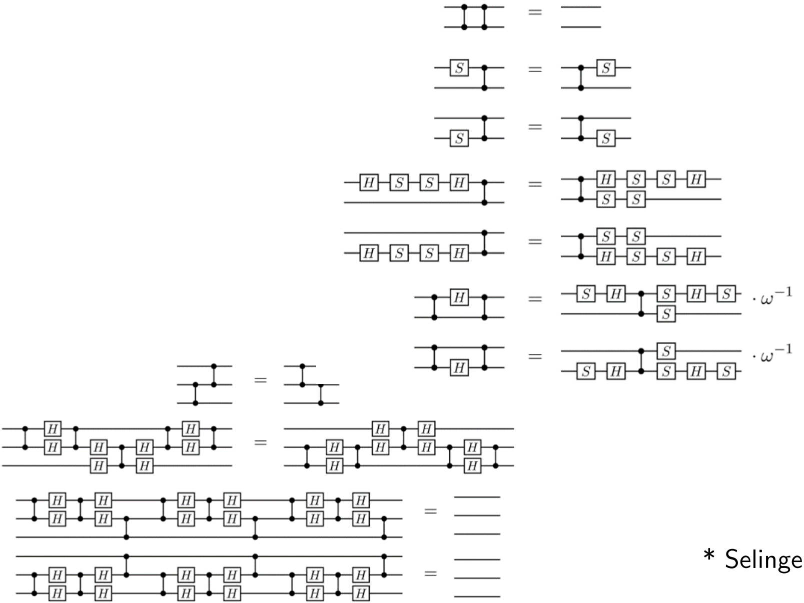

0
0
0
0
0
H
S
H
H
X
T
Z
Z


0
0
0
0
0
H
S
H
H
X
T
Z
Z


0
0
0
0
0
H
S
H
H
X
T
Z
Z

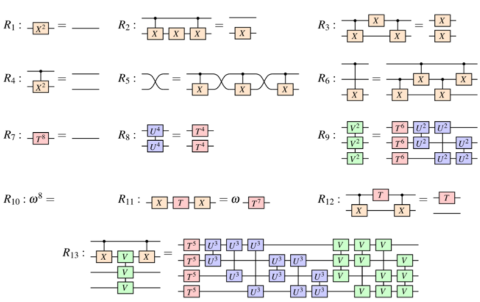

H
S
Z
=

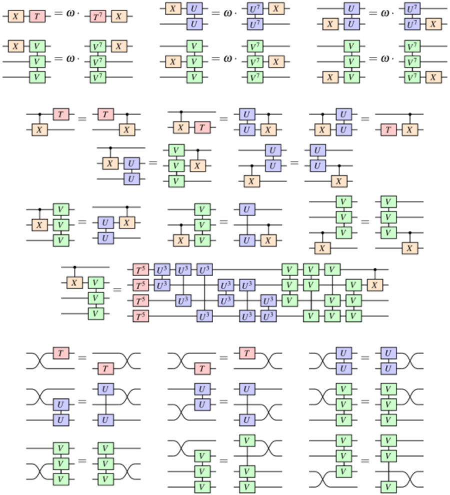
. . .



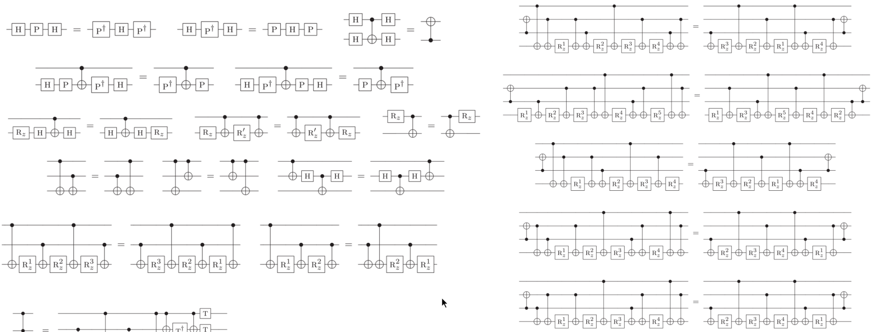
* Selinger 2015

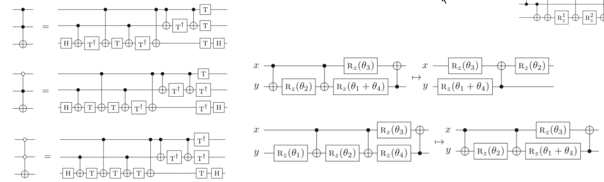
* Selinger 2015



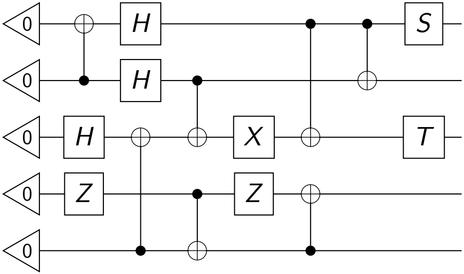


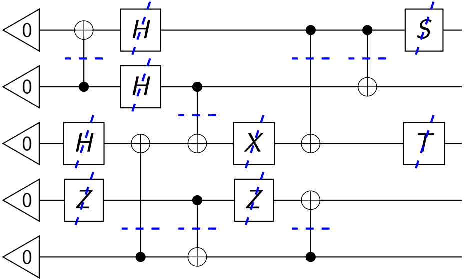
* Amy, Chen, & Ross 2018



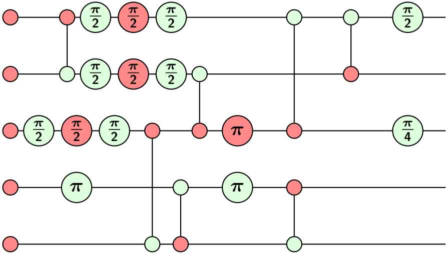


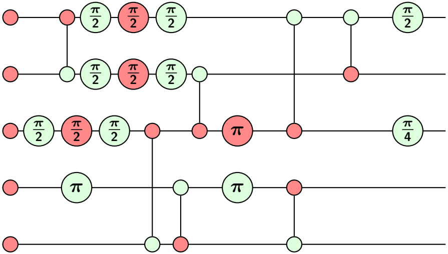
* Nam *et al* 2018


0
0
0
0
0
H
H
S
H
X
T
Z
Z



0
0
0
0
0
H
H
H
Z
X
Z
S
T


π/2 π/2 π/2 π/2
π/2 π/2 π/2
π/2 π/2 π/2 π π/4
π π

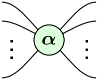


ZX-diagram

ZX-diagrams

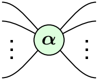
ZX-diagrams

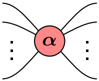
...are like circuits, but made from **spiders** instead of gates:

$\mathcal{Z}_\alpha \;=$ 

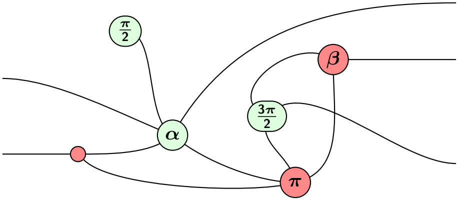
ZX-diagrams

...are like circuits, but made from **spiders** instead of gates:

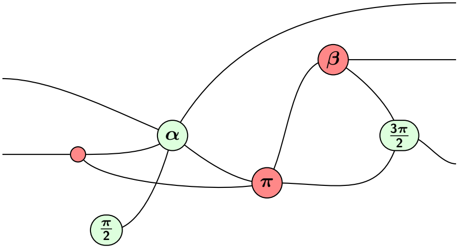
$\mathcal{Z}_\alpha$ = 

$\mathcal{X}_\alpha$ = 

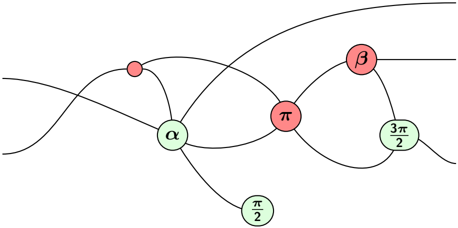
ZX-diagrams are bendy



ZX-diagrams are bendy



ZX-diagrams are bendy

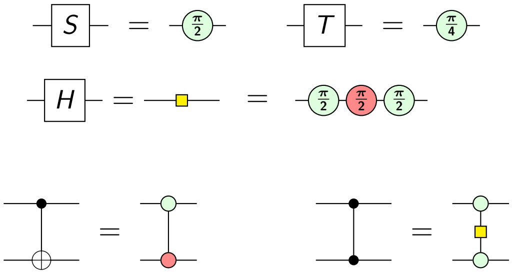


Why spiders?

- They generate **all** linear maps $\mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$.

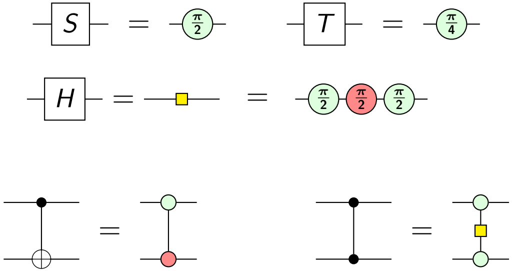
Why spiders?

- They generate **all** linear maps $\mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$.
- Handy for building most common gates, e.g.

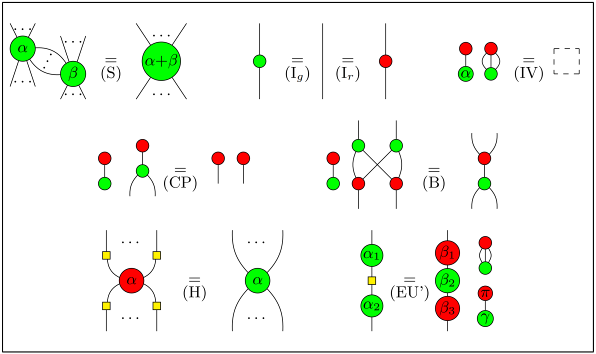


Why spiders?

- They generate **all** linear maps $\mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$.
- Handy for building most common gates, e.g.



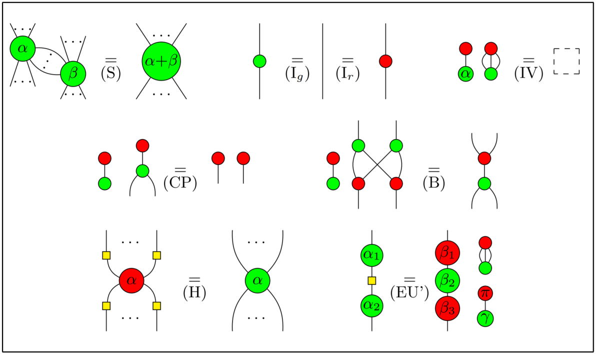
- ....and....



[2]

ZX calculus

[2] Vilmart 2018. arXiv:1812.09114

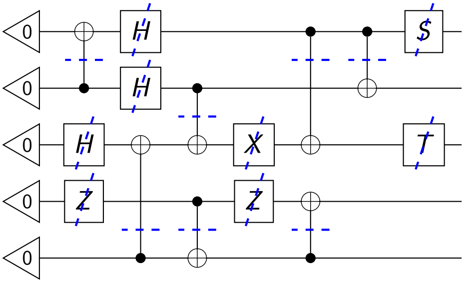


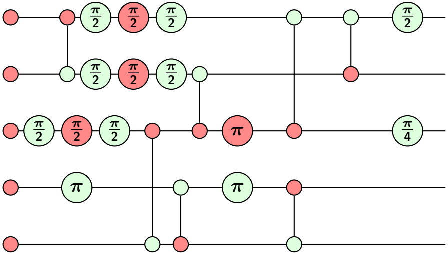
[2]

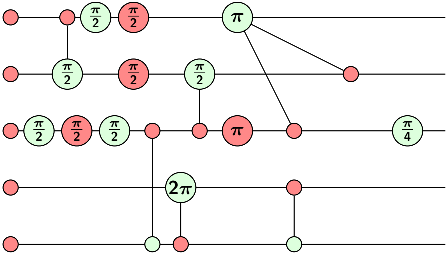
ZX calculus

these 8 rules $\implies$ everything before

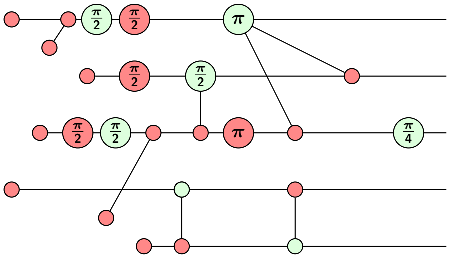
[2] Vilmart 2018. arXiv:1812.09114


0
0
0
0
0
H
H
H
Z
X
Z
S
T

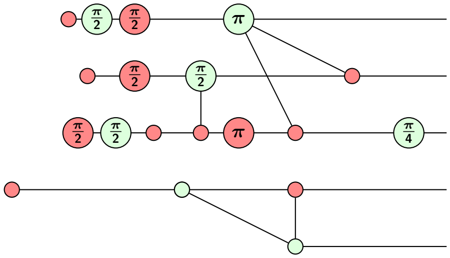

π/2
π/2
π/2
π/2
π/2
π/2
π/2
π/2
π/2
π/2
π
π/4
π
π

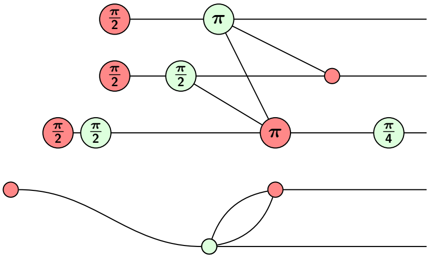


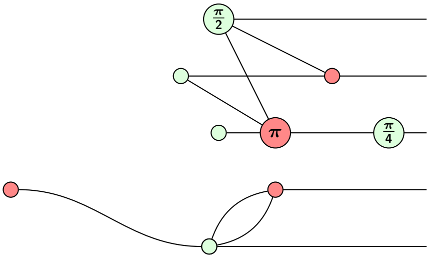
π/2
π/2
π
π/2
π/2
π/2
π/2
π/2
π/2
π
π/4
2π

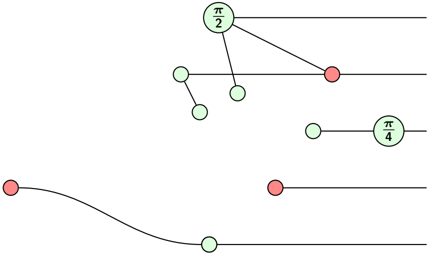


π/2
π/2
π
π/2
π/2
π/2
π/2
π
π/4

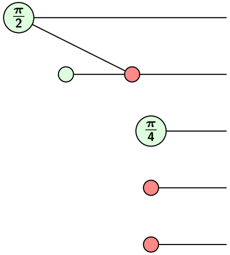

π/2
π/2
π
π/2
π/2
π/2
π/2
π
π/4

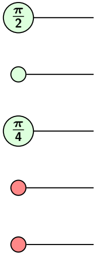

π/2
π
π/2
π/2
π/2
π/2
π
π/4

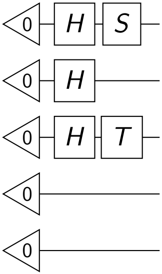

π
2
π
π
4



π/2
π/4

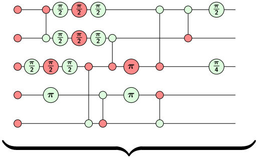

π
2
π
4


π
2
π
4


0
H
S
0
H
0
H
T
0
0

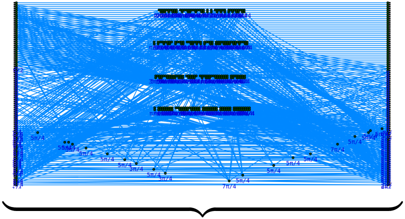
Q: How do we scale up?

Q: How do we scale up?



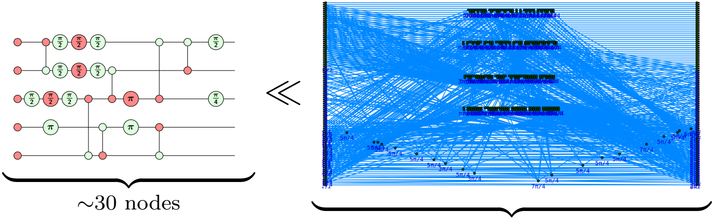
$\sim$30 nodes

$\ll$



10^2-10^5 nodes

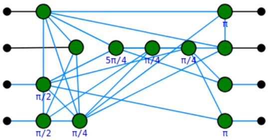
Q: How do we scale up?



~30 nodes ≪ 10^2-10^5 nodes

A: Automation.

In [11]:
```
zx.simplify.clifford_simp(g)
g.normalise()
zx.d3.draw(g)
```



In this course...

In this course...

- We'll look at **fundamental structures** underlying quantum computations:

$$\textbf{ZX-diagrams} \qquad \longleftrightarrow \qquad \begin{cases} \text{graph states} \\ \text{stabiliser groups} \\ \text{exponentiated Paulis} \\ \text{phase polynomials} \\ \ldots \end{cases}$$

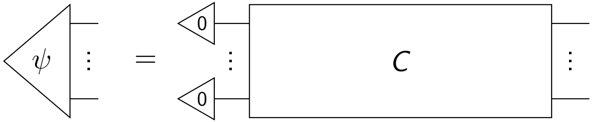
In this course...

- We'll look at **fundamental structures** underlying quantum computations:

$$\textbf{ZX-diagrams} \qquad \longleftrightarrow \qquad \begin{cases} \text{graph states} \\ \text{stabiliser groups} \\ \text{exponentiated Paulis} \\ \text{phase polynomials} \\ \ldots \end{cases}$$

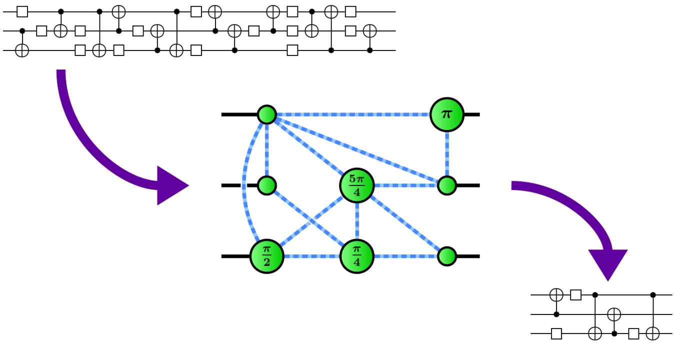
- And apply them to...

Classical simulation

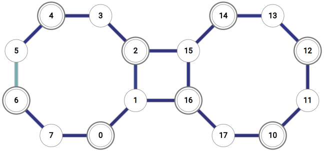


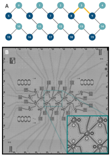
$$\text{Prob}\left(i \,\middle|\, |\psi\rangle\right) = ???$$

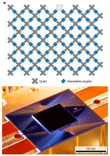
Circuit optimisation

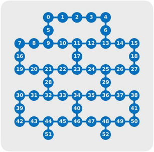


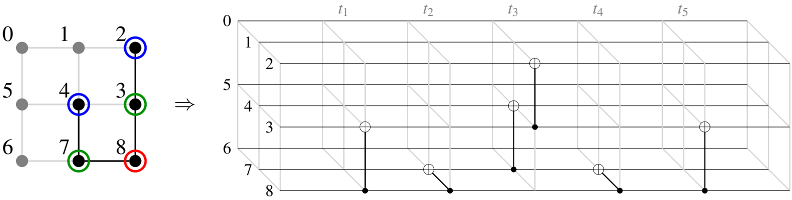
Circuit routing



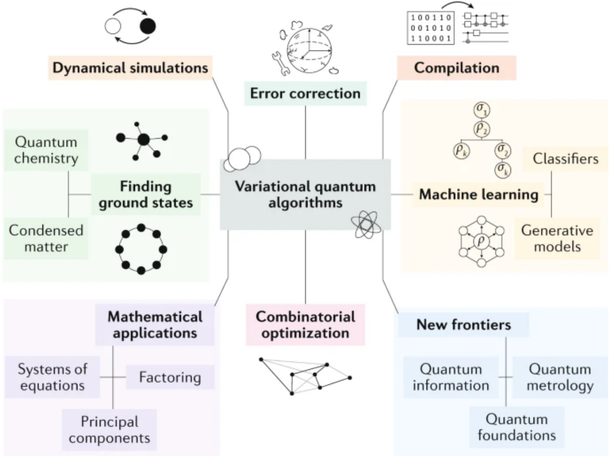








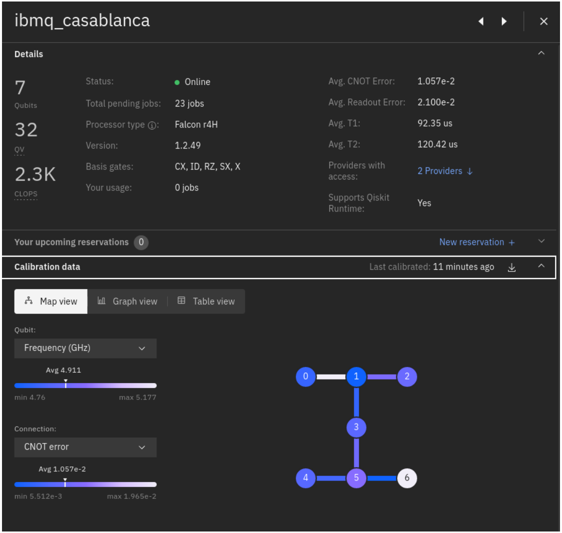
(NISQ) quantum algorithms



[3]

[3] https://www.nature.com/articles/s42254-021-00348-9

Implementation



[4]

[4] IBMQ calibration data. `https://quantum-computing.ibm.com`

Format of the course

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)
- 5 weeks theory (Aleks)
  - *basics, ZX, classical simulation, quantum compilation, quantum error correction*

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)
- 5 weeks theory (Aleks)
  - *basics, ZX, classical simulation, quantum compilation, quantum error correction*
- 3 weeks live coding (Stefano)
  - *algorithms, implementation on IBMQ*

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)
- 5 weeks theory (Aleks)
  - *basics, ZX, classical simulation, quantum compilation, quantum error correction*
- 3 weeks live coding (Stefano)
  - *algorithms, implementation on IBMQ*
- Materials:
  - *handwritten lecture notes, matching Aleks' lectures*
  - *textbook: preprint of Picturing Quantum Software. Kissinger & van de Wetering*
  - *Jupyter notebooks from live coding*

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)
- 5 weeks theory (Aleks)
  - *basics, ZX, classical simulation, quantum compilation, quantum error correction*
- 3 weeks live coding (Stefano)
  - *algorithms, implementation on IBMQ*
- Materials:
  - *handwritten lecture notes, matching Aleks' lectures*
  - *textbook: preprint of Picturing Quantum Software. Kissinger & van de Wetering*
  - *Jupyter notebooks from live coding*
  - *Picturing Quantum Processes*. Coecke & Kissinger. CUP (optional)
  - *Quantum Computation and Quantum Information*. Nielsen & Chuang. CUP (optional)

Format of the course

- 8 weeks, 24 lectures, 6 problem sheets + classes (week 3, 4, 5, 6, 7, 8)
- 5 weeks theory (Aleks)
  - *basics, ZX, classical simulation, quantum compilation, quantum error correction*
- 3 weeks live coding (Stefano)
  - *algorithms, implementation on IBMQ*
- Materials:
  - *handwritten lecture notes, matching Aleks' lectures*
  - *textbook: preprint of Picturing Quantum Software. Kissinger & van de Wetering*
  - *Jupyter notebooks from live coding*
  - *Picturing Quantum Processes*. Coecke & Kissinger. CUP (optional)
  - *Quantum Computation and Quantum Information*. Nielsen & Chuang. CUP (optional)
- Exam is by take-home miniproject
  - *expect a theory and a (Python) coding component*