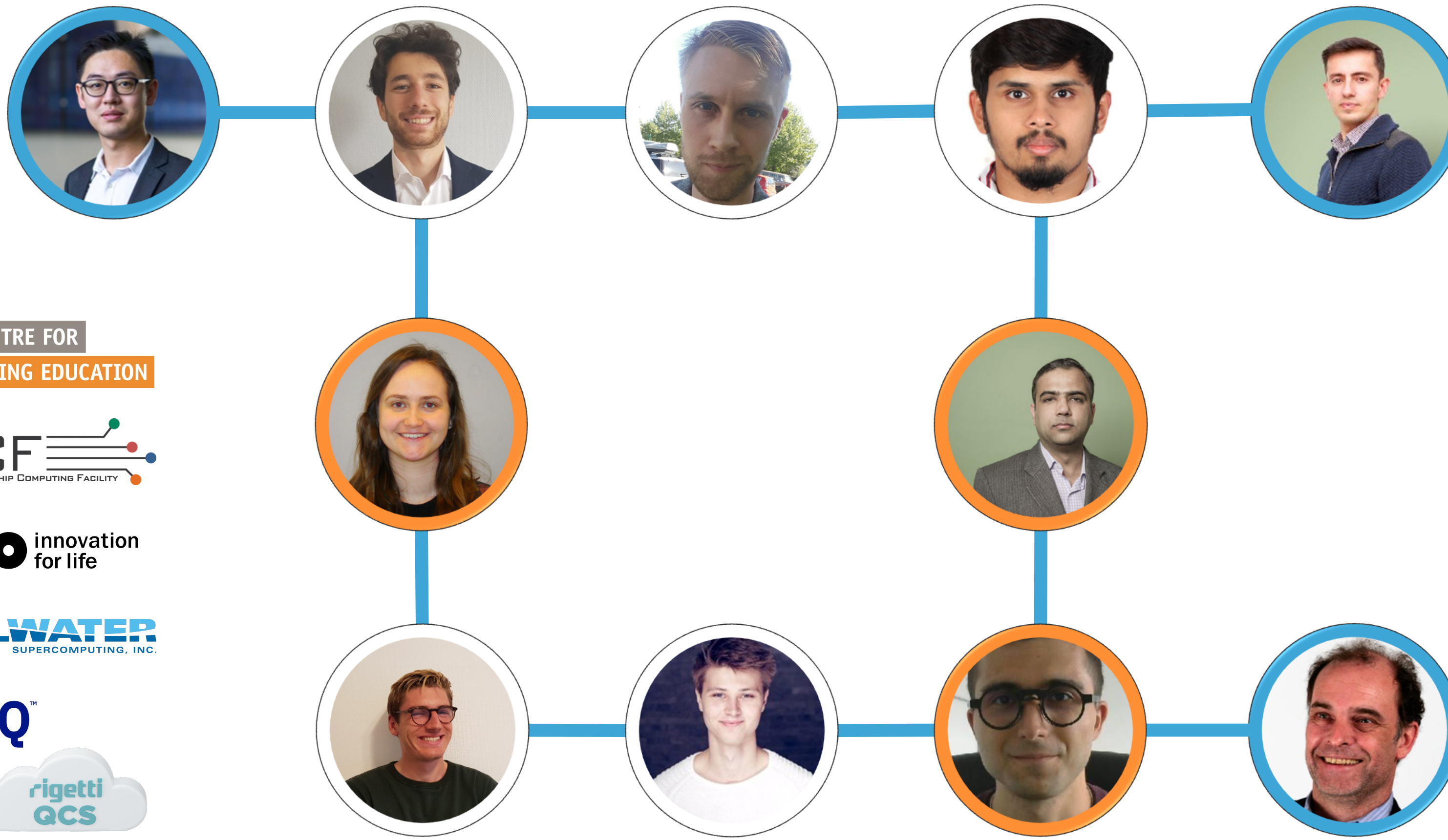
quantum-accelerated scientific computing: *concepts, programming tools and applications*

Matthias Möller

Delft University of Technology
Delft Institute of Applied Mathematics

collaborations, funding and support



4TU. CENTRE FOR ENGINEERING EDUCATION

OLCF
Oak Ridge Leadership Computing Facility

TNO innovation for life

STILLWATER SUPERCOMPUTING, INC.

IBM Q™

rigetti QCS

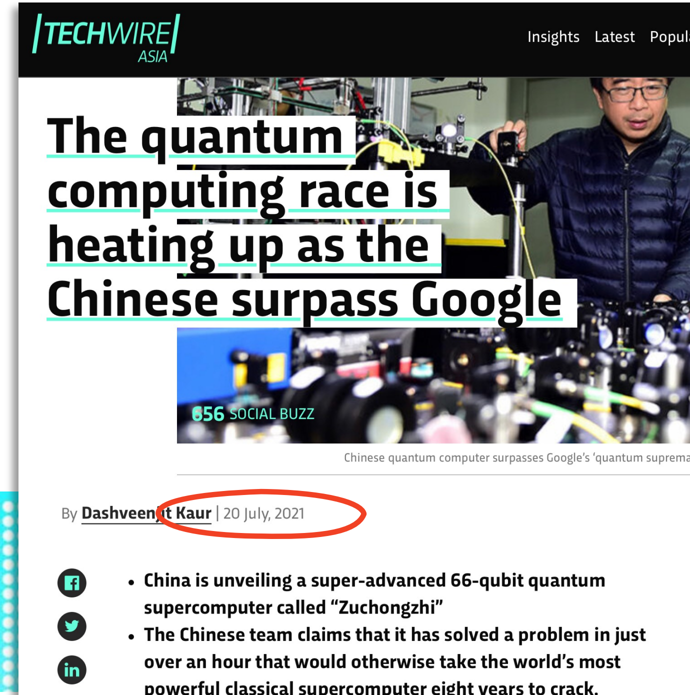
quantum computing in the news

Article

Quantum supremacy using a programmable superconducting processor

https://doi.org/10.1038/s41586-019-1666-5
Received: 22 July 2019
Accepted: 20 September 2019
Published online: 23 October 2019

Frank Arute[1], Kunal Arya[1], Ryan Babbush[1], Dave Bacon[1], Joseph C. Bardin[1,2], Rami Barends[1], Rupak Biswas[3], Sergio Boixo[1], Fernando G. S. L. Brandao[1,4], David A. Buell[1], Brian Burkett[1], Yu Chen[1], Zijun Chen[1], Ben Chiaro[5], Roberto Collins[1], William Courtney[1], Andrew Dunsworth[1], Edward Farhi[1], Brooks Foxen[1,5], Austin Fowler[1], Craig Gidney[1], Marissa Giustina[1], Rob Graff[1], Keith Guerin[1], Steve Habegger[1], Matthew P. Harrigan[1], Michael J. Hartmann[1,6], Alan Ho[1], Markus Hoffmann[1], Trent Huang[1], Travis S. Humble[7], Sergei V. Isakov[1], Evan Jeffrey[1],



October 21, 2019 | Written by: Edwin Pednault, John Gunnels & Dmitri Maslov, and Jay Gambetta

Recent advances in quantum computing have resulted in two 53-qubit processors: one from our group in IBM and a device described by Google in a paper published in the journal *Nature*. In the paper, it is argued that their device reached "quantum supremacy" and that "a state-of-the-art supercomputer would require approximately 10,000 years to perform the equivalent task." *We argue that an ideal simulation of the same task can be performed on a classical system in 2.5 days and with far greater fidelity*. This is in fact a conservative, worst-case estimate, and we expect that with additional refinements the classical cost of the simulation can be further reduced.

quantum computing in Europe

Computertechnologie

Ein Quantensprung für Deutschland?

Stand: 15.06.2021 18:25 Uhr

In Ehningen bei Stuttgart wurde Europas erster Quantencomputer eingeweiht. Der ultraschnelle Rechner der Firma IBM soll der Wirtschaft helfen, im Wettstreit mit China und den USA zu bestehen.

TECHNOLOGY NEWS MAY 11, 2021 / 10:52 AM / UPDATED 5 MONTHS AGO

Germany to support quantum computing with 2 billion euros

By Reuters Staff 2 MIN READ

TNO FOCUS AREAS COLLABORATION OUR IMPACT ABOUT TNO CAREER

THE FIRST EUROPEAN ONLINE QUANTUM COMPUTER PLATFORM

4 May 2020 • 3 min reading time

Leading universities and quantum hubs from China to America and the Netherlands are working on the development of a usable quantum computer. Within QuTech, TNO is working on innovative quantum technology in collaboration with Delft University of Technology and with some success, because a new version of the 'Quantum Inspire' quantum computing platform was launched on 20 April 2020. It is, in fact, the first European quantum computer platform that is generally accessible online.

09.04.2021 . Awards

Quantum Delta NL awarded 615 million euro from Netherlands' National Growth Fund to accelerate quantum technology

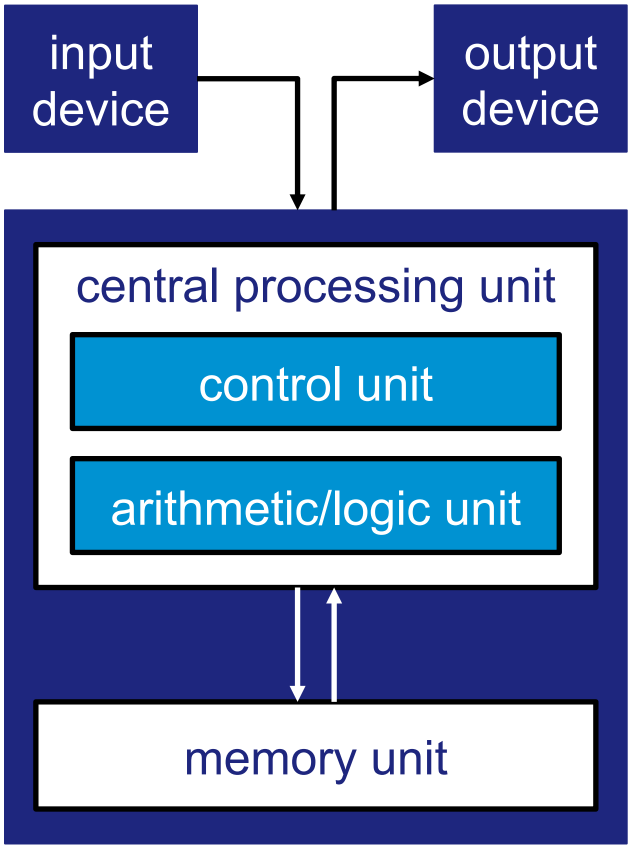
quantum-accelerated scientific computing

- **concepts**
  - *qubits, gates, and simple algorithms*
- **programming tools**
  - *LibKet and generation of resource-optimal quantum circuits*
- **applications**
  - *quantum linear solvers and optimization algorithms*
- **summary**

concepts

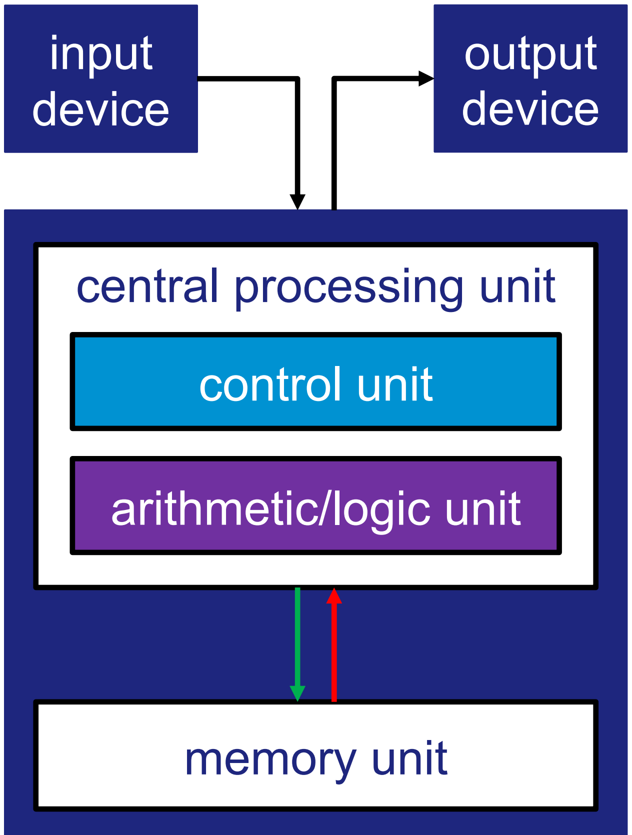
qubits, gates, and simple algorithms

von Neumann model



```
int a = 1;
int b = 2;
int c = a+b;
```

von Neumann model

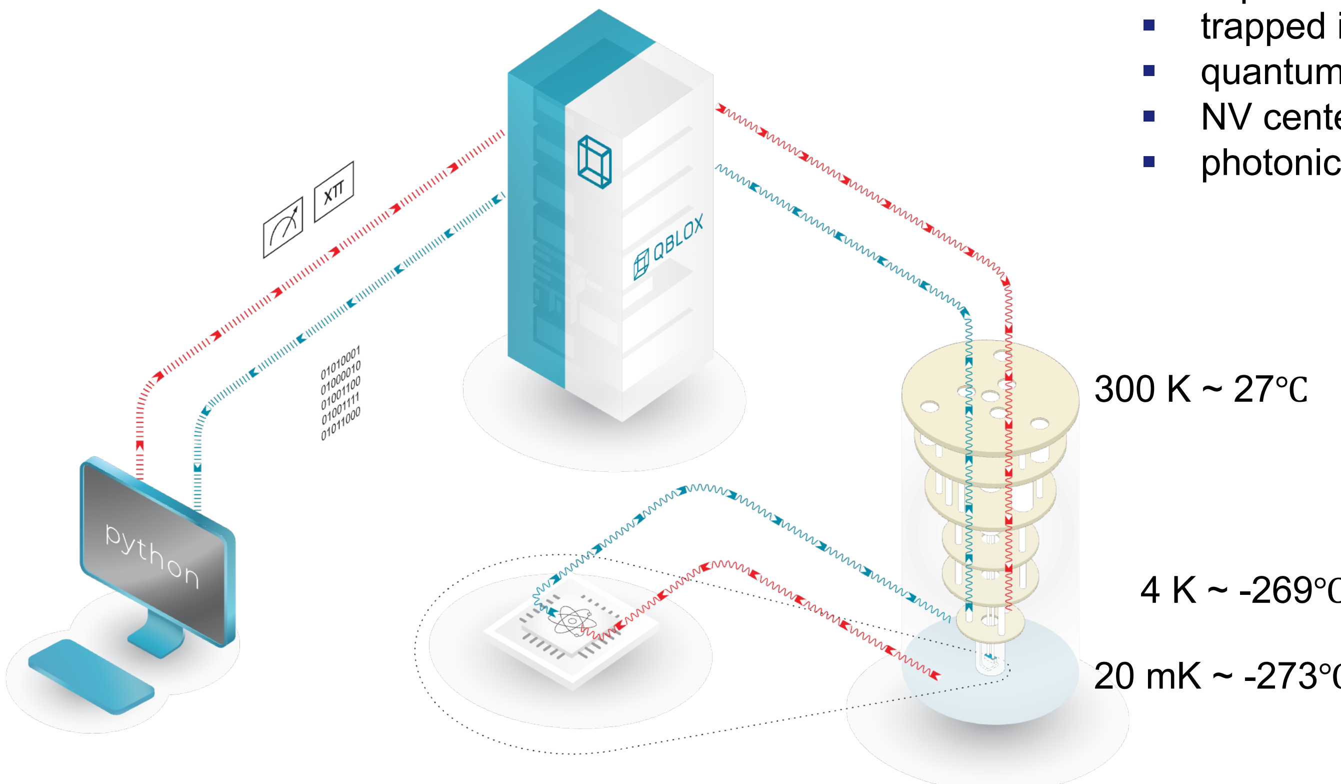


```
int a = 1;
int b = 2;
int c = a+b;
```

```
ld  r0 mem(a)      10001100000001010000001100000100000
ld  r1 mem(b)      10001100010010110000001001100010
add r0 r1 r2       10101101100010100000010110100110
sd  r2 mem(c)      10000100100010100000010000010011
```

a quantum computer model

- superconducting
- trapped ion
- quantum dots
- NV centers
- photonics (room temperature)

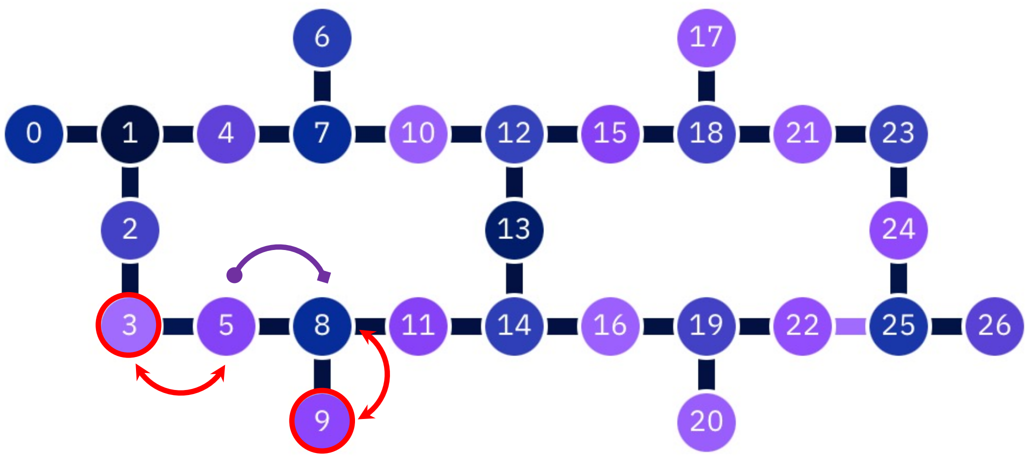


IBM's 27-qubit processor

controlled-NOT gate
between q3 and q9

```
swap q3 q5
swap q9 q8
cnot q5 q8
```

data is 'stored' in qubits
and can by manipulated
by 1- and 2-qubit 'gates'



2-qubit gates between nonadjacent
qubits require additional 'swap' ops

quantum bits

- **qubit**: quantum version of a bit

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1$$

- computational **basis**

$$\mathcal{E} = (|0\rangle, |1\rangle) = \left(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right)$$

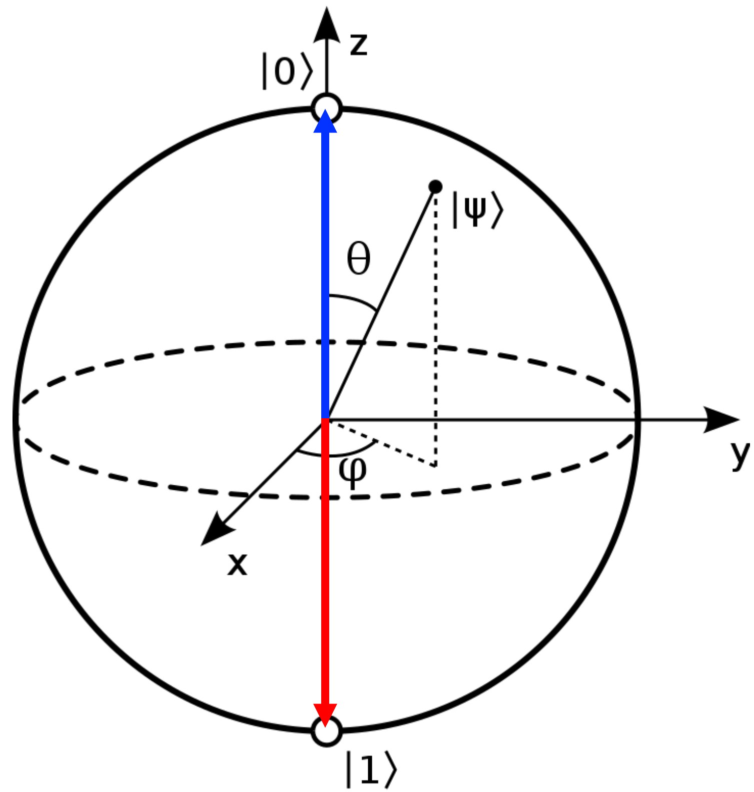
- coefficients α, β are the **probability amplitues** and $|\alpha|^2$ and $|\beta|^2$ are the **probabilities** of measuring the basis states $|0\rangle$ and $|1\rangle$, respectively

single-qubit states

- **Bloch sphere**

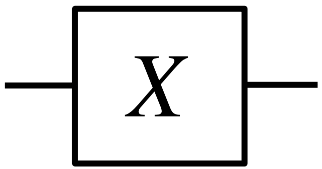
$$|\psi\rangle = e^{i\delta}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right)$$

  - polar angle $\theta \in [0,\pi]$
  - azimutal angle $\varphi \in [0,2\pi)$
  - ~~global phase δ~~



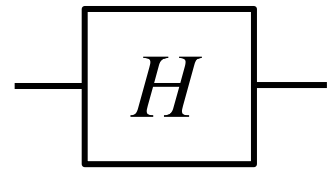
quantum gates

- **Pauli X**



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

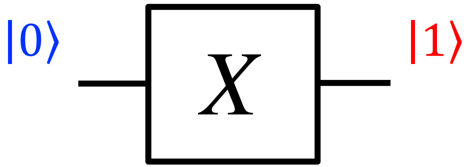
- **Hadamard**

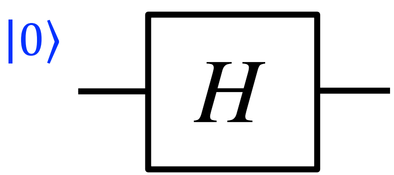


$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

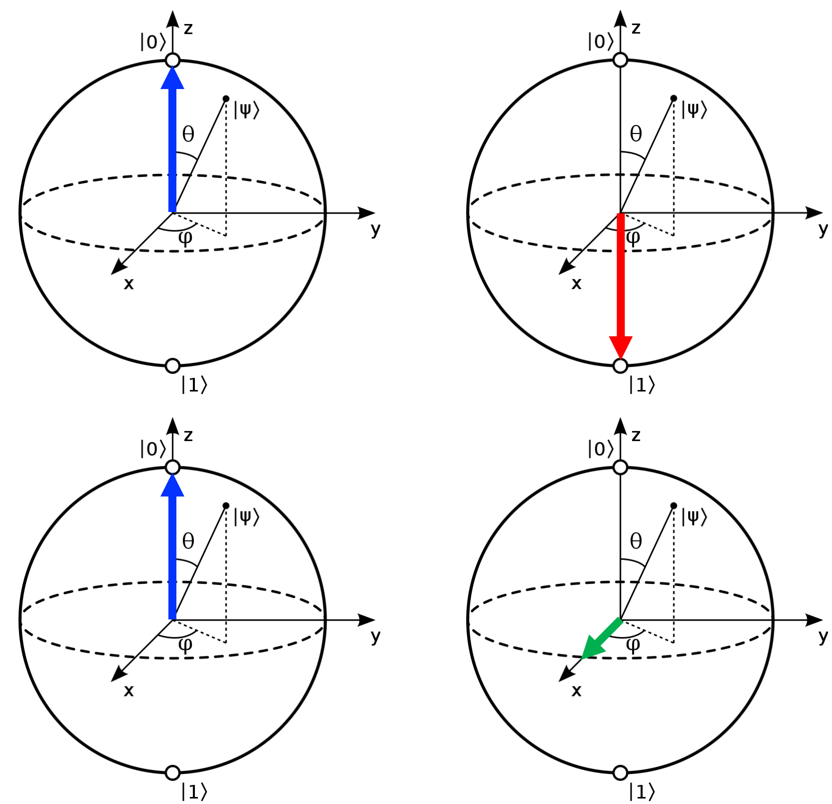
- unitary operations represented by unitary matrices
- all quantum gates are reversible, e.g. $HH^{\dagger} = I$

single-qubit gates

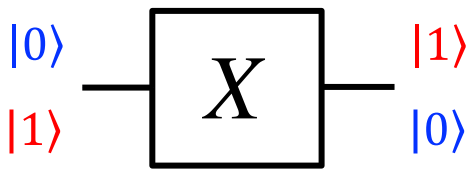


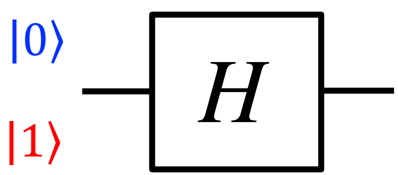


$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$



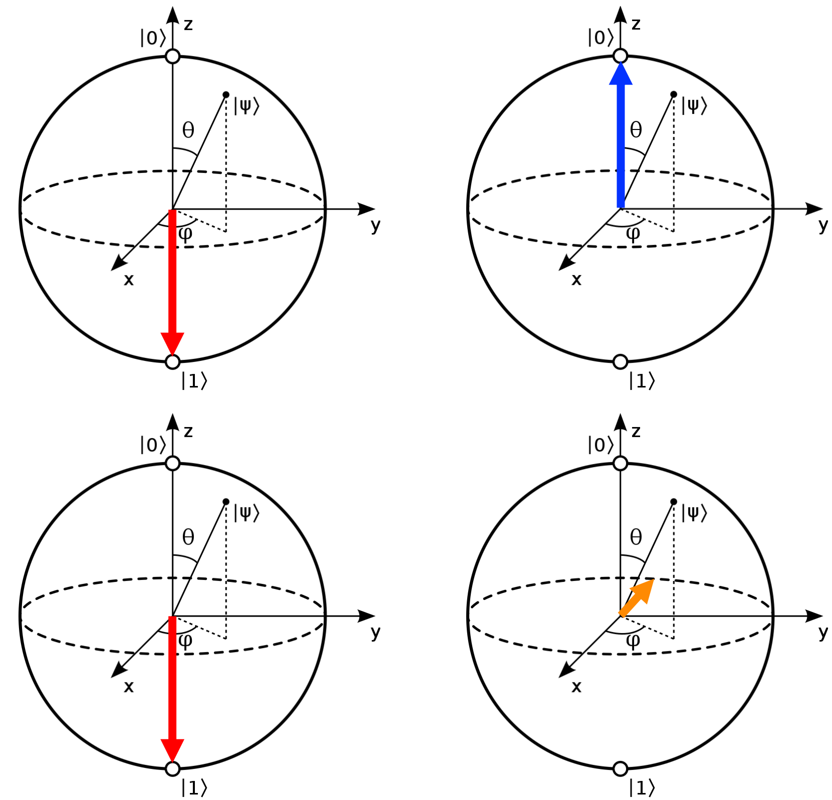
single-qubit gates



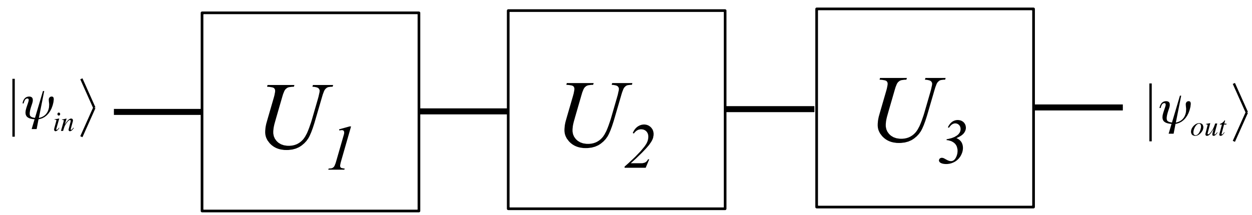


$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$



single-qubit circuits



- single-qubit gates U_k are **unitary matrices**, i.e.

$$U_k U_k^\dagger = U_k^\dagger U_k = I$$

- quantum circuits are sequences of matrix-vector multiplications

$$|\psi_{out}\rangle = U_3 U_2 U_1 \; |\psi_{in}\rangle$$

multi-qubit states

- $|\psi_0\rangle = \alpha_0|0\rangle + \beta_0|1\rangle = \alpha_0\begin{pmatrix}1\\0\end{pmatrix} + \beta_0\begin{pmatrix}0\\1\end{pmatrix}$
- $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle = \alpha_1\begin{pmatrix}1\\0\end{pmatrix} + \beta_1\begin{pmatrix}0\\1\end{pmatrix}$

tensor product

$$|A\rangle \otimes |B\rangle = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix}$$

- **tensor product** of two single-qubit states

$$|\psi_0\rangle \otimes |\psi_1\rangle = \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle =: |\psi_0\psi_1\rangle$$

with

$$|\alpha_0\alpha_1|^2 + |\alpha_0\beta_1|^2 + |\beta_0\alpha_1|^2 + |\beta_0\beta_1|^2 = 1$$

multi-qubit states

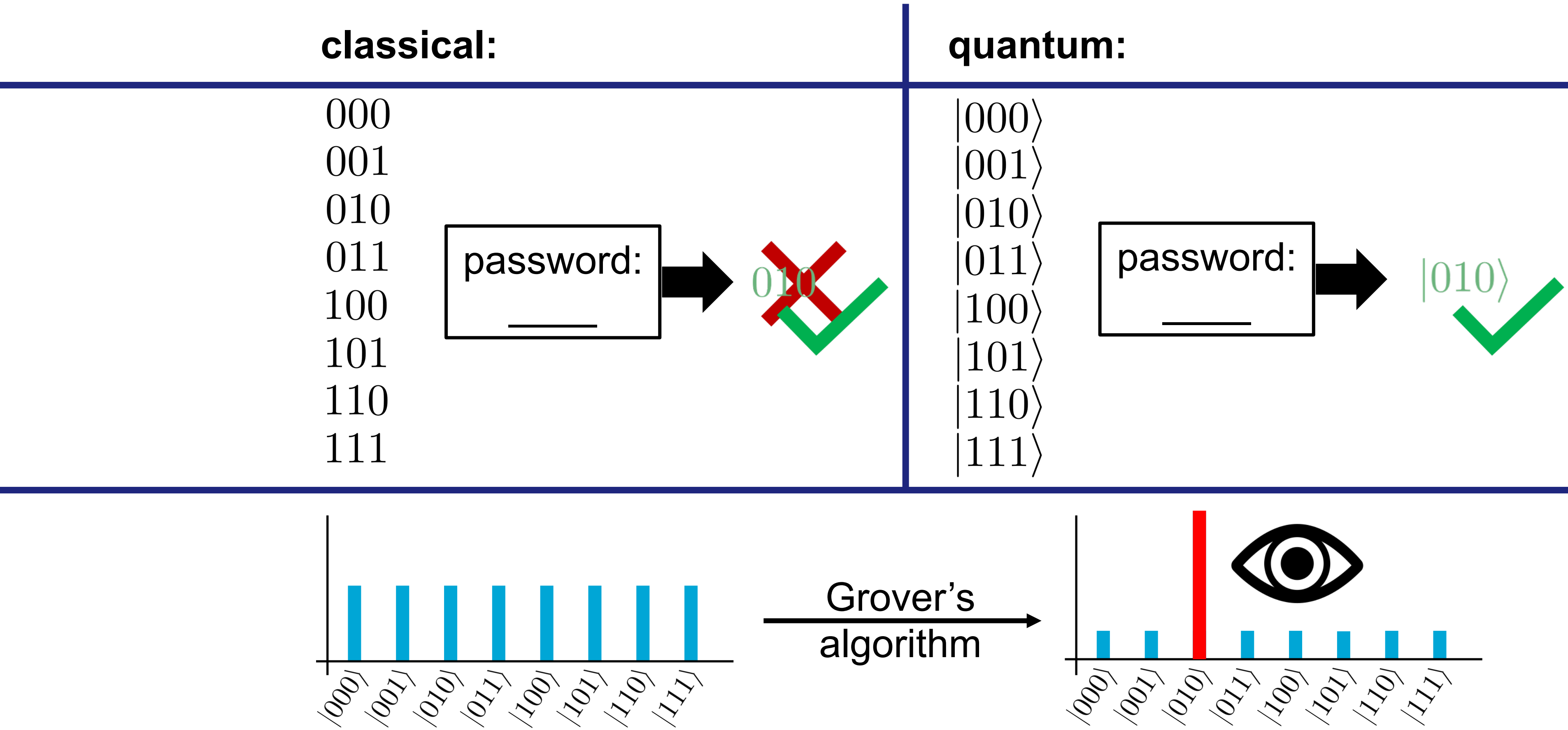
- **tensor product** of n single-qubit states

$$|\psi_0 \dots \psi_n\rangle = \gamma_{0\dots00}|0 \dots 00\rangle + \gamma_{0\dots01}|0 \dots 01\rangle + \cdots + \gamma_{1\dots11}|1 \dots 11\rangle$$

- an n-qubit register can hold the 2^n inputs ‘simultaneously’ in **superposition**

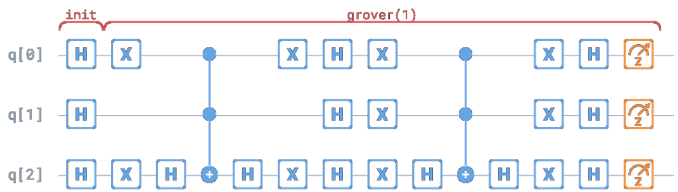
- **a few words of caution**
  - it is impossible to obtain the γ’s; one obtains a single binary answer, say, $|001101\rangle$ with probability $|\gamma_{001101}|^2$ upon **measurement**
  - a single run of a quantum circuit is not very useful; many runs are required to measure the correct answer with sufficient certainty

example: 3-bit password

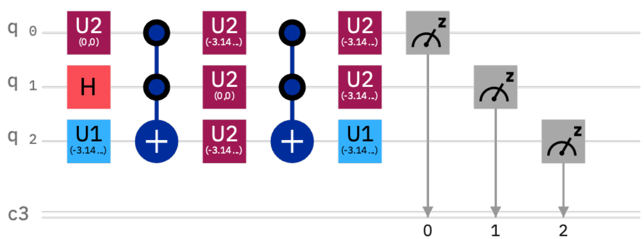


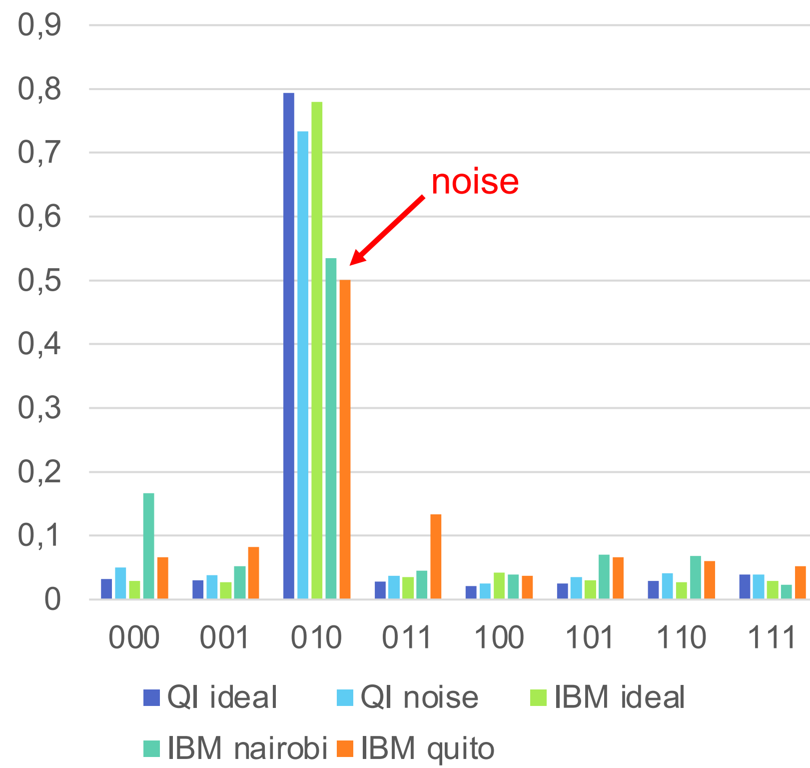
Grover’s algorithm

- quantum circuit on QI

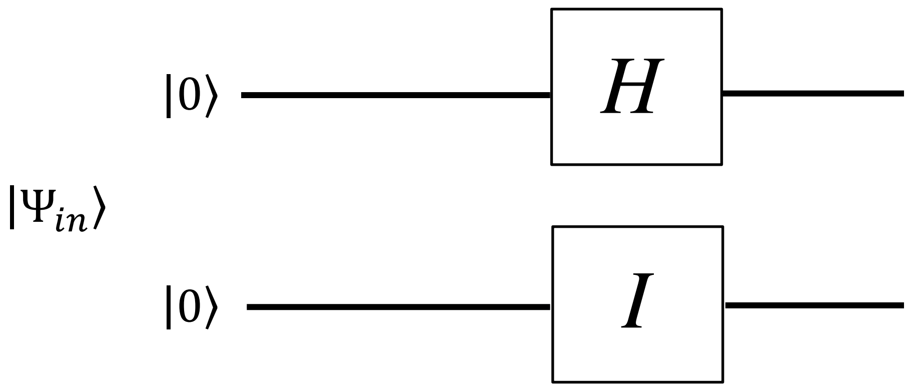


- quantum circuit on IBM





multi-qubit gates



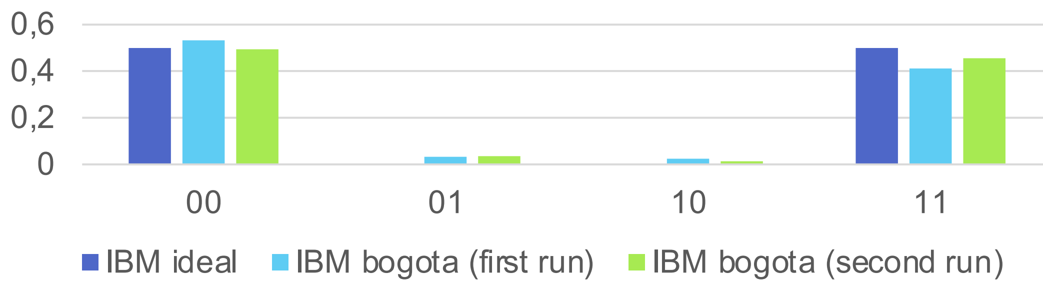
$$|\Psi_{out}\rangle = H \otimes I|\Psi_{in}\rangle$$

$$H \otimes I|00\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{|00\rangle + |10\rangle}{\sqrt{2}} = \frac{\left(|0\rangle + |1\rangle\right) \otimes |0\rangle}{\sqrt{2}}$$

entanglement

$$CNOT(H \otimes I)|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

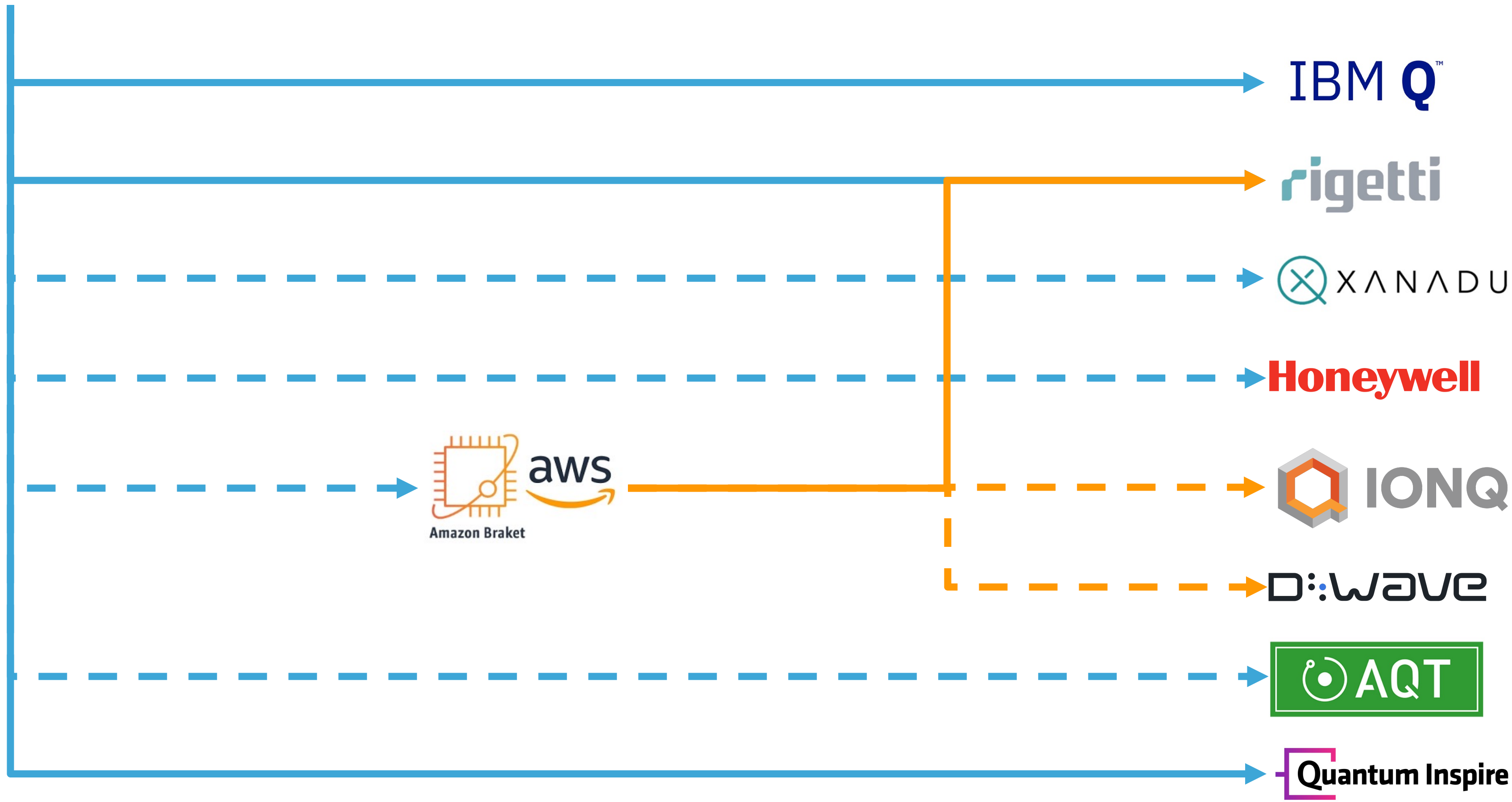
- Bell state is maximally entangled. By measuring one of the two qubits one knows the value of the other qubit without a further measurement



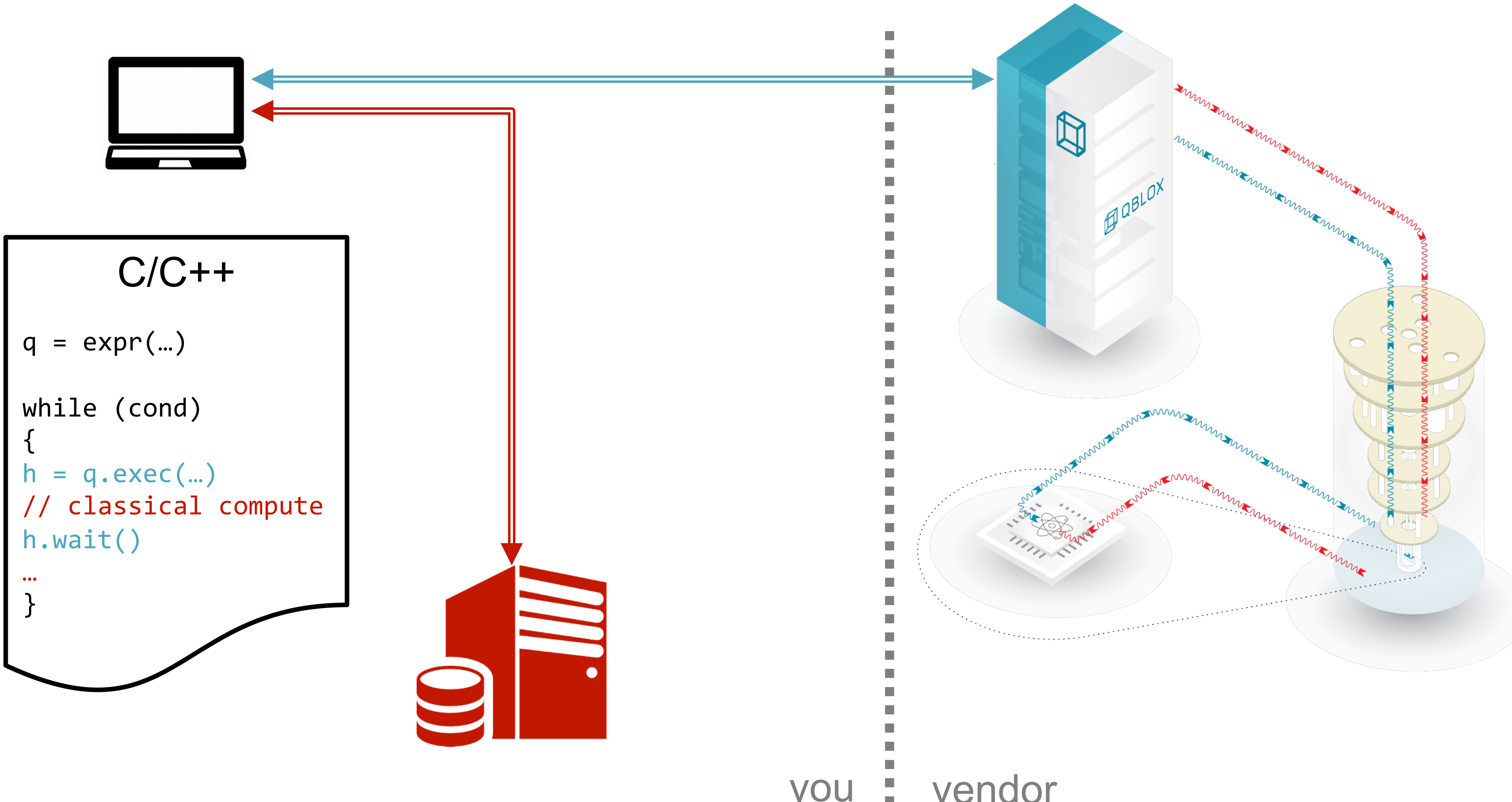
programming tools

LibKet and generation of resource-optimal quantum circuits

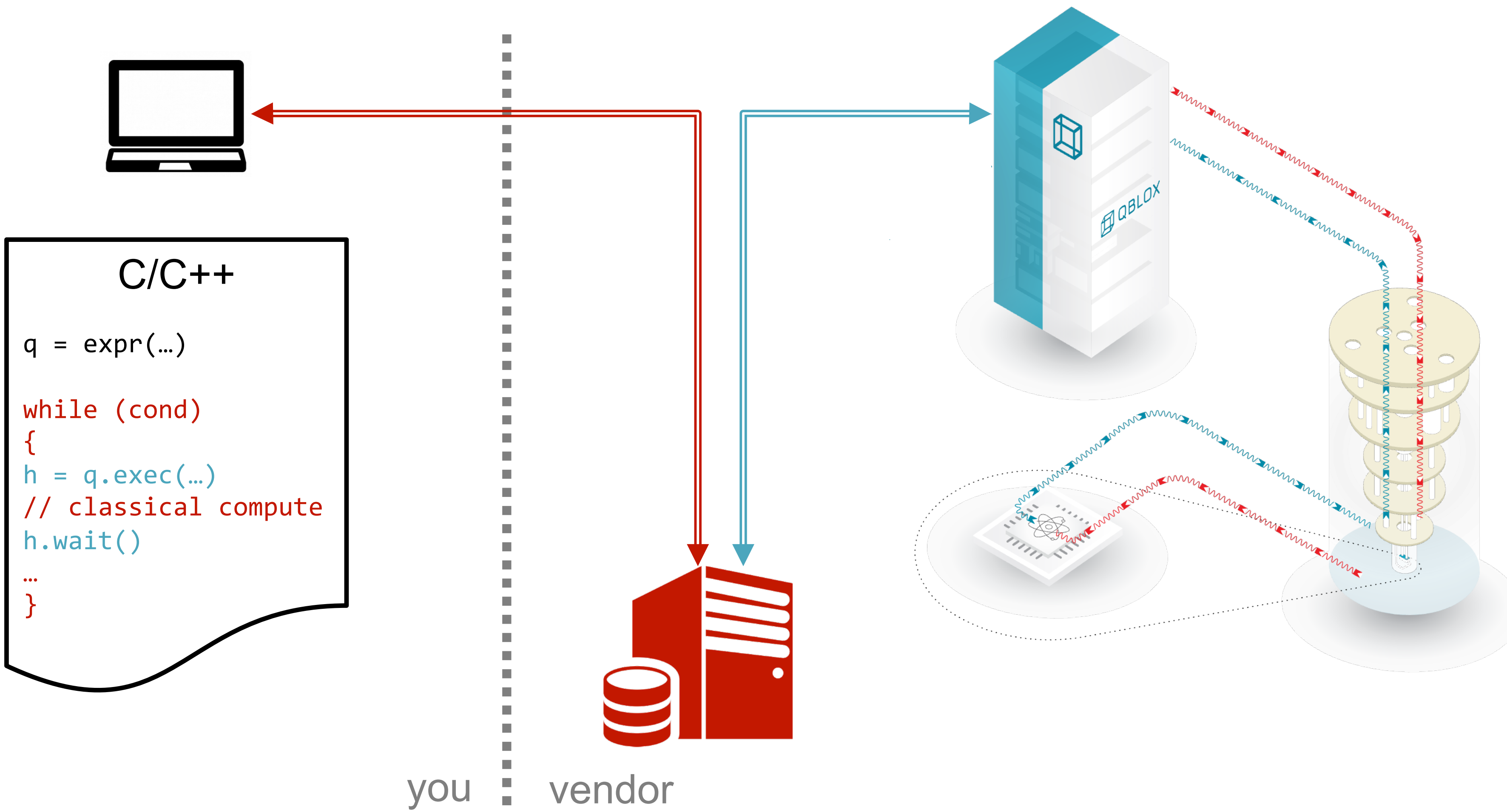
|Lib⟩ – The **k**wantum **e**xpression **t**emplate **Lib**rary



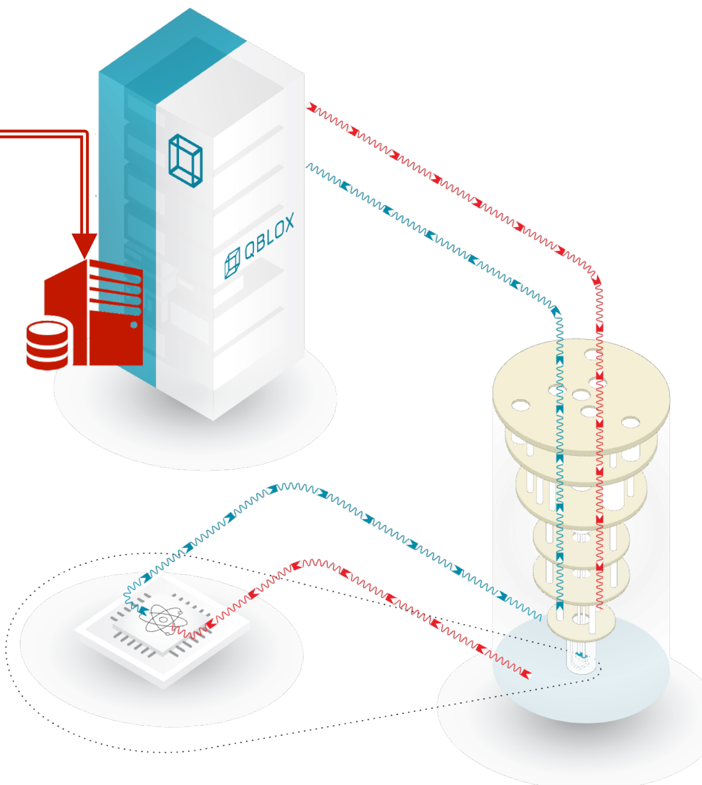
quantum acceleration workflow



quantum acceleration workflow



quantum acceleration workflow



C/C++

```
q = expr(…)

while (cond)
{
h = q.exec(…)
// classical compute
h.wait()
…
}
```

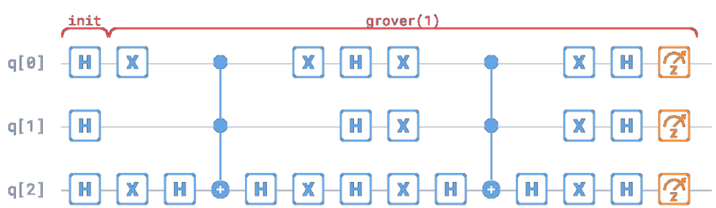
you | vendor

different programming philosophies

standard quantum SDKs

- apply gates to individual qubits

```
H   q[0:2]
X   q[0,2]
H   q[2]
CCX q[0], q[1], q[2]
...
```



LibKet

- ‘stream’ qubits through gates

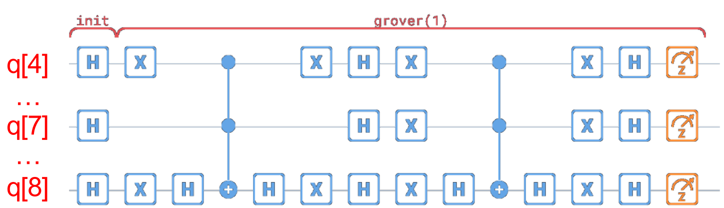
```
...CCX(q[0],
     q[1],
     q[2](
       H(q[2](
         X(q[0,2](
           H(q[0:2]())
         ))
       ))
     ))
```

different programming philosophies

standard quantum SDKs

- apply gates to individual qubits

```
H   q[4,7,8]
X   q[4,8]
H   q[8]
CCX q[4], q[7], q[8]
…
```



LibKet

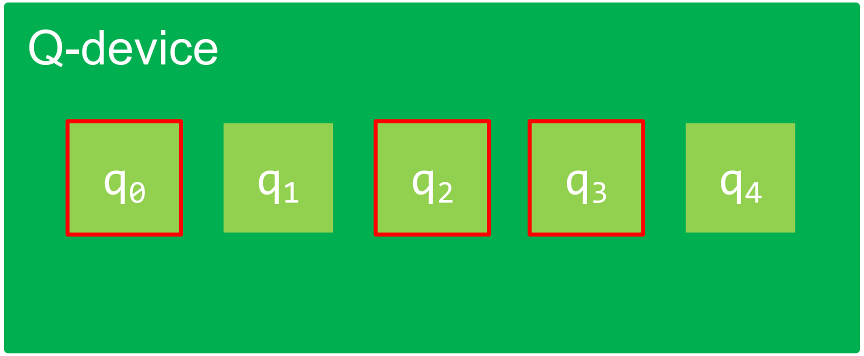
- ‘stream’ qubits through gates

```
…CCX(q[0],
     q[1],
     q[2](
       H(q[2](
         X(q[0,2](
           H(q[0:2](q[4,7,8]))
          ))
        ))
    ))
```

filters

- selective ‘views’ on the qubits

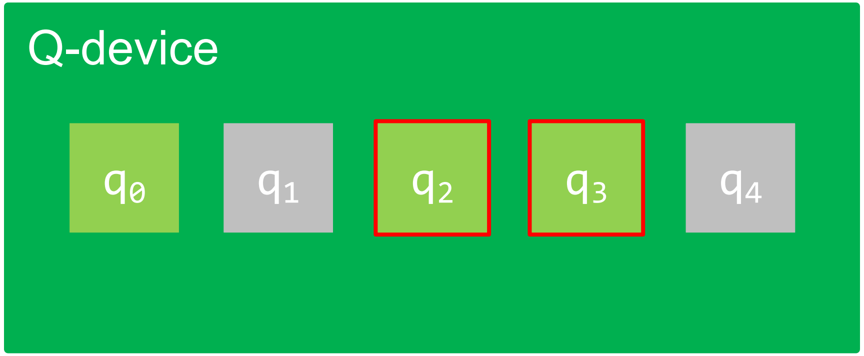
```
auto f0 = select<0,2,3>();
```



filters

- selective ‘views’ on the qubits

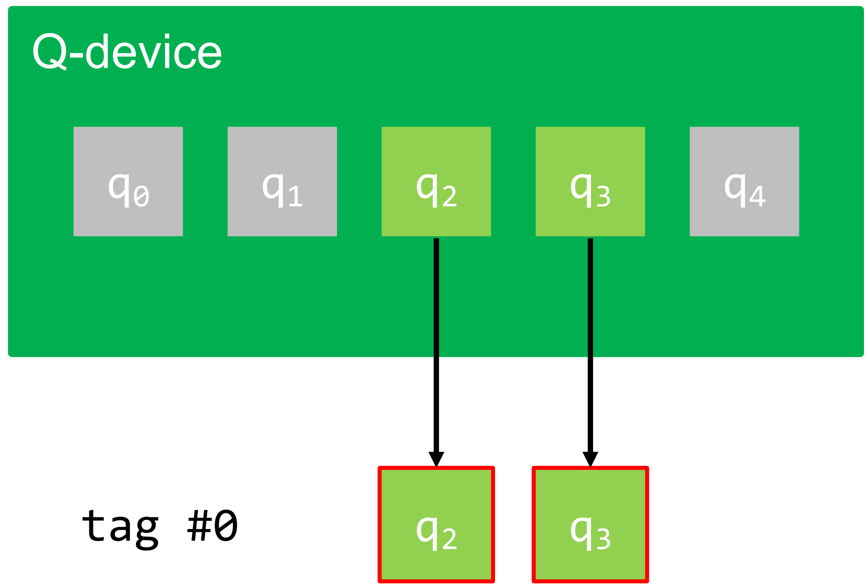
```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
```



filters

- selective 'views' on the qubits

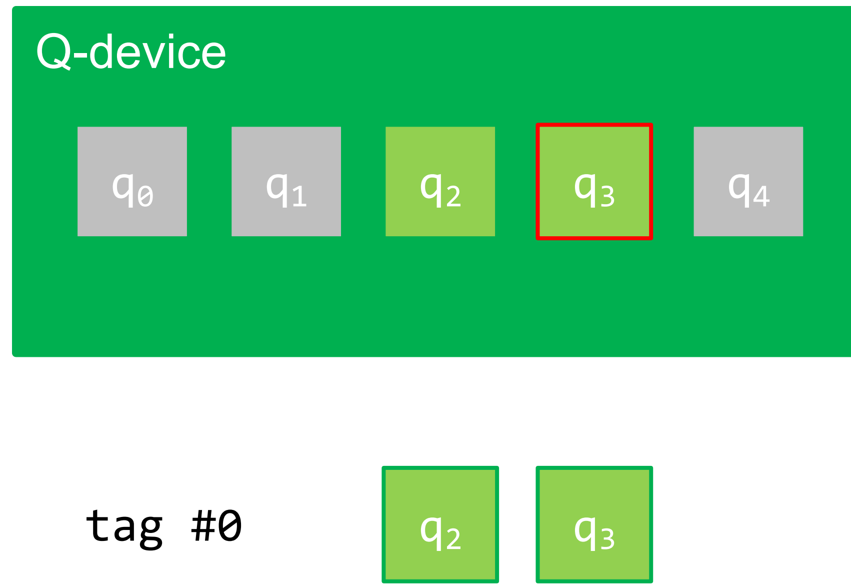
```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
auto f2 = tag<0>(f1);
```



filters

- selective ‘views’ on the qubits

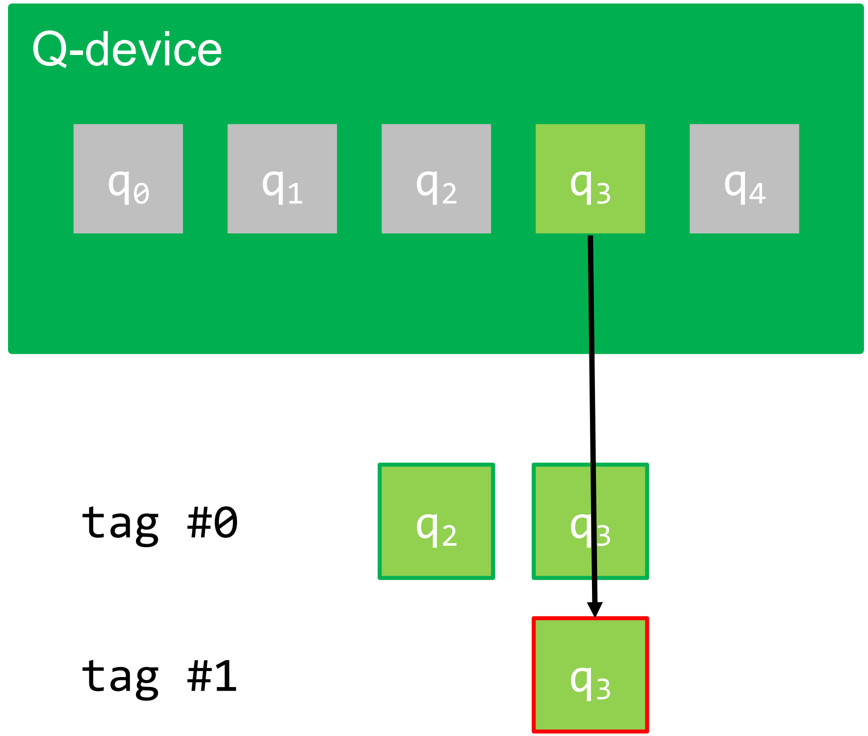
```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
auto f2 = tag<0>(f1);
auto f3 = qubit<1>(f2);
```



filters

- selective ‘views’ on the qubits

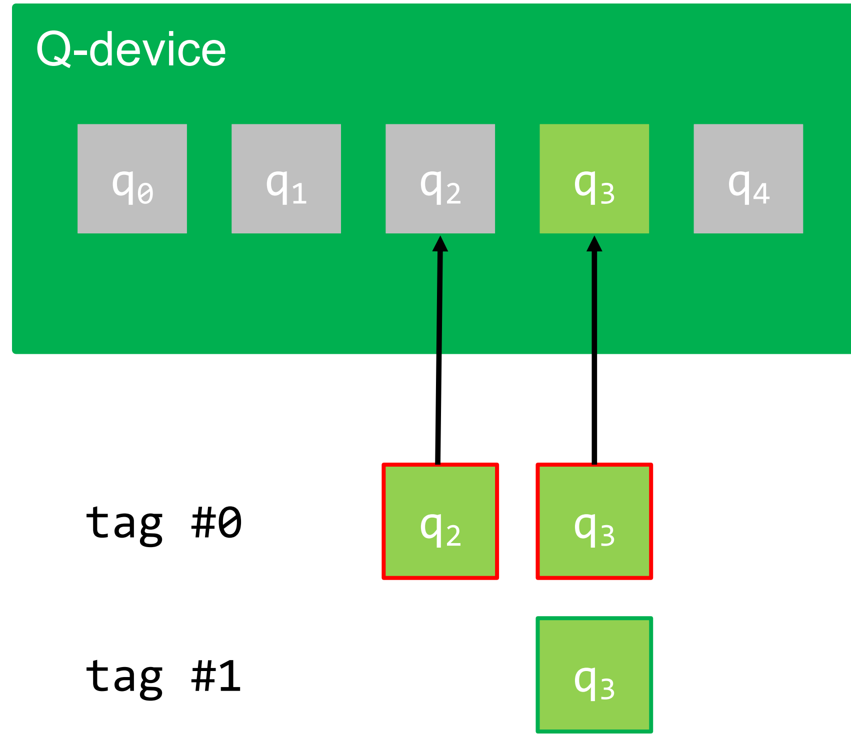
```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
auto f2 = tag<0>(f1);
auto f3 = qubit<1>(f2);
auto f4 = tag<1>(f3);
```



filters

- selective 'views' on the qubits

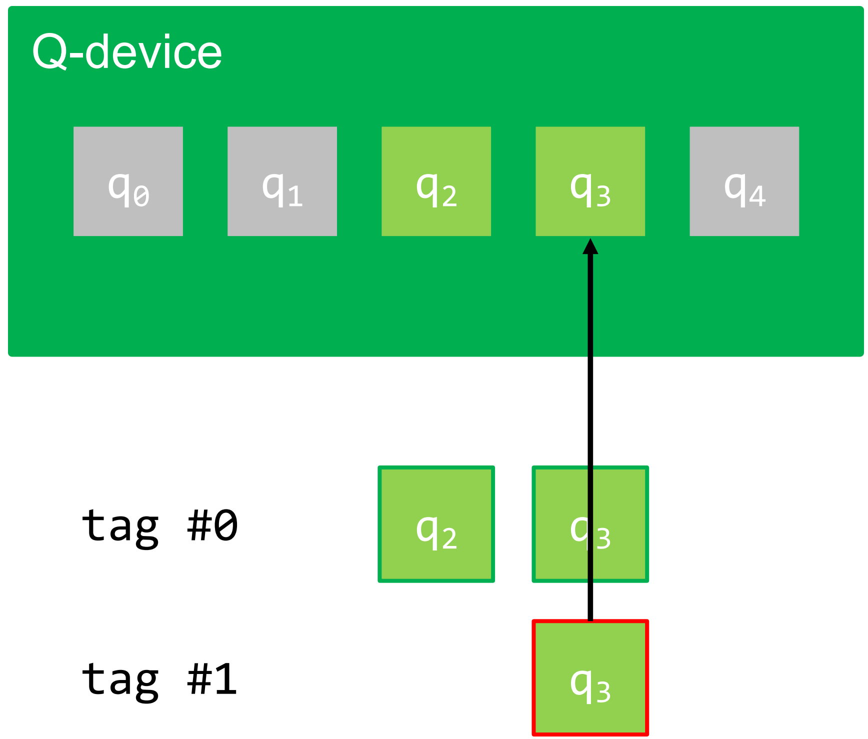
```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
auto f2 = tag<0>(f1);
auto f3 = qubit<1>(f2);
auto f4 = tag<1>(f3);
auto f5 = gototag<0>(f4);
```



filters

- selective ‘views’ on the qubits

```
auto f0 = select<0,2,3>();
auto f1 = range<1,2>(f0);
auto f2 = tag<0>(f1);
auto f3 = qubit<1>(f2);
auto f4 = tag<1>(f3);
auto f5 = gototag<0>(f4);
auto f6 = gototag<1>(f5);
```



gates

- SIMD-like quantum operation on all qubits of the current filter chain

```
auto e0 = init();
```

gates

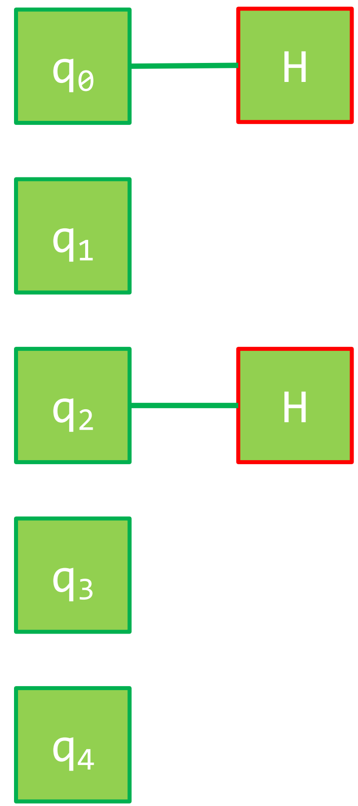
- SIMD-like quantum operation on all qubits of the current filter chain

```
auto e0 = init();
auto e1 = sel<0,2>(e0);
```

gates

- SIMD-like quantum operation on all qubits of the current filter chain

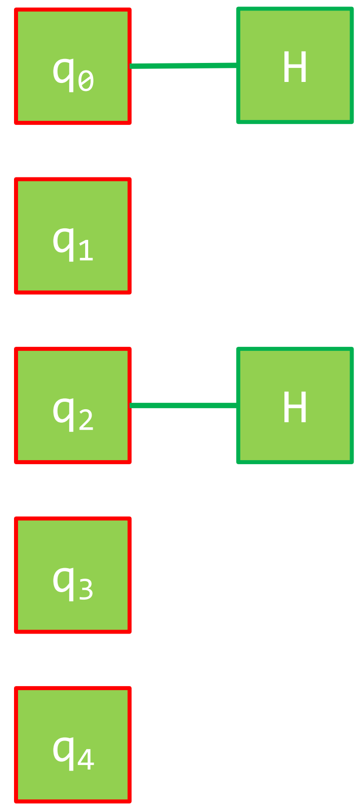
```
auto e0 = init();
auto e1 = sel<0,2>(e0);
auto e2 = h(e1);
```



gates

- SIMD-like quantum operation on all qubits of the current filter chain

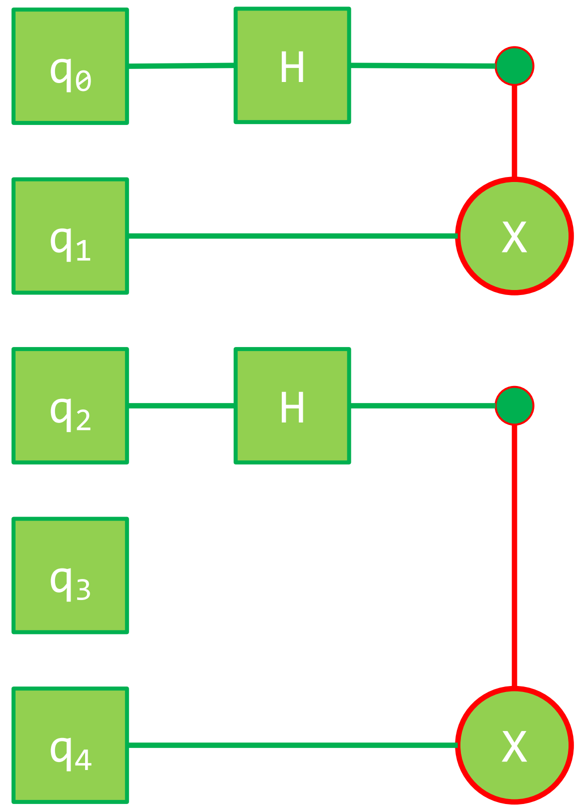
```
auto e0 = init();
auto e1 = sel<0,2>(e0);
auto e2 = h(e1);
auto e3 = all(e2);
```



gates

- SIMD-like quantum operation on all qubits of the current filter chain

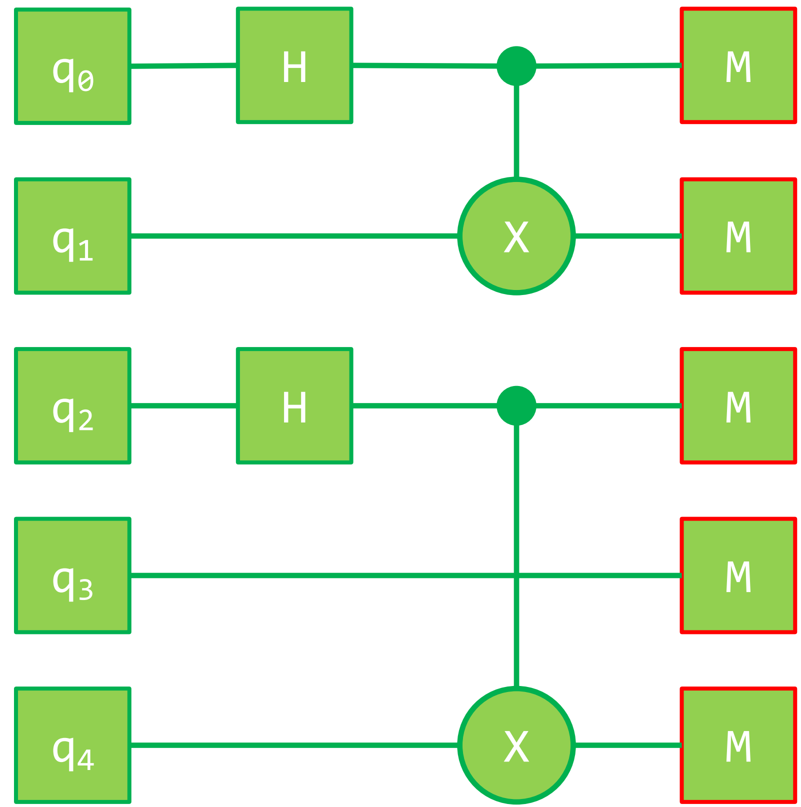
```
auto e0 = init();
auto e1 = sel<0,2>(e0);
auto e2 = h(e1);
auto e3 = all(e2);
auto e4 = cnot(
            sel<0,2>(),
            sel<1,4>(e3)
          );
```



gates

- SIMD-like quantum operation on all qubits of the current filter chain

```
auto e0 = init();
auto e1 = sel<0,2>(e0);
auto e2 = h(e1);
auto e3 = all(e2);
auto e4 = cnot(
            sel<0,2>(),
            sel<1,4>(e3)
         );
auto e5 = measure(all(e4));
```

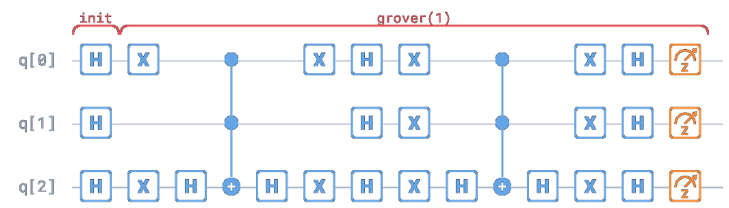


3-qubit Grover's algorithm

```
auto oracle = [](auto expr) {
  return x(sel_<0>(x(h(sel_<2>(ccnot(sel_<0>(),
                                     sel_<1>(),
                                     sel_<2>(h(x(sel_<2>(x(sel_<0>(expr))))))))))); };
```

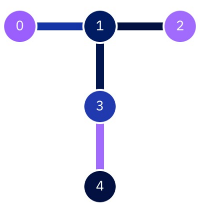
```
auto diffusion = [](auto expr) {
  return h(x(all(h(sel_<2>(ccnot(sel_<0>(),
                                 sel_<1>(),
                                 sel_<2>(h(sel_<2>(x(h(all(expr))))))))))); };
```

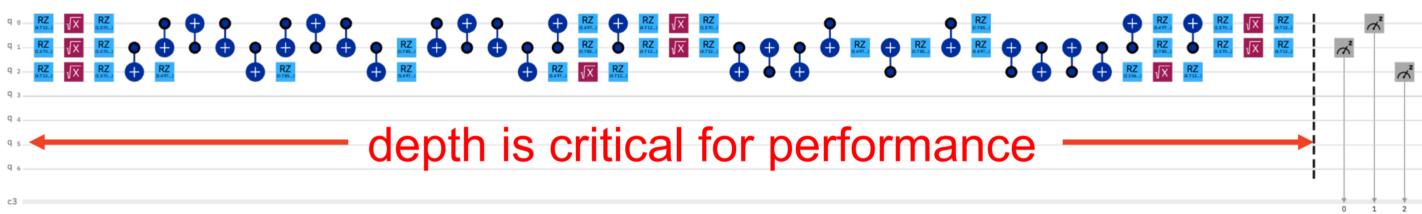
```
auto expr = measure(diffusion(oracle(h(init()))));
QDevice<backend, 3> device;
utils::json res = device(expr).eval(shots);
cout << device.get<QResultType::best>(res) << endl;
```



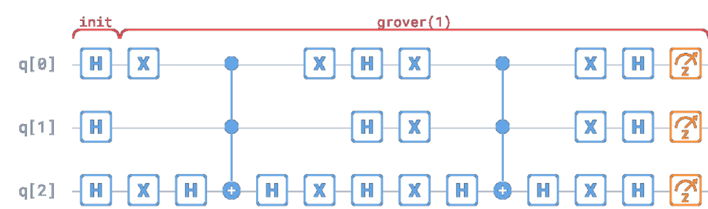
3-qubit Grover's algorithm

- IBM's basis gates: CX, ID, RZ, SX, X
- *executable* quantum circuit generated by IBM's quantum compiler






```
auto expr = measure(diffusion(oracle(h(init()))));
QDevice<QDeviceType::ibmq_quito, 3> device;
utils::json res = device(expr).eval(shots);
cout << device.get<QResultType::best>(res) << endl;
```



traditional quantum circuit compilation

- gate substitution rules

$$H \to R_x(\pi)R_y(\pi/2), \qquad H \to R_y(-\pi/2)R_x(\pi), \qquad \ldots$$

- cancelling of inverse gates

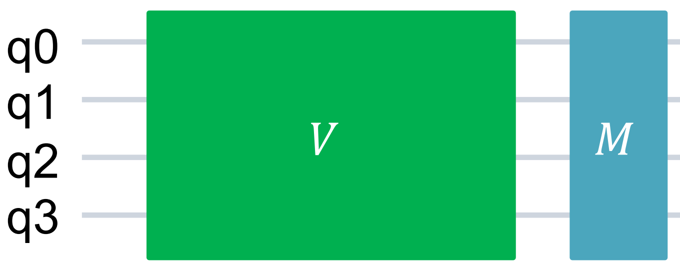
$$CZ\, CZ^\dagger = I, \qquad R_x(\theta)R_x(-\theta) = I, \qquad \ldots$$

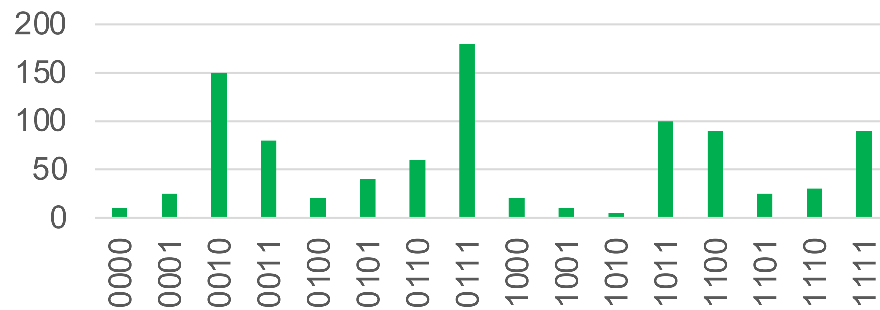
- aggregation using commutativity or fusion rules

$$HR_z(\theta)H = R_x(\theta), \theta \in \{\pi, \pm\pi/2\}, \qquad R_z(\theta_1)R_z(\theta_2) = R_z(\theta_1 + \theta_2), \qquad \ldots$$

approximate computing

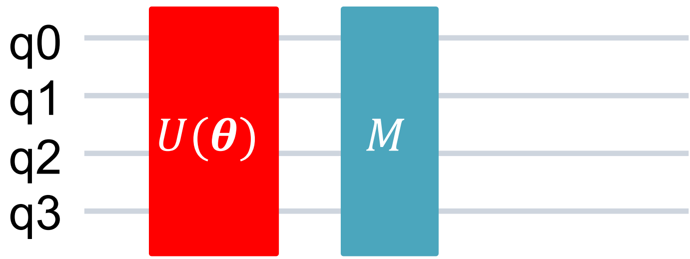
- our aim is to generate a resource-efficient directly executable circuit $U(\boldsymbol{\theta})$ that mimics the expectation-value behavior of the textbook circuit V

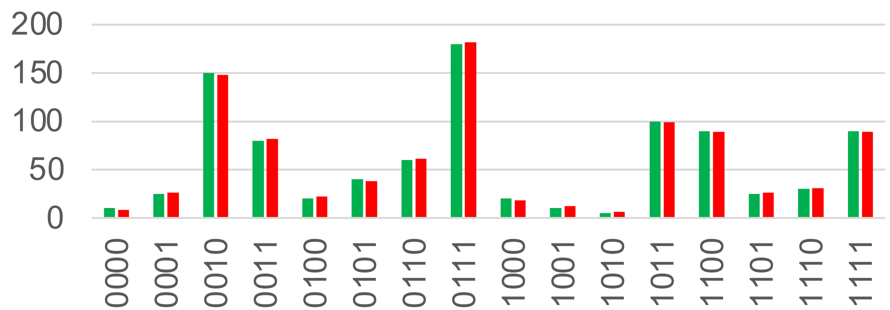




approximate computing

- our aim is to generate a resource-efficient directly executable circuit $U(\boldsymbol{\theta})$ that mimics the expectation-value behavior of the textbook circuit V





approximate computing

$$U_{\text{opt}} = \underset{U \in \mathcal{U}_s}{\operatorname{argmin}} \min_{\boldsymbol{\theta}_U} \max_{|\psi\rangle \in \Psi} F\big(|\psi\rangle; V, U(\boldsymbol{\theta}_U)\big), \qquad s \to \min$$

- cost function

$$F\big(|\psi\rangle; V, U(\boldsymbol{\theta}_U)\big) = \sum_k \Big(\big\langle A^k \big\rangle_{V|\psi\rangle} - \big\langle A^k \big\rangle_{U(\boldsymbol{\theta}_U)|\psi\rangle}\Big)$$

- A^k is an observable, e.g., Pauli-X, Y, Z gate
- expectation value of state $|\psi\rangle$ upon application of operator P

$$\big\langle A^k \big\rangle_{P|\psi\rangle} = \big\langle (P\psi)^\dagger \big| A^k \big| P\psi \big\rangle$$

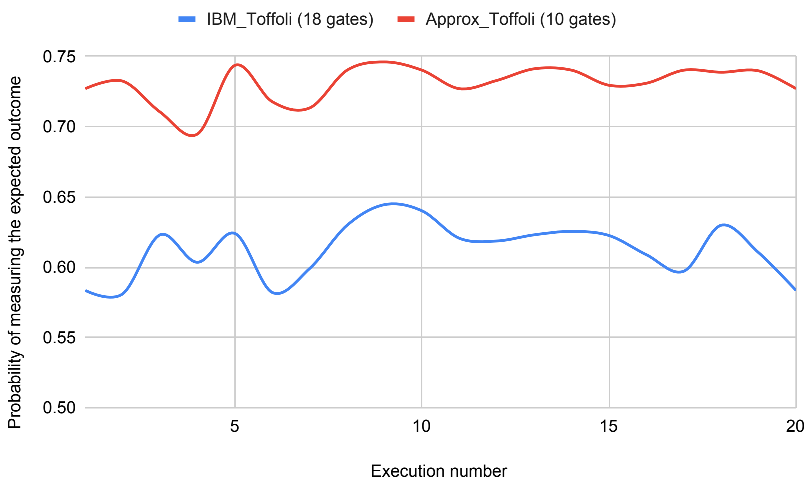
- $\mathcal{U}_s$ is the set of all admissible quantum circuits of size s
- $U(\boldsymbol{\theta}_U)$ is one parametrized quantum circuit with $\boldsymbol{\theta}_U = (\theta_1, \dots, \theta_N)^\top$

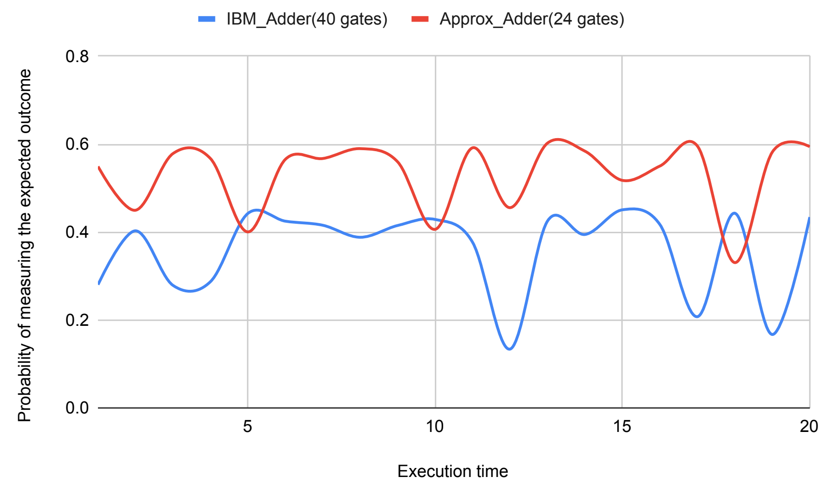
selected results

algo	#qubits	rigetti	ours		ibm	ours	
2q qft	2	18	16	11%	12-42	11	8-73%
3q qft	5	118	92	22%	57-73	56	1-23%
ccnot	3	33	15	54%	18	10	44%
4q add	8	197	132	33%	116-131	74	36-43%
8q add	16	474	312	34%	272-299	174	36-42%
mc3x	4	90	76	15%	46-102	36	21-64%
mc4x	5	195	164	16%	94-150	76	23-49%
2q grover	2	15	8	46%	16-27	8	50-71%
bv	4	21	11	47%	22	11	50%

S. Adarsh, M. Möller: Resource Optimal Executable Quantum Circuit Generation Using Approximate Computing. To appear the Proceedings of IEEE International Conference on Quantum Computing and Engineering (QCE21), 2021.

selected results





S. Adarsh, M. Möller: Resource Optimal Executable Quantum Circuit Generation Using Approximate Computing. To appear the Proceedings of IEEE International Conference on Quantum Computing and Engineering (QCE21), 2021.

applications

quantum linear solvers and optimization algorithms

potential quantum applications

$$\min_{\theta} x_\theta^\dagger M_\theta x_\theta \quad \text{s.t.} \quad A_\theta x_\theta = b_\theta$$

- **HHL-type quantum linear solver**

$$\text{Find} \quad x^\dagger M x \quad \text{s.t.} \quad Ax = b$$

  - sparse matrices $O(\log(N)\kappa^2/\epsilon)$ [Harrow, Hassidim, Lloyd 2009]
    $\text{polylog}(1/\epsilon)$ [Childs, Kothari, Somma 2017]
  - dense matrices $O(\sqrt{N}\log(N)\kappa^2/\epsilon)$ [Wossnig et al. 2018]

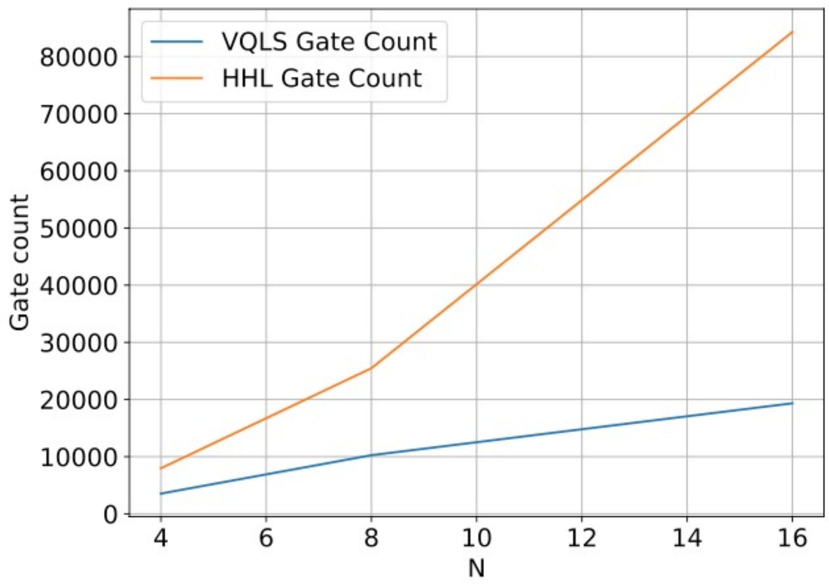
- **applications**
  - linear differential equations [Berry 2010, Xin et al. 2018]
  - nonlinear differential equations [Leyton, Osborne 2008, Liu et al. 2021]
  - Poisson equation [Cao et al. 2013, Montanaro 2015]
  - principal component analysis [Lloyd et al. 2014]
  - data fitting [Wiebe et al. 2012]
  - machine learning [Lloyd et al. 2013, Adcock et al. 2015, Biamonte et al. 2017, Schuld et al. 2018, Perdomo-Ortiz et al. 2018, …]

caveats

- you don't get the solution vector x but a scalar value $x^\dagger M x$
- circuits are impractical for near-future quantum computers

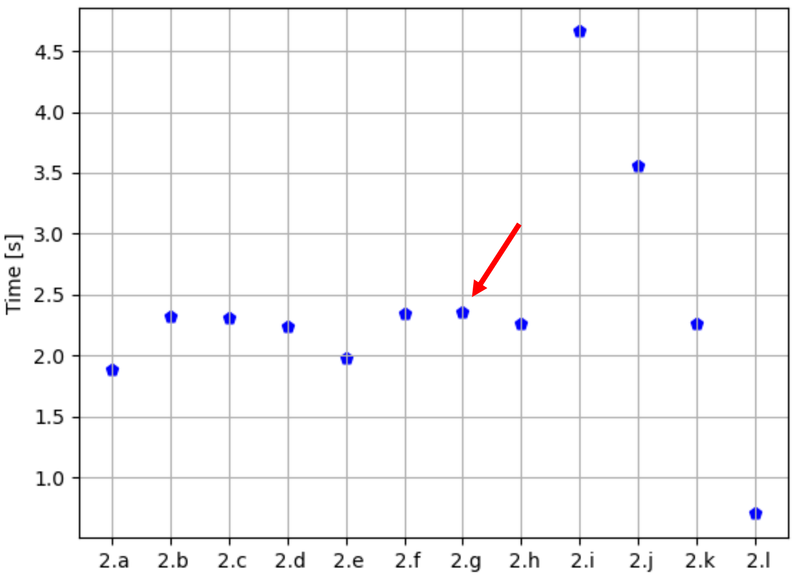
- Recent step-by-step HHL algorithm walk-through by Morrell and Wong (08/2021):

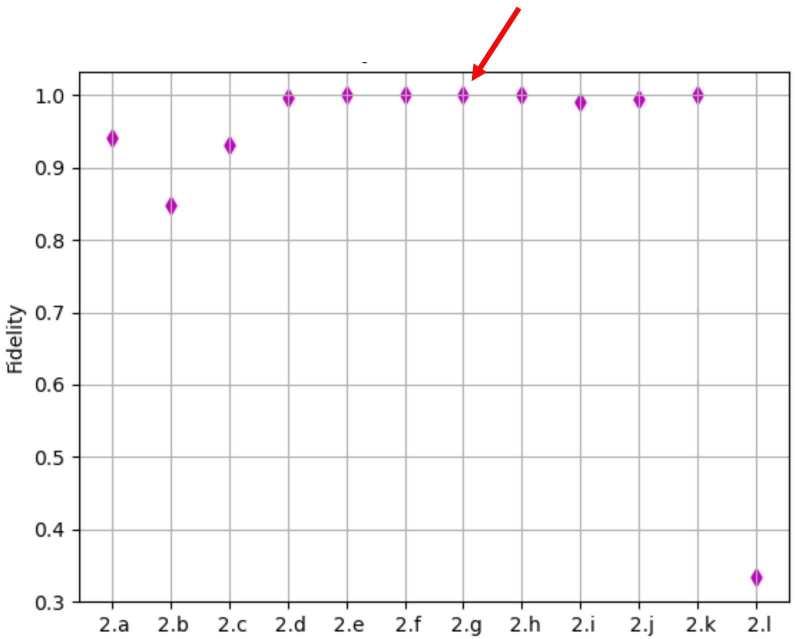
  "[…] due to the imperfection and noise in a real quantum computer (ibmq_santiago), the hardware execution of the same circuit (for a 2x2 matrix) does not give satisfactory result"
  arXiv:2108.09004



E. Cappanera: Variational quantum linear solver for finite element problems, Master Thesis TU Delft, 2021.

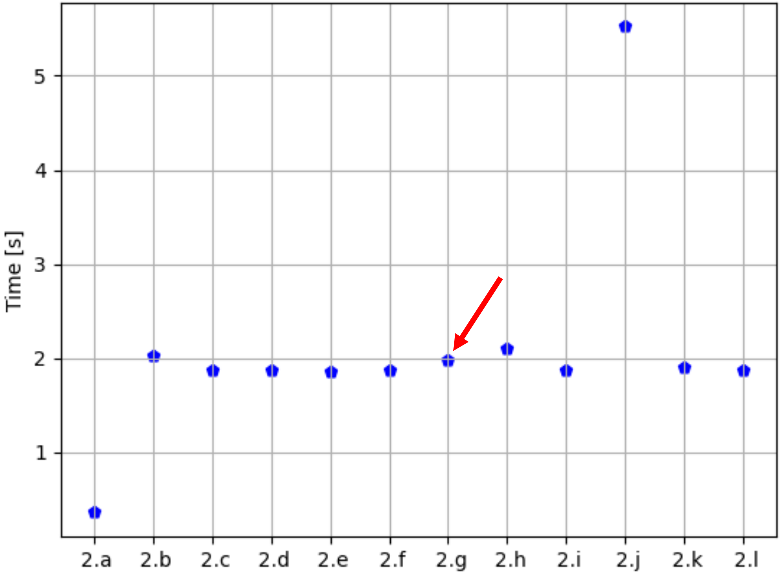
HHL simulation with Qiskit: 2x2 matrices, w/o noise

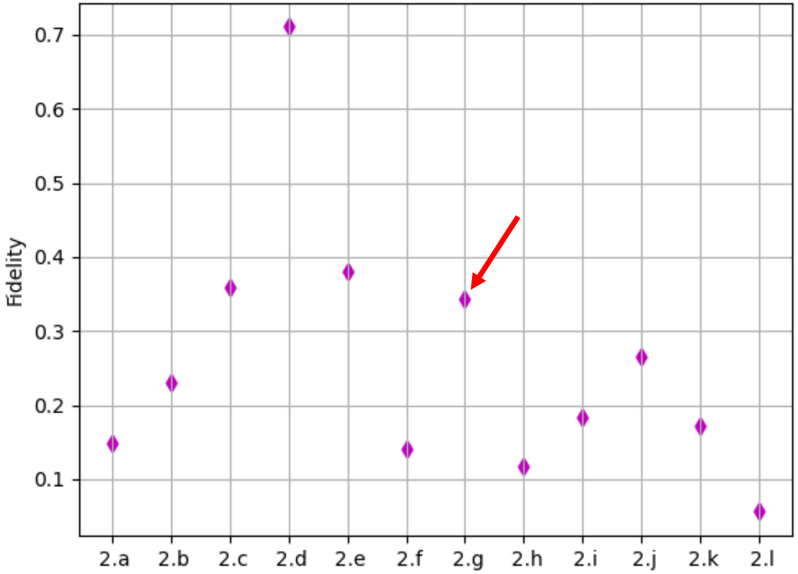




S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

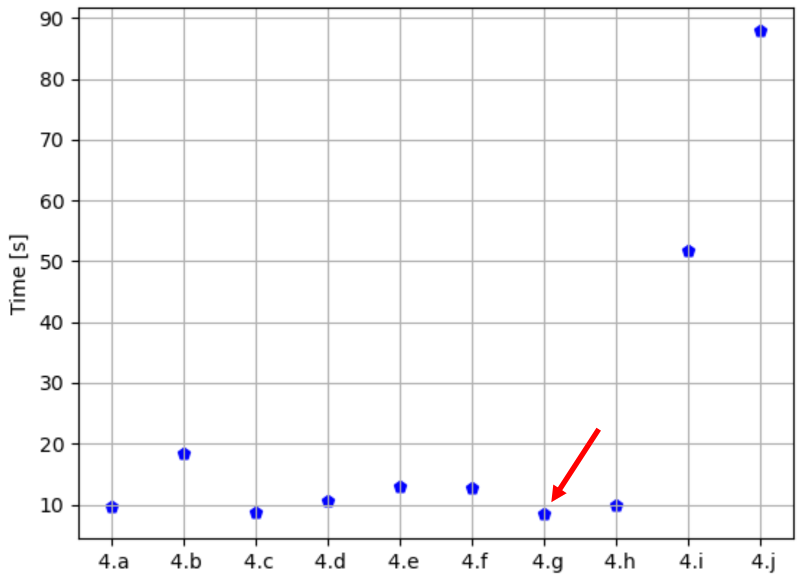
HHL simulation with Qiskit: 2x2 matrices, with noise

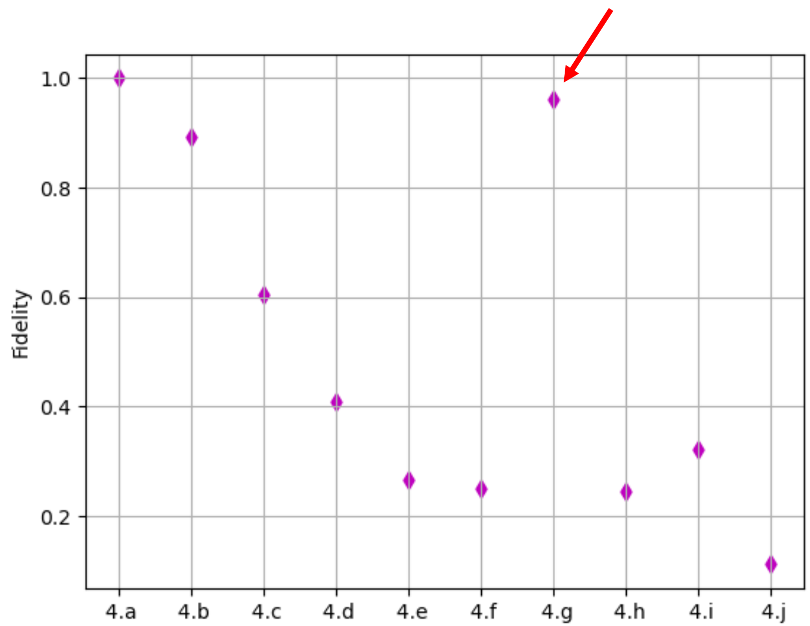




S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

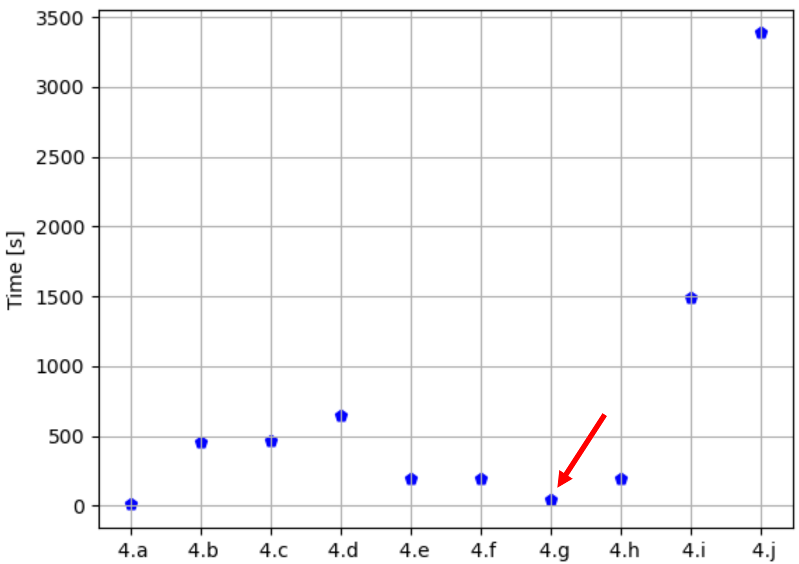
HHL simulation with Qiskit: 4x4 matrices, w/o noise

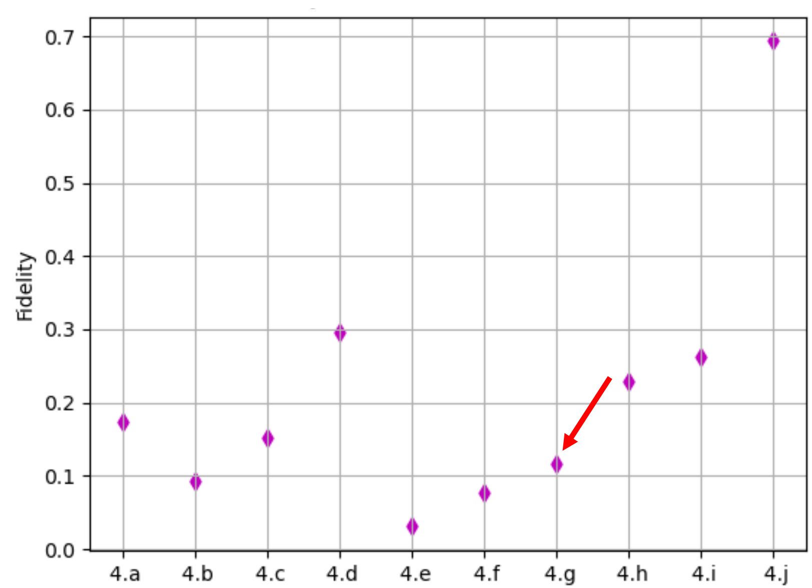




S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

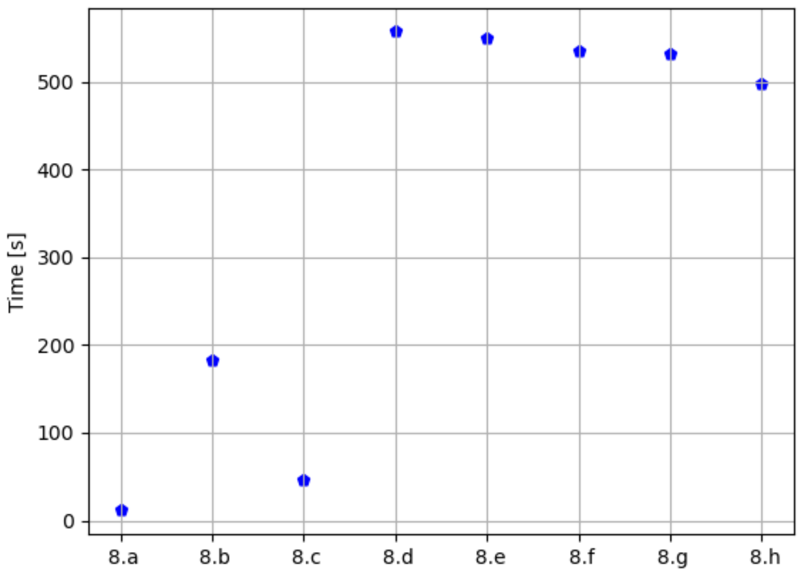
HHL simulation with Qiskit: 4x4 matrices, with noise

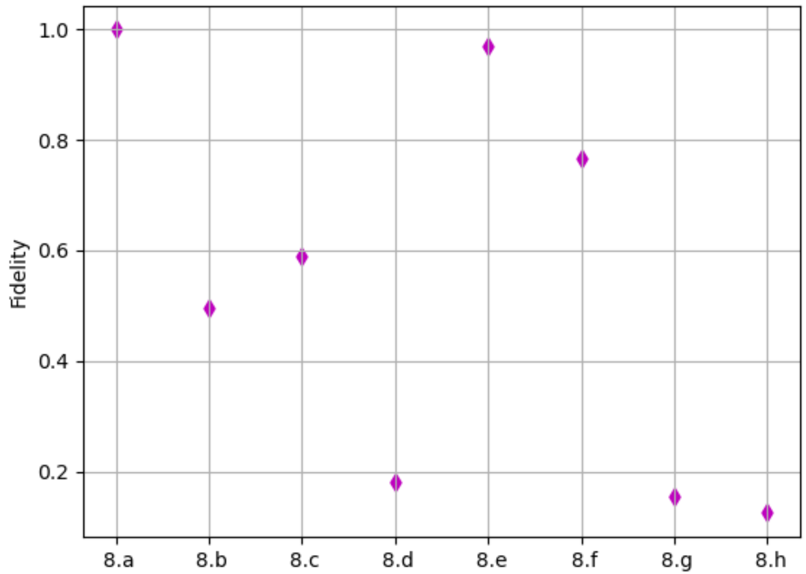




S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

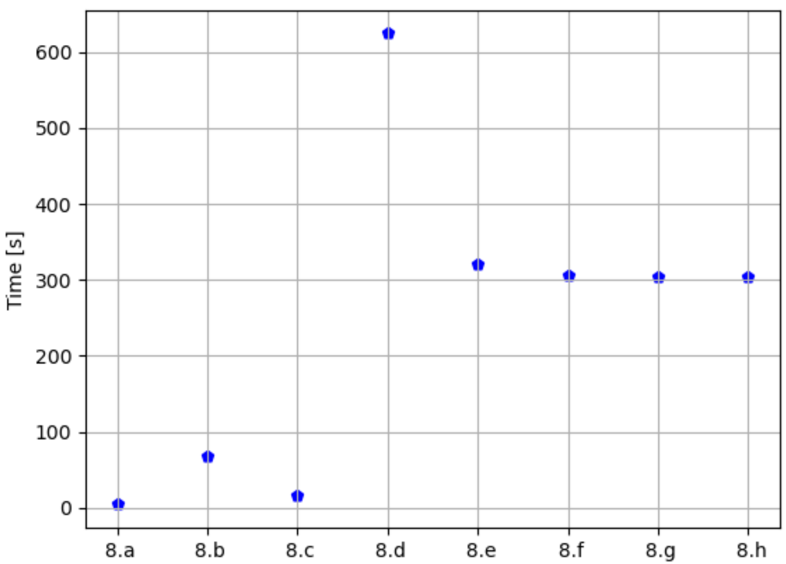
HHL simulation with Qiskit: 8x8 matrices, w/o noise

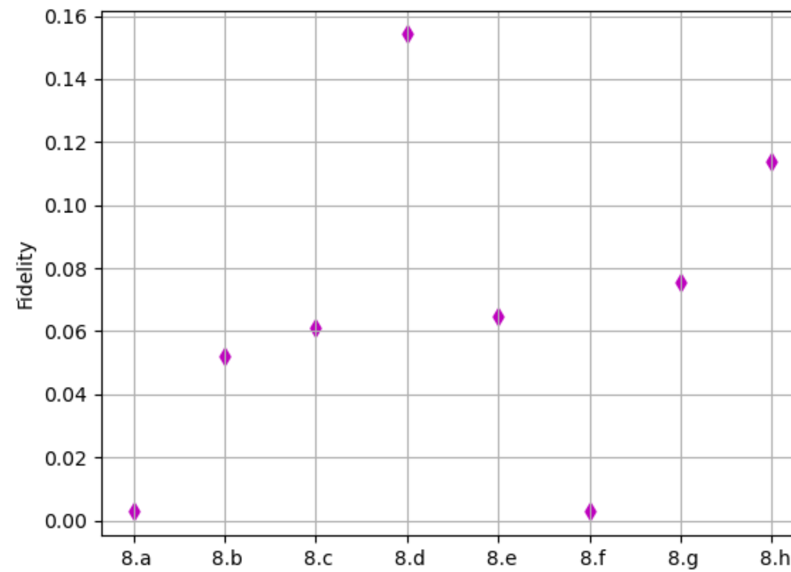




S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

HHL simulation with Qiskit: 8x8 matrices, with noise



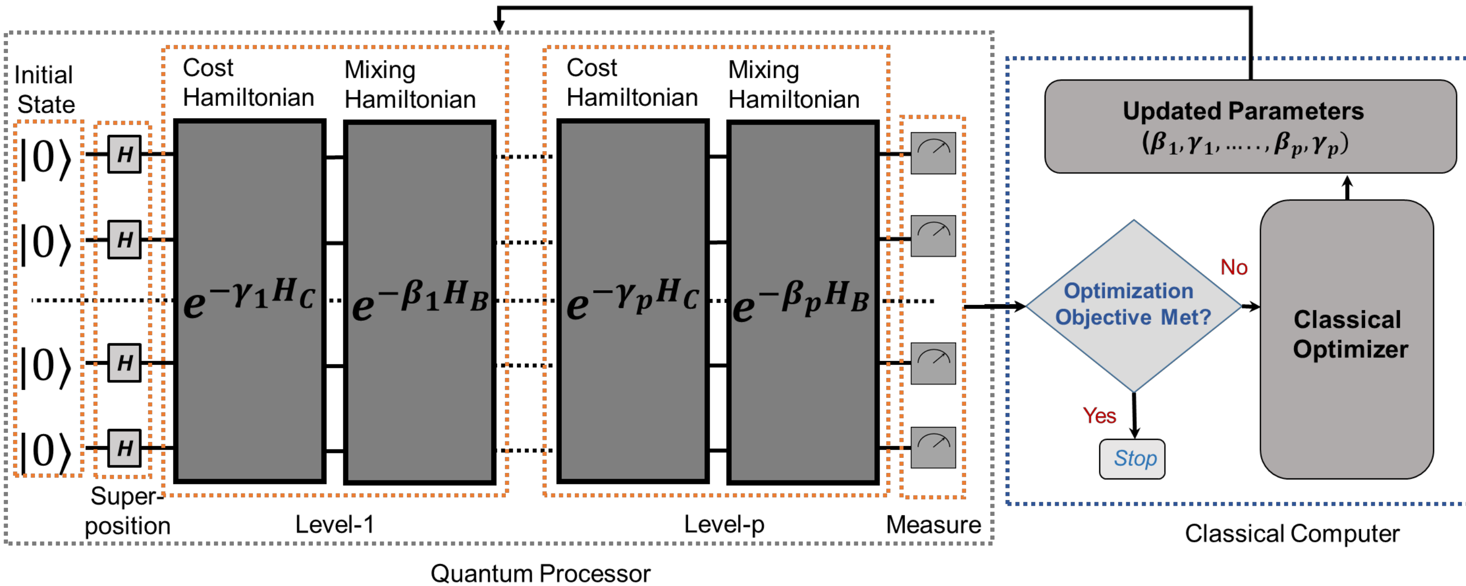


S. Sigurdsson: Implementations of quantum algorithms for solving linear systems, Master Thesis TU Delft, 2021.

potential *near-future* quantum applications in SciComp

- **hybrid quantum-classical algorithms**
  - quantum approximate optimization algorithm (QAOA) [Farhi et al. 2014]
  - quantum alternating operator ansatz (QAOA) [Hadfield et al. 2017]
  - variational quantum eigensolver (VQE) [Peruzzo et al. 2014]
  - variational quantum linear solver (VQLS) for sparse matrices [Bravo-Prieto et al. 2019 & Xu et al. 2019]

QAOA workflow

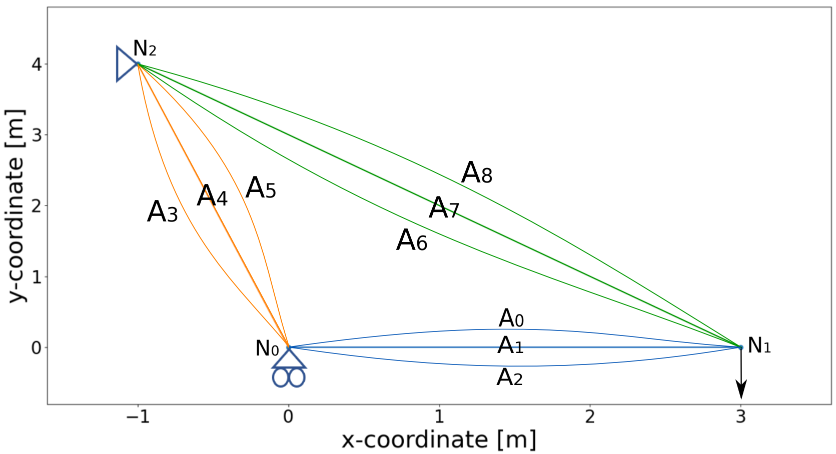


M. Alam, A. Ash-Saki, S. Ghosh: Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits. arXiv: 1907.09631 (2019)

truss structure optimization

3-truss structure

options: $2^{\#\text{trusses} \times \#\text{areas}} = 512$



valid options

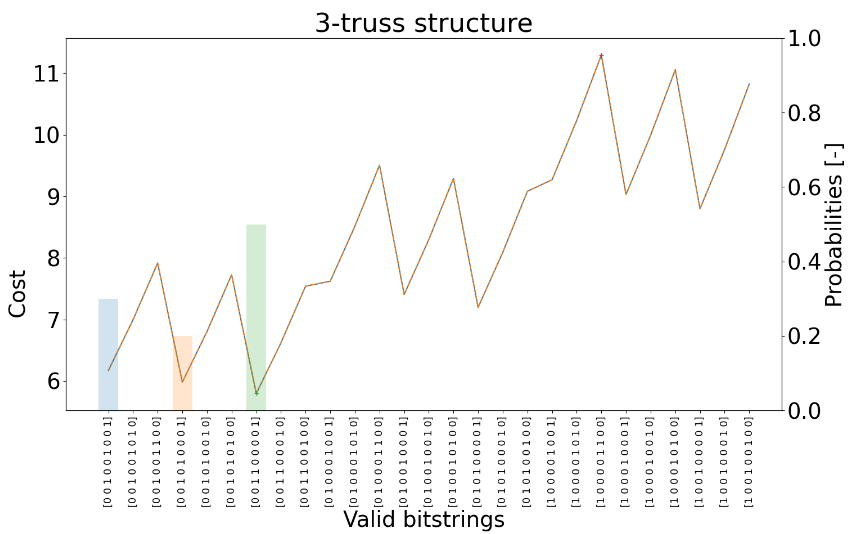
Option	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
1	0	0	1	0	0	1	0	0	1
2	0	0	1	0	0	1	0	1	0
3	0	0	1	0	0	1	1	0	0
4	0	0	1	0	1	0	0	0	1
5	0	0	1	0	1	0	0	1	0
6	0	0	1	0	1	0	1	0	0
7	0	0	1	1	0	0	0	0	1
8	0	0	1	1	0	0	0	1	0
9	0	0	1	1	0	0	1	0	0
10	0	1	0	0	0	1	0	0	1
11	0	1	0	0	0	1	0	1	0
12	0	1	0	0	0	1	1	0	0
13	0	1	0	0	1	0	0	0	1
14	0	1	0	0	1	0	0	1	0
15	0	1	0	0	1	0	1	0	0
16	0	1	0	1	0	0	0	0	1
17	0	1	0	1	0	0	0	1	0
18	0	1	0	1	0	0	1	0	0
19	1	0	0	0	0	1	0	0	1
20	1	0	0	0	0	1	0	1	0
21	1	0	0	0	0	1	1	0	0
22	1	0	0	0	1	0	0	0	1
23	1	0	0	0	1	0	0	1	0
24	1	0	0	0	1	0	1	0	0
25	1	0	0	1	0	0	0	0	1
26	1	0	0	1	0	0	0	1	0
27	1	0	0	1	0	0	1	0	0

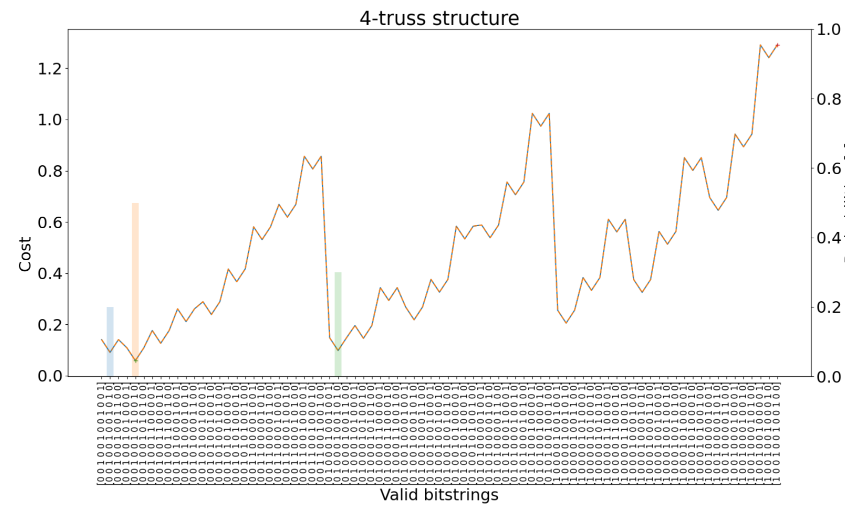
invalid options

485

J. De Zoete: A practical quantum algorithm for solving structural optimization problems. Master Thesis, TU Delft, 2021.

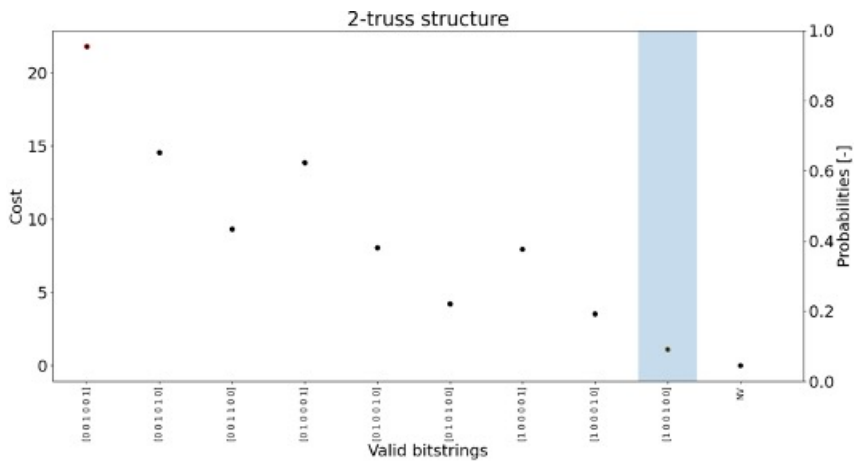
preliminary results using Rigetti's simulator

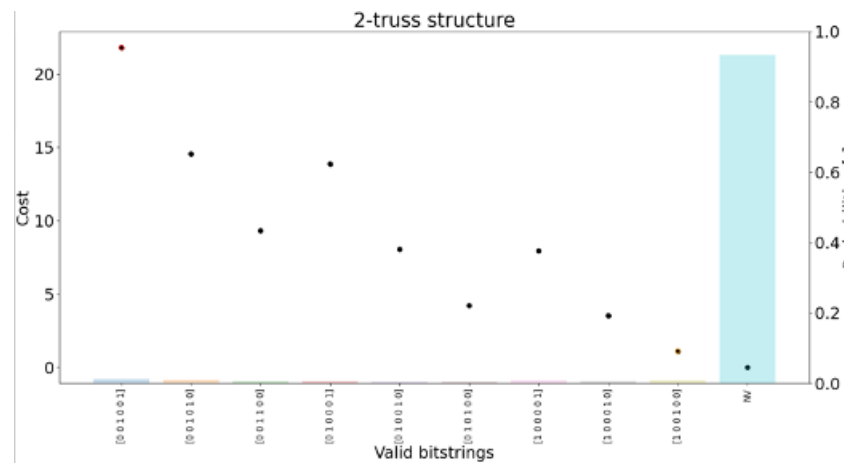




J. De Zoete: A practical quantum algorithm for solving structural optimization problems. Master Thesis, TU Delft, 2021.

preliminary results using Rigetti's Aspen-9 processor





only 9 out of 64 options are valid and the exclusion criterion is sensitive to noise

J. De Zoete: A practical quantum algorithm for solving structural optimization problems. Master Thesis, TU Delft, 2021.

summary

- QC is not just for physicists and electrical engineers but should interest the entire CSE community as a potential future accelerator technology
- building quantum computers is just the beginning, the time has come to develop practical algorithms and software for solving real-world problems
- early experience with quantum-accelerated applications will hopefully guide QC vendors in the development of practically usable devices for end-users

Thank you for your attention and enjoy your dinner!