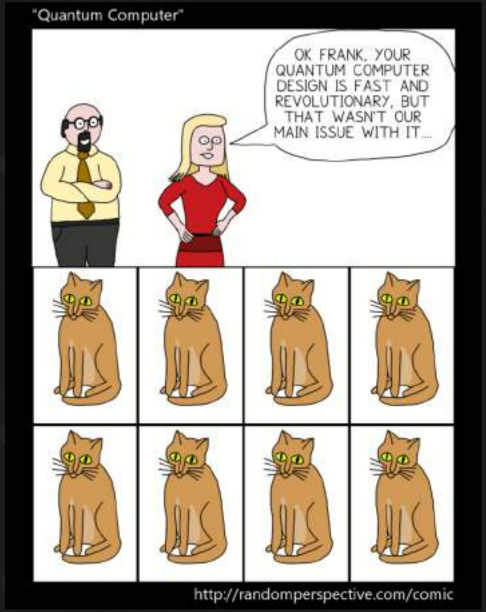
ProjectQ: Software for Quantum Computing

Damian Steiger
Work done at ETH Zurich
(now quantum researcher at Huawei)

What can we do with a Quantum Computer ?

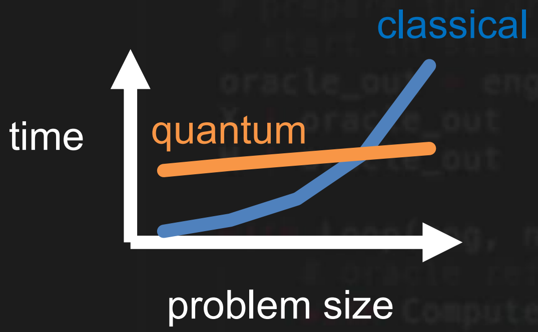


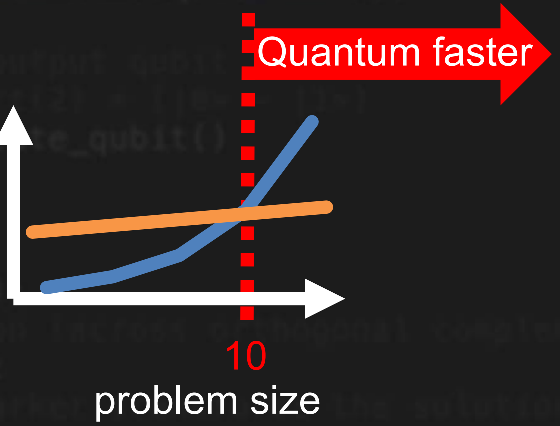
Finding a competitive application

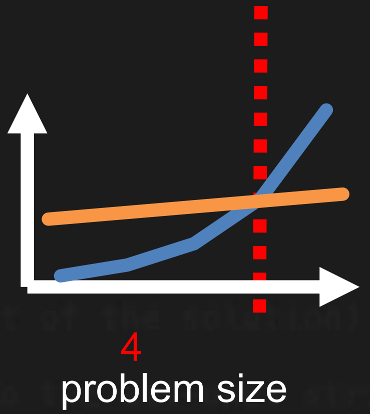
1 Find a quantum algorithm which scales better

2 Determine the exact cost

3 Improve the quantum algorithm



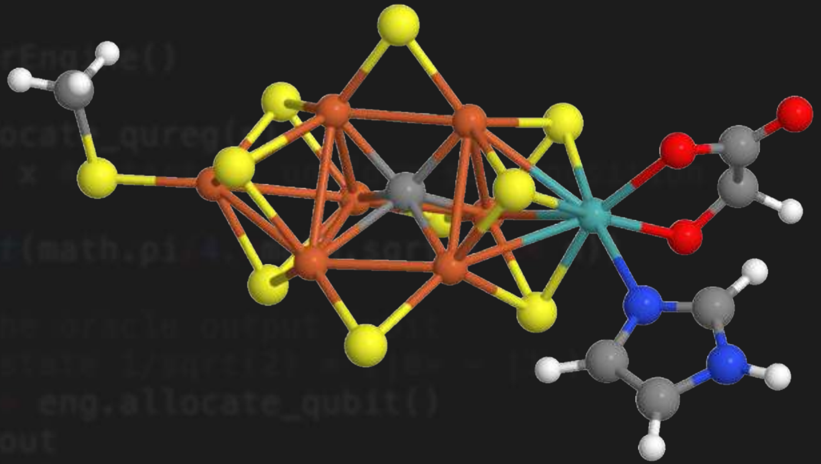




What can we do with a Quantum Computer ?

21 = 7 * 3

What else can we do with a quantum computer?



Quantum Simulation

ProjectQ (.ch)

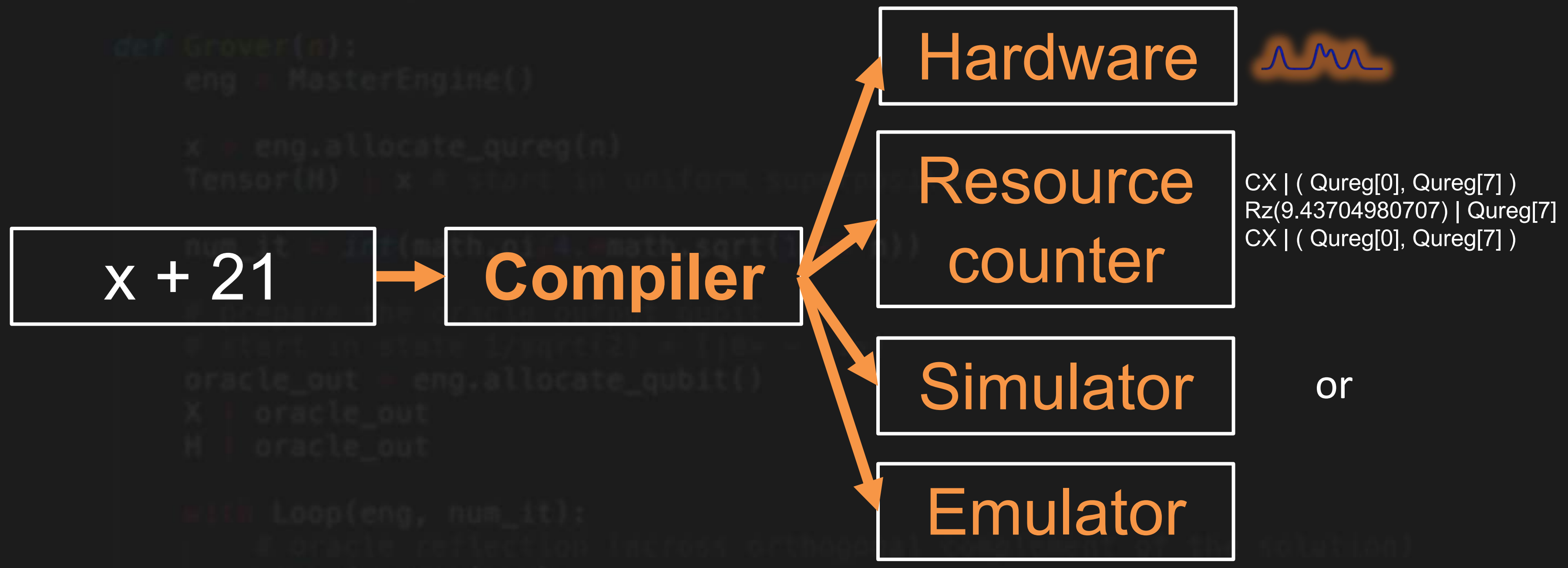
Develop standard tools for development of quantum algorithms, programming and controlling quantum computers

- **Open & Free:** *Apache 2 license*
- **Easy to Use:** *Simple learning curve by using common languages and intuitive syntax*
- **Extensible:** *Easily extensible to support new hardware or gate sets*

How to compute x+21?

```
CX | ( Qureg[4], Qureg[7] )
Rz(9.62112750162) | Qureg[7]
CX | ( Qureg[5], Qureg[6] )
CX | ( Qureg[4], Qureg[7] )
Rz(2.74889357189) | Qureg[7]
R(2.74889357189) | Qureg[5]
CX | ( Qureg[5], Qureg[7] )
Rz(9.81747704247) | Qureg[7]
H | Qureg[6]
CX | ( Qureg[5], Qureg[7] )
Rz(2.35619449019) | Qureg[7]
R(2.35619449019) | Qureg[6]
CX | ( Qureg[6], Qureg[7] )
Rz(10.2101761242) | Qureg[7]
CX | ( Qureg[6], Qureg[7] )
H | Qureg[7]
```

Unified framework



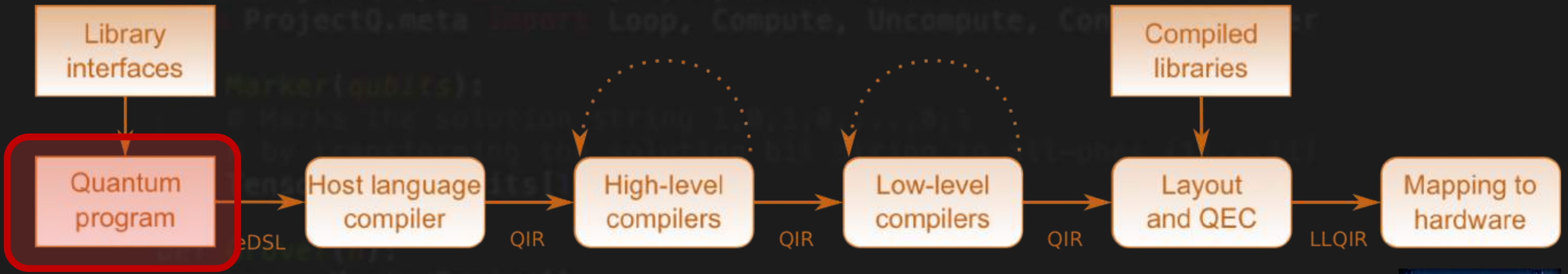
ProjectQ Syntax

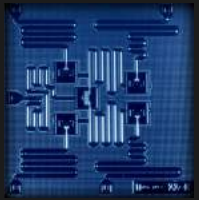
```
eng = projectq.MainEngine(backend)

qubit = eng.allocate_qubit()
qureg = eng.allocate_qureg(10)

U(classical_param) | qureg
```

$$U(theta)\,|\text{qureg}\rangle$$

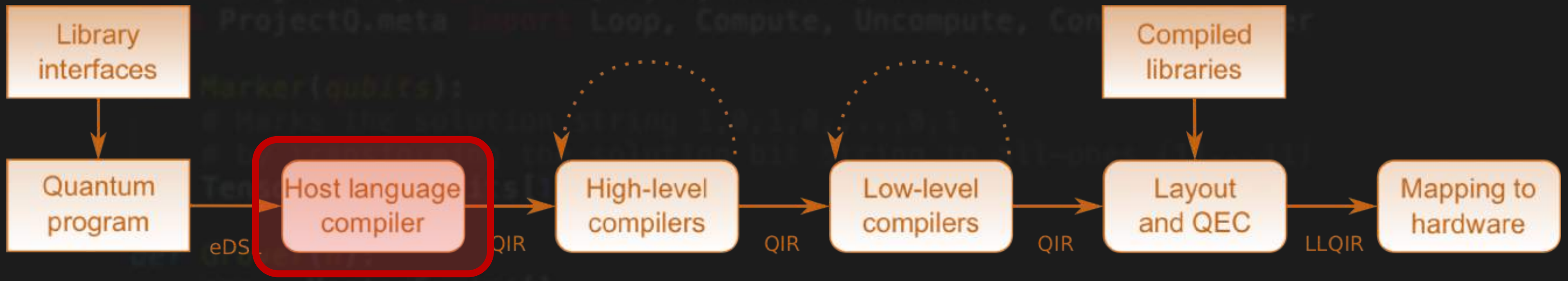
ProjectQ Example Operations

```
Rx(0.1) | qubit
H | qubit
CNOT | (qubit, qureg[0])
Measure | qubit

QFT | qureg
MultiplyByConstantModN(10, N=21) | qureg
```

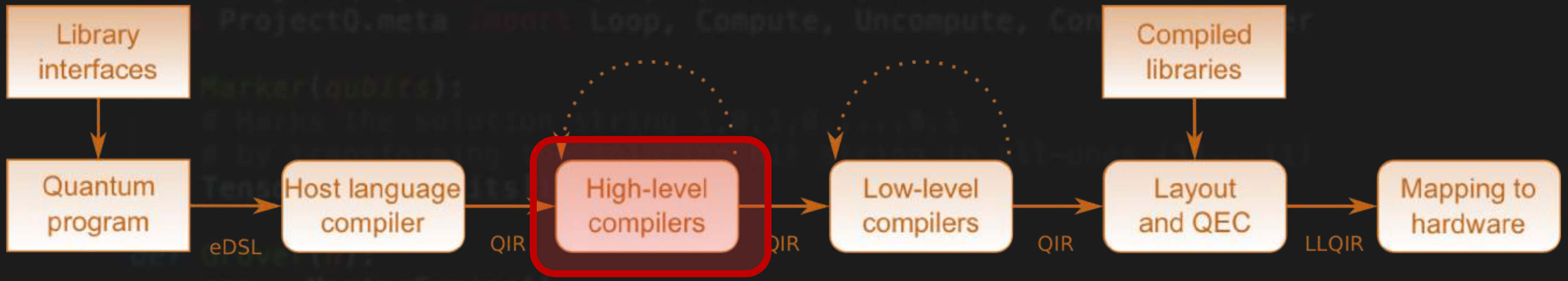


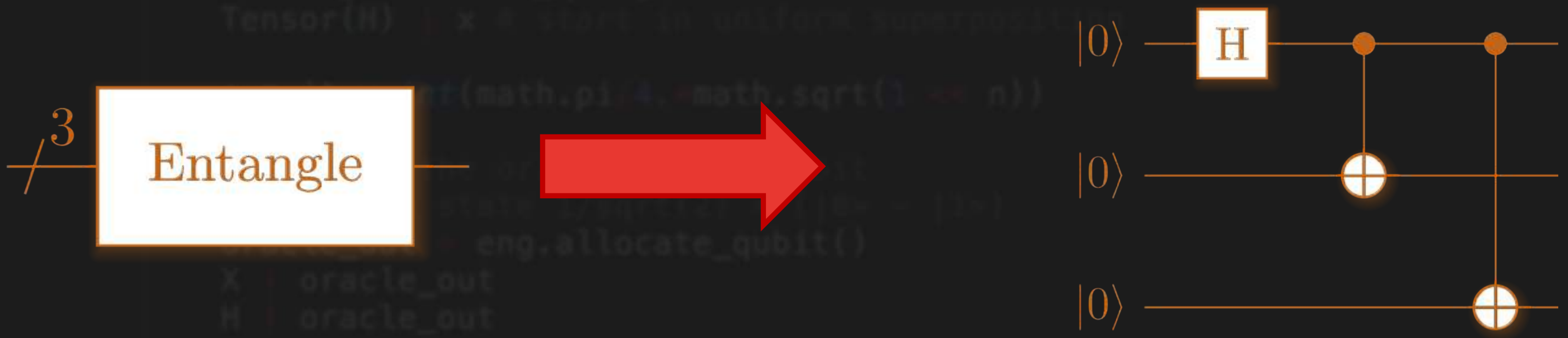


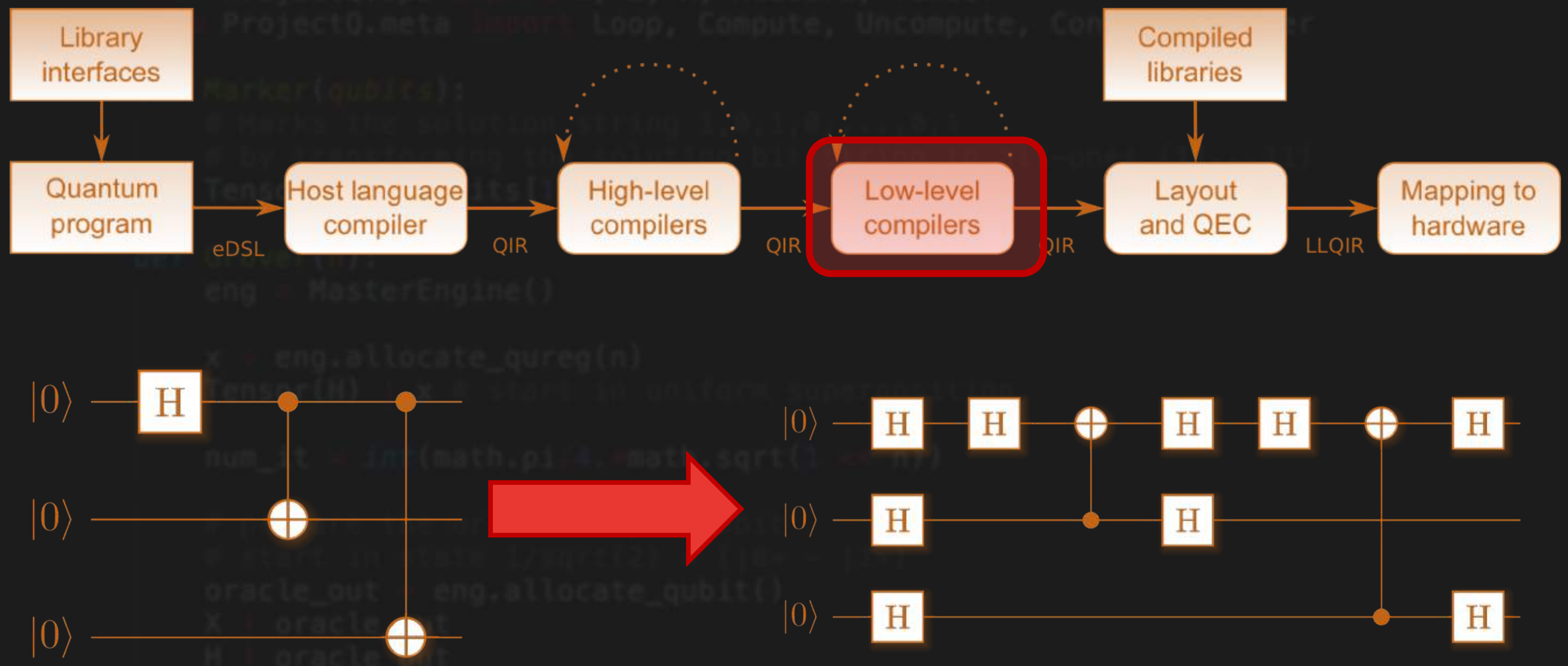
```
qureg = eng.allocate_qureg(3)
# entangle the qureg
Entangle | qureg
# measure; should be all-0 or all-1
Measure | qureg
```

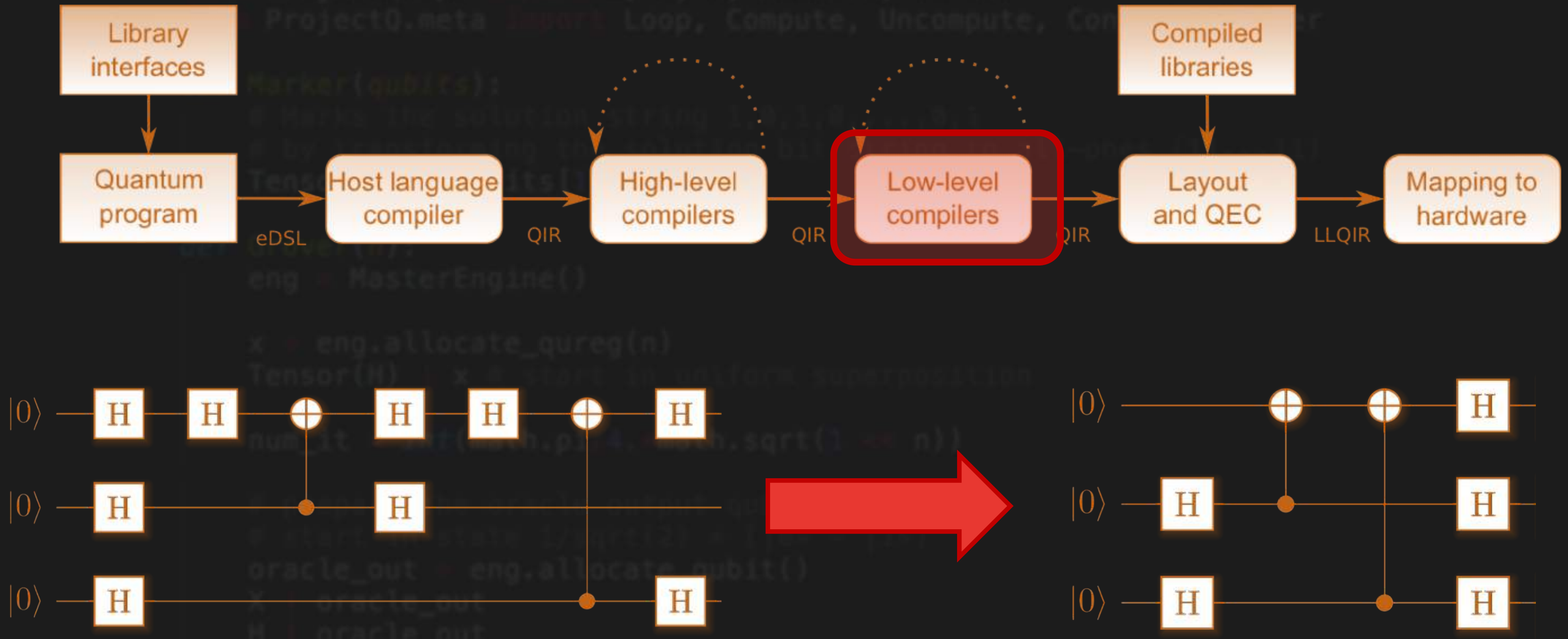


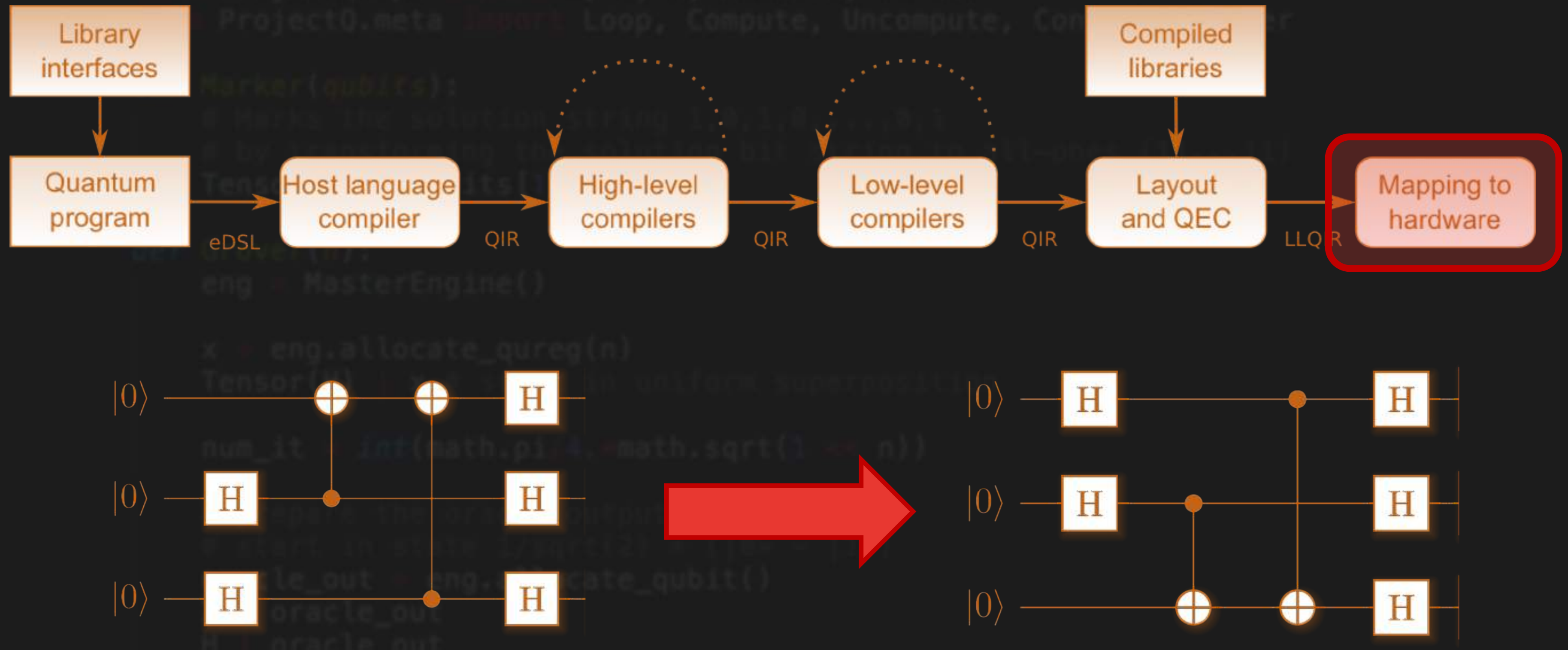

```
# entangle the qureg
Entangle | qureg
```


Library interfaces
Compiled libraries
Quantum program
eDSL
Host language compiler
QIR
High-level compilers
QIR
Low-level compilers
QIR
Layout and QEC
LLQIR
Mapping to hardware


3
Entangle
|0⟩
|0⟩
|0⟩
H


Library interfaces
Quantum program
eDSL
Host language compiler
QIR
High-level compilers
QIR
Low-level compilers
QIR
Compiled libraries
Layout and QEC
LLQIR
Mapping to hardware
|0⟩
|0⟩
|0⟩
H
|0⟩
|0⟩
|0⟩
H H H H H H H H H


Library interfaces
Quantum program
eDSL
Host language compiler
QIR
High-level compilers
QIR
Low-level compilers
QIR
Compiled libraries
Layout and QEC
LLQIR
Mapping to hardware


Library interfaces
Quantum program
eDSL
Host language compiler
QIR
High-level compilers
QIR
Low-level compilers
QIR
Compiled libraries
Layout and QEC
LLQIR
Mapping to hardware
|0⟩
|0⟩
|0⟩
H
H
H
H
H
|0⟩
|0⟩
|0⟩
H
H
H
H
H
H

Resource estimates

Shor's algorithm

```
x = eng.allocate_qureg(n)
X | x[0]
ctrl_qubit = eng.allocate_qubit()

for k in range(2 * n):
    current_a = pow(a, 1 << (2 * n - 1 - k), N)
    # one iteration of 1-qubit QPE
    H | ctrl_qubit
    with Control(eng, ctrl_qubit):
        MultiplyByConstantModN(current_a, N) | x

    # perform inverse QFT --> Rotations conditioned on previous outcomes
    for i in range(k):
        if measurements[i]:
            R(-math.pi/(1 << (k - i))) | ctrl_qubit
    H | ctrl_qubit

    # and measure
    Measure | ctrl_qubit
    eng.flush()
    measurements[k] = int(ctrl_qubit)
    if measurements[k]:
        X | ctrl_qubit
```

Beauregard's implementation

- Factoring a number with *n bits*
- 2n + 3 qubits
- $O(n^4)$ gates
- $O(n^3)$ circuit depth

Resource for factoring 15

- 11 qubits
- Circuit depth of 28477
- Number of gates
  - CNOT: 17832
  - Rz: 13624
  - T and inverse T: 11312
  - H: 4576
  - X: 129

Simulator backend

- Calibration and benchmarking of quantum devices
- Debugging quantum algorithms

Quantum Simulator

Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch

Quantum Simulator

Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch
20	16 MByte	milliseconds on a smartphone

Quantum Simulator

Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch
20	16 MByte	milliseconds on a smartphone
30	16 GByte	seconds on a laptop

Quantum Simulator

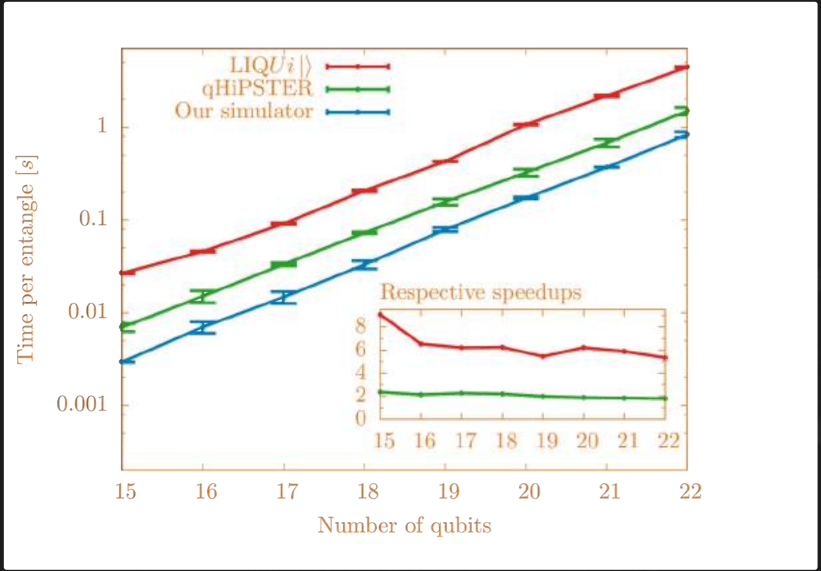
Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch
20	16 MByte	milliseconds on a smartphone
30	16 GByte	seconds on a laptop
40	16 TByte	minutes on a supercomputer

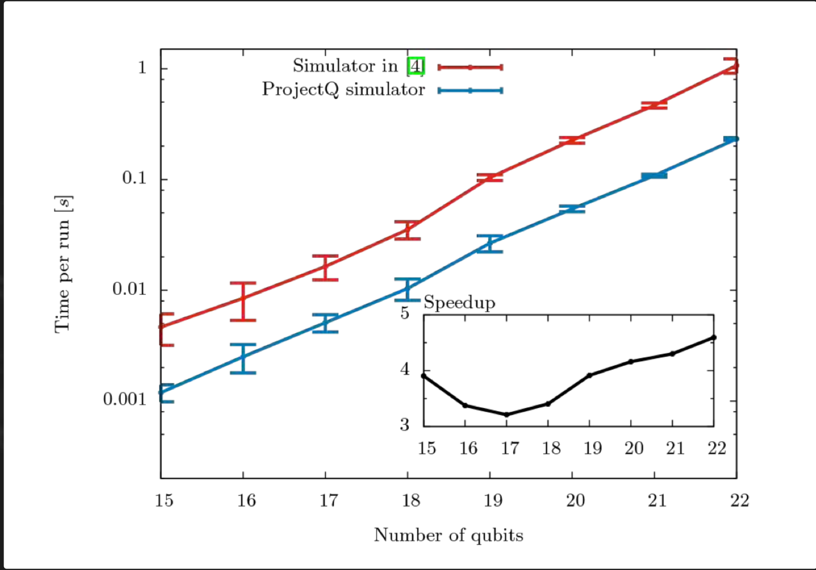
Quantum Simulator

Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch
20	16 MByte	milliseconds on a smartphone
30	16 GByte	seconds on a laptop
40	16 TByte	minutes on a supercomputer
260	each particle of visible universe	age of universe

T. Häner, D. Steiger, M. Troyer

Simulator performance





Our simulator performance of spring 2016

ProjectQ simulator (winter 2016) vs our own simulator of spring 2016

T. Häner, D. Steiger, M. Troyer

Another simulator of ours

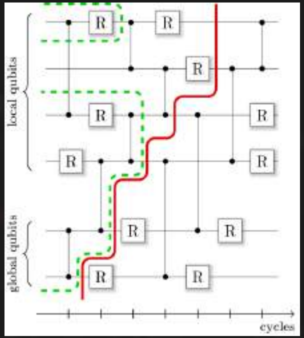
0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit

Thomas Häner*, Damian S. Steiger*
**Institute for Theoretical Physics, ETH Zurich, 8093 Zurich, Switzerland*

45-Qubit Simulation

The worlds largest quantum computer simulation at that time

Cori Supercomputer
0.5 Petabyte of memory
2.2 Million logical cores



>10x speedup
45 Qubits
(previously 42)

In collaboration with
Thomas Häner

Supercomputing's monster in the closet, IEEE

ETH Zurich

T. Häner and DS, SC’17

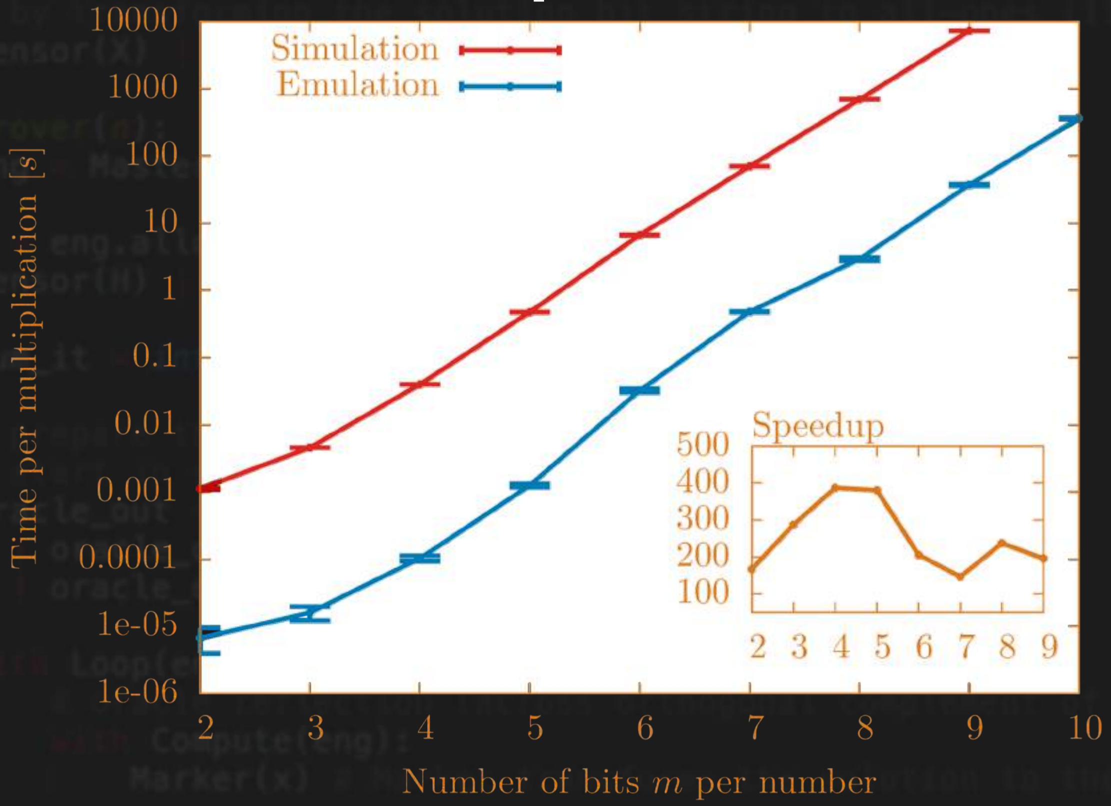
T. Häner, D. Steiger, M. Troyer

High Performance Emulation of Quantum Circuits

Thomas Häner*, Damian S. Steiger*, Mikhail Smelyanskiy†, and Matthias Troyer*‡§
*Institute for Theoretical Physics, ETH Zurich, 8093 Zurich, Switzerland
† Parallel Computing Lab, Intel Corporation
‡ Quantum Architectures and Computation Group, Microsoft Research, Redmond, WA (USA)
§ Corresponding Author: troyer@phys.ethz.ch

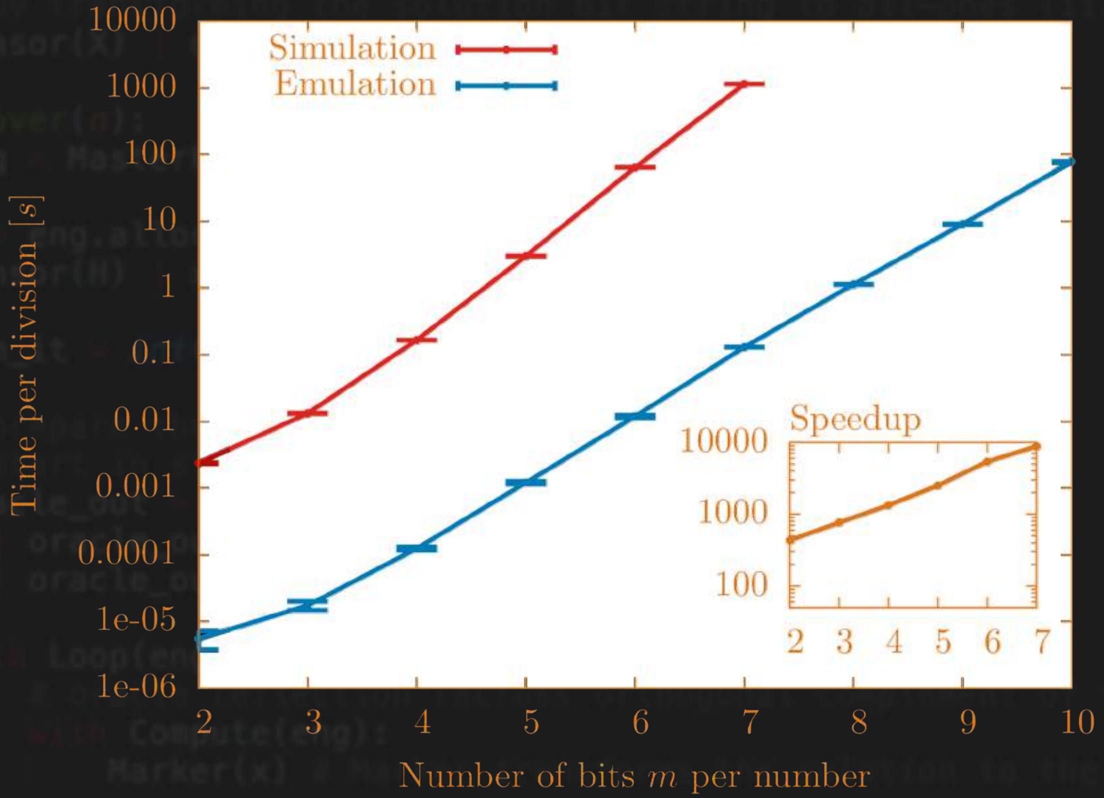
T. Häner, D. Steiger, M. Troyer

Multiplication



T. Häner, D. Steiger, M. Troyer

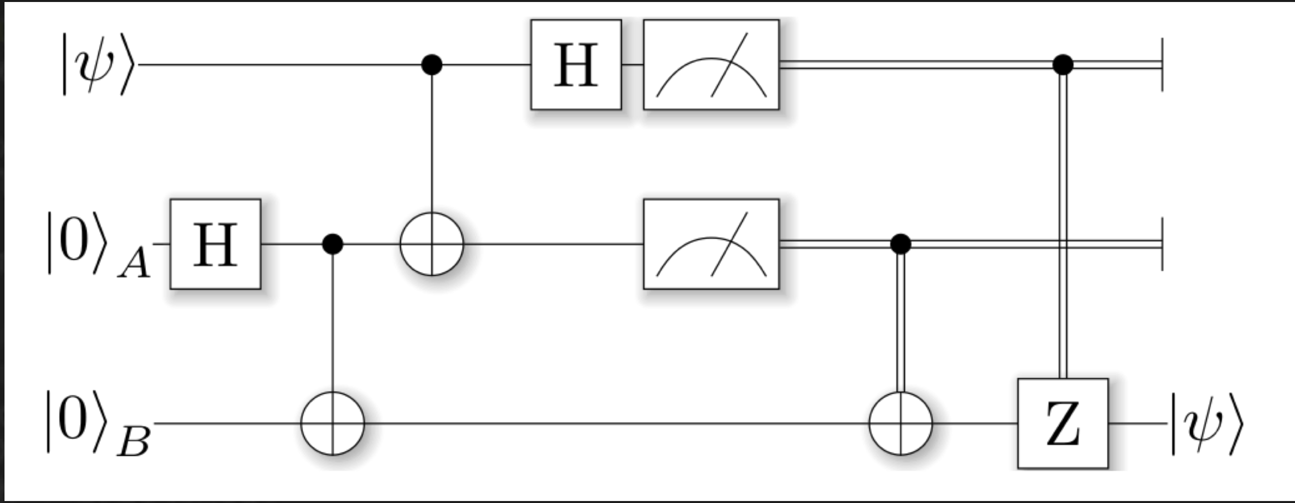
Division



Advantages of emulation

- 10x for QFT
- 400x for multiplication
- 10000x for division
- much more for more complex functions (exp, sqrt, arcsin, ...)

Latex Drawer backend



RevKit Library

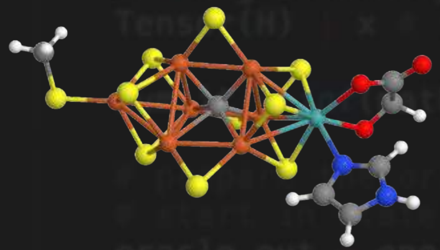
```
def f(a, b, c, d):
    return (a and b) ^ (c and d)

qubits = eng.allocate_qureg(4)
PhaseOracle(f) | qubits  # (-1)**f(qubits)
```

Programming Quantum Computers Using Design Automation ***arXiv:1803.01022***

By Mathias Soeken, Thomas Häner, Martin Roetteler

Quantum Simulation Library

- FermiLib



$$E_0\,,\ |\phi_0\rangle\ ,\ \ldots$$

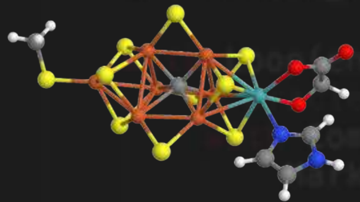
In collaboration with Ryan Babbush, Jarrod McClean, Ian Kivlichan

FermiLib

- Interface to electronic structure packages

Input:

```
geometry = [('H', (0., 0., 0.)),
            ('H', (0., 0., 0.7))]
basis = 'sto-3g'
multiplicity = 1
charge = 0
molecule = MolecularData(geometry,
                         basis,
                         multiplicity,
                         charge)
```



Output:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

Optional classical results:

- Full CI energy
- MP2 energy
- CCSD energy
- …

FermiLib

- Transform fermionic to spin operators:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \quad \Rightarrow \quad H = \sum_{j=0}^{N} \alpha_j P_j$$

Choose between:

Jordan-Wigner transform

Bravyi-Kitaev transform

...

FermiLib - ProjectQ

FermiLib provides: $$H = \sum_{j=0}^{N} \alpha_j P_j$$

Use one of the quantum algorithms in ProjectQ to find static properties using methods such as:

"Quantum Algorithm for Spectral Measurement with a Lower gate Count" *by D. Poulin, A. Kitaev, DS, M. Hastings, M. Troyer.* PRL 121, 010501 (2018)

Collaborators

Thomas Häner

Matthias Troyer

Acknowledgments

We would like to thank

Lev S. Bishop, Fran Cabrera, Jorge Carballo, Jerry M. Chow, Andrew W. Cross, Jana Darulová, Michele Dolfi, Ismael Faro, Jay M. Gambetta, Alan Geller, Dominik Gresch, Jonathan Home, Mario Könz, Matteo Marinelli, Paco Martin, Anne Matsuura, Jarrod McClean, Vlad Negnevitsky, Ryan Babbush, Hartmut Neven, Natalie Pearson, Mark Ritter, Donjan Rodic, Daniel Sank, Slava Savenko, Mikhail Smelyanskiy, John Smolin, Krysta Svore, Dave Wecker, and Camillo Zapata Ocampo.

Thanks for your attention!

www.projectq.ch