Design Automation and Software Tools for Quantum Computing

Robert Wille
Technical University of Munich
Software Competence Center Hagenberg GmbH
robert.wille@tum.de
https://iic.jku.at/eda/research/quantum/

Connect on LinkedIn

Follow @rbrtwll

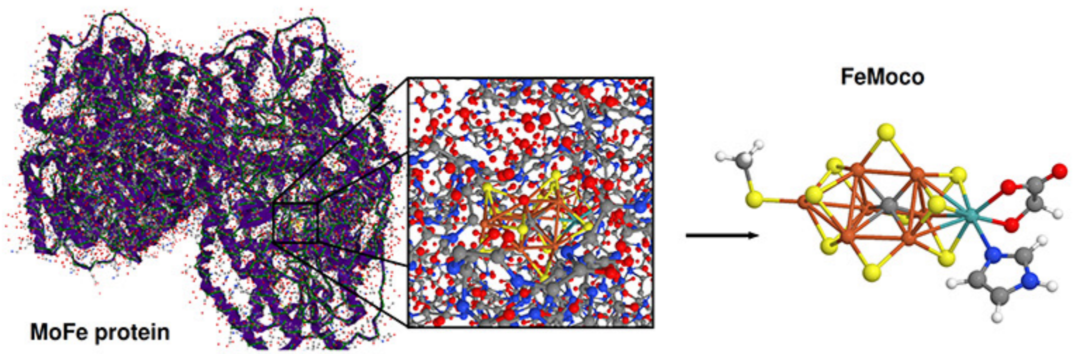
scch { }
software competence center hagenberg

erc
European Research Council
Established by the European Commission

Quantum Computing: The next big Thing

- Global Players are heavily investing
  - IBM, Google, Microsoft, Amazon
  - Startups: Rigetti, IonQ, …
  - **Exponential improvements** in the best case

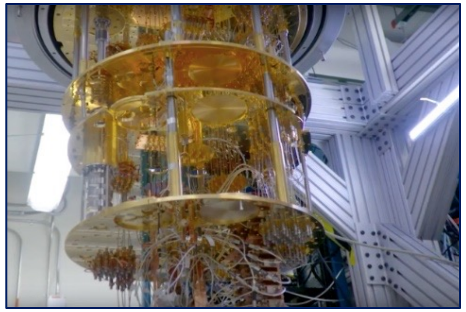
- Killer Application: Haber-Bosch Process
  - 1-2% of world‘s energy consumption
  - 3-5% of world‘s gas production ($11 Billion)
  - Bacteria does similar with fewer energy
  - Quantum computers simulate underlying process
- Integer factoring in polynomial time
  - Break current internet cryptography
- Machine learning, physics simulation, unstructured search, …

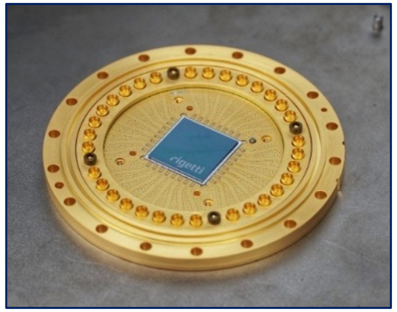


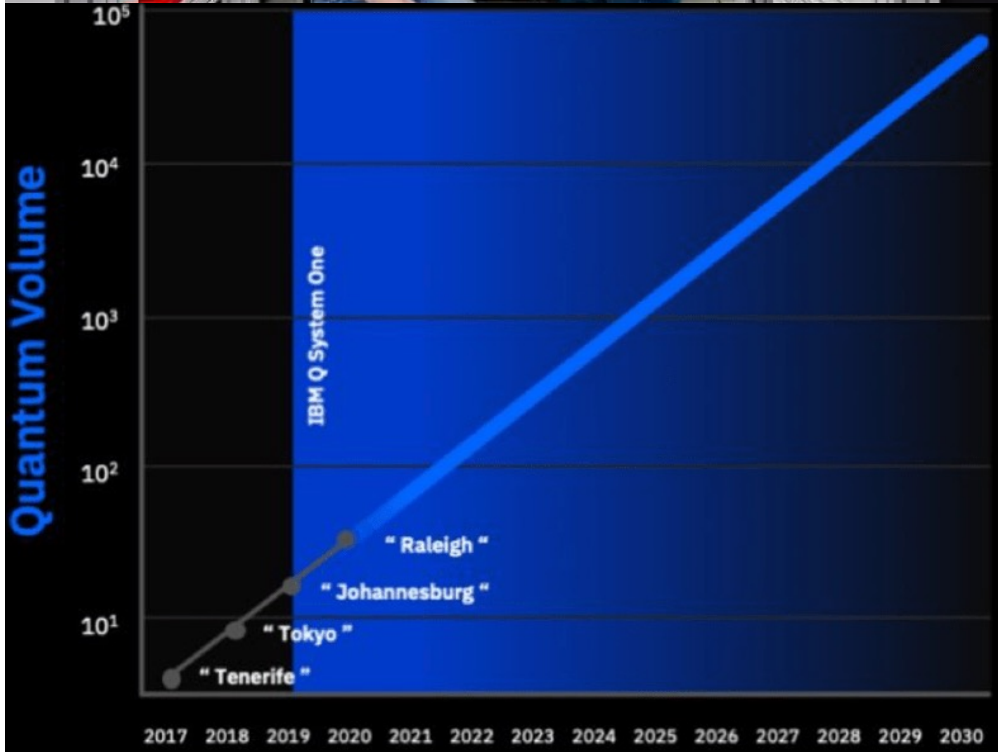
Quantum Hardware

- Significant progress in recent years
  - Publicly available quantum computers by IBM (2017)
  - Google claims quantum supremacy (2019)
- IBM Roadmap:
  - 2020: 65 qubits
  - 2021: 127 qubits
  - 2022: 433 qubits
  - 2023: 1.127 qubits
  - Beyond: Million of qubits

IBM Google rigetti



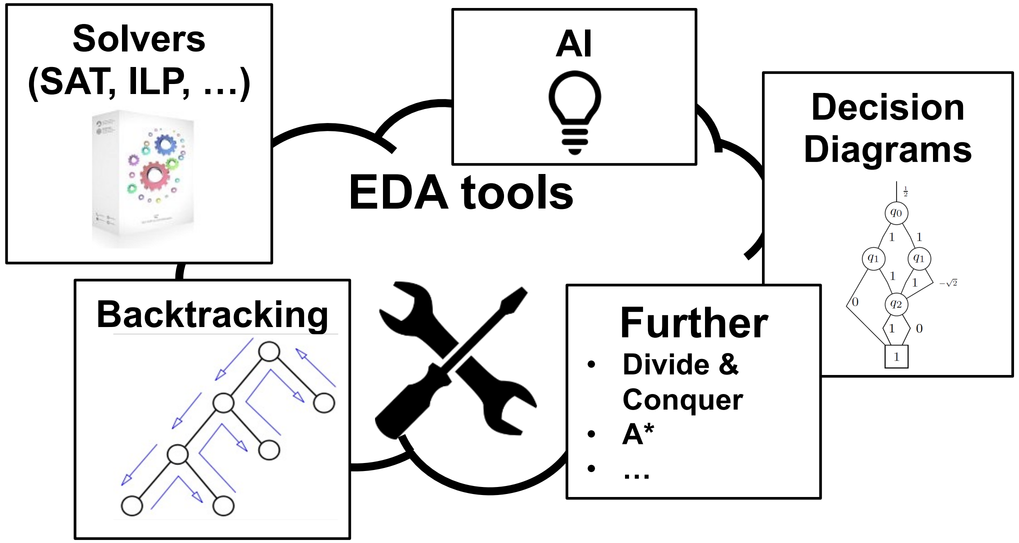


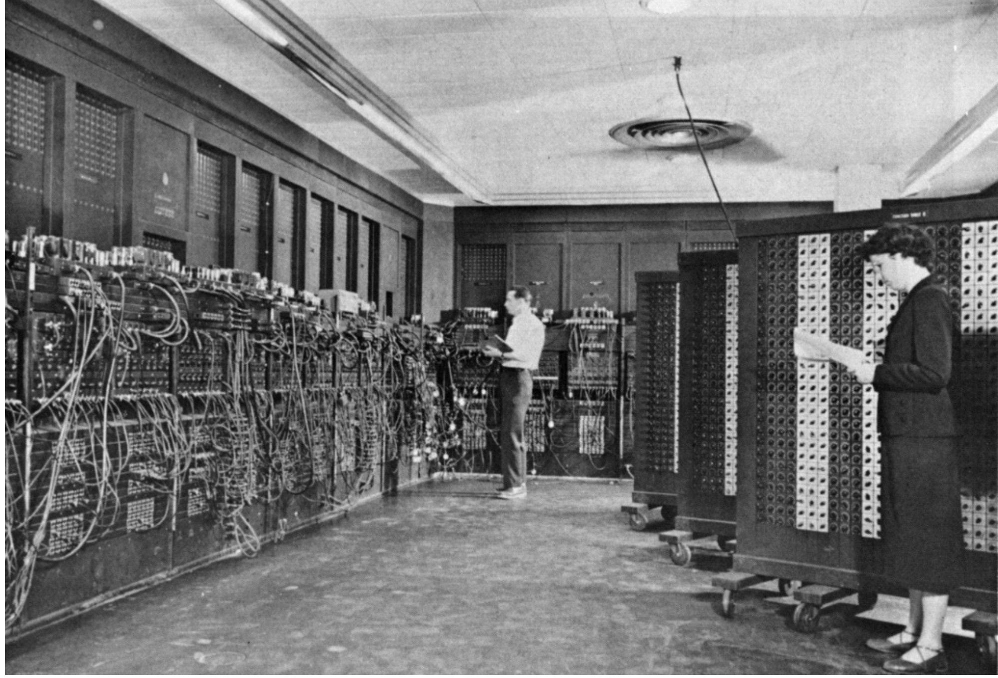


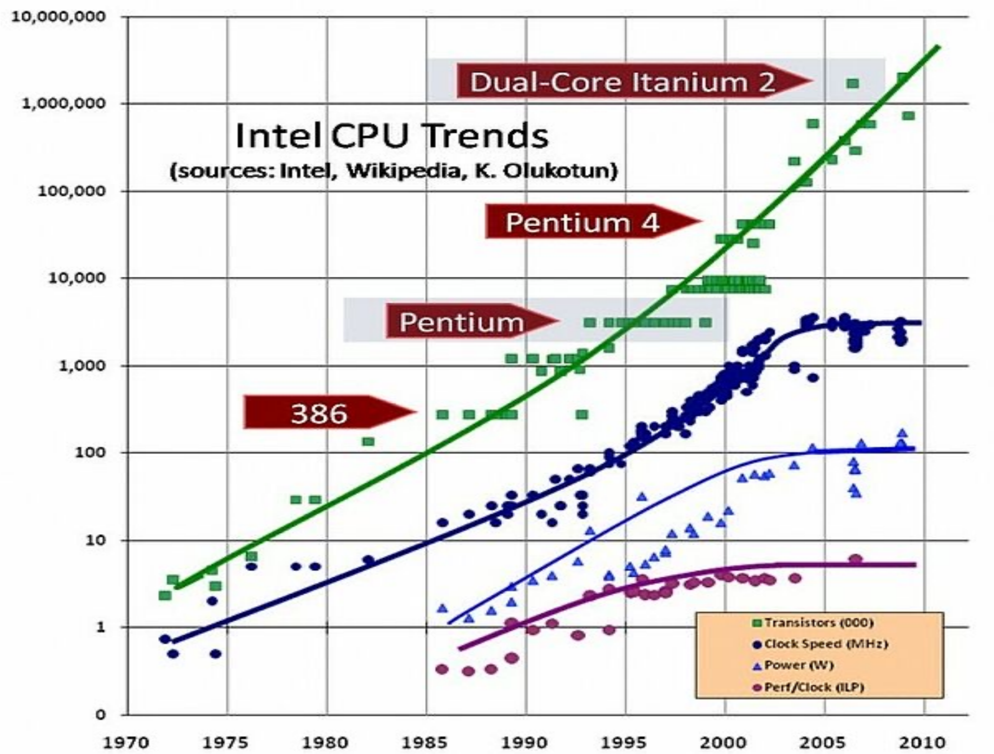
Analogy to Conventional Computers

- Similar picture if we look back in time
  - First, bulky computers
  - Moores law
  - Digital revolution

- **EDA** tools enabled their success
  - **Handle** problems with **enormous complexity**



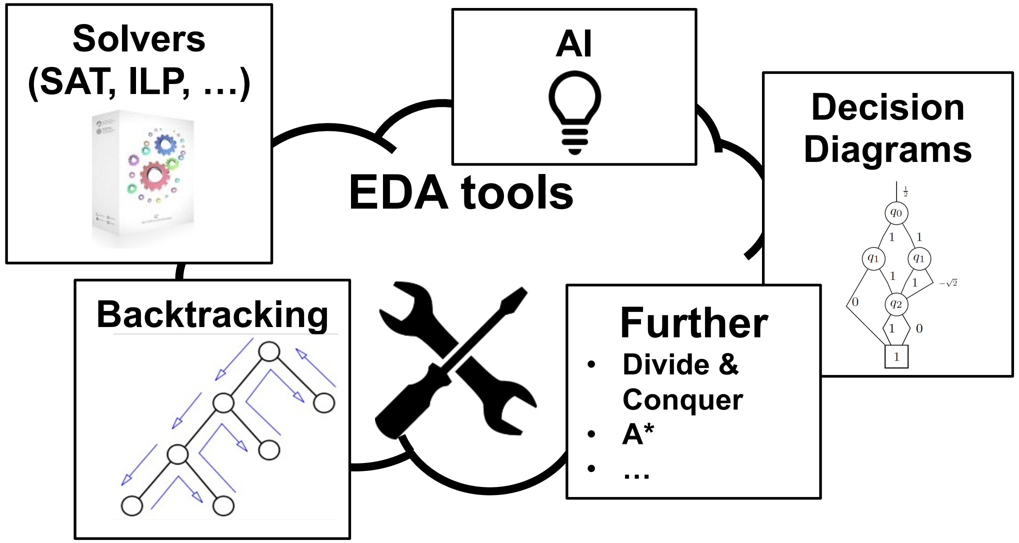




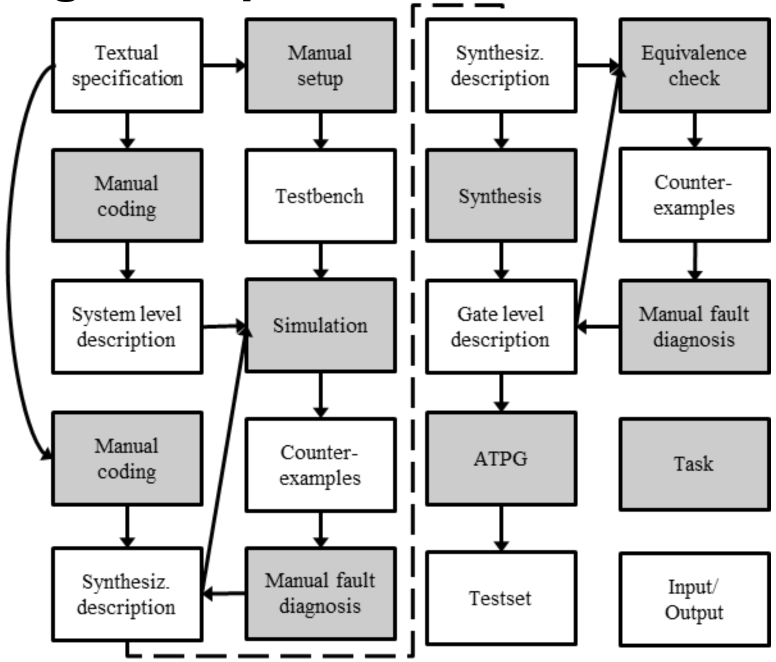
Analogy to Conventional Computers

- Similar picture if we look back in time
  - First, bulky computers
  - Moores law
  - Digital revolution

- **EDA** tools enabled their success
  - **Handle** problems with **enormous complexity**



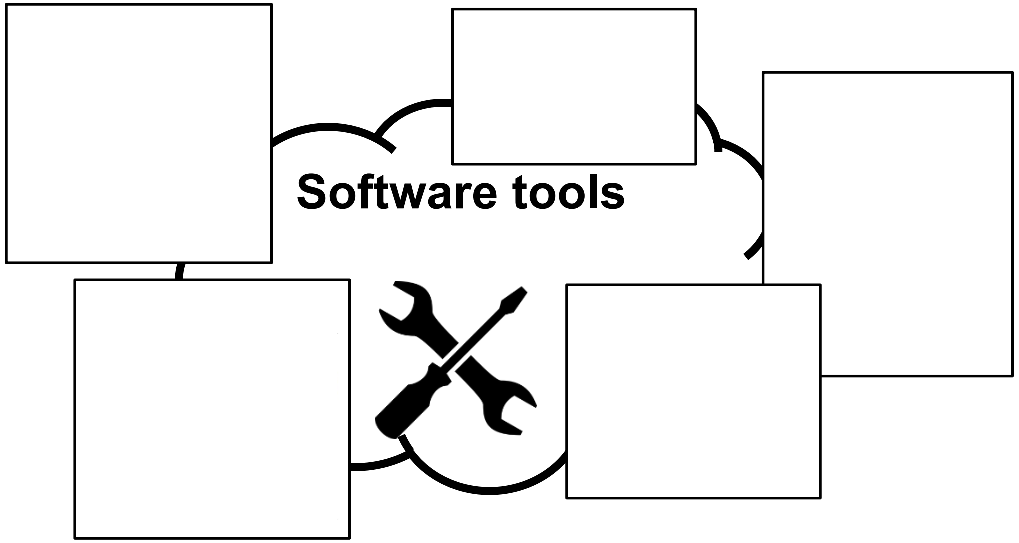
- **Emerge of powerful design/compilation flows**



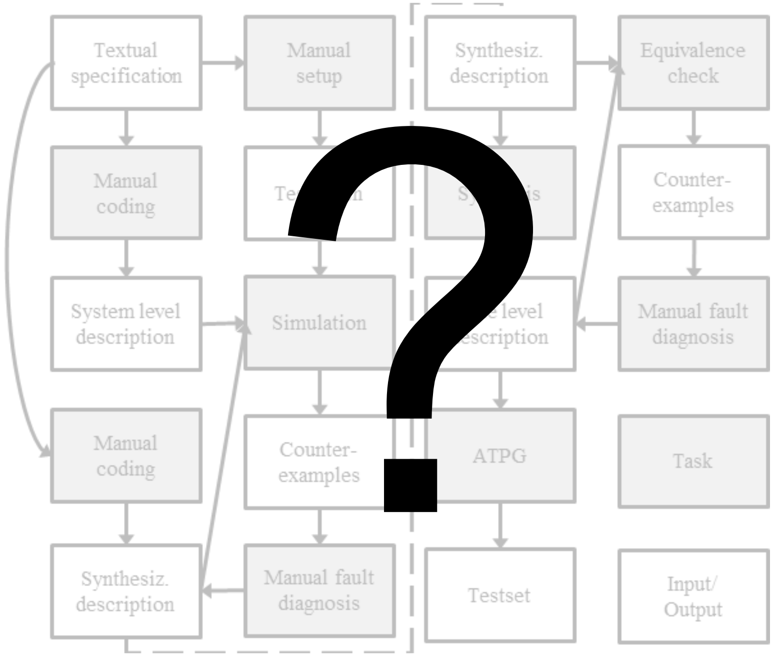
EDA for Quantum Computers

We may end up in a situation where we have quantum computers but no efficient methods to use or design them

- **Software** tools



- **Emerge of powerful design/compilation flows**



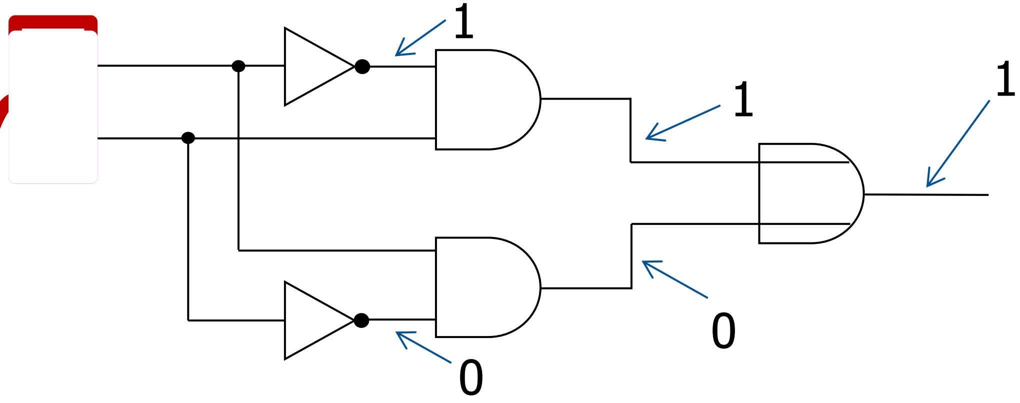
Challenges (Examples)

- “Programming” (different paradigms)
- Every quantum operation is inherently reversible

```
        a=4 b=5 c=0
c+=a*b
        a=4 b=5 c=20
```

- Not to mention
  - Dealing with “0 and 1 at the same time”
  - Understanding/Handling quantum operations
  - Physical constraints
  - Error correction
  - …

- Simulation



$$|\psi\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix}$$

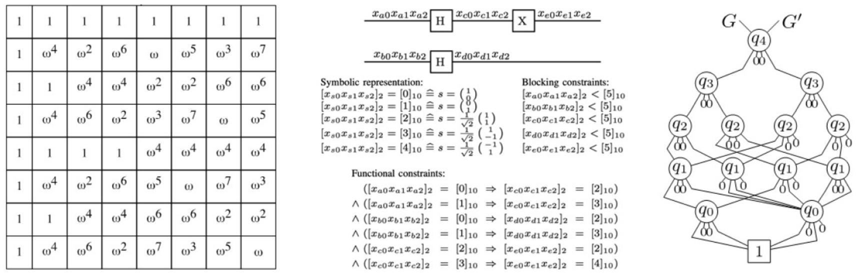
Exponential complexity

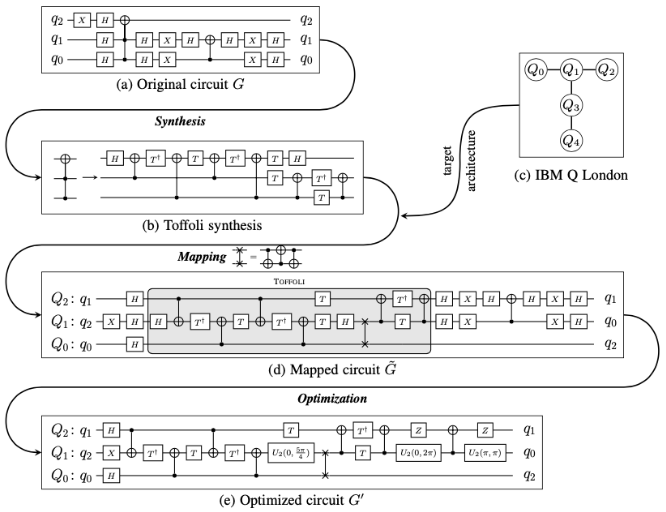
Already simple tasks are substantially harder for quantum computing than for conventional circuits/systems

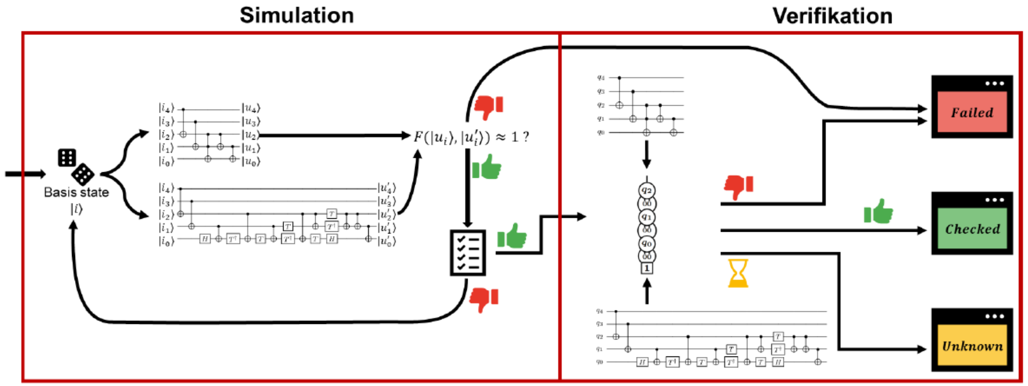
- Further:
  - Interdisciplinarity
  - Terminology and Formalizations
  - …

Focus of This Talk

- **Development of (automatic) methods and tools for the design of quantum algorithms, quantum circuits, and quantum systems**
  - **Core Methods and Data-structures**
  - **Compilation/Synthesis** quantum functionality to quantum devices (e.g. IBM QX architectures, Ion-Traps)
  - **Simulation and verification** of quantum computations
  - …





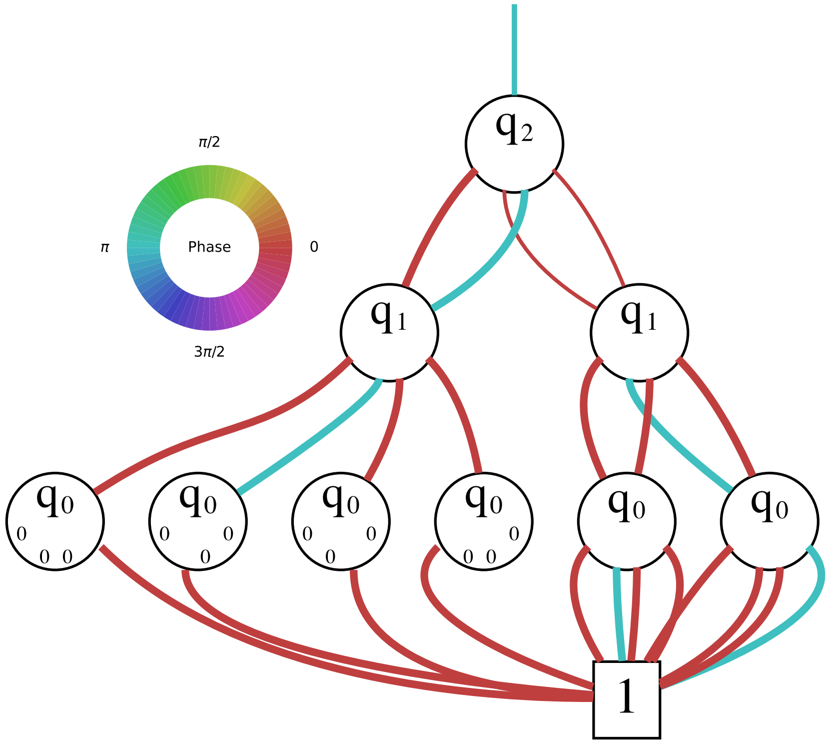


Data-structures #1

Functionality

$$-\frac{1}{\sqrt{2}}\left(\begin{array}{cc:cc|cc:cc}
0 & 0 & 0 & -1 & 1/2 & -1/2 & -1/2 & -1/2 \\
0 & 1 & 0 & 0 & 1/2 & 1/2 & -1/2 & 1/2 \\
\hdashline
0 & 0 & 1 & 0 & 1/2 & -1/2 & 1/2 & 1/2 \\
1 & 0 & 0 & 0 & 1/2 & 1/2 & 1/2 & -1/2 \\
\hline
0 & 0 & 0 & 1 & 1/2 & -1/2 & -1/2 & -1/2 \\
0 & -1 & 0 & 0 & 1/2 & 1/2 & -1/2 & 1/2 \\
\hdashline
0 & 0 & -1 & 0 & 1/2 & -1/2 & 1/2 & 1/2 \\
-1 & 0 & 0 & 0 & 1/2 & 1/2 & 1/2 & -1/2
\end{array}\right)$$

Decision Diagram



Eventually allows for compact representation and efficient manipulation in many cases

Simulation #1

- Matrix vector multiplication:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{CNOT} \times \underbrace{\begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix}}_{Input} = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \boldsymbol{\alpha_{11}} \\ \boldsymbol{\alpha_{10}} \end{bmatrix}$$

- Example

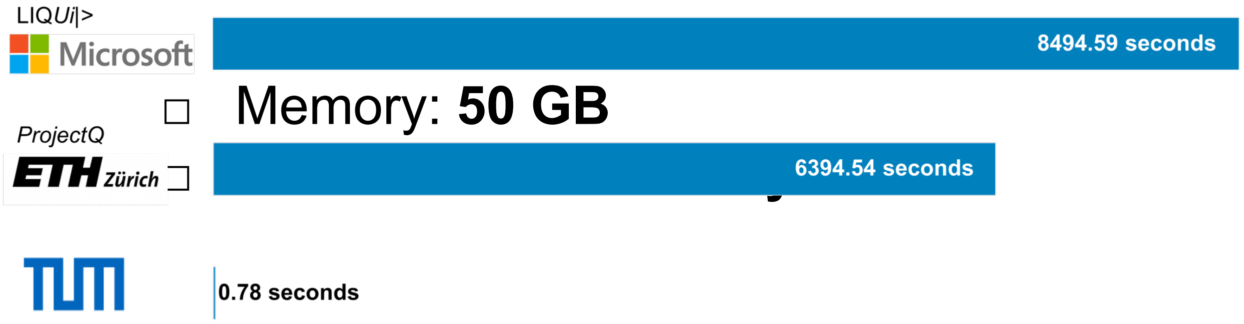
$$H|1\rangle$$

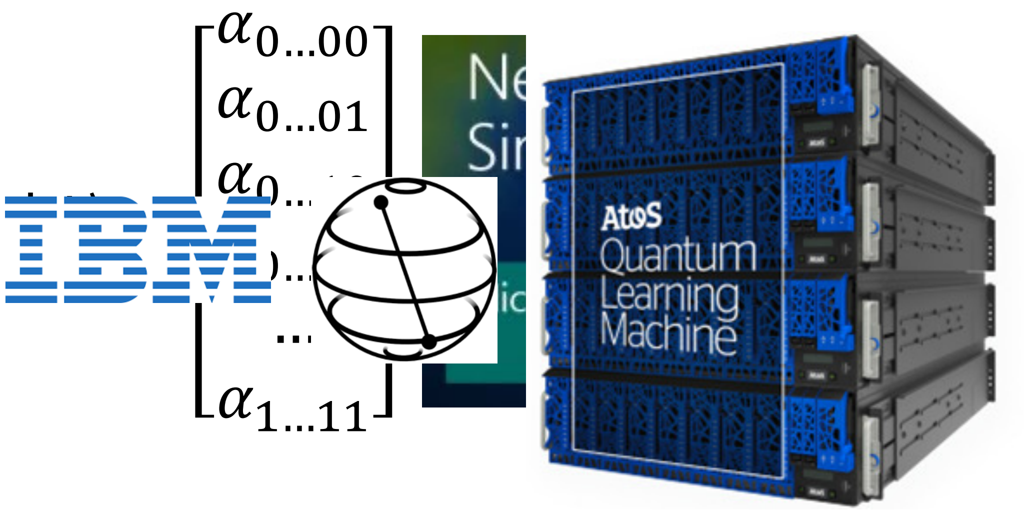
- Matrices and state vectors grow exponentially with respect to the number of qubits

➔ **Efficient representation and manipulation of quantum systems**

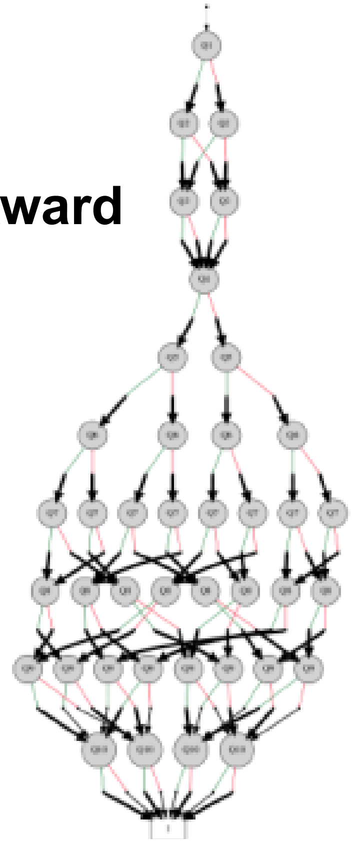
Simulating Shor‘s algorithm (31 Qubits)

- Microsoft‘s simulator
  - Exponential complexity
  - Memory: **50 GB**





- **DD-based simulation**
  - Exploit redundancies by shared nodes
  - Computations in **O(n²)**
  - Memory: **< 100 MB**
  - Simulation time: **< 1 Min**



- **Google Faculty Research Award**

Google

- Integration into
  - **IBM Qiskit**
  - **ATOS QLM**

Simulation – MQT DDSIM DEMO

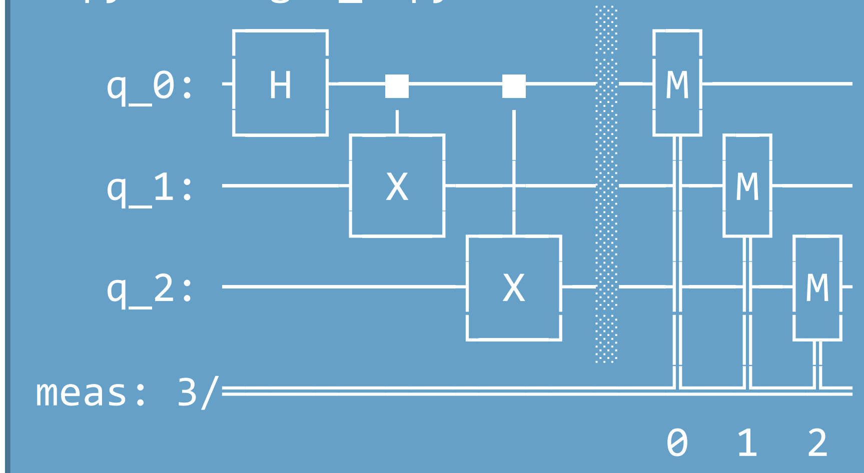
https://github.com/cda-tum/ddsim
pip install mqt.ddsim

ghz_3.py

```python
from qiskit import *
from jkq import ddsim

circ = QuantumCircuit(3)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
circ.measure_all()
print(circ.draw())

provider = ddsim.JKQProvider()
backend = provider.get_backend('qasm_simulator')
job = execute(circ, backend, shots=100000)
result = job.result()
counts = result.get_counts(circ)
print(counts)
```

Terminal

```
$ python3 ghz_3.py
```




```
{'000': 50149, '111': 49851}
```

Compilation

Conceptional algorithm

Limited Gate Set

Synthesis

Limited Connectivity

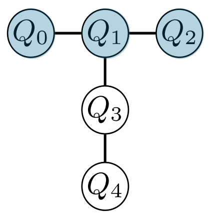
Mapping

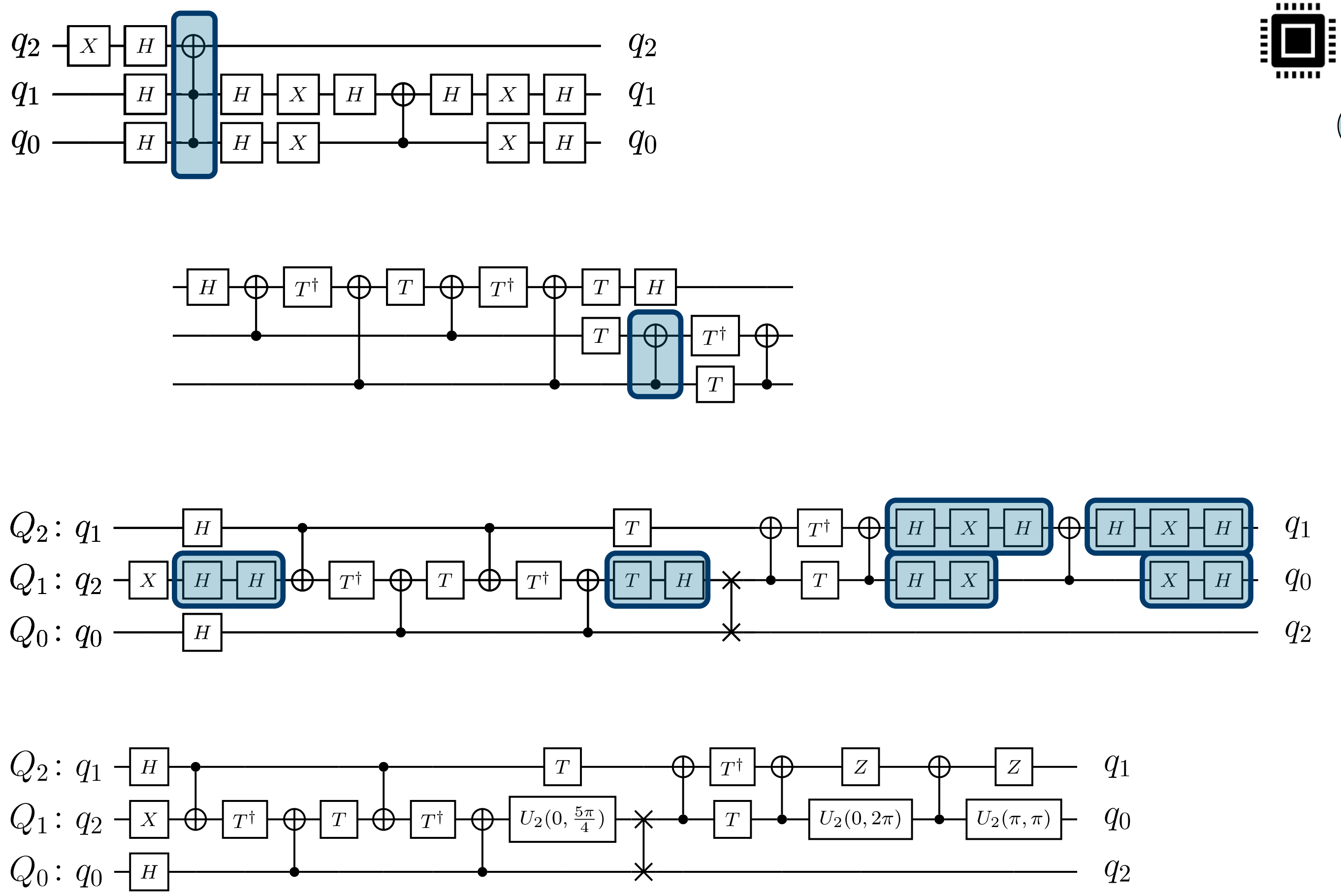
Limited Fidelity and Coherence

Optimizations

Actual Realization

Target Device





➔ **Kind of "typical" EDA problems** (some of them proven NP-hard)

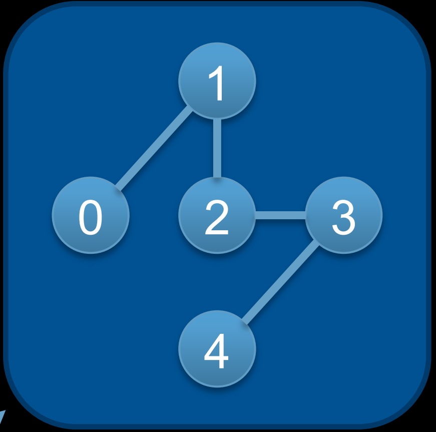
Compilation – MQT QMAP DEMO – JKQ DDSIM

https://github.com/cda-tum/qmap

pip install mqt.qmap

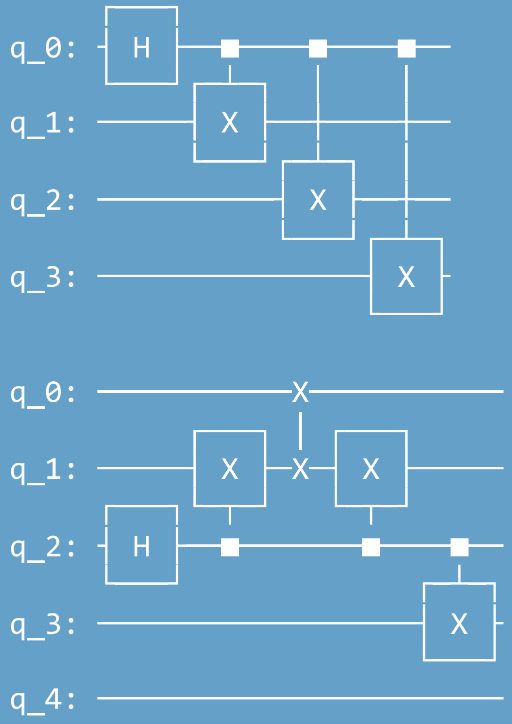
ghz_4.py

```python
from qiskit import *
from jkq import qmap

circ = QuantumCircuit(4)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
circ.cx(0, 3)
print(circ.draw())

arch = qmap.Arch.IBMQ_Bogota
method = qmap.Method.exact

result = qmap.compile(circ, arch=arch, method=method)

qasm = result['mapped_circuit']['qasm']
m_circ = QuantumCircuit.from_qasm_str(qasm)
print(m_circ.draw())
```



Terminal

```
$ python3 ghz_4.py
```



Verification

Conceptional algorithm

Limited Gate Set

Synthesis

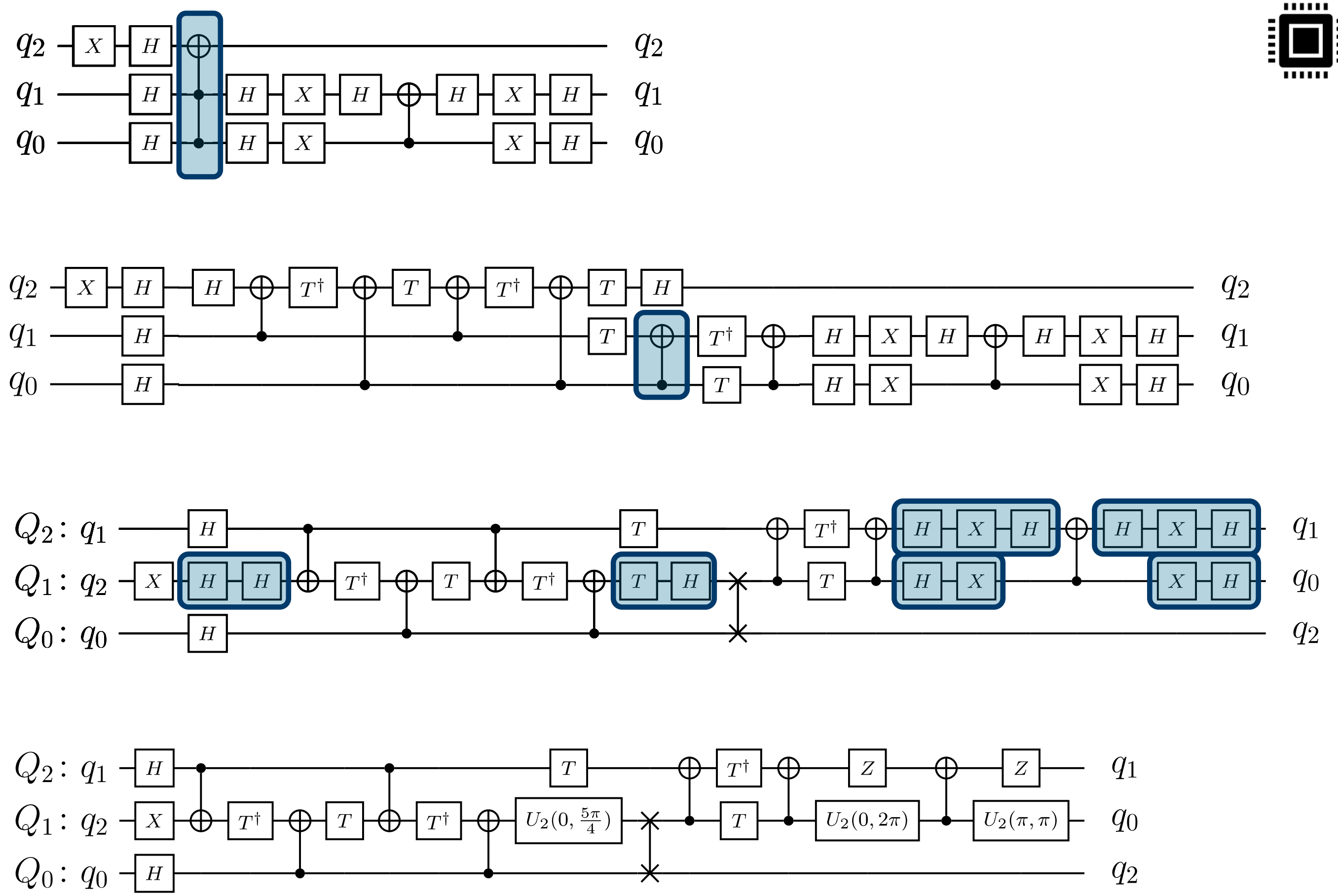
Limited Connectivity

Mapping

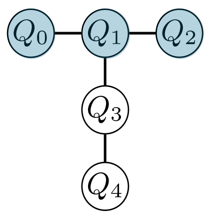
Limited Fidelity and Coherence

Optimizations

Actual Realization

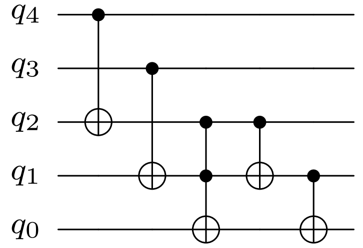


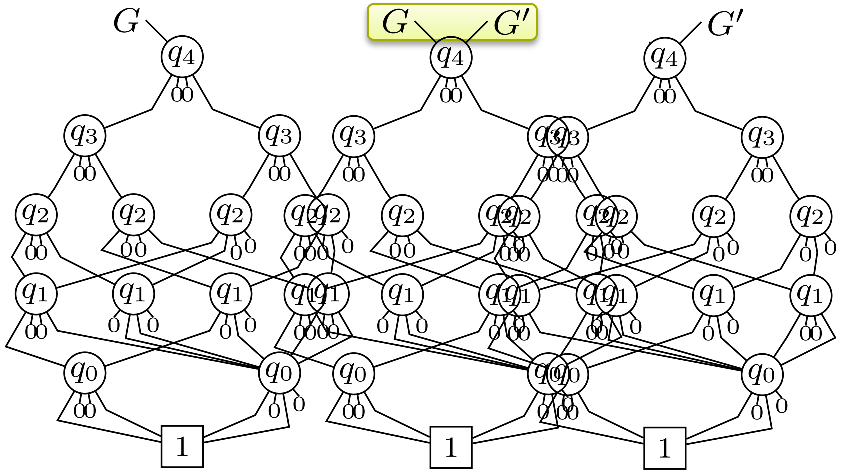
Target Device

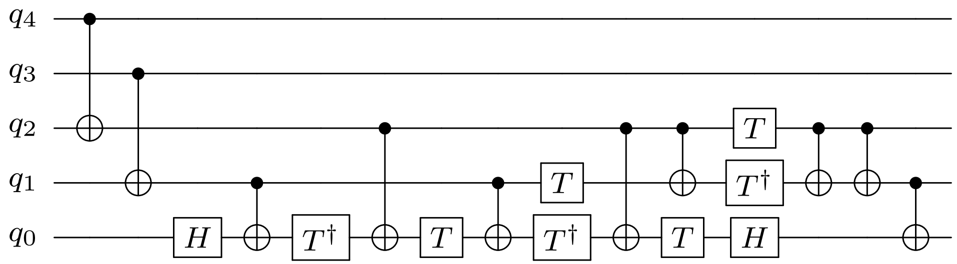


➔ **Kind of “typical” EDA problems** (some of them proven NP-hard)

Verification





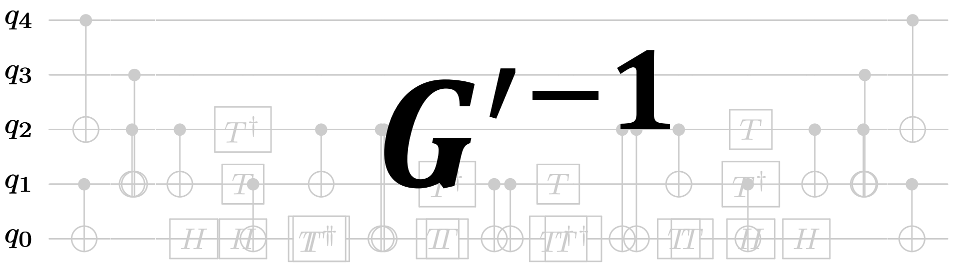


$$[U_{m-1}]\cdots[U_0] \quad [U] \quad \stackrel{?}{=} \quad [U'] \quad [U'_{m'-1}]\cdots[U'_0]$$

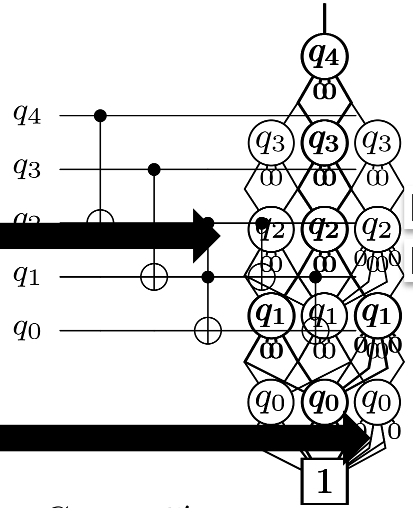
We can do even better

- If $G \equiv G'$, then $G'^{-1} \cdot G \equiv I$

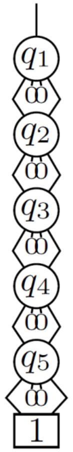
  - G'^{-1} easy due to reversibility

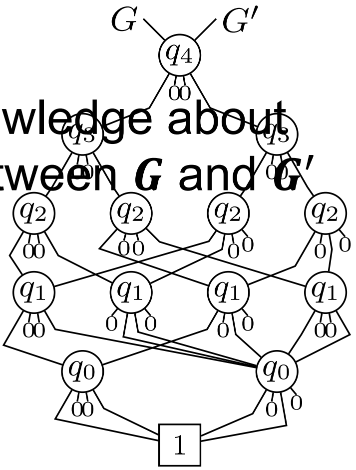


G'^{-1}



$\overset{?}{\equiv}$ I G





Strategies?

Complete DD for G'^{-1}

Exploit knowledge about relation between G and G'

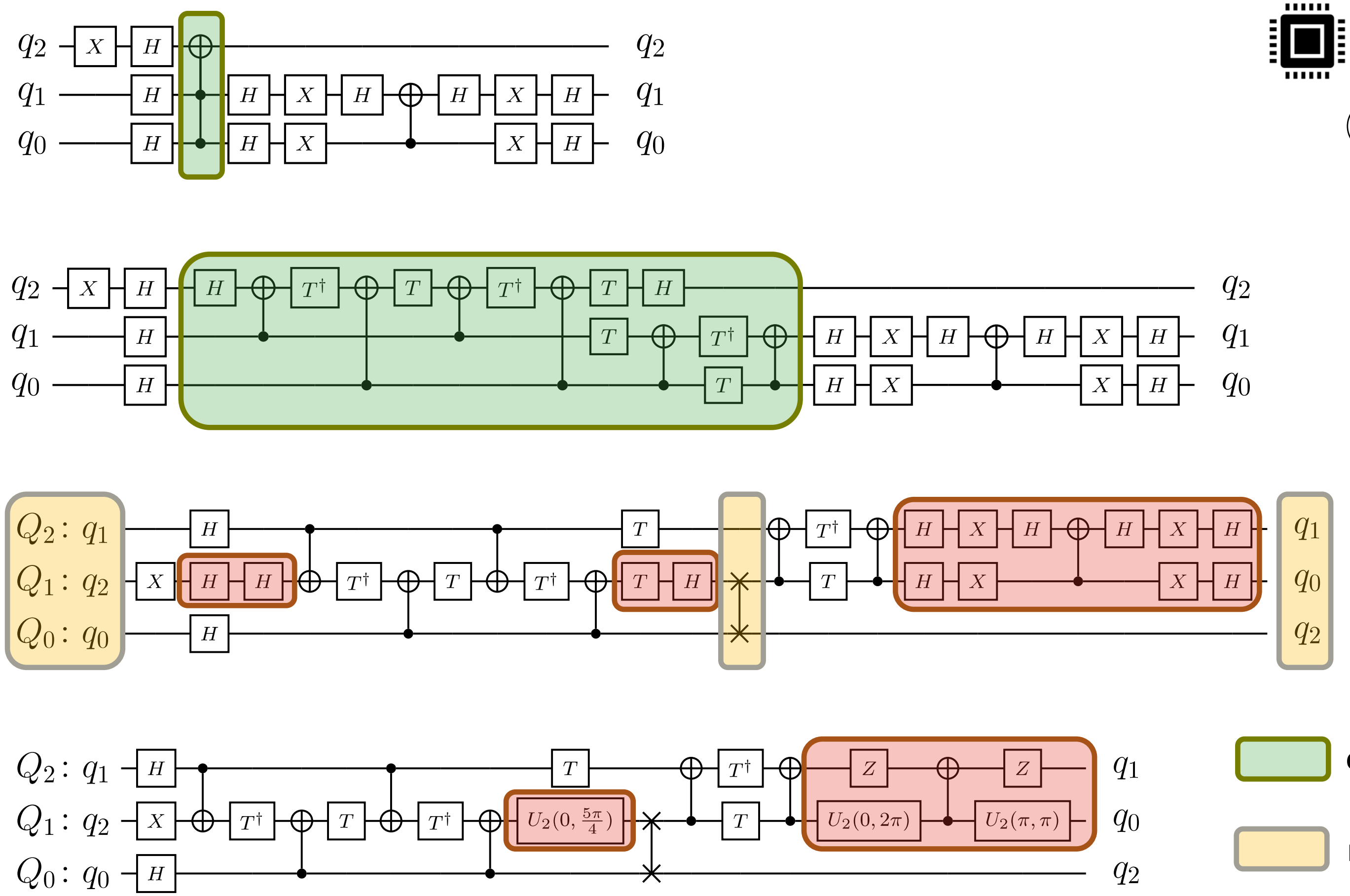
Efficient verification of compilation results

Exploiting Knowledge about the Compilation Flow

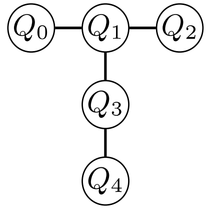
Conceptional algorithm

- Limited Gate Set
- Synthesis
- Limited Connectivity
- Mapping
- Limited Fidelity and Coherence
- Optimizations

Actual Realization



Target Device



easy to incorporate

necessary

potentially hard

Verification – MQT QCEC DEMO

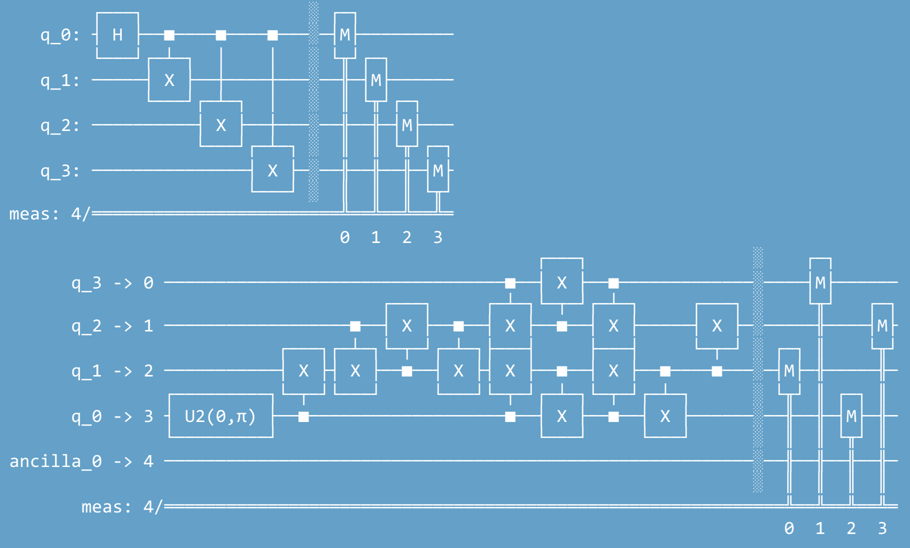
https://github.com/cda-tum/qcec
pip install mqt.qcec

ghz_4.py

```python
from qiskit import *
from qiskit.test.mock import FakeBogota
from jkq import qcec

circ = QuantumCircuit(4)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
circ.cx(0, 3)
circ.measure_all()
print(circ.draw())

backend = FakeBogota()
m_circ = transpile(circ, backend=backend, optimization_level=2)
print(m_circ.draw())

config = qcec.Configuration()
config.strategy = qcec.Strategy.compilationflow

result = verify(circ, m_circ, config)
print(result.equivalence)
```

Terminal

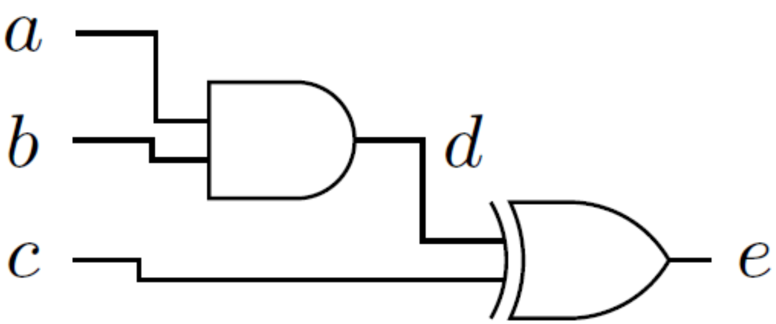
```
$ python3 ghz_4.py
```




```
Equivalence.equivalent
```

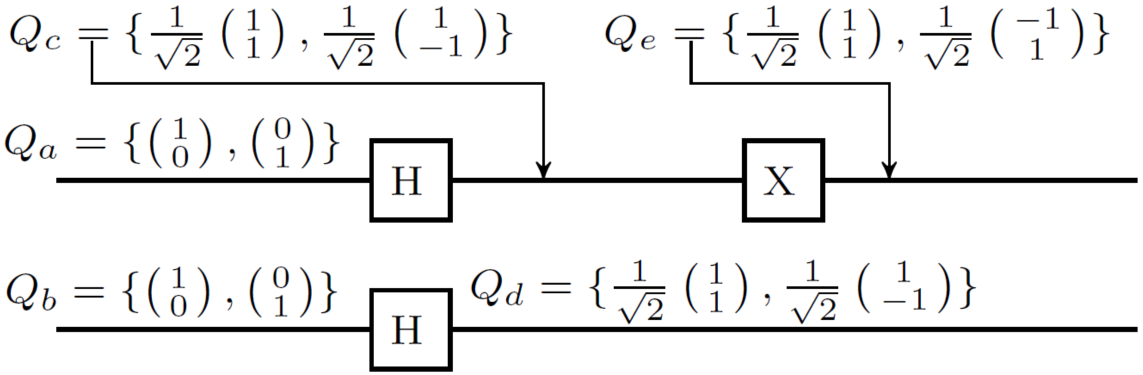
Core Methods

- SAT Solving (classical)



$$(x_d = x_a \wedge x_b)$$
$$\wedge\ (x_e = x_c \oplus x_d)$$

- SAT Solving (quantum)

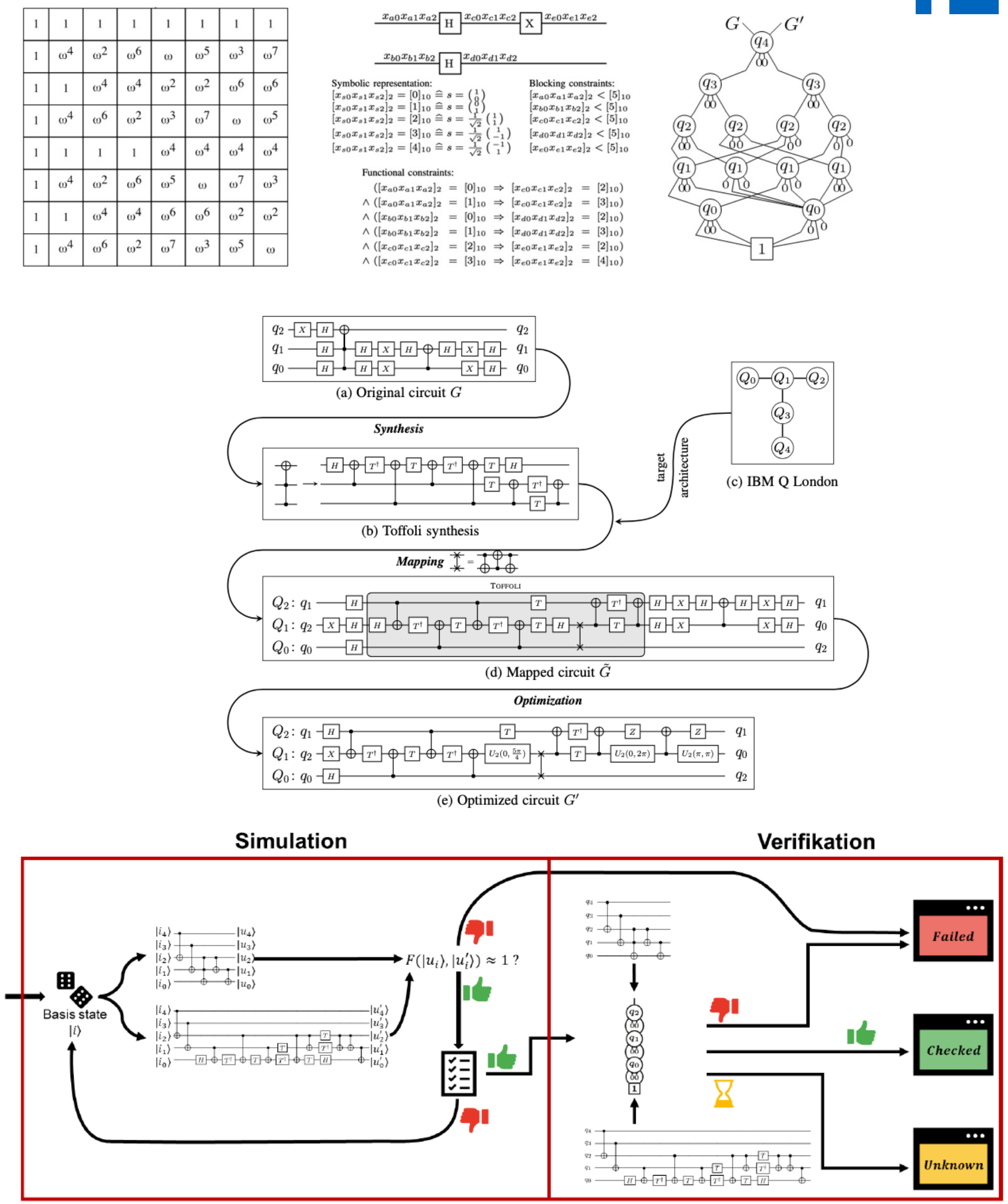


	$\mathcal{U}$	
	H	X
$q_0 = \binom{1}{0}$	$\frac{1}{\sqrt{2}}\binom{1}{1}$	-
$q_1 = \binom{0}{1}$	$\frac{1}{\sqrt{2}}\binom{1}{-1}$	-
$q_2 = \frac{1}{\sqrt{2}}\binom{1}{1}$	-	$\frac{1}{\sqrt{2}}\binom{1}{1}$
$q_3 = \frac{1}{\sqrt{2}}\binom{1}{-1}$	-	$\frac{1}{\sqrt{2}}\binom{-1}{1}$
$q_4 = \frac{1}{\sqrt{2}}\binom{-1}{1}$	-	-

- Exploitation of decades of design automation
- Goal: Generic interface/library providing a stack of alternatives to be used in a black-box fashion

Focus of This Talk

- **Development of (automatic) methods and tools for the design of quantum algorithms, quantum circuits, and quantum systems**
  - **Core Methods and Data-structures**
  - **Compilation/Synthesis** quantum functionality to quantum devices (e.g. IBM QX architectures, Ion-Traps)
  - **Simulation and verification** of quantum computations
  - …

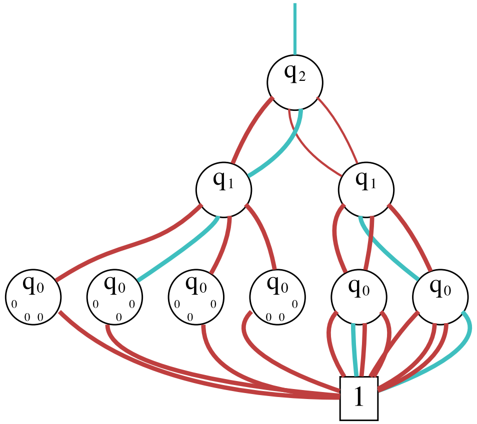


Selected Results

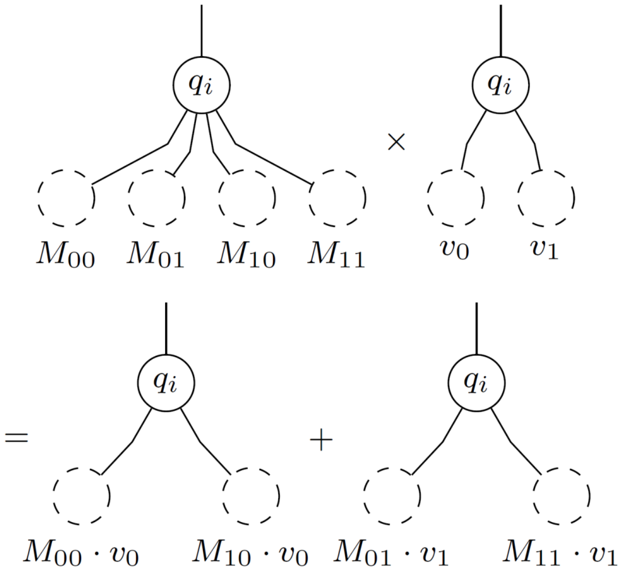
- Methods have been integrated in toolkits such as **IBM’s Qiskit** or **Atos’ QLM**
- The work on simulation of quantum computations has been recognized by a **Google Research Award**
- The work on compilation to IBM QX architectures received the **first price in the IBM QISKIT Developer Challenge**
- Work on Design Automation for Quantum Computing supported by **ERC Consolidator Grant**

Tools for Quantum Computing

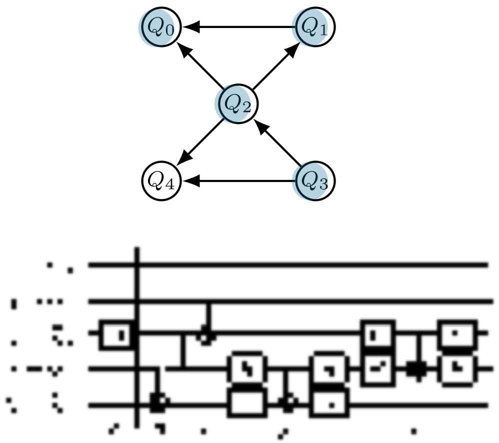
- Representation



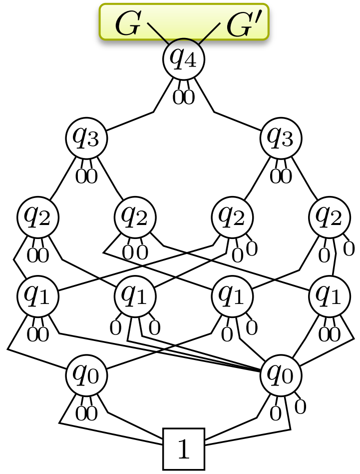
- Simulation



- Compilation



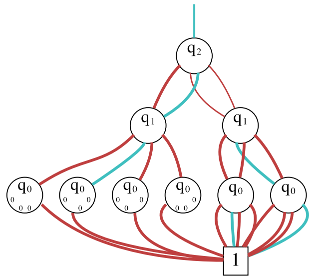
- Verification



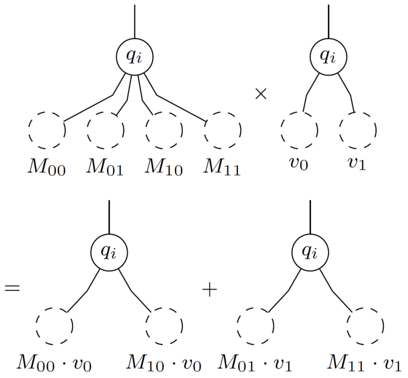
https://github.com/cda-tum

Tools for Quantum Computing

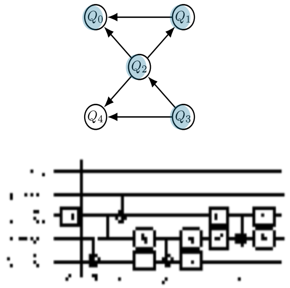
- Representation



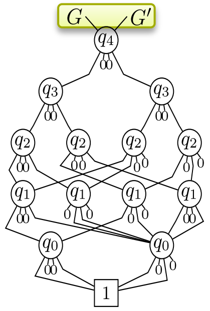
- Simulation



- Compilation

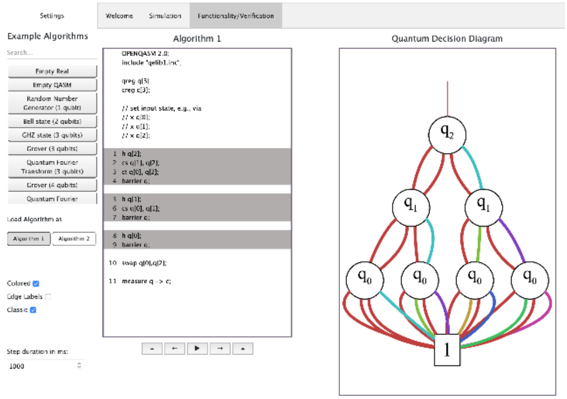


- Verification



https://github.com/cda-tum

- Web-based and installation-free tool



https://www.cda.cit.tum.de/app/ddvis/

- Can be used as black box (pure interface to conduct quantum operations; no understanding of the internal working necessary)

- More information: **https://www.cda.cit.tum.de/research/quantum/**