MQT QMAP:
Efficient Quantum Circuit Mapping

Robert Wille & Lukas Burgholzer
Technical University of Munich
Software Competence Center Hagenberg GmbH
Johannes Kepler University Linz

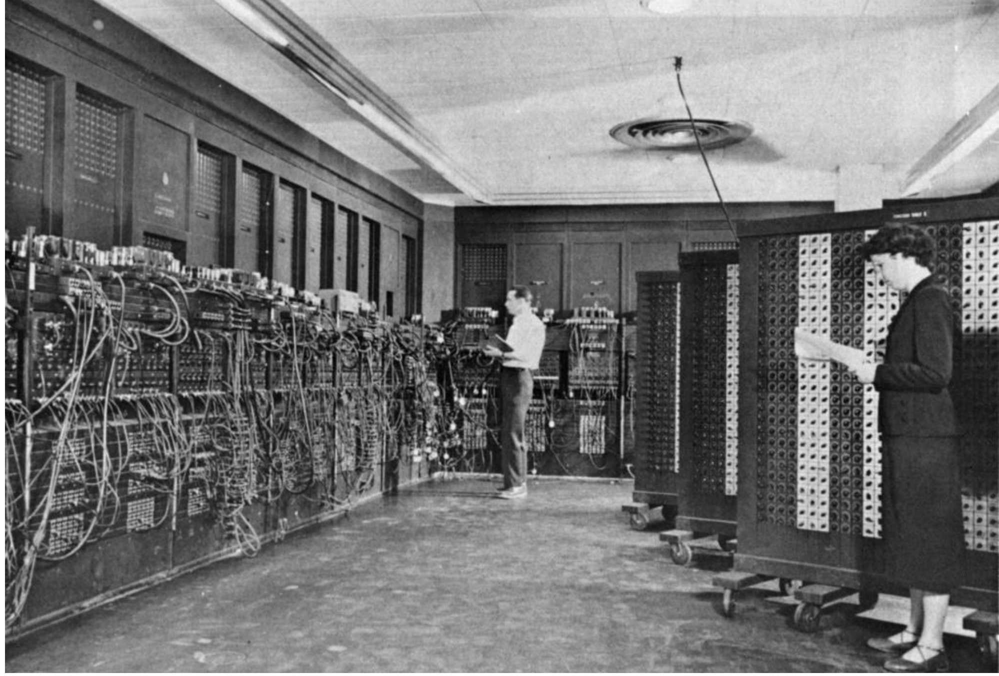
robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/
Connect on LinkedIn
Follow @rbrtwll

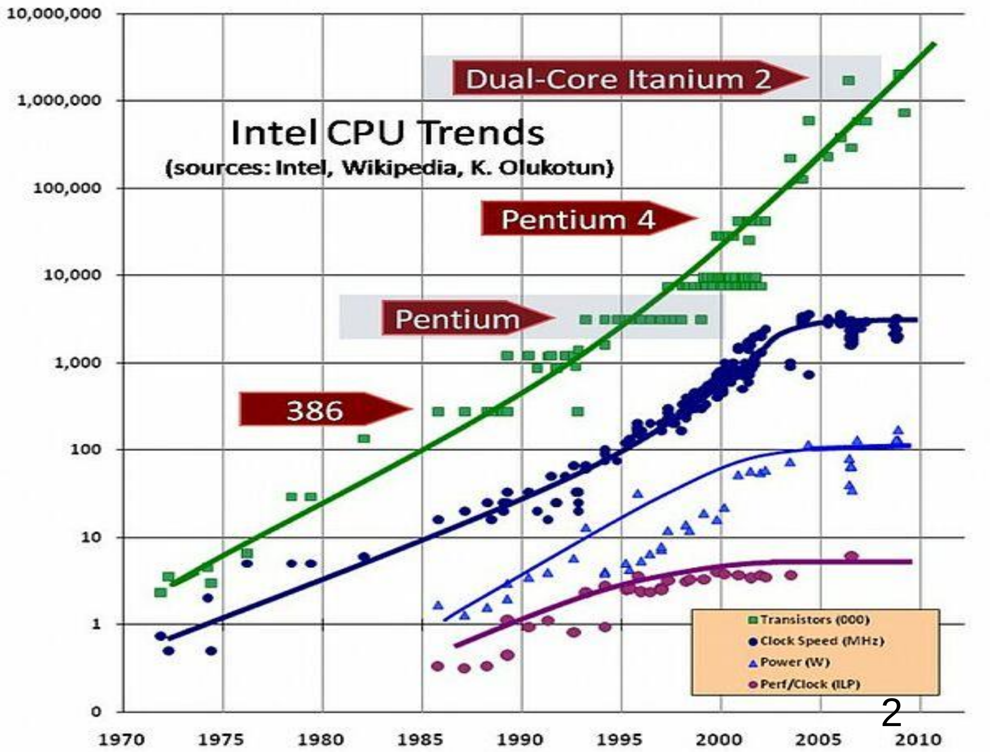
software competence center hagenberg

European Research Council

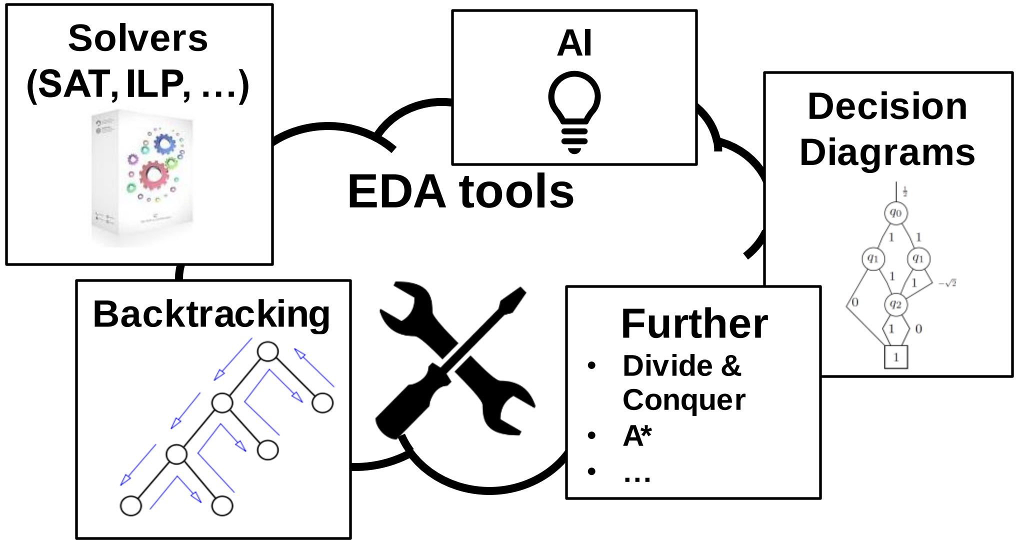
Design for Conventional Computing

- First, bulky computers
- Moores law
- Digital revolution





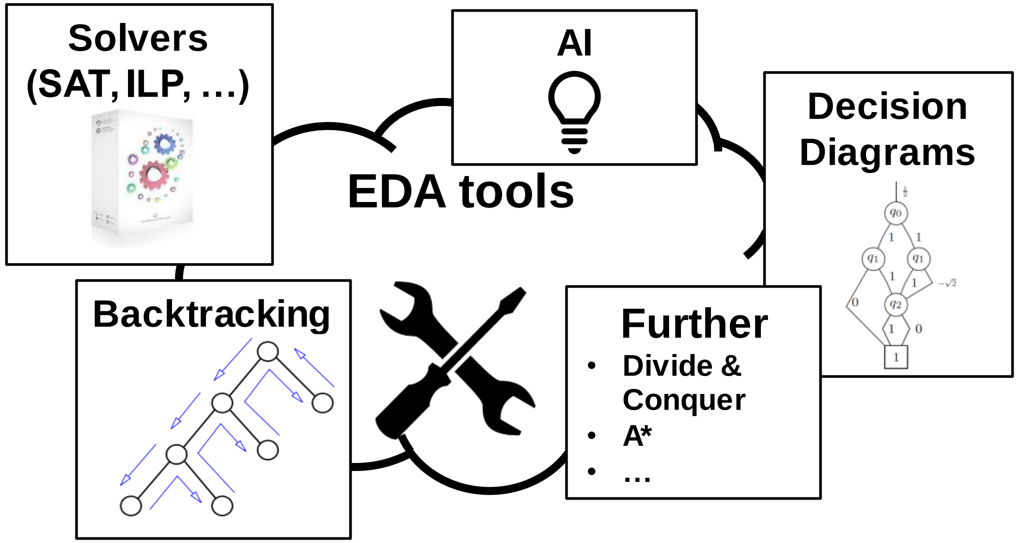
- **Design Automation** tools enabled their success
  - **Handle** problems with **enormous complexity**



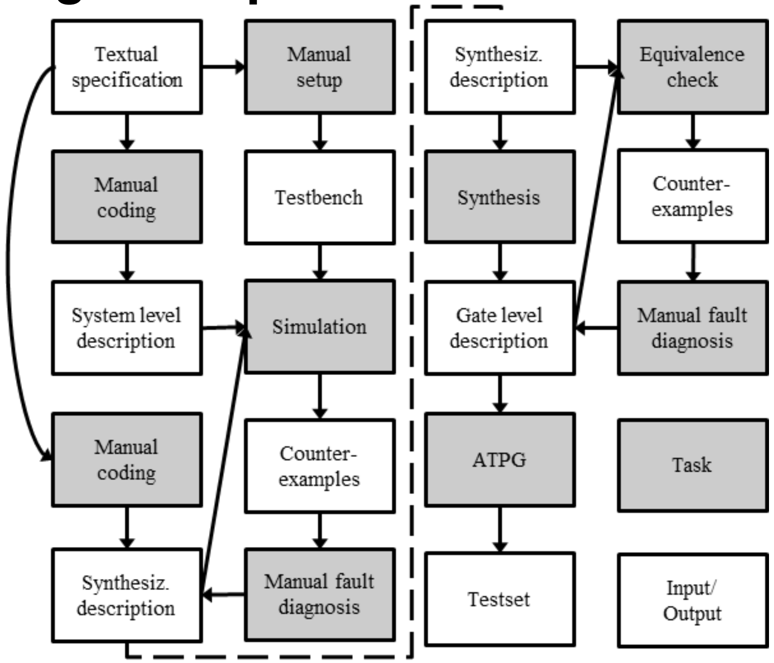
Design for Conventional Computing

- First, bulky computers
- Moores law
- Digital revolution

- **Design Automation** tools enabled their success
  - **Handle** problems with **enormous complexity**



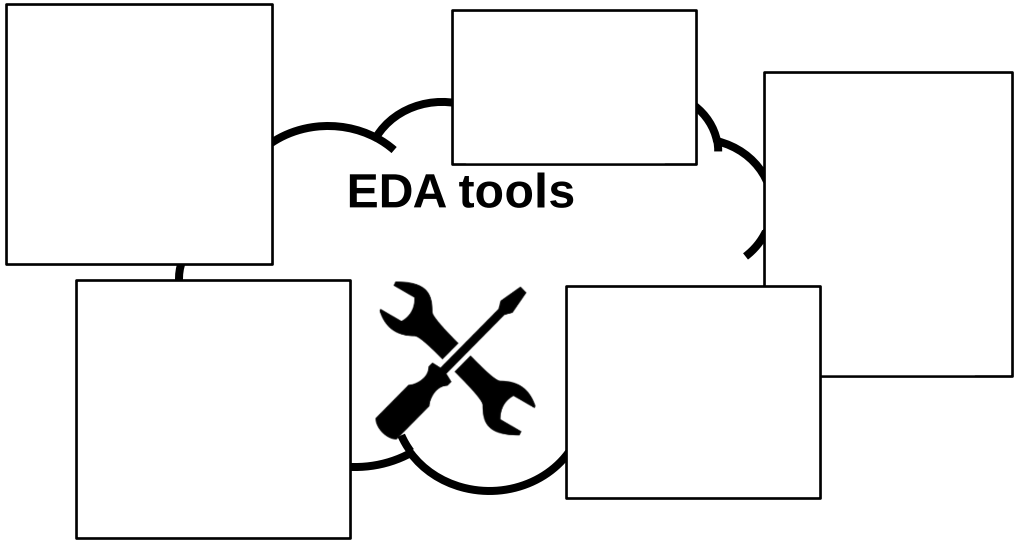
- **Emerge of powerful design/compilation flows**



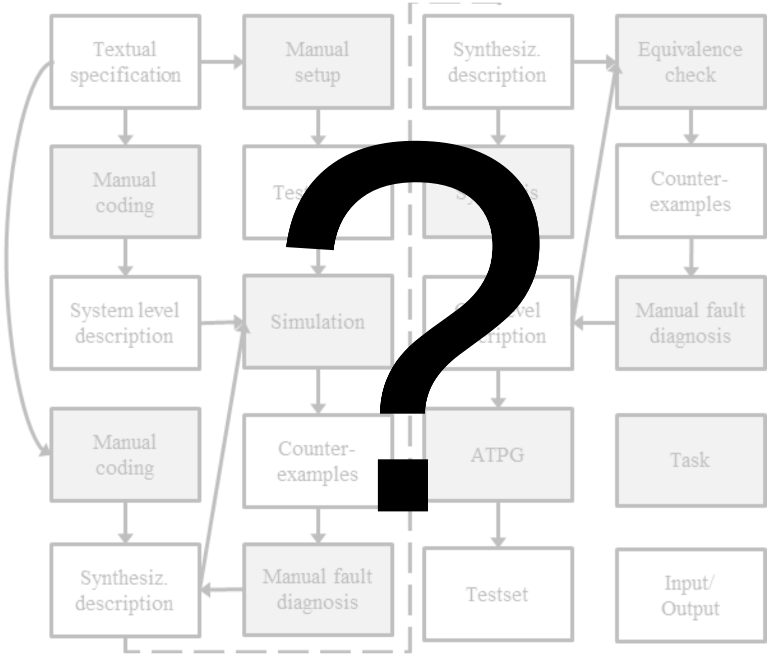
Design for Quantum Computing

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

- **Design Automation** tools



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

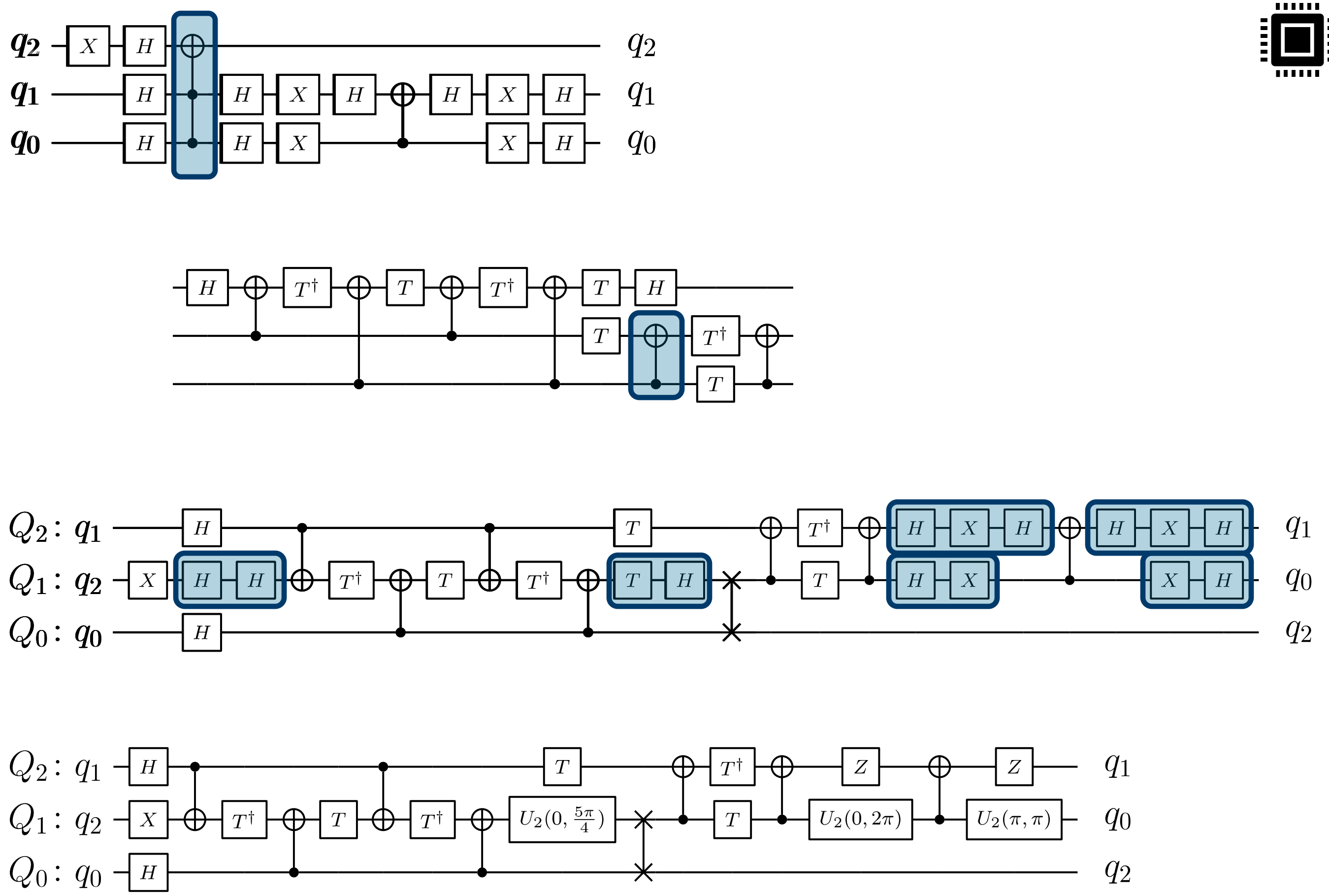


Quantum Circuit Mapping

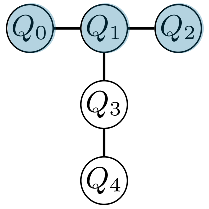
Conceptional algorithm

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

Actual Realization



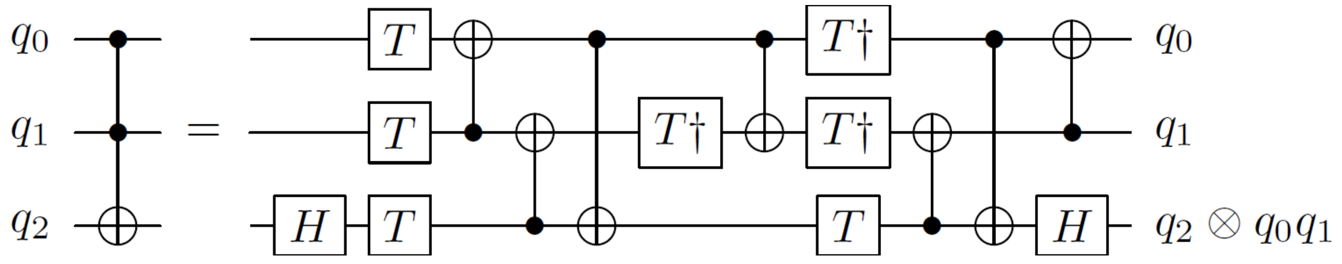
Target Device



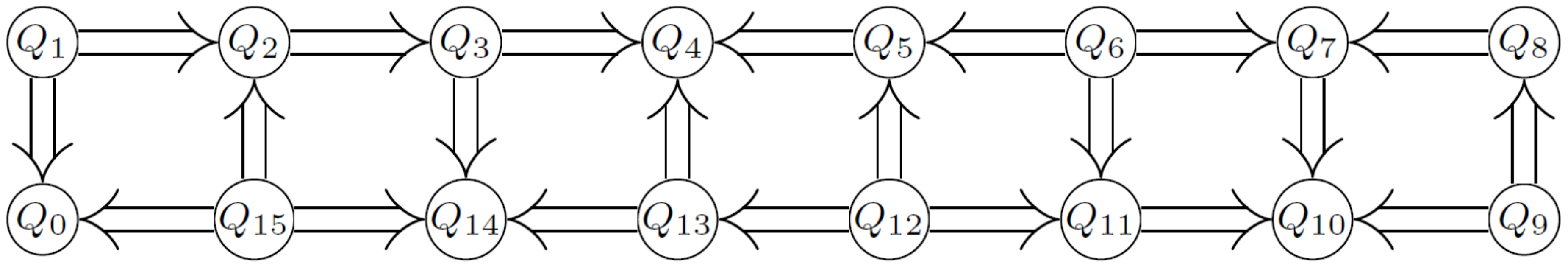
➔ **Kind of "typical" EDA problems** (some of them proven NP-hard)

Mapping to IBM Devices

- Decomposition into elementary operations
  - Single qubit gates $U(\theta,\phi,\lambda) = R_z(\phi)R_y(\theta)R_z(\lambda)$
  - **Controlled Not (CNOT)**



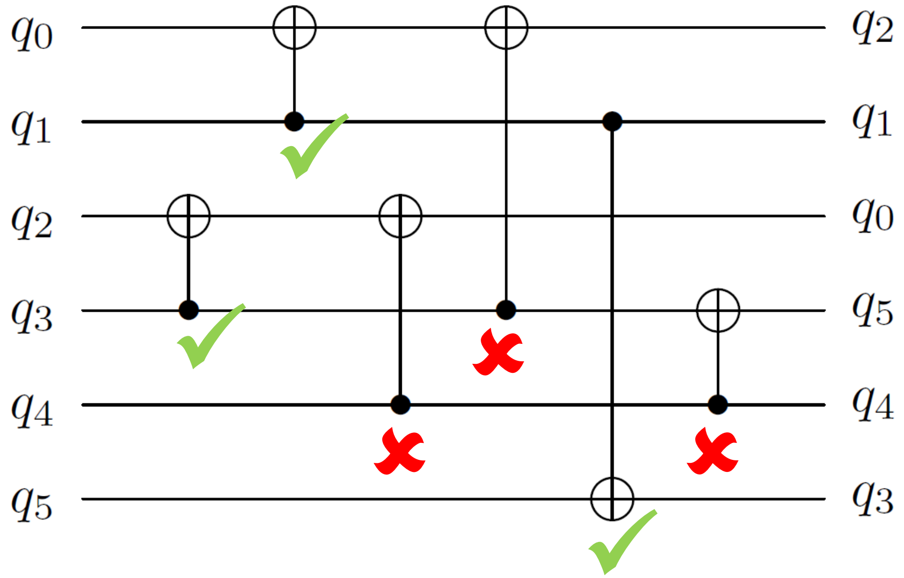
- Satisfying coupling constraints
  - Given by the architecture

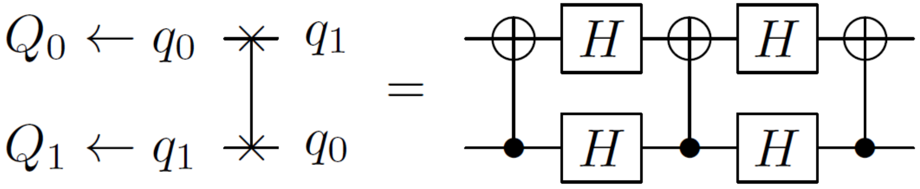


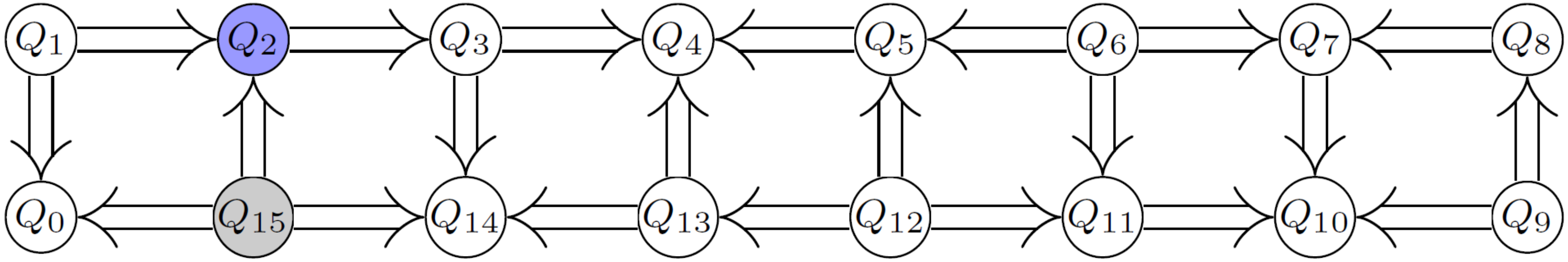
Satisfying coupling constraints

- Map logic qubits to physical qubits

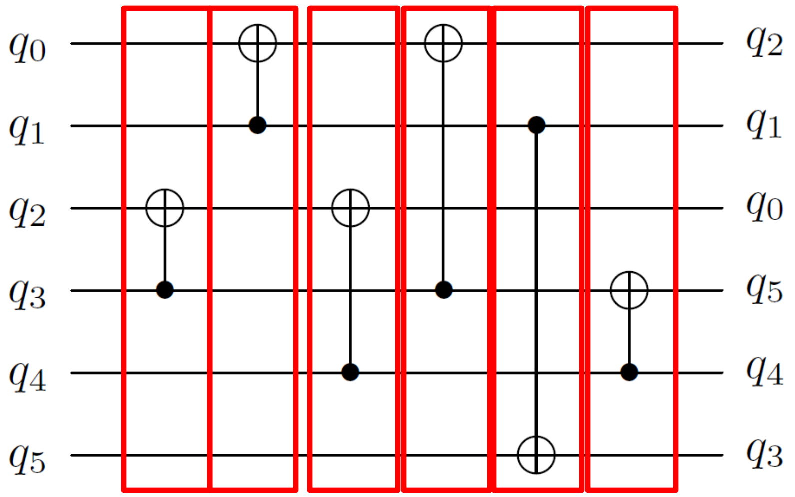
- Usually not possible to find such a mapping
  - Mapping has to change dynamically
  - Insert SWAP gates







Naive Solution



$Q_{14} \leftarrow q_0$

$Q_{15} \leftarrow q_1$

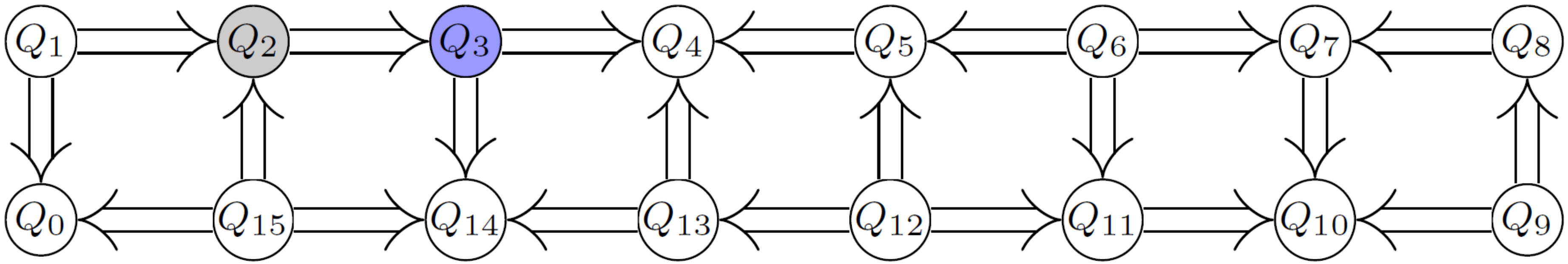
$Q_0 \leftarrow q_2$

$Q_1 \leftarrow q_3$

$Q_3 \leftarrow q_4$

$Q_2 \leftarrow q_5$

Required SWAPs: 5

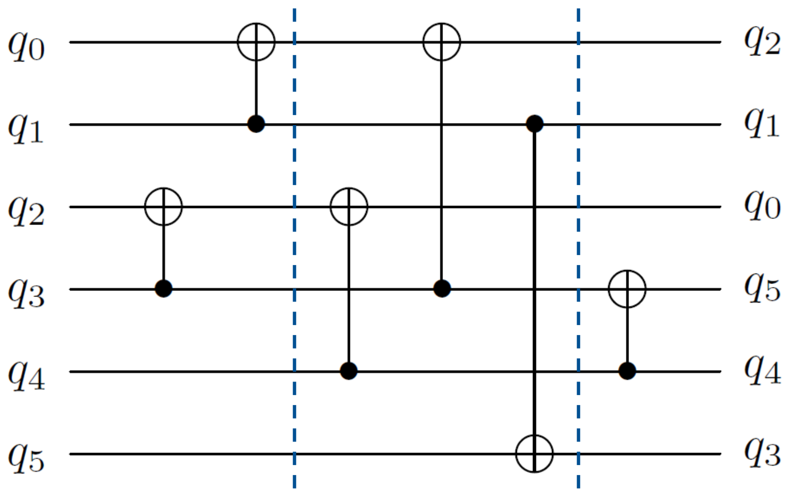


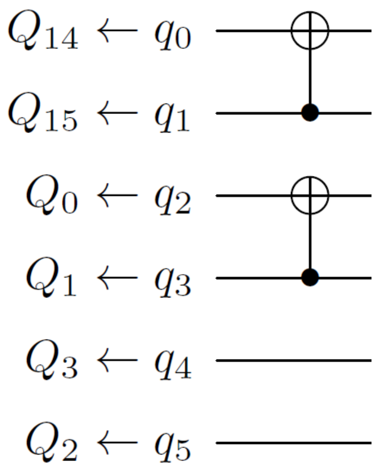
Heuristic Solution

- General idea: partition into layers
  - Map layers successively ($l_0 \pi_1 l_1 \pi_2 l_2$)
  - Fix mapping at beginning

Find locally optimal permutations π_i

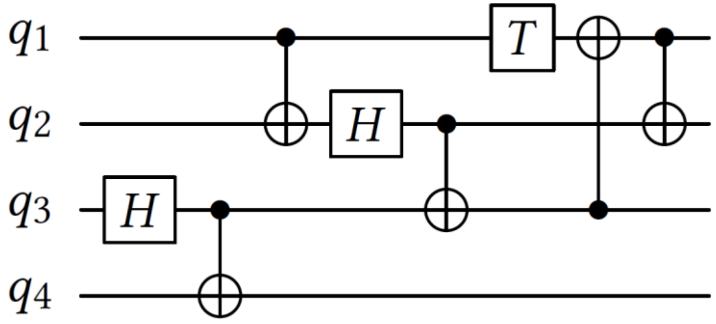
- Use A* search to cope with complexity



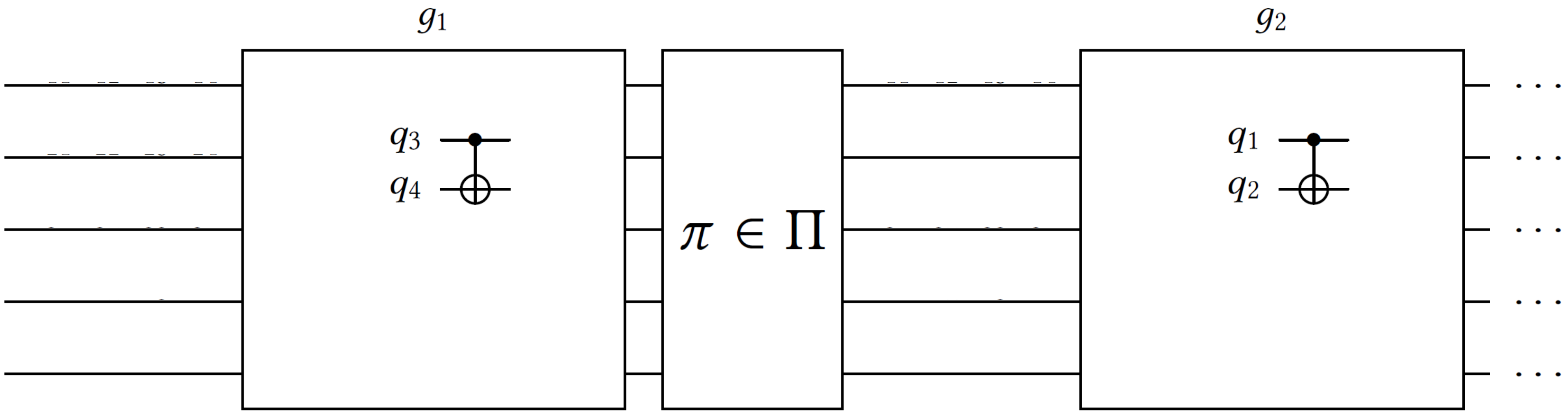


Required SWAPs: 4

Exact Solution



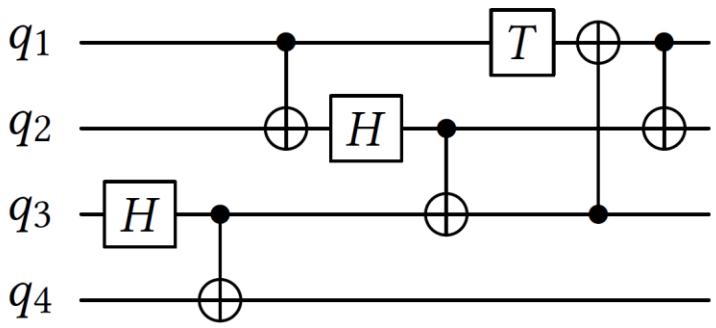
- Consideration of all possible permutations before each gate



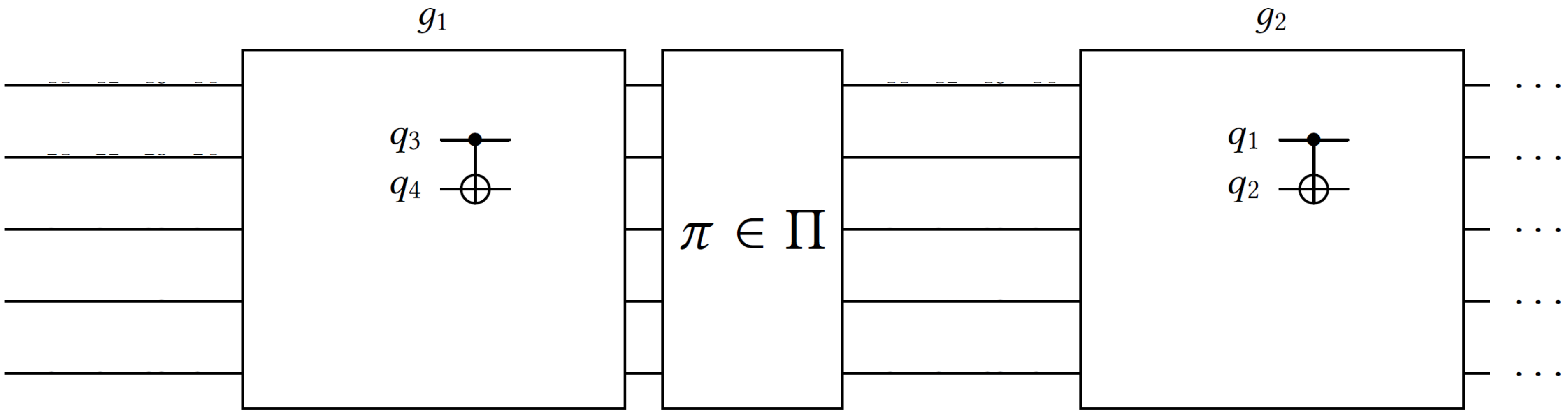
- Naïve Approach:
  - Enumerately consider all possible permutations for all gates of the given circuit
  - For each permutation satisfying the coupling constraints, calculate the number of SWAP gates required to realize the permutation
  - Afterwards, take the one with the smallest costs

  ➔ n!d possible combinations
  (n…number of lines, d…number of gates)

Exact Solution



- Consideration of all possible permutations before each gate



- Consistency constraints:

$$\bigwedge_{k=1}^{|G|}\left(\bigwedge_{j=1}^{n}\left(\sum_{i=1}^{m} x_{ij}^{k} = 1\right) \wedge \bigwedge_{i=1}^{m}\left(\sum_{j=1}^{n} x_{ij}^{k} \leq 1\right)\right)$$

- Coupling map constraints:

$$\bigwedge_{g_k = CNOT_k(q_c,q_t)\in G}\left(\bigvee_{(p_i,p_j)\in CM} (x_{ic}^{k} \wedge x_{jt}^{k}) \vee (x_{it}^{k} \wedge x_{jc}^{k})\right)$$

- Permutation constraints:

$$\bigwedge_{k=2}^{|G|}\left(\bigwedge_{\pi\in\Pi}\left(\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{n}\left(x_{ij}^{k-1} = x_{\pi(i)j}^{k}\right)\right) \Leftrightarrow y_{\pi}^{k}\right)$$

- Cost function:

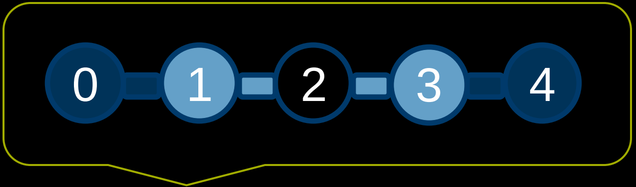
$$\mathcal{F} = \sum_{k=2}^{|G|}\sum_{\pi\in\Pi}\left(7 \cdot \mathrm{swaps}(\pi)\, y_{\pi}^{k} \qquad \right)$$

Quantum Circuit Mapping – MQT QMAP

https://github.com/cda-tum/qmap or simply `pip install mqt.qmap`

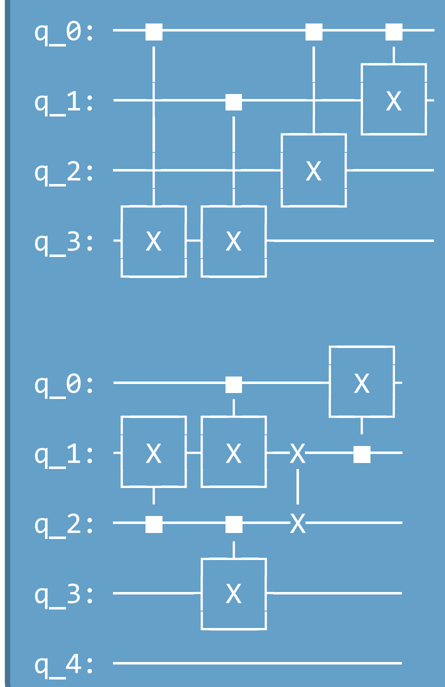
map.py

```
from qiskit import *
from qiskit.providers.fake_provider import FakeBogota
from mqt import qmap

q = QuantumRegister(4, 'q')
circ = QuantumCircuit(q)
circ.cx(q[0], q[3])
circ.cx(q[1], q[3])
circ.cx(q[0], q[2])
circ.cx(q[0], q[1])
print(circ.draw())

m_circ, results = qmap.compile(circ, arch=FakeBogota(), method='exact')

print(m_circ.draw())
```



Terminal

```
$ python3 map.py
```



Munich Quantum Toolkit (MQT)

Towards Applications and End Users

Prediction of “Good” Compilation Options

Automated Framework for Realizing Quantum Solutions

Cross-Level Benchmark Suite MQT Bench

Datastructures

Decision Diagrams

Tensor Networks

ZX-Calculus

Conceptional algorithm

Limited Gate Set

Synthesis

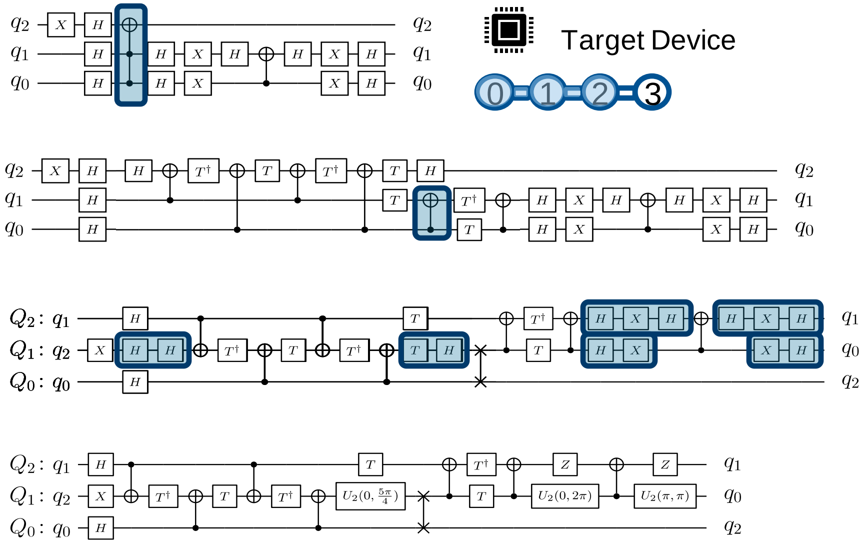
Limited Connectivity

Mapping

Limited Fidelity and Coherence

Optimizations

Actual Realization



Design Methods

Classical Simulation of Quantum Circuits

Verification of Compilation Results

Equivalence Checking

Towards Physical Realizations

Error Correcting Codes

Ion-Trap Compilation

Multi-Level Compilation