

Supporting End Users in Implementing Quantum Computing Applications



Nils Quetschlich (nils.quetschlich@tum.de), Supervisor: Robert Wille

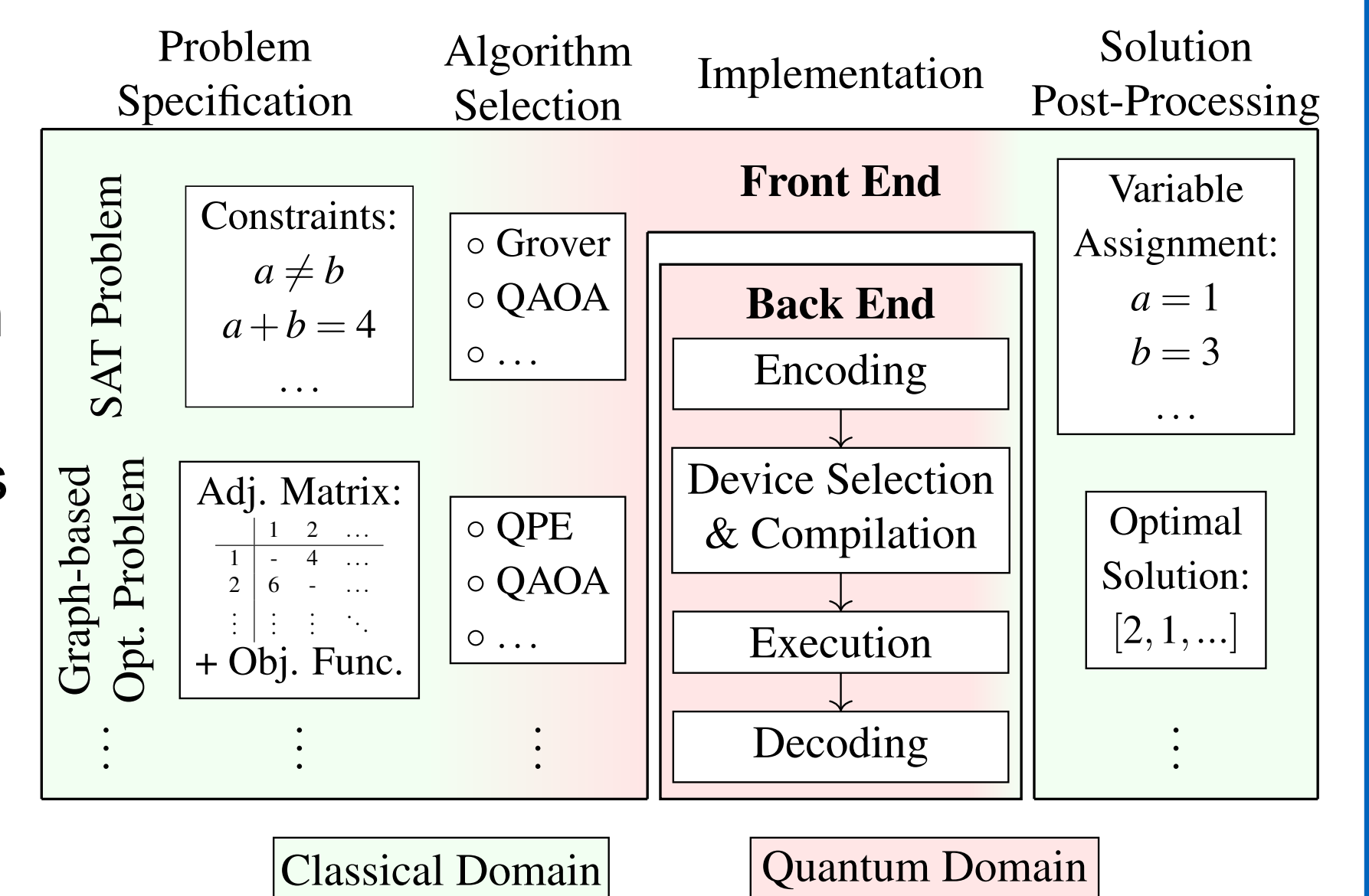
Abstract

Implementing quantum computing applications requires expert knowledge and four steps must be conducted to solve a considered problem after a suitable quantum algorithm is chosen: (1) encoding the problem as a quantum circuit based on the chosen algorithm, (2) selecting and compiling for a suitable device, (3) executing the circuit, and (4) decoding the results. This creates a high entry barrier for end users with limited quantum expertise who need solutions to domain-specific problems. This poster highlights methods developed to support end users, resulting in multiple open-source tools in the **Munich Quantum Toolkit (MQT)** on GitHub:

- **End User Workflow (MQT ProblemSolver)**: Providing a workflow from a classical problem as input to a quantum computing solution.
 - **Quantum Device Selection and Compilation (MQT Predictor)**: Selecting and efficiently compiling for the most suitable quantum device.
 - **Benchmark Suite (MQT Bench)**: Offering representative test cases in a benchmark suite of quantum applications.
- These tools simplify quantum computing to make it more accessible.

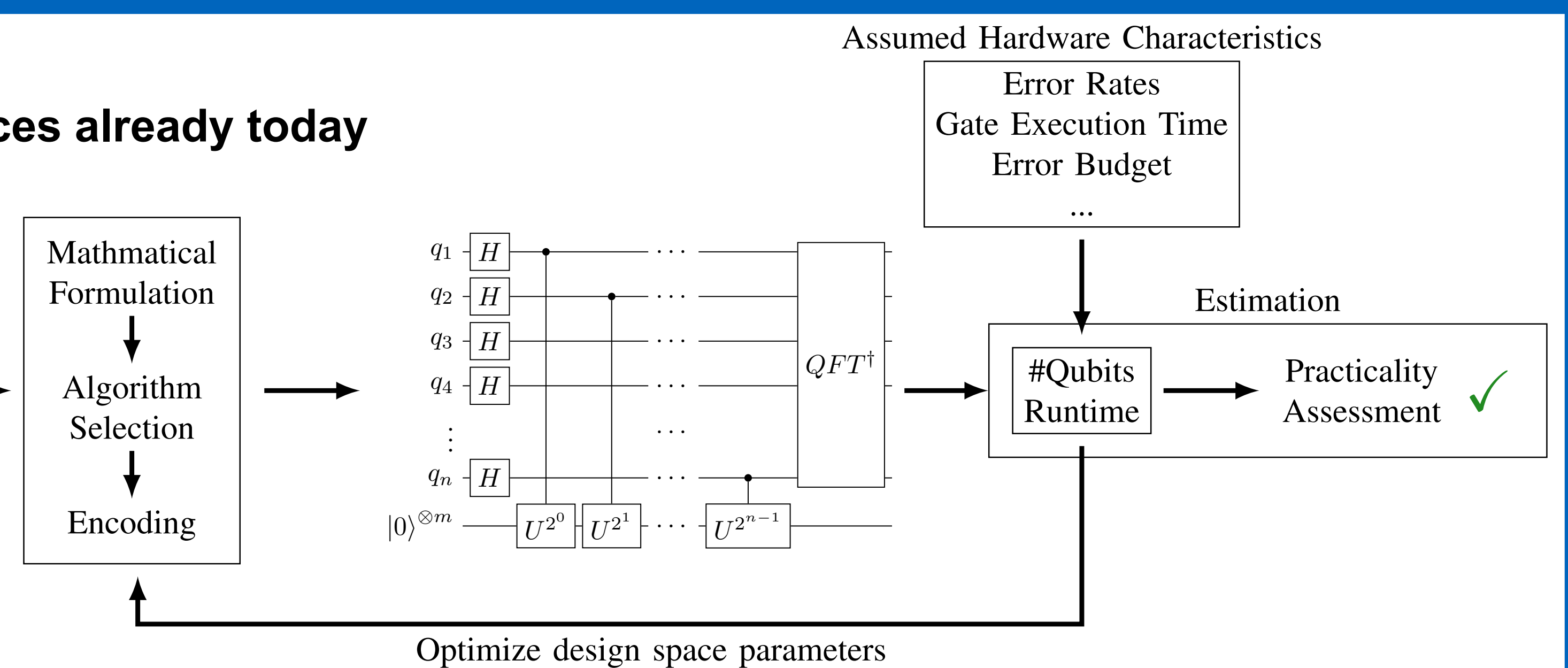
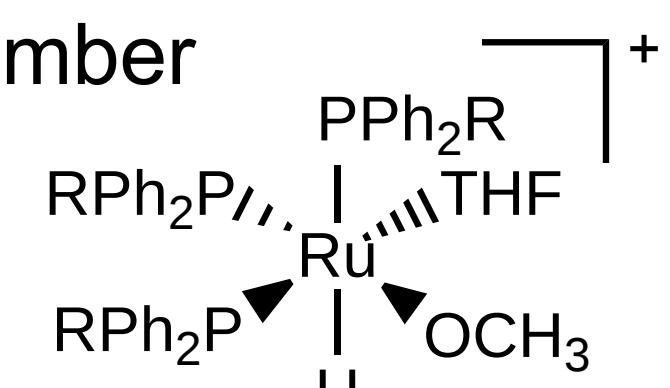
MQT ProblemSolver

- **Goal: Shielding end users as much as possible from the quantum domain**
- Automation of solving problems from various problem classes
- Providing the same interfaces as classical solvers



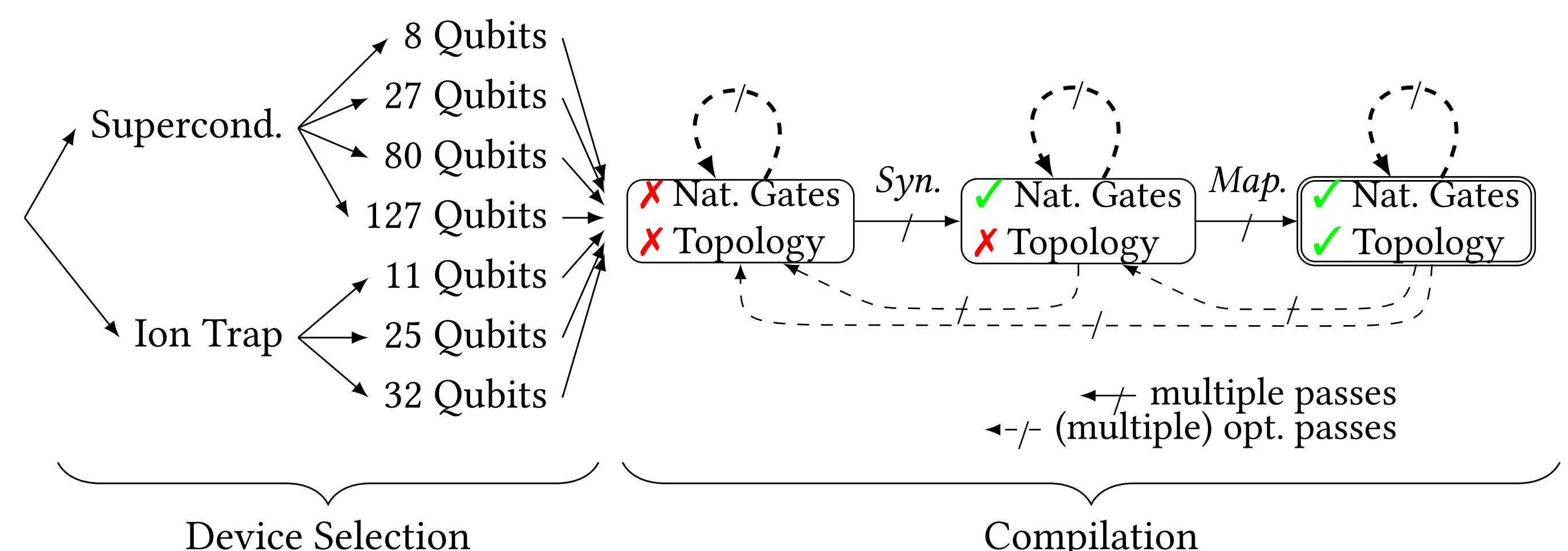
Extended MQT ProblemSolver Workflow Using Resource Estimation

- **Goal: Allowing end users to consider real-world problem instances already today**
- Due to the limited capacities of both simulators and NISQ devices, only toy-size problem instances can be considered
- Rather than executing a quantum circuit, its number of qubits and expected runtime can be estimated to explore possible optimizations of real-world problem instances across the entire design space



MQT Predictor

- **Goal: Guiding end users through the compilation process**
- Executing an application implemented in a quantum circuit requires the selection of a quantum technology, a respective device, and compilation options; often overwhelming end users
- The MQT Predictor framework automatically makes those decisions for a given circuit using supervised machine learning and reinforcement learning



MQT Bench

- **Goal: Increased comparability, reproducibility, and transparency**
- Huge benchmark suite with > 70,000 benchmarks of various algorithms on four abstraction levels
- Integrated into Xanadu's PennyLane



Open-source Tools



mqt-problemsolver



mqt-predictor



mqt-bench

Publications & Accomplishments

- **2 Journal Publications (ACM Transactions on Quantum Computing and the Quantum journal)**
- **12+ Conference Publications**
 - 2 x Design Automation Conferences (**DAC** and **DATE**)
 - 5 x International Conference on Quantum Computing and Engineering (**QCE**)
 - 5 x International Conference on Quantum Software (**QSW**) (including an invited paper)
- **Further Activities**
 - Participation in numerous coding challenges
 - Mentor at the 12th NYUAD Quantum Computing Hackathon
 - Tutorial on the **Munich Quantum Toolkit (MQT)** at the **Supercomputing Conference (SC) 2024**
 - **Collaborations with Industry** (Microsoft, BMW, Xanadu), **Academia** (University of British Columbia, Canada; Politecnico di Torino, Italy; Ludwig Maximilian University, Germany; University of Cambridge, United Kingdom), and **Research Institutes** (Fraunhofer, German Aerospace Center)
- **Open-source Tools**: > 190 GitHub stars and > 90k downloads