Design Automation for Quantum Computing

Intermediate Stage Report of the ERC Consolidator Grant "DAQC"

Robert Wille
Chair for Design Automation, Technical University of Munich, Germany
robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

***Abstract*—We are at the dawn of a new "computing age" in which quantum computers hopefully will find their way into practical applications. However, while impressive accomplishments can be observed in the physical realization of quantum computers, the development of automated tools and methods that provide assistance in the design and realization of applications for those devices is at risk of not being able to keep up with this development anymore—leaving a situation where we might have powerful quantum computers but hardly any proper means to actually use them.**

The ERC Consolidator project "Design Automation for Quantum Computing" aims to provide a solution for this upcoming design gap by developing efficient and practically relevant design methods for this emerging technology. While the current state of the art suffers from the interdisciplinarity of quantum computing (leading to the consideration of inappropriate models, inconsistent interpretations, and "wrong" problem formulations), this project builds a bridge between the design automation community and the quantum computing community. This will allow to fully exploit the potential of design automation which is hardly utilized in quantum computing yet.

This intermediate stage report provides an overview of the motivation and approach of the project as well as showcases selected results and outreach activities conducted in the first two years of the project.

I. Introduction & Motivation

In the 1970s, researchers started to utilize quantum mechanics to address questions in computer science and information theory—establishing new research directions such as quantum computing [1]. Here, quantum bits (i.e., qubits) serve as elementary information unit, which—in contrast to classical bits—can not only be in one of its two orthogonal basis states (denoted $|0\rangle$ and $|1\rangle$ using Dirac notation), but also in an almost arbitrary superposition of both (i.e., $\alpha|0\rangle + \beta|1\rangle$, where the complex factors α and β satisfy $\alpha\alpha^* + \beta\beta^* = 1$). This allows an n-qubit quantum system to represent 2^n different complex values at once—exponentially many more than classical n-bit systems (which can only represent n different Boolean values at a time). Together with further quantum-mechanical phenomena such as entanglement, this allows for substantial improvements in information density as well as computation power and motivated the establishment of dedicated research areas in computer science and information theory investigating and exploiting this potential.

One of these research areas, namely *quantum computing*, covers the development of applications that exploit these quantum-mechanical effects in order to solve certain problems significantly faster (in the best case, exponentially faster) than classical computers. First quantum applications such as Grover's Search or Shor's Algorithm which improve database search [2] or enable integer factorization in polynomial time [3] (a "killer application" that would change the way cryptography works today [4]), respectively, have already been proposed in the last century. Since then, corresponding developments have significantly broadened and, in the meantime, cover a huge variety of quantum applications for quantum chemistry, solving systems of linear equations, physical simulations, machine learning, quantum finance, and many more [5]–[7].

These developments are significantly triggered by the fact that quantum computers are reaching feasibility, i.e., are maturing from an "academic vision" implemented in laboratory experiments towards a practical reality supported in compute centers. In fact, a broad spectrum of technologies, including superconducting qubits [8], [9], ion traps [10], [11], neutral atoms [12], [13], and more are gaining momentum. This is also confirmed by the involvement of "big players" such as IBM, Google, Microsoft, Eviden, Intel, as well as the emerge of several specialized startups such as AQT, Rigetti, IonQ, Quantinuum, IQM, and many more.

All these accomplishments are leading to an increasing complexity in the design of corresponding quantum computing applications and systems that often can and soon will not be handled manually anymore. This demands for dedicated design automation solutions. In the classical realm (i.e., for electronic circuits and systems), sophisticated design tools for that are taken for granted today and constitute a main reason for the utilization and penetration of electronic devices into almost all parts of our daily life. In contrast, the development of design methods for quantum computing is in its infancy and far away from the performance and impact of their classical counterparts. The three main reasons for this are:

1) *Complexity:* Many design problems in these areas are of combinatorial and exponential nature (some have even been proven to be NP-complete [14], coNP-hard [15], or QMA-complete [16]). Even presumably simple tasks such as the logic simulation of a circuit (which has a linear complexity in the classical realm) suddenly yield on exponential complexity (since an exponential number of amplitudes must be considered). The experience of the design automation community may be able to tackle this complexity. In fact, many efficient solutions have been developed for problems with similar complexity in the classical realm. But most of the experts in quantum computing do not have a background in design automation and, hence, this potential is left untapped.

979-8-3503-4859-0/DATE24/© 2024 EDAA

2) *Terminology and Formalizations*: Vice versa, experts in design automation often do not have a sufficient background in quantum computing. Since, additionally, terminologies as well as formalizations are often unclear and/or ambiguous to design automation experts, frequently "wrong problems" are addressed or inappropriate models, faulty assumptions, or misleading cost metrics are used. Because of that, the potential of design automation (which dominates the design of classical systems today) is not fully exploited yet.
3) *Interdisciplinarity*: A closer interaction between the quantum computing community and the design automation community would help addressing many of these problems. But bridging the gap between both communities requires lots of resources and time to exchange with experts and stakeholders of the field. Different "languages" as well as "cultures" in the respective fields additionally complicates interactions.

Overall, the little interaction and exchange between experts of design automation and quantum computing are the main reasons why many relevant problems in the design of quantum computing have not or insufficiently been considered yet. As a result, we may end up in a situation, where we may have powerful quantum computers but hardly any proper means to actually use them.

II. Approach

The ERC Consolidator project "Design Automation for Quantum Computing" aims to provide a solution for this upcoming design gap by developing efficient and practically relevant methods for corresponding design tasks. To this end, the following approach is applied:

- *Categorizing and defining design tasks:* As indicated above, a significant problem is that design tasks for quantum computing are often not clearly defined and that existing tools are often not categorized in a fashion which is accessible for the design automation community. Even worse, there often is no clear and precise terminology for these tasks. For example, in the context of compilation terms like *synthesis*, *decomposition*, and *mapping* are used, referring to the same or similar concepts. These are all justified since quantum computations are commonly described by so-called *quantum circuits* even though they describe a sequence of operations applied to a set of qubits (i.e., more like a *quantum program*). However, there is no clear definition how the different parts of the design flow are termed—leading to a situation where different terms are often used for the same task or where different tasks are referred to by the same term. Hence, categorizing and defining design tasks is a first important step.
- *Identifying and disseminating design problems:* Next, we identify practically relevant design problems in the quantum domain for which no or only unsatisfying solutions are available and disseminate the corresponding problem formulations and relevant models (based on the categorization and definition from the first step) to the design automation community. By this, a broad community with knowledge in clever data structures, reasoning engines, search algorithms, etc. is supplied with design tasks that often contain combinatorial problems—allowing for the development of accordingly adjusted versions of existing design tools and, thus, advancing research in the quantum domain by the establishment of *design automation for quantum computing*.
- *Developing design solutions:* Eventually, the previous two steps serve as the basis to develop design automation tools for quantum computing—the main step of this project. They will tackle practically relevant tasks that have not yet been addressed at all or only in an unsatisfactory fashion. By this tool support, the current state of the art in quantum computing is significantly advanced and it is expected to handle the increasing complexity of future quantum computers. The outcome of this objective is not supposed to serve as a stand-alone tool for quantum computing (such as IBM's *Qiskit* [17], Google's *Cirq* [18], or Microsoft's *Quantum Azure* [19]), but rather a free and open source set of tools from which certain methods can be incorporated into the stakeholder's tools.

To make this approach work, the project requires a significant amount of cooperation with stakeholders. Thus far, there is still far too little coordination between the design automation community and the quantum community. Consequently, many design approaches proposed in the past have either addressed the "wrong problems" or failed to reach the end users (as partially discussed above). The project aims for addressing this issue by building a bridge between both "sides": providing techniques and new ideas from the design automation domain, but also expertise and insights from the quantum domain. To this end, a broad network of partners and contacts has been established (see Fig. 1) which is used for the development of a common terminology as well as taking the role of an interpreter or mediator between both communities.

III. Selected Results

In the following, selected results which have been obtained in the first two years of the project are briefly presented. To this end, the corresponding design tasks are briefly reviewed first. Afterwards, a brief sketch shows how design automation expertise has been applied to address these tasks. All solutions have been made available as open-source implementations. A brief Python code snippet for each design task shows how easy the resulting methods can be applied.

For more details, references for further reading are provided. Moreover, further design tasks considered in the project are described online at https://www.cda.cit.tum.de/research/quantum/. Finally, all methods developed in the projects thus far are integrated into the *Munich Quantum Toolkit (MQT)*—a collection of design automation tools and software for quantum computing. The overarching objective of this toolkit is to provide solutions for design tasks across the entire quantum

Fig. 1: Partners and contacts

software stack, which are particularly driven by expertise from the design automation community. More information about the MQT is available at https://www.cda.cit.tum.de/research/quantum/mqt/.

A. Quantum Circuit Simulation

Simulating quantum circuits on a classical machine is essential for the development of (prototypes of) quantum algorithms, for considering the behavior of physical quantum computers, as well as for studying error models. In fact, classical simulations of quantum computations provide deeper insights, since they allow one to observe the individual amplitudes of a quantum state (which is impossible when executing a quantum computation on a quantum device). Moreover, costly executions on real quantum devices—whose availability is still rather limited—can be avoided when using simulators.

The classical simulation of quantum circuits is commonly conducted by performing consecutive matrix-vector multiplication, which many simulators realize by storing a dense representation of the complete state vector in memory and evolving it correspondingly (see, e.g., [20]–[24]). This approach quickly becomes intractable due to the exponential growth of the quantum state with respect to the number of qubits—quickly rendering such simulations infeasible even on supercomputing clusters. Simulation methodologies based on decision diagrams [25]–[27] are a promising complementary approach that frequently allows to reduce the required memory by exploiting redundancies in the simulated quantum state.

MQT offers the decision diagram-based quantum circuit simulator *DDSIM* (available at https://github.com/cda-tum/mqt-ddsim and https://pypi.org/project/mqt.ddsim/) [27]–[33] that simulates quantum circuits defined in various formats such as OpenQASM or Qiskit Quantum Circuits.

Example 1: The following listing shows how to use DDSIM as a backend for IBM Qiskit for the 3-qubit GHZ state.

```
from qiskit import *
from mqt import ddsim
circ = QuantumCircuit(3)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
provider = ddsim.DDSIMProvider()
backend = backend.get_backend('qasm_simulator')
job = execute(circ, backend, shots=10000)
print(job.result().get_counts(circ))
```

B. Quantum Circuit Mapping

Since several quantum computers in the NISQ era are bound by connectivity constraints, only support a limited set of elementary gates, and are heavily affected by noise, high-level descriptions of quantum algorithms have to be *compiled* through different layers of abstraction before being executable on the actual quantum computer. A major part of this *compilation* flow consists of *mapping*, i.e., making an already decomposed circuit conform to the device's connectivity constraints (usually provided as a coupling map).

A circuit is typically *mapped* to the actual device by inserting $SWAP$ operations into the circuit—dynamically permuting the location of the circuit's logical qubits on the device's physical qubits such that each operation conforms to the coupling map. Due to the inherent influence of noise and short coherence times of today's quantum computers, it is of utmost importance to keep the overhead induced by the mapping procedure as low as possible. As this problem has been shown to be NP-complete [14], there is a high demand for automated and efficient solutions.

MQT offers the quantum circuit mapping tool *QMAP* (available at https://github.com/cda-tum/mqt-qmap and https://pypi.org/project/mqt.qmap/) that allows one to generate circuits which satisfy all constraints given by the targeted architecture and, at the same time, keep the overhead in terms of additionally required quantum gates as low as possible. More precisely, different approaches based on design automation techniques are provided, which are generic and can be easily configured for future architectures. Among them is a general solution for arbitrary circuits based on informed-search algorithms [34], [35] as well as a solution for obtaining mappings ensuring minimal overhead with respect to SWAP gate insertions [36], [37].

Example 2: Assume we want to perform the computation simulated in Ex. 1 on the five-qubit IBMQ London quantum computer. Then, mapping the circuit to that device merely requires the following lines of Python:

```
from mqt import qmap
from qiskit.providers.fake_provider import FakeLondon
backend = FakeLondon()
circ_mapped, results = qmap.compile(circ, backend)
```

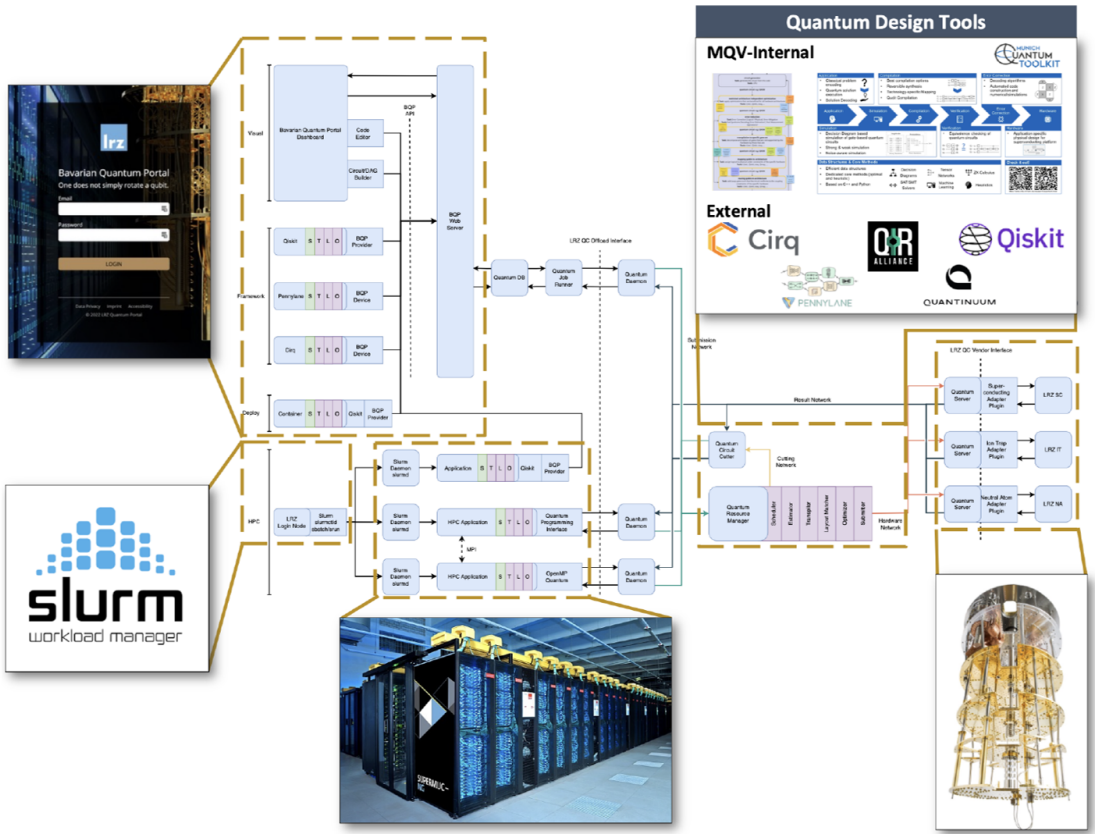

Fig. 2: The Munich Quantum Software Stack [38]

C. Quantum Circuit Verification

Compiling quantum algorithms results in different representations of the considered functionality, which significantly differ in their basis operations and structure but are still supposed to be functionally equivalent. Consequently, checking whether the originally intended functionality is indeed maintained throughout all these different abstractions becomes increasingly relevant in order to guarantee an efficient, yet correct design flow. Existing solutions for equivalence checking of quantum circuits suffer from significant shortcomings due to the immense complexity of the underlying problem—which has been proven to be QMA-complete [16]. However, certain quantum mechanical characteristics provide impressive potential for efficient equivalence checking of quantum circuits.

MQT offers the quantum circuit equivalence checking tool *QCEC* (available at https://github.com/cda-tum/mqt-qcec and https://pypi.org/project/mqt.qcec/) which explicitly exploits these characteristics based on the ideas outlined in [39]–[41].

Example 3: Verifying that the quantum circuit from Ex. 1 has been compiled correctly in Ex. 2 merely requires the following lines of Python:

```
from mqt import qcec
results = qcec.verify(circ, circ_mapped)
print(results)
```

The available methods include a strategy especially suited for verifying compilation results [42], as well as dedicated random stimuli generation schemes [43].

IV. Selected Outreach Activities

To "get the word out" and, indeed, "build a bridge" between the design automation community and the quantum computing community, numerous outreach activities have been conducted in the project. This includes several tutorials, special sessions, panel organizations, and more in conferences from both communities (such as DATE, DAC, Quantum Week, etc.). Besides that, the project has actively reached out to partners and contacts (see above), is represented in committees and advisory boards, and initiated as well as supported a significant number of initiatives. In the following, two selected examples of the project's outreach activities are briefly presented.

A. Munich Quantum Software Stack

Eventually, to make quantum computing a success, a sophisticated software stack is required which is able to connect the end users (usually domain experts from the respective application areas) with the experimentalists who provide the actual quantum computing hardware platforms. However, the development of such a software stack is far from trivial: end users expect to use the respective platforms without having to understand the specific physical underpinnings, while platform developers, who are typically physical experimentalists,

assume that quantum circuits developed to be executed on their respective platforms follow specific physical constraints and are aware of system specific constraints and conditions. Any software stack must be able to act as a "glue" between these two worlds, which keeps everything together.

The *Munich Quantum Valley*—a huge quantum computing initiative composed of over 300 researchers from various domains such as physics, engineering, computer science, domain experts, etc.) aims at developing such a software stack in a holistic and full-stack manner. The currently developed stack will support direct access for experiments, integration with HPC systems, and driving implicit quantum compilation as well as optimization toolkits which, eventually, enable multiple quantum computer backends.

The currently developed approach is sketched in Fig. 2 (see previous page, taken from [38]). Users can access the system either via a dedicated portal with multiple language backends (top left) or via HPC systems using traditional schedulers like SLURM (bottom left), which drive hybrid applications capable of offloading parts of their computation to quantum computing backends. The relevant parts of the programs are then processed via a quantum resource manager as well as corresponding quantum design tools (top-right). Finally, the resulting quantum circuits are executed by a suitable backend system (bottom right). This workflow is opaque for the user, hides the particular complexities, and still enables "expert paths" for experimental computations.

Obviously, the tools developed within this project perfectly address the needs of the quantum design tools sketched in the top-right corner of Fig. 2. Accordingly, the project heavily works together with the MQV-initiative—contributing to one of the world's leading software stacks for quantum computing.

B. Munich Quantum Software Forum

In October 2023, the Munich Quantum Software Forum brought the "who's who" in quantum computing software together for a two-days exchange meeting. The forum featured renowned representatives from academia and industry who presented existing software tools as well as recent developments, including:

- Leon Stok (IBM) covering Qiskit
- Austin Fowler (Google) covering Stim, Pymatching, and Scinter
- Mathias Soeken (Microsoft) covering Azure Quantum
- Eric Kessler (Amazon) covering Amazon Braket
- Ross Duncan (Quantinuum) covering TKET
- Fred Chong (UChicago) covering the ColdQuanta platform
- Costin Iancu (Lawrence Berkeley National Laboratory) covering BSQKit
- Ivana Kurečić (Xanadu) covering PennyLane
- Laura Schulz (LRZ) covering the Munich Quantum Ecosystem
- Lukas Burgholzer (TU Munich) covering the Munich Quantum Toolkit (MQT)

Fig. 3: Munich Quantum Software Forum

Additionally, 16 further software tools and initiatives were presented through brief pitch presentations and poster sessions. A video available at https://youtu.be/x99N7uOKJ1U provides a brief summary of the event.

Overall, the event attracted more than 200 participants covering the entire spectrum of the community including computer scientists, physicists, engineers, and mathematicians, representatives from universities and research centers but also start-ups and established companies, graduate students, PhD students, and postdocs but also junior developers, senior developers, and managers, as well as long-term quantum computing "veterans" and beginners.

Video recordings, presentation slides, as well as posters from the event are available at https://www.cda.cit.tum.de/research/quantum/2023_mqsf_summary/.

V. Conclusions

Quantum computing is becoming a reality and the corresponding complexity of its applications demands for sophisticated design and software solutions. In the past two years, the ERC Consolidator project "Design Automation for Quantum Computing" has worked towards providing the basis for that. To this end, it explicitly tries to build a bridge between the design automation community and the quantum computing community. The results obtained thus far showcase the potential of design automation for this emerging technology and the corresponding outreach activities confirm its potential. It is the goal of this project to eventually establish design automation for quantum computing—a goal for which the current results provide a great basis and which shall be completed in the remaining duration of the project.

Acknowledgments

Huge thanks to Lukas Burgholzer, Lucas Berent, Jagatheesan Kunasaikaran, Kevin Mato, Tom Peham, Nils Quetschlich, Damian Rovara, Aaron Sander, Ludwig Schmid, and Daniel Schönberger who form the quantum computing team at the Chair for Design Automation at the Technical University of Munich and who tirelessly work to realize the vision of this project! Furthermore, many thanks to all our partners and everyone who supported us during this journey as well as everyone that contributed to the development of the *Munich Quantum Toolkit*!

Special thanks go to Alwin Zulehner, Stefan Hillmich, Hartwig Bauer, Sarah Schneider, Smaran Adarsh, Alexander Ploier, and Thomas Grurl for their specific contributions in the past!

Obviously, this work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001318).

REFERENCES

[1] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Theory of computing*, 1996.

[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.

[4] E. Gerjuoy, "Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers," *American Journal of Physics*, vol. 73, no. 6, pp. 521–540, 2005.

[5] A. Montanaro, "Quantum algorithms: An overview," *NPJ Quantum Information*, vol. 2, p. 15 023, 2016.

[6] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[7] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, *et al.*, "Quantum algorithm implementations for beginners," *arXiv:1804.03719*, 2018.

[8] M. H. Devoret and R. J. Schoelkopf, "Superconducting circuits for quantum information: An outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, 2013.

[9] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021 318, 2019.

[10] H. Haeffner, C. F. Roos, and R. Blatt, "Quantum computing with trapped ions," *Physics Reports*, vol. 469, no. 4, pp. 155–203, 2008.

[11] F. Bernardini, A. Chakraborty, and C. Ordóñez. "Quantum computing with trapped ions: A beginner's guide." arXiv: 2303 . 16358 [cond-mat, physics:physics, physics:quant-ph]. (2023), [Online]. Available: http://arxiv.org/abs/2303.16358 (visited on 05/05/2023), preprint.

[12] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020.

[13] K. Wintersperger, F. Dommert, T. Ehmer, A. Hoursanov, J. Klepsch, W. Mauerer, *et al.* "Neutral atom quantum computing hardware: Performance and end-user perspective." arXiv: 2304.14360 [quant-ph]. (2023), [Online]. Available: http://arxiv.org/abs/2304.14360 (visited on 05/05/2023), preprint.

[14] A. Botea, A. Kishimoto, and R. Marinescu, "On the complexity of quantum circuit compilation," in *Symp. on Combinatorial Search*, 2018.

[15] M. Soeken, R. Wille, O. Keszocze, D. M. Miller, and R. Drechsler, "Embedding of large boolean functions for reversible logic," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 4, p. 41, 2016.

[16] D. Janzing, P. Wocjan, and T. Beth, ""Non-identity check" is QMA-complete," *Int'l Journal of Quantum Information*, 2005.

[17] H. Abraham *et al.*, *Qiskit: An open-source framework for quantum computing*, 2019.

[18] *Cirq*, https://github.com/quantumlib/Cirq, Accessed: 2020-07-22, 2020.

[19] *Microsoft Quantum Development Kit*, https://www.microsoft.com/en-us/quantum/development-kit, Accessed: 2020-07-22, 2020.

[20] T. Häner and D. S. Steiger, "0.5 petabyte simulation of a 45-Qubit quantum circuit," in *Int'l Conf. for High Performance Computing, Networking, Storage and Analysis*, 2017.

[21] J. Doi, H. Takahashi, R. Raymond, T. Imamichi, and H. Horii, "Quantum computing simulator on a heterogenous HPC system," in *Int'l Conf. on Computing Frontiers*, 2019, pp. 85–93.

[22] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "QuEST and high performance simulation of quantum computers," in *Scientific Reports*, 2018.

[23] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. D. Sawaya, "Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, 2020.

[24] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Int'l Conf. for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–24.

[25] G. F. Viamontes, I. L. Markov, and J. P. Hayes, *Quantum Circuit Simulation*. Springer, 2009.

[26] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient Quantum Function Representation and Manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 86–99, 2016.

[27] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019.

[28] A. Zulehner and R. Wille, "Matrix-vector vs. matrix-matrix multiplication: Potential in DD-based simulation of quantum computations," in *Design, Automation and Test in Europe*, 2019.

[29] S. Hillmich, I. L. Markov, and R. Wille, "Just like the real thing: Fast weak simulation of quantum computation," in *Design Automation Conf.*, 2020.

[30] S. Hillmich, R. Kueng, I. Markov, and R. Wille, "As accurate as needed, as efficient as possible: Approximations in dd-based quantum circuit simulation," in *Design, Automation and Test in Europe*, 2021.

[31] T. Grurl, R. Kueng, J. Juß, and R. Wille, "Stochastic quantum circuit simulation using decision diagrams," in *Design, Automation and Test in Europe*, 2021.

[32] L. Burgholzer, H. Bauer, and R. Wille, "Hybrid Schrödinger-Feynman simulation of quantum circuits with decision diagrams," in *Int'l Conf. on Quantum Computing and Engineering*, 2021.

[33] L. Burgholzer, A. Ploier, and R. Wille, "Simulation paths for quantum circuit simulation with decision diagrams: What to learn from tensor networks, and what not," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2022. arXiv: 2203.00703.

[34] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2018.

[35] S. Hillmich, A. Zulehner, and R. Wille, "Exploiting quantum teleportation in quantum circuit mapping," in *Asia and South Pacific Design Automation Conf.*, 2021.

[36] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Design Automation Conf.*, 2019.

[37] L. Burgholzer, S. Schneider, and R. Wille, "Limiting the search space in optimal quantum circuit mapping," in *Asia and South Pacific Design Automation Conf.*, 2022.

[38] M. Schulz, L. Schulz, M. Ruefenacht, and R. Wille, "Towards the Munich Quantum Software Stack: Enabling efficient access and tool support for quantum computers," in *Int'l Conf. on Quantum Computing and Engineering*, 2023, pp. 399–400.

[39] L. Burgholzer and R. Wille, "Advanced equivalence checking of quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2021.

[40] T. Peham, L. Burgholzer, and R. Wille, "Equivalence checking of parameterized quantum circuits: Verifying the compilation of variational quantum algorithms," in *Asia and South Pacific Design Automation Conf.*, 2023.

[41] T. Peham, L. Burgholzer, and R. Wille, "Equivalence checking of quantum circuits with the ZX-Calculus," 2022.

[42] L. Burgholzer, R. Raymond, and R. Wille, "Verifying results of the IBM Qiskit quantum circuit compilation flow," in *Int'l Conf. on Quantum Computing and Engineering*, 2020.

[43] L. Burgholzer, R. Kueng, and R. Wille, "Random stimuli generation for the verification of quantum circuits," in *Asia and South Pacific Design Automation Conf.*, 2021.