The Basis of Design Tools for Quantum Computing:

Arrays, Decision Diagrams, Tensor Networks, and ZX-Calculus

(Invited Paper)

Robert Wille[1,2*], Lukas Burgholzer[3*], Stefan Hillmich[3*], Thomas Grurl[3], Alexander Ploier[3], Tom Peham[3]
[1] Chair for Design Automation, Technical University of Munich, Germany
[2] Software Competence Center Hagenberg (SCCH) GmbH, Austria
[3] Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
[*]Corresponding Authors: robert.wille@tum.de, lukas.burgholzer@jku.at, stefan.hillmich@jku.at
https://www.cda.cit.tum.de/research/quantum/

***Abstract*—Quantum computers promise to efficiently solve important problems classical computers never will. However, in order to capitalize on these prospects, a fully automated quantum software stack needs to be developed. This involves a multitude of complex tasks from the classical simulation of quantum circuits, over their compilation to specific devices, to the verification of the circuits to be executed as well as the obtained results. All of these tasks are highly non-trivial and necessitate efficient data structures to tackle the inherent complexity. Starting from rather straight-forward arrays over decision diagrams (inspired by the design automation community) to tensor networks and the ZX-calculus, various complementary approaches have been proposed. This work provides a look "under the hood" of today's tools and showcases how these means are utilized in them, e.g., for simulation, compilation, and verification of quantum circuits.**

***Index Terms*—quantum computing, data structures, arrays, decision diagrams, tensor networks, ZX-calculus**

I. Introduction

We are at the dawn of a new computing age in which quantum computers will find their way into practical applications such as cryptography [1], chemistry [2], medicine [3], physics [4], finance [5], and machine learning [6]. In many instances, quantum computing is believed to provide efficient solutions for problems which are out of reach for classical computers. Besides the ongoing discovery of new potential applications, the capabilities of currently available quantum computers are rapidly improving as, e.g., witnessed by IBM's ambitious road map for scaling quantum technology to more than 1000 qubits by 2023 [7].

Due to an increased number of qubits with increased coherence time as well as faster operations with higher fidelity, increasingly large quantum circuits can reliably be executed on actual devices. With this increase in computational power comes the need for corresponding software solutions and tools that aid users and developers in making best use of the available hardware. Similar to the design of classical circuits and systems, realizing conceptual quantum algorithms on actual devices requires a multitude of complex design tasks. Some of the most important tasks are:

- *Classical simulation*: Simulating the execution of a quantum circuit on classical computers is an extremely important task in the development and testing of new applications and use cases. In addition to lower costs, it offers detailed insights on the quantum state during the execution of a quantum circuit that is physically unavailable when running the circuit on an actual quantum computer [8]–[13].
- *Compilation*: Similar to classical circuits and systems, quantum circuits are initially described at a rather high abstraction level and need to be compiled to a representation that adheres to all the constraints imposed by the target device (e.g., limited gate-set and/or limited connectivity) [14]–[18].
- *Verification*: Since compilation significantly changes the structure of quantum circuits, it is crucial to ensure that the resulting circuits still realize the originally intended functionality. To this end, verification (or, more precisely, equivalence checking) methods are employed to guarantee equivalence [17], [19]–[25].

Either due to the inherent exponential size of the underlying representations of quantum states and operations or the huge amount of degrees of freedom, each of these design tasks represents a computationally hard challenge. Consequently, efficient data structures and methods are needed to tackle these challenges. In this work, we provide a brief overview of various complementary data structures that have been proposed in the past and briefly discuss how each of them has been used to efficiently solve the above mentioned design tasks. With this, we hope to provide the interested reader with an intuition on the different kinds of approaches available and the necessary pointers to dive deeper into the wide range of possible methods and solutions.

The rest of this work is structured as follows: Section II reviews the basics of quantum computing and shows how quantum states and operations are represented as one- and two-dimensional arrays in a straight-forward, yet hardly efficient fashion. Section III introduces decision diagrams which enable representing quantum states and functionality in a more compact fashion in many cases by exploiting redundancies in the underlying representations. Section IV covers the basics of tensor networks which, instead of capitalizing on redundancies in the underlying representations, take advantage of the topological structure of certain quantum states and algorithms. Section V demonstrates how the ZX-calculus—a graphical notation for quantum circuits equipped with a

powerful set of rewrite rules—enables diagrammatic reasoning about quantum computing. Finally, Section VI concludes the paper.

II. Arrays

In quantum computing, vectors and matrices are often considered to be the most intuitive data structure for representing quantum objects. These structures can be directly realized using arrays and can be used for design automation tasks. Here, we introduce this data structure along with a brief introduction to quantum computing. The interested reader can find an in-depth introduction in [26].

Similar to classical bits, *quantum bits* (*qubits*) can assume the states 0 or 1. These are are called computational basis states and—using Dirac notation—written as $|0\rangle$ and $|1\rangle$. Additionally, they can also assume an (almost) arbitrary linear combination (i.e., a *superposition*) of these two basis states. More precisely, the state of a qubit $|\psi\rangle$ is given by $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$, with $\alpha_0, \alpha_1 \in \mathbb{C}$ such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The two factors α_0 and α_1 are the *amplitudes* and denote how much the qubit is related to each of the two basis states. Measuring a qubit returns 0 with probability $|\alpha_0|^2$ and 1 with probability $|\alpha_1|^2$, respectively. The individual amplitudes in a qubit are fundamentally not observable and measurements are the only way to extract information out of a qubit.

The concepts of a single qubit can be generalized to describe states composed of multiple qubits—commonly referred to as *quantum registers*. An n qubit register can assume 2^n basis states and is described by amplitudes $\alpha_0, \alpha_1, \dots \alpha_{2^n-1}$, which must satisfy the normalization constraint $\sum_{i \in \{0,1\}^n} |\alpha_i|^2 = 1$. Quantum states are often shortened to state vectors containing only the amplitudes, e.g., $\begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{10} & \alpha_{11} \end{bmatrix}^{\mathrm{T}}$ for two qubits.

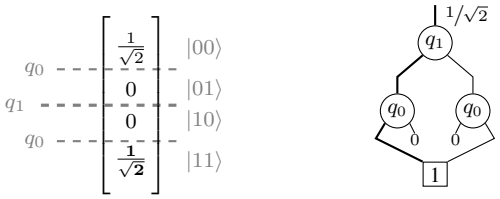
Quantum states can be manipulated using quantum operations. Quantum operations are inherently reversible and are described by unitary matrices. They are applied to quantum states by matrix-vector multiplication. Important single-qubit operations include the NOT $= \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ operation, which negates the state of a qubit, and the Hadamard operation H $= 1/\sqrt{2}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$, which transforms a qubit from a basis state into a superposition. There are also multi-qubit operations. The most prominent two-qubit operation is the controlled-NOT operation (CNOT), which negates the state of its target qubit iff the control qubit is in state $|1\rangle$.

Example 1. *Consider the quantum register $|\psi\rangle$ composed of two qubits, which is in the state $1/\sqrt{2} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$. Applying a* CNOT *operation with control on the first and target on the second qubit yields the output state $|\psi'\rangle$ determined by*

$$\underbrace{\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{bmatrix}}_{\text{CNOT}} \cdot \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{|\psi\rangle} = \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{|\psi'\rangle}.$$

Measuring $|\psi'\rangle$ (also known as Bell state*) collapses the state and returns $|00\rangle$ or $|11\rangle$, each with probability $|1/\sqrt{2}|^2 = 1/2$.*

The concepts reviewed above can be realized in a straightforward fashion: Vectors and matrices are described in terms of 1-dimensional and 2-dimensional arrays, respectively. While such a representation has huge potential for concurrent execution, it incurs a huge memory footprint, since the involved arrays growth exponentially with each considered qubit. As a consequence, these memory requirements limit array-based simulation methods to rather small/moderate quantum computations (today's practical limit is less than 50 qubits [27]).



(a) Vector (b) Decision diagram

Fig. 1. Different representations of the Bell state

III. Decision Diagrams

The general idea of decision diagrams [28], [29] is about uncovering and exploiting redundancies within the involved quantum states and operations. More precisely, consider a quantum register composed of n qubits $q_{n-1}, \dots, q_1, q_0$, where q_{n-1} represents the most significant qubit. The first 2^{n-1} entries of the corresponding state vector represent amplitudes for basis states where q_{n-1} is $|0\rangle$ and the remaining 2^{n-1} entries represent amplitudes where q_{n-1} is $|1\rangle$. This is represented in a decision diagram by a node labeled q_{n-1} connected to two successor nodes labeled q_{n-2}, representing the zero- and one-successor. This process is repeated recursively until sub-vectors of size 1 (i.e., individual complex numbers) remain, which are connected to terminal nodes.

During this decomposition process, equivalent sub-vectors are represented by the same node—reducing the overall size of the decision diagram. Furthermore, instead of having distinct terminal nodes for all amplitudes, edge weights are used to store common factors of the amplitudes. Having encoded a state vector into a decision diagram, specific amplitudes can be reconstructed multiplying the edge weights along the corresponding path. To improve the readability of decision diagrams, edge weights of 1 are typically omitted from the visualization and nodes with an incoming edge weight of zero are shown as 0-stubs to indicate that the whole sub-part is zero.

Example 2. *Fig. 1 depicts the quantum register $|\psi'\rangle$ in both, the vector and the decision diagram representation. The annotations of the state vector in Fig. 1a indicate how the corresponding decision diagram is constructed. In order to reconstruct specific amplitudes from the decision diagram, the edge weights of the corresponding path need to be multiplied. For example, reconstructing the amplitude of the state $|00\rangle$ (bold line in the figure) requires multiplying the edge weight of the root edge ($1/\sqrt{2}$) with the right edge of q_1 (1) as well as q_0 (1), i.e. $1/\sqrt{2} \cdot 1 \cdot 1 = 1/\sqrt{2}$.*

Decision diagram representation of matrices are constructed in an analogous fashion to vectors, decomposing the matrix recursively into quarters instead of halves. Just as the underlying

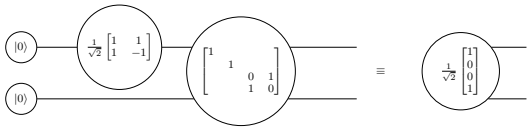


Fig. 2. Tensor network representation of the quantum circuit to create the Bell state

vectors and matrices, decision diagrams support multiplication and addition, enabling their usage in different design automation tasks, such as quantum circuit simulation (e.g., [9]) or equivalence checking (e.g., [20]). A web-based visualizing tool providing an intuition of decision diagrams is available at https://iic.jku.at/eda/research/quantum_dd/tool/ [30].

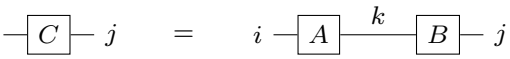
IV. Tensor Networks

Tensor networks can help alleviate the complexity of the array-based simulation by exploiting redundancies in the topological structure of the quantum circuit [31], [32]. To translate a quantum circuit into a tensor networks, each object, be it a state or a operation, is represented by a multidimensional array of complex numbers, a *tensor*, connecting to other tensors according to the underlying quantum circuit. The extraction of useful information from such a network then typically requires the pairwise contraction of tensors into a single remaining tensor.

Example 3. *Let A, B, C be matrices in $\mathbb{C}^{N\times N}$. Further, let the matrix product $C = AB$ be given by*

$$C_{i,j} = \sum_{k=0}^{N-1} A_{i,k} B_{k,j},$$

with $i, j = 0, \ldots, N-1$. Then, this corresponds to the contraction of the rank-2 tensors $A = [A_{i,k}]$ and $B = [B_{k,j}]$ over the shared index k. This is conveniently represented graphically as:



The order in which all the tensors are contracted is called *contraction plan*. The main goal of such a plan is to keep the intermediate tensors and their dimension of contracted indices (also referred to as *bond dimension*) during the computation in check—a task proven to be NP-hard [33]. Therefore, a plethora of methods have been developed to efficiently determine suitable contraction plans [34].

Example 4. *Consider again the Bell state from Fig. 1a. Fig. 2 shows how this translates to a tensor network. Each individual tensor is illustrated by a "bubble" containing the actual data of the tensor. This representation only requires a linear amount of memory with regard to the total number of qubits and gates (in contrast to the exponential representation in the array-based method). The final state vector, on the other hand, still is of size 2^n, where n denotes the number of qubits in the system.*

As shown by the example, the computation of the complete output state vector with tensor networks is generally infeasible. Different specialized types of tensor networks have been proposed to alleviate that complexity by imposing certain structures by decomposing the whole state into smaller tensors (see [35] and the references therein).

(a) Bell circuit (b) Bell state (c) Graph-like diagram

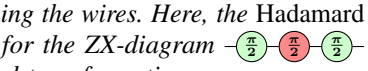
Fig. 3. ZX-diagrams for the Bell state

This is used, e.g., in classical quantum circuit simulation, where it is desirable to determine a single scalar quantity, such as the expected value of some observable or an individual amplitude. Methods based on tensor networks accomplish this by fixing the output indices of the circuit's tensor network, i.e., adding "bubbles" at the end of the circuit. Contracting this network results in a single rank-0 tensor—a scalar. Whenever the size and bond dimension of intermediate tensors can be kept in check, this can be done extremely efficient.

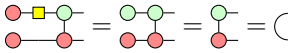
V. ZX-Calculus

The ZX-calculus [36], [37] is a graphical notation for quantum circuits equipped with a powerful set of rewrite rules that enable diagrammatic reasoning about quantum computing. A ZX-diagram is made up of colored nodes (called *spiders*) that are connected by wires. Each spider can either be green (Z-spider ○) or red (X-spider ●) and is optionally attributed a scalar phase. Spiders without inputs are called *states*, whereas spiders with no outputs are called *effects*.

An important concept for ZX-diagrams is the *only connectivity matters* paradigm, which expresses the fact that two ZX-diagrams are considered equal if one can be transformed into the other simply by bending wires.

Example 5. *Consider the Bell circuit in Fig. 3a. It is equivalent to the ZX-diagram*  *because they can be transformed into each other by (un-)crossing the wires. Here, the* Hadamard box *is a short notation for the ZX-diagram*  *and represents the Hadamard transformation.*

To see how this circuit acts on the  $= |0\rangle$ *states, we can plug them into the ZX-diagram and simplify with the ZX-calculus:*



At the end, the Bell state shown in Fig. 3b is obtained.

Any quantum circuit can be interpreted as a ZX-diagram. ZX-diagrams are more general than quantum circuits however, and allow for representations that do not have meaningful interpretations as quantum circuits. It is this flexibility of being able to leave the quantum circuit formalism that makes the ZX-calculus a good intermediate language when working with quantum circuits.

While the basic axiomatization of the ZX-calculus is a powerful language for quantum information theory, it is hard to apply directly in automated reasoning. The reason for this is the lack of normal-forms for ZX-diagrams—an important feature for automated rewriting. The backbone of many automated methods using the ZX-calculus is an alternate representation of ZX-diagrams using only Z-spiders and wires with Hadamard boxes, called *graph-like* ZX-diagrams. The graph-like ZX-diagram corresponding to the Bell circuit is

shown in Fig. 3c. Additional rewrite rules based on graph-theoretic simplification are defined for these graph-like diagrams enabling the definition of a terminating rewriting procedure [38].

This automated rewriting as well as the ability to efficiently work with quantum operations such as *phase polynomials* has led to many algorithms based on the ZX-calculus being used to solve problems in the design automation of quantum circuits such as compilation and optimization as well as simulation and verification [39]–[42].

VI. Conclusion

In this work, we briefly reviewed the basics of arrays, decision diagrams, tensor networks, and ZX-calculus as well as their applications in design automation for quantum computing. Each of these complementary data structures provides a certain trade-off between memory consumption, performance, and conceptional complexity. Picking the most suitable data structure for the job at hand is crucial to have an efficient workflow in design automation for quantum computing. We hope this work provides the interested reader with an intuition on the data structures and necessary pointers to further explore the wide range of possible methods and applications.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001318). It is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus and was partially supported by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, 1997.

[2] A. Kandala, A. Mezzacapo, *et al.*, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, 2017.

[3] B. A. Cordier, N. P. D. Sawaya, G. G. Guerreschi, and S. K. McWeeney, *Biology and medicine in the landscape of quantum advantages*, 2021. arXiv: 2112.00760.

[4] H.-S. Zhong, H. Wang, *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, 2020.

[5] D. Herman, C. Googin, *et al.*, *A survey of quantum computing for finance*, 2022. arXiv: 2201.02773.

[6] H.-Y. Huang, R. Kueng, *et al.*, *Provably efficient machine learning for quantum many-body problems*, 2022. arXiv: 2106.12627.

[7] J. Gambetta, "IBM's Roadmap For Scaling Quantum Technology," *IBM Research Blog*, 2020. [Online]. Available: https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/.

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

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

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

[11] S. Bravyi and D. Gosset, "Improved classical simulation of quantum circuits dominated by Clifford gates," *Phys. Rev. Lett.*, vol. 116, 25 2016.

[12] S. Hillmich, R. Kueng, I. L. 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*, 2020.

[13] T. Grurl, J. Fuß, and R. Wille, "Noise-aware quantum circuit simulation with decision diagrams," 2022.

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

[15] 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*, 2019.

[16] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, "A software methodology for compiling quantum programs," *Quantum Sci. Technol.*, vol. 3, no. 2, 2018.

[17] K. N. Smith and M. A. Thornton, "A quantum computational compiler and design tool for technology-specific targets," in *Int'l Symp. on Computer Architecture*, 2019.

[18] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, 2019.

[19] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," in *Int'l Symp. on Nanoscale Architectures*, 2010.

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

[21] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Checking equivalence of quantum circuits and states," in *Int'l Conf. on CAD*, 2007.

[22] P. Niemann, R. Wille, and R. Drechsler, "Equivalence checking in multi-level quantum systems," in *Int'l Conf. of Reversible Computation*, 2014.

[23] S.-A. Wang, C.-Y. Lu, I.-M. Tsai, and S.-Y. Kuo, "An XQDD-based verification method for quantum circuits," in *IEICE Trans. Fundamentals*, 2008.

[24] M. Amy, "Towards large-scale functional verification of universal quantum circuits," in *International Conference on Quantum Physics and Logic*, 2019.

[25] X. Hong, X. Zhou, *et al.*, *A tensor network based decision diagram for representation of quantum circuits*, 2020. arXiv: 2009.02618.

[26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[27] H. De Raedt *et al.*, "Massively parallel quantum computer simulator, eleven years later," *Computer Physics Communications*, vol. 237, 2019.

[28] P. Niemann, R. Wille, *et al.*, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2016.

[29] A. Zulehner, S. Hillmich, and R. Wille, "How to efficiently handle complex values? Implementing decision diagrams for quantum computing," in *Int'l Conf. on CAD*, 2019.

[30] R. Wille, L. Burgholzer, and M. Artner, "Visualizing decision diagrams for quantum computing," in *Design, Automation and Test in Europe*, 2021.

[31] M. Fannes, B. Nachtergaele, and R. F. Werner, "Finitely correlated states on quantum spin chains," *Commun. Math. Phys.*, vol. 144, no. 3, 1992.

[32] J. C. Bridgeman and C. T. Chubb, "Hand-waving and Interpretive Dance: An Introductory Course on Tensor Networks," *J. Phys. A: Math. Theor.*, 2017.

[33] L. Chi-Chung, P. Sadayappan, and R. Wenger, "On optimizing a class of multi-dimensional loops with reduction for parallel execution," *Parallel Process. Lett.*, 1997.

[34] J. Gray and S. Kourtis, "Hyper-optimized tensor network contraction," *Quantum*, vol. 5, 2021.

[35] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, "Matrix product states and projected entangled pair states: Concepts, symmetries, theorems," *Rev. Mod. Phys.*, vol. 93, 4 2021.

[36] J. van de Wetering, *ZX-calculus for the working quantum computer scientist*, 2020. arXiv: 2012.13966.

[37] B. Coecke and A. Kissinger, "Picturing quantum processes," in *Diagrammatic Representation and Inference*, 2018.

[38] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering, "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus," *Quantum*, vol. 4, 2020.

[39] A. Kissinger and J. van de Wetering, "Reducing T-count with the ZX-calculus," *Phys. Rev. A*, 2020.

[40] A. Kissinger and J. van de Wetering, "Simulating quantum circuits with zx-calculus reduced stabiliser decompositions," *Quantum Science and Technology*, 2022.

[41] N. de Beaudrap, X. Bian, and Q. Wang, "Techniques to reduce $\pi/4$-parity-phase circuits, motivated by the ZX calculus," *Electron. Proc. Theor. Comput. Sci.*, vol. 318, 2020.

[42] A. Cowtan, W. Simmons, and R. Duncan, *A generic compilation strategy for the unitary coupled cluster ansatz*, 2020. arXiv: 2007.10515.