Decision Diagrams for Quantum Computing

Robert Wille[1,2][0000−0002−4993−7860],
Stefan Hillmich[3][0000−0003−1089−3263], and Lukas Burgholzer[3][0000−0003−4699−1316]

[1] Chair for Design Automation, Technical University of Munich, Germany
[2] Software Competence Center Hagenberg GmbH (SCCH), Austria
[3] Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
robert.wille@tum.de   stefan.hillmich@jku.at   lukas.burgholzer@jku.at
https://www.cda.cit.tum.de/research/quantum/

Abstract. Quantum computing promises to solve some important problems faster than conventional computations ever could. Currently available NISQ devices on which first practical applications are already executed demonstrate the potential—with future fault-tolerant quantum hardware for more demanding applications on the horizon. Nonetheless, the advantages in computing power come with challenges to be addressed in the design automation and software development community. In particular, non-quantum representations of states and operations, which provide the basis, e.g., for quantum circuit simulation or verification, require an exponential amount of memory. We propose to use decision diagrams as data structure to conquer the exponential memory requirements in many cases. In this chapter, we review the fundamentals on decision diagrams and highlight their applicability in the tasks of quantum circuit simulation with and without errors as well as in verification of quantum circuits. The tools presented here are all available online as open source projects.

1   Introduction

Quantum computing promises to speed up many important applications even in the current NISQ era [1] and more so once fault-tolerance is achieved. The underlying primitives of quantum computing are fundamentally different to conventional computations. This introduces new challenges for design automation and software development such as the exponential memory requirement to store arbitrary quantum states and operations on non-quantum hardware.

The design automation community in the conventional domain has spent decades to successfully solve many difficult problems. One of these solutions which especially addresses memory consumption is the usage of decision diagrams to represent information. In the conventional domain, there exists a

plethora of different types such as *Binary Decision Diagrams* (BDDs, [2]), *Binary Moment Diagrams* (BMDs, [3]), *Zero-suppressed Decision Diagrams* (ZDDs, [4]), or *Tagged BDDs* [5]. Inspired by the results achieved with decision diagram in the conventional domain, several types have been invented for the quantum domain, such as *X-decomposition Quantum Decision Diagrams* (XQDDs, [6]), *Quantum Decision Diagrams* (QDDs, [7]), *Quantum Information Decision Diagrams* (QuIDDs, [8]), or *Quantum Multiple-valued Decision Diagrams* (QMDDs, [9], [10]). However, many researchers and engineers working in the domain of quantum computing are still rather unfamiliar with the concepts of decision diagrams and, hence, often cannot fully exploit this potential.

In this chapter, we review decision diagrams as data structure to compactly represent quantum states and quantum operators. To this end, we explain how decision diagrams are obtained from decomposing state vectors along with an explanation of the graphical notation. The vector decomposition is subsequently extended to obtain decision diagrams for matrices. Afterwards, we cover selected applications of decision diagrams for design and validation work. More precisely, we begin by covering error-free quantum circuit simulation, which is essentially matrix-vector multiplication. In the next step, we discuss noisy quantum circuit simulation and the advantages decision diagrams have in this application. Additionally to simulation, we present as a main aspect of quantum circuit verification an efficient procedure to check the equivalence of quantum circuits using decision diagrams.

The remainder of this chapter is structured as follows. Section 2 gives the background on decision diagrams, specifically how vectors and matrices are represented. In Section 3 and Section 4, we show how decision diagrams can be employed to conduct quantum circuit simulation without and with noise, respectively. Section 5 explains how decision diagrams lead to more efficient equivalence checking procedures. Finally, Section 6 concludes the chapter.

2 Decision Diagrams

In this section, we describe how decision diagrams exploit redundancies in vectors and matrices to enable a compact representation in many cases. More precisely, we first detail the representation for state vectors which we subsequently extend by a second dimension to compactly represent matrices for quantum operations.

2.1 Representation of State Vectors

The representation of a system composed of n qubits on non-quantum hardware is commonly achieved through 2^n-dimensional vector—an exponential representation. However, a closer look at state vectors unveils that they are frequently composed of redundant entries which provide potential for a more compact representation.

Example 1. *Consider a quantum system with $n = 3$ qubits situated in a state given by the following vector:*

$$\psi = \left[0, 0, \tfrac{1}{2}, 0, \tfrac{1}{2}, 0, -\tfrac{1}{\sqrt{2}}, 0\right]^T.$$

Although exponential in size ($2^3 = 8$ entries), this vector only assumes three different values, namely 0, $\tfrac{1}{2}$, and $-\tfrac{1}{\sqrt{2}}$.

Redundancy in the considered data can be exploited to attain a compact representation. To this end, we propose to employ decision diagrams. For conventional computations, decision diagrams such as the *Binary Decision Diagram* (BDD, [2]) are a tried and tested data structure and have been used for decades. For BDDs, a decomposition scheme is employed which reduces a function to be represented into corresponding sub-functions. Since the sub-functions usually include redundancies as well, equivalent sub-functions result which can be shared—eventually yielding a much more compact representation. In a similar fashion, the concept of decomposition can also be applied to represent state vectors in a more compact fashion.

Similar to decomposing a function into sub-functions, we decompose a given state vector with its complex entries into sub-vectors. To this end, consider a quantum system with qubits $q_{n-1}, q_1, \ldots q_0$, whereby q_{n-1} represents the most significant qubit.[4] Then, the first 2^{n-1} entries of the corresponding state vector represent the amplitudes for the basis states with q_{n-1} set to $|0\rangle$; the other entries represent the amplitudes for states with q_{n-1} set to $|1\rangle$. This decomposition is represented in a decision diagram structure by a node labeled q_{n-1} and two successors leading to nodes representing the sub-vectors. The sub-vectors are recursively decomposed further until vectors of size 1 (i.e., a complex number) result. This eventually represents the amplitude α_i for the complete basis state and is given by a terminal node. During this decomposition, equivalent sub-vectors are represented by the same nodes—enabling sharing and, hence, a reduction of the complexity of the representation. An example illustrates the idea.

Example 2. *Consider again the quantum state from Example 1. Applying the decomposition described above yields a decision diagram as shown in Fig. 1a. The left (right) outgoing edge of each node labeled q_i points to a node representing the sub-vector with all amplitudes for the basis states with q_i set to $|0\rangle$ ($|1\rangle$). Following a path from the root to the terminal yields the respective entry. For example, following the path highlighted bold in Fig. 1a provides the amplitude for the basis state with $q_2 = |1\rangle$ (right edge), $q_1 = |1\rangle$ (right edge), and $q_0 = |0\rangle$ (left edge), i.e., $-\tfrac{1}{\sqrt{2}}$ which is exactly the amplitude for basis state $|110\rangle$ (seventh entry in the vector from Example 1). Since some sub-vectors are equal (e.g., $\left[\tfrac{1}{2}, 0\right]^T$ represented by the left node labeled q_0), sharing is possible.*

---

[4] The terminology *most-significant qubit* refers to a position in the basis states of a quantum system and does not signify the importance of the qubit itself.

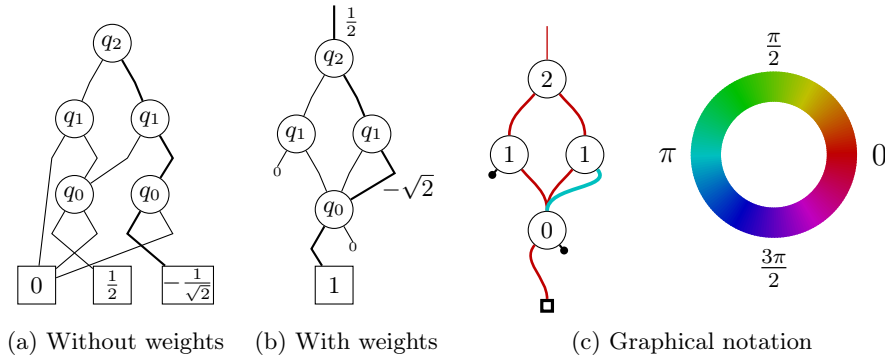(a) Without weights   (b) With weights   (c) Graphical notation

Fig. 1: Different representations of the state vector from Example 1

However, there is more potential for sharing. In fact, many entries of the state vectors differ by a common factor only (e.g., the state vector from Example 1 has entries $\frac{1}{2}$ and $-\frac{1}{\sqrt{2}}$ which differ by the factor $-\sqrt{2}$). This is exploited in the decision diagram representation by denoting common factors of amplitudes as weights to the edges of the decision diagram. Then, the value of an amplitude for a basis state is determined by not only following the path from the root to the terminal, but additionally multiplying the weights of the edges along this path. Note that for a more readable notation, we use zero stubs to indicate zero vectors (i.e., vectors only containing zeroes) and omit edge weights that are equal to one. Again, an example illustrates the idea.

Example 3. *Consider again the quantum state from Example 1 and the corresponding decision diagram shown in Fig. 1a. As can be seen, the sub-graphs rooting the node labeled q_0 are structurally equivalent and only differ in their terminal values. Moreover, they represent sub-vectors $\left[\frac{1}{2}, 0\right]^T$ and $\left[-\frac{1}{\sqrt{2}}, 0\right]^T$ which only differ in a common factor.*

In the decision diagram shown in Fig. 1b, both sub-graphs are merged. This is possible since the corresponding value of the amplitudes is now determined not by the terminals, but the product of weights on the respective paths. As an example, consider again the path highlighted bold representing the amplitude for the basis state $|110\rangle$. Since this path includes the weights $\frac{1}{2}$, 1, $-\sqrt{2}$, and 1, an amplitude value of $\frac{1}{2} \cdot 1 \cdot (-\sqrt{2}) \cdot 1 = -\frac{1}{\sqrt{2}}$ results.

There exist various possibilities to factorize an amplitude. Hence, we apply a normalization scheme to the decision diagrams, resulting in a representation which is canonical w.r.t order of qubits [9]. The outgoing edges of a node are often normalized by dividing both weights by the weight of the left-most edge (when $\neq 0$), and multiplying this factor to the incoming edges. However, it has been found in [11], that it is more effective to divide by the norm of the vector containing both edge weights and adjust the incoming edges accordingly. This normalizes the sum of the squared magnitudes of the outgoing edge weights to 1

and is consistent with the quantum semantics, where basis states $|0\rangle$ and $|1\rangle$ are observed after measurement with probabilities that are squared magnitudes of the respective weights. Furthermore, to ease the graphical notation we represent the complex number in polar plane as $r \cdot \mathrm{e}^{\mathrm{i}\alpha}$. The magnitude r of an edge weight is represented by the edge's thickness and the angle α according to the HLS color wheel [12]. The graphical notation reflects that one is most often only interested in the structure of the decision diagram instead of the exact values of edge weights. Of course, the edge weights can be put in the notation if necessary.

Example 4. *Consider again the quantum state from Example 1 and the normalized decision diagram with edge weights shown in Fig. 1b. Fig. 1c shows the graphical notation of this decision diagram where the line width represents the magnitude of the edge weight and the color the respecting angle when considering the polar notation of complex numbers.*

In Fig. 1c, the edge to the root node (having a weight of $1/2$) is notably thinner than the other edges (with weights 1 and $-\sqrt{2}$). The with weight $-\sqrt{2}$ is slightly thicker than the edges with weight 1 and, more visible in the figure, has a different phase, i.e., $-\sqrt{2} = \sqrt{2} \cdot e^{i\pi}$, encoded in the line color.

Overall, the discussions from above lead to the following definition of decision diagrams for quantum states.

Definition 1. *The decision diagram representing a 2^n-dimensional state vector is a directed acyclic multi-graph with one terminal node labeled 1 that has no successors and represents a 1-dimensional vector with the element 1. All other nodes are labeled q_i, $0 \leq i < n$ (representing a partition over qubit q_i) and have two successors. Additionally, there is an edge pointing to the root node of the decision diagram. This edge is called* root edge. *Each edge of the graph has a complex number attached as weight. An entry of the state vector is then determined by the product of all edge weights along the path from the root towards the terminal. Without loss of generality, the nodes of the decision diagram are ordered by the significance of their label, i.e., the successor of a node labeled q_i are labeled with a less significant qubit q_j. Finally, the nodes are normalized, which means that the sum of the squared magnitudes of the outgoing edge weights equals one and the common factor is propagated upwards in the decision diagram.*

2.2   Representation of Matrices

While quantum states are commonly represented by vectors, quantum operations are described by matrices. These matrices are unitary (its conjugate transpose is also its inverse) and $2^n \times 2^n$-dimensional for a n-qubit system. Similar to state vectors, matrices often include redundancies, which can be exploited for a more compact representation. To this end, the decomposition scheme for state vectors is extended by a second dimension – yielding a decomposition scheme for $2^n \times 2^n$ matrices.

The entries of a unitary matrix $U = [u_{i,j}]$ indicate how much the operation U affects the mapping from a basis state $|i\rangle$ to a basis state $|j\rangle$. Considering
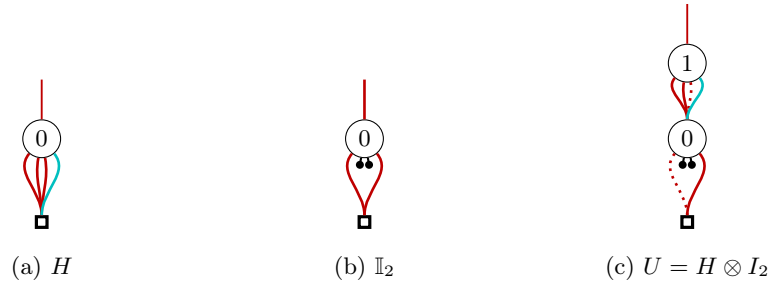
(a) H (b) $\mathbb{I}_2$ (c) $U = H \otimes I_2$

Fig. 2: Representation of matrices

again a quantum system with qubits $q_{n-1}, \ldots, q_1, q_0$, whereby q_{n-1} represents the most significant qubit, the matrix U is decomposed into four sub-matrices with dimension $2^{n-1} \times 2^{n-1}$: All entries in the left upper sub-matrix (right lower sub-matrix) provide the values describing the mapping from basis states $|i\rangle$ to $|j\rangle$ with both assuming $q_0 = |0\rangle$ ($q_0 = |1\rangle$). All entries in the right upper sub-matrix (left lower sub-matrix) provide the values describing the mapping from basis states $|i\rangle$ with $q_0 = |1\rangle$ to $|j\rangle$ with $q_0 = |0\rangle$ ($q_0 = |0\rangle$ to $q_0 = |1\rangle$).

This decomposition is represented in a decision diagram structure by a node labeled q_{n-1} and four successors leading to nodes representing the sub-matrices. The sub-matrices are recursively decomposed further until a 1×1 matrix (i.e., a complex number) results. This eventually represents the value $u_{i,j}$ for the corresponding mapping. Also during this decomposition, equivalent sub-matrices are represented by the same nodes and weights. As for decision diagrams representing state vectors, a corresponding normalization scheme is employed. To this end, all edges-weights are divided by the leftmost entry with the largest magnitude. Again, zero stubs are used to indicate zero matrices (i.e., matrices that contain zeros only) and edge weights equal to one are omitted. Similar to decision diagrams for quantum states, magnitude and phase of edge weights are encoded as thickness and color, respectively (see Example 4). Again, an example illustrates the idea.

Example 5. *Consider the matrices of the Hadamard operation H, the identity $\mathbb{I}_2$, and their combination $U = H \otimes \mathbb{I}_2$, i.e.,*

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \mathbb{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad U = H \otimes I_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

Fig. 2 shows the corresponding decision diagram representations. Following the path with dotted lines in Fig. 2c defines the entry $u_{2,0}$: a mapping from $|0\rangle$ to $|1\rangle$ for q_1 (third edge from the left) and from $|0\rangle$ to $|0\rangle$ for q_0 (first edge). Consequently the path describes the entry for a mapping from $|00\rangle$ to $|10\rangle$. Multiplying all factors on the path (including the root *edge) yields $\frac{1}{\sqrt{2}} \cdot 1 \cdot 1 = \frac{1}{\sqrt{2}}$, which is the value of $u_{2,0}$.*

Overall, the concepts described above yield to the definition of a decision diagram representing a unitary matrix as follows.

Definition 2. *The decision diagram representing a $2^n \times 2^n$-dimensional unitary matrix is a directed acyclic graph with one terminal node labeled 1 that has no successors and represents a 1×1 matrix with the element 1. All other nodes are labeled q_i, $0 \leq i < n$ (representing a partition over qubit q_i) and have four successors. Additionally, there is an edge pointing to the root node of the decision diagram. This edge is called* root edge. *Each edge of the graph has attached a complex number as weight. An entry of the unitary matrix is then determined by the product of all edge weights along the path from the root towards the terminal. Without loss of generality, the nodes of the decision diagram are ordered by the significance of their label, i.e., the successor of a node labeled q_i are labeled with a less significant qubits q_j. Finally, the nodes are normalized, which means that all edges-weights are divided by the leftmost entry with the largest magnitude. The common factor is propagated upwards in the decision diagram.*

A performance-oriented implementation handling decision diagrams and operations as described in this section is freely available under the MIT license at `https://github.com/cda-tum/dd_package`. Furthermore, to give a better intuition and make decision diagrams for quantum computing more accessible, an installation-free web-tool that visualizes decision diagrams for state vectors as well as matrices is available at `https://www.cda.cit.tum.de/app/ddvis/`.

Given the decision diagrams as described in this section, the following sections showcase the applicability in different areas of design and verification work. More precisely, we cover the simulation of quantum circuits without noise in Section 3 and with noise in Section 4 as well as verification of quantum circuits in Section 5.

3 Simulation of Quantum Circuits

Despite physical quantum computers being available in the cloud nowadays, the simulation of quantum circuits on non-quantum hardware remains paramount for the development and design of future quantum computing applications. Additionally, simulations on non-quantum hardware provide insights into the inner workings of a quantum system that are fundamentally hidden in physical quantum computers. This enables designers to analyze quantum algorithms or verify the output of physical quantum computers. To this end, simulating a quantum circuit entails the successive application of all individual gates of the circuit to the initial state of a quantum system in order to obtain the final state. The final state is measured to obtain the result in the computational bases. While straight-forward in principle, this quickly amounts to a hard task due to the required memory on non-quantum hardware and the subsequently difficult manipulation of 2^n complex amplitudes for an n-qubit system.

Decision diagrams, as described in Section 2, provide a promising technique that aims at compactly representing the 2^n complex amplitudes of a quantum
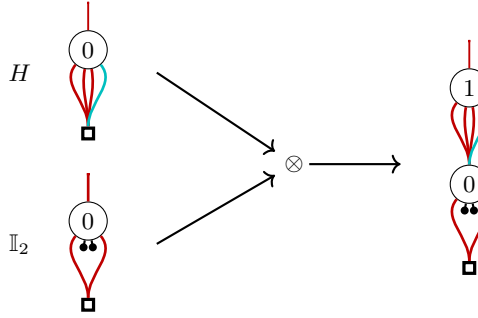
Fig. 3: Creation of $H \otimes \mathbb{I}_2$ using decision diagrams

system and the corresponding operations applied to it. Having the ability to compactly represent state vectors and unitary matrices, all that is left is to provide corresponding methods to form the Kronecker product, multiply vectors with matrices, as well as measure the quantum system. Since the introduced decision diagrams closely relate to vectors and matrices, we can implement the required operations by slight adaptations only.

3.1   Kronecker Product

The Kronecker product enables composition of multiple matrices to attain the suitable size $2^n \times 2^n$ matrix to be applied to an n-qubit system. Given two matrices A and B, the Kronecker product is defined as in Eq. (1).

$$A \otimes B = \begin{bmatrix} a_{0,0} \cdot B & \cdots & a_{0,2^k-1} \cdot B \\ \vdots & \ddots & \vdots \\ a_{2^k-1,0} \cdot B & \cdots & a_{2^k-1,2^k-1} \cdot B \end{bmatrix} \tag{1}$$

In other words, the Kronecker product replaces each element $a_{i,j}$ of A by $a_{i,j} \cdot B$. While this constitutes a computationally expensive task using straightforward realizations by means of array-based implementations of A and B, it is very cheap to form the Kronecker product of two matrices given as decision diagrams.

Since $a_{i,j}$ is given as product of the edge weights from A's root node to the terminal and we can easily determine $a_{i,j} \cdot B$ by adjusting the weight of the edge pointing to B's root node. All that has to be done to determine $A \otimes B$ is replacing A's terminal with the root node of B. Additionally, the weight of A's root edge has to be multiplied by the weight of B's root edge.

Example 6. *Recall the matrices considered in Fig. 2c. The Kronecker product $U = H \otimes \mathbb{I}_2$ can efficiently be constructed by taking the decision diagram representation of H and replacing its terminal node with the root node of the decision diagram representing $\mathbb{I}_2$. Since the root edge of $\mathbb{I}_2$ has weight 1, the value of the root node of U is equal to the weight of A's root edge. This is illustrated in Fig. 3.*
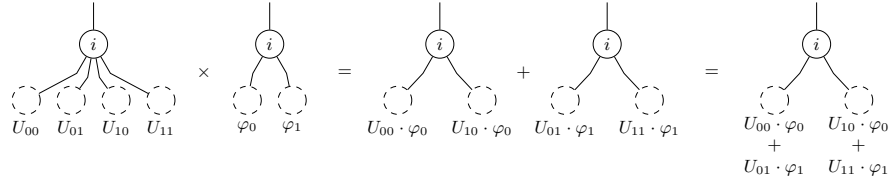
Fig. 4: Recursive structure of multiplication and addition using decision diagrams

3.2   Adding and Multiplying Unitary Matrices

The multiplication of a unitary matrix U and a state vector $|\varphi\rangle$ can be broken down into sub-computations according to Eq. (2).

$$\begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix} \cdot \begin{bmatrix} \varphi_0 \\ \varphi_1 \end{bmatrix} = \begin{bmatrix} (U_{00}\varphi_0 + U_{01}\varphi_1) \\ (U_{10}\varphi_0 + U_{11}\varphi_1) \end{bmatrix} \tag{2}$$

For decision diagrams, recursively determining the four sub-products $U_{00} \cdot \varphi_0$, $U_{01} \cdot \varphi_1$, $U_{10} \cdot \varphi_0$, and $U_{11} \cdot \varphi_1$ realizes the multiplication. The decompositions of multiplication and addition are recursively applied until 1×1 matrices or 1-dimensional vectors result. Since these represent just complex numbers, their multiplication and addition is well defined.

As shown in the middle of Fig. 4, these sub-products are then combined with a decision diagram node to two intermediate state vectors. Finally, these intermediate state vectors have to be added. This addition is recursively decomposed similarly, namely as in Eq. (3).

$$\psi + \phi = \begin{bmatrix} \psi_0 \\ \psi_1 \end{bmatrix} + \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} \psi_0 + \phi_0 \\ \psi_1 + \phi_1 \end{bmatrix} \tag{3}$$

The recursively determined sub-sums $\psi_0 + \phi_0$ and $\psi_1 + \phi_1$ are composed by a decision diagram node as shown on the right-hand side of Fig. 4.

Moreover, decomposition into sub-products and sub-sums does not change the decision diagram structure. Hence, the complexity of them remains bounded by the number of nodes of the original representations. Furthermore, redundancies can again be exploited by caching sub-products and sub-sums.

3.3   Measurement

Measurement can efficiently be conducted on the decision diagram structure. Without loss of generality, consider that the most significant qubit (which is represented by the root node of the corresponding decision diagram) of the state vector should be measured. This can be accomplished by applying a SWAP operation or by re-arranging the nodes and edges of the decision diagram. Then, the probability of choosing either the left or right edge is given by the *upstream* probability of the successor nodes weighted by the corresponding edge

(a) Measure $q_2 = |1\rangle$ (b) Normalize amplitudes
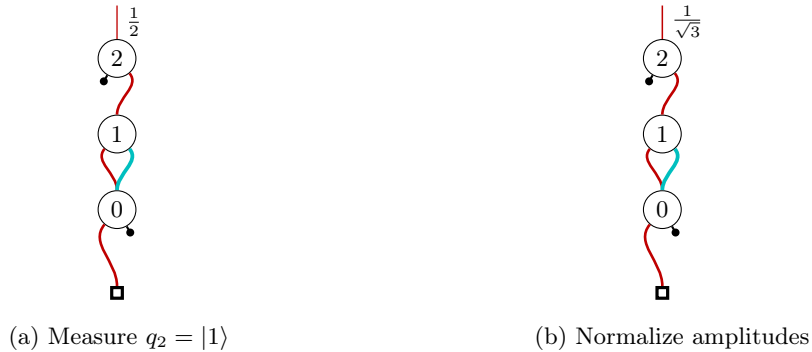
Fig. 5: Measurement of qubit q_2

weights [13]. Depending on the used normalization scheme, this calculation may be simplified [14]. An example illustrates the idea.

Example 7. *Consider again the quantum state discussed in Example 1 and its corresponding decision diagram shown in Fig. 1b. Then, the probabilities for measuring $q_2 = |0\rangle$ and $q_2 = |1\rangle$ are $|\frac{1}{2}|^2 \cdot |1|^2 \cdot 1 = \frac{1}{4}$ and $|\frac{1}{2}|^2 \cdot |1|^2 \cdot 3 = \frac{3}{4}$, respectively.*

Having the probabilities for collapsing the most significant qubit to basis state $|0\rangle$ and $|1\rangle$ allows to sample its new value. If we obtain basis state $|0\rangle$ ($|1\rangle$), the amplitudes for all basis states with $q_{n-1} = |1\rangle$ ($q_{n-1} = |0\rangle$) are set to zero. In the decision diagram, the collapse is performed by multiplying with the M_0 (M_1) non-unitary matrix from Eq. (4).

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad M_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

Afterwards the decision diagram is renormalized and the probability of the considered qubit being measurement again in the same basis state in subsequent measurements is 1.

Example 7 (continued). *Assume we measure basis state $|1\rangle$ for qubit q_2. Fig. 5a shows the resulting decision diagram. To fulfil the normalization constraint, we renormalize the decision diagram – eventually resulting in the decision diagram shown in Fig. 5b.*

Measuring all qubits can be conducted in a similar fashion. In fact, we repeat the procedure discussed above sequentially for all qubits $q_{n-1}, q_{n-2}, \ldots, q_0$. Assume that qubit q_i shall be measured, and that all qubits q_j where $j > i$ are already measured. Then, there exists only one node labeled q_i, which is the root node of the sub-vector to be measured.

Overall, this allow for an more efficient quantum circuit simulation in many cases. An implementation of this method is available at `https://github.com/cda-tum/ddsim` and via the corresponding Python package `mqt.ddsim`. In addition, `https://www.cda.cit.tum.de/app/ddvis/` provides an installation-free visualization scheme of the procedure.

4 Noise-Aware Simulation of Quantum Circuits

The methods reviewed above allow for an efficient simulation of *perfect*, i.e., error-free, quantum circuits. While already an important step towards proper design and evaluation of certain applications, physical quantum computers do not work perfectly and are affected by noise effects, which cause errors during quantum computations. Considering those errors during simulation enables a more accurate and realistic evaluation of the respectively considered quantum circuits. At the same time, considering errors introduces new challenges for the already exponentially hard problem of quantum circuit simulation. In this section, we review how to conduct noisy simulation with decision diagrams. To this end, we first review typical noise effects in quantum computing, i.e., recap frequently occurring errors, and, afterwards, discuss two complementary solutions for noise-aware simulation based on decision diagrams (originally proposed in [15], [16]).

4.1 Errors in Quantum Computations

During quantum computations, a large variety of errors may occur and affect the output of the corresponding executions. Most prominently, two types of errors are distinguished [17]:

Gate errors: Any errors that may alter the originally intended functionality of an operation or may lead to an operation not being executed at all.
Decoherence errors: Any errors caused by the effect that qubits can only hold information for a limited amount of time.

Gate errors heavily depend on the underlying quantum computer technology and even on the qubits to which the respective operations are applied. The effect is that the operation either is not executed at all or that a different operation is employed. Often, they are approximated using depolarization errors [18], [19] and, hence, defined by altering the qubit to a completely random state [20]. More detailed descriptions of the respective effects are additionally often provided by the vendors of the respective quantum computer, e.g., in case of IBM at [21].

Decoherence Errors occur due to the fragile nature of quantum systems. Because of this, qubits can only hold information for a limited time and, hence, qubits in a high-energy state ($|1\rangle$) tend to relax towards a low energy state ($|0\rangle$) (i.e., after a certain amount of time, qubits eventually decay towards $|0\rangle$). Moreover, when a qubit interacts with the environment, further errors (such as phase-flip errors) might occur.

Example 8. *Consider a 2-qubit system which is in state $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and assume that a gate error might affect this state with probability p. Then, with probability $1-p$, nothing happens (the state remains unchanged) while, with probability p, a certain error effect is imposed. Both scenarios can be captured by either employing an I-operation or an operation describing the error effect (e.g., a polarization using X, Y, or Z or a completely random effect), respectively.*

In a similar fashion, consider the same quantum state but assume a decoherence error at the second qubit (more precisely, a dampening error which makes the second qubit decay to $|0\rangle$) with a probability $p = 0.3$. Then, a measurement of this state would not lead to $|00\rangle$ or $|11\rangle$ with equal probabilities anymore (as in the error-free case), but to $|00\rangle$ in 50% of the cases, $|10\rangle$ in 15% of the cases, and $|11\rangle$ in 35% of the cases. That is, the probability that the second qubit decays to $|0\rangle$ is substantially larger due to the decoherence.

Overall, errors effects can be seen as (unwanted) operations employed on a quantum system. Accordingly, they could in principle be simulated like any other quantum operation—using, e.g., the methods described before in Section 3. The main challenge, however, is that whether an error effect happens or not depends on certain probabilities. These need to be captured during the simulation. Existing solutions doing that have been proposed, e.g., in [19], [22]–[29].

4.2   Simulation Methods Using Decision Diagrams

In this section, we review how decision diagrams may help in providing a solution for noise-aware quantum circuit simulation. The first solution thereby relies on a stochastic approach which employs the main concepts reviewed in Section 3, while the second solution aims for a deterministic consideration of noise—requiring a more complex representation of quantum states and operations.

More precisely, the first solution (stochastic simulation as proposed in [16]) is based on the vanilla decision diagram-based circuit simulator described in Section 3. Then, whenever the considered quantum computer might make an error during its simulated operation, it either mimics the effect of the error by additionally employing an error operation (with corresponding probability p) or leaves the state untouched (with probability $1 - p$). Consequently, a *single* output state is sampled from the whole spectrum of possible output states by such a run.

By iteratively sampling sufficiently many output states (using, e.g., stochastic Monte-Carlo approximation) a rather accurate approximation of the quantum circuit's behavior under the influence of noise effects can be obtained. The benefit of this approach is that it does not substantially increase the complexity of individual simulation runs when compared to the error-free circuit simulation. Furthermore, individual simulations are independent and, hence, can be executed in parallel. However, this approach remains stochastic, i.e., it cannot guarantee the best possible accuracy (although evaluations summarized in [30] show that a sufficient accuracy can be achieved for practically relevant use cases).

If an exact consideration of noise is desired, a more elaborate solution is required which describes all noise effects in a deterministic fashion. To this end, the representation of quantum states and quantum operations in terms of vectors and matrices (as used thus far) is not sufficient anymore. More precisely, a description is needed which incorporates all possible states a quantum system

may reside in (including the original state but also states resulting from any noise effects with certain probabilities).

This is accomplished by extending the state vector representation to *density matrices* (also known as *density operators*) [31]). More precisely, let $|\phi\rangle$ be a complex vector representing the state of a quantum system. Then, the corresponding *density matrix* is defined as $\rho = |\phi\rangle\langle\phi|$ with $\langle\phi| := |\phi\rangle^\dagger$.

Example 9. *Consider again the quantum state* $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ *from Example 8. The corresponding* density matrix ρ *is given by*

$$
\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}.
\tag{5}
$$

This representation properly describes the quantum state while, additionally, allowing to store information about the noise-effects on the state. For example, the diagonal entries encode the probabilities for measuring $|00\rangle, |01\rangle, |10\rangle,$ *and* $|11\rangle$, *respectively, which is in line with the probabilities obtained from the state vector representation* $(|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2})$.

Based on this representation, various error effects can now be applied by a tuple $(E_0, E_1, \ldots, E_m)$ of *Kraus matrices* satisfying the condition

$$
\sum_{i=0}^{m} E_i^\dagger E_i = \mathbb{I}.
\tag{6}
$$

For example, a decoherence error which makes a qubit decay to $|0\rangle$ can be represented by [20]

$$
(E_0, E_1) \text{ with } E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix} \text{ and } E_1 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix},
\tag{7}
$$

where the variable p represents the probability of the error occurring. Other noise effects can be described in a similar fashion. Applying these error descriptions to a quantum system given by the density matrix ρ yields the density matrix [20]:

$$
\rho' = \sum_{i=0}^{m} E_i \rho E_i^\dagger.
\tag{8}
$$

This formalism allows to deterministically capture all noise effects as illustrated in the following example:

Example 10. *Consider again the quantum state from above and the decoherence error from Example 8 making the second qubit decay to* $|0\rangle)$ *with probability* $p = 0.3$. *Then, the error's effect can be calculated deterministically by constructing the correctly sized matrices with the Kronecker product and summing*
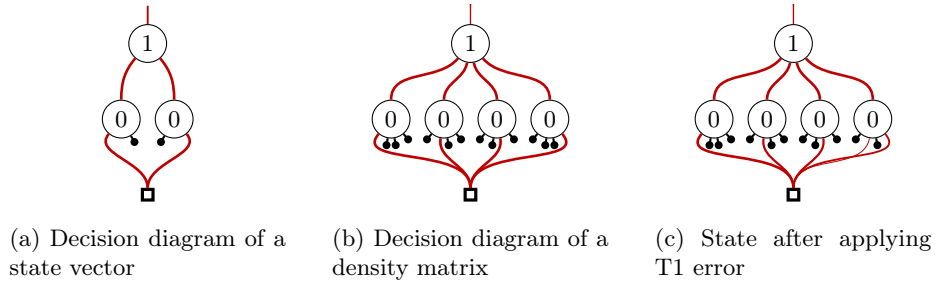
(a) Decision diagram of a state vector

(b) Decision diagram of a density matrix

(c) State after applying T1 error

Fig. 6: Decision diagram representation of states

them as in Eq. (9).

$$
\begin{bmatrix} 0.5 & 0 & 0 & 0.418 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.418 & 0 & 0 & 0.35 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 & 0.418 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 \\ 0.418 & 0 & 0 & 0.35 \end{bmatrix} \quad (9)
$$

$$
\underbrace{\hphantom{xxxxxxxxxxx}}_{E_0 \rho E_0^\dagger} \qquad \underbrace{\hphantom{xxxxxxxxxx}}_{E_1 \rho E_1^\dagger} \qquad \underbrace{\hphantom{xxxxxxxxxxx}}_{\rho'}
$$

Again the diagonal encodes the probabilities for measuring $|00\rangle , |01\rangle , |10\rangle ,$ *and* $|11\rangle ,$ *which are in-line with the values covered before in Example 8.*

The resulting representations are substantially larger than the vectors and matrices needed for error-free quantum circuit simulation. For example, rather than vectors of size 2^n, matrices of size $2^n \times 2^n$ are needed to represent an n-qubit quantum state. However, density matrices can be represented in terms of decision diagrams as well. In fact, employing the same decomposition scheme for matrices as reviewed in Section 3 yields corresponding decision diagrams for density matrices.

Example 11. *Consider again the quantum state from Example 9 in both the vector as well as density matrix representation. The corresponding decision diagram representations are provided in Fig. 6a and Fig. 6b, respectively. The decision diagram resulting after applying the error affect (as considered in Example 10) is shown in Fig. 6c.*

Obviously, the decision diagram representations of the density matrices (providing a deterministic representation of all employed error effects) is larger than the original state representation. After all, a substantially larger amount of information needs to be stored. Nevertheless, the examples show that, also in these cases, decision diagrams may offer a more compact representation than offered by a direct representation in terms of a $2^n \times 2^n$-matrix. After all, this helps in improving deterministic, noise-aware quantum circuit simulation.

Overall, this section showed that decision diagrams can be employed for noise-aware quantum circuit simulation—both, stochastically as well as deterministically. Further details and evaluations on the respective methods are available in [15], [16]. An implementation of the stochastic method is available at https://github.com/cda-tum/ddsim.

(a) QFT Circuit G

(b) Functionality U ($\omega = (1+i)/\sqrt{2}$)

$$\frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^3 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix}$$
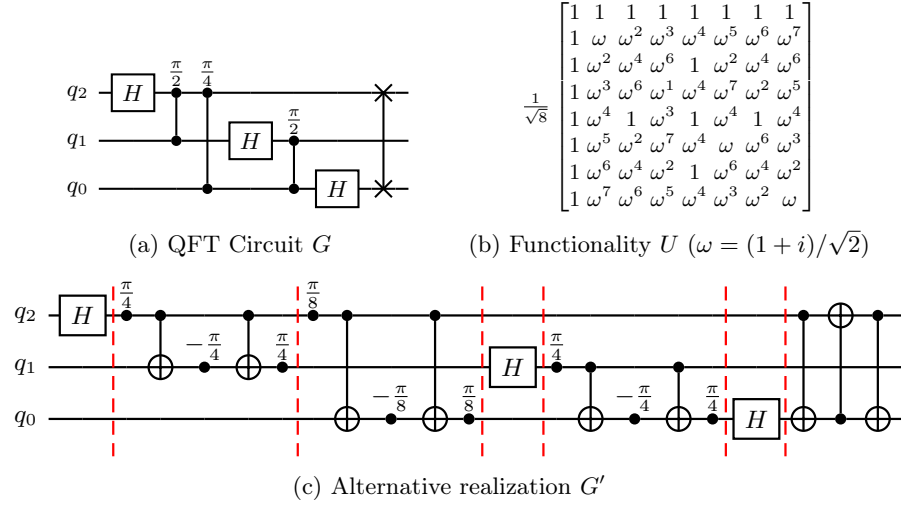
(c) Alternative realization G'

Fig. 7: The QFT, its functionality, and an alternative realization

5   Verification of Quantum Circuits

As a final example for the utilization of decision diagrams in design/software for quantum computing, we consider the task of verification – more precisely equivalence checking. Here, the question is whether two quantum circuits G and G' realize the same functionality. This is motivated by the design flow in which a given circuit is decomposed, mapped, and optimized [32]–[42]. During all these steps, it has to be ensured that the functionality of the correspondingly resulting circuit descriptions does not change. In the following, we first give an explicit description of this problem and, then, describe two complementary approaches for tackling it using decision diagrams.

5.1   The Quantum Circuit Equivalence Checking Problem

Equivalence checking in the domain of quantum computing—as we consider it in this work—is about proving that two quantum circuits G and G' are functionally equivalent (i.e., realize the same function), or to show the non-equivalence of these circuits by means of a counterexample. To this end, consider two quantum circuits $G = g_0 \ldots g_{m-1}$ and $G' = g'_0 \ldots g'_{m'-1}$ operating on n qubits. Then, the functionality of each circuit can be uniquely described by the respective system matrices $U = U_{m-1} \cdots U_0$ and $U' = U'_{m'-1} \cdots U'_0$, where the matrices $U_i^{(')}$ describe the functionality of the i-th gate of the respective circuit (with $0 \leq i < m^{(')}$). Consequently, deciding the equivalence of both computations amounts to comparing the system matrices U and U'. More precisely, U and U' are considered equivalent, if they at most differ by a global phase factor (which is fundamentally unobservable [43]), i.e., $U = e^{i\alpha}U'$ with $\alpha \in [0, 2\pi)$.
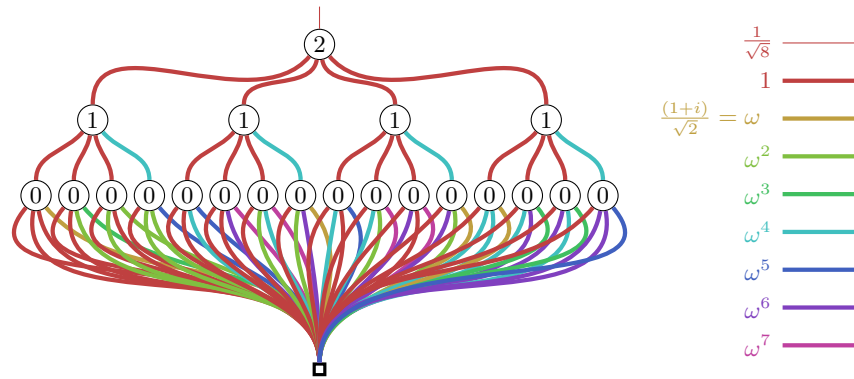
Fig. 8: Decision diagram for the functionality U of G shown in Fig. 7

Example 12. *Consider the circuit G of the three qubit Quantum Fourier Transform shown in Fig. 7a. Its corresponding functionality is described by the densely-populated 8×8 matrix U shown in Fig. 7b [43]. Additionally, Fig. 7c shows an alternative realization G' of the functionality of G. Since both circuits exhibit the same system matrix U, they are considered equivalent.*

Unfortunately, the whole functionality U (and similarly U') is not readily available for performing this comparison, but has to be constructed from the individual gate descriptions g_i—requiring the subsequent matrix-matrix multiplications $U^{(0)} = U_0$, $\quad U^{(j)} = U_j \cdot U^{(j-1)}$ for $j = 1, \ldots, m - 1$ to construct the whole system matrix $U = U^{(m-1)}$. While conceptually simple (as the matrix-vector multiplication for simulation discussed in Section 3), this quickly constitutes an extremely complex task due to the exponential size of the involved matrices with respect to the number of qubits. In fact, equivalence checking of quantum circuits has been shown to be QMA-complete [44].

Due to their potential for compactly representing and efficiently manipulating the functionality of a quantum circuit, decision diagrams are a perfect fit for this task. However, merely using decision diagrams to construct a representation of both circuits' functionality and comparing them, still has significant shortcomings. In fact, representing the entire functionality of a quantum circuit still might be exponential in the worst case. However, this can be addressed by additionally exploiting the reversibility of quantum circuits.

Example 13. *Consider the functionality U of the QFT circuit G from Fig. 7. Its corresponding decision diagram (shown in Fig. 8 with color legend to the right-hand side) is as densely populated as the matrix it represents since no redundancy can be exploited, i.e., each node's successors point to unique child nodes.*

5.2   Exploiting Reversibility

Most classical logic operations are not reversible (e.g., neither $x \wedge y = 0$ nor $x \vee y = 1$ allows one to determine the values of x and y). As there is no bijective mapping between input and output states, in general, the concept of the *inverse* of a classical operation (or a sequence thereof) is not meaningful. In contrast, all quantum operations are inherently *reversible*. Consider an operation g described by the unitary matrix U. Then, its inverse U^{-1} is efficiently calculated as the conjugate-transpose $U^\dagger$. Given a sequence of m operations $g_0, \ldots, g_{m-1}$ with associated matrices $U_0, \ldots, U_{m-1}$, the inverse of the corresponding system matrix $U = U_{m-1} \cdots U_0$ is derived by reversing the operations' order and inverting each individual operation, i.e., $U^{-1} = U^\dagger = U_0^\dagger \cdots U_{m-1}^\dagger$.

This characteristics can be exploited to improve the performance of the verification approach presented above. To this end, consider two quantum circuits G and G'. In case both circuits are functionally equivalent, this allows for the conclusion that concatenating one circuit with the inverse of the other realizes the identity function $\mathbb{I}$, i.e., $G'^{-1} \cdot G \equiv \mathbb{I}$. This offers significant potential since the identity constitutes the best case for decision diagrams (the identity can be represented by a decision diagram of linear size). Unfortunately, creating such a concatenation in a naive fashion, e.g., by computing $U \cdot U'^\dagger$ hardly yields any advantage because, even if the final decision diagram would be as compact as possible, the full (and potentially exponential) decision diagram of at least one of the circuits would be generated as an intermediate result.

Instead, the full potential of this observation is utilized if the associativity of the respective multiplications is fully exploited. More precisely, given two quantum circuits G and G', it holds that

$$G'^{-1} \cdot G = (g'^{-1}_{m'-1} \ldots g'^{-1}_0) \cdot (g_0 \ldots g_{m-1})$$
$$\equiv (U_{m-1} \cdots U_0) \cdot (U'^\dagger_0 \cdots U'^\dagger_{m'-1})$$
$$= U_{m-1} \cdots U_0 \cdot \mathbb{I} \cdot U'^\dagger_0 \cdots U'^\dagger_{m'-1}$$
$$=: G \to \mathbb{I} \leftarrow G'.$$

Here, $G \to \mathbb{I} \leftarrow G'$ symbolizes that, starting from the identity $\mathbb{I}$, either gates from G can be "applied from the left", or (inverted) gates from G' can be "applied from the right". If the respective gates of G and G' are applied in a fashion frequently yielding the identity, the entire equivalence checking process can be conducted on rather small (intermediate) decision diagrams. This is illustrated by the following example.

Example 14. *Consider again the two circuits G and G' from Example 12 and assume that, starting with a decision diagram representing the identity, for every gate applied from G all gates from G' until the next red barrier shown in Fig. 7c are applied. Applying the gates from G and G' in such a particular order "from the left" and "from the right", respectively, yields situations where the impact of a gate from circuit G (increasing the size of the decision diagram) is reverted by multiplications with inverted gates from G' (decreasing the size of the decision*

diagram back to the representation of the identity function). This way, the equivalence check can be conducted on much smaller intermediate representations and, hence, much more efficiently.

Moreover, even if the considered circuits G and G' are *not* functionally equivalent (and, hence, identity is not achieved), the observations from above still promise improvements compared to creating the complete decision diagrams for G and G'. This is, because in this case, the result of $G \to \mathbb{I} \leftarrow G'$ inherently provides an efficient representation of the circuit's difference that allows one to obtain counterexamples almost "for free" (while those have to be explicitly generated using additional inversion and multiplication operations otherwise).

Overall, following those ideas, equivalence checking of two quantum circuits can be conducted very efficiently on rather compact decision diagrams, as shown in [45]. But determining when to apply gates from G and when to apply (inverted) gates from G' is not at all obvious. Designing dedicated strategies for specific applications is a topic of ongoing research. As an example, a dedicated strategy for verifying the results of compilation flows can be derived by exploiting knowledge about the compilation flow itself [46]. An implementation of this method is available at `https://github.com/cda-tum/qcec` and via the corresponding Python package `mqt.qcec`. In addition, `https://www.cda.cit.tum.de/app/ddvis/` provides an installation-free visualization scheme of the procedure that also can be used to try out different gate-application schemes.

5.3   The Power of Simulation

The second characteristic we are exploiting rests on the observation that simulation is much more powerful for equivalence checking of quantum circuits than for equivalence checking of classical circuits. More precisely, in the classical realm, it is certainly possible to simulate two circuits with random inputs to obtain counterexamples in case they are not equivalent. However, this often does not yield the desired result. In fact, due to masking effects and the inevitable information loss introduced by many classical gates, the chance of detecting differences in the circuits within a few arbitrary simulations is greatly reduced (e.g., $x \wedge 0$ masks any difference that potentially occurs during the calculation of x). Consequently, sophisticated schemes for constraint-based stimuli generation [47]–[50], fuzzing [51], [52], etc. are employed in order to verify classical circuits.

In quantum computing, the inherent reversibility of quantum operations dramatically reduces these effects and frequently yields situations where even small differences remain unmasked and affect entire system matrices—showing the power of random simulations for checking the equivalence of quantum circuits. Because of that, it is in general not necessary to compare the *entire* system matrices—in particular when two circuits are *not* equivalent and, hence, their system matrices differ from each other.

Given two unitary matrices U and U', we define their *difference* D as the unitary matrix $D = U^\dagger U'$ and it holds that $U \cdot D = U'$. In case both matrices are identical (i.e., the circuits are equivalent), it directly follows that $D = \mathbb{I}$. One

characteristic of the identity function $\mathbb{I}$ resulting in this case is that all diagonal entries are equal to one, i.e., $\langle i | U^\dagger U' | i \rangle = 1$ for $i \in \{0, \dots, 2^n - 1\}$, where $|i\rangle$ denotes the ith computational basis state. More generally—in case of a potential relative/global phase difference between G and G'—all diagonal elements have modulus one, i.e., $|\langle i | U^\dagger U' | i \rangle|^2 = 1$. This expression can further be rewritten to

$$1 = |\langle i | U^\dagger U' | i \rangle|^2 = |(U |i\rangle)^\dagger (U' |i\rangle)|^2 = ||u_i\rangle^\dagger |u_i'\rangle|^2$$
$$= |\langle u_i | u_i' \rangle|^2,$$

where $|u_i\rangle$ and $|u_i'\rangle$ denote the ith column of U and U', respectively. This essentially resembles the simulation of both circuits with the initial state $|i\rangle$ and, afterwards, calculating the fidelity $\mathcal{F}$ between the resulting states $|u_i\rangle$ and $|u_i'\rangle$. Hence, if only one simulation yields $\mathcal{F}_i := \mathcal{F}(|u_i\rangle, |u_i'\rangle) \not\approx 1$, then $|i\rangle$ proves the non-equivalence of G and G'.

This constitutes an exponentially easier task than constructing the entire system matrices U and U'—although the complexity of simulation still remains exponential with respect to the number of qubits (for which the decision diagram-based solution reviewed above provides an efficient solution in many cases). Regarding the complexity, creating the entire system matrices corresponds to simulating the respective circuit with all 2^n different computational basis states. All this, of course, does not guarantee that any difference is indeed detected by just simulating a limited number of arbitrary computational basis states $|i\rangle$. This brings up the following question: How significantly do the matrices U and U' differ from each other in case of non-equivalence, i.e., how many computational basis states $|i\rangle$ yield $\mathcal{F}_i \not\approx 1$ for a given difference matrix D. Since the difference D of both matrices is unitary itself, it may as well be interpreted as a quantum circuit G_D. In the following, we assume that each gate of G_D either represents a single-qubit or a multi-controlled operation[5].

Example 15. *Assume that G_D only consists of one (non-trivial) single-qubit operation defined by the matrix U_s applied to the first of n qubits. Then, the system matrix D is given by $diag(U_s, \dots, U_s)$. The process of going from U to U', i.e., calculating $U \cdot D$, impacts* all *columns of U. Thus, an error is detected by a* single *simulation with* any *computational basis state.*

Among all quantum operations, single-qubit operations possess a system matrix least similar to the identity matrix due to the tensor product structure of their corresponding system matrix.

Example 16. *In contrast to Example 15, assume that G_D only consists of one operation U_s targeted at the first qubit and controlled by the remaining $n - 1$ qubits. Then, the corresponding system matrix is given by $diag(\mathbb{I}_2, \dots, \mathbb{I}_2, U_s)$.*

---

[5] This does not limit the applicability of the following findings, since arbitrary single-qubit operations combined with CNOT form a universal gate-set [43].

In this case, applying D to U only affects the last two columns of U. As a consequence, a maximum of two columns (out of 2^n) may serve as counterexamples—the worst case scenario.

These basic examples cover the extreme cases when it comes to the difference of two unitary matrices. In case G_D exhibits no such simple structure, the analysis is more involved, e.g., generally quantum operations with $c \in \{0, \ldots, n-1\}$ controls will exhibit a difference in 2^{n-c} columns. Furthermore, given two operations showing a certain number of differences, the matrix product of these operations in most cases (except when cancellations occur) differs in as many columns as the maximum of both operands.

The gate-set provided by (current) quantum computers typically includes only (certain) single-qubit gates and a specific two qubit gate, such as the $CNOT$ gate. Thus, multi-controlled quantum operations usually only arise at the most abstract algorithmic description of a quantum circuit and are then *decomposed* into elementary operations from the device's gate-set before the circuit is mapped to the target architecture. As a consequence, errors occurring during the design flow will typically consist of (1) single-qubit errors, e.g., offsets in the rotation angle, or (2) errors related to the application of $CNOT$ or $SWAP$ gates. In both cases, non-equivalence can be efficiently concluded by a limited number of simulations with arbitrary computational basis states. If a counterexample was not obtained after a few simulations, this yields a highly probable estimate of the circuit's equivalence—in contrast to the classical realm, where this generally does not allow for any conclusion. Further details and evaluations on the respective methods are available in [45], [53].

6 Conclusions

The power of quantum computing comes with new computing primitives and the need for suitable design automation methods. In this work, we reviewed decision diagrams for quantum computing as well as their application in quantum circuit simulation (with and without noise) as well as the verification of quantum circuits. Decision diagrams offer a complementary approach for tackling the complexity of these tasks with a potential impact comparable to their conventional counterparts. With this work, we want to encourage their usage in the quantum (design automation) community. Implementations of the methods presented here are available at the corresponding GitHub repositories mentioned above.

Acknowledgments

References

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

[2]   Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, 1986.

[3]   R. E. Bryant and Y.-A. Chen, "Verification of arithmetic circuits using binary moment diagrams," *Software Tools for Technology Transfer*, vol. 3, no. 2, pp. 137–155, 2001.

[4]   S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," in *Design Automation Conf.*, 1993, pp. 272–277.

[5]   T. van Dijk, R. Wille, and R. Meolic, "Tagged BDDs: Combining reduction rules from different decision diagram types," in *Int'l Conf. on Formal Methods in CAD*, 2017, pp. 108–115.

[6]   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, pp. 584–594.

[7]   A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006.

[8]   G. F. Viamontes, I. L. Markov, and J. P. Hayes, "High-performance QuIDD-Based simulation of quantum circuits," in *Design, Automation and Test in Europe*, 2004.

[9]   D. Miller and M. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, 2006.

[10] 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.

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

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

[13] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 848–859, 2019.

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

[15] T. Grurl, J. Fuß, and R. Wille, "Considering decoherence errors in the simulation of quantum circuits using decision diagrams," in *Int'l Conf. on CAD*, 2020.

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

[17] S. S. Tannu and M. K. Qureshi, "Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers," in *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 987–999.

[18] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudéver, and K. Bertels, "QX: A high-performance quantum computer simulation platform," in *Design, Automation and Test in Europe*, D. Atienza and G. D. Natale, Eds., 2017, pp. 464–469.

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

[20] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[21] Gambetta, Jay and Sheldon, Sarah, *Cramming more power into a quantum device*, `https : / / www . ibm . com / blogs / research / 2019 / 03 / power - quantum - device/`, Accessed: 2021-04-08, 2019.

[22] Atos SE, *Quantum learning machine*, atos.net/en/products/quantum-learning-machine, Accessed: 2021-04-08, 2016.

[23] N. Khammassi, I. Ashraf, X. Fu, C. Almudever, and K. Bertels, "QX: A high-performance quantum computer simulation platform," in *Design, Automation and Test in Europe*, 2017.

[24] D. Wecker and K. M. Svore, "LIQUi|>: A software design architecture and domain-specific language for quantum computing," *CoRR*, vol. abs/1402.4467, 2014.

[25] C. Developers, *Cirq*, 2021.

[26] T. Jones, A. Brown, I. Bush, and S. Benjamin, "Quest and high performance simulation of quantum computers," *arXiv:1802.08032*, 2018.

[27] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik, "qHiPSTER: The quantum high performance software testing environment," *CoRR*, vol. abs/1601.07195, 2016.

[28] B. Villalonga *et al.*, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, pp. 1–16, 2019.

[29] *Forest SDK*, `https://www.rigetti.com/systems`, Accessed: 2020-07-22, 2020.

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

[31] T. Grurl, J. Fuß, and R. Wille, "Considering decoherence errors in the simulation of quantum circuits using decision diagrams," in *Int'l Conf. on CAD*, IEEE, 2020, 140:1–140:7.

[32] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, no. 5, pp. 3457–3467, 1995.

[33] D. Maslov, "On the advantages of using relative phase Toffolis with an application to multiple control Toffoli optimization," *Phys. Rev. A*, vol. 93, no. 2, p. 022 311, 2016.

[34] R. Wille, M. Soeken, C. Otterstedt, and R. Drechsler, "Improving the mapping of reversible circuits to quantum circuits using multiple target lines," in *Asia and South Pacific Design Automation Conf.*, 2013.

[35] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for Noisy Intermediate-Scale Quantum computers," in *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1015–1029.

[36] M. Y. Siraichi, V. F. dos Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proc. Int'l Symp. on Code Generation and Optimization*, 2018, pp. 113–125.

[37] 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*, vol. 38, no. 7, pp. 1226–1236, 2019.

[38] 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.

[39] 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.

[40] A. Matsuo, W. Hattori, and S. Yamashita, "Reducing the overhead of mapping quantum circuits to IBM Q system," in *IEEE International Symposium on Circuits and Systems*, 2019.

[41] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo, "Optimization of quantum circuit mapping using gate transformation and commutation," *Integration*, vol. 70, pp. 43–50, 2020.

[42] G. Vidal and C. M. Dawson, "Universal quantum circuit for two-qubit transformations with three controlled-NOT gates," *Phys. Rev. A*, vol. 69, no. 1, p. 010 301, 2004.

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

[44] D. Janzing, P. Wocjan, and T. Beth, ""Non-identity check" is QMA-complete," *Int. J. Quantum Inform.*, vol. 03, no. 03, pp. 463–473, 2005.

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

[46] 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, pp. 356–365.

[47] J. Yuan, C. Pixley, and A. Aziz, *Constraint-based verification*. Springer, 2006.

[48] J. Bergeron, *Writing Testbenches using System Verilog*. Springer, 2006.

[49] N. Kitchen and A. Kuehlmann, "Stimulus generation for constrained random simulation," in *Int'l Conf. on CAD*, 2007, pp. 258–265.

[50] R. Wille, D. Große, F. Haedicke, and R. Drechsler, "SMT-based stimuli generation in the SystemC Verification library," in *Forum on Specification and Design Languages*, 2009.

[51] K. Laeufer, J. Koenig, D. Kim, J. Bachrach, and K. Sen, "RFUZZ: Coverage-directed fuzz testing of RTL on FPGAs," in *Int'l Conf. on CAD*, 2018.

[52] H. M. Le, D. Große, N. Bruns, and R. Drechsler, "Detection of hardware trojans in SystemC HLS designs via coverage-guided fuzzing," in *Design, Automation and Test in Europe*, 2019, pp. 602–605.

[53] 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.