Visualizing Decision Diagrams for Quantum Computing
(Special Session Summary)

Robert Wille*† Lukas Burgholzer* Michael Artner*
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
iic-quantum@jku.at
https://iic.jku.at/eda/research/quantum/

***Abstract*—With the emergence of more and more applications for quantum computing, also the development of corresponding methods for design automation is receiving increasing interest. In this respect, decision diagrams provide a promising basis for many design tasks such as simulation, synthesis, verification, and more. However, users of the corresponding tools often do not have a proper background or an intuition about how these methods based on decision diagrams work and what their strengths and limits are. In an effort to make decision diagrams for quantum computing more accessible, we present a visualization tool which visualizes quantum decision diagrams and allows to explore their behavior when used in the design tasks mentioned above. The installation-free web-tool allows users to interactively learn how decision diagrams can be used in quantum computing, e.g., to (1) compactly represent quantum states and the functionality of quantum circuits, (2) to efficiently simulate quantum circuits, and (3) to verify the equivalence of two circuits. The tool is available at https://iic.jku.at/eda/research/quantum_dd/tool.**

I. Introduction

Quantum computers are steadily improving in terms of their computational power to the extent that first computations are being performed that are no longer feasible on conventional machines [1], [2]. Achieving these milestones is only possible through interdisciplinary efforts by physicists, mathematicians, computer scientists, and many others. Just as in the design of conventional circuits and systems, the development of design automation tools for quantum computing will be one of the key factors for the success of the technology.

In the 80's, decision diagrams were proposed as a data structure for efficiently representing and manipulating Boolean functions [3]. Following this development, several types of decision diagrams such as BDDs, FBDDs, KFDDs, MTBDDs, ZDDs, etc. emerged (see, e.g., [4]–[9]) which established them as a core asset in the design of today's circuits and systems. Due to their success in the past, the use of decision diagrams has also been proposed in the domain of quantum computing [10]–[14]. In particular for design tasks such as *simulation* [10], [15], [16], *synthesis* [17]–[19], and *verification* [12], [20], [21], they found great interest recently.

However, to date, there is still a huge gap to bridge between the quantum computing and the design automation community. In fact, these promising techniques have hardly reached the core of the quantum computing community until now and are not yet established—despite showing promising results. Due to the interdisciplinarity of the field, quite often, users of the corresponding tools are hardly familiar with the underlying theory and lack an intuition about how these tools work.

This special session summary paper (and the tool it proposes) shall serve as an entry point for both, the design automation community in order to better understand the problems and tasks in quantum computing, but even more so the quantum community to get familiar with the concept of decision diagrams. In an effort to make decision diagrams for quantum computing more accessible, we present a tool which visualizes quantum decision diagrams and allows to explore their behavior when used in the design tasks mentioned above. We believe that this will allow researchers and engineers who are unfamiliar with decision diagrams to more easily grasp the concept of them and foster an understanding of how they can be applied in practice.

To properly introduce the tool, we first review the basics of quantum computing and settle the terminology used throughout this work in Sec. II. Based on that, we then describe the construction of decision diagrams for quantum computing and illustrate their application to the design problems of *simulation* and *verification* with several examples in Sec. III. Afterwards, Sec. IV shows how the developed tool can be used to visualize (1) the compact representation of quantum states and the functionality of quantum circuits, (2) the simulation scheme based on decision diagrams, and (3) the equivalence checking scheme based on decision diagrams. Finally, Sec. V concludes the paper. To keep the effort of using the visualization as small as possible, it has been implemented as a web tool which can be started by simply accessing https://iic.jku.at/eda/research/quantum_dd/tool.

II. Quantum Computing

In this section, we give an overview of the main concepts in quantum computing. While the following descriptions are kept brief, we refer to [22] for a thorough introduction.

In quantum computing the main computational unit is the *qubit*. In contrast to classical bits, a qubit cannot only be in one of the computational basis states $|0\rangle$ or $|1\rangle$, but also in an arbitrary *superposition* of these states. Specifically, the state $|\varphi\rangle$ of a qubit is described by two *amplitudes* $\alpha_0, \alpha_1 \in \mathbb{C}$ such that

$$|\varphi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \equiv \alpha_0 \begin{bmatrix}1\\0\end{bmatrix} + \alpha_1 \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha_0\\\alpha_1\end{bmatrix}$$

and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The resulting column vector is also referred to as *state vector*. In an n-qubit system[1] there exist 2^n computational basis states $|i\rangle$ with $i \in \{0,1\}^n$. Analogously, the state of an n-qubit system is described by 2^n complex amplitudes $\alpha_i \in \mathbb{C}$ such that $|\varphi\rangle = \sum_{i\in\{0,1\}^n} \alpha_i |i\rangle$ and $\sum_{i\in\{0,1\}^n} |\alpha_i|^2 = 1$—which can again be interpreted as a state vector, i.e., $|\varphi\rangle \equiv [\alpha_{0\ldots0}, \ldots, \alpha_{1\ldots1}]^\top$.

Example 1. *Consider a two-qubit system whose state is described by the state vector*

$$1/\sqrt{2}\,[1,\,0,\,0,\,1]^\top \equiv 1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|11\rangle\,.$$

This is a valid quantum state, since $|1/\sqrt{2}|^2 + |1/\sqrt{2}|^2 = 1$. *Furthermore, it demonstrates a key phenomenon unique to quantum computing called* entanglement. *While the complete*

[1] Throughout this work, big-endian convention is used for ordering the qubits, i.e., we interpret a certain basis state as $|q_{n-1}\ldots q_0\rangle$, where q_{n-1} denotes the "most-significant" qubit.

978-3-9819263-5-4/DATE21/©2021 EDAA

system's state can be accurately described (by the above state vector), its individual parts, i.e., the state of the individual qubits, cannot. More precisely, the state $|\varphi\rangle$ cannot be split into $|q_1\rangle \otimes |q_0\rangle$, where $|q_i\rangle$ describes the state of the i^{th} qubit and $\otimes$ denotes the tensor product of both vectors.

In a real quantum system, the individual amplitudes α_i are not directly observable. Instead, a *measurement* operation collapses the system's state to one of the computational basis states $|i\rangle$—each with probability $|\alpha_i|^2$—which can then be read-out classically.

Example 2. *The state $1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|11\rangle$ from Ex. 1 represents an equal superposition of two basis states. Measuring one qubit of this state would yield $|0\rangle$ in 50% of the cases and $|1\rangle$ otherwise. The resulting value for the second qubit is completely determined by the output of the first measurement due to the entanglement of the two qubits.*

The state of a quantum system is manipulated by *quantum operations* (often also referred to as *quantum gates*). Typically, these operations only operate on a small subset of a system's qubits. An operation acting on $k \leq n$ qubits (most frequently $k = 1$ or $k = 2$) is described by a $2^k \times 2^k$ unitary matrix[2] U. Applying such an operation to an n-qubit system in the state $|\varphi\rangle$ corresponds to extending the (local) $2^k \times 2^k$ matrix to a $2^n \times 2^n$ *(system) matrix* using tensor products and, then, calculating the matrix-vector product $U\,|\varphi\rangle$—resulting in a new state $|\varphi'\rangle$.

Example 3. *Consider the all-zero state $|\varphi\rangle = |00\rangle \equiv [1,0,0,0]^\top$ and the single-qubit Hadamard operation, whose action is described by the unitary 2×2 matrix H shown in Fig. 1(a). Assume this operation is to be applied to the "most-significant" qubit of the two-qubit system above. Then, the full $2^2 \times 2^2$ matrix corresponding to this operation is given by $H \otimes \mathbb{I}_2$, where $\mathbb{I}_2$ denotes the 2×2 identity matrix. Its application to the state vector $|\varphi\rangle$ then yields the new state $|\varphi'\rangle = 1/\sqrt{2}\,[1,0,1,0]^\top = 1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|10\rangle$.*

Quantum computations are just sequences of quantum operations applied to the state of a system. This is predominantly described by *quantum circuits*. There, horizontal wires indicate the system's individual qubits, while gates that are placed on these wires indicate the sequence of operations to apply.

Example 4. *Fig. 1(c) shows one of the simplest quantum circuits comprised of two qubits q_0, q_1 and two gates g_0, g_1. The first gate represents a* Hadamard *operation (as already discussed in Ex. 3), while the second operation represents a* controlled-NOT *operation whose corresponding matrix is shown in Fig. 1(b). A negation of the qubit state is applied to a target qubit (indicated by $\oplus$) if and only if certain control qubits (indicated by $\bullet$) are in state $|1\rangle$.*

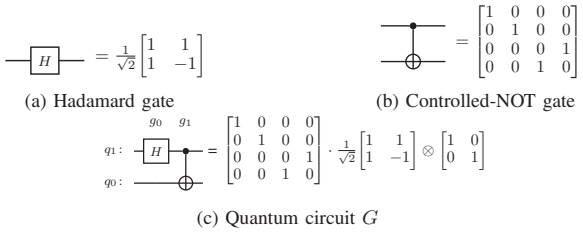
Executing the quantum circuit $G = g_0 \dots g_{m-1}$ for a given initial state $|\varphi\rangle$ (also called *simulation* when conducted on a classical computer) leads to an evolution of the state $|\varphi\rangle$ according to $U_{m-1} \cdots U_0\,|\varphi\rangle = U\,|\varphi\rangle = |\varphi'\rangle$.

Example 5. *Consider again the circuit G shown in Fig. 1(c) and assume that the computation starts out in the all-zero state $|00\rangle$. Consecutively multiplying the corresponding gate matrices depicted in Fig. 1(c) to the state vector leads to the following evolution of states:*

$$|00\rangle \xrightarrow{H\otimes\mathbb{I}_2} 1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|10\rangle \xrightarrow{CNOT} 1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|11\rangle\,,$$

which precisely yields the quantum state illustrated in Ex. 1.

Since each gate g_i represents a corresponding unitary matrix U_i and this property is preserved through matrix multiplication, the functionality of a given circuit $G = g_0 \dots g_{m-1}$ can be obtained as a unitary *system matrix* U itself by determining $U = U_{m-1} \cdots U_0$. This is essential concerning the *verification* of quantum circuits, i.e., the question whether two given circuits G and G' indeed realize the same functionality. Deciding the equivalence of two quantum computations can be reduced to comparing their system matrices U and U'.



(a) Hadamard gate (b) Controlled-NOT gate

(c) Quantum circuit G

Fig. 1. Quantum operations and their unitary matrices

III. Decision Diagrams and Their Applications

State vectors and operation matrices of a quantum system are exponential in size with respect to the number of qubits—quickly rendering the representation of a system state or the construction of a system matrix U an extremely difficult task. *decision diagrams* have been proposed as an efficient way for representing and manipulating quantum functionality [10]–[21]. While they are still exponential in the worst-case, decision diagrams have been shown to lead to very compact representations in many cases. In the following, we review how decision diagrams for quantum computing work and how they can be applied to the problems of *simulation* and *verification*.

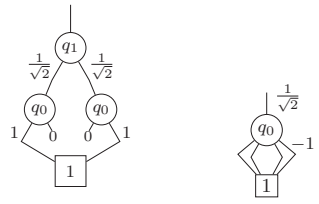
A. Decision Diagrams

As discussed in Section II, the state of an n-qubit system is represented by a state vector of size 2^n—an exponential representation. However, the inherent tensor product structure of many quantum states and redundancies in their description provide ground for a more compact representation. To this end, a given state vector with its complex entries is decomposed into sub-vectors according to

$$\begin{array}{ccccc} & & [\alpha_{00\dots0}, \dots, \alpha_{1\dots1}]^\top & & \\ & [\alpha_{0x}]^\top & & [\alpha_{1x}]^\top & \\ [\alpha_{00y}]^\top & & [\alpha_{01y}]^\top & [\alpha_{10y}]^\top & [\alpha_{11y}]^\top \\ & & \vdots & & \end{array}$$

with $x \in \{0,1\}^{n-1}$ and $y \in \{0,1\}^{n-2}$, until only complex numbers remain. This gives rise to a decision diagram structure with n levels of nodes (labeled q_{n-1} to q_0) and the individual amplitudes as its leaves. Each node has exactly two successors—indicating whether the corresponding path leads to an amplitude where qubit q_i is in the state $|0\rangle$ or $|1\rangle$, respectively. During these decompositions, equivalent sub-vectors can be represented by the same node—allowing

[2] A complex matrix U is unitary if $U^\dagger U = UU^\dagger = \mathbb{I}$, where $U^\dagger$ denotes the conjugate-transpose of U and $\mathbb{I}$ the identity matrix.



(a) $|\varphi\rangle = 1/\sqrt{2}\,[1,\ 0,\ 0,\ 1]^\top$ (b) Hadamard gate (c) Controlled-NOT gate

Fig. 2. Decision diagram representations for quantum states and operations

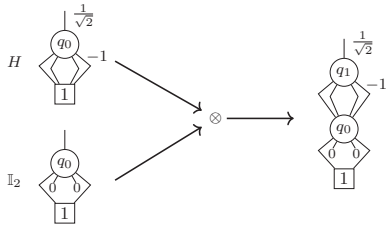


Fig. 3. Creation of $H \otimes \mathbb{I}_2$ using decision diagrams

for sharing and, hence, a reduction of the complexity of the representation. Further compaction is achieved by introducing edge weights and employing normalization schemes[3], in order to unify sub-vectors only differing by a common factor and further exploit possible redundancies. The amplitude of a given basis state can then be reconstructed from the multiplication of the edge weights along the path from the root node to the terminal node. An example illustrates the idea.

Example 6. *Consider again the state* $|\varphi\rangle \equiv 1/\sqrt{2}\,[1,\ 0,\ 0,\ 1]^\top$. *Recursively splitting this vector into sub-vectors results in a decision diagram as shown in Fig. 2(a). It consists of 3 nodes (the terminal node is usually not counted towards a decision diagram's size). The two paths leading from the root edge to the terminal node encode the states* $|00\rangle$ *and* $|11\rangle$ *respectively—each with an amplitude of* $1/\sqrt{2} \cdot 1 = 1/\sqrt{2}$. *Sub-vectors composed solely of* 0 *entries are typically denoted by* 0*-stubs, while edge weights equal to* 1 *are frequently omitted.*

A similar construction is employed for representing quantum operations. Instead of two successors, each node in a decision diagram representing the matrix U of an operation has four successors—corresponding to four equally sized sub-matrices U_{ij}. At each level l, this splitting corresponds to the action of U depending on the value the qubit q_l, i.e., U_{ij} describes how the rest of the system is transformed given that q_l is mapped from $|j\rangle$ to $|i\rangle$ for $i, j \in \{0, 1\}$.

Example 7. *Consider again the single-qubit Hadamard operation and the two-qubit controlled-NOT operation shown in Fig. 1(a) and Fig. 1(b), respectively. Their corresponding representations as decision diagrams are shown in Fig 2(b) and Fig. 2(c), respectively. To this end, the first (second) edge points to the node corresponding to the upper-left (upper-right) sub-matrix* U_{00} *(*U_{01}*), while the third (fourth) edge points to the lower-left (lower-right) sub-matrix* U_{10} *(*U_{11}*).*

As discussed in Section II, individual gate matrices have to be extended to the full system size using tensor products before being applied to the current state of a system. This extension is a natural operation on decision diagrams. Given two decision diagrams representing matrices U and V, their tensor product $U \otimes V$ is obtained by just replacing the terminal node in the decision diagram of U with the root node of V's decision diagram (and potentially relabelling the nodes).

Example 8. *In Ex. 3 the* 2×2 *matrix of the Hadamard gate was extended to a* 4×4 *representation by computing the tensor product with the* 2×2 *identity matrix* $\mathbb{I}_2$. *Fig. 3 now illustrates this process using decision diagrams.*

Decision diagrams do not only allow one to efficiently represent quantum states and matrices, but also to manipulate it. In the following, we illustrate their application for two particularly important design task in quantum computing: *simulation* and *verification*.

B. Simulation

Recall that the simulation of a quantum computation entails the consecutive calculation of the matrix-vector product between the state vector $|\varphi\rangle$ and the current operation matrix U_i until all operations have been applied. The following example sketches how matrix-vector multiplication is realized on decision diagrams.

Example 9. *The multiplication of a (unitary) matrix* U *and a (state) vector* $|\varphi\rangle$ *can be broken down into sub-computations according to:*

$$\begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix} \cdot \begin{bmatrix} \alpha_{0x} \\ \alpha_{1x} \end{bmatrix} = \begin{bmatrix} (U_{00}\alpha_{0x} + U_{01}\alpha_{1x}) \\ (U_{10}\alpha_{0x} + U_{11}\alpha_{1x}) \end{bmatrix}.$$

Now, the U_{ij}*-submatrices precisely correspond to the four successors of the matrix's root node, while the* α_{ix} *correspond to the two successors of the state vector's root node. This is illustrated in Fig. 4. Thus, by recursively decomposing these sub-computations further until only operations on complex numbers remain, an efficient scheme for matrix multiplication using decision diagrams is devised* [4].

Measuring the resulting state, i.e., sampling from the corresponding decision diagram, can be efficiently conducted by a randomized single-path traversal of the decision diagram [16]. At each node, the squared magnitude of the left (right) successor gives the probability of the qubit associated to the node being $|0\rangle$ ($|1\rangle$), while the probability of an individual basis state is the product of all probabilities along the path. In contrast to quantum computations on real quantum computers, measurements of classically simulated quantum states are non-destructive, i.e., they can be repeated on the same state without having to repeat the whole calculation.

C. Verification

In order to realize a conceptual quantum algorithm on an actual device, the algorithm's description is transformed through various levels of abstraction—including steps usually called compilation, synthesis, transpilation, mapping, and/or similar. To this end, several methods have been proposed [23]–[27]. During this process, it is of utmost importance to guarantee that the resulting circuit is still functionally equivalent to the

[3]Normalization can be performed by, e.g., dividing the weight of the outgoing edges of a node by the norm of the vector containing both edge weights and multiplying this factor to the incoming edges—allowing for efficient sampling from the resulting decision diagram [16].

[4]Decision diagram packages employ *unique tables* and *compute tables* for the decision diagram nodes as well as the complex edge weights in order to further exploit possible redundancies and to reduce the number of computations necessary, see, e.g., [14].

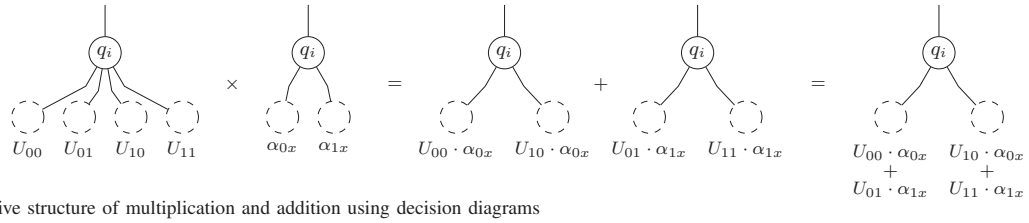
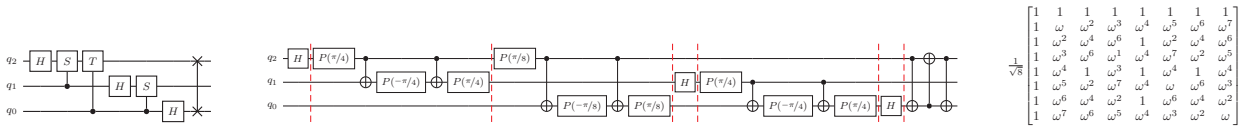


Fig. 4. Recursive structure of multiplication and addition using decision diagrams



(a) Three-qubit QFT (b) Compiled circuit (c) Functionality

Fig. 5. The Quantum Fourier Transformation and its functionality

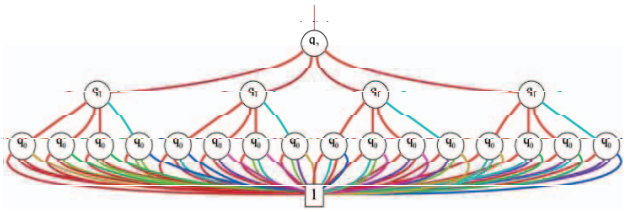


Fig. 6. Decision diagram for the functionality of the three-qubit QFT

original algorithm. Recall that the functionality of a quantum circuit is described by the unitary system matrix U arising from the matrix-matrix multiplications of the individual gate descriptions. Thus, the equivalence of two circuits can be reduced to the comparison of their system matrices. In the following, we illustrate this verification scenario using the *Quantum Fourier Transform* (QFT, a popular building block in many quantum algorithms) [22] as an example.

Example 10. *The circuit of the three-qubit QFT is shown in Fig. 5(a). It consists of Hadamard gates, controlled phase gates, i.e., rotations with an angle that is a certain fraction of π (e.g., $S = p(\pi/2)$, $T = p(\pi/4)$) controlled on the value of another qubit, and a SWAP gate (which swaps the value of the two qubits indicated by $\times$). The latter two types of gates are not native to any current quantum computer and, thus, need to be compiled into sequences of gates that are supported. Fig. 5(b) shows one possible compiled version of the abstract QFT circuit. Both circuits realize the functionality described by the 8×8 matrix shown in Fig. 5(c), where $\omega = e^{i\pi/4} = \sqrt{i} = {}^{1+i}/\sqrt{2}$.*

While conceptually simple, the exponential size of the involved matrices quickly renders techniques purely based on matrices infeasible. Decision diagrams are a prominent candidate for checking the equivalence of two circuits since they (1) can represent quantum functionality in a very compact fashion in many cases, and, (2) still offer a canonic representation (with respect to a given variable order and normalization scheme). Thus, the equivalence of two decision diagrams can be concluded by comparing their root pointers (and the corresponding edge weight).

Example 11. *Constructing the decision diagrams for the two circuits shown in Fig. 5(a) and Fig. 5(b), respectively, results in a decision diagram as shown in Fig. 6 in both cases*[5]*. Hence, both circuits are considered equivalent.*

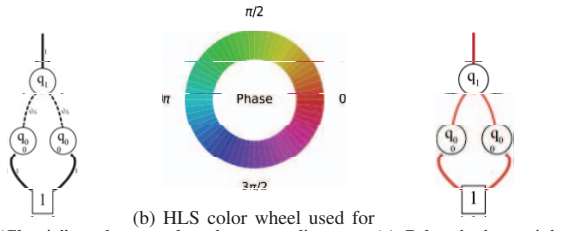
As seen in the previous example, decision diagrams can still grow exponentially large in the worst case—again presenting a severe obstacle for verifying the correctness of circuits. However, as recently shown in [20], this complexity can be drastically reduced in many cases by exploiting the inherent reversibility of quantum operations. The general idea is as follows: If two quantum circuits G and G' are equivalent, then concatenating the first circuit G with the inverse G'^{-1} of the second circuit would realize the identity function $\mathbb{I}$. The potential now lies in the order in which the operations from either circuit are applied. Whenever a strategy can be employed so that the respective gates from G and G' are applied in a fashion frequently yielding the identity, the entire procedure can be conducted rather efficiently (e.g., in case of verifying the results of compilation flows [28]).

Example 12. *Consider again the two circuits realizing the QFT shown earlier in Fig. 5. Then, their equivalence can be concluded by (1) starting with a decision diagram resembling the identity, (2) applying one gate from the circuit shown in Fig. 5(a), and, (3) applying all gates from the circuit shown in Fig. 5(b) up to the next barrier (indicated by dashed lines in Fig. 5(b)). The resulting decision diagram resembles the identity, and hence the equivalence of both circuits can be concluded. Conducting the verification in this fashion only requires a maximum of 9 nodes (as opposed to 21 nodes for building the entire system matrix).*

IV. Visualization

In the previous sections, we have shown that decision diagrams provide a promising basis for many design tasks such as simulation and verification. However, users of the corresponding tools often do not have a corresponding background or an intuition about how these methods based on decision diagrams work and what their strengths and limits are. In

[5]Note that, for sake of clarity, edge weights are not explicitly annotated to this decision diagram anymore. Instead, a color code is used (also explained in detail later in Section IV and Fig. 7(b)).



(a) "Classic" mode (b) HLS color wheel used for complex phase encoding (c) Colored edge weights

Fig. 7. Visualization options for vector decision diagrams

an effort to make decision diagrams for quantum computing more accessible, we developed a tool which visualizes quantum decision diagrams and allows to explore their behavior when used in the design tasks covered above. To keep the effort of using the visualization as small as possible, the tool has been implemented as a web tool which can be started by simply accessing https://iic.jku.at/eda/research/quantum_dd/tool. In the following, we illustrate how our tool (1) visualizes decision diagrams for vectors and matrices, (2) can be used for simulating quantum circuits, and (3) can be used for verifying quantum circuits.

A. Decision Diagrams

In order to provide the most accessible user interface possible, the tool offers several options for customizing how decision diagrams are visualized. Fig. 7 illustrates the available styles for decision diagrams representing vectors, i.e., states of a quantum system. The "classic" mode (see Fig. 7(a)) offers a look and feel that is most similar to what is found in research papers (as, e.g., shown throughout Section III). Edges with a corresponding weight not equal to 1 are drawn using dashed lines and 0-stubs are retracted into the nodes themselves. Since the explicit annotation of edge weights quickly requires lots of space and leads to unreadable decision diagrams, there is also an option for removing these edge labels. Instead the magnitude of an edge weight can be reflected by the thickness of the line, while its complex phase can be color-coded using the HLS color wheel shown in Fig 7(b). Examples using this color code are shown in Fig. 6 and Fig. 7(c). In addition to the above, the tool provides a more "modern" look for the decision diagram nodes, where the connection to the underlying state vector is expressed in a more straight-forward fashion (shown later in Fig. 8). Such a "modern" look is also available for matrices (shown later in Fig. 9). These should allow less accommodated users to more easily grasp the concept of decision diagrams. In the following, we provide a deeper look into the individual features of the tool and how simulation and verification can be conducted using the tool.

B. Simulation

The simulation feature of the tool provides a settings panel, an algorithm box for entering or loading a quantum algorithm/circuit, and an interactive decision diagram box which displays the current system state in terms of a decision diagram. Loading quantum algorithms/circuits into the tool is as easy as drag- and dropping an algorithm/circuit file (in either *.qasm* or *.real* format) into the corresponding algorithm box, or starting to enter your own description using one of the templates provided in the "Example Algorithms" list. Once a valid algorithm/circuit has been entered/loaded in the algorithm box, the simulation can be controlled using the navigation buttons below it, i.e.:

- → / ← : Go one step forward or backward. Can be used to step through the simulation.
- ↠ / ↞ : Go straight to the end (or the next special operation; see below) or back to the beginning.
- ▷/|| : Start/Pause a slide show where the simulation advances step-by-step in an automated fashion.

Some operations are considered *special operations* since they do not directly correspond to the application of a unitary matrix:

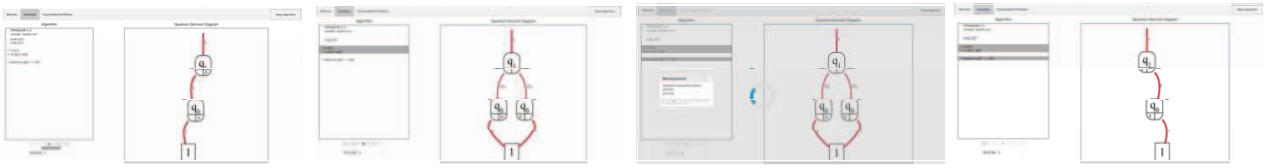
- Barrier statements (e.g., *"barrier q;"*) can be used as breakpoints when stepping forward with ↠.
- Measurement operations (e.g., *"measure q[0] -> c[0];"*) collapse the state of a qubit to one of its basis states. Whenever a qubit is about to get measured and it has a non-zero probability of being in either $|0\rangle$ or $|1\rangle$ (i.e., it is in superposition), a pop-up dialog appears which displays the probabilities for obtaining $|0\rangle$ and $|1\rangle$, respectively. Upon choosing one of the options, the decision diagram is irreversibly collapsed. Measurements also act as breakpoints due to their non-unitary (and, thus, non-reversible) nature. The tool supports OpenQASM's classically-controlled operations, where a certain gate is only applied if some classical bits obtained from measurements are set.
- Reset operations (e.g., *"reset q[0];"*) discard a qubit and initialize it to $|0\rangle$ as if it were a new qubit. Mathematically, this corresponds to taking the partial trace of the whole state and, then, setting the qubit to $|0\rangle$. However, the partial trace maps pure states to mixed states and can thus in general not be represented by the same kind of decision diagram used for representing state vectors. The tool supports resets in a probabilistic fashion (similar to measurements). Whenever a reset operation is encountered where the considered qubit has a non-zero probability of being in either $|0\rangle$ or $|1\rangle$, a pop-up dialog appears which displays the probabilities for obtaining $|0\rangle$ and $|1\rangle$, respectively. Upon choosing one of the options, the other decision diagram branch is discarded and the remaining branch is set as the $|0\rangle$ branch. Resets also act as breakpoints due to their non-unitary (and, thus, non-reversible) nature.

Example 13. *Fig. 8 illustrates the process of simulating the quantum circuit shown in Fig. 1(c) using the tool. The first screenshot (Fig. 8(a)) shows the initial quantum circuit and its state $|00\rangle$. Then, the two gates are applied—yielding the resulting state $1/\sqrt{2}|00\rangle + 1/\sqrt{2}|11\rangle$ (Fig. 8(b)). If, now, the first qubit is to be measured, there is a 50% chance of it being either $|0\rangle$ or $|1\rangle$ (Fig. 8(c)). Assume, that the measurement outcome is $|1\rangle$. Then, the value of the second qubit is completely determined due to the entanglement of both qubits—resulting in the final state $|11\rangle$ (Fig. 8(d)).*

C. Verification

The verification feature of the tool provides a similar settings panel and decision diagram box as the simulation tab, but now features two algorithm boxes. In case only one algorithm/circuit is loaded in the left (right) algorithm box, the tool can be used to build the (inverse) functionality of the corresponding circuit.

Example 14. *Creating the QFT circuit shown in Fig. 5(a) in the left algorithm box and applying all the operations precisely yields the decision diagram shown in Fig. 6.*



(a) Initial state $|00\rangle$ (b) Resulting state $1/\sqrt{2}\,|00\rangle + 1/\sqrt{2}\,|11\rangle$ (c) Measuring qubit q_0 (d) Post-measurement state $|11\rangle$

Fig. 8. Visualizing the simulation of the circuit shown in Fig. 1(c)

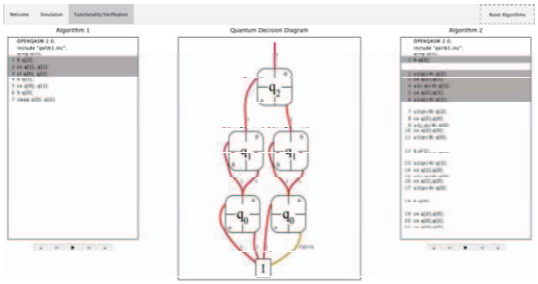


Fig. 9. Visualizing the verification of the QFT circuits shown in Fig. 5

Once a valid algorithm/circuit has been entered in each of the algorithm boxes, their equivalence can be checked by successively applying operations from both circuits (using the corresponding $\rightarrow$ / $\twoheadrightarrow$ controls) and checking whether the final result resembles the identity. As for the simulation, *Barrier* statements can be used as breakpoints when stepping through both algorithms/circuits. In contrast to simulation, *Measurement*, *Reset*, and *Classicaly-Controlled Operations* are currently not supported due to their non-unitary nature. When verifying two algorithms/circuits, the tool expects both algorithms/circuits to have the same number of qubits and the same variable order. If you want to verify algorithms/circuits using different numbers of ancillary and/or garbage qubits and/or different variable orderings, you can check out our fully-fledged equivalence checking tool JKQ QCEC (available at https://github.com/iic-jku/qcec).

Example 15. *Consider again the two circuits realizing the three-qubit QFT shown in Fig. 5(a) and Fig. 5(b). Fig. 9 shows how the tool is used to verify the equivalence of both circuits. The first three gates have already been applied from the left circuit, while six operations have been applied from the right circuit. The corresponding decision diagram in the middle only slightly differs from the identity (by the right node labeled q_0). Continuing the computation as discussed in Ex. 12 eventually allows to verify the equivalence of both circuits while remain close to the identity throughout the whole process.*

V. Conclusions

In an effort to bridge the gap between the design automation and the quantum community and to make decision diagrams for quantum computing more accessible, we provided a comprehensive summary on how decision diagram techniques can be used for the design of quantum circuits. Additionally, we have presented an easily-accessible tool which visualizes quantum decision diagrams and their applications. The corresponding tool is hosted at https://iic.jku.at/eda/research/quantum_dd/tool. We sincerely hope that our efforts help interested researchers to learn and adopt these techniques.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

[1] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019.
[2] H.-S. Zhong *et al.* (2020). "Quantum computational advantage using photons." arXiv: 2012.01625.
[3] Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, 1986.
[4] R. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 293–318, 1992.
[5] I. Wegener, *Branching programs and binary decision diagrams: theory and applications*. SIAM Jour. of Comp., 2000.
[6] J. Gergov and C. Meinel, "Efficient boolean manipulation with OBDD's can be extended to FBDD's," *IEEE Trans. Comput.*, vol. 43, no. 10, pp. 1197–1209, 1994.
[7] R. Drechsler *et al.*, "Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams," in *Design Automation Conf.*, 1994, pp. 415–419.
[8] R. I. Bahar *et al.*, "Algebraic decision diagrams and their applications," in *Int'l Conf. on CAD*, 1993, pp. 188–191.
[9] S. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," in *Design Automation Conf.*, 1993, pp. 272–277.
[10] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Improving gate-level simulation of quantum circuits," *Quantum Information Processing*, vol. 2, no. 5, pp. 347–380, 2003.
[11] D. Miller and M. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, 2006.
[12] 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.
[13] P. Niemann *et al.*, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 86–99, 2016.
[14] 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.
[15] 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.
[16] S. Hillmich, I. L. Markov, and R. Wille, "Just Like the Real Thing: Fast Weak Simulation of Quantum Computation," in *Design Automation Conf.*, 2020.
[17] P. Niemann, R. Wille, and R. Drechsler, "Efficient synthesis of quantum circuits implementing Clifford group operations," in *Asia and South Pacific Design Automation Conf.*, 2014, pp. 483–488.
[18] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006.
[19] M. Soeken *et al.*, "Synthesis of reversible circuits with minimal lines for large functions," in *Asia and South Pacific Design Automation Conf.*, 2012, pp. 85–92.
[20] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2021.
[21] K. N. Smith and M. A. Thornton, "Quantum logic synthesis with formal verification," *IEEE Int Midwest Symp Circuits Syst.*, pp. 73–76, 2019.
[22] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.
[23] 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.
[24] 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.
[25] P. Murali *et al.*, "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.
[26] 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.
[27] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *Int'l Conf. on CAD*, 2020.
[28] 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.