# Advanced Equivalence Checking for Quantum Circuits

Lukas Burgholzer* *Student Member, IEEE* Robert Wille*† *Senior Member, IEEE*
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†Software Competence Center Hagenberg GmbH (SCCH), Austria
lukas.burgholzer@jku.at robert.wille@jku.at

***Abstract*—In the not-so-distant future, quantum computing will change the way we tackle certain problems. It promises to dramatically speed-up many chemical, financial, cryptographical, and machine-learning applications. However, in order to capitalize on those promises, complex design flows composed of steps such as compilation, decomposition, mapping, or transpilation need to be employed before being able to execute a conceptual quantum algorithm on an actual device. This results in many descriptions at various levels of abstraction which may significantly differ from each other. The complexity of the underlying design problems makes it ever more important to not only provide efficient solutions for the single steps, but also to verify that the originally intended functionality is preserved throughout all levels of abstraction. This motivates methods for equivalence checking of quantum circuits. However, most existing methods for this are inspired by equivalence checking in the classical realm and have merely been extended to support quantum circuits (i.e., circuits which do not only rely on 0's and 1's, but also employ superposition and entanglement).**

**In this work, we propose an advanced methodology which takes the different paradigms of quantum circuits not only as a burden, but as an opportunity. In fact, the proposed methodology explicitly utilizes characteristics unique to quantum computing in order to overcome the shortcomings of existing approaches. We show that, by exploiting the reversibility of quantum circuits, complexity can be kept feasible in many cases. Moreover, we show that, in contrast to the classical realm, simulation is very powerful in verifying quantum circuits. Experimental evaluations confirm that the resulting methodology allows one to conduct equivalence checking dramatically faster than ever before—in many cases just a single simulation run is sufficient. An implementation of the proposed equivalence checking flow is publicly available at https://iic.jku.at/eda/research/quantum_verification/.**

## I. Introduction

Quantum computers [1] aim to change the way we tackle certain problems in the future. As more and more big companies like Google, IBM, Intel, and Microsoft as well as start-ups like Rigetti and IonQ set foot in this domain, the need for design methods that allow to use this new technology is steadily increasing. Without appropriate methods, we might reach a point where we have quantum computers readily available but no means to actually use them. In order to utilize the theoretical advantage of quantum computers in practice, a multitude of (computationally complex) design tasks have to be conducted—resulting in descriptions of quantum algorithms at various abstraction levels which may significantly differ in their basis operations and structure. This is similar to the classical realm where, e.g., descriptions at the *Electronic System Level*, the *Register Transfer Level*, and the *Gate Level* exist.

More precisely, a high-level description of a quantum algorithm has to be *compiled* to a low-level description satisfying all constraints imposed by the targeted device. Today's quantum computers only support a very limited (yet universal) set of quantum operations natively. Thus, non-native operations first have to be *decomposed* into sequences of native operations [2]–[6]. Furthermore, most quantum computers limit the pairs of qubits (the main computational unit in quantum computing) that may directly interact with each other. Realizing a generic quantum algorithm on such a device therefore requires a *mapping* step (sometimes also called *transpilation* or *qubit routing*), where a given circuit is made compliant to the imposed connectivity constraints by inserting $SWAP$ or $H$ gates [7]–[14]. Finally, current devices are heavily affected by noise and decoherence effects and are considered part of the *Noisy-Intermediate-Scale-Quantum* (NISQ) era of quantum computing [15]. On the one hand, this motivates quantum circuit optimizations, such as gate fusion, gate cancellation, or block-wise re-synthesis—which aim to reduce the overall gate count of circuits to be executed in order to reduce the effect of noise and allow the computation to stay coherent [16]–[24]. On the other hand, mapping techniques which take the targeted device's calibration and error data into account to achieve noise-adaptive mappings shift into focus [25], [26].

All this substantially changes the circuit description during the design process. At the same time, all these steps should obviously preserve the originally intended functionality of the quantum circuit—even if other basic operations and structures are eventually used for the realization. In order to check that, *equivalence checking* is usually conducted to prove whether two circuits (the originally given quantum algorithm and the quantum circuit resulting from the compilation process) are equivalent or to determine a counterexample showing the non-equivalence between them. In the classical realm, equivalence checking is conducted using design automation expertise leading to efficient methods in order to guarantee correctness throughout the design [27]–[30].

Inspired by these methods, several solutions for equivalence checking of quantum circuits have been proposed in the recent past, e.g., based on re-writing [24], [31], Boolean satisfiability [32], and decision diagrams [33]–[36]. However, all of them merely extended classical methods in order to additionally support quantum circuits (i.e., are extended to support superposition or entanglement) and, by this, take the different computation paradigm only as a burden to be addressed. Consequently, those methods remain unsatisfactory in many cases.

In this work[1], we propose a different take. Rather than a burden, we see the computation paradigm of quantum circuits as an opportunity. We propose an advanced equivalence checking methodology which explicitly utilizes characteristics unique to quantum computing in order to substantially improve existing approaches. More precisely, we unearth potential which rests on the following two observations:

[1] Preliminary versions of this work have been published in [37], [38].

Quantum circuits are inherently reversible. Because of that, 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 $I$, i.e., $G \cdot G'^{-1} = I$. 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 since the identity constitutes the best case for most representations of quantum functionality (e.g., linear in the number of qubits for decision diagrams).

Moreover, even in the case where the two considered quantum circuits are not equivalent, quantum characteristics can be exploited. In fact, we observed that, again due to the inherent reversibility of quantum operations, even small differences in quantum circuits frequently affect the *entire* functional representation. Hence, it may not always be necessary to check the complete functionality, but it is highly likely that the simulation of both computations with a couple of arbitrary input states (i.e., considering only a small part of the whole functionality) will already provide a counterexample showing the non-equivalence. This is in stark contrast to the classical realm, where the inevitable information loss introduced by many logic gates and the resulting masking effects often require a complete consideration of *all* possible input states or sophisticated schemes for constraint-based stimuli generation [39]–[42], fuzzing [43], [44], etc.

Both observations provide the nucleus of an advanced equivalence checking methodology in which the non-equivalence is often detected by a few simulation runs (which can be conducted dramatically faster than the actual equivalence check). Moreover, passing several simulation runs leading to the same results for both circuits provides an indication (albeit no proof) that the circuits might be equivalent. The proof itself can, afterwards, be significantly accelerated by using strategies which keep the check for $G \cdot G'^{-1} = I$ close to the identity $I$. Combining these complementary ideas into a comprehensive equivalence checking flow allows for efficient verification of quantum circuits.

Experimental evaluations confirm that the resulting flow allows one to conduct equivalence checking dramatically faster than ever before—in many cases, just a single simulation run is sufficient. By this, we do not only show ways to handle the complexity of verifying quantum circuits, but also show the potential of simulation for this task. An implementation of the proposed equivalence checking flow is publicly available at https://iic.jku.at/eda/research/quantum_verification/.

The remainder of this paper is structured as follows: Section II reviews the background needed to keep this work self-contained. Section III motivates the considered problem and discusses the related work. Then, Section IV introduces and illustrates the general ideas proposed in this work. Based on that, Section V describes the dedicated equivalence checking schemes resulting from the ideas and provides a theoretical discussion on the power of simulation for equivalence checking of quantum circuits. All these findings eventually result in an advanced equivalence checking methodology which is described and discussed in Section VII. Finally, Section VIII summarizes the obtained experimental results before Section IX concludes the paper.

## II. Background

In this section, we review the main concepts of quantum computing and illustrate decision diagrams as one way of efficiently representing and manipulating quantum functionality. While the following descriptions are kept brief, we refer to [1] and [34] for more details on either topic.

### A. Quantum Computing

The main computational unit in quantum computing is the *qubit*. In contrast to classical bits, a qubit cannot only be in one of the computational basis states $|0\rangle$ or $|1\rangle$ (written in Dirac notation), 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 a system of $n$ qubits, there exist $2^n$ computational basis states $|i\rangle$ with $i$ from $0$ to $2^n - 1$. Analogously, the state of an $n$-qubit system is described by $2^n$ complex amplitudes $\alpha_i \in \mathbb{C}$ such that $|\varphi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ and $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$—which can again be interpreted as a state vector, i.e., $|\varphi\rangle \equiv [\alpha_0, \ldots, \alpha_{2^n-1}]^\top$.

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

$$\tfrac{1}{\sqrt{2}}[1, 0, 0, 1]^\top \equiv \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|3\rangle = \tfrac{1}{\sqrt{2}}|(00)_2\rangle + \tfrac{1}{\sqrt{2}}|(11)_2\rangle.$$

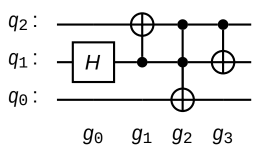
*This is a valid quantum state, since* $|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{\sqrt{2}}|^2 = 1$. *Furthermore, it demonstrates a key phenomenon unique to quantum computing called* entanglement*. While the complete system's state can be accurately described (by the above statevector), its individual parts, 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 $\alpha_i$ 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* $\frac{1}{\sqrt{2}}|(00)_2\rangle + \frac{1}{\sqrt{2}}|(11)_2\rangle$ *from Ex. 1 represents an equal superposition of two basis states. Measuring this state would yield* $|(00)_2\rangle$ *in half of the cases and* $|(11)_2\rangle$ *otherwise.*

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

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



(a) Quantum circuit $G$

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

(b) System matrix $U$

Figure 1: Quantum computations

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 $|\varphi'\rangle$[3].

**Example 3.** *Consider a single-qubit operation (i.e., $k = 1$) represented by the unitary matrix $U_S \in \mathbb{C}^{2\times 2}$. Further assume that this operation is to be applied to the second qubit in a system consisting of $n = 3$ qubits. Then, the full $2^3 \times 2^3$ matrix corresponding to this operation is given by $I_2 \otimes U_S \otimes I_2$, where $I_2$ denotes the $2 \times 2$ identity matrix. Its application to the $2^3$-dimensional state vector $|\varphi\rangle \equiv [\alpha_0, \ldots, \alpha_7]^\top$ then yields the new state $|\varphi'\rangle$.*

*Quantum computations* are just sequences of quantum operations applied to the state of a system. This is predominantly described by *quantum circuits* and visualized as *quantum circuit diagrams*. There, horizontal wires indicate the system's individual qubits, while gates that are placed on these wires indicate the sequence of operations to apply—operating from left to right.

**Example 4.** *Fig. 1a shows a quantum circuit $G$ comprised of three qubits $q_0, q_1, q_2$ and four gates $g_0, g_1, g_2, g_3$. The first gate represents a* Hadamard *operation (indicated by a box with the label $H$), which can be used to create an equal superposition from a given basis state (cf. Ex. 2). The remaining three gates represent* controlled *operations. A specific operation (in this case 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 $|1\rangle$.*

Each gate $g_i$ represents a corresponding unitary matrix $U_i$ that is subsequently applied during the execution of a quantum circuit. Thus, the functionality of a given circuit $G = g_0 \ldots g_{m-1}$ can be obtained as a unitary *system matrix* $U$ itself by determining $U = U_{m-1} \cdots U_0$. Moreover, executing the quantum circuit 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 $U_{m-1} \cdots U_0|\varphi\rangle = U|\varphi\rangle = |\varphi'\rangle$. Note that, if $|\varphi\rangle = |i\rangle$ for some $i \in \{0, \ldots, 2^n - 1\}$, i.e., $|\varphi\rangle$ is a computational basis state, the simulation precisely calculates the $i^{th}$ column $u_i$ of $U$, i.e., $U|i\rangle = |u_i\rangle$.

**Example 5.** *Consider again the circuit $G = g_0 g_1 g_2 g_3$ shown in Fig. 1a. Consecutively multiplying the corresponding gate matrices $U_3 \cdot U_2 \cdot U_1 \cdot U_0$ eventually yields the system matrix $U$ shown in Fig. 1b. Given the initial state $|4\rangle = |(100)_2\rangle$, executing the quantum circuit $G$ precisely transforms this state to the state described by column four of $U$, i.e., $U|(100)_2\rangle = |u_4\rangle = \frac{1}{\sqrt{2}}[0\,0\,1\,0\,0\,0\,1\,0]^\top = \frac{1}{\sqrt{2}}|(010)_2\rangle + \frac{1}{\sqrt{2}}|(110)_2\rangle$.*

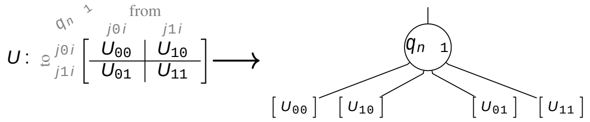


Figure 2: Decomposition scheme

Since quantum operations are inherently reversible, the inverse of a whole quantum circuit $G = g_0 \ldots g_{m-1}$ can always be determined by reversing the order of operations and inverting each individual operation, i.e., $G^{-1} = (g_0 \ldots g_{m-1})^{-1} = g_{m-1}^{-1} \ldots g_0^{-1}$—corresponding to the matrix operations $U_0^\dagger \cdots U_{m-1}^\dagger = U^\dagger$.

### B. Decision Diagrams for Quantum Computing

State vectors and operation matrices of a quantum system are exponential in size with respect to the number of qubits—quickly rendering the simulation of a quantum circuit $G$ and even more the construction of its system matrix $U$ an extremely difficult task. *Decision diagrams* (DDs) have been proposed as an efficient way for representing and manipulating quantum functionality [33], [34], [45]–[47]. While they are still exponential in the worst-case, decision diagrams have been shown to lead to very compact representations for the vectors and matrices in many cases. In the following, we review how decision diagrams can be used to compactly represent matrices (e.g., the system matrix $U$). As we will see, (state) vectors are just a special case of matrices that can be handled analogously.

Consider a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$. Then, $U$ can be split into four $2^{n-1} \times 2^{n-1}$-sized sub-matrices $U_{ij}$ as shown on the left side of Fig. 2. This splitting corresponds to the action of $U$ depending on the value of the topmost qubit $q_{n-1}$, i.e., $U_{ij}$ describes how the rest of the system is transformed given that $q_{n-1}$ is mapped from $|j\rangle$ to $|i\rangle$ for $i, j \in \{0, 1\}$. In the corresponding decision diagram, this manifests as a node with label $q_{n-1}$ and four successor nodes as shown on the right side of Fig. 2.

This decomposition scheme can now be applied recursively until only single matrix entries (i.e., complex numbers) remain—eventually forming the decision diagram's terminal nodes. During this process, identical sub-matrices can be represented by the same node which allows to capitalize on possible redundancies in the matrix $U$.

**Example 6.** *The most compact decision diagram is achieved by the identity $I$ since all sub-matrices $U_{00}$ and $U_{11}$ are recursively identical and* all *sub-matrices $U_{01}$ and $U_{10}$ are solely composed of 0-entries—yielding a decision diagram with exactly $n$ nodes as shown in Fig. 3 for three qubits*[4].

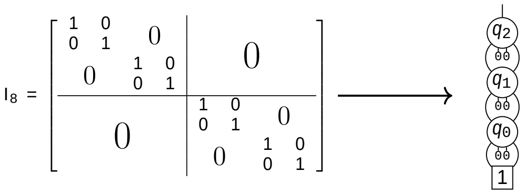


Figure 3: Identity matrix and DD for $n = 3$

[3] As a consequence, quantum operations are inherently reversible, i.e., $U^\dagger|\varphi'\rangle = U^\dagger U|\varphi\rangle = |\varphi\rangle$.

[4] Sub-matrices only containing zero entries are represented as 0-stubs.

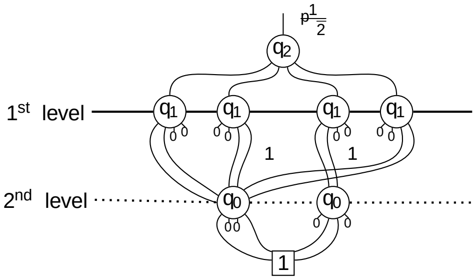


Figure 4: Decision diagram for $U$

Further compaction is achieved by employing edge weights in order to unify sub-matrices only differing by a common factor. As a consequence, one terminal node always suffices and, hence, is typically not counted towards the overall size (i.e., node count) of a decision diagram.

*Example 7.* Consider again the matrix $U$ as shown in Fig. 1b. Recursively decomposing this matrix yields the decision diagram shown in Fig. 4—representing the entire matrix $U$. While, at the first decomposition level (indicated by a thick bold line in Fig. 1b) all sub-matrices have a different structure, two nodes suffice to completely represent the entire functionality at the second decomposition level (indicated by a dotted line in Fig. 1b). Due to the utilization of edge weights and exploitation of redundancies, just seven nodes suffice to capture the whole functionality of $U$.

Constructing decision diagrams in this fashion results in a canonic representation of the functionality[5]. Moreover, vectors can be interpreted as matrices where each node's $U_{10}$ and $U_{11}$ successors are zero.

Decision diagrams do not only allow one to efficiently represent quantum functionality, but also to manipulate it. Here, we just illustrate how the most prominent operation—matrix-matrix multiplication—is conducted using decision diagrams. However, many of the typical operations on matrices and vectors can directly be carried over for decision diagrams in a similar fashion. This includes, but is not limited to, addition, (conjugate) transposition, fidelity and (partial) trace calculation. The multiplication of two matrices $U$ and $V$ is recursively broken down into sub-expressions according to

$$\begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix} \cdot \begin{bmatrix} V_{00} & V_{01} \\ V_{10} & V_{11} \end{bmatrix} = \begin{bmatrix} (U_{00}V_{00} + U_{01}V_{10}) & (U_{00}V_{01} + U_{01}V_{11}) \\ (U_{10}V_{00} + U_{11}V_{10}) & (U_{10}V_{01} + U_{11}V_{11}) \end{bmatrix},$$

until only operations on complex numbers remain. Note that the individual sub-matrices exactly correspond to the respective successors of the currently considered decision diagram node. Eventually, this allows decision diagrams to efficiently represent and manipulate quantum functionality in many cases.

## III. Motivation

As reviewed in Section I, the typical design flow for realizing a quantum algorithm on a real quantum computer requires several steps in which the desired functionality is decomposed, mapped, and optimized. During all these steps, it has to be ensured that the functionality of the correspondingly resulting circuit descriptions does not change—motivating equivalence checking for quantum circuits. In this section, we first give an explicit description of this problem. Then, we review how equivalence checking is conducted to date and discuss why current solutions are still unsatisfactory. Those discussions provide the overall motivation of our work.

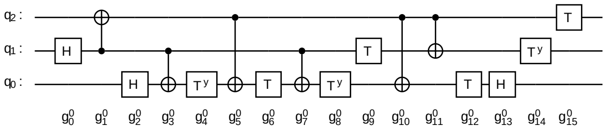


Figure 5: Alternative circuit realization $G'$

### A. The Quantum Circuit Equivalence Checking Problem

The functionality of a quantum algorithm is most prominently described in the form of a quantum circuit—potentially consisting of high-level operations, such as Grover iterations [48], the quantum Fourier transform [1], etc. Technically, one could also consider other "functional descriptions" such as a unitary matrix, a decision diagram, or a tensor network. However, those are usually internal representations used by tools and hardly a functional description provided by a designer. Consequently, 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, as reviewed in the previous section, the functionality of each circuit can uniquely be described by the respective system matrices $U = U_{m-1} \cdots U_0$ and $U' = U'_{m'-1} \cdots U'_0$, where the matrices $U_i^{(\prime)}$ describe the functionality of the $i$-th gate of the respective circuit (with $0 \leq i < m^{(\prime)}$). 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 [1]), i.e., $U = e^{i\theta} U'$ with $\theta \in [0, 2\pi)$.

*Example 8.* Consider again the circuit $G$ from Fig. 1a and an alternative realization $G'$ as shown in Fig. 5. Since the functionality of $G'$ exhibits the same system matrix $U$ as shown before in Fig. 1b, both circuits $G$ and $G'$ are indeed equivalent.

If $U$ and $U'$ differ in any column $i$ (by more than a global phase factor $e^{i\theta}$), then the corresponding circuits $G$ and $G'$ are not equivalent and $|i\rangle$ serves as a counterexample showing that

$$U|i\rangle = |u_i\rangle \not\equiv |u'_i\rangle = U'|i\rangle .$$

Here, the fidelity $F$ between two states $|x\rangle$ and $|y\rangle$ is typically used as a similarity measure for comparing quantum states, where $F$ is calculated as the squared overlap between the states, i.e., $F = |\langle x|y\rangle|^2 \in [0, 1]$. Two states are considered equivalent if the fidelity between them is 1 (up to a given tolerance $\varepsilon$).

[5] Canonicity is achieved for a fixed variable order and under the assumption that a suitable normalization scheme is employed. For example, nodes can be normalized through division by the leftmost non-zero edge weight.

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & \omega^3 & 0 & \omega^3 & 0 & 0 & 0 & 0 \\ \omega^3 & 0 & \omega^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \omega^3 & 0 & \omega^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & \omega^3 & 0 & \omega^3 \end{bmatrix}$$

Figure 6: Erroneous circuit matrix $\tilde{U}'$ ($\omega = \frac{1+i}{\sqrt{2}}$)

*Example 9.* Consider the same scenario as in Ex. 8, but additionally assume that, due to an error, the last $T$-gate of $G'$ (i.e., $g'_{15}$) is not applied (yielding a new circuit $\tilde{G}'$). Then, a new functionality results, which is described by the system matrix $\tilde{U}'$ shown in Fig. 6. Since $U$ and $\tilde{U}'$ are obviously not identical anymore, the circuits $G$ and $\tilde{G}'$ have been shown to be non-equivalent. Moreover, since both matrices differ, e.g., in column four, $|4\rangle$ serves as a counterexample showing that

$$U|4\rangle = |u_4\rangle = \frac{1}{\sqrt{2}}[0\;0\;1\;0\;0\;0\;1\;0]^\top;$$

while

$$\tilde{U}'|4\rangle = |\tilde{u}'_4\rangle = \frac{1}{\sqrt{2}}[0\;0\;1\;0\;0\;0\;\tfrac{1-i}{\sqrt{2}}\;0]^\top:$$

It holds that $F(U|4\rangle, \tilde{U}'|4\rangle) = F(|u_4\rangle, |\tilde{u}'_4\rangle) \approx 0.92 < 1$.

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

$$U^{(0)} = U_0; \quad U^{(j)} = U_j \cdot U^{(j-1)} \text{ for } j = 1,\ldots,m-1$$

to construct the whole system matrix $U = U^{(m-1)}$. While conceptually simple, this quickly constitutes an extremely complex task due to the exponential size of the involved matrices (and vectors) with respect to the number of qubits. In fact, equivalence checking of quantum circuits has been shown to be QMA-complete [49].

### B. Related Work

In order to tackle the underlying complexity, equivalence checking of quantum circuits has been intensively considered in the past. Inspired by methods for equivalence checking of classical circuits [27]–[30], methods based on rewriting [24], [31], Boolean satisfiability [32], and decision diagrams [33]–[36] have been proposed. However, approaches based on re-writing suffer from the fact that they are either incomplete, provide inconclusive results in case of non-equivalence, or may require the enumeration of an exponential number of possibilities to check. Solvers for Boolean satisfiability may cope with these problems but, in the case of quantum circuits, face the problem that an infinite number of quantum states needs to be encoded.[6] Because of that, approaches based on decision diagrams (also called DD-based equivalence checking) still constitute the state of the art for equivalence checking of quantum circuits.

Indeed, decision diagrams offer several benefits for this task: As reviewed in Section II-B, they provide means for compact representation and efficient manipulation of the respective functionality realized by a quantum circuit. Moreover, most decision diagrams provide a canonical way of representing quantum functionality (usually with respect to a given variable order and normalization scheme). Because of that, once the functionalities of different quantum circuits are represented as decision diagrams, checking their equivalence can be conducted in constant time by simply comparing their root edges (and the associated weights in case both circuits differ only by a global phase factor).

*Example 10.* Consider again the two quantum circuits $G$ and $G'$ from Ex. 8. Creating decision diagrams for both circuits yields the representation as shown in Fig. 4. Since representations for both, $U$ and $U'$, would point to the (same) root node of this decision diagram, $G$ and $G'$ are proven to be functionally equivalent.

However, despite the benefits discussed above, DD-based equivalence checking of quantum circuits as conducted thus far still has significant shortcomings. In fact, representing the entire functionality of a quantum circuit still might be exponential in the worst case. Even if the representation of the overall functionality of a circuit might be compact, intermediate results may require significantly more space. As an example, the final decision diagram discussed in Ex. 10 and shown in Fig. 4 is composed of seven nodes; however, intermediate decision diagrams generated during the construction required up to nine nodes.[7] Moreover, the generation of a counterexample (which is supposed to be provided in case two quantum circuits are not equivalent) requires further potentially expensive manipulations of the decision diagrams. In fact, in order to generate such a counterexample, the "difference" between the two non-equivalent decision diagrams has to be determined. To this end, one decision diagram has to be inverted (i.e., the conjugate-transpose representation has to be generated) and multiplied with the other decision diagram—requiring non-trivial operations on potentially large representations. Then, any path in the resulting difference decision diagram containing a node not resembling the identity constitutes a counterexample.

*Example 11.* Consider again the quantum circuit $G$ shown in Fig. 1a and the erroneous circuit $\tilde{G}'$ from Ex. 9. Then, the decision diagram representation of $\tilde{U}'$ would not point to the same root node as the decision diagram for $U$—from which it can be concluded that they are not equivalent. In order to generate counterexamples, the difference of both circuits has to be determined—requiring the inversion of the decision diagram for $\tilde{U}'$ (i.e., calculating the conjugate-transposed) and multiplication with the decision diagram for $U$ (see Fig. 4). Eventually, this results in the decision diagram shown in Fig. 7. From this, one may obtain, e.g., the counterexample $|(100)_2\rangle = |4\rangle$ as observed previously in Ex. 9, since the corresponding path contains the node $q_2$—which does not

[6] This problem has partially been addressed in [32] by employing a detailed structural analysis that restricts the number of possible states to a finite one. However, the resulting encoding does not support key features of quantum circuits such as entanglement and, hence, is not applicable for almost all relevant quantum circuits.

[7] An increase by two nodes might not seem substantial. However, for realistic quantum circuits which represent much more complex functionality than this example, the difference in the number of nodes between the final and intermediate decision diagrams easily sums up to several orders of magnitudes.

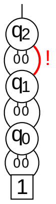


Figure 7: Difference DD of $G$ and $G'$

resemble the identity due to the rightmost edge weight being equal to $\frac{1}{\sqrt{2}}(1+i) \neq 1$.

## IV. General Ideas

In this section, we illustrate the general ideas developed in this work to address the shortcomings of the related work discussed above and, indeed, to solve the equivalence checking problem dramatically faster than ever before—in many cases even just with a single simulation run. To this end, we utilize certain characteristics of quantum computing. While those, at a first glance, make the problem of verification harder compared to the classical realm (circuits have to be supported which do not only rely on 0's and 1's, but also on superposition or entanglement), they also offer potential which has not really been exploited yet. In the following, we sketch this unearthed potential before dedicated equivalence checking schemes and flows resulting from these ideas are covered in the remaining sections of this paper.

### A. Exploiting Reversibility

Many classical logic operations are irreversible (e.g., $x \wedge y = 0$ does not allow one to determine the precise 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) does not make sense. 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., $U^{-1} = U^\dagger = U_0^\dagger \cdots U_{m-1}^\dagger$.

Now 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 $I$, i.e.,

$$G \cdot G'^{-1} = G'^{-1} \cdot G = G^{-1} \cdot G' = G' \cdot G^{-1} \equiv I.^{8}$$

Since, as discussed in Section II-B, the identity constitutes the best case for decision diagrams (the identity can be represented by a decision diagram of linear size), this offers significant potential.

Unfortunately, creating such a concatenation in a naive fashion, e.g., by computing $U \cdot U'^\dagger$ hardly yields any benefits compared to the state of the art. That is, because, even if the final decision diagram would be as compact as possible, the full (and potentially exponential) decision diagrams representing $U \equiv G$ and $U'^\dagger \equiv G'^{-1}$ would still be generated as intermediate results. Even if the whole sequence of $m + m'$ operations is considered as one huge quantum circuit whose functionality is to be constructed and compared to the identity, this would still entail the construction of the full decision diagram of one circuit for the first $m$ ($m'$) operations.

Example 12. Consider again the two circuits $G$ and $G'$ from Fig. 1a and Fig. 5 as discussed before in Ex. 10. Concatenating $G$ and $G'^{-1}$ yields a quantum circuit $\tilde{G} = G \cdot G'^{-1}$ with $4 + 16 = 20$ gates. Since both computations are equivalent (see Ex. 8) constructing the functionality of $\tilde{G}$ would eventually yield an identity decision diagram as shown in Fig. 3. However, during the construction, the first four operations would essentially construct the decision diagram for $G$ as shown in Fig. 4. Thus, there is no real benefit over the state of the art when conducting equivalence checking in this fashion.

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

$$\begin{aligned} G'^{-1} \cdot G &= (g'^{-1}_{m'-1} \cdots g'^{-1}_{0}) \cdot (g_0 \cdots g_{m-1}) \\ &\equiv (U_{m-1} \cdots U_0) \cdot (U'^\dagger_0 \cdots U'^\dagger_{m'-1}) \\ &= U_{m-1} \cdots U_0 \cdot I \cdot U'^\dagger_0 \cdots U'^\dagger_{m'-1} \\ &=: G \rightarrow I \leftarrow G'. \end{aligned}$$

Here, $G \rightarrow I \leftarrow G'$ symbolizes that, starting from the identity $I$, 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 with rather small (intermediate) decision diagrams only. This is illustrated by the following example.

Example 13. Consider again the two circuits $G$ and $G'$ from Ex. 12 and assume that, starting with a decision diagram representing the identity, gates from $G$ and $G'$ are applied as sketched in Fig. 8. Here, the top of the figure indicates the respectively applied gates, and the bottom provides the corresponding sizes of the decision diagrams. As can be seen, applying the gates from $G$ and $G'$ in a particular order "from the left" and "from the right", respectively, frequently yields situations where the impact of a gate from circuit $G$ (potentially increasing the size of the decision diagram) is reverted by multiplications with inverted gates from $G'$ (potentially decreasing the size of the decision diagram back to the representation of the identity function).

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 $G \rightarrow I \leftarrow G'$ inherently provides an efficient representation of the circuit's difference that allows one to obtain counterexamples almost "for free" (while state-of-the-art solutions have to explicitly create those using additional inversion and multiplication operations as discussed in Section III-B).

[8] Throughout this paper we particularly focus on the arrangement $G'^{-1} \cdot G$ as it results in the "intuitive" order of matrix multiplication $U \cdot U'^\dagger$. However, all findings in this paper hold for any possible arrangement of $G$ and $G'$.

Gate application sequence: $I - g_0 - g_0'^{\dagger} - g_1 - g_1'^{\dagger} - g_2'^{\dagger} - g_3'^{\dagger} - g_4'^{\dagger} - g_2 - g_3 - g_5'^{\dagger} - g_6'^{\dagger} - g_7'^{\dagger} - g_8'^{\dagger} - g_9'^{\dagger} - g_{10}'^{\dagger} - g_{11}'^{\dagger} - g_{12}'^{\dagger} - g_{13}'^{\dagger} - g_{14}'^{\dagger} - g_{15}'^{\dagger}$

DD node count: 3—3—3—4—3—3—4—4—6—6—7—7—7—5—5—4—3—3—3—3—3

Figure 8: Illustration of the general idea: $G \rightarrow I \leftarrow G'$ with $G$ and $G'$ from Ex. 10 using decision diagrams

*Example 14.* Consider again the scenario of Ex. 11. In contrast to the state-of-the-art equivalence checking routine, applying the general idea proposed above follows the same steps as illustrated in Fig. 8 up to the very last multiplication—which is dropped. This precisely leads to the decision diagram obtained in Ex. 11 and shown in Fig. 7. Thus, counterexamples can easily be derived from this decision diagram (see Ex. 11).

Overall, exploiting this characteristic and following those ideas, equivalence checking of two quantum circuits can be conducted much more efficiently and compactly than before. But determining when to apply gates from $G$ and when to apply (inverted) gates from $G'$ is not at all obvious. Corresponding strategies for this purpose will be presented and evaluated in Section V and Section VIII, respectively.

### B. 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 [39]–[42], fuzzing [43], [44], etc. are employed in order to verify classical circuits.

This is significantly different in quantum computing. Here, 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. Then, rather than constructing the overall matrices $U$ and $U'$ for both computations, it is sufficient to just compare a couple of individual columns. If any of those columns differ, the two circuits have been shown to be *non-equivalent*. This is illustrated by the following example.

*Example 15.* Consider again the circuit $G$ and the erroneous circuit $G'$ from Ex. 9. As discussed earlier, the respective system matrices $U$ and $U'$ are shown in Fig. 1b and Fig. 6, respectively. Upon comparison of $U$ and $U'$, the non-equivalence of both circuits is clearly seen. Moreover, since $U$ and $U'$ differ in all their columns, this non-equivalence can also be detected by constructing *any* two columns $|u_i\rangle$ and $|u'_i\rangle$, rather than constructing the entire matrices $U$ and $U'$. Specifically, it holds that $F(|u_i\rangle, |u'_i\rangle) \approx 0.92$ for all $i$ from 0 to 7.

While the construction of the matrix $U$ (and accordingly of $U'$) requires expensive matrix-matrix multiplications $U_{m-1} \cdots U_0$, the construction of a single column $|u_i\rangle$ with $i \in \{0, \dots, 2^n - 1\}$ equates to simulating $G$ with the computational basis state $|i\rangle$ as input, i.e., performing the matrix-vector multiplications

$$|u_i^{(0)}\rangle = U_0 |i\rangle, \quad |u_i^{(j)}\rangle = U_j \cdot |u_i^{(j-1)}\rangle \text{ for } j \in \{1, \dots, m-1\}.$$

If the results of those simulations (respectively yielding the $i^{\text{th}}$ columns $|u_i\rangle = |u_i^{(m-1)}\rangle$ and $|u'_i\rangle = |u_i'^{(m'-1)}\rangle$) differ, the two circuits have been shown to be *non-equivalent*.

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. 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 $|i\rangle$. But motivated by these observations, a more detailed consideration of this direction was triggered whose results are summarized in Section VI. In combination with the ideas from Section IV-A, this eventually leads to the proposal of an advanced equivalence checking flow (described in detail in Section VII) which, first, quickly checks for a possible non-equivalence with some simulation runs. If no counterexample has been obtained by this, the (highly probable) equivalence is proven afterwards. As evaluations summarized in Section VIII confirm, this dramatically outperforms the state of the art.

## V. Strategies for conducting $G \rightarrow I \leftarrow G'$

Following the general ideas outlined in Section IV-A potentially allows one to conduct equivalence checking of quantum circuits with decision diagrams in a significantly more efficient fashion than before. However, to fully exploit the idea's potential, a "good" strategy for how to eventually conduct $G \rightarrow I \leftarrow G'$ (i.e., when to apply gates from $G$ and when to apply inverted gates from $G'$) is essential. In this section, we propose several promising strategies and illustrate their application.

### A. Naive Strategy

The first strategy is motivated by the (rather *naive*) assumption that a given circuit $G$ is checked against itself, i.e. $G \rightarrow I \leftarrow G$. Then, obviously the best possible strategy is to alternate between applications of gates from $G$ and their respective inverse—yielding the identity function after each pair of operations. In case that $G \neq G'$ (and, without loss of generality, assuming that $m < m'$, i.e., that $G'$ has more gates), this strategy alternates between $G$ and $G'$ until all

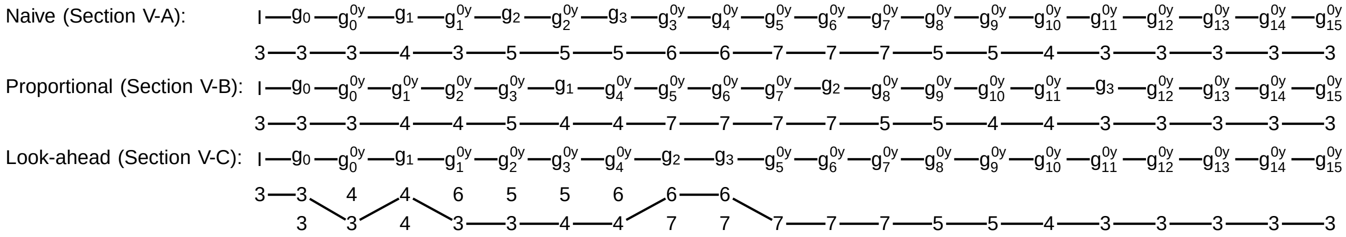

Naive (Section V-A):
I — g0 — g0′† — g1 — g1′† — g2 — g2′† — g3 — g3′† — g4′† — g5′† — g6′† — g7′† — g8′† — g9′† — g10′† — g11′† — g12′† — g13′† — g14′† — g15′†
3 — 3 — 3 — 4 — 3 — 5 — 5 — 5 — 6 — 6 — 7 — 7 — 7 — 5 — 5 — 4 — 3 — 3 — 3 — 3 — 3
Proportional (Section V-B):
I — g0 — g0′† — g1′† — g2′† — g3′† — g1 — g4′† — g5′† — g6′† — g7′† — g2 — g8′† — g9′† — g10′† — g11′† — g3 — g12′† — g13′† — g14′† — g15′†
3 — 3 — 3 — 4 — 4 — 5 — 4 — 4 — 7 — 7 — 7 — 7 — 5 — 5 — 4 — 4 — 3 — 3 — 3 — 3 — 3
Look-ahead (Section V-C):
I — g0 — g0′† — g1 — g1′† — g2′† — g3′† — g4′† — g2 — g3 — g5′† — g6′† — g7′† — g8′† — g9′† — g10′† — g11′† — g12′† — g13′† — g14′† — g15′†
3 — 3 4 4 6 5 5 6 6 — 6
3 3 4 3 — 3 — 4 — 4 7 7 7 — 7 — 7 — 5 — 5 — 4 — 3 — 3 — 3 — 3 — 3

Figure 9: Application of the proposed strategies and corresponding node counts (for the circuits $G$ and $G'$ from Ex. 10)

gates of $G$ have been applied. Afterwards, the "left-over" gates from $G'$ are applied. This strategy supposedly works well if $G$ and $G'$ are very similar, but obviously looses its benefits if both circuits significantly differ in their structure (in particular if one circuit has significantly more gates than the other, i.e., if $m \ll m'$).

Example 16. Consider again the two circuits $G$ and $G'$ as discussed before in Ex. 10. Applying the naive strategy leads to an order of gate applications as shown at the top of Fig. 9 with the corresponding node count listed below the respective identifier. During this process, the size of the (intermediate) decision diagrams never exceeds $7$ nodes, while the average node count is $4.43$. This is significantly less than required by the state-of-the-art approach which has a maximal node count of 9 and an average one of $5.95$.

## B. Proportional Strategy

Applying the naive strategy to circuits which, structurally, are significantly different obviously leads to an imbalance since a huge portion of "left-over" gates are applied—possibly neglecting the effect of staying close to the identity function. In order to avoid that, the *proportional* strategy aims for a more balanced approach. To this end, first, the ratio with respect to the number of gates for both circuits is determined. Afterwards, the gates from $G$ and $G'$ are proportionally applied according to this ratio.

Example 17. Consider again the two circuits $G$ and $G'$ as discussed before in Ex. 10. The ratio between their gate counts is $4 : 16 = 1 : 4$. Hence, applying the proportional strategy leads to an order of gate applications as shown in the second row of Fig. 9 with the corresponding node count listed below the respective identifier. During this process, the size of the (intermediate) decision diagrams never exceeds $7$ nodes, while the average node count is $4.33$. For the problem at hand, this strategy constitutes near optimal performance.

## C. Look-ahead Strategy

Despite strategies considering structural elements of the given circuits for deciding the actions to be performed, also schemes based on the actual size of the (intermediate) decision diagrams may provide a good indication of how to proceed. Recall that the general aim is to stay as close as possible to the identity function (leading to the smallest possible decision diagram). Hence, the decision to apply a gate either from $G$ or $G'$ can be based on which case actually leads to a smaller decision diagram. This is conducted by the *look-ahead* scheme. While this potentially doubles the number of multiplications to be performed (since both alternatives have to be checked out), it may lead to smaller decision diagrams and, by this, a more efficient equivalence checking routine.

Example 18. Consider again the two circuits $G$ and $G'$ as discussed before in Ex. 10. Applying the look-ahead strategy leads to an order of gate applications as shown in the third row of Fig. 9. As long as there are still gates left to be multiplied in both circuits, the bottom of Fig. 9 indicates the node count of both alternatives ($G$ on top, $G'$ on the bottom). The bold line indicates which path was chosen. The resulting sequence of operations is exactly the one shown in Fig. 8 which was employed in Ex. 13—resulting in a maximum of $7$ nodes, while the average node count is $4.24$.

Even for the small example showcased throughout this section, all proposed strategies perform significantly better in terms of maximum as well as average decision diagram size when compared to the state-of-the-art approach.

## VI. Investigating the Power of Simulation

In Section IV-B, we illustrated the potential power of simulation for equivalence checking and argued that, in case two quantum circuits $G$ and $G'$ operating on $n$ qubits are not equivalent, even a few simulation runs will likely yield a counterexample. This idea triggered a more detailed consideration in which we elaborated how significantly the matrices $U$ and $U'$ differ from each other in case of non-equivalence and whether this would make an incomplete coverage of the functionality feasible. The results obtained by this consideration are summarized in this section.

To this end, we first introduce the notion of the *difference* of two unitary matrices. Given two unitary matrices $U$ and $U'$, their difference $D$ is defined as the unitary matrix $D = U^\dagger U'$ and it holds that $U \cdot D = U'$.[9] In case both matrices are identical (i.e., the circuits are equivalent), it directly follows that $D = I$. One characteristic of the identity function $I$ resulting in this case is that all diagonal entries are equal to one, i.e., $\langle i|U^\dagger U'|i\rangle = 1$ for $i \in \{0, \ldots, 2^n - 1\}$, where $|i\rangle$ denotes the $i^{th}$ 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

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

[9] Similarly to picking $G'^{-1} \cdot G$ in Section IV-A, the particular arrangement $D = U^\dagger U'$ is just one of the four possibilities to introduce the notion of the *difference* of two unitaries.

where $|u_i\rangle$ and $|u'_i\rangle$ denote the $i^{th}$ 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 $F$ between the resulting states $|u_i\rangle$ and $|u'_i\rangle$ (see Section III-A). Hence, if only one simulation yields $F_i := F(|u_i\rangle, |u'_i\rangle) \neq 1$, then $|i\rangle$ proves the non-equivalence of $G$ and $G'$.

Now, the question is how many computational basis states $|i\rangle$ yield $F_i \neq 1$ for a given difference matrix $D$, i.e., how likely it is for an arbitrary simulation to detect possible differences. Since the difference $D$ of both matrices is unitary itself, it may as well be interpreted as a quantum circuit $G_D$. For the purpose of this theoretical consideration, we assume that each gate of $G_D$ either represents a single-qubit or a (multi-)controlled operation.[10]

*Example 19.* 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

$$D = I_{2^{n-1}} \otimes U_s = \begin{bmatrix} U_s & & \\ & \ddots & \\ & & U_s \end{bmatrix}.$$

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 posses a system matrix least similar to the identity matrix due to the tensor product structure of their corresponding system matrix.

*Example 20.* In contrast to Ex. 19, 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

$$D = \begin{bmatrix} I_2 & & & \\ & \ddots & & \\ & & I_2 & \\ & & & U_s \end{bmatrix}.$$

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.

*Example 21.* Consider a two-qubit system and two circuits $G$ and $G'$ exhibiting a difference-circuit $G_D$ consisting of two gates—the first of which is a Hadamard operation $H(q_1)$ on the second qubit, while the second is a CNOT operation $\mathrm{CNOT}(q_1, q_0)$ with target on the first and control on the second qubit. As discussed in Ex. 19 and Ex. 20, the application of the single-qubit matrix $U_H$ affects all columns, while the application of the controlled two-qubit operation matrix $U_{CNOT}$ only affects two of the four columns. Their combination, however, still affects all columns of the result.

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.

Of course, this cannot be guaranteed—otherwise, the problem would hardly be QMA-complete. However, following the above discussion and considering that comparing single columns (i.e., a partial consideration of the functionality) is substantially cheaper than a full functional coverage, the simulation of arbitrary computational basis states is a promising option. Moreover, as also confirmed by the experimental evaluation in Section VIII-D, 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.

## VII. Resulting Advanced Equivalence Checking Flow

The general ideas proposed in Section IV and elaborated in Section V as well as Section VI complement each other in many different ways. Trying to keep $G \rightarrow I \leftarrow G'$ close to the identity (see Section IV-A) proves very efficient in case two circuits are indeed equivalent—provided a "good" strategy can be employed. Conducting simulations with randomly chosen computational basis states $|i\rangle$ (see Section IV-B) on the other hand allows to quickly detect non-equivalence even in cases where both circuits only differ slightly. This section first describes how these findings eventually result in an advanced equivalence checking flow—followed by a discussion of the resulting methodology.

### A. Resulting Flow

The motivation and considerations above eventually led us to an advanced equivalence checking flow as shown in Fig. 10. Here, instead of constructing and comparing the complete matrix representations of both circuits, we propose to first perform a limited number of $r \ll 2^n$ simulation runs with randomly chosen computational basis states. Should one of those simulations yield different outputs in both circuits (i.e., a fidelity $F_i \neq 1$), the non-equivalence of the circuits under consideration has been shown. If this is not the case, the equivalence checking routine $G \rightarrow I \leftarrow G'$ is utilized to complete the task. Moreover, if simulation has not revealed any differences, the likelihood of the circuits being non-equivalent is significantly reduced as shown in the discussions from Section VI and confirmed by the evaluations in Section VIII. Overall, this eventually leads to three possible outcomes:

[10] This does not limit the applicability of the following findings, since arbitrary single-qubit operations combined with CNOT form a universal gate-set [1].

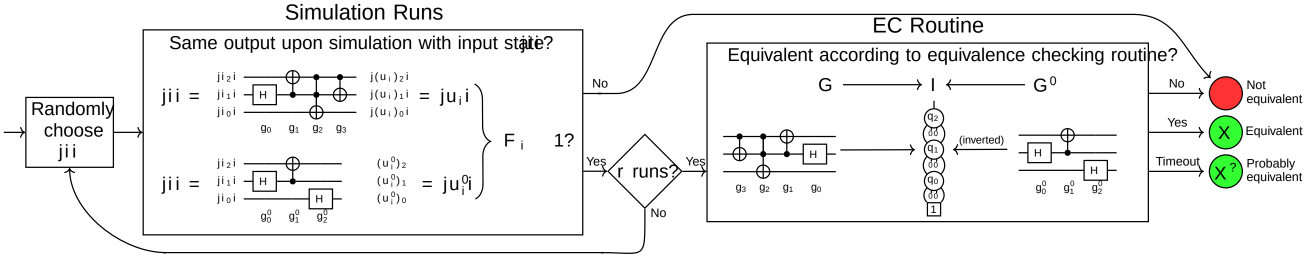


Figure 10: Proposed equivalence checking flow

- Not equivalent, if any simulation run yields $F_i$ 6 1 or if the equivalence checking routine is employed after the simulation runs and yields this result.[11] As confirmed by the evaluations summarized later in Section VIII, this can be conducted very efficiently in many cases, while state-of-the-art equivalence checking routines require substantial runtime or even time out frequently.
- Equivalent, if, after $r$ simulation runs, the equivalence checking routine is employed and yields that result. If this is the case, the simulation runs conducted before (which did not lead to a conclusive result) only constitute a negligible runtime overhead. As discussed earlier, this is because constructing the whole functionality is complexity-wise equivalent to simulating all $2^n$ computational basis states and we have chosen $r \; 2^n$.
- Timeout, if the simulation runs did not lead to a counterexample and the equivalence checking routine was not able to complete the task within a given resource limit. If this is the case, we at least get an indication that both circuits might be equivalent (since the conducted simulations did not provide a counterexample which, according to the discussions from Section VI, is rather rare). Even if this does not provide a guarantee of non-equivalence, this is a stronger result than provided by the state of the art thus far, which does not allow for any kind of conclusion in this case.

### B. Discussion

The resulting equivalence checking flow with the combination of using the power of simulation together with the dedicated $G \quad I \quad G^0$ scheme dramatically improves the state of the art. In contrast to constructing and comparing the whole functionality of two circuits, simulation can typically be conducted significantly faster while already providing valuable insights, i.e., a counterexample or an indication that both circuits might be equivalent. If the results of simulation remain inconclusive, i.e., no counterexample was found, the subsequent functional comparison can be improved by utilizing the $G \quad I \quad G^0$ scheme. Even if the latter step (the actual proof) cannot be completed due to time or resource limits, a meaningful result has already been obtained after conducting the simulations—namely that the circuits are presumably equivalent—while previously proposed approaches provide no indication on the equivalence at all.

Naturally, there are cases where even single simulation runs prove too resource-consuming to conduct. Especially in cases where simulation fails due to the immense complexity of the involved state vector, the $G \quad I \quad G^0$ scheme still provides a complementary alternative. If a strategy can be employed which manages to keep the intermediate decision diagrams close to the identity, the whole procedure can again be conducted with a rather low memory footprint. The design of corresponding strategies could benefit from specific knowledge about the origin of both circuits, e.g., if $G^0$ is the result of compiling $G$ to a certain target device. It is left for future work to explore such application-specific strategies based on the $G \quad I \quad G^0$ scheme.

Besides that, our discussions in Section VI show that choosing computational basis states uniformly at random as input for the simulation part of the proposed equivalence checking flow promises high success rates in the quantum realm. This is in stark contrast to the (simulative) verification of classical circuits, where sophisticated techniques such as constrained-based stimuli generation [39]–[42], fuzzing [43], [44], etc. have to be employed for such techniques to work at all. Our experimental results (summarized in the next section) empirically show that such sophisticated techniques might not be that urgently needed in the quantum realm. First results towards the study of quantum stimuli generation schemes have been obtained in [50].

Furthermore, previous works on equivalence checking of quantum circuits often employed several optimizations such as template replacement [24], [31]. Potential candidates include optimizations frequently used during compilation [18], [21]–[24] (as reviewed in Section I). Those optimization techniques may of course also be applied in the methodology proposed here—as long as the optimization itself is valid (which could be verified beforehand using the proposed methodology). In general, lowering the gate count of the respective circuits (or their concatenation in case of the $G \quad I \quad G^0$ scheme) can be expected to improve the performance of the proposed equivalence checking flow (since less operations have to be applied).

Finally, there is no need to conduct both complementary steps (simulation and the $G \quad I \quad G^0$ scheme) in a sequential fashion as proposed in Fig. 10. In fact, the simulations and the $G \quad I \quad G^0$ scheme could also be started in parallel (provided sufficient memory and processor resources). If then, any of the simulations yields a counterexample, the remaining calculations can be aborted. Vice versa, if the equivalence checking routine returns “equivalent”, any ongoing simulation runs can be stopped.

## VIII. Experimental Evaluation

In order to evaluate the ideas and methods proposed above, the advanced methodology for equivalence checking of quantum circuits (as summarized in Fig. 10) has been implemented

[11] As a consequence of the discussions from Section VI it is far more likely that this result already occurs during simulation.

and integrated into the JKQ toolset [51]. For simulation, the method proposed in [46] based on the decision diagram package proposed in [47] has been used and we fixed $r = 16$, i.e., we conduct up to sixteen random simulations before opting for the equivalence checking routine if no counterexample has been obtained. The dedicated strategies for conducting $G \quad I \quad G^0$ (proposed in Section V) have been implemented on top of the above decision diagram package as well. In this section, we summarize the obtained results. To this end, we first review the further setup, before the results themselves are covered.

## A. Setup

In order to evaluate the performance of the proposed equivalence checking flow, we used benchmarks frequently considered for evaluating quantum circuit compilers, simulators, and equivalence checking tools—taken from [52]. Each benchmark consists of a "high-level" description $G$ (directly obtained from [52]) as well as a "low-level" description $G^0$ generated by compiling the given descriptions into elementary gates and/or mapping them for a certain quantum computer architecture using IBM Qiskit [53].

The performance is compared against the state-of-the-art equivalence checking technique taken from [34] (see Ex. 8). All computations have been performed on a 4 GHz Amazon EC2 z1d instance running Ubuntu 18.04 with at least 32 GB per job using GNU Parallel [54]. A hard timeout of 1 h (i.e., 3 600 s) was set for each run.

In the following, we provide a representative subset of the obtained results. Additionally, the implementation of the proposed methodology (together with an archive of all benchmarks) is publicly available at https://iic.jku.at/eda/research/quantum_verification for further evaluation.

## B. Showing Equivalence

In a first series of evaluations, we considered cases where the two given circuits $G$ and $G^0$ are equivalent. Table I provides the obtained results. The first columns describe the benchmarks themselves (i.e., their name, number of qubits, $n$, as well as the gate count $|G|$ and $|G^0|$), followed by the runtime $t_{sota}$ (in CPU seconds) for the state-of-the-art approach. Then, the performance of the proposed flow is listed—split into the runtime $t_{sim}$ for the simulation runs as well as the runtime of the individual $G \quad I \quad G^0$ strategies. Finally, the last columns list the total runtime $t_{proposed}$ of the proposed scheme and its relative improvement compared to the state-of-the-art routine[12].

From the results it can be seen that, in the majority of cases, the dedicated strategies proposed in this work reduce the equivalence checking time down to a half or a third compared to the state-of-the-art routine. Especially the *proportional* strategy performs rather well in this regard—sometimes leading to magnitudes of improvement. Furthermore, the additionally conducted simulations only lead to a negligible run-time overhead to the overall flow. Hence, although they cannot prove equivalence, they also do not "hurt" much. On the contrary, as discussed in Section VII and further confirmed by the following evaluations, they often provide indications of the circuits' equivalence—in particular for cases where no decisive answer could be obtained by any equivalence checking routine due to timeouts (e.g., for the topmost four benchmarks in Table I), this is better than no result at all.

## C. Showing Non-equivalence

In a second series of evaluations, we considered cases where the two given circuits $G$ and $G^0$ are *not* equivalent. To this end, we injected errors into each benchmark by removing either one or a total of three random gates from the corresponding circuit $G^0$. This led to benchmarks, where the difference between $G$ and $G^0$ is difficult to detect (usually, differences caused by errors in the compilation flow would have much more severe effects). The respectively obtained results are provided in Table II.

Using the state-of-the-art approach, detecting these differences indeed causes substantial runtimes—or cannot be completed within the given time limit at all. In contrast, the proposed methodology can determine the non-equivalence for all benchmarks in just a matter of seconds or even less. Moreover, in the vast majority of cases, simulation alone is capable of showing the non-equivalence—in more or less negligible run-time. Only in the case of mlp4_245 (and only assuming the removal of a single gate, i.e., a most subtle difference), an actual equivalence checking routine is needed. Here, the $G \quad I \quad G^0$ strategies can finish the job in feasible runtime.

On top of that, the results also show that it is indeed rather unlikely that differences get masked so that they cannot be detected by simulation. While this indeed happens in case just one gate is removed (as reported in Table IIa for benchmark mlp4_245), not a single such instance exists in our evaluation if three gates are removed (which still constitute rather subtle differences compared to the effect an actual error in the compilation flow would have on $G^0$). In fact, Table IIb shows that, in the majority of cases, a single simulation run (and never more than 3 runs) are necessary to detect the error—clearly demonstrating the power of simulation.

## D. The Power of Simulation

In order to further evaluate the power of simulation, we finally conducted a third series of evaluations in which the success probability of the simulation part was evaluated in detail. To this end, not just a single erroneous instance has been considered for each benchmark, but a hundred different instances with either one or three gates removed have been created. Afterwards, we again executed $r = 16$ simulation runs to check whether this shows the respective non-equivalence. Table III provides the obtained results, i.e., the average number of simulations, the average and the maximum simulation runtime, as well as the success probability, i.e., the probability that non-equivalence was detected within sixteen simulations.

In cases where just a single gate was removed from the circuit (which, thus far, was really hard, if not impossible, to detect), the success probability is 85.2 % on average. Already this provides a rather strong argument that, when checking the

[12] The total runtime $t_{proposed}$ is calculated as the sum of the simulations' runtime and the runtime of the proportional scheme which, as discussed next, turns out to be the most efficient scheme, i.e., $t_{proposed} = t_{sim} + t_{prop}$.

Table I: Equivalent benchmarks

| Benchmark | | | | State-of-the-art EC Routine | Proposed Equivalence Checking Flow | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Simulation Runs | | EC Routine | | | | |
| Name | $n$ | $\|G\|$ | $\|G^0\|$ | $t_{sota}$ [s] | #sims | $t_{sim}$ [s] | $t_{naive}$ [s] | $t_{prop}$ [s] | $t_{look}$ [s] | $t_{proposed}$ [s] | $\frac{t_{proposed}}{t_{sota}}$ |
| apla_203 | 22 | 80 | 13 548 | >3 600.00 | **16** | **2.68** | >3 600.00 | >3 600.00 | >3 600.00 | >3 600.00 | **-** |
| cm151a_211 | 28 | 33 | 3 889 | >3 600.00 | **16** | **1.01** | >3 600.00 | >3 600.00 | >3 600.00 | >3 600.00 | **-** |
| rd84_313 | 34 | 113 | 2 295 | >3 600.00 | **16** | **0.65** | >3 600.00 | >3 600.00 | >3 600.00 | >3 600.00 | **-** |
| mod5adder_306 | 32 | 110 | 2 348 | >3 600.00 | **16** | **0.66** | >3 600.00 | >3 600.00 | >3 600.00 | >3 600.00 | **-** |
| sym9_317 | 27 | 64 | 1 540 | >3 600.00 | 16 | 0.39 | >3 600.00 | **63.61** | **1 484.83** | **64.00** | **-** |
| c2_182 | 35 | 305 | 2 545 | >3 600.00 | 16 | 0.73 | **2 664.79** | **2 500.79** | **3 081.50** | **2 501.52** | **-** |
| cm163a_213 | 29 | 39 | 3 539 | >3 600.00 | 16 | 0.94 | **496.95** | **183.93** | **503.43** | **184.87** | **-** |
| cm150a_210 | 22 | 53 | 31 595 | 3 452.70 | 16 | 3.43 | >3 600.00 | **0.59** | **10.24** | **4.03** | 0.001 |
| rd73_312 | 25 | 76 | 1 426 | 1 108.62 | 16 | 0.36 | **598.57** | **196.10** | **761.75** | **196.47** | 0.177 |
| c2_181 | 35 | 116 | 2 708 | 403.80 | 16 | 0.77 | **60.53** | **4.06** | **62.58** | **4.83** | 0.012 |
| example2_231 | 16 | 157 | 19 411 | 187.63 | 16 | 3.31 | **133.82** | **79.62** | **132.26** | **82.93** | 0.442 |
| cu_219 | 25 | 40 | 5 031 | 177.15 | 16 | 1.12 | **124.00** | **72.20** | **121.63** | **73.32** | 0.414 |
| C7552_205 | 21 | 80 | 6 384 | 158.30 | 16 | 1.26 | **93.96** | **80.38** | **94.50** | **81.64** | 0.516 |
| add6_196 | 19 | 229 | 262 179 | 150.56 | 16 | 25.92 | **79.49** | **13.85** | **78.62** | **39.78** | 0.264 |
| dk17_224 | 21 | 49 | 6 650 | 105.19 | 16 | 1.22 | **39.77** | **36.56** | **39.78** | **37.78** | 0.359 |
| mlp4_245 | 16 | 131 | 14 513 | 79.06 | 16 | 2.41 | **33.13** | **22.69** | **34.95** | **25.10** | 0.317 |
| alu1_198 | 20 | 32 | 1 377 | 13.19 | 16 | 0.36 | **8.44** | **4.02** | **7.23** | **4.39** | 0.333 |
| 5xp1_194 | 17 | 85 | 5 487 | 10.78 | 16 | 1.10 | **5.07** | **4.59** | **5.16** | **5.70** | 0.529 |
| pcler8_248 | 21 | 22 | 1 522 | 3.44 | 16 | 0.38 | **2.06** | **1.66** | **1.99** | **2.04** | 0.593 |
| dk27_225 | 18 | 24 | 1 208 | 2.83 | 16 | 0.31 | **1.82** | **1.48** | **1.48** | **1.78** | 0.629 |

$n$: Number of qubits $jGj$: Gate count of $G$ $jG^0j$: Gate count of $G^0$ $t_{sota}$: Runtime of state-of-the-art EC routine #sims: Number of simulation runs
$t_{sim}$: Runtime of simulation runs $t_{naive}$=$t_{prop}$=$t_{look}$: Runtime of EC routine $t_{proposed}$: Runtime of proposed EC flow

equivalence of quantum circuits, simulation frequently is sufficient (in contrast to the classical realm, where masking effects frequently make simulation insufficient). Moreover, removing just two more gates (still constituting a rather subtle difference compared to the effect an actual error in the compilation flow may have on $G^0$) results in an average success probability which is very close to 100 %. Additionally considering that the runtime for that is almost always negligible, this clearly shows the power of simulation to either quickly show the non-equivalence of two circuits or, at least, provide an indication of equivalence.

## IX. Conclusions

In this paper, we proposed an advanced methodology for equivalence checking of quantum circuits which takes the different paradigms of quantum computing not as burden, but as an opportunity. If a "good" strategy for keeping $G \quad I \quad G^0$ close to the identity is available, obtaining the complete proof of two circuit's equivalence can be dramatically accelerated. Furthermore, we showed that, in contrast to classical computing, inherent characteristics of quantum circuits allow for the conclusion that simulations provide an indication on whether two circuits are equivalent—even in the case of rather minor differences. Experimental evaluations confirmed that the resulting methodology allows one to check the equivalence or non-equivalence magnitudes faster than ever before—and, in many cases, by simulation only.

Specifically, strategies for keeping $G \quad I \quad G^0$ close to the identity may be derived from utilizing information about how a given circuit $G$ is compiled to a resulting implementation $G^0$. First studies on verifying the results of IBM's quantum circuit compilation flow Qiskit [53] employing the proposed methodology have been conducted in [55]. In contrast to the classical realm, this could eventually make verifying the results of sophisticated design flows feasible in general. An implementation of the proposed equivalence checking flow is publicly available at https://iic.jku.at/eda/research/quantum_verification/.

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

[2] B. Giles and P. Selinger, "Exact synthesis of multiqubit Clifford+T circuits," *Phys. Rev. A*, vol. 87, no. 3, p. 032 332, 2013.

[3] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013.

[4] 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.

[5] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," in *Int'l Symp. on Multi-Valued Logic*, 2011.

[6] 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.

[7] 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.

[8] 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.

[9] A. Zulehner and R. Wille, "Compiling SU(4) quantum circuits to IBM QX architectures," in *Asia and South Pacific Design Automation Conf.*, Tokyo, Japan, 2019, pp. 185–190.

[10] K. N. Smith and M. A. Thornton, "A quantum computational compiler and design tool for technology-specific targets," in *Int'l Symp. on Computer Architecture*, 2019, pp. 579–588.

[11] 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.

[12] A. Matsuo and S. Yamashita, "An efficient method for quantum circuit placement problem on a 2-D grid," in *Int'l Conf. of Reversible Computation*, 2019, pp. 162–168.

[13] M. Amy and V. Gheorghiu, "Staq—A full-stack quantum processing toolkit," *Quantum Sci. Technol.*, vol. 5, no. 3, p. 034 016, 2020.

[14] A. Cowtan *et al.*, "On the qubit routing problem," in *Theory of quantum computation, communication and cryptography*, 2019.

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

Table II: Non-equivalent benchmarks

(a) Removed 1 random gate

| Benchmark | | | | State-of-the-art EC Routine | Proposed Equivalence Checking Flow | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Simulation Runs | | EC Routine | | | |
| Name | $n$ | $\|G\|$ | $\|G^0\|$ | $t_{sota}$ [s] | #sims | $t_{sim}$ [s] | $t_{naive}$ [s] | $t_{prop}$ [s] | $t_{look}$ [s] | $t_{proposed}$ [s] |
| apla_203 | 22 | 80 | 13 547 | >3 600.00 | **1** | **0.15** | - | - | - | **0.15** |
| cm151a_211 | 28 | 33 | 3 888 | >3 600.00 | **2** | **0.12** | - | - | - | **0.12** |
| rd84_313 | 34 | 113 | 2 294 | >3 600.00 | **4** | **0.16** | - | - | - | **0.16** |
| mod5adder_306 | 32 | 110 | 2 347 | >3 600.00 | **4** | **0.17** | - | - | - | **0.17** |
| c2_182 | 35 | 305 | 2 544 | >3 600.00 | **1** | **0.05** | - | - | - | **0.05** |
| cm150a_210 | 22 | 53 | 31 594 | >3 600.00 | **1** | **0.62** | - | - | - | **0.62** |
| rd73_312 | 25 | 76 | 1 425 | >3 600.00 | **1** | **0.02** | - | - | - | **0.02** |
| sym9_317 | 27 | 64 | 1 539 | >3 600.00 | **1** | **0.02** | - | - | - | **0.02** |
| cm163a_213 | 29 | 39 | 3 538 | >3 600.00 | **3** | **0.17** | - | - | - | **0.17** |
| cu_219 | 25 | 40 | 5 030 | >3 600.00 | **1** | **0.09** | - | - | - | **0.09** |
| c2_181 | 35 | 116 | 2 707 | 899.19 | **1** | **0.05** | - | - | - | **0.05** |
| example2_231 | 16 | 157 | 19 410 | 194.51 | **1** | **0.18** | - | - | - | **0.18** |
| C7552_205 | 21 | 80 | 6 383 | 160.90 | **3** | **0.23** | - | - | - | **0.23** |
| add6_196 | 19 | 229 | 262 178 | 143.05 | **1** | **0.33** | - | - | - | **0.33** |
| dk17_224 | 21 | 49 | 6 649 | 106.03 | **11** | **0.84** | - | - | - | **0.84** |
| mlp4_245 | 16 | 131 | 14 512 | 80.97 | 16 | 2.36 | **33.81** | **23.35** | **33.02** | **25.71** |
| alu1_198 | 20 | 32 | 1 376 | 42.56 | **1** | **0.03** | - | - | - | **0.03** |
| 5xp1_194 | 17 | 85 | 5 486 | 11.43 | **1** | **0.07** | - | - | - | **0.07** |
| pcler8_248 | 21 | 22 | 1 521 | 4.41 | **8** | **0.19** | - | - | - | **0.19** |
| dk27_225 | 18 | 24 | 1 207 | 3.45 | **1** | **0.02** | - | - | - | **0.02** |

$n$: Number of qubits $jGj$: Gate count of $G$ $jG^0j$: Gate count of $G^0$ $t_{sota}$: Runtime of state-of-the-art EC routine #sims: Number of simulation runs
$t_{sim}$: Runtime of simulation runs $t_{naive}$=$t_{prop}$=$t_{look}$: Runtime of EC routine $t_{proposed}$: Runtime of proposed EC flow

(b) Removed 3 random gates

| Benchmark | | | | State-of-the-art EC Routine | Proposed Equivalence Checking Flow | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Simulation Runs | | EC Routine | | | |
| Name | $n$ | $\|G\|$ | $\|G^0\|$ | $t_{sota}$ [s] | #sims | $t_{sim}$ [s] | $t_{naive}$ [s] | $t_{prop}$ [s] | $t_{look}$ [s] | $t_{proposed}$ [s] |
| apla_203 | 22 | 80 | 13 545 | >3 600.00 | **3** | **0.50** | - | - | - | **0.50** |
| cm151a_211 | 28 | 33 | 3 886 | >3 600.00 | **1** | **0.08** | - | - | - | **0.08** |
| rd84_313 | 34 | 113 | 2 292 | >3 600.00 | **1** | **0.04** | - | - | - | **0.04** |
| mod5adder_306 | 32 | 110 | 2 345 | >3 600.00 | **1** | **0.04** | - | - | - | **0.04** |
| c2_182 | 35 | 305 | 2 542 | >3 600.00 | **1** | **0.04** | - | - | - | **0.04** |
| sym9_317 | 27 | 64 | 1 537 | >3 600.00 | **1** | **0.03** | - | - | - | **0.03** |
| cm163a_213 | 29 | 39 | 3 536 | >3 600.00 | **3** | **0.18** | - | - | - | **0.18** |
| c2_181 | 35 | 116 | 2 705 | >3 600.00 | **1** | **0.05** | - | - | - | **0.05** |
| C7552_205 | 21 | 80 | 6 381 | >3 600.00 | **1** | **0.12** | - | - | - | **0.12** |
| add6_196 | 19 | 229 | 262 176 | >3 600.00 | **1** | **1.66** | - | - | - | **1.66** |
| cu_219 | 25 | 40 | 5 028 | >3 600.00 | **2** | **0.16** | - | - | - | **0.16** |
| rd73_312 | 25 | 76 | 1 423 | >3 600.00 | **1** | **0.02** | - | - | - | **0.02** |
| cm150a_210 | 22 | 53 | 31 592 | 2 982.74 | **1** | **0.19** | - | - | - | **0.19** |
| example2_231 | 16 | 157 | 19 408 | 1 215.63 | **1** | **0.26** | - | - | - | **0.26** |
| dk17_224 | 21 | 49 | 6 647 | 327.62 | **1** | **0.07** | - | - | - | **0.07** |
| mlp4_245 | 16 | 131 | 14 510 | 186.00 | **1** | **0.14** | - | - | - | **0.14** |
| alu1_198 | 20 | 32 | 1 374 | 84.76 | **1** | **0.03** | - | - | - | **0.03** |
| 5xp1_194 | 17 | 85 | 5 484 | 21.52 | **1** | **0.08** | - | - | - | **0.08** |
| dk27_225 | 18 | 24 | 1 205 | 5.22 | **1** | **0.02** | - | - | - | **0.02** |
| pcler8_248 | 21 | 22 | 1 519 | 4.03 | **1** | **0.02** | - | - | - | **0.02** |

$n$: Number of qubits $jGj$: Gate count of $G$ $jG^0j$: Gate count of $G^0$ $t_{sota}$: Runtime of state-of-the-art EC routine #sims: Number of simulation runs
$t_{sim}$: Runtime of simulation runs $t_{naive}$=$t_{prop}$=$t_{look}$: Runtime of EC routine $t_{proposed}$: Runtime of proposed EC flow

[16] W. Hattori and S. Yamashita, "Quantum circuit optimization by changing the gate order for 2D nearest neighbor architectures," in *Int'l Conf. of Reversible Computation*, 2018, pp. 228–243.

[17] Z. Sasanian and D. M. Miller, "Reversible and quantum circuit optimization: A functional approach," in *Int'l Conf. of Reversible Computation*, 2013, pp. 112–124.

[18] 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.

[19] T. Itoko, R. Raymond, T. Imamichi, A. Matsuo, and A. W. Cross, "Quantum circuit compilers using gate commutation rules," *Asia and South Pacific Design Automation Conf.*, pp. 191–196, 2019.

[20] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, "Automated optimization of large quantum circuits with continuous parameters," *npj Quantum Inf*, vol. 4, no. 1, p. 23, 2018.

[21] D. Maslov, G. Dueck, D. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 436–444, 2008.

[22] A. K. Prasad, V. V. Shende, I. L. Markov, J. P. Hayes, and K. N. Patel, "Data structures and algorithms for simplifying reversible circuits," *J Emerg Technol Comput Syst*, vol. 2, no. 4, pp. 277–293, 2006.

[23] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-Based quantum circuits," in *Design Automation Conf.*, 2002, pp. 419–424.

[24] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering. (2019). "Graph-theoretic simplification of quantum circuits with the ZX-calculus." arXiv: 1902.03178.

[25] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh, "MUQUT: Multi-constraint quantum circuit mapping on NISQ computers: Invited paper," in *Int'l Conf. on CAD*, 2019.

[26] 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.

[27] D. Brand, "Verification of large synthesized designs," in *Int'l Conf. on CAD*, 1993, pp. 534–537.

[28] P. Molitor and J. Mohnke, *Equivalence checking of digital circuits: Fundamentals, principles, methods*. Springer, 2010.

[29] J. Marques-Silva and T. Glass, "Combinational equivalence checking using satisfiability and recursive learning," in *Design, Automation and Test in Europe*, 1999.