The Power of Simulation for Equivalence Checking in Quantum Computing

Lukas Burgholzer Robert Wille
Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
lukas.burgholzer@jku.at robert.wille@jku.at
http://iic.jku.at/eda/research/quantum/

***Abstract*—The rapid rate of progress in the physical realization of quantum computers sparked the development of elaborate design flows for quantum computations on such devices. Each stage of these flows comes with its own representation of the intended functionality. Ensuring that each design step preserves this intended functionality is of utmost importance. However, existing solutions for equivalence checking of quantum computations heavily struggle with the complexity of the underlying problem and, thus, no conclusions on the equivalence may be reached with reasonable efforts in many cases. In this work, we uncover the power of simulation for equivalence checking in quantum computing. We show that, in contrast to classical computing, it is in general not necessary to compare the *complete* representation of the respective computations. Even small errors frequently affect the *entire* representation and, thus, can be detected within a couple of simulations. The resulting equivalence checking flow substantially improves upon the state of the art by drastically accelerating the detection of errors or providing a highly probable estimate of the operations' equivalence.**

***Index Terms*—equivalence checking, quantum computing, simulation, verification**

I. Introduction

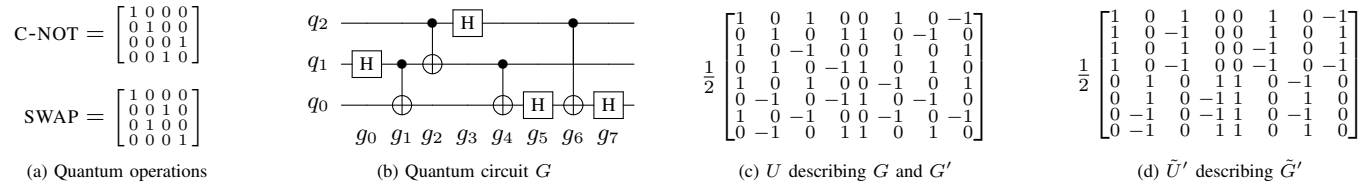
Quantum computers are at the brink of transcending from academic research to becoming commercially available as shown, e.g., by IBM unveiling their IBM Q System One in 2019 [1]. However, given a conceptual quantum algorithm, it is not a straightforward task to execute such a quantum computation on an actual quantum computer. The restriction to specific gate libraries and certain architectural constraints necessitate methods for *decomposing* a high-level description into low-level operations provided by the targeted architecture [2]–[5], as well as for *mapping* a resulting circuit to a description which complies to the target's constraints [6]–[10]. In between these steps, several optimizations may be applied in order to improve the performance of the respective computation [11], [12]. Supported by toolkits such as IBM's Qiskit [13], Microsoft's QDK [14], or Rigetti's Forest SDK [15], elaborate design flows emerged.

Each stage of these design flows creates a description of the intended functionality. During this process, it is of utmost importance that this functionality is preserved throughout all levels of abstraction and/or optimizations. To this end, equivalence checking aims to prove the functional equivalence of two quantum computations or to show by a counterexample that those computations are not equivalent. Several approaches addressing this task have been proposed in the past, e.g., based on re-writing [16], Boolean satisfiability [17], or decision diagrams [18]–[22]. However, all current approaches construct and compare the *complete* functionality of the considered computations. Even for small cases, this quickly amounts to a substantially complex task which often cannot be completed with reasonable efforts.

In this work, we propose to tackle this task from a new perspective by incorporating the power of simulation into current equivalence checking flows. More precisely, we observe that, due to the inherent reversibility of quantum operations, even small errors in general affect most (if not all) the functionality of a quantum system. This is in stark contrast to classical circuits where errors are frequently masked due to the inherent information loss introduced by many logic gates. As a consequence, in order to check the equivalence of two quantum computations it is in general not necessary to consider their complete functionality. Motivated by this, we propose to first compare the outcomes of a couple of *simulations* of both computations with arbitrary computational basis states – a task significantly less complex than covering the complete functionality. If none of those simulations reveal the non-equivalence of the circuits under consideration, any current state-of-the-art equivalence checking routine can still be utilized.

In the following, we show that this indeed leads to substantial improvements. In fact, simulation reliably allows for the detection of errors drastically faster than ever before – in most of the cases just a single run is sufficient. Additionally, if no sign of non-equivalence can be detected within a few simulations, this yields a highly-probable estimate (although no guarantee) of the operations' equivalence, while state-of-the-art equivalence checking routines frequently time out in these cases (which does not allow for any kind of conclusions). Those findings are confirmed by both theoretical as well as experimental evaluations.

The remainder of this paper is structured as follows: Section II covers the required background on quantum computing. Then, Section III discusses how equivalence checking is currently conducted in quantum computing and illustrates the power of simulation. Based on these observations, Section IV provides a theoretical discussion on the matter and describes the proposed equivalence checking flow. The substantial improvements achievable by the proposed flow are demonstrated by the experimental evaluations in Section V. Section VI concludes the paper.



(a) Quantum operations (b) Quantum circuit G (c) U describing G and G' (d) $\tilde{U}'$ describing $\tilde{G}'$

Fig. 1. Concepts of quantum computing

II. Background

To keep this paper self-contained, this section reviews the basics of quantum computing and introduces the main concepts used throughout this paper. The descriptions are kept brief, but we refer the interested reader to [23] for a more thorough introduction.

While classical computations only operate on bits, i.e., on 0 and 1, quantum computing works with so-called quantum bits (*qubits*). A single qubit $|\varphi\rangle$ cannot only be in one of the two computational basis states denoted by $|0\rangle$ and $|1\rangle$, but also in an arbitrary *superposition* $|\varphi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ with $\alpha_0, \alpha_1 \in \mathbb{C}$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Moreover, while one can determine with certainty if a classical bit is 0 or 1, one cannot determine the precise state of a qubit, i.e., the values α_i. Instead, measurement of a qubit yields a basis state ($|0\rangle$ or $|1\rangle$), with probability $|\alpha_0|^2$ and $|\alpha_1|^2$, respectively. Furthermore, a measurement collapses the state of the qubit to the computational basis state corresponding to the measurement result.

Example 1. *Consider $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This is a valid qubit state, since $|\frac{1}{\sqrt{2}}|^2 + |\frac{1}{\sqrt{2}}|^2 = 1$. After measurement, the state of this qubit collapses to $|0\rangle$ or $|1\rangle$ with probability $|\frac{1}{\sqrt{2}}|^2 = 0.5$, respectively.*

In an ensemble of n qubits, there exist 2^n computational basis states $\{|i\rangle\}_{i=0}^{2^n-1}$, where the individual qubit states correspond to the binary representation of $i \in \{0, \ldots, 2^n - 1\}$, i.e., $|i\rangle = |(i_{n-1} \ldots i_0)_2\rangle = |i_{n-1}\rangle \otimes \cdots \otimes |i_0\rangle$. The state of such a system is again described by a superposition of computational basis states, i.e., $|\varphi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$ with $\alpha_i \in \mathbb{C}$ and $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. Thus, the state of an n-qubit system may also be represented by a 2^n-dimensional complex unit vector, also called *state vector*, i.e., $|\varphi\rangle \equiv [\alpha_i]_{i=0}^{2^n-1}$.

Example 2. *Consider the so-called Bell state defined as $\frac{1}{\sqrt{2}}(|00_2\rangle + |11_2\rangle)$. The corresponding state vector representation is given by $\frac{1}{\sqrt{2}}[1\,0\,0\,1]^\top$. Measuring the first qubit of this state leads to a result of 0 or 1 – each with a probability of 0.5. By this, the state of the second qubit is already determined. The Bell state is an example of an entangled qubit state – a concept unique to quantum computing.*

In order to perform quantum computations, certain *quantum operations* (also called *quantum gates*) are applied to the qubits of a quantum system – transforming an arbitrary qubit state $|\varphi\rangle$ into another state $|\varphi'\rangle$. Such manipulations of quantum states are described as $2^n \times 2^n$ complex matrices U acting on the state vector of the qubits. Since the resulting vector shall again describe a quantum state, the matrix U must preserve the unit length of the ingoing state vector. Thus, matrices U describing quantum operations must be unitary[1].

Example 3. *Prominent representatives of single-qubit operations include the* X *operation (which negates the value of a qubit state) and the Hadamard operation* H *(which sets a qubit into superposition), whose matrix representations are given by $\left[\begin{smallmatrix}0 & 1\\ 1 & 0\end{smallmatrix}\right]$ and $\frac{1}{\sqrt{2}}\left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$, respectively. Most multi-qubit operations are so-called controlled-U operations, where a certain single-qubit operation U is applied to a specific target qubit only when the state of certain designated control qubits is $|1\rangle$. Here, the controlled-NOT (*CX *or* C-NOT*) 2-qubit operation is widely used, since in combination with arbitrary single-qubit operations it allows for universal quantum computing [23]. The functionality of the* C-NOT *operation is described by the $2^2 \times 2^2$ matrix shown in Fig. 1a.*

Finally, a *quantum computation* then consists of a sequence of quantum operations applied to an n-qubit system. This is typically visualized in terms of a *quantum circuit*, where the evaluation of qubit states is represented by individual wires and quantum operations to be performed on these qubits are represented by gates. More formally, a quantum circuit $G = g_0 g_1 \ldots g_{m-1}$ with gates $\{g_i\}_{i=0}^{m-1}$ operating on n qubits directly translates to a unitary system matrix U via $U = U_{m-1} \cdots U_0$ where U_i is the unitary matrix corresponding to gate g_i for $i \in \{0, \ldots, m-1\}$. In order to describe operations acting on a subset of the system's qubits, the matrix describing the quantum operation has to be "extended" to a $2^n \times 2^n$ representation by using tensor products of smaller matrices.

Example 4. *Fig. 1b shows an example of a quantum circuit with $m = 8$ gates operating on $n = 3$ qubits containing only Hadamard and* C-NOT *gates, where $\bullet$ indicates the control and $\oplus$ the target of a* C-NOT*. To construct the system matrix U of this circuit, the individual 2×2 or $2^2 \times 2^2$ gate matrices (cf. Example 3) have to be "extended" to $2^3 \times 2^3$ matrices and multiplied in reverse order. In case of the first Hadamard gate (applied to the second qubit), the corresponding "extended" matrix is given by $\mathbb{I}_2 \otimes \mathrm{H} \otimes \mathbb{I}_2$, where $\mathbb{I}_2$ denotes the 2×2 identity matrix. The whole functionality of the quantum computation described by G is represented by the matrix U shown in Fig. 1c.*

[1] A matrix U is unitary when $UU^\dagger = \mathbb{I}$, where $U^\dagger$ denotes the adjoint (or conjugate transpose) of U and $\mathbb{I}$ denotes the identity matrix. Thus, it holds for a unitary matrix U that $U^{-1} = U^\dagger$.

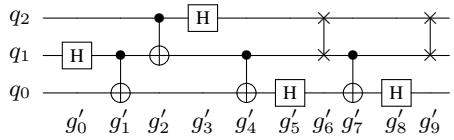


Fig. 2. Mapped quantum circuit G'

III. The Power of Simulation in Equivalence Checking of Quantum Circuits

In quantum computing, a multitude of design tasks have to be conducted in order to realize a conceptual quantum algorithm on an actual quantum computer. Here, ensuring that each design step preserves the intended functionality is of utmost importance. Equivalence checking aims to prove the functional equivalence of two quantum computations (usually provided as quantum circuits G and G') or to show by a counterexample that those computations are not equivalent. In this section, we discuss how equivalence checking is currently conducted in quantum computing and, afterwards, illustrate the power of simulation that addresses current limitations.

A. Motivation

Consider two quantum computations operating on n qubits and provided as circuits $G = g_0 \dots g_{m-1}$ and $G' = g'_0 \dots g'_{m'-1}$. As reviewed in the previous section, the functionality of both computations is uniquely described by the respective 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 respective circuit's gates. Consequently, deciding the equivalence of both computations reduces to comparing these matrices. In order to perform this comparison, these system matrices are constructed from the individual gate descriptions by subsequent matrix-matrix multiplication.

Example 5. *Consider again the circuit G from Fig. 1b. In order to realize this circuit on an actual quantum computer, restrictions in the interactions between qubits have to be satisfied. This can be done by adding* SWAP *gates*[2] *(cf. Fig. 1a for the matrix representation) and may result in a circuit G' as shown in Fig. 2 (qubits modified by* SWAP *gates are indicated by $\times$). Constructing the respective system matrices by multiplying the corresponding gate matrices $U_{m-1} \cdots U_0$ and $U'_{m'-1} \cdots U'_0$ yields the matrix U as shown in Fig. 1c in both cases. Thus, both circuits G and G' are equivalent.*

However, since the involved matrices are exponential in size with respect to the number of qubits, this quickly yields a substantially complex task (in fact, it has been proven that equivalence checking of quantum circuits is QMA-complete [24]). In order to tackle this complexity, several methods based on re-writing [16], Boolean satisfiability [17], or decision diagrams [18]–[22] have been proposed. However, while those methods indeed allow to cope with the complexity at least for small quantum circuits, the state of the art remains heavily restricted by the QMA-completeness of the underlying problem.

[2]For details on this mapping process, we refer to corresponding design works such as [6]–[10].

B. General Idea

In this work, we advocate to tackle equivalence checking of quantum circuits from a new perspective which allows to escape this complexity to some extent. The main idea rests on the observation that even small errors do not only lead to small changes in the overall behavior of a circuit, but frequently affect the circuit's system matrix in its entirety. Because of that, checking the *complete* system matrix is in general not necessary – in particular when two circuits are *not* equivalent. Then, rather than constructing the overall matrices U and U' for both computations (requiring the expensive matrix-matrix multiplications $U_{m-1} \cdots U_0$ and $U'_{m'-1} \cdots U'_0$), it is sufficient to just compare single columns (which can be constructed by less expensive matrix-vector multiplications). This is identical to simulating both circuits with a computational basis state $|i\rangle$ as input, i.e., conducting $U_{m-1} \cdots U_0 |i\rangle$ and $U'_{m'-1} \cdots U'_0 |i\rangle$ yielding the i^{th} column $|u_i\rangle$ and $|u'_i\rangle$, respectively. If those columns differ, the non-equivalence of the two given circuits has been shown.

Example 6. *Consider again the circuits G and G' from Fig. 1b and Fig. 2, respectively, but assume that a bug in the mapping tool led to a circuit $\tilde{G}'$ where the last* SWAP *gate (g'_{10}) is not correctly applied to the qubits q_1 and q_2, but rather to the qubits q_1 and q_0. Then, circuit $\tilde{G}'$ does not realize the function as described by the matrix U from Fig. 1c, but by the matrix $\tilde{U}'$ shown in Fig. 1d. Since U and $\tilde{U}'$ are obviously not identical anymore, also the circuits G and $\tilde{G}'$ have been shown to be non-equivalent. Moreover, since U and $\tilde{U}'$ differ in* all *their columns, this non-equivalence can be detected by constructing any two columns $|u_i\rangle$ and $|\tilde{u}'_i\rangle$, rather than constructing both matrices U and $\tilde{U}'$.*

Conducting equivalence checking in this fashion basically amounts to *simulating* the given circuits with arbitrary computational basis states $|i\rangle$ until a counterexample is obtained. In the classical realm, this is an established technique for preliminary testing (also called *random-stimuli simulation*), but has not led to a viable technique for equivalence checking in general. That is because, in classical circuits, masking effects and the inevitable information loss introduced by many logic gates greatly reduce the chance of detecting errors within a few arbitrary simulations – leading to many false positives after just conducting a couple of simulations. This is significantly different in quantum computing. In fact, the inherent reversibility of quantum operations drastically reduces these effects and frequently yields situations where even small errors remain unmasked and affect entire system matrices – emphasising the viability of random simulations.

All this, of course, does not guarantee that an error is indeed detected by just simulating a limited number of arbitrary stimuli $|i\rangle$. Moreover, in case two quantum circuits are equivalent, all 2^n possible computational basis states need to be checked, thus, leading to the same complexity as constructing the whole functionality. But motivated by these observations, a more detailed consideration of this direction was triggered which eventually leads to the proposal of an alternative equivalence checking flow that clearly outperforms the state of the art.

IV. Exploring and Exploiting the Power

The observations above showed that simulations of arbitrary computational basis states $|i\rangle$ are indeed viable to check the equivalence of two quantum computations without constructing and comparing their *complete* functionality. This section provides a theoretical discussion exploring the power of simulation (i.e., just partially covering the respective functionalities). Based on that, a correspondingly adjusted equivalence checking flow is proposed, which takes these findings into consideration and, by this, substantially improves the available state-of-the-art methods. Experimental evaluations (summarized later in Section V) confirm these improvements.

A. Theoretical Consideration

Consider two quantum computations (given as quantum circuits G and G') operating on n qubits, whose equivalence shall be checked. To this end, as discussed in Section III-A, the current state of the art covers the full functionality by constructing and comparing the corresponding circuit matrices U and U'. Motivated by the observations discussed in Section III-B, it is now of interest, how significantly the resulting matrices U and U' differ from each other in case of errors and whether this would make an incomplete coverage of the functionality feasible.

To this end, we 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 $U \cdot D = U'$. In the case that both matrices are identical (i.e., the circuits are equivalent), it directly follows that $D = \mathbb{I}$. One characteristic of the identity function $\mathbb{I}$ resulting in this case is that all diagonal entries are equal to one, i.e., $\langle i|U^\dagger U'|i\rangle = 1$ for $i \in \{0, \ldots, 2^n - 1\}$, where $|i\rangle$ denotes the i^{th} computational basis state. This expression can further be rewritten to

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

where $|u_i\rangle$ and $|u'_i\rangle$ denote the i^{th} column of U and U', respectively. Note that the inner product $\langle u_i|u'_i\rangle$ precisely describes the process of simulating both G and G' with initial state $|i\rangle$ and comparing the results. Hence, if only one simulation yields $\langle u_i|u'_i\rangle \neq 1$, then $|i\rangle$ proves the non-equivalence of G and G'.

Now, the question is how many computational basis states $|i\rangle$ yield $\langle u_i|u'_i\rangle \neq 1$ for a given difference D, i.e., how likely it is for an arbitrary simulation to detect possible errors. 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 this difference-circuit either represents a single-qubit or a controlled-U operation[3].

[3]Note that this does not limit the applicability of the following findings, since arbitrary single-qubit operations combined with C-NOT already form a universal gate-set (cf. Section II).

Example 7. *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*

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

The process of going from U to U', i.e., calculating $U \cdot D$, impacts all *columns of U. Thus, an error may be detected by 100% of the simulations.*

Among all quantum operations, single-qubit operations posses a system matrix least similar to the identity matrix due to their matrices' tensor product structure.

Example 8. *In contrast to Example 7, assume that G_D only consists of one operation targeted at the first qubit and controlled by the remaining $n-1$ qubits. Then, the corresponding system matrix is given by*

$$D = \begin{bmatrix} \mathbb{I}_2 & & \\ & \ddots & \\ & & \mathbb{I}_2 \\ & & & U_s \end{bmatrix}.$$

In this case, applying D to U only affects the last two columns of U. As a consequence, a maximum of two columns may serve as counterexamples—certainly a worst case scenario.

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

The gate-set provided by (current) quantum computers typically includes only (certain) single-qubit gates and a specific two qubit gate, such as the C-NOT 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 C-NOT 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, 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.

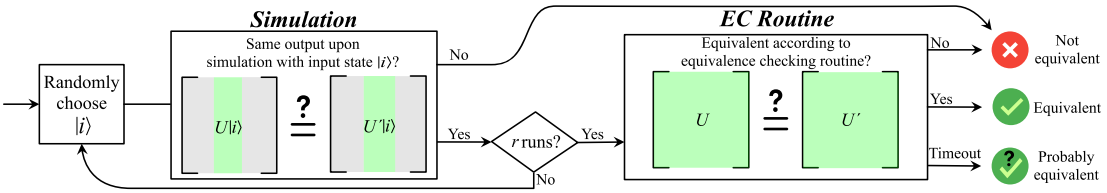


Fig. 3. Proposed equivalence checking flow

B. Proposed Equivalence Checking Flow

The motivation and considerations from above eventually led to an improved equivalence checking flow as shown in Fig. 3. 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. If one of those simulations yields different outputs, the non-equivalence of the circuits under consideration has already been shown. If this is not the case, still any current state-of-the-art equivalence checking routine (such as proposed in [16]–[22]) can be utilized to complete the task. Moreover, if no errors have been found by simulation, the likelihood of an error indeed being present is significantly reduced as shown in the discussions from Section IV-A. Overall, this eventually leads to three possible outcomes:

- *Not equivalent*, if different outputs are obtained by a simulation run.[4] As confirmed by the evaluations summarized later in Section V, 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 state-of-the-art 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.
- *Timeout*, if the simulation runs did not lead to a counterexample and the state-of-the-art equivalence checking routine was not able to complete the task within a given time limit. If this is the case, we at least get a strong indication that both circuits are equivalent (since the conducted simulations did not provide a counterexample which, according to the discussions from Section IV-A, 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.

Overall, this flow significantly improves upon the state of the art as also confirmed in evaluations which are summarized next.

V. Experimental Results

In order to evaluate the potential of the proposed approach (and, by this, the power of simulation for equivalence checking in quantum computing), we implemented the flow introduced in Fig. 3 in C++. For simulation we utilized the method proposed in [25] (downloaded from http://iic.jku.at/eda/research/quantum_simulation/) and as equivalence checking routine we used the approach which inherently comes with [26] (downloaded from http://iic.jku.at/eda/research/quantum_dd/)[5].

As benchmarks, we used quantum circuits realizing prominent quantum algorithms including Grover's algorithm, Quantum Fourier Transform, Quantum Chemistry, and Quantum Supremacy in combination with circuits from [27]. Each benchmark consists of a circuit G representing the "original" functionality and an alternative realization G', e.g., generated by decomposition [2]–[5], mapping to a dedicated architecture [6]–[10], or through optimizations [11], [12]. Common errors occurring during design flows involve altered single-qubit gates as well as misplaced/removed C-NOT gates (see discussions in Section IV-A). Thus, such errors were randomly inserted into each benchmark in order to evaluate the proposed method on non-equivalent instances. All computations have been performed on a 4.2 GHz Intel i7 machine with 32 GiB main memory running Ubuntu 16.04 using a hard timeout of one hour (3600 s) for each benchmark.

Table I provides a representative subset of the obtained results.[6] The first four columns provide the characteristics of the benchmarks (i.e., their name, number of qubits, and gate counts), while the remaining columns list the run-time (in CPU seconds) of the sole application of the equivalence checking routine (t_{ec}) as well as the run-time of the simulation part (t_{sim}). Additionally, the total number of simulation runs needed until a counterexample has been determined is shown for the non-equivalent cases. Table Ia provides the results for the non-equivalent benchmarks while Table Ib provides the results for the equivalent benchmarks.

The results clearly show the complexity of the problem. Solely considering a state-of-the-art equivalence checking routine frequently yields timeouts for both, non-equivalent as well as equivalent benchmarks. In these cases, no conclusions on the equivalence can be drawn at all. In contrast, the power of simulation becomes evident. In case of non-equivalent benchmarks, simulation is able to determine a counterexample in *all* cases – for most of them even *within a single simulation run*. This also confirms the discussions conducted in Section IV-A. Even if there is no guarantee, it is very unlikely that two circuits are not equivalent if no counterexample can

[4] In principle, this result is also possible if the state-of-the-art equivalence checking routine is employed after the simulation runs and, hence, is accordingly added in Fig. 3. However, in all our investigations, non-equivalence was already detected by simulation.

[5] Note that this approach was merely chosen as a representative. The improvements obtained in this work can be obtained for all currently existing equivalence checking routines.

[6] Due to space limitations, we needed to focus on a subset only. However, an implementation of the resulting design flow is publicly available at http://iic.jku.at/eda/research/quantum_verification/ for further evaluations.

TABLE I
EXPERIMENTAL RESULTS

(a) Non-equivalent benchmarks

Benchmark	n	$\|G\|$	$\|G'\|$	t_{ec} [s]	#sims	t_{sim} [s]
Quantum Chemistry 3x3	18	1 938	1 918	> 3600	1	1 369.85
urf4_187	11	32 004	470 414	> 3600	1	441.44
hwb9_119	9	1 544	745 685	> 3600	1	317.53
5xp1_194	17	85	47 487	> 3600	1	219.10
Supremacy 4 4 50	16	410	407	> 3600	1	137.36
Grover 9	15	6 453	6 421	> 3600	1	97.72
inc_237	16	93	40 194	> 3600	1	80.38
root_255	13	99	277 425	> 3600	1	52.70
cm85a_209	14	69	106 448	> 3600	1	29.02
dc2_222	15	75	63 932	> 3600	1	18.39
max46_240	10	107	366 115	> 3600	1	17.10
Supremacy 4 4 15	16	126	124	> 3600	1	14.29
rd84_253	12	111	119 165	> 3600	1	13.83
sqr6_259	18	81	13 802	> 3600	1	5.05
Grover 8	13	3 648	3 630	> 3600	1	3.52
Supremacy 4 4 05	16	95	92	> 3600	1	3.49
Grover 7	11	2 175	2 163	> 3600	1	1.38
pcler8_248	21	22	8 523	> 3600	1	1.32
sqn_258	10	76	36 315	> 3600	1	1.26
QFT 48	48	1 176	1 172	> 3600	2	0.22
QFT 64	64	2 080	2 070	> 3600	1	0.21
Grover 6	9	1 042	1 039	> 3600	1	0.03
Quantum Chemistry 2x2	8	368	364	5.13	1	0.02
Grover 5	9	830	827	0.17	1	0.02

(b) Equivalent benchmarks

Benchmark	n	$\|G\|$	$\|G'\|$	t_{ec} [s]	t_{sim} [s]
Supremacy 4 4 50	16	410	410	> 3600	1 339.13
Grover 9	15	6 453	6 453	> 3600	679.39
Quantum Chemistry 3x3	18	1 938	1 938	> 3600	159.63
Supremacy 4 4 15	16	126	126	> 3600	142.72
Grover 8	13	3 648	3 648	> 3600	54.08
Supremacy 4 4 05	16	95	95	> 3600	46.48
Grover 7	11	2 175	2 175	> 3600	12.27
QFT 64	64	2 080	2 080	> 3600	2.76
QFT 48	48	1 176	1 176	> 3600	1.26
Grover 6	9	1 042	1 042	> 3600	0.98
hwb9_119	9	1 544	760 994	577.19	39.12
max46_240	10	107	373 613	395.72	24.56
urf4_187	11	32 004	480 060	275.20	28.51
root_255	13	99	283 123	195.17	23.81
rd84_253	12	111	121 645	169.30	9.46
cm85a_209	14	69	108 651	121.26	9.21
5xp1_194	17	85	48 485	33.33	5.28
dc2_222	15	75	65 269	30.63	6.21
sqn_258	10	76	37 074	9.21	2.51
inc_237	16	93	41 033	7.41	4.10
pcler8_248	21	22	8 712	6.67	1.14
Quantum Chemistry 2x2	8	368	368	5.47	0.17
sqr6_259	18	81	14 111	4.94	1.60
Grover 5	9	830	830	0.07	0.18

n: Number of qubits $|G|$: Gate count of G $|G'|$: Gate count of G' t_{ec}: Equivalence checking time #sims: Performed simulations t_{sim}: Simulation time

be determined by a limited number of random simulations. Thus, it may be concluded that choosing $r \ll 2^n$, e.g., $r = 10$, suffices to reason about the operations' equivalence in practice, i.e., conducting ten random simulations before opting for the (standard) equivalence checking routine. Table Ib shows the runtime comparison of the equivalence checking routine (t_{ec}) itself and the simulation with $r = 10$ arbitrary computational basis states (t_{sim}). It can be seen, that these simulations only result in a negligible runtime overhead, while still providing a strong indication that the considered circuits are indeed equivalent – even if the equivalence checking routine fails to complete the check within reasonable time.

VI. CONCLUSIONS

In this paper, we uncovered the power of simulation for equivalence checking in quantum computing. To this end, we showed that, in contrast to classical computing, it is not always necessary to consider the whole functionality of two quantum computations in order to decide their equivalence. By conducting a limited number of simulations with arbitrary computational basis states, errors are consistently detected several magnitudes faster than with the current state of the art. The resulting equivalence checking flow drastically accelerates the detection of errors or provides a highly probable estimate of the considered operations' equivalence for a much larger class of problems. Thus, for the first time allowing to conduct equivalence checking in many cases, where no conclusion could be drawn to date.

ACKNOWLEDGMENTS

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

REFERENCES

[1] *IBM Q System One*. [Online]. Available: https://www.research.ibm.com/quantum-computing/system-one/ (visited on 03/25/2020).

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

[3] 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.

[4] 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.

[5] 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, pp. 145–150.

[6] 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.

[7] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *ASPLOS*, 2019.

[8] 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.

[9] 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.

[10] 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.

[11] Z. Sasanian and D. M. Miller, "Reversible and quantum circuit optimization: A functional approach," in *Int'l Conf. of Reversible Computation*, R. Glück and T. Yokoyama, Eds., 2013, pp. 112–124.

[12] 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.

[13] G. Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

[14] *Quantum Development Kit*, Microsoft. [Online]. Available: https://microsoft.com/en-us/quantum/development-kit (visited on 03/25/2020).

[15] *Forest SDK*, Rigetti. [Online]. Available: https://rigetti.com/forest (visited on 03/25/2020).

[16] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," in *Int'l Symp. on Nanoscale Architectures*, 2010, pp. 23–28.

[17] R. Wille, N. Przigoda, and R. Drechsler, "A compact and efficient SAT encoding for quantum circuits," in *IEEE Africon*, 2013.

[18] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Checking equivalence of quantum circuits and states," in *Int'l Conf. on CAD*, 2007, pp. 69–74.

[19] 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.

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

[21] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 86–99, 2016.

[22] L. Burgholzer and R. Wille, "Improved DD-based equivalence checking of quantum circuits," in *Asia and South Pacific Design Automation Conf.*, 2020, pp. 127–132.

[23] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

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

[25] 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.

[26] 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.

[27] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225.