Is Simulation the Only Alternative for Effective Verification of Dynamic Quantum Circuits?

Liam Hurwitz[1,2], Kamalika Datta[1,2], Abhoy Kole[2], and Rolf Drechsler[1,2]

[1] Institute of Computer Science, University of Bremen, Bremen, Germany
{hurwitz, kdatta, drechsler}@uni-bremen.de
[2] Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
abhoy.kole@dfki.de

Abstract. This paper investigates the verification gap of *Dynamic Quantum Circuits (DQC)* by analyzing state-of-the-art equivalence checking approaches. Today's *Noisy Intermediate-Scale Quantum (NISQ)* devices are limited in the number of qubits. DQCs drastically reduce the number of qubits required by guiding the outcome based on the intermediate results of the computations. Investigation of feasibility of existing verification tools with respect to DQC verification is needed. In order to verify the equivalence of DQCs, verification tools often transform dynamic primitives in order to reveal the underlying functionality of the circuits. This leads to restoration of their unitary functionality and allows existing equivalence checkers to reason about DQCs. Our objective is to provide empirical data that can be used to improve these tools by examining their capabilities and effectiveness. Equivalence checking methods that use ZX-Calculus, *Quantum Multi Valued Decision Diagrams (QMDD)* and simulators are considered in this regard. In order to gauge the effectiveness of the present tools, we use the Bernstein–Vazirani, Deutsch-Jozsa and Quantum Phase Estimation algorithms and their dynamic variants. Experiments reveal that the existing equivalence checking tools are limited in their effectiveness, while simulation based approaches provide correct verifications at the expense of high runtime overhead. Our results show that the different verification methods never achieves accuracy of more than 50%.

Keywords: Dynamic Quantum Circuit · Equivalence Checking · Verification · QMDD · ZX-Calculus · Quantum Simulation

1 Introduction

Quantum computing is a rapidly advancing field with the potential to revolutionize how we solve complex problems. Quantum computers take advantage of quantum mechanical principles, such as *superposition* and *entanglement* to solve certain problems. Currently, we are in the *Noisy Intermediate Scale Quantum (NISQ)* era, where the fabricated quantum computing devices are noisy and have limited number of qubits. While current quantum computers, such as the IBM

Condor, can scale up to 1211 qubits, they are not advanced enough for fault-tolerant computing. The long-term goal in quantum computation is to achieve *Fault-Tolerant Quantum Computing*, thus providing robustness and opening the door to more applications. Recently with the introduction of non-unitary operations such as mid-circuit measurement, active resets and classically controlled quantum operations, a new class of circuits known as *Dynamic Quantum Circuits (DQC)* have emerged [7]. The mid-circuit measurements in a computation enable the measurement of an outcome during the execution of an intermediate stage, where subsequent gate operations depend on this measurement. This allows non-unitary operations to be combined with unitary operations. One fundamental difference between *Static Quantum Circuits (SQC)* and DQC is that, future states of DQC depend on the outcome of measurements that occur in the circuit. Also, the measurement always occurs at the end of the circuit for SQC i.e. states of a SQC, have no dependencies [7]. This brings us one step closer to overcoming the limitations of NISQ era by reducing the number of qubits required for executing any algorithm.

DQCs allow the realization of any quantum algorithm with fewer qubits. Recent works have targeted the design of dynamic versions of Toffoli gate [14] thereby paving the way for dynamic realization of *Deutsch–Jozsa (DJ)* algorithm [8]. Another work [13] targets the dynamic realization of *Multiple Control Toffoli (MCT)* gate that further allows for dynamic realization of many more quantum algorithms. Now the task is to verify whether the functionality of the dynamically realized algorithm is correct, as compared to its static counterpart.

Existing verification tools have their merits and demerits with respect to dynamic circuit verification [4,10]. To verify the equivalence of DQCs, the verification tools transform the dynamic primitives to represent the underlying functionality of the circuits. This leads to restoration of their unitary functionality and permits existing equivalence checkers to reason about DQCs. Even when we transform a DQC to a SQC, there still exist some issues because of which available tools for SQC verification fail. The challenge is that the qubit order is lost during the unitary reconstruction of the DQC to the SQC, and hence exiting methods cannot be directly applied. In this paper we primarily focus on analyzing existing equivalence checking methods based on ZX-Calculus, and *Quantum Multi-Valued Decision Diagrams (QMDD)*. We also use the Qiskit Aer simulator [16] for verifying our results. We use the *Bernstein–Vazirani (BV)*, *Deutsch-Jozsa (DJ)* and *Quantum Phase Estimation (QPE)* algorithms and their dynamic variants [7,14] for the experiments. The goal of this work is to provide empirical data for evaluating the effectiveness of equivalence checkers for DQCs, as well as finding the best candidate from a set of DQCs for a quantum algorithm. Experimental results show that existing verification tools have certain limitations while verifying DQCs. In contrast, simulation based approaches provide accurate results; however, they are not scalable and become very expensive in terms of memory and run-time when the circuit size increases. Our experiments reveal that the accuracy level of the considered verification methods does not exceed 50% and depends on their implementation.

2 Background

2.1 Quantum Gates and Quantum Circuits

The state of a quantum system is manipulated by *quantum operations*, known as *quantum gates*. Every such operation on a quantum system is reversible and must yield a valid quantum state; thus, the operation is represented by a $2^n \times 2^n$ unitary matrix U, where n is the number of qubits on which the gate operates. The only constraint for a quantum operation is that the matrix U needs to be unitary.

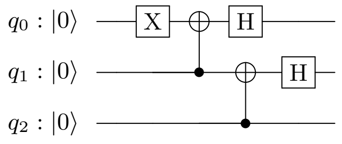


Fig. 1: A Quantum Circuit

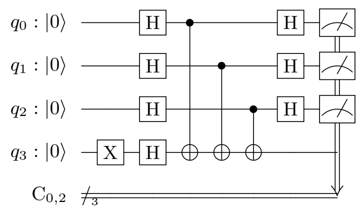
Fig. 1 shows a 3-qubit quantum circuit consisting of NOT (X), CNOT (CX) and *Hadamard (H)* gates. Generally, circuits are evaluated on physical quantum computers. This mode of computation is considered static, as the result of the outcome is available only after the execution of all the gates, and it does not have the capability to perform computations based on intermediate outcomes.

2.2 Dynamic Quantum Circuit

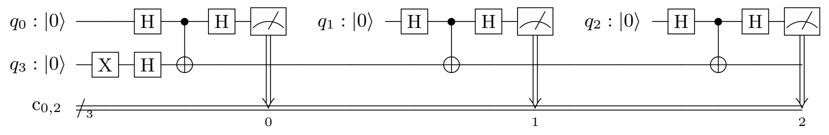
The traditional quantum circuit model forms the basis for implementing quantum circuits on physical quantum devices. Recently, IBM has introduced the capability to manipulate quantum circuits in real-time and perform mid-circuit measurements [7]. These advancements have laid the groundwork for the realization of DQCs. DQCs have the potential to realize any n qubit quantum circuit using minimum 2 qubits only, as opposed to traditional or static circuits requiring at least n-qubits.

Many recent works have shown the advantage of using DQCs for realizing various algorithms like QPE, BV and DJ. In [14] two methods were introduced for converting a Toffoli gate into its dynamic analogue. This work in particular shows the dynamic implementations of 3-qubit DJ circuits using two different dynamic Toffoli designs, namely *dynamic 1* and *dynamic 2*.

Fig. 2(a) shows the BV algorithm to determine a 3-qubit hidden string 111 from a given black-box function that implements the function $\mathcal{F}(x) = xyz$. This uses three data qubits (q_0, q_1 and q_2) that are initialized to $|0\rangle$ state, and an answer qubit (q_3) initialized to $|-\rangle \big(= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\big)$ state. The main idea of dynamic transformation is to transform the static circuit consisting of various data and answer qubits into a single data qubit and equal number of answer



(a) BV Algorithm



(b) Dynamic Realization of BV Algorithm

Fig. 2: Static and Dynamic Representation of BV Algorithm

qubits. In the case of the BV algorithm, we use one data qubit and one answer qubit. It may be noted that while performing the transformation, we need to consider independent and dependent 1- and 2-qubit operations. This information allows us to determine how the dynamic transformation will follow.

Fig. 2(b) shows the dynamic realization of BV algorithm. It can be observed that we require three iterations, involving one data qubit and one answer qubit. All the operations present between a reset and a measurement on the data qubit are performed within an iteration. The three iterations entail evaluation of all the gate operations between qubits (q_0, q_3), (q_1, q_3) and (q_2, q_3) respectively, along with a reset operation on the data qubit after the first and second iteration.

2.3 Verification of Quantum Circuits

Design and verification of quantum circuits is a challenging task due to the inherent complexity of quantum systems. Simulation can be an effective tool for small quantum circuits and their verification. It allows for the comparison of the measurement outcomes of circuits. However, quantum simulators are not effective for complex or large-scale quantum systems. Current scalable approaches to address this challenge use *Quantum Multiple-Valued Decision Diagrams (QMDDs)*, *Tensor Decision Diagrams (TDDs)* and ZX-Calculus. All of these approaches have been shown to be effective in verifying the equivalence of static quantum circuits, yet have different strengths and weaknesses.

Tensor Decision Diagrams *(TDD)* [10] offer a compact representation of Boolean functions and are inspired by tensor networks. They are used widely for formal verification, artificial intelligence and cryptography. In recent years, TDDs have attracted attention in the field of quantum computing, due to their ability to efficiently represent and manipulate quantum circuits. In [10] Hong et al. have provided efficient methods for verifying the equivalence of static and dynamic quantum circuits. They formally define DQCs and have characterized their functionality in terms of ensembles of linear operators. Equivalence is checked by verifying that the two DQCs have the same functionality. Here, the authors present a unified representation for each component of the DQC as a tensor. This allows DQCs to be interpreted as a tensor decision network and represented as a TDD. The equivalence checking operation then checks if the two DQCs share the same TDD representation.

Quantum Multivalued Decision Diagrams (QMDD) are a type of decision diagrams capable of representing multiple values simultaneously, making them well-suited for depicting quantum states. The fundamental concept behind QMDDs involves portraying a quantum state as a tree structure, where each node signifies a superposition of quantum states, and the edges denote transitions between different superpositions. Leveraging QMDDs allows for the efficient representation and manipulation of quantum states, a crucial aspect in verifying the equivalence of quantum circuits.

Several studies use QMDDs for verifying the equivalence of quantum circuits. For instance, in [20], the authors proposed a method for verifying the equivalence of two quantum circuits by constructing their corresponding QMDDs and comparing them. The authors demonstrated that their approach is efficient and scalable [21]. It is shown that decision diagrams can efficiently represent large quantum states by leveraging the principles of quantum mechanics [3].

ZX-Calculus has drawn the attention of researchers from different fields such as quantum circuit optimization, error correcting codes, circuit simulation, extraction and equivalence checking, as well as measurement based computing, tensor networks, variational circuits and quantum natural language processing [18]. A literature survey [6] was published by Bob Coecke. It has several advantages over traditional methods for equivalence checking. First, it provides a graphical representation of quantum circuits that is easy to understand and analyze [6]. Second, algorithms for equivalence checking exist [9,12]. The disadvantage of ZX-Calculus is that it is not efficient for quantum circuits containing Toffoli gates, as the rewrite rules take longer to terminate due to gate decomposition. The ZH-Calculus is a promising alternative for quantum circuits with multiple controlled Toffoli gates [1]. Equivalence checking using the ZX-Calculus has not yet been proven to be complete for quantum circuits. If the rewrite rules cannot prove equivalence, no conclusion can be drawn. In an earlier work Seiter et al. [17] have exploited the QMDD data structure for property checking in quantum circuits. As a future work we can also extend the concept for dynamic quantum circuit.

3 Verification of Dynamic Quantum Circuits (DQC)

Checking the equivalence of two quantum circuits relies on the reversibility of quantum operations [15]. Every quantum operation is unitary and hence reversible. The product of any quantum operation and its inverse (adjoint) will always yield identity. G is an abstract representation of a quantum circuit such as QMDD or ZX-Calculus, while U represents the matrix operator of the circuit. If U is equivalent to U', this implies G is equivalent to G' and vice versa. Given two quantum circuits G and G', where g and g' are the individual gates of each circuit, their equivalence is defined as:

Definition 1. *Given two quantum circuits*

$$G = g_o \dots g_{m-1} \text{ and } G' = g'_0 \dots g'_{n-1}$$

and the respective system matrices for the two circuits

$$U = U_{m-1} \dots U_0 \text{ and } U' = U'_{n-1} \dots U'_0,$$

the problem of equivalence checking *is to verify, whether*

$$U = e^{i\sigma} U' \text{ or } UU'^{\dagger} = e^{i\sigma}\mathbb{I},$$

given $\sigma \in (-\pi, \pi]$ denotes a physically unobservable global phase.

In general, checking the equivalence of quantum circuits becomes increasingly difficult because the size of the matrices grows exponentially with the number of qubits. Equivalence checking is *Quantum Merlin Arthur (QMA)*-Complete. Problems that are QMA-Complete can be solved efficiently using quantum computing algorithms.

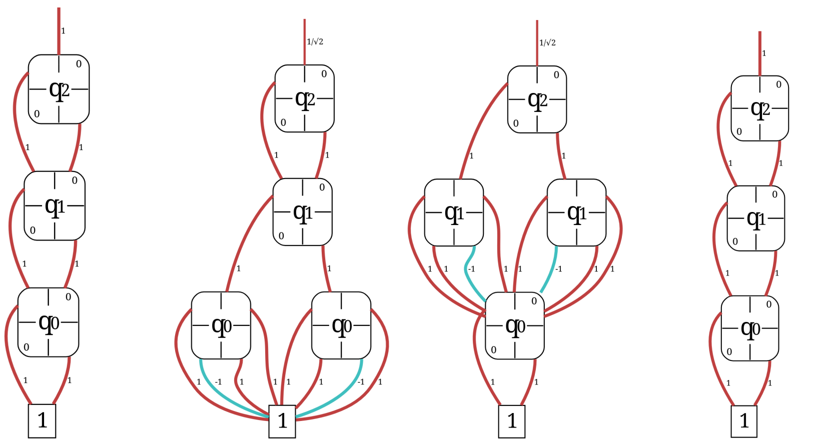
Def. 1 is not complete as it does not always consider circuits equivalent when the number of qubits differ, or the circuit has non-unitary operations or even when the system matrices differ. The cause for the non-equivalence between the system matrices could be numerical inaccuracies or permutations in the input or outputs of the circuits [4]. In order to ensure that compiled or optimized quantum circuits do not alter the functionality or the intended behavior, equivalence checking needs to be able to deal with the variations that exist in the system matrices [3]. Current equivalence checking tools [11, 20] try to avoid using the matrix based operator representation (system matrix), since the representation of an n qubit quantum circuit requires a matrix of the size $M^{2^n \times 2^n}$.

3.1 QMDD based Verification

The QMDD equivalence checking algorithm verifies the equality of two quantum circuits G and G' according to Def. 1. It first constructs a decision diagram representation of each circuit's functionality: first by applying each gate g from the circuit G onto the initial identity, and next by applying the inverse functionality of $G'^{-}1$ as individual gates g'^{-1}.

$$G'^{-1} \cdot G = (g'^{-1}_{m'-1} \cdots g'^{-1}_{0}) \cdot (g_0 \cdots g_{m-1}) = G' \rightarrow \mathbb{I} \leftarrow G \tag{1}$$

If the resulting diagram is the identity decision diagram, the circuits are equivalent; otherwise, they are not [20]. Fig. 3 exemplifies the process of verifying the equality of two quantum circuits, G and G' with QMDD. Fig. 3(a), labeled 'Initial Identity $\mathbb{I}$', shows the QMDD representing the initial identity operator. Next, Fig. 3(b) displays the QMDD of the given circuit G, In Fig. 3(c), referred to as 'Intermediate', we observe an intermediate state during the construction of the QMDD for the inverted circuit G'. Specifically, all the gates of the first qubit have already been applied to the previous diagram. Fig. 3(d) exhibits the resulting QMDD when applying $G' \rightarrow \mathbb{I} \leftarrow G$, proving they yield the identity. MQT DDVis [19] is a tool for visualizing simulation and verification for QMDD. While QMDDs typically enable compact representation of quantum systems,



(a) Initial Identity $\mathbb{I}$ (b) QMDD of G (c) Intermediate (d) $G' \rightarrow \mathbb{I} \leftarrow G$

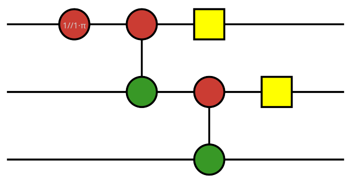
Fig. 3: Example of QMDD verification using MQT DDVis for Fig. 1.

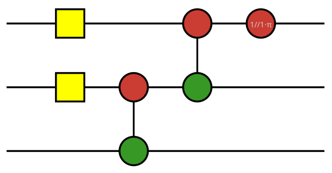
their size can still grow exponentially with the number of qubits in the worst-case scenario, where no redundancy in the state description can be exploited, resulting in minimal node sharing and exponential node growth [3]. They should be used for computing the full state vector, since the QMDD remains compact [2]. In conclusion, QMDDs have emerged as a powerful tool for verifying the equivalence of quantum circuits. Numerous studies have explored their application for this purpose, proposing innovative techniques for constructing, manipulating and visualizing QMDDs [20].

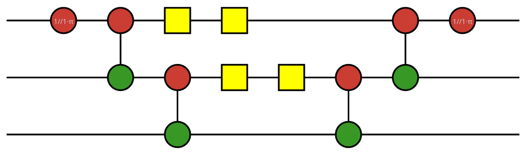
3.2 ZX-Calculus based Verification

ZX-Calculus is a rigorous graphical language for reasoning about quantum circuits and algorithms. It extends the categorical quantum mechanics school of reasoning [5], using the paradigms of *monoidal category theory*. ZX-Calculus provides a solution to the problem of equivalence checking. By representing quantum circuits as diagrams referred to as ZX-Diagrams, it is possible to manipulate these diagrams using a set of graphical rewrite rules [18]. These rules allow us to verify, if two quantum circuits, which were transformed into ZX-Diagrams, are equivalent [20].

The transformations utilized by the ZX-Calculus equivalence miter [11], proves that $G' \cdot G^\dagger = I$, where G and G' are quantum circuits and $G^\dagger$ is the adjoint (transposed complex conjugated) of G. The equivalence checking miter leverages Def. 1, which states that if two quantum circuits are equivalent, the adjoint of one circuit applied to the second circuit can be rewritten as the identity. If the reduced ZX-Diagram consists only of bare wires, both circuits are equivalent [18]. Fig. 4 shows how the equivalence checking miter proves the equality of the quantum circuit shown in Fig. 1 with itself.


(a) Step 0: Obtain ZX-Diagram D


(b) Step 1: Adjoint $D^\dagger$ of D


(c) Step 2: Concatenation of $D_m = D \cdot D^\dagger$

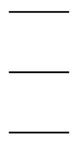

(d) Step 3: Bare Wires

Fig. 4: Equivalence Check using ZX-Calculus [4, 11] of Fig 1

The equivalence miter checking method involves several steps, that are depicted in Fig. 4(b), 4(c), and 4(d). A key benefit of utilizing ZX-Calculus rewrite rules lies in their non-deterministic nature, which enables the application of each rule until exhaustive application of the rule, followed by a subsequent rewrite rule without requiring a reapplication of the prior rewrite rule [12]. This non-deterministic rewrite rule application in ZX-Calculus allows for a flexible and efficient approach for reducing ZX-Diagrams. First, the adjoint ZX-Diagram $D^\dagger$ of Fig. 1 is obtained, as shown in Fig. 4(b). The adjoint operation switches the inputs with the outputs and vice versa, implicitly reversing the order of the operations for each wire. Additionally, every phase is negated; for instance, the X gate has a phase of π, which becomes $-\pi$ after negation. Because phases have the range $0 \leq \varphi < 2\pi$ and are periodic, the phase remains π. It can be observed that $-\pi = \pi \mod 2\pi$, since it represents the same angle in the periodic domain. The concatenation $D_m = D \cdot D^\dagger$ is formed by connecting the output of the first diagram with the inputs of the second diagram, as displayed in Fig. 4(c). The resulting diagram D_m is reduced as much as possible using ZX-Graph rules, resulting in the simplified diagram shown in Fig. 4(d). Finally, it is checked whether the reduced diagram consists of only bare wires. This can be done efficiently by verifying whether every input directly connects to an output. If so, the two original circuits are equivalent up to global phases, as demonstrated in [4, 11].

The advantage of the ZX-Calculus for equivalence checking is that computing the equivalence becomes computationally cheap, after the $D \cdot D'^\dagger = D_m$ has been reduced, since checking if a ZX-Diagram contains only 'bare' wires requires checking if every input node is directly connected to output node. If two circuits are not equal, Def. 1 will yield the difference between both circuits instead of the identity. The difference can be quantified with various methods, such as the trace-distance, fidelity or the Hilbert-Schmidt inner product. This provides a measure of how well the first circuit matches the behavior of the second circuit. Quantifying this difference in a memory efficient way is an active research topic.

3.3 Simulation

Our informal definition for equivalence of quantum circuits states that two quantum circuits are equal, if they have the same measurement distribution. To simulate a quantum circuit, the quantum registers should be initialized to the $|0\rangle$ state. After each operation is applied, measurements are conducted. This process is then repeated multiple times. The simulator returns the number of occurrences of a specific measurement result. The following points are considered for simulation based method:

1. Two circuits are equal if their measurement distributions are identical.
2. Two circuits can be considered similar if the difference in their measurement distributions falls within a certain percentage, attributable to their probabilistic nature.
3. If there is a significant discrepancy between the measurement distributions, we conclude that the circuits are unequal.

We utilized the Qiskit Aer simulator [16] to acquire the measurements for each quantum circuit. This requires transpiling the quantum circuits for the Aer simulator (a successor to the previously deprecated Qasm simulator). If a SQC and a DQC produce identical measurements, they are regarded as equivalent.

However, when they differ, the degree of similarity is determined using Eq. 2, where s represents the number of shots, n denotes the measurement distribution of the desired outcome for the SQC, and m signifies the corresponding measurement count for the DQC.

$$\text{Similarity} = \left(1 - \frac{|n-m|}{s}\right) \times 100 \tag{2}$$

3.4 Unitary Reconstruction for DQCs

Another approach to verify DQCs is known as unitary reconstruction that was proposed in [4]. This method involves transforming the dynamic primitives to reveal their underlying unitary functionality by overcoming mid-circuit resets, and applying the deferred measurement principle to delay all measurements to the end of the circuit.

The first step transforms a n-qubit circuit containing r active reset operations into a $(n+r)$-qubit quantum circuit and eliminates qubit reuse. In the second step, all phase rotations controlled by measurement outcomes are replaced by phase gates controlled directly by the respective qubit, that was measured previously. Next all measurements are moved to the end of the circuit, thus removing any mid-circuit measurements [4], applying the principle of deferred measurement [15]. The verification of the original and reconstructed SQCs is not straight forward, as their qubit order differs. Fig. 5 shows the unitary reconstruction of the dynamic BV-111 from Fig. 2(b).

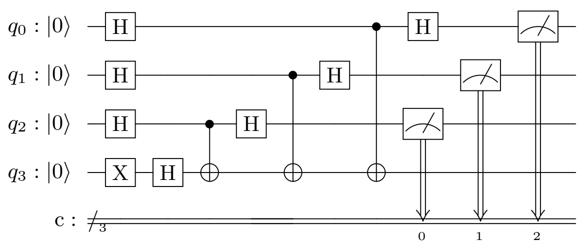


Fig. 5: Unitary Reconstruction of Fig. 2(b)

Unitary reconstruction enables existing equivalence checking tools to verify the equivalence of DQCs, as only unitary quantum operations are present.

4 Experimental Evaluation

To evaluate the effectiveness of existing equivalence checkers for verifying DQC, we perform the following series of experiments:

1. Compare the similarity between static and DQCs using Qiskit Aer quantum simulator.
2. Performance evaluation of existing verification tools for DQCs.

For performing the experiments, following algorithms are used as benchmarks:

– *Deutsch-Josza (DJ)* algorithm: A quantum algorithm designed to determining whether a boolean function is constant or balanced with just one query when given access to a quantum oracle. This circuit also consists of Hadamard, CNOT, and measurement gates.
– *Bernstein-Vazirani (BV)* algorithm: An extension of the DJ algorithm capable of finding the hidden string within a black box function using exponentially fewer queries compared to classical methods. The circuit involves Hadamard gates, controlled-NOT gates, and measurement operations.
– *Quantum Phase Estimation (QPE)*: A fundamental subroutine in many quantum algorithms, including Shor's factoring algorithm. It estimates the eigenvalues of a unitary operator with high precision.

One goal of this experiment is to publish a reproducible benchmark for increasing the accessibility and effectiveness of quantum circuit equivalence checkers. The benchmarking tool requires that all .qasm files follow the naming convention of MQT Bench. Each equivalence checker verifies the equivalence between variations of a quantum circuit and their hardware-independent versions, and exports the results in a CSV or XLS format. The source code for this Benchmark and Unitary Reconstruction can be found at `https://codeberg.org/QuantumHB/equivalence`. The benchmark folder contains a README file with a step-by-step guide to reproduce the results of this experiment, as well as further benchmarks that can be used to assess the efficiency and effectiveness of SQCs and DQCs.

4.1 Similarity Comparison

In this experiment, we simulate a set of SQC benchmarks, namely the *Bernstein-Vazirani (BV)* algorithm, *Deutsch-Josza (DJ)* algorithm, and *Quantum Phase Estimation (QPE)* algorithms along with their dynamic counterparts from [14]. Simulation is performed to verify the equality of DQCs. The simulation results and details are presented in Table 1. We use Qiskit's Aer simulator for this purpose. But it may not always be efficient or practical due to the resource requirements in simulating large quantum systems.

Table 1 shows the simulation results for SQC and DQC. The first column represent the name of the quantum circuit, the second column specifies which type of the quantum circuit was instantiated. Third to sixth column provide details about number of inputs N, depth D, number of single-qubit gates G_1, and number of two-qubit gates G_2 required for the static quantum circuit. The seventh till eleventh column show the version Ver, i.e. which dynamic variant

Table 1: Similarity Comparison of Execution Outcome of SQC and DQC

Circuit	Type	SQC N	SQC D	SQC G_1	SQC G_2	DQC Ver	DQC N	DQC D	DQC G_1	DQC G_2	Similarity
BV	001	4	5	8	1	-	2	13	8	1	100
	010	4	5	8	1	-	2	13	8	1	100
	011	4	6	8	2	-	2	14	8	2	100
	100	4	5	8	1	-	2	13	8	1	100
	101	4	6	8	2	-	2	14	8	2	100
	110	4	6	8	2	-	2	14	8	2	100
	111	4	7	8	3	-	2	15	8	3	100
DJ	And	3	15	12	6	1	2	23	17	6	75
						2	2	26	21	6	99
	Carry	4	35	26	18	1	2	60	41	18	100
						2	2	68	49	18	100
	Const 0	3	3	6	1	-	2	7	6	2	100
	Const 1	3	3	7	0	-	2	7	7	1	100
	Imply 1	3	17	13	7	1	2	26	18	7	74
						2	2	29	22	7	97
	Imply 2	3	17	13	7	1	2	25	18	7	74
						2	2	28	22	7	97
	Inhib 1	3	16	12	7	1	2	24	17	7	76
						2	2	27	21	7	99
	Inhib 1	3	16	12	7	1	2	25	17	7	73
						2	2	28	21	7	99
	Invert 1	3	6	7	1	-	2	10	7	2	100
	Invert 2	3	6	7	1	-	2	8	7	2	100
	Nand	3	16	13	6	1	2	24	18	6	72
						2	2	27	22	6	97
	Nor	3	18	13	8	1	2	27	18	8	74
						2	2	30	22	8	97
	Or	3	17	12	8	1	2	26	17	8	75
						2	2	29	21	8	99
	Pass 1	3	5	6	1	-	2	9	6	2	100
	Pass 2	3	5	6	1	-	2	8	6	2	100
	Xnor	3	7	7	2	-	2	11	7	2	100
	Xor	3	6	6	2	-	2	10	6	2	100
QPE	00	3	8	5	0	-	2	11	6	0	100
	01	3	8	5	0	-	2	11	6	0	100
	10	3	8	5	0	-	2	11	6	0	100
	11	3	8	5	0	-	2	11	6	0	100

from [14] was evaluated, number of inputs N, depth D, number of single-qubit gates G_1, and number of two-qubit gates G_2 required for the DQC. The last column contains the similarity between the SQCs and DQCs, which is calculated by running the Qiskit Aer simulator 1024 times. If both circuits both generate the same measurements distributions, they have a similarity of 100% and are considered equivalent and are marked green. Otherwise, their similarity is calculated with Equation 2. Yellow indicates that they behave similarly, and red signifies greater variance between the measurements counts.

4.2 Evaluation of Equivalence Checkers

In this evaluation, we have considered MQT and PyZX equivalence checkers. These tools are publicly available. Several TDD based checkers are available as well; however, all the TDD implementations are plagued by runtime errors, making it impossible to include them in the evaluation study. In order to compare the accuracy of the PyZX and MQT ZX-Calculus implementations for DQCs, a preprocessor was added to PyZX benchmark, which applies the unitary reconstruction if necessary.

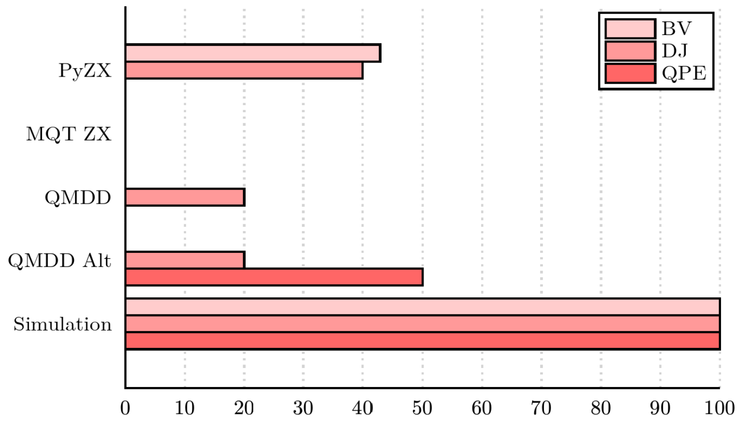


Fig. 6: Accuracy of PyZX, MQT QCEC and the Qiskit Aer Simulator

In Fig. 6, we present the result of various equivalence checkers after evaluating on our considered benchmarks. Only benchmarks, that were 100% equivalent using simulation are included in this part of the experiment. The Y-axis represents different methods for equivalence checking, and the X-axis represents the accuracy in percent. *PyZX* (version 0.8.0) coupled with the unitary reconstruction preprocessor achieved an accuracy of 42% for BV instances, 40% for DJ instances, and failed for all QPE instances, because the custom unitary reconstruction algorithm, which was added for this experiment, does not support all gates that are necessary for QPE.

Next, we evaluated different equivalence checking methods supported by version 2.4.3 of the *Quantum Circuit Equivalence Checker (QCEC)*, that is part of the *Munich Quantum Toolkit (MQT)* [3]. MQT ZX, the ZX-Calculus method, did not perform any correct evaluations, indicated by zero percent across all benchmark BV, DJ, and QPE instances.

The *Construction Equivalence Checker (QMDD)* [3] did not perform correct verifications for BV and QPE instances but managed to achieve an accuracy of 20% for the DJ instances (Invert2 and XNOR). It reported equivalence up to a global phase. The *Alternate Equivalence Checker Method (QMDD Alt)* [3], correctly classified 2 out of 4 QPE circuits and managed to correctly verify the equality of the Const-1 and Const-2 version of the DJ algorithm achieving an accuracy of 50% and 20%. Both QMDD methods combined achieved an accuracy of 30% for all circuits.

Table 2 provides a more in depth comparison of various state-of-the-art equivalence checkers. The table offers analysis consisting of individual equivalence check for each type of equivalence checker and for individual circuit version, that are 100% similar using simulation. An `X` indicates a false negative when the checker returned not equivalent, timeout or undecidable for an equivalent circuit. The ✓ denotes a correct verification and ⋆ represents a software crash. The *p* signifies that equivalence could be proven up to a global phase.

Table 2: Verification Results for BV, DJ and QPE Circuits

Circuit				**MQT**			**PyZX**
	Type	Qubits	Version	QMDD	Alt	ZX	
BV	001	4	-	x	x	x	x
	001	4	-	x	x	x	✓
	010	4	-	x	x	x	x
	011	4	-	x	x	x	x
	100	4	-	x	x	x	✓
	101	4	-	x	x	x	x
	001	4	-	x	x	x	✓
DJ	Carry	4	1	x	x	x	x
			2	x	x	x	x
	Const 0	3	1	x	✓	x	✓
	Const 1	3	1	x	✓	x	✓
	Invert 1	3	1	x	x	x	x
	Invert 2	3	1	p	x	x	x
	Pass 1	3	1	x	x	x	x
	Pass 2	3	1	x	x	x	x
	Xnor	3	1	p	x	x	✓
	Xor	3	1	x	x	x	✓
QPE	00	3	-	x	✓	x	⋆
	01	3	-	x	x	x	⋆
	01	3	-	x	✓	x	⋆
	01	3	-	x	x	x	⋆

4.3 Critical Analysis

We can safely say that no tool achieved an accuracy higher than 50% for any of the three quantum algorithms. This emphasizes the imperative for further development and improvement of equivalence checking methodologies tailored for DQCs. There is a need for refining tools for verifying equivalence of DQCs. Additionally, there is a call for defining missing functionality in current equivalence checkers, such as a similarity metric. Overall, the study provides valuable data for critical evaluation of existing equivalence checkers and motivation for proposing new approaches for checking the equivalence of DQCs. Furthermore, it is assumed that the accuracy of MQT-QCEC is low for DQCs because MQT-QCEC does not correctly handle the qubit permutations that occur during the unitary reconstruction of DQCs. The experiment has revealed that no equivalence check-

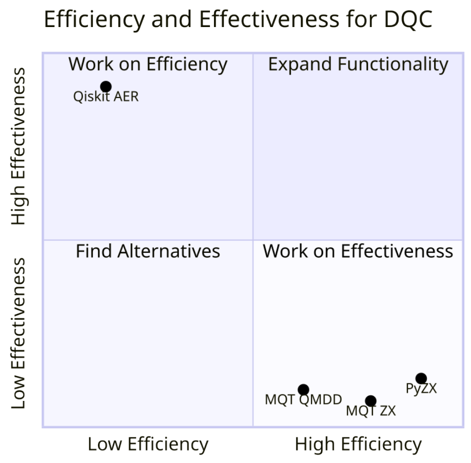


Fig. 7: Efficiency and Effectiveness for Dynamic Quantum Circuits

ing method except simulation is effective for dynamic quantum circuits, as their accuracy reaches, at best, a limited level. Fig. 7 is a depiction of the verification gap for DQCs. It shows that the current state-of-the-art equivalence checkers are not effective and efficient for verifying the equivalence of DQCs. State-of-the-art equivalence checkers need to enhance their accuracy to be effective for DQCs.

5 Conclusion

In this paper we investigate and analyze the verification gap of the existing state-of-the-art equivalence checking approaches for DQCs. In the current NISQ era, DQCs are considered a viable alternative because it reduces the number of qubits required to realize any algorithm. We particularly study and analyze the existing tools to verify the DQCs. We can safely say that none of the existing tools can fully verify the DQCs. The highest accuracy achieved by these tools for various benchmarks is 50%. This clearly demonstrates the need for designing improved verification tool for DQCs. Only simulation based method using Qiskit Aer provide 100% accuracy for a set of benchmarks. This answers the question that we raise in the title of the paper. As a future work we can also exploit various decision diagrams for property checking in quantum circuits.

Acknowledgment

We are grateful to Lennart Weingarten for his support in designing the quantum circuits and helpful guidance.

References

1. Backens, M., Kissinger, A.: ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity. Electronic Proceedings in Theoretical Computer Science **287**, 23–42 (Jan 2019)
2. Burgholzer, L., Ploier, A., Wille, R.: Tensor networks or decision diagrams? guidelines for classical quantum circuit simulation. arXiv preprint arXiv:2302.06616 (2023)
3. Burgholzer, L., Wille, R.: Advanced equivalence checking for quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **40**(9), 1810–1824 (2020)
4. Burgholzer, L., Wille, R.: Handling Non-Unitaries in Quantum Circuit Equivalence Checking. In: Design Automation Conference. pp. 529–534. ACM, San Francisco Californiaoffer (Jul 2022)
5. Coecke, B., Duncan, R.: Interacting Quantum Observables. In: Aceto, L., Damgård, I., Goldberg, L.A., Halld órsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming. pp. 298–310. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
6. Coecke, B., Horsman, D., Kissinger, A., Wang, Q.: Kindergarden quantum mechanics graduates ...or how I learned to stop gluing LEGO together and love the ZX-Calculus. Theoretical Computer Science **897**, 1–22 (2022)
7. Córcoles, A.D., Takita, M., Inoue, K., Lekuch, S., Minev, Z.K., Chow, J.M., Gambetta, J.M.: Exploiting Dynamic Quantum Circuits in a Quantum Algorithm with Superconducting Qubits. Physical Review Letters **127**(10), 100501 (Aug 2021)
8. Deutsch, D., Jozsa, R.: Rapid solution of problems by quantum computation. In: Proc. Royal Society of London. vol. 439, pp. 553–558 (1992)
9. Duncan, R., Kissinger, A., Perdrix, S., Van de Wetering, J.: Graph-theoretic Simplification of Quantum Circuits with the ZX-Calculus. Quantum **4**, 279 (Jun 2020)
10. Hong, X., Feng, Y., Li, S., Ying, M.: Equivalence Checking of Dynamic Quantum Circuits. In: International Conference on Computer-Aided Design (ICCAD). pp. 1–8 (2022)
11. Kissinger, A., Van de Wetering, J.: PyZX: Large Scale Automated Diagrammatic Reasoning. Electronic Proceedings in Theoretical Computer Science **318**, 229–241 (May 2020)

12. Kissinger, A., Van de Wetering, J.: Reducing the number of non-clifford gates in quantum circuits. Physical Review A **102**(2) (aug 2020)
13. Kole, A., Deb, A., Datta, K., Drechsler, R.: Dynamic Realization of Multiple Control Toffoli Gate. In: Design, Automation & Test in Europe Conference & Exhibition, DATE 2024. pp. 1–6 (2024)
14. Kole, A., Deb, A., Datta, K., Drechsler, R.: Extending the Design Space of Dynamic Quantum Circuits for Toffoli based network. In: Design, Automation & Test in Europe Conference & Exhibition, DATE 2023. pp. 1–6. IEEE (2023)
15. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, Cambridge; New York (2010)
16. Qiskit contributors: Qiskit: An open-source framework for quantum computing (2023). https://doi.org/10.5281/zenodo.2573505
17. Seiter, J., Soeken, M., Wille, R., Drechsler, R.: Property checking of quantum circuits using quantum multiple-valued decision diagrams. In: Reversible Computation: 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012. Revised Papers 4. pp. 183–196. Springer (2013)
18. Van de Wetering, J.: ZX-Calculus for the working quantum computer scientist, arXiv preprint arXiv:2012.13966. (2020)
19. Wille, R., Burgholzer, L., Artner, M.: Visualizing Decision Diagrams for Quantum Computing. In: Design, Automation and Test in Europe. pp. 768–773 (2021)
20. Wille, R., Burgholzer, L., Hillmich, S., Grurl, T., Ploier, A., Peham, T.: The Basis of Design Tools for Quantum Computing. In: Design Automation Conference. pp. 1367–1370. ACM (Jul 2022)
21. Zulehner, A., Hillmich, S., Wille, R.: How to Efficiently Handle Complex Values? Implementing Decision Diagrams for Quantum Computing. In: International Conference on Computer-Aided Design (ICCAD). pp. 1–7 (2019)