A Robust Approach to Detecting Non-equivalent Quantum Circuits Using Specially Designed Stimuli

Hsiao-Lun Liu, Yi-Ting Li, Yung-Chih Chen, and Chun-Yao Wang

ABSTRACT

As several compilation and optimization techniques have been proposed, equivalence checking for quantum circuits has become essential in design flows. The state-of-the-art to this problem observed that even small errors substantially affect the entire quantum system. As a result, it exploited random simulations to prove the non-equivalence of two quantum circuits. However, when errors occurred close to outputs, it was hard for the work to prove the non-equivalence of some non-equivalent quantum circuits under a limited number of simulations. In this work, we propose a novel simulation-based approach using a set of specially designed stimuli. The simulation runs of the proposed approach is linear rather than exponential to the number of quantum bits of a circuit. According to the experimental results, the success rate of our approach is 100% (100%) under a simulation run (execution time) constraint for a set of benchmarks, while that of the state-of-the-art is only 69% (74%) on average. Our approach also achieves a speedup of 26 on average.

ACM Reference Format:
Hsiao-Lun Liu, Yi-Ting Li, Yung-Chih Chen, and Chun-Yao Wang. 2023. A Robust Approach to Detecting Non-equivalent Quantum Circuits Using Specially Designed Stimuli. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567935

1 INTRODUCTION

Due to the potential of quantum computers, the research of quantum computing has become popular in both academia and industry. With the contributions from many researchers, the design flow of quantum circuits including preprocessing [10][11], compilation [6][13], optimization [3][4][9], and verification [1][2][8] has been studied. Furthermore, from the fact that quantum circuits are not easily accessible currently, simulation methods with classic computers have been proposed [5][14]. Also, some open-source toolkits such as IBM's Qiskit [16] and Microsoft's QDK [17] are available.

Since quantum circuits are generated by quantum compilation from conventional logic circuits [6][13], we must ensure that the functionalities from different quantum compilation techniques are equivalent. Moreover, we also need to guarantee that the functionality of a quantum circuit remains the same after optimization. Thus, the equivalence checking of quantum circuits is an important and necessary task in design flows.

Various methods about equivalence checking of quantum circuits were proposed recently [1][8]. However, these methods completely computed the functionalities of two quantum circuits, which was time-consuming. Therefore, a certain technique about accelerating the process of checking *non-equivalence* of two quantum circuits was proposed in the state-of-the-art [2]. The main idea of the state-of-the-art [2] is based on the observation that a small error frequently leads to huge effect on the entire quantum system. Because of the reversibility of quantum circuits, unlike the fault-effect in logic circuits that may be masked during propagation, the fault-effect in quantum circuits can be propagated out easily. Therefore, the state-of-the-art [2] exploited random simulations to detect the non-equivalence of two quantum circuits. However, the effectiveness of this approach strongly depends on the stimuli that check only one of 2^n basis states at a time, where n is the number of quantum bits (*qubits*) in the quantum circuit. In some cases, it may fail to detect the non-equivalence of two non-equivalent quantum circuits under a limited amount of simulations.

In this work, we propose a robust approach using a set of specially designed stimuli instead of random stimuli to detect the non-equivalence of two non-equivalent quantum circuits effectively and efficiently. Our approach simulates at most $(n + 1)$ patterns rather than 2^n patterns, where n is the number of qubits of the quantum circuit. To demonstrate the effectiveness of using such set of special stimuli, we create erroneous benchmarks by injecting errors into the circuits in RevLib [12]. The error types include randomly removing, altering, and misplacing one quantum gate. According to the experimental results, the success rate of our approach is 100% (100%), while that of the state-of-the-art is only 69% (74%) under a simulation run (execution time) constraint. These results are obtained by performing experiments with 100 different random seeds on average.

The main contributions of this work are as follows:

- We present a novel concept of simulation, which uses a set of specially designed stimuli rather than random stimuli.
- We propose an innovative approach, which is more robust for detecting non-equivalence of two non-equivalent quantum circuits than the state-of-the-art in terms of success rate.

The remainder of this paper is organized as follows. Section II introduces backgrounds. Section III presents the proposed approach. Section IV shows the experimental results. Finally, we conclude this paper in Section V.

2 PRELIMINARIES

In this section, we introduce the necessary backgrounds of quantum computing [7] and the advanced data structure used in quantum computing, i.e., decision diagrams [15]. Moreover, we introduce an equivalence checking model of quantum circuits.

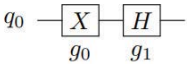
This work is supported in part by the National Science and Technology Council of Taiwan under MOST 109-2221-E-007-082-MY2, MOST 109-2221-E-155-047-MY2, MOST 110-2224-E-007-007, MOST 111-2218-E-007-010, MOST 111-2221-E-007-121, and MOST 111-2221-E-011-137-MY3.
H.-L. Liu, Y.-T. Li, and C.-Y. Wang are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300044, R.O.C. Y.-C. Chen is with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan 106335, R.O.C.

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567935

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad X = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

(a)



(b)

Figure 1: (a) The unitary matrices describing the Hadamard gate (H) and the swap gate (X). (b) A quantum circuit with a single qubit and two gates.

2.1 Quantum Computing

Quantum computing relies on qubits to represent the state of computation. Unlike logic bits used in classical computing, a qubit is not limited to either basis state $|0\rangle$ or $|1\rangle$, but both of them. This phenomenon is called superposition. Thus, a qubit is described as $\alpha_0|0\rangle + \alpha_1|1\rangle$, where $\alpha_0, \alpha_1 \in \mathbb{C}$. $|\alpha_0^2|$ and $|\alpha_1^2|$ represent the probabilities that this qubit will collapse into $|0\rangle$ and $|1\rangle$ after measurement, respectively. As a result, each qubit must satisfy the identity of $|\alpha_0^2| + |\alpha_1^2| = 1$. For a quantum system with n qubits, its state is described as $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle$, and it satisfies $\sum_{x\in\{0,1\}^n} |\alpha_x^2| = 1$. Typically, a state of an n-qubit quantum system is represented by a column vector $V \in \mathbb{C}^{2^n\times 1}$ and denoted as $[\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}]^T$.

Quantum operations are represented by quantum gates in quantum circuits and are able to manipulate single or multiple qubits, which changes the state of a quantum system. Each quantum operation in an n-qubit quantum circuit is represented by a unitary matrix $M \in \mathbb{C}^{2^n\times 2^n}$ for computation. A unitary matrix M is defined as $M^{-1} = M^*$, meaning that a unitary matrix's inverse is identical to its conjugate transpose. Fig. 1(a) shows two common quantum operations manipulating a single qubit. A quantum circuit G is represented by a sequence of quantum operations, i.e., $g_0 g_1 \ldots g_t$, and we can consider quantum computing to be a series of matrix-vector multiplications. To satisfy the form of matrix-vector multiplications, the unitary matrix describing G is $M_t \ldots M_1 M_0$, where M_i is the unitary matrix describing g_i, $i = 0 \sim t$. For example, Fig. 1(b) shows a simple quantum circuit $G = g_0 g_1 = M_1 M_0 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$. Typically, a pattern V simulating on a quantum circuit is formulated as $G \cdot V = M_t \ldots M_0 \cdot V$.

2.2 Decision Diagrams

The decision diagram is an advanced data structure to represent a quantum state or a quantum operation. By sharing data in vectors and matrices, this compact representation can extremely reduce the complexity in computation and memory usage.

A quantum state $V \in \mathbb{C}^{2^n\times 1}$ is decomposed into two sub-vectors $V_0, V_1 \in \mathbb{C}^{2^{n-1}\times 1}$, where $V = [V_0\ V_1]^T$, and this process is repeated until each sub-vector remains only one element. For each decomposition, a node is created in the decision diagram with two children representing the resultant sub-vectors. When some sub-vectors are identical, they share the same node. Each complex value in the represented quantum state is distributed to the edges of the decision diagram as the edge weights. We can retrieve a complex value by multiplying all the weights of edges in the corresponding path. Fig. 2 shows the decision diagram representing a quantum state $\left[\frac{1}{2}\ 0\ \frac{1}{2}\ 0\ 0\ 0\ \frac{1}{2}\ \frac{1}{2}\right]^T$. To maintain the canonical representation, we normalize weights by dividing each value by the first non-zero

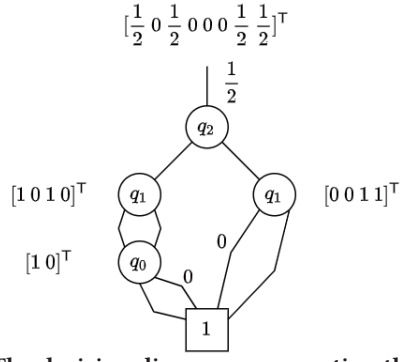


Figure 2: The decision diagram representing the quantum state of $\left[\frac{1}{2}\ 0\ \frac{1}{2}\ 0\ 0\ 0\ \frac{1}{2}\ \frac{1}{2}\right]^T$.

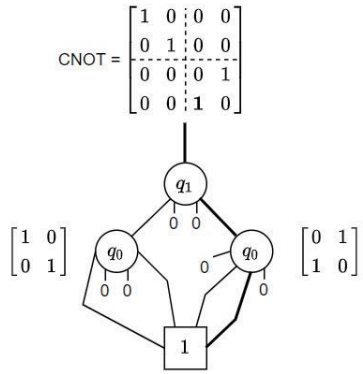


Figure 3: The decision diagram representing the CNOT gate.

value from the left, i.e., $\frac{1}{2}$ in Fig. 2. To make the decision diagram concise, the default weight of each edge is 1.

The way to create a decision diagram for a quantum operation is similar to that of a quantum state. A quantum operation $M \in \mathbb{C}^{2^n\times 2^n}$ is recursively decomposed into four sub-matrices $M_{00}, M_{01}, M_{10}, M_{11} \in \mathbb{C}^{2^{n-1}\times 2^{n-1}}$, where $M = \begin{bmatrix}M_{00} & M_{01}\\ M_{10} & M_{11}\end{bmatrix}$. Fig. 3 shows the decision diagram representing a controlled not gate (CNOT), which is represented by a unitary matrix $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$. When all the elements in a sub-matrix are 0s, we use the *0-sub* node marked as 0 in Fig. 3 to represent such a sub-matrix. Also, we can retrieve the bold complex value in the matrix by multiplying all the weights on the edges of the bold path.

After having the decision diagrams of quantum states and quantum operations, we can perform a matrix-vector multiplication via decision diagrams. The more details can be seen in [14] and [15].

2.3 Equivalence Checking Model of Quantum Circuits

In this work, we use the equivalence checking model proposed in [1] to examine if two quantum circuits G and R are equivalent or not. Let $G = M_t \ldots M_1 M_0$ consisting of $g_0 g_1 \ldots g_t$ and $R = N_{t'} \ldots N_1 N_0$ consisting of $r_0 r_1 \ldots r_{t'}$, where M_i and N_j are unitary matrices describing g_i and r_j, respectively, $i = 0 \sim t$, $j = 0 \sim t'$. We say that G and R are functionally equivalent if and only if $G = M_t \ldots M_1 M_0 = R = N_{t'} \ldots N_1 N_0$. Since every unitary matrix is inherently reversible,

it means that $G \cdot R^{-1} = I$, where I is the identity matrix. As a result, we can formulate the equivalence checking model of two quantum circuits G and R as EQ(1),

$$\begin{aligned} G \cdot R^{-1} &= (M_t \dots M_0) \cdot (N_{t'} \dots N_0)^{-1} \\ &= (M_t \dots M_0) \cdot (N_0^{-1} \dots N_{t'}^{-1}) \\ &= I \end{aligned} \tag{1}$$

and then check if EQ(1) holds or not. If EQ(1) holds for G and R, G and R are equivalent; otherwise, they are non-equivalent.

In the state-of-the-art [2], they exploited EQ(2) to detect the non-equivalence of two quantum circuits G and R, where V is a stimulus for simulation.

$$G \cdot V = (M_t \dots M_0) \cdot V \neq (N_{t'} \dots N_0) \cdot V = R \cdot V \tag{2}$$

That is, when $G \cdot V \neq R \cdot V$, $G \neq R$. However, when $G \cdot V = R \cdot V$, they conclude nothing, and need to try other stimuli.

In this work, we regard $G \cdot R^{-1}$ as a quantum system, and exploit EQ(3) to detect the non-equivalence of two quantum circuits G and R.

$$G \cdot R^{-1} \cdot V = (M_t \dots M_0) \cdot (N_0^{-1} \dots N_{t'}^{-1}) \cdot V \neq V \tag{3}$$

When EQ(3) holds, $G \cdot R^{-1} \neq I$; i.e., G and R are non-equivalent. Otherwise, it only indicates that $G \cdot R^{-1}$ has a high probability to be an identity matrix I.

3 THE PROPOSED APPROACH

In this section, we present the proposed approach, which uses a set of specially designed stimuli instead of randomly generated stimuli for detecting the non-equivalence of two quantum circuits.

3.1 Detecting Non-equivalent Quantum Circuits

In the traditional simulation for quantum circuits, a selected stimulus is limited to be one of 2^n basis states, where n is the number of qubits in the quantum circuit. Each basis state has *only one element of 1*, and *the others are all 0s*. As a consequence, it can only check one out of 2^n elements at a time in each row of the unitary matrices representing quantum circuits G and R. In other words, when G and R are slightly different, a large amount of simulations may be required. Thus, in this work, we propose to use a set of specially designed stimuli to detect the difference between G and R efficiently and effectively. Each designed stimulus is *deliberately* not in the form of a basis state. Hence, the output of the corresponding simulation is meaningless. However, we can exploit these designed stimuli to detect the non-equivalence of two non-equivalent quantum circuits G and R efficiently and effectively. This is the proposed *novel idea* about quantum simulation using *classic computers*. It is worth mentioning that the number of these specially designed stimuli grows linearly rather than exponentially with respect to the number of qubits n in the quantum circuit. Specifically, only $O(n)$ simulation runs are required in our approach.

Let T denote $G \cdot R^{-1}$, and the original objective of equivalence checking is to prove that $T = G \cdot R^{-1}$ is an identity matrix I. However, computing T completely is very time-consuming. Thus, we turn to check if T is an identity matrix via simulation. Note that for better readability, we use $T \cdot V_i$ to represent a simulation using a stimulus V_i on T. However, it is actually conducted as EQ(4),

$$(M_t \dots M_0) \cdot (N_0^{-1} \dots N_{t'}^{-1}) \cdot V_i \tag{4}$$

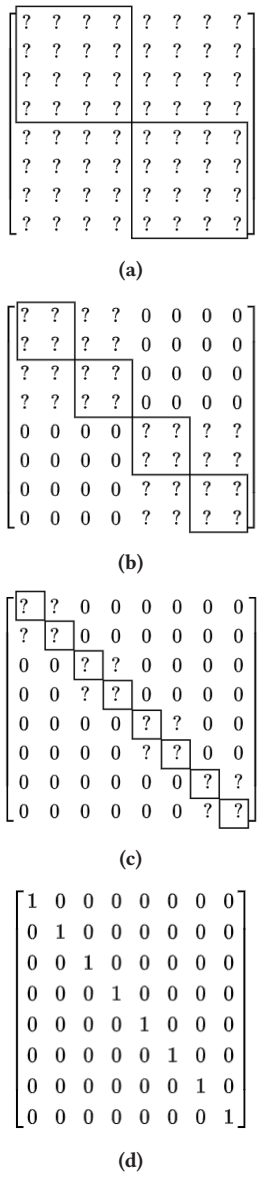


Figure 4: (a) The original T describing a 3-qubit quantum system. (b) The resultant T after the first run check. (c) The resultant T after the second run check. (d) The resultant T after passing the binary checking.

i.e., a series of multiplications of a matrix and a vector from right to left.

In the following paragraphs, we will introduce the set of specially designed stimuli, and explain why they work. First, we use a stimulus with *all the elements of 1s* for computation. To meet the identity of $\sum_{x=0}^{2^n-1} |\alpha_x^2| = 1$, we multiply all the elements by $\sqrt{\frac{1}{2^n}}$, and this stimulus is denoted as V_{init}. If $T \cdot V_{init} \neq V_{init}$, $T \neq I$, i.e., we detect the non-equivalence of G and R. Otherwise, we obtain $\sum_{j=0}^{2^n-1} x_{ij} = 1$ for $i = 0 \sim (2^n - 1)$, where x_{ij} is the complex value at the i^{th} row, and the j^{th} column in T. Namely, for each row in T,

the summation of all its complex values is 1. Based on our preliminary experiments, when a non-equivalent case passes this initial simulation, i.e., $T \cdot V_{init} = V_{init}$, T has a high probability of being structurally similar to the identity matrix, and the matrix T *may* satisfy EQ(5) and EQ(6),

$$\sum_{j=0}^{2^n-1} x_{ij} = 1,\ x_{ij} \in \{0,1\},\ for\ each\ row\ i \tag{5}$$

$$\sum_{i=0}^{2^n-1} x_{ij} = 1,\ x_{ij} \in \{0,1\},\ for\ each\ column\ j \tag{6}$$

which means that there is only an element of 1 in each row and each column. However, it is still possible that some elements of 1s are located at wrong positions, i.e., $x_{ij} = 1$, but $i \neq j$. Thus, based on this observation, we propose a set of specially designed stimuli to examine these cases, and the number of designed stimuli grows only linearly instead of exponentially to the number of qubits in the circuits.

Condition 1 states a necessary condition of an identity matrix I, where o_{ij} represents the element of 1 in I.

Condition 1 : *If I is an identity matrix, o_{ij} in the first half rows must stay in the first half columns, and o_{ij} in the second half rows must stay in the second half columns.*

We propose a method called *binary checking* to check if a matrix T meets Condition 1. We use an example to demonstrate the concept of binary checking in Fig. 4. Note that we assume T in Fig. 4(a) *satisfies* EQ(5) and EQ(6), and T describes a 3-qubit quantum system. Originally, the elements in the entire matrix T are undetermined, which are represented in question marks. To satisfy Condition 1, o_{ij} must stay in the framed regions in Fig. 4(a). Therefore, in the first run, we use the stimulus V_1, $V_1 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^\top$ for simulation. When $T \cdot V_1 = V_1$, the undetermined regions of T shrink into two sub-matrices T_{00} and T_{11} as shown in Fig. 4(b), where $T = \begin{bmatrix} T_{00} & T_{01} \\ T_{10} & T_{11} \end{bmatrix}$. In the second run, we use the stimulus V_2, $V_2 = [1\ 1\ 0\ 0\ 1\ 1\ 0\ 0]^\top$ for simulation. If T passes the second run check, i.e., $T \cdot V_2 = V_2$, the resultant T is as shown in Fig. 4(c). Similarly, in the last run, we use the stimulus V_3, $V_3 = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0]^\top$, to check the undetermined regions. When T passes all the simulation runs in the binary checking, T is an identity matrix; otherwise, T is not an identity matrix, and we detect the non-equivalence of two quantum circuits.

The total number of runs in the binary checking is $\log 2^n = n$, which is linear instead of exponential growth with respect to the number of qubits in a quantum circuit. Due to its regularity, stimuli used in the binary checking can be transformed into the corresponding decision diagrams easily. Also, to meet the identity of $\sum_{x=0}^{2^n-1} |\alpha_x^2| = 1$, we multiply all the elements of each stimulus used in the binary checking by $\sqrt{\frac{1}{2^{n-1}}}$.

These stimuli are designed based on our observation that when T passes the initial simulation, i.e., $T \cdot V_{init} = V_{init}$, T will satisfy EQ(5) and EQ(6) with a very high probability. Although this probability is not 100%, according to the experimental results, our approach with these designed stimuli can detect the non-equivalence of all the non-equivalent benchmarks. As for the equivalent cases, which will pass the binary checking, we can use other methods proposed in [1] and [8] to prove their equivalence, which is not the scope of this paper.

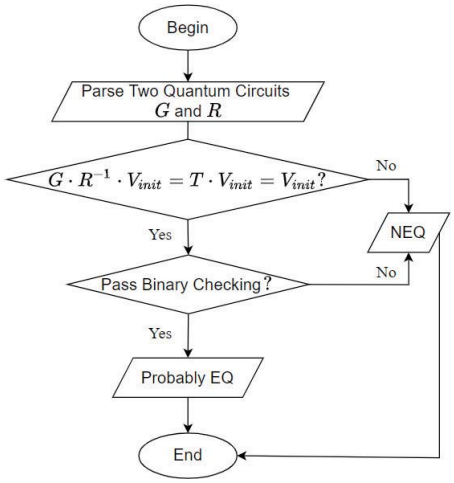


Figure 5: The overall flow of the proposed approach.

3.2 Overall Flow of the Proposed Approach

In this work, we propose a robust approach using a set of specially designed stimuli, which is novel and counter-intuitive. The overall flow of the proposed approach is shown in Fig. 5. First, we run an initial simulation with the stimulus $V_{init} = [\alpha_0\ \alpha_1\ \ldots\ \alpha_{2^n-1}]^\top$, where $\alpha_i = \sqrt{\frac{1}{2^n}}$ for $i = 0 \sim (2^n - 1)$. When it passes the initial simulation, we perform the binary checking algorithm. Otherwise, we detect the non-equivalence of two quantum circuits G and R. During the process of the binary checking, when a simulation conducted in the binary checking detects the non-equivalence (NEQ) of two quantum circuits G and R, the process is terminated. Passing the binary checking indicates that two quantum circuits G and R are probably equivalent (Probably EQ), and then the methods proposed in [1] and [8] can be used to prove their equivalence.

4 EXPERIMENTAL RESULTS

We implemented the proposed approach in C++ language on top of the decision diagram package released by [15]. The implementation of the state-of-the-art method was from the source code released by [2]. The experiments were conducted on a Linux platform of Ubuntu 20.04.02 LTS with an Intel i9-11900 CPU (2.5 GHz and 64 GB RAM).

To examine the effectiveness of our approach for detecting the non-equivalence of two non-equivalent quantum circuits, we create erroneous benchmarks by injecting errors into original benchmarks provided in RevLib [12]. The error types include randomly removing, altering, and misplacing one quantum gate. For instance, we take an original benchmark as a golden quantum circuit G, and we randomly remove one gate from G to form an erroneous quantum circuit R. To compare the effectiveness and the efficiency of our approach against the state-of-the-art [2], we set the simulation run (execution time) constraint to be 100 times (1 second).

The comparison of experimental results for the erroneous benchmarks is summarized in TABLEs 1 ∼ 4. Since the state-of-the-art [2] exploited random simulations to detect the non-equivalence of two quantum circuits G and R, the experimental results of the

Table 1: The comparison of experimental results for the erroneous benchmarks created by removing one gate.

Benchmark	Qubit	$\|gate\|$	The state-of-the-art [2]				Ours			
			#Sims	CPU (s)	Success rate (%)		#Sims	CPU (s)	Success rate (%)	
					Sims	Time			Sims	Time
4_49	4	16	1.93	0.01	100	100	3	0.01	100	100
alu4	**22**	**1063**	**550.45**	**3.25**	**15**	**28**	**5**	**0.58**	**100**	**100**
dist	13	185	1.00	0.01	100	100	2	0.01	100	100
dk17	21	49	145.17	0.03	44	100	4	0.01	100	100
ex1010	20	2611	408.90	5.25	20	15	4	0.03	100	100
example2	16	157	7.34	0.01	100	100	4	0.01	100	100
f51m	22	663	1.00	0.01	100	100	3	0.02	100	100
hwb9	9	1959	2.18	0.02	100	100	2	0.07	100	100
misex3	28	1752	524.48	6.55	19	18	6	0.19	100	100
rd84	15	28	2.16	0.01	100	100	4	0.01	100	100
ryy6	17	44	33.04	0.01	94	100	2	0.01	100	100
sym10	11	194	1.00	0.01	100	100	2	0.01	100	100
tial	**22**	**1041**	**1990.82**	**11.85**	**4**	**7**	**4**	**0.06**	**100**	**100**
Average			282.27	2.08	69	74	3.46	0.08	100	100
Ratio				26.0				1		

Table 2: The comparison of experimental results for the erroneous benchmarks created by altering one gate.

Benchmark	Qubit	$\|gate\|$	The state-of-the-art [2]				Ours			
			#Sims	CPU (s)	Success rate (%)		#Sims	CPU (s)	Success rate (%)	
					Sims	Time			Sims	Time
4_49	4	16	1.00	0.01	100	100	2	0.01	100	100
alu4	22	1063	477.19	2.76	16	32	6	0.35	100	100
dist	13	185	8.10	0.01	100	100	2	0.01	100	100
dk17	21	49	71.86	0.02	79	100	2	0.01	100	100
ex1010	20	2611	266.67	3.36	32	25	3	0.03	100	100
example2	16	157	66.74	0.04	83	100	3	0.01	100	100
f51m	22	663	1846.52	6.75	6	14	4	0.14	100	100
hwb9	9	1959	1.00	0.01	100	100	2	0.08	100	100
misex3	28	1752	567.71	6.93	17	16	4	0.12	100	100
rd84	15	28	1.79	0.01	100	100	4	0.01	100	100
ryy6	17	44	975.15	0.16	11	100	3	0.01	100	100
sym10	11	194	52.41	0.03	85	100	2	0.01	100	100
tial	22	1041	262.89	1.53	35	50	3	0.07	100	100
Average			353.09	1.66	59	72	3.08	0.07	100	100
Ratio				23.7				1		

Table 3: The comparison of experimental results for the erroneous benchmarks created by misplacing one gate.

Benchmark	Qubit	$\|gate\|$	The state-of-the-art [2]				Ours			
			#Sims	CPU (s)	Success rate (%)		#Sims	CPU (s)	Success rate (%)	
					Sims	Time			Sims	Time
4_49	4	16	1.34	0.01	100	100	2	0.01	100	100
alu4	22	1063	3.47	0.03	100	100	2	0.07	100	100
dist	13	185	2.38	0.01	100	100	2	0.01	100	100
dk17	21	49	3.47	0.01	100	100	2	0.01	100	100
ex1010	20	2611	118.03	1.52	53	42	4	0.12	100	100
example2	16	157	138.03	0.08	51	100	3	0.01	100	100
f51m	22	663	4.42	0.02	100	100	2	0.03	100	100
hwb9	9	1959	8.83	0.07	100	100	2	0.08	100	100
misex3	28	1752	254.38	3.19	32	24	6	0.45	100	100
rd84	15	28	1.29	0.01	100	100	7	0.01	100	100
ryy6	17	44	2.19	0.01	100	100	2	0.01	100	100
sym10	11	194	253.82	0.15	31	100	2	0.01	100	100
tial	22	1041	119.65	0.72	59	73	5	0.38	100	100
Average			70.1	0.45	79	88	3.62	0.02	100	100
Ratio				22.5				1		

state-of-the-art are the averaged results obtained by performing experiments for 100 times with random seeds. TABLE 1 shows the experimental results for the erroneous benchmarks created by randomly removing one quantum gate. The column $|gate|$ represents the number of quantum gates in the corresponding quantum circuit G. The column *#Sims* (*CPU*) represents the number of simulation runs (execution time with one CPU core) until detecting the non-equivalence of two quantum circuits G and R. The column *Sims* (*Time*) under *Success rate* represents the percentage in the

Table 4: The comparison of experimental results for the erroneous benchmarks created by injecting three types of errors.

Benchmark	Qubit	$\|gate\|$	The state-of-the-art [2]				Ours			
			#Sims	CPU (s)	Success rate (%)		#Sims	CPU (s)	Success rate (%)	
					Sims	Time			Sims	Time
4_49	4	16	1.28	0.01	100	100	2	0.01	100	100
alu4	22	1063	1.00	0.02	100	100	2	0.20	100	100
dist	13	185	2.59	0.01	100	100	2	0.01	100	100
dk17	21	49	1.00	0.01	100	100	2	0.01	100	100
ex1010	20	2611	1.84	0.04	100	100	2	0.06	100	100
example2	16	157	8.06	0.01	100	100	3	0.01	100	100
f51m	22	663	1.00	0.01	100	100	3	0.10	100	100
hwb9	9	1959	1.19	0.01	100	100	2	0.07	100	100
misex3	28	1752	265.88	3.28	27	21	2	0.16	100	100
rd84	15	28	1.00	0.01	100	100	3	0.01	100	100
ryy6	17	44	18.87	0.01	98	100	2	0.01	100	100
sym10	11	194	1.00	0.01	100	100	2	0.01	100	100
tial	22	1041	30.43	0.19	95	99	4	0.32	100	100
Average			25.78	0.27	94	94	2.38	0.08	100	100
Ratio				3.4				1		

corresponding method meeting the simulation run (execution time) constraint.

For example, the state-of-the-art [2] spent 550.45 runs of simulations and 3.25 seconds to detect the non-equivalence of G and R of *alu4* benchmark on average. Only 15 (28) experiments met the simulation run (execution time) constraint in 100 random experiments. However, for the proposed approach, the success rates are both 100%, and the averaged simulation runs and CPU time are only 5 and 0.58 seconds, respectively. For other benchmarks such as *dk17, ex1010, misex3*, and *tial*, the averaged simulation runs in the state-of-the-art are also more than 100 times. Especially, for *tial* benchmark, the success rate of the state-of-the-art is only 4% (7%) under the simulation run (execution time) constraint. According to TABLE 1, the averaged success rate of our approach is 100% (100%), while that of the state-of-the-art is only 69% (74%) under the simulation run (execution time) constraint. Our speedup is 26.0 on average.

TABLEs 2 and 3 show the experimental results for the erroneous benchmarks created by randomly altering or misplacing one quantum gate, respectively. According to TABLEs 2 and 3, the averaged success rates of our approach are both 100% (100%), while those of the state-of-the-art are 59% (72%) and 79% (88%) under the simulation run (execution time) constraint. TABLE 4 shows the experimental results for the erroneous benchmarks created by injecting all three types of errors including randomly removing, altering, and misplacing one quantum gate. According to TABLE 4, the averaged success rate of our approach is 100% (100%), and that of the state-of-the-art is 94% (94%) under the simulation run (execution time) constraint.

From these experimental results, we realized that when fewer errors are injected into the benchmarks, the success rate of the state-of-the-art is lower, which means that random simulation approach is hard to detect the non-equivalence of two more similar circuits. However, in the proposed novel approach, the success rates are all 100%, and the averaged CPU time is less than 0.1 seconds for all the experiments, which indicates that our approach is more efficient and more robust than the state-of-the-art.

5 CONCLUSION

In this work, we propose a robust approach to detect non-equivalent quantum circuits. Our approach uses a set of specially designed stimuli instead of random stimuli, which is out of the traditional frame in quantum circuit simulation. Moreover, our approach is more efficient and more robust for detecting the non-equivalence of two non-equivalent quantum circuits than the state-of-the-art in terms of success rate in different sets of erroneous benchmarks.

REFERENCES

[1] L. Burgholzer and R. Wille, "Improved DD-based Equivalence Checking of Quantum Circuits," in *Proc. of ASP-DAC*, 2020, pp. 1-6.
[2] L. Burgholzer and R. Wille, "The Power of Simulation for Equivalence Checking in Quantum Computing," in *Proc. of DAC*, 2020, pp. 1-6.
[3] J. Golden, A. Bartschi, D. O'Malley, and S. Eidenbenz, "Threshold-Based Quantum Optimization," in *Proc. of QCE*, 2021, pp. 1-11.
[4] 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.
[5] S. Hillmich, R. Kueng, I. L. Markov, and R. Wille, "As Accurate as Needed, as Efficient as Possible: Approximations in DD-based Quantum Circuit Simulation," in *Proc. of DATE*, 2021, pp. 1-6.
[6] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. De. Micheli, "The role of multiplicative complexity in compiling low T-count oracle circuits," in *Proc. of ICCAD*, 2019, pp. 1-8.
[7] M. A. Nielsen and I. L. Chuang, "Quantum Computation and Quantum Information: 10th Anniversary Edition," Cambridge University Press, 2010.
[8] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, 2016, pp. 86–99.
[9] R. Shaydulin, I. Safro, and J. Larson, "Multistart Methods for Quantum Approximate Optimization," *Proc. of HPEC*, 2019, pp. 1–8.
[10] E. Testa, M. Soeken, L. Amaru and G. De. Micheli, "Reducing the multiplicative complexity in logic networks for cryptography and security applications," in *Proc. of DAC*, 2019, pp. 1-6.
[11] E. Testa, M. Soeken, H. Riener, L. Amaru and G. De. Micheli, "A logic synthesis toolbox for reducing the multiplicative complexity in logic networks," in *Proc. of DATE*, 2020, pp. 568-573.
[12] 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.
[13] R. Wille, S. Hillmich, and L. Burgholzer, "Efficient and Correct Compilation of Quantum Circuits," in *Proc. of ISCAS*, 2020, pp. 1-5.
[14] A. Zulehner and R. Wille, "Advanced Simulation of Quantum Computations," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.38, no. 5, 2019, pp. 848-859.
[15] A. Zulehner, S. Hillmich, and R. Wille, "How to Efficiently Handle Complex Values? Implementing Decision Diagrams for Quantum Computing," in *Proc. of ICCAD*, 2019, pp. 1-7.
[16] "Qiskit," IBM, [Online]. Available: https://qiskit.org/.
[17] "QDK," Microsoft, [Online]. Available: https://github.com/microsoft/qdk-python.