

# A Compact and Efficient SAT Encoding for Quantum Circuits

Robert Wille\*<sup>†</sup>

Nils Przigoda\*

Rolf Drechsler\*<sup>†</sup>

\*Institute of Computer Science, University of Bremen, 28359 Bremen, Germany

<sup>†</sup>Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany

{rville,przigoda,drechsle}@informatik.uni-bremen.de

**Abstract**—Promising applications of quantum computation motivated the consideration of corresponding design methods for this emerging technology. Here, researchers are faced with the problem that signals in quantum circuits may (theoretically) assume an infinite number of states. As a consequence, design approaches based on Boolean satisfiability (SAT) were subject to restrictions so far. In this work, we propose a compact and efficient SAT encoding for quantum circuits that loses these restrictions. For this purpose, a structural analysis is introduced which determines an upper bound on possible quantum states. The applicability of the encoding is exemplarily demonstrated by a SAT-based equivalence checker.

## I. INTRODUCTION

Quantum computation [1] established itself as a promising emerging technology. Important problems including factorization [2], [3] (and its application in cryptography), database search [4], or graph and algebraic problems [5] can be addressed significantly faster when exploiting the underlying quantum mechanical phenomena in comparison to conventional computing paradigms. Consequently, the development of corresponding design methods received significant attention in the last years. Besides a quite thorough consideration of synthesis questions (see e.g. [6], [7] for overviews), recently, also approaches for verification or test became a focal point of interest [8], [9], [10], [11], [12], [13], [14].

Here, researchers are particularly struggling with the complexity of quantum circuits. Theoretically, qubits<sup>1</sup> may assume an infinite number of values, i.e. not only the Boolean values 0 and 1 known from conventional computations need to be considered, but also quantum states representing the *superposition* of them. To address this problem, approaches have been introduced which (1) rely on simulation [15] and often require an exhaustive consideration of the input vectors or (2) make use of decision diagram-like data-structures [16], [9], [10], [12], [14] that suffer from memory explosion.

As an alternative, methods based on *Boolean satisfiability* (SAT) are considered in this work. Here, the corresponding verification or test problem is encoded as a SAT instance which, afterwards, is addressed by corresponding solving engines (e.g. [17]). Due to the impressive performance of today's SAT solvers, Boolean satisfiability is an established core technology in the verification and the test of conventional circuits. Applications can be found e.g. in equivalence checking [18], [19], property checking [20], [21], or automatic test pattern generation [22], [23].

However, considering SAT for quantum circuits still requires an efficient handling of the (theoretically) infinite number of possible quantum values qubits may assume. An existing

solution presented in [12] solved this issue by rigorously restricting the possible number of quantum states to 4. While this enabled e.g. SAT-based equivalence checking for selected quantum circuits, obviously, the assumed restrictions significantly reduce the applicability of this encoding.

In this work, we propose a compact and efficient SAT encoding for quantum circuits that loses these restrictions. We exploit thereby an observation that quantum circuits are inherently restricted by themselves in the number of values their respective qubits may assume. Based on this information, a corresponding SAT encoding can be derived. To this end, a two-stage approach is presented: First, a structural analysis is conducted that precisely derives the number of possible quantum states to be considered. Afterwards, the extracted information is utilized to create a proper SAT encoding.

The proposed encoding can be applied to a variety of applications, e.g. SAT-based equivalence checking, SAT-based property checking, or SAT-based automatic test pattern generation of quantum circuits. To demonstrate the applicability and the efficiency of the proposed encoding, a SAT-based equivalence checker has been developed which exploits the contributions of this work. An experimental evaluation confirms that the SAT encoding is capable of handling various quantum circuits with different quantum states.

The remainder of this paper is structured as follows: The next section provides a brief overview of the background to this work, i.e. quantum computation and Boolean satisfiability are reviewed. Section III revisits existing SAT encodings for both, conventional and quantum circuits, and, by this, builds the motivation for the work at hand. The proposed SAT encoding is then described in detail in Section IV. Afterwards, the applicability of this encoding is discussed in Section V and experimentally evaluated by means of the equivalence checker application in Section VI. Finally, the paper is concluded in Section VII.

## II. BACKGROUND







In order to keep the paper self-contained, basics on quantum computation and quantum circuits as well as on Boolean satisfiability are briefly reviewed in this section.

### A. Quantum Logic

Quantum computation is an emerging technology that enables the computation of many relevant problems, e.g. factorization or data-base search, in less complexity than conventional computing paradigms [1]. Every quantum circuit works on qubits instead of bits. In contrast to Boolean logic, qubits do not only allow to represent Boolean 0's and Boolean 1's, but also the superposition of both.

<sup>1</sup>In quantum computation, a qubit is the equivalent to a bit.

TABLE I  
QUANTUM GATES

Hadamard-Gate 	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Pauli-X-Gate 	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y-Gate 	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Pauli-Z-Gate 	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
V-Gate 	$\frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$	$V^\dagger$ -Gate 	$\frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$

More formally, a qubit is a two level quantum system, described by a two dimensional complex Hilbert space. The two orthogonal quantum states  $|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  and  $|1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$  are used to represent the Boolean values 0 and 1. Any state of a qubit may be written as  $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ , where  $\alpha$  and  $\beta$  are complex numbers with  $|\alpha|^2 + |\beta|^2 = 1$ . The quantum state of a single qubit is denoted by the vector  $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ . The state of a quantum system with  $n > 1$  qubits is given by an element of the tensor product of the respective state spaces and can be represented as a normalized vector of length  $2^n$ , called the state vector.

Operations on  $n$ -qubits states are performed through multiplication of appropriate  $2^n \times 2^n$  unitary matrices. Thus, each quantum computation is inherently reversible but manipulates qubits rather than pure logic values. At the end of the computation, a qubit can be measured. Then, depending on the current state of the qubit, either a 0 (with probability of  $|\alpha|^2$ ) or a 1 (with probability of  $|\beta|^2$ ) returns. After the measurement, the state of the qubit is destroyed.

*Example 1:* Given an input state  $|x\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  and an operation H defined by the unitary matrix  $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ . Applying  $|x\rangle$  to H leads to a new quantum state  $|x'\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ , i.e. a state with  $\alpha = \beta = \frac{1}{\sqrt{2}}$ . Measuring this qubit would either lead to a Boolean 0 or a Boolean 1 with a probability of  $|\frac{1}{\sqrt{2}}|^2 = 0.5$  each. This computation represents one of the simplest quantum computers – a random number generator.

Quantum computations are usually represented by *quantum circuits*. Here, the respective qubits are denoted by solid *circuit lines*. Operations are represented by *quantum gates*. Table I lists common quantum gates together with the corresponding unitary matrices describing their operation. In order to perform operations on more than one qubit, *controlled quantum gates* are applied. These gates are composed of a *target line*  $|t\rangle$  and a control line  $|c\rangle$  and realize the unitary operation represented by the matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U & 0 \\ 0 & 0 & 0 & U \end{pmatrix},$$

where  $U$  denotes the operation applied to the target line. That is, if  $|c\rangle = |0\rangle$  all states remain unchanged and if  $|c\rangle = |1\rangle$  the operation  $U$  is applied to the target line  $|t\rangle$ . In all other cases, the vector  $(\alpha_{|c\rangle}\alpha_{|t\rangle}, \alpha_{|c\rangle}\beta_{|t\rangle}, \beta_{|c\rangle}\alpha_{|t\rangle}, \beta_{|c\rangle}\beta_{|t\rangle})$  is applied to  $M$ . This leads to an entangled quantum state [1].

*Example 2:* Fig. 1 shows a quantum circuit together with a possible assignment to its signals.

<sup>2</sup>This operation represents the well known Hadamard operation [1].

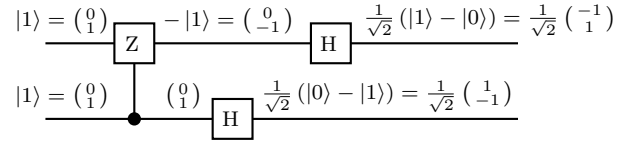


Fig. 1. Quantum circuit

In this work, a quantum circuit is formally denoted by the tuple  $C(G, S, I, O)$  whereby  $G$  denotes the gates,  $S$  denotes the signals,  $I$  denotes the primary inputs, and  $O$  denotes the primary outputs of the circuit. A gate  $g \in G$  of a quantum circuit is formally denoted by the tuple  $g(I_g, O_g, U_g)$  whereby  $I_g$  denotes the input to that gate,  $O_g$  denotes the output to that gate, and  $U_g$  denotes the unitary operation invoked at the gate.

### B. Boolean Satisfiability

The *Boolean Satisfiability* (SAT) problem is defined as follows: Let  $f$  be a Boolean function in *Conjunctive Normal Form* (CNF), i.e. a product-of-sum representation. Then, the SAT problem is to determine an assignment for the variables of  $f$  so that  $f$  evaluates to 1 or to prove that no such assignment exists.

The CNF consists of a conjunction of clauses. A clause is a disjunction of literals and each literal is a propositional variable or its negation.

*Example 3:* Let  $f = (x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2)(\bar{x}_2 + x_3)$ . Then,  $x_1 = 1, x_2 = 1$  and  $x_3 = 1$  is a satisfying assignment for  $f$ . The value of  $x_1$  ensure that the first clause becomes satisfied, the value of  $x_2$  ensures this for the second clause and  $x_3$  ensures this for the remaining clause.

SAT is one of the central  $\mathcal{NP}$ -complete problems. In fact, it was the first known  $\mathcal{NP}$ -complete problem that was proven by Cook in 1971 [24]. But, in the past efficient solving algorithms (so called *SAT solvers*) have been proposed (see e.g. [17]). Instead of simply traversing the complete space of assignments, intelligent decision heuristics, *conflict based learning*, and sophisticated engineering of the implication algorithm by *Boolean Constraint Propagation* (BCP) lead to an effective search procedure. Once it is proven that no solution exists, an instance is called *unsatisfiable* (UNSAT), otherwise *satisfiable* (SAT). Due to these efficient algorithms, problem instances consisting of hundreds of thousands of variables, millions of clauses, and tens of millions of literals can be handled.

### III. SAT ENCODINGS OF CIRCUITS

Due to the impressive performance of today's SAT solvers, Boolean satisfiability is an established core technology in the verification and the test of conventional circuits. Applications can be found e.g. in equivalence checking [18], [19], property checking [20], [21], or test pattern generation [22], [23].

In all these approaches, the considered circuit is encoded as a SAT instance as follows: For each signal  $s$  of the circuit, a corresponding SAT variable  $x_s$  representing this signal is introduced. Then, for each gate  $g$  of the circuit, a corresponding set of functional constraints is introduced. For this, the SAT variables representing the input-signals and the output signals of  $g$  are applied. Afterwards, this description is converted into a CNF.

*Example 4:* Consider the (conventional) circuit as shown in the top of Fig. 2(a). All signals  $a, b, c, d, e$  of this circuit are represented by SAT variables  $x_a, x_b, x_c, x_d, x_e$ . Then, the constraints as shown in the bottom of Fig. 2(a) encode the functionality of each gate to be converted into a CNF.

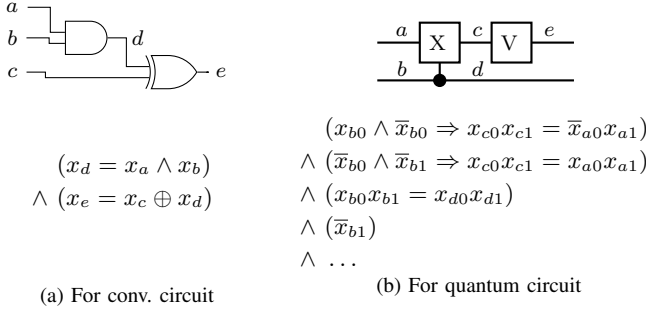


Fig. 2. Existing SAT encodings

Passing such an encoding to a SAT solver, only satisfiable assignments to SAT variables and, hence, only valid assignments to each circuit signal are determined. By adding further constraints to the SAT instance, certain scenarios of this circuit can be checked. For example, SAT solvers can be applied e.g. to generate counterexamples (i.e. assignments to all circuit signals showing that an unwanted behavior indeed may occur). If a SAT solver shows the unsatisfiability of an instance, it can be concluded that the formulated scenario can never occur. By this, verification and test problems can be efficiently solved. For more details on that, we refer e.g. to [18], [19], [20], [21], [22], [23].

In principle, this scheme can also be applied for quantum circuits. In fact, initial approaches doing equivalence checking using Boolean satisfiability have been introduced in [12]. However, while each signal  $s$  in a conventional circuit may assume either Boolean 0 or Boolean 1 (which can easily be represented by a single SAT variable  $x_s$ ), qubits in quantum circuits may assume any quantum state  $\alpha|0\rangle + \beta|1\rangle$  with  $|\alpha|^2 + |\beta|^2 = 1$ , i.e. an infinite number in the worst case. In [12], this problem has been addressed by rigorously restricting the possible number of quantum states and the use of a multiple-valued SAT encoding. More precisely, only a quantum library composed of Pauli-X gate, controlled Pauli-X gate, V-gate, and  $V^\dagger$ -gate was applied. Furthermore, the input to a quantum circuit as well as to each control line of a gate has been restricted to 0 and 1. This has the effect that the value of each qubit is restricted to one value of the set  $\{0, 1, V_0, V_1\}$ , i.e. a 4-valued logic with  $V_0 = \frac{1+i}{2} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$  and  $V_1 = \frac{1+i}{2} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$  was applied. Consequently, each signal  $s$  was no longer represented by a single SAT variable  $x_s$ , but by two SAT variables  $x_{s0}, x_{s1}$  that represent the respective values as follows:

$x_{s0}$	$x_{s1}$	
0	0	0
0	1	$V_0$
1	0	1
1	1	$V_1$

*Example 5:* Consider the restricted quantum circuit as shown in the top of Fig. 2(b). For each signal, two SAT variables are introduced. Then, constraints as partially shown in the bottom of Fig. 2(b) encode the functionality of each gate.

Such an encoding, in fact, enables the consideration of verification and test issues of quantum circuit by means of Boolean satisfiability (as e.g. done in [12] for equivalence checking). But obviously, the assumed restrictions significantly reduce the applicability of this encoding: As soon as another gate type or further quantum states may occur in the considered circuit, this 4-valued encoding is not applicable any longer.

At the same time, simply extending the encoding e.g. from a 4-valued encoding to an 8-valued encoding eventually will lead to the same restrictions as long as the total number of possible quantum states in the considered circuit is unknown.

#### IV. PROPOSED SAT ENCODING

In this work, we propose a SAT encoding that looses the restrictions from previous approaches. To this end, an observation is exploited that many quantum circuits are inherently restricted in the number of states their respective qubits can assume. A precise structural analysis can further reduce the number of possible quantum states to be considered. In doing so, compact and efficient SAT encodings for quantum circuits can be created. In the following, the observation as well as the approaches resulting from it are described in detail.

##### A. Observation

In general, qubits may assume an infinite number of states. Hence, the consideration of SAT technologies in the verification and the test of quantum circuits may seem unfeasible as an  $\infty$ -valued encoding is required in the worst case. However, actual quantum circuits are inherently restricted in the number of states their respective qubits can assume.

*Lemma 1:* Let  $|G|$  be the total number of gates a quantum circuit  $C$  is composed of. Furthermore, let  $v$  be the total number of quantum values which may be applied to the primary inputs  $I$  of  $G$ . Then, the total number of quantum states which may occur in  $C$  is restricted by  $2^{|G|} \cdot v$ .

*Proof:* Each gate may introduce new quantum states to the circuit depending on its possible input values. More precisely, if a gate  $g$  performing the operation  $U$  is fed by quantum values from the set  $Q$ , a new set  $Q'$  of quantum values with  $Q' = \{U \cdot |x\rangle : |x\rangle \in Q\}$  results. Hence, each gate may only double the number of quantum states occurring in a circuit. Considering the initial number  $v$  of quantum states, this leads to  $2^{|G|} \cdot v$ . ■

This observation can be used as an upper bound for a SAT encoding: Given a circuit to be considered, each signal  $s$  is represented by a  $(2^{|G|} \cdot v)$ -valued encoding, i.e. SAT variables  $x_{s0} \dots x_{s2^{|G|} \cdot v - 1}$  are applied that represent the respective values.

*Example 6:* Consider the quantum circuit shown in Fig. 3(a) composed of  $|G| = 3$  gates. Assuming Boolean inputs values only, i.e.  $v = 2$ , it can be deduced that never more than  $2^3 \cdot 2 = 16$  different quantum states may occur in this circuit. Hence, a 4-valued encoding for each signal is sufficient to encode this circuit as a SAT instance. If further (non-Boolean) inputs values are assumed, the encoding has to be extended accordingly.

##### B. Structural Analysis

The observation from above constitutes an upper bound for quantum states that may occur in a considered circuit. However, the resulting value can further be refined. In fact, each gate does not necessarily introduce a new quantum state to be considered, but may lead to a state that already occurred before.

*Example 7:* Consider again the quantum circuit shown in Fig. 3(a). If Boolean inputs are assumed, i.e.  $v = 2$ , the first two gates always produce either  $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$  (if the input is  $|0\rangle$ ) or  $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$  (if the input is  $|1\rangle$ ). That is, in contrast to the

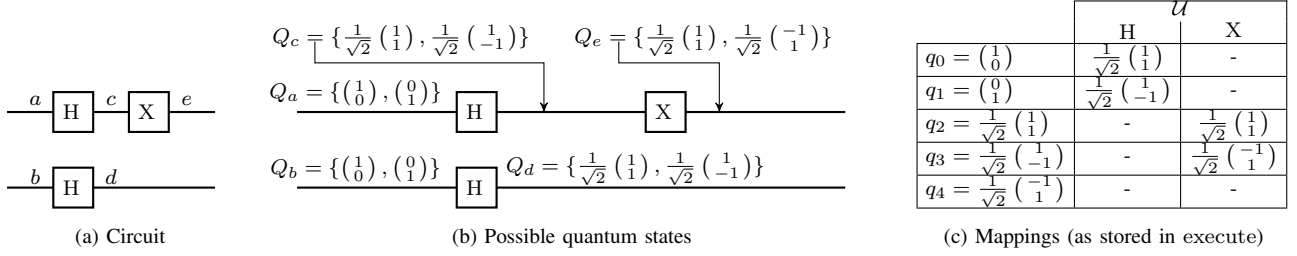


Fig. 3. Structural analysis

upper bound not  $2^2 \cdot 2 = 8$  different quantum states need to be considered for the first two gates, but just 4.

The precise value of total quantum states possible in a circuit  $C(G, S, I, O)$  can be determined by a structural analysis as follows:

- 1) For each signal  $s \in S$  of the considered quantum circuit, a set  $Q_s$  is introduced.  $Q_s$  stores all possible quantum states the signal  $s$  may assume. At the beginning, each  $Q_s$  is initialized empty.
- 2) All input signals  $s \in I$  of the circuits are traversed. The initial quantum states which may be applied to these circuit's inputs are defined by the designer. Accordingly, the respective sets  $Q_s$  are initialized with the corresponding values.
- 3) The circuit is traversed from the input to the output of the considered circuit. For each gate  $g(I_g, O_g, U_g) \in G$ , a local simulation is performed. By this, a precise mapping from all possible values of the gate inputs  $I_g$  to the corresponding values of the gate outputs  $O_g$  is obtained, i.e. for each possible input value  $q \in Q_s$  with  $s \in I_g$ , the corresponding value  $q' \in Q_{s'}$  with  $s' \in O_g$  is obtained considering the function  $U$  represented by the gate. These input-output mappings are stored in a function  $\text{execute} : \mathcal{U} \times Q \rightarrow Q$  with  $\mathcal{U}$  denoting the set of all unitary operations  $U$  in the circuit. This function gets a quantum operation  $U \in \mathcal{U}$  as well as a quantum state  $q \in Q$  and maps it to the corresponding quantum state  $q' \in Q$  so that  $q' = U \cdot q$ .

*Example 8:* The proposed structural analysis is applied to the quantum circuit shown in Fig. 3(a). First, for each signal  $s$ , a corresponding set  $Q_s$  is introduced. The sets  $Q_s$  representing the possible values of the primary inputs  $s \in I$  of the circuit are initialized with  $Q_s = \{|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\}$  (assuming Boolean inputs only); all remaining sets  $Q_s$  with  $s \in S \setminus I$  are initialized empty. Then, for each gate local simulations as described above are performed. This leads to the precise quantum states  $Q_s$  each signal  $s \in S$  in the circuit may assume. This is shown in Fig. 3(b). At the same time, all possible mappings are stored in the function  $\text{execute}$ . Fig. 3(c) shows all these mappings by means of a matrix. Columns and rows denote thereby the respective inputs (i.e. the considered operation  $U$  and the input quantum state  $q$ , respectively), while the entries denote the output (i.e. the resulting quantum state  $q'$ ).

By this, a finite set  $Q$  of quantum states is revealed, i.e. the signals in the considered quantum circuit may assume only quantum states from  $Q = \bigcup_{s \in S} Q_s$ . At the same time, all possible mappings are available through the function  $\text{execute}$ .

The cost of this structural analysis is thereby moderate: Since all simulations are performed locally, the run-time of structural analysis basically depends on a (linear) traversal of each gate and the required simulation time for all possible input values of each gate. In all our evaluations, this run-time was negligible. From the results of this structural analysis, the precise encoding can be derived as described next.

### C. Resulting SAT Encoding

To eventually encode a given quantum circuit as a SAT instance, a Boolean function  $f$  is created as follows:

For each signal  $s \in S$  variables  $x_{s0} \dots x_{s[\log_2 |Q|]-1}$  are introduced which symbolically represent all possible quantum states  $Q = \{q_0, \dots, q_{|Q|-1}\}$  the signal  $s$  may assume. More precisely, assignments to  $x_{s0} \dots x_{s[\log_2 |Q|]-1}$  are a symbolic representation of the corresponding quantum state, i.e.  $[x_{s0} \dots x_{s[\log_2 |Q|]-1}]_2 = [i]_{10}$  represents that signal  $s$  assumes the quantum state  $q_i$  with  $0 \leq i < |Q|$ . In order to avoid illegal assignments, constraints are introduced that block assignments for which no corresponding quantum state exists, i.e. constraints  $[x_{s0} \dots x_{s[\log_2 |Q|]-1}]_2 < [|Q|]_{10}$  are added to  $f$ .

Afterwards, for each gate of the circuit, a corresponding set of functional constraints is introduced. For this, the mapping stored in the function  $\text{execute}$  is utilized. More precisely, for each gate  $g(I_g, O_g, U_g)$ , a constraint

$$\bigwedge_{s \in I} \bigwedge_{q_i \in Q_s} [x_{s0} \dots x_{s[\log_2 |Q|]-1}]_2 = [i]_{10} \\ \Leftrightarrow [x_{s'0} \dots x_{s'[\log_2 |Q|]-1}]_2 = [j]_{10}$$

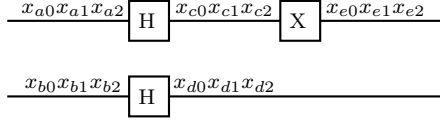
with  $q_j = \text{execute}(q_i, U_g)$  is added, where

- $x_{s0} \dots x_{s[\log_2 |Q|]-1}$  represents the quantum state of the gate inputs  $s \in I_g$ ,
- $x_{s'0} \dots x_{s'[\log_2 |Q|]-1}$  represents the quantum state of the gate outputs  $s' \in O_g$ , and
- $q_j$  represents the quantum state obtained from  $\text{execute}^4$ .

*Example 9:* Fig. 4 shows the resulting SAT encoding for the quantum circuit from Fig. 3(a). Since the total number  $|Q|$  of possible quantum states is 5,  $\lceil \log_2 5 \rceil = 3$  variables are needed to represent the assignment to a signal, i.e. each signal  $s$  is represented by variables  $x_{s0} x_{s1} x_{s2}$  as shown by the symbolic representation in Fig. 4. Since three variables allow to represent more quantum states than the signals can actually assume, constraints blocking illegal assignments are added as shown on the right-hand side of Fig. 4. Finally, functional constraints as partially shown in the bottom of Fig. 4 are added to the instance.

<sup>3</sup>Note that the description only covers the consideration of unary quantum gates. However, controlled quantum gates are handled analogously.

<sup>4</sup>Note that, also here, the encoding of unary gates only is described, while controlled gates can be encoded analogously.



Symbolic representation:

$$\begin{aligned} [x_{s0}x_{s1}x_{s2}]_2 = [0]_{10} &\hat{=} s = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ [x_{s0}x_{s1}x_{s2}]_2 = [1]_{10} &\hat{=} s = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ [x_{s0}x_{s1}x_{s2}]_2 = [2]_{10} &\hat{=} s = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ [x_{s0}x_{s1}x_{s2}]_2 = [3]_{10} &\hat{=} s = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\ [x_{s0}x_{s1}x_{s2}]_2 = [4]_{10} &\hat{=} s = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \end{aligned}$$

Blocking constraints:

$$\begin{aligned} [x_{a0}x_{a1}x_{a2}]_2 < [5]_{10} \\ [x_{b0}x_{b1}x_{b2}]_2 < [5]_{10} \\ [x_{c0}x_{c1}x_{c2}]_2 < [5]_{10} \\ [x_{d0}x_{d1}x_{d2}]_2 < [5]_{10} \\ [x_{e0}x_{e1}x_{e2}]_2 < [5]_{10} \end{aligned}$$

Functional constraints:

$$\begin{aligned} ([x_{a0}x_{a1}x_{a2}]_2 = [0]_{10} &\Rightarrow [x_{c0}x_{c1}x_{c2}]_2 = [2]_{10}) \\ \wedge ([x_{a0}x_{a1}x_{a2}]_2 = [1]_{10} &\Rightarrow [x_{c0}x_{c1}x_{c2}]_2 = [3]_{10}) \\ \wedge ([x_{b0}x_{b1}x_{b2}]_2 = [0]_{10} &\Rightarrow [x_{d0}x_{d1}x_{d2}]_2 = [2]_{10}) \\ \wedge ([x_{b0}x_{b1}x_{b2}]_2 = [1]_{10} &\Rightarrow [x_{d0}x_{d1}x_{d2}]_2 = [3]_{10}) \\ \wedge ([x_{c0}x_{c1}x_{c2}]_2 = [2]_{10} &\Rightarrow [x_{e0}x_{e1}x_{e2}]_2 = [2]_{10}) \\ \wedge ([x_{c0}x_{c1}x_{c2}]_2 = [3]_{10} &\Rightarrow [x_{e0}x_{e1}x_{e2}]_2 = [4]_{10}) \end{aligned}$$

Fig. 4. Resulting SAT encoding

Having the formulation for  $f$ , all these constraints are converted into CNF – the common input format of SAT solvers. Since only Boolean and relational operations are used, this can be done quite easily (see e.g. [25]). Afterwards, depending on the considered design tasks, additional constraints for verification or test purposes may be added (e.g. in case of equivalence checking, a miter structure as described later in Section VI-A). Then, the SAT solver determines only assignments that solve the respective design task with respect to the functionality of the quantum circuit.

## V. DISCUSSION

For the first time, the SAT formulation proposed above enables the encoding of quantum circuits with a non-restricted gate library. As motivated in Section III, this builds the basis for many applications e.g. in the SAT-based verification or the SAT-based automatic test pattern generation of/for quantum circuits. Nevertheless, the proposed encoding still inherits some disadvantages.

First of all, the respective operations of the quantum circuits are not encoded functionally but enumeratively. While e.g. conventional gates like *AND*, *OR*, etc. can easily be represented through functional constraints like  $c = a \wedge b$ ,  $c = a \vee b$ , etc., respectively, the proposed encoding relies on the complete enumeration of all possible input-output mappings in the considered gate. This makes the encoding larger but, still, is more feasible than a functional encoding of generic unitary operations.

Besides that, entanglement (see [1]) is not supported yet. Here, relations between more than one qubit have to be considered simultaneously. This poses a separate research question on its own. Hence, the support of entanglement is left for future work.

Finally, the performance of the proposed encoding obviously degrades with an increasing number of quantum states to be considered. The more quantum states may occur in the circuit, the larger gets the corresponding SAT encoding. Note that equivalent quantum states are thereby already handled by an equal encoding.

However, despite these drawbacks, the proposed encoding indeed enables the efficient consideration of design issues for quantum circuits through SAT-based approaches. This was also confirmed by an experimental evaluation whose results are discussed in the next section.

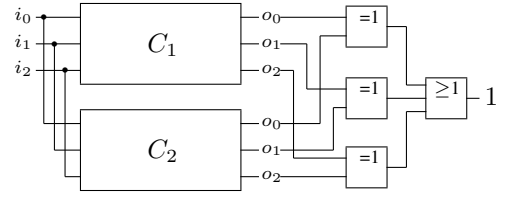


Fig. 5. Miter structure for equivalent checking

## VI. EXPERIMENTAL EVALUATION

In order to demonstrate the applicability and the efficiency of the proposed encoding, a SAT-based equivalent checker for quantum circuits has been developed which utilizes the proposed structural analysis and formulations. This section briefly reviews this application before results obtained with it are documented and discussed.

### A. SAT-based Equivalence Checker

*Equivalence checking* addresses the problem of checking whether two (quantum) circuits are equivalent or not. In typical design flows, this is crucial e.g. to determine whether an optimized version of a circuit still realizes the intended functionality.

To solve this problem with the proposed SAT encoding, a so-called *miter structure* as introduced in [18] for conventional circuits is built. Fig. 5 shows the general structure of the formulation. By applying the input assignments in turn to both circuits  $C_1$  and  $C_2$ , differences at the corresponding outputs are observed by XOR operations. If at least one XOR evaluates to 1 (determined by an additional OR operation), the two circuits are not equivalent. Hence, a decision problem results asking for a satisfying assignment to all circuit signals so that at least one output pair of  $C_1$  and  $C_2$  are different.

Using the encoding introduced in the previous section, this structure is represented as SAT instance and passed to a corresponding SAT solver. If the SAT solver returns *satisfiable*, an assignment has been found which leads to different outputs for  $C_1$  and  $C_2$ , i.e. a counterexample is returned. Otherwise, if the solver returns *unsatisfiable*, it has been proven that no such assignment exists and, hence, both circuits indeed realize the same functionality.

### B. Obtained Results

The described equivalence checker has been implemented in C++ on top of RevKit [26] and evaluated on a set of benchmarks taken from RevLib [27] and [1]. More precisely, quantum circuits from these sources have been taken and compared to each other leading to the resulting equivalence checking instances. In order to observe both, equivalent behavior and non-equivalent behavior, errors have been injected in some circuits by arbitrarily altering, adding, or deleting gates. Afterwards, all instances have been encoded into SAT and passed to *Boolector* [17] as underlying SAT solver. All experiments have been conducted on an Intel Pentium with 3.6 GHz and 2 GB of memory running Linux.

The results are summarized in Table II. The first columns denote thereby the name of the considered circuits, their respective number of circuit lines, as well as their gate count (for each circuit in the miter-structure). Column EQUIV.? denotes whether the respective circuits are equivalent (y) or not (n). Finally, the last three columns provide the number of variables needed for the encoding, the measured run-time in

TABLE II  
RESULTS OBTAINED WITH EQUIVALENCE CHECKER

BENCHMARK	LINES	GATES*	EQUIV.?	SAT VARS.	TIME	QUA. STATES
add64-184-a	193	272/256	y	516,221	13.86	31
add64-184-b	193	272/384	y	767,377	23.72	33
add64-184-c	193	272/256	y	516,221	13.92	31
add64-184-d	193	272/384	y	767,377	23.54	33
add64-184-e	193	272/386	n	769,697	11.27	33
add64-nct-fl	193	256/384	y	251,529	12.64	4
add64-nct-g	193	256/386	n	252,305	10.86	4
add64-ncv-h	193	386/384	n	301,969	16.23	4
add8-nct-a	25	46/32	y	4,569	0.59	48
add8-nct-b	25	46/48	y	4,621	0.76	48
add8-nct-c	25	46/49	n	5,453	0.63	48
c2-182-a	35	305/116	y	9,437	9.99	4
c2-182-b	35	305/116	n	9,435	1.97	4
c2-182-c	35	305/305	y	31,563	18.95	4
c2-182-d	35	305/304	n	31,491	5.42	4
c2-ncv-a	35	304/305	n	45,357	5.42	4
ckt2-cycle-a	8	5030/620	n	81,023	1021.70	2
hwb9-122-a	9	1539/1958	n	45,357	24.11	2
hwb9-123-a	9	1958/1959	n	57,518	261.64	2
hwb9-123-b	9	1984/1959	y	23,5457	483.55	48
hwb9-123-c	9	1984/1958	n	23,5395	264.80	48
toffoli-a	3	4/29	y	789	0.18	48
toffoli-b	3	4/88	y	2,115	0.18	48

\*The two numbers denote the number of gates for each circuit in the miter-structure.

CPU seconds, as well as the total number of different quantum states which occurred in the considered circuits.

The results confirm that, indeed, the number of different quantum states can be kept small. Obviously, this depends on the considered circuits: For example, *ckt2-cycle-a* is composed of Boolean gates only, and, hence requires just a 2-valued encoding. In contrast, *hwb9-123-b* incorporates nine different quantum gates types so that the number of quantum states to be considered significantly increases (to 48 in this case). Nevertheless, the proposed SAT encoding remains efficient. Even if dozens of quantum states need to be considered, results for circuits composed of thousands of gates still can be determined in a reasonable time. Compared to previous work, SAT-based verification of quantum circuits allowing various quantum states became possible for the first time.

## VII. CONCLUSIONS

In this work, we introduced a compact and efficient SAT encoding for quantum circuits. We particularly addressed thereby the problem that qubits (theoretically) may assume an infinite number of quantum states. We showed that, in fact, quantum circuits are inherently restricted in their number of possible states. This was exploited by a structural analysis whose results are utilized to create a proper SAT encoding. The applicability of the proposed approach has been demonstrated by means of a SAT-based equivalence checker. Quantum circuits composed of thousands of gates and qubits assuming dozens of quantum states can efficiently be handled. By this, a SAT formulation enabling the encoding of quantum circuits with a non-restricted gate library has been proposed for the first time. This builds the basis for many applications in the domain of verification and test of quantum circuits.

## REFERENCES

- [1] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.
- [2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.
- [3] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, p. 883, 2001.
- [4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Theory of computing*, 1996, pp. 212–219.
- [5] S. Dörn and T. Thierauf, "The quantum complexity of group testing," in *Conf. on Current Trends in Theory and Practice of Computer Science*, 2008, pp. 506–518.
- [6] R. Drechsler and R. Wille, "From truth tables to programming languages: Progress in the design of reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2011, pp. 78–85.
- [7] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits - a survey," *ACM Computing Surveys*, 2012.
- [8] I. Polian, T. Fiehn, B. Becker, and J. P. Hayes, "A family of logical fault models for reversible circuits," in *Asian Test Symp.*, 2005, pp. 422–427.
- [9] 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.
- [10] S.-A. Wang, C.-Y. Lu, I.-M. Tsai, and S.-Y. Kuo, "An XQDD-based verification method for quantum circuits," *IEICE Transactions*, vol. 91-A, no. 2, pp. 584–594, 2008.
- [11] S. Yamashita, S. Minato, and D. M. Miller, "An efficient verification of quantum circuits under a practical restriction," in *Int'l Conf. on Computer and Information Technology*, 2008, pp. 873–879.
- [12] R. Wille, D. Große, D. M. Miller, and R. Drechsler, "Equivalence checking of reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2009, pp. 324–330.
- [13] R. Wille, H. Zhang, and R. Drechsler, "ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization," in *IEEE Annual Symposium on VLSI*, 2011, pp. 120–125.
- [14] J. Seiter, M. Soeken, R. Wille, and R. Drechsler, "Property checking of quantum circuits using quantum multiple-valued decision diagrams," in *Workshop on Reversible Computation*, 2012.
- [15] G. F. Viamontes, M. Rajagopalan, I. L. Markov, and J. P. Hayes, "Gate-level simulation of quantum circuits," in *ASP Design Automation Conf.*, 2003, pp. 295–301.
- [16] D. M. Miller and M. A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, 2006, p. 6.
- [17] R. Brummayer and A. Biere, "Boolector: An efficient SMT solver for bit-vectors and arrays," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2009, pp. 174–177.
- [18] D. Brand, "Verification of large synthesized designs," in *Int'l Conf. on CAD*, 1993, pp. 534–537.
- [19] J. Marques-Silva and T. Glass, "Combinational equivalence checking using boolean satisfiability and recursive learning," in *Design, Automation and Test in Europe*, 1999, pp. 145–149.
- [20] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu, "Symbolic model checking using SAT procedures instead of BDDs," in *Design Automation Conf.*, 1999, pp. 317–320.
- [21] P. Williams, A. Biere, E. Clarke, and A. Gupta, "Combining decision diagrams and SAT procedures for efficient symbolic model checking," in *Computer Aided Verification*, ser. LNCS, vol. 1855. Springer Verlag, 2000, pp. 124–138.
- [22] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. on CAD*, vol. 11, pp. 4–15, 1992.
- [23] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille, "On acceleration of SAT-based ATPG for industrial designs," *IEEE Trans. on CAD*, vol. 27, pp. 1329–1333, 2008.
- [24] S. A. Cook, "The complexity of theorem proving procedures," in *Symposium on Theory of Computing*, 1971, pp. 151–158.
- [25] G. Tseitin, "On the complexity of derivation in propositional calculus," in *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, 1968, pp. 115–125, (Reprinted in: J. Siekmann, G. Wrightson (Ed.), *Automation of Reasoning*, Vol. 2, Springer, Berlin, pp. 466–483, 1983.).
- [26] M. Soeken, S. Fehse, R. Wille, and R. Drechsler, "RevKit: An Open Source Toolkit for the Design of Reversible Circuits," in *Reversible Computation 2011*, ser. Lecture Notes in Computer Science, vol. 7165, 2012, pp. 64–76, RevKit is available at [www.revkit.org](http://www.revkit.org).
- [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, RevLib is available at <http://www.revlib.org>.