Towards the Design Automation of Quantum Circuits

Sidi Mohamed Beillahi and Sofiène Tahar
Department of Electrical and Computer Engineering
Concordia University, Montreal, Canada
Email: beillahi, tahar@ece.concordia.ca

Mohamed Yousri Mahmoud
Department of Electrical Engineering and Computer Science,
University of Ottawa, Ottawa, Ontario, Canada
Email: myousri@uottawa.ca

Abstract—Quantum computing systems promise to increase the efficiency of solving problems which classical computers cannot handle efficiently, such as integers factorization. In this paper, we introduce the idea of design automation for quantum circuits, where we develop a rich library of quantum gates and a robust decision procedure in higher-order logic, to reason about quantum circuits formally. We adopt a low level modeling approach of quantum gates at the optics physics level. Interestingly, this approach facilitates the modeling of optimized implementations for certain quantum gates that cannot be optimally modeled in the available analysis tools. The constructed library contains a variety of quantum gates ranging from 1-qubit to 3-qubit gates that are sufficient to model most of the existing quantum circuits. To eliminate the user’s interaction with the theorem prover, we develop a decision procedure and tactics that fully automate the analysis process. As real world applications, we present the formal analysis of several quantum circuits including the quantum full adder and the Grover’s oracle circuits, for which we have proved the behavioral correctness and calculated the operational success rate which has never been provided in the literature.

I. INTRODUCTION

Quantum computers are expected to be employed in high computing domains and to secure communications (e.g., monetary transactions on the Internet) [10], [12]. In [13], the authors have shown that it is possible to create a “universal” quantum computer using only single photon sources, detectors, and linear optical elements (e.g., beam splitter and phase shifter) [14]. However, there still many practical challenges that impedes the realization of an optical quantum computer such as: initialization of quantum bits (i.e., two-state quantum systems), measurements, decoherence (i.e., unwanted interaction between a quantum system and its environment).

Quantum systems like any other engineering systems need Computer Aided Design (CAD) tools to facilitate their design and deployment in real life application. These CAD tools help in the realization of new designs and evaluation of their efficiency without the need for expensive laboratory setups. However, due to the inherent complexity of quantum circuits, which rely on mathematical models of quantum physics principles, numerical simulations based CAD tools are incomplete: the computation space increases exponentially with the size of the quantum circuit. Even though, there exist several methods and tools for numerical simulation and modeling of quantum circuits, where quantum gates are described as matrices and applied to quantum states using matrix-vector multiplication, e.g., [22], [5]. In these works, the simulations were performed at the gates level without modeling the physical elements of these gates which may facilitate the optimization process. Nevertheless, in [5] for instance a time-out is reached when simulating 15 qubits (quantum bits) circuits. On the other hand, quantum circuits synthesis has been an active area of design automation research (e.g., [30], [6]), where the authors proposed methods for the synthesis of quantum circuits built using elementary gates at the behavioral level. However, the physical design of the elementary gates was not tackled which limits the underlying works to find the optimized circuit. For example, in [30], [6], the Toffoli gate circuit was synthesized into five 2-qubit gates, however, it is possible to implement it in quantum optics using only three 2-qubit gates as shown in [24], by utilizing the photons polarization for the qubits.

Higher-order logic (HOL) theorem proving [11] offers rich mathematical foundation to fulfill the main requirement for modeling quantum systems. Because quantum theory is mainly based on infinite linear spaces, we believe that HOL can assist in the analysis of quantum circuits. In this paper, we propose to use the HOL Light theorem prover [8] to handle the automated design, modeling and verification of quantum circuits due to its rich support for multivariate calculus and Hilbert spaces theories [7], [18] which are essential to reason about quantum physics.

In [19], [17], the authors used the Hilbert-spaces library of HOL Light [7] to formalize flip (NOT) and controlled-not (CNOT) gates. Thus in [27], Hadamard, non-linear sign(NS), controlled-phase (CZ) gates were also formalized in HOL Light using the tensor product projection. However, the underlying gates are simple and cannot form an universal library. In this paper, we are building a rich quantum gates library and developing a decision procedure to automate the analysis of quantum circuits constructed based on the underlying library. Therefore, in order to be able to offer a rich library that can model a variety of quantum circuits, we formalize flip (the underlying technology of the flip gate is different than the one implemented in [19]), SWAP, Toffoli sign, Toffoli and Fredkin gates, which are widely used quantum gates. Afterwards, we develop a decision procedure and a set of automated tactics to perform automatically the analysis of any quantum circuits constructed using the gates library. Finally, we demonstrate the
effectiveness of our proposed framework by showing several quantum circuits including quantum full adder, decoder, grover oracle, etc.

The most related work to us in [26], where a quantum process calculus is used to model linear optical quantum systems, which was applied to model the CNOT gate. The main limitation of this work is that the beam splitters parameters are considered as real numbers, however, in the general context of quantum optics they can be complex numbers as in the case of quantum interferometer [20]. In another related work [9], the authors used model checking to formally synthesize quantum circuits based on a fixed set of 1-qubit and 2-qubits gates. Both works have not showed the scalability of their approaches.

It is important to note that most of the existing quantum circuits models in the literature are not physically feasible due to the adjacency principle between qubits [1]. The principle states that in order to apply a gate to a set of qubits, those qubits need to be adjacent [1]. Therefore, in order to make qubits adjacent to each other, we add SWAP gates to the circuit to switch qubits between each other. This shows the valuable merits of our approach for doing an expressive low level modeling of quantum circuits, over the existing techniques, e.g., [29], [26].

To the best of our knowledge, this is the first time a systematic automated formal low level modeling, design, and verification of quantum circuits is tackled. The source code of our formalization is available for download [2] and can be utilized by other researchers and engineers.

The rest of the paper is organized as follows: In Section II we present a short overview of quantum optics. We highlight our proposed framework in Section III. Then, we present the formalization of the quantum gates library in HOL Light in Section IV. In Sections V we describe the flow of the automation procedure. Next, in Section VI we present the results of the formal analysis of several quantum circuits. Finally, we conclude the paper in Section VII.

II. QUANTUM OPTICS

Quantum optics is considered as one of the rich and promising approaches under investigation for realizing quantum computers [31], [3]. This is because photons decohere slowly, photons can move quickly (at the speed of light), and photons can be experimented with at room temperature. For quantum optics, the physical state of an optical beam is a probability density function which provides the probability of the number of photons inside the beam, typically written as \( \psi_j \). The most elementary optical quantum states are called fock states. An optical beam is in a fock state \( |n\rangle \) if it contains exactly \( n \) photons. The set of fock states represents an orthonormal basis for the linear space that contains all optical quantum states. Therefore any quantum state \( |\psi\rangle \) of a given mode (i.e., a light beam) is a superposition of the fock states, i.e., \( |\psi\rangle = \sum_n |\psi_n\rangle |n\rangle \). In the case of \( n \) independent optical beams, the joint quantum state \( |\Psi\rangle \) is described as the tensor product of the states \( |\psi_j\rangle \):

\[
|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle
\]  

(1)

where \( |\psi_j\rangle \) is the quantum state of the \( j^{th} \) beam. Given the quantum states \( |\psi_1\rangle \ldots |\psi_n\rangle \) of \( n \) optical beams, the function that describes the joint probability of the \( n \) beams is then the point-wise multiplication of all the states. Hence, we define the tensor product for \( n \)-beam quantum state as follows:

\[
\lambda y_1 \ldots y_n. \left( |\psi_1\rangle \otimes \ldots \otimes |\psi_n\rangle \right) (y_1 \ldots y_n) = |\psi_1\rangle y_1 \ldots * |\psi_n\rangle y_n.
\]

Thus, we formally define the tensor product for \( n \) beams in HOL Light, recursively, as follows:

**Definition 1 (Tensor Product):**

\[
\vdash \text{tensor } 0 \text{ modes} = (\lambda y. \ 1) \wedge \\
\text{tensor } n+1 \text{ modes} = (\lambda y. \ ((\text{tensor } n \text{ modes}) \ y) \ * \ (\text{modes}(n+1) \ y(n+1)))
\]

where \( \text{mode} \) is a vector of size \( n \) that contains \( n \) modes.

The basic case of zero mode \( n = 0 \) is a trivial case; it is a constant function (i.e., \( y \rightarrow 1 \)) and it guarantees a terminating definition. We have proved that this tensor satisfies the property of bi-linearity.

III. PROPOSED FORMAL ANALYSIS FRAMEWORK

The proposed framework for the analysis of quantum circuits, given in Figure 1 outlines the necessary steps to encode theoretical fundamentals in higher-order logic. Quantum circuits are usually modeled as a set of connected gates. In order to represent a given quantum circuit in HOL, the first step is the formalization of quantum primitives gates using optics elements. Consequently, the gates library is expanded to include 2-qubits and 3-qubits gates. Thus the formed library of quantum gates can model a big variety of quantum circuits.

The next step is to facilitate the usage of this library for the formal modeling and analysis of quantum circuits. In this step, we develop a decision procedure that captures the quantum circuit model and extracts a matrix that captures the circuit structure. Subsequently, based on this matrix, the procedure generates the proof steps and required simplification rules to conduct the formal proof of quantum circuits properties automatically without the need for the user interaction. Finally, we demonstrate the effectiveness of the framework through the formal analysis of several quantum circuits.

IV. FORMALIZATION OF QUANTUM GATES LIBRARY

Building upon the existing quantum gates (i.e., Hadamard, non-linear sign, CZ, and CNOT), we have formally modeled 5 additional important quantum gates, namely flip, SWAP, Toffoli sign, Toffoli, and Fredkin gates. The flip gate implementation is based on single photon source. For the Toffoli sign gate model, it is taken from [24] in which the gate has 3 inputs, where one input is qutrit (i.e., has 3 quantum states) and the rest are qubits. This makes it impossible to model this gate using existing tools because of their boolean foundation (i.e., compatible with only two states entity). Based on the Toffoli sign gate we can model an optimal implementation of the
Fig. 1: Proposed Analysis Framework for Quantum Circuits

The Toffoli gate. In this section, we will present the formalization of the flip, SWAP, Toffoli sign, Toffoli, and Fredkin gates.

A. Formalization of Flip Gate

The majority of existing gates are implemented using a single photon source. However, to the best of our knowledge, the only existing optical implementation for the flip (not gate) is using coherent light source [23]. Using our formalization we are able to derive a new design of this gate using a single photons source. The proposed design has a success probability (i.e., the probability in which the circuit produces the right output) 100% and is composed of two beam splitters and a phase shifter as shown in Figure 2. We verify that our design and the one in [23] are equivalent by checking that the underlying design behaves according to the flip gate truth table. Hence this shows the practical aspect of our methods of enabling the design, modeling and verification of new physical designs for quantum gates. The flip gate flips the input state: if the possible input is $|\phi\rangle_{\text{input}} = \alpha |0\rangle + \beta |1\rangle$ then the output is $|\phi\rangle_{\text{output}} = \alpha |1\rangle + \beta |0\rangle$. We formally define the flip gate

$$\text{FLIP Outputs: } |0\rangle \iff \text{FLIP} \rightarrow \text{FLIP}_\text{In} \rightarrow \text{FLIP}_\text{Out} \iff \text{FLIP}_\text{In} \rightarrow \text{FLIP}_\text{Out}$$

Theorem 1 (Inputs: $|1\rangle_a \equiv \text{LV } a$ and $|0\rangle_a \equiv \text{LH } a$):\[ \text{FLIP}_\text{GATE} (a,d,ten) \Rightarrow \text{tensor 1 } (\lambda_i. \text{LV } a) = \text{tensor 1 } (\lambda_i. \text{LH } d) \land \text{tensor 1 } (\lambda_i. \text{LH } a) = \text{tensor 1 } (\lambda_i. \text{LV } d) \]

Notice that, the proposed design for the flip gate is physically feasible using the available optics technology as it requires only two beam splitters, a phase shifter and a single photons source and detector.

B. Formalization of SWAP Gate

The SWAP gate is a two qubits gate which swaps the states of the two input qubits. It has a crucial role in the design of quantum circuits where the SWAP gate is used to swap the qubits between each other in order to fulfill the requirement that computations should only be performed between adjacent qubits [23]. Also in [15], the authors show the role of SWAP gates for the storage of quantum information, where the SWAP gate swaps the information of qubits between flying qubits which are not suitable for storage of quantum information and statics qubits. The physical implementation of the SWAP gate requires three CNOT gates, as shown in Figure 3. A CNOT gate has two inputs which are the control and target qubits. If

\[
\begin{align*}
\text{a}_1 & \rightarrow \text{c}_1 \\
\text{d}_1 & \rightarrow \text{b}_1
\end{align*}
\]

Fig. 3: SWAP Gate

where LH and LV are employed to describe the representation of qubits using the vertical and horizontal polarization of photon, respectively. For example, LV a1 (resp., a2) represents a vertically (resp., horizontally) polarized photon in the mode a1 which describes the qubit in the state $|1\rangle$ (resp., $|0\rangle$). is_beam_splitter accepts the beam splitter parameters: input $(a(1),1,a(2),2)$ and output ports $(b(1),1,b(2),2)$, tensor operator ten, and the beam splitter $2 \times 2$ matrix $(\begin{array}{c} 1/\sqrt{2} \, -1/\sqrt{2} \\ 1/\sqrt{2} \, 1/\sqrt{2} \end{array})$. Note that, for such parameters, the inputs and outputs will be related as follows: $a(1) = \frac{1}{\sqrt{2}} b(1) - \frac{1}{\sqrt{2}} b(2)$ and $a(2) = \frac{1}{\sqrt{2}} b(1) + \frac{1}{\sqrt{2}} b(2)$. phase_shifter accepts the parameters: input port $(b(2),2)$, output port $(c(2),2)$, tensor operator and the angle $\pi$. FLIP_GATE takes as parameters the gate input and output ports (i.e., $a$ and $d$, respectively), and the tensor operator. As the two states of the qubit are represented by the internal polarization degree of freedom of a single photon, $|0\rangle_L = \text{LH } |1\rangle \otimes |0\rangle \equiv |1,0\rangle$ and $|1\rangle_L = \text{LV } |0\rangle \otimes |1\rangle \equiv |0,1\rangle$. Therefore, we defined FLIP_In_Output an input/output wrapper of the flip gate in order to facilitate the use of the gate (i.e., instead of considering the gate input as two polarization modes we describe it as a single quantum logical input). We succeeded to verify the equivalence of our design with the one proposed in [23] (and verified in [19]) by formally proving the result of applying the flip gate on the two possible inputs $|0,1\rangle_a$ and $|1,0\rangle_a$.

Definition 2 (Flip Gate):

$$\text{FLIP } (a,d,ten) \iff (\forall b \, c.
\text{phase} \_\text{shifter}(ten,a,b,c,d) \land
\text{is} \_\text{beam} \_\text{splitter}(\begin{array}{cc}
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{array},ten,a(1),1,a(2),2,
b(1),1,b(2),2) \land
\text{is} \_\text{beam} \_\text{splitter}(\begin{array}{cc}
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{array},ten,b(1),1,b(2),2,d(1),1,c(2),2) \land
\text{FLIP}_\text{In} \_\text{Output}(a,d,b,c,LH,LV))$$

Fig. 2: Single photon source design of flip gate
the control qubit is \( |1 \rangle \) then the target qubit will be inverted and nothing change if the control qubit is \(|0\rangle\). We formally define the structure of the SWAP gate in HOL as follows:

**Definition 3 (SWAP Gate Structure):**

\[ \vdash \text{SWAP\_GATE}(x_1, x_2, y_1, y_2, \text{ten}, \text{LV}, m_{\text{proj}}) \iff (\forall c1, c2, d1, d2. \text{CNOT\_GATE}(c1, c2, d1, d2, \text{ten}, \text{LV}, m_{\text{proj}}) \land \text{CNOT\_GATE}(x_1, x_2, c1, c2, \text{ten}, \text{LV}, m_{\text{proj}}) \land \text{CNOT\_GATE}(d1, d2, y_1, y_2, \text{ten}, \text{LV}, m_{\text{proj}})) \]

where \( m_{\text{proj}} \) is the tensor product projection operator, it is used to project the circuits outputs on expected output in order to eliminate the undesirable states, more details can be found in [27], [3]. Accordingly, we employed this definition to prove the result of the application of SWAP gate over an input in the general form \(|a_1, a_2\rangle = (\alpha_1 \% |1\rangle + \alpha_2 \% |0\rangle) \otimes (\beta_1 \% |1\rangle + \beta_2 \% |0\rangle) \) (i.e., each state is the superposition of the two states \(0\) and \(1\)).

**Theorem 2 (SWAP Gate Input: \(|x, y\rangle\):**

\[ \vdash \text{SWAP\_GATE}(a1, a2, j1, j2, \text{ten}, \text{LV}, m_{\text{proj}}) \Rightarrow \text{tensor } 2 \left( \lambda_i. \text{if } i = 1 \text{ then } (\alpha_1 \% \text{LV } a1 + \alpha_2 \% \text{LV } a1) \text{ else } (\beta_1 \% \text{LV } a2 + \beta_2 \% \text{LV } a2) \right) \]

We can notice that there is a scalar multiplied by the output state, \( \frac{1}{\sqrt{2}} \), which represent the success rate of the gate (i.e., the probability at which the gate produces the correct output). In the following section, we will present the Toffoli Sign gate.

**C. Formalization of Toffoli Sign Gate**

The Toffoli sign (TS) gate is a 3-qubit reversible gate that applies a sign shift on one of the state components, and the identity to other inputs. The main benefit of TS is to use it for construction of the Toffoli gate. The TS gate is composed of two CNOT gates (first and last gates) and a CZ gate. The gate structure is shown in Figure 4. We formally define the structure of TS gate in HOL as follows:

**Definition 4 (TS Gate Structure):**

\[ \vdash \text{TS\_GATE}(a1, a2, a3, b1, b2, b3, \text{ten}, \text{LV}, m_{\text{proj}}) \iff (\forall k, c1, c2, d1. \text{CNOT\_GATE}(a1, a2, c1, c2, \text{ten}, \text{LV}, m_{\text{proj}}) \land \text{CZ\_GATE}(c2, a3, d2, b3, \text{ten}, \text{LV}, m_{\text{proj}}) \land \text{CNOT\_GATE}(c1, d2, b1, b2, \text{ten}, \text{LV}, m_{\text{proj}}) \land \text{TS\_outputs}(k, b1, b2, b3, \text{LV}, \text{LV}) \land \text{TS\_inputs}(k, a1, a2, a3, \text{LV}, \text{LV})) \]

Notice that in the Figure 4 TS gate has four input modes, however, TS is a 3-qubits gate, where the first logical qubit is represented by two optical modes. In our formal definition of the TS gate, for an input \(|x, y, z\rangle\), \(x\) is the qutrit input. We formally verify the result of applying TS transformation on all the possible inputs combination. We present here one combination of them i.e., \([101]\):

**Theorem 3 (TS Input: \([101]\):**

\[ \vdash \text{TS\_GATE}(a1, a2, a3, b1, b2, b3, \text{ten}, \text{LV}, m_{\text{proj}}) \Rightarrow \text{tensor } 3 \left( \lambda_i. \text{if } i = 1 \text{ then } \text{LV } a1 \text{ else } i = 2 \text{ then } \text{LV } a2 \text{ else } \text{then } \text{LV } a3 \right) \]

Here the sign shift for the output state \(|0, 1, 1\rangle\) which is also multiplied by a scalar value that represents the gate success probability: \( \frac{1}{\sqrt{2}} \). For the reset possible combinations for the inputs there will be no sign shift. Now, after formally modeling and verifying the Toffoli Sign gate, we are ready to tackle the formalization of the Toffoli gate.

**D. Formalization of Toffoli Gate**

The Toffoli gate is a 3-qubit gate that flips the logical state of the target qubit if the two control qubit are in the state \(|1\rangle\) and does nothing for the rest of cases. The Toffoli is one of the most important quantum gates and has many quantum applications including universal reversible classical computation, quantum error correction and fault tolerance. The simplest known design of the Toffoli gate when restricted to operating on qubits at the behavioral level is a circuit that requires five 2-qubit gates, the gate is shown in Figure 5. However, it was shown that it is possible to realize a Toffoli gate using the Toffoli sign gate which is composed of only three 2-qubit gates, flip, and Hadamard gates [24]. The Toffoli gate can be implemented in two forms; 1) the first qubit is the target and the rest are the control qubits; or 2) the third qubit is the target and the rest are the control qubits. Therefore, we have formally defined these two kinds of Toffoli gate in HOL. We provide here the formal definition of the second type structure of Toffoli gate as follows:

**Definition 5 (Toffoli Gate Structure):**

\[ \vdash \text{TOFFOLI\_GATE}(a1, a2, a3, b1, b2, b3, \text{ten}, \text{LV}, m_{\text{proj}}) \iff (\forall c3, c1, c2, d1. \text{HADAMARD\_GATE}(a3, c3, \text{ten}, \text{LV}, \text{LV}) \land \text{HADAMARD\_GATE}(a1, c1, \text{ten}, \text{LV}, \text{LV}) \land \text{HADAMARD\_GATE}(a2, c2, \text{ten}, \text{LV}, \text{LV})) \]

where HADAMARD\_GATE describes the structures and Hadamard gates. More details about the gate formalization can be found in [3]. From this definition, we verify the result of applying
Toffoli on the input $|111\rangle$, where the two control qubits are $|1\rangle$.

Theorem 4 (Toffoli Input: $|111\rangle$):
\[ \vdash \text{TOFFOLI\_GATE}(a_1, a_2, a_3, b_1, b_2, b_3, \text{ten}, \text{LV}, m, \text{proj}) \]
\[ \Rightarrow \text{CNOT\_GATE}(a_2, a_3, c_2, c_3, \text{ten}, \text{LV}, m, \text{proj}) \]
\[ \land \]
\[ \text{TOFFOLI\_GATE}(a_1, c_2, c_3, b_1, d_2, d_3, \text{ten}, \text{LV}, m, \text{proj}) \]
\[ \land \]
\[ \text{CNOT\_GATE}(a_2, d_2, b_3, \text{ten}, \text{LV}, m, \text{proj}) \]

Notice that the success probability of the Toffoli gate is the same as the one of the Toffoli Sign: \( \frac{1}{4} \). Contrarily, if the Toffoli gate was constructed using five 2-qubit gates, the success probability will be \( \frac{1}{1024} \).

E. Formalization of Fredkin Gate

The Fredkin gate or the controlled-2x2 reversible quantum switch gate (or controlled SWAP gate) is a 3-qubits gate [21]. One of the qubits is designated as the control qubit and is left unchanged by the gate, and the rest two qubits are the target qubits. If the control qubit is zero the two target qubits remain unchanged. If the control qubit is one the two target qubits are interchanged. The Fredkin gate has an important role in quantum computing and quantum computations error-correcting [16]. Moreover, Fredkin gate is a universal gate for reversible computing which means that any logical or arithmetic operation can be constructed entirely using this gate [21]. The gate circuit is shown in Figure 6 which is composed of two CNOT and one Toffoli gates. We formally model the structure of the Fredkin gate in HOL as follows:

Definition 6 (Fredkin Gate Structure):
\[ \vdash \text{FREDKIN\_GATE}(a_1, a_2, a_3, b_1, b_2, b_3, \text{ten}, \text{LV}, m, \text{proj}) \]
\[ \iff (\forall c_2 c_3 d_2 d_3. \]
\[ \text{CNOT\_GATE}(a_2, a_3, c_2, c_3, \text{ten}, \text{LV}, m, \text{proj}) \land \]
\[ \text{TOFFOLI\_GATE}(a_1, c_2, c_3, b_1, d_2, d_3, \text{ten}, \text{LV}, m, \text{proj}) \land \]
\[ \text{CNOT\_GATE}(a_2, d_2, b_3, \text{ten}, \text{LV}, m, \text{proj}) \]

From this definition, we verify the result of applying the Fredkin gate on the input $|zxy\rangle$:

Theorem 5 (Fredkin Input: $|zxy\rangle$):
\[ \vdash \text{FREDKIN\_GATE}(a_1, a_2, a_3, b_1, b_2, b_3, \text{ten}, \text{LV}, m, \text{proj}) \]
\[ \Rightarrow \text{tensor} \ 3 \ (\lambda i. \ if \ i = 1 \ then \ (\alpha_1 \% \text{LV} \ a_1 + \alpha_2 \% \text{LV} \ a_1) \]
\[ \text{else} \ if \ i = 2 \ then \ (\beta_1 \% \text{LV} \ a_2 + \beta_2 \% \text{LV} \ a_2) \]
\[ \text{else} \ (\theta_1 \% \text{LV} \ a_3 + \theta_2 \% \text{LV} \ a_3) = \alpha_1 \% \]
\[ \text{tensor} \ 3 \ (\lambda i. \ if \ i = 1 \ then \ \text{LV} \ b_1 \text{ else} \]
\[ \text{else} \ if \ i = 2 \ then \ (\beta_1 \% \text{LV} \ b_2 + \beta_2 \% \text{LV} \ b_2) \]
\[ \text{else} \ (\theta_1 \% \text{LV} \ b_3 + \theta_2 \% \text{LV} \ b_3) + \alpha_2 \% \]
\[ \text{tensor} \ 3 \ (\lambda i. \ if \ i = 1 \ then \ \text{LV} \ b_1 \text{ else} \]
\[ \text{else} \ if \ i = 2 \ then \ (\beta_1 \% \text{LV} \ b_2 + \beta_2 \% \text{LV} \ b_2) \]
\[ \text{else} \ (\theta_1 \% \text{LV} \ b_3 + \beta_2 \% \text{LV} \ b_3) \]

Notice that the success probability of the Fredkin gate is \( \frac{1}{1024} \) and it is very small.

By this, we have covered the formal modeling, design and verification of a set of quantum gates which can be used in the analysis of a variety of quantum circuits.

V. Decision Procedure

The main idea behind the developed decision procedure is to develop tactics that can fully remove the user interaction with the HOL Light theorem prover, paving the path for the construction of an automated tool for the analysis of quantum circuits. The decision procedure is built such that it can be customized for the underlying circuit analysis. Here we mean by customization is that based on a circuit description the procedure constructs the required automation tactics for the circuit analysis.

Generally, any quantum circuit is a collection of gates that are connected to each other either sequential or parallel. Therefore, the main proof steps for the analysis of any quantum circuit are: 1) to unfold the input tensor product to elementary tensors to be inputted to parallel gates; 2) to apply the required gates transformation; 3) to fold the tensor product back. Then we repeat this process until the input tensor goes through all the gates transformations. Finally, we rewrite the obtained result to the standard format using some existing HOL Light theorems (e.g., linearity of linear algebra spaces).

Hence, the core of the procedure is an Ocaml function that extracts from a textual quantum circuit description a matrix that contains information about the circuit gates, their inputs/outputs and their orders. The information contained in this matrix are crucial to perform the three steps explained earlier. This function searches in the given circuit description: 1) if two gates are sequential and which one is first applied to the circuit input; and 2) if two gates are parallel what is their inputs order within the circuit input vector. Knowing this information helps in unfolding the input tensor product to elementary tensors for each particular gate.

Then a second Ocaml function takes this matrix and generates the proof steps (tactics). This function based on the matrix provides proof steps and subgoals and lemmas to automatically proof the required theorems for the underlying circuit.

For example, let consider the quantum circuit given in Figure 4. The circuit is for quantum full adder and it is composed of two SWAP, three CNOT and one Fredkin gates. Using the first Ocaml function described previously we extract

Fig. 6: Fredkin Gate

Fig. 7: Quantum Full Adder
the following matrix:
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

In the matrix each row contains the description of gates that can be applied in parallel to the circuits and their orders and the matrix rows are ordered in term of dependability between the gates.

To illustrate the task of the second Ocaml function and the flow of the decision procedure and the lemmas involved let consider a n-qubits circuit that contains a m-qubits gate (m ≤ n) and the general form of its input is: tensor n (λi. if i = 1 then x_i else ⋯ else x_n). Two of the most important properties of tensor product are the ability to write tensor as tensor of tensor (lemma1 in Table[I] and vice versa (lemma6 in Table[I]). Then the first step is to rewrite the main tensor product (circuit input) using lemma1, lemma2, lemma3 and lemma6 in the form:

\[
\begin{align*}
(λy. (\text{tensor } k1 (λi. if i = 1 then x_i else ⋯ else x_n+1)) y)
& \times (\text{tensor } m (λi. if i = 1 then x_{n+1} else ⋯ else x_{n+k})) \\
& (λi. y(i+1)) \times (\text{tensor } k2 (λi. if i = 1 then x_{n+k} else ⋯ else x_n)) \times (λi. y(i+k+1))
\end{align*}
\]

where n = k1 + m + k2. After rewriting each elementary tensor in the standard form as in the equation above, we are now capable of replacing the term tensor (λi. if i = 1 then x_i else ⋯ else x_n) with its transformation under the m-qubits gate. Thus the circuit input will be as follows:

\[
\begin{align*}
(λy. (\text{tensor } k1 (λi. if i = 1 then x_i else ⋯ else x_n+1)) y)
& \times (a \% \text{ tensor } m (λi. if i = 1 then x_{n+1} else ⋯ else x_{n+k})) \\
& (λi. y(i+1)) \times (\text{tensor } k2 (λi. if i = 1 then x_{n+k} else ⋯ else x_n)) \times (λi. y(i+k+1))
\end{align*}
\]

The last step consists of folding back the tensor product by using lemma4, lemma5, lemma6, and lemma7 of Table[I]. Thus the circuit input will be in the form:

\[
a \% \text{ tensor } n (λi. if i = 1 then x_i else ⋯ else if i = k1 + 1 then x_{k1+1} else ⋯ else if i = k1 + m + 1 then x_{k1+m+1} else ⋯ else x_n)
\]

Thus we have successfully applied the m-qubit gate transformation to the m-qubits circuit input. We repeat the same procedure to all the circuit gates transformation over the input until reaching the final value of the circuit output. Notice that this decision procedure can be applied to any quantum circuit that is constructed based on the gates library.

In case of the quantum adder that is a 4-qubits circuit which means that the input vector is in the form of tensor 4 mode. When the second Ocaml function takes the circuit matrix, in the first row in the matrix we have two parallel gates, accordingly we should unfold the input tensor to two elementary tensors: tensor 2 mode1 and tensor 2 mode2 and apply the two CNOT gates to the two tensors. Then we fold back to the main tensor. Consequently, in the second row we have one gate, however, this gate order is in the middle of the main tensor. Therefore, we should unfold the main tensor to three elementary tensors: tensor 1 mode1, tensor 2 mode2 and tensor 1 mode3 and apply the CNOT gate to the elementary tensor tensor 2 mode2. Then we fold back to the main tensor. Subsequently, we repeat the same procedures for the remaining two rows of the matrix until all gates are applied, and the final tensor product is obtained which is the circuit output.

In the following sections we will show several quantum circuits that have been automatically analyzed using the underlying decision procedure.

VI. DECISION PROCEDURE: APPLICATIONS

In this section we provide the results of the formal analysis of several quantum circuits. We have analysed several quantum benchmarks circuits taken from the online library of reversible and quantum circuits at [25]. All the circuits are provided in the online library do not meet the adjacency criteria in quantum computing. The adjacency criteria implies that a quantum gate cannot be applied to qubits that are not adjacent. This criteria is supported experimentally and theoretically [1]. For example, in order to apply a 2-qubit gate to two element x_k1 and x_k2 of n-qubit input, the input should be in the form tensor n (λi. if i = 1 then x_1 ⋯ else if x = k1 then x_k1 else if x = k2 then x_k2 else ⋯ else x_n). Therefore, we added SWAP gates to all the quantum circuits taken from [25] to move the qubits to be adjacent to each other when they are applied to the same gate. The set of benchmarks circuits that were analyzed are:

- \text{gf2}^3\text{mult} finds product of two elements of a field \text{GF}(2^3), a = a_0 + a_1x + a_2x^2 and b = b_0 + b_1x + b_2x^2 with the output, ab = c = c_0 + c_1x + c_2x^2 written on the last 3 qubits.
- 2-to-4 decoder that has 3 inputs and 4 outputs. If the enable qubit is low, all the output qubits will be zero. If the enable qubit is high, one of the four output qubits will become high selected by the remaining two input qubits.
- \text{hwb}4 is the hidden weighted bit function with parameter (N=4).
- \text{ham}3 is the size 3 Hamming optimal coding function.
- mod5, the Grover’s oracle, which has 4 inputs and 1 output. Its output is 1 if and only if the binary number represented by its input is divisible by 5.
- 6sym has 6 inputs and 1 output. Its output is 1 if and only if the number of ones in the input pattern is 2, 3 or 4.
- nth_prime3_inc used to find primes with up to 3 binary digits.

The result of the formal analysis of these quantum circuits is given in Table[I]. The second column provides the number of gate in each circuit before adding SWAP gate, and the third column provides the total number of gates. The last column contains the success probability of these circuits if they were implemented using the elementary gates that have
been fabricated experimentally. In the following part we will perform thoroughly the analysis of quantum full adder.

A. Formal Verification and Modeling of Full Adder

In this section, we demonstrate the basic building of any quantum computing algorithm which is the quantum full adder, using elementary gates. In particular, our design is mainly inspired from the design presented in \[4\]. Since it has been physically proven that the qubits need to be adjacent in order to be applied to the same gate, we added two swap gates to exchange the qubits before feeding them to the Fredkin gate. This was not addressed in \[4\] and shows the practicality of our approach in modeling quantum circuits that are physically feasible to build. By identifying the limitation of the proposed design in literature when it is modeled using tensor product. For example, mathematically we cannot feed two components of a tensor product that are not adjacent to the same quantum gate. The circuit has four inputs; the two operands, the carry, and an extra input which is initialized to the state $|0\rangle$. We formally define the structure of the quantum full adder as follows:

**Definition 7 (Full Adder Circuit):**

\[
\text{FULL ADDER}(a_0, a_1, a_2, a_3, e_0, e_1, e_2, e_3, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \equiv (\forall b_0, b_1, b_2, b_3, c_0, c_1, c_2, d_1, d_2.
\]

\[
\text{CNOT2\_GATE}(a_0, b_0, b_1, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \land
\text{CNOT2\_GATE}(a_2, a_3, b_2, b_3, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \land
\text{CNOT2\_GATE}(b_1, b_2, c_1, c_2, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \land
\text{SWAP\_GATE}(b_0, c_1, c_0, d_1, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \land
\text{SWAP\_GATE}(c_2, b_3, d_2, e_3, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \land
\text{FREDKIN3\_GATE}(c_0, d_1, d_2, e_0, e_1, e_2, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj})
\]

Based on this definition, we formally verify the functionality of quantum full adder in the general case where we have the two input values to be added: $|x\rangle = x_1|0\rangle a_1 + x_2|1\rangle a_1$ and $|y\rangle = y_1|0\rangle a_2 + y_2|1\rangle a_2$, and the carry: $|z\rangle = z_1|0\rangle a_3 + z_2|1\rangle a_3$. We formally verify the result as follows:

**Theorem 6 (Full Adder):**

\[
\text{let constraints} = \text{FULL\_ADDER}(a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, \text{ten}, \text{LH}, \text{LV}, \text{m}_\text{proj}) \in
\text{let input} = \text{tensor} 4 (\lambda i. \text{if} \ i = 1 \text{then} (x_1 \text{LH} a_1 + x_2 \text{LV} a_1) \text{if} \ i = 2 \text{then} (y_1 \text{LH} a_2 + y_2 \text{LV} a_2) \text{else} \ i = 3 \text{then} (z_1 \text{LH} a_3 + z_2 \text{LV} a_3) \text{else} \text{LH} a_4) \in
\text{let output1} = \text{tensor} 4 (\lambda i. \text{if} \ i = 1 \text{then} \text{LH} b_1 \text{else} \ i = 2 \text{then} \text{LH} b_2 \text{else} \ i = 3 \text{then} \text{LH} b_3 \text{else} \text{LH} b_4) \in
\text{let output2} = \text{tensor} 4 (\lambda i. \text{if} \ i = 1 \text{then} \text{LV} b_1 \text{else} \ i = 2 \text{then} \text{LV} b_2 \text{else} \ i = 3 \text{then} \text{LV} b_3 \text{else} \text{LV} b_4) \in
\text{let output3} = \text{tensor} 4 (\lambda i. \text{if} \ i = 1 \text{then} \text{LV} b_1 \text{else} \ i = 2 \text{then} \text{LV} b_2 \text{else} \ i = 3 \text{then} \text{LV} b_3 \text{else} \text{LV} b_4) \in
\]

### Table I: HOL Lemmas

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemma1</td>
<td>$\text{tensor } m + n \text{ mode } = (\lambda y. (\text{tensor } m \text{ mode }) y) \star (\text{tensor } n (\lambda i. \text{ mode } (i + m))) (\lambda i. y(i + m))$</td>
</tr>
<tr>
<td>lemma2</td>
<td>$(\lambda y. (\text{tensor } m \text{ mode }) y) \star (\text{tensor } n \text{ (i + m))) = (\lambda i. \text{ mode } (i + m)) y(i + m)$</td>
</tr>
<tr>
<td>lemma3</td>
<td>$(\lambda i. (\text{ tensor } m \text{ mode } n))) i = (\lambda i. (\text{ tensor } n \text{ mode } m))) i$</td>
</tr>
<tr>
<td>lemma4</td>
<td>$(\lambda i. (\text{ tensor } m \text{ mode } n))) i = (\lambda i. (\text{ tensor } n \text{ mode } m))) i$</td>
</tr>
<tr>
<td>lemma5</td>
<td>$(\lambda i. (\text{ tensor } m \text{ with } n))) i = (\lambda i. (\text{ tensor } n \text{ with } m))) i$</td>
</tr>
<tr>
<td>lemma6</td>
<td>$(\lambda i. (\text{ tensor } m \text{ mode } n))) i = (\lambda i. (\text{ tensor } n \text{ mode } m))) i$</td>
</tr>
<tr>
<td>lemma7</td>
<td>$(\lambda i. (\text{ tensor } m \text{ mode } n))) i = (\lambda i. (\text{ tensor } n \text{ mode } m))) i$</td>
</tr>
</tbody>
</table>

### Table II: Formalized Quantum Circuits

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Qubits</th>
<th>Gates without SWAP</th>
<th>Total Gates</th>
<th>Success Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>nth_prime3_inc</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>$5.9 \times 10^{-6}$</td>
</tr>
<tr>
<td>ham3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>$9.5 \times 10^{-7}$</td>
</tr>
<tr>
<td>hwb4</td>
<td>4</td>
<td>12</td>
<td>22</td>
<td>$1.2 \times 10^{-29}$</td>
</tr>
<tr>
<td>full adder</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>$3.7 \times 10^{-9}$</td>
</tr>
<tr>
<td>mod5, the Grover’s oracle</td>
<td>5</td>
<td>8</td>
<td>18</td>
<td>$2 \times 10^{-28}$</td>
</tr>
<tr>
<td>2-4 dec</td>
<td>6</td>
<td>3</td>
<td>8</td>
<td>$8.6 \times 10^{-19}$</td>
</tr>
<tr>
<td>gf23mult</td>
<td>9</td>
<td>11</td>
<td>61</td>
<td>$1.7 \times 10^{-108}$</td>
</tr>
<tr>
<td>6sym</td>
<td>10</td>
<td>20</td>
<td>61</td>
<td>$1.7 \times 10^{-102}$</td>
</tr>
</tbody>
</table>
let output4 = tensor 4 (λi. if i = 1 then LH b1 else if i = 2 then LV b2 elseif i = 3 then LH b3 else LH b4) in
let output5 = tensor 4 (λi. if i = 1 then LH b1 else if i = 2 then LH b2 elseif i = 3 then LV b3 else LV b4) in
let output6 = tensor 4 (λi. if i = 1 then LV b1 else if i = 2 then LV b2 elseif i = 3 then LH b3 else LH b4) in
let output7 = tensor 4 (λi. if i = 1 then LV b1 else if i = 2 then LV b2 elseif i = 3 then LH b3 else LH b4) in
let output8 = tensor 4 (λi. if i = 1 then LH b1 else if i = 2 then LV b2 elseif i = 3 then LH b3 else LV b4) in

constraints ⇒ input = Cx((x1 pow 7)* (z1 * x1 * y1) % output1 + (z1 * x1 * y2) % output2
+ (z1 * x2 * y1) % output3 + (z1 * x2 * y2) % output4
+ (z2 * x1 * y1) % output5 + (z2 * x1 * y2) % output6
+ (z2 * x2 * y1) % output7 + (z2 * x2 * y2) % output8)

Here we have eight possible outputs from the adder, as the combinations of the three inputs gives 8 possibilities. Notice that the success probability of the quantum full adder is very low: (x111 pow 7). This completes the formal analysis of the quantum full adder.

VII. CONCLUSION AND DISCUSSION

In this paper, we presented an approach for the design automation of quantum circuits based on HOL Light theorem prover. In particular, we formalized a rich quantum gates library and developed a powerful automation procedure. In order to demonstrate the usability of the presented framework, we modeled and analyzed a benchmark quantum circuits. The analysis result proved the maturity of the approach and its capability to help in the advancement of CAD tools usage in the quantum domain. The low level modeling of quantum gates provides a crucial path to perform an effective optimization of quantum circuits as the case of Toffoli gate. One of the main features in the proposed approach is the capability to extract a quantum circuit success probability. The success probability can be considered as an optimization parameter when it comes to choose the effective design. To the best of our knowledge, we are the first to provide the success probabilities of the analyzed benchmark circuits in Section VI. We can notice that all the success probabilities given in Table II are very low. This is due to the fact that the only available optical implementation for the 2-qubit gates that have been fabricated are probabilistic. The rich mathematical foundation of our approach gives the opportunity to invent new quantum designs can either perform totally new functionalities or existing functionalities but in more efficient fashion in term of success probability. Compared to existing related works, the presented approach is complete (e.g., rich quantum gates library), and generic (i.e., the analysis is done at the quantum physics level), and can be considered as complimentary to the existing works in the synthesis and optimization of quantum circuits. It is important to note here that the underlying automation procedure does not depend on the quantum gates library and it is possible to add or remove gates and the automation procedure still perform the circuit analysis automatically using the available gates in the library.

The framework described in this paper can be expanded to cover the synthesis of functions to quantum circuits, and also the optimization of these circuits. An immediate future plan is to develop an optimization procedure to optimize the number of SWAP gates added to a quantum circuit in order to meet the adjacency principle.

REFERENCES


