https://doi.org/10.1007/s00165-019-00480-5
BCS © 2019
Formal Aspects of Computing (2019) 31: 321–351

Formal Aspects of Computing

A modeling and verification framework for optical quantum circuits

Sidi Mohamed Beillahi[1], Mohamed Yousri Mahmoud[2], and Sofiène Tahar[1]

[1]Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada.
[2]Department of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada.
e-mail: beillahi@ece.concordia.ca, myousri@uottawa.ca, tahar@ece.concordia.ca.

Abstract. Quantum computing systems promise to increase the capabilities for solving problems which classical computers cannot handle adequately, such as integers factorization. In this paper, we present a formal modeling and verification approach for optical quantum circuits, where we build a rich library of optical quantum gates and develop a proof strategy in higher-order logic to reason about optical quantum circuits automatically. The constructed library contains a variety of quantum gates ranging from 1-qubit to 3-qubit gates that are sufficient to model most existing quantum circuits. As real world applications, we present the formal analysis of several quantum circuits including quantum full adders 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. We show through several case studies the efficiency of the proposed framework in terms of the scalability and modularity.

Keywords: Formal methods; Interactive theorem proving; High-order logic; Quantum computing; Quantum optics.

1. Introduction

Quantum computers are expected to be employed in high computing domains and for secure communications (e.g., monetary transactions on the Internet) [BMK10]. In [KLM01], 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*). However, there are still many practical challenges that impede the realization of an efficient optical quantum computer such as: initialization of quantum bits (i.e., two-level quantum systems), measurements, decoherence (i.e., unwanted interaction between a quantum system and its environment), etc.

Correspondence and offprint requests to: Sofiène Tahar, E-mail: tahar@ece.concordia.ca

Quantum systems, like any other engineering systems, need Computer Aided Design (CAD) tools to facilitate their design and deployment in real life applications. These CAD tools help in the realization of new designs and the evaluation of their efficiency and security. However, due to the inherent complexity of quantum circuits, which rely on mathematical models of quantum physics principles, numerical simulations based CAD tools are not sufficient: the computation space increases exponentially with the size of the quantum circuit. 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., [BL04, VRM⁺03], where the simulations were performed at the gate level without modeling the physical elements of these gates which leads to a loss of accuracy. On the other hand, quantum circuits synthesis has been an active area of design automation research (e.g., [GFM10, SWM12]), where several methods were proposed for the synthesis of quantum circuits built using elementary gates at the behavioral level.[1] However, the physical design of the elementary gates was not tackled. This limits the underlying work to find the optimized circuit. For example, in [GFM10, SWM12], the Toffoli gate circuit was synthesized into five 2-qubit gates. However, it is possible to implement the Toffoli gate in quantum optics using only three 2-qubit gates as shown in [RRG07], by utilizing the photons polarization for the qubits.[2]

Higher-order logic (HOL) theorem proving [Har09] offers a rich mathematical foundation to fulfill the main requirements 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 interactive theorem prover [Har96] to handle the modeling and automated verification of quantum circuits. This is due to its rich support for multivariate calculus and Hilbert spaces theories [MAT13] which are essential to reason about quantum circuits.

In this framework, we are building a rich quantum gates library and developing a proof strategy to automate the analysis of quantum circuits constructed from the underlying library. In particular, we are considering the quantum optics implementation of quantum gates, rather than considering general high level behavioral gates description which is already covered in the existing work in the literature, e.g., [VRM⁺03, WS14, NWM⁺16]. Furthermore, we tackle the explicit mathematical modeling of tensor product and projection. We first formalize several gates including the Flip,[3] SWAP, Toffoli sign, Toffoli and Fredkin gates. These gates are widely used in building quantum circuits. Afterwards, we develop a proof strategy to automatically perform the analysis of any quantum circuits constructed using the gates library. Furthermore, we show that the developed framework is indeed modular and provides a scalable platform for efficient quantum circuits verification. Finally, we demonstrate the effectiveness of the proposed framework by showing several quantum circuits including Shor's algorithm, full adders, decoder, Grover's oracle, etc.

Related Work. There are several work in the literature for developing CAD tools for the analysis of quantum circuits. However, the usage of HOL based tools for the analysis of physical quantum circuits did not yet get much attention. In [MAT14, MPT15], the authors used the Hilbert-spaces library of HOL Light to formalize the flip (NOT) and controlled-not (CNOT) gates. In [BMT16a], the Hadamard, non-linear sign (NS), and controlled-phase (CZ) gates were formalized in HOL Light using the tensor product projection. However, the underlying gates are simple and cannot form a universal library. Another related work is [FGP13], 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 [MW95]. More recently, two theorem provers were employed for quantum programs [LLW⁺16] and protocols [BKN15] verification. In [LLW⁺16], the authors modeled and verified manually quantum programs using Isabelle/HOL and quantum Hoare logic [Yin11] by providing the invariants during the proof process. In [BKN15], the authors modeled and verified some quantum gates and a simple quantum protocol using Coq and the matrix theory. Both work cannot be used for the modeling and verification of an arbitrary quantum circuit. In [RPZ17a, RPZ17b], Coq is used to define a programming language denoted QWIRE to model quantum circuits and verify properties of these circuits using matrix calculus. The approach provides a Coq's library for matrix transformation and proof of soundness and safety of the semantics of QWIRE. The QWIRE language is built upon the (Quantum

[1] In the paper, we differentiate between the physical level, where we take to account the physical implementations (e.g, optics, ion-trap, solid-state,...) and the behavioral level, where the physical implementations are abstracted.

[2] Using our approach we are able to model and verify the optimized implementation of Toffoli given in [RRG07].

[3] The underlying technology of the flip gate is different than the one implemented in [MAT14] which is not compatible with the present modeling approach.

Random Access Memory) QRAM model [Kni96] with more formal structure, better operational semantics and type system. The QRAM approach assumes quantum bit and operation are perfect and does not consider a certain technology of implementation, whereas our approach is working at the physical implementation level. Hence, it is not possible to extract and compare the success probabilities of quantum circuits that are created using different optical elements.

In other less related work, several approaches addressing quantum circuits synthesis have been proposed at the behavioral level (e.g., [HSY+04, HSY+06, SBM06]). For example, in [HSY+04], the authors used model checking to synthesis quantum circuits based on a fixed set of 1-qubit and 2-qubits gates. Likewise, in [SBM06], the authors employed the Shannon expansion to decompose quantum functions in terms of CNOT gates. On the other hand, in [GWD+09] the authors used Boolean satisfiability (SAT) and SAT modulo theory (SMT) for the synthesis of Toffoli networks at the behavioral level. Even though these works provide suitable environments for quantum synthesis at the behavioral level, the verification part was not tackled. More recently, a new approach using matrices manipulations was presented in [NWM+16]. The authors developed an approach to model and synthesize quantum circuits using a new form of decision diagrams.

It is important to note that most of the existing quantum circuits behavioral models in the literature are not physically feasible due to the adjacency principle constraint between qubits [BBC+95]. This is because most of the existing works tackle quantum circuits at the behavioral level and do not consider a particular technology for implementing these circuits. The adjacency principle states that in order to apply a gate to a set of qubits, those qubits need to be adjacent (i.e., neighborhood to each other) [BBC+95]. 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 merit of considering low level modeling, as it helps us to know how far we are from implementing real world quantum computers. This makes our approach an important step in the modeling and designing process of quantum circuits that completes the mission of the existing techniques and helps to have more insights about real implementations and their constraints.

To the best of our knowledge, this is the first time a systematic formal low level modeling and automated verification of optical quantum circuits is tackled using mechanized interactive theorem prover. The source code of our formalization is available for download at [BM19] and can be utilized by other researchers and scientists. In this paper, we assume that the reader might not be familiar with the basic notions of quantum theory and computing, therefore, in Sects. 2 and 3, we review the necessary background about quantum optics and computing and the formalization of two optical elements.

The rest of the paper is organized as follows: We highlight our proposed framework in Sect. 4. In Sect. 5, we describe the formalization of tensor product, projection, and tensor product projection. Then, we present the HOL formalization of the quantum gates library in Sect. 6. In Sect. 7, we describe the flow of the automation procedure. Next, in Sect. 8, we present the results of the analysis of several quantum circuits. Finally, Sect. 9 concludes the paper.

2. Preliminary

2.1. Linear spaces and quantum mechanics

We define an inner product space over the complex field C using the tuple $(A, +, *, I)$, where A is a set of vectors, $+ : A \times A \longrightarrow A$ and $* : C \times A \longrightarrow A$ are two operators over A and an inner product function $I : A \times A \longrightarrow C$. Given a vector $X \in C$, we denote $\| X \| = \sqrt{I(X, X^*)} = \sqrt{\langle X \mid X \rangle}$, the norm of X where X^* is the complex conjugate of X. Given an inner product space $(A, +, *, I)$, a sequence $\{f_n\}_{n \in N}$ of elements $f_n \in A$ is a Cauchy sequence iff for every $\epsilon > 0$, there is some $N \geq 0$ such that:

$$I(f_m, f_n) < \epsilon \ \mathit{forall}\ m,\ \ n \in N.$$

We say $(A, +, *, I)$ is complete if every Cauchy sequence $\{f_n\}_{n \in N}$ converges to a unique limit:

$$\exists\ f \in A.\ \lim_{n \to \infty} I(f_n, f) < \epsilon$$

A Hilbert space H is defined as an inner product space $(A, +, *, I)$ that is complete.

Example 2.1 The set of all square-integrable complex-valued functions (i.e., $\int_{\infty}^{\infty} f(x)f^{*}(x)dx < \infty$) is a Hilbert space (denoted L^2 in the literature) [Pac74] with operators and inner product defined: $\forall f\ g \in L^2.\ +(f, g) : x \to f(x) + g(x), \star(k, f) : x \to k \star f(x)$, and $I(f, g) = \int_{\infty}^{\infty} f(x)g^{*}(x)dx$.

A quantum state is described as a complex-valued function (i.e., $\mid \psi\rangle : A \to C$[4]) and the set of all possible quantum states forms an L^2 Hilbert space.

Quantum operators are mapping functions between quantum states (i.e., $op : (A \to C) \longrightarrow (A \to C)$). The operators are used to either transform a quantum state, or measure one.

2.2. Quantum optics

Quantum optics particles (i.e., photons) are described by the L^2 Hilbert space of type $(real \to complex)$. In quantum literature, we call the quantum states that constitute the basis of the L^2 Hilbert space Fock states such that a Fock state denoted by $\mid n\rangle$ describes an optical field with n photons. Mathematically, Fock states form an orthonormal basis:

$$\langle m \mid n\rangle = 1\ if\ n = m;\ \ 0\ otherwise.$$

The two main operators that transform Fock states are: annihilation operator (annihilator) and creation operator (creator) which transform a given Fock state by decreasing and increasing the number of particles (e.g., photons) by 1, respectively:

$$\hat{a} \mid n\rangle = \sqrt{n} \mid n-1\rangle\ \ and\ \ \hat{a}^{\dagger} \mid n\rangle = \sqrt{n+1} \mid n+1\rangle\,.$$

Notice that a new definition of Fock state can be derived using the creation operator as follows:

$$\mid n\rangle = \frac{(\hat{a}^{\dagger})^{n} \mid 0\rangle}{n!}$$

where the quantum state $\mid 0\rangle$ is the vacuum state which does not contain any physical particle (i.e., photon).

To describe the quantum state ($\mid \Psi\rangle$) of multiple optical fields ($\mid \psi\rangle_i\ i \in 0 \cdots n$) we use the mathematical notion of tensor product of states, $\otimes$, which attaches the sub-states together, $\mid \Psi\rangle = \mid \psi\rangle_1 \otimes \mid \psi\rangle_2 \otimes \cdots \otimes \mid \psi\rangle_n : (A^n \to C)$, such that $\mid \Psi\rangle$ is the point-wise multiplication of $\mid \psi\rangle_i : A \to C\ i \in 0 \ldots n$:

$$\mid \Psi\rangle\,(y_1, \ldots, y_n) = (\mid \psi\rangle_1 \otimes \cdots \otimes \mid \psi\rangle_n)(y_1, \ldots, y_n) = \mid \psi\rangle_1\, y_1 * \cdots * \mid \psi\rangle_n\, y_n \qquad (1)$$

The new state $\mid \Psi\rangle$ describes all the sub-states ($\mid \psi\rangle_i\ \ i \in 0 \cdots n$) together.

Similarly, we define the tensor product of operators which act on quantum states that are tensor product of multiple quantum sub-states as: $Op = (op_1 \oplus \cdots \oplus op_n)$ such that:

$$Op\ (\mid \psi\rangle_1 \otimes \cdots \otimes \mid \psi\rangle_n) = (op_1 \oplus \cdots \oplus op_n)(\mid \psi\rangle_1 \otimes \cdots \otimes \mid \psi\rangle_n) = (op_1 \mid \psi\rangle_1 \otimes \cdots \otimes op_n \mid \psi\rangle_n) \qquad (2)$$

2.3. Linear optical quantum computing

Quantum optics is considered as one of the rich and promising approaches under investigation for realizing quantum computers [KLM01]. This is because photons decohere slowly, move quickly (at the speed of light), and can be experimented with at room temperature.

In [KLM01] the authors showed that combining single-photon sources, single-photon detectors, and linear optics would suffice to implement efficient quantum computation. Moreover, most of existing universal quantum gate architectures in the literature are built using only linear-optical elements. A linear optical element is such that optical modes transformation can be described by matrices u and v, which transform the modes linearly, such that,

$$\hat{a}_j^{\dagger} \to \sum_k u_{kj}\, \hat{a}_k^{\dagger}\ \ and\ \ \hat{a}_j \to \sum_k v_{kj}\, \hat{a}_k$$

In literature, the above transformation is named Bogoliubov transformation [Bog47]. The most widely employed linear optics elements are beam splitter and phase shifter.

[4] A designates an abstract type and C designates the type of complex numbers.

2.4. HOL light theorem proving

We provide a couple of examples to illustrate how definitions and theorems are written in HOL Light:

We define a function `f_min` that takes two real values as parameters and returns the minimum one, the corresponding HOL expression of such a function is as follows:

$$\vdash \texttt{f\_min} = \lambda\,(\texttt{x}:\texttt{real})\,(\texttt{y}:\texttt{real}).\ \texttt{if x} > \texttt{y then y else x} \tag{3}$$

where λ is the lambda abstraction in HOL Light for defining the function. `f_min x y` returns the minimum of the given parameters `x` and `y` which are of types *real*. The type of `f_min` is `real` $\rightarrow$ `real` $\rightarrow$ `real`.

Another way to define a function in HOL Light is by using a predicate which is used when the concrete implementation of a function is not available but rather its specifications. For instance, in below we show *is_even* predicate which accepts an integer parameter and returns true if this integer is even and false otherwise:

$$\vdash \texttt{is\_even}\,(\texttt{n}:\texttt{num}) \Leftrightarrow (\texttt{if}\ (\texttt{n rem 2} = \texttt{0})\ \texttt{then T else F}) \tag{4}$$

Here, we are using the equivalence symbol $\Leftrightarrow$ to define `is_even` since it is a predicate (i.e., the return value is Boolean) not a function (can return any type). Thus, the type of `is_even` is `int` $\rightarrow$ `bool`. Also, we use a predefined HOL Light function `rem` which returns the remainder of integer division. The style of the underlying definition is mostly employed when the concrete implementation of a function is not available but rather its specifications. In our formalization of quantum gates we use both styles.

3. Formalization of optical elements

In this section, we present the HOL formalization of the two optical elements: beam splitter and phase shifter. In the formalization, we define the transformation of an optical element using the relations between the annihilator and creator operators of the input modes and the annihilator and creator operators of the output modes of the given optical element.

We provide in Table 1, the HOL definitions and types for some quantum notions and symbols that are used in our formalization.[5]

3.1. Phase shifter

The phase shifter provides a phase shift in a given optical mode: $\hat{a}^{\dagger}_{o1} = e^{ii\theta}\,\hat{a}^{\dagger}_{i1}$. A phase shifter ($P_\theta$, θ is the angle of the phase shifter) is a passive linear optical element. Phase shifter transformation is associated with a unitary operator described by: $U(P_\theta) = e^{ii\phi}$. Physically, a phase shifter is a slab of transparent material with an index of refraction that is different from that of free space. The phase shifter transformation is defined in HOL as follows:

Definition 3.1 (Phase shifter)
1 $\texttt{phase\_shifter}(\texttt{ten}, (\theta:\texttt{real}), (\texttt{i1}:\texttt{sm}), (\texttt{m1}:\texttt{natural}), (\texttt{o1}:\texttt{sm}), (\texttt{m2}:\texttt{natural})) \Longleftrightarrow$
2 $\texttt{is\_sm i1} \wedge \texttt{is\_sm o1} \wedge \texttt{w i1} = \texttt{w o1} \wedge \texttt{vac i1} = \texttt{vac o1}\ \wedge$
3 $\texttt{pos ten (cr i1) m1} = \texttt{e}^{(-\texttt{ii}\times\theta)}\ \%\ \texttt{pos ten (cr o1) m2}\ \wedge$
4 $\texttt{pos ten (anh i1) m1} = \texttt{e}^{(\texttt{ii}\times\theta)}\ \%\ \texttt{pos ten (anh o1) m2}$

where `ten` and `pos` are the tensor product of operators and the positioning operator, respectively, defined in Table 1. `anh x` and `cr x` designate the annihilator and creator operators for the mode x, respectively. `i1` and `o1` are the input mode and output mode, respectively. Notice that the formal definition of the phase shifter relates the input operators (`anh i1` and `cr i1`) in terms of the output operators (`anh o1` and `cr o1`), see Lines 2 and 3. In Line 1, the parameters {`m1`, `m2`} define the order of each mode in the tensor product of operators. Line 1 ensures that the two modes (input and output) are proper optical single modes, and working with the same frequency and vacuum state (i.e., the state of zero photons).

[5] For the detailed HOL definitions of the mathematics for quantum physics and optics and related theorems, we refer the reader to [M15]

Table 1. Terms and types in the HOL formalization

HOL Term and type	Description
`bqs` : `real` $\rightarrow$ `complex`	Type of optical quantum state
`qop` : `bqs` $\rightarrow$ `bqs`	Type of quantum operator
`sm` : (`qop#qop#real#bqs`)	Type of optical single mode which is composed of the tuple (`anh`, `cr`, `f`, `vac`), annihilator `anh`, creator `cr`, resonance frequency `f`, and vacuum state `vac`
`is_sm` : `sm` $\rightarrow$ `bool`	A predicate to indicate that a given single mode `sm` of type `sm` does indeed satisfy the properties of quantum single mode
`anh` : `sm` $\rightarrow$ `qop`	The annihilator operator such that `anh` (`sm` = (`anh`, `cr`, `f`, `vac`)) = `anh`
`cr` : `sm` $\rightarrow$ `qop`	The creator operator such that `cr` (`sm` = (`anh`, `cr`, `f`, `vac`)) = `cr`
`vac` : `sm` $\rightarrow$ `bqs`	The vacuum state such that `vac` (`sm` = (`anh`, `cr`, `f`, `vac`)) = `vac`
`fock` : `sm` $\rightarrow$ `natural` $\rightarrow$ `bqs`	Designates a Fock state such that `fock sm n` designates the Fock state in the optical single mode `sm` with `n` photons and it is defined using the creator operator. In the case `n` = `0`, we require that `fock sm 0` = `vac sm`, Fock state with zero photons to designate the vacuum state
`tensor` : `natural` $\rightarrow$ `bqs`$^{\mathtt{N}}$ $\rightarrow$ (`real`$^{\mathtt{N}}$ $\rightarrow$ `complex`)	Tensor product which accepts a vector of quantum states and its size and returns the tensor product of states
`ten` : `qop`$^{\mathtt{N}}$ $\rightarrow$ (`real`$^{\mathtt{N}}$ $\rightarrow$ `complex`) $\rightarrow$ (`real`$^{\mathtt{N}}$ $\rightarrow$ `complex`)	Operators tensor product which accepts a vector of operators and returns tensor product of operators which accepts and returns tensor product of states
`pos` : `ten` $\rightarrow$ `qop` $\rightarrow$ `natural` $\rightarrow$ (`real`$^{\mathtt{N}}$ $\rightarrow$ `complex`)	Designates the positioning operator such that `pos ten op m` = `ten` (λ `i`. `if(i = n) then op else I`) places a given operator `op` in a specific mode n and leaves the other modes with the identity operator `I`

3.1.1. *Mirror*

By choosing the shifting angle θ to be $\frac{\pi}{2}$, the phase shifter operates as a mirror. Thus, the optical mirror is formalized in HOL as follows:

Definition 3.2 (Mirror)
`mirror(ten, (i1 : sm), (m1 : natural), (o1 : sm), (m2 : natural))` $\Longleftrightarrow$
`phase_shifter(ten,` $\frac{\pi}{2}$`, (i1 : sm), (m1 : natural), (o1 : sm), (m2 : natural))`

3.2. Beam splitter

Beam splitter is a two modes passive linear optical element that consists of a semi reflective mirror. When a light beam falls on this mirror, a part will be reflected and a part will be transmitted. Mathematically, a beam splitter has two parameters θ and φ, where $\cos\theta$ and $\sin\theta$ are the probability amplitudes and φ is the relative phase. Let a_{i1} and a_{i2} denote the two incoming modes of the beam splitter by and a_{o1} and a_{o2} denote the outgoing modes, as shown in Fig. 1 then the beam splitter transformation of the annihilator and creator is:

$$\begin{pmatrix} \hat{a}_{o1}^{\dagger} \\ \hat{a}_{o2}^{\dagger} \end{pmatrix} = \begin{pmatrix} \cos\theta & ii e^{-ii\varphi}\sin\theta \\ ii e^{ii\varphi}\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \hat{a}_{i1}^{\dagger} \\ \hat{a}_{i2}^{\dagger} \end{pmatrix}$$

The commonly used beam splitter is the one of relative phase $\varphi = 0$. Accordingly, the beam splitter transformation is defined in HOL as follows:

Definition 3.3 (Beam splitter)
```
1 is_beam_splitter((p1 : complex), p2, p3, p4, ten, (i1 : sm), (m1 : natural), i2, m2, o1, m3, o2, m4) ⟺
2 is_sm i1 ∧ is_sm i2 ∧ is_sm o1 ∧ is_sm o2 ∧
3 w i1 = w i2 ∧ w i2 = w o1 ∧ w o1 = w o2 ∧
4 vac i1 = vac i2 ∧ vac i2 = vac o1 ∧ vac o1 = vac o2 ∧
5 pos ten (anh i1) m1 = p1 % pos ten (anh o1) m3  + p2 % pos ten (anh o2) m4 ∧
6 pos ten (anh i2) m2 = p3 % pos ten (anh o1) m3  + p4 % pos ten (anh o2) m4 ∧
```

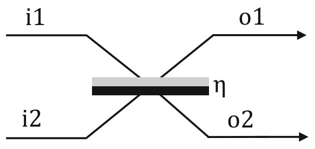


Fig. 1. Schematics of beam splitter

```
7 pos ten (cr i1) m1 = p1* % pos ten (cr o1) m3  + p2* % pos ten (cr o2) m4 ∧
8 pos ten (cr i2) m2  = p3* % pos ten (cr o1) m3  + p4* % pos ten (cr o2) m4
```

where `i1`, `i2` and `o1`, `o2` represent the beam splitter inputs and outputs modes, respectively. `m1` and `m2` (resp. `m3` and `m4`) represent the order of the two beam splitter input (resp. output) modes within the inputs (resp. outputs) tensor vector of operators. Similar to the phase shifter, the formal definition of the beam splitter relates the inputs annihilator and creator operators in terms of the outputs annihilator and creator operators (see Lines 5–8). The parameters {p1,p2,p3,p4} are complex numbers that are the inverse of the beam splitter matrix. Lines 2, 3, and 4 ensure that the four modes are proper single modes and working with the same frequency and vacuum state. More details about the formalization of the basics notions of quantum mechanics and optics can be found in [M15].

4. Proposed formal analysis framework

The proposed framework for the analysis of quantum circuits using HOL theorem proving is depicted in Fig. 2. Our ultimate goal is to evaluate the output for a given input and circuit, supplemented with the success probability of this output. The first step is to formalize the notion of tensor product, projection and tensor product projection which form the mathematical foundation of the framework. The second step is to build a library of quantum gates models in HOL on top of an existing library of optics elements [MAT14, MPT15]. The gates' library includes 1-qubit, 2-qubit and 3-qubit gates, which would enable the modeling of a wide variety of quantum circuits. The next step is to take a quantum circuit netlist and a set of inputs and establish a HOL theorem for the input–output relations and success probabilities. To facilitate the proof of the HOL theorem automatically, without the need of user guidance, we develop a proof strategy that takes the quantum circuit netlist and the respective inputs and builds the required tactics[6] that automate the proof. The proof strategy first reads the quantum circuit netlist and generates a matrix that captures the circuit structure. The proof strategy uses the extracted matrix to generate the required folding/unfolding lemmas: these lemmas are responsible for preparing the quantum wires (inputs and outputs) in the correct order before and after they pass through a set of parallel gates. Finally, using a set of simplification rules, we construct the final automation tactics to conduct the formal proof of the quantum circuit properties.

5. Formalization of tensor product and projection

In Sect. 2, we gave an introduction to quantum optics where we showed the importance of tensor product for the analysis of quantum states in multi-modes. This section covers in detail the higher-order logic formalization of the tensor product along with linear projection. Furthermore, in the last part of this section we combine the tensor product and linear projection to obtain the tensor product projection.

5.1. Formalization of tensor product

As described in the previous section, the tensor product for n optical beams is the point-wise multiplication of n complex-valued functions. We formally define it in HOL, recursively, as follows:

[6] A tactic is a function written in OCaml which partially automates the process of theorem proving in HOL Light

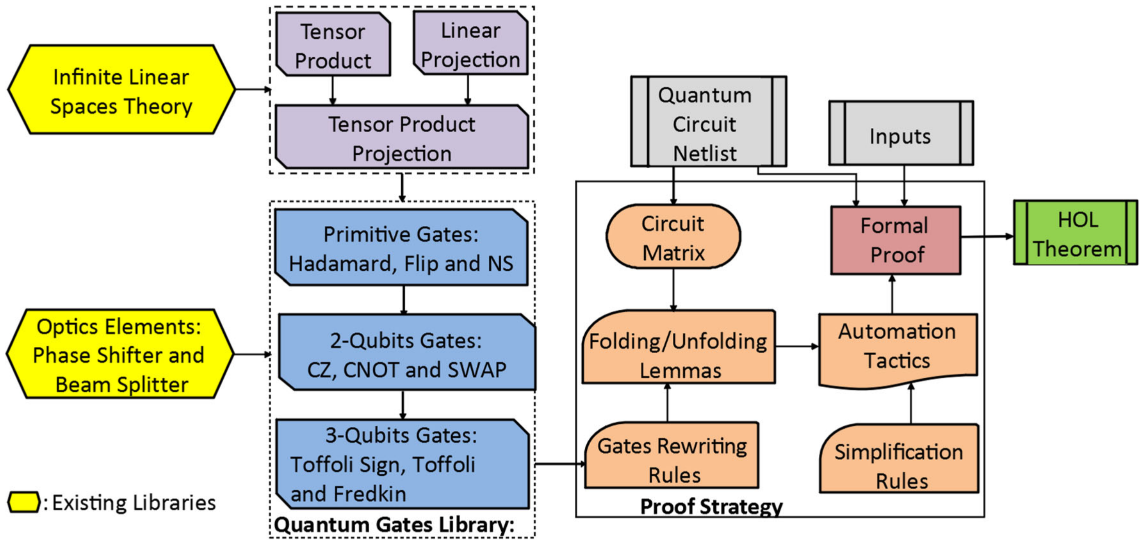


Fig. 2. Proposed analysis framework for quantum circuits

Definition 5.1 (Tensor product)
$\vdash \texttt{tensor 0 (mode : bqs}^{\texttt{N}}\texttt{)} = (\lambda(\texttt{y : real}^{\texttt{N}}).\ 1)\ \wedge$
$\quad \texttt{tensor (n : natural) + 1 (mode : bqs}^{\texttt{N}}\texttt{)} = (\lambda(\texttt{y : real}^{\texttt{N}}).\ ((\texttt{tensor n mode) y}) * (\texttt{mode(n + 1) y(n + 1)}))$

where the symbol λ is the Lambda abstraction. Note that mode is a vector of size n that contains n modes. The basic case of zero mode $\texttt{n} = 0$ is a trivial case; it is a constant function (i.e., $y \rightarrow 1$) and it guarantees a terminating definition. tensor is of type $\texttt{natural} \rightarrow \texttt{bqs}^{\texttt{N}} \rightarrow (\texttt{real}^{\texttt{N}} \rightarrow \texttt{complex})$. Mathematically, to validate that the underlying tensor product is well defined, we should prove the bi-linearity property:

Theorem 5.1 (Tensor: Bi-Linearity)
$\vdash \texttt{n+1} \leq \texttt{N} \wedge \texttt{k} \leq \texttt{n+1} \wedge 0 < \texttt{k} \wedge \texttt{mode(k)} = (\texttt{a : complex})\ \%\ (\texttt{x : bqs}) \implies$
$\quad \texttt{tensor n+1 mode} = \texttt{a}\ \%\ \texttt{tensor n+1 (}\lambda\,\texttt{i. if i=k then x else mode(i))}$
$\vdash \texttt{n+1} \leq \texttt{N} \wedge \texttt{k} \leq \texttt{n+1} \wedge 0 < \texttt{k} \wedge \texttt{mode(k)} = \texttt{x}_1\texttt{+x}_2 \implies$
$\quad \texttt{tensor n+1 mode} = \texttt{tensor n+1 (}\lambda\,\texttt{i. if i=k then x}_1\texttt{ else mode(i))} +$
$\quad \texttt{tensor n+1 (}\lambda\,\texttt{i. if i=k then x}_2\texttt{ else mode(i))}$

where the symbol % denotes the scalar multiplication. The two assumptions $k \leq n{+}1$ and $0 < k$ ensure that the element k is indeed part of the tensor. The assumption $\texttt{mode}(k) = a\ \%\ x$ ($\texttt{mode}(k) = x_1{+}x_2$) provides a linear decomposition of the k^{th} element of the tensor product as multiplication of scalar and vector (summation of two vectors). With the two parts of Theorem 5.1 we have verified that our definition is a tensor product. The proof steps for the two theorems are based on using induction where the base case is trivial and in the inductive step we use the lemma $\texttt{k} \leq \texttt{n+2} \iff (\texttt{k} \leq \texttt{n+1} \vee \texttt{k} = \texttt{n+2})$ then using the induction hypothesis for the first case and the definition of tensor product for the second case. With these properties of bi-linearity, we have validated our tensor product definition.

An important property in the manipulation of tensor product is when we have a tensor product $v_1 \otimes \cdots \otimes v_m \otimes u_1 \otimes \cdots \otimes u_n$ constructed out of two elementary tensors $((v_1 \otimes \cdots \otimes v_m)$ and $(u_1 \otimes \cdots \otimes u_n))$. In this case, the property states that $v_1 \otimes \cdots \otimes v_m \otimes u_1 \otimes \cdots \otimes u_n$ can be written in the form $(v_1 \otimes \cdots \otimes v_m) \otimes (u_1 \otimes \cdots \otimes u_n)$.

Theorem 5.2 (Tensor: Multiplication)
$\vdash \texttt{tensor m+n (mode : bqs}^{\texttt{N}}\texttt{)} =$
$\quad (\lambda(\texttt{y : real}^{\texttt{N}}).\ ((\texttt{tensor m mode) y}) * (\texttt{tensor n (}\lambda(\texttt{i : natural}).\ \texttt{mode(i+m))) (}\lambda\,\texttt{i. y(i+m)))})$

Here, $\texttt{tensor m mode}$ and $\texttt{tensor n} (\lambda\,\texttt{i. mode(i + m)})$ represent the elementary tensors $(v_1 \otimes \cdots \otimes v_m)$ and $(u_1 \otimes \cdots \otimes u_n)$, respectively. A typical usage of this theorem is to separate elementary tensors for the sake of conducting quantum transformations over the two elementary tensors independently from each other. Then using the same theorem, we return back to the initial tensor. The proof of this theorem is done using induction on the size of the second tensor product (i.e., n).

5.2. Formalization of linear projection

In linear algebra, a projection is a linear transformation p from a vector space to itself that maintains the idempotent property; $p^2 = p$. In the quantum context, for a pure state $\mid \psi \rangle$, the projection is defined as $p = \mid \psi \rangle \langle \psi \mid$ which is a self-adjoint and linear transformation. In particular, for a quantum circuit design, the expected circuit output is the projection of all possible outputs over the appropriate Fock states. For example, let us consider the state $\mid \phi \rangle = \frac{1}{3} \mid n \rangle + \frac{1}{3} \mid n\text{-}1 \rangle + \frac{1}{3} \mid n{+}1 \rangle$ and the projection $p_n = \mid n \rangle \langle n \mid$. The result of the projection of $\mid \phi \rangle$ is $p_n(\mid \phi \rangle) = \mid n \rangle \langle n \mid (\frac{1}{3} \mid n \rangle + \frac{1}{3} \mid n\text{-}1 \rangle + \frac{1}{3} \mid n{+}1 \rangle) = \frac{1}{3} \mid n \rangle$, because the Fock states form an orthonormal basis. Therefore, we define the projection on Fock states as follows:

Definition 5.2 (Linear Projection)
$\vdash \forall\ \mathtt{x}.\ (\mathtt{proj}\ (\mid \mathtt{n} \rangle_{\mathtt{sm}} : \mathtt{bqs}))\ (\mid \mathtt{x} \rangle : \mathtt{bqs}) = \langle \mathtt{n} \mid \mathtt{x} \rangle\ \ \%\ \ \mid \mathtt{n} \rangle_{\mathtt{sm}}$

where proj $\mid \mathtt{n} \rangle_{\mathtt{sm}}$ is the quantum linear projection over the Fock state and accepts as parameter x. proj is of type bqs $\rightarrow$ bqs $\rightarrow$ bqs. A quantum linear projection should meet the three mathematical requirements which are linearity, idempotent, and self-adjoint properties. We have formally proven these properties in HOL Light. The HOL theorem for the linearity of the projection is as follows..

Theorem 5.3 (Projection: Linearity)
$\vdash \mathtt{is\_sm\ sm} \Rightarrow \forall\ (\mathtt{x} : \mathtt{bqs})\ (\mathtt{y} : \mathtt{bqs})\ (\mathtt{a1} : \mathtt{complex})\ (\mathtt{a2} : \mathtt{complex}).$
$\quad (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ (\mathtt{a1}\%\mathtt{x} + \mathtt{a2}\%\mathtt{y}) = \mathtt{a1}\ \%\ (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ \mathtt{x} + \mathtt{a2}\%(\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ \mathtt{y}$

where the assumption is_sm sm is to maintain that the beam *sm* is indeed a quantum single mode beam and that it meets all the requirements [MPT15]. In what follows, we show the projection idempotency property.

Theorem 5.4 (Projection: Idempotency)
$\vdash \mathtt{is\_sm\ sm} \Longrightarrow \forall \mathtt{x}.\ (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ ((\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ \mathtt{x}) = (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ \mathtt{x}$

where $(\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ ((\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}})\ \mathtt{x})$ is the application of the projection proj $\mid \mathtt{n} \rangle_{\mathtt{sm}}$ twice on x. Next, we show the property of self-adjoint for the projection operator (i.e., $\langle p(u) \mid v \rangle = \langle u \mid p(v) \rangle$).

Theorem 5.5 (Projection: Self-Adjoint)
$\vdash \mathtt{is\_sm\ sm} \Longrightarrow \forall\ \mathtt{x\ y}.\ (\mathtt{x} : \mathtt{bqs}), (\mathtt{y} : \mathtt{bqs}) \in \mathtt{sq\_integrable} \Longrightarrow$
$\quad \mathtt{r\_inprod\ x}\ (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}}\ \mathtt{y}) = \mathtt{r\_inprod}\ (\mathtt{proj}\ \mid \mathtt{n} \rangle_{\mathtt{sm}}\ \mathtt{x})\ \mathtt{y}$

where sq_integrable is the linear inner product space formed by the square integrable functions space and Lebesgue integral. The proof of the three theorems is based on conjugate symmetry (i.e., $\langle \mathtt{x} \mid \mathtt{y} \rangle = \langle \mathtt{y} \mid \mathtt{x} \rangle^*$) and linearity of inner product.

5.3. Formalization of tensor product projection

In this section, we combine the tensor product for multi-mode and linear projection for single-mode together. We do this in order to obtain the tensor product projection, or in other words the multi-mode projection. In some realization of quantum optics, the quantum gates are implemented using *ancilla* resources. These are extra qubits that have a secondary role in a computation and are used for detecting the correct output [KMN+07]. During the design process of a quantum circuit, the state of the ancilla is measured after it leaves the circuit using a detector. The correct output is known to have been produced whenever the detector registers the expected ancilla. In our formalization, we implement the design process of detecting the expected ancillas in the output ports of a quantum circuit as the tensor product projection of the circuit outputs. By doing this, we eliminate the undesirable outputs and keep only the "correct" ones. In addition, we obtain the projected state multiplied by a scalar value which is the success probability of the circuit, or the probability in which we detect the expected ancilla. To the best of our knowledge, this is the first time tensor product projection is used as a mathematical analysis tool for quantum optics detection. We define the projection of multi-mode over multi-mode as follows:

Definition 5.3 (Tensor Projection)
$\vdash \mathtt{is\_tensor\_proj\ m\_proj} \iff \forall\ \mathtt{mode1\ mode2\ n}.$
$\quad \mathtt{is\_linear\_cop}\ (\mathtt{m\_proj}\ (\mathtt{tensor\ n\ mode1})) \wedge$
$\quad \mathtt{m\_proj}\,((\mathtt{tensor\ n\ mode1}) : (\mathtt{real}^{\mathtt{N}} \rightarrow \mathtt{complex}))\,(\mathtt{tensor\ n\ mode2}) =$
$\quad \mathtt{tensor\ n}\ (\lambda\, \mathtt{i}.((\mathtt{proj\ mode1(i)})\ \mathtt{mode2(i)}))$

where `m_proj tensor n mode1` is a linear projection operator defined over the multi-mode state `tensor n mode1`, and takes as parameter `tensor n mode2` which is the projected multi-mode state. The projection produces the state: `tensor n (λ i. ((proj (mode1(i))) (mode2(i))))` which is the tensor of the projection of each single-mode state. `m_proj` is of type $(\texttt{real}^{\texttt{N}} \rightarrow \texttt{complex}) \rightarrow (\texttt{real}^{\texttt{N}} \rightarrow \texttt{complex}) \rightarrow (\texttt{real}^{\texttt{N}} \rightarrow \texttt{complex})$. The function `is_linear_cop op` ensures that the operator `op` is indeed a linear operator. Using this definition, we prove a crucial property in the analysis of quantum circuits, which states that $(p_1 \otimes \cdots \otimes p_n)(u_1 \otimes \cdots \otimes u_n) = p_1(u_1) \otimes \cdots \otimes p_n(u_n)$.

Theorem 5.6 (Tensor Projection: Multiplication)

```
⊢ is_tensor_proj m_proj ∧ 1 ≤ n ⟹
  (m_proj tensor m+n mode1) tensor m+n mode2 =
  (λ y. ((m_proj tensor m mode1) tensor m mode2) y ∗
  (m_proj tensor n (λ i.mode1(i+m)) tensor n (λ i. mode2(i+m))) (λ i. y(i+m)))
```

The verification of this theorem is based on Theorem 5.2. This property is very useful when projecting multi-mode state which is applied to parallel quantum gates as the case for the controlled-phase gate (see Fig. 6 where the multi-mode state $|\, b(1), b(2), b(3), d(1), d(2), d(3)\rangle$ is fed to the two parallel NS gates). Using the tensor product lemma $v_1 \otimes \cdots \otimes 0 \otimes \cdots \otimes v_n = 0$, we prove the following property.

Theorem 5.7 (Tensor Projection: Fock States)

```
⊢ is_tensor_proj m_proj ∧ 0 < k ∧ mode1(k) = | m1⟩_sm ∧ mode2(k) = | m2⟩_sm ∧
  m1 ≠ m2 ∧ is_sm sm ∧ k ≤ n+1 ⟹ (m_proj tensor n+1 mode1) tensor n+1 mode2 = 0
```

This theorem is very important for the measurement of photons as it indicates that for two multi-mode states, where in the first state, the single mode k contains the Fock state $|\, m1\rangle$ (i.e., $|\, mode1(1), \ldots, m1, \ldots, mode1(n)\rangle$) and in the second state, the single mode k contains the Fock state $|\, m2\rangle$ (i.e., $|\, mode2(1), \ldots, m2, \ldots, mode2(n)\rangle$). If $m1$ and $m2$ are different ($m1$ and $m2$ describe the number of photons in each Fock state), then the projection of the first multi-mode state over the other is zero (the zero constant function).

We have covered the required mathematical tools for the formal modeling and analysis of optical quantum gates and circuits. In particular, we have covered the notions of tensor product, linear projection, tensor product projection, and have proved several important theorems related to these three mathematical notions. To the best of our knowledge, this is the first time a systematic formalization of tensor product, linear projection, and tensor product projection is tackled in the context of quantum optics circuits analysis. In addition, our mathematics formalization is general and can be used in other quantum circuits analysis. In the following section, we will utilize the developed mathematical foundation to formalize a variety of quantum gates.

6. Formalization of quantum gates library

In this section, we build upon the mathematical foundation to formally model nine important quantum gates, namely Hadamard, flip, non-linear sign, controlled-not, controlled-phase, SWAP, Toffoli sign, Toffoli, and Fredkin gates. The formalized set of quantum gates covers 1-qubit, 2-qubit, and 3-qubit gates and includes a new implementation for the flip gate based on a single photon source. Another interesting gate is the Toffoli sign gate. This is taken from [RRG07] and has three inputs, where one input is qutrit (i.e., has three quantum states) and the rest are qubits. Due to this, it is impossible to model the Toffoli sign using a Boolean model of the qubits (i.e., only supporting two states). Therefore, prior formal techniques could not model the Toffoli sign. Based on our Toffoli sign gate formalisation, we can model an optimal implementation of the Toffoli gate.

6.1. Quantum bit

A logical qubit is represented by a quantum system with two basis states which are $|\, 0\rangle_L$ and $|\, 1\rangle_L$. In the literature of quantum optics, the qubit can be realized through different configuration of the optical light beams weather using coherent light or single photon beam. In our formalization, we are using single photon approach for realizing the qubit which is called dual-rail qubit [KMN+07]. The qubit state describes the possibility of finding a given single photon in one of two different optical modes.

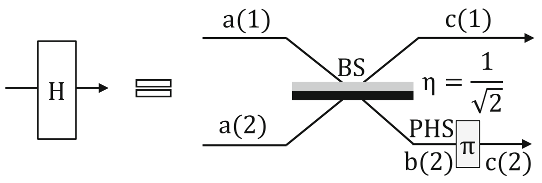


Fig. 3. Schematics of Hadamard gate

Thus, the two states of the qubit are represented by the internal polarization degree of freedom of a single photon. In particular, the logical states described as follows: $|\,0\rangle_L = |\,1\rangle \otimes |\,0\rangle \equiv |\,1, 0\rangle$ and $|\,1\rangle_L = |\,0\rangle \otimes |\,1\rangle \equiv |\,0, 1\rangle$ where the logical zero state represents the state such that the first optical mode contains a single photon and the second mode is in the vacuum state and the inverse for the logical one state.

6.2. Hadamard gate

The Hadamard gate [NC10] is an 1-qubit universal gate which exploits quantum superposition to create a new state, where the qubit logical states $|\,0\rangle$ and $|\,1\rangle$ are mapped to two superpositions of $|\,0\rangle$ and $|\,1\rangle$ with equal weight. For example, if the input is $|\,\phi\rangle_{input} = \alpha\,|\,0\rangle + \beta\,|\,1\rangle$, then the output is $|\,\phi\rangle_{output} = \alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Hadamard gates are usually used to initialize the quantum states of a circuit or to add random information to a quantum circuit. The authors in [PMO09] implemented Shor's algorithm for factorization of the number 15 using six Hadamard gates in a photonics chip by employing the quantum optics single photon technology. In this section, we present the formalization of the Hadamard gate in HOL Light as an example of a single qubit gate that can be constructed by using a beam splitter (BS) and a phase shifter (PHS) together. The dual-rail representation is used to describe the qubit. The gate circuit is shown in Fig. 3 and is composed of a beam splitter with a probability amplitude $\eta = \frac{1}{\sqrt{2}}$ and a relative phase $\varphi = 0$, and a phase shifter of angle $\theta = \pi$. The formal definition of the gate structure is as follows:

Definition 6.1 (Hadamard Gate)
$\vdash$ HADAMARD((in : sm), (out : sm), ten, (LH : sm $\rightarrow$ bqs), LV) $\iff$ ($\exists$ (a : sm$^{\mathtt{N}}$)(b : sm$^{\mathtt{N}}$)(c : sm$^{\mathtt{N}}$).
is_beam_splitter($\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$, ten, a(1), 1, a(2), 2, c(1), 1, b(2), 2) $\wedge$
phase_shifter(ten, π, b(2), 2, c(2), 2) $\wedge$ Hadamard_In_Output(in, out, a, c, LH, LV))

where LH and LV are employed to describe the representation of qubits using the photon vertical or horizontal polarization. LH and LV are both of type sm $\rightarrow$ bqs Here, LV in (resp., LH in) represents a vertically (resp., horizontally) polarized photon in the single mode a(1) which describes the qubit in the state $|\,1\rangle$ (resp., $|\,0\rangle$). HADAMARD takes as parameters all gate input/output ports (in, out), and the tensor product operator $tens$. As shown in Fig. 3 the optical implementation of Hadamard gate has two optical inputs to realize a single logical qubit input. Thus, $|\,0\rangle_L \equiv LH\ in \equiv |\,1, 0\rangle_a$ and $|\,1\rangle_L \equiv LV\ in \equiv |\,0, 1\rangle_a$. In order to make use of this gate in quantum circuits, where the computation is at the qubit level without getting to the detail of the qubit representation, we developed an input/output behavioral description for the Hadamard gate Hadamard_In_Outputs(in, out, a, c, LH, LV) [BMT16b]. Using the above definition, we formally verify the result of applying the Hadamard gate on the two possible inputs.

Theorem 6.1 (Inputs: $|\,0, 1\rangle_a \equiv LV\ in$ and $|\,1, 0\rangle_a \equiv LH\ in$)
$\vdash$ HADAMARD((in : sm), out, ten, LH, LV) $\Longrightarrow$
tensor 1 (λ i. LV in) = $\frac{1}{\sqrt{2}}$ % tensor 1 (λ i. LH out) $-$ $\frac{1}{\sqrt{2}}$ % tensor 1 (λ i. LV out) $\wedge$
tensor 1 (λ i. LH in) = $\frac{1}{\sqrt{2}}$ % tensor 1 (λ i. LV out) + $\frac{1}{\sqrt{2}}$ % tensor 1 (λ i. LV out)

Notice that we did not use the projection because we do not employ ancilla, therefore, there will be no detection required. What you have actually done, it seems to me, is to produce a new design somehow, and then used your formalization to prove that this design is correct.

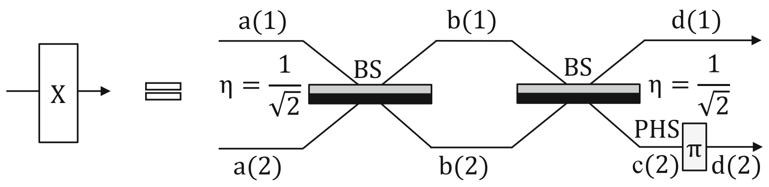


Fig. 4. Single photon source design of flip gate

6.3. 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 a coherent light source [RGM+03]. In the course of formalizing the library of quantum gates that are implemented using a single photon source, we discovered a *new* design of the flip gate using a single photons source. Using our formalization, we are able to certify that the *new* design of the flip gate is behaviorally correct. The proposed design has a success probability (i.e., the probability in which the circuit produces the right output) of 100% and is composed of two beam splitters and a phase shifter as shown in Fig. 4. We verify that our design and the one in [RGM+03] are equivalent by checking that the underlying design behaves according to the gate truth table. The gate flips the input state: if the possible input is $\mid \phi\rangle_{input} = \alpha \mid 0\rangle + \beta \mid 1\rangle$ then the output is $\mid \phi\rangle_{output} = \alpha \mid 1\rangle + \beta \mid 0\rangle$. We define the flip gate in HOL as follows:

Definition 6.2 (Flip Gate)
$\vdash$ FLIP((in : sm), out, LH, LV, ten) $\Longleftrightarrow$ ($\exists$ (a : sm$^{\mathtt{N}}$) b c d.
is_beam_splitter($\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$, ten, a(1), 1, a(2), 2, b(1), 1, b(2), 2) $\wedge$
is_beam_splitter($\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$, ten, b(1), 1, b(2), 2, d(1), 1, c(2), 2) $\wedge$
phase_shifter(ten, π, c(2), 2, d(2), 2) $\wedge$ FLIP_In_Output(in, out, a, d, LH, LV))

where FLIP takes as parameters the gate input and output ports (i.e., LH, LV, in and out) and the tensor operator. Similar to the Hadamard gate, we define FLIP_In_Output as 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 [RGM+03] (and verified in [MAT14]) by proving the result of applying the gate on the two possible inputs $\mid 0, 1\rangle_a$ and $\mid 1, 0\rangle_a$.

Theorem 6.2 (Inputs: $\mid 0, 1\rangle_a \equiv LV\ in$ and $\mid 1, 0\rangle_a \equiv LH\ in$)
$\vdash$ FLIP (a, d, LH, LV, ten) $\Longrightarrow$
tensor 1 (λ i. LV in) = tensor 1 (λ i. LH out) $\wedge$
tensor 1 (λ i. LH in) = tensor 1 (λ i. LV out)

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

6.4. Non-linear sign gate

In [KLM01], the authors realized the universal controlled-phase gate using the nondeterministic non-linear sign (NS) quantum gate (Fig. 5), which is composed of three beam splitters. The NS gate operates as follows: when a superposition of the vacuum state $\mid 0\rangle$, the one photon state $\mid 1\rangle$ and the two-photon state $\mid 2\rangle$ is input into the NS gate, the gate flips the sign (or the phase) of the amplitude of the $\mid 2\rangle$ component. Contrary to the Hadamard and flip gates, the NS gate contains two ancillas, one with a single photon and the other in vacuum as shown in Fig. 5. Thus, a successful transformation of the input by the gate corresponds to the event when the first ancilla contains one photon and the other contains zero photon. NS gate shows the feasibility of realizing simple non-linear operations using ancilla photons, linear optics, and linear projection (i.e., postselection) [KLM01].

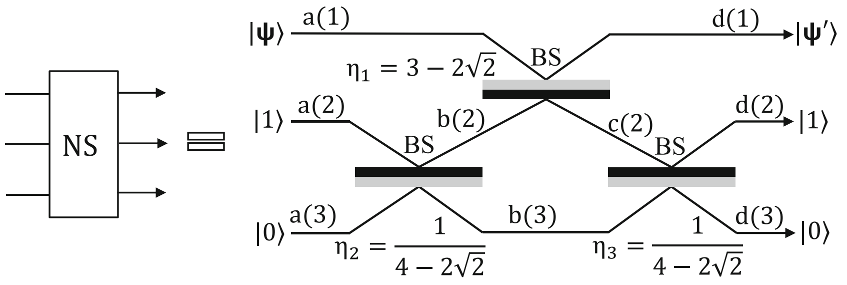


Fig. 5. Schematics of NS gate

For instance, if the input is like $(|\psi\rangle_1 = \alpha|0\rangle_1 + \beta|1\rangle_1 + \gamma|2\rangle_1) \otimes |1\rangle_2 \otimes |0\rangle_3$, then when we measure a single photon at port $d(2)$ and vacuum at port $d(3)$, the gate operation is considered successful. In this case we have the output state $|\psi\rangle_1' = \alpha|0\rangle_1 + \beta|1\rangle_1 - \gamma|2\rangle_1$ at port $d(1)$. Given this structure, the probability of measuring a single photon at the ancilla port $d(2)$ and vacuum at the ancilla port $d(3)$ is then $\frac{1}{4}$. Accordingly, we define the NS gate structure in HOL as follows:

Definition 6.3 (NS Gate)
$\vdash \mathtt{NS}((\mathtt{a}:\mathtt{sm}^{\mathtt{N}}),\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{ten}) \Longleftrightarrow$
$\quad \mathtt{is\_beam\_splitter}(\sqrt{2\sqrt{2}-2}, \sqrt{3-2\sqrt{2}}, -\sqrt{3-2\sqrt{2}}, \sqrt{2\sqrt{2}-2}, \mathtt{ten}, \mathtt{b}(2), 2, \mathtt{a}(1), 1, \mathtt{d}(1), 1, \mathtt{c}(2), 2)\ \wedge$
$\quad \mathtt{is\_beam\_splitter}(\frac{1}{\sqrt{4-2\sqrt{2}}}, \frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}, \frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}, -\frac{1}{\sqrt{4-2\sqrt{2}}}, \mathtt{ten}, \mathtt{a}(2), 2, \mathtt{a}(3), 3, \mathtt{b}(2), 2, \mathtt{b}(3), 3)\ \wedge$
$\quad \mathtt{is\_beam\_splitter}(\frac{1}{\sqrt{4-2\sqrt{2}}}, \frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}, \frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}, -\frac{1}{\sqrt{4-2\sqrt{2}}}, \mathtt{ten}, \mathtt{c}(2), 2, \mathtt{b}(3), 3, \mathtt{d}(2), 2, \mathtt{d}(3), 3)$

where NS takes as parameters the two input vectors (a, b), the two output vectors (c, d), and the tensor operator ten. Using this definition of NS gate, we formally verify the expected output and its joint success probability by projecting all NS gate outputs on the expected output. We prove that for an input $|2,1,0\rangle_a$ the projection of NS gate output on the states $|0,1,0\rangle_d$ and $|1,1,0\rangle_d$ gives zero. On the contrary, the projection on the state $|2,1,0\rangle_d$ gives $-\frac{1}{2}$ (success probability $(\frac{1}{2})^2 = \frac{1}{4}$). We repeat the same procedure for the two other possible inputs (i.e., $|0,1,0\rangle_a$ and $|1,1,0\rangle_a$).

Theorem 6.3 (NS Input: $|2\rangle$, Projection: $|2\rangle$)
$\vdash$ let $|2,1,0\rangle_{\mathtt{d}} = \mathtt{tensor\ 3\ (\lambda i.\ if\ i=1\ then}\ |2\rangle_{\mathtt{d(1)}}\ \mathtt{elseif\ i=2\ then}\ |1\rangle_{\mathtt{d(2)}}\ \mathtt{else}\ |0\rangle_{\mathtt{d(3)}})$ in
$\quad$ let $|2,1,0\rangle_{\mathtt{a}} = \mathtt{tensor\ 3\ (\lambda i.\ if\ i=1\ then}\ |2\rangle_{\mathtt{a(1)}}\ \mathtt{elseif\ i=2\ then}\ |1\rangle_{\mathtt{a(2)}}\ \mathtt{else}\ |0\rangle_{\mathtt{a(3)}})$ in
$\quad \mathtt{NS}(\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{ten}) \Longrightarrow (\mathtt{m\_proj}\ \ |2,1,0\rangle_{\mathtt{d}})\ \ |2,1,0\rangle_{\mathtt{a}} = -\frac{1}{2}\%|2,1,0\rangle_{\mathtt{d}}$

Next, we show the projection of the previous input ($|2,1,0\rangle$) over a different quantum state $|1,1,0\rangle$. The result of this projection is the zero constant function, as follows:

Theorem 6.4 (NS Input: $|2\rangle$, Projection: $|1\rangle$)
$\vdash$ let $|1,1,0\rangle_{\mathtt{d}} = \mathtt{tensor\ 3\ (\lambda i.\ if\ i=1\ then}\ |1\rangle_{\mathtt{d(1)}}\ \mathtt{elseif\ i=2\ then}\ |1\rangle_{\mathtt{d(2)}}\ \mathtt{else}\ |0\rangle_{\mathtt{d(3)}})$ in
$\quad$ let $|2,1,0\rangle_{\mathtt{a}} = \mathtt{tensor\ 3\ (\lambda i.\ if\ i=1\ then}\ |2\rangle_{\mathtt{a(1)}}\ \mathtt{elseif\ i=2\ then}\ |1\rangle_{\mathtt{a(2)}}\ \mathtt{else}\ |0\rangle_{\mathtt{a(3)}})$ in
$\quad \mathtt{NS}(\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{ten}) \Longrightarrow (\mathtt{m\_proj}\ \ |1,1,0\rangle_{\mathtt{d}})\ \ |2,1,0\rangle_{\mathtt{a}} = 0$

where, m_proj $(|1,1,0\rangle_{\mathtt{d}})$ is the projection on the state $|1,1,0\rangle_d$.

6.5. Controlled phase gate

The controlled-phase (CZ) gate is a two qubits gate which transforms the input state $|x,y\rangle$ to the output $e^{i\pi x.y}|x,y\rangle$, $x, y \in \{0,1\}$. In other words, if the possible input is $|\phi\rangle_{input} = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, then the output is $|\phi\rangle_{output} = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle - \delta|11\rangle$. The CZ gate is constructed with the use of two NS gates and two beam splitters, as shown in Fig. 6.

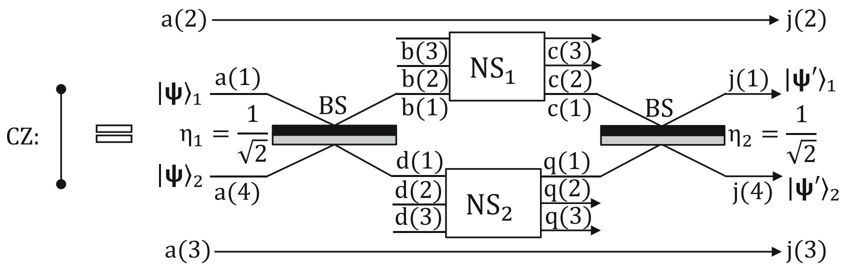


Fig. 6. Schematics of CZ gate

The probability of measuring the ancilla state $|\,1, 0\rangle$ in both NS gates is $\frac{1}{16}$, which is the success probability of the CZ gate (otherwise the gate fails, i.e., the result of the measurement of the ancilla states is other than $|\,1, 0\rangle$). We formally define the CZ gate as follows:

Definition 6.4 (CZ Gate)

$\vdash$ CZ((x1 : sm), x2, y1, y2, ten, LH, LV, m_proj) $\Longleftrightarrow$ ($\exists$ (a : sm^N) b c j d q k l m p.
is_sm a(3) $\wedge$ is_sm a(2) $\wedge$ b(5)=d(2) $\wedge$ b(6)=d(3) $\wedge$ b(4)=d(1) $\wedge$
q(1)=c(4) $\wedge$ q(2)=c(5) $\wedge$ q(3)=c(6) $\wedge$ NS(d, m, p, q, ten) $\wedge$ NS(b, l, k, c, ten) $\wedge$
is_beam_splitter($\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$, ten, a(1), 1, a(4), 4, b(1), 1, b(4), 4) $\wedge$
is_beam_splitter($\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$, ten, c(1), 1, c(4), 4, j(1), 1, j(4), 4) $\wedge$
CZ_INPUTS(x1, x2, a, b, c, LH, LV, m_proj) $\wedge$ CZ_OUTPUTS(y1, y2, a, c, j, LH, LV))

Notice that we renamed the input and output ports for the second NS gate in order to match the order of the modes in the definition of the gate, i.e., instead of $|\,b(4), b(5), b(6)\rangle$ and $|\,c(4), c(5), c(6)\rangle$ we have $|\,d(1), d(2), d(3)\rangle$ and $|\,q(1), q(2), q(3)\rangle$, respectively. From this definition, we formally verify the CZ gate operations and its success probability. There are four possible combinations of inputs. Here we are providing one of them as example and the rest can be found in [BM19].

Theorem 6.5 (CZ Input: $|\,1, 1\rangle_{x1x2} \equiv |\,1\rangle_{x1} \otimes |\,1\rangle_{x2}$)

$\vdash$ let $|\,2, 1, 0, 0, 1, 0, 0, 0\rangle_{cq}$ = tensor 8 (λ i. if i=1 then $|\,2\rangle_{c(1)}$ elseif i=2 then $|\,1\rangle_{c(2)}$
elseif i=5 then $|\,1\rangle_{q(2)}$ else $|\,0\rangle_{c(3)}$) in
let $|\,0, 1, 0, 2, 1, 0, 0, 0\rangle_{cq}$ = tensor 8 (λ i. if i=2 then $|\,1\rangle_{c(2)}$ elseif i=4 then $|\,2\rangle_{q(1)}$
elseif i=5 then $|\,1\rangle_{q(2)}$ else $|\,0\rangle_{c(3)}$) in
let $|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{cq}$ = tensor 8 (λ i. if i=1 then $|\,1\rangle_{c(1)}$ elseif i=2 then $|\,1\rangle_{c(2)}$
elseif i=4 then $|\,1\rangle_{q(1)}$ elseif i=5 then $|\,1\rangle_{q(2)}$ else $|\,0\rangle_{c(3)}$) in
let $|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{ab}$ = tensor 8 (λ i. if i=1 then $|\,1\rangle_{a(1)}$ elseif i=2 then $|\,1\rangle_{b(2)}$
elseif i=4 then $|\,1\rangle_{a(4)}$ elseif i=5 then $|\,1\rangle_{b(5)}$ else $|\,0\rangle_{b(3)}$) in
let $|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{cj}$ = tensor 8 (λ i. if i=1 then $|\,1\rangle_{j(1)}$ elseif i=2 then $|\,1\rangle_{c(2)}$
elseif i=4 then $|\,1\rangle_{j(4)}$ elseif i=5 then $|\,1\rangle_{c(5)}$ else $|\,0\rangle_{c(3)}$) in
CZ(a, b, c, j, ten, LH, LV, m_proj) $\Longrightarrow$
(m_proj $|\,2, 1, 0, 1, 0, 0, 0, 0\rangle_{cq}$ + m_proj $|\,0, 1, 0, 1, 2, 0, 0, 0\rangle_{cq}$ + m_proj $|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{cq}$)
($|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{ab}$) = $-\ \frac{1}{4}$ % $|\,1, 1, 0, 1, 1, 0, 0, 0\rangle_{cj}$

Notice that the output of the CZ gate has been projected over three different states. This is because of the fact that we have two photons at the input ($|\,1, 1\rangle_{x1x2}$) which results in three possibilities at the input of the two parallel NS gates: 1) two photons go through the first NS gate; 2) two photons go through the second NS gate; and 3) one photon goes through the first NS gate and the other goes through the second one. The verification of the CZ has been done using Theorem 6 in order to subdivide the tensor product projection to the tensor of two tensor product projections each fed to an NS gate. Moreover, through the projection we eliminate the undesired outputs with their own probability and the remaining output is the correct. Thus, the square of the amplitude of the produced output is the probability of obtain this output.

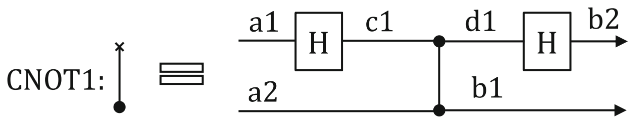


Fig. 7. Schematics of CNOT gate

As shown in Fig. 6, the CZ gate has 8 input optical modes. However, the CZ gate is a 2-qubit gate, where each logical qubit is represented by two optical modes and the remaining four modes are ancillas. In order to facilitate the use of this gate in quantum circuits where the computation is at the qubit level, we developed `CZ_INPUTS(x1, x2, a, b, c, LH, LV, m_proj)` and `CZ_OUTPUTS(y1, y2, a, c, j, LH, LV)`, a behavioral description for CZ where instead of eight inputs and eight outputs, we have two inputs and two outputs [BMT16b]. This completes the formal analysis of the CZ gate for the input "11". The analysis for the inputs "10", "01", and "00" follows the similar pattern.

6.6. Controlled-not gate

The Controlled-not (CNOT) gate is a two inputs/two outputs gate, namely control and target signals. The gate functionality is to invert the target bit whenever the control bit is equal to one, and nothing changes as long as the control bit is equal to zero. The control bit is always transmitted as it is. In other words, if the possible input is $|\,\phi\rangle_{input} = \alpha\,|\,00\rangle + \beta\,|\,01\rangle + \gamma\,|\,10\rangle + \delta\,|\,11\rangle$, then the output is $|\,\phi\rangle_{output} = \alpha\,|\,00\rangle + \beta\,|\,11\rangle + \gamma\,|\,10\rangle + \delta\,|\,01\rangle$. Here, we will show the gate implementation using CZ and Hadamard gates, as shown in Fig. 7, however, it can also be implemented using five beam splitters as shown in [RLB+02] and verified in [MPT15]. Contrary to CZ, CNOT is not symmetric (i.e., an exchange in the order of inputs implies a modification in the design of the gate). Therefore, we have formally defined two versions of the CNOT, where for the first version the target qubit is fed to the first input and for the second version it is fed to the second input. We provide here the definition of the first version of CNOT gate structure (we name it CNOT1). The second one (CNOT2) is given in [BMT16b].

Definition 6.5 (CNOT Gate)

```
⊢ CNOT1((a1 : sm), a2, b1, b2, ten, LH, LV, m_proj) ⟺ (∃ c1 d1.
  HADAMARD(a1, c1, ten, LH, LV) ∧ HADAMARD(d1, b2, ten, LH, LV) ∧ CZ(c1, a2, d1, b1, ten, LH, LV, m_proj))
```

Here, the Hadamard gate is applied on the first input, which is the target qubit. We formally verified that this definition maintains the truth table of the CNOT gate. As an example we provide the result of applying the CNOT on the input $|\,0, 1\rangle$:

Theorem 6.6 (CNOT Gate Input: $|\,0, 1\rangle$)

```
⊢ CNOT1(a1, a2, b1, b2, ten, LH, LV, m_proj)  ⟹
  tensor 2 (λ i. if i=1 then LH a1 else LV a2) =
  ¼ % tensor 2 (λ i. if i=1 then LV b1 else LV b2)
```

Notice that the CNOT gate has the same success probability as the CZ gate. The success probability of CZ gate is calculated based on the probabilities of Hadamard and CZ gates. Because the Hadamard and CZ gates are connected in series to form CNOT gate, then the success probability of CZ gate is the multiplication of the success probabilities of Hadamard and CZ gates. As the Hadamard has a success probability of 1 then CNOT has the same success probability as the CZ gate. Similar to CZ, CNOT uses 8 optical modes as well.

6.7. SWAP gate

The SWAP gate is a two qubits gate which swaps the states of 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 fulfil the requirement that computations should only be performed between adjacent qubits [SWH+12].

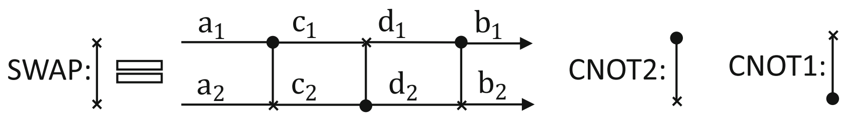


Fig. 8. SWAP gate

Also in [LL05], 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*. In [FDH04], it was shown that the SWAP gate plays an important role in the implementation of Shor's algorithm [Shor97] based on linear nearest neighbor architecture, where the SWAP gate rearranges the qubits. The physical implementation of the SWAP gate requires three CNOT gates, as shown in Fig. 8. We define the SWAP gate in HOL as follows:

Definition 6.6 (SWAP Gate)
$\vdash$ `SWAP ((a1 : sm), a2, b1, b2, ten, LH, LV, m_proj)` $\Longleftrightarrow$ `(`$\exists$ `c1 c2 d1 d2.`
`CNOT2(a1, a2, c1, c2, ten, LH, LV, m_proj)` $\wedge$ `CNOT1(c1, c2, d1, d2, ten, LH, LV, m_proj)` $\wedge$
`CNOT2(d1, d2, b1, b2, ten, LH, LV, m_proj))`

where `m_proj` is the tensor product projection operator. In contrast to the presentation of previous gates, for the SWAP gate we present its application over the general form of the input: $|\, a1, a2\rangle = (\alpha_1\% \,|\, 1\rangle + \alpha_2\% \,|\, 0\rangle) \otimes (\beta_1\% \,|\, 1\rangle + \beta_2\% \,|\, 0\rangle) = (\alpha_1 * \beta_1)\% \,|\, 1, 1\rangle + (\alpha_2 * \beta_1)\% \,|\, 0, 1\rangle + (\alpha_1 * \beta_2)\% \,|\, 1, 0\rangle + (\alpha_2 * \beta_2)\% \,|\, 0, 0\rangle$. Notice that this input is a superposition of four possible inputs. Therefore, in order to prove the output and input relation theorem for this input in HOL, we should first prove output and input relation theorems for each elementary input of the four possible inputs and then by using the bi-linearity tensor product we obtain the following theorem.

Theorem 6.7 (SWAP Gate Input: $|\, x, y\rangle$)
$\vdash$ `SWAP(a1, a2, j1, j2, ten, LH, LV, m_proj)` $\Longrightarrow$
`tensor 2 (`λ `i. if i=1 then (`α_1`%LV a1 +` α_2`%LH a1) else (`β_1`%LV a2 +` β_2`%LH a2))` $=$
$\frac{1}{64}$ `% tensor 2 (`λ `i. if i=1 then (`β_1`%LH j1 +` β_2`%LV j1) else (`α_1`%LH j2 +` α_1`%LV j2))`

where $\frac{1}{64}$ is the success rate of the gate. We can notice that in Theorem 6.7, all behaviors of the gate were considered.

6.8. Toffoli sign gate

The Toffoli Sign (TS) gate is a quantum gate that applies a sign shift on one of the inputs, and the identity to the rest. The main benefit of the TS gate is to construct the Toffoli gate. In the optical implementation of TS, `vac k` refers to the vacuum state, i.e., where both polarization modes are unoccupied, in the single mode `k`. Thus, we will introduce a third level of representation of a *qutrit* (i.e., a superposition of three orthogonal quantum states [RRG07] where each state is represented by an optical polarization mode). The TS gate structure is shown in Fig. 9. The realization of TS is based on using two qubits a(1), a(3) (i.e., $0_L = |\, LH\rangle$ and $1_L = |\, LV\rangle$) and a *qutrit* (i.e., $1_L = |\, vac, LV\rangle_{a(2)k}$, $0_L = |\, LH, vac\rangle_{a(2)k}$, and $2_L = |\, LV, vac\rangle_{a(2)k}$).

To our best knowledge, the sole purpose of TS gate is to construct the Toffoli gate [RRG07]. Therefore, the third input of the gate (i.e., the *qutrit*) can only receive the values $1_L = |\, vac, LV\rangle_{a(2)k}$ and $0_L = |\, LH, vac\rangle_{a(2)k}$ when it is implemented as part of Toffoli gate. Thus, it is sufficient in the formalization to only consider the values 0_L and 1_L for the inputs of the TS gate. Then, we convert the 0_L and 1_L for the third input based on the *qutrit* definition and convert 0_L and 1_L for the first two inputs based on the qubit definition. Thus, behaviorally the TS gate receives and transforms the same set of inputs as a 3-qubit gate but physically the TS gate interprets the third input as a *qutrit*.

The structure of the TS gate is composed of two CNOT gates and a CZ gate. The two CNOT gates operate as normal at the qubit levels and implement the identity if the target is at the *qutrit* level ($|\, vac, LV\rangle$). We define the TS gate in HOL as follows:

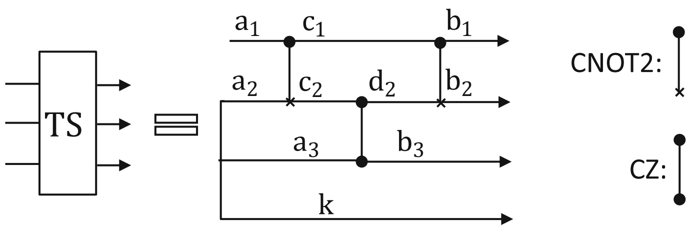


Fig. 9. TS gate

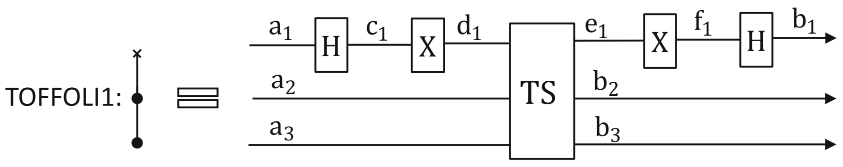


Fig. 10. Toffoli gate

Definition 6.7 (TS Gate)

$\vdash$ `TS ((a1 : sm), a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longleftrightarrow$ `(`$\exists$ `k c1 c2 d2.`
`CNOT2(a1, a2, c1, c2, ten, LH, LV, m_proj)` $\wedge$ `CNOT2(c1, d2, b1, b2, ten, LH, LV, m_proj)` $\wedge$
`CZ(c2, a3, d2, b3, ten, LH, LV, m_proj)` $\wedge$ `TS_outputs(k, b1, b2, b3, LH, LV)` $\wedge$ `TS_inputs(k, a1, a2, a3, LH, LV))`

In our formal definition of TS gate, for an input $\mid x, y, z\rangle$, x is the qutrit. For example, for the input $\mid 011\rangle$, the 4-qubit format is given as $\mid 1\rangle_{a(1)} \otimes \mid vac\rangle_{a(2)} \otimes \mid 1\rangle_{a(3)} \otimes \mid 1\rangle_k$. The following is the result of the TS transformation over the input $\mid 011\rangle$:

Theorem 6.8 (TS Input: $\mid 011\rangle$)

$\vdash$ `TS(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longrightarrow$
`tensor 3 (`λ `i. if i=1 then LH a1 elseif i=2 then LV a2 else LV a3)` $=$
$-\frac{1}{64}$ `% tensor 3 (`λ `i. if i=1 then LH b1 elseif i=2 then LV b2 else LV b3)`

Notice that there is a sign shift on the output state $\mid 011\rangle$. Also the gate success probability is $\frac{1}{64}$. For the rest of the possible combinations for the inputs there will be no sign shift. Now, after modeling and verifying the Toffoli Sign gate, we are ready to tackle the formalization of the Toffoli gate.

6.9. Toffoli gate

The Toffoli is a 3-qubit gate that flips the logical state of the target qubit if the two control qubits are in the state $\mid 1\rangle$ and does nothing for the rest of the 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. Inspired from [RRG07], using the TS, flip, and Hadamard gates we can construct the Toffoli gate as shown in Fig. 10. Because the TS gate design is at the physical level and requires only three 2-qubit gates, the number of required 2-qubit gates for Toffoli is only three. However, when restricted to the modeling at the behavioral level, the simplest known design of the Toffoli gate requires five 2-qubit gates [HSY+04, SWM12]. Therefore, it is possible to reduce the number of required 2-qubit gates from five to three [RRG07], thanks to the low level modeling approach. Similar to the CNOT gate, the Toffoli gate can be used in two forms: 1) the first qubit is the target; and 2) the third qubit is the target (we name them TOFFOLI1 and TOFFOLI3, respectively). Therefore, we have formally defined these two kinds of Toffoli gate in HOL. More details about the first kind of Toffoli gate can be found at [BM19]. We provide here the definition of the first type of Toffoli gate (`TOFFOLI1`):

Definition 6.8 (Toffoli Gate)

$\vdash$ `TOFFOLI1((a1 : sm), a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longleftrightarrow$ `(`$\exists$ `c3 d3 c1 d1.`
`HADAMARD(a1, c1, ten, LH, LV)` $\wedge$ `HADAMARD(f1, b1, ten, LH, LV)` $\wedge$ `FLIP(c1, d1, ten, LH, LV)` $\wedge$
`FLIP(e1, f1, ten, LH, LV)` $\wedge$ `TS(d1, a2, a3, e1, b2, b3, ten, LH, LV, m_proj))`

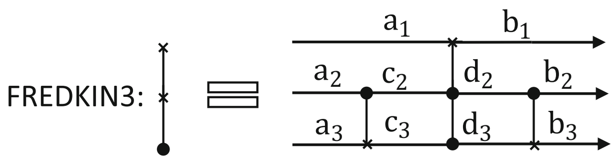


Fig. 11. Fredkin gate

From this definition, we verify the result of applying Toffoli on the input $|\,111\rangle$:

Theorem 6.9 (Toffoli Input: $|\,111\rangle$)
$\vdash$ `TOFFOLI1(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longrightarrow$
`tensor 3 (`λ` i. if i=1 then LV a1 elseif i=2 then LV a2 else LV a3)` $=$
$\frac{1}{64}$ `% tensor 3 (`λ` i. if i=1 then LH b1 elseif i=2 then LV b2 else LV b3)`

Because the two control qubits are $|\,1\rangle_L$, the state of the first qubit is flipped. Notice that the success probability of the gate is $\frac{1}{64}$. In contrast, if the Toffoli gate was constructed using five 2-qubit gates, the success probability will be $\frac{1}{1024}$.

6.10. Fredkin gate

The Fredkin gate, or the controlled-2 × 2 reversible quantum switch gate (or controlled SWAP gate), is a 3-qubit gate [Mil89]. One of the qubits is designated as the control qubit and is left unchanged by the gate, and the remaining 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 inter-changed. The Fredkin gate has an important role in quantum computing and quantum computations error-correcting [NC10]. Moreover, it is a universal gate for reversible computing. This means that any logical or arithmetic operation can be constructed entirely using this gate [Mil89]. The gate circuit is shown in Fig. 11, which is composed of two CNOT and one Toffoli gates. Considering the two versions of the Toffoli gate formalized in the previous section, we get two versions of the Fredkin gate: (1) the first qubit is the control (FREDKIN1); and (2) the third qubit is the control (FREDKIN3). We model the second version in HOL as follows:

Definition 6.9 (Fredkin Gate)
$\vdash$ `FREDKIN3((a1 : sm), a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longleftrightarrow$ `(`$\exists$` c2 c3 d2 d3.`
`TOFFOLI1(a1, c2, c3, b1, d2, d3, ten, LH, LV, m_proj)` $\wedge$ `CNOT2(a2, a3, c2, c3, ten, LH, LV, m_proj)` $\wedge$
`CNOT2(d2, d3, b2, b3, ten, LH, LV, m_proj))`

Similar to our presentation of SWAP gate, we provide here the result of applying the Fredkin gate on the input in the general form $|\,xzy\rangle$ (i.e., $|\,xzy\rangle = (\alpha_2*\beta_2*\theta_2)\%\,|\,1,1,1\rangle + (\alpha_2*\beta_2*\theta_1)\%\,|\,1,1,0\rangle + (\alpha_2*\beta_1*\theta_2)\%\,|\,1,0,1\rangle + (\alpha_2*\beta_1*\theta_1)\%\,|\,1,0,0\rangle + (\alpha_1*\beta_2*\theta_2)\%\,|\,0,1,1\rangle + (\alpha_1*\beta_2*\theta_1)\%\,|\,0,1,0\rangle + (\alpha_1*\beta_1*\theta_2)\%\,|\,0,0,1\rangle + (\alpha_1*\beta_1*\theta_1)\%\,|\,0,0,0\rangle$, the input is a superposition of eight possible inputs):

Theorem 6.10 (Fredkin Input: $|\,zxy\rangle$)
$\vdash$ `FREDKIN3(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj)` $\Longrightarrow$
`tensor 3 (`λ` i. if i=1 then (`α_1`%LH a1 + `α_2`%LV a1) elseif i=2 then (`β_1`%LH a2 + `β_2`%LV a2)`
`else (`θ_1`%LH a3 + `θ_2`%LV a3))` $=$
$\frac{1}{1024}$ `% (`α_1` % tensor 3 (`λ` i. if i=1 then (`α_1`%LH b1 + `α_2`%LV b1) elseif i=2 then`
`(`β_1`%LH b2 + `β_2`%LV b2) else LH b3) +`
θ_2 `% tensor 3 (`λ` i. if i=1 then (`β_1`%LH b + `β_2`%LV b1) elseif i=2 then`
`(`α_1`%LH b2 + `α_2`%LV b2) else LV b3))`

Notice that the success probability of the Fredkin gate is $\frac{1}{1024}$ and hence very small. We have covered the formal modeling and verification of a set of quantum gates which can be used in the analysis of a variety of quantum circuits as it will be described in Sect. 8.

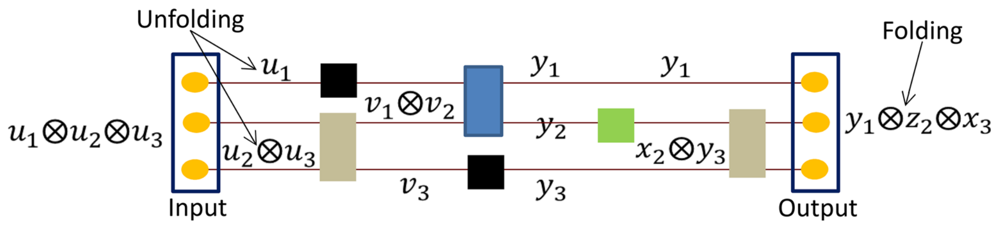


Fig. 12. Tensor product unfolding

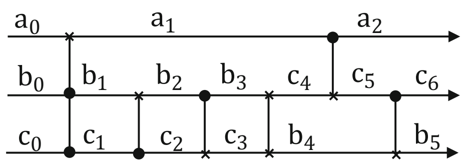

a_0 a_1 a_2
b_0 b_1 b_2 b_3 c_4 c_5 c_6
c_0 c_1 c_2 c_3 b_4 b_5

Fig. 13. Hamming function of size 3

7. Proof strategy

While the developed mathematics and gates library are rich enough to model and verify a variety of quantum circuits, the verification process for a quantum circuit in an interactive theorem prover is not an easy task. This is due to the need of user expertise to guide the proof process. Therefore, some kind of strategies and proof automation is required for the framework to be usable by non-experts. In this section, we propose a proof strategy, which shall fully eliminate the need for user interaction with the theorem prover. This will tremendously aid the use of our framework by engineers and physicists who want to conduct the analysis of quantum circuits.

Generally, any quantum circuit is a collection of gates that are connected to each other either sequentially or in parallel. Accordingly, our proof process proceeds through three major steps: 1) if the main input will go through parallel gates or a single gate with an input size less than the main input size, we unfold the main input tensor product to elementary tensors to be input to parallel gates or the single gate, as shown in Fig. 12; 2) applying the required gates transformation; 3) folding the tensor product back. Then, we repeat this process until the input tensor goes through all gates transformations that are sequential. Finally, we rewrite the obtained result to the final format using some linear algebra theorems.

7.1. Verification process of a quantum circuit

In this section, we present the verification process for any quantum circuit using the developed mathematical foundation and the gates library. We use the Hamming optimal coding function of size 3 circuit as an example to illustrate the proof steps for verifying quantum circuits using our framework. The circuit is composed of one SWAP, one Toffoli, and four CNOT gates, as shown in Fig. 13, and it has three inputs/outputs. The inputs are initialized to the state $|\,\psi\rangle_{in} = |\,1, 0, 0\rangle_{a0b0c0}$. Accordingly, we define the circuit structure in HOL as follows:

Definition 7.1 (Hamming Circuit)

```
⊢ HAM3((a0 : sm), b0, c0, a2, b5, c6, ten, LH, LV, m_proj) ⟺ (∃ a1 b1 c1 b2 c2 b3 c3 c4 b4 c5.
  TOFFOLI1(a0, b0, c0, a1, b1, c1, ten, LH, LV, m_proj) ∧ CNOT1(b1, c1, b2, c2, ten, LH, LV, m_proj)
  CNOT2(b2, c2, b3, c3, ten, LH, LV, m_proj) ∧ SWAP(b3, c3, c4, b4, ten, LH, LV, m_proj) ∧
  CNOT2(a1, c4, a2, c5, ten, LH, LV, m_proj) ∧ CNOT2(c5, b4, c6, b5, ten, LH, LV, m_proj))
```

The Hamming circuit input is: `tensor 3 (λ i. if i=1 then LV a0 elseif i=2 then LH b0 else LH c0)`. As shown in Fig. 13, the Toffoli gate is the first to act on the circuit input, therefore, we do not need to unfold the input. The Toffoli gate outputs the state: `(1/64) % tensor 3 (λ i. LV a1 elseif i=2 then LH b1 else LH c1)`. After Toffoli, the input undertakes a CNOT transformation, therefore, we need to unfold the tensor to two elementary tensors by rewriting the goal using tensor product theorems and the lemmas in Table 2.

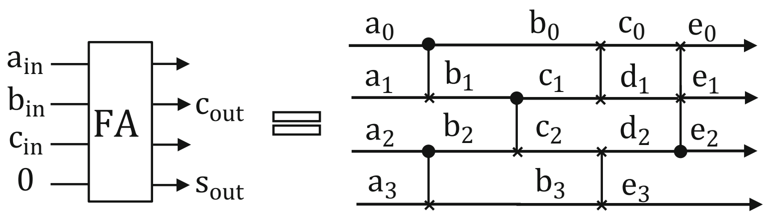


Fig. 14. Quantum full adder

The resulting expression is in the form: `(1/64) % (λ y. (tensor 1 (λ i. LV a1) y(1)) * (tensor 2 (λ i. if i= 1 then LH b1 else LH c1) y(2)))`. We now apply the CNOT over this expression which yields the following output: `(1/64 * 1/4) % (λ y. (tensor 1 (λ i. LV a1) y(1)) * (tensor 2 (λ i. if i=1 then LH b2 else LH c2) y(2)))`. Before folding back the tensor product, we apply the next CNOT and the SWAP gate. To apply the third CNOT gate, we need to fold back the two elementary tensor back to obtain the following expression: `(1/256 * 1/4* 1/64) % tensor 3 (λ i. LV a1 elseif i=2 then LH c4 else LH b4)`. Thereupon, we unfold it to: `(1/65536) % (λ y. (tensor 2 (λ i.if i=1 then LV a1 else LH c4) y(1)) * (tensor 1 (λ i. LH b4) y(2)))`. Next, we apply the CNOT gate and obtain the following expression: `(1/65536 * 1/4) % (λ y. (tensor 2 (λ i. if i=1 then LV a2 else LV c5) y(1)) * (tensor 1 (λ i. LH b4) y(2)))`. Finally, we fold back the elementary tensors, unfold the obtained tensor, and apply the last CNOT gate. After applying all gates transformations, we obtain the final expression given in the RHS of Theorem 7.1.

Theorem 7.1 (Hamming Function Size 3**)**

```
⊢ HAM3(a0, b0, c0, a2, b5, c6, ten, LH, LV, m_proj) ⟹
  tensor 3 (λ i. if i=1 then LV a0 elseif i=2 then LH b0 else LH c0) =
  1/1048576 % tensor 3 (λ i. if i=1 then LV a2 elseif i=2 then LV c6 else LV b5)
```

The process of verifying the Hamming function of size 3 was not easy, involving more than 10 lemmas to prove. In addition, the proof of Theorem 7.1 required more than 300 lines of HOL Light proof script. Based on this result, the proof of circuits that involve dozens of gates may involve thousands lines of HOL Light proof script, which is very tedious even for an expert in HOL Light. Hence, providing automation is necessary for our framework to be used in the analysis of quantum circuits. In the next section, we describe a proof strategy that fully automates the analysis process.

7.2. The engineering of the proof strategy

The proposed proof strategy (third block of the framework given in Fig. 2) takes a quantum circuit netlist and its inputs and builds tactics that automate the analysis proof in HOL Light. The core of the procedure is an Ocaml function *matrix_procedure* that extracts a matrix description from a textual quantum circuit netlist description 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 circuit description for two pieces of information: 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 *quantum_tac* takes this matrix and generates the required folding/unfolding lemmas and tactics. This function takes the extracted matrix to provide the proof steps, subgoals and lemmas to automatically prove the required theorems for the underlying circuit. It utilizes a combination of tactics that are set up for each particular gate of the library.[7]

[7] The proof strategy and the associated tactics and lemmas are available in tactics_1.ml and tactics_2.ml of [BM19].

Table 2. Tensor product folding/unfolding lemmas

lemma1:	$\texttt{tensor m + n mode} =$ $(\lambda\, y.\, (\texttt{tensor m mode})\, y * (\texttt{tensor n}\, (\lambda\, i.\, \texttt{mode(i+m)}))\, (\lambda\, i.\, y(i+m)))$
lemma2:	$(\texttt{if } i \le k1 \wedge k2 \le i \texttt{ then } (\texttt{if } i{=}k \texttt{ then } x_k \texttt{ else} \cdots \texttt{if } i{=}k2 \texttt{ then } x_{k2}$ $\texttt{else} \cdots \texttt{if } i{=}k1 \texttt{ then } x_{k1} \texttt{ else} \cdots \texttt{else } x_m) \texttt{ else } y) =$ $(\texttt{if } i \le k1 \wedge k2 \le i \texttt{ then } (\texttt{if } i{=}k2 \texttt{ then } x_{k2} \texttt{ else} \cdots \texttt{else } x_{k1}) \texttt{ else } y)$
lemma3:	$\forall\, i\, j\, k \in \mathbb{N}.\ (i + j = k) \Leftrightarrow (\texttt{if } (j \le k) \texttt{ then } (i{=}k\text{-}j) \texttt{ else FALSE})$
lemma4:	$\texttt{tensor m mode} = \texttt{tensor m}\, (\lambda\, i.\, \texttt{if } i \le m \wedge 1 \le i \texttt{ then mode(i) else } y)$
lemma5:	$(f_1\, x_1) * \cdots * ((a_k \,\%\, f_k)\, x_k) * \cdots * (f_n\, x_n) =$ $a_k * ((f_1\, x_1) * \cdots * (f_k\, x_k) * \cdots * (f_n\, x_n))$
lemma6:	$(\lambda\, y.\, ((\texttt{tensor m mode1})\, y) * (\texttt{tensor n mode2})\, (\lambda\, i.\, y(i + m))) =$ $\texttt{tensor}\, (m + n)\, (\lambda\, i.\, \texttt{if } i \le m \texttt{ then mode1(i) else mode2(i)})$
lemma7:	$(\texttt{if } i \le m \wedge 1 \le i \texttt{ then } (\texttt{if } i \le k \texttt{ then } (\texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \cdots \texttt{else } x_k)$ $\texttt{else } (\texttt{if } i{=}1 \texttt{ then } x_{k+1} \texttt{ else} \cdots \texttt{else } x_m)) \texttt{ else } y) =$ $(\texttt{if } i \le m \wedge 1 \le i \texttt{ then } (\texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \cdots \texttt{else } x_m) \texttt{ else } y)$

For example, consider the quantum circuit given in Fig. 14. It is a quantum full adder composed of two SWAP gates, three CNOT gates and one Fredkin gate. Using *matrix_procedure*, we extract the following matrix:

$$\begin{bmatrix} \begin{bmatrix}\texttt{0 CNOT2 2 a0 a1 b0 b1}\end{bmatrix} & \begin{bmatrix}\texttt{2 CNOT2 2 a2 a3 b2 b3}\end{bmatrix} \\ \begin{bmatrix}\texttt{1 CNOT2 2 b1 b2 c1 c2}\end{bmatrix} & \\ \begin{bmatrix}\texttt{0 SWAP 2 b0 c1 c0 d1}\end{bmatrix} & \begin{bmatrix}\texttt{2 SWAP 2 c2 b3 d2 c3}\end{bmatrix} \\ \begin{bmatrix}\texttt{0 FREDKIN1 3 c0 d1 d2 e0 e1 e2}\end{bmatrix} & \end{bmatrix}$$

In the matrix, the first row contains the description of the gates that are applied in parallel to the circuit inputs. The subsequent rows describe, in order, the subsequent gates applied to previous gates outputs. Each element of the matrix contains the order of the gate (i.e., order of the gate inputs with regard to the circuit inputs, e.g., 1 means that the gate input starts from the second element of the circuit input), its type, number of inputs, and the list of inputs and outputs. To illustrate the task of the second Ocaml function and the flow of the proofs and the lemmas involved, consider a n-qubits circuit that contains a m-qubits gate ($m \le n$), where the general form of the circuit input is: $\texttt{tensor n}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \ldots \texttt{else } x_n)$. Two of the most important properties of the tensor product are the ability to write tensor as tensor of tensor (*lemma1* and *lemma6* in Table 2). Then the first step in the proof is to rewrite the main tensor product (circuit input) using *lemma1*, *lemma2*, *lemma3* and *lemma4* in the form:

$$\begin{aligned}&(\lambda\, y.\, (\texttt{tensor k1}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \cdots \texttt{else } x_{k1}))\, y\, * \\ &(\texttt{tensor m}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_{k1+1} \texttt{ else} \cdots \texttt{else } x_{k1+n}))\, (\lambda\, i.\, y(i + k1))\, * \\ &(\texttt{tensor k2}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_{k1+n+1} \texttt{ else} \cdots \texttt{else } x_m))\, (\lambda\, i.\, y(i + k1 + m))\end{aligned}$$

where $n = k1 + m + k2$. After rewriting each elementary tensor as in the above equation, we replace the term $\texttt{tensor m}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_{k1+1} \texttt{ else} \cdots \texttt{else } x_{k1+m})$ with its transformation under the m-qubit gate, which is $a \,\%\, \texttt{tensor m}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } z_{k1+1} \texttt{ else} \cdots \texttt{else } z_{k1+m})$. Thus the circuit input becomes:

$$\begin{aligned}&(\lambda\, y.\, (\texttt{tensor k1}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \cdots \texttt{else } x_{k1}))\, y\, * \\ &(a \,\%\, \texttt{tensor m}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } z_{k1+1} \texttt{ else} \cdots \texttt{else } z_{k1+n}))\, (\lambda\, i.\, y(i + k1))\, * \\ &(\texttt{tensor k2}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_{k1+n+1} \texttt{ else} \cdots \texttt{else } x_m))\, (\lambda\, i.\, y(i + k1 + m))\end{aligned}$$

The last step consists of folding back the tensor product by using *lemma4*, *lemma5*, *lemma6*, and *lemma7* of Table 2. Thereafter, the circuit input will become in the form:

$$\begin{aligned}&a \,\%\, \texttt{tensor n}\, (\lambda\, i.\, \texttt{if } i{=}1 \texttt{ then } x_1 \texttt{ else} \cdots \texttt{if } i{=}k1 + 1 \texttt{ then } z_{k1+1} \texttt{ else} \cdots \\ &\texttt{if } i{=}k1 + m + 1 \texttt{ then } x_{k1+m+1} \texttt{ else} \cdots \texttt{else } x_n)\end{aligned}$$

We repeat the same procedure to all circuit gates transformation over the input until reaching the final value of the circuit output. Notice that this proof procedure can be applied to any quantum circuit that is constructed based on the formalized gates library.

For the quantum adder circuit, the input is in the form: `tensor 4 mode`. The second Ocaml function takes the circuit matrix. In the first row of the matrix we have two parallel gates, thus, 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 as shown in Fig. 15. Then, we fold back to the main tensor.

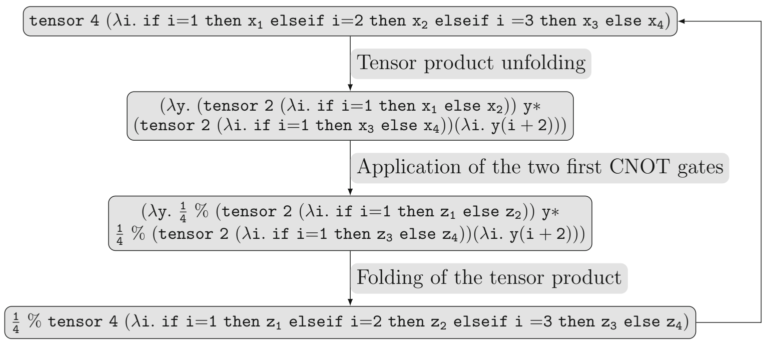

tensor 4 (λi. if i=1 then x₁ elseif i=2 then x₂ elseif i =3 then x₃ else x₄)
Tensor product unfolding
(λy. (tensor 2 (λi. if i=1 then x₁ else x₂)) y∗
(tensor 2 (λi. if i=1 then x₃ else x₄))(λi. y(i + 2)))
Application of the two first CNOT gates
(λy. 1/4 % (tensor 2 (λi. if i=1 then z₁ else z₂)) y∗
1/4 % (tensor 2 (λi. if i=1 then z₃ else z₄))(λi. y(i + 2)))
Folding of the tensor product
1/4 % tensor 4 (λi. if i=1 then z₁ elseif i=2 then z₂ elseif i =3 then z₃ else z₄)

Fig. 15. Tensor product unfolding

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 and thus the circuit output.

7.3. Handling hierarchical verification

To help make the proposed framework generic, we designed the proof strategy such that if a theorem about a new circuit behavior was proven, it is possible to embed the circuit behavior to the gates library for later use. Thus, if copies of this circuit are embedded as part of a larger circuit, then the proven theorem can be used for the proof of the behavior of the larger circuit. To this aim, we developed a procedure that takes the proved theorems about a circuit's behavior and adds them to the existing theorems about the behavior of the gates in the library. So, in this case the procedure will treat the circuit as a gate and use its theorems for the proof of the behavior of any larger circuit that contains the underlying circuit.

This will help in improving the scalability of our approach. In particular, instead of doing the proof of a circuit from scratch we can reuse the proved lemmas and save significant amounts of time in the proof process. For example, consider the 3-bit Ripple Adder. The circuit is composed of three full adders (FA) and four SWAP gates to move the carry to the correct position, as shown in Fig. 16a. However, if our framework does not offer the possibility to reuse the theorems of full adder, we will be forced to express the circuit of 3-bit Ripple Adder in terms of the primitive gates described in Sect. 6, as shown in Fig. 16b.

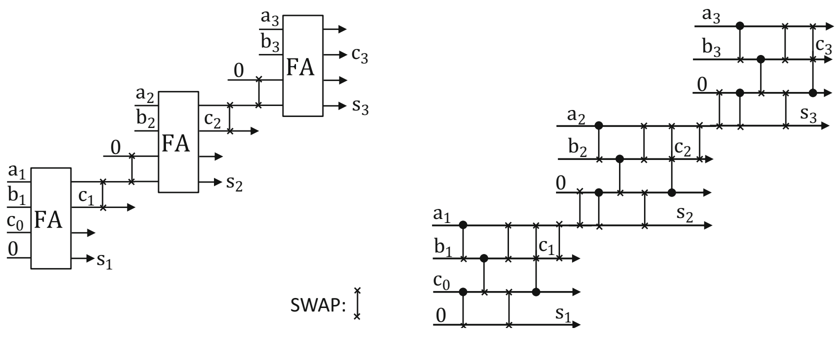
In Table 3, we provide the result of the analysis of the two types of 3-bit Ripple Adder. It is clear that the full adder module had a major effect on our speed. In contrast to the existing work such as in [NWD14] which uses a fixed set of quantum gates such as the Clifford group for quantum circuits synthesis and optimization, our work is flexible in terms of the set of quantum gates and it is possible to add gates in order to have more optimized designs or to speedup the analysis.

8. Experimental results

We have analysed several quantum benchmarks circuits taken from the online library of reversible and quantum circuits at [RevLib]. The provided circuits do not meet the adjacency criteria in quantum computing.

Table 3. 3-bit ripple adder analysis

Ripple adder circuit type	Number of qubits	Number of elements	Proof time (in seconds)
Hierarchical	10	7	1506.9
Flat	10	22	4551.17



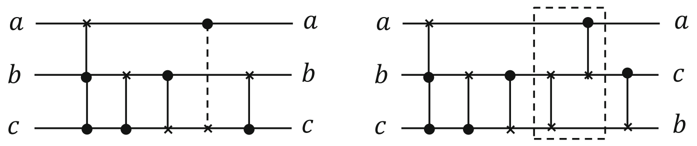
(a) Hierarchical **(b)** Flat

Fig. 16. 3-bit ripple adder

This criterion is supported experimentally and theoretically [BBC+95]. For example, consider the Hamming function of size 3 circuit given in Fig. 17:

In this circuit, the fourth gate (with dashed line) is applied to two inputs that are not adjacent (Fig. 17a). In order to meet the adjacency principle, we added a SWAP gate before the fourth gate as shown in Fig. 17b. Following this rule, we added SWAP gates to all quantum circuits taken from [RevLib] 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:

- **gf23mult** is to find the product of two elements of a field $\mathtt{GF}(2^3)$, $\mathtt{a} = \mathtt{a}_0 + \mathtt{a}_1\mathtt{x} + \mathtt{a}_2\mathtt{x}^2$ and $\mathtt{b} = \mathtt{b}_0 + \mathtt{b}_1\mathtt{x} + \mathtt{b}_2\mathtt{x}^2$ with the output, $\mathtt{ab} = \mathtt{c} = \mathtt{c}_0 + \mathtt{c}_1\mathtt{x} + \mathtt{c}_2\mathtt{x}^2$ written on the last 3 qubits.
- **2-to-4 decoder** that has 3 inputs and 4 outputs. If the enable qubit is zero, all outputs will be set to zero. If the enable qubit is one, then one of the four outputs that is selected based on the values of remaining two input qubits will be set to one.
- **hwb4** is the hidden weighted bit function with four inputs/outputs. Its output equals its input shifted left by the number of positions equal to the number of 1 in the input pattern.
- **ham3** is the size 3 Hamming optimal coding function.
- **mod5** is 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 1 in the input pattern is 2, 3 or 4.
- **nth_prime3_inc** is used to find primes with up to 3 binary digits.



(a) Without SWAP Gate **(b)** With SWAP Gate

Fig. 17. Hamming function of size 3

Table 4. Formalized quantum circuits

Circuit's name	Number of qubits	Number of gates without SWAP	Total number of gates	Success probability	Proof time (in seconds)
nth_prime3_inc	3	4	6	$5.9 * 10^{-8}$	18.71
ham3	3	5	6	$9.5 * 10^{-7}$	22
hwb4	4	12	22	$1.2 * 10^{-29}$	277.81
Shor's algorithm	4	8	8	$3.1 * 10^{-2}$	60
full adder	4	4	6	$3.7 * 10^{-9}$	38.32
mod5 (Grover's oracle)	5	8	18	$2 * 10^{-28}$	273.83
2-4 dec	6	3	8	$8.6 * 10^{-19}$	132.21
gf23mult	9	11	61	$1.7 * 10^{-108}$	13634.25
6sym	10	20	61	$1.7 * 10^{-102}$	20679.37
5-qubits adder	16	30	66	$7.49 * 10^{-96}$	33523.31

The result of the formal analysis of these circuits is given in Table 4.[8] The second column provides the number of gates in each circuit before adding the SWAP gate, and the third column provides the total number of gates. The case studies that were examined demonstrate that our proof strategy significantly improve on the degree of automation. Instead of using thousands of lines of HOL tactics to conduct the proof, we were able to achieve it automatically using the developed proof strategy. We believe that without the proposed proof strategy, we will not be able to formally analyze large circuits such as **6sym** and **gf23mult** that contain 61 gates.

From Table 4, it can be noticed that the proof time for the last two circuits increases dramatically and this is due to the larger size of the circuits which effects the frequency of tensor unfolding and folding. The efficient way to solve these kinds of issues is to employ the same techniques used for 3-bit Ripple Adder. The results presented in Table 4 show the efficiency of the implemented proof strategies.

A comparison of the number of 2-qubit gates required for Toffoli gate in our approach and the approaches in [NWD14, SBM06] shows that we achieved a more efficient design with two 2-qubit gates less. Note that this efficiency is a direct consequence of the low level modeling of the gates. On the other hand, it was not possible to carry out an accurate comparison of our work and existing CAD approaches based on the run-time given in Table 4 because all related approaches are developed for the behavioral level, whereas our proposed approach is interested in the physical level.

Nevertheless, we tried to make a rough comparison by developing Tables 5 and 6. In the second column of Table 5, we show the number of optical models, which analogs to the number of logical quantum bits metric in the behavioral approaches, used in each circuit of Table 4. In the third column of Table 5, we provide the number of 2-qubit gates (CZ or CNOT) that compose the circuits shown in Table 4. Table 5 shows the scalability of our approach to model and verify quantum circuit with around 200 2-qubit gates. In Table 6, we compare our approach in term of scalability with four recent CAD tools in the area of quantum computing. Note that the four tools are designed for analysis of quantum circuits at the behavioral level and our approach starts the analysis from the physical level going up to the behavioral level. Therefore, in the second column of Table 6 we provide the number of optical modes instead of the number of qubits for our approach. For the two techniques from [KAF⁺17] and [HSS⁺16], we could not find an accurate estimation for the maximum number of 2-qubit gates. It is important to note that the simulation in [WS14] took more than 30 days. On the other hand, the remaining tools provide very good runtime compared to our approach and this is due to the high level implementation and runtime-efficient programming languages that were used to build these tools. For instance, the tool in [ZW17] was implemented using C++. However, to formalize the mathematical logic necessary for reasoning about the low level implementation of quantum circuit (such as tensor product, infinite dimension Hilbert space, and all the related theorems in Sect. 5) we need a High-Order Logic theorem prover that does support a mature mathematical foundation which are not possible with programming languages such as C and C++. The matter of the slow down caused by HOL Light inference is conventional with interactive theorem prover. For example in [Har95] it was shown that inference causes a slow down of a factor of 50 over a direct Standard ML implementation for Binary Decision Diagrams (much more slower compared to C and C++).

In the following, we detail the analysis of Shor's algorithm, the full adder, and the boson-sampling circuit (the details about the other benchmark circuits can be found at [BM19]).

[8] Theorems proved about these circuits are available in the APPLICATIONS.ml of [BM19].

Table 5. Quantum circuits at the optical implementation level

Circuit's name	Number of optical modes	Number of 2-qubit gates
nth_prime3_inc	54	12
ham3	46	10
hwb4	200	48
Shor's algorithm	4	8
full adder	62	14
mod5 (Grover's oracle)	170	46
2-4 dec	132	30
gf23mult	734	190
6sym	696	169
5-qubit adder	652	158

Table 6. Comparison between several CAD tools and our approach

Approach's name	Max number of optical modes or qubits	Max number of 2-qubit gates
qHiPSTER (Intel, from [SSA16])	40	820
LIQUi| ⟩ (Microsoft, from [WS14])	31	18200
QX (from [KAF+17])	34	–
Quantum emulator (from [HSS+16])	36	–
Quantum simulator (from [ZW17])	100	100
Our approach	734	190

8.1. Shor's algorithm

Shor's integer factorization [Shor97] is a quantum algorithm which can break cryptographic codes that are widely employed in monetary transactions on the Internet [BV97]. The algorithm is capable of computing the two primes factor of a given integer number much faster than classical algorithms can do. Our objective here is to show the formal modeling and verification of a compiled version (i.e., a designed version to find the prime factors of a specific input) of Shor's factoring for the number 15 [PMO09] using the previously presented formalization. The task of the underlying circuit is to find the minimum integer r that satisfies a^r $mode$ $N = 1$, where $N = 15$ and a is a randomly chosen co-prime integer to N, in our case $a = 2$. r is called the order of a modulo N, from which we compute the desired prime factors; $(a^{\frac{r}{2}} - 1)$ and $(a^{\frac{r}{2}} + 1)$.

The circuit is composed of six Hadamard and two CZ gates, as shown in Fig. 18, and it has four inputs/outputs. The inputs are initialized to the state $|\,\psi\rangle_{in} = |\,0, 0, 1, 0\rangle_{x1f1f2x2}$. From the computed output, $|\,\psi\rangle_{out} = |\,.,.,.,.\rangle_{\ddot{x}1\ddot{f}1\ddot{f}2\ddot{x}2}$, we extract the variable $z = |\,.,.,0\rangle_{\ddot{x}1\ddot{x}2}$, then we obtain $r = a^z$ mod 15. Accordingly, we formally define the circuit structure in HOL as follows:

Definition 8.1 (Shor's Circuit)

$\vdash$ shor((x1 : sm), x2, f1, f2, f̈1, f̈2, ẍ1, ẍ2, ten, LH, LV, m_proj) $\Longleftrightarrow$ ($\exists$ a2 b2 a1 a3 a4 b3.
CZ_GATE(a1, a2, ẍ1, b2, ten, LH, LV, m_proj) $\wedge$ CZ_GATE(a3, a4, b3, ẍ2, ten, LH, LV, m_proj) $\wedge$
HADAMARD_GATE(f1, a2, ten, LH, LV) $\wedge$ HADAMARD_GATE(f2, a3, ten, LH, LV) $\wedge$
HADAMARD_GATE(x2, a4, ten, LH, LV) $\wedge$ HADAMARD_GATE(b2, f̈1, ten, LH, LV) $\wedge$
HADAMARD_GATE(x1, a1, ten, LH, LV) $\wedge$ HADAMARD_GATE(b3, f̈2, ten, LH, LV))

From this definition, we formally verify the operation of the circuit as follows:

Theorem 8.1 (Shor's Factoring of 15**)**

$\vdash$ let $|\,0, 0, 1, 0\rangle_{\mathtt{f1x1f2x2}}$ = tensor 4 (λ i. if i = 1 then LH f1 elseif i = 2
then LH x1 elseif i = 3 then LV f2 else LH x2) in
let $|\,0, 0, 0, 1\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}}$ = tensor 4 (λ i. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LH f̈2 else LV ẍ2) in
let $|\,0, 0, 1, 0\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}}$ = tensor 4 (λ i. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LV f̈2 else LH ẍ2) in
let $|\,1, 1, 0, 1\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}}$ = tensor 4 (λ i. if i = 1 then LV f̈1 elseif i = 2 then LV ẍ1
elseif i = 3 then LH f̈2 else LV ẍ2) in
let $|\,1, 1, 1, 0\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}}$ = tensor 4 (λ i. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LV f̈2 else LH ẍ2) in
shor(x1, x2, f1, f2, f̈1, f̈2, ẍ1, ẍ2, ten, LH, LV, m_proj) $\Longrightarrow$

$$|\,0, 0, 1, 0\rangle_{\mathtt{f1x1f2x2}} = \tfrac{1}{32}\ \%\ (|\,1, 1, 1, 0\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}} + |\,1, 1, 0, 1\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}} + |\,0, 0, 1, 0\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}} + |\,0, 0, 0, 1\rangle_{\mathtt{\ddot{f}1\ddot{x}1\ddot{f}2\ddot{x}2}})$$

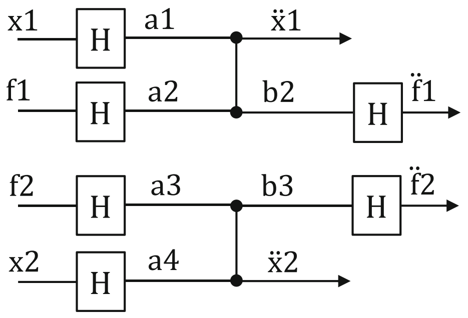


Fig. 18. Shor's factorization of number 15 circuit

Here, the circuit produces two categories of solutions: 1) $|\,000\rangle$ or $|\,100\rangle$, which are an expected failure of the algorithm; and 2) $|\,010\rangle$ or $|\,110\rangle \equiv z = 2$ or $z = 6$ which give $r = 4$ from which we obtain the 5 and 3 prime numbers.

8.2. Full adder

We consider a quantum full adder design inspired from [BTS+02], to which we have added two swap gates to exchange the qubits before feeding them to the Fredkin 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 8.2 (Full Adder Circuit)

```
⊢
FULL_ADDER((a0 : sm), a1, a2, a3, e0, e1, e2, e3, ten, LH, LV, m_proj) ⟺ (∃ b0 b1 b2 b3 c0 c1 c2 d1 d2.
  CNOT2(a0, a1, b0, b1, ten, LH, LV, m_proj) ∧ FREDKIN1(c0, d1, d2, e0, e1, e2, ten, LH, LV, m_proj) ∧
  CNOT2(a2, a3, b2, b3, ten, LH, LV, m_proj) ∧ CNOT2(b1, b2, c1, c2, ten, LH, LV, m_proj) ∧
  SWAP(b0, c1, c0, d1, ten, LH, LV, m_proj) ∧ SWAP(c2, b3, d2, e3, ten, LH, LV, m_proj))
```

Based on this definition, we formally verify the functionality of the quantum full adder in the general case where the two input values are added: $|\,x\rangle = x1\,|\,0\rangle_{a1} + x2\,|\,1\rangle_{a1}$ and $|\,y\rangle = y1\,|\,0\rangle_{a2} + y2\,|\,1\rangle_{a2}$, and the carry: $|\,z\rangle = z1\,|\,0\rangle_{a3} + z2\,|\,1\rangle_{a3}$.

Theorem 8.2 (Full Adder)

```
⊢ let input = tensor 4 (λ i. if i=1 then (x1%LH a1 + x2%LV a1) elseif i=2 then
  (y1%LH a2 + y2%LV a2) elseif i=3 then (z1%LH a3 + z2%LV a3) else LH a4) in
  let output1 = tensor 4 (λ i. if i=1 then LH b1 elseif i=2 then LH b2
  elseif i=3 then LH b3 else LH b4) in
  let output2 = tensor 4 (λ i. if i=1 then LV b1 elseif i=2 then LH b2
  elseif i=3 then LH b3 else LV b4) in
  let output3 = tensor 4 (λ i. if i=1 then LV b1 elseif i=2 then LH b2
  elseif i=3 then LV b3 else LV b4) in
  let output4 = tensor 4 (λ i. if i=1 then LH b1 elseif i=2 then LV b2
  elseif i=3 then LH b3 else LH b4) in
  let output5 = tensor 4 (λ i. if i=1 then LH b1 elseif i=2 then LH b2
  elseif i=3 then LV b3 else LV b4) in
  let output6 = tensor 4 (λ i. if i=1 then LV b1 elseif i=2 then LV b2
  elseif i=3 then LH b3 else LH b4) in
  let output7 = tensor 4 (λ i. if i=1 then LV b1 elseif i=2 then LV b2
  elseif i=3 then LV b3 else LH b4) in
  let output8 = tensor 4 (λ i. if i=1 then LH b1 elseif i=2 then LV b2
  elseif i=3 then LV b3 else LV b4) in
  FULL_ADDER(a1, a2, a3, a4, b1, b2, b3, b4, ten, LH, LV, m_proj) ⟹
  input = ((1/16)^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 cases in the outputs of the adder, as the combinations of the three inputs gives eight possibilities. Notice that the success probability of the quantum full adder is very low: $(\frac{1}{16})^7$. This completes the formal analysis of the quantum full adder. It is important to notice that using the bi-linearity of tensor product and the linearity of projection (Theorems 5.1 and 5.2 respectively) our approach can prove any generic input, such as the case above for the full adder circuit, because any generic input is a composition of a set of specific inputs.

8.3. Boson-sampling circuit

One of the most interesting benefits of beam splitters is to combine them with mirrors that reflect incident photon to realize multiphoton interferometer. Which are used in [WHL+17] to build quantum boson-sampling circuit. The 5-photon boson-sampling circuit is constructed using 9-mode interferometer circuit. Using our formalized library of quantum optics and developed HOL tactics, we verified several instances of the quantum boson-sampling circuits which are built using several interferometer circuits including 4-mode, 5-mode, and 6-mode interferometer circuits. For the sake of space, we only include here the formalization of 4-mode interferometer circuit since the remaining circuits are much bigger in term of size. The formalization of the remaining circuits are available at [BM19]. In Fig. 19, we show the optical circuit of 4-mode interferometer circuit composed of four mirrors and six beam splitters. The circuit has four inputs and four outputs. In below, we give the circuit formalization in HOL:

Definition 8.3 (4-mode interferometer Circuit)

```
⊢ Boson_four_Circuit((a : sm^N), e, ten) ⟺ (∃ b c d. mirror(ten, a(1), 1, b(1), 1) ∧
  mirror(ten, b(2), 1, c(2), 1) ∧ mirror(ten, c(3), 1, d(3), 1) ∧ mirror(ten, d(4), 1, e(4), 1) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, d(3), 1, c(4), 2, d(4), 1, e(3), 2) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, c(2), 1, b(3), 2, c(3), 1, d(2), 2) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, d(2), 2, b(4), 3, c(4), 2, e(2), 3) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, b(1), 1, a(2), 2, b(2), 1, c(1), 2) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, c(1), 2, a(3), 3, b(3), 2, d(1), 3) ∧
  is_beam_splitter(−0.9 * ii, 0.1, −0.42, 0.58 * ii, ten, d(1), 3, a(4), 4, b(4), 3, e(1), 4))
```

Using the above HOL definition of the circuit, we verify the 4-mode interferometer circuit for the input $| 0110\rangle$ where we have two photons in modes 2 and 3:

Theorem 8.3 (4-mode Interferometer Circuit: Input $| 0110\rangle$)

```
⊢ let input = tensor 4 (λ i. if i=2 then fock a(2) 1 elseif i=3 then fock a(3) 1
  else vac a(4)) in
  let output1 = tensor 4 (λ i. if i=2 then fock e(3) 1 elseif i=3 then fock e(2) 1
  else vac e(4)) in
  let output2 = tensor 4 (λ i. if i=4 then √2 % fock e(1) 2 else vac e(4))
  let output3 = tensor 4 (λ i. if i=1 then fock e(4) 1 elseif i=2 then fock e(3) 1
  else vac e(4)) in
  let output4 = tensor 4 (λ i. if i=3 then fock e(2) 1 elseif i=4 then fock e(1) 1
  else vac e(4)) in
  let output5 = tensor 4 (λ i. if i=1 then fock e(4) 1 elseif i=3 then fock e(2) 1
  else vac e(4)) in
  let output6 = tensor 4 (λ i. if i=3 then √2 % fock e(2) 2 else vac e(4))
  let output7 = tensor 4 (λ i. if i=2 then fock e(3) 1 elseif i=4 then fock e(1) 1
  else vac e(4)) in
  let output8 = tensor 4 (λ i. if i=1 then fock e(4) 1 elseif i=4 then fock e(1) 1
  else vac e(4)) in
  let output9 = tensor 4 (λ i. if i=1 then √2 % fock e(4) 2 else vac e(4))
  let output10 = tensor 4 (λ i. if i=2 then √2 % fock e(3) 2 else vac e(4))
  Boson_four_Circuit(a, e, ten) ⟹ input = −0.004% output1 − 0.0003 % output2 −
  0.17 % output3 − 0.005 % output4 + 0.15 % output5  − 0.016 % output6 −
  0.002 % output7 + 0.03 % output8 + 0.2 % output9 + 0.015 % output10
```

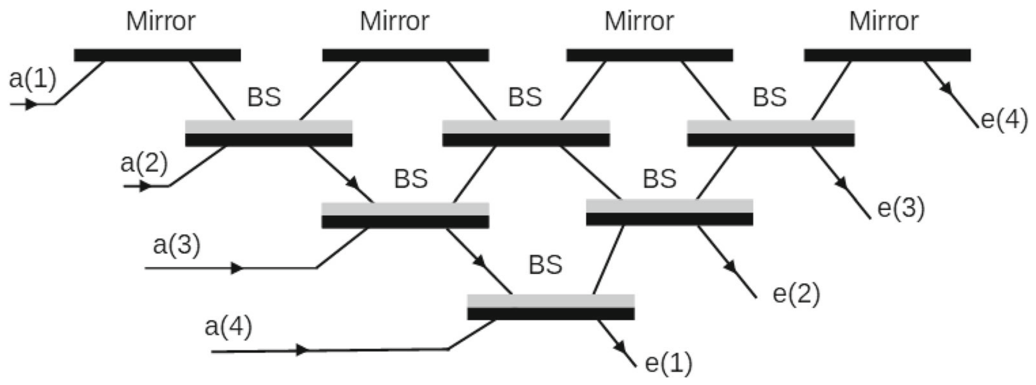


Fig. 19. Interferometer circuit: 4 optical modes

Notice that the probability weights for the outputs `output3`, `output5`, `output9` are at least 5 times bigger than the probability weights for the remaining outputs and are more than 30 times bigger than the probability weights of `output1`, `output2`, `output4`, `output7`. Thus, we may assume that the prospects of getting the outputs `output1`, `output2`, `output4`, `output7` are negligible compared to the outputs `output3`, `output5`, `output9`.

8.4. Success probability

Interesting insights that can be concluded from Table 4 are the very small success probabilities for most of the verified quantum circuits. These values show that it is near impossible to obtain the correct outputs from these circuits. These insights show the merit of our work in proving whether a quantum circuit is feasible or not. The empirical results in Table 4 validate the case that linear optics based quantum circuits have small success probabilities. Possible means to improve the success probabilities are to use teleportation, squeezed states and to combine linear optics with non-linear optics in building efficient quantum circuits.

Since a quantum circuit success probability decreases exponentially with the number of gates. As a consequence, in order to improve the success probability of a given circuit, it is preferred to investigate whether it is possible to build the circuit directly using optical elements instead of quantum gates. For example, in the case of 6-mode interferometer circuit which has 16 beam splitters and 6 mirrors we get an output with a probability around 0.2 which would not be feasible if the circuits was built a 2-qubit gate. If it is not possible to build a quantum circuit directly using optical elements, we try to avoid using quantum gates that have small success probability which include all 3-qubit gates and some 2-qubit gates such as the SWAP gate. For example, the circuit of `5`-qubit adder has 66 gates and has a success probability that is more than 10^6 times the success probability of `6`sym which contains 61 gates.

9. Conclusion and discussion

In this paper, we presented a framework for the modeling and the automated verification of optical quantum circuits based on the HOL Light theorem prover. In particular, we formalized a rich mathematical foundation that we utilized to build a rich optical quantum gates library and develop a proof automation strategy. In order to demonstrate the usability of the presented framework, we modeled and analyzed a set of well-known benchmark quantum circuits. We have applied our approach to verify the behavior of the boson-sampling circuit described in [WHL+17].[9] The analysis results 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 the quantum gates provides a crucial path to perform an effective optimization of quantum circuits, as the case of the Toffoli gate reported in this paper. One of the main features in the proposed approach is the capability to extract a quantum circuit success probability. To the best of our knowledge, the success probabilities of the analyzed benchmark circuits have not been investigated in existing literature. We can notice that all the success probabilities given in Table 4 are very low. This is due to the fact that the basic building blocks of these circuits are 2-qubit gates, which are probabilistic in nature. In contrast,

[9] The authors would like to thank Reviewer 1 for the suggestion on verifying the multiphoton boson sampling circuit.

existing CAD approaches do not consider quantum gates success probabilities which are crucial information about these gates. Moreover, the success probability is considered as an optimization parameter when it comes to choosing the effective design and employing error correction techniques.

The proposed approach provides a vital tool to model and automatically verify quantum circuits composed of hundreds of optical modes and hundreds of 2-qubit gates. Currently, we are investigating other means to improve the running time, which increases considerably when the numbers of qubits and gates increase. This is due to the low level analysis we are conducting, in which we use the explicit mathematical modeling of tensor product, where we need to unfold and fold the tensor product in order to apply the quantum gates transformations over the inputs. The proof time consumed by the unfolding and folding of a tensor product increases significantly when the size of the tensor product grows. Similarly, the number of times we do the unfolding and folding is proportional to the number of gates in the circuit. Furthermore, as we can notice from the previous theorems (e.g., Theorems 8.1 and 8.2), the underlying quantum circuits gates are listed in the assumptions of theorems for these circuits' behaviors. Therefore, if the number of gates increases this means the list of assumptions will expand which delays the HOL tactics application over the goals. To tackle these issues we succeeded in improving the proof time for the unfolding and folding lemmas. Additionally, we embedded a procedure to ease the proof for a complex quantum circuit by dividing it to several sub-circuits. Then, conducting the proofs for each sub-circuit to produce lemmas and using these lemmas for the original circuits proof. For example, we implemented specialized tactics instead of some generic HOL Light tactics such as *ARITH_TAC* in order to prove tensor product's folding and unfolding lemmas that improved runtime by 30%.[10]

The merit our approach provides is that it can help in building new quantum circuits designs that can either outperform existing designs or perform new functionalities in a more efficient fashion in terms of success probability. On the other hand, the success probabilities provided in Table 4 help in giving a valuable assessment on how far we are from reaching an efficient real world quantum computer. To summarize, compared to existing related work, the presented approach is comprehensive (i.e., rich quantum gates library), expressive (i.e., the analysis is done at the quantum physics level), and can be considered as complementary to existing quantum circuits modeling, synthesis and optimization approaches.

The framework described in this paper can be expanded to cover the formal synthesis of security protocols to quantum circuits, and also the optimization of these circuits. An immediate future plan is to investigate the existing CAD work for optimizing the number of SWAP gates added to a quantum circuit and to develop a HOL Light procedure. Moreover, the work presented here opens the doors to consider the formalization of some other methods such as quantum error correction for improving the success probabilities. We believe that our approach can be as well extended to check if given two well-formed quantum programs that were proven to be equivalent using Qwire [RPZ17b] or Hoare logic [Yin11]. Then, the two generated quantum circuits from the two programs are also equivalent. This helps when looking for a program such that the corresponding quantum circuit is optimum in term of success probability. Also, formally linking our work to these two works allows to see whether given an optimum quantum circuit is it possible to prove certain properties of the underlying program from which the circuit is generated.

We formalized the concepts of quantum optics over a general formalization of quantum mechanics rules that is currently part of the HOL Light repository. Furthermore, the presented framework and proof strategy are generic and not specific to quantum optics. Hence, our framework is directly applicable for formalizing other quantum computer realization technologies such as coherent states, squeezed states, quantum ion traps, and cavity quantum electrodynamic. In fact, the formalization of flip gate using coherent light was addressed in [MAT14]. The logical quantum bits $|1\rangle_L$ and $|0\rangle_L$ were realized using the coherent state and the vacuum state, respectively. The definition of coherent state required the formalization of the infinite summation over complex-valued functions which was done using the same functional spaces library we are using in the current work [MAT14]. Similarly, the same approach applies to the formalization of quantum ion traps which necessitates the formalization of the ground and excited states to describe Fock states and the formalization of the transformation of energy transition of ions between different electronic levels using physical elements such as laser pulses. However, such formalizations are not straightforward, due to the need to formalize the physical elements and the qubits representations in these technologies.

[10] The authors would like to thank John Harrison for helpful suggestions in implementing these tactics.

Acknowledgements

This work has been supported in part by the Canadian federal research funding agency NSERC, Grant No. RGPIN/06809-2015. We wish also to thank the Associate Editor and the Anonymous Referees for their comments which helped to improve the paper.

References

[BBC$^+$95] Barenco A, Bennett CH, Cleve R, DiVincenzo DP, Margolus N, Shor P, Sleator T, Smolin JA, Weinfurter H (1995) Elementary gates for quantum computation. Phys Rev A Am Phys Soc 52(5):3457–3467

[BM19] Beillahi SM, Mahmoud MY (2019) HOL-light source code. https://github.com/beillahi/FMV-QC-HOL.git

[BMT16a] Beillahi SM, Mahmoud MY, Tahar S (2016) Hierarchical verification of quantum circuits. In: Proceedings of NASA formal methods (NFM 2016), LNCS 9690. Springer, pp 344–352

[BMT16b] Beillahi SM, Mahmoud MY, Tahar S (2016) Optical quantum gates formalization in hol light. Technical Report, ECE Department, Concordia University, Montreal, QC, Canada

[BV97] Bernstein E, Vazirani U (1997) Quantum complexity theory. SIAM J Comput 26(5):1411–1473

[BL04] Black PE, Lane AW (2004) Modeling quantum information systems. In: Proceedings of quantum information and computation II. SPIE 5436, pp. 340–347

[BKN15] Boender J, Kammüller F, Nagarajan R (2015) Formalization of quantum protocols using Coq. In: Proceedings of international workshop on quantum physics and logic (QPL 2015), pp 71–83

[Bog47] Bogoliubov NN (1947) On the theory of superfluidity. J Phys (USSR) 11(1):23–32

[BMK10] Brown KL, Munro WJ, Kendon VM (2010) Using quantum computers for quantum simulation. Entropy 12(11):2268–2307

[BTS$^+$02] Bruce JW, Thornton MA, Shivakumaraiah L, Kokate PS, Li X (2002) Efficient adder circuits based on a conservative reversible logic gate. In: Proceedings of IEEE computer society annual symposium on VLSI (ISVLSI 2002), pp 83–88

[FDH04] Fowler AG, Devitt SJ, Hollenberg LCL (2004) Implementation of Shor's algorithm on a linear nearest neighbour qubit array. Quantum Inf Comput 4(4):237–251

[FGP13] Franke-Arnold S, Gay SJ, Puthoor IV (2013) Quantum process calculus for linear optical quantum computing. In: Proceedings of reversible computation (RC 2013), LNCS 7948. Springer, pp 234–246

[GFM10] Golubitsky O, Falconer SM, Maslov D (2010) Synthesis of the optimal 4-bit reversible circuits. In: Proceedings of design automation conference (DAC 2010), pp. 653–656

[GWD$^+$09] Grosse D, Wille R, Dueck GW, Drechsler R (2009) Exact multiple-control toffoli network synthesis with SAT techniques. IEEE Trans Comput Aided Des Integ Circuits Syst 28(5):703–715

[HSS$^+$16] Häner T, Steiger DS, Smelyanskiy M, Troyer M (2016) High performance emulation of quantum circuits. In: Proceedings of international conference for high performance computing, networking, storage and analysis (SC 2016), pp 866–874

[Har09] Harrison J (2009) Handbook of practical logic and automated reasoning. Cambridge University Press, Cambridge

[Har96] Harrison J (1996) HOL light: a tutorial introduction. proc. formal methods in computer-aided design (FMCAD 1996), LNCS 1166. Springer, pp 265–269

[Har95] Harrison J (1995) Binary decision diagrams as a HOL derived rule. Comput J 38(2):162–170,

[HSY$^+$06] Hung WNN, Song X, Yang G, Yang J, Perkowski MA (2006) Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans Comput Aided Des Integr Circuits Syst 25(9):1652–1663

[HSY$^+$04] Hung WNN, Song X, Yang G, Yang J, Perkowski MA (2004) Quantum logic synthesis by symbolic reachability analysis. In: Proceedings of design automation conference (DAC 2004), pp 838–841

[KAF$^+$17] Khammassi N, Ashraf I, Fu X, Almudever CG, Bertels K (2017) QX: a high-performance quantum computer simulation platform. In: Proceedings of design, automation and test in Europe (DATE 2017), pp 464–469

[KMN$^+$07] Kok P, Munro WJ, Nemoto K, Ralph TC, Dowling JP, Milburn GJ (2007) Linear optical quantum computing with photonic qubits. Rev Mod Phys Am Phys Soc 79(1):135–174

[KLM01] Knill E, Laflamme R, Milburn GJ (2001) A scheme for efficient quantum computation with linear optics. Nature 409:46–52

[Kni96] Knill EH (1996) Conventions for quantum pseudocode. Technical Report, LAUR-96-2724, Los Alamos National Laboratory

[LL05] Liang L, Li C (2005) Realization of quantum SWAP gate between flying and stationary qubits. Phys Rev A Am Phys Soc 72(2):024303

[LLW$^+$16] Liu T, Li Y, Wang S, Mingsheng Y, Zhan N (2016) A theorem prover for quantum hoare logic and its applications. CoRR. arXiv:1601.03835

[M15] Mahmoud MY (2015) Formal analysis of quantum optics. Ph.D. Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec

[MPT15] Mahmoud MY, Panangaden P, Tahar S (2015) On the formal verification of optical quantum gates in HOL. In: Proceedings of formal methods for industrial critical systems (FMICS 2015), LNCS vol 9128. Springer, pp 198–211

[MAT14] Mahmoud MY, Aravantinos V, Tahar S (2014) Formal verification of optical quantum flip gate. In: Proceedings interactive theorem proving (ITP 2014), LNCS vol 8558. Springer, pp 358–373

[MAT13] Mahmoud MY, Aravantinos V, Tahar S (2013) Formalization of infinite dimension linear spaces with application to quantum theory. In: Proceedings NASA formal methods (NFM 2013), LNCS vol 7871. Springer, pp 413–427

[MW95] Mandel L, Wolf E (1995) Optical coherence and quantum optics. Cambridge University Press, Cambridge

[Mil89] Milburn GJ (1989) Quantum optical Fredkin gate. Phys Rev Lett Am Phys Soc 62(18):2124–2127

[NC10] Nielsen MA, Chuang IL (2010) Quantum computation and quantum information. Cambridge University Press, Cambridge

[NWM$^+$16] Niemann P, Wille R, Miller DM, Thornton MA, Drechsler R (2016) QMDDs: efficient quantum function representation and manipulation. IEEE Trans Comput Aided Des Integr Circuits Syst 35(1):86–99

[NWD14] Niemann P, Wille R, Drechsler R (2014) Efficient synthesis of quantum circuits implementing Clifford group operations. In: Proceedings of Asia and South Pacific design automation conference (ASP-DAC 2014), pp 483–488

[Pac74] Packel EW (1974) Hilbert space operators and quantum mechanics. Am Math Mon 81(8):863–873

[PMO09] Politi A, Matthews JCF, O'Brien JL (2009) Shor's quantum factoring algorithm on a photonic chip. Science 325(5945):1221–1221

[RRG07] Ralph TC, Resch KJ, Gilchrist A (2007) Efficient Toffoli gates using qudits. Phys Rev A Am Phys Soc 75(2):022313

[RGM$^+$03] Ralph TC, Gilchrist A, Milburn GJ, Munro WJ, Glancy S (2003) Quantum computation with optical coherent states. Phys Rev A Am Phys Soc 68(4):042319

[RLB$^+$02] Ralph TC, Langford NK, Bell TB, White AG (2002) Linear optical controlled-NOT gate in the coincidence basis. Phys Rev A Am Phys Soc 65(6):062324

[RevLib] RevLib: an online resource for benchmarks within the domain of reversible and quantum circuits. http://www.revlib.org/ 2018.

[RPZ17a] Rand R, Paykin J, Zdancewic S (2018) QWIRE practice: formal verification of quantum verification in Coq. In: Proceedings of international conference on quantum physics and logic (QPL 2017), EPTCS, vol 266, pp 119–13

[RPZ17b] Rand R, Paykin J, Zdancewic S (2017) QWIRE: a core language for quantum circuits. In: Proceedings of ACM SIGPLAN symposium on principles of programming languages (POPL 2017), ACM, pp 846–858

[SWH$^+$12] Soeken M, Wille R, Hilken C, Przigoda N, Drechsler R (2012) Synthesis of reversible circuits with minimal lines for large functions. In: Proceedings of Asia and South Pacific design automation conference (ASP-DAC 2012), pp 85–92

[SWM12] Sasanian Z, Wille R, Miller DM (2012) Realizing reversible circuits using a new class of quantum gates. In: Proceedings of design automation conference (DAC 2012), pp 36–41

[SBM06] Shende VV, Bullock SS, Markov IL (2006) Synthesis of quantum logic circuits. IEEE Trans Comput Aided Des Integr Circuits Syst 25(6):1000–1010

[Shor97] Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput 26(5):1484–1509

[SSA16] Smelyanskiy M, Sawaya NPD, Aspuru-Guzik A (2016) qHiPSTER: the quantum high performance software testing environment. CoRR. arXiv:1601.07195

[VRM$^+$03] Viamontes GF, Rajagopalan M, Markov IL, Hayes JP (2003) Gate level simulation of quantum circuits. In: Proceedings of Asia and South Pacific design automation conference (ASP-DAC 2003), pp 295–301

[WHL$^+$17] Wang H, He Y, Li YH, Su ZE, Li B, Huang HL, Ding X, Chen MC, Liu C, Qin J, Li JP, He YM, Schneider C, Kamp M, Peng CZ, Höfling S, Lu CY, Pan JW (2017) High-efficiency multiphoton Boson sampling. Nat Photon 11:361–365,

[WS14] Wecker D, Svore KM (2014) LIQUi$|\,\rangle$: A software design architecture and domain-specific language for quantum computing. CoRR. arXiv:1402.4467

[WLD14] Wille R, Lye A, Drechsler R (2014) Optimal SWAP gate insertion for nearest neighbor quantum circuits. In: Proceedings of Asia and South Pacific design automation conference (ASP-DAC 2014), pp 489–494

[YM10] Yamashita S, Markov IL (2010) Fast equivalence-checking for quantum circuits. Quantum Inf Comput 10(9):721–734

[Yin11] Ying M (2011) Floyd–Hoare logic for quantum programs. ACM Trans Program Lang Syst 33(6):19:1–19:49

[ZW17] Zulehner A, Wille R (2017) Advanced simulation of quantum computations. CoRR. arXiv:1707.00865

Received 11 January 2018
Accepted in revised form 24 February 2019 by Gerwin Klein
Published online 15 March 2019