Optical Quantum Gates Formalization in HOL Light

Sidi Mohamed Beillahi, Mohamed Yousri Mahmoud, and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, QC, Canada
{beillahi, mo_solim, tahar}@ece.concordia.ca

Technical Report

February, 2016

Abstract

Quantum computers are expected to have high capability, and are considered good candidates to replace classical cryptography and supercomputing systems. Among many implementations, quantum optics systems provide a promising platform to implement quantum computers, since they link quantum computation and quantum communication in the same framework. However, the modeling, analysis, and verification of quantum optics systems is challenging since they exploit quantum physics laws that require complex mathematical foundations. In this report, we present the formalization of a set of optical quantum gates built hierarchically in higher-order logic. We also describe two applications based on this set of gates, which are Shor's factorization and a quantum full adder.

1 Introduction

Quantum mechanics is concerned with studying microscopic physics systems. The main difference between classical and quantum mechanics lies in two principles: superposition (i.e., an object can be in two places at the same time) and entanglement (i.e., two objects in remote locations without physical connection can be instantaneously connected). Quantum computing systems are computation systems (or machines) which employ quantum mechanics principles. The study of quantum computers started in 1982 when it was proved that classical machines (classical means non-quantum) cannot simulate quantum physics efficiently [32].

Quantum computers promise to increase the efficiency of solving problems which classical computers cannot handle, such as integers factorization. Hence, they are expected to be used in high computing domains and to secure communications (e.g., monetary transactions on the Internet) [18, 21].

Since the early days of quantum computing, scientists are investigating several approaches for building these systems and finding solutions for the main practical limitations for building quantum computers such as: initialization of quantum bits (i.e., two-state quantum systems), measurements, and decoherence (i.e., unwanted interaction between a quantum system and its environment). In 2000, DiVincenzo [10] proposed five criteria, which should be satisfied by a mature quantum computer. These criteria were used in [40] to evaluate the progress in the most widely proposed approaches to implement quantum computers. Some of these approaches have shown significant progress, including ion trap, and superconductors, and optics. In this work we are focusing on quantum optics computing systems where the quantum bits (qubits) are considered as single photons that can be in two different modes which is called a dual-rail representation [23].

Quantum computers like any other physical or engineering systems, need Computer Aided Design (CAD) tools to facilitate their design and deployment in real life application. Whereas, quantum computers can best realize their computational potential by invoking quantum superposition and parallelism (e.g., a quantum computer performs many computations simultaneously). The current languages and tools are based on Boolean algebra and thus cannot exploit this feature and benefit of the full potential of quantum systems. Therefore, building tools that can model, synthesize, verify, and ensure the full functionalities of quantum algorithms and computers is also as important as building the physical quantum computer. Nowadays, many computer scientists are working to develop algorithms which can exploit quantum features, including languages which can be supported on quantum computers, and tools that can verify and simulate the functionalities of quantum systems.

Formal methods [44] based techniques allow to develop a mathematical model for the given system and analyze this model using computer-based mathematical reasoning. Hence they have the potential to improve the analysis of quantum systems. Particularly higher-order logic (HOL) theorem proving [19], since it offers rich mathematical foundation to fulfil the main requirement for modeling quantum systems. Because quantum logic is mainly based on Hilbert spaces theory which is not available in CAD tools that are used for the analysis of Boolean based machines. HOL theorem proving tools have been widely employed to verify generic properties of a wide class of software and hardware systems. Within the theorem proving tool, the system properties are described as theorems in higher-order logic [19]. Then once these theorems are proved inside the tool, we say that the system is formalized. The proposed work is conducted within HOL Light interactive theorem prover because it

provides rich multivariate analysis libraries [15]. HOL Light is written in the functional programming language Objective CAML (OCaml) [20]. The main components of the logical kernel of HOL Light are: types, terms, theorems, rules of inference, and axioms. Proofs in HOL Light are based on the concepts of tactics and tacticals that break goals into simple subgoals. There are many automatic proof and decision procedures available in HOL Light which help the user in directing the proof to the end [20]. HOL Light has been successfully used as a verification framework for both software and hardware verification as well as a platform for the formalization of pure mathematical theorems. In Appendix A, we provide the mathematical interpretations of some HOL Light notations that will be used in this report.

Recently, a comprehensive linear algebra library was formalized in HOL Light [26]. Using this library, the authors formalized optical coherent states, beam splitters, phase shifters, flip gate, quantum optics single mode, Mach-Zehnder interferometer, and controlled-not (CNOT) gate [27]. In our work, we are building upon the existing work in HOL Light. First, we enrich the mathematical formalization with tensor product to model quantum optics system in multi-modes, linear projection and tensor product projection to model quantum measurement. Second, using these mathematical tools we model and verify a hierarchical library of quantum gates and circuits that include the most widely used quantum gates in quantum computating. In this report, we demonstrate the hierarchical library that has been formalized within HOL Light. Afterwards, we show the application of this library to a compiled version of Shor's factorization of number 15 and a quantum full adder.

The rest of the report is organized as follows: In Section 2, we present the related work. Then, we highlight an overview of tensor products, quantum mechanics, quantum computers, and quantum optics in Section 3. In Sections 4 and 5, we describe the formal modeling and verification of quantum primitive gates and quantum circuits, respectively. In order to demonstrate the practical effectiveness of our work, in Section 6 we present the verification of quantum algorithms. Finally, we conclude the report in Section 7.

2 Related Work

We can subdivide the works in CAD approaches related to quantum computing to three main categories: quantum circuits simulations, synthesis of quantum circuits, and the modeling and verification of quantum circuits.

2.1 Quantum Circuits Simulations and Emulations

Numerical simulations are used to study the behaviour of physical systems through computer assisted calculation. Nowadays, numerical simulations are the most popular approach in CAD. However, due to the inherent quantum circuits complexities, which rely on Hilbert spaces and quantum principles, numerical simulations are incomplete. Since in order to measure the effect of different initial conditions or parametric variation over the quantum circuit, it is necessary to perform exhaustive tests that do not cover the complete space. Nevertheless, a number of approaches have been persuaded to simulate quantum circuits using emulation and numerical simulation. For example, several methods and tools for numerical simulation have been proposed, where quantum gates are described as matrices and applied to quantum states using matrix-vector multiplication [43, 30, 11]. In these work, the simulations were performed at the gate level without modelling the physical elements of

the quantum gates. In fact quantum circuits are built using different elementary physical devices, which in turn limit the ability of such tools to accurately perform the analysis of these circuits. On the other hand, in [5] the authors used an FPGA emulator for the emulation of quantum circuits, however, this approach is limited by the size of the FPGA under consideration which constrains it to circuits with a limited number of qubits.

2.2 Quantum Circuits Synthesis

Motivated by the capacity of quantum computers, researchers and engineers started to actively consider the logic synthesis of quantum circuits. In order to design scalable quantum computers, automatic methods for computer-aided design and synthesis are required. Accordingly, efficient quantum circuit synthesis became an active field of research in design automation. Several Approaches addressing quantum circuits synthesis directly [17, 41], exploiting synthesis methods for reversible circuits [39, 12, 37] or synthesis of quantum circuit that are using a fixed set of gates such as Clifford group circuits [31] have been proposed. Another area of research in quantum circuits design is the optimization of the number of SWAP gates (i.e., a 2-qubit gate that switches the states of the two inputs) inserted in a quantum circuit where quantum gates are restricted to work on adjacent qubits (nearest neighbor quantum circuits). In this work, the swap gates are inserted in a quantum circuit in order to move the qubits to be adjacent to each other. The drawback of these generic approaches is that synthesis and optimization of the quantum circuits is at the behavioral level, and the physical design of the elementary gates was not tackled which limits these work to find the optimized circuit. For example, in [39, 12], the Toffoli gate circuit was synthesized into five 2-qubit gates, however, it is possible to implement this gate in quantum optics using only three 2-qubit gates as shown in [34], by utilizing the photons polarization structure of the qubits. In another related work [16], the authors used model checking to formally synthesise quantum circuits based on a fixed set of 1-qubit and 2-qubits gates. The authors have demonstrated the optimized design for Toffoli and Fredkin gates, however, this remains valid only when we are dealing with circuits at the behavioral level. Moreover, another work [13] using formal techniques: Boolean satisability (SAT) and SAT modulo theory (SMT) for the synthesis of Toffoli networks at the behavioral level which limits the underlying approach in producing the most optimized design.

2.3 Modeling and Verification of Quantum Circuits

In [38], Binary Decision Diagrams (BDD) has been used for the equivalence checking of reversible quantum circuits, by classifying quantum circuits into two types: properly-quantum and non-properly-quantum. A circuit is properly-quantum if it contains quantum gates that exploit superposition quantum, e.g., Hadamard gate. Thus this limits significantly the underlying technique to perform the complete modeling analysis of any quantum circuits. Another related work proposed in [36] to model and analyse quantum communication protocols using the CWB-NC model checker. The authors have verified the quantum coin-flipping protocol. The proposed technique can be used only for protocols that can be described in finite-state models. However, the focus of the work in [36] is on quantum cryptography (i.e., the use of quantum mechanics to encrypt data) not quantum computing which is the main target of our work. The most related work to ours is [35], in which a special quantum process calculus is used to model linear optical quantum systems. As an application, the authors modeled and verified the controlled-not gate. The main limitation of this work is that the

authors consider beam splitters parameters as real numbers, whereas in the general context of quantum optics they can be complex numbers as in the case of quantum interferometer [28]. So far, the proposed process calculus has not been mechanized in a verification tool.

3 Preliminaries

In this section, we will have a brief coverage of quantum mechanics in single mode and in multi-modes using tensor product. Then, we will give a description of a branch of quantum mechanics, namely quantum computing. In the following sub-section, we will show one of the approaches under investigation for implementing a scalable quantum computing machine, namely quantum optics.

3.1 Quantum Mechanics

Quantum mechanics can be described as the study of physics at very small length and microscopic scales. In quantum theory, physical particles have wavelike properties and their behaviors are governed by the Schrodinger equation. Generally, quantum mechanics is composed of many physics aspects such as: the quantum measurement, Bell's theorem, and wave-particle duality. The two main mathematical objects in quantum mechanics are wavefunctions and operators. The wavefunction describes the system of interest (such as a spin or an electron); if the wavefunction is known, it is possible to extract all the properties of the system: The square of the wavefunction gives the probability of finding the particle at that point. During the mathematical analysis of quantum mechanics, a wavefunction is represented by a "ket" $|\ldots\rangle$; labels used to represent the wavefunctions as follows:

$$f_q \text{ is written as } |q\rangle \text{ or sometimes } |f_q\rangle \tag{1}$$

The complex conjugate of a wavefunction is written as a "bra" $\langle\ldots|$:

$$(f_{q'})^* \text{ is written as } \langle q'| \tag{2}$$

Another rule is that if a bra appears on the left side and a ket on the right side, integration over $\mathrm{d}\tau$ is implied:

$$\langle q'|q\rangle \text{ implies } \int (f^*_{q'}) f_q d\tau \tag{3}$$

where the notation $d\tau$ is taken in quantum mechanics to mean integration over the full range of all relevant variables. Mathematically, we call this notation the inner product which is a multiplication operation that maps any pair of vectors of a linear complex vector space onto a number. A linear complex vector space in which an inner product is defined, is called the Hilbert space. Traditionally, quantum mechanical operators and wavefunctions can be represented as linear transformations and functions in Hilbert space, respectively. In order to describe a multi-states quantum system, we use the notion of tensor product in the next section.

3.2 Tensor Product

In this section, we give an overview of the tensor products of Hilbert spaces. Given two complex vector spaces V and W, then the complex vector space $V \otimes W$ is called the tensor

product, whose elements are linear combinations of vectors of the form $v \otimes w$, with $v \in V$, $w \in W$. To generalize, suppose we have n vector spaces over $\mathbb{C}$ (i.e., V_k, $k \in [1,\ n]$) then the tensor product $V_1 \otimes V_2 \otimes ... \otimes V_n$ is a new complex vector space:

$$v_1 \otimes v_2 \otimes ... \otimes v_n \in V_1 \otimes V_2 \otimes ... \otimes V_n when\ \forall k \in [1,\ n].\ v_k \in V_k. \tag{4}$$

Each tensor product is bilinear and all the elements of the tensor vector space $V_1 \otimes V_2 \otimes ... \otimes V_n$ should satisfy the following properties of multinearity: 1. If the vector v_k is scaled by a complex scalar number, this is equivalent to scaling the main vector.

$$v_1 \otimes ... \otimes (av_k) \otimes ... \otimes v_n = a(v_1 \otimes ... \otimes v_k \otimes ... \otimes v_n), a \in C. \tag{5}$$

2. If the vector v_k is a superposition of two vectors, then the main vector is also a superposition.

$$v_1 \otimes ... \otimes (v_{k1} + v_{k2}) \otimes ... \otimes v_n = v_1 \otimes ... \otimes v_{k1} \otimes ... \otimes v_n + v_1 \otimes ... \otimes v_{k2} \otimes ... \otimes v_n. \tag{6}$$

The main usage of tensor product is to describe multi-states quantum system. For example, given a 2-particle quantum system where $v \in V$ and $w \in W$ describe the state for the first and second particles, respectively, then the element $v \otimes w \in V \otimes W$ describes the joint states of the two particles. The element $v_1 \otimes v_2 \otimes ... \otimes v_n$ is a vector in a tensor vector space $V_1 \otimes V_2 \otimes ... \otimes V_n$ that represents the description of the quantum states of the system of n single particles. Moreover, for a multi-particle quantum system the operators are also tensor products of single state operators. For example, for n operators applied on a quantum state of n particles we will have the following formulas:

$$(\hat{a}_1^\dagger \otimes \hat{a}_2^\dagger \otimes ... \otimes \hat{a}_n^\dagger)\ (|\psi\rangle_1 \otimes |\psi\rangle_2 \otimes ... \otimes |\psi\rangle_n) = \hat{a}_1^\dagger |\psi\rangle_1 \otimes \hat{a}_2^\dagger |\psi\rangle_2 \otimes ... \otimes \hat{a}_n^\dagger |\psi\rangle_n \tag{7}$$

Furthermore, the tensor product of linear operators is also multi-linear and satisfies Equations 5 and 6.

3.3 Quantum Computing

The observation that certain quantum mechanical effects cannot be simulated efficiently on a classical computer leads to thinking that quantum based computation can be more efficient than the classical computation. This speculation was justified when a polynomial time quantum algorithm for factoring integers was developed. Since then, a tremendous amount of research has been conducted on quantum information hoping to solve some problems that cannot efficiently be solved by classical algorithms. In quantum systems, the computational space (the size of Hilbert space) increases exponentially with the size of the system which enables exponential parallelism. This parallelism can lead to exponentially faster quantum algorithms than possible with current machines. However, accessing the results, which requires measurement that may destroy the information, requires new non-traditional programming techniques.

The main element of quantum computer is a quantum bit (qubit), which is a unit vector in a two dimensional complex vector space for which a particular basis, denoted by $|0\rangle, |1\rangle$, has been fixed. In one of the realization of quantum systems, the orthonormal basis $|0\rangle$ and $|1\rangle$ correspond to the $|\uparrow\rangle$ and $|\rightarrow\rangle$ polarizations of a photon, respectively. Generally, all measurements in two dimensional quantum systems are made with respect to the standard

basis for quantum computation, $|0\rangle, |1\rangle$. For the purposes of quantum computation, the basis states $|0\rangle$ and $|1\rangle$ are used to represent the classical logic bit values 0 and 1, respectively. Unlike classical bits, however, qubits can be in a superposition of $|0\rangle$ and $|1\rangle$ such as:

$$a|0\rangle + b|1\rangle, \text{ where } a, b \in \mathbb{C} \text{ and such that } |a|^2 + |b|^2 = 1. \tag{8}$$

Furthermore, the probability that the measured value is $|0\rangle$ is $|a|^2$ and the probability that the measured value is $|1\rangle$ is $|b|^2$. In quantum computation the resulting state space for a system of n qubits is a space of 2^n, however, in classical computation the possible states of a system of n bits, form a vector space of $2n$ dimensions. This demonstrates the exponential speed-up of computation on quantum computers over classical computers. In quantum computing, the states of qubits are combined using the tensor product. For example, the state space for two qubits, each with basis $|0\rangle, |1\rangle$, has basis $|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle$ which can be written more compactly as $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. More generally, an n-qubit system has 2^n basis vectors. On the other hand, the states which cannot be decomposed into their individual components are called entangled states. These states represent situations that have no classical counterpart, and for which we have no intuition. The state $|00\rangle + |11\rangle$ is an example of a quantum state that cannot be described in terms of the state of each of its components separately. In other words, we cannot find a1, a2, b1, b2 such that $(a1|0\rangle + b1|1\rangle) \otimes (a2|0\rangle + b2|1\rangle) = |00\rangle + |11\rangle$ since $(a1|0\rangle + b1|1\rangle) \otimes (a2|0\rangle + b2|1\rangle) = a1a2|00\rangle + a1b2|01\rangle + b1a2|10\rangle + b1b2|11\rangle$ and $a1b2 = 0$ implies that either $a1a2 = 0$ or $b1b2 = 0$. Note that it would require vast resources to simulate even a small quantum system on traditional computers.

3.4 Quantum Optics

Quantum optics is considered as one of the rich and promising approaches under investigation for realizing quantum computers [40, 22]. This is because, photons decohere slowly, photons can move quickly (at the speed of light), and photons can be experimented with at room temperature. For quantum optics, the state of a quantum system is a probability density function which provides the probability of the number of photons inside the optical beam, typically written as $|\psi\rangle$. The corresponding linear Hilbert space is the space of square integrable functions and the inner product is the complex Lebesgue integral. The two main quantum optics operators are: annihilation operator (annihilator) and creation operator (creator) for photons. Because they lower and increase the photon count by 1, respectively. The two operators satisfy the Boson commutation relation: $[\hat{a}_j, \hat{a}_j^\dagger] = 1$. Another quantum optics operator is the number operator: $\hat{n}_j = \hat{a}_j^\dagger \hat{a}_j$. The most elementary optical quantum states, namely fock states which are pure states (means that they cannot be modeled by other states). The fock state $|n\rangle_j$ describes the number of photons in a mode j, also they are eigenstates of the number operator $\hat{n}_j$, i.e., $\hat{n}_j |n\rangle_j = n_j |n\rangle_j$ with integer eigenvalues. The set of fock states represents an orthonormal basis for the linear functional space. Therefore any quantum state $|\psi\rangle$ of a given mode is a superposition of fock states, i.e., $|\psi\rangle \equiv \sum_n |\psi\rangle_n |n\rangle$. The main operators which act on fock states are the annihilation and creation operators:

$$\hat{a}_j |n\rangle_j = \sqrt{n}\,|n-1\rangle_j \text{ and } \hat{a}_j^\dagger |n\rangle_j = \sqrt{n+1}\,|n+1\rangle_j, \text{ respectively.} \tag{9}$$

For the case of a quantum state $|\Psi\rangle$ where there is N single modes $|\psi\rangle_j$ ($|\psi\rangle_j$ can be independent from each other or entangled between each other), the state $|\Psi\rangle$ is described as the

tensor product of the states $|\psi\rangle_j$:

$$|\Psi\rangle \equiv \bigotimes_{j=1}^{N} |\psi\rangle_j = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \ldots \otimes |\psi\rangle_N \tag{10}$$

In optical quantum computing systems the qubit is usually taken as a single photon that can be in two different modes of polarization, $|0\rangle_L = |1\rangle \bigotimes |0\rangle \equiv |1,0\rangle$ and $|1\rangle_L = |0\rangle \bigotimes |1\rangle \equiv |0,1\rangle$ which is called dual_rail. When the two modes represent the internal polarization degree of freedom of the photon ($|0\rangle_L = |H\rangle$ and $|1\rangle_L = |V\rangle$), we call it a polarization qubit. In [22] the authors showed that given single photon sources and single-photon detectors, linear optics alone would suffice to implement efficient quantum computation. Moreover, most of existing universal quantum gate architectures are built using only linear-optical networks, a linear optical element is such that the mode transformation under evolution, U can be described by matrices u and v, which transform the modes linearly, such that, $\hat{a}_j^\dagger \rightarrow \sum_k u_{kj}\hat{a}_k^\dagger + v_{kj}\hat{a}_k$. The most widely employed linear optics elements are beam splitters and phase shifters:

3.4.1 Phase Shifter

An important optical component is the single-mode phase shifter which provides a phase shift in a given mode: $\hat{a}_{o1}^\dagger = e^{i\theta}\hat{a}_{i1}^\dagger$. A phase shifter ($P_\theta$, θ is the angle of the phase shifter) is a passive linear optical element with Hamiltonian: $H_{P_\theta}(\theta) = \theta\hat{a}^\dagger\hat{a}$. Thus, the Hamiltonian is proportional to the number operator, which means that the photon number is conserved. Phase shifter transformation is associated with a unitary operator described by: $U(P_\theta) = e^{i\phi}$. Physically, a phase shifter is a slab of transparent material with an index of refraction that is different from that of free space. We formally define the phase shifter transformation in HOL as follows:

Definition 1 (Phase Shifter).

1 `phase_shifter(ten, f, i1, m1, o1, m2)` $\Leftrightarrow$
2 `is_sm i1` $\wedge$ `is_sm o1` $\wedge$ `w i1` $=$ `w o1` $\wedge$ `vac i1` $=$ `vac o1` $\wedge$
3 `pos ten (cr i1) m1` $=$ `e`$^{(-\mathtt{j}*\mathtt{f})}$ $\%$ `pos ten (cr o1) m2` $\wedge$
4 `pos ten (anh i1) m1` $=$ `e`$^{(\mathtt{j}*\mathtt{f})}$ $\%$ `pos ten (anh o1) m2`)

Note that the formal definition of phase shifter relates the input operators in terms of the output operators (see Lines 3 and 4). In Line 1, the parameters {m1,m2} define the order of each mode in the whole circuit. Line 2 ensures that the two modes are proper single modes, and working with the same frequency and vacuum state (i.e., the state of zero photons).

3.4.2 Beam Splitter

Beam splitter is a two mode 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. Beam splitters are central components in linear optical quantum computing systems. 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 the two incoming modes on either side of the beam splitter be denoted by a_{i1} and a_{i2} and the outgoing modes by a_{o1} and a_{o2}, as shown in Figure 1. Its transformation is then:

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

The Hamiltonian H_{BS} of the beam-splitter transformation is given as: $H_{BS} = \theta e^{i\varphi}\hat{a}^{\dagger}_{i1}\hat{a}_{i2} + \theta e^{-i\varphi}\hat{a}_{i1}\hat{a}^{\dagger}_{i2}$. Since the Hamiltonian operator H_{BS} commutes with the total number operators $[H_{BS}, \hat{n}] = 0$, then the photon number is conserved in the beam splitter. The commonly used beam splitter is the one of relative phase $\varphi = 0$.

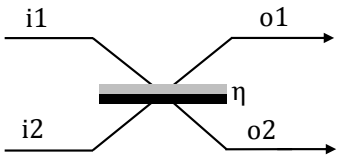

Fig. 1: Schematics of Beam Splitter

Accordingly, we formally define the beam splitter transformation in HOL Light as follows:

Definition 2 (Beam Splitter).

```
1 is_beam_splitter (p1,p2,p3,p4,ten,i1,m1,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 ∧
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
```

Similar to phase shifter the formal definition of beam splitter relates the inputs operators in terms of the outputs operators (see Lines 5, 6, 7, and 8). The parameters {p1,p2,p3,p4} are the inverse of 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.

Many universal quantum gates have been implemented using linear optics elements: 1-qubit gates such as: Hadamard, Flip, Non-linear sign gates, 2-qubits gates such as: controlled-phase (CZ) and controlled-not (CNOT) gates which can be constructed using Non-linear sign (NS) gates [23], and 3-qubits gates such as Toffoli and Fredkin gates which can be constructed using CZ, Hadamard, and Flip gates. Therefore, quantum linear optics does indeed maintain DiVincenzo's fourth criteria for building scalable quantum computing machines [23]. Although most the existing implementation of quantum gates in linear optics are probabilistic, however, it has been proposed to use teleportation to improve the efficiency to deterministic (near-deterministic) by teleporting the successful gate outputs [23].

4 Verification of Quantum Primitive Gates

In this section, we show in detail the formal modeling and verification of a set of quantum primitive gates. Specifically, we formalize the Hadamard, non-linear sign and flip gates. All the mentioned transformations are fully implemented using linear optics elements.

4.1 Hadamard Gate

The Hadamard gate [25] is an one-qubit universal gate which exploits quantum superposition to create a new state, where we have a combination of $|0\rangle$ and $|1\rangle$ with the same probability. For example, if the possible 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 states and to add random information to a quantum circuit. The authors in [2] implemented the Shor's algorithm for factoring 15 using six Hadamard gates (see Section 6.1). In this section, we present the formal verification of the Hadamard gate in HOL Light as an example of a single qubit gate that can be constructed by using a beam splitter and a phase shifter together. The dual-rail representation is used to describe the qubit. The gate circuit is shown in Figure 2 with a beam splitter ($\eta = \frac{1}{\sqrt{2}}$) and a phase shifter of angle $\theta = \pi$ ($\varphi = 0$). The formal definition of

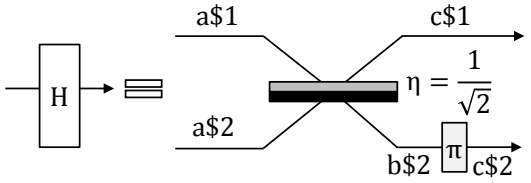

Fig. 2: Schematics of Hadamard Gate

the gate structure in HOL is as follows:

Definition 3 (Hadamard Gate).

$\texttt{HADAMARD\_GATE}\ (\texttt{a},\texttt{c},\texttt{ten},\texttt{LH},\texttt{LV}) \Leftrightarrow (\forall\ \texttt{b.phase\_shifter}\ (\texttt{ten},\pi,\texttt{b\$2},2,\texttt{c\$2},2)\ \wedge$
$\texttt{is\_beam\_splitter}(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\texttt{ten},\texttt{a\$1},1,\texttt{a\$2},2,\texttt{c\$1},1,\texttt{b\$2},2))$

where `is_beam_splitter` accepts the beam splitter parameters: input ports $(a\$1, 1, a\$2, 2)$, output ports $(c\$1, 1, b\$2, 2)$, tensor operator `ten`, and beam splitter 2×2 matrix($\frac{1}{\sqrt{2}}$, $-\frac{1}{\sqrt{2}}; \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$). Note that, for such parameters, the inputs and outputs will be related as follows: $\hat{a\$1} = \frac{1}{\sqrt{2}}\ \%\ \hat{c\$1} - \frac{1}{\sqrt{2}}\ \%\ \hat{b\$2}$ and $\hat{a\$2} = \frac{1}{\sqrt{2}}\ \%\ \hat{c\$1} + \frac{1}{\sqrt{2}}\ \%\ \hat{b\$2}$. `phase_shifter` accepts the parameters: input port $(b\$2, 2)$, output port $(c\$2, 2)$, tensor operator and phase shifter angle π. `LH` and `LV` are employed to describe the representation of qubits using the photon polarization vertical or horizontal, where `LV a$1` (resp., `LH a$1`) 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_GATE` takes as parameters all the gate input/output ports (a, c), and the tensor operator. Using this definition, we formally verify the result of applying the Hadamard gate on the two possible inputs: $|0,1\rangle_a$ and $|1,0\rangle_a$:

Theorem 1 (Hadamard Input: $|1\rangle_L \equiv |0,1\rangle_a$).

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{HADAMARD\_GATE}(\texttt{a},\texttt{c},\texttt{ten},\texttt{LH},\texttt{LV})\ \texttt{in}$
$\texttt{let}\ |0,1\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda\texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{a\$2}}\ \texttt{else}\ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$
$\texttt{let}\ |1,0\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda\texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ |1\rangle_{\texttt{c\$1}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$
$\texttt{let}\ |0,1\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda\texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{c\$2}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$
$\texttt{constraints} \Rightarrow |0,1\rangle_{\texttt{a}} = \frac{1}{\sqrt{2}}\%\,|1,0\rangle_{\texttt{c}} - \frac{1}{\sqrt{2}}\%\,|0,1\rangle_{\texttt{c}}$

Theorem 2 (Hadamard Input: $|0\rangle_L \equiv |1,0\rangle_a$).

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{HADAMARD\_GATE}(\texttt{a},\texttt{c},\texttt{ten},\texttt{LH},\texttt{LV})\ \texttt{in}$
$\texttt{let}\ |1,0\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ |1\rangle_{\texttt{a\$1}}\ \texttt{else}\ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$
$\texttt{let}\ |1,0\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ |1\rangle_{\texttt{c\$1}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$
$\texttt{let}\ |0,1\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{c\$2}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$
$\texttt{constraints} \Rightarrow |1,0\rangle_{\texttt{a}} = \frac{1}{\sqrt{2}} \% |1,0\rangle_{\texttt{c}} + \frac{1}{\sqrt{2}} \% |0,1\rangle_{\texttt{c}}$

Note that, we did not use the projection because we do not employ ancilla, therefore, there will be no detection required. As shown in Figure 2 the optical implementation of Hadamard gate has two inputs to describe the input qubit. 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, $\texttt{Hadamard\_In\_Outputs}(\texttt{x0},\texttt{y0},\texttt{a},\texttt{c},\texttt{LH},\texttt{LV})$ presented in Appendix B, an input/output behavioral description for Hadamard gate. This completes the formal analysis of the Hadamard gate for inputs $|0,1\rangle$ and $|1,0\rangle$.

4.2 Flip Gate

Contrary to the Hadamard gate design, the flip gate implementation is fully new. Another proposed design of this gate based on coherent state source was proposed in [33]. However, to the best of our knowledge, there is no optical design which is based on the single photon source technique. In this section, we detail a new implementation of the optical flip gate (not gate or X gate) based on single photon technology. The flip gate flips the input state: if the possible input is $|\phi\rangle_{input} = \alpha|0\rangle + \beta|1\rangle$ then the output is $|\phi\rangle_{output} = \alpha|1\rangle + \beta|0\rangle$. The intended implementation of the gate, shown in Figure 3, is composed of two beam splitters and a phase shifter. We formaly define the flip gate structure in HOL as follows:

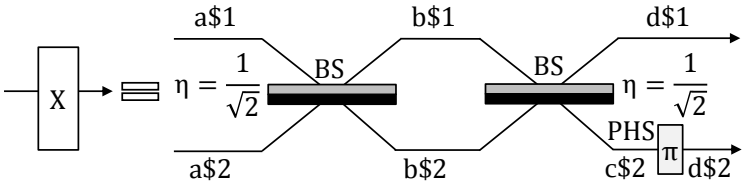

Fig. 3: Schematics of Flip Gate

Definition 4 (Flip Gate).

$\texttt{FLIP\_GATE}\ (\texttt{a},\texttt{d},\texttt{ten},\texttt{LH},\texttt{LV}) \Leftrightarrow (\forall\ \texttt{b c. phase\_shifter}\ (\texttt{ten},\pi,\texttt{c\$2},2,\texttt{d\$2},2)\ \wedge$
$\texttt{is\_beam\_splitter}(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\texttt{ten},\texttt{a\$1},1,\texttt{a\$2},2,\texttt{b\$1},1,\texttt{b\$2},2)\ \wedge$
$\texttt{is\_beam\_splitter}(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\texttt{ten},\texttt{b\$1},1,\texttt{b\$2},2,\texttt{d\$1},1,\texttt{c\$2},2))$

FLIP_GATE takes as parameters all the gate input a and output ports d, the tensor operator. Using this definition, we formally verify the result of applying the flip gate on the two possible inputs $|0,1\rangle_a$ and $|1,0\rangle_a$:

Theorem 3 (Flip Input: $|1\rangle_L \equiv |0,1\rangle_a$).

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{FLIP\_GATE}\ (\texttt{a}, \texttt{d}, \texttt{ten}, \texttt{LH}, \texttt{LV}, \texttt{i})\ \texttt{in}$
$\texttt{let}\ \ |0,1\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ \ |1\rangle_{\texttt{a\$2}}\ \texttt{else}\ \ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$
$\texttt{let}\ \ |1,0\rangle_{\texttt{d}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{d\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{d\$2}})\ \texttt{in}$
$\texttt{constraints} \Rightarrow |0,1\rangle_{\texttt{a}} = |1,0\rangle_{\texttt{d}}$

Theorem 4 (Flip Input: $|0\rangle_L \equiv |1,0\rangle_a$).

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{FLIP\_GATE}(\texttt{a}, \texttt{d}, \texttt{ten}, \texttt{LH}, \texttt{LV}, \texttt{i})\ \texttt{in}$
$\texttt{let}\ \ |1,0\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{a\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$
$\texttt{let}\ \ |0,1\rangle_{\texttt{d}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ \ |1\rangle_{\texttt{d\$2}}\ \texttt{else}\ \ |0\rangle_{\texttt{d\$2}})\ \texttt{in}$
$\texttt{constraints} \Rightarrow |1,0\rangle_{\texttt{a}} = |0,1\rangle_{\texttt{d}}$

Similar to the Hadamard gate, we developed $\texttt{Flip\_In\_Outputs}\ (\texttt{x0}, \texttt{y0}, \texttt{a}, \texttt{d}, \texttt{LH}, \texttt{LV}, \texttt{i})$ which is presented in Appendix C. This is an input/output behavioral description for flip gate.

So far, we have presented two primitive gates without using ancilla resources which are extra qubits that have a secondary role in a computation and are used for detecting the correct output [23]. However, many of the existing optical quantum circuits are using detectors to measure the state of the ancilla after it leaves the circuit. In the approach proposed by [22], the quantum circuit is considered properly implemented only when the detector produces a positive outcome (expected outcome), i.e., the circuit is nondeterministic. In the next section, we will describe a nondeterministic non-linear sign (NS) gate.

4.3 Non-Linear Sign Gate

In [22], the authors formed the universal controlled-phase gate using the nondeterministic non-linear sign (NS) gate (Figure 4), which is a primitive quantum gate composed of three beam splitters. The NS gate operates as follows: When a superposition of the vacuum state $|0\rangle$, the one photon state $|1\rangle$ and the two-photon state $|2\rangle$ is input into the NS gate, the gate flips the sign (or the phase) of the amplitude of the $|2\rangle$ component. Contrary to the Hadamard gate, the NS gate contains two ancillas, one with a single photon and the other in vacuum as shown in Figure 4. 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 $d\$1$.

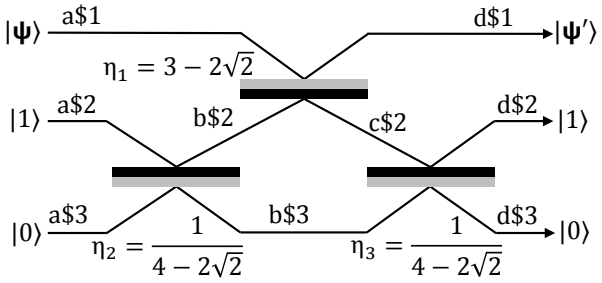

Fig. 4: Schematics of NS Gate

Given this structure, the probability of measuring a single photon at port $d\$2$ and vacuum at port $d\$3$ is then $\frac{1}{4}$. Accordingly, we formally define the NS gate in HOL as follows:

Definition 5 (NS Gate).

$\vdash$ $\texttt{NS\_GATE}(\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{ten}) \Leftrightarrow \texttt{is\_beam\_splitter}(\sqrt{2\sqrt{2}-2},$
$\sqrt{3-2\sqrt{2}},-\sqrt{3-2\sqrt{2}},\sqrt{2\sqrt{2}-2},\texttt{ten},\texttt{b}\$2,2,\texttt{a}\$1,1,\texttt{d}\$1,1,\texttt{c}\$2,2)\wedge$
$\texttt{is\_beam\_splitter}(\frac{1}{\sqrt{4-2\sqrt{2}}},\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}},\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}},-\frac{1}{\sqrt{4-2\sqrt{2}}},\texttt{ten},\texttt{a}\$2,2,\texttt{a}\$3,3,\texttt{b}\$2,2,\texttt{b}\$3,3)\wedge$
$\texttt{is\_beam\_splitter}(\frac{1}{\sqrt{4-2\sqrt{2}}},\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}},\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}},-\frac{1}{\sqrt{4-2\sqrt{2}}},\texttt{ten},\texttt{c}\$2,2,\texttt{b}\$3,3,\texttt{d}\$2,2,\texttt{d}\$3,3)$

where NS_GATE 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 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 inputs $|0,1,0\rangle_a$ and $|1,1,0\rangle_a$.

Theorem 5 (NS Input: $|2\rangle$, Projection: $|2\rangle$).

$\vdash$
$\texttt{let constraint} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge \texttt{NS\_GATE}(\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{ten})\ \texttt{in}$
$\texttt{let}\ |2,1,0\rangle_{\texttt{d}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 1\ \texttt{then}\ |2\rangle_{\texttt{d}\$1}\ \texttt{elseif i} = 2\ \texttt{then}\ |1\rangle_{\texttt{d}\$2}$
$\texttt{else}\ |0\rangle_{\texttt{d}\$3})\ \texttt{in}$
$\texttt{let}\ |2,1,0\rangle_{\texttt{a}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 1\ \texttt{then}\ |2\rangle_{\texttt{a}\$1}\ \texttt{elseif i} = 2\ \texttt{then}\ |1\rangle_{\texttt{a}\$2}$
$\texttt{else}\ |0\rangle_{\texttt{a}\$3})\ \texttt{in}$
$\texttt{constraint} \Rightarrow (\texttt{m\_proj}\ |2,1,0\rangle_{\texttt{d}})\ |2,1,0\rangle_{\texttt{a}} = -\frac{1}{2}\%\,|2,1,0\rangle_{\texttt{d}}$

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 (NS Input: $|2\rangle$, Projection: $|1\rangle$).

$\vdash$
$\texttt{let constraint} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge \texttt{NS\_GATE}(\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{ten})\ \texttt{in}$
$\texttt{let}\ |1,1,0\rangle_{\texttt{d}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 1\ \texttt{then}\ |1\rangle_{\texttt{d}\$1}\ \texttt{elseif i} = 2\ \texttt{then}\ |1\rangle_{\texttt{d}\$2}$
$\texttt{else}\ |0\rangle_{\texttt{d}\$3})\ \texttt{in}$
$\texttt{let}\ |2,1,0\rangle_{\texttt{a}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 1\ \texttt{then}\ |2\rangle_{\texttt{a}\$1}\ \texttt{elseif i} = 2\ \texttt{then}\ |1\rangle_{\texttt{a}\$2}$
$\texttt{else}\ |0\rangle_{\texttt{a}\$3})\ \texttt{in}$
$\texttt{constraint} \Rightarrow (\texttt{m\_proj}\ |1,1,0\rangle_{\texttt{d}})\ |2,1,0\rangle_{\texttt{a}} = 0$

Theorem 7 (NS Input: $|2\rangle$, Projection: $|0\rangle$).

$\vdash$
$\texttt{let constraint} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge \texttt{NS\_GATE}(\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{ten})\ \texttt{in}$
$\texttt{let}\ |0,1,0\rangle_{\texttt{d}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{d}\$2}\ \texttt{else}\ |0\rangle_{\texttt{d}\$3})\ \texttt{in}$
$\texttt{let}\ |2,1,0\rangle_{\texttt{a}} = \texttt{tensor 3}\ (\lambda\texttt{i. if i} = 1\ \texttt{then}\ |2\rangle_{\texttt{a}\$1}\ \texttt{elseif i} = 2\ \texttt{then}\ |1\rangle_{\texttt{a}\$2}$
$\texttt{else}\ |0\rangle_{\texttt{a}\$3})\ \texttt{in}$
$\texttt{constraint} \Rightarrow (\texttt{m\_proj}\ |0,1,0\rangle_{\texttt{d}})\ |2,1,0\rangle_{\texttt{a}} = 0$

where, $\texttt{m\_mode\_pro}\ (|1,1,0\rangle_{\texttt{d}})$ is the projection on the state $|1,1,0\rangle_d$. This completes the formal analysis of the NS gate, where we showed the result of projecting one input state on two different states.

5 Verification of Quantum Circuits

A quantum circuit is a system where certain primitive gates and optical elements are used as elementary building blocks. We have already built a library of widely used quantum primitive gates and with the help of the built-in mathematical properties we will be able to model and verify quantum circuits. In this section, we will demonstrate the idea of formalizing quantum circuits upon primitive gates by formally verifying the controlled-phase (CZ), controlled-not (CNOT), SWAP, Toffoli Sign (TS), Toffoli and Fredkin circuits. This set of quantum gates contains the most used quantum gates and is enough to implement a variety of quantum algorithms.

5.1 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 word: 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 Figure 5. The probability of measuring the ancilla state $|1,0\rangle$ in both NS gates is $\frac{1}{16}$, which is the success probability of 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:

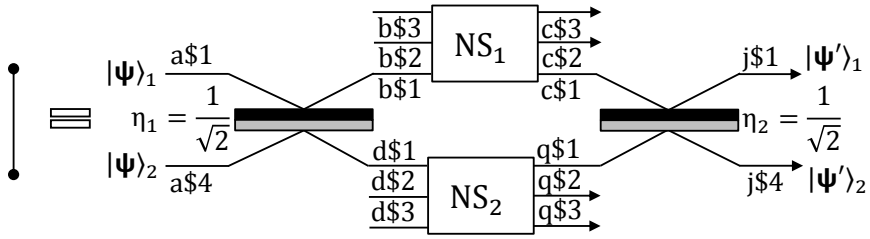

Fig. 5: Schematics of CZ Gate

Definition 6 (CZ Gate).

$\vdash\ \texttt{IS\_CZ\_GATE}\ (\texttt{a},\texttt{b},\texttt{c},\texttt{j},\texttt{ten},\texttt{LH},\texttt{LV},\texttt{m\_proj}) \Leftrightarrow (\forall\ \texttt{d q k l m p}.\texttt{NS\_GATE}(\texttt{d},\texttt{m},\texttt{p},\texttt{q},\texttt{ten})$
$\wedge\ \texttt{b\$4} = \texttt{d\$1} \wedge \texttt{b\$5} = \texttt{d\$2} \wedge \texttt{b\$6} = \texttt{d\$3} \wedge \texttt{NS\_GATE}(\texttt{b},\texttt{l},\texttt{k},\texttt{c},\texttt{ten}) \wedge \texttt{q\$1} = \texttt{c\$4}\ \wedge$
$\texttt{q\$3} = \texttt{c\$6} \wedge \texttt{is\_beam\_splitter}(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},\texttt{ten},\texttt{a\$1},1,\texttt{a\$4},4,\texttt{b\$1},1,\texttt{b\$4},4)\ \wedge$
$\texttt{q\$2} = \texttt{c\$5} \wedge \texttt{is\_beam\_splitter}(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},\texttt{ten},\texttt{c\$1},1,\texttt{c\$4},4,\texttt{j\$1},1,\texttt{j\$4},4)\ \wedge$
$\texttt{is\_sm a\$3} \wedge \texttt{is\_sm a\$2}$

Note that we rename 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, 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, we are providing here two of them as example, and the rest can be found in [1].

Theorem 8 (CZ Input: $|1,1\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$
IS_CZ_GATE (a, b, c, j, ten, LH, LV, m_proj) in
let $|2,1,0,0,1,0,0,0\rangle_{\mathtt{cq}}$ = tensor 8 (λi. if i = 1 then $|2\rangle_{\mathtt{c\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{c\$2}}$ elseif i = 5 then $|1\rangle_{\mathtt{q\$2}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
let $|0,1,0,2,1,0,0,0\rangle_{\mathtt{cq}}$ = tensor 8 (λi. if i = 2 then $|1\rangle_{\mathtt{c\$2}}$ elseif i = 4 then
$|2\rangle_{\mathtt{q\$1}}$ elseif i = 5 then $|1\rangle_{\mathtt{q\$2}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
let $|1,1,0,1,1,0,0,0\rangle_{\mathtt{cq}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{c\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{c\$2}}$ elseif i = 4 then $|1\rangle_{\mathtt{q\$1}}$ elseif i = 5 then $|1\rangle_{\mathtt{q\$2}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
let $|1,1,0,1,1,0,0,0\rangle_{\mathtt{ab}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{a\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{b\$2}}$ elseif i = 4 then $|1\rangle_{\mathtt{a\$4}}$ elseif i = 5 then $|1\rangle_{\mathtt{b\$5}}$ else $|0\rangle_{\mathtt{b\$3}}$) in
let $|1,1,0,1,1,0,0,0\rangle_{\mathtt{cj}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{j\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{c\$2}}$ elseif i = 4 then $|1\rangle_{\mathtt{j\$4}}$ elseif i = 5 then $|1\rangle_{\mathtt{c\$5}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
constraints $\Rightarrow$ (m_proj $|2,1,0,1,0,0,0,0\rangle_{\mathtt{cq}}$ + m_proj $|0,1,0,1,2,0,0,0\rangle_{\mathtt{cq}}$ +
m_proj $|1,1,0,1,1,0,0,0\rangle_{\mathtt{cq}}$) ($|1,1,0,1,1,0,0,0\rangle_{\mathtt{ab}}$) = $-\ \frac{1}{4}\ \%\ |1,1,0,1,1,0,0,0\rangle_{\mathtt{cj}}$

Note 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 port ($|1,1\rangle$) 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 NS gate. For the second input, it is as follows:

Theorem 9 (CZ Input: $|1,0\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$
IS_CZ_GATE (a, b, c, j, ten, LH, LV, m_proj) in
let $|1,1,0,0,1,0,0,0\rangle_{\mathtt{cq}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{c\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{c\$2}}$ elseif i = 5 then $|1\rangle_{\mathtt{q\$2}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
let $|0,1,0,1,1,0,0,0\rangle_{\mathtt{cq}}$ = tensor 8 (λi. if i = 2 then $|1\rangle_{\mathtt{c\$2}}$ elseif i = 4 then
$|1\rangle_{\mathtt{q\$1}}$ elseif i = 5 then $|1\rangle_{\mathtt{q\$2}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
let $|1,1,0,0,1,0,0,0\rangle_{\mathtt{ab}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{a\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{b\$2}}$ elseif i = 5 then $|1\rangle_{\mathtt{b\$5}}$ else $|0\rangle_{\mathtt{b\$3}}$) in
let $|1,1,0,0,1,0,0,0\rangle_{\mathtt{cj}}$ = tensor 8 (λi. if i = 1 then $|1\rangle_{\mathtt{j\$1}}$ elseif i = 2 then
$|1\rangle_{\mathtt{c\$2}}$ elseif i = 5 then $|1\rangle_{\mathtt{c\$5}}$ else $|0\rangle_{\mathtt{c\$3}}$) in
constraints $\Rightarrow$ (m_proj $|1,1,0,0,1,0,0,0\rangle_{\mathtt{cq}}$ + m_proj $|0,1,0,1,1,0,0,0\rangle_{\mathtt{cq}}$)
($|1,1,0,0,1,0,0,0\rangle_{\mathtt{ab}}$) = $\frac{1}{4}\ \%\ |1,1,0,0,1,0,0,0\rangle_{\mathtt{cj}}$

Here the CZ gate has been projected over two different states. This is because of the fact that we have one photon at the input port ($|1,0\rangle$) which results in two possibilities at the input of the two parallel NS gates: 1) one photon goes through the first NS gate; and 2) one photon goes through the second NS gate. The verification of the CZ gate has been done using Theorem 6 in-order to subdivided the tensor product projection to the tensor of two tensor product projections each fed to an NS gate.

As shown in Figure 5, the CZ gate has 8 input modes, however, the CZ is a 2-qubits gate, where each logical qubit is represented by two optical modes and the rest of the 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), details are presented in Appendix D. This completes the formal analysis of the CZ gate for the inputs "11" and "10". The analysis for the inputs "01", and "00" follows the similar pattern.

5.2 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 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 Figure 6, however, it can also be implemented using five beam splitters [42].

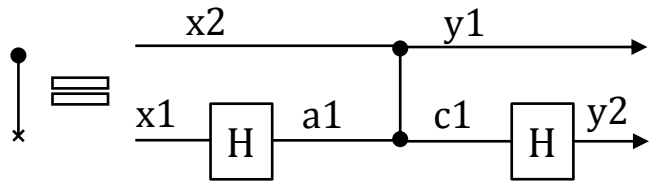

Fig. 6: Schematics of CNOT Gate

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 CNOT, where the target is the first qubit for the first version and it is the second qubit of the second version. We provide here the HOL definition of the first type structure and the second one is given in Appendix E.

Definition 7 (CNOT Gate).

$\vdash$ CNOT1_GATE(x1, x2, y1, y2, ten, LH, LV, m_proj) $\Leftrightarrow$
($\forall$a1 c1. HADAMARD_GATE(x1, a1, ten, LH, LV) $\wedge$ HADAMARD_GATE(c1, y2, ten, LH, LV)
$\wedge$ CZ_GATE(a1, x2, c1, y1, ten, LH, LV, m_proj))

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

Theorem 10 (CNOT Gate Input: $|0, 1\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$ 8 $\leq$ dimindex (:N) $\wedge$
CNOT1_GATE (a1, a2, j1, j2, ten, LH, LV, m_proj) in
let $|0, 1\rangle_{\mathtt{a}}$ = tensor 2 (λi. if i = 1 thenLH a1 else LV a2) in
let $|1, 1\rangle_{\mathtt{j}}$ = tensor 2 (λi. if i = 1 then LV j1 else LV j2) in
constraints $\Rightarrow$ $|0, 1\rangle_{\mathtt{a}} = \frac{1}{4}$ % $|1, 1\rangle_{\mathtt{j}}$

Furthermore, by employing the linearity of tensor product we formally proved the general case for an input in the form $|x, y\rangle = x1y1|11\rangle + x1y2|10\rangle + x2y1|01\rangle + x2y2|00\rangle$ as follows:

Theorem 11 (CNOT Gate Input: $|x, y\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$ $8 \leq$ dimindex $(:\mathtt{N})$ $\wedge$ CNOT1_GATE (a1, a2, j1, j2, ten, LH, LV, m_proj) in
let $\mathtt{x1y1}\,|11\rangle_{\mathtt{a}} + \mathtt{x1y2}\,|10\rangle_{\mathtt{a}} + \mathtt{x2y1}\,|01\rangle_{\mathtt{a}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{a}} =$ tensor 2 (λi. if i = 1 then (x1%LV a1 + x2%LH a1) else (y1%LV a2 + y2%LH a2)) in
let $\mathtt{x1y1}\,|01\rangle_{\mathtt{j}} + \mathtt{x2y1}\,|11\rangle_{\mathtt{j}} =$ tensor 2 (λi. if i = 1 then (x1%LH j1+ x2%LV j1) else y1%LV j2) in
let $\mathtt{x1y2}\,|10\rangle_{\mathtt{j}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{j}} =$ tensor 2 (λi. if i = 1 then (x1%LV j1+ x2%LH j1) else y2%LH j2) in
constraints $\Rightarrow$ $\mathtt{x1y1}\,|11\rangle_{\mathtt{a}} + \mathtt{x1y2}\,|10\rangle_{\mathtt{a}} + \mathtt{x2y1}\,|01\rangle_{\mathtt{a}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{a}} =$
$\frac{1}{4}$ % $(\mathtt{x1y1}\,|01\rangle_{\mathtt{j}} + \mathtt{x2y1}\,|11\rangle_{\mathtt{j}} + \mathtt{x1y2}\,|10\rangle_{\mathtt{j}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{j}})$

5.3 SWAP Gate

The SWAP gate is a two qubits gate which swaps the states of the two input qubits. It has a crucial role in the design of quantum circuits where the SWAP gate is used to swap the qubits between each other in order to fulfil the requirement that computations should only be performed between adjacent qubits [37]. Also in [24], 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 [4], it was shown that the SWAP gate plays an important role in the implementation of Shor's algorithm [3] based on linear nearest neighbour architecture, where the SWAP gate rearranges the qubits. The physical implementation of the SWAP gate requires three CNOT gates, as shown in Figure 7.

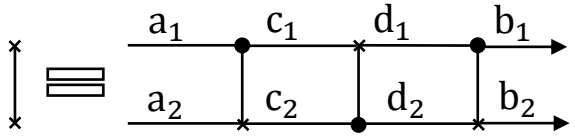

Fig. 7: Schematics of SWAP Gate

In the structure of SWAP gate, we can note the usage of the two versions of CNOT gate in the implementation. We formally define the structure of the SWAP gate in HOL as follows:

Definition 8 (SWAP Gate Structure).

$\vdash$ SWAP_GATE(x1, x2, y1, y2, ten, LH, LV, m_proj) $\Leftrightarrow$
($\forall$ c1 c2 d1 d2. CNOT2_GATE(d1, d2, y1, y2, ten, LH, LV, m_proj) $\wedge$
CNOT2_GATE(x1, x2, c1, c2, ten, LH, LV, m_proj) $\wedge$ CNOT1_GATE(c1, c2, d1, d2, ten, LH, LV, m_proj))

Accordingly, we employed this definition to prove the general case for an input to the SWAP gate in the form $|x, y\rangle = |x1\%1 + x2\%0, y1\%1 + y2\%0\rangle$ in HOL as follows:

Theorem 12 (SWAP Gate Input: $|x, y\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$ $8 \leq$ dimindex (:N) $\wedge$
SWAP_GATE(a1, a2, j1, j2, ten, LH, LV, m_proj) in
let $|$x1%1 + x2%0, y1%1 + y2%0$\rangle_{\text{a}}$ = tensor 2 (λi. if i = 1 then (x1%LV a1+
x2%LH a1) else (y1%LV a2 + y2%LH a2)) in
let $|$y1%1 + y2%0, x1%1 + x2%0$\rangle_{\text{j}}$ = tensor 2 (λi. if i = 1 then (y1%LH j1+
y2%LV j1) else (x1%LH j2 + x2%LV j2)) in
constraints $\Rightarrow$ $|$x1%1 + x2%0, y1%1 + y2%0$\rangle_{\text{a}} = \frac{1}{64}$ % $|$y1%1 + y2%0, x1%1 + x2%0$\rangle_{\text{j}}$

5.4 Toffoli Sign Gate

The Toffoli Sign (TS) gate is a three-qubit gate that applies a sign shift on one of the state components, and the identity to other inputs. The main benefit of TS gate is to construct the Toffoli gate. In the optical implementation of TS, vac a1 refers to the vacuum state, i.e., where both polarization modes are unoccupied, in the single mode a1. Thus, we will introduce a third level of representation of a *qutrit* (i.e., a superposition of three orthogonal quantum states). The realization of TS is based on using two qubits a2, a3 (i.e., $0_L = |LH\rangle$ and $1_L = |LV\rangle$) and a *qutrit* t (i.e., $1_L = |vac, LV\rangle_{a1a4}$ and $0_L = |LH, vac\rangle_{a1a4}$). The gate is composed of two CNOT gates (first and last two-qubit gates) on a CZ gate (middle two-qubit gate). The two CNOT gates operate as normal at the qubit levels and implement the identity if the target is at the *qutrit* level ($|vac, LV\rangle$). The TS gate structure is shown in Figure 8.

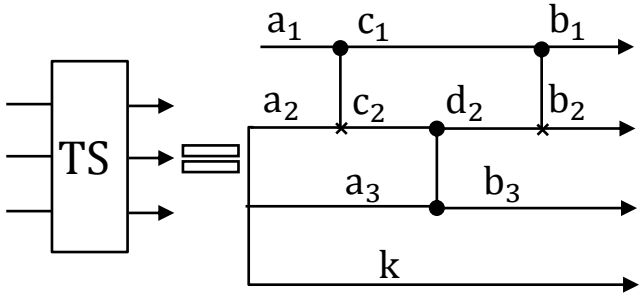

Fig. 8: Schematics of TS Gate

We formally define the structure of TS gate in HOL as follows:

Definition 9 (TS Gate Structure).

$\vdash$ TS_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) $\Leftrightarrow$
($\forall$ k c1 c2 d2. CNOT2_GATE(c1, d2, b1, b2, ten, LH, LV, m_proj) $\wedge$
CNOT2_GATE(a1, a2, c1, c2, ten, LH, LV, m_proj) $\wedge$ CZ_GATE(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 $|x, y, z\rangle$, x is the qutrit. We formally verify the result of applying TS transformation on two inputs form $|101\rangle$ and $|111\rangle$:*

Theorem 13 (TS Input: $|101\rangle$).
$\vdash$ let constraints $= 8 \leq$ dimindex $(:\mathtt{N}) \wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
TS_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) in
let $|0,1,1\rangle =$ tensor 3 (λi. if i $= 1$ then LH a1 elseif i $= 2$ then LV a2
else LV a3) in
let $|0,1,1\rangle =$ tensor 3 (λi. if i $= 1$ then LH a1 elseif i $= 2$ then LV a2
elseif LV a3) in
constraints $\Rightarrow |0,1,1\rangle_{\mathtt{a}} = -\frac{1}{64}$ % $|0,1,1\rangle_{\mathtt{b}}$

Theorem 14 (TS Input: $|111\rangle$).
$\vdash$ let constraints $= 8 \leq$ dimindex $(:\mathtt{N}) \wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
TS_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) in
let $|1,1,1\rangle =$ tensor 3 (λi. if i $= 1$ then LV a1 elseif i $= 2$ then LV a2
elseLV a3) in
let $|1,1,1,\mathtt{vac}\rangle =$ tensor 3 (λi. if i $= 1$ then LV b1 elseif i $= 2$ then LV b2
else LV b3) in
constraints $\Rightarrow |1,1,1\rangle_{\mathtt{a}} = \frac{1}{64}$ % $|1,1,1\rangle_{\mathtt{b}}$

From the Figure 8, we can note that TS gate has 4 input modes, however, TS is a 3-qubits gate, where the first logical qubit is represented by three optical modes. In order to facilitate the use of this gate in quantum circuits where the computation is at the qubit level and without getting into the detail of the qubit representation, we developed, TS_outputs (t, c4, t2, c3, LH, LV) and TS_inputs (t, t1, LH, LV)) presented in Appendix E, an input/output behavioral description for TS gate.

5.5 Toffoli Gate

The Toffoli gate is a three-qubit gate that flips the logical state of the target qubit conditional on the logical state of the two control qubits. The Toffoli gate is one of the most important quantum gates and has many quantum applications including; universal reversible classical computation, quantum error correction and fault tolerance. Furthermore, the combination of the Toffoli and Hadamard gates offers a simple universal quantum gate set [6]. The simplest known design of the Toffoli gate when restricted to operating on qubits at the behavioral level is a circuit that requires five two-qubit gates, the gate is shown in Figure 9. However, it was shown that it is possible to construct a Toffoli gate using the Toffoli sign gate which is composed of only three two-qubit gates, flip, and Hadamard gates [34].

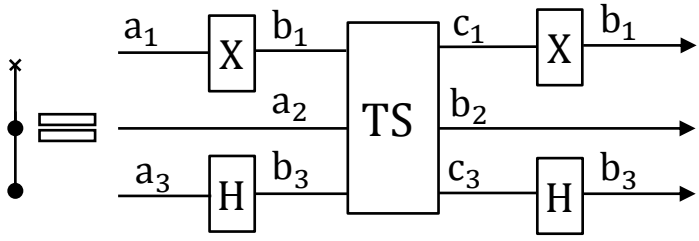

Fig. 9: Schematics of Toffoli Gate

Similar to CNOT, the Toffoli gate can be used in two forms; 1) the first qubit is the target; and 2) the third qubit is the target. Therefore, we have formally defined these two kinds of

Toffoli gate in HOL. We provide here the formal definition of the second type structure of Toffoli gate as follows:

Definition 10 (Toffoli Gate Structure).
$\vdash$ TOFFOLI3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) $\Leftrightarrow$ ($\forall$ c3 d3 c1 d1.
FLIP_GATE(a1, c1, LH, LV, ten) $\wedge$ TS_Gate(c1, a2, c3, d1, b2, d3, ten, LH, LV, m_proj) $\wedge$
HADAMARD_GATE(a3, c3, ten, LH, LV) $\wedge$ FLIP_GATE(d1, b1, LH, LV, ten) $\wedge$
HADAMARD_GATE(d3, b3, ten, LH, LV))

From this definition, we verify the result of applying Toffoli on the input $|111\rangle$, where the two control qubits are $|1\rangle_L$:

Theorem 15 (Toffoli Input: $|111\rangle$).
$\vdash$ let constraints = 8 $\leq$ dimindex(: N) $\wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
TOFFOLI3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) in
let $|111\rangle_{\mathtt{a}}$ = tensor 3 (λi. if i = 1 then LV a1 elif i = 2 then LV a2 else
LV a3) in
let $|011\rangle_{\mathtt{b}}$ = tensor 3 (λi. if i = 1 then LH b1 elif i = 2 then LV b2 else
LV b3) in
constraints $\Rightarrow$ $|111\rangle_{\mathtt{a}} = \frac{1}{64}\% |011\rangle_{\mathtt{b}}$

We provide also the result of applying Toffoli on the input $|011\rangle$, where the target is $|0\rangle_L$:

Theorem 16 (Toffoli Input: $|011\rangle$).
$\vdash$ let constraints = 8 $\leq$ dimindex(: N) $\wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
TOFFOLI3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) in
let $|011\rangle_{\mathtt{a}}$ = tensor 3 (λi. if i = 1 then LH a1 elif i = 2 then LV a2 else
LV a3) in
let $|111\rangle_{\mathtt{b}}$ = tensor 3 (λi. if i = 1 then LV b1 elif i = 2 then LV b2 else
LV b3) in
constraints $\Rightarrow$ $|011\rangle_{\mathtt{a}} = \frac{1}{64}\% |111\rangle_{\mathtt{b}}$

5.6 Fredkin Gate

The Fredkin gate or the controlled-2x2 quantum switch gate (or controlled SWAP gate) is a 3-qubits gate [29]. One of the qubits is designated as the control qubit and is left unchanged by the gate, and the rest, two qubits are the target qubits. If the control qubit is zero, the two target qubits remain unchanged. If the control qubit is one, the two target qubits are inter-changed. The Fredkin gate has an important role in quantum computing, error-correcting quantum computations, and information processing [25]. Moreover, the controlled SWAP gate is a universal gate for quantum (reversible) computing such that any logical or arithmetic operation can be constructed entirely of Fredkin gates [29]. The gate circuit, shown in Figure 10, is composed of two CNOT gates and one Toffoli. AND, OR, and XOR gates, flip-flops, etc. can all be constructed using the Fredkin gate.

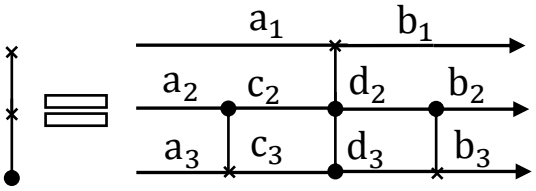

Fig. 10: Schematics of Fredkin Gate

We formally model the structure of the Fredkin gate in HOL as follows:

Definition 11 (Fredkin Gate Structure).
$\vdash$ FREDKIN3_GATE (a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) $\Leftrightarrow$ ($\forall$ c2 c3 d2 d3.
CNOT1_GATE(a2, a3, c2, c3, ten, LH, LV, m_proj) $\wedge$
TOFFOLI3_GATE(a1, c2, c3, b1, d2, d3, ten, LH, LV, m_proj) $\wedge$
CNOT1_GATE(d2, d3, b2, b3, ten, LH, LV, m_proj))

From this definition, we verify the result of applying the Fredkin gate on the input $|011\rangle$, where the two control qubits are $|1\rangle_L$, which means that there will be an exchange between the two target qubits.

Theorem 17 (Fredkin: Input: $|011\rangle$).
$\vdash$ let constraints = 8 $\leq$ dimindex(: N) $\wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
FREDKIN3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) in
let $|011\rangle_{\mathtt{a}}$ = tensor 3 (λi. if i = 1 then LH a1 elif i = 2 then LV a2 else
LV a3) in
let $|101\rangle_{\mathtt{b}}$ = tensor 3 (λi. if i = 1 then LV b1 elif i = 2 then LH b2 else
LV b3) in
constraints $\Rightarrow$ $|011\rangle_{\mathtt{a}} = \frac{1}{1024}\% |101\rangle_{\mathtt{b}}$

We also verify the result of applying the Fredkin gate on the input general form $|zxy\rangle$:

Theorem 18 (Fredkin: Input: $|zxy\rangle$).
$\vdash$ let constraints = 8 $\leq$ dimindex(: N) $\wedge$ is_tensor_pro m_proj $\wedge$ is_tensor ten $\wedge$
FREDKIN3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m$_{\mathtt{m}}$odes$_{\mathtt{p}}$ro) in
let $|zxy\rangle_{\mathtt{a}}$ = tensor3(λi. if i = 1 then (z1%LH a1 + z2%LV a1) elseif i = 2 then
(x1%LH a2 + x2%LV a2) else (y1%LH a3 + y2%LV a3)) in
let $|0xy\rangle_{\mathtt{b}}$ = tensor 3 (λi. if i = 1 then LH b1 elseif i = 2 then (x1%LH b2+
x2%LV b2) else (y1%LH b3 + y2%LV b3)) in
let $|1xy\rangle_{\mathtt{b}}$ = tensor 3 (λi. if i = 1 then LV b1 elseif i = 2 then (y1%LH b2+
y2%LV b2) else (x1%LH b3 + x2%LV b3)) in
constraints $\Rightarrow$ $|zxy\rangle_{\mathtt{a}} = \frac{1}{1024}\%(\mathtt{z1}\% |0xy\rangle_{\mathtt{b}} + \mathtt{z2}\% |1xy\rangle_{\mathtt{b}}$

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

6 Verification of Quantum Algorithms

In this section, we demonstrate the usefulness of the formalized library in implementing the only known quantum algorithm that is exponentially faster than its classical counterpart, specifically, we model and verify a compiled version of Shor's integer factorization of number 15 [2]. Moreover, we apply our library of reversible gates to design, model, and verify a quantum full adder circuit, our formalization is based on the design proposed in [8].

6.1 Verification of Shor's Factorization Algorithm

Shor's integer factorization [3] is a quantum algorithm which can break cryptographic codes that are widely employed in monetary transactions on the Internet [7]. The algorithm trick is that it can compute the two primes factor of a given integer number much faster than classical algorithms can do. Our objective here is to show the formal modelling and verification of a compiled version (i.e., a designed version to find the prime factors of a specific input) of Shor's factoring of number 15 [2] 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 Figure 11, and it has four inputs/outputs. 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:

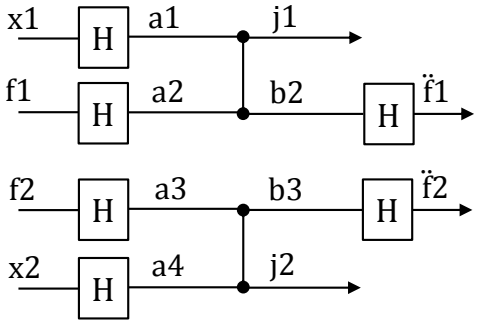

Fig. 11: Schematics of Shor's factoring of 15

Definition 12 (Shor Circuit).

$\vdash$ shor(x1, x2, f1, f2, f̈1, f̈2, j1, j2, ten, LH, LV, m_proj) $\Leftrightarrow$ ($\forall$a2 b2 a1 a3 a4 b3.
CZ_GATE(a1, a2, j1, b2, ten, LH, LV, m_proj) $\wedge$ CZ_GATE(a3, a4, b3, j2, ten, LH, LV, m_proj) $\wedge$
HADAMARD_GATE(x1, a1, ten, LH, LV) $\wedge$ 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(b3, f̈2, ten, LH, LV))

where the two modes j1$2 and j2$2 stand for ẍ1 and ẍ2, respectively. From this definition, we formally verify the operation of the circuit as follows:

Theorem 19 (Shor's Factoring of 15).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$
is_tensor ten $\wedge$ shor (x1, x2, f1, f2, f̈1, f̈2, j1, j2, ten) in
let $|0,0,1,0\rangle_{\mathtt{f1x1f2x2}}$ = tensor 4 (λi. if i = 1 then LH f1 elseif i = 2 then LH x1
elseif i = 3 then LV f2 else LH x2) in
let $|0,0,0,1\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$ = tensor 4 (λi. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LH f̈2 else LV ẍ2) in
let $|0,0,1,0\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$ = tensor 4 (λi. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LV f̈2 else LH ẍ2) in
let $|1,1,0,1\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$ = tensor 4 (λi. if i = 1 then LV f̈1 elseif i = 2 then LV ẍ1
elseif i = 3 then LH f̈2 else LV ẍ2) in
let $|1,1,1,0\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$ = tensor 4 (λi. if i = 1 then LH f̈1 elseif i = 2 then LH ẍ1
elseif i = 3 then LV f̈2 else LH ẍ2) in
constraints $\Rightarrow |0,0,1,0\rangle_{\mathtt{f1x1f2x2}} = \frac{1}{32}$ % ($|1,1,1,0\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2} + |1,1,0,1\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$
$+ |0,0,1,0\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2} + |0,0,0,1\rangle_{\ddot{\mathtt{f}}1\ddot{\mathtt{x}}1\ddot{\mathtt{f}}2\ddot{\mathtt{x}}2}$)

Here the circuit outputs two categories of solutions; 1) $|000\rangle$ or $|100\rangle$ an expected failure of the algorithm; 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. The verification of the compiled Shor's circuit has been done using Theorem 3 to subdivide the tensor to 4 tensors and apply Hadamard transformation on each elementary tensor, then recombine the elementary tensors to 2 tensor and apply CZs, finally subdivide to 4 tensors and apply Hadamards. Applying our formalized quantum gates library to perform the formal verification of Shor's factorization shows the richness of this library. Another added value of the modeling and verification of Shor's factorization of number 15 is the extraction of success probability of the quantum circuit.

6.2 Verification and Modeling of Full Adder

Reversible gates are circuits that have the same number of inputs and outputs where the input states can be always reconstructed from the output states. In this section, we demonstrate the basic building of any classical algorithm[1], which is the full adder, using reversible gates. In particular, our design is mainly inspired from the design presented in [8]. Since it has been physically proven that the qubits need to be adjacent in order to be applied to the same gate, we added two SWAP gates to exchange the qubits before feeding them to the Fredkin gate. This was not addressed in [8] and shows the practicality of our approach in designing quantum circuits that are physically feasible to build. By identifying the limitation of the proposed design when it is modeled using tensor product. For example, we cannot feed two components of a tensor product that are not adjacent to the same gate.

The quantum circuit of the quantum full adder is composed of three CNOT, two SWAP, and a Fredkin gates, as shown in Figure 12. The circuit has four inputs; the two operands, the carry, and an extra input which is initialized to $|0\rangle$.

[1]Traditionally, any classical Boolean function can be implemented using only NAND-gates that is logically irreversible (i.e., it has two inputs and one output) which leads to energy dissipation regardless of the realization technology.

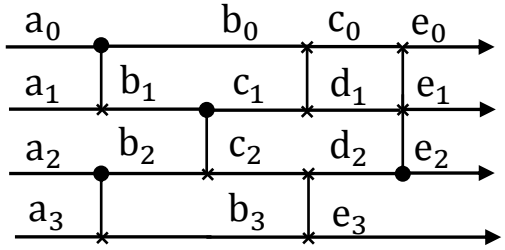

Fig. 12: Schematics of Full Adder Gate

We formally define the structure of the quantum full adder in HOL as follows:

Definition 13 (Full Adder Circuit).

$\vdash$ FULL_ADDER(a0, a1, a2, a3, e0, e1, e2, e3, ten, LH, LV, m_proj) $\Leftrightarrow$
($\forall$ b0 b1 b2 b3 c0 c1 c2 d1 d2. CNOT2_GATE(a0, a1, b0, b1, ten, LH, LV, m_proj) $\wedge$
CNOT2_GATE(a2, a3, b2, b3, ten, LH, LV, m_proj) $\wedge$ CNOT2_GATE(b1, b2, c1, c2, ten, LH, LV, m_proj) $\wedge$
SWAP_GATE(b0, c1, c0, d1, ten, LH, LV, m_proj) $\wedge$ SWAP_GATE(c2, b3, d2, e3, ten, LH, LV, m_proj)
$\wedge$ FREDKIN3_GATE(c0, d1, d2, e0, e1, e2, ten, LH, LV, m_proj))

Based on this definition, we formally verify the functionality of quantum full adder in the general case where have two variables: $|x\rangle = x1\%LH\ a1 + x2\%LV\ a1\ (x = x1\,|0\rangle_{a1} + x2\,|1\rangle_{a1})$ and $|y\rangle = y1\%LH\ a2 + y2\%LV\ a2\ (y = y1\,|0\rangle_{a2} + y2\,|1\rangle_{a2})$, and a carry: $|z\rangle = z1\%LH\ a3 + z2\%LV\ a3\ (z = z1\,|0\rangle_{a3} + z2\,|1\rangle_{a3})$. The HOL Light formalization is as follows:

Theorem 20 (Full Adder).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$
FULL_ADDER(a1, a2, a3, a4, b1, b2, b3, b4, ten, LH, LV, m_proj) in
let input = tensor 4 (λi.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 (λi. 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 (λi. 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 (λi. 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 (λi. 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 (λi. 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 (λi. 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 (λi. 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 (λi. if i = 1 then LH b1 elseif i = 2 then LV b2
elseif i = 3 then LV b3 else LV b4) in
constraints $\Rightarrow$ input = Cx(($\frac{\&1}{\&16}$) pow 7) * ((z1 * x1 * y1)%output1+

$(\texttt{z1} * \texttt{x1} * \texttt{y2})\%\texttt{output2} + (\texttt{z1} * \texttt{x2} * \texttt{y1})\%\texttt{output3} + (\texttt{z1} * \texttt{x2} * \texttt{y2})\%\texttt{output4} +$
$(\texttt{z2} * \texttt{x1} * \texttt{y1})\%\texttt{output5} + (\texttt{z2} * \texttt{x1} * \texttt{y2})\%\texttt{output6} + (\texttt{z2} * \texttt{x2} * \texttt{y1})\%\texttt{output7} +$
$(\texttt{z2} * \texttt{x2} * \texttt{y2})\%\texttt{output8})$

7 Conclusions

In this report, we presented an approach for the formal modeling, design and verification of quantum circuits, by formalizing a library of the most common employed quantum gates in HOL Light theorem prover. In particular, we described the formalization of Flip, Hadamard, NS, CNOT, CZ, SWAP, TS, Toffoli, and Fredkin gates. In order to demonstrate the usability of the presented library, we modeled and verified a compiled version of Shor's factorization of number 15 and a quantum full adder. This report encompasses several contributions namely: 1) The analysis was conducted using the very trusted, expressive, and sound high-order logic theorem proving and several fundamental mathematical theorems related to tensor product and linear projection were proved; 2) Tensor product projection was used to mathematically model the physical operation of measuring the output of quantum circuits; 3) A rich set of universal quantum gates which can implement a variety of quantum algorithms was formalized; 4) With the help of our library, we have developed a novel design of the Flip gate and formally verified it; 5) The designs of Toffoli gate and full adder were enhanced from the existing models in the literature for the sake of physically feasible implementation; 6) For all the formalized circuits the success probability was extracted, which gives an insight about the effectiveness of the proposed implementation. For example, we can note how little is the success probability of the quantum full adder, however, to our best knowledge it is the optimum design in terms of number of quantum gates. This result leads to considering alternative methods for implementing this circuit such as quantum teleportation. Note that such result has never been reported in the literature. Compared to existing related work, the presented approach is more complete (i.e., covers more quantum gates and circuits), and generic (i.e., the analysis is done at the quantum physics level).

The formalization described in this report can be used to examine more quantum algorithms e.g., Grover search [14] and quantum switching networks [9] algorithms. Also in the future, we plan to investigate the usage of quantum teleporation in quantum circuits to improve the success probability of these circuits. Another future plan is to develop techniques that could automate the analysis procedures and to build a tool with a graphical interface where the analysis of quantum circuits will be carried out.

Appendices

A HOL Light Symbols and Functions

HOL Symbol	Standard Symbol	Meaning
/\	and	Logical *and*
\/	or	Logical *or*
~	not	Logical *negation*
T	true	Logical true value
F	false	Logical false value
==>	$\longrightarrow$	Implication
<=>	$=$	Equality
!x.t	$\forall x.t$	for all $x : t$
?x.t	$\exists x.t$	for some $x : t$
@x.t	$\epsilon x.t$	an x such that : t
λx.t	$\lambda x.t$	Function that maps x to $t(x)$
num	$\{0, 1, 2, \ldots\}$	Positive Integers data type
real	All Real numbers	Real data type
complex	All complex numbers	Complex data type
suc n	$(n + 1)$	Successor of natural number
abs x	$\|x\|$	Absolute function
&a	$\mathbb{N} \to \mathbb{R}$	Typecasting from Integers to Reals
EL k L	$L[k]$	The k^{th} element of list L
CONS e L	*cons*	Add the element e to the list L
APPEND l1 l2	*append*	Joins two lists together
HD L	*head*	Return the head of the list L
TL L	*tail*	tail of list L
LENGTH L	*length*	Length of list L
(a, b)	$a \times b$	A pair of two elements
FST	$fst(a, b) = a$	First component of a pair
SND	$snd(a, b) = b$	Second component of a pair
{x|P(x)}	$\{x \mid P(x)\}$	Set of all x such that $P(x)$
Cx a	$\mathbb{R} \to \mathbb{C}$	Typecasting from Reals to Complexes
lambda x. v x	$\lambda\ x.\ v\ x$	Vectors lambda
x$i	x(i)	Vector indexing operator
x pow n	x^n	Real and complex power
a % V	a. V	Scalar multiplication
--	-	Arithmetic negation
A → B	$A \to B$	Domain to Codomain
λ	λ	Lambda abstraction (required for functions definition)
A:real	$A : \mathbb{R}$	Specify type operator
f o g	f o g	Function composition
**	*	Operator multiplication
**	*	Operator multiplication
[a; b; ..]	[a; b; ..]	Lists

B Hadamard Gate Behavioral Description

In this appendix, we define the behavioral description of the Hadamard gate, where instead of two inputs and two outputs we have one single input and single output.

Hadamard_In_Outputs (x0,y0,a, c, LH, LV)	
tensor 1 (λi. LH x0)	tensor 2 (λi. if i = 1 then fock a$1 1 else vac a$1)
tensor 1 (λi. LV x0)	tensor 2 (λi. if i = 2 then fock a$2 1 else vac a$1)
tensor 1 (λi. vac x0)	tensor 2 (λi. vac a$1)
tensor 2 (λi. if i = 1 then fock c$1 1 else vac c$2)	tensor 1 (λi. LH y0)
tensor 2 (λi. if i = 2 then fock c$2 1 else vac c$2)	tensor 1 (λi. LV y0)
tensor 2 (λi. vac c$2)	tensor 1 (λi. vac y0)

C Flip Gate Behavioral Description

In this appendix, we define the behavioral description of the Flip gate, where instead of two inputs and two outputs we have one single input and single output.

Flip_In_Outputs (x0,y0,a, c1, LH, LV)	
tensor 1 (λi. LH x0)	tensor 2 (λi. if i = 1 then fock a$1 1 else vac a$1)
tensor 1 (λi. LV x0)	tensor 2 (λi. if i = 2 then fock a$2 1 else vac a$1)
tensor 2 (λi. if i = 1 then fock c1$1 1 else vac c1$2)	tensor 1 (λi. LH y0)
tensor 2 (λi. if i = 2 then fock c1$2 1 eelse vac c1$2)	tensor 1 (λi. LV y0)

D CZ Gate Behavioral Description

In this appendix, we define the behavioral description for CZ gate, where instead of eight inputs and eight outputs, we have two inputs and two outputs.

CZ_INPUTS (x1,x2,a, b, c,LH, LV,m_proj)	
tensor 2 (λi. if i = 1 then LH x1 else LH x2)	m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$51 elif i = 7 then fock a$2 1 elif i = 8 then fock a$3 1 else vac c$3)) (tensor 8 (λi. if i = 2 then fock b$2 1 elif i = 5 then fock b$5 1 elif i = 7 then fock a$2 1 elif i = 8 then fock a$3 1 else vac b$3))
tensor 2 (λi. if i = 1 then LV x1 else LH x2)	(m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 1 then fock c$1 2 else vac c$3)) + m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 4 then fock c$4 2 else vac c$3))+ m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 4 then fock c$4 1 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 4 then fock c$4 1 else vac c$3))) (tensor 8 (λi. if i = 1 then fock a$1 1 elif i = 4 then fock a$41 elif i = 5 then fock b$21 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock b$51 else vac b$3))
tensor 2 (λi. if i = 1 then LV x1 else LH x2)	(m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 1 then fock c$11 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 elif i = 5 then fock c$5 1 else vac c$3)) + m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 4 then fock c$4 1 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 elif i = 5 then fock c$5 1 else vac c$3))) (tensor 8 (λi. if i = 1 then fock a$1 1 elif i = 2 then fock b$21 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 elif i = 5 then fock b$51 else vac b$3))
tensor 2 (λi. if i = 1 then LH x1 else LV x2)	(m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 1 then fock c$11 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3)) + m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 4 then fock c$41 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3))) (tensor 8 ($\lambda$i. if i = 4 then fock a$4 1 elif i = 2 then fock b$21 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 elif i = 5 then fock b$51 else vac b$3))

tensor 2 (λi. if i = 1 then vac x1 else LH x2)	m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$51 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 else vac c$3)) (tensor 8 (λi. if i = 2 then fock b$2 1 elif i = 5 then fock b$5 1 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 else vac b$3))
tensor 2 (λi. if i = 1 then vac x1 else LV x2)	(m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 1 then fock c$11 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3)) + m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 4 then fock c$41 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3))) (tensor 8 ($\lambda$i. if i = 4 then fock a$4 1 elif i = 2 then fock b$21 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock b$51 else vac b$3))
tensor 2 (λi. if i = 1 then LH x1 else vac x2)	m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$51 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 else vac c$3)) (tensor 8 (λi. if i = 2 then fock b$2 1 elif i = 5 then fock b$5 1 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 else vac b$3))
tensor 2 (λi. if i = 1 then LV x1 else vac x2)	(m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 1 then fock c$11 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3)) + m_modes_pro (tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 4 then fock c$41 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock c$5 1 else vac c$3))) (tensor 8 ($\lambda$i. if i = 1 then fock a$1 1 elif i = 2 then fock b$21 elif i = 7 then vac a$2 elif i = 8 then vac a$3 elif i = 5 then fock b$51 else vac b$3))

CZ_OUTPUTS (y1,y2,a, c, j,LH, LV)	
tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then fock a$2 1 elif i = 8 then fock a$3 1 else vac c$3)	tensor 2 (λi. if i = 1 then LH y1 else LH y2)
tensor 8 (λi. if i = 1 then fock j$1 1 elif i = 4 then fock j$4 1 elif i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then vac a$3 elif i = 7 then vac a$2 else vac c$3)	tensor 2 (λi. if i = 1 then LV y1 else LV y2)
tensor 8 (λi. if i = 1 then fock j$1 1 elif i = 2 then fock c$21 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 else vac c$3)	tensor 2 (λi. if i = 1 then LV y1 else LH y2)
tensor 8 (λi. if i = 4 then fock j$4 1 elif i = 2 then fock c$21 elif i = 5 then fock c$5 1 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 else vac c$3)	tensor 2 (λi. if i = 1 then LH y1 else LV y2)
tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then fock a$3 1 else vac c$3)	tensor 2 (λi. if i = 1 then vac y1 else LH y2)
tensor 8 (λi. if i = 4 then fock j$4 1 elif i = 2 then fock c$21 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then vac a$3 else vac c$3)	tensor 2 (λi. if i = 1 then vac y1 else LV y2)
tensor 8 (λi. if i = 2 then fock c$2 1 elif i = 5 then fock c$5 1 elif i = 7 then fock a$2 1 elif i = 8 then vac a$3 else vac c$3)	tensor 2 (λi. if i = 1 then LH y1 else vac y2)
tensor 8 (λi. if i = 1 then fock j$1 1 elif i = 2 then fock c$21 elif i = 5 then fock c$5 1 elif i = 7 then vac a$2 elif i = 8 then vac a$3 else vac c$3)	tensor 2 (λi. if i = 1 then LV y1 else vac y2)

E Second Type of CNOT Gate

In this appendix, we provide the HOL formalization of the second type of CNOT gate.

Definition 14 (CNOT Gate).

$\vdash$ CNOT2_GATE(x2, x1, y1, y2, ten, LH, LV, m_proj) $\Leftrightarrow$ ($\forall$ a1 c1.
HADAMARD_GATE(x1, a1, ten, LH, LV) $\wedge$
HADAMARD_GATE(c1, y2, ten, LH, LV) $\wedge$ CZ_GATE(x2, a1, y1, c1, ten, LH, LV, m_proj))

Here the Hadamard gate is applied on the second input which is the target qubit. Using this definition we formally verified that it maintains the truth table of the CNOT gate. The general case for an input in the form $|x, y\rangle = x1y1\,|11\rangle + x1y2\,|10\rangle + x2y1\,|01\rangle + x2y2\,|00\rangle$ is as follows:

Theorem 21 (CNOT Gate Input: $|x, y\rangle$).

$\vdash$ let constraints = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$ 8 $\leq$ dimindex (:N) $\wedge$
CNOT2_GATE (a1, a2, j1, j2, ten, LH, LV, m_proj) in
let x1y1 $|11\rangle_{\mathtt{a}}$ + x1y2 $|10\rangle_{\mathtt{a}}$ + x2y1 $|01\rangle_{\mathtt{a}}$ + x2y2 $|00\rangle_{\mathtt{a}}$ = tensor 2 (λi. if i = 1 then
(x1%LV a1 + x2%LH a1) else (y1%LV a2 + y2%LH a2)) in
let x1y1 $|01\rangle_{\mathtt{j}}$ + x2y1 $|11\rangle_{\mathtt{j}}$ = tensor 2 (λi. if i = 1 then (x1%LH j1+
x2%LV j1) else y1%LV j2) in
let x1y2 $|10\rangle_{\mathtt{j}}$ + x2y2 $|00\rangle_{\mathtt{j}}$ = tensor 2 (λi. if i = 1 then (x1%LV j1+
x2%LH j1) else y2%LH j2) in
constraints $\Rightarrow$ x1y1 $|11\rangle_{\mathtt{a}}$ + x1y2 $|10\rangle_{\mathtt{a}}$ + x2y1 $|01\rangle_{\mathtt{a}}$ + x2y2 $|00\rangle_{\mathtt{a}}$ =
$\frac{1}{4}$ % (x1y1 $|10\rangle_{\mathtt{j}}$ + x1y2 $|11\rangle_{\mathtt{j}}$ + x2y1 $|01\rangle_{\mathtt{j}}$ + x2y2 $|00\rangle_{\mathtt{j}}$)

F TS Gate Behavioral Description

In this appendix, we define the behavioral description for Toffoli Sign gate, where instead of four inputs and four outputs, we have three inputs and three outputs.

TS_outputs (t,y1,y2,y3,LH,LV)	
tensor 4 (λi. if i = 1 then LV y1 elif i = 2 then LV y2 elif i = 3 then LV y3 else vac t$4)	tensor 3 (λi. if i = 1 then LV y1 elif i = 2 then LV y2 else LV y3)
tensor 4 (λi. if i = 1 then LV y1 elif i = 2 then LV y2 elif i = 3 then LH y3 else vac t$4)	tensor 3 (λi. if i = 1 then LV y1 elif i = 2 then LV y2 else LH y3)
tensor 4 (λi. if i = 1 then LV y1 elif i = 2 then vac y2 elif i = 3 then LV y3 else LH t$4)	tensor 3 (λi. if i = 1 then LV y1 elif i = 2 then LH y2 else LV y3)
tensor 4 (λi. if i = 1 then LV y1 elif i = 2 then vac y2 elif i = 3 then LH y3 else LH t$4)	tensor 3 (λi. if i = 1 then LV y1 elif i = 2 then LH y2 else LH y3)
tensor 4 (λi. if i = 1 then LH y1 elif i = 2 then LV y2 elif i = 3 then LV y3 else vac t$4)	tensor 3 (λi. if i = 1 then LH y1 elif i = 2 then LV y2 else LV y3)
tensor 4 (λi. if i = 1 then LH y1 elif i = 2 then LV y2 elif i = 3 then LH y3 else vac t$4)	tensor 3 (λi. if i = 1 then LH y1 elif i = 2 then LV y2 else LH y3)
tensor 4 (λi. if i = 1 then LH y1 elif i = 2 then vac y2 elif i = 3 then LV y3 else LH t$4)	tensor 3 (λi. if i = 1 then LH y1 elif i = 2 then LH y2 else LV y3)
tensor 4 (λi. if i = 1 then LH y1 elif i = 2 then vac y2 elif i = 3 then LH y3 else LH t$4)	tensor 3 (λi. if i = 1 then LH y1 elif i = 2 then LH y2 else LH y3)

TF_inputs (t,x1,x2,x3,LH,LV)	
tensor 3 (λi. if i = 1 then LV x1 elif i = 2 then LV x2 else LV x3)	tensor 4 (λi. if i = 1 then LV x1 elif i = 2 then LV x2 elif i = 3 then LV x3 else vac t$4)
tensor 3 (λi. if i = 1 then LV x1 elif i = 2 then LV x2 else LH x3)	tensor 4 (λi. if i = 1 then LV x1 elif i = 2 then LV x2 elif i = 3 then LH x3 else vac t$4)
tensor 3 (λi. if i = 1 then LV x1 elif i = 2 then LH x2 else LV x3)	tensor 4 (λi. if i = 1 then LV x1 elif i = 2 then vac x2 elif i = 3 then LV x3 else LH t$4)
tensor 3 (λi. if i = 1 then LV x1 elif i = 2 then LH x2 else LH x3)	tensor 4 (λi. if i = 1 then LV x1 elif i = 2 then vac x2 elif i = 3 then LH x3 else LH t$4)
tensor 3 (λi. if i = 1 then LH x1 elif i = 2 then LV x2 else LV x3)	tensor 4 (λi. if i = 1 then LH x1 elif i = 2 then LV x2 elif i = 3 then LV x3 else vac t$4)
tensor 3 (λi. if i = 1 then LH x1 elif i = 2 then LV x2 else LH x3)	tensor 4 (λi. if i = 1 then LH x1 elif i = 2 then LV x2 elif i = 3 then LH x3 else vac t$4)
tensor 3 (λi. if i = 1 then LH x1 elif i = 2 then LH x2 else LV x3)	tensor 4 (λi. if i = 1 then LH x1 elif i = 2 then vac x2 elif i = 3 then LV x3 else LH t$4)
tensor 3 (λi. if i = 1 then LH x1 elif i = 2 then LH x2 else LH x3)	tensor 4 (λi. if i = 1 then LH x1 elif i = 2 then vac x2 elif i = 3 then LH x3 else LH t$4)

References

[1] HOL-Light Source Code. `http://hvg.ece.concordia.ca/projects/optics/tdaqc.html`.

[2] A. Politi, J. C. F. Matthews, and J. L. O'Brien. Shor's Quantum Factoring Algorithm on a Photonic Chip. *Science*, 325(5945):1221, 2009.

[3] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. In *the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, 1994.

[4] S. J. Devitt A. G. Fowler and L. C. L. Hollenberg. Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array. *Quantum Information & Computation*, 4(4):237–251, July 2004.

[5] A. U. Khalid, Z. Zilic and K. Radecka. FPGA Emulation of Quantum Circuits. In *International Conference on Computer Design*, pages 310–315, 2004.

[6] D. Aharonov. A simple proof that Toffoli and Hadamard are quantum universal. arXiv preprint quant-ph/0301040, 2003.

[7] E. Bernstein and U. Vazirani. Quantum Complexity Theory. In *Symposium on Theory of Computing, ACM*, pages 11–20, 1993.

[8] J. W. Bruce, M. A. Thornton, L. Shivakumaraiah, P. S. Kokate, and X. Li. Efficient Adder Circuits Based on a Conservative Reversible Logic Gate. In *IEEE Computer Society Annual Symposium on VLSI*, pages 74–79, 2002.

[9] S. T. Cheng and C. Y. Wang. Quantum Switching and Quantum Merge Sorting. *IEEE Transactions on Circuits and Systems I*, 53(2):316–325, February 2006.

[10] D. P. DiVincenzo. The Physical Implementation of Quantum Computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.

[11] G. F. Viamontes, M. Rajagopalan, I. L. Markov and J. P. Hayes. Gate Level Simulation of Quantum Circuits. In *Asia and South Pacific Design Automation Conference*, pages 295–301, 2003.

[12] O. Golubitsky, S. M. Falconer, and D. Maslov. Synthesis of the Optimal 4-bit Reversible Circuits. In *Design Automation Conference*, pages 653–656, 2010.

[13] D. Grosse, R. Wille, G.W. Dueck, and R. Drechsler. Exact Multiple-Control Toffoli Network Synthesis With SAT Techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):703–715, May 2009.

[14] L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.

[15] J. Harrison. The HOL Light Theory of Euclidean Space. *Journal of Automated Reasoning*, 50(2):173–190, February 2013.

[16] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski. Quantum Logic Synthesis by Symbolic Reachability Analysis. In *Design Automation Conference*, pages 838–841, 2004.

[17] W. N. N. Hung, S. Xiaoyu, Y. Guowu, Jin Y., and M. Perkowski. Optimal Synthesis of Multiple Output Boolean Functions Using a Set of Quantum Gates by Symbolic Reachability Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1652–1663, September 2006.

[18] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik. Simulation of Electronic Structure Hamiltonians Using Quantum Computers. *Molecular Physics*, 109(5):735–750, 2011.

[19] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[20] J. Harrison. HOL light: An overview. In *Theorem Proving in Higher Order Logics*, volume 5674 of *LNCS*, pages 60–66. Springer, 2009.

[21] K. L. Brown , W. J. Munro, and V. M. Kendon. Using Quantum Computers for Quantum Simulation. *Entropy*, 12:2268–2307, 2010.

[22] E. Knill, R. Laflamme, and G. J. Milburn. A Scheme for Efficient Quantum Computation with Linear Optics. *Nature*, 409:46–52, 2001.

[23] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn. Linear Optical Quantum Computing with Photonic Qubits. *Reviews of Modern Physics*, 79:135–174, 2007.

[24] L. Liang and C. Li. Realization of Quantum SWAP Gate Between Flying and Stationary Qubits. *Physical Review A*, 72:024303, August 2005.

[25] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2011.

[26] M. Y. Mahmoud, V. Aravantinos and S. Tahar. Formalization of Infinite Dimension Linear Spaces with Application to Quantum Theory. In *Nasa Formal Methods*, volume 7871 of *LNCS*, pages 413–427. Springer, 2013.

[27] M. Y. Mahmoud. *Formal Analysis of Quantum Optics*. PhD thesis, Concordia University, September 2015.

[28] L. Mandel and E. Wolf. *Optical Coherence and Quantum Optics*. Cambridge University Press, 1995.

[29] G. J. Milburn. Quantum optical Fredkin gate. *Physical Reviw Letter*, 62:2124–2127, May 1989.

[30] P. E. Black and A. W. Lane. Modeling Quantum Information Systems. In *SPIE, Quantum Information and Computation II*, pages 340–347, 2004.

[31] R. Wille P. Niemann and R. Drechsler. Efcient Synthesis of Quantum Circuits Implementing Clifford Group Operations. In *Asia and South Pacific Design Automation Conference*, pages 483–488, 2014.

[32] R. P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982.

[33] T. C. Ralph, A. Gilchrist, G. J. Milburn, W. J. Munro, and S. Glancy. Quantum Computation with Optical Coherent States. *Physical Review A*, 68:042319, October 2003.

[34] T. C. Ralph, K. J. Resch, and A. Gilchrist. Efficient Toffoli Gates Using Qudits. *Physical Review A*, 75:022313, February 2007.

[35] S. F. Arnold, S. J. Gay and I. V. Puthoor. Quantum Process Calculus for Linear Optical Computing. In *Reversible Computation*, volume 7948 of *LNCS*, pages 234–246. Springer, 2013.

[36] S. J. Gay, R. Nagarajan, and N. Papanikolaou. QMC: A Model Checker for Quantum Systems. In *Computer Aided Verification*, volume 5123 of *LNCS*, pages 543–547. Springer, 2008.

[37] C. Hilken S. Mathias, R. Wille and N. Przigoda. Synthesis of Reversible Circuits with Minimal Lines for Large Functions. In *Asia and South Pacific Design Automation Conference*, page 8592, 2012.

[38] S. Yamashita and I. L. Markov. Fast Equivalence-Checking for Quantum Circuits. In *IEEE/ACM International Symposium on Nanoscale Architectures*, page 2328. IEEE Press, 2010.

[39] Z. Sasanian, R. Wille, and D. M. Miller. Realizing Reversible Circuits Using a New Class of Quantum Gates. In *Design Automation Conference*, pages 36–41, 2012.

[40] Quantum Information Science and Technology Experts Panel. A Quantum Information Science and Technology Roadmap. `http://qist.lanl.gov/qcomp_map.shtml`, April, 2004.

[41] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of Quantum Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):10001010, June 2006.

[42] T. C. Ralph, N. K. Langford, T. B. Bell, and A. G. White. Linear Optical Controlled-Not Gate in the Coincidence Basis. *Physical Review A*, 65:062324, 2002.

[43] G. F. Viamontes, I. L. Markov, and J. P. Hayes. Graph-Based Simulation of Quantum Computation in the Density Matrix Representation, 2004.

[44] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal Methods: Practice and Experience. *ACM Computing Survey*, 41(4):19:1–19:36, 2009.