# Towards the Design Automation of Quantum Circuits

Sidi Mohamed Beillahi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical & Computer Engineering)

Concordia University

Montréal, Québec, Canada

June 2016

© Sidi Mohamed Beillahi, 2016

# Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Sidi Mohamed Beillahi**

Entitled: **Towards the Design Automation of Quantum Circuits**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Electrical & Computer Engineering)**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Dr. Robin Raut (Chair)

_____ Dr. Volker Haarslev (Examiner)

_____ Dr. Otmane Ait Mohamed (Examiner)

_____ Dr. Sofiène Tahar (Supervisor)

Approved by _____

Chair of the ECE Department

_____ 2016 _____

Dean of Engineering

# Abstract

Towards the Design Automation of Quantum Circuits

Sidi Mohamed Beillahi

Quantum mechanics based computing systems are expected to have high capabilities and are considered good candidates to replace classical cryptography and supercomputing systems. Among many implementations, quantum optics systems provide a promising platform to implement universal quantum computers, since they link quantum computation and quantum communication in the same framework. Recently, several quantum gates, circuits, and protocols have been experimentally realized using optics. Despite the fact that big advances in building the physical quantum computers were achieved, there are no currently available industrial computer aided tools that can perform the modeling, analysis, and verification of optical quantum computing systems. In this thesis, we tackle the idea of design automation for quantum circuits, where we use a sound language, higher order logic, to model and reason about quantum circuits formally. In particular, we propose a framework for the hierarchical modeling and automated verification of quantum computing circuits. The modeling approach captures quantum models built hierarchically from quantum gates, which models are readily available in a library. The analysis and verification of composed circuits is done seamlessly based on dedicated mathematical foundations formalized in the theorem prover. Specifically, the tensor product and linear projection are used to extract the quantum circuit outputs. Subsequently, a rich library of quantum gates which includes 1-qubit, 2-qubit, and 3-qubit gates is formalized. In order to automate the analysis process, we developed a decision procedure to eliminate the need of user guidance throughout the formal proofs. To demonstrate the effectiveness of

the proposed framework, we conduct the formal analysis of a benchmark of quantum circuits including the Shor's integer factorization algorithm, the Grover's oracle, and the quantum full adder.

**To My Family and Friends**

# Acknowledgments

To start, I would like to express my deep gratitude to my supervisor Dr. Sofiène Tahar, who provided me with full support and supervision to carry my Master's thesis in the best conditions. Without his continuous encouragement and guidance, I cannot imagine how difficult my thesis would be. Dr. Tahar taught me the precious academics skills that any scientist dreams of and I will forever be in debt for it.

I would like to thank very much all my colleagues from the Hardware Verification Group. Thank you for all the help and motivation and for the good times we spent together inside and outside the lab. Especially, I would like to thank Dr. Mohamed Yousri Mahmoud, who guided and helped me to successfully build and implement my thesis ideas in the area of quantum computing. I would like also to thank Ahmed Hachani and Muhammad Shirjeel for taking the time to go over the thesis.

I am deeply thankful to my parents, who have supported and guided me through the journey I am taking. The wisdom and love they planted in my life are invaluable. The great support of my family enabled me to focus on my studies during all the years. Thank you very much to my beloved mother, father, sisters, and brothers.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| BDD | Binary Decision Diagrams |
| BS | Beam Splitter |
| CAD | Computer Aided Design |
| CNOT | Controlled-Not |
| CZ | Controlled-Phase |
| FPGA | Field-Programmable Gate Array |
| FOL | First-Order Logic |
| HOL | Higher-Order Logic |
| NS | Non-Linear Sign |
| OCaml | Objective Caml |
| PSH | Phase Shifter |
| Qubit | Quantum Bit |
| SAT | Boolean Satisability |
| SMT | SAT Modulo Theory |
| TS | Toffoli Sign |

# Chapter 1

# Introduction

## 1.1 Motivation

Quantum physics [16] was developed at the beginning of the twentieth century to comprehend the fundamental forces of nature, in particular at microscopic scale. The main difference between classical and quantum physics lies in the two principles of *superposition* (i.e., an object could be in two places at the same time) and *entanglement* (i.e., two objects in remote locations without physical connection could be instantaneously connected). Due to these principles, it has been proved that classical machines cannot simulate quantum physics in polynomial times [11]. Accordingly, scientists were working to develop new systems, namely quantum computers, which employ quantum physics principles to increase the efficiency of the current computing and security systems. Throughout their research, quantum technologies showed a good potential for providing solutions to several challenges such as secure communication, and most significantly faster computation.

Quantum computing [36] is a typical example of reversible computing which requires that the output contains enough information to reconstruct the input, i.e., no input information is erased, which also means no energy dissipation due to information loss. On the other hand, traditional computing is characterized by energy dissipation because it is logically irreversible. Beside energy saving, the class of problems that

can be efficiently tackled by quantum computers includes several currently insolvable problems by classical computers (e.g., integer factorization algorithm). Interestingly, the integer factorization algorithm is crucial as it can be used to break cryptographic codes that are widely employed in monetary transactions on the Internet [7].

Quantum optics [14] is considered as one of the rich and promising approaches for realizing quantum machines. During the last two decades, research has shown significant progress in linear optics for building quantum computers. For instance, in [25], the authors have shown that it is possible to create a "universal" quantum computer using only single photon sources, detectors, and linear optical elements (e.g., *beam splitter* and *phase shifter*) [26]. However, like its peer approaches, optical quantum computing suffers from several practical limitations 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). Nevertheless, in this thesis we are focusing on quantum optics computing systems where the quantum bits (qubits) are considered as single photons that can be in two different optical modes (e.g., horizontal or vertical) which is called a dual-rail representation [26].

Quantum circuits [36] are networks of quantum gates connected by wires. The quantum gates represent quantum transformations while the wires represent the qubits on which the gates act. The main difference between classical and quantum gates is that quantum gates have the same number of inputs as outputs. In general, a quantum circuit computation is randomized and the probabilities can be negative. Quantum computing circuits like any other physical or engineering circuits need Computer Aided Design (CAD) tools to facilitate their design and deployment in real life applications. These CAD tools help in the realization of new designs and evaluation of their efficiency without the need for expensive laboratory setups. Generally, quantum circuits rely heavily on mathematical models of quantum physics principles (i.e., infinite linear spaces), whereas, current languages and tools are based on Boolean logic. Therefore, building tools that can model, synthesize, verify, and

ensure the full functionalities of quantum circuits is 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 machines and tools that can verify and simulate the functionalities of quantum machines.

Because quantum logic is mainly based on infinite linear spaces theory, they cannot be handled by current CAD tools, which were conceived for the analysis of Boolean based machines. Hence, we believe that there is a dire need of comprehensive and expressive computer-aided design and verification tools for quantum systems that cover both the mathematics and the principles of quantum physics. Formal methods based techniques [54] allow for expressive and accurate analysis of reversible quantum systems and have the potential to provide the necessary mathematical foundation. The main idea behind formal methods is to construct a computer based mathematical model of the given system. Higher-order-logic (HOL) theorem proving [19] is an effective formal methods approach to analyze physics and engineering systems, thanks to its solid mathematics foundations, which fulfil the main requirement for the modeling of quantum systems.

Recently, a comprehensive linear algebra library was formalized in the HOL Light theorem prover [30]. Using this library, the author of [29] has formalized the notions of quantum optics single mode and multi modes, beam splitter, and phase shifter. Based on this work, our ultimate goal is to build the necessary tools to formally model and verify quantum circuits composed using quantum gates that are built using only optical components such as beam splitter and phase shifter, in a hierarchical fashion. The first step towards this goal is to formally define in HOL the required mathematics which are the notions of linear projection, tensor product, and tensor product projection and to prove the associated properties. The second step is to apply this mathematical formalization to formally model and verify a library of quantum gates that is rich enough to model a variety of quantum circuits. This library includes 1-qubit, 2-qubit, and 3-qubit gates. Because of the underlying logic, HOL theorem

proving is an interactive approach. The analysis of any quantum circuit within HOL would require user guidance, which is tedious in particular for large circuits. Therefore, the third step is to develop a decision procedure that fully automate the analysis of any given quantum circuit that can be constructed using the existing gates library. The developed automation procedure eliminates the need for user interaction with the interactive HOL theorem prover. Finally, the last step is to demonstrate the utility and the effectiveness of the previous steps through the analysis of several real world quantum circuits.

## 1.2 Related Work

The field of quantum computing is one of the hottest in physics and computer science as many researchers believe it is only a matter of time and intensified effort before a large-scale quantum computer is built. Due to the tremendous amount of research conducted in the field of quantum computing, scientists and engineers use different approaches to build CAD tools to analyze the corresponding systems based on analytical and numerical models. In this section, we provide an overview of these techniques and highlight their strengths and weaknesses. In particular, we can subdivide CAD approaches related to quantum computing into three main categories: quantum circuits simulation and emulation, synthesis of quantum circuits, and formal verification of quantum circuits.

### 1.2.1 Quantum Simulation and Emulation

Numerical simulations are used to study the behavior of engineering 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 [40] and quantum principles, numerical simulations are incomplete: the computation space increases exponentially with the size of the quantum circuit. Nevertheless, a number of approaches have been persuaded to simulate

quantum circuits using existing emulation and numerical simulation techniques. For example, several methods and tools for numerical simulation have been proposed, where the quantum gates are described as matrices and applied to quantum states (described as vectors) using matrix-vector multiplication (e.g., [8, 51, 52]). In these work, the simulations were performed at the gate level without modeling 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 [24] 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 quantum circuits with a limited number of qubits.

### 1.2.2 Quantum 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 synthesis are required. Accordingly, efficient quantum circuit synthesis became an active field of research in design automation. Several approaches addressing quantum circuits synthesis have been proposed, whether universally (i.e., synthesis with no restriction on the gates types) [23, 48], exploiting synthesis methods for reversible circuits [15, 34, 47] or using a fixed set of quantum gates such as the Clifford group (a set of quantum gates) [37]. Another area of research in quantum circuits synthesis 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 the quantum gates are restricted to work on adjacent qubits (nearest neighbor quantum circuits) [28, 53]. The drawback of all the above approaches is that synthesis and optimization of the quantum circuits is conducted at the behavioral level rather than the physical level of the elementary gates design, which limits these work in finding the optimal design. For example, in [15, 47], the Toffoli gate circuit was synthesized into five 2-qubit gates, however, it is

possible to implement it in quantum optics using only three 2-qubit gates as shown in [45], by utilizing the photons polarization structure of the qubits. In another related work [22], the authors used model checking to formally synthesise quantum circuits based on a fixed set of 1-qubit and 2-qubit gates. The authors have demonstrated the optimized design for Toffoli and Fredkin gates, however, this remains valid only when dealing with circuits at the behavioral level. Moreover, another work [17] using formal techniques: Boolean satisability (SAT) and SAT modulo theory (SMT) for the synthesis of Toffoli networks at the behavioral level. However, this work suffers from the same limitation described earlier regarding the level at which the analysis is conducted.

### 1.2.3 Quantum Formal Verification

Developing methods and tools for the formal modeling and verification for quantum circuits is a promising subject in CAD research, since formal methods have rich mathematics foundation and quantum circuits exhibit many mathematical notions that need to be tackled. In [55], Binary Decision Diagrams (BDD) have been used for the equivalence checking of reversible quantum circuits through the classification of 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 a generic modeling analysis of quantum circuits. In [13], the CWB-NC model checker has been used to formally analyse quantum communication protocols that are implemented as quantum circuits. The authors have verified the quantum coin-flipping protocol. The proposed technique, however, can only be used for protocols that are described as finite-state models. The focus of the work in [13] 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. Another related work in using formal methods is [3], in which the authors developed a special quantum process calculus 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 [33]. So far, the proposed process calculus has not been demonstrated on scalable quantum circuits.

Using the formalized linear algebra library in HOL Light theorem prover, in [31], the authors conducted the formalization[1] of the flip gate based on coherent states. Then in [32], the authors formalized the optical beam splitter, phase shifter, Mach-Zehnder interferometer and the controlled-not (CNOT) gate.

## 1.3 Proposed Methodology

The objective of this thesis is mainly targeted towards the development of a theorem proving based automated analysis framework for quantum optics circuits that can handle the modeling and analysis of real-world quantum circuits. In particular, we propose to develop a framework in HOL Light characterizing:

- The ability to formally express the notions of measurement in quantum optics (i.e., measurement of quantum circuits outputs) in a mathematical form.
- The ability to use the developed infrastructure to analyze different types of optical quantum gates and circuits.
- The ability to formally model the quantum circuits in a systematic way with no restriction on the number of quantum gates.
- The ability to perform the formal analysis of quantum circuits automatically without the need for the user to guide the theorem prover.

[1] A system is considered as formalized if it is stated in a formal language, with enough detail that a computer program (proof assistant or theorem prover) can mechanically verify properties of the system and thereby certifying its correctness.

The proposed framework, given in Figure 1.1, outlines the above mentioned characteristics and the main idea about the automated formal design, modeling, and verification of quantum optical circuits within the sound core of HOL theorem proving. The whole framework can be decomposed into three major parts. First, the *mathematical foundation*, where we have the main requirement to model quantum states in multi-modes and the measurement of quantum states in single mode (i.e., quantum state contains only one mode of light) and multi-modes (i.e., quantum state contains several modes of light). In particular, we formalize the notions of tensor product projection and its associated properties using the existing formalization of infinite linear spaces. Having the required mathematical tools to analyze any quantum system, the second part of this thesis is to build a rich library of the most important quantum gates which are built hierarchically from simple gates based on optical elements (i.e., beam splitter and phase shifter) to complicated gates that are composed of other gates. The third part of the framework is to take a quantum circuit netlist and a set of inputs and establish a HOL theorem for the outputs relations and success probability (i.e., the probability at which the quantum circuit produces the correct outputs). To facilitate the proof of the HOL theorem automatically without the need of user guidance, we developed a decision procedure that automates the proof. Note that the underlying framework can also be used to formally check if inputs and outputs correspond to each other for a given quantum circuit.

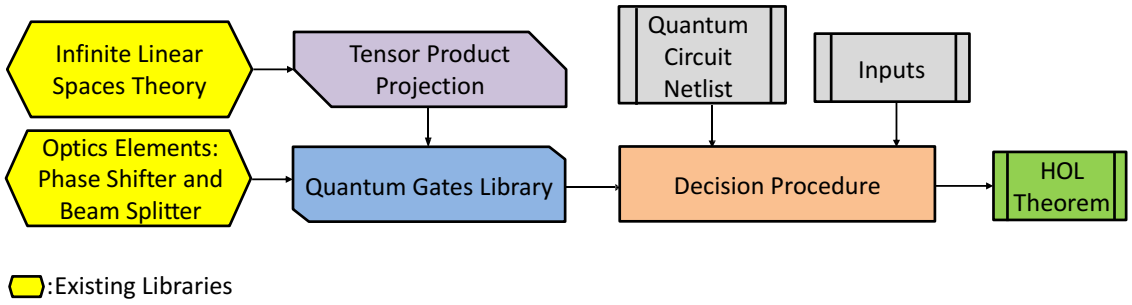


Figure 1.1: Modeling and Verification Methodology

## 1.4 Thesis Contributions

The main contribution of this thesis is about the idea of applying HOL theorem prover to handle the hierarchical analysis of optical quantum computing circuits. We develop a formal framework on top of the trusted kernel of HOL Light theorem prover which ultimately allows the precise analysis of quantum optics circuits. Furthermore, the high expressiveness of HOL Light provides a suitable environment for dealing with all kinds of mathematics and hence it is the proper tool to tackle the complete analysis of optical quantum computing systems. Our proposed approach can be considered as a complementary method to other state-of-the-art but less accurate and complete techniques like numerical simulation and lab simulation based analysis. We list below the main contributions of this thesis:

- The formalization of the basic notions of tensor product, linear projection and tensor product projection.
- The formalization of a rich library of optical quantum gates that contains: 1-qubit gates, 2-qubit gates, and 3-qubit gates.
- The development of an automated decision procedure to fully automate the analysis process without the need for user interaction with the theorem prover.
- The formalization of several practical quantum applications including the quantum full adder, Grover's oracle, and Shor's algorithm for factoring the number 15.

## 1.5 Thesis Outline

The rest of this thesis is organized as follows: In Chapter 2, we introduce some basic concepts of quantum mechanics including the notions of quantum states, fock states, and quantum operators. Afterward, we describe the tensor product to model quantum states in multi-modes. We also present the field of quantum computing and the optics

approach for implementing a quantum machine. Then, we provide an overview of the HOL Light theorem prover along with some of its useful features. Finally, we describe the formalization of the phase shifter and beam splitter which our work is based on.

In Chapter 3, we describe the formalization of tensor product and related properties such as bilinearity, tensor of zero, and tensor of tensor. We also present the development of linear projection and its properties such as linearity, idempotent, and self-adjoint. We then combine these two notions to formalize the tensor product projection and its properties such as: projection of tensor of tensor, and projection of tensor of fock states.

In Chapter 4, we present the HOL formalization of quantum optical gates. In particular, we describe the formal modeling and verification of the Hadamard, the non-linear sign, and the flip gates, which are 1-qubit gates. Then, we present the formalization of the controlled phase, the controlled not, and the SWAP gates which are 2-qubit gates. Finally, we provide the formal modeling and verification of the Toffoli sign, the Toffoli, and the Fredkin gates which are 3-qubit gates. In this chapter, we highlight the usage of tensor product projection properties to obtain the expected output of the non-linear sign and controlled phase gates. We also discuss a novel design of the flip gate and how the Toffoli gate design can be optimized in terms of the number of gates compared to existing models.

In Chapter 5, we present the automated verification of quantum circuits. In particular, we demonstrate the process of verifying a quantum circuit through the verification of the Shor's integer factorization of the number 15 circuit. Then, we present a decision procedure to fully automate the quantum circuits verification process. We describe the idea behind the decision procedure and the lemmas for tensor product folding and unfolding involved in the construction of the final tactics to be used to perform the automatic analysis. In order to highlight the effectiveness and the benefit of the proposed framework, we provide a detailed analysis of the quantum full adder. Finally, we provide the result of the analysis for a number of benchmark quantum circuits that were taken from the online library for quantum circuits [46].

This chapter also highlights the benefit of our approach of extracting the success probabilities of the analyzed quantum circuits.

Finally, Chapter 6 concludes this thesis by providing some remarks about the developed framework including a description of some challenging aspects of our work and potential future research directions.

# Chapter 2

# Preliminaries

In this chapter, we start by giving a preliminary overview on quantum mechanics along with the tensor product. Then, we give an introduction about quantum computing and quantum optics. Subsequently, we provide a brief overview of theorem proving technique and HOL Light theorem prover. Finally, we present the existing formalization of two important quantum optics elements, namely beam splitter and phase shifter. The intent of this chapter is to introduce the basic theories along with some notations of the theorem proving environment that we use in the rest of this thesis.

## 2.1 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 [16]. Generally, quantum mechanics is composed of many physics aspects such as: the quantum measurement, Bell's theorem, and wave-particle duality [16]. 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 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 using the "ket" (i.e., $|\ldots\rangle$) notation 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 into 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.

## 2.2 Tensor Product

In this section, we give an overview of the tensor product of Hilbert spaces [40]. 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 [40] and all elements of the tensor vector space $V_1 \otimes V_2 \otimes ... \otimes V_n$ should satisfy the following properties of biltinearity:
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 [1]. 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 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.

## 2.3 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 measurements 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 realizations 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:

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

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 $a1$, $a2$, $b1$, and $b2$ 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.

## 2.4 Quantum Optics

Quantum optics is considered as one of the rich and promising approaches under investigation for realizing quantum computers [25, 42]. 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., $|\psi\rangle \equiv \sum_n \psi_n\,|n\rangle$. The main operators which act on fock states are

the annihilation and creation operators:

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

For the case of a quantum state $|\Psi\rangle$ where there are $N$ 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 $|\psi\rangle_j$:

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

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 [25], the authors showed that given single photon sources and single-photon detectors, linear optics alone would suffice to implement efficient quantum computation. Moreover, most 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.

### 2.4.1 Phase Shifter

An important optical component is the single-mode phase shifter which provides a phase shift in a given mode: $\hat{a}_{o1} = e^{i\theta}\hat{a}_{i1}$ [14]. A phase shifter $P_\theta$ ($\theta$ 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.

### 2.4.2 Beam Splitter

Beam splitter [14] is a two mode passive linear optical element that consists of a semi reflective mirror: when a light beam falls on the 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 $\theta$ and $\varphi$, where $\cos\theta$ and $\sin\theta$ are the probability amplitudes and $\varphi$ 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 2.1. 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 $\varphi = 0$.

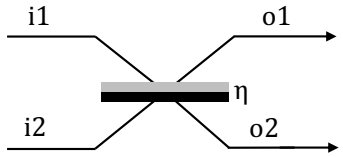


Figure 2.1: Schematics of Beam Splitter

### 2.4.3 Quantum Gates

Many universal quantum gates have been implemented using linear optics elements: such as the above described phase shifter and beam splitter. These include 1-qubit gates such as the Hadamard, Flip, and Non-linear sign gates; 2-qubit gates such as the controlled-phase (CZ) and controlled-not (CNOT) gates which are constructed using

Non-linear sign (NS) gates [26]; and 3-qubit gates such as the 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 [26]. Although most existing implementations of quantum gates in linear optics are probabilistics, it has been proposed to use teleportation to improve the efficiency to become deterministic (near-deterministic) by teleporting the successful gate outputs [26].

## 2.5 Theorem Proving

High-order logic (HOL) is a mathematical logic language which encompasses the notions of quantification over functions, and functions defined on functions. Hence, this gives HOL an advantage over the other types of mathematical logic (i.e., propositional logic, first-order logic, and second-order logic). HOL is employed for the reasoning about systems as mathematical objects in order to prove properties about them. There exist several HOL based theorem provers, the most popular are Isabelle/HOL [38], Coq [10], HOL4 [50], HOL Light [20] and PVS [39].

The work proposed in this thesis is conducted within HOL Light because it provides rich multivariate analysis libraries [21]. HOL Light has been employed to verify generic properties of a wide class of software, hardware and physical systems as well as a platform for the formalization of pure mathematical theorems. 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]. In this thesis, we make use of the HOL Light theories of complex numbers, transcendental functions, functional spaces, and multivariate analysis. In fact, one of the primary motivations of selecting the HOL Light theorem prover for

our work was to benefit from these built-in mathematical theories. In Table 2.1, we provide the mathematical interpretations of some HOL Light notations that will be used in this thesis. In the following, we provide a couple of examples to illustrate how definitions and theorems are written in HOL Light:

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

$$\vdash \texttt{f\_min} \iff (\lambda\ (\texttt{x} : \texttt{real})\ (\texttt{y} : \texttt{real}).\ \texttt{if x} \geq \texttt{y then y else x}) \tag{11}$$

The if statement is part of the HOL Light which allows to choose between two alternatives according to a given condition. `f_min x y` returns the minimum of the given parameters `x` and `y`. The type of `f_min` is `real` $\rightarrow$ `real` $\rightarrow$ `real`.

Another way to define in HOL Light is by using a predicate which accepts an integer parameter and returns true if this integer is even and false otherwise:

$$\vdash \texttt{is\_even}\ (\texttt{n} : \texttt{num}) \iff (\texttt{if}\ (\texttt{n rem 2} = \texttt{0})\ \texttt{then T else F}) \tag{12}$$

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

## 2.6 Formalization of Quantum Optics Elements

In this section, we present the formalization of beam splitter and phase shifter (cf. Section 2.4), which have been developed in [29].

### 2.6.1 Phase Shifter

The annihilator and creator operators for the input and output of the phase shifter transformation are related as: $\hat{a}_{o1} = e^{i\theta}\hat{a}_{i1}$ and $\hat{a}^{\dagger}_{o1} = e^{-i\theta}\hat{a}^{\dagger}_{i1}$, respectively. Hence, the

Table 2.1: 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 |
| 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 |
| A:real | $A : \mathbb{R}$ | Specify type operator |
| suc n | $(n + 1)$ | Successor of natural number |
| abs x | $\|x\|$ | Absolute function |
| &a | $\mathbb{N} \to \mathbb{R}$ | Typecasting from Integers to Reals |
| Cx a | $\mathbb{R} \to \mathbb{C}$ | Typecasting from Reals to Complexes |
| [a; b; ..] | [a; b; ..] | Lists |
| {x\|P(x)} | $\{x \mid P(x)\}$ | Set of all x such that $P(x)$ |
| lambda x. v x | $\lambda\, x.\ v\ x$ | Vectors lambda |
| λ | $\lambda$ | Lambda abstraction (required for functions definition) |
| x$i | x(i) | Vector indexing operator |
| x pow n | $x^n$ | Real and complex power |
| a % V | a . V | Scalar multiplication |
| -- | - | Arithmetic negation |
| ** | * | Operator multiplication |
| A → B | $A \to B$ | Domain to Codomain |
| f o g | f o g | Function composition |
| let var = exp1 in exp2 (var) | exp2 (exp1) | exp1 is evaluated first then exp2 is evaluated with var bound to the value produced by the evaluation of exp1 |

phase shifter transformation is formally defined in HOL as follows [29]:

**Definition 2.1** (Phase Shifter)**.**

$\vdash$ `phase_shifter(ten, θ, i1, m1, o1, m2)` $\Longleftrightarrow$

1 `(is_sm i1)` $\wedge$ `(is_sm o1)` $\wedge$ `(w i1 = w o1)` $\wedge$ `(vac i1 = vac o1)` $\wedge$

2 `(pos ten (cr i1) m1 =` $\mathtt{e}^{(-\mathtt{j}*\theta)}$ `% pos ten (cr o1) m2)` $\wedge$

3 `(pos ten (anh i1) m1 =` $\mathtt{e}^{(\mathtt{j}*\theta)}$ `% pos ten (anh o1) m2)`

where `ten` is the tensor product operator and `pos` is used to position a given operator in a specific mode (based on its order in the input vector) and leave the other modes with the identity operator. `anh x` and `cr x` designate the annihilator and creator operators for the mode $x$, respectively. `i1` and `o1` are the input and output,

respectively. Notice that the formal definition of the phase shifter relates the input operators (anh i1 and cr i1) in terms of the output operators (anh o1 and cr o1), see Lines 2 and 3. In Line 1, the parameters {m1, m2} define the order of each mode in the input vector. Line 1 ensures that the two modes (input and output) are proper optical single modes, and working with the same frequency and vacuum state (i.e., the state of zero photons).

### 2.6.2 Beam Splitter

The beam splitter transformation described in Section 2.4.2 can be formally defined in HOL as follows [29]:

**Definition 2.2** (Beam Splitter)**.**

$\vdash$ is_beam_splitter (p1, p2, p3, p4, ten, i1, m1, i2, m2, o1, m3, o2, m4) $\Longleftrightarrow$

1 (is_sm i1) $\wedge$ (is_sm i2) $\wedge$ (is_sm o1) $\wedge$ (is_sm o2)$\wedge$

2 (w i1 = w i2) $\wedge$ (w i2 = w o1) $\wedge$ (w o1 = w o2) $\wedge$

3 (vac i1 = vac i2) $\wedge$ (vac i2 = vac o1) $\wedge$ (vac o1 = vac o2) $\wedge$

4 (pos ten (anh i1) m1 = p1 % pos ten (anh o1) m3 + p2 % pos ten (anh o2) m4) $\wedge$

5 (pos ten (anh i2) m2 = p3 % pos ten (anh o1) m3 + p4 % pos ten (anh o2) m4) $\wedge$

6 (pos ten (cr i1) m1 = p1$^*$ % pos ten (cr o1) m3 + p2$^*$ % pos ten (cr o2) m4) $\wedge$

7 (pos ten (cr i2) m2 = p3$^*$ % pos ten (cr o1) m3 + p4$^*$ % pos ten (cr o2) m4)

where i1, i2 and o1, o2 represent the beam splitter inputs and outputs modes, respectively. m1 and m2 (resp. m3 and m4) represent the order of the two beam splitter input (resp. output) modes within the inputs (resp. outputs) vector. Similar to the phase shifter, the formal definition of the beam splitter relates the inputs annihilator and creator operators in terms of the outputs annihilator and creator operators (see Lines 4, 5, 6, and 7). The parameters {p1,p2,p3,p4} are the inverse of the beam splitter matrix. Lines 1, 2, and 3 ensure that the four modes are proper single modes, and working with the same frequency and vacuum state.

# Chapter 3

# Formalization of Tensor Product Projection

In Chapter 2, we gave an introduction to quantum theory that shows the importance of tensor product for the analysis of quantum states in multi-modes. This chapter covers in detail the higher-order formalization of the tensor product along with linear projection. In the last part of this chapter, we combine the tensor product and linear projection to obtain the tensor product projection. In our formalization, we are building on top of a linear algebra library [30] which is available as part of the HOL Light libraries [18]. We chose this library as it provides the theory of Hilbert-spaces and the main theorems required for our formalization. Note that Hilbert-spaces are the primary resource for modeling quantum systems.

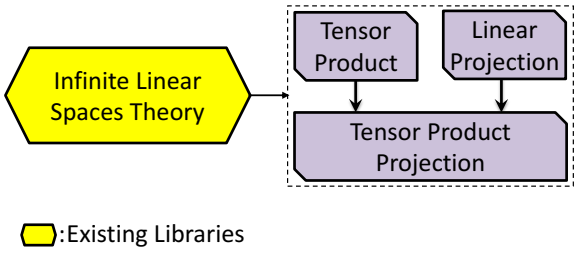


Figure 3.1: Mathematical Foundation

Figure 3.1 shows the main components of the mathematical foundation we develop in this chapter, including tensor product, linear projection, and tensor product projection. For each one we prove the associated properties such as: linearity and self-adjoint for projection, bilinearity and tensor of tensor for tensor product, and projection of tensor of tensor for tensor product projection. Notice that this mathematical foundation is built over the HOL Light library for infinite linear spaces [29].

## 3.1 Formalization of Tensor Products

Given the quantum states $|\psi\rangle_1 \ldots |\psi\rangle_n$ of $n$ optical beams, the function that describes the joint probability of the $n$ beams is then the point-wise multiplication of all the states. Hence, we define the tensor product for an n-beam quantum state as follows: $\lambda\ y_1 \ldots y_n.\ (|\psi\rangle_1 \otimes \ldots \otimes |\psi\rangle_n)(y_1 \ldots y_n) = |\psi\rangle_1 y_1 * \ldots * |\psi\rangle_n y_n$. Thus, we formally define the tensor product for $n$ optical beams in HOL, recursively, as follows:

**Definition 3.1** (Tensor Product)**.**

$\vdash$ `tensor 0 mode` $=$ $(\lambda$`y. 1`$)\ \wedge$

`tensor n + 1 mode` $=$ $(\lambda$`y. ((tensor n mode) y) * (mode$(n + 1) y$(n + 1))`$)$

where `mode` is a vector of size $n$ that contains $n$ modes. The basic case of zero mode $n = 0$ is a trivial case; it is a constant function (i.e., $y \Longrightarrow 1$) and it guarantees a terminating definition. Mathematically, to validate that the underlying tensor product is well defined, we should prove the bi-linearity property. Therefore, the defined tensor should satisfy the following two properties of bi-linearity:

**Theorem 3.1** (Tensor Product: Bi-Linearity)**.**

$\vdash$ `n + 1 ≤ dimindex (:N) ∧ k ≤ n + 1 ∧ 0 < k ∧ mode$k = a % x ⟹`

`tensor n + 1 mode = a % tensor n + 1 (λi. if i = k then x else mode$i)`

$\vdash$ `n + 1 ≤ dimindex (:N) ∧ k ≤ n + 1 ∧ 0 < k ∧ mode$k = x₁ + x₂ ⟹`

`tensor n + 1 mode = tensor n + 1 (λi. if i = k then x₁ else mode$i)`

`+ tensor n + 1 (λi. if i = k then x₂ else mode$i)`

where $\texttt{dimindex}\ (:\texttt{N})$ returns the size of the set $\{\texttt{1}\cdots\texttt{N}\}$. Notice that the number of modes is $n+1$ as this property does not hold for $n=0$, where the tensor product is the constant functions (not linear). The two assumptions $k \leq n+1$ and $0 < k$ ensure that the element $k$ is indeed part of the tensor. The assumption $\texttt{mode\$}k = a \ \%\ x$ ($\texttt{mode\$}k = x_1 + x_2$) ensures that the element can be written in the form of multiplication of scalar and vector (summation of two vectors). With the previous two theorems we have verified that our definition is indeed a tensor product. The proof steps for the two theorems are based on using induction where the base case is trivial and in the inductive step we use the lemma $\texttt{k} \leq \texttt{n}+\texttt{2} \iff (\texttt{k} \leq \texttt{n}+\texttt{1} \vee \texttt{k} = \texttt{n}+\texttt{2})$ then using the induction hypothesis for the first case and the definition of tensor product for the second case. With these properties of bi-linearity, we have validated our tensor product definition.

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

**Theorem 3.2** (Tensor Product: Multiplication (1/2))**.**

$\vdash\ \texttt{m}+\texttt{n} \leq \texttt{dimindex}\ (:\texttt{N}) \Longrightarrow \texttt{tensor m}+\texttt{n mode}\ =$

$(\lambda\texttt{y}.\ ((\texttt{tensor m mode})\ \texttt{y}) * (\texttt{tensor n}\ (\lambda\texttt{i}.\ \texttt{mode\$}(\texttt{i}+\texttt{m})))\ (\lambda\texttt{i}.\ \texttt{y\$}(\texttt{i}+\texttt{m})))$

Here, $\texttt{tensor m mode}$ and $\texttt{tensor n}\ (\lambda\texttt{i}.\ \texttt{mode\$}(\texttt{i}+\texttt{m}))$ represent the elementary tensors $(v_1 \otimes ... \otimes v_m)$ and $(u_1 \otimes ... \otimes u_n)$, respectively. As an example of this property utility, suppose that one of the elementary tensors goes through a quantum transformation alone, using this theorem we can isolate the elementary tensor under consideration and substitute it by the result of the transformation without modifying the state of the second elementary tensor. Furthermore, in order to return back to the initial main tensor we use the following property:

**Theorem 3.3** (Tensor Product: Multiplication (2/2))**.**

$\vdash$ `m + n ≤ dimindex (:N) ⟹`

`(λy. ((tensor m mode1) y) * (tensor n mode2 (λi. y$(i + m))) =`

`tensor (m + n) (λ i. if(i ≤ m) then mode1$i else mode2$i)`

An important property of tensor product which is very useful for orthogonal projection is that $v_1 \otimes ... \otimes 0 \otimes ... \otimes v_n = 0$, which is given in HOL as:

**Theorem 3.4** (Tensor Product: Zero Element)**.**

$\vdash$ `n + 1 ≤ dimindex (:N) ∧ mode$k = |0⟩ ∧ 0 < k ∧ k ≤ n + 1 ⟹`

`tensor n + 1 mode = |0⟩`

The proof of this theorem is done using the induction on the size of the tensor product (i.e., $n$). Finally, we provide an important property in the process of unfolding and folding of the tensor product as follows:

**Theorem 3.5** (Tensor Product: Rewriting)**.**

$\vdash$ `n ≤ dimindex (:N) ⟹`

`tensor n mode = tensor n (λ i. if(1 ≤ i ∧ i ≤ n) then mode$i else g i)`

where g can be any vector as it has no effect in the analysis and its role is only to have valid if, else expression. In the next section, we describe the linear projection and provide some of its important properties.

## 3.2 Formalization of Linear Projection

In linear algebra, a projection is a linear transformation $p$ from a vector space to itself that maintains the idempotent property; $p^2 = p$. In the quantum context, a pure state associated with a state vector $|\psi\rangle \in H$ from a Hilbert space H, the projection over this state is given by $p = |\psi\rangle \langle\psi|$, which is a self-adjoint linear projection. In particular, for a quantum circuit design, the expected circuit output is the projection of all possible outputs over the appropriate fock states. For example, let us consider

the state $|\phi\rangle = \frac{1}{3}|n\rangle + \frac{1}{3}|n-1\rangle + \frac{1}{3}|n+1\rangle$ which is a mixture of three fock states, and the projection $p_n = |n\rangle\langle n|$ over fock state. The result of the projection of $|\phi\rangle$ is $p_n(|\phi\rangle) = |n\rangle\langle n|(\frac{1}{3}|n\rangle + \frac{1}{3}|n-1\rangle + \frac{1}{3}|n+1\rangle) = \frac{1}{3}|n\rangle$, because the fock states form an orthonormal basis (i.e., $\langle n|n\rangle = 1$ and $\langle n|m\rangle = 0$ for $m \neq n$). Therefore, we define the projection on fock states as follows:

**Definition 3.2** (Linear Projection)**.**

$\vdash \forall$ x. (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x $= \langle \mathtt{n}_{\mathtt{sm}}|\mathtt{x}\rangle$ % $|\mathtt{n}\rangle_{\mathtt{sm}}$

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

**Theorem 3.6** (Projection: Linearity)**.**

$\vdash$ is_sm sm $\Longrightarrow$ ($\forall$ x y a. x, y $\in$ sq_integrable $\Longrightarrow$

(proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) (x + y) = (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x + (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) y $\wedge$

(proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) (a % x) = a % ((proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x))

where the assumption is_sm sm is to maintain that the beam *sm* is indeed a quantum single mode beam and that it meets all the requirements [29]. Here sq_integrable is the linear inner product space formed by the square integrable functions space and Lebesgue integral. The proof of this theorem is based on the linearity of inner product. In what follows, we show the projection idempotent property:

**Theorem 3.7** (Projection: Idempotent)**.**

$\vdash$ is_sm sm $\Longrightarrow \forall$x. (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) ((proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x) = (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x

where (proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) ((proj $|\mathtt{n}\rangle_{\mathtt{sm}}$) x) is the application of the projection proj $|\mathtt{n}\rangle_{\mathtt{sm}}$ twice on $x$. The proof of this theorem is based on the conjugate (i.e., $\langle \mathtt{x}|\mathtt{y}\rangle = \langle \mathtt{y}|\mathtt{x}\rangle^*$) and linearity of inner product. Next, we show the property of the self-adjoint for the projection operator (i.e., $\langle p(u)|v\rangle = \langle u|p(v)\rangle$):

**Theorem 3.8** (Projection: Self-Adjoint)**.**

$\vdash$ `is_sm sm` $\Longrightarrow$ ($\forall$ `x y. x,y` $\in$ `sq_integrable` $\Longrightarrow$

`r_inprod x (proj` $|\mathtt{n}\rangle_{\mathtt{sm}}$ `y) = r_inprod (proj` $|\mathtt{n}\rangle_{\mathtt{sm}}$ `x) y`

The proof of this theorem is based on the conjugate (i.e., $\langle \mathtt{x}|\mathtt{y}\rangle = \langle \mathtt{y}|\mathtt{x}\rangle^*$) and linearity of inner product.

## 3.3 Formalization of Tensor Product Projection

In this section, we combine the tensor product for multi-mode and linear projection for single-mode together to obtain the tensor product projection, or in other words the multi-mode projection. In some realization of quantum optics, the quantum gates are implemented using *ancilla* resources which are extra qubits that have a secondary role in a computation and are used for detecting the correct output [26]. During the design process of a quantum circuit, the state of the ancilla is measured after it leaves the circuit using a detector. The correct output is known to have been produced whenever the detector registers the expected ancilla. In our formalization, we implement the design process of detecting the expected ancillas in the output ports of a quantum circuit as the tensor product projection of the circuit outputs. By doing this, we eliminate the undesirable outputs and keep only the "correct" ones. In addition, we obtain the projected state multiplied by a scalar value which is the success probability of the circuit, or the probability in which we detect the expected ancilla. To the best of our knowledge, this is the first time tensor product projection is used as a mathematical analysis tool for quantum optics detection. We define the projection of multi-mode over multi-mode as follows:

**Definition 3.3** (Tensor Projection)**.**

$\vdash$ `is_tensor_proj m_proj` $\Longleftrightarrow \forall$ `mode1 mode2 n.`

`is_linear_cop (m_proj (tensor n mode1))` $\wedge$

`m_proj (tensor n mode1) (tensor n mode2) =`

`tensor n` ($\lambda$`i.((proj mode1$i) mode2$i))`

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

**Theorem 3.9** (Tensor Projection: Multiplication)**.**
$\vdash$ `is_tensor_proj m_proj` $\wedge$ `1` $\leq$ `n` $\Longrightarrow$
(`m_proj tensor m` + `n mode1`) `tensor m` + `n mode2` =
($\lambda$`y`. ((`m_proj tensor m mode1`) `tensor m mode2`) `y` $*$
(`m_proj tensor n` ($\lambda$`i.mode1$(i` + `m`)) `tensor n` ($\lambda$`i.mode2$(i` + `m`))) ($\lambda$`i.y$(i` + `m`)))

The verification of this theorem is based on Theorem 3.2. This property is very useful when projecting multi-mode state which is applied to parallel quantum gates as the case for the controlled-phase gate (see Figure 4.5 where the multi-mode state $|b\$1, b\$2, b\$3, d\$1, d\$2, d\$3\rangle$ is fed to the two parallel NS gates). Using the tensor product lemma $v_1 \otimes ... \otimes 0 \otimes ... \otimes v_n = 0$, we prove the following property:

**Theorem 3.10** (Tensor Projection: Fock States)**.**
$\vdash$ `is_tensor_proj m_proj` $\wedge$ `0` $<$ `k` $\wedge$ `mode1$k` $= |$`m1`$\rangle_{\text{sm}}$ $\wedge$ `mode2$k` $= |$`m2`$\rangle_{\text{sm}}$ $\wedge$
`m1` $\neq$ `m2` $\wedge$ `is_sm sm` $\wedge$ `k` $\leq$ `n` + `1` $\Longrightarrow$
(`m_proj tensor n` + `1 mode1`) `tensor n` + `1 mode2` = `0`

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

state), then the projection of the first multi-mode state over the other is zero (the zero constant function).

## 3.4 Summary

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

# Chapter 4

# Quantum Gates Library

In this chapter, we build upon the mathematical foundation described in the previous chapter to formally model a set of nine quantum gates that includes 1-qubit, 2-qubit, and 3-qubit gates. The gates library includes a new implementation of the flip gate based on a single photon source. Another interesting gate is the Toffoli sign gate which is taken from [45] and has three inputs, where one input is qutrit (i.e., has three quantum states) and the rest are qubits. This makes it impossible to model the Toffoli sign using existing tools because of their Boolean foundation (i.e., compatible with only two states entity). Based on the Toffoli sign gate we can model an optimal implementation of the Toffoli gate.

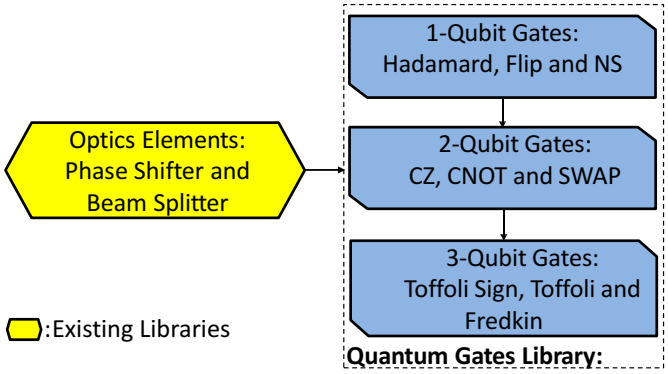


Figure 4.1: Quantum Gates Library

Figure 4.1 depicts the formalized quantum gates library using the existing formalization of the optical elements. We subdivide quantum gates to three categories based on the number of qubits: 1) 1-qubit gates which are the simplest type of gates that are built using only optical elements (i.e., beam splitter and phase shifter). This set contains two deterministic gates (Hadamard and flip gates) and a non-deterministic gate (non-linear sign (NS) gate); 2) 2-qubit gates which are mostly constructed using 1-qubit gates and optical elements and can also be built using 2-qubit gates such as the case for SWAP gate; and 3) 3-qubit gates which contain the most complicated set of gates that are constructed using a mixture of 1-qubit, 2-qubit, and 3-qubit gates.

## 4.1 Formalization of 1-qubit gates

In this section, we show the formal modeling and verification of the Hadamard, bit flip (flip) and non-linear sign (NS) gates using the mathematical formalization presented in the previous chapter. The modeling and verification of other 1-qubit gates follow the same pattern. Throughout our formal analysis of the 1-qubit gates, we extract their success probabilities and verify their correct outputs.

### 4.1.1 Hadamard Gate

The Hadamard gate [36] is an 1-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 the quantum states of a circuit or to add random information to a quantum circuit. The authors in [41] implemented the Shor's algorithm for factorization of the number 15 using six Hadamard gates in a photonics chip by employing the quantum optics single photon technology. In this section, we present the formalization of the Hadamard gate in HOL Light as an example of a single qubit gate that can be constructed by using a beam splitter (BS)

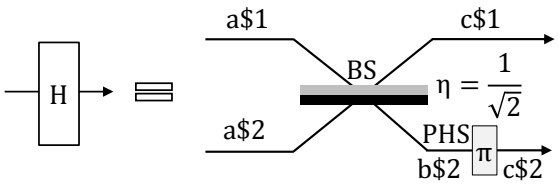


Figure 4.2: Schematics of Hadamard Gate

and a phase shifter (PSH) together. The dual-rail representation is used to describe the qubit. The gate circuit is shown in Figure 4.2 with a beam splitter ($\eta = \frac{1}{\sqrt{2}}$) and a phase shifter of angle $\theta = \pi$ ($\varphi = 0$).

The formal definition of the gate structure in HOL is as follows:

**Definition 4.1** (Hadamard Gate)**.**

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

where LH and LV are employed to describe the representation of qubits using the photon vertical or horizontal polarization. Here, 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 gate input/output ports $(a, c)$, and the tensor product operator *tens*. 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 4.1** (Hadamard Input: $|1\rangle_L \equiv |0,1\rangle_a$)**.**

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{HADAMARD\_GATE}(\texttt{a}, \texttt{c}, \texttt{ten}, \texttt{LH}, \texttt{LV})\ \texttt{in}$

$\texttt{let}\ |0,1\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{a\$2}}\ \texttt{else}\ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$

$\texttt{let}\ |1,0\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ |1\rangle_{\texttt{c\$1}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$

$\texttt{let}\ |0,1\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ |1\rangle_{\texttt{c\$2}}\ \texttt{else}\ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$

$\texttt{constraints} \Longrightarrow |0,1\rangle_{\texttt{a}} = \frac{1}{\sqrt{2}} \% |1,0\rangle_{\texttt{c}} - \frac{1}{\sqrt{2}} \% |0,1\rangle_{\texttt{c}}$

**Theorem 4.2** (Hadamard Input: $|0\rangle_L \equiv |1,0\rangle_a$).

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{HADAMARD\_GATE}(\texttt{a},\texttt{c},\texttt{ten},\texttt{LH},\texttt{LV})\ \texttt{in}$

$\texttt{let}\ \ |1,0\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{a\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$

$\texttt{let}\ \ |1,0\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{c\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$

$\texttt{let}\ \ |0,1\rangle_{\texttt{c}} = \texttt{tensor 2}\ (\lambda \texttt{i.}\ \texttt{if i} = 2\ \texttt{then}\ \ |1\rangle_{\texttt{c\$2}}\ \texttt{else}\ \ |0\rangle_{\texttt{c\$2}})\ \texttt{in}$

$\texttt{constraints} \Longrightarrow |1,0\rangle_{\texttt{a}} = \frac{1}{\sqrt{2}} \% |1,0\rangle_{\texttt{c}} + \frac{1}{\sqrt{2}} \% |0,1\rangle_{\texttt{c}}$

Notice that we did not use the projection because we do not employ ancilla, therefore, there will be no detection required. As shown in Figure 4.2 the optical implementation of the 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 an input/output behavioral description for the Hadamard gate $\texttt{Hadamard\_In\_Outputs}(\texttt{a0},\texttt{b0},\texttt{a},\texttt{c},\texttt{LH},\texttt{LV})$ presented in Table 4.1.

Table 4.1: Hadamard Gate Behavioral Description

| **Hadamard_In_Outputs (a0,b0,a,c,LH,LV)** | |
|---|---|
| $\texttt{tensor 1}\ (\lambda \texttt{i. LH a0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i. if i} = 1\ \texttt{then}\ \ \|1\rangle_{\texttt{a\$1}}\ \texttt{else}\ \ \|0\rangle_{\texttt{a\$2}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i. LV a0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i. if i} = 2\ \texttt{then}\ \ \|1\rangle_{\texttt{a\$2}}\ \texttt{else}\ \ \|0\rangle_{\texttt{a\$1}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i. vac a0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i.}\ \ \|0\rangle_{\texttt{a\$1}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i. LH b0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i. if i} = 1\ \texttt{then}\ \ \|1\rangle_{\texttt{c\$1}}\ \texttt{else}\ \ \|0\rangle_{\texttt{c\$2}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i. LV b0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i. if i} = 2\ \texttt{then}\ \ \|1\rangle_{\texttt{c\$2}}\ \texttt{else}\ \ \|0\rangle_{\texttt{c\$1}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i. vac b0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i.}\ \ \|0\rangle_{\texttt{c\$2}})$ |

Generally, an optical photon is horizontally (resp. vertically) polarized if in the first optical mode there is one fock state and in the second mode we have vacuum $|10\rangle$ (resp. vacuum in the first optical mode and one fock state in the second mode $|01\rangle$). The first three rows (resp. last three rows) of Table 4.1 represent the relation between the behavioral description and optical modes representation for the inputs (resp. outputs). The first column of the table contains the behavioral description of the inputs and the second column contains the optical modes representation of the inputs and outputs. A special case where there is no photon in neither the horizontal

and the vertical polarization modes, we named it `vac`. For example in the third and last rows of Table 4.1, we have $\texttt{vac a0} = |\texttt{00}\rangle_{\texttt{a}}$ and $\texttt{vac b0} = |\texttt{00}\rangle_{\texttt{b}}$.

### 4.1.2 Flip Gate

A design of the flip gate based on coherent state source is proposed in [43]. However, to the best of our knowledge, there is no optical design 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 4.3, is composed of two beam splitters and a phase shifter.

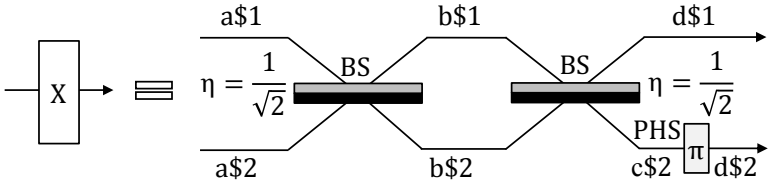


Figure 4.3: Schematics of Flip Gate

We formally define the flip gate structure in HOL as follows:

**Definition 4.2** (Flip Gate)**.**

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

`FLIP_GATE` takes as parameters all gate input $a$ and output ports $d$, the tensor operator, and the parameter $i$ to specify the order of the signal in the main tensor on which the gate is applied. 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 4.3** (Flip Input: $|1\rangle_L \equiv |0,1\rangle_a$)**.**

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{FLIP\_GATE}(\texttt{a},\texttt{d},\texttt{ten},\texttt{LH},\texttt{LV})\ \texttt{in}$

$\texttt{let}\ \ |0,1\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 2\ \texttt{then}\ \ |1\rangle_{\texttt{a\$2}}\ \texttt{else}\ \ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$

$\texttt{let}\ \ |1,0\rangle_{\texttt{d}} = \texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{d\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{d\$2}})\ \texttt{in}$

$\texttt{constraints} \Longrightarrow |0,1\rangle_{\texttt{a}} = |1,0\rangle_{\texttt{d}}$

**Theorem 4.4** (Flip Input: $|0\rangle_L \equiv |1,0\rangle_a$)**.**

$\texttt{let constraints} = \texttt{is\_tensor ten} \wedge \texttt{FLIP\_GATE}(\texttt{a},\texttt{d},\texttt{ten},\texttt{LH},\texttt{LV})\ \texttt{in}$

$\texttt{let}\ \ |1,0\rangle_{\texttt{a}} = \texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 1\ \texttt{then}\ \ |1\rangle_{\texttt{a\$1}}\ \texttt{else}\ \ |0\rangle_{\texttt{a\$1}})\ \texttt{in}$

$\texttt{let}\ \ |0,1\rangle_{\texttt{d}} = \texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 2\ \texttt{then}\ \ |1\rangle_{\texttt{d\$2}}\ \texttt{else}\ \ |0\rangle_{\texttt{d\$2}})\ \texttt{in}$

$\texttt{constraints} \Longrightarrow |1,0\rangle_{\texttt{a}} = |0,1\rangle_{\texttt{d}}$

Similar to the Hadamard gate, we developed an input/output behavioral description for flip gate $\texttt{Flip\_In\_Outputs}(\texttt{a0},\texttt{b0},\texttt{a},\texttt{d},\texttt{LH},\texttt{LV})$, which is presented in Table 4.2. In Table 4.2, the input qubit $a0$ is described using the photon polarization on the two optical modes $a\$1$ and $a\$2$ and the output qubit $b0$ is described using the photon polarization on the two optical modes $c\$1$ and $c\$2$. The first two rows (resp. last two rows) of Table 4.2 represent the relation between the behavioral description and optical modes representation for the inputs (resp. outputs). The first column of the table contains the behavioral description and the second column contains the optical modes representation of the inputs and outputs.

Table 4.2: Flip Gate Behavioral Description

| **Flip_In_Outputs (a0,b0,a,c1,LH,LV)** | |
|---|---|
| $\texttt{tensor 1}\ (\lambda \texttt{i}.\ \texttt{LH a0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 1\ \texttt{then}\ \ \|1\rangle_{\texttt{a\$1}}\ \texttt{else}\ \ \|0\rangle_{\texttt{a\$2}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i}.\ \texttt{LV a0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 2\ \texttt{then}\ \ \|1\rangle_{\texttt{a\$2}}\ \texttt{else}\ \ \|0\rangle_{\texttt{a\$1}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i}.\ \texttt{LH b0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 1\ \texttt{then}\ \ \|1\rangle_{\texttt{c\$1}}\ \texttt{else}\ \ \|0\rangle_{\texttt{c\$2}})$ |
| $\texttt{tensor 1}\ (\lambda \texttt{i}.\ \texttt{LV b0})$ | $\texttt{tensor 2}\ (\lambda \texttt{i}.\ \texttt{if i} = 2\ \texttt{then}\ \ \|1\rangle_{\texttt{c\$2}}\ \texttt{else}\ \ \|0\rangle_{\texttt{c\$1}})$ |

So far, we have presented two 1-qubit gates without using the ancilla resources which are extra qubits that have a secondary role in a quantum computation and are

used for detecting the correct output [26]. However, many optical quantum circuits implementations are using quantum detectors to measure the states of the ancillas after they leave the quantum circuit. In the approach proposed by [25], the quantum circuit is considered properly implemented only when the detector produces a positive outcome (expected outcome), i.e., the circuit is nondeterministic (sometimes we say probabilistic). In the next section, we describe a nondeterministic gate namely the non-linear sign gate.

### 4.1.3 Non-Linear Sign Gate

In [25], the authors formed the universal controlled-phase gate using the nondeterministic non-linear sign (NS) gate (Figure 4.4), which is a 1-qubit 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 4.4. 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 $d\$1$.

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

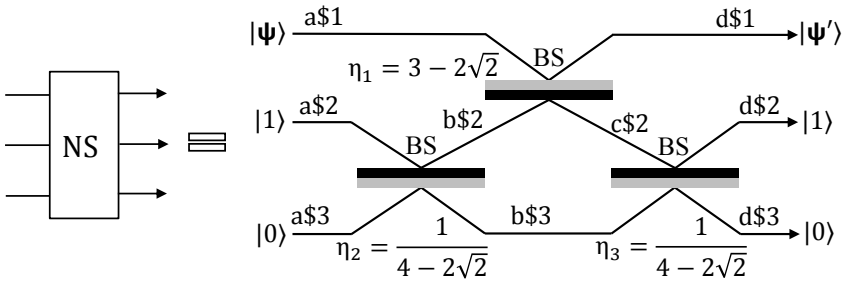


Figure 4.4: Schematics of NS Gate

**Definition 4.3** (NS Gate)**.**

$\vdash$ NS_GATE(a, b, c, d, ten) $\iff$ is_beam_splitter($\sqrt{2\sqrt{2}-2}$, $\sqrt{3-2\sqrt{2}}$, $-\sqrt{3-2\sqrt{2}}$, $\sqrt{2\sqrt{2}-2}$, ten, b$2, 2, a$1, 1, d$1, 1, c$2, 2) $\wedge$ is_beam_splitter($\frac{1}{\sqrt{4-2\sqrt{2}}}$, $\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}$, $\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}$, $-\frac{1}{\sqrt{4-2\sqrt{2}}}$, ten, a$2, 2, a$3, 3, b$2, 2, b$3, 3) $\wedge$ is_beam_splitter($\frac{1}{\sqrt{4-2\sqrt{2}}}$, $\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}$, $\frac{\sqrt{3-2\sqrt{2}}}{\sqrt{4-2\sqrt{2}}}$, $-\frac{1}{\sqrt{4-2\sqrt{2}}}$, ten, c$2, 2, b$3, 3, d$2, 2, 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 success probability by projecting all NS gate outputs on the expected output. We prove that for an input $|2,1,0\rangle_a$ the projection of NS gate output on the states $|0,1,0\rangle_d$ and $|1,1,0\rangle_d$ gives zero, on the contrary the projection on the state $|2,1,0\rangle_d$ gives $-\frac{1}{2}$ (success probability $(\frac{1}{2})^2 = \frac{1}{4}$). We repeat the same procedure for the two other possible inputs (i.e., $|0,1,0\rangle_a$ and $|1,1,0\rangle_a$).

**Theorem 4.5** (NS Input: $|2\rangle$, Projection: $|2\rangle$)**.**

$\vdash$ let constraint = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$
NS_GATE(a, b, c, d, ten) in
let $|2,1,0\rangle_{\mathtt{d}}$ = tensor 3 ($\lambda$i. if i = 1 then $|2\rangle_{\mathtt{d\$1}}$ elseif i = 2 then $|1\rangle_{\mathtt{d\$2}}$ else $|0\rangle_{\mathtt{d\$3}}$) in
let $|2,1,0\rangle_{\mathtt{a}}$ = tensor 3 ($\lambda$i. if i = 1 then $|2\rangle_{\mathtt{a\$1}}$ elseif i = 2 then $|1\rangle_{\mathtt{a\$2}}$

$$\texttt{else } |0\rangle_{\texttt{a\$3}}) \texttt{ in}$$
$$\texttt{constraint} \Longrightarrow (\texttt{m\_proj } |2,1,0\rangle_{\texttt{d}}) \ |2,1,0\rangle_{\texttt{a}} = -\tfrac{1}{2}\% \, |2,1,0\rangle_{\texttt{d}}$$

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

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

**Theorem 4.7** (NS Input: $|2\rangle$, Projection: $|0\rangle$)**.**

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

where, $\texttt{m\_mode\_pro } (|1,1,0\rangle_{\texttt{d}})$ is the projection on the state $|1,1,0\rangle_d$.

## 4.2 Formalization of 2-Qubit Gates

In this section, we focus on the 2-qubit quantum gates that can be constructed using the three 1-qubit gates described in the previous section. In particular, we formally model and verify the controlled phase (CZ), controlled not (CNOT), and SWAP gates.

The CZ and CNOT are considered as two universal gates in quantum computing and many quantum computing circuits are based on these two gates. On the other hand, the SWAP gate has a crucial role in interchanging the qubits between each other inside a quantum circuit.

### 4.2.1 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 words, if the possible input is $|\phi\rangle_{input} = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, then the output is $|\phi\rangle_{output} = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle - \delta|11\rangle$. The CZ gate is constructed with the use of two NS gates and two beam splitters, as shown in Figure 4.5. The probability of measuring

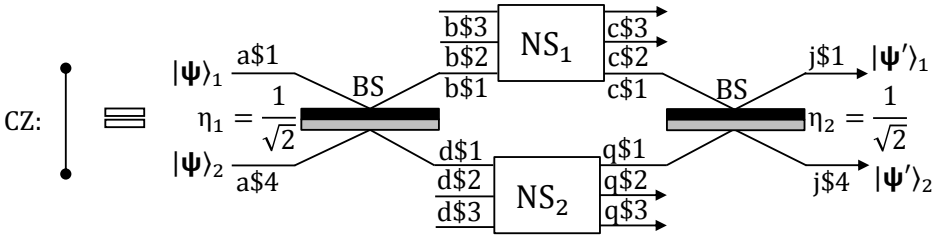


Figure 4.5: Schematics of CZ Gate

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

**Definition 4.4** (CZ Gate)**.**

$\vdash$ IS_CZ_GATE (a, b, c, j, ten, LH, LV, m_proj) $\iff$ ($\forall$ d q k l m p.

NS_GATE(d, m, p, q, ten) $\wedge$ b$4 = d$1 $\wedge$ b$5 = d$2 $\wedge$ b$6 = d$3 $\wedge$

NS_GATE(b, l, k, c, ten) $\wedge$ q$1 = c$4 $\wedge$ is_sm a$3 $\wedge$ is_sm a$2 $\wedge$

q$3 = c$6 $\wedge$ is_beam_splitter($\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$, ten, a$1, 1, a$4, 4, b$1, 1, b$4, 4) $\wedge$

q$2 = c$5 $\wedge$ is_beam_splitter($\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$, ten, c$1, 1, c$4, 4, j$1, 1, j$4, 4))

Notice that we renamed the input and output ports for the second NS gate in order to match the order of the modes in the definition of the gate, 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 [5].

**Theorem 4.8** (CZ Input: $|1,1\rangle$)**.**

$$\vdash \texttt{let constraints} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge$$
$$\texttt{IS\_CZ\_GATE (a, b, c, j, ten, LH, LV, m\_proj) in}$$
$$\texttt{let } |2,1,0,0,1,0,0,0\rangle_{\texttt{cq}} = \texttt{tensor 8 } (\lambda \texttt{i. if i} = 1 \texttt{ then } |2\rangle_{\texttt{c\$1}} \texttt{ elseif i} = 2$$
$$\texttt{then } |1\rangle_{\texttt{c\$2}} \texttt{ elseif i} = 5 \texttt{ then } |1\rangle_{\texttt{q\$2}} \texttt{ else } |0\rangle_{\texttt{c\$3}}) \texttt{ in}$$
$$\texttt{let } |0,1,0,2,1,0,0,0\rangle_{\texttt{cq}} = \texttt{tensor 8 } (\lambda \texttt{i. if i} = 2 \texttt{ then } |1\rangle_{\texttt{c\$2}} \texttt{ elseif i} = 4$$
$$\texttt{then } |2\rangle_{\texttt{q\$1}} \texttt{ elseif i} = 5 \texttt{ then } |1\rangle_{\texttt{q\$2}} \texttt{ else } |0\rangle_{\texttt{c\$3}}) \texttt{ in}$$
$$\texttt{let } |1,1,0,1,1,0,0,0\rangle_{\texttt{cq}} = \texttt{tensor 8 } (\lambda \texttt{i. if i} = 1 \texttt{ then } |1\rangle_{\texttt{c\$1}} \texttt{ elseif i} = 2$$
$$\texttt{then } |1\rangle_{\texttt{c\$2}} \texttt{ elseif i} = 4 \texttt{ then } |1\rangle_{\texttt{q\$1}} \texttt{ elseif i} = 5 \texttt{ then } |1\rangle_{\texttt{q\$2}} \texttt{ else } |0\rangle_{\texttt{c\$3}}) \texttt{ in}$$
$$\texttt{let } |1,1,0,1,1,0,0,0\rangle_{\texttt{ab}} = \texttt{tensor 8 } (\lambda \texttt{i. if i} = 1 \texttt{ then } |1\rangle_{\texttt{a\$1}} \texttt{ elseif i} = 2$$
$$\texttt{then } |1\rangle_{\texttt{b\$2}} \texttt{ elseif i} = 4 \texttt{ then } |1\rangle_{\texttt{a\$4}} \texttt{ elseif i} = 5 \texttt{ then } |1\rangle_{\texttt{b\$5}} \texttt{ else } |0\rangle_{\texttt{b\$3}}) \texttt{ in}$$
$$\texttt{let } |1,1,0,1,1,0,0,0\rangle_{\texttt{cj}} = \texttt{tensor 8 } (\lambda \texttt{i. if i} = 1 \texttt{ then } |1\rangle_{\texttt{j\$1}} \texttt{ elseif i} = 2$$
$$\texttt{then } |1\rangle_{\texttt{c\$2}} \texttt{ elseif i} = 4 \texttt{ then } |1\rangle_{\texttt{j\$4}} \texttt{ elseif i} = 5 \texttt{ then } |1\rangle_{\texttt{c\$5}} \texttt{ else } |0\rangle_{\texttt{c\$3}}) \texttt{ in}$$
$$\texttt{constraints} \Longrightarrow (\texttt{m\_proj } |2,1,0,1,0,0,0,0\rangle_{\texttt{cq}} + \texttt{m\_proj } |0,1,0,1,2,0,0,0\rangle_{\texttt{cq}} +$$
$$\texttt{m\_proj } |1,1,0,1,1,0,0,0\rangle_{\texttt{cq}}) \ (|1,1,0,1,1,0,0,0\rangle_{\texttt{ab}}) = -\ \tfrac{1}{4}\ \%\ |1,1,0,1,1,0,0,0\rangle_{\texttt{cj}}$$

Notice that the output of the CZ gate has been projected over three different states. This is because of the fact that we have two photons at the input 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 4.9** (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 ($\lambda$i. 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 ($\lambda$i. 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 ($\lambda$i. 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 ($\lambda$i. 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 $\Longrightarrow$ (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 subdivide the tensor product projection to the tensor of two tensor product projections each fed to an NS gate.

As shown in Figure 4.5, the CZ gate has 8 input modes, however, the CZ gate is a 2-qubit 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 (behavioral level), we developed CZ_INPUTS (a, b, c, LH, LV, m_proj) and CZ_OUTPUTS (a, c, j, LH, LV)[2], details are presented in Appendix A. 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.

[2]A behavioral description for CZ gate, where instead of eight inputs and eight outputs, we have two inputs and two outputs.

### 4.2.2 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 4.6, however, it can also be implemented using five beam splitters as shown in [44] and verified in [32].

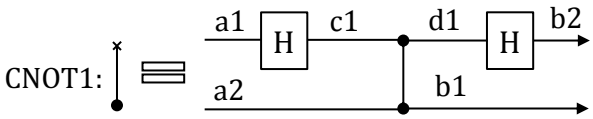


Figure 4.6: Schematics of CNOT Gate

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 the CNOT, where for the first version the target qubit feds to the first input and for the second version it is fed to the second input. We provide here the HOL definition of the first version of CNOT gate structure and the second one (we name them `CNOT1` and `CNOT2`, respectively). We formally define CNOT1 structure as follows (CNOT2 is given in Appendix B):

**Definition 4.5** (CNOT Gate)**.**

$\vdash$ `CNOT1_GATE`(`a1`, `a2`, `b1`, `b2`, `ten`, `LH`, `LV`, `m_proj`) $\Longleftrightarrow$ ($\forall$`c1 d1`.
`HADAMARD_GATE`(`a1`, `c1`, `ten`, `LH`, `LV`) $\wedge$ `HADAMARD_GATE`(`d1`, `b2`, `ten`, `LH`, `LV`)
$\wedge$ `CZ_GATE`(`c1`, `a2`, `d1`, `b1`, `ten`, `LH`, `LV`, `m_proj`))

Here, the Hadamard gate is applied on the first input, which is the target qubit. We formally verified that this definition maintains the truth table of the CNOT gate. As

an example we provide the result of applying the CNOT on the input $|0,1\rangle$:

**Theorem 4.10** (CNOT Gate Input: $|0,1\rangle$)**.**

$\vdash$ `let constraints` $=$ `is_tensor_proj m_proj` $\wedge$ `is_tensor ten` $\wedge$
`CNOT1_GATE (a1, a2, b1, b2, ten, LH, LV, m_proj)` $\wedge$ `8` $\leq$ `dimindex (:N) in`
`let` $|0,1\rangle_{\mathtt{a}}$ $=$ `tensor 2 (`$\lambda$`i. if i = 1 then LH a1 else LV a2) in`
`let` $|1,1\rangle_{\mathtt{b}}$ $=$ `tensor 2 (`$\lambda$`i. if i = 1 then LV b1 else LV b2) in`
`constraints` $\Longrightarrow$ $|0,1\rangle_{\mathtt{a}} = \frac{1}{4}$ % $|1,1\rangle_{\mathtt{b}}$

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

**Theorem 4.11** (CNOT Gate Input: $|x,y\rangle$)**.**

$\vdash$ `let constraints` $=$ `is_tensor_proj m_proj` $\wedge$ `is_tensor ten` $\wedge$
`CNOT1_GATE (a1, a2, b1, b2, ten, LH, LV, m_proj)` $\wedge$ `8` $\leq$ `dimindex (:N) 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 (`$\lambda$`i. if i = 1`
`then (x1`%`LV a1` $+$ `x2`%`LH a1) else (y1`%`LV a2` $+$ `y2`%`LH a2)) in`
`let` $\mathtt{x1y1}\,|01\rangle_{\mathtt{b}} + \mathtt{x2y1}\,|11\rangle_{\mathtt{b}}$ $=$ `tensor 2 (`$\lambda$`i. if i = 1 then`
`(x1`%`LH b1` $+$ `x2`%`LV b1) else y1`%`LV b2) in`
`let` $\mathtt{x1y2}\,|10\rangle_{\mathtt{b}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{b}}$ $=$ `tensor 2 (`$\lambda$`i. if i = 1 then`
`(x1`%`LV b1` $+$ `x2`%`LH b1) else y2`%`LH b2) in`
`constraints` $\Longrightarrow$ $\mathtt{x1y1}\,|11\rangle_{\mathtt{a}} + \mathtt{x1y2}\,|10\rangle_{\mathtt{a}} + \mathtt{x2y1}\,|01\rangle_{\mathtt{a}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{a}} =$
$\frac{1}{4}$ % $(\mathtt{x1y1}\,|01\rangle_{\mathtt{b}} + \mathtt{x2y1}\,|11\rangle_{\mathtt{b}} + \mathtt{x1y2}\,|10\rangle_{\mathtt{b}} + \mathtt{x2y2}\,|00\rangle_{\mathtt{b}})$

Here, $x1y1$, $x1y2$, $x2y1$, and $x2y2$ represent the probabilities that the state $|x,y\rangle$ is in the basics quantum states $|11\rangle$, $|10\rangle$, $|01\rangle$, and $|00\rangle$, respectively. Notice that the CNOT gate has the same success probability as the CZ gate.

### 4.2.3 SWAP Gate

The SWAP gate is a two qubits gate which swaps the states of two input qubits. It has a crucial role in the design of quantum circuits where the SWAP gate is used to swap the qubits between each other in order to fulfil the requirement that computations should only be performed between adjacent qubits [34]. Also in [27], 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 [12], it was shown that the SWAP gate plays an important role in the implementation of Shor's algorithm [49] based on linear nearest neighbor architecture, where the SWAP gate rearranges the qubits. The physical implementation of the SWAP gate requires three CNOT gates, as shown in Figure 4.7. In the structure of the SWAP gate, we can note the

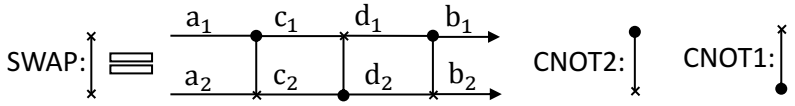


Figure 4.7: Schematics of SWAP Gate

usage of the two versions of the CNOT gate in the implementation of SWAP: the first (CNOT1) where the target qubit is represented by the first input and the second (CNOT2) where the target qubit is represented by the second input, more details can be found in [6]. We formally define the structure of the SWAP gate in HOL as follows:

**Definition 4.6** (SWAP Gate Structure)**.**

$\vdash$ SWAP_GATE(a1, a2, b1, b2, ten, LH, LV, m_proj) $\Longleftrightarrow$

($\forall$ c1 c2 d1 d2. CNOT2_GATE(d1, d2, b1, b2, ten, LH, LV, m_proj) $\wedge$

CNOT2_GATE(a1, a2, c1, c2, ten, LH, LV, m_proj) $\wedge$

CNOT1_GATE(c1, c2, d1, d2, ten, LH, LV, m_proj))

We now use this definition to prove the general case for an input to the SWAP gate in the form $|a1, a2\rangle = |x1\%1 + x2\%0, y1\%1 + y2\%0\rangle$ in HOL as follows:

**Theorem 4.12** (SWAP Gate Input: $|x, y\rangle$)**.**

$$\vdash \texttt{let constraints} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge$$
$$\texttt{8} \leq \texttt{dimindex (:N)} \wedge \texttt{SWAP\_GATE(a1, a2, b1, b2, ten, LH, LV, m\_proj) in}$$
$$\texttt{let } |\texttt{x1\%1} + \texttt{x2\%0}, \texttt{y1\%1} + \texttt{y2\%0}\rangle_{\texttt{a}} = \texttt{tensor 2 } (\lambda \texttt{i. if i} = \texttt{1 then}$$
$$\texttt{(x1\%LV a1} + \texttt{x2\%LH a1) else (y1\%LV a2} + \texttt{y2\%LH a2)) in}$$
$$\texttt{let } |\texttt{y1\%1} + \texttt{y2\%0}, \texttt{x1\%1} + \texttt{x2\%0}\rangle_{\texttt{b}} = \texttt{tensor 2 } (\lambda \texttt{i. if i} = \texttt{1 then}$$
$$\texttt{(y1\%LH b1} + \texttt{y2\%LV b1) else (x1\%LH b2} + \texttt{x2\%LV b2)) in}$$
$$\texttt{constraints} \Longrightarrow |\texttt{x1\%1} + \texttt{x2\%0}, \texttt{y1\%1} + \texttt{y2\%0}\rangle_{\texttt{a}} =$$
$$\tfrac{1}{64} \ \% \ |\texttt{y1\%1} + \texttt{y2\%0}, \texttt{x1\%1} + \texttt{x2\%0}\rangle_{\texttt{b}}$$

Here, the two assumptions `is_tensor_proj m_proj` and `is_tensor ten` are to maintain that the two operators `m_proj` and `ten` are indeed the tensor product projection and the tensor product operator, respectively. The assumption `8` $\leq$ `dimindex (:N)` is to make sure that the dimension of the tensor product is more than the size of the quantum circuit under consideration. We can notice that there is a scalar multiplication by the output state, $\frac{1}{64}$, which represents the success rate of the gate (i.e., the probability at which the gate produces the correct output).

## 4.3 Formalization of 3-Qubit Gates

In the previous two sections, we showed the formalization of 1-qubit and 2-qubit gates. Both sets of gates are an indispensable part for building quantum circuits. In this section, we present the formalization of another set of gates, namely 3-qubit gates. In particular, we formally model and verify two most prominent 3-qubit gates, namely Toffoli and Fredkin gates. For the former, we were required to formalize another 3-qubit gate which is the Toffoli Sign gate. Also, it is important to notice that the Fredkin gate design is based on the Toffoli gate design.

### 4.3.1 Toffoli Sign Gate

The Toffoli Sign (TS) gate is a 3-qubit reversible gate that applies a sign shift on one of the state components, and the identity to other inputs. The main benefit of the TS gate is to construct the Toffoli gate. In the optical implementation of TS, `vac t$1` refers to the vacuum state, i.e., where both polarization modes are unoccupied, in the single mode `t$1`. Thus, we will introduce a third level of representation of a *qutrit* (i.e., a superposition of three orthogonal quantum states [45]).

The realization of TS is based on using two qubits `t$2`, `t1$2` (i.e., $0_L = |LH\rangle$ and $1_L = |LV\rangle$) and a *qutrit* t (i.e., $1_L = |vac, LV\rangle_{t\$1t\$3}$ and $0_L = |LH, vac\rangle_{t\$1t\$3}$). The TS gate structure is shown in Figure 4.8. It is composed of two CNOT gates and a CZ gate. The two CNOT gates operate as normal at the qubit levels and implement the identity if the target is at the *qutrit* level ($|vac, LV\rangle$).

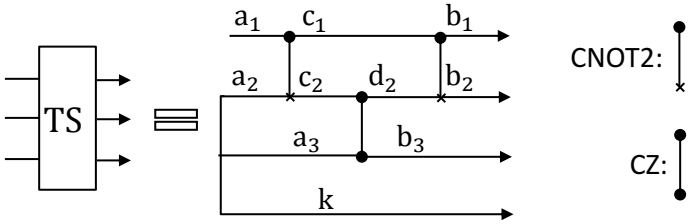


Figure 4.8: Schematics of TS Gate

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

**Definition 4.7** (TS Gate Structure)**.**

$\vdash$ `TS_GATE`(`a1`, `a2`, `a3`, `b1`, `b2`, `b3`, `ten`, `LH`, `LV`, `m_proj`) $\Longleftrightarrow$

($\forall$ `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`))

where `CZ_GATE` describes the structure of a CZ gate. 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 the

TS transformation on two inputs forms: $|101\rangle$ and $|111\rangle$, which in a 4-qubit format are given as $|1, vac, 1, 1\rangle$ and $|1, 1, 1, vac\rangle$, respectively. Following is the result of the TS transformation over the input $|0, 1, 1\rangle$:

**Theorem 4.13** (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`$(\mathtt{a1}, \mathtt{a2}, \mathtt{a3}, \mathtt{b1}, \mathtt{b2}, \mathtt{b3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj})$ `in`
`let` $|0, 1, 1\rangle_{\mathtt{a}} =$ `tensor 3` $(\lambda$`i. if i` $= 1$ `then LH a1 elseif i` $= 2$ `then LV a2`
`else LV a3) in`
`let` $|0, 1, 1\rangle_{\mathtt{b}} =$ `tensor 3` $(\lambda$`i. if i` $= 1$ `then LH a1 elseif i` $= 2$ `then LV a2`
`else LV a3) in`
`constraints` $\Longrightarrow |0, 1, 1\rangle_{\mathtt{a}} = -\frac{1}{64}$ % $|0, 1, 1\rangle_{\mathtt{b}}$

Notice here the sign shift for the output state $|0, 1, 1\rangle$, which is also multiplied by a scalar value that represents the gate success probability, $\frac{1}{64}$. However, for the input $|1, 1, 1\rangle$ there will be no sign shift:

**Theorem 4.14** (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`$(\mathtt{a1}, \mathtt{a2}, \mathtt{a3}, \mathtt{b1}, \mathtt{b2}, \mathtt{b3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj})$ `in`
`let` $|1, 1, 1\rangle_{\mathtt{a}} =$ `tensor 3` $(\lambda$`i. if i` $= 1$ `then LV a1 elseif i` $= 2$ `then LV a2`
`else LV a3) in`
`let` $|1, 1, 1\rangle_{\mathtt{b}} =$ `tensor 3` $(\lambda$`i. if i` $= 1$ `then LV b1 elseif i` $= 2$ `then LV b2`
`else LV b3) in`
`constraints` $\Longrightarrow |1, 1, 1\rangle_{\mathtt{a}} = \frac{1}{64}$ % $|1, 1, 1\rangle_{\mathtt{b}}$

Note that in the Figure 4.8, the TS gate has four input modes, however, TS is a 3-qubit gate, where the first logical qubit is represented by two 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 details of the qubit representation, we developed an input/output behavioral description of the gate which can be found in

Appendix C. Now, after formally modeling and verifying the Toffoli Sign gate, we are ready to tackle the formalization of the Toffoli gate.

### 4.3.2 Toffoli Gate

The Toffoli gate is a three-qubit reversible 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 [2].

The simplest known design of the Toffoli gate when restricted to operating on qubits at the behavioral level is a circuit that requires five 2-qubit gates. However, it was shown in [45] that it is possible to construct a Toffoli gate using the Toffoli sign flip and Hadamard gates, the Toffoli gate is shown in Figure 4.9. In following, we formally formalize this optimized design of the Toffoli gate.

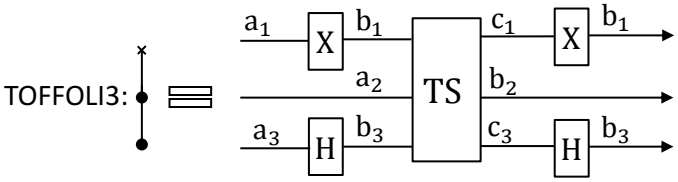


Figure 4.9: Schematics of Toffoli Gate

Similar to the CNOT gate, the Toffoli gate can be used in two forms: 1) the first qubit is the target; and 2) the third qubit is the target (we call them `TOFFOLI1` and `TOFFOLI3`, respectively). Therefore, we have formally defined these two kinds of Toffoli gate in HOL. More details about the first kind of Toffoli gate can be found at [5]. We provide here the formal definition of the second type structure of Toffoli gate (`TOFFOLI3`) as follows:

**Definition 4.8** (Toffoli Gate Structure)**.**

$\vdash$ TOFFOLI3_GATE(a1, a2, a3, b1, b2, b3, ten, LH, LV, m_proj) $\iff$ ($\forall$ 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))

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

**Theorem 4.15** (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 ($\lambda$i. if i = 1 then LV a1 elif i = 2 then LV a2 else LV a3) in
let $|011\rangle_{\mathtt{b}}$ = tensor 3 ($\lambda$i. if i = 1 then LH b1 elif i = 2 then LV b2 else LV b3) in
constraints $\Longrightarrow$ $|111\rangle_{\mathtt{a}} = \frac{1}{64}\% \, |011\rangle_{\mathtt{b}}$

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

**Theorem 4.16** (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 ($\lambda$i. if i = 1 then LH a1 elif i = 2 then LV a2 else LV a3) in
let $|111\rangle_{\mathtt{b}}$ = tensor 3 ($\lambda$i. if i = 1 then LV b1 elif i = 2 then LV b2 else LV b3) in
constraints $\Longrightarrow$ $|011\rangle_{\mathtt{a}} = \frac{1}{64}\% \, |111\rangle_{\mathtt{b}}$

Notice that the success probability of the Toffoli gate is the same as the one of the Toffoli Sign: $\frac{1}{64}$. Note that if the Toffoli gate was constructed using five 2-qubit gates, the success probability will be around $\frac{1}{16}$ of $\frac{1}{64}$.

### 4.3.3 Fredkin Gate

The Fredkin gate or the controlled-2x2 reversible quantum switch gate (or controlled SWAP gate) is a 3-qubit gate [35]. One of the qubits is designated as the control qubit and is left unchanged by the gate, and the other 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 plays an important role in quantum computing, error-correcting quantum computations, and information processing [36]. Moreover, the controlled SWAP gate is a universal gate for classical (reversible) computing which means that any logical or arithmetic operation can be constructed entirely out of Fredkin gates [35]. As there is two versions of the Toffoli gate, it results of having two versions of the Fredkin gate: 1) the control qubit is the first qubit (FREDKIN1); and 2) the control qubit is the third qubit (FREDKIN3). The Fredkin gate circuit shown in Figure 4.10 is composed of two CNOT gates and one Toffoli gate. Other gates such as AND, OR, and XOR gates, flip-flops, etc. can also be constructed using the Fredkin gate.

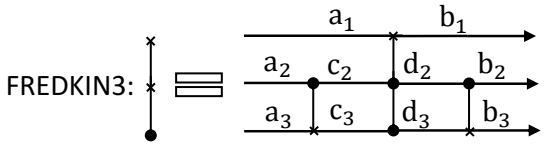


Figure 4.10: Schematics of Fredkin Gate

We formally model the structure of the second version of the Fredkin gate (`FREDKIN3` in HOL as follows:

**Definition 4.9** (Fredkin Gate Structure)**.**

$\vdash$ `FREDKIN3_GATE` $(\mathtt{a1}, \mathtt{a2}, \mathtt{a3}, \mathtt{b1}, \mathtt{b2}, \mathtt{b3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \Longleftrightarrow (\forall$ `c2 c3 d2 d3`.

$\mathtt{CNOT1\_GATE}(\mathtt{a2}, \mathtt{a3}, \mathtt{c2}, \mathtt{c3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \wedge$

$\mathtt{TOFFOLI3\_GATE}(\mathtt{a1}, \mathtt{c2}, \mathtt{c3}, \mathtt{b1}, \mathtt{d2}, \mathtt{d3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \wedge$

$\mathtt{CNOT1\_GATE}(\mathtt{d2}, \mathtt{d3}, \mathtt{b2}, \mathtt{b3}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}))$

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 4.17** (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 $|011\rangle_a$ = tensor 3 ($\lambda$i. if i = 1 then LH a1 elif i = 2 then LV a2
else LV a3) in
let $|101\rangle_b$ = tensor 3 ($\lambda$i. if i = 1 then LV b1 elif i = 2 then LH b2
else LV b3) in
constraints $\Longrightarrow |011\rangle_a = \frac{1}{1024}\% |101\rangle_b$

Using the bilinearity property of tensor product, we also verify the result of applying the Fredkin gate on the input in the general form $|zxy\rangle$:

**Theorem 4.18** (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_proj) in
let $|zxy\rangle_a$ = tensor 3 ($\lambda$i. if i = 1 then (z1%LH a1 + z2%LV a1) else
if i = 2 then (x1%LH a2 + x2%LV a2) else (y1%LH a3 + y2%LV a3)) in
let $|0xy\rangle_b$ = tensor 3 ($\lambda$i. 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 $|1xy\rangle_b$ = tensor 3 ($\lambda$i. 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 $\Longrightarrow |zxy\rangle_a = \frac{1}{1024}\%(z1\% |0xy\rangle_b + z2\% |1xy\rangle_b$

Here, the Fredkin gate inputs $z$ represents the control input and $x$ and $y$ are the target inputs. We can notice that when $z = |0\rangle$ the two target inputs do not change and when $z = |1\rangle$ the two target inputs switch between each other. It is important to notice that the success probability of the Fredkin gate is $\frac{1}{1024}$ and it is very small. By

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

## 4.4 Summary

In this chapter, we have covered the formal modeling, design and verification of a complete library of quantum gates which can be used in the design and analysis of a variety of quantum algorithms and circuits. We have demonstrated the usefulness of the developed mathematical foundation in carrying the analysis of the underlying gates library. In particular, we have used the tensor product projection to obtain the correct expected output, to eliminate the undesired outputs, and to obtain the success probability for the NS and CZ gates. It is important to notice that we generalized the CNOT, Toffoli, and Fredkin gates modeling by formalizing two versions (configurations) of each gate. Also, we demonstrated the usability of our approach to discover new designs for 1-qubit gates through the flip gate design that is based on quantum optics single photon technology. We then verified that the new flip gate design has 100% success probability and checked that the gate outputs for all possible inputs ($|0\rangle$ and $|1\rangle$) match the gate truth table. Finally, we have shown the success probability of each quantum gate that was analyzed.

In the next chapter, we will provide the description of a decision procedure to automate the analysis of quantum circuits constructed using the formalized quantum gates library and several quantum circuits that were analyzed using the proposed framework.

# Chapter 5

# Automated Quantum Circuits Verification

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

In the first part of this chapter, we present the verification process for quantum circuits taking as an example the Shor's algorithm for integer factorization of the number 15 circuit [41]. In the second part, we describe the developed decision procedure and use the quantum full adder [9] as a running example. Finally, we provide experimental results of applying the developed framework on the analysis of several benchmark quantum applications, including the above mentioned Shor's algorithm and quantum full adder.

## 5.1 Quantum Circuits Verification

In this section, we present the verification process for any quantum circuit using the developed mathematical foundation and the gates library. The verification process involves multiple rewriting of the tensor product using Theorems 3.2 and 3.3 and substituting one gate input by the output of the gate transformation when applied on that input. We use the Shor's integer factorization of the number 15 circuit as an example to illustrate the proof steps for verifying quantum circuits.

### 5.1.1 Shor's Algorithm

Shor's integer factorization [49] 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 modeling and verification of a compiled version (i.e., a designed version to find the prime factors of a specific input) of Shor's factoring for the number 15 [41] 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 5.1, and it has four inputs/outputs. The inputs are initialized to the state $|\psi\rangle_{in} = |0, 0, 1, 0\rangle_{x1f1f2x2}$. From the computed output, $|\psi\rangle_{out} = |., ., ., .\rangle_{\ddot{x}1\ddot{f}1\ddot{f}2\ddot{x}2}$, we extract the variable $z = |., ., 0\rangle_{\ddot{x}1\ddot{x}2}$, then we obtain $r = a^z \ mod \ 15$. Accordingly, we formally define the circuit structure in HOL as follows:

**Definition 5.1** (Shor's Circuit)**.**

$\vdash$ `shor`$(\mathtt{x1}, \mathtt{x2}, \mathtt{f1}, \mathtt{f2}, \ddot{\mathtt{f}}\mathtt{1}, \ddot{\mathtt{f}}\mathtt{2}, \ddot{\mathtt{x}}\mathtt{1}, \ddot{\mathtt{x}}\mathtt{2}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \Longleftrightarrow (\forall \mathtt{a2\ b2\ a1\ a3\ a4\ b3}.$

$\mathtt{CZ\_GATE}(\mathtt{a1}, \mathtt{a2}, \ddot{\mathtt{x}}\mathtt{1}, \mathtt{b2}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \ \wedge$

$\mathtt{CZ\_GATE}(\mathtt{a3}, \mathtt{a4}, \mathtt{b3}, \ddot{\mathtt{x}}\mathtt{2}, \mathtt{ten}, \mathtt{LH}, \mathtt{LV}, \mathtt{m\_proj}) \ \wedge$

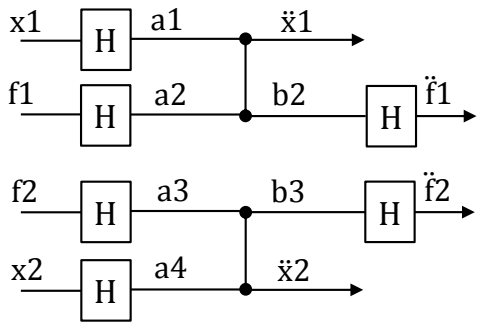


Figure 5.1: Shor's Factorization of Number 15 Circuit

$$\texttt{HADAMARD\_GATE}(\texttt{x1}, \texttt{a1}, \texttt{ten}, \texttt{LH}, \texttt{LV}) \wedge \texttt{HADAMARD\_GATE}(\texttt{f1}, \texttt{a2}, \texttt{ten}, \texttt{LH}, \texttt{LV}) \ \wedge$$
$$\texttt{HADAMARD\_GATE}(\texttt{f2}, \texttt{a3}, \texttt{ten}, \texttt{LH}, \texttt{LV}) \wedge \texttt{HADAMARD\_GATE}(\texttt{x2}, \texttt{a4}, \texttt{ten}, \texttt{LH}, \texttt{LV}) \ \wedge$$
$$\texttt{HADAMARD\_GATE}(\texttt{b2}, \ddot{\texttt{f}}\texttt{1}, \texttt{ten}, \texttt{LH}, \texttt{LV}) \wedge \texttt{HADAMARD\_GATE}(\texttt{b3}, \ddot{\texttt{f}}\texttt{2}, \texttt{ten}, \texttt{LH}, \texttt{LV}))$$

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

**Theorem 5.1** (Shor's Factoring of 15)**.**

$$\vdash \texttt{let constraints} = \texttt{is\_tensor\_proj m\_proj} \wedge \texttt{is\_tensor ten} \wedge$$
$$\texttt{shor } (\texttt{x1}, \texttt{x2}, \texttt{f1}, \texttt{f2}, \ddot{\texttt{f}}\texttt{1}, \ddot{\texttt{f}}\texttt{2}, \ddot{\texttt{x}}\texttt{1}, \ddot{\texttt{x}}\texttt{2}, \texttt{ten}) \texttt{ in}$$
$$\texttt{let } |0,0,1,0\rangle_{\texttt{f1x1f2x2}} = \texttt{tensor 4 } (\lambda \texttt{i. if i} = 1 \texttt{ then LH f1 elseif i} = 2$$
$$\texttt{then LH x1 elseif i} = 3 \texttt{ then LV f2 else LH x2}) \texttt{ in}$$
$$\texttt{let } |0,0,0,1\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} = \texttt{tensor 4 } (\lambda \texttt{i. if i} = 1 \texttt{ then LH } \ddot{\texttt{f}}\texttt{1 elseif i} = 2$$
$$\texttt{then LH } \ddot{\texttt{x}}\texttt{1 elseif i} = 3 \texttt{ then LH } \ddot{\texttt{f}}\texttt{2 else LV } \ddot{\texttt{x}}\texttt{2}) \texttt{ in}$$
$$\texttt{let } |0,0,1,0\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} = \texttt{tensor 4 } (\lambda \texttt{i. if i} = 1 \texttt{ then LH } \ddot{\texttt{f}}\texttt{1 elseif i} = 2$$
$$\texttt{then LH } \ddot{\texttt{x}}\texttt{1 elseif i} = 3 \texttt{ then LV } \ddot{\texttt{f}}\texttt{2 else LH } \ddot{\texttt{x}}\texttt{2}) \texttt{ in}$$
$$\texttt{let } |1,1,0,1\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} = \texttt{tensor 4 } (\lambda \texttt{i. if i} = 1 \texttt{ then LV } \ddot{\texttt{f}}\texttt{1 elseif i} = 2$$
$$\texttt{then LV } \ddot{\texttt{x}}\texttt{1 elseif i} = 3 \texttt{ then LH } \ddot{\texttt{f}}\texttt{2 else LV } \ddot{\texttt{x}}\texttt{2}) \texttt{ in}$$
$$\texttt{let } |1,1,1,0\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} = \texttt{tensor 4 } (\lambda \texttt{i. if i} = 1 \texttt{ then LH } \ddot{\texttt{f}}\texttt{1 elseif i} = 2$$
$$\texttt{then LH } \ddot{\texttt{x}}\texttt{1 elseif i} = 3 \texttt{ then LV } \ddot{\texttt{f}}\texttt{2 else LH } \ddot{\texttt{x}}\texttt{2}) \texttt{ in}$$
$$\texttt{constraints} \Longrightarrow |0,0,1,0\rangle_{\texttt{f1x1f2x2}} = \tfrac{1}{32} \ \% \ (|1,1,1,0\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} + |1,1,0,1\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}}$$
$$+ |0,0,1,0\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}} + |0,0,0,1\rangle_{\ddot{\texttt{f}}\texttt{1}\ddot{\texttt{x}}\texttt{1}\ddot{\texttt{f}}\texttt{2}\ddot{\texttt{x}}\texttt{2}})$$

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

The Shor's circuit input is as follows: `tensor 4` ($\lambda$`i. if i = 1 then LV x1 else if i = 2 then LH f1 else if i = 3 then LH f2 else LH x2`). As shown in Figure 5.1, the four parallel Hadamard gates are the first to act on the circuit input, therefore, we need to unfold the input to four elementary tensors of size one each. To perform this step of the proof, we rewrite the goal using tensor product theorems and the lemmas in Appendix D. Thereupon, the resulting output becomes in the form: ($\lambda$ `y.` (`tensor 1` ($\lambda$`i. LV x1`) `y$1`) $*$ (`tensor 1` ($\lambda$`i. LH f1`) `y$2`) $*$ (`tensor 1` ($\lambda$`i. LH f2`) `y$3`) $*$ (`tensor 1` ($\lambda$`i. LH x2`) `y$4`)). Next, we apply the four Hadamard gate transformations over the four elementary tensors to rewrite the goal with the Hadamard gate formalization and the lemmas in Appendix D. As explained in the previous chapter, the Hadamard gate output is a superposition of two states, therefore, each elementary tensor is replaced by two tensors. Thus, when we spread the main expression, we get sixteen terms ($2*2*2*2$). The first term is as follows: (`1/4`) % ($\lambda$ `y.` (`tensor 1` ($\lambda$`i. LV a1`) `y$1`) $*$ (`tensor 1` ($\lambda$`i. LH a2`) `y$2`) $*$ (`tensor 1` ($\lambda$`i. LH a3`) `y$3`) $*$ (`tensor 1` ($\lambda$`i. LH a4`) `y$4`)) $+ \cdots$.

After the four Hadamard gate transformations, the input will undertake two parallel CZ transformations, therefore, we need to fold back the four elementary tensors to one tensor and unfold this tensor to two elementary tensors by rewriting the goal using tensor product theorems and the lemmas in Appendix D. The resulting expression for the first term is as follows: (`1/4`) % ($\lambda$ `y.` (`tensor 2` ($\lambda$`i. if i = 1 then LV a1 else LH a2`) `y$1`) $*$ (`tensor 2` ($\lambda$`i. if i = 1 then LH a3 else LH a4`) `y$2`)) $+ \cdots$. We now apply the two CZ gate transformations over this expression to rewrite the goal with the CZ gate formalization and the lemmas in Appendix D. The first term of the expression is now as follows: (`1/64`) % ($\lambda$ `y.` (`tensor 2` ($\lambda$`i. if i = 1 then LV ẍ1 else LH b2`) `y$1`) $*$ (`tensor 2` ($\lambda$`i. if i = 1 then LH b3 else LH ẍ2`) `y$2`)) $+ \cdots$.

After folding and unfolding the tensor product, we apply the last two parallel

Hadamard gate transformations to rewrite the goal with the tensor product lemmas, the Hadamard gate formalization and the lemmas in Appendix D. Finally, after applying all the gate transformations, we obtain the final expression given in the RHS of the Theorem 5.1.

The process of verifying the Shor's integer factorization of the number 15 was not easy and involved more than 10 lemmas to prove. In addition, the proof of Theorem 5.1 required more than 150 lines of HOL Light proof script. Based on this result, the proof of circuits that involves dozens of gates may involve thousands of HOL Light proof script which is very tedious even for an expert in HOL. Hence, providing automation is necessary for our framework to be used in the analysis of quantum circuits. In the next section, we will describe a decision procedure that fully automates the analysis process.

## 5.2 Decision Procedure

Generally, any quantum circuit is a collection of gates that are connected to each other either sequentially or in parallel. Therefore, the main proof steps for the analysis of any quantum circuit are: 1) unfold the input tensor product to elementary tensors to be input to parallel gates as shown in Figure 5.2; 2) apply the required gates transformation; 3) fold the tensor product back. Then, we repeat this process until the input tensor goes through all the gates transformations that are sequential. Finally, we rewrite the obtained result to the final format using some linear algebra theorems.

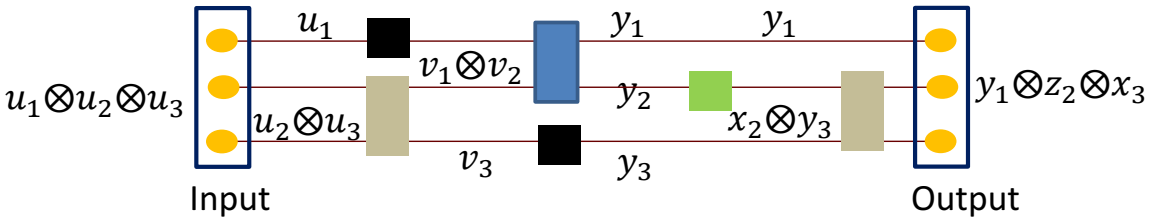


Figure 5.2: Tensor Product Unfolding

To facilitate the proof of the HOL theorem automatically without the need of user guidance, we have developed a decision procedure, given in Figure 5.3, that takes a quantum circuit netlist and its inputs and builds tactics[3] that automate the proof. The decision procedure first reads the quantum circuit netlist and generates a matrix that captures the circuit structure. For each gate of the circuit, the procedure then uses special rewriting rules that rewrite a gate outputs in terms of its inputs. Based on these rules and the extracted circuit matrix, the procedure generates the required folding/unfolding lemmas (which are related to the number of times the tensor product is going to be unfolded and folded back). Finally, using a set of simplification rules, we construct the final automation tactics to conduct the formal proof of the quantum circuit properties. Note that the decision procedure can also be used to formally validate if given inputs and outputs correspond to each other for a given circuit (i.e., we apply the quantum circuit to the given inputs and compare the obtained result with the expected outputs in order to validate it).

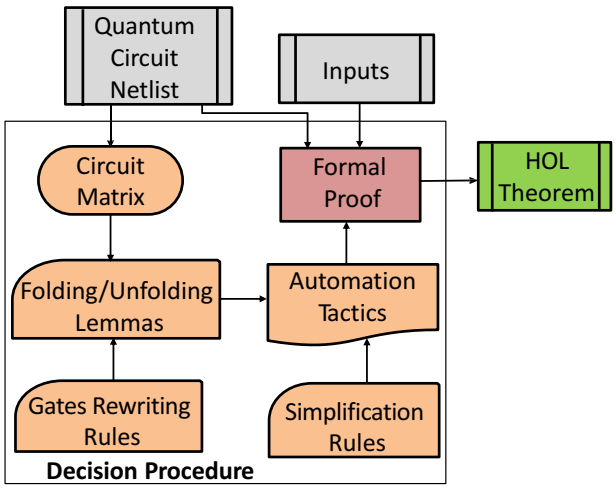


Figure 5.3: Decision Procedure

The core of the procedure is an Ocaml function that extracts from a textual quantum circuit netlist description a matrix that contains information about the

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

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

A second Ocaml function takes this matrix and generates the folding/unfolding lemmas and tactics. This function uses the extracted matrix to provide the proof steps, subgoals and lemmas to automatically prove the required theorems for the underlying circuit.

For example, consider the quantum circuit given in Figure 5.4, which is a quantum full adder composed of two SWAP gates, three CNOT gates and one Fredkin gate. Using the first Ocaml function described previously, we extract the following matrix:

```
[      [ 0 CNOT2 2 a0 a1 b0 b1 ]      [ 2 CNOT2 2 a2 a3 b2 b3 ] ]
|      [ 1 CNOT2 2 b1 b2 c1 c2 ]                                |
|       [ 0 SWAP 2 b0 c1 c0 d1 ]       [ 2 SWAP 2 c2 b3 d2 c3 ] |
[ [ 0 FREDKIN1 3 c0 d1 d2 e0 e1 e2 ]                            ]
```

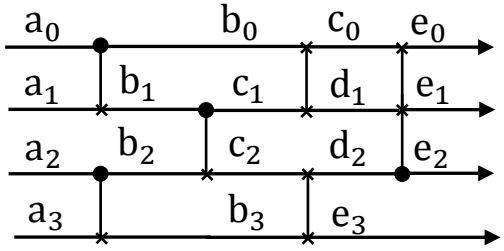


Figure 5.4: Quantum Full Adder

In the matrix, the first row contains the description of the gates that are applied in parallel to the circuit inputs. The subsequent rows describe, in order, the subsequent gates applied to previous gates outputs. Each element of the matrix contains the order of the gate (i.e., order of the gate inputs with regard to the circuit inputs, e.g., 1 means

that the gate input starts from the second element of the circuit input), its type, the number of inputs and the list of inputs and outputs. To illustrate the task of the sec-

Table 5.1: Tensor Product Folding/Unfolding Lemmas

| | |
|---|---|
| lemma1 | $\texttt{tensor m + n mode} = (\lambda \texttt{y}.\ (\texttt{tensor m mode})\ \texttt{y} *$ $(\texttt{tensor n}\ (\lambda \texttt{i}.\ \texttt{mode(i + m)}))\ (\lambda \texttt{i}.\ \texttt{y(i + m)}))$ |
| lemma2 | $(\texttt{if i} \leq \texttt{k1} \wedge \texttt{k2} \leq \texttt{i then}\ (\texttt{if i} = \texttt{k then x}_\texttt{k}\ \texttt{else} \cdots \texttt{if i} = \texttt{k2 then x}_\texttt{k2}$ $\texttt{else} \cdots \texttt{if i} = \texttt{k1 then x}_\texttt{k1}\ \texttt{else} \cdots \texttt{else x}_\texttt{m})\ \texttt{else y}) =$ $(\texttt{if i} \leq \texttt{k1} \wedge \texttt{k2} \leq \texttt{i then}\ (\texttt{if i} = \texttt{k2 then x}_\texttt{k2}\ \texttt{else} \cdots \texttt{else x}_\texttt{k1})\ \texttt{else y})$ |
| lemma3 | $\forall\ \texttt{i j k} \in \mathbb{N}.\ \ (\texttt{i + j} = \texttt{k}) \Longleftrightarrow (\texttt{if}\ (\texttt{j} \leq \texttt{k})\ \texttt{then}\ (\texttt{i} = \texttt{k} - \texttt{j})\ \texttt{else FALSE})$ |
| lemma4 | $\texttt{tensor m mode} = \texttt{tensor m}\ (\lambda \texttt{i}.\ \texttt{if i} \leq \texttt{m} \wedge \texttt{1} \leq \texttt{i then mode(i) else y})$ |
| lemma5 | $(\texttt{f}_1\ \texttt{x}_1) * \cdots * ((\texttt{a}_\texttt{k}\ \%\ \texttt{f}_\texttt{k})\ \texttt{x}_\texttt{k}) * \cdots * (\texttt{f}_\texttt{n}\ \texttt{x}_\texttt{n}) =$ $\texttt{a}_\texttt{k} * ((\texttt{f}_1\ \texttt{x}_1) * \cdots * (\texttt{f}_\texttt{k}\ \texttt{x}_\texttt{k}) * \cdots * (\texttt{f}_\texttt{n}\ \texttt{x}_\texttt{n}))$ |
| lemma6 | $(\lambda \texttt{y}.\ ((\texttt{tensor m mode1})\ \texttt{y}) * (\texttt{tensor n mode2})\ (\lambda \texttt{i}.\ \texttt{y(i + m)})) =$ $\texttt{tensor}\ (\texttt{m + n})\ (\lambda \texttt{i}.\ \texttt{if i} \leq \texttt{m then mode1(i) else mode2(i)})$ |
| lemma7 | $(\texttt{if i} \leq \texttt{m} \wedge \texttt{1} \leq \texttt{i then}\ (\texttt{if i} \leq \texttt{k then}\ (\texttt{if i} = \texttt{1 then x}_1\ \texttt{else} \cdots \texttt{else x}_\texttt{k})$ $\texttt{else}\ (\texttt{if i} = \texttt{1 then x}_{\texttt{k+1}}\ \texttt{else} \cdots \texttt{else x}_\texttt{m}))\ \texttt{else y}) =$ $(\texttt{if i} \leq \texttt{m} \wedge \texttt{1} \leq \texttt{i then}\ (\texttt{if i} = \texttt{1 then x}_1\ \texttt{else} \cdots \texttt{else x}_\texttt{m})\ \texttt{else y})$ |

ond Ocaml function and the flow of the decision procedure and the lemmas involved, consider a n-qubits circuit that contains a m-qubits gate ($\texttt{m} \leq \texttt{n}$), where the general form of the circuit input is: $\texttt{tensor n}\ (\lambda \texttt{i}.\ \texttt{if i} = \texttt{1 then x}_1\ \texttt{else} \cdots \texttt{else x}_\texttt{n})$. Two of the most important properties of the tensor product are the ability to write tensor as tensor of tensor (*lemma1* in Table 5.1) and vice versa (*lemma6* in Table 5.1). Then the first step in the proof is to rewrite the main tensor product (circuit input) using *lemma1*, *lemma2*, *lemma3* and *lemma4* in the form:

$(\lambda \texttt{y}.\ (\texttt{tensor k1}\ (\lambda \texttt{i}.\ \texttt{if i} = \texttt{1 then x}_1\ \texttt{else} \cdots \texttt{else x}_\texttt{k1}))\ \texttt{y} *$
$(\texttt{tensor m}\ (\lambda \texttt{i}.\ \texttt{if i} = \texttt{1 then x}_\texttt{k1+1}\ \texttt{else} \cdots \texttt{else x}_\texttt{k1+n}))\ (\lambda \texttt{i}.\ \texttt{y(i + k1)}) *$
$(\texttt{tensor k2}\ (\lambda \texttt{i}.\ \texttt{if i} = \texttt{1 then x}_\texttt{k1+n+1}\ \texttt{else} \cdots \texttt{else x}_\texttt{m}))\ (\lambda \texttt{i}.\ \texttt{y(i + k1 + m)}))$

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

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

The last step consists of folding back the tensor product by using *lemma4*, *lemma5*, *lemma6*, and *lemma7* of Table 5.1. Thereafter, the circuit input will become in the form:

$\mathtt{a}\ \%\ \mathtt{tensor\ n}\ (\lambda \mathtt{i}.\ \mathtt{if\ i} = \mathtt{1\ then\ x_1\ else} \cdots \mathtt{if\ i} = \mathtt{k1} + \mathtt{1\ then}$
$\mathtt{z_{k1+1}\ else} \cdots \mathtt{if\ i} = \mathtt{k1} + \mathtt{m} + \mathtt{1\ then\ x_{k1+m+1}\ else} \cdots \mathtt{else\ x_n})$

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

For this 4-qubit quantum adder, which input vector is in the form of `tensor 4 mode`, when the second Ocaml function takes the circuit matrix, in the first row we have two parallel gates, accordingly we should unfold the input tensor to two elementary tensors: `tensor 2 mode1` and `tensor 2 mode2` and apply the two CNOT gates to the two tensors as shown in Figure 5.5. Then, we fold back to the main tensor. Consequently, in the second row we have one gate, however, this gate order is in the middle of the main tensor. Therefore, we should unfold the main tensor to three elementary tensors: `tensor 1 mode1`, `tensor 2 mode2` and `tensor 1 mode3` and apply the CNOT gate to the elementary tensor `tensor 2 mode2`. Then, we fold back to the main tensor. Subsequently, we repeat the same procedures for the remaining two rows of the matrix until all gates are applied, and the final tensor product is obtained which is the circuit output.

In the following, we provide the detailed analysis of the quantum full adder circuit in HOL. The circuit model of quantum full adder is based on the design proposed in [9], which we have modified to meet the adjacency principle.

We consider a quantum full adder design depicted in Figure 5.4 to which we have

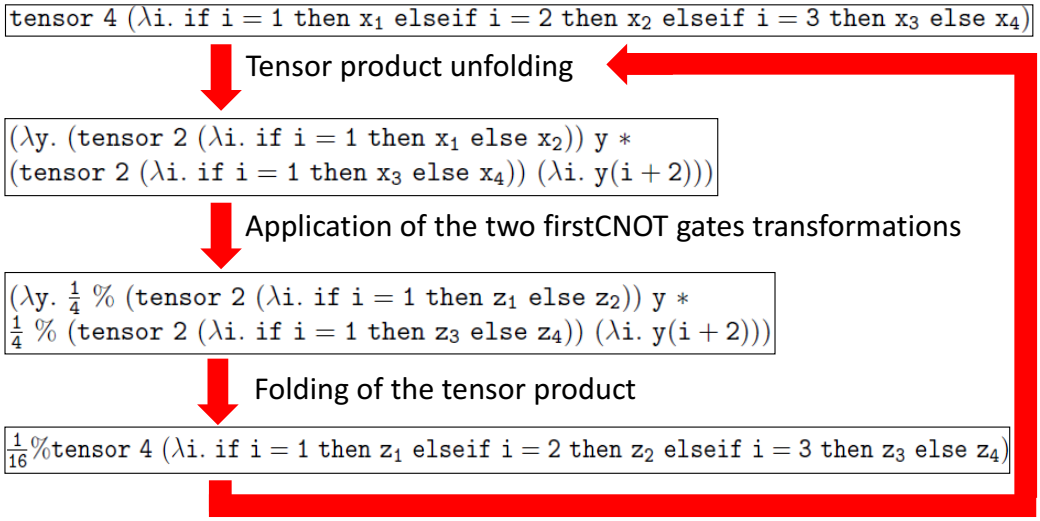


Figure 5.5: Proof Steps for Full Adder Circuit

added two swap gates to exchange the qubits before feeding them to the Fredkin gate. The circuit has four inputs; the two operands, the carry, and an extra input which is initialized to the state $|0\rangle$. We formally define the structure of the quantum full adder as follows:

**Definition 5.2** (Full Adder Circuit)**.**

$\vdash$ FULL_ADDER(a0, a1, a2, a3, e0, e1, e2, e3, ten, LH, LV, m_proj) $\iff$

($\forall$ 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 the quantum full adder in the general case where the two input values are added: $|x\rangle = x1\,|0\rangle_{a1} + x2\,|1\rangle_{a1}$ and $|y\rangle = y1\,|0\rangle_{a2} + y2\,|1\rangle_{a2}$, and the carry: $|z\rangle = z1\,|0\rangle_{a3} + z2\,|1\rangle_{a3}$.

**Theorem 5.2** (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 ($\lambda$i.if i = 1 then (x1%LH a1 + x2%LV a1)
elseif i = 2 then (y1%LH a2 + y2%LV a2) elseif i = 3 then
(z1%LH a3 + z2%LV a3) else LH a4) in
let output1 = tensor 4 ($\lambda$i. if i = 1 then LH b1 elseif i = 2 then
LH b2 elseif i = 3 then LH b3 else LH b4) in
let output2 = tensor 4 ($\lambda$i. if i = 1 then LV b1 elseif i = 2 then
LH b2 elseif i = 3 then LH b3 else LV b4) in
let output3 = tensor 4 ($\lambda$i. if i = 1 then LV b1 elseif i = 2 then
LH b2 elseif i = 3 then LV b3 else LV b4) in
let output4 = tensor 4 ($\lambda$i. if i = 1 then LH b1 elseif i = 2 then
LV b2 elseif i = 3 then LH b3 else LH b4) in
let output5 = tensor 4 ($\lambda$i. if i = 1 then LH b1 elseif i = 2 then
LH b2 elseif i = 3 then LV b3 else LV b4) in
let output6 = tensor 4 ($\lambda$i. if i = 1 then LV b1 elseif i = 2 then
LV b2 elseif i = 3 then LH b3 else LH b4) in
let output7 = tensor 4 ($\lambda$i. if i = 1 then LV b1 elseif i = 2 then
LV b2 elseif i = 3 then LV b3 else LH b4) in
let output8 = tensor 4 ($\lambda$i. if i = 1 then LH b1 elseif i = 2 then
LV b2 elseif i = 3 then LV b3 else LV b4) in
constraints $\Longrightarrow$ input = Cx(($\frac{\&1}{\&16}$) pow 7) $*$ ((z1 $*$ x1 $*$ y1)%output1+
(z1 $*$ x1 $*$ y2)%output2 + (z1 $*$ x2 $*$ y1)%output3 + (z1 $*$ x2 $*$ y2)%output4+
(z2 $*$ x1 $*$ y1)%output5 + (z2 $*$ x1 $*$ y2)%output6 + (z2 $*$ x2 $*$ y1)%output7+
(z2 $*$ x2 $*$ y2)%output8)

Here we have eight possible cases in the outputs of the adder, as the combinations of the three inputs gives eight possibilities. Notice that the success probability of the

quantum full adder is very low: $(\frac{1}{16})^7$.

## 5.3 Experimental Results

In this section we provide the results of the formal analysis of several quantum circuits. We have analysed several quantum benchmarks circuits taken from the online library of reversible and quantum circuits at [46]. The provided circuits do not meet the adjacency criteria in quantum computing. This criteria is supported experimentally and theoretically [4]. For example, in order to apply a 2-qubit gate to two elements $\mathtt{x_{k1}}$ and $\mathtt{x_{k2}}$ of an n-qubit input, the input should be in the form $\mathtt{tensor\ n}\ (\lambda \mathtt{i.if\ i} = \mathtt{1\ then} \cdots \mathtt{if\ x} = \mathtt{k1\ then\ x_{k1}\ elseif\ x} = \mathtt{k2\ then\ x_{k2}\ else} \cdots \mathtt{x_n})$. Therefore, we added SWAP gates to all quantum circuits taken from [46] to move the qubits to be adjacent to each other when they are applied to the same gate.

For example, consider the size 3 Hamming optimal coding function circuit given in Figure 5.4(a) [46]. In this circuit, the fourth gate (with dashed line) is applied to


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

Figure 5.6: Size 3 Hamming Circuit

two inputs that are not adjacent. Therefore, in order to meet the adjacency principle we added a SWAP gate before the fourth gate as shown in Figure 5.4(b).

We experimented with the following benchmark quantum circuits, which we automatically analysed using the developed decision procedure.

- **gf23mult** is about finding the product of two elements of a field $\mathtt{GF}(2^3)$, $\mathtt{a} = \mathtt{a_0} + \mathtt{a_1x} + \mathtt{a_2x^2}$ and $\mathtt{b} = \mathtt{b_0} + \mathtt{b_1x} + \mathtt{b_2x^2}$ with the output, $\mathtt{ab} = \mathtt{c} = \mathtt{c_0} + \mathtt{c_1x} + \mathtt{c_2x^2}$ written on the last 3 qubits.

- **2-to-4 decoder** that has 3 inputs and 4 outputs. If the enable qubit is low, all output qubits will be zero. If the enable qubit is high, one of the four output qubits will become high selected by the remaining two input qubits.
- **hwb4** is the hidden weighted bit function with four inputs/outputs. Its output equals its input shifted left by the number of positions equal to the number of ones in the input pattern.
- **ham3** is the size 3 Hamming optimal coding function.
- **mod5** is Grover's oracle, which has 4 inputs and 1 output. Its output is 1 if and only if the binary number represented by its input is divisible by 5.
- **6sym** has 6 inputs and 1 output. Its output is 1 if and only if the number of ones in the input pattern is 2, 3 or 4.
- **nth_prime3_inc** is used to find primes with up to 3 binary digits.

The result of the formal analysis of these quantum circuits is given in Table 5.2. The second column provides the number of gates in each circuit before adding the SWAP gate, and the third column provides the total number of gates. Details about the benchmark circuits can be found in [5].

The case studies that we have conducted demonstrate that our decision procedure significantly improves the degree of automation. Instead of using thousands of lines of HOL tactics to conduct the proof, we were able to achieve it automatically using the decision procedure. We believe that without the proposed decision procedure, we will not be able to formally analyse the circuits **6sym** and **gf23mult** that contain 61 gates.

## 5.4 Summary

In this chapter, we tackled a crucial subject in the interactive theorem provers which is the automation and the elimination of the need for user interaction with the theorem

Table 5.2: Formalized Quantum Circuits

| Circuit Name | Qubits | Gates without SWAP | Total Gates | Success Probability |
|---|---|---|---|---|
| nth_prime3_inc | 3 | 4 | 6 | $5.9 * 10^{-8}$ |
| ham3 | 3 | 5 | 6 | $9.5 * 10^{-7}$ |
| hwb4 | 4 | 12 | 22 | $1.2 * 10^{-29}$ |
| Shor's algorithm | 4 | 8 | 8 | $3.1 * 10^{-2}$ |
| full adder | 4 | 4 | 6 | $3.7 * 10^{-9}$ |
| Grover's oracle | 5 | 8 | 18 | $2 * 10^{-28}$ |
| 2-4 dec | 6 | 3 | 8 | $8.6 * 10^{-19}$ |
| gf2$\hat{3}$mult | 9 | 11 | 61 | $1.7 * 10^{-108}$ |
| 6sym | 10 | 20 | 61 | $1.7 * 10^{-102}$ |

prover. For instance, we have developed an automation procedure that fully automate the analysis process for any quantum circuit. This automation procedure helps in speeding up the analysis of new quantum circuits designs. Moreover, we showed the practicality of the proposed framework in the formal modeling and analysis of several real world quantum computing applications. We have tackled different kinds of quantum circuits that perform different functionalities. We have formally modeled and verified the Shor's algorithm for factorization of the number 15 and the Grover's oracle which are basically the most prominent quantum computing algorithms. Thus, we have formally verified the success probability and the outputs of the algorithms circuits.

In our formalization approach we have added SWAP gates to the quantum circuits that do not satisfy the principle of adjacency. Hence, this makes the designs more physically practical be constructed [4]. We also formally verified the quantum full adder circuit outputs and its success probability in the case where the inputs are superpositions of multi quantum states. In Table 5.2, we showed the usefulness of our approach in the analysis of optical quantum computing circuits by demonstrating how little is the success probability of the analyzed circuits. These results will help in finding new physical approaches for building quantum computing circuits and improving their success probabilities.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

Quantum computing systems are widely considered the next generation of computing system that will revolutionize the industry of computing and secure communication. Optics systems is one of several approaches under investigation for building a scalable quantum computer. Due to the novelty and nature of quantum computing, current CAD tools are not sufficient enough for these systems. Therefore, there is dire need for CAD tools to carry the systematic analysis of quantum computing systems. The development of these tools is believed to accelerate the pace for building quantum computers by providing environments for developing new designs, verifying these designs, writing quantum algorithms and synthesizing these algorithms to quantum gates and circuits. During the last decade, several software environments and CAD tools have been proposed for the analysis of quantum circuits and developing quantum algorithms. However, these tools work at the behavioral level which limits their efficient deployment for the development of quantum computers. In fact, all efforts to build quantum circuits are undertaken at the physical level using some specific technologies. There is hence a dire need for CAD tools that can conduct the analysis of quantum circuits at the physical level.

In this thesis, we proposed to leverage upon the expressiveness and accuracy of

higher-order-logic theorem proving to develop a framework for the formal analysis of quantum optics computing circuits. The main contributions of the proposed framework are: First, the development of several fundamental mathematical theories related to tensor product, which model the physical operation of measuring the output of quantum circuits. Second, the development of a library of commonly used quantum gates that are composed of optical elements. Third, the development of a decision procedure to automate the analysis process of any quantum circuits constructed using the quantum gates library. Fourth, the application of the framework on a set of benchmark circuits including the Shor's algorithm and the quantum full adder

The developed framework allowed us to discover a novel design of the flip gate and formally verified it. Moreover, it enabled us to extract the success probability of all formalized circuits, which provides an insight about the effectiveness of the proposed implementation. For instance, our analysis provided a very small success probability for the design of the quantum full adder, described in Chapter 5, which is, to the best of our knowledge, an optimum circuit in terms of the number of quantum gates. This result leads to considering alternative methods for implementing quantum circuits 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 proposed formal analysis framework along with the above mentioned practical quantum applications provide some thoughtful indications: theorem proving systems have reached to the maturity, where complex physical models can be expressed with less efforts than ever before; and formal methods can assist in the verification of futuristic quantum computing systems which are largely becoming the main research trend in computing industry nowadays. However, the question of the utilization of higher-order-logic theorem proving in an industrial settings and physical laboratories still persists due to the huge amount of time required to formalize the underlying mathematical theories. We believe that an important factor is the gap between the

theorem proving and engineering and physicists communities which limits its usage in industrial settings. For example, it is hard to find engineers (or physicists) with theorem proving background and vice-versa. One of the several solutions to tackle this issue is the continuous formal development of quantum theories including the synthesis and optimization of quantum circuits. The work presented in this thesis can be considered as a one step towards this goal.

The proof script of the formalization presented in this thesis require around 5500 lines of HOL Light code and 500 lines of OCaml code available at [5].

## 6.2 Future Work

The formalization and verification results, presented in this thesis, can be used as a complementary approach to provide a more expressiveness and accuracy to the existing techniques. In the following, we list some future research directions based on our experience and lessons learned during the course of this thesis:

- An immediate extension of this thesis is to build a simple graphical interface where the quantum circuit will be depicted as blocks connected between each other and a textual description of the circuit will be generated and fed to the developed framework in this thesis to conduct the automated formal verification and analysis with the help of the developed decision procedure.

- Another short term extension is to investigate the optimization of the number of gates in a circuit that performs a given functionality. Also, to develop a procedure to optimize the number of SWAP gates added to a quantum circuit in order to meet the principle of adjacency.

- A longer term extension of this thesis is to investigate the usage of quantum teleportation in quantum optics circuits to improve the success probabilities of these circuits. Quantum teleportation principle is such that the successful

result for a given quantum gate is teleported to the rest of the gates within the quantum circuit.

- Quantum cryptography has shown recently a good capability of securing communication transaction and several companies have started commercializing quantum cryptography products that are built using optics elements. Our work can be extended to cover the formal analysis of quantum cryptography protocols based on optics. This approach will provide an accurate analysis method which may interest the new born industry of quantum cryptography.

- It is possible to use high-order logic to do the formal synthesis of quantum functions and algorithms to quantum optics gates and circuits. This approach will be more efficient compared to the existing quantum circuits synthesis methods due to the low level design of the gates and circuits which gives more liberty to optimize the quantum circuits in term of number of gates. Another feature of this approach will be the synthesis of quantum functions and algorithms to circuits that are practically feasible to build in laboratory setups. For example several of the existing quantum synthesis approaches do not taken into account the adjacency constraint.

- The library built in this thesis contains only gates that are constructed using single photon technology which is the most common optics approach. However, it will be interesting to build other libraries that contain gates constructed from different technologies (i.e., squeezed states or coherent states). This will enable the possibility to compare the efficiency for a given quantum algorithm that is modeled using these different optics approaches and to choose the most efficient one in terms of the number of gates, success probability, and number of optics modes.

# Appendix A

# CZ Gate Behavioral Description

In this appendix, we define the behavioral description for CZ gate, where instead of eight inputs and eight outputs of optical modes, we have two behavioral inputs and two behavioral outputs. The first table is for the outputs behavioral description and the second table is for inputs behavioral description.

| **CZ_INPUTS (x1,x2,a, b, c,LH, LV,m_proj)** | |
|---|---|
| tensor 2 ($\lambda$i. if i = 1 then LH x1 else LH x2) | m_modes_pro (tensor 8 ($\lambda$i. 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$3)) (tensor 8 ($\lambda$i. 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$3)) |
| tensor 2 ($\lambda$i. if i = 1 then LV x1 else LH x2) | (m_modes_pro (tensor 8 ($\lambda$i. 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$3)) + m_modes_pro (tensor 8 ($\lambda$i. 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$3))+ m_modes_pro (tensor 8 ($\lambda$i. 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$3))) (tensor 8 ($\lambda$i. 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 ($\lambda$i. if i = 1 then LV x1 else LH x2) | (m_modes_pro (tensor 8 ($\lambda$i. 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$3)) + m_modes_pro (tensor 8 ($\lambda$i. 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$3))) (tensor 8 ($\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 fock a$3 1 elif i = 5 then fock b$51 else vac b$3)) |
| tensor 2 ($\lambda$i. if i = 1 then LH x1 else LV x2) | (m_modes_pro (tensor 8 ($\lambda$i. 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$3)) + m_modes_pro (tensor 8 ($\lambda$i. 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$3)) |
| tensor 2 ($\lambda$i. if i = 1 then vac x1 else LH x2) | m_modes_pro (tensor 8 ($\lambda$i. 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$3)) (tensor 8 ($\lambda$i. 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$3)) |
| tensor 2 ($\lambda$i. if i = 1 then vac x1 else LV x2) | (m_modes_pro (tensor 8 ($\lambda$i. 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$3)) + m_modes_pro (tensor 8 ($\lambda$i. 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$3)) |
| tensor 2 ($\lambda$i. if i = 1 then LH x1 else vac x2) | m_modes_pro (tensor 8 ($\lambda$i. 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$3)) (tensor 8 ($\lambda$i. 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 ($\lambda$i. if i = 1 then LV x1 else vac x2) | (m_modes_pro (tensor 8 ($\lambda$i. 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$3)) + m_modes_pro (tensor 8 ($\lambda$i. 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 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LH y1 else LH y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LV y1 else LV y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LV y1 else LH y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LH y1 else LV y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then vac y1 else LH y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then vac y1 else LV y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LH y1 else vac y2) |
| tensor 8 ($\lambda$i. 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$3) | tensor 2 ($\lambda$i. if i = 1 then LV y1 else vac y2) |

# Appendix B

# Second Version of CNOT Gate

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

**Definition B.1** (CNOT Gate)**.**
$\vdash$ CNOT2_GATE(a2, a1, b1, b2, ten, LH, LV, m_proj) $\iff$ ($\forall$ c1 d1.
HADAMARD_GATE(a1, c1, ten, LH, LV) $\wedge$
HADAMARD_GATE(d1, b2, ten, LH, LV) $\wedge$ CZ_GATE(a2, c1, b1, d1, 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 B.1** (CNOT Gate Input: $|x, y\rangle$)**.**
$\vdash$ let const = is_tensor_proj m_proj $\wedge$ is_tensor ten $\wedge$ 8 $\leq$ dimindex (: N)
$\wedge$ CNOT2_GATE (a1, a2, b1, b2, ten, LH, LV, m_proj) in
let x1y1 $|11\rangle_{\text{a}}$ + x1y2 $|10\rangle_{\text{a}}$ + x2y1 $|01\rangle_{\text{a}}$ + x2y2 $|00\rangle_{\text{a}}$ = tensor 2 ($\lambda$i. if i = 1
then (x1%LV a1 + x2%LH a1) else (y1%LV a2 + y2%LH a2)) in
let x1y1 $|01\rangle_{\text{b}}$ + x2y1 $|11\rangle_{\text{b}}$ = tensor 2 ($\lambda$i. if i = 1 then
(x1%LH b1 + x2%LV b1) else y1%LV b2) in
let x1y2 $|10\rangle_{\text{b}}$ + x2y2 $|00\rangle_{\text{b}}$ = tensor 2 ($\lambda$i. if i = 1 then
(x1%LV b1 + x2%LH b1) else y2%LH b2) in
const $\Longrightarrow$ x1y1 $|11\rangle_{\text{a}}$ + x1y2 $|10\rangle_{\text{a}}$ + x2y1 $|01\rangle_{\text{a}}$ + x2y2 $|00\rangle_{\text{a}}$ =
$\frac{1}{4}$ % (x1y1 $|10\rangle_{\text{b}}$ + x1y2 $|11\rangle_{\text{b}}$ + x2y1 $|01\rangle_{\text{b}}$ + x2y2 $|00\rangle_{\text{b}}$)

# Appendix C

# TS Gate Behavioral Description

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

| **TS_outputs (t,y1,y2,y3,LH,LV)** | |
|---|---|
| tensor 4 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LV y2 elif i = 3 then LV y3 else vac t$4) | tensor 3 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LV y2 else LV y3) |
| tensor 4 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LV y2 elif i = 3 then LH y3 else vac t$4) | tensor 3 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LV y2 else LH y3) |
| tensor 4 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then vac y2 elif i = 3 then LV y3 else LH t$4) | tensor 3 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LH y2 else LV y3) |
| tensor 4 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then vac y2 elif i = 3 then LH y3 else LH t$4) | tensor 3 ($\lambda$i. if i = 1 then LV y1 elif i = 2 then LH y2 else LH y3) |
| tensor 4 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then LV y2 elif i = 3 then LV y3 else vac t$4) | tensor 3 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then LV y2 else LV y3) |
| tensor 4 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then LV y2 elif i = 3 then LH y3 else vac t$4) | tensor 3 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then LV y2 else LH y3) |
| tensor 4 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then vac y2 elif i = 3 then LV y3 else LH t$4) | tensor 3 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then LH y2 else LV y3) |
| tensor 4 ($\lambda$i. if i = 1 then LH y1 elif i = 2 then vac y2 elif i = 3 then LH y3 else LH t$4) | tensor 3 ($\lambda$i. 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 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LV x2 else LV x3) | tensor 4 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LV x2 elif i = 3 then LV x3 else vac t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LV x2 else LH x3) | tensor 4 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LV x2 elif i = 3 then LH x3 else vac t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LH x2 else LV x3) | tensor 4 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then vac x2 elif i = 3 then LV x3 else LH t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then LH x2 else LH x3) | tensor 4 ($\lambda$i. if i = 1 then LV x1 elif i = 2 then vac x2 elif i = 3 then LH x3 else LH t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LV x2 else LV x3) | tensor 4 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LV x2 elif i = 3 then LV x3 else vac t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LV x2 else LH x3) | tensor 4 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LV x2 elif i = 3 then LH x3 else vac t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LH x2 else LV x3) | tensor 4 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then vac x2 elif i = 3 then LV x3 else LH t$4) |
| tensor 3 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then LH x2 else LH x3) | tensor 4 ($\lambda$i. if i = 1 then LH x1 elif i = 2 then vac x2 elif i = 3 then LH x3 else LH t$4) |

# Appendix D

# Proof Lemmas

In this appendix, we provide a set of lemmas used during the proof process for the verification the operations of the circuit of Shor's integer factorization for the number 15.

| |
|---|
| $4 = ((1 + 1) + 1) + 1 \wedge 1 <= 1$ |
| $1 + 1 = 2 \wedge 2 + 1 = 3 \wedge 3 + 1 = 4 \wedge 2 = 1 + 1 \wedge 1 <= 1$ |
| $4 = 2 + 2 \wedge 1 <= 2 \wedge 1 + 1 = 2 \wedge 2 + 2 = 4$ |
| $4 <= \mathtt{dimindex}(: \mathtt{N}) <=> (4 <= \mathtt{dimindex}(: \mathtt{N}) \wedge 3 <= \mathtt{dimindex}(: \mathtt{N}) \wedge$ $2 <= \mathtt{dimindex}(: \mathtt{N}) / \ 1 <= \mathtt{dimindex}(: \mathtt{N}))$ |
| $(1 <= \mathtt{i} + 3) \wedge (1 <= \mathtt{i} + 2) \wedge (1 <= \mathtt{i} + 1) \wedge$ $(4 <= \mathtt{dimindex}(: \mathtt{N}) ==> (\mathtt{i} <= 1 ==> (\mathtt{i} + 3 <= \mathtt{dimindex}(: \mathtt{N})$ $\wedge \ \mathtt{i} + 2 <= \mathtt{dimindex}(: \mathtt{N}) \wedge \mathtt{i} + 1 <= \mathtt{dimindex}(: \mathtt{N}))))$ |
| $((\mathtt{i} : \mathtt{num}) + \mathtt{j} = \mathtt{k}) <=> (\mathtt{if}\ (\mathtt{j} <= \mathtt{k})\ \mathtt{then}\ \mathtt{i} = \mathtt{k} - \mathtt{j}\ \mathtt{else}\ \mathtt{F})$ |
| $(\mathtt{i} <= 1 \wedge 1 <= \mathtt{i}) <=> \mathtt{i} = 1$ |
| $((\mathtt{a1}\%\mathtt{f1})\ \mathtt{y}) * ((\mathtt{a2}\%\mathtt{f2})\ \mathtt{x}) * ((\mathtt{a3}\%\mathtt{f3})\ \mathtt{z}) * (\mathtt{a4}\%\mathtt{f4})\ \mathtt{t} =$ $(\mathtt{a1} * \mathtt{a2} * \mathtt{a3} * \mathtt{a4}) \ * \ \mathtt{f1}\ \mathtt{y} * \mathtt{f2}\ \mathtt{x} * \mathtt{f3}\ \mathtt{z} * \mathtt{f4}\ \mathtt{t}$ |
| $(\lambda \mathtt{y}.\mathtt{a} * \mathtt{f1}\ \mathtt{y}) = \mathtt{a}\%(\lambda \mathtt{y}.\mathtt{f1}\ \mathtt{y})$ |
| $(\mathtt{x} : \mathtt{complex}) * \mathtt{y} * \mathtt{z} * \mathtt{t} = ((\mathtt{x} * \mathtt{y}) * \mathtt{z}) * \mathtt{t}$ |
| $((\mathtt{a1}\%\mathtt{f1})\ \mathtt{y}) * ((\mathtt{a2}\%\mathtt{f2})\ \mathtt{x}) = (\mathtt{a1} * \mathtt{a2}) * \mathtt{f1y} * \mathtt{f2x}$ |
| $(\sim (\mathtt{i} <= 3) ==> 1 <= \mathtt{i} - 3) \wedge (\sim (\mathtt{i} <= 2) ==> 1 <= \mathtt{i} - 2) \wedge$ $(\sim (\mathtt{i} <= 1) ==> 1 <= \mathtt{i} - 1) \wedge ((4 <= \mathtt{dimindex}(: \mathtt{N})) ==> (\mathtt{i} <= 4 ==>$ $(\mathtt{i} <= \mathtt{dimindex}(: \mathtt{N}) \wedge \mathtt{i} - 3 <= \mathtt{dimindex}(: \mathtt{N}) \wedge \mathtt{i} - 2 <= \mathtt{dimindex}(: \mathtt{N})$ $\wedge \mathtt{i} - 1 <= \mathtt{dimindex}(: \mathtt{N}))))$ |
| $1 <= \mathtt{i} + 2 \wedge ((4 <= \mathtt{dimindex}(: \mathtt{N})) ==> (\mathtt{i} <= 2 ==> (\mathtt{i} + 2 <= \mathtt{dimindex}(: \mathtt{N}))))$ |
| $(\mathtt{if}(\mathtt{i} <= 2 \wedge 1 <= \mathtt{i})\ \mathtt{then\ if\ i} <= 3\ \mathtt{then\ if\ i} <= 1\ \mathtt{then\ x1\ else\ x2\ else\ x3\ else}$ $\mathtt{g\ i}) = (\mathtt{if}\ (\mathtt{i} <= 2 \wedge 1 <= \mathtt{i})\ \mathtt{then\ if\ i} = 1\ \mathtt{then\ x1\ else\ x2\ else\ g\ i}) \wedge$ $(\mathtt{if}\ (\mathtt{i} <= 2 \wedge 1 <= \mathtt{i})\ \mathtt{then\ if\ i} <= 1\ \mathtt{then\ if\ i} <= 0\ \mathtt{then\ x1\ else\ x2\ else\ x3}$ $\mathtt{else\ g\ i}) = (\mathtt{if}\ (\mathtt{i} <= 2 \wedge 1 <= \mathtt{i})\ \mathtt{then\ if\ i} = 1\ \mathtt{then\ x2\ else\ x3\ else\ g\ i})$ |
| $(4 <= \mathtt{dimindex}(: \mathtt{N}) <=> (4 <= \mathtt{dimindex}(: \mathtt{N}) \wedge 1 <= \mathtt{dimindex}(: \mathtt{N})))$ |
| $1 <= \mathtt{i} + 1 \wedge ((4 <= \mathtt{dimindex}(: \mathtt{N})) ==>$ $(\mathtt{i} <= 1 ==> (\mathtt{i} + 1 <= \mathtt{dimindex}(: \mathtt{N}))))$ |

| |
|---|
| $(\texttt{f1 y}) * (\texttt{f2 x}) * (\texttt{f3 z}) * \texttt{f4 t} = (\texttt{f1 y} * \texttt{f2 x}) * (\texttt{f3 z} * \texttt{f4 t})$ |
| $(((\texttt{a1} \% \texttt{f1})\ \texttt{y}) * (\texttt{f2 x})) * (((\texttt{a2} \% \texttt{f3})\ \texttt{z}) * \texttt{f4 t}) =$<br>$(\texttt{a1} * \texttt{a2}) * ((\texttt{f1 y} * \texttt{f2 x}) * (\texttt{f3 z} * \texttt{f4 t}))$ |
| $(\sim (\texttt{i} - 2 <= 1) ==> 1 <= \texttt{i} - 2 - 1) \wedge (\sim (\texttt{i} <= 2) ==> 1 <= \texttt{i} - 2) \wedge$<br>$(\sim (\texttt{i} <= 1) ==> 1 <= \texttt{i} - 1) \wedge ((4 <= \texttt{dimindex}(: \texttt{N})) ==> (\texttt{i} <= 4 ==>$<br>$(\texttt{i} <= \texttt{dimindex}(: \texttt{N}) \wedge \texttt{i} - 2 - 1 <= \texttt{dimindex}(: \texttt{N}) \wedge \texttt{i} - 2 <= \texttt{dimindex}(: \texttt{N})$<br>$\wedge\ \texttt{i} - 1 <= \texttt{dimindex}(: \texttt{N}))))$ |
| $(\texttt{if}\ (\texttt{i} <= 4 \wedge 1 <= \texttt{i})\ \texttt{then if i} <= 2\ \texttt{then if i} <= 1\ \texttt{then x1 else x2 else}$<br>$\texttt{if i} - 2 <= 1\ \texttt{then x3 else x4 else x5}) = (\texttt{if}\ (\texttt{i} <= 4 \wedge 1 <= \texttt{i})\ \texttt{then}$<br>$\texttt{if i} = 1\ \texttt{then x1 else if i} = 2\ \texttt{then x2 else if i} = 3\ \texttt{then x3 else x4 else x5})$ |

# Bibliography


[1] D. Aerts and I. Daubechies. Physical Justification for Using the Tensor Product to Describe two Quantum Systems as one Joint System. *Helvetica Physica Acta*, 51:661–675, 1978.

[2] D. Aharonov. A Simple Proof that Toffoli and Hadamard are Quantum Universal. arXiv preprint quant-ph/0301040, 2003.

[3] 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.

[4] A. Barenco, C. H. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary Gates for Quantum Computation. *The American Physical Society*, 52:3457–3467, 1995.

[5] S. M. Beillahi. HOL-Light Source Code. `http://hvg.ece.concordia.ca/projects/optics/quantumcad.html`.

[6] S. M. Beillahi, M. Y. Mahmoud, and S. Tahar. Optical Quantum Gates Formalization in HOL Light. Technical report, ECE Department, Concordia University, Montreal, QC, Canada, February 2016.

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

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

[9] 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.

[10] Coq Proof Assistant. `https://coq.inria.fr`, 2016.

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

[12] A. G. Fowler, S. J. Devitt, 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.

[13] 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.

[14] C. Gerry and P. Knight. *Introductory Quantum Optics*. Cambridge University Press, 2005.

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

[16] D. J. Griffiths. *Introduction to Quantum Mechanics*. Pearson Prentice Hall, USA, 1995.

[17] 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.

[18] J. Harisson. HOL-Light Revision r200. `https://code.google.com/p/hol-light/source/detail?r=200`, October, 2014.

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

[22] 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.

[23] 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.

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

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

[26] 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.

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

[28] A. Lye, R. Wille, and R. Drechsler. Determining the Minimal Number of Swap Gates for Multi-Dimensional Nearest Neighbor Quantum Circuits. In *Asia and South Pacific Design Automation Conference*, pages 178–183, 2015.

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

[30] 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.

[31] M. Y. Mahmoud, V. Aravantinos, and S. Tahar. Formal Verification of Optical Quantum Flip Gate. In *Interactive Theorem Proving*, volume 8558 of *LNCS*, pages 358–373. Springer, 2014.

[32] M. Y. Mahmoud, P. Panangaden, and S. Tahar. On the Formal Verification of Optical Quantum Gates in HOL. In *Formal Methods for Industrial Critical Systems*, volume 9128 of *LNCS*, pages 198–211. Springer, 2015.

[33] L. l Mande and E. Wolf. *Optical Coherence and Quantum Optics*. Cambridge University Press, 1995.

[34] S. Mathias, R. Wille, C. Hilken, and N. Przigoda. Synthesis of Reversible Circuits with Minimal Lines for Large Functions. In *Asia and South Pacific Design Automation Conference*, pages 85–92, 2012.

[35] G. J. Milburn. Quantum Optical Fredkin Gate. *Physical Reviw Letter*, 62:2124–2127, May 1989.

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

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

[38] T. Nipkow, M. Wenzel, and L. C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer-Verlag, 2002.

[39] S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer, 1992.

[40] K. R. Parthasarathy. *An Introduction to Quantum Stochastic Calculus*. Springer, 1992.

[41] 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.

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

[43] 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.

[44] 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.

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

[46] RevLib. An Online Resource for Benchmarks Within the Domain of Reversible and Quantum Circuit. `http://www.revlib.org/`, 2016.

[47] 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.

[48] 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):1000–1010, June 2006.

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

[50] K. Slind and M. Norrish. A Brief Overview of HOL4. In *Theroem Proving in Higher Order Logics*, volume 5170 of *LNCS*, pages 28–32. Springer, 2008.

[51] G. F. Viamontes, I. L. Markov, and J. P. Hayes. Graph-Based Simulation of Quantum Computation in the Density Matrix Representation. *Quantum Information and Computation*, 5:113–130, 2005.

[52] 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.

[53] R. Wille, A. Lye, and R. Drechsler. Optimal SWAP Gate Insertion for Nearest Neighbor Quantum Circuits. In *Asia and South Pacific Design Automation Conference*, pages 489–494, 2014.

[54] 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.

[55] S. Yamashita and I. L. Markov. Fast Equivalence-Checking for Quantum Circuits. In *IEEE/ACM International Symposium on Nanoscale Architectures*, pages 23–28. IEEE Press, 2010.