



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

An online-accessible discrete variable photonic one-way
quantum computer

verfasst von / submitted by

Felix Johannes Zilk, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA 066 876

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Masterstudium Physics

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Ing. Dr. Philip Walther

Abstract

Several quantum computing resources have been made available online by cloud providers and research labs in recent years. The majority of these platforms use gate-based QPUs or simulators. The one-way quantum computing model is an alternative model of quantum computing that uses single-qubit measurements on a highly entangled resource state and classical feed-forward. This approach is especially attractive for photonic implementations. In this work, we build a photonic experiment that processes up to four polarization-encoded qubits. The experiment is accessible online via a REST API, and users can execute the desired circuit using a graphical user interface.

In den vergangenen Jahren wurden mehrere Quantencomputerressourcen von Cloud Providern und Forschungslaboren online zur Verfügung gestellt. Die meisten dieser Plattformen verwenden Gatter-basierte QPUs oder Simulatoren. Das Einweg-Quantencomputer-Modell ist ein alternatives Modell für Quantencomputer, das eine Reihe von Einzel-Qubit-Messungen auf einem hochgradig verschränkten Ressourcenzustand und klassisches Feed-forward verwendet. Dieser Ansatz ist besonders für photonische Implementierungen attraktiv. In dieser Arbeit bauen wir ein photonisches Experiment, das bis zu vier Polarisations-Qubits verarbeitet. Das Experiment ist online über eine REST API verfügbar, und Benutzer können

einen gewünschten Schaltkreis mithilfe einer grafischen Benutzeroberfläche ausführen.

Keywords:

Quantum Computing, Photonic QC, Measurement-based QC, One-way QC, Online QC, Cloud-based QC

Acknowledgement

Above all, I would like to thank *Philip Walther* for his enormous leap of faith, without which I would never have been able to work on this exciting project.

I would also like to thank *Juan Lored* for his support with all experiment-related matters since the beginning of the project.

I'd like to thank *Tobias Guggemos* for his guidance during the past five months, in particular for his contributions to the PhotonQ compiler.

I am thankful to *Omar Ortiz* for assisting the project day and night with programming work, mostly with the software backend in the lab.

I also owe thanks to *Daniel Petutschnigg* for being a great mentor and instructor to me when I first started with Django and Django REST.

Thank you also to *Florian Kleber* for all of the hours we spent together building up the network and server infrastructure for the first demo, which took place in May 2022 as part of the CDL Review Meeting. I am glad that Florian has agreed to take up PhotonQ's IT maintenance and development.

Finally, I am grateful to *Esther Sztatecsny*, *Francesco Giorgino*, and *Daniel Beckord*, who read the drafts of this thesis and provided me with feedback and support.

Contents

1	Introduction	1
1.1	Quantum algorithms	2
1.2	Quantum hardware	2
1.3	Quantum cloud services	4
1.4	Aim and method of this work	5
2	Quantum information fundamentals	7
2.1	Basic concepts of quantum information	7
2.1.1	Single and multiple qubits	7
2.1.2	Projective measurements	12
2.1.3	The density operator	14
2.1.4	Entanglement	15
2.2	The quantum circuit model	18
2.2.1	Single-qubit gates	19
2.2.2	Two-qubit gates	22
2.2.3	Measurements	25
2.3	The one-way model of quantum computing	25
2.3.1	Graph states and cluster states	26
2.3.2	One-way quantum computation	29
3	Photonic quantum computing	37
3.1	Photons as qubits	37
3.2	Single-qubit gates	39
3.3	Two-qubit gates	41
3.4	Measuring polarization qubits	44

4 Experimental setup	47
4.1 Source	47
4.2 Graph state preparation	49
4.3 Detection stage	50
5 PhotonQ system stack	53
5.1 System stack overview	53
5.2 Web application	54
5.3 CDL web service	58
5.4 Laboratory-sided backend	60
6 Results & outlook	65
References	75
Nomenclature	77
List of Figures	83
List of Tables	85
A Full Experiment Payload Schema	87
B Published Material	89

Chapter 1

Introduction

“That was a big revolution in classical computing, giving access to everyone. Through trial and error, that led to huge advances, and I think this is going to be true with quantum computing as well.”

Alexander Keesling, *CEO at QuEra Computing Inc.*

In May 2016, IBM launched the *IBM Quantum Experience*, a service within its cloud that allowed everyone with an internet connection to remotely program their five-qubit in-house superconducting quantum processor [16]. The commercialization of quantum computer technology begins at this point, at the very least.

Since then, an entire quantum computing (QC) ecosystem has emerged: according to a McKinsey report from December 2021 [12], there are approximately 60 QC hardware manufacturers, more than 120 software providers, more than ten cloud computing vendors, and more than 40 industrial companies working on applications. Tens of billions of dollars have been invested in the QC ecosystem alone (for example, through the Quantum Flagship Program in Europe), and the number of startups has nearly quintupled in the last three years.

1.1 Quantum algorithms

The properties of quantum information's superposition and entanglement enable a new computational paradigm that is fundamentally different from the classical view of computation. This new type of computing allows for the creation of novel algorithms, thereby opening up the field of quantum algorithms.

Since theoretical computer scientists started creating algorithms for quantum computers in the 1990s, solving important problems that classical computers cannot (efficiently) deal with, QC has received a lot of consideration and attention. The algorithm developed by Peter Shor in 1991 is a well-known example [48]. It solves the problems of prime factorization and discrete logarithms in polynomial time. This highly affects digital security for online payments and cloud computing [40, 41].

Also, quantum algorithms for simulating quantum mechanical systems achieve exponential speedups when compared to the best known classical algorithms [39]. The resulting applications in quantum chemistry range from material science (identification of better materials that satisfy desirable properties) to medicine (discovery of new drugs) [4].

Another well-known quantum algorithm that provides a quadratic speedup over classical algorithms when performing search-based tasks is Grover's algorithm [23]. The website Quantum Algorithm Zoo provides an extensive collection of quantum algorithms [25]. Current research is focusing on whether there are more problems for which quantum algorithms outperform classical ones [39, 41].

1.2 Quantum hardware

The theoretical descriptions of many powerful quantum algorithms assume full fault-tolerance of the hardware. Fault-tolerant quantum computers require thousands to millions of stable physical qubits with long coherence times, as well as the ability to perform high-fidelity quantum logic operations and correct for errors that are likely to occur

during computation [1, 11, 32, 38]. The scaling of the hardware while at the same time preserving the quantum coherence of the system is a major challenge in the engineering of quantum computers [1].

After all, today's quantum computers are still in an early stage of development. Qubits have been realized in various physical systems, including atoms and ions, photons, superconducting systems, electronic semiconductor qubits, and spin systems in solids. Each platform has its own unique advantages and drawbacks [1, 30]. Most modern experiments involve dozens of qubits, with limited coherence times and significant error rates for single- and two-qubit gates. Such erroneous hardware accumulates noise during the computation, which restricts the number of quantum operations that can be reliably executed. However, noisy QC devices of intermediate scale already exist, which gave rise to the term *noisy intermediate scale quantum* (NISQ) computer [43]. NISQ hardware is expected to improve its performance in the coming decades [1, 11].

The development of quantum hardware is an active field of research and is progressing continuously. Advancements in QC hardware are evident in experiments by Google [5] and a research group in China [56], which have demonstrated quantum computational supremacy for specific tasks using a superconducting quantum processor (Google, 2019) and photonic systems (China, 2020). In addition, scientists from the University of Innsbruck and Alpine Quantum Technologies have launched the first 19-inch trapped-ion rackmount quantum computer ready for integration into server rooms [42]. Furthermore, IBM has recently announced its latest quantum chip, which reaches 127 qubits, making it the first quantum processor to reach a three-digit number in qubit count [7].

On the road to fault-tolerant QC, there are still many technical hurdles to overcome in improving the error rates of quantum devices, and fault-tolerance may be many decades away. Until then, it is important to develop methods for finding valuable solutions with error-prone devices [38].

1.3 Quantum cloud services

Traditional cloud companies offer various services for accessing quantum hardware and simulators [6, 24, 37]. This is often referred to as *quantum cloud computing*, or more accurately, *cloud-based quantum computing* [26, 33]. Typically, a web service provides an API that acts as a gateway to the hardware in the laboratory. Such services are especially important for integrating quantum processing units (QPUs) as high-performance computing (HPC) accelerators in data centers [10]. The users submit their programs in the form of quantum circuits by sending requests to the web service that contain the required circuit parameters and meta data for translation and execution of the circuit on the actual quantum device. A quantum circuit is a sequential set of quantum logic operations and measurements that manipulate the data stored in qubits.

The technical definition of a cloud is given in [35]. QC is yet in an experimental stage, however, we can use cloud services to make QC available to the general public. Even if current QPUs cannot solve practical problems of relevant size, cloud-based QC allows users to compare architectures [22], test theories [28], and realize QC protocols [18].

Moreover, the cloud offers the ideal environment for the execution of quantum programs. Many services required for the execution of practical quantum applications, such as user management, network, and other traditional computer services, are already deployed as cloud services, and the quantum service only needs to be added to that infrastructure [33].

Current research on cloud-based QC has multiple layers. Lower-level research focuses on improving latency, CPU-QPU interaction, compilation tasks, and translating abstract instructions to physical execution [26]. High-level topics include developing better software development environments for quantum computers, testing and debugging quantum software [3, 54], deployment methodologies for hybrid quantum-classical systems [33], optimal input and output data formats [33], and different interfaces to meet the needs of different user groups [17]. System architectural questions include the

optimal integration of all components of the quantum cloud computing stack, as well as integration with classical technologies for container applications and cloud orchestration [33, 36].

1.4 Aim and method of this work

The aim of this thesis is an online-accessible photonic quantum computer based on the one-way model. Discrete variable (DV) photonic one-way processors have yet to be investigated as a backend in quantum cloud computing and as accelerators in the HPC environment.

As part of the Christian Doppler Laboratory for Photonic Quantum Computing’s online QC project, named *PhotonQ*, this thesis focused on

1. building the experimental photonic setup that serves as the physical hardware backend. The experimental setup is conceptually similar to the experiment in [34].
2. developing the REST API that exposes the hardware backend on the web. We choose a REST API with JSON support to expose our backend as online service.

This thesis is structured as follows: Chapter 2 revises the fundamental concepts of quantum information and the standard quantum circuit model and introduces the measurement-based model of QC with cluster states. Chapter 3 explains the realization and properties of photonic qubits and photonic quantum gates. Chapter 4 discusses the experimental photonic setup that serves as PhotonQ’s hardware backend. Chapter 5 outlines the implemented system stack of PhotonQ and specifies the calls, endpoints, and data structures of the REST API. Chapter 6 concludes with results and an outlook.

In the course of the work, a compiler framework for photonic one-way computers is also developed and presented at the SC22 conference in conjunction with the *Third International Workshop on Quantum Computing Software*.

The preprint of the workshop paper, as well as a conference poster presented at the *SFB BeyondC Conference 2022*, are printed in Appendix B.

Chapter 2

Quantum information fundamentals

“To me quantum computation is a new and deeper and better way to understand the laws of physics, and hence understanding physical reality as a whole.”

David Deutsch, *Visiting
Professor at Oxford University*

2.1 Basic concepts of quantum information

This section reviews the underlying physical concepts of quantum information. Most of this chapter is based on [41] and [8], unless otherwise referenced.

2.1.1 Single and multiple qubits

The fundamental unit of information in classical computation is a *bit*, which can take the binary values 0 or 1. *Quantum bits*, or *qubits* for short, are the quantum-mechanical analog concept of a classical bit in

quantum computation and quantum information. A qubit's logical state is represented as a unit vector in the two-dimensional complex Hilbert space $\mathcal{H} = \mathbb{C}^2$. Qubits can also be in a linear combination of states, also referred to as *superposition*. Any normalized superposition of basis states in $\mathbb{C}^2$ is a valid state for a qubit. This is expressed formally as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \quad (2.1)$$

The states $|0\rangle$ and $|1\rangle$ form an orthonormal basis in the Hilbert space $\mathcal{H}$ of the qubit's state, and are often called *computational basis states*. After a measurement, the qubit is found in state $|0\rangle$ with probability $|\alpha_0|^2$ and in state $|1\rangle$ with probability $|\alpha_1|^2$ (see Section 2.1.2). The complex amplitudes α_0 and α_1 satisfy the normalization constraint $\langle\psi|\psi\rangle = |\alpha_0|^2 + |\alpha_1|^2 = 1$; and $\alpha_0, \alpha_1 \in \mathbb{C}$.

Single-qubit states can be intuitively visualized on a three-dimensional unit sphere, the so-called *Bloch sphere* (see Figure 2.1). Using the polar form of complex numbers, the normalization constraint, and a few straightforward algebraic operations, we are able to derive the representation

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \quad (2.2)$$

of the state $|\psi\rangle$ from Equation 2.1.

The real numbers θ and ϕ in this geometric representation correspond to the polar and azimuthal angles in the Bloch sphere picture, respectively (see Figure 2.1). In these terms, every quantum state $|\psi\rangle$ can be mapped to a point on a three-dimensional unit sphere with spherical coordinates (θ, ϕ) . Furthermore, single-qubit gate operations are easily illustrated using the Bloch sphere picture (see Section 2.2.1).

Measurements alter the state of a qubit and project it onto one of the basis states, which is one of the fundamental postulates of quantum mechanics. The measurement process for projective measurements will be explained more formally in Section 2.1.2. However, as a practical result of this fact, only one bit of information can be extracted from a single qubit with a single measurement.

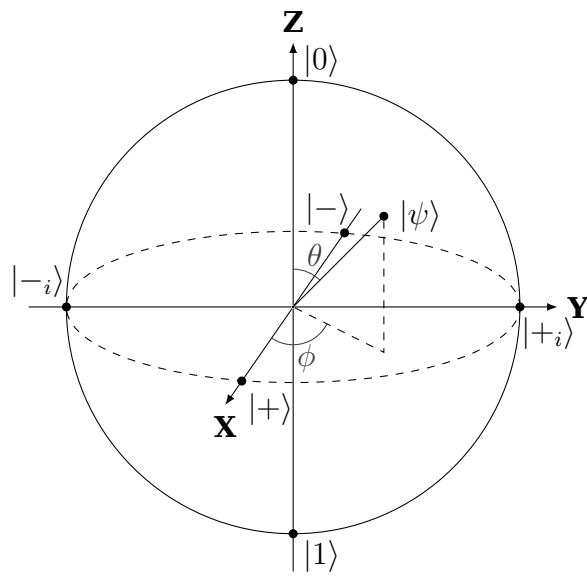


Figure 2.1: The Bloch sphere visualizes single-qubit states on a three-dimensional unit sphere. The state $|\psi\rangle$ from Equation 2.1 can be rewritten to be characterized by two real numbers, θ and ϕ , which correspond to the polar and azimuthal angles on the Bloch sphere.

Pauli Matrix	Eigenvalues	Eigenvectors
σ_z	± 1	$\{ 0\rangle, 1\rangle\}$
σ_x	± 1	$\{ +\rangle, -\rangle\}$
σ_y	± 1	$\{ +_i\rangle, -_i\rangle\}$

Table 2.1: Eigenvalues and eigenvectors of the Pauli matrices.

In addition, copying quantum information in the general case of $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ is not possible.

Other relevant basis states in $\mathbb{C}^2$ are

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.3)$$

and

$$|+_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \quad |-_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle) \quad (2.4)$$

In general, any two linearly independent unit vectors form a basis in $\mathbb{C}^2$. The bases $\{|+\rangle, |-\rangle\}$ and $\{|+_i\rangle, |-_i\rangle\}$ are eigenvectors of the Pauli matrices σ_x and σ_y , respectively; σ_z has eigenvectors $\{|0\rangle, |1\rangle\}$.

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (2.5)$$

The *eigenvalues* with *eigenvectors* of the Pauli matrices are listed in Table 2.1 and can be easily verified by writing the states in matrix notation:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (2.6)$$

Together with the 2×2 identity matrix the Pauli matrices form a basis of the space of linear operators on $\mathcal{H} = \mathbb{C}^2$.

Multi-qubit states are described using the same formalism as for single-qubit states. The tensor product becomes the glue between the Hilbert spaces of the individual qubits. A system of two qubits is

described with a state $|\Psi_2\rangle \in \mathcal{H}_2 = \mathbb{C}^2 \otimes \mathbb{C}^2$, where $\otimes$ denotes the tensor product. Thus, a set of basis states of $\mathcal{H}_2$ is given by

$$|00\rangle \equiv |0\rangle_1 \otimes |0\rangle_2, \quad (2.7)$$

$$|01\rangle \equiv |0\rangle_1 \otimes |1\rangle_2, \quad (2.8)$$

$$|10\rangle \equiv |1\rangle_1 \otimes |0\rangle_2, \quad (2.9)$$

$$|11\rangle \equiv |1\rangle_1 \otimes |1\rangle_2, \quad (2.10)$$

where the subscripts label the qubits. A general two-qubit state can therefore be written as a superposition of these four basis states:

$$|\Psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle, \text{ with } \alpha_{ij} \in \mathbb{C} \quad (2.11)$$

The normalization condition is $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

After measuring the system described by $|\Psi_2\rangle$, we find that it is in state $|00\rangle$ (respectively $|01\rangle$, $|10\rangle$, $|11\rangle$) with probability $|\alpha_{00}|^2$ (respectively $|\alpha_{01}|^2$, $|\alpha_{10}|^2$, $|\alpha_{11}|^2$).

The generalization of this method to n -qubit states is straight forward:

$$|\Psi_n\rangle = \sum_{i=1}^{2^n} \alpha_i |x_1\rangle \otimes |x_2\rangle \otimes \dots \otimes |x_n\rangle, \quad (2.12)$$

or as in short-hand notation

$$|\Psi_n\rangle = \sum_{i=1}^{2^n} \alpha_i |x_1 x_2 \dots x_n\rangle \quad (2.13)$$

Due to the constraint that probabilities sum to one, the probability amplitudes fulfill $\sum_i |\alpha_i|^2 = 1$, and $x_i \in \{0, 1\}$.

A quantum computer with n qubits can process an exponential number of basis states; 2^n states, to be precise. Thus, the description of such a state and its evolution over time involves an exponential amount of parameters (complex amplitudes), which provides a heuristic explanation for why quantum computation is difficult to simulate on classical computers [55].

2.1.2 Projective measurements

The measurement process is fundamental in quantum physics, particularly in quantum information, and its formalization is one of the postulates of quantum mechanics. A measurement is understood as the interaction of a measuring instrument with the measured quantum system. When the state of the system is measured, it is projected onto the eigenspace of an observable $\mathcal{O}$, and the measured value is the associated eigenvalue. This is known as *projective measurement*.

The observable $\mathcal{O}$ is fully characterized by a hermitian operator $\hat{\mathcal{O}}$ acting on the state space $\mathcal{H}$ of the system. The operator $\hat{\mathcal{O}}$ has a spectral decomposition

$$\hat{\mathcal{O}} = \sum_m m \hat{P}_m, \quad (2.14)$$

where $\hat{P}_m$ are the projectors onto the eigenspace of $\hat{\mathcal{O}}$ with eigenvalue m . The set of eigenvalues represents the range of possible measurement outcomes. By measuring $\mathcal{O}$ on the state $|\psi\rangle$, the probability $p(m)$ of getting the result m is given by

$$p(m) = \langle \psi | \hat{P}_m | \psi \rangle, \quad (2.15)$$

and the state of the system is projected onto the final state

$$|\psi_f\rangle = \frac{\hat{P}_m |\psi\rangle}{\sqrt{p(m)}} \quad (2.16)$$

after the measurement. Take note of how the factor $1/\sqrt{p(m)}$ renormalizes the post-measurement state.

In the Bloch sphere picture, the observable $\mathcal{O}$ defines an axis through the center of the Bloch sphere, and the two intersecting poles represent the two potential outcomes [55]. Unless the quantum state vector perfectly coincides with one of these intersecting poles, the measurement result seems to be one of the axis's two poles at random with respective probability $p(m)$. The computational basis $\{|0\rangle, |1\rangle\}$ is represented by the axis linking the north and south poles of the Bloch sphere (see Figure 2.1). However, measuring along the X axis

which links two points on the the equator yields two possible outputs corresponding to $|+\rangle$ and $|-\rangle$. In practice, such an X axis measurement may be obtained by doing the measurement in the computational basis after a Hadamard rotation to produce the basis shift (from X to Z or vice versa).

Suppose we want to measure the qubit from Equation 2.1 in the computational basis $\{|0\rangle, |1\rangle\}$. The operator $\hat{O}$ is then specifically the combination of the two projectors $\hat{M}_0 = |0\rangle\langle 0|$ and $\hat{M}_1 = |1\rangle\langle 1|$:

$$\hat{O} = \sum_m m \hat{P}_m = \hat{M}_0 - \hat{M}_1 \quad (2.17)$$

The probability to obtain $|0\rangle$ with corresponding eigenvalue 1 as an outcome is given by:

$$p(1) = \langle \psi | \hat{M}_0 | \psi \rangle = |\alpha_0|^2 \quad (2.18)$$

Similarly for $|1\rangle$ with corresponding eigenvalue -1 :

$$p(-1) = \langle \psi | \hat{M}_1 | \psi \rangle = |\alpha_1|^2 \quad (2.19)$$

We can see from Table 2.1 that the operator for this measurement is $\hat{O} = \hat{\sigma}_z \equiv \hat{Z}$. If we were to measure the qubit's state in the $\{|+\rangle, |-\rangle\}$ basis, we would be using the observable described by the operator $\hat{\sigma}_x \equiv \hat{X}$, which is the sum of $\hat{M}_+ = |+\rangle\langle +|$ and $\hat{M}_- = |-\rangle\langle -|$:

$$\hat{\sigma}_x = \hat{M}_+ - \hat{M}_- \quad (2.20)$$

The probability to obtain $|+\rangle$ with corresponding eigenvalue 1 as an outcome is given by

$$p(1) = \langle \psi | \hat{M}_+ | \psi \rangle = |\alpha_+|^2 = \frac{|\alpha_0 + \alpha_1|^2}{2}, \quad (2.21)$$

and similarly for $|-\rangle$ with corresponding eigenvalue -1 (see Table 2.1),

$$p(-1) = \langle \psi | \hat{M}_- | \psi \rangle = |\alpha_-|^2 = \frac{|\alpha_0 - \alpha_1|^2}{2} \quad (2.22)$$

We will be primarily concerned with projective measurements in various applications of QC and quantum information. For a more general treatment of the measurement process, the reader is referred to [41].

2.1.3 The density operator

We have only discussed *pure states* up to this point, which are states that describe systems that are in a state of $|\psi\rangle$ with certainty. When a quantum system is in the state $|\psi_i\rangle$ with a given probability p_i , one speaks of a *mixed state*. Mixed states are statistical mixtures of pure states. A practical method for formally treating mixed states is the *density operator* formalism. If a quantum system is in the state $|\psi_i\rangle$ with the probability p_i , its density operator ρ^1 is

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \text{ with } \sum_i p_i = 1 \quad (2.23)$$

A quantum system is said to be in a pure state if $p_j = 1$ for only one $i = j$ and $p_i = 0$ for all $i \neq j$, and its density operator is simply $\rho = |\psi\rangle\langle\psi|$ in this case. Otherwise, ρ is considered to be a mixture of different pure states. It is important to notice that a density operator is always:

- trace-preserving: $\text{Tr}(\rho) = 1$, where Tr is the trace
- positive semi definite: $\rho \geq 0$, meaning the eigenvalues are non-negative
- self-adjoint: $\rho = \rho^\dagger$

Determining the density matrix representation ρ of an experimentally produced quantum state is critical for assessing various properties and evaluating the quantum state's quality. Two useful examples of such properties are a state's purity P and fidelity F .

The purity P of a state ρ discriminates whether the state is mixed or pure, and is defined as

¹The hat symbol for operators will from now be omitted.

$$P = \text{Tr}(\rho^2) \quad (2.24)$$

Pure states satisfy $P = 1$, while for mixed states $P < 1$ holds.

The fidelity F is a useful metric for determining how similar two quantum states are, or how much they overlap. For a pure state $|\psi\rangle$ and an arbitrary state ρ the fidelity F is defined as

$$F(\rho, |\psi\rangle) = \langle \psi | \rho | \psi \rangle, \quad (2.25)$$

and between two mixed states ρ and $\tilde{\rho}$ the fidelity F is given by

$$F(\rho, \tilde{\rho}) = \left(\text{Tr} \left(\sqrt{\sqrt{\tilde{\rho}} \rho \sqrt{\tilde{\rho}}} \right) \right) \quad (2.26)$$

The density matrix can also be used to calculate values of entropy and the amount of entanglement present in a given state.

2.1.4 Entanglement

Mathematically, an *entangled state* $|\psi\rangle$ cannot be expressed as a tensor product of individual qubit states $|\psi_1\rangle$ and $|\psi_2\rangle$,

$$|\psi\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle \quad (2.27)$$

Unlike two classical bits, which can only exist in one of the four states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ at a time, a pair of qubits can also exist in a superposition $|\Psi_2\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$ of these states, for some $\alpha, \beta, \gamma, \delta \in \mathbb{C}$. *Entanglement* occurs as a natural result of the superposition principle [44]. In particular, the four Bell states

$$|\Phi^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1 \otimes |0\rangle_2 \pm |1\rangle_1 \otimes |1\rangle_2), \quad (2.28)$$

$$|\Psi^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1 \otimes |1\rangle_2 \pm |1\rangle_1 \otimes |0\rangle_2), \quad (2.29)$$

are examples of entangled states. Moreover, they form another orthonormal basis for the two-qubit state space $\mathcal{H}_2$.

A measurement on one of their individual qubits will always return a random result, either $|0\rangle$ or $|1\rangle$. Hence, when analyzed independently

from each other, the individual qubit's state is completely mixed. However, when one performs a joint measurement on one of the four Bell states, the outcomes exhibit perfect correlations (results are always equal to one another) for $|\Phi^\pm\rangle$ and perfect anticorrelations (results are always opposite to one another) for $|\Psi^\pm\rangle$ [44].

Since the famous publication by A. Einstein, B. Podolsky, and N. Rosen [21], who pointed out the non-classical features of entangled states, these correlations have been the subject of considerable theoretical and experimental investigation. John Bell significantly improved the understanding of entangled states by proving that their measurement outcomes reveal stronger correlations than could ever exist between classical systems [9].

In fact, these correlations hold regardless of the measurement basis. The whole information of the composite system is encoded in the correlations that the subsystems have with regard to each other rather than in the states of the individual subsystems [44].

As a result, the behavior of the composite system can only be understood by its joint properties. Determining experimentally whether a given state is entangled or not is not an easy task, especially when dealing with mixed states. Characterizing the type of entanglement in a given system is, however, critical for quantum information experiments [44].

One commonly used measure for a system with two subsystems, A and B , described by a density matrix ρ_{AB} is the concurrence C , defined by

$$C = \max(\sqrt{\lambda_1} - \sqrt{\lambda_2} - \sqrt{\lambda_3} - \sqrt{\lambda_4}, 0), \quad (2.30)$$

where λ_i are the eigenvalues of the matrix $\rho_{AB}\tilde{\rho}_{AB}$ in decreasing order with

$$\tilde{\rho}_{AB} = (\sigma_y \otimes \sigma_y) \rho_{AB}^* (\sigma_y \otimes \sigma_y) \quad (2.31)$$

and ρ_{AB}^* being the complex conjugate of ρ_{AB} . A concurrence of $C = 1$ corresponds to a maximally entangled state, whereas $C = 0$ corresponds to a separable state.

SLOCC class	States
completely seperable 1-2-3	$ \psi_1\rangle \otimes \psi_2\rangle \otimes \psi_1\rangle$
biseparable 1-23	$ \psi_1\rangle \otimes \psi_{23}\rangle$
biseparable 3-12	$ \psi_3\rangle \otimes \psi_{12}\rangle$
biseparable 2-13	$ \psi_2\rangle \otimes \psi_{13}\rangle$
genuinely tripartite entangled W	$ W_3\rangle$
genuinely tripartite entangled GHZ	$ GHZ_3\rangle$

Table 2.2: Six SLOCC classes can be distinguished for three qubits. The fully separable state $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$, three biseparable classes in which two qubits are entangled but separated from the third (1-23, 3-12, 2-13), and two classes with genuine tripartite entanglement, namely $|GHZ_3\rangle$ and $|W_3\rangle$.

For bipartite systems, i.e., systems composed of two subsystems, entanglement and its measures can be defined in a standardized way. This definition, however, can lead to ambiguities and requires more careful treatment for systems with multiple subsystems.

Multipartite entanglement refers to entangled systems with more than two subsystems. Multi-qubit systems, as opposed to two-qubit systems, can have different types of entanglement, revealing distinct properties such as robustness against decoherence [44].

A useful framework is the classification of multi-qubit entanglement based on stochastic local operations and classical communication (SLOCC), in which distinct entanglement classes are characterized by their invariance with respect to local operations. With three qubits, there is already more than one type of entanglement, and six SLOCC classes can be distinguished [44], as categorized in Table 2.2:

the fully separable state $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle$, three biseparable classes in which two qubits are entangled but separated from the third, and two classes with genuine tripartite entanglement, namely $|GHZ_3\rangle$ (named after D. Greenberger, M. Horne, and A. Zeilinger) and $|W_3\rangle$.

The $|\text{GHZ}_3\rangle$ state, for example, violates locality and realism more than the $|\text{W}_3\rangle$ state, whereas W-states are more robust to qubit loss, retaining some entanglement even if one of the qubits is measured². In contrast, measuring any of the qubits in a GHZ state completely disentangles the remaining state [44].

$$|\text{W}_3\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle) \quad (2.32)$$

$$|\text{GHZ}_3\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \quad (2.33)$$

With four qubits, an infinite number of SLOCC classes can be distinguished. However, these classes can be divided into nine SLOCC families, which correspond to nine different types of entanglement in the case of four qubits [44].

2.2 The quantum circuit model

The quantum circuit model is frequently compared with the boolean (electrical) circuit model in classical computing. Here, electronic circuits are made up of wires that contain data as strings of bits (for example, "01001...110") and binary logic gates that manipulate those bits. Modern CPUs implement bits in the physical state of transistors. By combining electrical components on a chip, different logic gates such as NOT, AND, OR, XOR, NAND, and so on can be constructed.

Qubits can be realized by utilizing two-level quantum systems. Section 3.1 goes into greater detail about how photons are used as qubits. Quantum logic gates are made up of unitary operations, and a set of physical actions on qubits is used to implement these operations. When compared to classical computers, this allows for an entirely different approach to processing information in a quantum computer. The basic building blocks of a quantum circuit are single- and two-qubit gates, which are introduced in the subsections below based on [41] and [8] unless otherwise noted.

²Measuring one qubit of a $|\text{W}_3\rangle$ state: if the outcome is 1, the remaining state is fully separable; if the outcome is 0, the remaining state is biseparable.

2.2.1 Single-qubit gates

A single-qubit gate is a unitary operation U that acts on one qubit:

$$|\psi'\rangle = U|\psi\rangle \quad (2.34)$$

The single-qubit unitary operation transforms the qubit's state from

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \rightarrow |\psi'\rangle = \alpha'_0|0\rangle + \alpha'_1|1\rangle \quad (2.35)$$

In the quantum circuit picture, the single-qubit gate is illustrated as

$$|\psi\rangle \text{ --- } \boxed{U} \text{ --- } |\psi'\rangle$$

and in matrix notation, it becomes a 2×2 unitary matrix:

$$\begin{pmatrix} \alpha'_0 \\ \alpha'_1 \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \quad (2.36)$$

The unitarity follows from the condition that the output state must be another valid single-qubit state, and therefore, the operation must preserve the norm.

The Pauli operators σ_x , σ_y , σ_z are important examples of single-qubit gates. Other important examples are the Hadamard gate (H), the phase gate (S), and the $\pi/8$ gate (T). A list of commonly used single-qubit gates in QC is given in Table 2.3. The H gate, for example, transforms $|0\rangle \rightarrow |+\rangle$ and $|1\rangle \rightarrow |-\rangle$. It thus turns computational basis states into equally weighted superposition states and vice versa. The S gate adds a $\pi/2$ phase to the basis state $|1\rangle$, while the T gate adds a $\pi/8$ phase to the basis state $|1\rangle$.

All single-qubit gates can be represented geometrically on the Bloch sphere. The H gate, for example, rotates the state vector 180 deg around the $X + Z$ axis. This effectively produces the superposition states from the computational basis states, as shown in Figure 2.2.

Some useful gate identities are:

$$HXH = Z \quad (2.37)$$

$$HZH = X \quad (2.38)$$

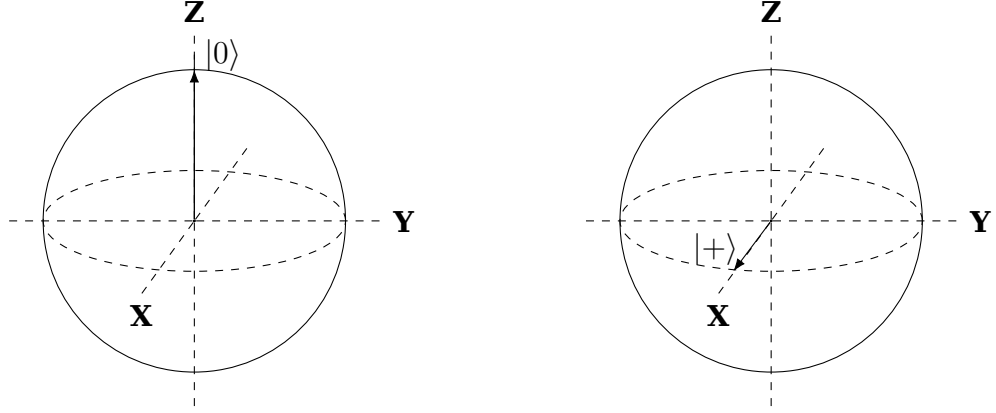


Figure 2.2: On the Bloch sphere, the Hadamard gate rotates the $|0\rangle$ state to the $|+\rangle$ state: The state before applying a Hadamard gate is shown on the left. Right, the state after applying a Hadamard gate.

$$H = (X + Z)/\sqrt{2} \quad (2.39)$$

$$T^2 = S \quad (2.40)$$

$$H^2 = \mathbb{1} \quad (2.41)$$

When the Pauli matrices are potentiated, they generate the following three rotation operators around the X , Y , and Z axes of the Bloch sphere:

$$R_z(\theta) \equiv e^{-i\theta Z/2} \quad (2.42)$$

$$R_x(\theta) \equiv e^{-i\theta X/2} \quad (2.43)$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} \quad (2.44)$$

Pauli gates can also be used to express a rotation around an arbitrary axis $\vec{n} = (n_x, n_y, n_z)$:

$$R_{\vec{n}}(\theta) \equiv e^{-i\theta \vec{n} \cdot \vec{\sigma}/2} \quad (2.45)$$

$\vec{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$ is a vector of matrices and the $\cdot$ represents the inner product of the two vectors.

Single-qubit gate	Matrix
$\sigma_z \equiv Z$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$\sigma_x \equiv X$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$\sigma_y \equiv Y$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
H	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$
$R_z(\theta)$	$\begin{pmatrix} e^{-\frac{\theta}{2}} & 0 \\ 0 & e^{\frac{\theta}{2}} \end{pmatrix}$
$R_x(\theta)$	$\begin{pmatrix} \cos\frac{\theta}{2} & -i \sin\frac{\theta}{2} \\ -i \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
$R_y(\theta)$	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
$\mathbb{1} \equiv Identity$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Table 2.3: A table of commonly used single-qubit gates in QC.

2.2.2 Two-qubit gates

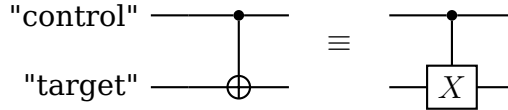
Analogously, two-qubit gates take two qubits as input and perform unitary operations on them. Two-qubit gates are essential for QC, and operations can be conditioned on the state of one of the input qubits. One prominent example is the controlled-NOT (CNOT) gate, or controlled- X (CX) gate. The CNOT gate takes two qubits, i and j , and does the following operation:

$$\text{CNOT} |i\rangle |j\rangle = |i\rangle |i \oplus j\rangle, \quad (2.46)$$

where $\oplus$ stands for addition modulo two. If the control qubit i is in the state $|1\rangle$, the target qubit's state is changed from $|0\rangle$ to $|1\rangle$ (or vice versa), otherwise the target qubit j remains unchanged. The following truth table summarizes the operation of the CNOT gate on the different input states:

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |11\rangle \\ |11\rangle &\rightarrow |10\rangle \end{aligned} \quad (2.47)$$

In a quantum circuit, the CNOT gate is illustrated by the symbol:



The top wire represents the control qubit, and the bottom wire represents the target qubit, on which the operation is conditionally performed.

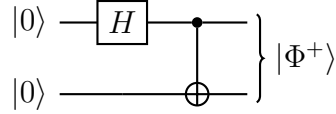
The matrix representation of the CNOT gate with respect to the computational basis is

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.48)$$

Consider a two-qubit register that begins in the state $|00\rangle$ and a circuit that applies an H gate to the first qubit before applying the CNOT gate to both qubits. This results in:

$$\begin{aligned} \text{CNOT}_{12}(H_1 \otimes \mathbb{1}_2)|00\rangle &= \text{CNOT}_{12}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle\right) \\ &= \text{CNOT}_{12}\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\Phi^+\rangle, \end{aligned} \quad (2.49)$$

where the subscripts indicate actions on the first and second qubits, respectively. This quantum circuit of an H gate and a CNOT gate generates the entangled state $|\Phi^+\rangle$, which is one of the four Bell states (see Equation 2.28). In the quantum circuit picture, this looks like:



In fact, this quantum circuit effectively entangles (or disentangles) two qubits.

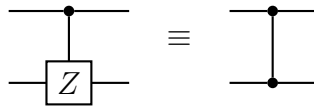
Another important two-qubit gate is the controlled-phase gate (CZ). The CZ gate applies the following operation:

$$\text{CZ} |i\rangle|j\rangle = (-1)^{ij} |i\rangle|j\rangle \quad (2.50)$$

The input qubits acquire a phase of -1 if they are in the state $|11\rangle$. The truth table for the CZ looks like this:

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |01\rangle &\rightarrow |01\rangle \\ |10\rangle &\rightarrow |10\rangle \\ |11\rangle &\rightarrow -|11\rangle \end{aligned} \quad (2.51)$$

In the quantum circuit diagram, the CZ gate is represented by the symbol:

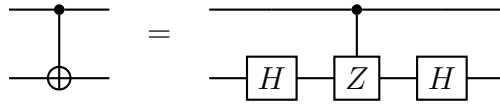


The CZ gate is represented by the unitary matrix

$$\text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (2.52)$$

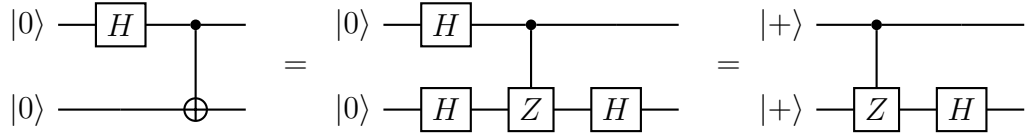
The CNOT and the CZ gates are examples of two-qubit entangling gates, as they can produce entanglement between two unentangled qubits but also disentangle two qubits.

The two gates are also closely related. Consider the quantum circuit identity below, which includes additional H gates acting on the target qubit before and after the CZ gate [44]:



$$\text{CNOT}_{12} = (\mathbb{1}_1 \otimes H_2) \text{CZ}_{12} (\mathbb{1}_1 \otimes H_2) \quad (2.53)$$

As a result, we can generate an entangled state by initializing two separable qubits in the $|+\rangle$ state and applying the CZ gate and an additional H gate on the second qubit. In circuit notation, this becomes:



A quantum computer works mathematically by applying a $2^n \times 2^n$ unitary matrix U to a vector of 2^n coefficients representing n quantum bits. Any such unitary matrix U can be decomposed into a series of single- and two-qubit gates, with single-qubit gates performing local unitary operations and two-qubit (entangling) gates generating entanglement between two qubits [55].

A universal set of gates allows for the implementation of an arbitrary unitary matrix. The set $\{T, H, \text{CNOT}\}$, for example, constitutes a universal gate set capable of implementing any unitary operation with arbitrary precision.

Another example is the CNOT gate and an arbitrary single-qubit rotation, $e^{i\alpha} R_{\hat{n}}(\theta)$, i.e., $\{e^{i\alpha} R_{\hat{n}}(\theta), \text{CNOT}\}$.

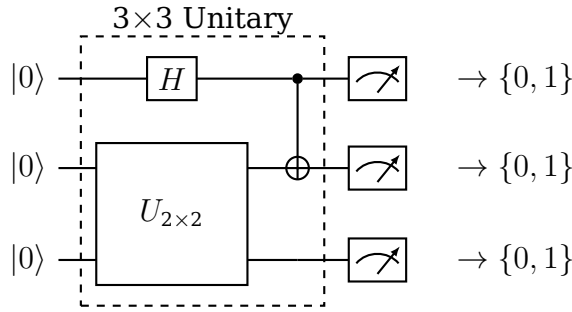
Since all quantum gates are unitary, they are also reversible:

$$U^\dagger U = \mathbb{1} \iff U^{-1} = U^\dagger \quad (2.54)$$

Similarly, in classical computing, any classical function can be computed using $\{\text{AND}, \text{OR}, \text{NOT}\}$, resulting in a universal gate set; however, those gates are not reversible.

2.2.3 Measurements

A qubit must be measured in order to read out its state. Quantum state measurements cannot be reversed, in contrast to unitary gate operations. Measurements produce probabilistic results, and the distribution of those results depends on the measurement basis (the projection axis on the Bloch sphere) [55]. In the quantum circuit picture, the *meter symbol* represents a measurement in the computational basis:



2.3 The one-way model of quantum computing

The quantum circuit model processes quantum information with quantum logic gates, which perform coherent unitary operations on qubits. The *one-way model* of QC, however, achieves universal quantum computation through a sequence of adaptive

single-qubit measurements on a highly entangled multipartite state, a so-called *cluster state* [47]. The one-way model is thus an entirely *measurement-based* model of QC (MBQC). Both models, the quantum circuit model and the one-way model, are computationally equivalent [8, 13]. In the following sections, we will focus on the one-way model of quantum computation.

2.3.1 Graph states and cluster states

Graph states are multipartite quantum states that are highly entangled and can be represented by a mathematical graph. A graph G is made up of vertices V , and edges E that connect them. Each such graph corresponds to an N -qubit quantum state, with vertices representing physical qubits and edges denoting entanglement between qubits [8].

A graph state $|G\rangle$ is defined mathematically by a set of eigenvalue equations [44],

$$K_G^{(a)}|G\rangle = (-1)^{k_a}|G\rangle, \quad (2.55)$$

with $K_G^{(a)}$ as an operator for each vertex,

$$K_G^{(a)} = \sigma_x^{(a)} \bigotimes_{b \in N(a)} \sigma_z^{(b)} \quad (2.56)$$

$N(a)$ refers to all vertices in the neighborhood of vertex a , i.e., all nearest vertices b connected to a by an edge.

The resource state for the one-way quantum computer is a type of graph state known as a *cluster state*, in which the underlying graph structure forms a two-dimensional lattice (see Figure 2.3).

An arbitrary two-dimensional (2D) cluster state can thus be expressed as

$$|\mathcal{C}_{2D}\rangle = \bigotimes_{a \in \mathcal{C}} \left(|0\rangle_a \bigotimes_{\gamma \in \Gamma} \sigma_z^{(a+\gamma)} + |1\rangle_a \right), \quad (2.57)$$

with $\Gamma = \{(1, 0), (0, 1)\}$ and the convention $\sigma_z^{(a+\gamma)} \equiv 1$ for $a + \gamma \notin \mathcal{C}$.

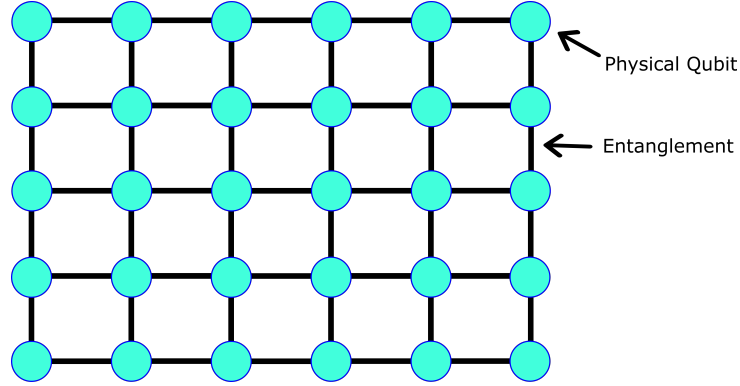


Figure 2.3: Visualization of a 2D cluster state where vertices (cyan) represent physical qubits and edges (black) indicate entanglement.

For a one-dimensional chain of N qubits this becomes [14]

$$|\mathcal{C}_N\rangle = \frac{1}{2^{N/2}} \bigotimes_{a=1}^N \left(|0\rangle_a \sigma_z^{(a+1)} + |1\rangle_a \right) \quad (2.58)$$

For $N = 2$ this yields:

$$\begin{aligned} |\mathcal{C}_2\rangle &= \frac{1}{2} (|0\rangle_1 \sigma_z^{(2)} + |1\rangle_1) \otimes (|0\rangle_2 + |1\rangle_2) \\ &= \frac{1}{2} (|0\rangle_1 \otimes (|0\rangle_2 - |1\rangle_2) + |1\rangle_1 \otimes (|0\rangle_2 + |1\rangle_2)) \\ &= \frac{1}{\sqrt{2}} (|0\rangle_1 \otimes |-\rangle_2 + |1\rangle_1 \otimes |+\rangle_2) \\ &\equiv \frac{1}{\sqrt{2}} (|0\rangle_1 |-\rangle_2 + |1\rangle_1 |+\rangle_2) \end{aligned} \quad (2.59)$$

$|\mathcal{C}_2\rangle$ is locally equivalent to the Bell state $|\Psi^+\rangle$ under the unitary transformation $\mathbb{1}_1 \otimes H_2$,

$$|\Psi^+\rangle = (\mathbb{1}_1 \otimes H_2) |\mathcal{C}_2\rangle, \quad (2.60)$$

and locally equivalent up to the unitary transformation $\mathbb{1}_1 \otimes X_2 H_2$ to the Bell state $|\Phi^+\rangle$,

$$|\Phi^+\rangle = (\mathbb{1}_1 \otimes X_2 H_2) |\mathcal{C}_2\rangle \quad (2.61)$$

Thus,

$$|\mathcal{C}_2\rangle \stackrel{l.u.}{=} |\Phi^+\rangle, \quad (2.62)$$

where *l.u.* indicates equality up to a local unitary transformation on one or more qubits.

Similarly, for $N = 3, 4$:

$$\begin{aligned} |\mathcal{C}_3\rangle &= \frac{1}{\sqrt{2}}(|+\rangle_1|0\rangle_2|-\rangle_3 - |-\rangle_1|1\rangle_2|+\rangle_3) \\ &\stackrel{l.u.}{=} \frac{1}{\sqrt{2}}(|0\rangle_1|0\rangle_2|0\rangle_3 + |1\rangle_1|1\rangle_2|1\rangle_3) \\ &\equiv |\text{GHZ}_3\rangle \end{aligned} \quad (2.63)$$

$$|\mathcal{C}_4\rangle \stackrel{l.u.}{=} \frac{1}{2}(|0\rangle|0\rangle|0\rangle|0\rangle + |0\rangle|0\rangle|1\rangle|1\rangle + |1\rangle|1\rangle|0\rangle|0\rangle - |1\rangle|1\rangle|1\rangle|1\rangle) \quad (2.64)$$

While $|\mathcal{C}_3\rangle$ corresponds to a GHZ state of three qubits, $|\mathcal{C}_4\rangle$ is not equivalent to a four-qubit GHZ state $|\text{GHZ}_4\rangle^3$. As a result, local unitary operations cannot transform $|\mathcal{C}_4\rangle$ into a four-qubit GHZ state, and the entanglement properties of the two states differ [14, 44].

Figure 2.4 shows different cluster states. The horseshoe cluster is a two-dimensional cluster that can implement two-qubit logic gates [8].

One way to prepare a graph state $|G\rangle$ is to initialize each physical qubit in the $|+\rangle$ state for each vertex and apply a CZ gate between all connected qubits. This recipe can be mathematically expressed as

$$|G\rangle = \prod_{(a,b) \in E} \text{CZ}_{ab} |+\rangle^{\otimes V}, \quad (2.65)$$

with V representing the number of vertices and E representing the set of edges that comprise the graph $G(V, E)$.

Several experimental techniques to grow large cluster states have been developed and cluster states have been realized on multiple physical platforms including superconducting circuits [2], trapped ions [31], and photons [45, 52]. Recently, cluster states of up to 14 photonic qubits have been demonstrated [51].

³ $|\text{GHZ}_4\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$

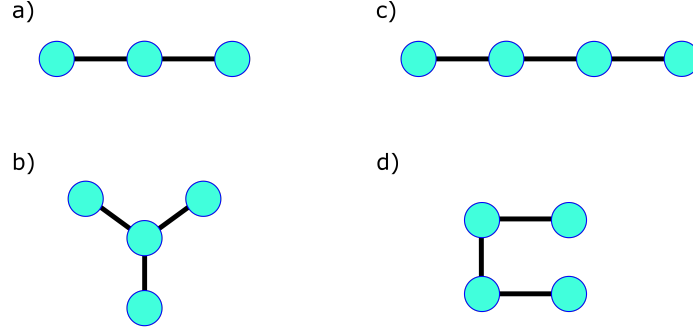


Figure 2.4: Overview of different cluster states. a) shows a three-qubit GHZ state $|\text{GHZ}_3\rangle$, b) shows a four-qubit GHZ state $|\text{GHZ}_4\rangle$, c) a four-qubit linear cluster state, and d) a so-called horseshoe cluster.

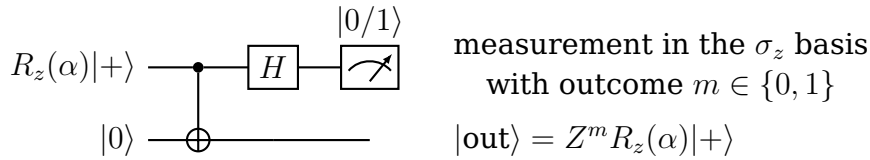
2.3.2 One-way quantum computation

In contrast to the coherent unitary operations in the standard circuit model of QC, the one-way quantum computer achieves universal quantum computation through a series of single-qubit measurements on a highly entangled cluster state and active feed-forward. The 2D cluster state is a universal resource for one-way QC in the sense that any quantum logic can be implemented on a sufficiently large and appropriately structured cluster state [8, 44].

Measuring a qubit j of a linear one-dimensional cluster in the basis $\mathcal{B}_j(\alpha) = \{|\pm\alpha\rangle_j, |-\alpha\rangle_j\}$,

$$|\pm\alpha\rangle_j = \frac{1}{\sqrt{2}}(|0\rangle_j \pm e^{-i\alpha}|1\rangle_j), \quad (2.66)$$

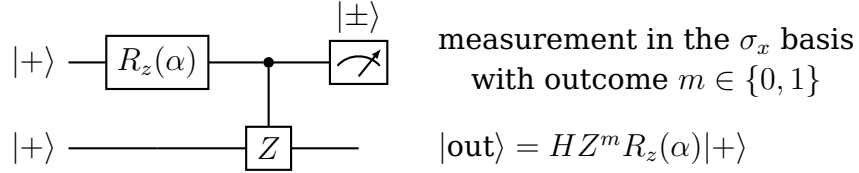
effectively performs a $HR_z(\alpha)$ operation on an encoded logical qubit. Consider the following teleportation circuit⁴:



⁴The derivation follows the steps in [8] and [44].

The measurement on the first qubit in the σ_z basis gives the result $m \in \{0, 1\}$. Depending on the outcome, a Pauli $\sigma_z \equiv Z$ operation needs to be applied to the second qubit.

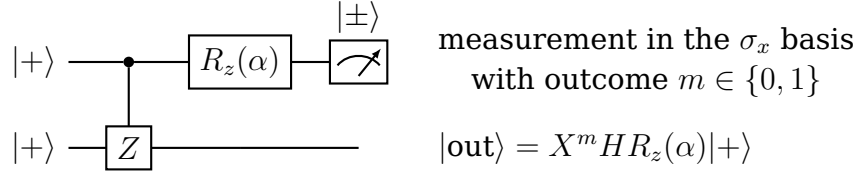
We now use Equation 2.53 to rewrite this circuit as:



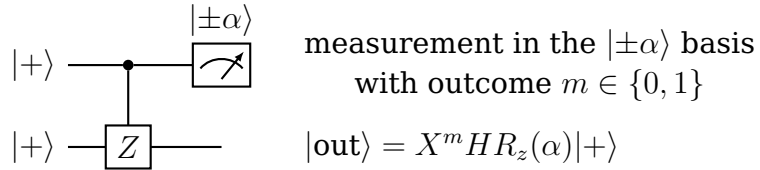
Here, we placed $R_z(\alpha)$ within the circuit and absorbed the H gates in the measurement basis and the target input qubit state ($|0\rangle \rightarrow |+\rangle$). Using Equation 2.38 we obtain the following equality:

$$|\text{out}\rangle = H Z^m R_z(\alpha) |+\rangle = X^m H R_z(\alpha) |+\rangle \quad (2.67)$$

Both the $R_z(\alpha)$ rotation and the CZ gate are diagonal in the computational basis, and thus they commute, which yields:



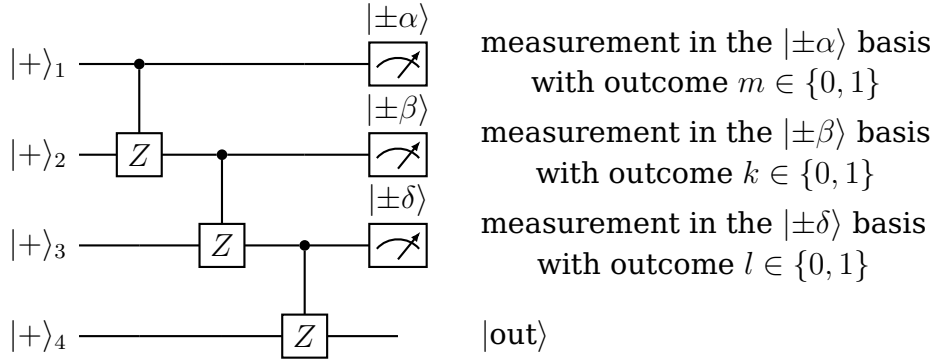
Absorbing $R_z(\alpha)$ into the measurement basis results in a measurement along an axis in the Bloch sphere's equatorial X - Y plane, defined by $|\pm\alpha\rangle$:



As a result, measuring the first qubit in the basis $\mathcal{B}(\alpha) = \{|+\alpha\rangle, |-\alpha\rangle\}$ effectively implements a single-qubit rotation $H R_z(\alpha)$ on the state $|+\rangle$ up to a Pauli X correction and teleports the state onto the second qubit. These equatorial measurements process quantum information, whereas

measurements in the computational basis $\{|0\rangle_j, |1\rangle_j\}$ are used to change the shape of the cluster by removing qubit j from the cluster [8, 44].

If the one-dimensional cluster state is large enough (at least four qubits), this scheme may be extended to implement an arbitrary single-qubit rotation via a series of single-qubit measurements:



The output state $|\text{out}\rangle$ of this circuit is, therefore⁵,

$$|\text{out}\rangle = \underbrace{X^l H R_z(\delta)}_{\text{meas. in } \mathcal{B}_3(\delta)} \underbrace{X^k H R_z(\beta)}_{\text{meas. in } \mathcal{B}_2(\beta)} \underbrace{X^m H R_z(\alpha)}_{\text{meas. in } \mathcal{B}_1(\alpha)} |+\rangle_1 \quad (2.68)$$

Now, using $ZX = -XZ$ we can proof

$$\begin{aligned} R_z(\theta)X &= \sum_{n=0}^{\infty} \left(\frac{i\theta}{2}\right)^n \left(\frac{Z^n}{n!}\right)X \\ &= \sum_{n=0}^{\infty} \left(\frac{-i\theta}{2}\right)^n \left(\frac{XZ^n}{n!}\right) = XR_z(-\theta), \end{aligned} \quad (2.69)$$

and therefore we can commute the Pauli X gates through to the left, such that:

⁵Note how the state $|+\rangle_1$ is teleported through the circuit.

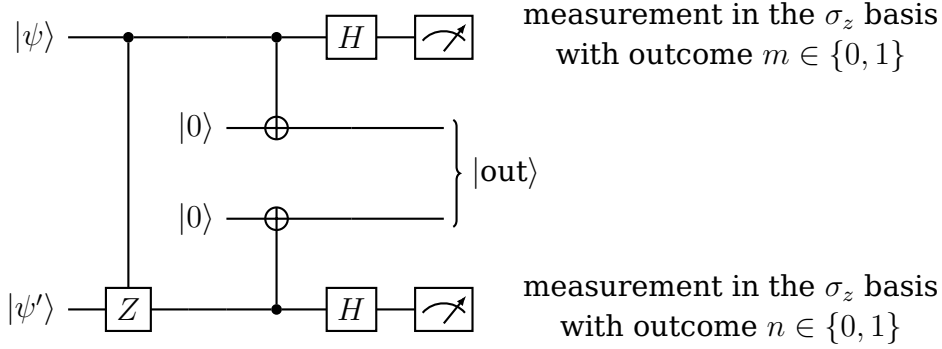
$$\begin{aligned}
|\text{out}\rangle &= X^l H \underbrace{R_z(\delta) X^k}_{X^k R_z((-1)^k \delta)} H \underbrace{R_z(\beta) X^m}_{X^m R_z((-1)^k \beta)} H R_z(\alpha) |+\rangle_1 \\
&= X^l H X^k R_z((-1)^k \delta) H X^m \underbrace{\mathbb{1}}_{HH} R_z((-1)^m \beta) H R_z(\alpha) |+\rangle_1 \\
&= X^l H X^k R_z((-1)^k \delta) \underbrace{H X^m H}_{Z^m} H R_z((-1)^m \beta) H R_z(-\alpha) |+\rangle_1 \\
&= X^l H X^k \underbrace{R_z((-1)^k \delta) Z^m}_{Z^m R_z((-1)^k \delta)} H R_z((-1)^m \beta) H R_z(-\alpha) |+\rangle_1 \quad (2.70) \\
&= X^l H X^k \underbrace{Z^m}_{H X^m H} R_z((-1)^k \delta) H R_z((-1)^m \beta) H R_z(-\alpha) |+\rangle_1 \\
&= X^l \underbrace{H X^k H}_{Z^k} X^m H R_z((-1)^k \delta) \underbrace{H R_z((-1)^m \beta) H}_{R_x((-1)^m \beta)} R_z(-\alpha) |+\rangle_1 \\
&= X^l Z^k X^m H R_z((-1)^k \delta) R_x((-1)^m \beta) R_z(-\alpha) |+\rangle_1
\end{aligned}$$

Since an arbitrary single-qubit rotation can be decomposed into rotations around the X and Z axes using a combination of three Euler angles $R_z(\theta)R_x(\eta)R_z(\zeta)$, the three subsequent measurements on the four-qubit linear cluster's qubits 1, 2, and 3 effectively achieve any single-qubit operation on the encoded logical qubit $|+\rangle_1$ up to an H gate (see Figure 2.5) [8, 44].

However, the desired operation is only carried out if the measurement outcomes are $m = k = l = 0$. Because quantum measurement is inherently probabilistic, the measurement outcomes $m, k, l = 1$ occur with some probability, and the computation proceeds in an unwanted way. These intrinsic errors are accounted for by applying Pauli corrections and adjusting subsequent measurement bases according to previous outcomes, a process known as *feed-forward*. The feed-forward mechanism also determines the computation's temporal orientation [8, 44].

For universal QC, two-qubit entangling gates are necessary. Here, the CZ gate comes in handy, as it is already inherently employed in cluster state generation (see Equation 2.65). We will now look at how vertical edges in a 2D cluster can implement CZ gates⁶.

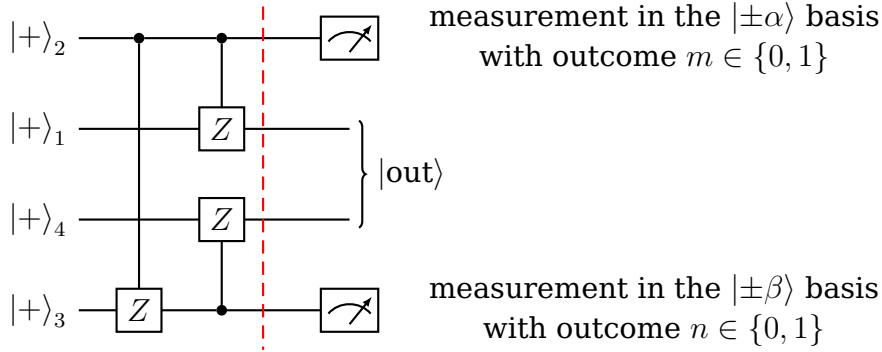
⁶The derivation follows the steps in [8].



The circuit above teleports the state $CZ|\psi\rangle|\psi'\rangle$ up to local Pauli corrections onto the output state

$$|out\rangle = (Z^m \otimes Z^n)CZ|\psi\rangle|\psi'\rangle \quad (2.71)$$

Setting $|\psi\rangle = R_z(\alpha)|+\rangle$ and $|\psi'\rangle = R_z(\beta)|+\rangle$, and following similar calculations as in the one-dimensional case, this circuit transforms into:



This circuit prepares a cluster state (part before the dashed red line), namely the horseshoe cluster seen in Figure 2.4 by applying CZ gates to the four qubits in the $|+\rangle$ state. Subsequent measurements in the bases $\mathcal{B}_2(\alpha)$ and $\mathcal{B}_3(\beta)$ on the qubits 2 and 3, respectively, carry out the computation on the two encoded logical qubits (see Figure 2.6).

As a result, the output state is

$$\begin{aligned} |out\rangle &= (H \otimes H)(Z^m \otimes Z^n)(R_z(\alpha) \otimes R_z(\beta))CZ|+\rangle|+\rangle \\ &= (X^m \otimes X^n)(H \otimes H)(R_z(\alpha) \otimes R_z(\beta))CZ|+\rangle|+\rangle \end{aligned} \quad (2.72)$$

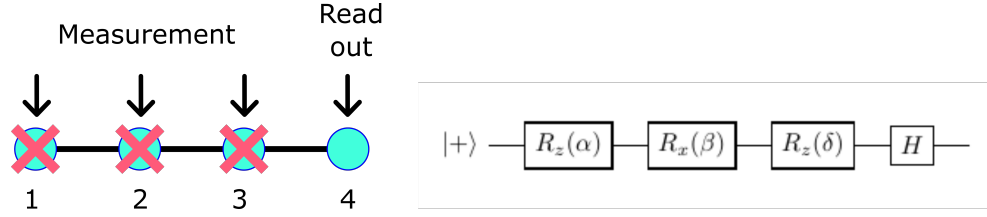


Figure 2.5: The sequential measurements of qubits 1, 2, and 3 of the four-qubit linear cluster (indicated by red crosses) effectively implement the single-qubit rotation $HR_z(\delta)R_x(\beta)R_z(\alpha)$ on the logical input state $|+\rangle$. The logical qubit is teleported through the cluster and can be read out from qubit 4.

This verifies the implementation of CZ gates on encoded logical qubits by the vertical edges that connect the one-dimensional cluster chains in a 2D cluster [8]. In fact, 2D cluster states are a universal resource for MBQC.

Note that other graph structures⁷ are known to be universal and 3D lattices in particular allow for the implementation of fault tolerant topological codes [13].

Summarizing, after the cluster state is prepared, the actual computation is carried out by a series of single-qubit measurements, with the measurement basis for each qubit depending on the outcomes from previous measurements. The horizontal chains in a 2D cluster encode logical qubits, and the vertical edges implement CZ gates between them. Since the measurement procedure is irreversible, the computation is *one-way*. Moreover, every quantum circuit can be converted from the standard circuit model into a measurement pattern on a universal resource state [8, 44].

⁷triangular, hexagonal, kagome [13]

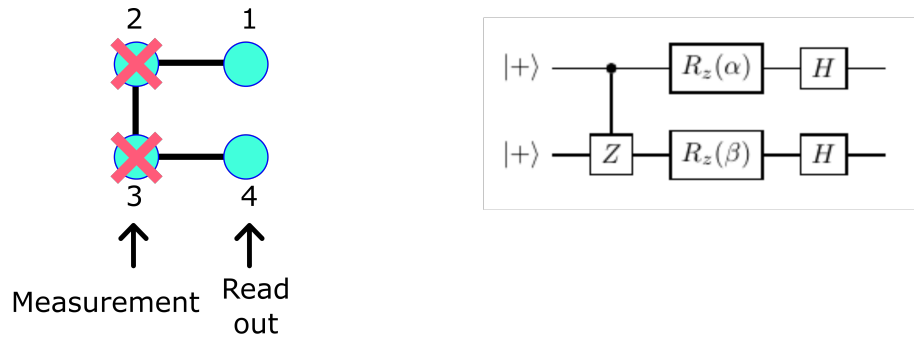


Figure 2.6: The horseshoe cluster's vertical edge implements a CZ operation, and measurements on qubits 2 and 3 (indicated by red crosses) implement rotations $HR_z(\alpha)$ and $HR_z(\beta)$ on the two encoded logical qubits, respectively. The encoded logical qubits are teleported onto qubits 1 and 4, from which the resulting logical state can be read out.

Chapter 3

Photonic quantum computing

“I don’t think there will be one qubit platform in 2050, there will be quite a few. And there might even be platforms that we haven’t thought of yet. In the existing platforms, there are 3-4 which are very promising, and several others are catching up. It’s a race.”

Rainer Blatt, *Professor at
University of Innsbruck*

3.1 Photons as qubits

Photons are excellent carriers of quantum information because they show low decoherence and may be easily transmitted over large distances using optical fibers. Furthermore, photonic systems can be manipulated extremely fast and accurately using current photonic technologies. Experiments have demonstrated the efficient creation of pure, and indistinguishable single-photons, up to the detection of 14 photons in different spatial modes [51, 53].

Logical state	Polarization state	Axis
$ 0/1\rangle$	$ H/V\rangle$	Z
$ +/-\rangle$	$ D/A\rangle \equiv 1/\sqrt{2}(H\rangle \pm V\rangle)$	X
$ +_i/-_i\rangle$	$ R/L\rangle \equiv 1/\sqrt{2}(H\rangle \pm i V\rangle)$	Y

Table 3.1: The horizontal and vertical polarization states $|H/V\rangle$ define the logical $|0/1\rangle$ states. Similarly, $|D/A\rangle$ represent the $|+/-\rangle$ states, and $|R/L\rangle$ represent the $|+_i/-_i\rangle$ states. **Z**, **X**, and **Y** label the corresponding orthogonal axes in the Bloch sphere picture.

There are many different ways to realize photonic qubits. Most commonly, quantum information is encoded in non-overlapping propagation paths or orthogonal polarizations; however, transverse spatial, frequency mode, time bin, and temporal degrees of freedom can act as a qubit [49].

In this work, the two orthogonal polarization states of single-photons, horizontal $|H\rangle$ and vertical $|V\rangle$, are used as qubits. $|H\rangle$ and $|V\rangle$ represent the computational bases $|0\rangle$ and $|1\rangle$, respectively:

$$|0\rangle \equiv |H\rangle \quad |1\rangle \equiv |V\rangle \quad (3.1)$$

Right and left circularly polarized light states are denoted by $|R\rangle$ (right) and $|L\rangle$ (left), while linearly polarized light at $\pm 45^\circ$ is denoted by $|D\rangle$ (diagonal, $+45^\circ$) and $|A\rangle$ (antidiagonal, -45°).

Table 3.1 displays the relationship between the polarization states with their associated logical states and axes on the Bloch sphere. **Z**, **X**, and **Y** label the corresponding orthogonal axes [44].

Polarization states can be described by a 2D vector $\vec{J}$ in the Jones formalism,

$$\vec{J} = \begin{pmatrix} a_H e^{i\phi_H} \\ a_V e^{i\phi_V} \end{pmatrix}, \quad (3.2)$$

where a_H and a_V are the amplitudes of the wave vector in the horizontal and vertical directions, with phases ϕ_H and ϕ_V , respectively [8].

3.2 Single-qubit gates

Uniaxial, birefringent crystals can be used to build single-qubit gates for polarization-encoded qubits. When light passes through such a crystal, it experiences different refractive indices for its ordinary (o) and extraordinary (e) polarization components. The extraordinary polarization is aligned parallel to the plane spanned by the optical axis (OA) and the $\vec{k}$ vector of the pump beam, while the ordinary polarization lies perpendicular to that plane. After passing through a crystal of thickness d , the two perpendicular polarization components acquire a phase difference of

$$\Delta\varphi = \frac{2\pi}{\lambda}d|n_e - n_o|, \quad (3.3)$$

where λ is the wavelength and n_e and n_o are the refractive indices for the extraordinary (e) and ordinary (o) polarization components, respectively. As a result, the polarization of the incident light is changed [8, 44].

The fast axis is the axis along which the phase velocity is fastest, whereas the slow axis is the axis along which the phase velocity is slowest. When the fast axis of a crystal is oriented horizontally, the vertical component of the light experiences a phase shift of $-\phi$. In the Jones formalism, this transformation is expressed by the matrix [8]

$$T(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\phi} \end{pmatrix} \quad (3.4)$$

If the crystal's fast axis is oriented along an angle Θ with respect to the horizontal axis, the transformation matrix can be calculated using rotation matrices $R(\Theta)$ ¹:

$$\underbrace{R(\Theta)T(\phi)R(-\Theta)}_{T'(\phi)} = \begin{pmatrix} \cos^2(\Theta) + e^{-i\phi}\sin^2(\Theta) & (1 - e^{-i\phi})\cos(\Theta)\sin(\Theta) \\ (1 - e^{-i\phi})\cos(\Theta)\sin(\Theta) & e^{-i\phi}\cos^2(\Theta) + \sin^2(\Theta) \end{pmatrix} \quad (3.5)$$

¹The angle Θ corresponds to a rotation in the physical 2D space in the laboratory, not a logical rotation on the Bloch sphere. In the Bloch sphere picture, the rotation about an angle Θ is twice as large.

Wave plate	Θ in deg	Single-qubit gate
HWP	0 deg	Z
HWP	45 deg	X
HWP	22.5 deg	H
QWP	0 deg	$\begin{pmatrix} 1+i & 0 \\ 0 & 1-i \end{pmatrix}$
QWP	90 deg	$\begin{pmatrix} 1-i & 0 \\ 0 & 1+i \end{pmatrix}$

Table 3.2: HWP and QWP implement different single-qubit gates for different values of Θ .

The most important crystals for building single-qubit gates are known as half-wave plates (HWP) and quarter-wave plates (QWP). Given the correct plate thickness d , a HWP introduces a phase shift of $\phi = \pi$ while a QWP introduces a phase shift of $\phi = \pi/2$. The transformations by a HWP (U_{HWP}) and QWP (U_{QWP}) in the Jones formalism are as follows [8, 44]:

$$U_{HWP}(\Theta) = \begin{pmatrix} \cos(2\Theta) & \sin(2\Theta) \\ \sin(2\Theta) & -\cos(2\Theta) \end{pmatrix} \quad (3.6)$$

$$U_{QWP}(\Theta) = \begin{pmatrix} 1 + i \cos(2\Theta) & i \sin(2\Theta) \\ i \sin(2\Theta) & 1 - i \cos(2\Theta) \end{pmatrix} \quad (3.7)$$

For different choices of Θ , different quantum logic gates can be directly implemented on polarization-encoded qubits with HWPs and QWPs.

Table 3.2 provides some values for Θ for HWP and QWP, as well as the implemented single-qubit gate for polarization-encoded qubits.

Combining multiple wave plates allows the realization of every possible single-qubit gate on polarization-encoded qubits. This is typically accomplished using a HWP sandwiched between two QWPs:

$$U = U_{QWP}(\Theta_3)U_{HWP}(\Theta_2)U_{QWP}(\Theta_1) \quad (3.8)$$

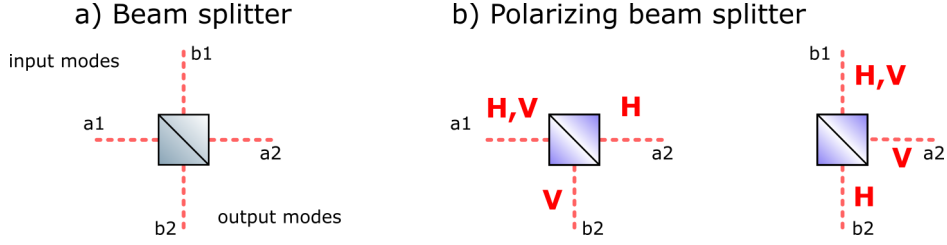


Figure 3.1: The physical action of beam splitter (BS) and polarizing beam splitter (PBS). a) Depending on the splitting ratio, the BS divides incident light into two output modes. b) A PBS reflects vertically polarized light while transmitting horizontally polarized light.

3.3 Two-qubit gates

Two-qubit entangling gates are required to complete a universal gate set. To implement such gates with photonic qubits, beam splitters (BS) and polarizing beam splitters (PBS) are used, which are devices made of two prisms with a thin metallic layer between them [44].

A perfect BS reflects a fraction R of the incident light while transmitting $T = 1 - R$. This can be expressed in second quantization notation as

$$\begin{aligned} a_2^\dagger &= \sqrt{T} a_1^\dagger + ie^{-i\phi} \sqrt{R} b_1^\dagger, \\ b_2^\dagger &= ie^{i\phi} \sqrt{R} a_1^\dagger + \sqrt{T} b_1^\dagger, \end{aligned} \quad (3.9)$$

where $a_1^\dagger$ ($a_2^\dagger$) and $b_1^\dagger$ ($b_2^\dagger$) are the creation operators in the input (output) modes, respectively. The coating of the BS defines the phase shift ϕ between the reflected and transmitted modes. Unitarity requires the phase shift ϕ_1 for photons entering the BS in mode a_1 and the phase shift ϕ_2 for photons entering the BS in mode b_1 to satisfy $\phi_1 + \phi_2 = \pi$.

A symmetric BS splits incoming light evenly into two output modes (see Figure 3.1), i.e. $R = T = 1/2$ and $\phi = 0$ [8].

A PBS divides incoming light based on its polarization. An ideal PBS reflects all light that is vertically polarized, while horizontally polarized light is transmitted (see Figure 3.1), i.e. $R_V = 1$, $R_H = 0$ [44].

However, deterministic two-photon interaction cannot be achieved with linear optical elements, i.e., mirrors, phase shifters (wave plates), and beam splitters. Consider two photons in the state

$$|+\rangle_a |+\rangle_b = (a_{1,H}^\dagger + a_{1,V}^\dagger)(b_{1,H}^\dagger + b_{1,V}^\dagger)|0\rangle \quad (3.10)$$

passing through a PBS from two input ports. The PBS transforms the creation operators:

$$a_{1,H}^\dagger \rightarrow a_{2,H}^\dagger \quad b_{1,H}^\dagger \rightarrow b_{2,H}^\dagger \quad (3.11)$$

$$a_{1,V}^\dagger \rightarrow b_{2,V}^\dagger \quad b_{1,V}^\dagger \rightarrow a_{2,V}^\dagger \quad (3.12)$$

Therefore:

$$\begin{aligned} |+\rangle_a |+\rangle_b &= (a_{1,H}^\dagger + a_{1,V}^\dagger)(b_{1,H}^\dagger + b_{1,V}^\dagger)|0\rangle \\ &= (a_{1,H}^\dagger b_{1,H}^\dagger + a_{1,H}^\dagger b_{1,V}^\dagger + a_{1,V}^\dagger b_{1,H}^\dagger + a_{1,V}^\dagger b_{1,V}^\dagger)|0\rangle \\ &\xrightarrow{PBS} (a_{2,H}^\dagger b_{2,H}^\dagger + a_{2,H}^\dagger a_{2,V}^\dagger + b_{2,V}^\dagger b_{2,H}^\dagger + b_{2,V}^\dagger a_{2,V}^\dagger)|0\rangle \end{aligned} \quad (3.13)$$

Equation 3.13 explains how the PBS acts on the input $|+\rangle_a |+\rangle_b$ and maps both photons to different outputs ($a_{2,H}^\dagger b_{2,H}^\dagger$ and $b_{2,V}^\dagger a_{2,V}^\dagger$) exactly half the time and both photons to the same output ($a_{2,H}^\dagger a_{2,V}^\dagger$ and $b_{2,V}^\dagger b_{2,H}^\dagger$) the other half of the time. In this way, the polarization-entangled state

$$|\Phi\rangle^+ = |H\rangle_a |H\rangle_b + |V\rangle_a |V\rangle_b \quad (3.14)$$

is generated probabilistically, and verification of the desired state requires destructive photon measurements in the output modes a and b . This process is called *post-selection*.

Various experimental realizations of linear optical two-qubit gates have been demonstrated. Although it is simple to implement a CZ gate, verifying its correct operation necessarily involves measurements that destroy the photons and make subsequent gates impossible. In principle, a linear optics CNOT gate implementation is scalable and can be used in succession, but it requires two entangled ancilla photons per gate, and heralding in the ancilla modes [8].

E. Knill, R. Laflamme, and G. Milburn [27] proposed a scheme that achieves near-deterministic two-photon interaction with a success probability of $n^2/(n+1)^2$ at the expense of $2n$ entangled ancilla photons and using auxiliary entangled states as well as Bell state measurements. Another method for creating nearly deterministic two-qubit gates with photons is using optical non-linearities [8]. However, photonic two-qubit gates are intrinsically probabilistic².

This is where the one-way model of QC comes in handy. The one-way model is especially suited to photonic implementations since single-photon measurements can be easily and reliably performed with high precision, and computational schemes based on cluster states can be made robust to photon loss [49]. Moreover, fast feed-forward protocols are technically feasible and have been successfully demonstrated in experiments where a two-qubit Grover's algorithm was executed with a photonic one-way processor [45]. One-way QC thus shifts the challenge of performing unitary operations on a quantum register to the generation of a sufficiently large cluster state.

In the absence of deterministic photonic two-qubit gates, the cluster state resource may be created using non-deterministic operations prior to applying any quantum logic (i.e., measurement). This allows us to make as many attempts as necessary to build the cluster state. Then, once created, the measurements required to implement the algorithm are executed. As a consequence, all probabilistic operations are outsourced to state preparation, while the actual computation is carried out deterministically via single-qubit measurements and feed-forward.

In this thesis, two types of four-qubit cluster states are generated with a setup consisting of three consecutive PBSs, a HWP, and four photonic qubits initialized in the $|+\rangle$ state. Chapter 4 describes the experimental setup in more detail.

Alternatively, photonic cluster states can be efficiently generated using so-called fusion gates [15]. Recent research has also demonstrated the direct generation of photonic cluster states with up to 14 entangled photons from a single atom in a cavity [51].

²This is a reflection of their bosonic properties.

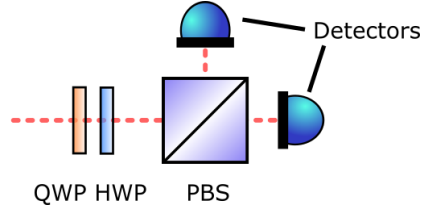


Figure 3.2: Detection stage for measuring the polarization state of light. The PBS projects the incoming light onto the $|H/V\rangle$ basis. Both outputs of the PBS are connected to single-photon detectors. The combination of QWP (orange) and HWP (blue) can rotate the eigenstates of an arbitrary basis to eigenstates of the σ_z basis. As a result, any basis spanned by two orthogonal polarization states can be measured using this configuration.

3.4 Measuring polarization qubits

To read out the state of a polarization qubit, we use a detection stage that includes a QWP, a HWP, and a PBS (as shown in Figure 3.2). The PBS projects the incident light onto the $|H/V\rangle$ basis, allowing the qubit to be measured in the σ_z basis. Both output ports of the PBS are connected to single-photon detectors.

To measure the qubit in an arbitrary basis, the QWP and HWP must be adjusted to the appropriate angles for that basis. To measure the qubit in the $|R/L\rangle$ basis, for example, we must first perform the transformation $|R\rangle \rightarrow |H\rangle$ ($|L\rangle \rightarrow |V\rangle$), which is accomplished by setting QWP to 45 deg and HWP to 0 deg. Thus, a photon detected in transmission at the PBS was in state $|R\rangle$, whereas a photon detected in reflection was in state $|L\rangle$ [44].

Table 3.3 lists the angles of the QWP and HWP for measurements in the σ_z , σ_x , and σ_y basis.

Basis	QWP	HWP
$ H/V\rangle$	0 deg	0 deg
$ D/A\rangle$	45 deg	22.5 deg
$ R/L\rangle$	45 deg	0 deg

Table 3.3: Angles for the QWP and HWP with their corresponding measurement bases.

Chapter 4

Experimental setup

“Doing research, especially in experimental physics, requires so many different skills, including being okay with the fact that the problems aren’t so well defined, and exploring that by asking the right questions. It’s so much more than just what you learned in physics class.”

Yiwen Chu, *Professor at ETH
Zürich*

4.1 Source

The photon source used for PhotonQ experiments is an InGaAs semiconductor quantum dot (QD) inside an electrically controlled micropillar cavity. We obtained the sample commercially [46], and built our own home-made optical setups for extraction and collection of single-photons, see Figure 4.1. The QD sample is placed inside a cryostat and cooled to ≈ 4 K. The cryostat is integrated into an optical table (attoDRY800), which sits on an ultra-low vibration platform.

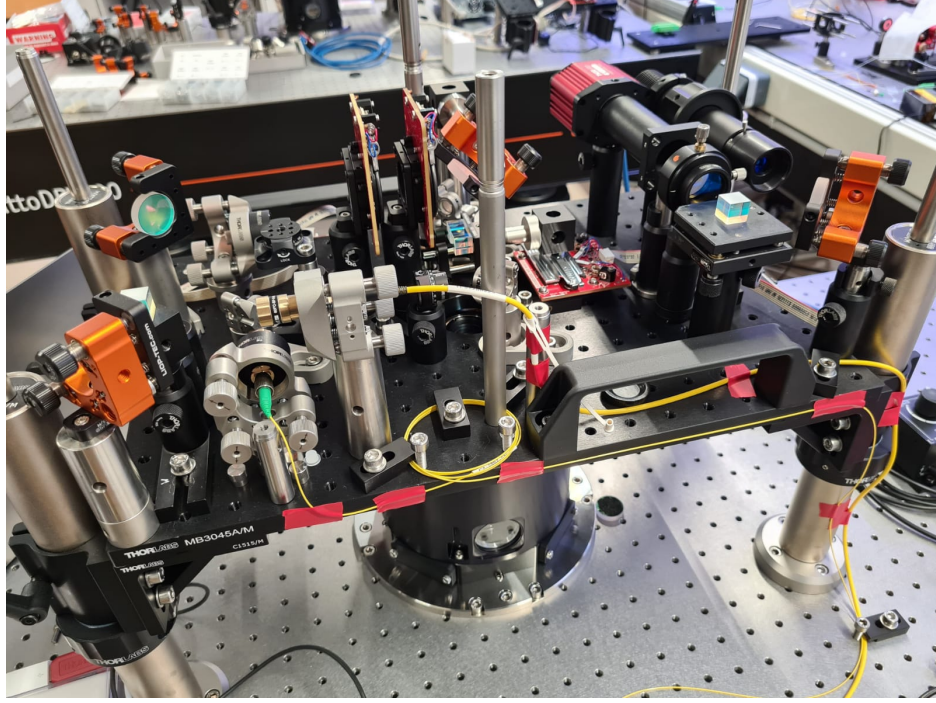


Figure 4.1: The microscope setup for QD excitation and single-photon collection is mounted on an optical breadboard on top of the cryostat's vacuum chamber (round aperture beneath the breadboard). The QD sample is placed within the vacuum chamber, and an optical window (not visible) provides an optical channel to the sample. The temperature inside the vacuum chamber is ≈ 4 K.

A pulsed Ti:Sapphire laser (Spectra-Physics Tsunami Series) with a wavelength of around 922 nm excites the QD resonantly with 10 ps long pulses (FWHM) at an 80 MHz repetition rate through a fiber-coupled confocal microscope setup. The setup is mounted on a breadboard outside the cryogenic vacuum chamber (see Figure 4.1). Back-reflected laser light is suppressed by orthogonal polarization states in the microscope's excitation and detection arms; conceptually similar as in [29]. We measured ≈ 18 MHz resonance fluorescence single-photon count rates and a photon life time of ≈ 200 ps using superconducting nanowire single-photon detectors (SNSPDs) of $\approx 89\%$ efficiency.

Three cascaded fast optical switches, each consisting of a Pockels cell (PC) and a PBS, actively demultiplex the collected stream of single-photons into four spatial modes. After that, the four photon streams are each coupled into single-mode fibers. The relative time delay between the four modes is compensated for by different fiber lengths and translation stages in each mode; conceptually similar to the setup from [34].

4.2 Graph state preparation

The photon streams propagate through fiber links to the optical setup that implements the photonic QPU. The setup consists of a state preparation stage and a detection stage (see Figure 4.2). Prior to the preparation stage, fibers wrapped in paddles (not shown in the Figure) control the polarization of the photons in each spatial mode. The polarization is tuned such that the reflection at a PBS is minimized (not depicted in Figure 4.2; see the crystals in front of the red rotation mounts in Figure 4.3). The polarization of the transmitted photons is thereby set to $|H\rangle$.

To prepare the graph state, HWPs at 22.5° in each mode initialize the photons in the $|+\rangle$ state before injecting them into three sequentially ordered PBSs (see PBS-1-3 in Figure 4.2). The extinction ratio of the PBSs is $> 1000:1$. Translation stages and different fiber lengths provide perfect temporal overlap for the interference of the four photons in the PBSs. For perfectly pure and indistinguishable photons interfering at the ideal PBS-based interferometer, the output state is a coherent superposition of 16 photon polarization configurations if the HWP between PBS-2 and PBS-3 is set to 0° . The only terms that create four-fold coincidence events are $|HHHH\rangle$ and $|VVVV\rangle$. Since these two possibilities are quantum mechanically indistinguishable, a four-fold coincidence detection corresponds to the successful post-selection of the four-photon polarization-entangled GHZ state $|\text{GHZ}_4\rangle$:

$$|\text{GHZ}_4\rangle = \frac{1}{\sqrt{2}}(|H\rangle_1|H\rangle_2|H\rangle_3|H\rangle_4 + |V\rangle_1|V\rangle_2|V\rangle_3|V\rangle_4) \quad (4.1)$$

When the HWP between the two PBSs is set to 22.5 deg instead, the resulting state is a superposition of 36 terms. Detection of four-fold coincidence events postselects the generation of the four-photon polarization-entangled linear cluster state $|C_4\rangle$:

$$|C_4\rangle = \frac{1}{2}(|H\rangle_1|H\rangle_2|H\rangle_3|H\rangle_4 + |H\rangle_1|H\rangle_2|V\rangle_3|V\rangle_4 + |V\rangle_1|V\rangle_2|H\rangle_3|H\rangle_4 - |V\rangle_1|V\rangle_2|V\rangle_3|V\rangle_4) \quad (4.2)$$

A local unitary operation $U = H_1 I_2 I_3 H_4$ transforms the $|C_4\rangle$ state into the standard four-photon linear graph state $|\mathcal{C}_4\rangle$.

Similarly, when two or three photons are fed into the setup, a two-photon Bell state or a three-photon GHZ state is produced. By iteratively adding PBS and HWP, this setup may be expanded to produce any photon number linear cluster state [34].

4.3 Detection stage

The detection step consists of QWP, HWP, and PBS and two single-photon detectors in each mode (see Figure 4.2). The QWPs and HWPs are installed in motorized rotation mounts, and their operation is fully automated by control software. The PBS output ports are fiber-coupled and routed to the detection ports of superconducting nanowire single-photon detectors (SNSPD, from Single Quantum) that operate at ≈ 2.5 K and have a detection efficiency of $\approx 89\%$. A time tagger module (Time Tagger X from Swabian Instruments) registers and processes detection events.

The QWP and HWP, in combination with the PBS, are used to read out the polarization of the photons and perform polarization analysis, e.g., quantum state tomography.

However, the correct choice of angles for the QWP and HWP also implements measurements in the basis $\mathcal{B}_j(\alpha)$ as well as local unitary operations, enabling the implementation of a quantum circuit on the prepared entangled resource states $|\text{GHZ}_4\rangle$ and $|C_4\rangle$.

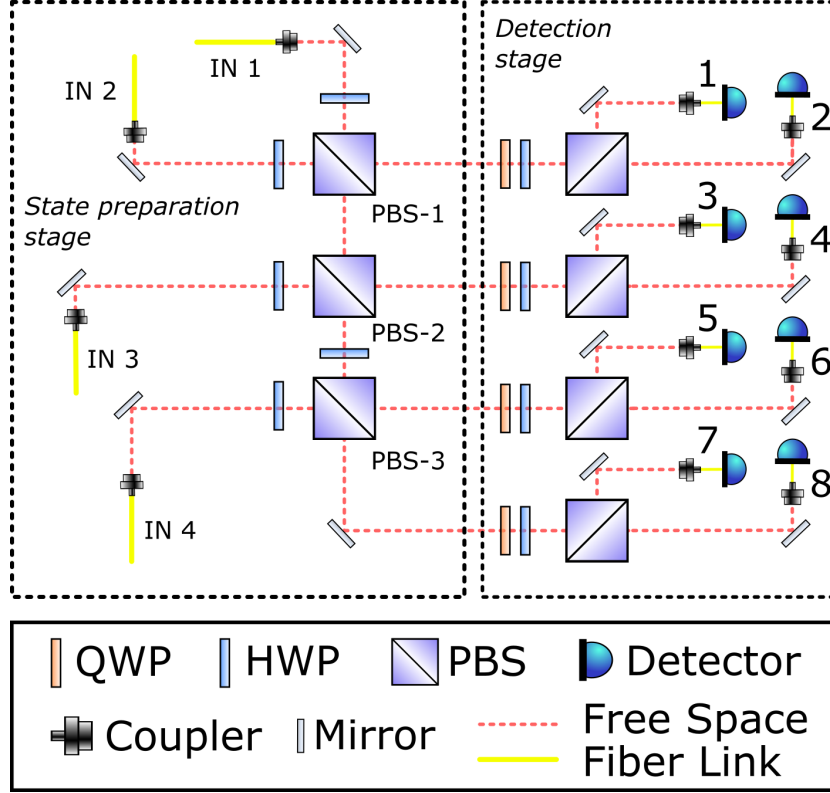


Figure 4.2: Graphic representation of the optical setup. HWPs in each input arm (IN 1-4) initialize the photons in the $|+\rangle$ state before they interfere in the PBS1-3. The state preparation stage produces a four-photon polarization-entangled GHZ state $|\text{GHZ}_4\rangle$ when the HWP between PBS-2 and PBS-3 is set to 0° , and the linear cluster state $|C_4\rangle$ when it is set to 22.5° . At the detection stage, polarization analysis, single-qubit unitaries, and measurements in an arbitrary basis are implemented using a pair of QWP and HWP in combination with a PBS. Both PBS output ports are fiber-coupled and connected to single-photon detectors.

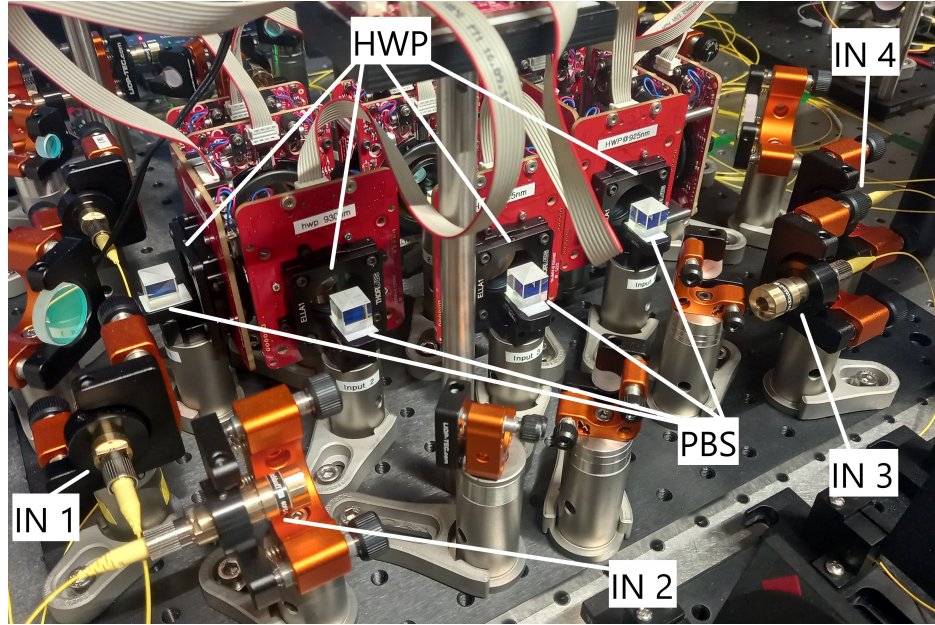


Figure 4.3: A view of the optical setup from the input side perspective (IN 1-4). The four crystals (PBSs) in this picture allow only $|H\rangle$ polarized photons to pass while reflecting $|V\rangle$ polarized photons. Paddles (not visible) adjust the polarization to minimize the intensity of the light in the reflection. The polarization of the photons is thereafter set to $|+\rangle$ by four HWPs (also depicted in Figure 4.2; here installed in the red rotation mounts right after the PBSs).

Chapter 5

PhotonQ system stack

“Contrary to popular perception, achieving quantum advantage isn’t as simple as replacing an enterprise’s classical computing infrastructure with shiny new quantum computers. Quantum computers will always make up just one component - albeit a unique, powerful one - of a more significant technology stack consisting mainly of classical computers.”

Itamar Sivan, *CEO at Quantum Machines Ltd.*

5.1 System stack overview

The PhotonQ system stack includes a website with a graphical user interface through which the user can submit instructions to the laboratory experiment. A web service defines the interface for incoming requests to the laboratory.

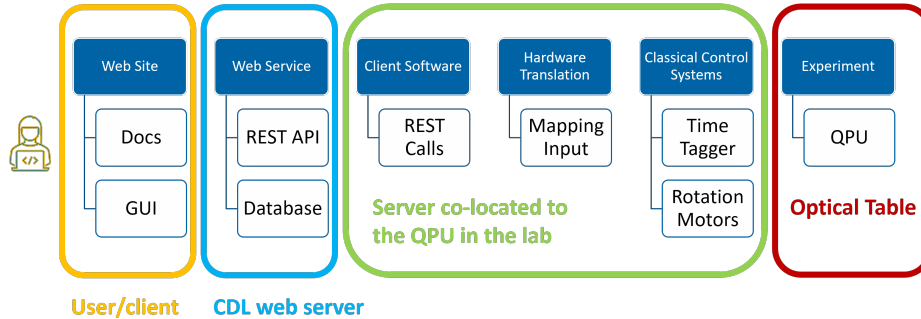


Figure 5.1: An overview of the current PhotonQ system stack. The user interacts with the PhotonQ web site (orange), which includes a graphical user interface, for sending instructions to the experiment. The web server (blue) handles client requests, database queries, and user authentication. The lab server software (green) is composed of REST calls for interacting with the REST API and Python APIs for control of the laboratory instruments used in the optical setup (red).

In the laboratory, a desktop computer functions as a server, interacts with the web service, and executes the incoming experimental instructions on the hardware. Each component is covered in detail in the next sections. A schematic of the system stack is provided in Figure 5.1.

5.2 Web application

The quantum circuits that can be built in the first generation of PhotonQ are limited by the hardware’s small number of qubits. With only four physical photonic qubits, the PhotonQ platform offers the user a limited collection of quantum circuits.

Hence, a graphical user interface (GUI) is a valuable solution. Figure 5.2, Figure 5.3, and Figure 5.4 show mockups¹ of the graphical user interface.

¹The mockups were created in collaboration with SQUER Solutions.

EXPERIMENT - 30.06.2021 EDITOR RESULTS RUN ▼

Source

Wavelength	925.00 nm	Purity	95.00 %
Single Photon Rate	16.00 MHz	Indistinguishability	90.00 %
Brightness	20.00 %	Temperature	4.00 K

Demultiplexer

The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several separate output line.

Cluster State Preparation

Choose your number of qubits and preset settings to see the cluster state.

Number of Qubits

1 2 3 **4**

Preset Settings

Linear Cluster

Cluster State

$$|L_4\rangle = \frac{1}{4} (|H\rangle_1|H\rangle_2|H\rangle_3|H\rangle_4 + |H\rangle_1|H\rangle_2|V\rangle_3|V\rangle_4 + |V\rangle_1|V\rangle_2|H\rangle_3|H\rangle_4 - |H\rangle_1|V\rangle_2|V\rangle_3|V\rangle_4)$$

Qubit Computing

Implement different quantum circuits by rearranging the prepared cluster state.

Circuit Configuration

Encoded qubits 2

CPhase gate 1

α °

β °

γ °

Circuit

$|+\rangle \xrightarrow{R_1^\alpha} R_2^\beta \xrightarrow{R_3^\gamma} R_4^\delta \xrightarrow{H} e1$

Qubit Measurements

Choose the basis on which to measure each qubit and perform a measurement based quantum computation.

Encoded qubit 1

θ °

ϕ °

Encoded qubit 3

θ °

ϕ °

Encoded qubit 2

θ °

ϕ °

Encoded qubit 4

θ °

ϕ °

Measurement

$$|E\rangle = \cos \frac{\theta}{2} |H\rangle + \sin \frac{\theta}{2} e^{i\phi} |V\rangle$$

$e1 \rightarrow |E\rangle \langle E|$

Reset Save

Figure 5.2: The PhotonQ platform's graphical user interface for controlling the QPU. The user sees information about the photon source (top block highlighted in lighter grey) and may select her preferred computing settings, including the number of physical qubits, the computational resource state, linear cluster or GHZ (*Cluster State Preparation* block), and calculation parameters (*Qubit Computing* and *Qubit Measurement* blocks); here α , β , γ , θ , ϕ .

The screenshot displays the PhotonQ System Stack interface, which is divided into several sections for configuring a quantum experiment.

- Source:** Parameters include Wavelength (925.00 nm), Purity (95.00 %), Single Photon Rate (16.00 MHz), Indistinguishability (90.00 %), Brightness (20.00 %), and Temperature (4.00 K).
- Demultiplexer:** A description states: "The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several separate output line."
- Cluster State Preparation:**
 - Number of Qubits:** A selection menu with options 1, 2, 3, and 4 (highlighted).
 - Preset Settings:** A dropdown menu showing "Linear Cluster".
 - Cluster State:** A mathematical expression for the 4-qubit cluster state:
$$|L_4\rangle = \frac{1}{4} (|H\rangle_1|H\rangle_2|H\rangle_3|H\rangle_4 + |H\rangle_1|H\rangle_2|V\rangle_3|V\rangle_4 + |V\rangle_1|V\rangle_2|H\rangle_3|H\rangle_4 - |H\rangle_1|V\rangle_2|V\rangle_3|V\rangle_4)$$
- Qubit Computing:**
 - Implement different:** A toggle switch set to "on".
 - Circuit Configuration:** A menu showing various qubit encodings (1, 2, 3, 4 qubits) with a horseshoe configuration highlighted in black.
 - Circuit:** A diagram showing two parallel circuits for qubits e1 and e2. Each circuit starts with a $|+\rangle$ state, followed by a R_z^{θ} gate, an H gate, and a CNOT gate.
 - Qubit Measurement:**
 - Choose the basis:** A dropdown menu.
 - Encoded qubit:** A selection menu for qubits 1, 2, 3, and 4.
 - Measurement:** A diagram showing two measurement bases, E_1 and E_2 , with corresponding measurement results.

At the bottom right, there are "Reset" and "Save" buttons.

Figure 5.3: By clicking on the *Circuit Configuration* button, the user may choose from a variety of circuit encodings, which are displayed in the menu appearing in the foreground and highlighted in black. A horseshoe configuration is selected in this figure. Each encoding represents a different circuit (visible in orange on the right).

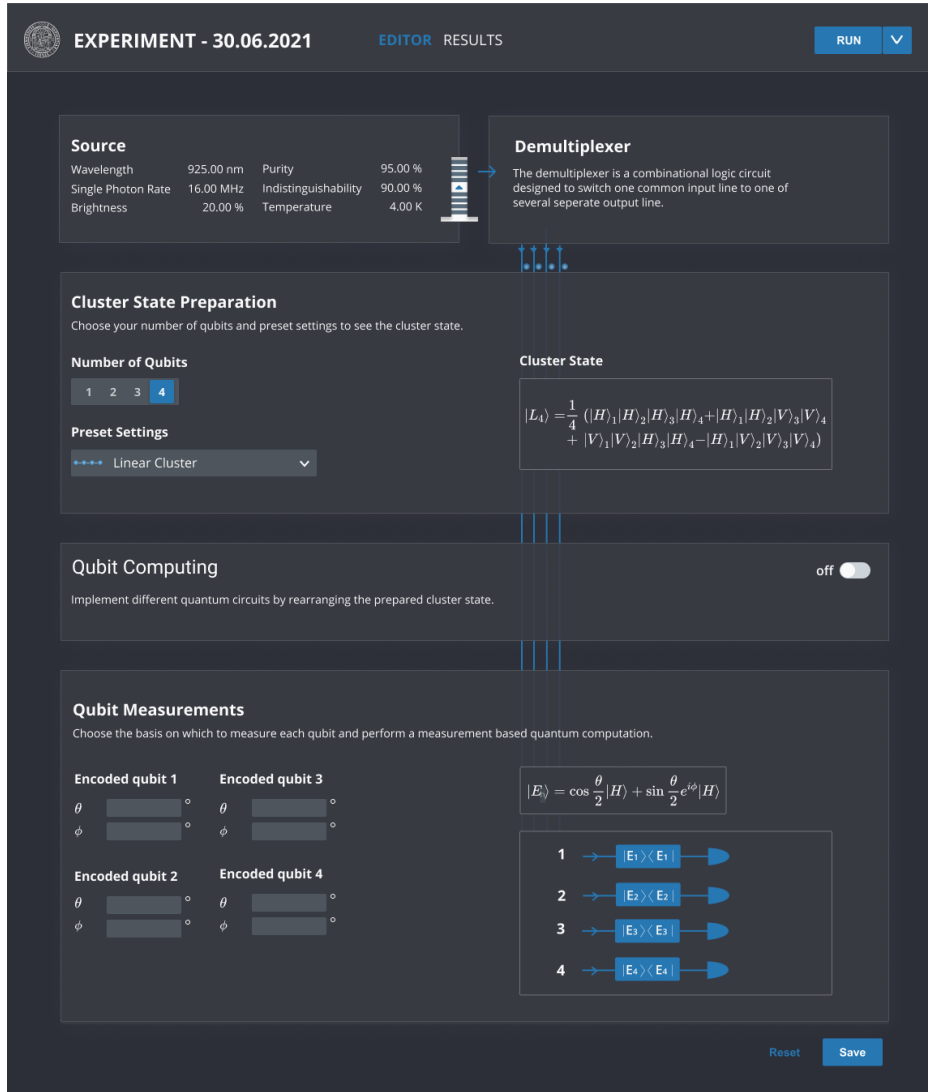


Figure 5.4: The user can also turn off the preselected encoding by flipping the switch in the *Qubit Computing* block. Then she may choose the measurement basis for each individual photon and conduct the experiment according to her interests. The *Reset* button clears all of the chosen fields, while the *Save* button saves the experiment without executing it.

The GUI visualizes the optical setup described in Section 4.2 and collects all the inputs, including experimental data and metadata, from the user that are required to execute an experiment in the lab. By clicking on the RUN button, the user sends a request with all the collected data to the CDL web service. After the request has been processed by the server and the experiment has been successfully executed, the experimental results can be retrieved under RESULTS.

5.3 CDL web service

The CDL web service provides a RESTful web API as the interface to the experiment. The API defines the protocols and standards for exchanging data with the quantum computing backend in the laboratory. The CDL web service is written in Python and built with Django [19] and the Django REST framework [20], utilizing the built-in classes for Django models and APIViews. It forms the link between our web app, databases, and the computing backend.

The REST API manages two resources: experiments and results. Experiments and results can be created, read, updated, or deleted using HTTP requests. The queue, the experiment’s list view and detail view, and the result’s list view and detail view each have their unique API endpoint (see Figure 5.5).

A client can create a new experiment via an HTTP **POST** request to the /experiment API endpoint. A **GET** call to the same endpoint returns all of the user’s experiments. The laboratory server uses the **GET** method to fetch the next experiment from the queue. With an HTTP **PATCH** request to /experiments/<slug:experiment_id>, the lab server updates the status of a specific experiment (IN QUEUE, RUNNING, FAILED, DONE) before and after it feeds the hardware control systems with the incoming instructions. **DELETE** and **GET** are also allowed on this endpoint to delete or view individual experiments.

experiments API endpoints to manipulate the experiments

GET	/experiments	Returns list of all experiments to user
POST	/experiments	Creates a new experiment
GET	/experiments/queue	Returns the oldest experiment from the queue to the lab computer
GET	/experiments/<slug:experiment_id>	Returns details about experiment with given experiment_id
DELETE	/experiments/<slug:experiment_id>	Deletes experiment with given experiment_id
PATCH	/experiments/<slug:experiment_id>	Updates status of experiment with given experiment_id

results API endpoints to manipulate the results

GET	/results	Returns list of all results to user
POST	/results	Creates a new result for an experiment
GET	/experiments/<slug:experiment_id>/results	Returns result details for given experiment_id
DELETE	/experiments/<slug:experiment_id>/results	Deletes result for given experiment_id

Figure 5.5: Overview of the API endpoints and their allowed HTTP methods (GET, POST, DELETE, PATCH) for manipulation of the experiment and result instances. The graphic was created with the Swagger Editor [50].

Only the laboratory computer has permission to post results to the API. If the experiment is carried out successfully, the results are sent to the API via HTTP **POST** on the /results endpoint. **GET** on /results retrieves all results associated with a user. Individual results can also be viewed by sending a **GET** request or deleted with a **DELETE** request to /experiments/<slug:experiment_id>/results.

The experiment schema is shown in Figure 5.6, and the result schema is shown in Figure 5.7. The entire request body from a **POST** /experiment client request is provided in Appendix A. In its current form, the data structure is in close proximity to the actual implementation of the photonic experiment (QPU). The submitted values from the client request (Figure 5.6) are converted into experimental instructions to perform physical operations on the quantum processor. Once the results are computed, they are returned to the API as explained.

5.4 Laboratory-sided backend

The PhotonQ laboratory backend is composed of the photonic experimental setup and a standard desktop computer that directly controls laboratory equipment and interacts with the REST API as a client. As soon as it is done with the previous experiment, the lab client periodically queries the queue endpoint of the API to fetch the next experiment. Using standard Python methods for handling strings and JSON data, the incoming data is parsed and piped to the hardware control systems. Helper functions perform the necessary algebraic transformations and conversions (e.g., single-qubit unitaries or degrees to radians) to set the correct angles on the wave plates and determine the channels for post-selection.

Figure 5.8 illustrates which wave plates in the setup can be directly controlled by the user. The software for the PhotonQ laboratory backend is completely written in Python and has direct access to the hardware that performs physical operations on the photonic setup. All devices are easy to integrate because they utilize Python APIs provided by the instrument manufacturer. This provides a versatile software environment in which the rotation mounts within the PhotonQ photonic setup can be controlled in real-time, and the time tagger output (photon counts and coincidences) is returned directly as a Python object.

```

1 {
2   "experimentName": "test-hs",
3   "maxRuntime": 3,
4   "circuitId": 12,
5   "ComputeSettings": {
6     "encodedQubitMeasurements": [
7       {
8         "encodedQubitIndex": 1,
9         "theta": 0.00,
10        "phi": 0.00
11      },
12      {
13        "encodedQubitIndex": 2,
14        "theta": 0.00,
15        "phi": 0.00
16      }
17    ],
18    "qubitComputing": {
19      "circuitAngles": [
20        {
21          "circuitAngleName": "alpha",
22          "circuitAngleValue": "45.000",
23        },
24        {
25          "circuitAngleName": "beta",
26          "circuitAngleValue": "45.000",
27        }
28      ]
29    },
30    "clusterState": {
31      "presetSettings": "linear",
32      "amountQubits": 4
33    }
34  }
35 }

```

Figure 5.6: This figure shows the JSON schema for an experiment. The shown fields must be filled out by the user before being sent to the API via POST. In this example, we have two encoded qubits that are read out in the Z basis, as well as two measurements $\mathcal{B}_3(\alpha)$ and $\mathcal{B}_2(\beta)$ on the physical qubits 2 and 3. The experiment's run time in the lab in seconds is determined by the value in *maxRuntime*, and *circuitId* specifies in hard-coded form which configuration of the cluster is used for the calculation (i.e., horseshoe, linear, etc.).

```

1 {
2   "experiment": "09027380-6e07-48ce-8dd6-eb8e7dc8477",
3   "startTime": "2022-10-21T23:48:50.248063Z",
4   "totalCounts": 48000000,
5   "numberOfDetectors": 8,
6   "singlePhotonRate": "1500.00",
7   "totalTime": 3,
8   "experimentData": {
9     "countRatePerDetector": {
10       "d1": 6000000,
11       "d2": 6000000,
12       "d3": 6000000,
13       "d4": 6000000,
14       "d5": 6000000,
15       "d6": 6000000,
16       "d7": 6000000,
17       "d8": 6000000
18     },
19     "coincidenceCounts": {
20       "c00": "4236",
21       "c11": "4646"
22     }
23   }
24 }

```

Figure 5.7: The JSON schema for the experiment is returned from the lab server. The result has a time stamp and the *experimentId* of the associated experiment. The result contains the current single-photon count rates in the experiment as well as the total photon counts for this execution. The counts in each detector as well as the coincidence counts are given in the result. *d1* is detector 1, and so on. Similarly, *c00* holds the coincidence counts for $|HH\rangle$ and so on. In the figure shown, however, these are dummy values and not measured values.

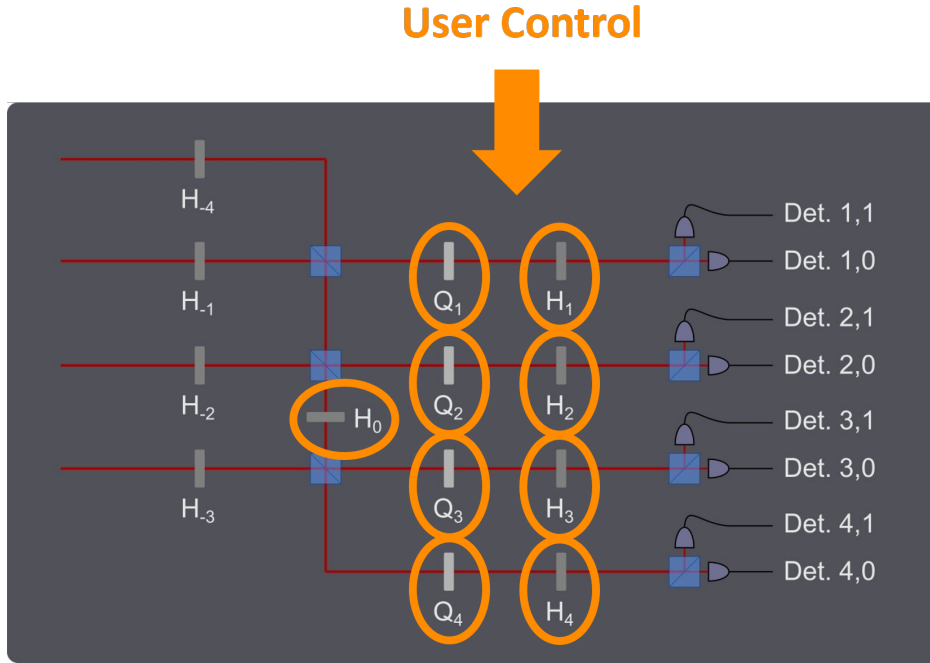


Figure 5.8: In order to provide complete control over the physical setup (QPU), the user has access to setting the values for the angles of the wave plates highlighted in orange. The input from the GUI is fed into the hardware control systems, which drive the rotation mounts to their positions as specified by the user. Wave plates $H_{-1}, H_{-2}, H_{-3}, H_{-4}$ are fixed to 45° to initialize incoming photons in the $|+\rangle$ state. The necessary post-selection for the executed circuit is automatically executed in the backend and not directly controlled by the user. *Det. 1,1* labels the reflected port in output arm 1, and *Det. 1,0* describes the transmitted port on the output arm 1, and so on.

Chapter 6

Results & outlook

“This job is a great scientific adventure. But it’s also a great human adventure. Mankind has made giant steps forward. However, what we know is really very, very little compared to what we still have to know.”

— Fabiola Gianotti
Director-General at CERN

This work is part of the PhotonQ project of the Christian Doppler Laboratory for Photonic Quantum Computing at the Faculty of Physics at the University of Vienna. The final goal of the CDL is to build a large-scale photonic one-way computer that is accessible online.

This work creates the first version of an online platform for discrete variable photonic one-way quantum computing that is built around a lightweight, standardized web interface (REST API and JSON) written in Python. The source code of the API is publicly available in the CDL repository on Github at

<https://github.com/CDL-Uni-Vienna/photonq-cdl-django>

We chose to work with Django and Django REST for this project. Django and Django REST, in particular, have a large online community that is extremely helpful throughout the development process.

As a first step, a graphical user interface for controlling the photonic one-way computer is implemented. This approach takes no optimization into account and represents a first attempt to control the photonic one-way computer via the internet. The current version of the PhotonQ web site and GUI is accessible at <https://photonq.at/> and the source code is available at

<https://github.com/CDL-Uni-Vienna/photonq-jaen>.

The source code for the laboratory backend is available at <https://github.com/CDL-Uni-Vienna/photonic-backend>.

However, this toy model is particularly useful for non-programmers and beginners. It also provides the basis for future work in the CDL with regard to the construction of a complete stack for photonic one-way computers. On the other hand, quantum application developers require programmatic access to run complex computations on real photonic hardware. Instead of a graphical user interface, the vision of the DV photonic one-way quantum computer with several dozen to several hundred or thousand photons must be programmed using code-based programming toolkits and domain-specific languages. Thus, the next step will be to integrate the CDL hardware as a backend into existing frameworks for programming quantum computers, such as Qiskit and others. Compiler tools for MBQC models are critical to achieving this milestone. Appendix B provides the Arxiv preprint of a proposed compiler pipeline for photonic one-way computers and its proof-of-concept implementation.

Depending on the specific hardware implementation, the interface requires modification. In its current state, developers must be familiar with the one-way model and the CDL hardware to use the API directly. The desired level of abstraction, as well as the target user group, define the optimal degree of API exposure.

This work also revealed several useful insights for the creation of an online quantum computing system compared to a single demonstration experiment. With its unique approach and hardware, it is likely that the CDL will position itself as a hardware provider for quantum cloud

services. In this case, much of the future work will concentrate on API development, as well as the programming of the lab's physical equipment. Compact and quick control software for quantum hardware will also play an important role in HPC integration. Further research questions regarding quantum software and the deployment of quantum computing services in the cloud are raised in [33].

The cloud provides manyfold classical computational resources, orchestration tools, and other services. Today's quantum computers, which are accessible through the cloud, are interfaced as standalone devices. While this approach has many benefits for research and education, it cannot solve large-scale productive problems. When quantum computing becomes an essential part of productive cloud services in the future, it will be included as an accelerator device alongside HPC systems. How such hybrid systems will look like is an open research question at the intersection of quantum physics and information technology. When CDL quantum computer backends get integrated into an HPC environment, the Python laboratory backend may be rewritten in C++, Rust, or GO as well.

References

- [1] Antonio Acín et al. “The quantum technologies roadmap: a European community view”. In: *New Journal of Physics* 20.8 (Aug. 2018), p. 080201. doi: 10.1088/1367-2630/aad1ea. url: <https://doi.org/10.1088/1367-2630/aad1ea>.
- [2] F. Albarrán-Arriagada et al. “One-way quantum computing in superconducting circuits”. In: *Phys. Rev. A* 97 (3 2018), p. 032320. doi: 10.1103/PhysRevA.97.032320. url: <https://link.aps.org/doi/10.1103/PhysRevA.97.032320>.
- [3] Shaukat Ali, Tao Yue, and Rui Abreu. “When software engineering meets quantum computing”. In: *Communications of the ACM* 65.4 (Apr. 2022), pp. 84–88. doi: 10.1145/3512340. url: <https://doi.org/10.1145/3512340>.
- [4] and Andreas Bayerstadler et al. “Industry quantum computing applications”. In: *EPJ Quantum Technology* 8.1 (Nov. 2021). doi: 10.1140/epjqt/s40507-021-00114-x. url: <https://doi.org/10.1140/epjqt/s40507-021-00114-x>.
- [5] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. doi: 10.1038/s41586-019-1666-5. url: <https://doi.org/10.1038/s41586-019-1666-5>.
- [6] AWS. *Amazon Braket*. url: <https://aws.amazon.com/braket/>.
- [7] Philip Ball. “First quantum computer to pack 100 qubits enters crowded race”. In: *Nature* 599.7886 (Nov. 2021), pp. 542–542. doi: 10.1038/d41586-021-03476-5. url: <https://doi.org/10.1038/d41586-021-03476-5>.

- [8] Stefanie Barz. “Quantum computing with photons: introduction to the circuit model, the one-way quantum computer, and the fundamental principles of photonic experiments”. In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 48.8 (Mar. 2015), p. 083001. doi: 10.1088/0953-4075/48/8/083001. url: <https://doi.org/10.1088/0953-4075/48/8/083001>.
- [9] J. S. Bell. “On the Einstein Podolsky Rosen paradox”. In: *Physique Physique Fizika* 1 (3 1964), pp. 195–200. doi: 10.1103/PhysiquePhysiqueFizika.1.195. url: <https://link.aps.org/doi/10.1103/PhysiquePhysiqueFizika.1.195>.
- [10] K. Bertels et al. *Quantum Accelerator Stack: A Research Roadmap*. 2021. doi: 10.48550/ARXIV.2102.02035. url: <https://arxiv.org/abs/2102.02035>.
- [11] Kishor Bharti et al. “Noisy intermediate-scale quantum algorithms”. In: *Reviews of Modern Physics* 94.1 (Feb. 2022). doi: 10.1103/revmodphys.94.015004. url: <https://doi.org/10.1103/revmodphys.94.015004>.
- [12] M. Biondi et al. *Quantum computing: An emerging ecosystem and industry use cases*. 2021. url: <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know>.
- [13] H. J. Briegel et al. “Measurement-based quantum computation”. In: *Nature Physics* 5.1 (Jan. 2009), pp. 19–26. doi: 10.1038/nphys1157. url: <https://doi.org/10.1038/nphys1157>.
- [14] Hans J. Briegel and Robert Raussendorf. “Persistent Entanglement in Arrays of Interacting Particles”. In: *Phys. Rev. Lett.* 86 (5 2001), pp. 910–913. doi: 10.1103/PhysRevLett.86.910. url: <https://link.aps.org/doi/10.1103/PhysRevLett.86.910>.
- [15] Daniel E. Browne and Terry Rudolph. “Resource-Efficient Linear Optical Quantum Computation”. In: *Phys. Rev. Lett.* 95 (1 2005), p. 010501. doi: 10.1103/PhysRevLett.95.010501. url: <https://link.aps.org/doi/10.1103/PhysRevLett.95.010501>.

- [16] Davide Castelvechi. “IBM's quantum cloud computer goes commercial”. In: *Nature* 543.7644 (Mar. 2017), pp. 159–159. doi: 10.1038/nature.2017.21585. url: <https://doi.org/10.1038/nature.2017.21585>.
- [17] Antonio D. Corcoles et al. “Challenges and Opportunities of Near-Term Quantum Computing Systems”. In: *Proceedings of the IEEE* 108.8 (Aug. 2020), pp. 1338–1352. doi: 10.1109/jproc.2019.2954005. url: <https://doi.org/10.1109/jproc.2019.2954005>.
- [18] Simon J. Devitt. “Performing quantum computing experiments in the cloud”. In: *Phys. Rev. A* 94 (3 2016), p. 032329. doi: 10.1103/PhysRevA.94.032329. url: <https://link.aps.org/doi/10.1103/PhysRevA.94.032329>.
- [19] *Django*. url: <https://www.djangoproject.com/>.
- [20] *Django REST framework*. url: <https://www.django-rest-framework.org/>.
- [21] A. Einstein, B. Podolsky, and N. Rosen. “Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?” In: *Phys. Rev.* 47 (10 1935), pp. 777–780. doi: 10.1103/PhysRev.47.777. url: <https://link.aps.org/doi/10.1103/PhysRev.47.777>.
- [22] C. Greganti et al. “Cross-Verification of Independent Quantum Devices”. In: *Phys. Rev. X* 11 (3 2021), p. 031049. doi: 10.1103/PhysRevX.11.031049. url: <https://link.aps.org/doi/10.1103/PhysRevX.11.031049>.
- [23] Lov K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, 212–219. isbn: 0897917855. doi: 10.1145/237814.237866. url: <https://doi.org/10.1145/237814.237866>.
- [24] IBM. *IBM Quantum*. url: <https://quantum-computing.ibm.com/>.

- [25] S. Jordan. *Quantum Algorithm Zoo*. 2011. url: <https://quantumalgorithmzoo.org/> (visited on 07/28/2022).
- [26] Peter J Karalekas et al. "A quantum-classical cloud platform optimized for variational hybrid algorithms". In: *Quantum Science and Technology* 5.2 (Apr. 2020), p. 024003. doi: 10.1088/2058-9565/ab7559. url: <https://doi.org/10.1088/2058-9565/ab7559>.
- [27] E. Knill, R. Laflamme, and G. J. Milburn. "A scheme for efficient quantum computation with linear optics". In: *Nature* 409.6816 (Jan. 2001), pp. 46–52. doi: 10.1038/35051009. url: <https://doi.org/10.1038/35051009>.
- [28] Huan-Yu Ku et al. "Experimental test of non-macrorealistic cat states in the cloud". In: *npj Quantum Information* 6.1 (Dec. 2020). doi: 10.1038/s41534-020-00321-x. url: <https://doi.org/10.1038/s41534-020-00321-x>.
- [29] Andreas V. Kuhlmann et al. "A dark-field microscope for background-free detection of resonance fluorescence from single semiconductor quantum dots operating in a set-and-forget mode". In: *Review of Scientific Instruments* 84.7 (July 2013), p. 073905. doi: 10.1063/1.4813879. url: <https://doi.org/10.1063/1.4813879>.
- [30] T. D. Ladd et al. "Quantum computers". In: *Nature* 464.7285 (Mar. 2010), pp. 45–53. doi: 10.1038/nature08812. url: <https://doi.org/10.1038/nature08812>.
- [31] B. P. Lanyon et al. "Measurement-Based Quantum Computation with Trapped Ions". In: *Phys. Rev. Lett.* 111 (21 2013), p. 210501. doi: 10.1103/PhysRevLett.111.210501. url: <https://link.aps.org/doi/10.1103/PhysRevLett.111.210501>.
- [32] Frank Leymann and Johanna Barzen. "The bitter truth about gate-based quantum algorithms in the NISQ era". In: *Quantum Science and Technology* 5.4 (Sept. 2020), p. 044007. doi: 10.1088/2058-9565/abae7d. url: <https://doi.org/10.1088/2058-9565/abae7d>.

- [33] Frank Leymann et al. *Quantum in the Cloud: Application Potentials and Research Opportunities*. 2020. doi: 10.48550/ARXIV.2003.06256. url: <https://arxiv.org/abs/2003.06256>.
- [34] Jin-Peng Li et al. "Multiphoton Graph States from a Solid-State Single-Photon Source". In: *ACS Photonics* 7.7 (June 2020), pp. 1603–1610. doi: 10.1021/acsphotonics.0c00192. url: <https://doi.org/10.1021/acsphotonics.0c00192>.
- [35] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing*. en. 2011. doi: <https://doi.org/10.6028/NIST.SP.800-145>.
- [36] *Method and system to create and deploy Cloud containerized quantum-based web applications using API-exposed quantum computers as back-end*. May 2019. url: <https://priorart.ip.com/IPCOM/000258553/Method-and-system-to-create-and-deploy-Cloud-containerized-quantum-based-web-app>.
- [37] Microsoft. *Azure Quantum - Quantum Cloud Computing Service: Microsoft azure*. url: <https://azure.microsoft.com/en-us/products/quantum/>.
- [38] Masoud Mohseni et al. "Commercialize quantum technologies in five years". In: *Nature* 543.7644 (Mar. 2017), pp. 171–174. doi: 10.1038/543171a. url: <https://doi.org/10.1038/543171a>.
- [39] Ashley Montanaro. "Quantum algorithms: an overview". In: *npj Quantum Information* 2.1 (Jan. 2016). doi: 10.1038/npjqi.2015.23. url: <https://doi.org/10.1038/npjqi.2015.23>.
- [40] Michele Mosca. *Cybersecurity in an era with quantum computers: will we be ready?* Cryptology ePrint Archive, Paper 2015/1075. 2015. url: <https://eprint.iacr.org/2015/1075>.
- [41] Michael A. Nielsen and Isaac L. Chuang. "Quantum Computation and Quantum Information (10th Anniversary edition)". In: 2010.
- [42] I. Pogorelov et al. "Compact Ion-Trap Quantum Computing Demonstrator". In: *PRX Quantum* 2.2 (June 2021). doi: 10.1103/prxquantum.2.020343. url: <https://doi.org/10.1103/prxquantum.2.020343>.

- [43] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. doi: 10.22331/q-2018-08-06-79. url: <https://doi.org/10.22331/q-2018-08-06-79>.
- [44] Robert Prevedel. “Experimental all-optical one-way quantum computing”. PhD thesis. University of Vienna, 2009.
- [45] Robert Prevedel et al. “High-speed linear optics quantum computing using active feed-forward”. In: *Nature* 445.7123 (Jan. 2007), pp. 65–69. doi: 10.1038/nature05346. url: <https://doi.org/10.1038/nature05346>.
- [46] *Quandela*. url: <https://www.quandela.com/>.
- [47] Robert Raussendorf and Hans J. Briegel. “A One-Way Quantum Computer”. In: *Physical Review Letters* 86.22 (May 2001), pp. 5188–5191. doi: 10.1103/physrevlett.86.5188. url: <https://doi.org/10.1103/physrevlett.86.5188>.
- [48] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM J. Comput.* 26.5 (1997), 1484–1509. issn: 0097-5397. doi: 10.1137/S0097539795293172. url: <https://doi.org/10.1137/S0097539795293172>.
- [49] Sergei Slussarenko and Geoff J. Pryde. “Photonic quantum information processing: A concise review”. In: *Applied Physics Reviews* 6.4 (Dec. 2019), p. 041303. doi: 10.1063/1.5115814. url: <https://doi.org/10.1063/1.5115814>.
- [50] *Swagger Editor*. url: <https://editor.swagger.io/>.
- [51] Philip Thomas et al. “Efficient generation of entangled multiphoton graph states from a single atom”. In: *Nature* 608.7924 (Aug. 2022), pp. 677–681. doi: 10.1038/s41586-022-04987-5. url: <https://doi.org/10.1038/s41586-022-04987-5>.
- [52] P. Walther et al. “Experimental one-way quantum computing”. In: *Nature* 434.7030 (Mar. 2005), pp. 169–176. doi: 10.1038/nature03347. url: <https://doi.org/10.1038/nature03347>.

- [53] Hui Wang et al. “Boson Sampling with 20 Input Photons and a 60-Mode Interferometer in a 10^{14} -Dimensional Hilbert Space”. In: *Phys. Rev. Lett.* 123 (25 2019), p. 250503. doi: 10.1103/PhysRevLett.123.250503. url: <https://link.aps.org/doi/10.1103/PhysRevLett.123.250503>.
- [54] Benjamin Weder et al. “The Quantum Software Lifecycle”. In: *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*. APEQS 2020. Virtual, USA: Association for Computing Machinery, 2020, 2–9. isbn: 9781450381000. doi: 10.1145/3412451.3428497. url: <https://doi-org.uaccess.univie.ac.at/10.1145/3412451.3428497>.
- [55] Tzu-Chieh Wei. *Measurement-Based Quantum Computation*. Mar. 2021. doi: 10.1093/acrefore/9780190871994.013.31. url: <https://doi.org/10.1093/acrefore/9780190871994.013.31>.
- [56] Han-Sen Zhong et al. “Quantum computational advantage using photons”. In: *Science* 370.6523 (Dec. 2020), pp. 1460–1463. doi: 10.1126/science.abe8770. url: <https://doi.org/10.1126/science.abe8770>.

Nomenclature

Abbreviations

API	Application programming interface
BS	Beam splitter
CPU	Central processing unit
DV	Discrete variables
FWHM	Full Width at Half Maximum
GHZ	Greenberger-Horne-Zeilinger
GUI	Graphical user interface
HPC	High-performance computing
HWP	Half-wave plate
JSON	JavaScript Object Notation
MBQC	Measurement-based quantum computation
OA	Optical axis
PBS	Polarizing beam splitter
PC	Pockels cell
QC	Quantum computing
QD	Quantum dot
QPU	Quantum processing unit
QWP	Quarter-wave plate
REST	Representational State Transfer
SLOCC	Stochastic local operations and classical communication
SNSPD	Superconducting nanowire single-photon detector

List of Figures

2.1	The Bloch sphere visualizes single-qubit states on a three-dimensional unit sphere. The state $ \psi\rangle$ from Equation 2.1 can be rewritten to be characterized by two real numbers, θ and ϕ , which correspond to the polar and azimuthal angles on the Bloch sphere.	9
2.2	On the Bloch sphere, the Hadamard gate rotates the $ 0\rangle$ state to the $ +\rangle$ state: The state before applying a Hadamard gate is shown on the left. Right, the state after applying a Hadamard gate.	20
2.3	Visualization of a 2D cluster state where vertices (cyan) represent physical qubits and edges (black) indicate entanglement.	27
2.4	Overview of different cluster states. a) shows a three-qubit GHZ state $ \text{GHZ}_3\rangle$, b) shows a four-qubit GHZ state $ \text{GHZ}_4\rangle$, c) a four-qubit linear cluster state, and d) a so-called horseshoe cluster.	29
2.5	The sequential measurements of qubits 1, 2, and 3 of the four-qubit linear cluster (indicated by red crosses) effectively implement the single-qubit rotation $HR_z(\delta)R_x(\beta)R_z(\alpha)$ on the logical input state $ +\rangle$. The logical qubit is teleported through the cluster and can be read out from qubit 4.	34

- 2.6 The horseshoe cluster's vertical edge implements a CZ operation, and measurements on qubits 2 and 3 (indicated by red crosses) implement rotations $HR_z(\alpha)$ and $HR_z(\beta)$ on the two encoded logical qubits, respectively. The encoded logical qubits are teleported onto qubits 1 and 4, from which the resulting logical state can be read out. 35
- 3.1 The physical action of beam splitter (BS) and polarizing beam splitter (PBS). a) Depending on the splitting ratio, the BS divides incident light into two output modes. b) A PBS reflects vertically polarized light while transmitting horizontally polarized light. 41
- 3.2 Detection stage for measuring the polarization state of light. The PBS projects the incoming light onto the $|H/V\rangle$ basis. Both outputs of the PBS are connected to single-photon detectors. The combination of QWP (orange) and HWP (blue) can rotate the eigenstates of an arbitrary basis to eigenstates of the σ_z basis. As a result, any basis spanned by two orthogonal polarization states can be measured using this configuration. 44
- 4.1 The microscope setup for QD excitation and single-photon collection is mounted on an optical breadboard on top of the cryostat's vacuum chamber (round aperture beneath the breadboard). The QD sample is placed within the vacuum chamber, and an optical window (not visible) provides an optical channel to the sample. The temperature inside the vacuum chamber is ≈ 4 K. 48

- 4.2 Graphic representation of the optical setup. HWPs in each input arm (IN 1-4) initialize the photons in the $|+\rangle$ state before they interfere in the PBS1-3. The state preparation stage produces a four-photon polarization-entangled GHZ state $|\text{GHZ}_4\rangle$ when the HWP between PBS-2 and PBS-3 is set to 0deg, and the linear cluster state $|C_4\rangle$ when it is set to 22.5deg. At the detection stage, polarization analysis, single-qubit unitaries, and measurements in an arbitrary basis are implemented using a pair of QWP and HWP in combination with a PBS. Both PBS output ports are fiber-coupled and connected to single-photon detectors. 51
- 4.3 A view of the optical setup from the input side perspective (IN 1-4). The four crystals (PBSs) in this picture allow only $|H\rangle$ polarized photons to pass while reflecting $|V\rangle$ polarized photons. Paddles (not visible) adjust the polarization to minimize the intensity of the light in the reflection. The polarization of the photons is thereafter set to $|+\rangle$ by four HWPs (also depicted in Figure 4.2; here installed in the red rotation mounts right after the PBSs). 52
- 5.1 An overview of the current PhotonQ system stack. The user interacts with the PhotonQ web site (orange), which includes a graphical user interface, for sending instructions to the experiment. The web server (blue) handles client requests, database queries, and user authentication. The lab server software (green) is composed of REST calls for interacting with the REST API and Python APIs for control of the laboratory instruments used in the optical setup (red). 54

- 5.2 The PhotonQ platform's graphical user interface for controlling the QPU. The user sees information about the photon source (top block highlighted in lighter grey) and may select her preferred computing settings, including the number of physical qubits, the computational resource state, linear cluster or GHZ (*Cluster State Preparation* block), and calculation parameters (*Qubit Computing* and *Qubit Measurement* blocks); here $\alpha, \beta, \gamma, \theta, \phi$ 55
- 5.3 By clicking on the *Circuit Configuration* button, the user may choose from a variety of circuit encodings, which are displayed in the menu appearing in the foreground and highlighted in black. A horseshoe configuration is selected in this figure. Each encoding represents a different circuit (visible in orange on the right). 56
- 5.4 The user can also turn off the preselected encoding by flipping the switch in the *Qubit Computing* block. Then she may choose the measurement basis for each individual photon and conduct the experiment according to her interests. The *Reset* button clears all of the chosen fields, while the *Save* button saves the experiment without executing it. 57
- 5.5 Overview of the API endpoints and their allowed HTTP methods (GET, POST, DELETE, PATCH) for manipulation of the experiment and result instances. The graphic was created with the Swagger Editor [50]. 59
- 5.6 This figure shows the JSON schema for an experiment. The shown fields must be filled out by the user before being sent to the API via POST. In this example, we have two encoded qubits that are read out in the Z basis, as well as two measurements $\mathcal{B}_3(\alpha)$ and $\mathcal{B}_2(\beta)$ on the physical qubits 2 and 3. The experiment's run time in the lab in seconds is determined by the value in *maxRuntime*, and *circuitId* specifies in hard-coded form which configuration of the cluster is used for the calculation (i.e., horseshoe, linear, etc.). 61

- 5.7 The JSON schema for the experiment is returned from the lab server. The result has a time stamp and the *experimentId* of the associated experiment. The result contains the current single-photon count rates in the experiment as well as the total photon counts for this execution. The counts in each detector as well as the coincidence counts are given in the result. *d1* is detector 1, and so on. Similarly, *c00* holds the coincidence counts for $|HH\rangle$ and so on. In the figure shown, however, these are dummy values and not measured values. 62
- 5.8 In order to provide complete control over the physical setup (QPU), the user has access to setting the values for the angles of the wave plates highlighted in orange. The input from the GUI is fed into the hardware control systems, which drive the rotation mounts to their positions as specified by the user. Wave plates H_{-1} , H_{-2} , H_{-3} , H_{-4} are fixed to 45° to initialize incoming photons in the $|+\rangle$ state. The necessary post-selection for the executed circuit is automatically executed in the backend and not directly controlled by the user. *Det. 1,1* labels the reflected port in output arm 1, and *Det. 1,0* describes the transmitted port on the output arm 1, and so on. 63

List of Tables

2.1	Eigenvalues and eigenvectors of the Pauli matrices.	10
2.2	Six SLOCC classes can be distinguished for three qubits. The fully separable state $ \psi_1\rangle \otimes \psi_2\rangle \otimes \psi_3\rangle$, three biseparable classes in which two qubits are entangled but separated from the third (1-23, 3-12, 2-13), and two classes with genuine tripartite entanglement, namely $ \text{GHZ}_3\rangle$ and $ \text{W}_3\rangle$	17
2.3	A table of commonly used single-qubit gates in QC.	21
3.1	The horizontal and vertical polarization states $ H/V\rangle$ define the logical $ 0/1\rangle$ states. Similarly, $ D/A\rangle$ represent the $ +/-\rangle$ states, and $ R/L\rangle$ represent the $ +_i/-_i\rangle$ states. Z , X , and Y label the corresponding orthogonal axes in the Bloch sphere picture.	38
3.2	HWP and QWP implement different single-qubit gates for different values of Θ	40
3.3	Angles for the QWP and HWP with their corresponding measurement bases.	45

Appendix A

Full Experiment Payload Schema

```
1 {
2   "user": "felix.zilk@univie.ac.at",
3   "status": "IN QUEUE",
4   "created": "2022-10-21T23:43:03.834876Z",
5   "experimentName": "test-hs",
6   "projectId": "55",
7   "maxRuntime": 3,
8   "experimentId": "09027380-6e07-48ce-8dd6-eb8e7dc8477",
9   "circuitId": 12,
10  "ComputeSettings": {
11    "encodedQubitMeasurements": [
12      {
13        "id": 5,
14        "encodedQubitIndex": 1,
15        "theta": "0.00",
16        "phi": "0.00",
17        "ComputeSettings": 3
18      },
19      {
20        "id": 6,
21        "encodedQubitIndex": 2,
22        "theta": "0.00",
```

```
23         "phi": "0.00",
24         "ComputeSettings": 3
25     },
26 ],
27 "qubitComputing": {
28     "id": 3,
29     "circuitAngles": [
30         {
31             "id": 5,
32             "circuitAngleName": "alpha",
33             "circuitAngleValue": "45.000",
34             "qubitComputing": 3
35         },
36         {
37             "id": 6,
38             "circuitAngleName": "beta",
39             "circuitAngleValue": "45.000",
40             "qubitComputing": 3
41         }
42     ]
43 },
44 "clusterState": {
45     "id": 3,
46     "presetSettings": "linear",
47     "amountQubits": 4
48 }
49 }
50 }
```

Appendix B

Published Material

This appendix contains a conference poster and the Arxiv preprint of a publication that builds on top of the work in this master's thesis:

- Zilk, F., Guggemos, T., Ortiz, O., Lored, J. & Walther, P. (2022). *Towards online-accessible photonic one-way quantum computing*. Poster presented at: SFB BeyondC Conference 2022 Frontiers of Quantum Information Science, 4 - 9 September, University of Vienna.
- Zilk, F., Staudacher, K., Guggemos, T., Furlinger, K., Kranzlmüller, D. & Walther, P. (2022). *A compiler for universal photonic quantum computers*. Paper presented at: Third International Workshop on Quantum Computing Software held in conjunction with SC22: The International Conference for High Performance Computing, Networking, Storage, and Analysis, 13 - 18 November, Dallas, Texas, USA.

The poster was presented at the poster session at the *SFB BeyondC Conference 2022: Frontiers of Quantum Information Science*¹, which took place in Vienna in September 2022. The poster gives an overview about the background of our efforts at the Christian Doppler Laboratory towards an online-accessible photonic one-way computer and reports the current status and results, as well as an outlook for future work.

¹<https://sfb-beyondc-conference2022.univie.ac.at/>

Towards online-accessible photonic one-way quantum computing



Felix Zilk¹, Tobias Guggemos¹, Omar Ortiz², Juan Loredo¹, Philip Walther¹

¹Christian Doppler Laboratory for Photonic Quantum Computing, Faculty of Physics, University of Vienna, Vienna, Austria

²Departamento de Ciencias, Sección Física, Pontificia Universidad Católica del Perú, Lima, Peru

Abstract

Various quantum systems have become accessible via the internet, e.g., IBM's superconducting devices. Current hardware implements circuit-based models, annealers, or continuous-variable computing systems.

The discrete-variable one-way model of quantum computing is of special interest in photonic systems, although not limited to them. Quantum processors based on this model are not yet available to users. Here, we assess **(1) access**, **(2) applications**, **(3) software**, and **(4) benchmarking** for an online-accessible photonic one-way quantum computer implementation [1].

Online access & cloud

- **Web services** make quantum computing resources available online 1
- **Cloud-native** quantum computing services make it easier to integrate with traditional cloud services for orchestration, logging, analytics, visualization, and storage
- ➔ Real photonic quantum hardware based on the discrete-variable one-way model is currently not available to users
- ➔ **We want to provide such a service**

Quantum applications on near-term devices

- Game theory applications 2
- Database searching
- Studies of quantum-classical gaps in computational complexity
- Measurement-based variational quantum eigensolvers [2-4]

Software stack

- Quantum algorithms must be **implemented as software**
- Open-source quantum computing software does not support discrete-variable photonic one-way processors [5]
- The core of our photonic one-way computers' software stack is a **compiler** that converts quantum assembly code into a graph-like executable 3
- **Current research** is focused on the development of software components and the design of a software development life cycle for quantum computing software

Benchmarking

- Existing metrics include the number of qubits, connectivity, error-rates, and software stack performance 4
- Cross-verification techniques for comparing the system performance of photonic one-way computers to other platforms [6]

Outlook

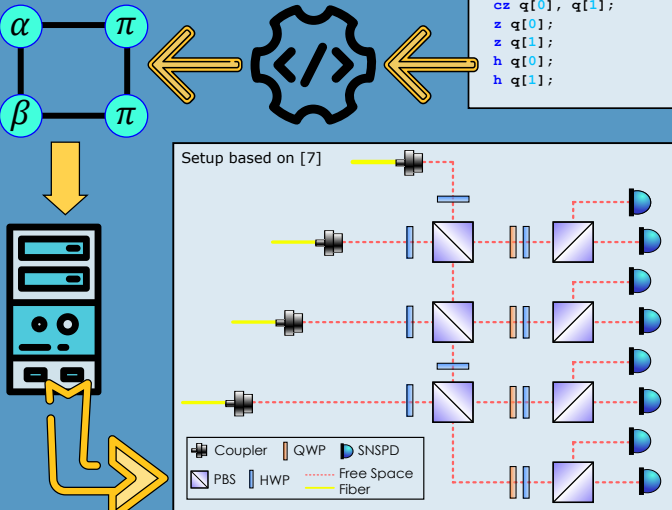
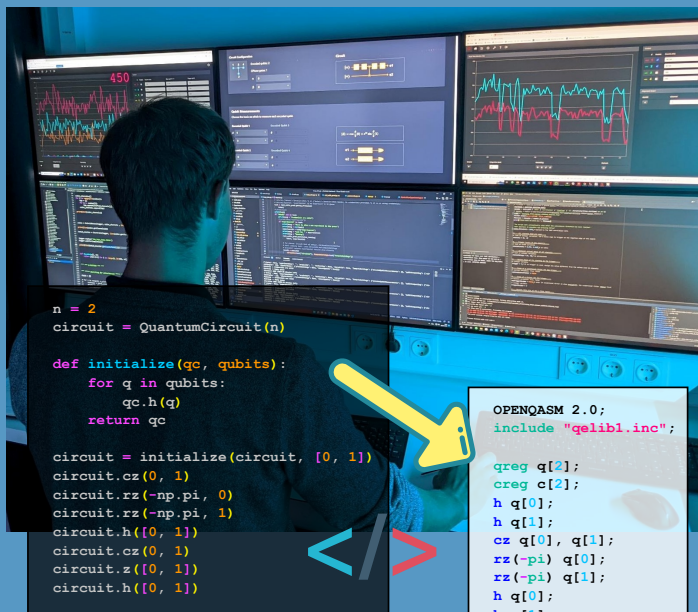
Circuits with **4 qubits** can currently be run via our platform at <https://photonq.at> using a graphical web interface:

SCAN ME



Hardware ➔ focusing on large cluster state generation

Software ➔ to integrate with existing frameworks such as Qiskit, we created a compiler concept, and we are now working on the compiler's prototype implementations



References

- [1] A. D. Córcoles et al., Proceedings of the IEEE, vol. 108, 1338-1352 (2020)
- [2] R. R. Ferguson et al., Phys. Rev. Lett., vol. 126, 220501 (2021)
- [3] M. S. Tame et al., Phys. Rev. Lett., vol. 113, 200501 (2014)
- [4] R. Prevedel et al., Nature, vol. 445, 65-69 (2007) & New J. Phys., vol. 9, 205 (2007)
- [5] M. Fingerhuth et al., PLOS ONE, vol. 13, 1-28 (2018)
- [6] C. Greganti et al., Phys. Rev. X, vol. 11, 031049 (2021)
- [7] J.-P. Li et al., ACS Photonics, vol. 7, 1603-1610 (2020)

Acknowledgments:

The authors are grateful to Borivoje Dakic, Francesco Giorgino, and Florian Kleber for helpful discussions, and Mathieu Bozzio for editing help.

In collaboration with colleagues from the MNM team from Ludwig Maximilians University in Munich and the Leibniz Supercomputing Centre, we further present a ZX-Calculus-based compiler that converts QASM input into optimized instructions for our photonic one-way quantum computers. The development of compiler tools for MBQC models becomes a critical step for the realization of practical photonic quantum computers. The paper was presented at the *SC22 conference*² in conjunction with the *Third International Workshop on Quantum Computing Software*³, which took place in Dallas, TX, in November 2022.

²<https://sc22.supercomputing.org/>

³https://sc22.supercomputing.org/presentation/?id=ws_qcs114&sess=sess423

A compiler for universal photonic quantum computers

1st Felix Zilk
Christian Doppler Laboratory
for Photonic Quantum Computer
Faculty of Physics
University of Vienna
Vienna, Austria
felix.zilk@univie.ac.at

2nd Korbinian Staudacher
MNM-Team
Ludwig-Maximilians-
Universität (LMU)
Munich, Germany
staudacher@nm.ifi.lmu.de

3rd Tobias Guggemos
Christian Doppler Laboratory
for Photonic Quantum Computer
Faculty of Physics
University of Vienna
Vienna, Austria
tobias.guggemos@univie.ac.at

4th Karl Furlinger
MNM-Team
Ludwig-Maximilians-
Universität (LMU)
Munich, Germany
fuerlinger@nm.ifi.lmu.de

5th Dieter Kranzlmüller
MNM-Team
Leibniz Supercomputing
Centre (LRZ)
Garching, Germany
dieter.kranzlmueeller@lrz.de

6th Philip Walther
Christian Doppler Laboratory
for Photonic Quantum Computer
Faculty of Physics
University of Vienna
Vienna, Austria
philip.walther@univie.ac.at

Abstract—Photons are a natural resource in quantum information, and the last decade showed significant progress in high-quality single photon generation and detection. Furthermore, photonic qubits are easy to manipulate and do not require particularly strongly sealed environments, making them an appealing platform for quantum computing. With the *one-way model*, the vision of a universal and large-scale quantum computer based on photonics becomes feasible. In *one-way* computing, the input state is not an initial product state $|0\rangle^{\otimes n}$, but a so-called cluster state. A series of measurements on the cluster state’s individual qubits and their temporal order, together with a feed-forward procedure, determine the quantum circuit to be executed. We propose a pipeline to convert a *QASM* circuit into a graph representation named *measurement-graph* (*m-graph*), that can be directly translated to hardware instructions on an optical *one-way* quantum computer. In addition, we optimize the graph using *ZX-Calculus* before evaluating the execution on an experimental discrete variable photonic platform.

Index Terms—Quantum Computing, Photonic QC, Measurement Based QC, One-way QC, ZX-Calculus

I. INTRODUCTION

Photons are a natural candidate for quantum computing, yet such systems are not very prevalent in Cloud or High-Performance Computing (HPC) platforms. However, photonic systems should be considered a valid competitor to other platforms; recent findings show setups with up to 14 entangled photons [1].

HPC is in an era of specialization, where an increasing number of accelerator devices are integrated in general-purpose computing machines [2]. Quantum computers represent an especially powerful type of accelerator, promising speed-ups for unstructured search and combinatorial optimization problems [3]–[5]. As quantum technology matures, it is important to enable integration of quantum processing units

(QPUs) in the HPC ecosystem and to support heterogeneous programming that integrates classical and quantum aspects, for example by means of offloading [6]–[11]. Here, quantum algorithms are usually expressed as offloaded computational kernels in a domain-specific language for the quantum circuit model (e.g., *QASM*) [8], [12].

Although photons are an attractive platform for QPUs, a direct translation of the quantum circuit model into photonic components is impractical. Photonic two-qubit gates are intrinsically probabilistic; hence, an increasing number of gates comes with an exponential decrease in the circuit’s success probability. That is why measurement-based schemes [13]–[15] are an appealing alternative, in particular the *one-way model* of quantum computing [15]–[18]. Here, computation is carried out solely by single-qubit measurements on highly-entangled multipartite states – so-called *cluster states* [19]. The model is equivalent to the circuit model [20], but efficient methods for translation are still rare [21], [22].

Contribution: This paper describes our efforts to develop a compiler for *QASM* kernels that targets discrete variable photonic platforms. We propose a pipeline to translate from *QASM* to a graph representation, named *measurement-graph*, or *m-graph*. The *m-graph* is optimized with *ZX-Calculus* [23] and mapped to hardware instructions for a photonic one-way processor. This marks a first step towards accessing photonic QPUs with HPC systems.

II. BACKGROUND

The quantum circuit model performs computations by sequentially applying unitary gates to qubits in a quantum register. Typically, one initializes a register of n qubits in the product state $|0\rangle^{\otimes n}$ [4].

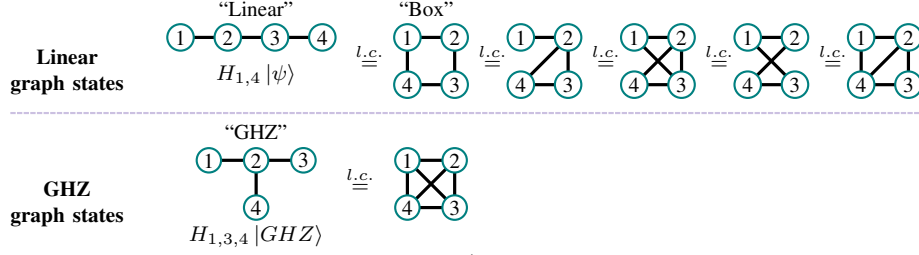


Fig. 1. Generation of the 4-qubit *Linear* and *GHZ* graph state from $|\psi\rangle = \frac{1}{2}(|0000\rangle + |0011\rangle + |1100\rangle - |1111\rangle)$ and $|GHZ\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$. The graph states in each are local complementary (l.c.) with each-other, and can be use as input states for the one-way-model.

In contrast to the circuit model, the paradigm of quantum annealing [24] bases on the unitary evolution of the underlying system Hamiltonian.

A. The measurement-based one-way model

The one-way model, however, bases entirely on adaptive single-qubit measurements that drive the computation [14], [15], [25]. Here, the initial state of individual qubits is the $|+\rangle$ state, and they are pairwise coupled via CZ operations to form a graph state, which serves as the computational resource. Then, single-qubit measurements on this resource state achieve universal quantum computation.

A *cluster state* is a type of graph state in which the underlying graph structure has the form of a two-dimensional orthogonal grid. *Graph states* are highly-entangled multipartite states, and we represent them mathematically as a graph $G(V, E)$, with vertices V representing physical qubits and edges E indicating entanglement between qubits. An arbitrary graph state $|G\rangle$ of V qubits and E edges is [26]

$$|G\rangle = \left(\prod_{(a,b) \in E} CZ_{a,b} \right) \bigotimes_{v \in V} |+\rangle_v \quad (1)$$

Fig. 1 shows a collection of 4-qubit cluster states, arranged as a two-dimensional lattice. The states are *locally complementary* (l.c.) if one can be transformed into the other with single-qubit transformations and *SWAP* operations only.

To carry out computation, we subsequently measure on connected physical qubits j in the *equatorial basis* $\mathcal{B}_j(\alpha) = \{|\alpha\rangle_j, |-\alpha\rangle_j\}$, where $|\pm\alpha\rangle_j = 1/\sqrt{2}(|0\rangle_j \pm e^{i\alpha}|1\rangle_j)$. Measurements of physical qubits in the basis $\mathcal{B}_j(\alpha)$ induce the rotation $HR_z(-\alpha)|\psi\rangle$ on encoded logical qubits up to a Pauli- X correction (cf. Fig. 2). CZ gates are inherently built into the computational resource state as links between two qubits.

Thus, the *native gate set* in the one-way model may be defined as $\mathcal{G} = \{HR_z(-\alpha), CZ\}$, which is indeed universal.

Unlike the unitary evolution in the gate-based model, the non-unitary action of measurement is irreversible. As each measurement's outcome is random, the desired result only occurs in some cases. Hence, a *feed-forward* protocol [18] compensates undesired results by adapting future measurement bases according to earlier outcomes.

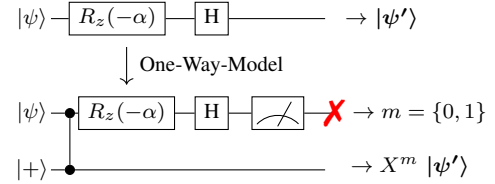


Fig. 2. Circuit representation of the conceptual difference between the circuit model with unitary transformation and the one-way model. Upper circuit: shows the transformation of a qubit in state $|\psi\rangle$ with the unitary $HR_z(-\alpha)$ to state $|\psi'\rangle$. Lower circuit: shows the same transformation with the one-way model, where the new state $|\psi'\rangle$ is then teleported to the bottom qubit by measuring the upper one (up to a Pauli- X correction, that is based on the measurements output m).

As an example, refer to the single-qubit computation in Fig. 2. The result m of the upper wire measurement (red cross mark) influences the Pauli- X correction on the lower output wire. If the outcome is $m = 0$, the algorithm works as expected; however, if $m = 1$, a Pauli error is introduced and corrected before the final measurement. The cascaded execution of this procedure allows for the implementation of arbitrary single-qubit rotations. In fact, feed-forward control makes one-way quantum computation deterministic.

Furthermore, any quantum circuit can be converted to a measurement pattern on a sufficiently large cluster state [14], [15], [25].

B. Implementation with a photonic processor

Photons are excellent candidates for building quantum computers; they are easy to generate and detect, robust against decoherence, and optical experiments can realize accurate single-qubit gates easily. However, deterministic interactions of two photons are experimentally impossible, and photonic two-qubit gates are of probabilistic nature [15], [16].

In the one-way model, these nondeterministic operations prepare the cluster state, just before any logical computation takes place [15], [16].

Post-selection techniques ensure successful generation of cluster states, such that it ignores certain detection events from the results where the cluster state generation failed [17], [18].

High-precision measurements in an arbitrary basis are achieved, for example, with phase retarders (wave plates)

TABLE I
SOME ELEMENTARY QUANTUM GATES AND THEIR RESPECTIVE REPRESENTATION IN A ZX-DIAGRAM.

Name	Z	Z-Phase	X	X-Phase	CNOT	CZ
Matrix	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{2} \begin{pmatrix} 1 + e^{i\alpha} & 1 - e^{i\alpha} \\ 1 - e^{i\alpha} & 1 + e^{i\alpha} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
Gate						
Spider/ Wire						

for qubits encoded in photon polarization. After successfully generating the multi-photon cluster, the computation proceeds deterministically by sequential execution of such single-qubit measurements.

Despite its limitations for implementing a pure circuitual model, photonic qubits are well-suited to the one-way model. Hence, the challenge to realizing universal photonic quantum computing lies in the efficient creation of sufficiently large cluster states. Various such protocols have been proposed and implemented [1], [27]–[29]. In fact, smaller clusters can already implement interestingly large circuits such as the Grover search in Fig. 5.

Fig. 3 shows how a photonic QPU can run a quantum circuit on a cluster state created from four photons. The input state generation step may be viewed independently from the computing and measurement steps that executes the necessary measurements for the post-selection and information processing.

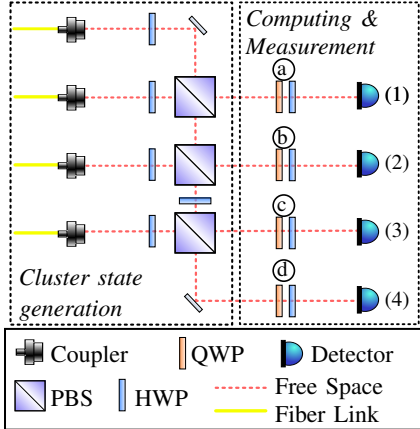


Fig. 3. A photonic quantum processing unit (QPU) to execute a *graph state* of 4 nodes, as in Fig. 1. The input are four single-photons in four fiber links. Half-wave plates (HWPs) initialize their states to $|D\rangle \equiv |+\rangle$ before they are pairwise entangled by polarizing beam splitters (PBSs) to create the cluster state. The combination of a quarter-wave plate (QWP) and a HWP before the detectors allow measurements in arbitrary bases.

C. ZX-Calculus

The ZX-Calculus is a diagrammatic language for quantum computation. A ZX-diagram consists of spiders and wires, which represent linear maps. Like the known quantum circuit notation, wires entering the diagram from the left are called input wires, and wires exiting to the right are called output wires.

Spiders are distinguished into Z (green) and X (red) spiders and can have any number of input and output wires. In Dirac notation, their linear maps are:

$$\begin{array}{c}
 m \vdots \text{Z-spider} \vdots n \\
 := |0\rangle^{\otimes m} \langle 0|^{\otimes n} + e^{i\alpha} |1\rangle^{\otimes m} \langle 1|^{\otimes n}
 \end{array}
 \quad
 \begin{array}{c}
 m \vdots \text{X-spider} \vdots n \\
 := |+\rangle^{\otimes m} \langle +|^{\otimes n} + e^{i\alpha} |-\rangle^{\otimes m} \langle -|^{\otimes n}
 \end{array}
 \quad (2)$$

Apart from the normal wire corresponding to $\mathbb{I}_2$, it is convenient to introduce another wire type for representing the Hadamard matrix:

$$\text{Hadamard wire} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{Hadamard wire} := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (3)$$

Two diagrams can be composed horizontally by joining the output wires of one diagram to the input wires of the other, or vertically by placing them side by side. In this way, any quantum circuit can be represented as a ZX-diagram, moreover, most elementary quantum gates can be directly translated, as shown in Table I.

ZX-calculus is equipped with a set of sound and complete rules, allowing the transformation of ZX-diagrams into equivalent ones wrt. their linear map. Using those rules, we can transform any ZX-diagram into an equivalent *graph-like* diagram, where we allow only Z spiders connected via Hadamard wires [30] (cf. Fig. 4a).

These are especially useful for the one-way-model. They can be directly interpreted as measurement patterns where the graph spanned by spiders and Hadamard wires corresponds to a graph state. The spider's angles correspond to measurements in the equatorial basis of the Bloch sphere [31].

Recent research has shown promising results in reducing the complexity of graph-like diagrams in terms of spiders and wires [30], [32].

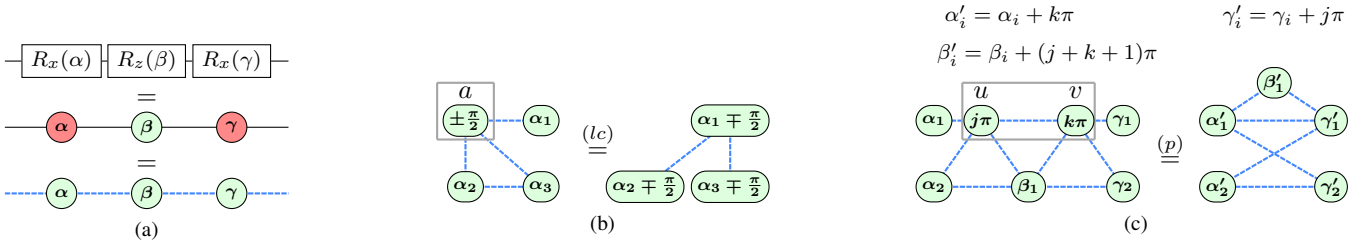


Fig. 4. (a) Transformation of a circuit for arbitrary single-qubit rotations into an equivalent graph-like diagram. (b) Local complementation (lc) and (c) Pivoting (p) rule applied on exemplary graph-like ZX-diagrams.

This is mainly done by using two graph theoretic rewrite rules named Local complementation and Pivoting which we can use to eliminate most of the “Clifford” part in diagrams, i.e., spiders with an angle of $k\pi/2$, $k \in \mathbb{Z}$. Local complementation applied on a spider with an odd multiple of $\pi/2$ removes the spider and flips the Hadamard connections between the neighboring spiders, whereas Pivoting applied on a pair of spiders with an even multiple of $\pi/2$ removes the pair and flips the connections between different sets of neighboring spiders (cf. Fig. 4b and 4c). Since every spider in a graph-like diagram corresponds to a qubit in the one-way model, we can use those rules to reduce the number of qubits needed for the computation.

III. RELATED WORK

Several compiler frameworks and tools are currently available for executing algorithms on gate-based QPUs or annealing devices [8], [12], [33]. Most high-level programming tools use sequential execution of basic quantum gates [6], [9], [12], [33], [34].

The one-way quantum computing paradigm is fundamentally different from the standard quantum circuit model (see Section II). To allow integrating photonic one-way hardware into current programming frameworks and the HPC ecosystem, we require translation from the circuit model.

The framework by Zhang Hezi et al. [21] aims particularly at mapping a quantum circuit to a specific photonic hardware architecture. This hardware abstraction model relies on a 2D lattice arrangement of resource state generators (RSGs) that create a 3-qubit graph state in each clock cycle.

For translation, the scheme takes an input circuit and transpiles it to the universal gate set $\{J(\alpha), CZ\}$ where $(J(\alpha) \equiv HR_z(\alpha))$ and construct an (in general) non-planar graph out of it. They separate this graph into multiple planar sub-graphs, each of which may contain high-degree nodes, i.e., nodes with a large number of edges. Further decomposition results in numerous appropriate low-degree (in their case, the 3-qubit GHZ-states) graph states and additional fusion operations. A clock cycle marks the execution of one planar graph state mapped to the physical layer on the considered hardware.

Vijayan Madhav Krishnan et al. [22] propose a framework using the stabilizer formalism. It converts a high-level description of a quantum circuit into a set of instructions for graph state preparation together with local operations and non-Pauli measurements. The approach is a modification of the Initialization-CNOT-Measurement (ICM) model, together with the single-qubit teleportation technique.

First, they decompose the quantum circuit into Clifford and T gate operations, the latter of which are performed via measurement-induced T and $T^\dagger$ gate teleportation. The resulting intermediate circuit – called *inverse ICM decomposition* – consists of a block of Clifford operations, Pauli corrections, and measurements in bases $\mathcal{B}_j(\pm \frac{\pi}{4})$. Since Clifford operations can be efficiently simulated, a classical computer calculates the output state of the Clifford block. The resulting stabilizer state translates to a graph state. With that at hand, the hardware instructions include the resulting graph state together with single-qubit unitaries (due to conversion) and non-Pauli measurements.

IV. COMPILER

A. Compiler pipeline

To execute an arbitrary *QASM* file on a photonic one-way processor, we perform the following translation and optimization steps (see also Fig. 5):

Step 1: We transpile an arbitrary *QASM* file as an input in the universal gate set $\{R_x(\theta), R_z(\theta), H, CX, CZ\}$. Various tools for this were proposed in the past [34]–[37].

Step 2: We rewrite the circuit as a sequence of $HR_z(\theta)$ and CZ operations as follows:

$$\begin{aligned} R_z(\theta) &= [HR_z(0)][HR_z(\theta)] \\ H &= HR_z(0) \\ R_x(\theta) &= [HR_z(\theta)][HR_z(0)] \\ CNOT &= [\mathbb{I} \otimes H]CZ[\mathbb{I} \otimes H] \end{aligned} \quad (4)$$

Step 3: The *m-graph* is defined by the sequence of $HR_z(\theta)$ and CZ : The number of qubits in the circuit defines the number of rows of the *m-graph*. A horizontal edge between two vertices in the *m-graph* corresponds to a logical $HR_z(-\alpha)$ operation and a physical measurement in the basis $\mathcal{B}_j(\alpha)$

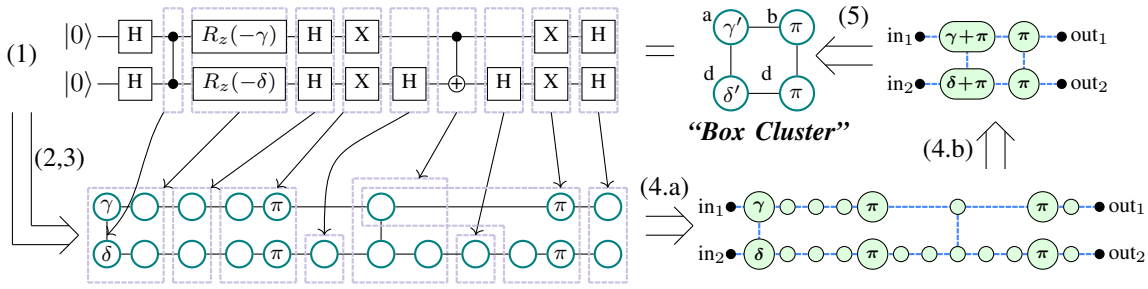


Fig. 5. The “Box” Cluster for four physical qubits (centrally located) translates into full Grover search for two qubits. Our pipeline takes the circuit description of this Grover algorithm as an input and translates it to the m -graph which is then optimized and corresponds to a “Box” Cluster. The measurement angles γ', δ' are adapted according to the hardware.

on the next physical qubit to the right. Similarly, a CZ corresponds to a vertical connection between two qubits [38].

Step 4: We interpret this m -graph as a graph-like (4.a) diagram in ZX-calculus and optimize it (4.b).

Step 5: The resulting m -graph defines the required input graph state. Each vertex shows the angle (α) and its position in the graph defines its order of execution.

B. Optimization

We optimize the *graph-like* diagram of ZX-Calculus with phase teleportation from [39] to eliminate redundant non-Clifford spiders, followed by the algorithm from [30] to eliminate Clifford spiders. Since both optimization strategies preserve the graph-like property, we can still interpret the diagram as m -graph.

V. HARDWARE INSTRUCTIONS

A. Cluster state generation

For universal computation, we need an arbitrary connected graph state as an input. In particular, the photonic setup in Fig. 3 generates $|GHZ\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$ and $|\psi\rangle = \frac{1}{2}(|0000\rangle + |0011\rangle + |1100\rangle - |1111\rangle)$ as input states [28]. By activating¹ or deactivating the half-wave plate (HWP) between the two bottom polarizing beam splitters (PBS), we produce the $|GHZ\rangle$ or $|\psi\rangle$ states, respectively. With that, we can realize arbitrary 4-photon graph states of Fig. 1 as such:

Linear: The 4-qubit linear cluster is equivalent to $|\psi\rangle$ up to a local operation $H_1 H_4$ (see row 1, column 1 in Fig. 1). Up to additional local (1-qubit + *SWAP*) operations, it is also equivalent to arbitrary connected 4-qubit clusters (e.g., the “Box”, see row 1 in Fig. 1). The only exceptions are the GHZ graph and the fully connected graph (see row 2 in Fig. 1)

GHZ: In the 4-qubit GHZ cluster, all nodes connect towards a single node (see row 2, column 1 in Fig. 6). Up to local (1-qubit) operations, it is equivalent to the $|GHZ\rangle$ state and to a fully connected 4-qubit graph (see row 2 in Fig. 1)

¹In an actual hardware setup, *activating* corresponds to an angle of 22.5° , while *deactivating* corresponds to 0° . For simulation, we can certainly just remove the HWP from the optical circuit.

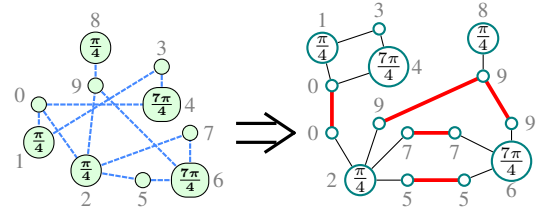


Fig. 6. Decomposition of an arbitrarily connected ZX-graph to a set of graph states, the respective measurement angles, and fusion operations (red connections). This case shows two linear (nodes 0-1-3-4 and 8-9) and two GHZ graph states (nodes 2-0-5-7-9 and 6-5-7-9). To generate the actual graph state, the nodes that occur multiple times in different graphs (nodes 0, 5, 7, 9) are fused.

In theory, the setup allows extension for producing $|GHZ\rangle$ and $|\psi\rangle$ states for an arbitrary number of qubits, however, for larger graphs, other setups are more suitable [1]. Please note that for three qubits, the GHZ and linear cluster states are identical and GHZ with > 4 qubits does not fit the definition of a cluster state.

B. Graph state generation with fusion

After the optimization in Step 4, we are left with a m -graph that is not locally complementary to the GHZ or linear graph state. Hence, we decompose the m -graph to a set of GHZ- and linear graph states, respectively. The extraction works as follows:

- 1) Find all nodes that have > 2 edges and order them with respect to number of their direct neighbors. Those are the nodes which will become “roots” of a GHZ graph state.
- 2) Extract all GHZ graph states and remove them from the ZX-graph until no node with > 2 neighbors is left.
- 3) All other spiders that have ≤ 2 neighbors are considered to be part of linear graph states and are extracted until the graph is empty.

Fig. 6 shows the decomposition of a ZX-graph that was extracted from the “bell-n4” circuit from *QASMBench* [40] with four logical qubits and 22 gates (of which 7 are two-qubit gates). After optimization, we are left with a ZX-graph of 10 spiders, which is decomposed into 4 graph states that can be implemented on hardware such as the one in Fig. 3.

Intersecting nodes between two graph states are connected using fusion gates [41], [42]. Fig. 6 shows that the photons labeled with “0”, “5”, “7” and “9” occur in multiple graph states. These labels identify the fusion gates that must be performed on hardware after the GHZ and linear graph states are generated. To perform fusion, one (in case of Type-1) or two (in case of Type-2) photons are measured, and we are left with one photon that connects two clusters. That way, we can construct arbitrary input states with any hardware setup that is able to produce GHZ and linear graph states.

C. Measurements basis

The measurement basis of each photon (1-4) on the right of Fig. 3 is defined by the angle of the corresponding vertex in the *m-graph*. As an example, node “a” in the “Box Cluster” in Fig. 5 configures the angles of the QWP/HWP “a” in Fig. 3.

VI. IMPLEMENTATION AND EVALUATION

We implement the presented pipeline with the following libraries:

- **Qiskit** [37] to transform an arbitrary circuit to the desired gate set (Step 1),
- **PyZX** [43] to translate the circuit to a graph-like diagram and optimize (Step 2-4), and
- **Perceval** [44] to create a photonic circuit as in Fig. 3, that simulates the graph states and fusion operations.

Our software is accessible on GitHub² and includes a simulation of the execution of the hardware instructions with *Perceval*. We also include a parser from *QASM* files into the basic gate set $\{HR_z(\alpha), CZ\}$.

We show the differences between our resulting optical circuits and those provided by the converter package from *Perceval*. The latter translates *Qiskit* quantum circuits into standard optical circuits with probabilistic photonic gates. However, it should be noted that this evaluation has only limited expressiveness, as the one-way model and circuit model are conceptually very different. The strength of the one-way model is not necessarily the reduction of the number of photons, but the fact that the overall computation can be performed deterministically.

A. Simulation

We simulate the photonic architecture depicted in Fig. 3 with *Perceval* using standard optical components and extend the setup for > 4 photons. To connect two graph states as in Fig. 6, we implement fusion gates using single polarizing beam splitters and half-wave plates. We choose to work with Type-1 fusion gates, for which one of the fused photons is measured; the fusion is successful if the detector registers a single photon.

B. Evaluation

We apply our compiler to extract *m-graphs* from circuits given by the *QASMBench* [40] project to demonstrate the validity of our pipeline and implementation. We execute these *m-graphs* on a simulation of our photonic setups, such as that shown in Fig. 3 in the case of four photons. Table II shows the number of required GHZ and linear graph states, together with the number of photons and optical elements – the latter are counted with *Perceval*.

First and foremost, it shows that we were able to decompose all circuits from this project and translate them to a photonic setup – which is not true for the converter provided by *Perceval* [45].

Second, we compare the number of required photons and optical elements against the quantum circuit conversion provided by *Perceval*. For some cases (e.g. *grover-n2*, *hs4-n4* or *bv-n14*), Table II shows the full strength of the one-way model, where a single input graph state can perform rather complex calculations. However, especially larger circuits with complex calculation show the excessive use of photons in the one-way model.

VII. CONCLUSION

Photons are well-known candidates for quantum computing, but they are currently underrepresented in publicly available environments, such as HPC centers. The well-established quantum circuit model, however, is impracticable for photonic hardware. Hence, measurement-based quantum computing (MBQC) is the only feasible alternative for large-scale photonic quantum computing. The quantum circuit model and the measurement-based one-way model of quantum computation are known to be computationally equivalent. Still, compilers that translate circuits into the one-way model are not established yet [21], [22].

In this paper, we describe a pipeline for a compiler that converts an arbitrary quantum circuit to an optimized instruction set for a one-way QPU. We demonstrate the translation to a discrete variable photonic platform, but the concept should be applicable to other quantum computing technologies that implement the one-way model, e.g., superconducting qubits or trapped ion qubits [46], [47].

Our implementation constructs a *measurement graph* from the *QASM* input by using simple transformation rules and optimizes it with the built-in methods of ZX-Calculus. We find that ZX-Calculus is ideally suited to extract specific hardware instructions from *graph-like* diagrams.

This work is only a first demonstration of the power of a ZX-Calculus-based compiler for photonic QPUs. We plan to further investigate the optimization for specific hardware architectures and other platforms, such as the recent proposal where 14-photon GHZ and 12-photon linear states were experimentally realized [1].

²<https://github.com/CDL-Uni-Vienna/photonq-compiler>

TABLE II

RESULTS OF COMPILED MEASUREMENT-PATTERNS FOR SEVERAL CIRCUITS OF THE OPENQASM BENCHMARK. THE COLUMNS ARE STRUCTURED AS FOLLOWS: **QASM-circuit**: NAME OF THE CIRCUIT IN THE OPENQASM BENCHMARK, **GHZ**: NUMBER OF GHZ GRAPHS, **LINEAR**: NUMBER OF LINEAR GRAPHS, **PHOTONS**: NUMBER OF TOTAL PHOTONS, **COMP.**: NUMBER OF OPTICAL COMPONENTS. WE COMPARE WITH THE CIRCUITAL-MODEL TRANSLATION AS BUILT-IN FUNCTION OF *Perceval*

QASM-circuit	Ours				Perceval	
	GHZ	Linear	Photons	Comp.	Photons	Comp.
adder-n4	6	4	37	167	46	182
bell-n4	2	2	18	76	34	135
cat-state-n4	0	1	6	17	18	42
deutsch-n2	0	1	3	8	6	11
fredkin-n3	4	3	29	126	36	144
grover-n2	0	1	4	11	10	35
hs4-n4	1	2	9	31	22	74
iswap-n2	0	1	3	8	10	32
linearsolver-n3	2	1	11	44	20	72
qft-n4	3	3	34	147	54	236
teleportation-n3	0	1	4	11	12	29
toffoli-n3	2	2	23	91	28	112
wstate-n3	3	3	29	122	40	176
adder-n10	25	10	218	1210	—	—
bv-n14	1	0	15	43	54	119
multiply-n13	17	9	157	754	106	380
multiplier-n15	51	19	767	4706	474	2096
qf21-n15	18	14	299	1467	260	1148
qft-n15	14	14	462	2234	450	2097
qpe-n9	11	7	121	568	104	432
sat-n11	68	34	834	4836	—	—
simon-n6	2	2	15	65	40	140

ACKNOWLEDGMENT

The authors would like to thank Francesco Giorgino for the fruitful discussions and the patient help with some of the theory.

This work is partially funded from the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association. This work is additionally partially supported by the German Federal Ministry of Education and Research (BMBF) under the funding program Quantum Technologies - From Basic Research to Market under contract number 13N16077.

REFERENCES

- [1] P. Thomas, L. Ruscio, O. Morin, and G. Rempe, “Efficient generation of entangled multiphoton graph states from a single atom,” *Nature*, vol. 608, no. 7924, pp. 677–681, 2022.
- [2] J. Shalf, “Hpc interconnects at the end of moore’s law,” in *Optical Fiber Communication Conference*. Optical Society of America, 2019, pp. Th3A–1.
- [3] M. Cerezo, A. Arrasmith *et al.*, “Variational quantum algorithms,” *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.
- [4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. USA: Cambridge University Press, 2011.
- [5] K. Bharti, A. Cervera-Lierta *et al.*, “Noisy intermediate-scale quantum algorithms,” *Rev. Mod. Phys.*, vol. 94, p. 015004, Feb 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.94.015004>
- [6] A. D. Córcoles, A. Kandala *et al.*, “Challenges and opportunities of near-term quantum computing systems,” *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1338–1352, 2020.
- [7] A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, and T. S. Humble, “Xacc: a system-level software infrastructure for heterogeneous quantum-classical computing,” *Quantum Science and Technology*, vol. 5, no. 2, p. 024002, 2020.
- [8] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, “ $t|ket\rangle$: a retargetable compiler for NISQ devices,” *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, 2020.
- [9] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, “Open quantum assembly language,” 2017. [Online]. Available: <https://arxiv.org/abs/1707.03429>
- [10] F. Leymann, J. Barzen, M. Falkenthal, D. Vietz, B. Weder, and K. Wild, “Quantum in the cloud: Application potentials and research opportunities,” 2020.
- [11] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, “A quantum-classical cloud platform optimized for variational hybrid algorithms,” *Quantum Science and Technology*, vol. 5, p. 024003, 2020.
- [12] R. LaRose, “Overview and Comparison of Gate Level Quantum Software Platforms,” *Quantum*, vol. 3, p. 130, Mar. 2019. [Online]. Available: <https://doi.org/10.22331/q-2019-03-25-130>
- [13] W.-B. Gao, X.-C. Yao *et al.*, “Experimental measurement-based quantum computing beyond the cluster-state model,” *Nature Photonics*, vol. 5, no. 2, pp. 117–123, Jan. 2011. [Online]. Available: <https://doi.org/10.1038/nphoton.2010.283>
- [14] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. V. den Nest, “Measurement-based quantum computation,” *Nature Physics*, vol. 5, no. 1, pp. 19–26, Jan. 2009. [Online]. Available: <https://doi.org/10.1038/nphys1157>
- [15] S. Barz, “Quantum computing with photons: introduction to the circuit model, the one-way quantum computer, and the fundamental principles of photonic experiments,” *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 48, no. 8, p. 083001, mar 2015. [Online]. Available: <https://doi.org/10.1088/0953-4075/48/8/083001>
- [16] S. Slussarenko and G. J. Pryde, “Photonic quantum information processing: A concise review,” *Applied Physics Reviews*, vol. 6, no. 4, p. 041303, 2019. [Online]. Available: <https://doi.org/10.1063/1.5115814>
- [17] P. Walther, K. Resch *et al.*, “Experimental one-way quantum computing,” *Nature*, vol. 434, pp. 169–76, 04 2005.
- [18] R. Prevedel, P. Walther *et al.*, “High-speed linear optics quantum computing using active feed-forward,” *Nature*, vol. 445, pp. 65–9, 02 2007.
- [19] R. Raussendorf and H. J. Briegel, “A one-way quantum computer,” *Physical Review Letters*, vol. 86, no. 22, p. 5188, 2001.
- [20] —, “Computational model for the one-way quantum computer:

- Concepts and summary,” 2002. [Online]. Available: <https://arxiv.org/abs/quant-ph/0207183>
- [21] H. Zhang, A. Wu *et al.*, “A compilation framework for photonic one-way quantum computation,” 2022. [Online]. Available: <https://arxiv.org/abs/2209.01545>
- [22] M. K. Vijayan, A. Paler, J. Gavriel, C. R. Myers, P. P. Rohde, and S. J. Devitt, “Compilation of algorithm-specific graph states for quantum circuits,” 2022. [Online]. Available: <https://arxiv.org/abs/2209.07345>
- [23] B. Coecke and A. Kissinger, “Picturing quantum processes,” in *International Conference on Theory and Application of Diagrams*. Springer, 2018, pp. 28–31.
- [24] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, “Perspectives of quantum annealing: methods and implementations,” *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, may 2020. [Online]. Available: <https://doi.org/10.1088/1361-6633/ab85b8>
- [25] T.-C. Wei, “Measurement-based quantum computation,” *Oxford Research Encyclopedia of Physics*, 2021.
- [26] A. Russo, E. Barnes, and S. E. Economou, “Generation of arbitrary all-photonic graph states from quantum emitters,” *New Journal of Physics*, vol. 21, no. 5, p. 055002, may 2019. [Online]. Available: <https://doi.org/10.1088/1367-2630/ab193d>
- [27] D. Istrati, Y. Pilnyak *et al.*, “Sequential generation of linear cluster states from a single photon emitter,” vol. 11, no. 1, 2020. [Online]. Available: <https://doi.org/10.1038/s41467-020-19341-4>
- [28] J.-P. Li, J. Qin *et al.*, “Multiphoton graph states from a solid-state single-photon source,” *ACS Photonics*, vol. 7, no. 7, pp. 1603–1610, 2020. [Online]. Available: <https://doi.org/10.1021/acsp Photonics.0c00192>
- [29] S. Bartolucci, P. M. Birchall *et al.*, “Creation of entangled photonic states using linear optics,” 2021. [Online]. Available: <https://arxiv.org/abs/2106.13825>
- [30] R. Duncan, A. Kissinger, S. Perdrix, and J. Van De Wetering, “Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus,” *Quantum*, vol. 4, p. 279, 2020.
- [31] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering, “There and back again: A circuit extraction tale,” *Quantum*, vol. 5, p. 421, 2021.
- [32] K. Staudacher, T. Guggemos, S. Grundner-Culemann, and W. Gehrke, “Reducing 2-qubit gate count for zx-calculus based quantum circuit optimization,” in *Proceedings 19th International Conference on Quantum Physics and Logic, Oxford, 27 June – 1 July 2022*, ser. *Electronic Proceedings in Theoretical Computer Science*, S. Gogioso and M. Hoban, Eds., 2022.
- [33] M. Fingerhuth, T. Babej, and P. Wittek, “Open source software in quantum computing,” *PLOS ONE*, vol. 13, pp. 1–28, 2018.
- [34] Microsoft, “qsharp-language,” <https://github.com/microsoft/qsharp-language>.
- [35] F. Chong, D. Franklin, and M. Martonosi, “Programming languages and compiler design for realistic quantum hardware,” *Nature*, vol. 549, pp. 180–187, 09 2017.
- [36] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, “Quipper: a scalable quantum programming language,” in *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, 2013, pp. 333–342.
- [37] M. S. ANIS, Abby-Mitchell *et al.*, “Qiskit: An open-source framework for quantum computing,” 2021.
- [38] C. Greganti, T. Demarie *et al.*, “Cross-verification of independent quantum devices,” *Physical Review X*, vol. 11, no. 3, Sep. 2021. [Online]. Available: <https://doi.org/10.1103/physrevx.11.031049>
- [39] A. Kissinger and J. van de Wetering, “Reducing the number of non-clifford gates in quantum circuits,” *Physical Review A*, vol. 102, no. 2, p. 022406, 2020.
- [40] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, “QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation,” *arXiv preprint arXiv:2005.13018*, 2021.
- [41] D. E. Browne and T. Rudolph, “Resource-efficient linear optical quantum computation,” *Phys. Rev. Lett.*, vol. 95, p. 010501, Jun 2005. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.95.010501>
- [42] S. Bartolucci, P. Birchall *et al.*, “Fusion-based quantum computation,” 2021. [Online]. Available: <https://arxiv.org/abs/2101.09310>
- [43] A. Kissinger and J. van de Wetering, “PyZX: Large Scale Automated Diagrammatic Reasoning,” in *Proceedings 16th International Conference on Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, ser. *Electronic Proceedings in Theoretical Computer Science*, B. Coecke and M. Leifer, Eds., vol. 318. Open Publishing Association, 2020, pp. 229–241.
- [44] N. Heurtel, A. Fyrrillas *et al.*, “Perceval: A Software Platform for Discrete Variable Photonic Quantum Computing,” 2022. [Online]. Available: <https://arxiv.org/abs/2204.00602>
- [45] “Convertors.” [Online]. Available: <https://perceval.quandela.net/docs/convertors.html>
- [46] S. N. Shah, “Realizations of measurement based quantum computing,” 2021. [Online]. Available: <https://arxiv.org/abs/2112.11601>
- [47] B. P. Lanyon, P. Jurcevic *et al.*, “Measurement-based quantum computation with trapped ions,” *Phys. Rev. Lett.*, vol. 111, p. 210501, Nov 2013. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.111.210501>