INSTITUTION FOR INFORMATICS

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Gate Cutting for Neutral Atom Based Synthesis

INSTITUTION FOR INFORMATICS

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Gate Cutting for Neutral Atom Based Synthesis

Author:	Emily Tme
Supervisor:	Prof. Christian Mendl
Advisor:	Yanbin Chen
Submission Date:	14.05.2025

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 14.05.2025 Emily Tme

Acknowledgments

I would like to thank my thesis supervisor, Yanbin Chen, for supporting me in this work. His ideas and guidance allowed me to explore a fascinating topic with purpose. I am grateful for his encouragement and expertise, which made this thesis possible.

Abstract

Neutral atom quantum computing has gained momentum because of its high connectivity and scalability. Regardless, noise and decoherence still pose challenges for the feasibility of this computing method. A compilation framework named Geyser optimizes circuits by reducing the laser pulses required to execute them. This work builds on Geyser by introducing ZX gate cutting to cut multi-qubit gates before circuit compilation. A series of experiments evaluates the circuit pulses and execution duration of Geyser in combination with gate cutting. The findings demonstrate that ZX gate cutting in combination with Geyser increase block count and, consequently, pulse count. Most drastically, the execution duration is significantly higher, making the strategy inefficient. The overhead from circuit sampling required by gate cutting suggests that this method is not suitable for compilation for neutral atom platforms.

Kurzfassung

Die Quantenberechnung mit neutralen Atomen hat aufgrund ihrer hohen Konnektivität und Skalierbarkeit an Bedeutung gewonnen. Dennoch stellen Rauschen und Dekohärenz weiterhin Herausforderungen für die Realisierbarkeit dieser Rechenmethode dar. Ein Kompilierungsframework namens Geyser optimiert Quantenschaltkreise, indem es die benötigten Laserpulse zur Ausführung reduziert. Diese Arbeit baut auf Geyser auf, indem sie das ZX-Gateschneiden einführt, um Mehr-Qubit-Gates vor der Schaltkreiskompilierung zu zerschneiden. Eine Reihe von Experimenten bewertet die Anzahl der Schaltkreis-Pulse und die Ausführungsdauer von Geyser in Kombination mit dem Gateschneiden. Die Ergebnisse zeigen, dass das ZX-Gateschneiden in Kombination mit Geyser die Blockanzahl und folglich auch die Anzahl der Pulse erhöht. Am auffälligsten ist die signifikant längere Ausführungsdauer, was die Strategie ineffizient macht. Der durch die Schaltkreissampling verursachte Overhead beim Gateschneiden deutet darauf hin, dass diese Methode für die Kompilierung auf neutralen Atomplattformen ungeeignet ist.

Contents

1 Introduction

As Seth Lloyd once said, "The history of the universe is, in effect, a huge and ongoing quantum computation. The universe is a quantum computer" [1]. These innovative words set the foundation for the development of quantum computing. In the past, classical computing has lead to the path to more powerful computation. However, it has reached its limits, unable to solve the burning questions posed by modern science. Quantum computing takes the rules that the universe has already provided and manipulates them to compute problems that classical computers would take eons to complete. Like bits, the qubits can be manipulated into state changes, where quantum mechanics allows qubits to be in multiple states simultaneously, or in superposition. This feature is what makes this computation method so powerful, allowing for the parallel processing of vast quantum information.

The hardware behind quantum computing branches into many different categories. The qubits can be represented in various ways. One such category is neutral atom quantum computing. It uses real neutral atoms found in nature to simulate computations. Optical tweezers hold these atoms in place, while laser pulses are beamed on the atoms to manipulate their quantum states and create entanglement. This strategy has some favorable characteristics over other types of quantum computing because it allows for more scalability and connectivity. The major advantage, however, is its ability to natively implement multi-qubit gates. In many cases, this allows for fewer pulses, and thus, fewer errors and noise.

As with the other quantum computing platforms, neutral atom quantum computers still face many challenges due to noise and errors. Qubits may lose their intended state over time, or atoms may be lost during the computation all together. These limitations present themselves as a hardware problem. However, they can be alleviated using software solutions. Compilation plays an important role when translating human-readable code into instructions executable by the machine. By optimizing the compilation, the circuits executed can be minimized to use fewer pulses to calculate the same calculations. One such compilation strategy specific to neutral atom quantum computing is proposed by the Geyser framework, which uses the platform's native qubit implementation to reduce the number of laser pulses needed by a circuit [2]. It separates the circuit into sections that can be executed in parallel and then attempts to replace each of these sections with blocks containing CCZ gates. This reduces the laser pulses needed to execute the circuit, allowing for lower error and noise.

This work builds on the Geyser framework by exploring the implementation of ZX gate cutting in combination with Geyser. ZX gate cutting uses ZX calculus rules to split a multi-qubit gate into gates that can be executed independently [3]. ZX calculus is a diagrammatic

language for representing quantum circuits that aids in circuit optimization. The notation and rules it uses facilitate finding an equivalent quantum circuit with fewer gates. After applying numerous ZX calculus rules, the decomposition allows for a sum of terms where the multi-qubit gate is split into smaller gates. The idea behind the experiments in this thesis is to cut the multi-qubit gate within a circuit before Geyser compilation is applied. The intention was that blocks within the circuit separated by Geyser would become bigger. Bigger blocks would allow for less pulses because replacing one big block would likely require fewer gates than replacing two smaller blocks.

Chapter 2 of this thesis will go through the necessary background information to help understand the later experiments, focusing on quantum computing fundamentals, specific characteristics of neutral atom quantum computing, the steps of the Geyser framework, and the math behind XZ gate cutting.

Chapter 3 presents a comprehensive overview of the methodology employed in conducting the experiments. It details the experimental setup, including the selection and definition of variables, as well as the specific hardware and software tools utilized throughout the process.

Chapter 4 thoroughly examines the outcomes of the experiments, offering a presentation of the data collected. It also includes an in-depth analysis and interpretation of the results, highlighting key patterns, insights, and any unexpected findings.

Chapter 5 provides a conclusive summary of the research, drawing together the main points and findings. Additionally, it reflects on the limitations encountered during the study and suggests directions for future work.

2 Background

2.1 Quantum Computing Fundamentals

Quantum computing has gained much attention in recent years because of its perspective for more powerful computation. It combines computer science, quantum information, and quantum physics to process data in ways that classical computers are not capable of.

2.1.1 Qubits

Classical computing uses bits, a unit of information that can be in the state 0 or 1. Qubits, or quantum bits are similar in that they also have a state. Their state is written in Dirac notation using "$|\,\rangle$". Qubits can be in the 0 or 1 state as well, just like a classical bit, written as $|0\rangle$ or $|1\rangle$. However, qubits posses the ability to take on other states as well by using a linear combination of states. This combination of states is called superposition, written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

In equation (2.1), alpha and beta represent the amplitude as complex numbers. However, the qubit state must be a unit vector in the complex vector space, meaning that alpha and beta must adhere to the following condition:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.2}$$

The same applies for describing the state of a multi-qubit system. To find the state of multiple qubits, one must simply multiply the states using the tensor product. This is called the Composition of Systems Postulate [4]. The square of the amplitudes will still sum up to one to ensure normalization. However, there will now be 2^n possible states as well as terms in the equation, with n being the number of qubits, as seen in the example in equation(2.3) [5].

$$\begin{aligned} |\Psi\rangle &= (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle) \\ |\Psi\rangle &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle \end{aligned} \tag{2.3}$$

The condition in equation (2.2) ensures that the value of the qubit seen in equation (2.1) has the probability $|\alpha|^2$ of being $|0\rangle$ and the probability $|\beta|^2$ of being $|1\rangle$ when measured [4]. However, this Composition of Systems Postulate only applies to closed systems where the qubits are isolated from interacting with each other. Qubits can be entangled with one another, meaning their system is impossible to write in tensor product form because they depend so heavily on each other they are essentially only one system. In this case, the

Compositon of Systems Postulate does not apply to each of the entangled qubits [4]. Once the qubit is measured, it loses its superposition and collapses into one of its basis states. Since a qubit can take on any superposition of these basis states, it is useful to have visual representation of the state of the qubit. For this, the Bloch sphere is a good tool. However, for the qubit to be visualized on the Bloch sphere, the equation of a state of a qubit must be adjusted. Any qubit state can be described as a unit vector on the Bloch Sphere using the linear combination in equation (2.4). The parameter θ changes the amplitude of the qubit,and thus, the measurement probabilities of each state. The parameter ϕ modifies the phase of the qubit, which can have an impact on the outcome of measurements, but not on the probability of the states of the qubit in question [6].

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.4}$$

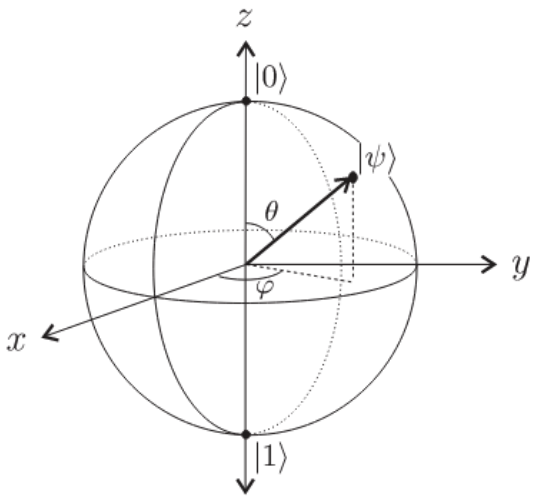


Figure 2.1: A visual representation of the state of a qubit on a Bloch sphere [4]

Until now, the basis states considered have been $|0\rangle$ and $|1\rangle$. However, this is simply the most common basis state for qubits, called the computational basis. Basis states can be arbitrary as long as the two states are orthonormal. The Bloch sphere shows the basis states $|0\rangle$ and $|1\rangle$, respectively, on the south and north pole of the Z-axis. Once an orthonormal basis state is chosen, a qubit will collapse into one of those states when measured with respect to that basis [4]. Another example of a common basis state is the Hadamard basis state with possible states $|+\rangle$ and $|-\rangle$, shown in equation (2.5). The basis state $|+\rangle$ is on the positive X-axis and $|-\rangle$ is on the negative X-axis of the Bloch sphere.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.5}$$

2.1.2 Quantum Gates

Just as classical bits have gates such as AND, OR, NOT, etc., qubits have quantum gates to manipulate their quantum information. A quantum gate is written mathematically as a complex matrix with the dimensions 2^n by 2^n, with n being the number of qubits acted on by the gate. Any matrix can represent a quantum gate as long as it is unitary [6]. A unitary matrix is a matrix that, when multiplied by its conjugate, results in an identity matrix. Equation (2.6) demonstrates this property. Because of the unique characteristics of unitary matrices, their operations are reversible. In classical computing, only the NOT operator is reversible. In contrast, all quantum gates can be reversed by their conjugate matrix.

$$UU^\dagger = U^\dagger U = I \tag{2.6}$$

In this work, we will review specific quantum gates that will be used later in the experiments. One of these is the U3 gate, which can place a qubit in any desired superposition using the matrix in equation (2.7) with angle parameters $\theta, \phi, \lambda \in [0, 2\pi]$ [7]. Referring back to the Bloch sphere, the U3 gate can change the position of the qubit's unit vector to any other direction.

$$U3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{bmatrix} \tag{2.7}$$

The Z gate is also a single qubit gate which changes the phase of a qubit. The gate leaves $|0\rangle$ unchanged but changes the sign of $|1\rangle$ to $-|1\rangle$ [6]. As explained in 2.1.1, changing the phase means the amplitude of the probabilities of each state remains the same. In the Bloch sphere, this means the unit vector of the qubit is rotated 180° around the X-axis. Equation (2.9) shows the matrix of a z gate operation.

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.8}$$

The S gate is not much different from the Z gate, except that it changes the phase of a qubit differently. The S gate is also a single qubit gate that changes the phase of a qubit by applying a 90° rotation on the Z-axis of the Bloch sphere. Equation (2.9) shows the corresponding matrix of the S gate [8].

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \tag{2.9}$$

Another important gate for later chapters of this work is the CZ gate. The CZ gate acts on two qubits, where one is referred to as the control qubit and the other the target qubit. If the control qubit is in the state $|1\rangle$, the gate applies a Z gate to the target qubit. However, if the control qubit is in state $|0\rangle$, the Z gate is not applied to the target qubit. Similarly, the CCZ gate does the same operation except on three qubits, two being control qubits and one being the target qubit. In a CCZ gate, both control qubits must be in the state $|1\rangle$ for the Z gate to be applied to the target qubit. Equation (2.10) demonstrates the matrices of the CZ gate and the CCZ gate [7].

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad CCZ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.10}$$

2.1.3 Quantum Circuits

Simply put, a quantum circuit is a collection of gates acting on one or more qubits. A quantum circuit consists of wires, each one corresponding to the state of one qubit in the circuit. These wires contain consecutive quantum gates or operations. A wire can be seen as the passage of time, read from left to right. With two gates on the wire, for example, the left gate is applied before the right gate. The order of the gates matters and can change the outcome. Unlike circuits in classical computing, quantum circuits are acyclic and the wires do not connect to themselves or each other [6]. An arbitrary U3 gate on the wire can be represented by a box containing the letter U. More common gates such as S gates or Z gates can be represented the same way but with S and Z in the box, respectively, instead of U. Additionally, a measurement is an operation that collapses the state of the qubit to either $|0\rangle$ or $|1\rangle$. This measure is then stored by a classic bit. The measurement operation is also placed on the wire as a meter symbol. In the circuit pictured in figure 2.2, the small Z pictured on the measurement operation means the qubit is measured in the Z basis, or computational basis.

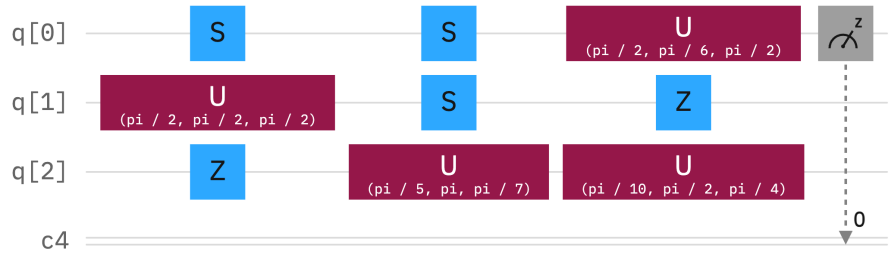


Figure 2.2: A quantum circuit with $U3$, X, and S gates acting on three qubits, followed by a measurement stored in the classical bit c4 [8].

Gates acting on multiple qubits are visualized by placing the box of the gate on all the wires connecting to the qubits on which the gate is applied. Controlled gates vary from this method, they instead place a filled dot on the wire of the control qubit and the standard gate box on the target qubit. CNOT gates, or controlled X gates which flip the target qubit when

the control qubit is $|1\rangle$, have a specific symbol on quantum circuits where the control qubit is still represented by a black filled in circle but the gate box with the X is replaced by a circle with a plus symbol inside. CZ gates also have unique notation, where both the control and the target bit are filled in black dots connected by a line.

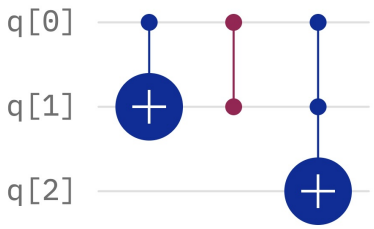


Figure 2.3: A quantum circuit with CX, CZ, and CCX gates acting on three qubits. [8].

An important differentiation between quantum gates and measurements in a circuit, is that quantum gates represent a closed system without outside interaction [4]. Meanwhile, operations like measurements are an interaction with some sort of measuring device or observer. Because the measurement is an external interaction, it is not a unitary operation and is thus also not reversible.

2.2 Neutral Atom Quantum Computing

Physically realizing quantum computers can be a devious task. The quantum computer must be well isolated to avoid decoherence, or quantum noise, because it sabotages the quantum information. In a trade-off, the quantum computer must also be accessible enough to carry out operations on the qubits and manipulate states [6]. DiVincenzo's criteria specifies a list of characteristics that a quantum computer must fullfill [9]:

1. A scalable physical system of qubits
2. The ability to initialize qubits to a fiducial state
3. Longer decoherence times than gate operation times
4. A universal set of quantum gates

Neutral atom quantum computing has been a rising quantum computing platform because of its advantageous characteristics over other platforms such as superconducting quantum computers and trapped ion quantum computers. It has been successfully used for simulating quantum systems to advance science and its use for computational problems such as quantum algorithms is an ongoing topic of research. As this work focuses specifically on neutral atom quantum computing platforms, the following sections will focus its architecture, computation methods, and key takeaways. These explanations will fulfill DaVincenzo's criteria of a quantum computer.

2.2.1 Neutral Atoms As Qubits

As the name implies, neutral atom quantum computers use neutral atoms to represent qubits. One atom corresponds to one qubit. Commonly used atoms include Rubidium, Cesium, and Strontium. Being in the alkali metal and alkaline-earth metal groups of the periodic table, these atoms offer an advantageous electron composition for the cooling and trapping steps, explained in depth later on, used in neutral atom quantum computing [10].

To represent the states of the qubit, either electronic spin states or nuclear spin states of the atom are used. When using electronic spin states, an interaction between electronic spin and nuclear spin leads to hyperfine splitting, which is a small change in energy level. The lower hyperfine energy state (e.g., $|F = 1, m_F = 0\rangle$) is typically assigned to $|0\rangle$, while the higher hyperfine state (e.g., $|F = 2, m_F = 0\rangle$) is assigned to $|1\rangle$. This difference in spin can be realized with a microwave, allowing precise control of a qubit's state [10]. Strontium is a strong candidate for using nuclear spin to define qubit states. This strategy trades longer decoherence times for decreased control of the qubit. Being that the nucleus is more isolated, it is more protected from outside interference than electrons but is harder to access.

2.2.2 Preparing the Quantum Register

To begin, the neutral atoms must first be slowed down through cooling, which starts in a room temperature high-vacuum system. Magneto-optical traps (MOTs) are employed where a light laser is beamed onto the atom, charging it with photons and, thus, giving the atom momentum and energy. These photons are then scattered from the atom in a random direction. When the atom moves towards the direction of the beam, the photons are more likely to be absorbed, meaning the net momentum transfer will shift opposite to the atom's direction. This leads the atom to slow down and cool to below 1 mK [10]. This effect of the light laser is called Doppler cooling. The quantum registers are stored in optical tweezer arrays, which

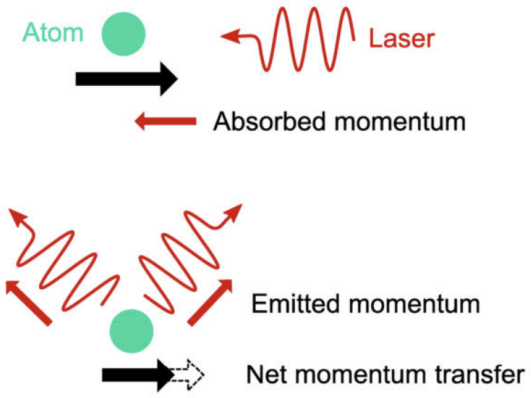


Figure 2.4: The process of Doppler cooling on an atom [10]

have one atom in one slot. To trap the atoms, spacial light modulators (SLMs) beam tightly

coupled light lasers. The word array can be misleading because these lasers can trap the atoms in any 2D or 3D spacial geometric arrangement; it does not necessarily have to be a 2D grid. The light beams of the SLMs trap the atom using dipole forces. The magnetic field of the laser separates the positive and negative charges in the atom, creating either an attractive or repulsive dipole force between the atom and the laser. In most cases, the force is attractive and the atom is then held in place by the laser. To ensure the quantum operations can still be carried out, there remains a gap between the atom and the laser despite the attraction. The lifetime of the atom in the trap before decoherence is limited to 10-20 seconds because of the unfavorable conditions of the vacuum pressure [10].

Following cooling and trapping, each trap in the optical tweezer array has either one or zero atoms. Each slot has about a 50% chance of containing a neutral atom [11]. To arrange the atoms in the desired positions, the atoms are first illuminated by fluorescent imaging using a charge-couple device (CCD) to know which traps contain an atom and which do not. The imaging shows filled traps as light spots and empty traps remain dark. To prepare the initial state of the quantum system for computation, optical tweezers rearrange the atoms to the desired traps with a 99% success rate [11]. Acousto-optical deflectors (AODs) realize this rearrangement by adjusting the tweezers holding the atoms. Once rearrangement is complete the atoms are once again imaged to confirm their position. The quantum registers are now ready for computation. The entire process for preparing the neutral atom qubits must be repeated for each computation.

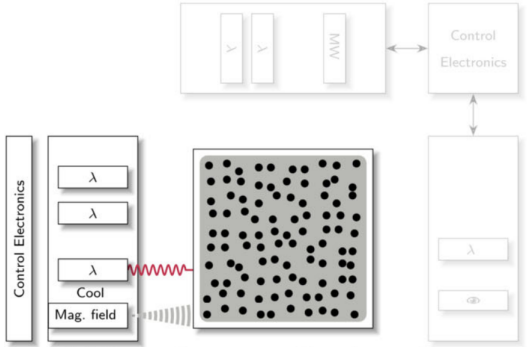


Figure 2.5: The atoms are cooled using MOTs [10].

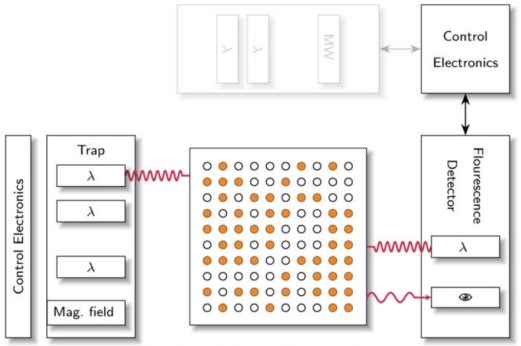


Figure 2.6: The atoms are trapped using SLMs [10].

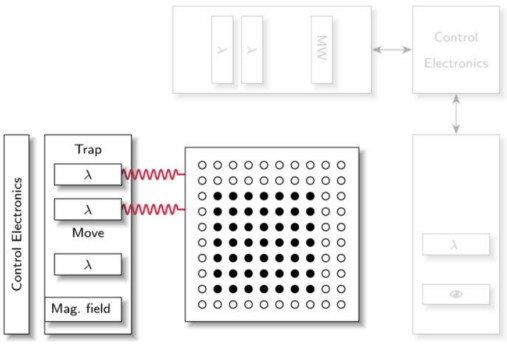


Figure 2.7: The atoms are rearranged to the desired positions using AODs [10].

2.2.3 Computations

There are two types of processing used for neutral atom quantum computing: digital and analog. Digital processing uses logical gates in a circuit to carry out a computation, while analog processing involves realizing a Hamiltonian over time using Schrödinger's equation. However, analog processing is out of scope of this work. Therefore, the next sections will discuss how gates are physically realized on the neutral atoms.

As mentioned in 2.2.1, the difference in the energy of a qubit in state $|0\rangle$ or state $|1\rangle$, is in the range of a microwave. However, beaming the atom with a microwave is insufficient because the atoms are so closely packed to each other that the length of the microwave cannot be focused on a single atom without impacting the others. To alleviate this problem, single qubit gates are implemented using Raman transitions. The Raman transition is realized using two lasers to create an intermediate state in the qubit without unnecessarily losing energy. The laser realizes the transitions $|0\rangle \leftrightarrow |a\rangle$ or $|1\rangle \leftrightarrow |a\rangle$ with intermediate transition state "a" instead of directly jumping into the state transitions $|0\rangle \leftrightarrow |1\rangle$ [12]. The intermediate transition increases precision and control over the qubit. Using the equation in (2.11), the qubit can have any arbitrary rotation (x, y, z) on the bloch sphere by adjusting the lasers' parameters, and, thus, the Raman transition is capable of implementing any single qubit gate [11].

$$(x, y, z) = (\Omega\tau \cos\varphi, \Omega\tau \sin\varphi, \delta\tau) \tag{2.11}$$

The parameters correlate to the settings for the laser: Ω is the Rabi frequency, δ is the detuning, φ is the relative phase and τ is its duration. Using the parameters $\Omega\tau = \pi$ and $\varphi = 0$, for example, would result in rotation angle (π,0,0) on the Bloch sphere, or in other words, a NOT gate is applied on the qubit.

The ground states of the neutral atom qubits in the quantum register do not have enough energy to interact with each other to execute two-qubit gates. As a solution, the neutral atom architecture, like it does to trap to atoms in the first place, takes advantage of dipole-dipole interactions again. The atom is excited to a higher energy state called a Rydberg state using

laser pulses. Recall that the atom types chosen are from group one or two of the periodic table. Thus, using Rubidium as an example, there is only one electron in the valence shell of the atom. Being farthest from the nucleus, this electron excited by the Rydberg state has a higher energy that allows it to move farther away. This gives the dipole-dipole interaction of the excited atom a larger range to interact with other atoms. To induce the Rydberg state, two lasers are focused on the two atoms in the two-qubit gate. Because of the high energy of a Rydberg atom, it cannot be held by the trapping lasers. Therefore, the trapping lasers are temporarily switched off while the atom is in this state [10].

Since two Rydberg atoms repulse each other, other atoms within the range of an atom in the Rydberg state are blocked from being excited to the Rydberg state themselves. This concept is known as the Rydberg blockade. Equation (2.12) shows how the blockade radius is calculated, with C_6 being the state-specific constant depending on the specific atom and Rydberg state, Ω being the Rabi frequency of the laser, and $\hbar$ being Planck's constant [13].

$$r_b \approx \left(\frac{C_6}{\hbar\Omega}\right)^{1/6} \tag{2.12}$$

The blockade is exactly how the two-qubit gates are manipulated. However, this leads to unwanted side effects because other atoms in the radius are also restricted. This can leave those atoms idle and, consequently, cause a waste of resources. This concept is called the restriction radius.

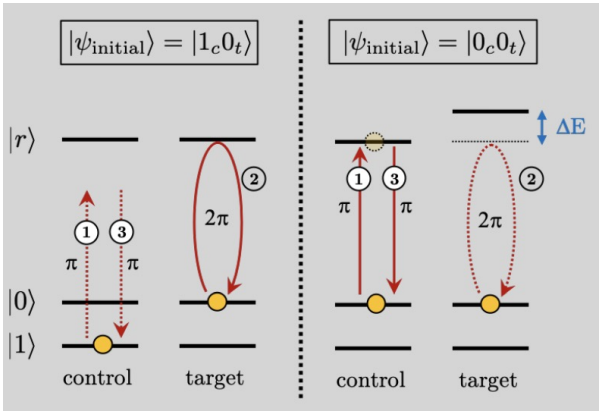


Figure 2.8: First a π pulse is applied to the control atom. Next, a 2π pulse is applied to the target atom. Finally, another π pulse is applied to the control atom [11].

To demonstrate how Rydberg states can carry out a two qubit gate, we use a CZ gate as an example. The CZ gate uses one control and one target qubit. First, a π pulse is applied to the control qubit. If the control qubit is in state $|1\rangle$, it is excited to the Rydberg state and forms a Rydberg blockade for nearby atoms in the blockade radius. If the control qubit is in state $|0\rangle$, nothing happens. Next, a 2π pulse is applied to the target atom. If the control atom was excited to the Rydberg state and the target qubit is in the state $|0\rangle$, the 2π pulse

is off-resonant and has no effect on the target qubit. If the control atom was excited to the Rydberg state and the target qubit is in state $|1\rangle$, the pulse attemps to excite the target qubit to the Rydberg state but fails to do so because of the Rydberg blockade of the control qubit. Thus, it instead induces a phase shift of -1 as expected from a CZ gate. If the control qubit remains in state $|0\rangle$ before the second pulse on the target qubit, there is no Rydberg blockade and the 2π pulse is either off-resonant if the target qubit is in state $|0\rangle$ or the pulse simply excites the target qubit to the Rydberg state and brings it back down to its starting state of $|1\rangle$, causing no change to the qubit. Finally, a π pulse is again applied to the control qubit, this either calms the control qubit back down from the Rydberg state to $|1\rangle$ or is off-resonant and leaves the control qubit as it is in state $|0\rangle$.

2.2.4 Advantages

The neutral atom architecture provides many unique advantages over other quantum computer architectures. One of these advantages is increased connectivity. When a quantum computing platform has little connectivity, SWAP gates are needed to transfer the information from one qubit to another. When a SWAP gate is not a native gate of the architecture, it must be decomposed into multiple gates that carry out the equivalent operation. This increases circuit depth and can be costly in algorithms which require many interactions between qubits [10]. However, neutral atom quantum computers mitigate this cost because all atoms within the Rydberg blockade can execute gates with the excited Rydberg atom without the use of SWAP gates. The Rydberg blockade radius can be adjusted, allowing it to extend either to the nearest neighbor of the Rydberg atom or to multiple trapped atoms, depending on the desired interaction range. The larger the connectivity radius, the less overhead is needed to implement additional gates [11]. Although a larger range brings higher connectivity, there is a trade-off with the atoms caught in the cross-fire of the Rydberg interaction [13]. Even atoms that are not meant to be part of the operation, but are in the restriction radius, are affected by the dipole-dipole forces and cannot execute other operations in the meantime. Besides

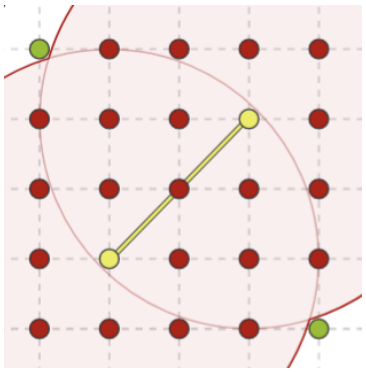


Figure 2.9: The interaction range is shown as the yellow line between the two qubits and the restriction radius is shown as red [13].

higher connectivity, another major benefit of the neutral atom architecture is the ability to

natively realize multi-qubit gates without decomposition. This can make a huge difference in circuit depth and execution time because these multi-qubit gates can be realized in just a few pulses instead of the many pulses needed for the many gates resulting from decomposition. An important example for later experiments is the CCZ gate. When decomposed, a CCZ gate requires a combination of 8 single qubit U3 gates and 6 two qubit CZ gates to simulate an equivalent operation. While these decomposed gates would need 26 pulses, a native CCZ gate only needs 5 pulses [2]. Fewer pulses and less gates creates a more favorable circuit because of reduced noise and execution time.

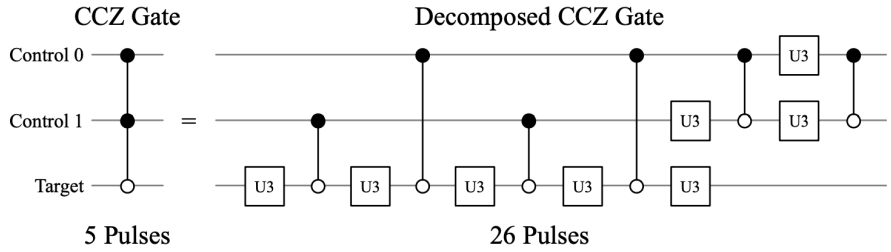


Figure 2.10: The circuit shows the equivalent decomposition of a CCZ gate into U3 and CZ gates, demonstrating the efficiency of the native implementation of the CCZ gate in neutral atom architectures. [2].

Depending on the Rydberg blockade radius, the architecture can realize any n-qubit gates. Theoretically, the interaction radius of the Rydberg blockade could span across the entire qubit array. However, this is not ideal since a larger Rydberg blockade comes with a larger restriction radius. When more qubits are restricted, parallelization decreases and the circuit must be executed in serial [14]. Therefore, there must be a balance between prioritizing multi-qubit gates and parallel execution. This choice is dependent on the program being executed. A program that uses multi-qubit gates and has a serial manner of gate execution could benefit most from native multi-qubit gates. In the case of this work, the multi controlled Z-gate is an important example of the neutral atom capabilities.

To implement a C_kZ gate with k control qubits, there are two options: using a laser that individually addresses qubits or using a laser that globally addresses qubits. A laser that individually addresses qubits requires more beams since each laser beam is targeted at one qubit at a time. It begins by sequentially targeting each of the control qubits with a π pulse. This excites each control qubit into the Rydberg state if it is in state $|0\rangle$ beforehand. Next, a 2π pulse is applied to the target qubit. If none of the control qubits are excited into the Rydberg state from the first k pulses, the 2π pulse applies the Z gate on the target qubit. Thus, the target changes phase ($|1\rangle \rightarrow -|1\rangle$) or remains the same($|0\rangle \rightarrow -|0\rangle \rightarrow |0\rangle$). If any control qubits are excited to the Rydberg state, the 2π pulse takes no effect on the target qubit because it is hindered by the Rydberg blockade. Lastly, each control qubit is hit with a π pulse again in reverse order this time to bring any excited control atoms to the ground state. This means there are k more pulses, and in total, $2k + 2$ pulses to complete the entire C_kZ

gate [15].

Alternatively, it is possible to globally address the atoms by using a laser that targets multiple atoms simultaneously. In the context of a C_kZ gate, this only requires four laser beams [15]. A π pulse inducing the Rydberg state $|s\rangle$ to every qubit in state $|0\rangle$ is applied to all control qubits at once using a single pulse. It is crucial that this specific Rydberg state level chosen does not interact with other atoms in the same state so that any control qubit can be excited to the Rydberg state. As with individual addressing, a 2π pulse is then applied to the target qubit to induce a Z-gate if no Rydberg blockade is present from the control qubits. Finally, another single global π pulse is applied to the control qubits simultaneously, bringing them back to their original state.

Unlike the trapped-ion quantum computing architecture, neutral atom quantum computing uses atoms instead of artificial qubits. Nature rules that atoms with the same atomic makeup are completely identical, ensuring that each qubit is uniform. Having innately identical qubits reduces the error rate because it avoids any deviations in the qubits during their manufacturing process. Additionally, the atoms are held in place by the SLM traps, which can be changed and adjusted to different patterns without having to rebuild the hardware as one typically would have to do with, for example, a superconducting quantum computer [11]. With guaranteed uniformity of qubits and adjustable quantum registers, neutral atom quantum computing offers high scalability. Its scalability is limited only by decoherence and laser power.

2.2.5 Limitations

Although the neutral atom architecture presents many advantages over other platforms, its performance remains limited due to errors. These errors can be classified into two categories: idle errors and gate errors. Idle errors occur even when no operation is being executed on the atom and gate errors occur as a result of applying lasers to trigger gate operations.

Idle Errors

A major culprit in the limitations of neutral atom architecture is decoherence and noise. When an atom interacts with its environment, it loses the desired quantum properties needed for computations. Amplitude damping falls under this category of errors. It occurs when atoms decay from their higher energy state to their ground state. The time for this to occur is called the relaxation time, T_1. Another challenge occurs when atoms inevitably interact with their environment, such as the magnetic fields or gases in the vacuum, and undergo dephasing. Dephasing alters the atom's phase so it is compromised in the computation and unusable. The time it takes for the atom to dephase is the dephasing time, T_2. Equation (2.13) demonstrates the complete coherence time for atoms in neutral atom quantum computers [13].

$$T_{\text{eff}} = \frac{T_1 T_2}{T_1 + T_2} \tag{2.13}$$

Another idle error source is atom loss. One way atom loss can occur when the qubit atom collides with a rogue atom existing in the vacuum of the array. Luckily the chances are very low of this happening, with a probability of 0.0068 [14]. A bigger risk is atom loss through readout. When the fluorescent imaging hits an atom not in the targeted state, it can be ejected out of the trapping lasers. Atom loss from readout happens 50% of the time [14]. The atoms lost can be replaced, but this can be costly so it is better avoided. It is also a hurdle because quantum algorithms are usually run hundreds of times, meaning high atom loss requires rerunning the program many more times than would be necessary if the atoms were stable.

These errors are a hardware problem but can be mitigated with software solutions [14]. Since these errors occur within a certain time frame, it is favorable to reduce the time of execution. This can be done by parallelizing gates and reducing circuit depth at compilation. This helps to not only reduce the computation time, but also reduces noise.

Gate Errors

Neutral atom quantum computing relies on Rydberg states to execute multi-qubit gates and allow for long range interactions. Despite this being an advantage to the platform, it brings its own problems. Controlling Rydberg states while avoiding noise is challenging. Exciting an atom to a Rydberg state introduces computation errors and leakage errors. Computation errors occur when the Rydberg state decays before the desired operation on the atoms is complete and leakage errors occur when the qubit unintentionally transitions to a different electronic state [13].

A significant source of computation errors is finite temperature Doppler dephasing. This occurs when atoms in their trap are not properly cooled, and thus not still enough to avoid Doppler detuning when excited by the Rydberg state. In essence, small shifts in the atom due to motion can cause dephasing and cause errors [13]. Another root cause behind computational errors is the physical characteristics of the laser beam used to induce the Rydberg state. The amplitude, phase, and frequency noise must be consistent. In addition, the beam must precisely target the atom [13].

Focusing on unwanted phase changes, or leakage errors, a major hurdle is spontaneous emission from intermediate states [13]. There is a risk that the atom decays from the Rydberg state before the desired operation is complete, which would disrupt the calculation of the circuit. Rydberg states also present a sensitivity to the noise of background electric and magnetic fields [13]. They can be altered by this noise and knocked out of the desired computational basis, again compromising the circuit.

Another limitation is gate fidelity, which measures how close the executed gate resem-

bles the desired operation compared to the actual operation carried out. A higher fidelity is more desirable as it means the operation was closer to the ideal gate. Single qubit gates in neutral atom quantum computing have proven a reliably high fidelity of around 0.99 [13]. Unfortunately, multi-qubit gates are the problem. A CCZ gate, for example, has a fidelity of about 0.95, which is not enough for a reliable computation [13]. Despite the low fidelities of multi-qubit gates, it is actually still favorable to use them over multiple single qubit gates. Using multi-qubit gates decreases circuit depth and execution time, ultimately increasing the circuit fidelity despite having low fidelity by itself [16].

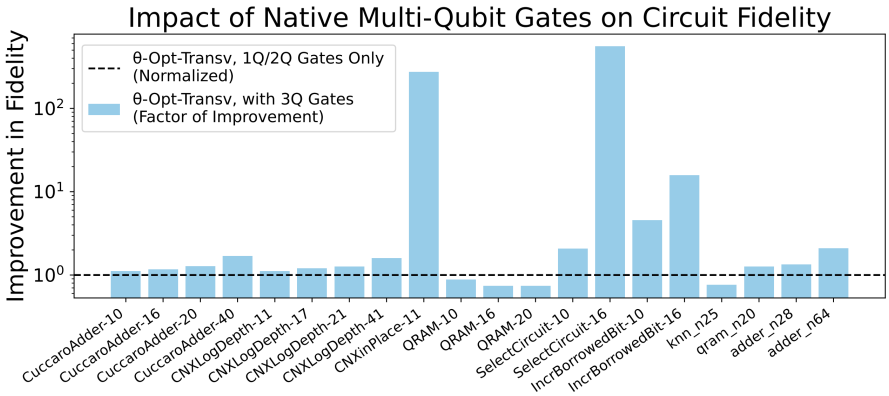


Figure 2.11: The graph pictures shows the increased circuit fidelity when using three-qubit gates as compared to single and two- qubit gates [16].

2.3 Compiling on Neutral Atom Hardware

This next section discusses compilation specific to the neutral atom platform. Like its meaning in traditional computing, compilation in quantum computing is translating abstract high-level instructions, like a circuit of gates, into low-level operations on hardware [13]. While there have been numerous developments in compilation strategies, this work uses the Geyser framework to conduct experiments because it takes advantage of the native multi-qubit gates of neutral atom quantum computers [7]. Thus, the next sections will outline each step involved in the compilation process of the Geyser framework: circuit mapping, circuit blocking, and block composition.

2.3.1 Circuit Mapping

As mentioned in 2.2.2, the physical qubits can be arranged in any shape or formation. The Geyser framework chooses to arrange these qubits in a triangular grid. One major advantage of this is the equidistance of each qubit to its neighbor. This ensures the interaction radius, and thus the restriction radius, does not need to be larger to reach farther qubits. In a square grid, for example, the diagonal neighbor of each qubit is farther away than its horizontal

neighbor, requiring a larger blockade radius. This in turn results in 12 restricted qubits that must remain idle until the operation is over. In comparison, a triangular grid only restricts 9 qubits [2]. In addition, the Geyser framework relies on using three-qubit gates later in the compilation process. Therefore, a triangular topology is a suitable choice to have 3 qubits in the interaction radius for a three-qubit gate. The first step in mapping

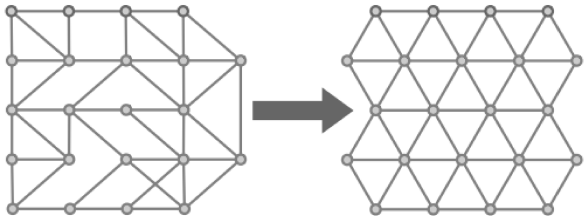


Figure 2.12: The logical circuit is mapped to a physical triangular topology of qubits [2].

qubits to this topology is decomposing the circuit into the native gates of the neutral atom platform, this set is known as $\sum^{\text{native}}$. In the case of Geyser, the gate set $U3, CZ$ is used. Since these gates are also native to the superconducting quantum computer, Geyser adapts this mapping strategy and uses Qiskit's transpile function to decompose the gates of the circuit into these gates [2]. Every quantum computer needs to be able to execute any quantum computation with its gate set, meaning it must have a universal gate set [13]. Thus the native gate set must also be a universal gate set. In this case, $\sum^{\text{native}}$ contains U3 to perform any arbitrary single gate rotation and CZ allows entangling of qubits so the set qualifies as universal.

Next, the triangular grid of qubits is split into three qubit blocks. Each block is tested against every other block to see if they can be actively used in an operation simultaneously without interfering with one another. Essentially any block of qubits within the interaction radius of another cannot be executed in parallel [13]. To differentiate, these blocks refer to the groups of three physical qubits in space, not to blocks of gates in the circuit as will be seen in the next section. For clarity, this work will use the term sections instead of blocks to refer to the physical three qubits.

2.3.2 Circuit Blocking

A unique characteristic of Geyser is that it uses pulse minimization instead of gate count to optimize the circuit. This is because less gates do not always mean an efficient circuit because not all gates require an equal number of pulses. For example, a CZ gate requires only 3 pulses while a CCZ gate requires 5 pulses [2]. From this it can be concluded the noise of the circuit depends on the number of pulses, not the number of gates. Using this evaluation criterion to optimize each block, the circuit blocking step of Geyser divides the circuit into blocks of gates that can be executed in isolation of other blocks. A qubit can be part of multiple blocks during the course of the circuit but can only be in one block at a time [2].

To create each block of operations in the circuit, Geyser takes each three qubit section created in the mapping phase and applies the maximum amount of operations possible on the qubits in the section. It tracks how far each qubit has progressed through its gate operations, calling it the "frontier" of the qubit. Each three qubit section is then scored based on the total amount of operations applied to the three qubits in the section. However, since CZ gates have a lower impact on circuit depth than U3 gates they are weighted higher when counted in the section's score. A key goal in circuit blocking is maximizing the size of each block, which

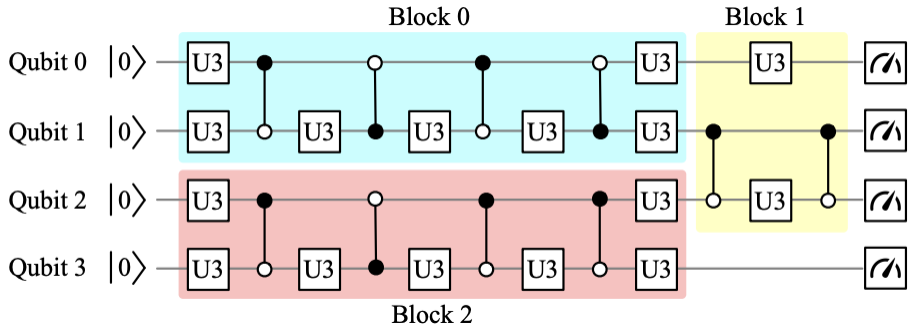


Figure 2.13: The circuit is split into one to three qubit blocks while maximizing block size and parallelism [2].

may help reduce the number of pulses in the last step, block composition. To do this, the framework recursively iterates through each three qubit section and attempts to combine them into block families with other three qubit sections while avoiding overlapping restriction zones [2]. The block family with the best score, meaning most number of operations, is then chosen to be a final block in the circuit.

However, there is a trade-off because the parallelization of blocks is also important to minimize execution time. So Geyser continues the circuit blocking by finding block families that can be executed in parallel without having overlapping qubits or restriction zones until the operations of all qubits have been fully traversed.

2.3.3 Block Composition

The final compilation step is what makes Geyser unique to other neutral atom compilation frameworks. It takes advantage of the platform's native multi-qubit gates to minimize pulse count. Recall from 2.2.4, a CCZ gate only requires 5 pulses as opposed to its decomposed counterpart, which uses 26 pulses. Therefore, Geyser takes each block created in the circuit blocking step and finds an equivalent block with less pulses. Every block can be represented by a unitary matrix, which describes the transformation of the operations applied to the qubits. The goal is to replace the original block with one requiring less pulses but having an almost identical unitary matrix. The similarity of these two blocks is measured by comparing the unitary matrices of each with their Hilbert-Schmidt distance. Hilbert-Schmidt distance between two unitary matrices is measured using equation 2.14. In the equation, $U^{\dagger}$ is the

transpose conjugate of the original unitary matrix, meaning the rows and columns of each element in the matrix are switched and the complex elements in the matrix are replaced by the their complex conjugate counterparts. The variable U' is the unitary matrix of the composed block and n is the number of qubits in the block. The trace of a unitary matrix is calculated by finding the sum of its diagonal elements. The Hilbert-Schmidt distance is in range $[0, 1]$, where a value closer to 1 means a larger distance and value closer to 0 means a smaller distance, and therefore higher equivalency between the two matrices.

$$1 - \frac{\left\|\mathrm{Tr}(U^{\dagger}U')\right\|}{2^n} \tag{2.14}$$

To create the composed block, the algorithm in Geyser first begins by using six U3 gates and one CCZ gate as seen in figure 2.14. This first attempt at a composed block would only need eleven pulses. To make this block as close in distance as possible to the original block, the algorithm optimizes the three rotational angles of each of the six U3 gates and the choice of the target qubit in the CCZ gate. These are 19 parameters in total that must be optimized to ensure that the composed block is as similar to the original block as possible. If the optimization fails to make the distance between these two blocks below a certain threshold, another layer of one CCZ gate and three U3 gates is appended to the block, requiring a total of 19 pulses. Next, the same optimization process is used with the now 29 parameters of the block to decrease Hilbert-Schmidt distance below the desired threshold. If the distance is

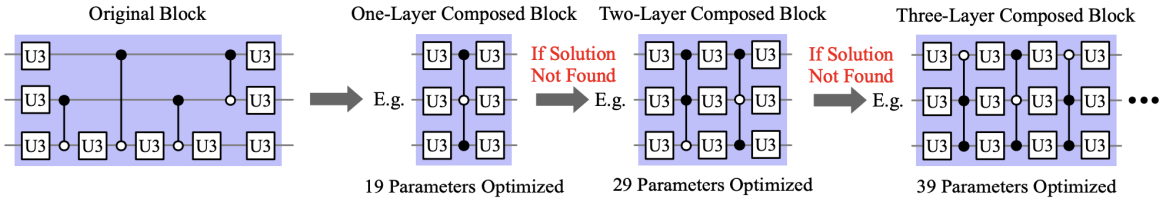


Figure 2.14: An attempt to optimize the block is made, adding a layer if such an attempt is unsuccessful [2].

still not satisfactory, another layer of one CCZ gate and three U3 gates is appended to the composed block. Again, the pulses required for this new composed block increases to 27 pulses and the number parameters to be optimized increases to 39. Each layer added increases the pulses needed by 8 and the parameters for optimization by 10. If the threshold is not met, this process continues and layers are continuously added until the the Hilbert-Schmidt distance is satisfactory or the number of pulses in the composed block is greater than the number of pulses in the original block. After each block from the circuit blocking stage undergoes this composition process, the circuit is then put together using either the original block or the replacement composition block, depending on which one requires fewer pulses.

To optimize the parameters of each layer as mentioned above, Geyser uses a dual annealing algorithm. Dual annealing uses a combination of two algorithms: simulated annealing is

used to find the global optimum, and the L-BFGS-B algorithm (Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Box Constraints) is used to refine the local optimum [17].

Thus, it is important to first understand simulated annealing. Simulated annealing comes from the process of physical annealing, where a crystalline solid is heated and then cooled very slowly to create a near perfect crystal lattice structure [18]. This phenomenon inspired the strategy behind simulated annealing to find the global minima in a high-dimensional search space. Before starting the algorithm, the cost function $f : \Omega \to \Re$ must be defined, which in the case of Geyser is the Hilbert-Schmidt distance between the original unitary and the composed block. The solution space, Ω is the combination of different parameter options for the U3 gate rotational angles and the target qubit of the CCZ gate. The objective is to find the global minimum $w* \in \Omega$ where $f(w) \geq f(w*)$ for all $w \in \Omega$ [18]. The algorithm begins by choosing a solution $w \in \Omega$ at random. Another solution $w' \in N(w)$, with $N(w)$ being the neighboring solutions of solution w, is chosen at random or using a rule. The unique advantage of simulated annealing is that it sometimes chooses a worse solution candidate to avoid getting trapped in a local minimum. Thus, the decision of whether to continue with solution w or w' is chosen based on the probability of the Metropolis Acceptance Criterion, seen in equation (2.15) [18].

$$P\{\text{Accept } \omega' \text{ as next solution}\} = \begin{cases} \exp\left[-\frac{f(\omega')-f(\omega)}{t_k}\right], & \text{if } f(\omega') - f(\omega) > 0, \\ 1, & \text{if } f(\omega') - f(\omega) \leq 0. \end{cases} \tag{2.15}$$

The probability of accepting a worse solution candidate, or a higher Hilbert-Schmidt distance in this case, is also influenced by t_k, the temperature parameter at iteration k. As with physical annealing, this temperature must be slowly lowered with each iteration of the algorithm to hone in the best solution. With each iteration, the probability of accepting a worse solution decreases with $\lim_{k\to+\infty} t_k = 0$ [18]. Once a global optimum is chosen by simulated annealing, the second step of dual annealing is finding the local optimum using the L-BFGS-B algorithm.

L-BFGS-B is an optimization algorithm suitable for high dimensional cost functions with bounded parameters. Since more dimensions can be costly for calculating gradients, approximating the gradients using a limited history of gradients saves memory in contrast to storing them in a full Hessian matrix [19]. The algorithm favors parameters in the direction of the steepest descent to then find the minimum. If this dual annealing process finds a solution within the specified threshold of the Hilbert-Schmidt distance after a limited amount of iterations, the composed unitary replaces the original unitary. As mentioned earlier, if the threshold is not met, another layer is added onto the composed block until its pulse count exceeds the original block.

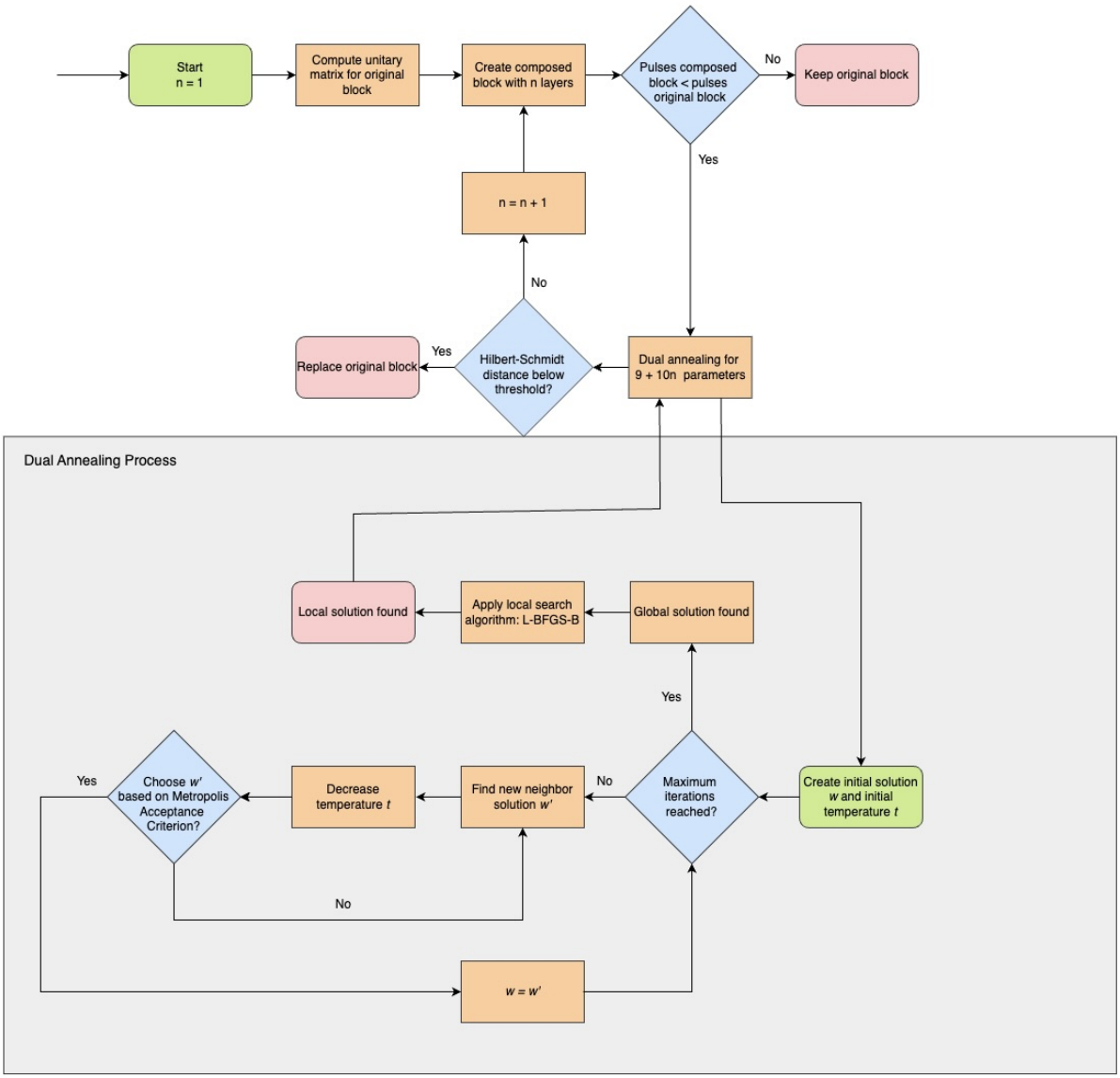


Figure 2.15: The flowchart demonstrates the decision flow of the Geyser block composition step along with the dual annealing process it uses.

2.4 Gate Cutting

Near-term quantum computers still face many challenges in proving their quantum advantage because of their limited connectivity, high error rates, and limited number of qubits [20]. To counter these limitations, researchers have attempted to reduce circuit depth and size by reducing the circuit size through decomposition in a process called circuit knitting. Circuit knitting can be done by either gate cutting, wire cutting, or a combination of the two [21]. Gate cutting, as the name suggests, refers to splitting a gate in a quantum circuit in a "space-like" manner. Wire cutting, which has garnered more attention in the scientific community, refers to cutting a wire in a "time-like" manner. The difference between these two types of circuit knitting are visualized in figure 2.16. Because the experiments in this work utilize gate cutting, further explanation of wire cutting is out of scope.

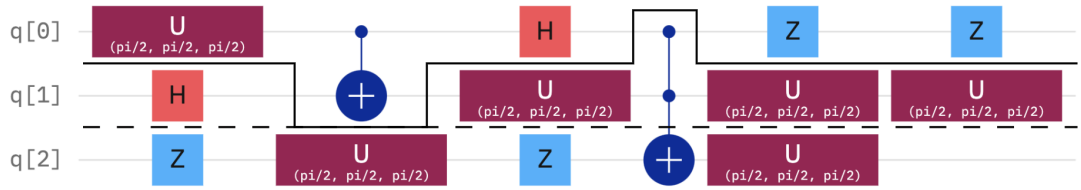


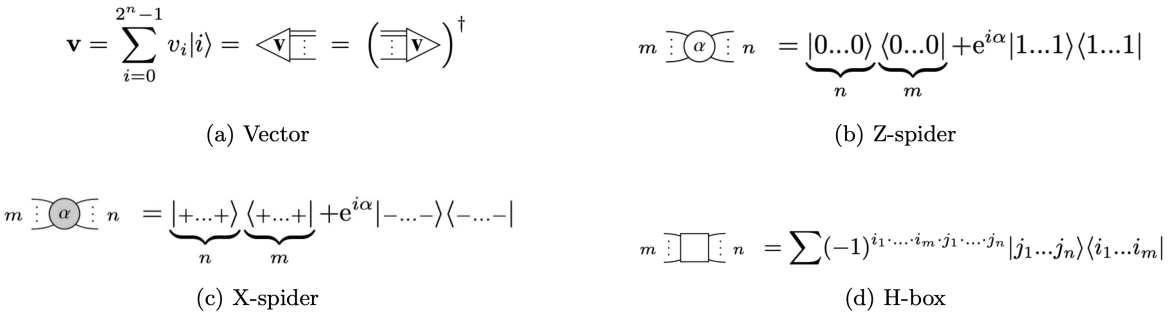
Figure 2.16: The circuit demonstrates a gate cut through the CCX gate using a dotted black line and a wire cut using a filled black line.

Most strategies for gate cutting simulate entanglement using sampling and post-processing. Chefer et al. (2022) explain the use of quasiprobability simulation as using the expectation value of samples from a circuit where the nonlocal operations are replaced by local operations [22]. This allows the circuit to by divided and executed on separate quantum computers. The disadvantage of this method is that it is probabilistic and, therefore, requires post-processing to reconstruct the measurements of the original circuit. If quasiprobability simulation is not used in combination with classical communication, the overhead is $O(9^n)$, with n being the number of cut gates [23]. The cost increases with cut count, making it costly to implement at larger scales. In addition, these methods focus on cutting two-qubit gates. The experiments in this work use instead an alternative method proposed by Ufrecht et al. (2023) that focuses on cutting multi-controlled Z gates (MCZ) with three or more qubits using ZX calculus [3]. Thus, the following section will explain how this specific gate cutting tactic is employed.

2.4.1 Gate Cutting using ZX calculus

ZX calculus is the basis for the decomposition of MCZ gates proposed by Ufrecht et al. (2023) [3]. ZX calculus is a diagram notation that can be used to represent any quantum circuit. The reason it can be helpful in optimizing quantum circuits is its versatile rewrite rules and inherent symmetric properties [24]. Any ZX diagram wires can be moved around at will,

only the inputs and outputs of elements must remain the same. This section will begin by explaining the basic elements of ZX calculus and continue by introducing rules that contribute to the decomposition of MCZ gates.



(a) Vector

(b) Z-spider

(c) X-spider

(d) H-box

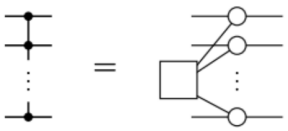
Figure 2.17: The basic elements of ZX calculus used for gate cutting [3].

To begin, one of the simplest elements in ZX calculus is a triangle representing an n-qubit column vector [3]. This triangle is hollow with the vector name v inside. The wires representing the output of the triangle correspond to one qubit each. For example, a triangle with one wire exiting it would represent a two-dimensional vector.

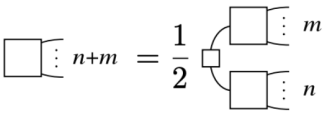
A central element of ZX diagrams are Z-spiders and X-spiders. Visually, Z-spiders are pictured as a white dot, representing an operation, with an arbitrary amount of wires, representing quantum systems, being input and output from it. Inside of the white dot is angle α, which is the $R_z(\alpha)$ phase gate of the operation. If $\alpha = 0$, the white dot is left blank and is assumed to have angle 0. The reasoning behind the name Z-spider is that these operations are defined on the eigenbasis of the Z matrix using $|0\rangle$ and $|1\rangle$ [24]. Following the same structure, X-spiders are essentially the same as Z-spiders except their center dot is grey instead of white and they perform a different operation. As the name again implies, X-spiders use the eigenbasis of the X matrix, $|-\rangle$ and $|+\rangle$ [24]. Figure 2.17 shows the interpretation of the Z-spider's and X-spider's linear map into Dirac notation.

Because it is difficult to concisely represent controlled multi-qubit gates with just ZX calculus, ZH calculus introduces the structure H-box as an extension [24]. An H-box operation represents a matrix filled with 1 in every position except the bottom right corner, which has the value a [24]. It can be seen as a rescaled Hadamard gate, with an arbitrary number of inputs and outputs. An H-box is pictured as a hollow square containing the value a inside, with any amount of wires connecting to and from it. As with spiders, the box is left empty if $a = 0$.

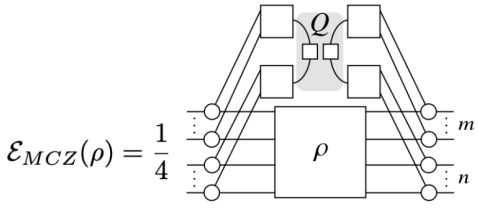
These elements along with some rewrite rules are all that is needed to decompose an MCZ gate using ZX calculus. The following decomposition steps were established by Ufrecht et al. (2023) [3]. The steps are visualized in figure 2.18. In step one, an MCZ gate is represented as



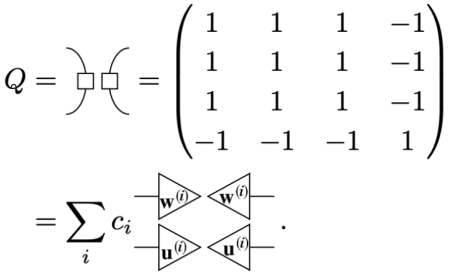
(a) In step 1, the MCZ gate is represented as an H-Box with output wires



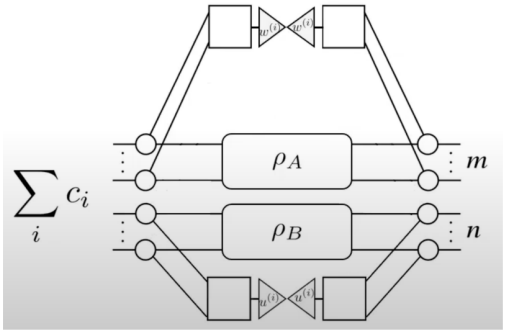
(b) Step 2 applies the H-box fusion rule.



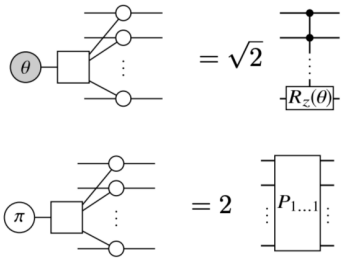
(c) Step 3 incorporated the MCZ gate into an arbitrary circuit.



(d) In step 4, rank one decomposition separates region Q into vectors.



(e) Step 5 is just the split gate within the circuit.



(f) Step 6 uses these identities to replace the vectors with unitaries and projectors.

Figure 2.18: The process of splitting an MCZ gate into two separate partitions [3].

an H-box with zero input wires. Since an MCZ gate has the matrix form diag(1,1,...,-1), an H-box is a fitting portrayal. The actual decomposition begins in step 2, where the H-box is split into a scalar of $\frac{1}{2}$, a Hadamard gate, and two H-boxes on the basis of the H-box fusion rule. Next, in step 3, the split H-box depiction of the MCZ gate is applied to an arbitrary circuit, where the sole connection between the circuit lies between the two Hadamard gates of the split H-boxes (this point of connection will be referred to as Q). To continue, Q is a 4 by 4 matrix that can be decomposed. Step 4 decomposes Q using rank one decomposition to factorize the matrix into single qubit vectors $\boldsymbol{w}^{(i)}$ and $\boldsymbol{u}^{(i)}$. At this point, the gate is effectively cut since the two partitions are now separated, pictured in step 5. Finally, in step 6, the partitions are altered into projections or unitary rotational gates which are feasible for hardware execution. Ufrecht et al. (2023) provide a more depth explanation for the substitutions and changes leading to step 6 [3].

The final decomposition of any size MCZ gate is shown in figure 2.19. The P operation in the figure represents an intermediate measurement and the Z operation sums all combinations of one-qubit identities and Z gates over each unitary channel (pictured in figure 2.20). This finding allows MCZ gates to be computed separately since it breaks the dependence of the qubits on each other. Using this technique, the sampling overhead is independent of the number of control qubits, being $O(4.5^{2k})$. For a CCZ gate, it even drops to $O(4.5^{2k})$.

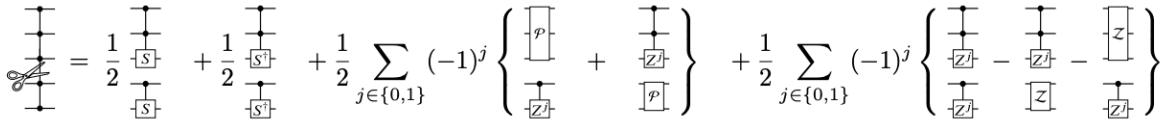


Figure 2.19: The full decomposition of a cut MCZ gate [3].

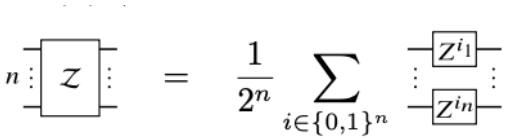


Figure 2.20: The Z gate in its equivalent form as a sum of combinations of one-qubit identities [3].

3 Methodology

3.1 Motivation

The Geyser framework provides a strategy to minimize pulse count by replacing parts of the circuit with more efficient blocks containing multi-qubit gates. As mentioned in 2.3.2, when dividing the circuit into blocks in the blocking step of Geyser, there is a tradeoff between maximizing parallelization and finding the largest possible blocks. Parallelization is important to minimize the depth of the circuit, therefore also minimizing execution time. A smaller execution time leads to less idle errors. A larger block helps minimize the pulse count because replacing one larger block with one composed block most likely uses less pulses than replacing two smaller blocks with two composed blocks. Thus, it is favorable to choose larger blocks over smaller blocks when parallelization is not considered. The objective of this work is to increase the size of the blocks in Geyser to ultimately reduce pulse count using the ZX gate cutting technique described in 2.4.1. Consider the following as pictured in figure 3.1: a circuit with four partitions connected to each other by only one multi-qubit gate. If this circuit is compiled using the Geyser framework, the circuit blocking step must keep each partition separate because the multi-qubit gate prevents combining the blocks without losing parallelism. To alleviate the separation of the blocks, this work uses gate cutting to split the multi-qubit gate into separate partitions of qubits so that the blocks can be combined. On this basis, the following experiments will test if cutting the gate results in more efficient compilation using Geyser.

3.1.1 Hardware and Software

This work was carried out on the LRZ Linux cluster with access gained through LMU. The specific cluster used was "*serial_std*", which is best utilized for serial jobs. It utilized 8 physical cores along with 8 logical cores and 50 GB of memory. It uses Qiskit 0.18.3 to simulate the circuits.

3.1.2 Experimental Setup

Each trial compares the effectiveness of using gate cutting with the Geyser framework, therefore, each iteration and change in variable was simulated using both the cut circuit and the uncut circuit as the control variable. To implement the cut circuit, the ZX gate cutting steps were applied to the circuit in each trial before the Geyser framework compiled it. This ensured that the multi-qubit gates are not dependent on each other, even when compiled into the native gate set in the first step of Geyser. The circuits used for the experiments were

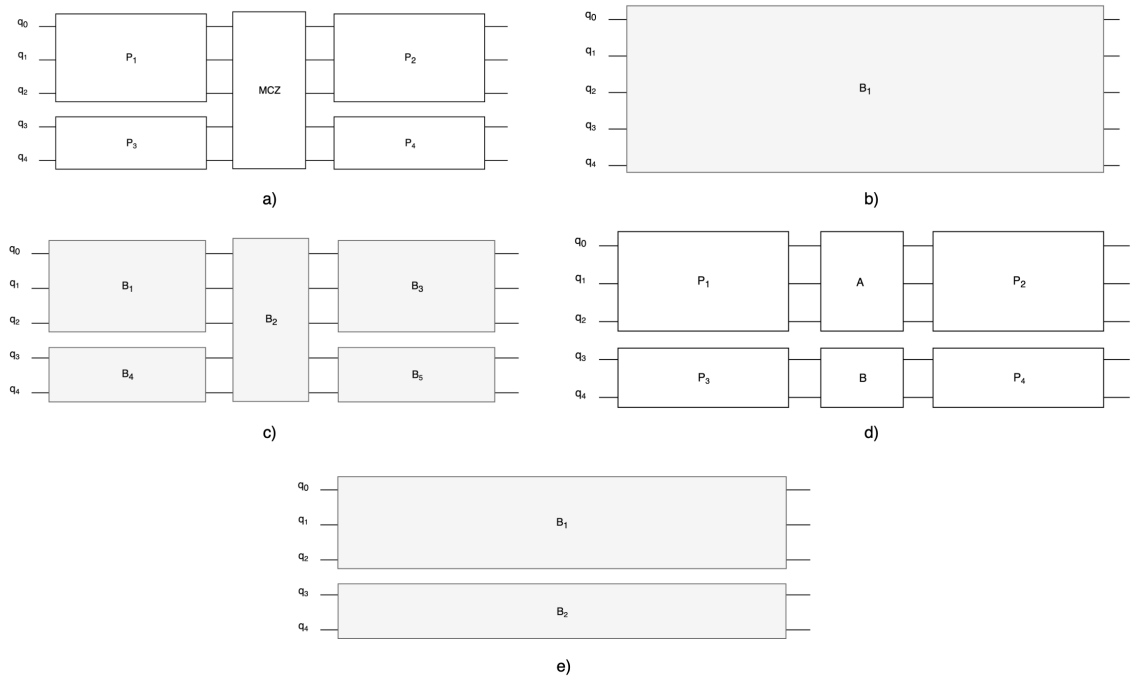


Figure 3.1: a) A circuit with an uncut multi-controlled gate separating partitions P_1 and P_2 as well as P_3 and P_4. b) The first possible blocking arrangement for the uncut circuit where the entire circuit is simplified into one block while losing parallelization. c) The second possible blocking arrangement for the uncut circuit with parallelization but a high number of blocks. d) The circuit after cutting the multi-control gate e) Blocking of the cut circuit where parallelization and large blocks are kept in tact by combining P_1, P_2, and A on the top and combing P_3, P_4, and B on the bottom.

altered based on two variables: gate count and qubit count. For each of these experimental parameters, three different values were tested to see how the compilation responded under different conditions. In each experiment, each value was simulated a total of 20 times, meaning each experiment consisted of 60 iterations for the cut circuit and 60 iterations for the full circuit. The number of trials was limited due to resource exhaustion. Besides the variables, the circuits tested were generated at random using a seed. However, the circuit was always the same for the full circuit and the cut circuit for each trial. To evaluate the performance of each compilation, the assessment criteria used were block count from the second Geyser step, pulse count, time to run, and Hilbert-Schmidt distance of the composed circuit in step 3 of Geyser to the original circuit.

4 Evaluation

4.1 Impact of Gate Count

The first experimental trials compared the performance of the Geyser compilation with circuit cutting to the original Geyser method by itself using various gate counts. The gate count values were set to 750, 1000, and 1250 gates. The block count, pulse count, duration, and H-distance metrics were used to compare the two methods at each value.

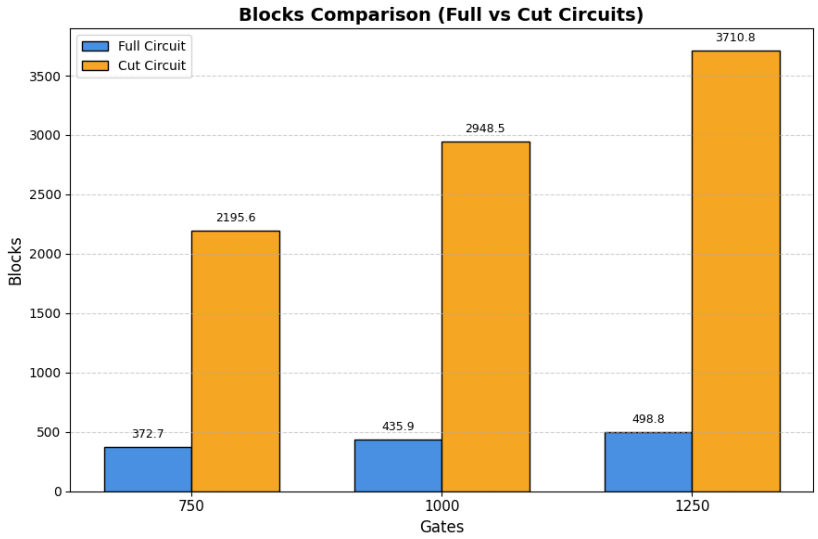


Figure 4.1: The average number of blocks computed in Geyser's circuit blocking step for uncut and cut circuits with 750, 1000, and 1250 gates.

The number of blocks for the cut circuit is approximately 7.5 times as many as for the full circuit at its worst and 5.9 times as many at its best. In each trial, there was clearly a larger block overhead for the cut circuit. In addition, an increase in gates caused a much steeper increase in blocks in the cut circuit than the full circuit. The cut circuit showed a 69% increase in blocks from a circuit with 750 gates to a circuit with 1250 gates, while the full circuit's pulses only increased by 34%.

The large discrepancy in the number of blocks needed for the cut circuit is most likely due to the sampling overhead of the cut circuit. It requires many more runs with different parts of the circuit, as seen in the decomposition in figure 2.19.

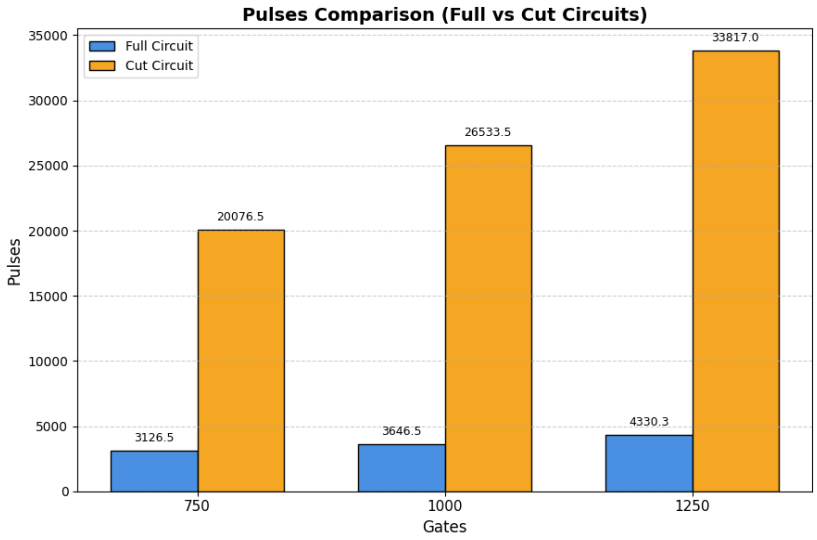


Figure 4.2: The average number of pulses for cut and uncut circuits needed to compute the composed blocks after Geyser's third step: block composition.

Just like the block count, the pulse count also demonstrates a notable difference between the cut and uncut circuits. The largest difference is seen with circuits using 1250 gates, where cut circuits require 7.8 as many pulses as full circuits. The smallest difference is seen in circuits with 750 gates, where cut circuits necessitate 6.4 times as many pulses as full circuits using the standard Geyser framework. As with the block count, the increase in pulses requires is also sharper for the cut circuit. The increase in pulses for the cut circuit from 750 gates to 1250 gates is 68%, while only 38% for the uncut circuit.

Although not precisely the same, these numbers reflect the same pattern seen in the block count. Again, the likely reasoning behind this is the costly sampling required for cut circuits. Even if one cut circuit by itself were more efficient than the uncut circuit, having to rerun it repeatedly adds to the block and pulse count. With this pulse count overhead, the noise created by the cut circuit demonstrates that this method is not a suitable addition to the Geyser framework in neutral atom compilation.

For 750, 1000, and 1250 gates, the cut circuit needed 25.3, 25.3, and 24.9 as long, respectively, as the uncut circuit. As opposed to the block and pulse count metrics, this gap in the overhead for the cut circuit did not correlate with the gate count. The increase in time was slightly more substantial with an increased gate count for the cut circuit, where the time increased by 68% from 750 to 1250 gates. The increase for the uncut circuits was only 40% in comparison.

The gap between the cut and uncut circuits can again be explained by the post-processing

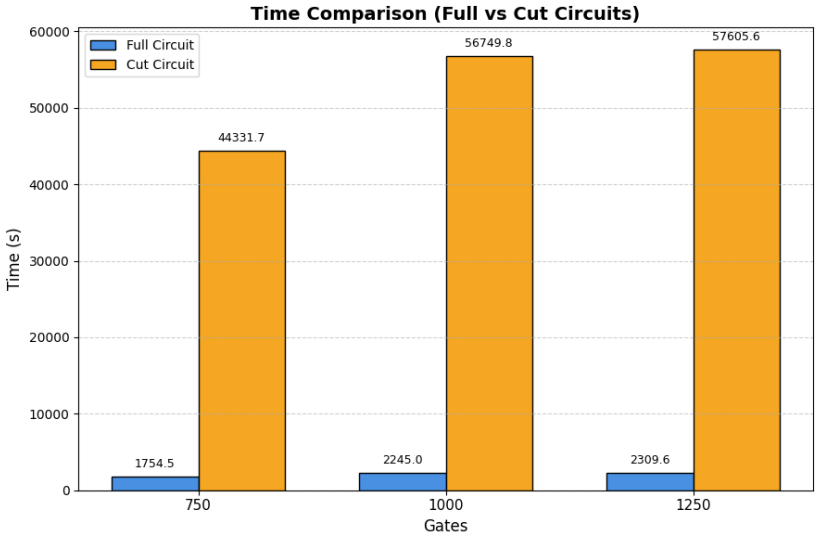


Figure 4.3: The average duration in seconds for cut and uncut circuits needed to compute the entire Geyser compilation process for the uncut circuit and the decomposed circuit.

sampling required by the cut gate decomposition. The time takes 25 times as long for the cut circuit on average, meaning that the cut circuit may have had 25 terms on which it had to apply Geyser to due to sampling.

After the consistently worse performance of the gate decomposition strategy, the Hilbert-Schmidt distance metric was a surprise. It was the only metric where the cut circuits performed notably better than the uncut circuits. For both the uncut circuit and cut circuit, there was variability in the distance that did not correlate with gate count. Cut and uncut circuits with 750 gates resulted in the highest distance. Compared to the lowest distance at 1000 gates, the highest distance for the uncut and cut circuits was 22 times and 5 times higher, respectively. However, the standard deviation showed that the results were not consistent for either type of circuit. Table 4.1 shows the strikingly high standard deviation. Thus, these results may not be as meaningful.

Because of the limitation of resources used in this experiment during the block composition step of the Geyser framework, it may be that the optimization of the dual annealing did not have enough resources to find an equivalent block for each block from the circuit blocking steps of Geyser. The dual annealing optimization in this work only called the local search function a maximum of 200 times and the global search function a maximum of 50,000 times. In contrast to the authors of the original Geyser framework who used 1000 and 100,000,000 local and global function calls, respectively, the optimization values in these experiments are significantly less. Thus, the optimization may not have been able to find a composed block to replace the original block with, thus leaving the H-distance between the two at zero since

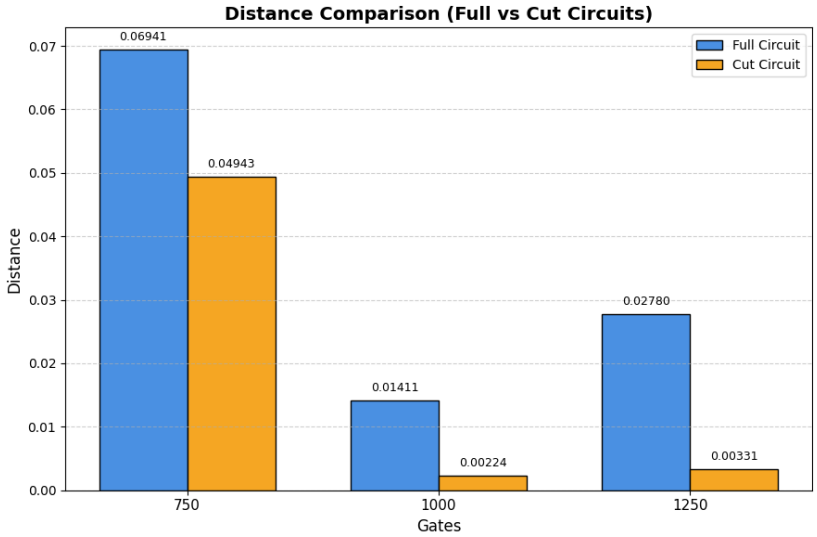


Figure 4.4: The average H-distance between the blocked and composed circuit after the completion of Geyser's compilation steps.

Gate Count	Full Circuit Mean	Full Circuit Std Dev (%)	Cut Circuit Mean	Cut Circuit Std Dev (%)
750	0.06941	318.3%	0.04943	427.8%
1000	0.01411	218.3%	0.00224	87.0%
1250	0.02780	163.6%	0.00331	73.0%

Table 4.1: Mean values and standard deviations (as a percentage of the mean) for full and cut circuits at different gate counts.

they remain identical. This is not ideal, since a composed block would require less pulses. Another potential reason could be that the decomposed gate must apply Geyser on parts of the circuit when sampling instead of the entire circuit. With less gates and less qubits, it is possible that a lack of flexibility during the blocking step created very small blocks since a partial circuit of, say 3 qubits, can only split these qubits into a block of 1 and 2 qubits or keep it as a block of all three qubits.

4.2 Impact of Qubit Count

The following experiments tested the decomposition method against the original Geyser method at different values of qubits in the circuit: 6, 8, and 10 qubits. In both methods, more qubits suggested more possibilities for parallel blocks during block scheduling in the second circuit blocking step of Geyser. Any qubit range below 6 was not tested because the decomposition step divides the circuit into subcircuits of qubits, meaning one subcircuit

would have either one or two qubits. This would result in having to apply Geyser to a subcircuit with less than three qubits, which would not be sufficient to create three-qubit gates in Geyser's third step, block composition. These experiments were tested to evaluate the same four metrics: block count, pulse count, duration, and Hilbert-Schmidt distance of the composed circuit.

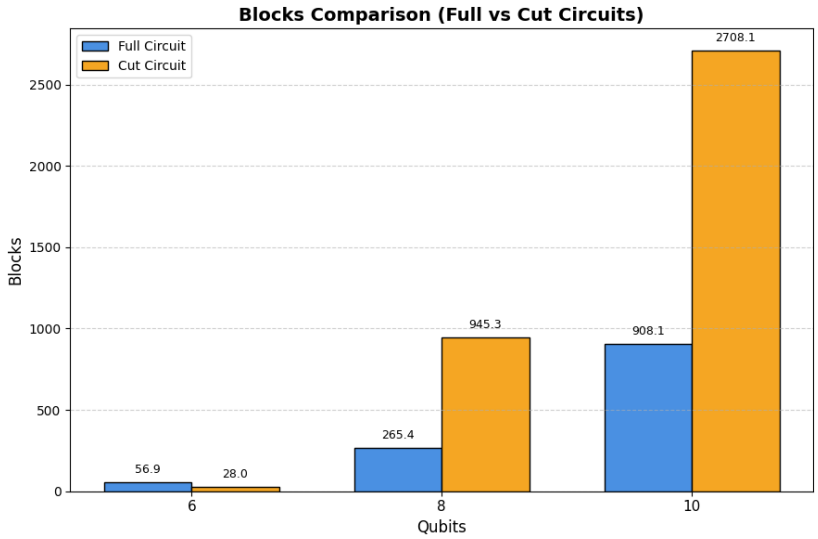


Figure 4.5: The average number of blocks computed in Geyser's circuit blocking step for uncut and cut circuits with 750, 1000, and 1250 gates.

The results for various qubits had a significant impact on the number of blocks formed during compilation for both types of circuit. For the uncut circuit, the blocks increased drastically, with the number of blocks for 10 qubit circuits increasing by 1,596% from the 6 qubit circuits. The cut circuit showed an even more drastic difference, with 10 qubit circuits increasing by 9,672%. However, the cut circuit most likely has a smaller amount of blocks in the 6 qubit circuits because decomposing the circuit results in 3 qubit subcircuits, which cannot be parallelized. Thus, the block in the circuit step of Geyser is always just formed using all three qubits of the circuit, meaning the circuit is blocked into only one block. However, as seen between the 8 qubit circuits and the 10 qubits circuits, there is still a major increase in blocks for both types of circuits because more qubits allow more parallelization of blocks. When comparing the cut and uncut circuit, the largest difference is seen at the 8 qubit mark with the cut circuit having approximately 3.6 as many blocks. The 10 qubit circuits demonstrate a smaller gap with the cut circuit having 2.98 as many blocks. Not considering the 6 qubits circuits because of their lack of flexibility in the blocking step of the cut circuit, the cut circuit required more gates. As in the previous experiments, this is likely again because of the sampling of the circuit which requires applying Geyser to many more circuits than just the full circuit.

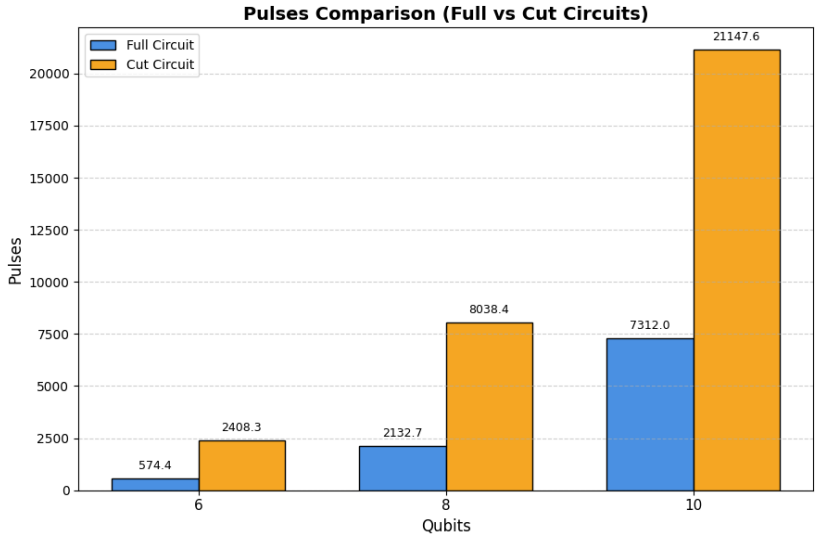


Figure 4.6: The average number of pulses for cut and uncut circuits needed to compute the composed blocks after Geyser's third step: block composition.

The number of pulses executed for the various compilation methods follows a similar pattern as the number of blocks. The uncut and cut circuit's pulses increased by 1,273% and 878%, respectively, from the 6 qubit circuits to the 10 qubit circuits. Surprisingly, the cut circuit needs more pulses than the uncut circuit despite using less blocks. This could be attributed to Geyser's inability to find a an optimized composed block replacement for the large blocks seen in the subcircuit of the 6 qubit cut circuit. In essence, even though the uncut circuit uses more blocks, the blocks it uses are successfully optimized using the block composition step of Geyser to reduce the number of pulses. Besides this anomaly, the pulses for each type of circuit show a clear advantage using the uncut circuit. The number of pulses for the cut circuit is about 3.8 times higher when run on the 8 qubit circuits and about 2.9 times higher when run on the 10 qubit circuits. The inefficiency of the cut circuit can, as with the other experiment, be blamed on the sampling overhead required for the gate decomposition. More circuits leads to more blocks, which ultimately leads to more pulses.

A consistent finding among all experiments was the drastic increase in time to complete the Geyser compilation for the cut circuit. Changing the qubit count did not seem to correlate with the duration of both the uncut and cut circuit. The worst performance for the cut circuit compared to the uncut circuit was evident when the experiments were run on the 8 qubit gates, with cut circuits requiring about 33.6 times as much time to complete. The best performance of the cut circuit was with 10 qubit circuits, requiring 14 times as much time to complete. Despite the time not correlating with the qubit count, it is still clear that the cut circuit is much more resource intensive than the uncut circuit. The most time intensive part of the Geyser compilation process is optimizing the block composition step to find an equivalent

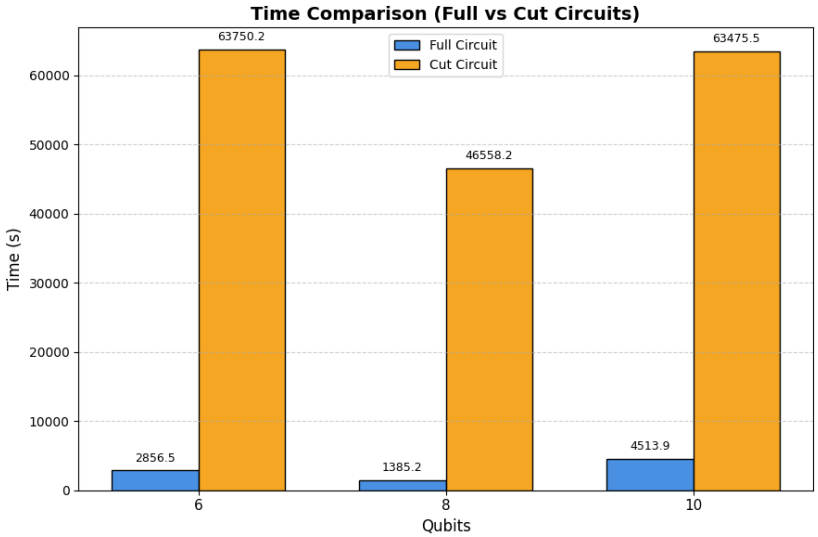


Figure 4.7: The average duration in seconds for cut and uncut circuits needed to compute the entire Geyser compilation process for the uncut circuit and the decomposed circuit.

block with reduced pulses. However, since this optimization can go on an indefinite amount of time, the dual annealing optimization is limited by the number of function calls. For the cut circuit, this limitation applies to each subcircuit of the decomposition step on which Geyser is called. Meaning each subcircuit is allowed the same time to find optimal composed blocks as one full circuit has. This explains why changing qubit count does not impact the duration of the compilation significantly for each method by itself and why the cap for the compilation time for the cut circuit was much higher.

As with the gate count experiments, the Hilbert-Schmidt distance metric between the composed circuit and the original circuit also improved using the cut circuit method. But as mentioned, this is likely due to the smaller quantity of equivalent composed circuits found in the third step of Geyser for the cut circuit. If the original blocks are used instead of composed blocks, the Hilbert-Schmidt distance is, of course, 0. This brings down the average distance for the cut circuit but results in more pulses, meaning it is less efficient. Additionally, the resources in the qubit count experiments were limited as well so the optimization process may have not had enough resources to find valid values in time. This can be demonstrated in the standard deviation for both compilation types as seen in table 4.2, where results are not at all consistent for the circuits distance.

The overall findings suggest that cutting circuits before applying the Geyser compilation method decreases efficiency because of a much higher duration and pulse count. Thus, it is

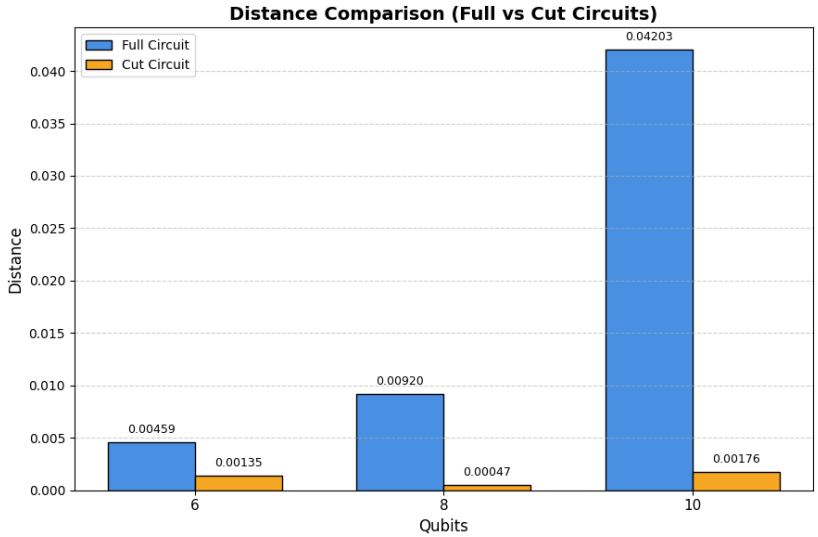


Figure 4.8: The average H-distance between the blocked and composed circuit after the completion of Geyser's compilation steps.

Qubits	Full Circuit Mean	Full Circuit Std Dev (%)	Cut Circuit Mean	Cut Circuit Std Dev (%)
6	0.00459	447.2%	0.00135	124.5%
8	0.00920	447.2%	0.00047	269.1%
10	0.04203	109.7%	0.00176	77.0%

Table 4.2: Mean values and standard deviations (as a percentage of the mean) for full and cut circuits across different qubit counts.

best to leave the circuit uncut.

5 Conclusion

This work sought to explore an adaptation to the Geyser framework by adding ZX gate cutting. It hypothesized that the combination of Geyser's optimization strategy with gate cutting would reduce pulse count and require overall less resources. Contrary to this proposal, the results showed a clear disadvantage when using ZX gate cutting in the compilation. Pulse count and time increased dramatically for every experiment using the gate cutting. In one experiment, the cut circuit's duration, on average, was 33.6 times higher than the Geyser compilation by itself. This aligns with the much higher block count needed for the cut circuit, which, at its worst, was 7.5 times higher. This discrepancy comes from the fact that many more circuits and gates need to be compiled by Geyser when the circuit is decomposed. A larger circuit calls for more blocks, which calls for more pulses, and ultimately takes more time. Geyser performed much better in its original form as it was written.

These findings demonstrate the complexity behind the current gate cutting methods. Although useful in scenarios where multi-qubit gates are not executable on quantum computing platforms without swap gates, the overhead the decomposed gates produce only inhibits neutral atom compilation for now. Further research into more efficient gate cutting could give the method more potential in combination with Geyser. Since overhead stays consistent regardless of circuit size, using this method at a larger scale with more resources would likely not make a difference. Additionally, these experiments only cut one multi-qubit gate per circuit. For circuits where multiple multi-qubit gates are cut when applying the gate cutting method, it would likely increase the overhead exponentially because each decomposition of the cut gate would be multiplied by each term of the other multi-qubit gates. Thus, it is clear that this method is not suitable in the context of neutral atom quantum computers.

Geyser's optimization allows much room for improvement. Future work could explore other gate cutting methods instead of the one used in this work. Perhaps a gate cutting method with little to no overhead could become a point of interest for Geyser in the future. Additionally, the recent interest in atom shuttling could benefit the Geyser method and should be further explored.

List of Figures

List of Tables

Bibliography

[1] S. Lloyd. *Programming the Universe: A Quantum Computer Scientist Takes on the Cosmos*. New York: Knopf, 2006. ISBN: 978-1-4000-4092-6.

[2] J. M. Baker, C. Duckering, A. Litteken, and F. T. Chong. "Geyser: A Compiler for Quantum Computing with Multi-Qubit Gates for Neutral Atoms". In: *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*. ACM, 2022, pp. 383–395. DOI: 10.1145/3470496.3527428. URL: https://dl.acm.org/doi/10.1145/3470496.3527428.

[3] C. Ufrecht, M. Periyasamy, S. Rietsch, D. D. Scherer, A. Plinge, and C. Mutschler. "Cutting multi-control quantum gates with ZX calculus". In: *Quantum* 7 (2023), p. 1147. DOI: 10.22331/q-2023-10-23-1147. URL: https://doi.org/10.22331/q-2023-10-23-1147.

[4] P. Kaye, R. Laflamme, and M. Mosca. *An introduction to quantum computing*. en. Oxford scholarship online. Oxford: Oxford University Press, 2020. ISBN: 978-0-19-857000-4 978-0-19-191672-4. DOI: 10.1093/oso/9780198570004.001.0001.

[5] N. D. Mermin. *Quantum Computer Science: An Introduction*. Cambridge, UK: Cambridge University Press, 2007. ISBN: 9780511341526.

[6] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge, UK: Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[7] T. Patel, D. Silver, and D. Tiwari. "A Compilation Framework for Quantum Computing with Neutral Atoms". In: *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*. New York, NY, USA: ACM, 2022, pp. 383–395. DOI: 10.1145/3470496.3527428. URL: https://dl.acm.org/doi/10.1145/3470496.3527428.

[8] IBM Quantum. *SGate - Qiskit Documentation*. Accessed: Feb. 16, 2025. 2025. URL: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.SGate.

[9] D. P. DiVincenzo. "The Physical Implementation of Quantum Computation". In: *Fortschritte der Physik* 48.9-11 (2000), pp. 771–783. DOI: 10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E. URL: https://onlinelibrary.wiley.com/doi/10.1002/1521-3978(200009)48:9/11%3C771::AID-PROP771%3E3.0.CO;2-E.

[10] K. Wintersperger, F. Dommert, T. Ehmer, A. Hoursanov, J. Klepsch, W. Mauerer, G. Reuber, T. Strohm, M. Yin, and S. Luber. "Neutral Atom Quantum Computing Hardware: Performance and End-User Perspective". In: *EPJ Quantum Technology* 10.1 (2023), p. 32. DOI: 10.1140/epjqt/s40507-023-00190-1. URL: https://link.springer.com/content/pdf/10.1140/epjqt/s40507-023-00190-1.pdf.

[11] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak. "Quantum Computing with Neutral Atoms". In: *Quantum* 4 (Sept. 2020), p. 327. DOI: 10.22331/q-2020-09-21-327. URL: https://quantum-journal.org/papers/q-2020-09-21-327/.

[12] S. G. Stanchev and N. V. Vitanov. "Characterization of high-fidelity Raman qubit gates". In: *arXiv preprint* (2023). arXiv: 2310.04228 [quant-ph]. URL: https://arxiv.org/abs/2310.04228.

[13] L. Schmid, D. F. Locher, M. Rispler, S. Blatt, J. Zeiher, M. Müller, and R. Wille. "Computational Capabilities and Compiler Development for Neutral Atom Quantum Processors: Connecting Tool Developers and Hardware Experts". In: *Quantum Science and Technology* 9.3 (2024), p. 033001. DOI: 10.1088/2058-9565/ad33ac. arXiv: 2309.08656 [quant-ph]. URL: https://arxiv.org/abs/2309.08656.

[14] J. M. Baker, A. Litteken, C. Duckering, H. Hoffman, H. Bernien, and F. T. Chong. "Exploiting Long-Distance Interactions and Tolerating Atom Loss in Neutral Atom Quantum Architectures". In: *arXiv preprint arXiv:2111.06469* (2021). URL: https://arxiv.org/abs/2111.06469.

[15] L. Isenhower, M. Saffman, and K. Mølmer. "Multibit C_kNOT quantum gates via Rydberg blockade". In: *Quantum Information Processing* 10.6 (2011), pp. 755–769. DOI: 10.1007/s11128-011-0292-4. arXiv: 1104.3916 [quant-ph]. URL: https://arxiv.org/abs/1104.3916.

[16] N. Nottingham, M. A. Perlin, D. Shah, R. White, H. Bernien, F. T. Chong, and J. M. Baker. "Circuit decompositions and scheduling for neutral atom devices with limited local addressability". In: *arXiv preprint arXiv:2307.14996* (2023). URL: https://arxiv.org/abs/2307.14996.

[17] T. S. Community. *SciPy Documentation: dual$_a$nnealing*. Accessed: March 12, 2025. 2024. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.dual_annealing.html.

[18] D. Henderson, S. H. Jacobson, and A. W. Johnson. "The Theory and Practice of Simulated Annealing". In: *Handbook of Metaheuristics*. Ed. by F. Glover and G. A. Kochenberger. Vol. 57. International Series in Operations Research & Management Science. Springer, 2003, pp. 287–319. DOI: 10.1007/0-306-48056-5_10. URL: https://doi.org/10.1007/0-306-48056-5_10.

[19] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization". In: *ACM Transactions on Mathematical Software* 23.4 (1997), pp. 550–560. DOI: 10.1145/279232.279236. URL: https://dl.acm.org/doi/10.1145/279232.279236.

[20] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario. "The future of quantum computing with superconducting qubits". In: *Journal of Applied Physics* 132.16 (2022), p. 160902. DOI: 10.1063/5.0107939. URL: https://pubs.aip.org/aip/jap/article/132/16/160902/2837574/The-future-of-quantum-computing-with.

[21] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *arXiv preprint arXiv:1909.07534* (2019).

[22] H. Chefer, S. Gur, and L. Wolf. "Transformer interpretability beyond attention visualization". In: *arXiv preprint arXiv:2205.00016* (2022).

[23] C. Piveteau and D. Sutter. "Circuit knitting with classical communication". In: *IEEE Transactions on Information Theory* (2023). arXiv:2205.00016. DOI: 10.1109/TIT.2023.3310797.

[24] J. van de Wetering. "ZX-calculus for the working quantum computer scientist". In: *arXiv preprint arXiv:2012.13966* (2020). arXiv:2012.13966. URL: https://arxiv.org/abs/2012.13966.