Optimizing Gate Decomposition for High-Level Quantum Programming

Evandro C. R. Rosa, Eduardo I. Duzzioni, and Rafael de Santiago

Universidade Federal de Santa Catarina, Grupo de Computação Quântica, 88040-900 Florianópolis, SC, Brazil

This paper presents novel methods for optimizing multi-controlled quantum gates, which naturally arise in high-level quantum programming. Our primary approach involves rewriting $U(2)$ gates as $SU(2)$ gates, utilizing one auxiliary qubit for phase correction. This reduces the number of CNOT gates required to decompose any multi-controlled quantum gate from $O(n^2)$ to at most $32n$. Additionally, we can reduce the number of CNOTs for multi-controlled Pauli gates from $16n$ to $12n$ and propose an optimization to reduce the number of controlled gates in high-level quantum programming. We have implemented these optimizations in the Ket quantum programming platform and demonstrated significant reductions in the number of gates. For instance, for a Grover's algorithm layer with 114 qubits, we achieved a reduction in the number of CNOTs from 101,252 to 2,684. This reduction in the number of gates significantly impacts the execution time of quantum algorithms, thereby enhancing the feasibility of executing them on NISQ computers.

1 Introduction

Quantum computing is an emerging computational paradigm that utilizes the principles of quantum mechanics to process information. Quantum computers can store and manipulate data using superposition and entanglement, enabling them to potentially solve certain problems more efficiently than any classical supercomputer. This novel approach to computation has impacts across various industries, including finance [20], logistics [35], chemistry [8], cryptography [17], and artificial intelligence [5].

Despite achieving the milestone of quantum advantage [37], where problems that would take days to solve on classical supercomputers can be solved in seconds [2, 30, 47, 50, 51], quantum computing technology is not yet ready to address industrial-scale problems in production environments. We are currently in the NISQ era [38], characterized by Noisy Intermediate-Scale Quantum computers containing dozens to a few hundred quantum bits, which are limited by the error rates of current systems.

Several companies and startups are currently developing quantum computers using various technologies, including superconductors [28], trapped ions [16], neutral atoms [19], quantum dots [48], and diamond NV centers [10]. While each technology possesses its unique capabilities, they all share the same programming paradigm: utilizing quantum bits, or qubits, to store information and quantum gates to perform computations. Although quantum computers offer a universal set of quantum gates capable of executing

Evandro C. R. Rosa: evandro.crr@posgrad.ufsc.br

any computation, there are two constraints for high-level quantum programming. Firstly, these gates are limited to operating on one or two qubits. Secondly, there exists only a finite set of quantum gates available.

Regarding the latter constraint, we can always map a one- or two-qubit gate into a combination of others, meaning that despite the limited options of quantum gates, there exists a function, which can be calculated beforehand, to perform this transformation. However, for the former case, decomposing an arbitrary n-qubit gate into a sequence of one- and two-qubit gates takes exponential time relative to the number of qubits [23, 24]. Fortunately, there exists a set of multi-qubit quantum gates that are useful for high-level quantum programming, along with known efficient decomposition algorithms.

A finite set of single-qubit quantum gates, coupled with the ability to add control qubits to these gates' applications, is sufficient to enable high-level quantum programming instructions [11], including support for quantum subroutines. These single-qubit gates must enable the preparation of a qubit into any arbitrary quantum state, global phase considered, as we will explore later in this article.

A comprehensive set of single-qubit quantum gates that encompass most quantum algorithms includes the Pauli gates, the Hadamard gate, the phase shift gate, and the parametrized rotation gates. For any single-qubit gate with n qubits of control, there exists an efficient decomposition algorithm with a time complexity of $O(n^2)$ [13], but specifically for multi-controlled rotation gates, there exists an $O(n)$ decomposition algorithm [45]. The rotation gates belong to the set of $SU(2)$ gates, which is a subset of $U(2)$ containing all single-qubit gates.

In this paper, we propose a series of optimizations for multi-control single-target quantum gate decomposition, implementing and evaluating the results in the Ket quantum programming platform [11]. The contributions of the paper are:

C.1 A schema to describe any multi-controlled $U(2)$ gate as multi-controlled $SU(2)$ gates, leveraging the $O(n)$ decomposition with the trade-off of adding a single auxiliary qubit. This allows programmers to describe their applications in terms of $U(2)$ without incurring the $O(n^2)$ decomposition cost.

C.2 Reducing the number of CNOTs for n-controlled $SU(2)$ gates from at most $20n$ to $16n$, a result previously limited to $SU(2)$ gates with at least one real diagonal [45].

C.3 Since that the auxiliary qubit in our proposed optimization for multi-controlled $U(2)$ gates is in the state $|0\rangle$, we can reduce the number of CNOTs to decompose Pauli gates from at most $16n$ [23] to $12n$.

C.4 Presenting methods to leverage reversible quantum computation to reduce the number of controlled gates in Ket's `with around` high-level quantum programming statement.

C.5 Implementing all the optimizations in the Ket quantum programming platform[1], allowing quantum programs to take advantage of these techniques without any changes to their quantum code.

The decomposition of multi-qubit gates represents a crucial first step in preparing a quantum application for execution on a quantum computer [31, 33, 41]. Consequently, optimizing this process significantly impacts the overall quantum compilation workflow. After decomposing quantum gates into one- and two-qubit gates, the next stage involves

[1] `https://quantumket.org`

adjusting the two-qubit gates to align with the connectivity constraints of the target quantum computer [21, 29, 46], a procedure referred to as quantum circuit mapping. Thus, the optimization introduced in this article, which reduces the number of quantum gates, directly influences the efficiency of quantum circuit mapping. This mapping problem is known to be NP-Hard [6, 40].

For the remainder of this paper, we will structure our discussion as follows: In Section 2, we will discuss how multi-controlled $U(2)$ gates enable the construction of quantum subroutines in high-level quantum programming, while also exploring how high-level quantum programming can be used to reduce the number of controlled operations. In Section 3, we will present the decomposition algorithms for multi-controlled $SU(2)$ and Pauli gates. Following that, in Section 4, we will demonstrate how to rewrite any $U(2)$ gate in terms of $SU(2)$ gates, allowing us to leverage the $O(n)$ decomposition for any multi-controlled single-target quantum gate. Subsequently, in Section 5, we will present numerical results showcasing the impact of this rewrite and other optimizations on the classical and quantum runtime. Finally, we conclude in Section 6, summarizing our findings and discussing future work.

2 High-Level Quantum Programming

At the lowest level, a quantum program consists of a sequence of radio frequency pulses [42], analogous to a binary program for a classical computer. Consequently, programming at this level is complex and heavily dependent on the target hardware. Therefore, quantum computers typically provide a set of one and two-qubit gates, where these gates have been pre-calibrated and have a known pulse format [22, 26]. This set of gates forms the equivalent of an assembly language for a quantum computer, where programs are still hardware-dependent, but abstraction allows for the development and testing of simple quantum programs.

Similar to a classical Instruction Set Architecture (ISA) [36] with a limited number of registers, quantum computers have a finite number of qubits, and the connectivity of these qubits limits which qubits can operate together in two-qubit gates. Therefore, the level of abstraction provided by a quantum assembly language is not suitable for the development of complex quantum applications. This necessitates the use of high-level quantum programming platforms such as Qiskit [25], Cirq [14], ProjectQ [43], PennyLane [4], Q#[44], and Ket [11]. At this level, we transition away from pulse definitions and hardware constraints towards the mathematical formalism of quantum mechanics. Using a quantum compiler, we can take a high-level quantum application and execute it on a quantum computer.

Table 1 presents a gate set commonly used in many quantum algorithms. Additionally, controlled versions of these gates are also utilized. For example, the CNOT gate, short for Controlled-NOT, is a Pauli X gate (also known as the NOT gate) with a control qubit. The matrix representation of the CNOT gate is as follows:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1}$$

A Controlled-U gate, where U is a unitary 2×2 matrix, is represented by the following matrix block form:

$$\text{Controlled-}U = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & U \end{bmatrix}, \tag{2}$$

Table 1: Commonly used single-qubit quantum gates. These gates form the foundational set implemented in Ket's runtime library, Libket. All other gates in Ket are composed using controlled versions of these base gates.

Gate Name	Matrix Representation	Gate Name	Matrix Representation
Pauli X	$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Phase	$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$
Pauli Y	$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	X Rotation	$R_X(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$
Pauli Z	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	Y Rotation	$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$
Hadamard	$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Z Rotation	$R_Z(\theta) = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$

where I represents an 2×2 identity matrix, and $\mathbf{0}$ signifies a 2×2 null matrix.

Controlled-gates operate with both a target qubit and a control qubit. The operation is applied to the target qubit only if the control qubit is in the state $|1\rangle$[2]. This notation can be extended for n-control qubits as

$$C^nU = \sum_{k=0}^{2^n-2} |k\rangle\langle k| \otimes I + |2^n{-}1\rangle\langle 2^n{-}1| \otimes U = \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & U_{0,0} & U_{0,1} \\ 0 & 0 & \cdots & 0 & U_{1,0} & U_{1,1} \end{bmatrix}}^{2^{n+1}\times 2^{n+1}}, \tag{3}$$

where k is an integer, with $|k\rangle$ representing an n-qubit state. The gate U is applied to the target qubit only if all n control qubits are in the $|1\rangle$ state.

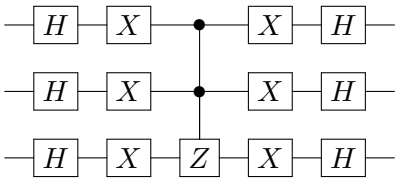
In quantum programming, integrating control qubits into quantum operations facilitates the straightforward expression of numerous quantum computations. It enables the description of various quantum subroutines, such as Grover's diffusion, a pivotal step in Grover's quantum search algorithm [18], and the Quantum Fourier Transformation, utilized in Shor's quantum factorization algorithm [39].

Figure 1a illustrates the implementation of Grover's diffusion operator in the Ket quantum programming platform. This example highlights the use of high-level constructions. The `with around` instruction takes advantage of the quantum circuit's reversibility, reducing the number of instructions needed to describe a quantum circuit that starts with a gate and ends with the inverse of that same gate. In Figure 1b, we present the circuit for a 3-qubit Grover's diffusion. Notably, each qubit line begins with a Hadamard followed by a Pauli X gate (XH) and ends with its inverse ($(XH)^\dagger = HX$). The use of the `with around` statement minimizes the number of high-level quantum programming instructions needed for this implementation. Also, in the middle of the Grover's diffusion circuit, we observe a multi-controlled Pauli Z gate. In Ket, this controlled-gate can be implemented by placing a Pauli Z gate within a `with control` scope, which effectively adds control qubits to any quantum gate called inside.

[2]While the control state of a qubit is typically $|1\rangle$, it can be any basis state. For simplicity, this paper assumes all control qubits are in the state $|1\rangle$.

```python
def grover_diffusion(qubits):
    with around(cat(H, X), qubits):
        with control(qubits[:-1]):
            Z(qubits[-1])
```

(a) Grover's diffusion operator implemented using Ket.



(b) Grover's diffusion operator for 3 qubits.

Figure 1: Grover's diffusion operator.

Figure 2 illustrates a state preparation algorithm implemented using the Ket quantum programming platform. The `prepare` function serves as a quantum gate, preparing a set of qubits into a quantum state encoded in two lists: one representing the measurement probabilities and the other encoding the phase of the state's amplitude probability. In this code snippet, lines 17 and 19 recursively invoke the `prepare` function, stacking control qubits with each call as they are nested within the scope of the `with control` in lines 16 and 18.

It is worth noting that although only single quantum gate applications are explicitly called in lines 9, 11-13, and 15, the recursive controlled call results in multi-controlled R_Y and multi-controlled $P(\theta)$ gates, as depicted in Figure 3. Additionally, it's noteworthy that due to optimizations implemented in the `with around` statement, Ket does not add controls for the Pauli X gates. This optimization, which will be further elucidated in Subsection 2.1, is also a contribution of this work.

```python
def prepare(
    qubits: Quant,
    prob: ParamTree | list[float | int],
    amp: list[float] | None = None,
):
    if not isinstance(prob, ParamTree):
        prob = ParamTree(prob, amp)
    head, *tail = qubits
    RY(prob.value, head)
    if prob.is_leaf():
        with around(X, head):
            PHASE(prob.phase0, head)
        PHASE(prob.phase1, head)
        return
    with around(X, head):
        with control(head):
            prepare(tail, prob.left)
    with control(head):
        prepare(tail, prob.right)
```

Figure 2: Recursive function for quantum state preparation using Ket. This implementation of the `ParamTree` class (detailed in Figure 10a) features a tree graph structure. The algorithm traverses this tree graph, using node information to adjust measurement probabilities with R_Y gates, and leaf node values to adjust the complex phase of the probability amplitude with Phase gates.

The construction illustrated in Figure 2, which allows calling a controlled function, is present in many high-level quantum programming platforms [11, 44]. It serves as an example of how controlled quantum operations can appear indirectly. As mentioned before,

quantum computers do not implement these operations directly. Instead, algorithms exist to decompose these operations into one and two quantum gates. However, some gate decompositions are more efficient than others. For instance, an arbitrary C^nU gate, such as the multi-controlled $P(\theta)$, decomposition results in a linear-depth circuit but has a decomposition time complexity of $O(n^2)$ [13]. On the other hand, if $U \in SU(2)$, such as the multi-controlled R_Y, the time complexity of the decomposition is linear [45]. In this paper, we present an optimization that allows using the more efficient $SU(2)$ decomposition for any C^nU gates. In the next sections, we illustrate how to perform the decomposition, and in section 4, we present this decomposition optimization.

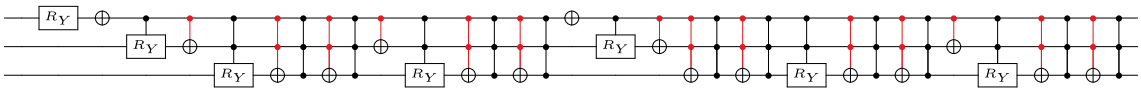


Figure 3: Quantum circuit resulting from executing the code from Figure 2 with three qubits. The red lines represent the control qubits that were optimized out in the `with around` statements, where multi-controlled Pauli X gates were replaced by Pauli X gates.

2.1 Reducing the Number of Controlled Gates

Another optimization proposed in this article is reducing the number of controlled operations in the `with around` construction (**C.4**). This statement, implemented in Ket, takes two unitary operations as arguments: the first is the `around` function arguments, and the second is the new code scope created in the next line, respectively, corresponding to the unitaries A and B. For example, in lines 11 and 12 of the code in Figure 2, the unitary A is the application of the Pauli X gate, and the unitary B is the application of the Phase gate.

The resulting unitary from a `with around` statement is $A^\dagger BA$. Prior to this optimization proposal, calling a `with around` statement with control qubits resulted in the unitary $(C^nA^\dagger) \cdot (C^nB) \cdot (C^nA)$, effectively adding control qubits to all operations. However, we can optimize this construction by adding control qubits only to the B operation, resulting in $A^\dagger \cdot (C^nB) \cdot A$. The intuition behind this optimization is that if the control qubits are in the control state (all qubits in $|1\rangle$), then the entire unitary $A^\dagger BA$ will be applied to the target qubits. Conversely, if the control qubits are not in the control state, the B unitary will not be applied, resulting in the application of $A^\dagger A = I$, which is equal to the identity operation, a no-operation.

With this optimization, we can replace all the controlled-NOT gates in the code from Figure 2, which would otherwise result from the recursive controlled calls of lines 11 and 15, with simple NOT gates. As shown in Figure 3, with only 3 qubits, this represents a reduction from 30 to 16 controlled gates. This is one example of how high-level quantum programming can result not only in fewer lines of code but also in more efficient code.

3 Quantum Gate Decomposition

Quantum gate decomposition is a crucial step in translating high-level quantum programming instructions into operations that a quantum computer can execute. In this paper, we focus on the decomposition of multi-control single-target quantum gates, as described in Equation (3). Specifically, we examine the gates listed in Table 1, although our results can be applied to any single-qubit unitary operation. To the best of our knowledge, the state-of-the-art in quantum gate decomposition is represented by the works of Iten et al.

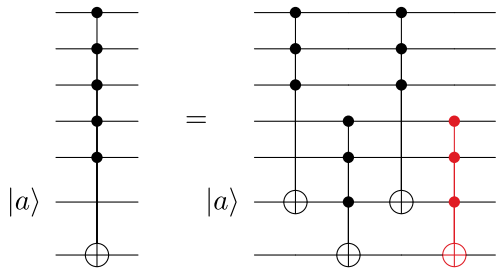


Figure 4: Multi-controlled Pauli X gate decomposition using a dirty auxiliary qubit. If the auxiliary qubit $|a\rangle$ is in the state $|0\rangle$, the last controlled operation in red can be discarded.

[23], Vale et al. [45], and da Silva and Park [13]. These works propose decompositions that result in linear-depth quantum circuits, with the first two limited to Pauli gates and $SU(2)$ gates, respectively, and the last one applicable to any $U(2)$ gate. For gates that are 2×2 special unitary operations ($SU(2)$), the decomposition can be performed with only a linear number of CNOTs, while for a general $U(2)$ acting with controlled qubit, the number of CNOTs is $O(n^2)$ [13]. From Table 1, it can be observed that the rotation gates belong to $SU(2)$. For other quantum gates, with the exception of Pauli gates, the decomposition results in a quadratic number of CNOTs gates.

For the Pauli X gate, Barenco et al. [3] proposed a circuit to decompose a k-controlled Pauli X gate using $k-2$ dirty[3] auxiliary qubits. This decomposition utilizes a chain of Toffoli gates (C^2X). Improving on this, Iten et al. [23] proposed using an approximate Toffoli gate that requires only 3 CNOTs instead of 6, as in the exact case. This results in a decomposition that requires at most $4n$ CNOTs, where n is the number of control qubits. Leveraging this optimization, Iten et al. [23] further proposed a decomposition for the n-controlled Pauli X gate that requires at most $16n$ CNOTs and a single dirty auxiliary qubit. This method breaks the n-controlled Pauli X into four $\frac{n}{2}$-controlled Pauli X decompositions, as illustrated in Figure 4, allowing the qubits that are not used in the multi-controlled Pauli X gate to be used as auxiliaries.

For the multi-controlled Pauli X gate decomposition, the auxiliary qubit remains in the same state after the gate application. Therefore, if we ensure the auxiliary qubit is in the state $|0\rangle$ before applying the quantum gates, we can reduce the decomposition to at most $12n$ CNOTs (**C.3**). This aligns with the optimization proposed in Section 4.1, which requires an auxiliary qubit in the state $|0\rangle$ and maintains the auxiliary qubit state, allowing the same auxiliary qubit to be reused for all decompositions. The intuition behind this optimization is that the first controlled gate flips the auxiliary qubit from $|0\rangle$ to $|1\rangle$ only if the first half of the control qubits are in the state $|1\rangle$. Consequently, the target qubit flips only if the second half of the control qubits and the auxiliary qubit are in the state $|1\rangle$. The third controlled operation flips the auxiliary qubit back to $|0\rangle$, preventing the execution of the last controlled operation.

To decompose the multi-controlled Pauli Y and Z gates, given that we have an auxiliary qubit, we can use the multi-controlled Pauli X gate decomposition, adding unitary operations around the target qubit. For instance, to implement a multi-controlled Z gate, we can apply a Hadamard gate around the target, since $HXH = Z$. Similarly, for a multi-controlled Y gate, we can use a Phase gate with $HP(\frac{\pi}{2})XP(\frac{-\pi}{2})H = Y$. Therefore, any n-controlled Pauli gate can be decomposed with at most $12n$ CNOTs.

The decomposition of multi-controlled $SU(2)$ gates builds on top of the presented multi-

[3]These qubits can be in any state.

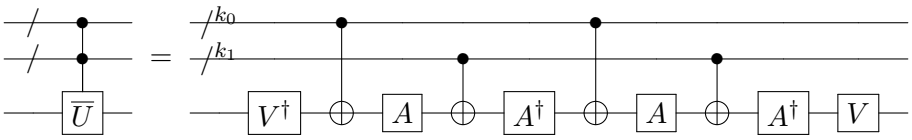


Figure 5: Multi-controlled $SU(2)$ gate decomposition with n-control qubits, where $k_0 = \lfloor \frac{n}{2} \rfloor$, $k_1 = n - k_0$, $\overline{U} = VDV^\dagger$, and $D = A^\dagger XAXA^\dagger XAX$.

controlled Pauli X decomposition. For a gate $\overline{U} \in SU(2)$ with at least one real diagonal element, Vale et al. [45] proposes a decomposition that takes at most $16n$ CNOTs, and up to $20n$ CNOTs when generalized for any $SU(2)$ gate. This generalization uses the eigendecomposition $\overline{U} = VDV^\dagger$, where the final controlled gate is $C^n\overline{U} = (C^nV) \cdot (C^nD) \cdot (C^nV^\dagger)$. The decomposition of (C^nD) can follow the same schema as the decomposition of a multi-controlled $SU(2)$ gate with a real diagonal, but due to CNOT cancellations that can occur in $(C^nD) \cdot (C^nV^\dagger)$, the entire decomposition takes at most $20n$ CNOTs. However, leveraging the proposed optimization **C.4**, there is no need to apply the V unitary with control qubits, since if the controlled-D is not applied, it results in the identity operation because $VV^\dagger = I$. Therefore, there is no need to decompose the $(C^nV^\dagger)$ and (C^nV) gates, and the final decomposition takes at most $16n$ CNOTs (**C.2**). Figure 5 presents the schema for decomposing a multi-controlled $SU(2)$ gate. Note that the eigendecomposition is not necessary when the gate has a real diagonal, although it streamlines the calculation of the A gate since it results in an R_Z gate, which can be performed virtually in superconducting qubits [32].

The decomposition proposed by da Silva and Park [13] works for any multi-controlled single-target quantum gate, and the resulting circuit has linear depth, meaning that, by executing the gates in parallel when possible, it takes linear time in the quantum computer. Additionally, this decomposition has the benefit of not requiring any auxiliary qubits. However, it adds a quadratic number of CNOTs, resulting in a time complexity of $O(n^2)$ to execute this algorithm on a classical computer. Furthermore, this schema requires computing fractions from π to $\frac{\pi}{2^{n-1}}$, which accumulates floating-point errors. In the following section, we will cover how to rewrite a multi-controlled $U(2)$ gate as a sequence of multi-controlled $SU(2)$ gates, allowing us to use the $O(n)$ decomposition by adding a single auxiliary qubit.

4 Quantum Gate Rewrite

In this section, we explore how to rewrite any $U(2)$ gate in terms of $SU(2)$ gates to leverage the efficient multi-controlled $SU(2)$ gate decomposition for any multi-controlled $U(2)$ gate. For any 2×2 unitary matrix U, there exists a corresponding 2×2 special unitary matrix $\overline{U}$, such that[4]

$$U = e^{i\phi}\overline{U}. \tag{4}$$

To determine the phase angle ϕ, we recall that since U is unitary, its determinant satisfies $\det(U) = e^{i\Phi}$. Moreover, since $\overline{U}$ is a special unitary matrix, we have $\det(\overline{U}) = 1$.

[4]We conclude that Equation (4) is valid based on Nielsen and Chuang [34, Theorem 4.1], since the product of $SU(2)$ gates, in the case of rotation gates, is also an element of $SU(2)$.

Substituting these properties into the determinant equation, we obtain

$$\begin{aligned} \det(U) &= \det(e^{i\phi}\overline{U}) \\ e^{i\Phi} &= e^{i2\phi}\det(\overline{U}) = e^{i2\phi}. \end{aligned} \tag{5}$$

Thus, it follows that $2\phi = \Phi$. The angle Φ can be determined as the phase of the complex number $e^{i\Phi}$, given by

$$\Phi = \arctan2\big(\mathrm{Im}(e^{i\Phi}),\ \mathrm{Re}(e^{i\Phi})\big). \tag{6}$$

Finally, solving for ϕ, we obtain

$$\phi = \frac{1}{2}\Phi = \frac{1}{2}\arctan2\big(\mathrm{Im}(\det U),\ \mathrm{Re}(\det U)\big). \tag{7}$$

Therefore, we can rewrite the $U(2)$ gates from Table 1 using the $SU(2)$ rotation gates. The results of these conversions are presented in Table 2. While the global phase difference between U and $\overline{U}$ has no measurement effect and can be ignored, for controlled operations, this global phase results in a relative phase difference. For instance, while $U = e^{i\phi}\overline{U} \equiv \overline{U}$,

$$C^1U = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & U \end{bmatrix} \not\equiv e^{i\phi}C^1\overline{U} = \begin{bmatrix} e^{i\phi}I & \mathbf{0} \\ \mathbf{0} & U \end{bmatrix} \text{ and} \tag{8}$$

$$C^1U = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & U \end{bmatrix} \not\equiv C^1\overline{U} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & \overline{U} \end{bmatrix}. \tag{9}$$

Table 2: $U(2)$ and $SU(2)$ quantum gate equivalence.

$U(2)$ **Gate**	$SU(2)$ **Equivalent**	**Global Phase**
Pauli X	$X \equiv R_X(\pi)$	$e^{i\frac{\pi}{2}}$
Pauli Y	$Y \equiv R_Y(\pi)$	$e^{i\frac{\pi}{2}}$
Pauli Z	$Z \equiv R_Z(\pi)$	$e^{i\frac{\pi}{2}}$
Phase	$P(\theta) \equiv R_Z(\theta)$	$e^{i\frac{\theta}{2}}$
Hadamard	$H \equiv R_X(\pi)R_Y(\frac{\pi}{2})$	$e^{i\frac{\pi}{2}}$

As presented, the global phase difference between a $U(2)$ and $SU(2)$ gate matters when using the gate in a controlled operation. Therefore, the simple rewrite of $U(2)$ into an $SU(2)$ is not sufficient to leverage the $SU(2)$ more efficient decomposition. A contribution of this paper is how to fix this phase difference and use the linear time decomposition algorithms for any quantum gate. We develop our solution in the next subsection. First, in Subsection 4.1, we provide a simple example of how we can fix the phase of a single-controlled gate and generalize the result for multi-controlled gates. Although the phase fixing results in the correct quantum gate, this workaround negates the proposed rewrite optimization. Therefore, in Subsection 4.2, we introduce an auxiliary qubit in the decomposition to enable the linear decomposition of any multi-controlled $U(2)$ gate. This optimization is especially beneficial for decomposition multi-controlled Phase and Hadamard gates.

4.1 Fixing the Phase

For a single-controlled quantum gate, considering that $U = e^{i\phi}\overline{U}$, we can see that a $P(\phi)$ gate applied to the control qubits can fix the phase.

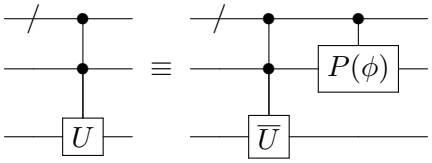


Figure 6: Multi-controlled $U(2)$ gate rewrite as a multi-controlled $SU(2)$ gate using a $C^{n-1}P(\phi)$ gate for phase correction.

$$\begin{aligned} C^1U &= (P(\phi)\otimes I)\cdot C^1\overline{U} \\ &= (P(\phi)\otimes I)\cdot\left(|0\rangle\langle 0|\otimes I + |1\rangle\langle 1|\otimes \overline{U}\right) \\ &= P(\phi)|0\rangle\langle 0|\otimes I + P(\phi)|1\rangle\langle 1|\otimes \overline{U} \\ &= |0\rangle\langle 0|\otimes I + e^{i\phi}|1\rangle\langle 1|\otimes \overline{U} \\ &= |0\rangle\langle 0|\otimes I + |1\rangle\langle 1|\otimes U. \end{aligned} \tag{10}$$

The intuition for this equation is that the phase must be applied only when the gate $\overline{U}$ is applied. In the controlled version, this is when the control qubit is in the state $|1\rangle$. Since the Phase gate has no effect on the $|0\rangle$ state, the phase gate applies a phase $e^{i\phi}$ when the control is in the state $|1\rangle$, corresponding to the application of the gate $\overline{U}$. Note that despite the phase being applied to the control qubits instead of the target, it has the same result as $(\alpha\cdot|\psi\rangle)\otimes|\varphi\rangle = |\psi\rangle\otimes(\alpha\cdot|\varphi\rangle)$.

Generalizing for n-control qubits, we can see that the circuit illustrated in Figure 6 using $C^{n-1}P(\phi)$ gate can fix the phase as

$$\begin{aligned} C^nU &= \left(C^{n-1}P(\phi)\otimes I\right)\cdot C^n\overline{U} \\ &= \left(\overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & e^{i\phi} \end{bmatrix}}^{2^n\times 2^n} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \cdot \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & \overline{U}_{0,0} & \overline{U}_{0,1} \\ 0 & 0 & \cdots & 0 & \overline{U}_{1,0} & \overline{U}_{1,1} \end{bmatrix}}^{2^{n+1}\times 2^{n+1}} \\ &= \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & e^{i\phi} & 0 \\ 0 & 0 & \cdots & 0 & 0 & e^{i\phi} \end{bmatrix}}_{2^{n+1}\times 2^{n+1}} \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & \overline{U}_{0,0} & \overline{U}_{0,1} \\ 0 & 0 & \cdots & 0 & \overline{U}_{1,0} & \overline{U}_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & U_{0,0} & U_{0,1} \\ 0 & 0 & \cdots & 0 & U_{1,0} & U_{1,1} \end{bmatrix}. \end{aligned} \tag{11}$$

We can follow the same intuition as in the single-control case, with the phase being applied only when all the control qubits are in the state $|1\rangle$. Note that for the multi-controlled case, the $C^{n-1}P(\phi)$ must be decomposed into one and two-qubit gates to execute on the quantum computer. However, since the Phase gate is not an $SU(2)$ gate, this decomposition takes quadratic compilation time and space, which eliminates the benefit of rewriting the U gate as $SU(2)$. In the next subsection, we introduce an auxiliary qubit and substitute the multi-control Phase gate with a multi-control Z rotation gate, therefore making all decompositions linear.

4.2 Adding an Auxiliary Qubit

With an auxiliary qubit initialized in the state $|0\rangle$, we can use a controlled R_Z gate to introduce a phase $e^{i\phi}$ after applying a $C^n\overline{U}$ gate, thereby implementing a C^nU gate. Starting with a single control, we can see that appending a $C_c^1 R_Z(-2\phi)_a$ after the controlled-$\overline{U}$ gate fixes the phase. To denote the control and target qubits of an operation, we use subscripts, with $C_c^1\overline{U}_t$ meaning the qubit c controls the application of the gate $\overline{U}$ on the qubit t, and $C_c^1 R_Z(-2\phi)_a$ meaning the qubit c controls the gate R_Z on qubit a, where c, t, and a stand for control, target, and auxiliary, respectively. Expanding the rewriting, we see that the application is equivalent to

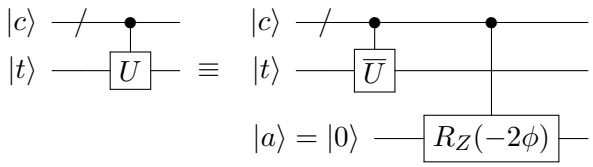


Figure 7: Multi-controlled single-target $U(2)$ gate rewrite to take advantage of $SU(2)$ decomposition.

$$
\begin{aligned}
C_c^1 U_t \equiv & C_c^1 R_Z(-2\phi)_a \cdot C_c^1 \overline{U}_t \\
= & \left(|0\rangle\langle 0|_c \otimes I_t \otimes I_a + |1\rangle\langle 1|_c \otimes I_t \otimes R_Z(-2\phi)_a\right) \\
& \cdot \left(|0\rangle\langle 0|_c \otimes I_t \otimes I_a + |1\rangle\langle 1|_c \otimes \overline{U}_t \otimes I_a\right) \\
= & |0\rangle\langle 0|_c \otimes I_t \otimes I_a + |1\rangle\langle 1|_c \otimes \overline{U}_t \otimes R_Z(-2\phi)_a \\
= & |0\rangle\langle 0|_c \otimes I_t \otimes I_a + |1\rangle\langle 1|_c \otimes \overline{U}_t \otimes \left(e^{i\phi}|0\rangle\langle 0|_a + e^{-i\phi}|1\rangle\langle 1|_a\right).
\end{aligned}
\tag{12}
$$

Since the auxiliary qubit is in the state $|0\rangle$, projecting it to $|0\rangle$ does not change the quantum state but allows us to state that

$$
\begin{aligned}
C_c^1 U_t \equiv & \left(C^1{}_c R_Z(-2\phi)_a C^1{}_c \overline{U}_t\right) \cdot (I \otimes I \otimes |0\rangle\langle 0|) \\
= & |0\rangle\langle 0|_c \otimes I_t \otimes |0\rangle\langle 0|_a + |1\rangle\langle 1|_c \otimes \overline{U}_t \otimes e^{i\phi}|0\rangle\langle 0|_a \\
= & (|0\rangle\langle 0|_c \otimes I_t + |1\rangle\langle 1|_c \otimes U_t) \otimes |0\rangle\langle 0|_a.
\end{aligned}
\tag{13}
$$

The intuition for this rewrite builds on the previous subsection. However, since the R_Z gate changes the phase of both $|0\rangle$ and $|1\rangle$ states, we need an auxiliary qubit that we know to be in the state $|0\rangle$, thereby selecting the desired phase from the R_Z gate. Note that the auxiliary qubit remains in the $|0\rangle$ state, allowing us to reuse it in other operations.

We can straightforwardly extend this concept to multiple control qubits, resulting in

the unitary matrix[5]

$$
\begin{aligned}
C_c^n U_t \equiv & C_c^n R_Z(-2\phi)_a \cdot C_c^n \overline{U}_t \\
= & \left(\sum_{l=0}^{2^n-2} |l\rangle\langle l|_c \otimes I_t \otimes I_a + |2^n-1\rangle\langle 2^n-1|_c \otimes I_t \otimes R_Z(-2\phi)_a \right) \\
& \cdot \left(\sum_{k=0}^{2^n-2} |k\rangle\langle k|_c \otimes I_t \otimes I_a + |2^n-1\rangle\langle 2^n-1|_c \otimes \overline{U}_t \otimes I_a \right) \\
= & \overbrace{\begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & e^{i\phi} & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & e^{-i\phi} & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & e^{i\phi} & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}}^{2^{n+2}\times 2^{n+2}} \cdot \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & \overline{U}_{0,0} & 0 & \overline{U}_{0,1} & 0 \\ 0 & \cdots & 0 & 0 & \overline{U}_{0,0} & 0 & \overline{U}_{0,1} \\ 0 & \cdots & 0 & \overline{U}_{1,0} & 0 & \overline{U}_{1,1} & 0 \\ 0 & \cdots & 0 & 0 & \overline{U}_{1,0} & 0 & \overline{U}_{1,1} \end{bmatrix} \\
= & \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & U_{0,0} & 0 & U_{0,1} & 0 \\ 0 & \cdots & 0 & 0 & e^{-i\phi}\overline{U}_{0,0} & 0 & e^{-i\phi}\overline{U}_{0,1} \\ 0 & \cdots & 0 & U_{1,0} & 0 & U_{1,1} & 0 \\ 0 & \cdots & 0 & 0 & e^{-i\phi}\overline{U}_{1,0} & 0 & e^{-i\phi}\overline{U}_{1,1} \end{bmatrix} \cdot
\end{aligned}
\tag{14}
$$

Since we know that the auxiliary qubit is in the state $|0\rangle$, we can project into $|0\rangle$ and trace out the auxiliary qubit, resulting in exactly the matrix for a multi-controlled U gate:

$$
\begin{aligned}
C^n{}_c U_t \equiv & \left(C^n{}_c R_Z(-2\phi)_a \cdot C^n{}_c \overline{U}_t \right) \cdot \left(I^{\otimes n}{}_c \otimes I_t \otimes |0\rangle\langle 0|_a \right) \\
= & \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & U_{0,0} & 0 & U_{0,1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & U_{1,0} & 0 & U_{1,1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xRightarrow{\mathrm{Tr}_a} \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & U_{0,0} & U_{0,1} \\ 0 & 0 & \cdots & 0 & U_{1,0} & U_{1,1} \end{bmatrix}}^{2^{n+1}\times 2^{n+1}} = C^n{}_c U_t
\end{aligned}
\tag{15}
$$

Figure 7 illustrates the proposed rewrite. Note that, since the $\overline{U}$ and R_Z gates are in $SU(2)$, the decomposition of the resulting circuit has a lower time complexity (with $16n$ CNOTs for each one) than the original circuit (with $O(n^2)$ CNOTs), requiring at most $32n$ CNOTs (**C.1**). Also, since the auxiliary qubit can be reused, the overhead of this rewriting is only a single auxiliary qubit for the entire quantum circuit. This result improves upon previous linear time decompositions. For instance, the proposal by Barenco

[5] In Equation (14), we use integer notation inside the kets for the control qubits, where $|l\rangle$ represents an n-qubit state.

et al. [3, Lemma 7.11] requires two auxiliary qubits, while the decomposition presented by Nielsen and Chuang [34, p. 184] requires exactly $n-1$ auxiliary qubits.

For a multi-controlled Phase gate, considering that $P(\theta) = e^{i\frac{\theta}{2}} R_Z(\theta)$, we can rewrite this operation with a single multi-controlled R_Z gate and one auxiliary qubit, as illustrated in Figure 8. By using the target qubit of the original instruction also as a control in the multi-controlled R_Z, the auxiliary qubit ends up in the state $R_Z(-2\theta)\,|0\rangle = e^{i\theta}\,|0\rangle$. Since the $P(\theta)$ gate only applies the relative phase $e^{i\theta}$ to the $|1\rangle$ state, particularly the control state, the target qubit acquires the phase $e^{i\theta}$ only if all the control qubits and the target qubit are in the state $|1\rangle$. With this scheme, the decomposition of a multi-controlled Phase gate takes at most $16n$ CNOTs.

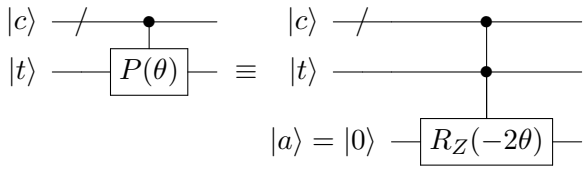


Figure 8: Multi-controlled $P(\theta)$ gate rewrite to take advantage of $SU(2)$ decomposition.

In the next section, we discuss the impact of this and other optimizations proposed in this paper on the decomposed quantum circuit. Additionally, we evaluate the implementation of these optimizations in the Ket quantum programming platform.

5 Results

Summarizing the optimizations proposed in this article, Table 3 shows that given an auxiliary qubit in the state $|0\rangle$, we can decompose any n-controlled quantum gate from Table 1 using a linear number of quantum gates CNOTs. The gates that benefit most from this optimization are the multi-controlled Phase and Hadamard gates, whose time complexity has been reduced from quadratic to linear given one auxiliary qubits. This reduction not only impacts the quantum runtime but also the classical runtime, as the classical computer computes the decompositions. For the rotation gates, since these $SU(2)$ gates fall under the special case presented in Vale et al. [45], having at least one real diagonal, the time complexity for decomposing their multi-controlled version remains the same.

In this section, we will explore how these optimizations impact the CNOT count of two algorithms: Grover's algorithm, which requires multi-controlled Pauli Z gates, and a state preparation algorithm that requires multi-controlled Phase gates. For our evaluation, we will compare the final number of CNOTs with and without the auxiliary qubit. All evaluations were performed using the Ket quantum programming platform, where we implemented all the decompositions and optimizations presented in this paper (**C.5**).

Note that enabling these optimizations requires no changes to the code. In Ket's architecture, the quantum execution target determines which features will be available for execution, including whether decomposition is necessary. By default, the `Process` class uses simulation, which does not require decomposition [12].

We have prepared a repository[6] containing the Ket version and the data generated in our experiments. Note that the Ket version used in this study does not match any official platform release. In this version, a `decompose` parameter is provided in the `Process`

[6] https://gitlab.com/quantum-ket/arxiv-2406.05581

Table 3: Comparison of the number of CNOTs required for n-controlled single-qubit gate decomposition. The subscript in the number of auxiliary qubits indicates whether the qubit can be dirty (d) or must be in the $|0\rangle$ state (0). Our results are marked in bold.

Gate Name	**Number of CNOTs/Auxiliary Qubits**					
		Comparison with State-of-the-Art				
	Our	[45]	[23]	[13]	[3]	[34]
Pauli X	$\mathbf{12n/1_0}$		$16n/1_d$; $4n/n-2_d$	$O(n^2)/0$		
Pauli Y	$\mathbf{12n/1_0}$		$16n/1_d$; $4n/n-2_d$	$O(n^2)/0$		
Pauli Z	$\mathbf{12n/1_0}$		$16n/1_d$; $4n/n-2_d$	$O(n^2)/0$		
Hadamard	$\mathbf{32n/1_0}$			$O(n^2)/0$	$32n/2_{0,d}$	$2n/n_0$
Phase	$\mathbf{16n/1_0}$			$O(n^2)/0$	$32n/2_{0,d}$	$2n/n_0$
X Rotation		$16n/0$				
Y Rotation		$16n/0$				
Z Rotation		$16n/0$				

class constructor for testing purposes, enabling the optimization by default when activated. Additionally, when decomposition is enabled and the optimization is not explicitly disabled[7], the simulation will execute with one additional qubit beyond what is specified in the constructor, allowing it to be used as an auxiliary qubit.

5.1 Evaluation: Grover's Algorithm

Grover's algorithm [18] can find an element in an unstructured list in time $O(\sqrt{N})$, where N is the number of elements in the list, using $n = \log_2(N)$ qubits. This algorithm has applications in various fields such as solving NP problems [9], optimization [15], and cryptography [1, 7]. It works by marking the search state in superposition and increasing the probability of measuring it, doing so in $\sqrt{N}$ iterations.

Figure 9a illustrates Grover's algorithm implemented using Ket. Each Grover layer contains two multi-controlled Pauli Z gates: one for the oracle and another for the diffusion operation. We used an oracle that marks the state $|11\ldots1\rangle$; however, marking any other basis state only requires applying Pauli X gates around to certain qubits. To evaluate the decomposition, we executed a single layer of the algorithm, resulting in a decomposition with at most $24n$ CNOTs when optimization is enabled. Even with just one layer, there is a substantial improvement in the number of CNOTs required for the decomposition.

Figure 9b illustrates the number of CNOT gates resulting from the decomposition when the algorithm is executed with 4 to 114 input qubits. Notably, enabling the optimization requires one additional auxiliary qubit. For instance, with 114 qubits, the optimization reduced the number of CNOTs from 101,252 to 2,684. Note that without the auxiliary qubit, only the rotation gates that belong to $SU(2)$ are decomposed with a linear decomposition, while the other gates are decomposed with the quadratic multi-controlled $U(2)$ decomposition.

5.2 Evaluation: State Preparation Algorithm

Our next evaluation analyzes the impact of the proposed decomposition optimization in the State Preparation Algorithm [27] shown in Figure 2. This algorithm prepares n-qubits in any possible quantum state, using two lists with 2^n parameters: one for the measurement

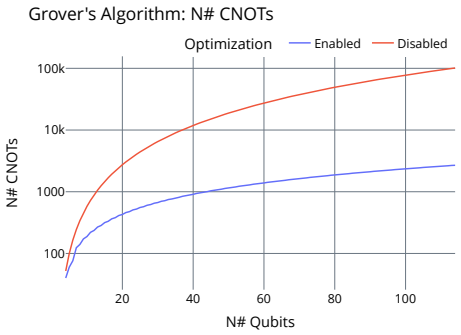
[7]To disable the optimization, set the environment variable `KET_DISABLE_DECOMPOSITION_OPTIMIZATION=true`.

```python
def oracle(qubits):
    """|11...1> -> -|11...1>"""
    ctrl(qubits[:-1], Z)(qubits[-1])

def grover_layer(qubits):
    oracle(qubits)
    grover_diffusion(qubits)

def grover(size):
    process = Process(
        num_qubits=size,
        decompose=True,
        execution="batch"
    )
    qubits = process.alloc(size)
    H(qubits)
    steps = int((pi / 4) * sqrt(2**size))
    for _ in range(steps):
        grover_layer(qubits)
    return measure(qubits)
```

(a) Grover's Algorithm implemented using Ket. The Function `grover_diffusion` implementation is presented in Figure 1a.



(b) Number of CNOTs resulting from the decomposition of a single layer of Grover's algorithm.

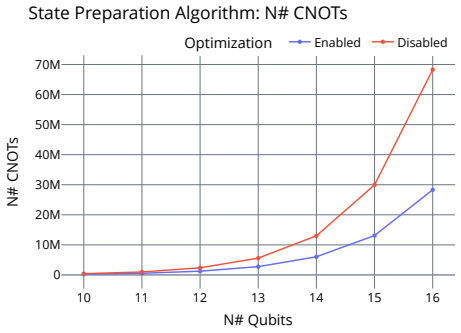
Figure 9: Grover's Algorithm evaluation.

probability and another for the phase of the probability amplitude. Although this algorithm requires an exponential number of parameters and its execution time grows exponentially with the number of qubits, making it unsuitable for solving practical problems, it is valuable for conducting quantum mechanics experiments with a quantum computer [49]. While this algorithm does not directly call for any controlled quantum gates, the recursion results in several multi-controlled R_Y and Phase gates, as illustrated in Figure 3. Figure 10b shows the number of CNOTs resulting from the execution and decomposition of this algorithm, with 4 to 16 qubits. Notably, with the execution using 16 qubits, the optimization reduced the number of CNOTs from 68,288,004 to 28,311,044.

```python
class ParamTree:
    def __init__(self, prob, amp):
        assert (len(prob)).bit_count() == 1
        total = sum(prob)
        assert total != 0
        prob = [p / total for p in prob]
        l_prob = prob[: len(prob) // 2]
        l_amp = amp[: len(prob) // 2]
        r_prob = prob[len(prob) // 2 :]
        r_amp = amp[len(prob) // 2 :]
        self.value = 2 * asin(sqrt(sum(r_prob)))
        if len(prob) > 2:
            self.l = ParamTree(l_prob, l_amp)
            self.r = ParamTree(r_prob, right_amp)
        else:
            self.l = None
            self.r = None
            self.phase0 = amp[0]
            self.phase1 = amp[1]

    def is_leaf(self):
        return self.l is None and self.r is None
```

(a) Class that implement part of the logics necessary for the State Preparation Algorithm presented in Figure 2.



(b) Number of CNOTs resulting from the decomposition of multi-controlled gates from the State Preparation Algorithm.

Figure 10: Evaluation of the State Preparation Algorithm from Figure 2.

6 Conclusion

In this paper, we have demonstrated that any multi-controlled single-target quantum gate can be decomposed into a linear number of CNOT gates in linear time, with the expense of a single auxiliary qubit in the state $|0\rangle$. In the worst case, which includes the multi-

controlled Hadamard gate, the decomposition takes at most $32n$ CNOT gates, while in the best case, such as for the multi-controlled Pauli gates, it takes at most $12n$ CNOTs. These efficient decompositions show that a quantum compiler can effectively decompose any quantum operation built with a single-qubit gate and the ability to add control qubits, as available in the Ket quantum programming platform. Therefore, decomposition should not be a concern for high-level quantum programming. However, quantum programmers will benefit from knowing that some quantum gates are more efficient than others, similar to how classical programmers know that a shift instruction takes fewer computing cycles in a CPU than a multiplication instruction.

One possible optimization for quantum circuits is combining the effect of consecutive multi-controlled operations that share the same control qubits and target, knowing that the upper bound for a decomposition is $32n$ CNOTs. This result comes from one of the main contributions of this paper: a method to rewrite $U(2)$ gates as $SU(2)$ gates with an auxiliary qubit to manage the phase differences in controlled operations. We have demonstrated that applying a controlled R_Z gate on an auxiliary qubit, initialized in the $|0\rangle$ state, can effectively fix the phase discrepancy. This technique allows us to utilize the more efficient decomposition of $SU(2)$ for any $U(2)$ gates, maintaining the integrity of controlled operations.

In conclusion, the optimization proposed in this paper, particularly the method for rewriting $U(2)$ gates as $SU(2)$ gates with phase correction using an auxiliary qubit, offers a viable path to optimizing quantum circuit designs. By minimizing the gate count, especially CNOT gates, we can improve the performance and reliability of quantum algorithms. This optimization is crucial for the practical realization of quantum computing, where gate efficiency directly impacts the overall feasibility of quantum computations.

Once all multi-controlled gates have been decomposed into CNOTs and single-qubit gates, the next step in the compilation process is adapting the quantum circuit to the connectivity constraints of the quantum computer, a process known as circuit mapping. This process involves assigning an initial physical qubit to each qubit in the program and adding SWAP gates (which can be implemented using three CNOTs) to satisfy the connectivity constraints. This step of the compilation process often increases the CNOT count of the final circuit. Evaluating the impact of the proposed optimization–using an auxiliary qubit for decomposition–on circuit mapping remains an area for future work. However, we believe that the circuit mapping process is unlikely to negate the benefits of the proposed optimization, as reducing the number of CNOTs minimizes potential connectivity mismatches that would otherwise require additional SWAP gates.

Future work may focus on exploring the impact of this optimization in other quantum algorithms. Additionally, investigating the effect of this optimization in real-world quantum hardware scenarios will be essential to fully understand and harness its benefits. Note that the auxiliary qubit cannot be used in parallel; therefore, allocating more auxiliary qubits may be beneficial. Also, due to the connectivity of the qubits in the quantum computer, allocating more auxiliary qubits in strategic locations may positively improve the quantum circuit mapping. Another possible optimization is to decompose the multi-controlled Pauli gates using dirty auxiliary qubits, with the quantum compiler automatically selects the auxiliary qubits, which would reduce the decomposition of those gates to at most $4n$ CNOTs without impacting high-level quantum programming.

Acknowledgement

EID acknowledges the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq through grant number 409673/2022-6 and ECRR acknowledges the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, Finance Code 001.

References

[1] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, July 2004. ISSN 0004-5411, 1557-735X. DOI: 10.1145/1008731.1008735.

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. ISSN 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-1666-5.

[3] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. ISSN 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.52.3457.

[4] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, and Nathan Killoran. PennyLane: Automatic differentiation of hybrid quantum-classical computations, 2018. URL https://doi.org/10.48550/arXiv.1811.04968.

[5] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, September 2017. ISSN 0028-0836, 1476-4687. DOI: 10.1038/nature23474.

[6] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. On the Complexity of Quantum Circuit Compilation. *Proceedings of the International Symposium on Combinatorial Search*, 9(1):138–142, September 2021. ISSN 2832-9163, 2832-9171. DOI: 10.1609/socs.v9i1.18463.

[7] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions: Invited paper. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Cláudio L. Lucchesi, and Arnaldo V. Moura, editors, *LATIN'98: Theoretical Informatics*, volume 1380, pages 163–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-64275-6 978-3-540-69715-2. DOI: 10.1007/BFb0054319.

[8] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum Chemistry in the Age of Quantum Computing. *Chemical Reviews*, 119(19):10856–10915, October 2019. ISSN 0009-2665, 1520-6890. DOI: 10.1021/acs.chemrev.8b00803.

[9] Nicolas J. Cerf, Lov K. Grover, and Colin P. Williams. Nested quantum search and structured problems. *Physical Review A*, 61(3):032303, February 2000. ISSN 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.61.032303.

[10] Lilian Childress and Ronald Hanson. Diamond NV centers for quantum computing and quantum networks. *MRS Bulletin*, 38(2):134–138, February 2013. ISSN 0883-7694, 1938-1425. DOI: 10.1557/mrs.2013.20.

[11] Evandro Chagas Ribeiro Da Rosa and Rafael De Santiago. Ket Quantum Programming. *ACM Journal on Emerging Technologies in Computing Systems*, 18(1):1–25, January 2022. ISSN 1550-4832, 1550-4840. DOI: 10.1145/3474224.

[12] Evandro Chagas Ribeiro da Rosa and Bruno G. Taketani. QSystem: Bitwise representation for quantum circuit simulations, 2020. URL `https://doi.org/10.48550/ARXIV.2004.03560`.

[13] Adenilton J. da Silva and Daniel K. Park. Linear-depth quantum circuits for multiqubit controlled gates. *Physical Review A*, 106(4):042602, October 2022. ISSN 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.106.042602.

[14] Cirq Developers. Cirq. Zenodo, May 2024. URL `https://doi.org/10.5281/ZENODO.4062499`.

[15] Christoph Durr and Peter Høyer. A Quantum Algorithm for Finding the Minimum, 1996. URL `https://doi.org/10.48550/ARXIV.QUANT-PH/9607014`.

[16] Laird Egan, Dripto M. Debroy, Crystal Noel, Andrew Risinger, Daiwei Zhu, Debopriyo Biswas, Michael Newman, Muyuan Li, Kenneth R. Brown, Marko Cetina, and Christopher Monroe. Fault-tolerant control of an error-corrected qubit. *Nature*, 598 (7880):281–286, October 2021. ISSN 0028-0836, 1476-4687. DOI: 10.1038/s41586-021-03928-y.

[17] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-04-15-433.

[18] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC '96*, pages 212–219, Philadelphia, Pennsylvania, United States, 1996. ACM Press. ISBN 978-0-89791-785-8. DOI: 10.1145/237814.237866.

[19] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-09-21-327.

[20] Dylan Herman, Cody Googin, Xiaoyuan Liu, Yue Sun, Alexey Galda, Ilya Safro, Marco Pistoia, and Yuri Alexeev. Quantum computing for finance. *Nature Reviews Physics*, 5(8):450–465, July 2023. ISSN 2522-5820. DOI: 10.1038/s42254-023-00603-1.

[21] Fei Hua, Meng Wang, Gushu Li, Bo Peng, Chenxu Liu, Muqing Zheng, Samuel Stein, Yufei Ding, Eddy Z. Zhang, Travis Humble, and Ang Li. QASMTrans: A QASM Quantum Transpiler Framework for NISQ Devices. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 1468–1477, Denver CO USA, November 2023. ACM. ISBN 979-8-4007-0785-8. DOI: 10.1145/3624062.3624222.

[22] You Huang, Mohammad T Amawi, Francesco Poggiali, Fazhan Shi, Jiangfeng Du, and Friedemann Reinhard. Calibrating single-qubit gates by a two-dimensional Rabi oscillation. *AIP Advances*, 13(3):035226, March 2023. ISSN 2158-3226. DOI: 10.1063/5.0139454.

[23] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. *Physical Review A*, 93(3):032318, March 2016. ISSN 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.93.032318.

[24] Raban Iten, Oliver Reardon-Smith, Emanuel Malvetti, Luca Mondada, Gabrielle Pauvert, Ethan Redmond, Ravjot Singh Kohli, and Roger Colbeck. Introduction to UniversalQCompiler, 2019. URL `https://doi.org/10.48550/ARXIV.1904.01072`.

[25] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, June 2024. URL `https://doi.org/10.48550/arXiv.2405.08810`.

[26] Shelby Kimmel, Guang Hao Low, and Theodore J. Yoder. Robust calibration of a universal single-qubit gate set via robust phase estimation. *Physical Review A*, 92(6):062315, December 2015. ISSN 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.92.062315.

[27] Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer, 2008. URL `https://doi.org/10.48550/ARXIV.0801.0342`.

[28] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. Superconducting Qubits: Current State of Play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, March 2020. ISSN 1947-5454, 1947-5462. DOI: 10.1146/annurev-conmatphys-031119-050605.

[29] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, Providence RI USA, April 2019. ACM. ISBN 978-1-4503-6240-5. DOI: 10.1145/3297858.3304023.

[30] Lars S. Madsen, Fabian Laudenbach, Mohsen Falamarzi. Askarani, Fabien Rortais, Trevor Vincent, Jacob F. F. Bulmer, Filippo M. Miatto, Leonhard Neuhaus, Lukas G. Helt, Matthew J. Collins, Adriana E. Lita, Thomas Gerrits, Sae Woo Nam, Varun D. Vaidya, Matteo Menotti, Ish Dhand, Zachary Vernon, Nicolás Quesada, and Jonathan Lavoie. Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912):75–81, June 2022. ISSN 0028-0836, 1476-4687. DOI: 10.1038/s41586-022-04725-x.

[31] Alexander J McCaskey, Dmitry I Lyakh, Eugene F Dumitrescu, Sarah S Powers, and Travis S Humble. XACC: A system-level software infrastructure for heterogeneous quantum–classical computing. *Quantum Science and Technology*, 5(2):024002, February 2020. ISSN 2058-9565. DOI: 10.1088/2058-9565/ab6bf6.

[32] David C. McKay, Christopher J. Wood, Sarah Sheldon, Jerry M. Chow, and Jay M. Gambetta. Efficient Z gates for quantum computing. *Physical Review A*, 96(2):022330, August 2017. ISSN 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.96.022330.

[33] Thien Nguyen, Anthony Santana, Tyler Kharazi, Daniel Claudino, Hal Finkel, and Alexander McCaskey. Extending C++ for Heterogeneous Quantum-Classical Computing, 2020. URL https://doi.org/10.48550/ARXIV.2010.03935.

[34] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge university press, Cambridge, 10th anniversary edition edition, 2010. ISBN 978-1-107-00217-3. DOI: 10.1017/CBO9780511976667.

[35] Eneko Osaba, Esther Villar-Rodriguez, and Izaskun Oregi. A Systematic Literature Review of Quantum Computing for Routing Problems. *IEEE Access*, 10:55805–55817, 2022. ISSN 2169-3536. DOI: 10.1109/ACCESS.2022.3177790.

[36] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann Publishers, an imprint of Elsevier, Cambridge, Massachusetts, risc-v edition edition, 2018. ISBN 978-0-12-812275-4. URL https://dl.acm.org/doi/10.5555/3153875.

[37] John Preskill. Quantum computing and the entanglement frontier, 2012. URL https://doi.org/10.48550/ARXIV.1203.5813.

[38] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. ISSN 2521-327X. DOI: 10.22331/q-2018-08-06-79.

[39] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. ISSN 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172.

[40] Marcos Yukio Siraichi, Vinícius Fernandes Dos Santos, Caroline Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125, Vienna Austria, February 2018. ACM. ISBN 978-1-4503-5617-6. DOI: 10.1145/3168822.

[41] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket⟩: A retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003, January 2021. ISSN 2058-9565. DOI: 10.1088/2058-9565/ab8e92.

[42] Leandro Stefanazzi, Kenneth Treptow, Neal Wilcer, Chris Stoughton, Collin Bradford, Sho Uemura, Silvia Zorzetti, Salvatore Montella, Gustavo Cancelo, Sara Sussman, Andrew Houck, Shefali Saxena, Horacio Arnaldi, Ankur Agrawal, Helin Zhang, Chunyang Ding, and David I. Schuster. The QICK (Quantum Instrumentation Control Kit): Readout and control for qubits and detectors. *Review of Scientific Instruments*, 93(4):044709, April 2022. ISSN 0034-6748, 1089-7623. DOI: 10.1063/5.0076249.

[43] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: An open source software framework for quantum computing. *Quantum*, 2:49, January 2018. ISSN 2521-327X. DOI: 10.22331/q-2018-01-31-49.

[44] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop 2018*,

pages 1–10, Vienna Austria, February 2018. ACM. ISBN 978-1-4503-6355-6. DOI: 10.1145/3183895.3183901.

[45] Rafaella Vale, Thiago Melo D. Azevedo, Ismael C. S. Araújo, Israel F. Araujo, and Adenilton J. Da Silva. Circuit Decomposition of Multicontrolled Special Unitary Single-Qubit Gates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(3):802–811, March 2024. ISSN 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2023.3327102.

[46] Robert Wille and Lukas Burgholzer. MQT QMAP: Efficient Quantum Circuit Mapping. In *Proceedings of the 2023 International Symposium on Physical Design*, pages 198–204, Virtual Event USA, March 2023. ACM. ISBN 978-1-4503-9978-4. DOI: 10.1145/3569052.3578928.

[47] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Physical Review Letters*, 127(18):180501, October 2021. ISSN 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.127.180501.

[48] Jun Yoneda, Kenta Takeda, Tomohiro Otsuka, Takashi Nakajima, Matthieu R. Delbecq, Giles Allison, Takumu Honda, Tetsuo Kodera, Shunri Oda, Yusuke Hoshi, Noritaka Usami, Kohei M. Itoh, and Seigo Tarucha. A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%. *Nature Nanotechnology*, 13(2):102–106, February 2018. ISSN 1748-3387, 1748-3395. DOI: 10.1038/s41565-017-0014-x.

[49] Marcelo S Zanetti, Douglas F Pinto, Marcos L W Basso, and Jonas Maziero. Simulating noisy quantum channels via quantum state preparation algorithms. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 56(11):115501, June 2023. ISSN 0953-4075, 1361-6455. DOI: 10.1088/1361-6455/accb76.

[50] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, December 2020. ISSN 0036-8075, 1095-9203. DOI: 10.1126/science.abe8770.

[51] Han-Sen Zhong, Yu-Hao Deng, Jian Qin, Hui Wang, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Dian Wu, Si-Qiu Gong, Hao Su, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Jelmer J. Renema, Chao-Yang Lu, and Jian-Wei Pan. Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light. *Physical Review Letters*, 127(18):180502, October 2021. ISSN 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.127.180502.