Efficient Simulation of High-Level Quantum Gates

Adam Husted Kjelstrøm, Andreas Pavlogiannis, and Jaco van de Pol

Department of Computer Science, Aarhus University

Quantum circuit simulation is paramount to the verification and optimization of quantum algorithms, and considerable research efforts have been made towards efficient simulators. While circuits often contain high-level gates such as oracles and multi-controlled X (C^kX) gates, existing simulation methods require compilation to a low-level gate-set before simulation. This, however, increases circuit size and incurs a considerable (typically exponential) overhead, even when the number of high-level gates is small. Here we present a gadget-based simulator which simulates high-level gates directly, thereby allowing to avoid or reduce the blowup of compilation. Our simulator uses a stabilizer decomposition of the magic state of non-stabilizer gates, with improvements in the rank of the magic state directly improving performance. We then proceed to establish a small stabilizer rank for a range of high-level gates that are common in various quantum algorithms. Using these bounds in our simulator, we improve both the theoretical complexity of simulating circuits containing such gates, and the practical running time compared to standard simulators found in IBM's Qiskit Aer library. We also derive exponential lower-bounds for the stabilizer rank of some gates under common complexity-theoretic hypotheses. In certain cases, our lower-bounds are asymptotically tight on the exponent.

1 Introduction

Testing software is the most standard approach to detecting errors [1]. In the quantum setting, this is often done by simulating quantum circuits on classical hardware [2]. Strong simulation provides the probability of observing some specific measurement of the qubits after applying the target circuit. Remarkably, strong simulation can also be used to prove two quantum circuits equivalent [3], which has immediate applications in circuit optimization [4].

The importance of strong simulation has led to the development of a variety of simulation techniques, targeting efficiency. Given a quantum circuit of n qubits and m gates, the simplest way is to store each quantum state explicitly, requiring time and space that is exponential in n, which is prohibitive. Feynman [5] proposed a recursive approach with time exponential in m, at the benefit of polynomial space. Aaronson and Gottesman [6] showed that stabilizer circuits can be simulated in $O(n^2m)$ time and $O(n^2)$ space. Although stabilizer circuits are not capable of arbitrary quantum computations, they are powerful enough to generate quantum effects such as superposition and entanglement.

Adam Husted Kjelstrøm: husted@cs.au.dk
Andreas Pavlogiannis: pavlogiannis@cs.au.dk
Jaco van de Pol: jaco@cs.au.dk

The stabilizer simulator has been extended to universal quantum circuits of stabilizer, T, and CCX gates [7, 8] by substituting each T and CCX gate by a small circuit that applies the same effect, using a, so called, "magic state" in the process [9]. Specifically for t T gates, the simulator runs in $O(2^{0.396t}n^2m)$ time and $O(t+n^2)$ space [10].

While T or CCX gates suffice to make stabilizer circuits universal for quantum computing, quantum circuits are generally composed from a much richer gate-set [11, 12, 13, 14], including multi-controlled Toffoli gates (C^kX), arbitrary multi-controlled single-qubit unitaries (C^kU), and oracle gates, which are a standard means of providing quantum access to a function. In particular, given a Boolean function $\varphi\colon \{0,1\}^k \to \{0,1\}$, the oracle gate $C_\varphi U$ applies the single qubit unitary U to the target qubit $k+1$ when the state of the control qubits $1,\ldots,k$ satisfies φ. Oracle gates are used in Grover's algorithm [15], the Deutsch-Josza algorithm [16], the Bernstein-Vazirani algorithm [17], Shor's algorithm [18], and Simon's algorithm [19], among others.

Circuits containing such high-level gates can still be simulated somewhat efficiently (at least in theory), by compiling the high-level gates to stabilizer+T equivalents, with active research on designing efficient compilation schemes [20, 21]. In particular for C^kU, these schemes convert each such gate into $\Theta(k)$ T gates. However, since the number of T gates contributes exponentially to the running time, simulating the compiled circuits has been a challenge. *Can we avoid the overhead of compilation schemes? Is it possible to simulate high-level circuits more efficiently?*

Theoretical contributions. We answer these questions by providing an extension to the existing stabilizer-based quantum simulation theory, generalizing from T and CCX gates. In particular, assume we want to simulate a target circuit C containing some high-level gates. Our method is to "implement" each such high-level gate G by some stabilizer circuit and some stabilizer decomposition of its magic input state. More precisely, let C be a circuit on n qubits, and assume that each high-level gate G in it can be implemented by a stabilizer circuit $\widehat{G}$ of $\leq \mu$ gates, applied to a magic state $|\psi\rangle$ that is decomposed as a weighted sum of k stabilizer states, each initialized by $\leq \mu$ gates. Note that $k \geq \chi(|\psi\rangle)$, the (minimal) stabilizer rank of $|\psi\rangle$ [7]. We generally restrict $\widehat{G}$ to using $O(n)$ ancillae. We call such an implementation a (μ,k)-decomposition of G. The following theorem describes the complexity of simulating C, given such an implementation.

Theorem 1. *Let C be a circuit over n qubits and m gates composed of stabilizer gates as well as t non-stabilizer gates $G_1, G_2, \ldots, G_t$. Let $x \in \{0,1\}^n$ be a measurement outcome. Assume that each G_j (for $1 \leq j \leq t$) has a (μ_j, k_j)-decomposition using $O(n)$ ancillae. Set $\chi = \prod_j k_j$, and $\mu = \max_j \mu_j$. Then we can compute $|\langle x|C|0^n\rangle|^2$, the probability of observing x, in time $O(\chi n^2(m+t\mu))$. The algorithm uses $O(\log\chi + n^2)$ space, and can be parallelized for a run-time of $O(n^2(m+t\mu))$ on χ threads.*

We note that the restriction to using $O(n)$ ancillae suffices for the high-level gates we consider in this work, and it simplifies the formulation since the cost of $O(n)$ ancillae is absorbed in time bound mentioned in Theorem 1.

We will illustrate applications to some high-level gates below. Compared to the standard process of compiling high-level gates to Clifford+T and then using existing simulation techniques, our bounds are smaller as long as the individual values k_j and μ_j are small. This motivates identifying an implementation for the high-level gates of interest that keeps these quantities small. To this end, we consider various multi-controlled and oracle gates G, and find (μ,k)-decompositions for which we establish small upper bounds k on the stabilizer rank, and gate size $\mu = O(n)$.

Towards a bound on the stabilizer rank of oracle gates $\chi(C_\varphi U)$, for some Boolean predicate $\varphi\colon \{0,1\}^k \to \{0,1\}$ and single-qubit unitary U, we define the "effectual" state $|E(\varphi)\rangle$ of φ as the sum of all basis states $|x\rangle$ for which $\varphi(x)$ is true:

$$|E(\varphi)\rangle \propto \sum_{x\in\{0,1\}^k\colon \varphi(x)} |x\rangle \ .$$

Here the "proportional to" notation $\propto$ denotes equality up to a scaling factor. We derive a bound on $\chi(C_\varphi U)$ as a function of $\chi(|E(\varphi)\rangle)$, generally establishing Theorem 2.

Theorem 2. *Let φ be a Boolean predicate. Then the following properties hold for the stabilizer rank, where $R_Z(\theta)$ and $R_X(\theta)$ are the standard rotation gates, and U is an arbitrary single-qubit unitary.*

$$\chi(C_\varphi R_Z(\theta)) \le \chi(|E(\varphi)\rangle) + 1 \tag{1}$$
$$\chi(C_\varphi R_X(\theta)) = \chi(C_\varphi R_Z(\theta)) \tag{2}$$
$$\chi(C_\varphi U) \le \begin{cases} 8, & \text{if } \chi(|E(\varphi)\rangle) = 1 \\ 8\chi(|E(\varphi)\rangle) + 8, & \text{otherwise.} \end{cases} \tag{3}$$

A simple strategy for bounding $\chi(|E(\varphi)\rangle)$ is to note that for all $x \in \{0,1\}^k$, $|x\rangle$ is a stabilizer state, giving $\chi(|E(\varphi)\rangle) \le |\{x : \varphi(x)\}| \le 2^k$. However, since φ is an arbitrary formula, we do not have non-trivial upper bounds on $\chi(|E(\varphi)\rangle)$. Nonetheless, as predicates are often inductively defined over logical connectives such as conjunctions, disjunctions, and negations, we derive more refined bounds by following this structure.

Lemma 1. *Let φ_1 and φ_2 be Boolean formulas. The following inequalities hold.*

$$\chi(|E(\neg\varphi_1)\rangle) \le \chi(|E(\varphi_1)\rangle) + 1 \tag{4}$$
$$\chi(|E(\varphi_1 \wedge \varphi_2)\rangle) \le \chi(|E(\varphi_1)\rangle)\chi(|E(\varphi_2)\rangle) \tag{5}$$
$$\chi(|E(\varphi_1 \vee \varphi_2)\rangle) \le \chi(|E(\varphi_1 \wedge \varphi_2)\rangle) + \chi(|E(\varphi_1)\rangle) + \chi(|E(\varphi_2)\rangle) \tag{6}$$

Note that $|\{x : \varphi(x)\}| + |\{x : \neg\varphi(x)\}| = 2^k$, which, together with Eq. (4), implies $\chi(|E(\varphi)\rangle) \le 2^{k-1}$ in general. Finally, we establish some direct bounds when φ has a specific form, in particular performing comparison between integers.

Theorem 3. *Let x and y be formal variables ranging over k-bit integers. The following (in)equalities hold.*

$$\chi(|E(x = y)\rangle) = 1 \tag{7}$$
$$\chi(|E(x > y)\rangle) \le k \tag{8}$$
$$\chi(|E(x + 1 = y)\rangle) \le k + 1 \tag{9}$$

Note that the equations also hold if x or y is a constant. Our work permits the simulation of any gate where an upper bound on an applicable magic state can be derived through Theorems 2 and 3 and Lemma 1. We proceed to give a number of concrete examples of how these inequalities enable the derivation of other bounds.

Application 1: MCX. Consider the multi-controlled gate $C^k\mathrm{X}$, also known as the MCX gate. This gate is omnipresent in quantum singular value transformations [22] as well as

in circuits performing arithmetic [23]. It applies an X when all k control bits $x_1, \ldots, x_k$ are 1. This means $C^k\mathrm{X} = C_{x_1 \ldots x_k = 1^k}\mathrm{X}$. Using Theorems 2 and 3, we obtain

$$\begin{aligned}
\chi(C^k\mathrm{X}) &= \chi(C^k\mathrm{R}_X(\pi)) && [\mathrm{X} = \mathrm{R}_X(\pi)] \\
&= \chi(C^k\mathrm{R}_Z(\pi)) && [\text{Eq. (2)}] \\
&\leq \chi(|E(x_1 \ldots x_k = 1^k)\rangle) + 1 && [\text{Eq. (1)}] \\
&\leq 2 && [\text{Eq. (7)}]
\end{aligned}$$

Combining with Theorem 1, we see that stabilizer circuits with t $C^k\mathrm{X}$ gates can be strongly simulated in $O(2^t n^2(m + kt))$ time. In particular, observe that the exponential factor of 2^t is independent of k, as opposed to $2^{O(kt)}$ achieved by standard compilation-driven techniques [20, 21]. This recovers a result of [24].

Application 2: C^kU. Consider a C^kU gate for an arbitrary, single-qubit unitary U. As $C^kU = C_{x_1 \ldots x_k = 1^k}U$, and $\chi(|E(x_1 \ldots x_k = 1^k)\rangle) = 1$, we apply Eq. (3) to obtain $\chi(C^kU) \leq 8$. Then a circuit with t C^kU gates, such as a CVO-QRAM circuit [11], suffers an exponential factor of 8^t, again avoiding an exponential dependency on k.

Application 3: An oracle gate. Consider the parameterized gate $C_{x \geq y}\mathrm{R}_Z(\theta)$. By combining Eq. (8) with Eq. (4), we see that

$$\chi(|E(x \geq y)\rangle) = \chi(|E(\neg(y > x))\rangle) \leq \chi(|E(y > x)\rangle) + 1 \leq k + 1.$$

By Eq. (1) and Eq. (8), $\chi(C_{x \geq y}\mathrm{R}_Z(\theta)) \leq k+2$, so by Theorem 1, we obtain an exponential factor of $(k+2)^t = 2^{O(t \log k)}$, an improvement over the compilation-driven $2^{O(kt)}$ [25].

With Theorems 2 and 3 and Lemma 1 concerning the rank of conditional single-qubit gates, a natural question is whether we can achieve similar performance improvements for multi-qubit gates. We consider query gates, a standard way to provide quantum access to an arbitrary Boolean mapping $\{0,1\}^\ell \to \{0,1\}^\ell$, which are used, e.g., in Simon's algorithm [19] and Shor's algorithm for period finding [18]. Our results on Boolean mappings are captured in Theorem 4.

Theorem 4. *Let $g\colon \{0,1\}^\ell \to \{0,1\}^\ell$ be a Boolean mapping, and define the query gate U_g to compute*

$$|x\rangle\,|0^\ell\rangle \xrightarrow{U_g} |x\rangle\,|g(x)\rangle\,.$$

Then we have

$$\chi(U_g) \leq \chi(|E(y = g(x))\rangle) + 1 \tag{10}$$

$$\chi(C_\varphi U_g) \leq \chi(|E(\varphi)\rangle)(\chi(|E(y = g(x))\rangle) + 1) + 2 \tag{11}$$

Moreover, we have

$$\chi(|E(y = g(x))\rangle) \leq 2^\ell \tag{12}$$

Although the bound of Eq. (12) is exponential in ℓ, note that the exponent is half the number of qubits that U_g acts on (i.e., 2ℓ qubits). While this is a general upper bound, tighter bounds can be obtained for specific choices of g.

Application 4: Increment. One such example is the increment gate INC_ℓ, which computes

$$|x\rangle\,|0^\ell\rangle \xrightarrow{\mathrm{INC}_\ell} |x\rangle\,|(x+1) \mod 2^\ell\rangle$$

for ℓ-bit numbers x. This gate is a useful component in quantum convolution [26], solving time-dependent linear differential equations [27], and in quantum walks [28]. Combining Theorem 4 with Eq. (9) of Theorem 3, we see that $\chi(\mathrm{INC}_\ell) \leq \ell+2$. Using compilation [29], we instead obtain $\chi(\mathrm{INC}_\ell) = 2^{O(\ell)}$.

As we have derived low (polynomial) bounds on the rank of certain high-level gates captured in our theorems, a natural question is whether other high-level gates also have polynomial rank. We consider the ADD_ℓ, MUL_ℓ, and QFT_ℓ gates, implementing ℓ-bit addition, Quantum Fourier Transform, and multiplication, respectively. By applying the Choi-Jamiołkowski isomorphism [30, 31], any ℓ-qubit unitary U can be implemented using a magic state $\frac{1}{\sqrt{2^\ell}} \sum_x |x\rangle (U|x\rangle)$, which, by taking the basis vectors as the stabilizer decomposition, gives upper-bounds $\chi(\mathrm{ADD}_\ell) \leq 2^{2\ell}$, $\chi(\mathrm{MUL}_\ell) \leq 2^{2\ell}$, and $\chi(\mathrm{QFT}_\ell) \leq 2^{2\ell}$, respectively. The next theorem states that the ranks of these gates are indeed expected to grow exponentially in ℓ, and thus significant improvements over the aforementioned implementations are unlikely.

Theorem 5. *Each of $\chi(ADD_\ell)$, $\chi(MUL_\ell)$, and $\chi(QFT_\ell)$ is hyper-polynomial unless* $\mathbf{P} = \mathbf{NP}$*, and is $2^{\Omega(\ell)}$ under ETH.*

Theorem 5 is derived by constructing, for each of the target gates, a family of circuits which contains a fixed number of such gates, and requires exponential time to simulate, under ETH [32]. Then applying Theorem 1 to the circuits, we find that the rank must be exponential under the hypothesis. For the ADD_ℓ, MUL_ℓ, and QFT_ℓ gates, our proofs further establish that the corresponding $2^{2\ell}$ upper bound on the stabilizer rank is asymptotically tight on the exponent. It is known that circuits composed of Clifford and arbitrarily many ADD_ℓ gates, each acting on $\ell = O(1)$ qubits, are universal for quantum computing and are generally expected to have hyper-polynomial rank. However, the use of arbitrarily many ADD_ℓ gates is insufficient to lower bound the stabilizer rank of a single ADD_ℓ gate. In contrast, our reduction uses circuits composed of Clifford and a constant number of ADD_ℓ gates (and only one $C^\ell\mathrm{X}$ gate), where ℓ is not bounded. This allows us to establish lower bounds on the rank of ADD_ℓ (and likewise for MUL_ℓ and QFT_ℓ).

Practical contributions. We implement our simulator behind Theorem 1 and evaluate its performance on simulating various benchmarks containing high-level gates, the stabilizer rank of which we have bounded in Theorem 2. In particular, we benchmark on CVO-QRAM state preparation circuits [11], as well as oracle gates and instances of Grover's algorithm. In all cases, we compare the performance of our direct simulation to the performance of the Qiskit Aer simulation library [33], which compiles the high-level circuit before simulating it. In all cases, our direct simulator outperforms the Qiskit Aer baselines by several orders of magnitude as instance sizes grow.

2 Preliminaries

Throughout this paper, we use $a, b, \cdots \in \{0,1\}$ to denote single (classical) bits, and $x, y, \cdots \in \{0,1\}^n$ to denote bit-strings of length n, where n is either explicit or clear from the context. Given such a bit-string x and some $j \in \{1, \ldots, n\}$, we denote by $x_j \in \{0,1\}$ the j'th bit of x. For two bit-strings x and y of equal length, we write $x \oplus y$ for their bitwise XOR operation.

Quantum states. An n-qubit quantum state is a 2^n-dimensional vector

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \psi_x |x\rangle$$

where the set of basis states $\{|x\rangle\}_{x\in\{0,1\}^n}$ form an orthonormal basis for a 2^n-dimensional vector space, and $\psi_x \in \mathbb{C}$ are complex numbers satisfying $\sum_{x\in\{0,1\}^n} |\psi_x|^2 = 1$. For convenience, we often write states such as $|0\rangle + |1\rangle$, with the understanding that they are scaled by real, non-negative constants such that $\sum_{x\in\{0,1\}^n} |\psi_x|^2 = 1$. To be explicit about this, we write, e.g., $|\psi\rangle \propto |0\rangle + |1\rangle$ to mean that, $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. For quantum states $|\psi_1\rangle$ and $|\psi_2\rangle$, we write their (tensor) product state as $|\psi_1\rangle |\psi_2\rangle$. One state that is used throughout this paper is the uniform superposition of n qubits, denoted as

$$|+^n\rangle \propto \sum_{x\in\{0,1\}^n} |x\rangle \ .$$

Quantum gates and circuits. A quantum state evolves by applications of quantum gates. A quantum gate G corresponds to a $2^n \times 2^n$-dimensional unitary matrix U_G. We write $U_G^\dagger$ for the conjugate transpose of U_G. Applying G to a state $|\psi\rangle$ means multiplying U_G onto $|\psi\rangle$, obtaining the state $U_G |\psi\rangle$. Since

$$U_G |\psi\rangle = \sum_{x\in\{0,1\}^n} \psi_x U_G |x\rangle \,,$$

a quantum gate can be described by how it acts on the basis states. Given unitaries U, U_1, U_2, we write $U_1 \otimes U_2$ for the tensor product between U_1 and U_2, and $U^{\otimes k}$ for the k-fold tensor product of U. If a gate D is diagonal (i.e., all non-diagonal entries of U_D are 0), we write D_x for the (x,x)-th entry of U_D, with $x \in \{0,1\}^n$. A diagonal gate D has the effect $|x\rangle \xrightarrow{D} D_x |x\rangle$ for each x.

A circuit C is a sequence of gates, where the i'th gate corresponds to a unitary U_i. Then C implements the unitary $U_C = U_m U_{m-1} \cdots U_1$. For convenience, given a state $|\psi\rangle$, we write $C |\psi\rangle$ to denote the state $U_C |\psi\rangle$. We say that two unitaries U_1 and U_2 are equivalent (up to a global phase), written as $U_1 \equiv U_2$, if there is a real number θ such that we have $U_1 = e^{i\theta} U_2$. For circuits C_1 and C_2, we take $C_1 \equiv C_2$ to mean $U_{C_1} \equiv U_{C_2}$, and $C_1 \equiv U_2$ to mean $U_{C_1} \equiv U_2$.

Clifford gates. The Clifford gate set consists of the Phase (S), Hadamard (H), and CNOT (CX) gates, and constitutes an important family of quantum gates, each acting on the basis states as follows.

$$|a\rangle \xrightarrow{\mathrm{S}} i^a |a\rangle \qquad |a\rangle \xrightarrow{\mathrm{H}} \frac{1}{\sqrt{2}} \sum_{b=0}^{1} (-1)^{ab} |b\rangle \qquad |a,b\rangle \xrightarrow{\mathrm{CX}} |a, a\oplus b\rangle$$

The Clifford gates suffice to construct other gates such as X and Z, and can be used to construct quantum states such as an EPR pair $1/\sqrt{2}(|00\rangle + |11\rangle)$.

Measurements. While gates are invertible and affect the state deterministically, measurements are probabilistic and are not invertible. Suppose we measure the j'th bit of a state $|\psi\rangle = \sum_x \psi_x |x\rangle$ in the 0,1-basis. The probability of outcome $b \in \{0,1\}$ of the measurement is $\sum_{x:\, x_j=b} |\psi_x|^2$. After measurement, the state is updated to $|\psi'\rangle \propto \sum_{x:\, x_j=b} \psi_x |x\rangle$. This notion naturally generalizes to simultaneous measurements on several qubits.

In simulation, as opposed to the real world, measurement outcomes can be post-selected, meaning the outcome is given instead of being sampled[1]. As an example, given a state $1/\sqrt{2}(|00\rangle + |11\rangle)$, the post-selected measurement of the first qubit to 0 is $|00\rangle$.

Stabilizer circuits. Together, Clifford gates and 0,1-measurements are called stabilizer gates [34]. Circuits consisting only of stabilizer gates are called stabilizer circuits. Any state of the form $C|0^n\rangle$ where C is a stabilizer circuit is called a stabilizer state. Concrete examples include any basis vector $|x\rangle$, for $x \in \{0,1\}^n$, as well as the uniform superposition state $|+^n\rangle$.

While stabilizer circuits can be efficiently simulated on a classical computer [6], they are not universal for quantum computing. States that cannot be expressed as $C|0^n\rangle$, for a stabilizer circuit C, are called non-stabilizer states, and can only be reached using some non-stabilizer gates.

Universal circuits. A common choice of a non-stabilizer gate is the T gate, defined as $|a\rangle \xrightarrow{\mathrm{T}} (e^{i\pi/4})^a |a\rangle$. Another choice is the R_Z gate, a gate which is parameterized by a real number θ, with $|a\rangle \xrightarrow{\mathrm{R}_Z(\theta)} (e^{i\theta})^a |a\rangle$. Access to T gates or R_Z gates makes stabilizer circuits universal for quantum computation, meaning that any quantum state can be approximated to arbitrary precision [35]. Both T and R_Z are diagonal gates, hence, in order to simulate arbitrary quantum computations, it suffices to simulate stabilizer+diagonal circuits.

Magic states and state injection. Given a circuit C, we can substitute any diagonal, non-stabilizer gate D for a gadget which consumes an appropriate non-stabilizer state, called a *magic state* [9], defined as

$$|D\rangle := \sum_{x\in\{0,1\}^n} D_x |x\rangle \ .$$

This is known as magic state injection, and is commonly simulated with post-selection [8], which reduces the size of the state injection gadget. Specifically, if D acts on n qubits, the magic state injection gadget uses n CX gates and n measurement gates with post-selection, as well as n ancillary bits to store $|D\rangle$.

Fig. 1 provides an illustration. The circuit performs the transformation $|x\rangle \to D_x |x\rangle$ via the following steps:

$$\begin{aligned} |x\rangle |D\rangle &\propto \sum_{y\in\{0,1\}^n} D_y |x\rangle |y\rangle \\ &\xrightarrow{\mathrm{CX}^{\otimes k}} \sum_{y\in\{0,1\}^n} D_{x\oplus y} |x\rangle |y\rangle \\ &\xrightarrow{Meas.} D_{x\oplus 0^k} |x\rangle |0^k\rangle = D_x |x\rangle |0^k\rangle \end{aligned}$$

Discarding the bits storing $|0^k\rangle$, we obtain $D_x |x\rangle$ as desired. As magic state injection gadgets consist only of stabilizer gates, the task of simulating arbitrary quantum circuits is thereby reduced to simulating stabilizer circuits on magic states. Although magic states are not (generally) stabilizer states, they can be represented as superpositions of stabilizer states, a process known as stabilizer decomposition.

[1]Naturally, we cannot post-select on an outcome that has 0 probability, as this would collapse the state of the system to the zero vector which is not a quantum state.

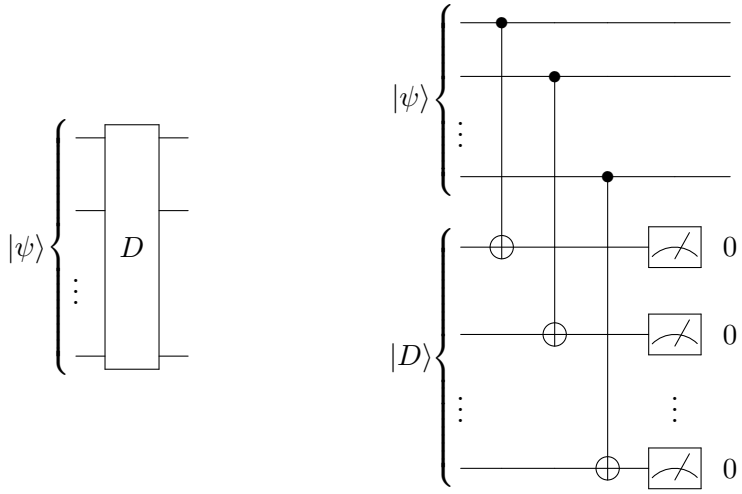

Figure 1: An implementation of a diagonal gate via magic state injection. *Left:* a diagonal gate D. *Right:* an equivalent state injection gadget consuming a magic state $|D\rangle$. The gadget uses state teleportation and measurements with post-selection on the outcome $00\ldots0$ to apply D on $|\psi\rangle$.

Stabilizer rank. Any arbitrary (not necessarily stabilizer) state can be represented as a weighted sum

$$|\psi\rangle = \sum_{j=1}^{k} c_j |\psi_j\rangle \ , \tag{13}$$

where $|\psi_j\rangle$ are stabilizer states and c_j are complex constants. Indeed, one can always pick $k = 2^n$ and let $|\psi_j\rangle$ be the basis states $|0^n\rangle, \ldots, |1^n\rangle$, giving the definition of a quantum state. The smallest value of k for which Eq. (13) holds is known as the stabilizer rank $\chi(|\psi\rangle)$ of $|\psi\rangle$ [7]. Note that by definition, $\chi(|\psi\rangle) = 1$ iff $|\psi\rangle$ is a stabilizer state. Bravyi et al. [7] show that the stabilizer rank is submultiplicative, i.e., for all states $|\phi\rangle$ and $|\psi\rangle$, we have

$$\chi(|\phi\rangle |\psi\rangle) \leq \chi(|\phi\rangle)\chi(|\psi\rangle). \tag{14}$$

3 A Gadget-based Simulation Algorithm

In this section, we turn our attention to the simulation of circuits containing high-level non-stabilizer gates, that are generally non-diagonal, proving Theorem 1. Towards this, we first generalize the concept of stabilizer rank from diagonal gates to arbitrary unitaries. This allows us to lift the standard simulation algorithm based on state injection of diagonal gates [8] to an algorithm handling arbitrary unitaries, with its complexity depending on the stabilizer rank of such unitaries. In later sections, we study the stabilizer rank of various high-level gates of general interest, thereby improving the simulation complexity of circuits containing such gates.

Stabilizer rank of arbitrary gates. For a stabilizer+diagonal circuit C with non-stabilizer diagonal gates $D_1, D_2, \ldots, D_k$, we define the magic state of C as $|C\rangle := |D_1\rangle |D_2\rangle \ldots |D_k\rangle$. This yields $\chi(|C\rangle) = \chi(|D_1\rangle |D_2\rangle \ldots |D_k\rangle)$. Next, we lift the notion of stabilizer rank to arbitrary unitaries. In particular, given some arbitrary gate G, we define the stabilizer rank of G as the minimum stabilizer rank of the magic state among

all stabilizer+diagonal circuits which are equivalent to G:

$$\chi(G) := \min_{C}\{\chi(|C\rangle)\colon C \equiv G \text{ and } C \text{ is a stabilizer+diagonal circuit}\} \ . \tag{15}$$

Observe that if C is a stabilizer+diagonal circuit with corresponding unitary U_C, we have $\chi(U_C) \leq \chi(|C\rangle)$. A strict inequality $\chi(U_C) < \chi(|C\rangle)$ implies that there exists another stabilizer+diagonal circuit C' which (i) defines the same unitary up to global phase, i.e., $C' \equiv C$, and (ii) its associated magic state has a smaller rank, i.e., $\chi(|C'\rangle) < \chi(|C\rangle)$. For example, consider a unitary U_C implemented by $C = \mathrm{TTT}$. Here $|C\rangle = |\mathrm{T}\rangle\,|\mathrm{T}\rangle\,|\mathrm{T}\rangle$, giving $\chi(|C\rangle) = 3$ [10]. By using the identity $\mathrm{S} = \mathrm{TT}$, we see that U_C is also implemented as $C' = \mathrm{ST}$. Here we obtain $|C'\rangle = |\mathrm{T}\rangle$, giving $\chi(|C'\rangle) = 2 < \chi(|C\rangle)$, since the state $|\mathrm{T}\rangle \propto |0\rangle + \sqrt{i}\,|1\rangle$ has stabilizer rank 2.

Low-rank implementation strategies. One way to simulate an arbitrary circuit is to first substitute its non-stabilizer gates for stabilizer+diagonal sub-circuits. The choice for such a substitution, and in particular the rank of each sub-circuit can significantly impact performance. We implement each non-stabilizer gate G by a stabilizer circuit $\widehat{G}$ applied to a magic state $|G\rangle$. In particular, $|G\rangle$ is given as $(c_1, C_1), (c_2, C_2), \ldots, (c_k, C_k)$ for stabilizer circuits C_j and constants c_j, encoding $|G\rangle = \sum_{j=1}^{k} c_j C_j\,|00\ldots0\rangle$. We call this a (μ, k)-decomposition of G where naturally, we have $\chi(G) \leq k$. We require that $\widehat{G}$ uses $O(n)$ ancillae, and that $\widehat{G}$ and each of $C_1, \ldots, C_k$ is over $\leq \mu$ gates. For correctness, we also require that $(G\,|\psi\rangle)\,|00\ldots0\rangle \equiv \widehat{G}\,|\psi\rangle\,|G\rangle$.

Strong simulation. Given a circuit C and a bit-string x, the task of strong simulation is to compute the probability of the state $|x\rangle$ after C has been executed on $|0^n\rangle$, written $|\,\langle x|C|0^n\rangle\,|^2$. If C contains non-stabilizer gates $G_1, G_2, \ldots, G_t$, the above probability can be computed by (i) substituting each G_j with a (μ_j, k_j)-decomposition $\widehat{G}_j$, thereby obtaining a "gadgetized" stabilizer circuit C' consuming a magic state $|C'\rangle = |G_1\rangle\,|G_2\rangle \ldots |G_t\rangle$, and (ii) evaluating

$$|\,\langle x|\,\langle 00\ldots0|\,C'\,|0^n\rangle\,|C'\rangle\,|^2 = |\,\langle x|C|0^n\rangle\,|^2 \ . \tag{16}$$

Using Eq. (14), we see that the stabilizer rank of the magic state $|C'\rangle$ is upper-bounded by

$$\chi(|C'\rangle) \leq \prod_{j=1}^{t} k_j \ .$$

Let $\chi := \prod_{j=1}^{t} k_j$, and thus we can write $|C'\rangle = \sum_{j=1}^{\chi} c_j\,|\psi_j\rangle$, where each $c_j = a_1 a_2 \ldots a_t$ is a product of unique choices of coefficients in the representations of the respective $|G_1\rangle\,|G_2\rangle \ldots |G_t\rangle$, and $|\psi_j\rangle$ is the tensor product of the corresponding stabilizer states (and thus a stabilizer state). Substituting this sum into Eq. (16), we obtain

$$\begin{aligned}\Big|\,\langle x|\,\langle 00\ldots0|\,C'\,|0^n\rangle\,|C'\rangle\,\Big|^2 &= \Big|\,\langle x|\,\langle 00\ldots0|\sum_{j=1}^{\chi} c_j C'\,|0^n\rangle\,|\psi_j\rangle\,\Big|^2 \\ &= \Big|\sum_{j=1}^{\chi} c_j\,\langle x|\,\langle 00\ldots0|\,C'\,|0^n\rangle\,|\psi_j\rangle\,\Big|^2 \ .\end{aligned} \tag{17}$$

We evaluate the sum of Eq. (17) by performing χ simulations, each of the stabilizer circuit C' applied to a stabilizer state $|0^n\rangle\,|\psi_j\rangle$, and then weighing and summing the resulting values $\langle x|\,\langle 00\ldots0|\,C'\,|0^n\rangle\,|\psi_j\rangle$. Strong simulation algorithms of stabilizer circuits generally provide $|\,\langle x|\,\langle 00\ldots0|\,C'\,|0^n\rangle\,|\psi_j\rangle\,|^2$, which does not suffice to compute

$\langle x| \langle 00\dots0| C' |0^n\rangle |\psi_j\rangle$, as the complex phase is lost. Hence, this approach requires a phase-sensitive simulation. Using directly the phase-sensitive simulator of [8] χ times, we obtain an algorithm with a total run-time of $O(\chi n_g^2 m_g)$, where n_g and m_g are the number of qubits and number of gates of C', respectively.

We can relate n_g and m_g to the number of qubits n and gates m of the original circuit C, as follows. Each magic state of $\widehat{G}_j$ is over $O(n)$ qubits, implying that $|C'\rangle$ is a state over $O(tn)$ qubits, therefore $n_g = O(tn)$. Moreover, each G_j in C is replaced by a (μ_j, k_j)-decomposition $\widehat{G}_j$ in C'. Defining $\mu := \max_j \mu_j$, we have $m_g \leq m + 2t\mu$. Thus, the complexity of the simulation is $O(\chi t^2 n^2 (m + t\mu))$.

In the remainder of this section, we outline some simple optimizations that remove the t^2 factor from the above run-time, thereby arriving at Theorem 1. In later sections, we provide (μ, k)-decompositions of various natural high-level gates G where the stabilizer rank k is small and $\mu = O(n)$.

Efficiency improvements. The time and space complexity of the above simulation process can be optimized further by storing the magic state $|C'\rangle$ more efficiently.

First, note that simply storing the full magic state $|C'\rangle$ requires $\Theta(tn)$ qubits. However, most of these qubits are only used once, which allows for the following optimization. Let $|\psi_1\rangle$ be the first magic state consumed by C'. During the simulation process, we initially store only $|\psi_1\rangle$ in ancillary qubits. After $|\psi_1\rangle$ has been consumed, post-selecting has reset all ancillary qubits, now storing $|00\dots0\rangle$. Thus we use the same ancillae for storing the second magic state $|\psi_2\rangle$, etc. Overall, our simulation now operates over circuits of $O(n)$ qubits, which improves the run-time by a factor of t^2 compared to the baseline simulation, thereby arriving at Theorem 1. As a side-note, one could permit $\omega(n)$ ancillae, which would further impact the asymptotic run-time of Theorem 1. We do not treat this general case in this work, as $O(n)$ ancillae suffice for the high-level gates we consider in later sections.

Second, the summation in Eq. (17) can also be naturally parallelized, yielding a parallel run-time of $O(\log \chi + n^2(m + t\mu)) = O(n^2(m + t\mu))$ on χ threads. Each term of the sum requires $O(n^2)$ memory to run the phase-sensitive simulator of [8]. As we perform χ such simulations in a row, we need $O(\log \chi)$ bits to track how many simulations have been performed so far. Hence, memory is bounded by $O(\log \chi + n^2)$, plus the space required to store two complex numbers, one for the phase on the on-going stabilizer simulation, and one for the running sum. Note that in the worst case, every gate G is non-stabilizer and has a (μ, k)-decomposition with $k = 2^{O(n)}$, thus χ is upper-bounded by $2^{O(nm)}$. Hence, the memory is polynomial in n and m (in particular, $O(nm + n^2)$), regardless of χ.

One potential downside of treating each non-stabilizer gate G_j independently is that we might not arrive at the best possible low-rank representation of C. As an example, consider a circuit C consisting of two T gates as $\mathrm{T} \otimes \mathrm{T}$. Treating each of them independently by substituting for $\widehat{\mathrm{T}}$ yields $\chi = 4$, since $|\mathrm{T}\rangle \propto |0\rangle + \sqrt{i}\,|1\rangle$. However, the combined gate $\mathrm{T} \otimes \mathrm{T}$ has a direct low rank magic state representation as $|\mathrm{T} \otimes \mathrm{T}\rangle \propto (|00\rangle + i\,|11\rangle) + \sqrt{i}(|01\rangle + |10\rangle)$, yielding a rank of 2 [10]. As a result, the bound of Theorem 1 is not tight.

4 Upper-bounds on the Stabilizer Rank

We now turn our attention to various natural high-level gates G, and provide bounds on their stabilizer rank $\chi(G)$, towards Theorems 2 to 4. We do so by presenting (μ, k)-decompositions $\widehat{G}$ for which k is small. In all cases, $\widehat{G}$ uses $O(n)$ gates, i.e., $\mu = O(n)$. We

will be using ancillary qubits in our constructions, but the qubit count remains $O(n)$, in particular avoiding an asymptotic impact on the run-time of Theorem 1.

4.1 Bounds on the Rank of Conditional Single-Qubit Gates

Conditional single-qubit gates are gates of the form $C_\varphi U$ where U is a single-qubit unitary. We derive Theorem 2 by establishing upper bounds on $\chi(C_\varphi U)$ for different choices of such $C_\varphi U$.

Bounds on $\chi(C_\varphi \mathrm{R}_Z)$. We apply controls to $\mathrm{R}_Z(\theta)$ specified by a Boolean predicate $\varphi\colon \{0,1\}^k \to \{0,1\}$. We define

$$C_\varphi \mathrm{R}_Z(\theta)\,|x,b\rangle = \begin{cases} e^{i\theta}\,|x,b\rangle\,, & \text{if } \varphi(x)\wedge b = 1 \\ |x,b\rangle\,, & \text{otherwise} \end{cases}$$

We implement this via the magic state $|C_\varphi \mathrm{R}_Z(\theta)\rangle \propto \sum_{x,b} (e^{i\theta})^{\varphi(x)\wedge b}\,|x,b\rangle$. We write this as

$$|C_\varphi \mathrm{R}_Z(\theta)\rangle \propto \sum_{x,b} |x,b\rangle + (e^{i\theta}-1) \sum_{x\colon \varphi(x)} |x,1\rangle$$

where x ranges over $\{0,1\}^k$, and b ranges over $\{0,1\}$. Given a Boolean function φ, we define its "effectual" state E as

$$|E(\varphi)\rangle \propto \sum_{x\in\{0,1\}^k\colon \varphi(x)} |x\rangle\,,$$

where x ranges over $\{0,1\}^k$. Then

$$|C_\varphi \mathrm{R}_Z(\theta)\rangle \propto \sum_{x,b} |x,b\rangle + (e^{i\theta}-1)\,|E(\varphi)\rangle\,|1\rangle \tag{18}$$

where x ranges over $\{0,1\}^k$, and b ranges over $\{0,1\}$. As $|+^{k+1}\rangle \propto \sum_{x,b} |x,b\rangle$ is the uniform superposition and hence a stabilizer state, we obtain Eq. (1):

$$\chi(C_\varphi \mathrm{R}_Z(\theta)) \le 1 + \chi(|E(\varphi)\rangle).$$

To further motivate our interest in $|E(\varphi)\rangle$, we rewrite Eq. (18) to obtain

$$|E(\varphi)\rangle\,|1\rangle \propto |C_\varphi \mathrm{R}_Z(\theta)\rangle - \sum_{x,b} |x,b\rangle\,,$$

implying that $\chi(|E(\varphi)\rangle) \le 1 + \chi(C_\varphi \mathrm{R}_Z(\theta))$. Combining with Eq. (1), any upper- or lower bound on $\chi(|E(\varphi)\rangle)$ yields a corresponding bound on $\chi(C_\varphi \mathrm{R}_Z(\theta))$.

Bounds on $\chi(C_\varphi \mathrm{R}_X)$. The X gate is a fundamental gate, providing access to the bit-flip operation. The conditional $C_\varphi \mathrm{X}$ gate is commonly used as an oracle gate [16, 17, 15, 19]. A generalization of $C_\varphi \mathrm{X}$ is the R_X gate, parameterized by a real number θ, given as

$$\mathrm{R}_X(\theta) := \mathrm{H}\mathrm{R}_Z(\theta)\mathrm{H}.$$

Specifically, we have $\mathrm{X} = \mathrm{R}_X(\pi)$. We implement $C_\varphi \mathrm{R}_X(\theta)$ by means of $C_\varphi \mathrm{R}_Z(\theta)$, as illustrated in Fig. 2.

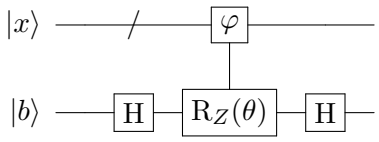

Figure 2: An implementation of a $C_\varphi \mathrm{R}_X(\theta)$ using a $C_\varphi \mathrm{R}_Z(\theta)$ gate and two H gates.

As $\mathrm{R}_Z(\theta) = \mathrm{H}\mathrm{R}_X(\theta)\mathrm{H}$, by substituting the $C_\varphi \mathrm{R}_Z$ gate in Fig. 2 for a $C_\varphi \mathrm{R}_X$ gate, we obtain an implementation of a $C_\varphi \mathrm{R}_Z$ gate by means of the $C_\varphi \mathrm{R}_X$ gate. This gives Eq. (2):

$$\chi(C_\varphi \mathrm{R}_X(\theta)) = \chi(C_\varphi \mathrm{R}_Z(\theta)).$$

Bounds on $\chi(C_\varphi U)$. It is known that an arbitrary single-qubit unitary U may be written as

$$U = e^{i\delta} \mathrm{R}_Z(\alpha)\mathrm{R}_X(\beta)\mathrm{R}_Z(\gamma)$$

for appropriate choices of real α, β, γ, and δ [36]. Hence, we can implement $C_\varphi U$ using two $C_\varphi \mathrm{R}_Z$ gates and a $C_\varphi \mathrm{R}_X$ gate as in Fig. 3 to see that

$$\begin{aligned} \chi(C_\varphi U) &\leq \chi(C_\varphi \mathrm{R}_Z)^2 \chi(C_\varphi \mathrm{R}_X) && \text{[Eq. (14)]} \\ &= \chi(C_\varphi \mathrm{R}_Z)^3 && \text{[Eq. (2)]} \\ &\leq (\chi(|E(\varphi)\rangle) + 1)^3 && \text{[Eq. (1)]} \end{aligned}$$

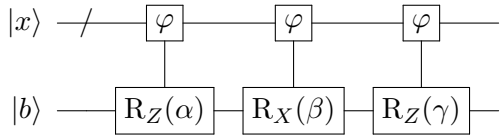

Figure 3: An implementation of a $C_\varphi U$ gate using two $C_\varphi \mathrm{R}_Z$ gates and a $C_\varphi \mathrm{R}_X$.

Alternatively, we can use a $C_\varphi \mathrm{X}$ gate and one ancillary qubit, as shown in Fig. 4.

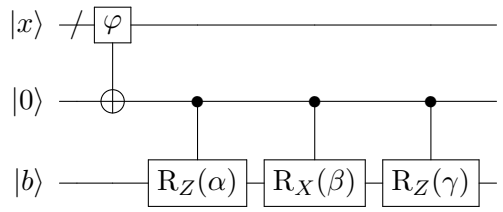

Figure 4: An implementation of a $C_\varphi U$ gate using an ancillary qubit, a single $C_\varphi \mathrm{X}$ gate, two $C\mathrm{R}_Z$ gates, and a $C\mathrm{R}_X$ gate.

Hence we also obtain

$$\begin{aligned}
\chi(C_\varphi U) &\leq \chi(C_\varphi \mathrm{R}_X)\chi(C\mathrm{R}_Z)^2\chi(C\mathrm{R}_X) && \text{[Eq. (14)]}\\
&\leq \chi(C_\varphi \mathrm{R}_Z)\chi(C\mathrm{R}_Z)^3 && \text{[Eq. (2)]}\\
&\leq (\chi(|E(\varphi)\rangle)+1)(\chi(|E(x=1)\rangle)+1)^3 && \text{[Eq. (1)]}\\
&\leq 8\chi(|E(\varphi)\rangle)+8 && \text{[Eq. (7)]}
\end{aligned}$$

Then $\chi(C_\varphi U) \leq \min\{(\chi(|E(\varphi)\rangle)+1)^3, 8\chi(|E(\varphi)\rangle)+8\}$.

By choosing the smaller expression based on $\chi(|E(\varphi)\rangle)$, we obtain Eq. (3):

$$\chi(C_\varphi U) \leq \begin{cases} 8, & \text{if } \chi(|E(\varphi)\rangle) = 1 \\ 8\chi(|E(\varphi)\rangle)+8, & \text{otherwise.} \end{cases}$$

4.2 Bounds on the Rank of Effectual States for Logical Connectives

Predicates are often inductively defined via conjunctions, disjunctions, and negations. In this section, we derive bounds that follow these inductive definitions. Specifically, we prove Eqs. (4) to (6) of Lemma 1.

Bounds on $\chi(|E(\varphi)\rangle)$ for negations. Consider the predicate $\neg\varphi$ for some $\varphi\colon \{0,1\}^k \to \{0,1\}$. We have

$$\begin{aligned}
|E(\neg\varphi)\rangle &\propto \sum_{x\in\{0,1\}^k\colon \neg\varphi(x)} |x\rangle \\
&\propto \sum_{x\in\{0,1\}^k} |x\rangle - \sum_{x\in\{0,1\}^k\colon \varphi(x)} |x\rangle \\
&\propto \sum_{x\in\{0,1\}^k} |x\rangle - |E(\varphi)\rangle\,.
\end{aligned}$$

Hence, we obtain Eq. (4):

$$\chi(|E(\neg\varphi)\rangle) \leq \chi(|E(\varphi)\rangle)+1.$$

Bounds on $\chi(|E(\varphi)\rangle)$ for conjunctions. Consider the predicates $\varphi_1\colon \{0,1\}^{k_1} \to \{0,1\}$ and $\varphi_2\colon \{0,1\}^{k_2} \to \{0,1\}$, and let us consider the predicate $\varphi(x,y) = \varphi_1(x) \wedge \varphi_2(y)$. If φ_1 and φ_2 operate over disjoint inputs, here x and y, we can write

$$\begin{aligned}
|E(\varphi)\rangle &\propto \sum_{x,y\colon \varphi_1(x)\wedge\varphi_2(y)} |x,y\rangle \\
&\propto \sum_{x\colon \varphi_1(x)} |x\rangle \sum_{y\colon \varphi_2(y)} |y\rangle \\
&\propto |E(\varphi_1)\rangle\,|E(\varphi_2)\rangle\ .
\end{aligned}$$

Here, x ranges over $\{0,1\}^{k_1}$ and y ranges over $\{0,1\}^{k_2}$. If φ_1 and φ_2 do not operate over disjoint inputs, we use entanglement and ancillary qubits to transform the circuit to one where two predicates do operate over disjoint sets of qubits, as follows.

Suppose we want to apply a gate $C_{\varphi_1(x,y)\wedge\varphi_2(y,z)}U$ for some choice of U, here with φ_1 and φ_2 overlapping on the bits y. Instead of applying the gate to the usual control registers $|x\rangle\,|y\rangle\,|z\rangle$, we extend with ancillary bits to obtain $|x\rangle\,|y\rangle\,|00\ldots0\rangle\,|z\rangle$, then apply CX gates

from the $|y\rangle$ register to the new $|00\ldots0\rangle$ register to obtain $|x\rangle\,|y\rangle\,|y\rangle\,|z\rangle$. We then apply a slightly modified gate $C_{\varphi_1(x,y_1)\wedge\varphi_2(y_2,z)}U$, where y_1 (resp. y_2) denotes the first (resp. second) register equal to $|y\rangle$. Finally, we reapply CX gates to reset the second register storing $|y\rangle$ to $|00\ldots0\rangle$, allowing it to be reused. Since we already use $O(n)$ qubits in our simulation, these ancillaries are subsumed in the big-O notation.

Fig. 5 captures an example of this transformation applied to a $C_{\varphi_1(a,b)\wedge\varphi_2(c,b)}\mathrm{R}_Z(\theta)$ gate, here overlapping on the bit b.

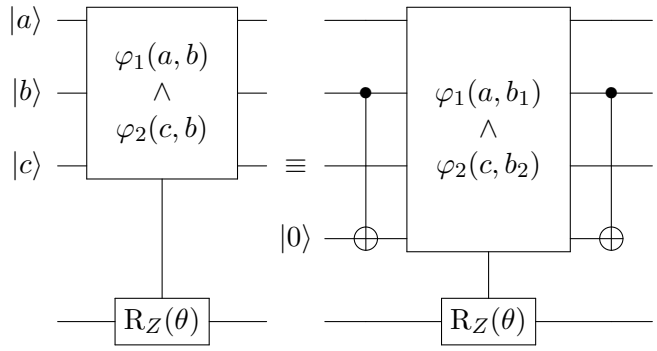

Figure 5: An example of our rewriting of conjunctions. By introducing ancillae, we split functions so that conjunctions are over separate variables.

By applying this technique as necessary, we assume without loss of generality that conjunctions of predicates always operate over disjoint qubits. This means in particular that

$$\begin{aligned}\chi(|E(\varphi_1\wedge\varphi_2)\rangle) &= \chi(|E(\varphi_1)\rangle\,|E(\varphi_2)\rangle)\\ &\leq \chi(|E(\varphi_1)\rangle)\chi(|E(\varphi_2)\rangle). \qquad \text{[Eq. (14)]}\ ,\end{aligned}$$

yielding Eq. (5):

$$\chi(|E(\varphi_1\wedge\varphi_2)\rangle) \leq \chi(|E(\varphi_1)\rangle)\chi(|E(\varphi_2)\rangle)\ .$$

Bounds on $\chi(|E(\varphi)\rangle)$ for disjunctions. We now turn to the case of $\varphi_1\vee\varphi_2$. Using our strategy as before, we assume without loss of generality that φ_1 and φ_2 operate over disjoint sets of qubits. We write

$$\begin{aligned}|E(\varphi_1\vee\varphi_2)\rangle &\propto \sum_{x,y:\varphi_1(x)\vee\varphi_2(y)} |x\rangle\,|y\rangle\\ &\propto \sum_{x,y:\ \varphi_1(x)} |x\rangle\,|y\rangle + \sum_{x,y:\varphi_2(y)} |x\rangle\,|y\rangle - \sum_{x,y:\ \varphi_1(x)\wedge\varphi_2(y)} |x\rangle\,|y\rangle\\ &\propto |E(\varphi_1)\rangle\,|+\cdots+\rangle + |+\cdots+\rangle\,|E(\varphi_2)\rangle - |E(\varphi_1\wedge\varphi_2)\rangle\,.\end{aligned}$$

Here, x ranges over $\{0,1\}^{k_1}$, while y ranges over $\{0,1\}^{k_2}$. Since $|+\rangle = \mathrm{H}\,|0\rangle$ is a stabilizer state, we have

$$\chi(|E(\varphi)\rangle\,|+\cdots+\rangle) = \chi(|+\cdots+\rangle\,|E(\varphi)\rangle) = \chi(|E(\varphi)\rangle).$$

Hence, we obtain Eq. (6):

$$\chi(|E(\varphi_1\vee\varphi_2)\rangle) \leq \chi(|E(\varphi_1)\rangle) + \chi(|E(\varphi_2)\rangle) + \chi(|E(\varphi_1\wedge\varphi_2)\rangle).$$

4.3 Bounds on the Rank of Effectual States for Predicates

In this section, we turn to establishing $\chi(|E(\varphi)\rangle)$ for various choices of predicates. Specifically, we prove Eqs. (7) to (9) of Theorem 3.

Bounds on $\chi(|E(\varphi)\rangle)$ for equality. We first consider functions performing comparison of binary integers x and y of length k. As a warm-up, let us examine the case $\varphi(x, y) = (x = y)$. We see that

$$|E(x = y)\rangle \propto \sum_{w\in\{0,1\}^k} |w\rangle\,|w\rangle\,.$$

This state can be initialized from $|0^k\rangle\,|0^k\rangle$ by applying H to each of the first k qubits, then applying CX with control bit j and target bit $j + k$ for $1 \leq j \leq k$. Hence $|E(x = y)\rangle$ is a stabilizer state, and we have Eq. (7):

$$\chi(|E(x = y)\rangle) = 1.$$

A special case of this result is when y is a fixed basis state, which enables checking if x equals a constant.

Bounds on $\chi(|E(\varphi)\rangle)$ for string comparison. Let us now examine the function $\varphi(x, y) = (x > y)$, where x and y are interpreted as binary representations of natural numbers. By definition, $x > y$ if there is a string w s.t. we have $x = w1x'$ and $y = w0y'$, where x' and y' have length based on w. Since the length of w ranges from 0 to $k - 1$, we have

$$\begin{aligned}|E(x > y)\rangle &\propto \sum_{x,y\in\{0,1\}^k\colon x>y} |x, y\rangle \\ &\propto \sum_{\ell=0}^{k-1} \sum_{w\in\{0,1\}^\ell} |w1 + \cdots +\rangle\,|w0 + \cdots +\rangle\,.\end{aligned}$$

It is easy to see that

$$\sum_{w\in\{0,1\}^\ell} |w1 + \cdots +\rangle\,|w0 + \cdots +\rangle$$

is a stabilizer state: we can initialize it from $|0^k\rangle\,|0^k\rangle$ by applying H to the first ℓ bits, then applying CX with control j and target $j + k$ for $1 \leq j \leq \ell$. Finally, applying X to the bit equaling $|1\rangle$, and H to each bit equaling $|+\rangle$, we obtain the desired state. Hence we write $|E(x > y)\rangle$ as a sum of k stabilizer states, giving Eq. (8):

$$\chi(|E(x > y)\rangle) \leq k.$$

Bounds on $\chi(|E(\varphi)\rangle)$ for incrementing. We now turn to $\varphi(x, y) = (y = x + 1)$. Here, the strings x and y are understood as binary numbers, and $+$ is binary addition modulo 2^k. By definition, we must either have some w of length ℓ s.t. $x = w01^{k-\ell-1}$ and $y = w10^{k-\ell-1}$, or we have exactly $x = 1\ldots1$ and $y = 0\ldots0$. As before, we have

$$\begin{aligned}|E(y = x + 1)\rangle &\propto \sum_{x,y\in\{0,1\}^k\colon y=x+1} |x, y\rangle \\ &\propto \sum_{\ell=0}^{k-1} \sum_{w\in\{0,1\}^\ell} |w01\ldots1\rangle\,|w10\ldots0\rangle \\ &\quad + |1\ldots1\rangle\,|0\ldots0\rangle\,.\end{aligned}$$

We can initialize $\sum_{w\in\{0,1\}^\ell} |w01\dots1\rangle\,|w10\dots0\rangle$ from $|0^k\rangle\,|0^k\rangle$ by applying H to the first ℓ bits, then applying CX with control j and target $j+k$ for $1 \le j \le \ell$. Finally, applying X to the bits equaling $|1\rangle$, we obtain the desired state. Hence it is a stabilizer state, and we obtain Eq. (9):

$$\chi(|E(y = x+1)\rangle) \le k+1.$$

4.4 The Stabilizer Rank of Query Gates

Query gates are a standard technique for providing quantum access to a Boolean mapping $g\colon \{0,1\}^\ell \to \{0,1\}^\ell$ [18, 19]. Given g, we define the query gate U_g to implement the transformation

$$|x\rangle\,|0^\ell\rangle \xrightarrow{U_g} |x\rangle\,|g(x)\rangle\,.$$

The gates captured by Theorem 2 only modify one of the qubits to which they are applied, whereas U_g modifies ℓ qubits. To implement U_g, we use post-selecting measurements to affect multiple qubits. Through this, we prove equations Eqs. (10) to (12) of Theorem 4. Our construction implementing U_g is captured in Fig. 6:

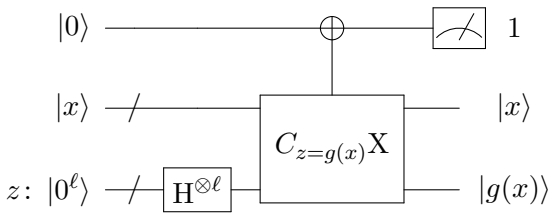

Figure 6: A circuit implementing U_g by applying post-selection.

The circuit performs the following transformation:

$$\begin{aligned}
|0\rangle\,|x\rangle\,|0^\ell\rangle &\xrightarrow{\mathrm{H}^{\otimes\ell}} \sum_{z\in\{0,1\}^\ell} |0\rangle\,|x\rangle\,|z\rangle \\
&\xrightarrow{C_{z=g(x)}\mathrm{X}} \sum_{z\in\{0,1\}^\ell} |1_{z=g(x)}\rangle\,|x\rangle\,|z\rangle \\
&\xrightarrow{Meas.} \sum_{z\in\{0,1\}^\ell\colon z=g(x)} |1\rangle\,|x\rangle\,|z\rangle \\
&= |1\rangle\,|x\rangle\,|g(x)\rangle
\end{aligned}$$

The qubit storing $|1\rangle$ can be set to $|0\rangle$ by applying an X gate, and can then be recycled. In particular, we suffer no asymptotic cost in ancillae. Doing so, we obtain the state $|x\rangle\,|g(x)\rangle$. Using this construction, we derive Eq. (10):

$$\chi(U_g) \le \chi(C_{z=g(x)}\mathrm{X}) \le \chi(|E(y = g(x))\rangle) + 1.$$

Since by Eq. (9) of Theorem 3, we have $\chi(|E(y = x+1)\rangle) \le \ell+1$, so we have the means to implement INC_ℓ:

$$|x\rangle\,|0^\ell\rangle \xrightarrow{\mathrm{INC}_\ell} |x\rangle\,|(x+1) \mod 2^\ell\rangle\,.$$

Specifically, we obtain

$$\chi(\mathrm{INC}_\ell) \leq \chi(C_{y=x+1}\mathrm{X}) \leq \ell + 2.$$

As a consequence, a circuit containing a constant number of INC_ℓ gates can be simulated in polynomial time. Given the linear bound on incrementing, it is natural to ask if this technique can be extended to also efficiently implement other arithmetic operations, such as addition and multiplication. In Section 5, we answer this in the negative, assuming common complexity theoretical hypotheses.

One natural question is whether this technique can be extended to general g. From Eq. (10), we simply need to establish $\chi(|E(y = g(x))\rangle)$. By examining

$$|E(y = g(x))\rangle \propto \sum_{x\in\{0,1\}^\ell} |x\rangle\,|g(x)\rangle\,,$$

we see that $|E(y = g(x))\rangle$ is a sum of 2^ℓ stabilizer states of the form $|x\rangle\,|g(x)\rangle$. This gives the upper bound of Eq. (12):

$$\chi(|E(y = g(x))\rangle) \leq 2^\ell.$$

Another natural question is whether we can apply controls to these gates, i.e. whether we can construct gates such as $C_\varphi U_g$. Specifically, we have

$$C_\varphi U_g\,|x\rangle\,|0^\ell\rangle = \begin{cases} |x\rangle\,|g(x)\rangle\,, & \text{if } \varphi(x) \\ |x\rangle\,|0^\ell\rangle\,, & \text{otherwise} \end{cases}$$

By defining a new mapping $h\colon \{0,1\}^\ell \to \{0,1\}^\ell$ as

$$h(x) := \begin{cases} g(x), & \text{if } \varphi(x) \\ 0^\ell, & \text{otherwise} \end{cases}$$

we see that $C_\varphi U_g = U_h$. Hence, we straightforwardly apply Eq. (12) to obtain a bound of $\chi(C_\varphi U_g) = \chi(U_h) \leq 2^\ell + 1$. However, the construction also enables us to derive a more refined upper bound.

By Eq. (10), we have $\chi(U_h) \leq \chi(|E(y = h(x))\rangle) + 1$, while $y = h(x)$ is equivalent to $(\varphi(x) \wedge y = g(x)) \vee (\neg\varphi(x) \wedge y = 0^\ell)$. Hence

$$\chi(|E(y = h(x))\rangle) = \chi(|E((\varphi(x) \wedge y = g(x)) \vee (\neg\varphi(x) \wedge y = 0^\ell))\rangle).$$

Towards deriving a bound on this value, we see that

$$\begin{aligned} \chi(|E(\neg\varphi(x) \wedge y = 0^\ell)\rangle) &\leq \chi(|E(\neg\varphi(x))\rangle)\chi(|E(y = 0^\ell)\rangle) && \text{[Eq. (5)]} \\ &= \chi(|E(\neg\varphi(x))\rangle) && \text{[Eq. (7)]} \\ &\leq \chi(|E(\varphi(x)\rangle) + 1. && \text{[Eq. (4)]} \end{aligned}$$

Furthermore, note that

$$(\varphi(x) \wedge y = g(x)) \wedge (\neg\varphi(x) \wedge y = 0^\ell) = \bot.$$

We have

$$\begin{aligned}
\chi(|E(y = h(x))\rangle) &= \chi(|E((\varphi(x) \wedge y = g(x)) \vee (\neg\varphi(x) \wedge y = 0^\ell))\rangle) \\
&\leq \chi(|E(\varphi(x) \wedge y = g(x))\rangle) + \chi(|E(\neg\varphi(x) \wedge y = 0^\ell)\rangle) \\
&\quad + \chi(|E((\varphi(x) \wedge y = g(x)) \wedge (\neg\varphi(x) \wedge y = 0^\ell))\rangle) && \text{[Eq. (6)]} \\
&\leq \chi(|E(\varphi(x) \wedge y = g(x))\rangle) + \chi(|E(\varphi(x))\rangle) + 1 + \chi(|E(\bot)\rangle) \\
&= \chi(|E(\varphi(x))\rangle)\chi(|E(y = g(x))\rangle) + \chi(|E(\varphi(x))\rangle) + 1 && \text{[Eq. (5)]} \\
&= \chi(|E(\varphi(x))\rangle)(\chi(|E(y = g(x))\rangle) + 1) + 1
\end{aligned}$$

Hence we obtain Eq. (11):

$$\chi(C_\varphi U_g) \leq \chi(|E(\varphi)\rangle)(\chi(|E(y = g(x))\rangle) + 1) + 2 \ .$$

This concludes our derivation of Theorem 4.

5 Lower-bounds on the Stabilizer Rank

We now turn our attention to other high-level gates, considering ADD_ℓ, MUL_ℓ, and QFT_ℓ, which implement ℓ-bit addition, multiplication, and Quantum Fourier Transform, respectively. We show Theorem 5, which states that the ranks of these gates cannot be polynomial assuming $\mathbf{P} \neq \mathbf{NP}$, while under the Exponential Time Hypothesis (ETH) [32], the exponent has to be linear in ℓ (e.g., it cannot be of the form $2^{O(\sqrt{\ell})}$).

Our proofs follow a sequence of reductions from 3-SAT, the famous **NP**-complete problem that is defined as follows. Given a propositional formula

$$\varphi(x_1, \ldots, x_k) := (a_1^1 \vee a_2^1 \vee a_3^1) \wedge \cdots \wedge (a_1^\ell \vee a_2^\ell \vee a_3^\ell)$$

where $a_j^i \in \{x_p, \bar{x}_p\}_{p=1,\ldots,k}$, do there exist $x_1, \ldots, x_k \in \{0, 1\}$ s.t. $\varphi(x_1, \ldots, x_k) = 1$? We say that φ has k variables and ℓ clauses. The Exponential Time Hypothesis (ETH) implies that for any algorithm solving 3-SAT, there are 3-SAT instances over k variables which require $2^{\Omega(k)}$ time. As each x_j must occur at least once, we have $k \leq \ell/3 = O(\ell)$, meaning 3-SAT has complexity $2^{\Omega(\ell)}$ under ETH.

Our sequence of reductions proceeds as follows. First, we show that a particular high-level oracle gate, which directly computes φ, has stabilizer rank $2^{\Omega(\ell)}$ under ETH. Second, we implement this oracle gate using two gates that perform ℓ parallel 3-bit logical OR operations. Third, we implement the parallel OR gate using a symmetrically defined parallel AND gate on ℓ qubits. Fourth, we implement the parallel AND gate using an ADD_ℓ gate. We then the parallel AND gate using a MUL_ℓ gate. Finally, we implement an ADD_ℓ gate using three QFT_ℓ gates.

Oracle gate. Given a 3-SAT formula φ over k variables and ℓ clauses, we can compute whether it is satisfiable by simulating a single $C_\varphi\mathrm{X}$ oracle gate on a superposition of all assignments to the input variables of φ, as shown in Fig. 7.

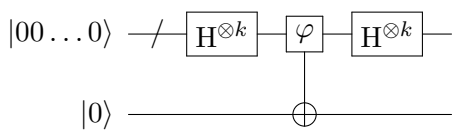

Figure 7: A circuit computing whether φ is satisfiable.

The output state is $|00\ldots0\rangle|0\rangle$ iff φ is unsatisfiable, which can be decided by simulating the circuit with target state $|00\ldots0\rangle|0\rangle$. The circuit has one non-stabilizer gate as well as $2k = O(\ell)$ stabilizer gates and operates over $k+1 = O(\ell)$ qubits. Thus, Theorem 1 implies that the running time for its simulation is bounded by

$$O(\chi O(\ell)^2(O(\ell)+O(\ell)1)) = O(\chi(C_\varphi \mathrm{X})\ell^3)\ .$$

This implies that $\chi(C_\varphi \mathrm{X})$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(C_\varphi \mathrm{X}) = 2^{\Omega(\ell)}$ under ETH.

OR_ℓ gates. We now proceed with implementing the oracle gate $C_\varphi \mathrm{X}$ using two OR_ℓ gates. These compute ℓ disjunctions over three variables each, in a parallel manner:

$$|x\rangle|0^\ell\rangle \xrightarrow{\mathrm{OR}_\ell} |x\rangle|x_1 \vee x_2 \vee x_3\rangle \ldots |x_{3\ell-2} \vee x_{3\ell-1} \vee x_{3\ell}\rangle\ .$$

The following construction implies a bound on $\chi(\mathrm{OR}_\ell)$ in terms of $\chi(C_\varphi \mathrm{X})$. Recall that a^i_j refers to the i-th literal of the j-th clause of φ. First we introduce the INI gadget that performs the following transformation:

$$|x\rangle|0^{3\ell}\rangle \xrightarrow{\mathrm{INI}} |x\rangle|a^1_1 a^1_2 a^1_3\rangle \ldots |a^\ell_1 a^\ell_2 a^\ell_3\rangle\ .$$

This construction can be implemented using CX and X gates. Fig. 8 shows an example for the transformation $|ab\rangle|0^3\rangle \xrightarrow{\mathrm{INI}} |ab\rangle|a\bar{a}b\rangle$.

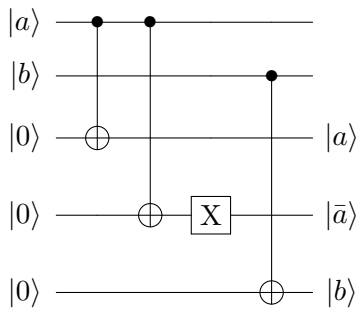

Figure 8: An INI gadget performing the transformation $|ab\rangle|0^3\rangle \xrightarrow{\mathrm{INI}} |ab\rangle|a\bar{a}b\rangle$.

We implement the $C_\varphi \mathrm{X}$ gate via the circuit of Fig. 9, which uses the INI gadget twice.

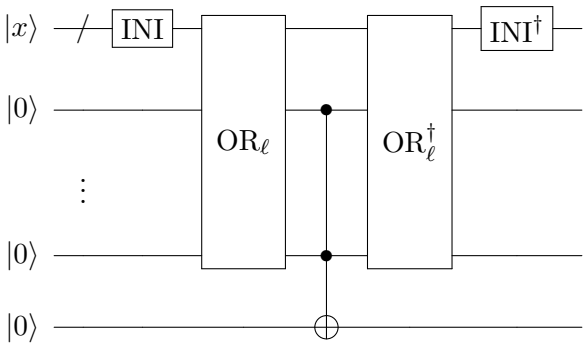

Figure 9: A circuit implementing a $C_\varphi \mathrm{X}$ gate using two OR_ℓ gates and a $C^\ell \mathrm{X}$ gate.

As the construction implements a $C_\varphi \mathrm{X}$ gate, its magic state rank is an upper bound on the magic state rank of $C_\varphi \mathrm{X}$, i.e.,

$$\chi(C_\varphi X) \leq \chi(C^\ell \mathrm{X})\chi(\mathrm{OR}_\ell)^2 \leq 2\chi(\mathrm{OR}_\ell)^2 \,, \tag{19}$$

where $\chi(C^\ell \mathrm{X}) \leq 2$ follows from Eqs. (1) and (2) of Theorem 2 and Eq. (7) of Theorem 3. We thus obtain that $\chi(\mathrm{OR}_\ell)$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(\mathrm{OR}_\ell) = 2^{\Omega(\ell)}$ under ETH.

AND_ℓ gates. As a second intermediate step, we show that an OR_ℓ gate can be implemented using an AND_ℓ gate, defined in the following manner:

$$|x\rangle\,|0^\ell\rangle \xrightarrow{\mathrm{AND}_\ell} |x\rangle\,|x_1 \wedge x_2 \wedge x_3\rangle \ldots |x_{3\ell-2} \wedge x_{3\ell-1} \wedge x_{3\ell}\rangle$$

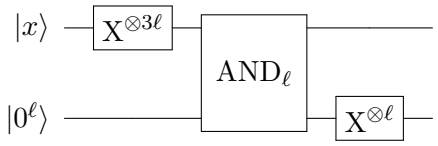

Figure 10: A circuit implementing an OR_ℓ gate using an AND_ℓ gate and 4ℓ X gates.

The construction, captured in Fig. 10, is a straightforward application of De Morgan's law, and shows that $\chi(\mathrm{OR}_\ell) \leq \chi(\mathrm{AND}_\ell)$. As a sidenote, if the AND_ℓ gate in Fig. 10 is substituted for an OR_ℓ gate, the construction instead yields an implementation of an AND_ℓ gate. This shows the stronger statement that $\chi(\mathrm{OR}_\ell) = \chi(\mathrm{AND}_\ell)$. Combined with Eq. (19), we obtain that $\chi(\mathrm{AND}_\ell)$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(\mathrm{AND}_\ell) = 2^{\Omega(\ell)}$ under ETH.

ADD_ℓ gates. We are now ready to derive a bound on the rank of the ADD_ℓ gate, which implements addition over ℓ-bit integers:

$$|u\rangle\,|v\rangle\,|0^\ell\rangle \xrightarrow{\mathrm{ADD}_\ell} |u\rangle\,|v\rangle\,|(u+v) \mod 2^\ell\rangle\,.$$

We show that an AND_ℓ gate can be implemented using an $\mathrm{ADD}_{3\ell}$ gate. We do this by arranging the input bit string x of AND_ℓ into two bit strings u and v, padded with zeros, which will serve as the input to the $\mathrm{ADD}_{3\ell}$ gate. In particular,

$$\begin{array}{lccccccccc} u := 0 & x_1 & x_2 & 0 & x_4 & x_5 & \ldots & 0 & x_{\ell-2} & x_{\ell-1} \\ v := 0 & 0 & x_3 & 0 & 0 & x_6 & \ldots & 0 & 0 & x_\ell \,. \end{array}$$

Let $z = u + v$, and $z_1 z_2 \ldots z_\ell$ be the bits of z, with z_1 being the most significant bit. Then we have $z_1 = x_1 \wedge x_2 \wedge x_3$, and generally, $z_{3j+1} = x_{3j+1} \wedge x_{3j+2} \wedge x_{3j+3}$. Thus the bit string $z_1 z_4 \ldots z_{3\ell-2}$ encodes the output of the AND_ℓ gate. This implies that $\chi(\mathrm{AND}_\ell) \leq \chi(\mathrm{ADD}_{3\ell})$, implying that $\chi(\mathrm{ADD}_\ell)$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(\mathrm{ADD}_\ell) = 2^{\Omega(\ell)}$ under ETH.

MUL_ℓ gates. We now derive a bound on the rank of the MUL_ℓ gate, which implements multiplication over ℓ-bit integers:

$$|u\rangle\,|v\rangle\,|0^\ell\rangle \xrightarrow{\mathrm{MUL}_\ell} |u\rangle\,|v\rangle\,|(uv) \mod 2^\ell\rangle\,.$$

We show that an AND_ℓ gate can be implemented via a $\mathrm{MUL}_{7\ell}$ gate, by a process similar to the one above. We arrange the input bit string x of AND_ℓ in a specific pattern padded with zeros, which will serve as the input to the $\mathrm{MUL}_{7\ell}$ gate. In particular,

$$u := 000x_1 0x_2 x_3 \qquad 000x_4 0x_5 x_6 \qquad \dots \qquad 000x_{\ell-2} 0x_{\ell-1} x_\ell$$

and $v := 00\dots01001$. Then the gate computes $uv = 9u = u + 8u$. Writing out these numbers, we see the pattern

$$\begin{array}{lccccccccccc} u = & & & 0 & 0 & 0 & x_1 & 0 & x_2 & x_3 & \dots \\ 8u = 0 & 0 & 0 & x_1 & 0 & x_2 & x_3 & 0 & 0 & 0 & \dots \end{array}$$

Then define $z = uv$, and let $z_1 z_2 \dots z_\ell$ be the bits of z. Then we have $z_5 = x_1 \wedge x_2 \wedge x_3$, while generally, $z_{7j+5} = x_{3j+1} \wedge x_{3j+2} \wedge x_{3j+3}$. Thus the bit-string $z_5 z_{12} \dots z_{7\ell-2}$ encodes the output of the AND_ℓ gate. This implies that $\chi(\mathrm{AND}_\ell) \leq \chi(\mathrm{MUL}_{7\ell})$, implying that $\chi(\mathrm{MUL}_\ell)$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(\mathrm{MUL}_\ell) = 2^{\Omega(\ell)}$ under ETH.

QFT_ℓ gates. We now derive a bound on the rank of the QFT_ℓ gate, which implements the Quantum Fourier Transform over ℓ-bit quantum states:

$$|x\rangle \xrightarrow{\mathrm{QFT}_\ell} \sum_{z \in \{0,1\}^\ell} e^{2i\pi x z 2^{-\ell}} |z\rangle \ .$$

We implement an ADD_ℓ gate via the circuit in Fig. 11, which uses three QFT_ℓ gates combined with ℓ post-selecting measurements.

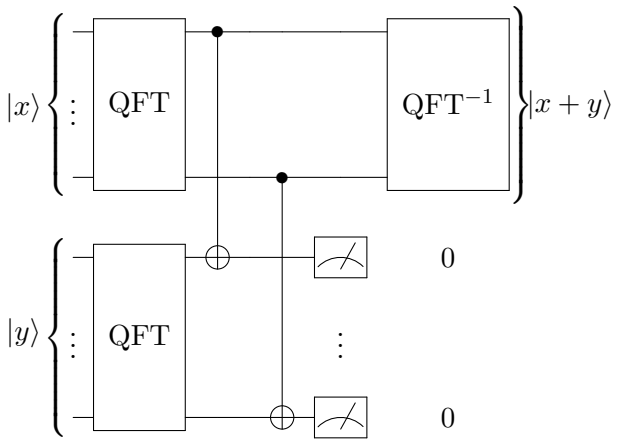

Figure 11: A circuit implementing binary addition using post-selection and three QFT gates.

The circuit performs the following transformation:

$$
\begin{aligned}
|x\rangle\,|y\rangle &\xrightarrow{\mathrm{QFT}_\ell^{\otimes 2}} \sum_{z\in\{0,1\}^\ell} e^{2i\pi xz2^{-\ell}}\,|z\rangle \sum_{w\in\{0,1\}^\ell} e^{2i\pi yw2^{-\ell}}\,|w\rangle \\
&\propto \sum_{z,w\in\{0,1\}^\ell} e^{2i\pi xz2^{-\ell}} e^{2i\pi yw2^{-\ell}}\,|z\rangle\,|w\rangle \\
&\propto \sum_{z,w\in\{0,1\}^\ell} e^{2i\pi (xz+yw)2^{-\ell}}\,|z\rangle\,|w\rangle \\
&\xrightarrow{\mathrm{CX}^{\otimes \ell}} \sum_{z,w\in\{0,1\}^\ell} e^{2i\pi (xz+y(w\oplus z))2^{-\ell}}\,|z\rangle\,|w\rangle \\
&\xrightarrow{Meas.} \sum_{z\in\{0,1\}^\ell} e^{2i\pi (xz+yz)2^{-\ell}}\,|w\rangle\,|0^\ell\rangle \\
&\propto \sum_{z\in\{0,1\}^\ell} e^{2i\pi (x+y)z2^{-\ell}}\,|w\rangle\,|0^\ell\rangle \\
&\xrightarrow{\mathrm{QFT}_\ell^{-1}} |x+y\rangle\,|0^\ell\rangle
\end{aligned}
$$

This yields $\chi(\mathrm{ADD}_\ell) \leq \chi(\mathrm{QFT}_\ell)^3$, implying that $\chi(\mathrm{QFT}_\ell)$ is not polynomial under $\mathbf{P} \neq \mathbf{NP}$, and $\chi(\mathrm{QFT}_\ell) = 2^{\Omega(\ell)}$ under ETH.

This concludes our derivation of Theorem 5.

6 Implementation and Experimentation

We implemented the simulator behind Theorem 1 (both the single-threaded and parallelizability results) as a prototype in Python, by extending the phase-sensitive stabilizer simulator of Reardon-Smith [37]. In particular, given a high-level circuit C and a basis state x, our simulator performs the following steps.

1. Gadgetize C to C' using (μ, k)-decompositions of each non-stabilizer gate, following the process described in Section 3 that re-uses the ancilla qubits that store the individual magic state of each gadget.
2. For each stabilizer state in the stabilizer decomposition, run the phase-sensitive simulator of [37] and obtain the coefficient on state x. The individual simulations can run in parallel, for a user-specified number of threads.
3. Maintain a running sum of results that is used to output the final measurement probability.

The rest of this section summarizes an experimental evaluation of this simulation approach of high-level circuits, and compares it against the baseline approach of first compiling to a low-level circuit and then using the standard Qiskit Aer simulator on the low-level circuit.

Benchmarks. We consider benchmarks from four categories: (i) the state preparation algorithm CVO-QRAM [11]; (ii) compositions of minimal oracle functions [38]; (iii) Grover's algorithm [15]; and (iv) a string comparison oracle. All these benchmarks contain high-level gates that operate on a variable number of qubits.

Low-level circuit simulators. For simulating the compiled, low-level circuits, we use the (weak) simulators implemented as part of the Qiskit Aer Python library [33]. These

simulators are parameterized by a number of samples to generate from the output distribution. We set this number to be 1. Although a single measurement reveals very little information about the measurement probability, we make this choice to ensure that the speedup of our simulator comes from its direct handling of the high-level gates, and not from the fact that it does not have to sample the output distribution.

We use 5 simulation baselines implemented in Aer: matrix product state [39, 40] (MProdS), state vector (SVec), density matrix (DMatr), and extended stabilizer [8] (ExtStab). The matrix product state method uses tensors to represent systems of qubits and generally performs well when the amount of entanglement, roughly corresponding to the number of CX gates, is low. The state vector method represents a pure state of an n-qubit system explicitly as a 2^n-dimensional vector. This is efficient for systems of few qubits. The density matrix method stores a mixed state over an n-qubit system by representing it as a $2^n \times 2^n$ dimensional matrix. This is also efficient for systems of few qubits. Finally, the extended stabilizer method uses magic state injection as our method but only does so for T gates. It works well when the number of these non-stabilizer gates is small.

Experimental setup. The experiments are run on an M3 MacBook using 10 GB of RAM and a time-out of 1 hour. In all our experiments, we compute the probability of measuring $x = 0^n$. Since not all baselines support multithreading, we disable it in our simulator for fair comparison.

6.1 Experiment: CVO-QRAM

State preparation is an important primitive in quantum software [41], with active research on designing efficient state preparation circuits [11, 42, 43]. As benchmarks, we generate circuits using the CVO-QRAM algorithm [11]. This algorithm initializes an n-qubit state of the form $\sum_{j=1}^{k} \psi_j |x_j\rangle$ for basis states $|x_j\rangle$ using k C^nU gates, $O(kn)$ CX gates, and one ancilla qubit. The structure of the circuits is captured in Fig. 12. To obtain the low-level circuits that Aer can handle, we used the Qiskit compiler [33] to obtain Clifford+T equivalents. To use our tool, we combined Eq. (3) of Theorem 2 and Eq. (7) of Theorem 3 in order to simulate the C^nU gates.

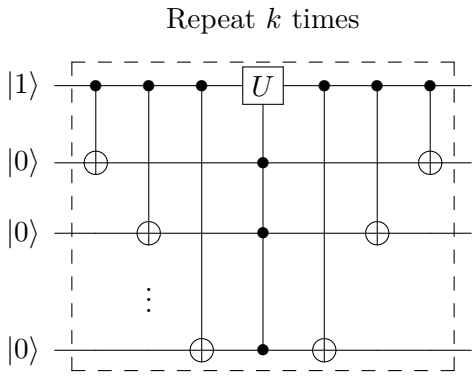

Figure 12: A circuit describing the structure of the CVO-QRAM experiment. In the j-th repetition, the repeated section applies a CX gate from the first qubit to any qubit which equals 1 in $|x_j\rangle$, then applies a C^nU gate, and uncomputes the CX gates.

Results. A summary of results is shown in Table 1, for varying numbers of qubits n

Benchmark		High-level (direct)	Low-level (transpiled)			
n	k	Our simulator	MProdS	SVec	DMatr	ExtStab
5	1	2ms	2ms	1ms	2ms	3.358s
5	2	2ms	1ms	1ms	1ms	1m15s
5	3	379ms	6ms	2ms	5ms	OOM
5	4	450ms	6ms	6ms	7ms	OOM
10	1	3ms	7ms	4ms	4.227s	OOM
10	2	41ms	6ms	5ms	3.154s	OOM
10	3	588ms	16ms	13ms	8.534s	OOM
10	4	7.603s	41ms	32ms	24.16s	OOM
20	1	6ms	53ms	1.778s	OOM	OOM
20	2	101ms	123ms	5.929s	OOM	OOM
20	3	1.179s	150ms	6.138s	OOM	OOM
20	4	12.89s	255ms	13.03s	OOM	OOM
50	1	11ms	305ms	OOM	OOM	OOM
50	2	211ms	811ms	OOM	OOM	OOM
50	3	2.689s	1.265s	OOM	OOM	OOM
50	4	26.88s	1.602s	OOM	OOM	OOM
500	1	172ms	2m29s	OOM	OOM	OOM
500	2	2.859s	5m29s	OOM	OOM	OOM
500	3	37.02s	8m13s	OOM	OOM	OOM
500	4	6m55s	11m9s	OOM	OOM	OOM
1000	1	537ms	24m35s	OOM	OOM	OOM
1000	2	7.702s	44m42s	OOM	OOM	OOM
1000	3	1m42s	TO	OOM	OOM	OOM
1000	4	25m12s	TO	OOM	OOM	OOM

Table 1: Experimental results for the CVO-QRAM experiment. An entry highlighted in green indicates the fastest run-time for that benchmark, while an entry in red indicates either out-of-memory (OOM) or time-out (TO).

and C^nU gates k. We observe that our direct simulator is generally faster, and typically by several orders of magnitude. On the other hand, SVec and MProdS perform efficient simulations on the smaller circuits, but either time out or go out of memory on circuits with many qubits. DMatr manages to handle circuits of few qubits, but is considerably slower than the above methods, and quickly runs out of memory on larger circuits. ExtStab quickly runs out of memory, handling only the smallest benchmarks.

6.2 Experiment: Oracle functions

In quantum computing, an oracle is a Boolean predicate $\varphi\colon \{0,1\}^k \to \{0,1\}$, access to which is usually provided via a C_φX gate. This is a very common construction, used in Grover's algorithm [15], the Deutsch-Josza algorithm [16], the Bernstein-Vazirani algorithm [17], and Simon's algorithm [19], among others.

There are 48 spectral equivalent classes over Boolean predicates of the form $\{0,1\}^5 \to \{0,1\}$ [38]. These 48 can be composed to give other predicates, and hence their minimal representations as stabilizer+T circuits have been studied [44]. For each of these classes, the optimal stabilizer+T circuit implementing a representative is provided in [45]. For our experimental setup, we take k representatives at random and chain them together as

Benchmark	High-level (direct)	Low-level (transpiled)			
Oracles k	Direct	MProdS	SVec	DMatr	ExtStab
1	5ms	1ms	1ms	1ms	1.678s
2	16ms	6ms	4ms	915ms	OOM
3	68ms	24ms	67ms	OOM	OOM
4	340ms	4.387s	678ms	OOM	OOM
5	3.331s	3.406s	1m8s	OOM	OOM
6	22.04s	1m49s	OOM	OOM	OOM
7	54.90s	TO	OOM	OOM	OOM
8	2m30s	TO	OOM	OOM	OOM
9	13m31s	TO	OOM	OOM	OOM
10	TO	TO	OOM	OOM	OOM

Table 2: Experimental results for the chained oracle experiment. An entry highlighted in green indicates the fastest run-time for that benchmark, while an entry in red indicates either out-of-memory (OOM) or time-out (TO).

illustrated in Fig. 13. To obtain the low-level circuits, we composed the minimal implementations. To run our simulator, we instead directly implemented the oracle gates using Eq. (12) of Theorem 4 combined with Eqs. (1) and (2) of Theorem 2.

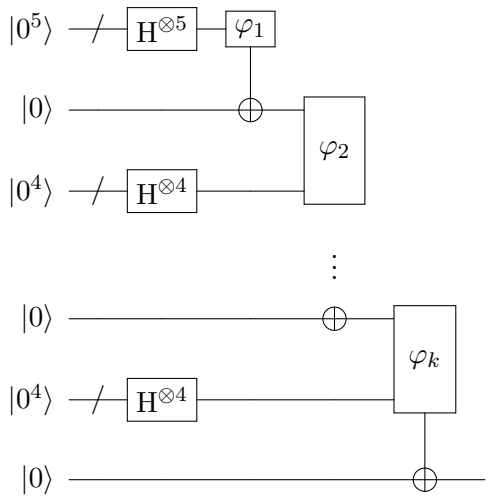

Figure 13: A circuit describing the composed oracle benchmark.

Results. A summary of results is shown in Table 2 for a varying number of chained oracles k. For non-trivially sized instances, we see that our direct simulator is faster. MProdS runs out of time on instances of more than 6 oracles. While SVec and DMatr perform well on small instances, they quickly run out of memory. ExtStab runs out of memory even on instances of more than one oracle.

6.3 Experiment: Grover's algorithm

Grover's algorithm [15] is a famous quantum algorithm for finding a satisfying assignment to a propositional formula φ, offering quadratic speedups over classical approaches. Fig. 14

Benchmark	High-level (direct)	Low-level (transpiled)			
Size n	Direct	MProdS	SVec	DMatr	ExtStab
5	2.072s	17ms	2ms	7ms	OOM
10	3.960s	221ms	2ms	26.68s	OOM
15	6.163s	1.407s	7ms	OOM	OOM
20	7.979s	5.350s	147ms	OOM	OOM
25	9.939s	8.545s	3.586s	OOM	OOM
30	12.01s	14.18s	OOM	OOM	OOM
40	15.92s	57.45s	OOM	OOM	OOM
45	17.98s	1m40s	OOM	OOM	OOM
50	19.92s	2m17s	OOM	OOM	OOM
75	30.57s	13m50s	OOM	OOM	OOM
100	41.30s	TO	OOM	OOM	OOM
200	1m28s	TO	OOM	OOM	OOM
500	4m53s	TO	OOM	OOM	OOM
1000	13m23s	TO	OOM	OOM	OOM

Table 3: Experimental results for simulation of Grover's algorithm with an oracle given a conjunction of negated variables. An entry highlighted in green indicates the fastest run-time for that benchmark, while an entry in red indicates either out-of-memory (OOM) or time-out (TO).

describes the layout of the circuit implementing the algorithm.

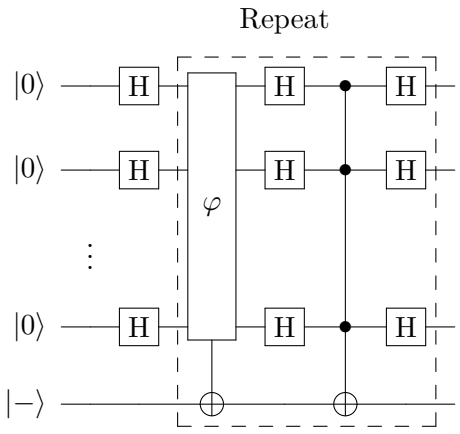

Figure 14: A circuit describing the structure of the Grover experiment.

We perform two experiments based on Grover's algorithm, using different choices of φ. For the first experiment, we choose $\varphi(x) = \bar{x_1} \wedge \cdots \wedge \bar{x_n}$, then generate and run instances for different choices of n. To obtain low-level circuits, we use the standard Qiskit compiler [33], while to run ours, we use Eqs. (2) and (18) combined with Eq. (7). For the second experiment, we use $\varphi = \bigwedge_{j=1}^{3} \bigvee_{\ell=1}^{n} a_{j,\ell}$, where each $a_{j,\ell} \in \{x_\ell, \bar{x}_\ell\}$ is a literal chosen uniformly at random. To obtain low-level circuits, we straightforwardly implement φ using three C^kX gates, some X gates, and a C^3X gate, then compile using the Qiskit compiler [33]. To run our simulator, we use Eqs. (5) and (6) to implement oracles defined from predicates in conjunctive normal form.

Benchmark	High-level (direct)	Low-level (transpiled)			
Size n	Direct	MProdS	SVec	DMatr	ExtStab
5	9.222s	14ms	1ms	624ms	OOM
10	15.30s	184ms	4ms	39m26s	OOM
15	23.29s	2.030s	48ms	OOM	OOM
20	30.01s	8.158s	1.271s	OOM	OOM
25	37.50s	21.08s	1m19s	OOM	OOM
30	46.02s	52.67s	OOM	OOM	OOM
40	58.89s	3m37s	OOM	OOM	OOM
45	1m5s	5m44s	OOM	OOM	OOM
50	1m11s	12m36s	OOM	OOM	OOM
75	1m45s	TO	OOM	OOM	OOM
100	2m25s	TO	OOM	OOM	OOM
200	5m19s	TO	OOM	OOM	OOM
500	16m49s	TO	OOM	OOM	OOM
1000	47m25s	TO	OOM	OOM	OOM

Table 4: Experimental results for simulation of Grover's algorithm with an oracle given as a function in conjunctive normal form (CNF). An entry highlighted in green indicates the fastest run-time for that benchmark, while an entry in red indicates either out-of-memory (OOM) or time-out (TO).

Results. Our results for the first experiment are captured in Table 3. For small instances, SVec is faster, but for all instances of 30 qubits or more, our method is more efficient, while from 100 qubits and beyond, our method is the only one that finishes within the bound on memory and time. Our results for the second experiment are captured in Table 4. For small instances, SVec is faster, while from instances of 30 qubits and beyond, our method is the fastest. From 75 qubits and beyond, our method is the only one that finishes within the bound on memory and time.

6.4 Experiment: String Comparison Oracle

Our final experiment involves simulating an oracle gate for the Boolean predicate $x > y$, defined for k-bit integers x and y. The oracle is provided as a $C_{x>y}\mathrm{X}$ gate. For different choices of k, we simulate comparing a uniform superposition over all choices of x and y, as described in Fig. 15.

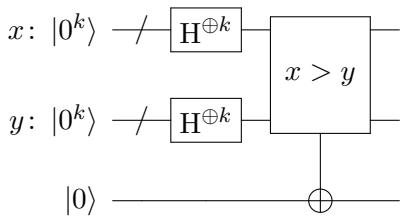

Figure 15: A circuit describing the structure of the $C_{x>y}\mathrm{X}$ benchmarks.

To obtain low-level circuits, we use the implementation provided in [46] to compile to Clifford+C^kX circuits, then compile to Clifford+T using the Qiskit compiler [33]. Their implementation uses k ancillary bits to store intermediate computations. Hence in Table 5,

Benchmark		High-level (direct)		Low-level (transpiled)			
k	n	Direct	As C^kX	MProdS	SVec	DMatr	ExtStab
1	4	0ms	0ms	1ms	0ms	1ms	107ms
2	7	0ms	9ms	1ms	0ms	1ms	314ms
3	10	1ms	117ms	3ms	0ms	238ms	4m47s
4	13	2ms	1.313s	8ms	1ms	32.13s	OOM
5	16	2ms	13.78s	22ms	3ms	OOM	OOM
6	19	3ms	2m22s	81ms	17ms	OOM	OOM
7	22	4ms	22m56s	646ms	116ms	OOM	OOM
8	25	5ms	TO	3m47s	985ms	OOM	OOM
9	28	6ms	TO	2m55s	8.514s	OOM	OOM
10	31	9ms	TO	37m52s	OOM	OOM	OOM
11	34	9ms	TO	TO	OOM	OOM	OOM
12	37	19ms	TO	TO	OOM	OOM	OOM
13	40	17ms	TO	TO	OOM	OOM	OOM
14	43	17ms	TO	TO	OOM	OOM	OOM
15	46	18ms	TO	TO	OOM	OOM	OOM

Table 5: Experimental results for simulating a $C_{x>y}$X gate. An entry highlighted in green indicates the fastest run-time for that experiment, while an entry in red indicates either out-of-memory (OOM) or time-out (TO).

we have $n = 3k + 1$, as we need $2k + 1$ qubits to store x, y, and the target qubit. To run our simulator, we use Eqs. (2) and (18) combined with Eq. (8) to simulate directly, while the results in the "As C^kX" column are produced by instead running the Clifford+C^kX circuits.

Results. See Table 5 for results. Between our results, we see that the direct simulation outperforms simulating Clifford+C^kX circuits on all benchmark sizes. Comparing against Qiskit Aer, SVec is generally faster for small instances, while on instances of 16 qubits and beyond, our direct simulation results are faster. Simulating the Clifford+C^kX circuits, while consistently outperforming DMatr and ExtStab, is itself consistently outperformed by MProdS and SVec, suggesting increased benefits when working with higher gate-levels. On instances of 34 qubits or more, our direct method is the only one which does not go out of time or memory.

7 Related Work

The ZX calculus can be applied to perform strong simulation of universal quantum circuits. The work of [47] converts Clifford+T and Clifford+CCZ circuits to ZX diagram representations, then recursively apply diagram simplifications and magic state decompositions to simulate these efficiently. These decompositions depend on the rank of the magic state $|\mathrm{T}\rangle^{\otimes t}$, which is shown to be $\chi(|\mathrm{T}\rangle^{\otimes t}) \leq 2^{0.396t}$, improving the earlier result of [10] which showed that this upper bound holds when t is sufficiently large. However, dealing with arbitrary high-level gates, this technique still requires a compilation step, typically introducing extra non-Clifford gates. In contrast, our technique sidesteps this overhead by providing magic state decompositions of high-level gates directly.

The work of [24] expands on the technique of [47] by using so-called "triangle nodes" in the ZX diagrams. This permits an efficient decomposition of C^kX gates, resulting in a 2^t

exponential factor for t C^kX gates. While our work also achieves a 2^t exponential factor for t C^kX gates, we generalize to arbitrary oracle gates beyond a single conjunction.

The work of [48] further extends the idea by considering a number of decompositions which apply to substructures of the ZX diagram and permit efficient decomposition when they are present. Heuristics determine which decomposition to apply, for which experiments seem to suggest practical improvements over a greedy strategy. The work of [49] studies partitioning the ZX diagram into smaller diagrams which can be individually reduced, and from which the simulation result can be derived. The efficacy each technique depends on the structure of the underlying graph, for which our technique is oblivious. Future work could explore a heuristic interleaving of partitions and decompositions to improve efficiency.

The recent work of [50] gives formal bounds on the run-time of simulation based on the "rank-width" R of the ZX diagram, giving a run-time of $\tilde{O}(4^R)$, where $\tilde{O}$ hides sub-exponential factors. The recent work of [51] unifies approaches from stabilizer decompositions and tensor network contraction to give run-times based on the structure of the underlying graph. For a stabilizer+T circuit C with rank-width $\texttt{rw}(C)$ and tree-width $\texttt{tw}(C)$, they give algorithms with respective run-times of $\tilde{O}(t^{\gamma\cdot\texttt{rw}(C)})$ and $\tilde{O}(t^{\texttt{tw}(C)})$, where t is the number of T gates of C, and $\gamma \approx 3.42$. Future work could generalize their work to our approach for high-level gates, by exploring the graph structure of C to improve efficiency.

Finally, the work of [52] proposes a technique for improving the simulation of the Clifford fragments of a general circuit. In high level, this approach recompiles the input circuits to permit efficient simulation via a sum-over-Cliffords technique. It is possible to apply a similar recompilation strategy to our work: Given a circuit C, write it as $C_tG_tC_{t-1}G_{t-1}\dots C_1G_1C_0$ for Clifford circuits C_j and non-Clifford gates G_j, then apply standard techniques to canonicalize the Clifford circuits to $O(n^2)$ gates in $O(nm + n^3)$ time. This brings the circuit size to $O(n^2t)$, resulting in a run-time of $O(\chi n^2(n^2t + t\mu) + nm + n^3) = O(\chi n^4 t + nm)$, where we use the fact that μ is the size of a Clifford circuit and hence bounded as $O(n^2)$. This strategy outperforms the baseline algorithm of Theorem 1 when $m = \omega(n^2t)$.

8 Conclusion

We have presented a gadget-based simulator which gadgetizes, then simulates high-level gates directly using magic state injection, thereby avoiding the blowup of compilation. We have also shown that a number of high-level gates that are often used in quantum computing enjoy a small stabilizer rank (in particular, independent of the number of qubits they act on), and can therefore be simulated more efficiently using our gadget-based simulator. Our experiments with various high-level gates indeed show reduced simulation time for several types of high-level circuits compared to Qiskit Aer that simulates via compilation. Exploring the limits of our technique, we have shown that the stabilizer rank of certain high-level gates is hyper-polynomial under $\mathbf{P} \neq \mathbf{NP}$, and exponential under ETH.

For each of the high-level gates which we have considered here, reducing the upper bounds on the rank of the corresponding magic states reduces the overall simulation complexity for circuits containing that gate. To complement such results, corresponding lower-bounds on the magic state rank help guide the search for efficient simulation, by avoiding searching for improvements where they are unlikely to be found. Furthermore, extending the high-level

gate simulation theory established here to other gates, more circuits might be simulated efficiently. With a tool for direct strong simulation of high-level gates, circuit equivalence can be decided without suffering the blow-up of compilation. If applied efficiently, this could extend the reach of formal quantum circuit verification.

9 Acknowledgments

This work was partially supported by a research grant (VIL42117) from VILLUM FONDEN.

References

[1] Leonidas Lampropoulos, Zoe Paraskevopoulou, and Benjamin C. Pierce. "Generating good generators for inductive relations". Proc. ACM Program. Lang. **2** (2017).

[2] Jiyuan Wang, Qian Zhang, Guoqing Harry Xu, and Miryung Kim. "QDiff: Differential Testing of Quantum Software Stacks". In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). Pages 692–704. (2021).

[3] Tom Peham, Nina Brandl, Richard Kueng, Robert Wille, and Lukas Burgholzer. "Depth-Optimal Synthesis of Clifford Circuits with SAT Solvers". In 2023 IEEE International Conference on Quantum Computing and Engineering (QCE). Volume 1, pages 802–813. IEEE (2023).

[4] Sarah Schneider, Lukas Burgholzer, and Robert Wille. "A SAT Encoding for Optimal Clifford Circuit Synthesis". In Proceedings of the 28th Asia and South Pacific Design Automation Conference. ASPDAC '23. ACM (2023).

[5] Richard P Feynman. "Simulating Physics with Computers". International Journal of Theoretical Physics **21**, 467–488 (1982).

[6] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". Physical Review A **70** (2004).

[7] Sergey Bravyi, Graeme Smith, and John A. Smolin. "Trading Classical and Quantum Computational Resources". Physical Review X **6** (2016).

[8] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. "Simulation of quantum circuits by low-rank stabilizer decompositions". Quantum **3**, 181 (2019).

[9] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. "Methodology for quantum logic gate construction". Physical Review A **62** (2000).

[10] Hammam Qassim, Hakop Pashayan, and David Gosset. "Improved upper bounds on the stabilizer rank of magic states". Quantum **5**, 606 (2021).

[11] Tiago M. L. de Veras, Leon D. da Silva, and Adenilton J. da Silva. "Double sparse quantum state preparation". Quantum Information Processing **21** (2022).

[12] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". Physical Review Letters **103** (2009).

[13] Mingchao Guo, Hailing Liu, Yongmei Li, Wenmin Li, Fei Gao, Sujuan Qin, and Qiaoyan Wen. "Quantum algorithms for anomaly detection using amplitude estimation". Physica A: Statistical Mechanics and its Applications **604**, 127936 (2022).

[14] Jing Li, Fei Gao, Song Lin, Mingchao Guo, Yongmei Li, Hailing Liu, Sujuan Qin, and QiaoYan Wen. "Quantum k-fold cross-validation for nearest neighbor classification algorithm". Physica A: Statistical Mechanics and its Applications **611**, 128435 (2023).

[15] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Page 212–219. STOC '96 New York, NY, USA (1996). Association for Computing Machinery.

[16] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**, 553–558 (1992).

[17] Ethan Bernstein and Umesh Vazirani. "Quantum Complexity Theory". In Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing. Page 11–20. STOC '93 New York, NY, USA (1993). Association for Computing Machinery.

[18] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". SIAM Journal on Computing **26**, 1484–1509 (1997).

[19] Daniel R. Simon. "On the Power of Quantum Computation". SIAM Journal on Computing **26**, 1474–1483 (1997).

[20] Ben Zindorf and Sougato Bose. "Efficient Implementation of Multi-Controlled Quantum Gates". Phys. Rev. Appl. **24**, 044030 (2025).

[21] 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". Trans. Comp.-Aided Des. Integ. Cir. Sys. **43**, 802–811 (2024).

[22] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing. Page 193–204. STOC 2019 New York, NY, USA (2019). Association for Computing Machinery.

[23] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. "Grand Unification of Quantum Algorithms". PRX Quantum **2**, 040203 (2021).

[24] Mark Koch, Richie Yeung, and Quanlong Wang. "Contraction of ZX diagrams with triangles via stabiliser decompositions". Physica Scripta **99**, 105122 (2024).

[25] Maxime Remaud. "Optimizing T and CNOT Gates in Quantum Ripple-Carry Adders and Comparators". In Proceedings of Recent Advances in Quantum Computing and Technology. Pages 56–61. Association for Computing Machinery (2024).

[26] Mingyoung Jeng, Alvir Nobel, Vinayak Jha, David Levy, Dylan Kneidel, Manu Chaudhary, Ishraq Islam, Muhammad Momin Rahman, and Esam El-Araby. "Generalized Quantum Convolution for Multidimensional Data". Entropy **25** (2023).

[27] Di Fang, Lin Lin, and Yu Tong. "Time-marching based quantum solvers for time-dependent linear differential equations". Quantum **7**, 955 (2023).

[28] Brendan L Douglas and JB Wang. "Efficient quantum circuit implementation of quantum walks". Physical Review A—Atomic, Molecular, and Optical Physics **79**, 052335 (2009).

[29] Craig Gidney. "Constructing Large Increment Gates" (2015). Accessed: 2025-03-27 at https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html.

[30] Man-Duen Choi. "Completely positive linear maps on complex matrices". Linear Algebra and its Applications **10**, 285–290 (1975).

[31] A. Jamiołkowski. "Linear transformations which preserve trace and positive semidefiniteness of operators". Reports on Mathematical Physics **3**, 275–278 (1972).

[32] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". Journal of Computer and System Sciences **62**, 367–375 (2001).

[33] Qiskit Development Team. "Qiskit Aer Documentation". (2024). url: https://qiskit.github.io/qiskit-aer/.

[34] Daniel Gottesman. "Class of quantum error-correcting codes saturating the quantum hamming bound". Phys. Rev. A **54**, 1862–1868 (1996).

[35] Christopher M. Dawson and Michael A. Nielsen. "The Solovay-Kitaev algorithm". Quantum Info. Comput. **6**, 81–95 (2006).

[36] Michael A. Nielsen and Isaac L. Chuang. "Quantum Computation and Quantum Information". Cambridge University Press. (2009). 10th Anniversary edition.

[37] Oliver Reardon-Smith. "PSCS: Phase-sensitive Clifford simulator". (2020). url: https://github.com/or1426/pscs.

[38] Giulia Meuli, Mathias Soeken, and Giovanni De Micheli. "SAT-based {CNOT, T} Quantum Circuit Synthesis". In Reversible Computation: 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings 10. Pages 175–188. Springer (2018).

[39] Guifré Vidal. "Efficient Classical Simulation of Slightly Entangled Quantum Computations". Physical Review Letters **91** (2003).

[40] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Annals of Physics **326**, 96–192 (2011).

[41] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum State Preparation with Optimal Circuit Depth: Implementations and Applications". Phys. Rev. Lett. **129**, 230504 (2022).

[42] Daniel K. Park, Francesco Petruccione, and June-Koo Kevin Rhee. "Circuit-Based Quantum Random Access Memory for Classical Data". Scientific Reports **9** (2019).

[43] Tiago M. L. de Veras, Ismael C. S. de Araujo, Daniel K. Park, and Adenilton J. da Silva. "Circuit-Based Quantum Random Access Memory for Classical Data With Continuous Amplitudes". IEEE Transactions on Computers **70**, 2125–2135 (2021).

[44] Giulia Meuli, Mathias Soeken, Martin Roetteler, and Giovanni De Micheli. "Enumerating Optimal Quantum Circuits using Spectral Classification". In 2020 IEEE International Symposium on Circuits and Systems (ISCAS). Pages 1–5. IEEE (2020).

[45] Mathias Soeken Giulia Meuli. "The Single-Target Gate (STG) benchmark suite". (2020). url: https://github.com/gmeuli/stg-benchmark.

[46] Shan-zhi Li. "Design of Quantum Comparator Based on Extended General Toffoli Gates with Multiple Targets". Computer Science **39**, 302–306 (2012). url: https://api.semanticscholar.org/CorpusID:63956655.

[47] Aleks Kissinger, John van de Wetering, and Renaud Vilmart. "Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions". In François Le Gall and Tomoyuki Morimae, editors, 17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022). Volume 232 of Leibniz International Proceedings in Informatics (LIPIcs), pages 5:1–5:13. Dagstuhl, Germany (2022). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[48] Wira Azmoon Ahmad and Matthew Sutcliffe. "Dynamic T-decomposition for classical simulation of quantum circuits". arXiv:2412.17182 (2024).

[49] Matthew Sutcliffe. "Smarter k-Partitioning of ZX-Diagrams for Improved Quantum Circuit Simulation". arXiv:2409.00828 (2024).

[50] Fedor Kuyanov and Aleks Kissinger. "Efficient Classical Simulation of Low-Rank-Width Quantum Circuits Using ZX-Calculus". arXiv:2603.06764 (2026).

[51] Julien Codsi and Tuomas Laakkonen. "Unifying Graph Measures and Stabilizer Decompositions for the Classical Simulation of Quantum Circuits". arXiv:2603.06377 (2026).

[52] Hammam Qassim, Joel J Wallman, and Joseph Emerson. "Clifford recompilation for faster classical simulation of quantum circuits". Quantum **3**, 170 (2019).