State preparation by shallow circuits using feed forward

Harry Buhrman[1], Marten Folkertsma[1], Bruno Loff[2], and Niels M. P. Neumann[1,3]

[1]QuSoft, CWI & University of Amsterdam, Amsterdam, the Netherlands
[2]LASIGE & Department of Mathematics, University of Lisbon
[3]The Netherlands Organisation for Applied Scientific Research (TNO), Delft, the Netherlands

Abstract

In order to achieve fault-tolerant quantum computation, we need to repeat the following sequence of four steps after we have initialized the quantum device. First, we perform 1 or 2 qubit quantum gates (in parallel if possible). Second, we do a syndrome measurement on a subset of the qubits. Third, we perform a fast classical computation to establish which errors have occurred (if any). And, fourth, depending on the errors, we apply a correction step. Then the procedure repeats with the next sequence of gates. These four steps are essential to accomplish fault-tolerant quantum computing.

In order for these four steps to succeed, we need the error rate of the gates to be below a certain threshold. Unfortunately, the error rates of current quantum hardware are still too high and do not meet this requirement. On the other hand, current quantum hardware platforms are designed with these four steps in mind. In this work we make use of this four-step scheme not to carry out fault-tolerant computations, but to enhance short, *constant*-depth, quantum circuits that perform 1 qubit gates and *nearest-neighbor* 2 qubit gates. To explore how this can be useful, we study a computational model which we call *Local Alternating Quantum Classical Computations* (LAQCC). In this model, qubits are placed in a grid and they can only interact with their direct neighbors; the quantum circuits are of constant depth with intermediate measurements; a classical controller can perform log-depth computations on these intermediate measurement outcomes and control future quantum operations based on the outcome. This model fits naturally between quantum algorithms in the NISQ era and full fledged fault-tolerant quantum computation.

We show how an LAQCC circuit can create long-ranged interactions, which constant-depth quantum circuits cannot achieve, and use it to construct a range of useful multi-qubit operations. With these gates, we create three new state preparation protocols for a uniform superposition over an arbitrary number of states, W-states and Dicke states, the generalization of W-states. Furthermore, we show that this type of model contains circuits which are unlikely to be classically simulatable, as well as bound the power of this model by showing an inclusion into QNC^1

1 Introduction

Current quantum hardware is unable to carry out universal quantum computations due to the buildup of errors that occur during the computation. The magnitude of the individual error is currently above the value that the Threshold Theorem requires in order to kick-start quantum error correction and quantum fault-tolerant computation [34, Section 10.6]. Although the experimentally achieved fidelity rates are promising and the error bounds are inching closer to the required threshold, we will have to work for the foreseeable future with quantum hardware with errors that build-up during the computation. This implies that we can only do a limited number of steps before the output of the computation has become completely uncorrelated with the intended one.

For fault-tolerant quantum computing, we repeat four steps: 1) We apply a number of single and two-qubit quantum gates, in parallel whenever possible; 2) We perform a syndrome measurement on a subset of the qubits; 3) We perform fast classical computations to determine which errors have occurred and how to correct them; and, 4) We apply correction terms based on the classical computations. We then repeat these four steps with a next sequence of gates. These four steps are essential to fault-tolerant quantum computing.

The starting point of this work is to use the four steps outlined above, not to carry out error correction and fault-tolerant computation, but to enhance short, constant-depth, *uncorrected* quantum circuits that perform single qubit gates and *nearest-neighbor* two qubit gates. Since in the long run we will have to implement error-correction and fault-tolerant computation anyhow, and this is done by such a four-step process, why not make other use of this architecture? Moreover, on some of the quantum hardware platforms, these operations are already in place. Embracing this idea we naturally arrive at the question: what is the computational power of *low-depth* quantum-classical circuits organized as in the four steps outlined above? We thus investigate circuits that execute a small, ideally constant, number of stages, where at each stage we may apply, in parallel, single qubit gates and *nearest-neighbor* two qubit gates, followed by measurements, followed by low-depth classical computations of which the outcome can control quantum gates in later stages. It is not clear, at first, whether such circuits, especially with constant depth, can do anything remotely useful. But we will see that this is indeed the case: many quantum computations can be done by such circuits in constant depth. By parallelizing quantum computations in this way, we improve the overall computational capabilities of these circuits, as we do not incur errors on qubits that are idle, simply because qubits are not idle for a very long time. Furthermore, reducing the depth of quantum circuits, at the cost of increasing width, allows the circuit to be run faster even if errors occur.

The first usage of such a four-step layout, not to do error correction, but to perform computations, can be found in the paradigm of measurement-based quantum computing [13, 19, 26, 42]: A universal form of quantum computing where a quantum state is prepared and operations are performed by measuring qubits in different bases, depending on previous measurements and intermediate measurements.

Pham and Svore were the first to formalize the four-step protocol for performing computations [38]. They included specific hardware topologies by considering two-dimensional graphs for imposing constraints on qubit interactions. In their model, they develop circuits for particularly useful multi-qubit gates, including specifying costs in the width, number of qubits, depth, number of concurrent time steps, size, and total number of non-Identity operations. As a result, they find an algorithm that factors integers in polylogarithmic depth.

More recently, Piroli, Styliaris, and Cirac introduced a scheme to implement unitary operations involving quantum circuits combined with Local Operations and Classical Communication (LOCC) channels: LOCC-assisted quantum circuits [39]. Similarly to the four-step scheme we just described, they allow for a short depth circuit to be run on the qubits, followed by one round of LOCC, in which ancilla qubits are measured and local unitaries are applied based on the measurement outcomes. They show that in this model any 1D transitionally invariant matrix-product state (MPS) with fixed bond dimension is in the same phase of matter as the trivial state. Similar ideas can be found in [48, 47]

In this work, we introduce a new model, called *Local Alternating Quantum-Classical Computations* (LAQCC). In this model we alternate between running quantum circuits (constrained by locality), ending in the measurement of a subset of qubits, and fast classical computations based on the measurement results. The outcome of the classical computations are then used to control future quantum circuits. We allow for flexibility in this model, by giving different constraints to the power of both the quantum circuits and the classical circuits as well as the number of alternations between them. Most attention will be given to LAQCC containing quantum circuits of constant depth, classical circuits of logarithmic depth and at most a constant number of alternations between them. Any circuit constructed in this model is considered to be of constant depth.

The definition of LAQCC sharpens the original definition of Pham and Svore by adding con-

straints to the intermediate classical computations. This allows us to bound the power of LAQCC from above. The LOCC-assisted circuits of Piroli, Styliaris, and Cirac are not low-depth, as they allow for long sequential measurement-based correction of the ancilla qubits, and this is required for their calculations. These measurement-based operations are considered as sequential alternations between the quantum and classical circuits in LAQCC, resulting in increasing the total depth.

We study the power of the LAQCC model with respect to state preparation, showing that even with only constant quantum-depth and logarithmic classical depth it remains possible to prepare states with long-range entanglement. Another surprising result is that it is unlikely that LAQCC circuits are classically simulatable. We show that any instantaneous quantum polynomial-time (IQP) circuit [11, 44] has an LAQCC implementation. Classical simulation of IQP circuits implies the collapse of the polynomial hierarchy to the third level, which is not believed to be true [12]. Therefore, we expect that LAQCC circuits are unlikely to be classically simulatable. We bound the power of LAQCC by showing that it is contained in QNC_1, the class of polynomial-size, log-depth circuits.

The main part of our paper concerns new efficient state-preparation circuits for three types of states. All states corresponding to these three types are non-stabilizer states. Efficient circuits for preparing stabilizer states have been known through measurement-based quantum computing, we discuss this result in more detail in Section 3.2. The first state is a uniform superposition over an arbitrary number of states. This circuit includes the use of Grover search as a subroutine, that turns a probabilistic circuit, with a known constant probability of success, into a deterministic circuit. We use the circuit for preparing a uniform superposition over an arbitrary number of states as a subroutine in the next two quantum state preparation protocols.

The second state is the W-state, the uniform superposition over all computational basis states of Hamming-weight 1, a natural long-ranged entangled state that displays a fundamentally nonequivalent type of entanglement from the Greenberger–Horne–Zeilinger state [16], for which LAQCC-type constant-depth circuits were previously known [38, 39]. The W-state is often used as benchmark for new quantum hardware [17, 23, 33]. A circuit for preparing the W-state was given in [39], but this implementation requires sequentially alternating measurements followed by local unitaries, which in the LAQCC model is not considered to be of constant depth. We improve this protocol by showing a LAQCC implementation of the W-state, at the cost of using extra ancilla qubits.

The third state considered is the Dicke state, a generalization of the W-state, a superposition over all computational basis states with Hamming-weight k [15]. Dicke states have relevance in various practical settings. For instance, for quantum game theory [37], quantum storage [4, 40], quantum error correction [36], quantum metrology [50], and quantum networking [41]. Dicke states have been used as a starting state for variational optimization algorithms, most notably Quantum Alternating Operator Ansatz (QAOA) [22], to find solutions to problems such as Maximum k-vertex Cover [10, 14]. The ground states of physical Hamiltonians describing one-dimensional chains tend to show a resemblance to Dicke states such as states resulting from the Bethe ansatz, making them an ideal starting state when investigating the ground state behavior of these Hamiltonians [8, 9, 51]. For instance, the algorithm by van Dyke et. al., who give an algorithm to prepare the Bethe ansatz eigenstates of the spin-1/2 XXZ spin chain, starts by first preparing a Dicke state [52]. Efficient deterministic circuit for preparing Dicke states have been proposed by Bärtschi and Eidenbenz [5, 6]. They provide a quantum circuit of depth $\mathcal{O}(k\log(\frac{n}{k}))$, allowing arbitrary connectivity, to prepare a Dicke state, which they conjecture to be optimal when k is constant. In this work, we provide a constant depth LAQCC circuits below their conjectured bound already for constant k. However, this does not directly disprove their conjecture, as we allow for intermediate measurements and classical computations. More significantly, we even construct constant-depth LAQCC circuits for $k = \mathcal{O}(\sqrt{n})$ greatly improving their bound. Finally, we give a log-depth circuit for every value of k.

We conclude by studying the power that intermediate classical calculations can add to quantum computations. In this study, we define a new model that goes beyond constant-depth quantum circuits and log-depth classical calculations, to polynomial-depth quantum circuits and unbounded classical power called LAQCC*. We study this model by doing a complexity theoretical analysis,

where we borrow inspiration from the notions of complexity given by Rosenthal and Yuen, Metger and Yuen, and Aaronson. All three complexity notions are based on the notion of state preparation, instead of more traditional definition of complexity such as the decidability of a computational problem. The first two consider classes based on sequences of quantum states preparable by a polynomial-sized quantum circuit, where the circuits are uniformly generated by a computational class, for instance, the class PSPACE, which results in the complexity class StatePSPACE [30, 43]. The third notion considers a relative complexity, where the complexity is measured between two given states, and is measured by the number of gates, from a given gate-set, required to transform one state in another state [1]. For our definition of state preparation complexity, we drop the uniformity constraint from [30, 43] and define a class as StateX, which refers to states preparable by circuits of type X. As an example, if $\mathsf{X} = \mathsf{QNC}^0$, this results in the class $\mathsf{StateQNC}^0$, which is the set of states preparable from the $|0\rangle^n$ state by poly-size constant-depth circuits. This notion is similar to the relative complexity from [1], where one state is the $|0\rangle^n$ state and instead of counting the number of gates we consider the set of states preparable by a fixed number of gates. Using this notion of complexity we show that any state preparable by an LAQCC^* circuit is also preparable by a PostQPoly circuit, which is the class of circuits of polynomial depth with post-selection.

Summary of results

- We give a new definition of a computational model that captures the power of the four step process: applying a constant number of layers of one- and two-qubit gates; performing a syndrome measurement; perform a fast classical calculation determining corrections; apply corrections. We call this model *Local Alternating Quantum Classical Computations*, or LAQCC for short. In this model we bound the allowed quantum operations, intermediate classical calculations, and number of rounds separately. In Section 3 we define this model and give a list of operations based on results from literature contained in this computational model. In some of these operations we explicitly use that we allow for multiple, but at most constant, rounds of corrections.

- We show show that there exist LAQCC circuits that can not be weakly simulated in Section 3.4. We further show that for every LAQCC circuit there exists a QNC^1 circuit simulating it perfectly, in Section 3.5.

- We show a protocol to prepare the uniform superposition state of size q in LAQCC using $\mathcal{O}(\log(\lceil q \rceil)^2)$ qubits in Section 4.1.

- We show a protocol to prepare the W_n state in LAQCC using $\mathcal{O}(n \log(n))$ qubits in Section 4.2.

- We show two ways of preparing the Dicke-(n, k) state. The first method is in LAQCC, works up to $k = \mathcal{O}(\sqrt{n})$, and uses $\mathcal{O}(n^2 \log(n)$ qubits, and is found in Section 4.3. The second method is in $\mathsf{LAQCC}-\mathsf{LOG}$ (an extension of LAQCC allowing for logarithmic number of alterations instead of constant), works for any k, and uses $\mathcal{O}(poly(n))$ qubits, and is found in Section 4.4.

- We introduce a new type computational complexity for preparing states and show that the extension of LAQCC where we allow a polynomial number of rounds and unbounded classical computation, is contained in PostQPoly, the class of polynomial circuits with post-selection, in Section 5.

Organization of the paper We first introduce relevant preliminaries in Section 2. In Section 3 we formally define the class of Local Alternating Quantum-Classical Computations (LAQCC). We also show that any Clifford circuit can be implemented in constant depth LAQCC (a result based on a result from measurement-based quantum computing [26]). This result allows us to give many useful multi-qubit gates and routines in Section 3.3. We conclude the section by showing that constant depth LAQCC circuits are contained in QNC^1 and that any IQP circuit has an LAQCC

implementation. In Section 4 we give LAQCC circuit implementations for preparing the uniform superposition over an arbitrary number of states, the W-state and the Dicke state up to $k = \mathcal{O}(\sqrt{n})$. We furthermore give a log-depth circuit implementation for preparing the Dicke state for any k. We conclude in Section 5 with an analysis of a more powerful instantiation of LAQCC and show an inclusion with respect to the class PostQPoly, which is the class of circuits of polynomial depth with post-selection.

2 Preliminaries

In this section we recap definitions used throughout the rest of the paper.

2.1 Complexity classes

The computational model introduced in this work uses complexity classes. Typically these classes are defined as classes of decision problems solvable by some type of circuits. Below we give definitions of some of these complexity classes in terms of the circuits contained in that class.

Definition 2.1. The class NC^k consists of all decision problems solvable by circuits of $\mathcal{O}((\log n)^k)$ depth and polynomial size, and consisting of bounded-fan-in AND- and OR-gates.

The class AC^k consists of all decision problems solvable by circuits of $\mathcal{O}((\log n)^k)$ depth and polynomial size, and consisting of unbounded-fan-in AND- and OR-gates.

The class TC^k consists of all decision problems solvable by circuits of $\mathcal{O}((\log n)^k)$ depth and polynomial size, and consisting of unbounded-fan-in AND-, OR- and Threshold$_t$-gates. A Threshold$_t$-gate evaluates to one if and only if the sum of the inputs is at least t.

These classes also have a quantum equivalent class.

Definition 2.2. The class QNC^k consists of all decision problems solvable by quantum circuits of $\mathcal{O}((\log n)^k)$ depth and polynomial size, and consisting of single- and two-qubit quantum gates.

Definitions for the quantum versions of AC^k and TC^k also exist. However, when equipping the class QNC^k with unbounded-fan-in parity gates, all three classes intersect [20, 31, 46].

Two other often used classes are P and L.

Definition 2.3. The class P consists of all decision problems solvable in polynomial time by a Turing machine.

The class L consists of all decision problems solvable using only a logarithmic amount of memory.

The first class poses a limit on the depth of the operations performed by the Turing machine. The second class instead limits the available memory. Note however that with a logarithmic amount of memory, only a polynomial number of states is available, and hence the computation time is polynomial as well.

Relations between different complexity classes exists: for instance, for all k, $\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{TC}^k$. By Johnson, we also have the inclusion $\mathsf{L} \subseteq \mathsf{AC}^1 \subseteq \mathsf{TC}^1$ [25].

In the remainder of this work we abuse notation and refer to X-circuits as circuits that correspond to a decision problem in the class X. For example, an NC^k circuit is a circuit of $\mathcal{O}((\log n)^k)$ depth and polynomial size that corresponds to a decision problem in NC^k.

Finally, we define the class of polynomial sized quantum circuits: [1]

Definition 2.4. The class $\mathsf{QPoly}(n)$ consists of all polynomial-sized quantum circuits (in n) that use single and two-qubit quantum gates.

[1] Due to the bounded-error-aspect associated to decision problems in BQP, we follow this definition instead of talking about BQP-circuits.

2.2 Quantum gate sets

First, recall the definition of the Pauli group and the Clifford group.

Definition 2.5. The one qubit Pauli-group P_1 consists of four matrices, the identity matrix I and the three Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

together with a global phase of ± 1 or $\pm i$.

The n-qubit Pauli-group P_n is the set of all 4^{n+1} possible tensors of length n of matrices from P_1, together with a global phase of ± 1 or $\pm i$.

The Clifford group forms the other well-known group of quantum circuits, as it stabilizes the Pauli group.

Definition 2.6. The Clifford group C_n consists of all n-qubit unitaries that leave the Pauli group P_n invariant under conjugation. That is, let $c \in C_n$ be any Clifford circuit, then for any $P \in P_n$, there exists a $P' \in P_n$, such that $cP = P'c$.

The Clifford group is generated by the $CNOT$-gate, the Hadamard gate H and the phase gate S, that act on computational basis states $|x\rangle$ and $|y\rangle$ via

$$CNOT : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus x\rangle , \qquad H : |x\rangle \mapsto |0\rangle + (-1)^x |1\rangle , \qquad S : |x\rangle \mapsto i^x |x\rangle .$$

Any circuit constructed using only these three gates is called a Clifford circuit.

Unsurprisingly, Clifford circuits only cover a small part of the possible quantum circuits. Moreover, on a linear nearest-neighbor architecture, $\mathcal{O}(n)$ deep Clifford circuits suffice to simulate any Clifford unitary of size $2^n \times 2^n$ [29]. We can furthermore simulate Clifford circuits efficiently [18]. Universal quantum computations require additional gates, though almost any quantum gate suffices. For example, adding the single qubit T-gate, $T : |x\rangle \mapsto e^{i\pi x/4} |x\rangle$, to the Clifford group gives a universal gate set.

2.3 Two quantum subroutines

This section discusses two quantum subroutines used in later sections. The first concerns Grover's algorithm with zero failure probability [28]. The second concerns parallelization of commuting gates using quantum fan-out gates [24].

Grover's search algorithm gives a quadratic speed-up for unstructured search [21]. After sufficient iterations, a measurement returns a target state with high probability. Surprisingly, if the exact number of target states is known, a slight modification of the Grover iterates allows for returning a target state with certainty, assuming noiseless computations. Lemma 3.14 uses the next lemma to prepare quantum states instead of to find a target state.

Lemma 2.7 ([28])**.** *Let L be a set of items and $T \subseteq L$ a set of targets, with $N = |L|$ and $m = |T|$ both known. Let $g : L \to \{0, 1\}$ label the items in L and define the oracle $O_g : |x\rangle |b\rangle \mapsto |x\rangle |b \oplus g(x)\rangle$.*

Then, there exists a quantum amplitude amplification algorithm that makes $\mathcal{O}(\sqrt{N/m})$ queries to O_g and prepares the quantum state $\sum_{x \in T} |x\rangle$.

For the other result, we use the quantum fan-out gate, that implements the map

$$|x\rangle |y_1\rangle \dots |y_n\rangle \mapsto |x\rangle |y_1 \oplus x\rangle \dots |y_n \oplus x\rangle .$$

Section 3.3 gives more details on how to implement this gate. Hoyer and Spalek introduced this gate and analyzed its properties. The state preparation protocols given in Section 4 use the property that the quantum fan-out gate allows parallelization of commuting quantum gates [24].

Lemma 2.8. ([24, Theorem 3.2]) *Let $\{U_i\}_{i=1}^n$ be a pairwise commuting set of gates on k qubits. Let $U_i^{x_i}$ be the gate U_i controlled by qubit $|x_i\rangle$. Let T be the unitary that mutually diagonalizes all U_i. Then there exists a quantum circuit, using quantum fan-out gates, computing $U = \prod_{i=1}^n U_i^{x_i}$ with depth $\max_{i=1}^n \text{depth}(U_i) + 4 \cdot \text{depth}(T) + 2$ and size $\sum_{i=1}^n \text{size}(U_i) + (2n+2) \cdot \text{size}(T) + 2n$, using $(n-1)k$ ancilla qubits.*

3 The LAQCC model

This section formally defines the LAQCC model. Next, we shows that all Clifford circuits have an efficient and equivalent LAQCC circuit. We then give quantum gates and tools constructable within LAQCC, such as the quantum fan-out gate and weighted threshold gate, and we conclude by showing that any LAQCC-circuit has an equivalent QNC^1-circuit.

3.1 Model definition

We define the computational model *Local Alternating Quantum-Classical Computations* (LAQCC) as follows:

Definition 3.1 (Local Alternating Quantum-Classical computations)**.** Let $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$ be the class of circuits such that

- every quantum layer implements a quantum circuit $Q \in \mathcal{Q}$ constrained to a grid topology;
- every classical layer implements a classical circuit $C \in \mathcal{C}$;
- there are d alternating layers of quantum and classical circuits;
- after every quantum circuit Q a subset of the qubits is measured;
- the classical circuit receives input from the measurement outcomes of previous quantum layers;
- the classical circuit can control quantum operations in future layers.

The allowed gates in the quantum and classical layers are given by $\mathcal{Q}$ and $\mathcal{C}$ respectively. Furthermore, we require a circuit in $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$ to deterministically prepare a pure state on the all-zeroes initial state.

The grid topology imposed on the quantum operations implies that qubits can only interact with their direct neighbors on the grid.

Remark 3.2. *Note that there exists ambiguity in the choices for $\mathcal{Q}$ and $\mathcal{C}$. For example, we have $\mathsf{LAQCC}(\mathsf{QPoly}(n), \mathsf{P}, \mathcal{O}(1)) = \mathsf{LAQCC}(\mathsf{QNC}_0, \mathsf{P}, \mathcal{O}(\text{poly}(n)))$. This follows as any P-circuit is in $\mathsf{QPoly}(n)$, and we can concatenate $\text{poly}(n)$ constant-depth quantum circuits with trivial intermediate classical computations.*

This ambiguity is non-trivial: consider for instance

$$\mathsf{LAQCC}(\mathsf{QNC}^1, \mathsf{NC}^1, \mathcal{O}(1)) \subseteq \mathsf{LAQCC}(\mathsf{QNC}^0, \mathsf{NC}^1, \mathcal{O}(\log(n))).$$

The inclusion from left to right follows immediately by same argument as above. It is however not obvious if the logarithmic number of measurement rounds, allowed in the right hand side, can be simulated by a QNC^1 circuit. Even stronger, we will show in Section 3.3 that threshold gates are available in $\mathsf{LAQCC}(\mathsf{QNC}^0, \mathsf{NC}^1, \mathcal{O}(1))$. From this fact it follows immediately that any TC^1-circuit is contained in $\mathsf{LAQCC}(\mathsf{QNC}^0, \mathsf{NC}^1, \mathcal{O}(\log(n)))$. It is unclear these circuits are also contained in $\mathsf{LAQCC}(\mathsf{QNC}^1, \mathsf{NC}^1, \mathcal{O}(1))$.

In the remainder of this work, we consider a specific instantiation of $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$.

Notation 3.3. We let LAQCC refer to the instance $\mathsf{LAQCC}(\mathsf{QNC}^0, \mathsf{NC}^1, \mathcal{O}(1))$, together with a grid nearest-neighbor topology and a quantum gate-set of all single-qubit gates and the two-qubit CNOT gate. The classical computations are bounded to logarithmic depth and of bounded-fan-in.

In its current definition, $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$, and hence also LAQCC, are classes of circuits. When considering the capabilities of $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$ in preparing states, it is helpful to define a related class that consists of states preparable by a circuit in $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$.

Definition 3.4. Let $\mathcal{H}_n$ be a Hilbert space on n qubits, then define

$$\mathsf{StateX}_{n,\varepsilon} = \{|\psi\rangle \in \mathcal{H}_n \mid \exists\, \mathsf{X}\text{-circuit } A : \langle A|0\rangle^{\otimes n}, |\psi\rangle\rangle \geq 1 - \varepsilon\}.$$

This is the subset of n-qubit states $|\psi\rangle$ such that there exists a circuit corresponding to the class X that prepares a quantum state that has inner product at least $1 - \varepsilon$ with $|\psi\rangle$.

Define $\mathsf{StateX}_\varepsilon = \bigcup_{n\in\mathbb{N}} \mathsf{StateX}_{n,\varepsilon}$.

This definition extends already existing ideas and definitions of state-complexity [2, 43, 45]. Our definition is very similar to state complexity defined in [30], where we are interested in which states are contained in a class, however we drop the uniformity requirement and instead study the set of states that can be generated by a specific class of circuits. An example of a circuit class is $\mathsf{StateLAQCC}(\mathcal{Q}, \mathcal{C}, d)_{n,\varepsilon}$.

Notation 3.5. The class $\mathsf{StateLAQCC}(\mathcal{Q}, \mathcal{C}, d)_{n,\varepsilon}$ consists of all n-qubit states $|\psi\rangle$ for which an $\mathsf{LAQCC}(\mathcal{Q}, \mathcal{C}, d)$ exists that prepares a state that has inner product at least $1 - \varepsilon$ with $|\psi\rangle$.

Another example is the circuit class of $\mathsf{PostQPoly}$.

Definition 3.6. The class $\mathsf{PostQPoly}$ consists of all polynomial-sized quantum circuits with one extra qubit, where the outcome state is considered conditional on the extra qubit being in the one state. If the extra qubit is in the zero state, the output state may be anything.

The class $\mathsf{StatePostQPoly}_{n,\varepsilon}$ consists of all n-qubit states $|\psi\rangle$ for which a polynomial-sized quantum circuit exists that prepares a state that, conditional on the extra qubit being one, has inner product at least $1 - \varepsilon$ with $|\psi\rangle$.

The next section shows that all polynomial-size Clifford circuits are in LAQCC. As a result, the quantum fan-out gate is also in LAQCC, a multi-qubit gate which enables many different subroutines (by applying Lemma 2.8). We give a list of such quantum gates and quantum subroutines accessible LAQCC in Section 3.3. We conclude by showing that any LAQCC-circuit corresponds to a QNC^1-circuit.

3.2 Clifford circuits

The concept of intermediate measurements with subsequent computations is closely related to measurement-based quantum computing. A famous result from measurement-based quantum computing us that all Clifford circuits can be paralellized using measurements. In this section we borrow techniques from this result to show that any Clifford circuit has an LAQCC implementation.

This result is best understood in the teleportation based quantum computing model [26], a specific instance of measurement-based quantum computing that applies quantum operations using bell measurements. In teleportation, qubits are measured in the Bell basis, which projects the measured qubits onto an entangled two-qubit, or ebit, state, up to local Pauli gates. This projection combined with an ebit state teleports a quantum state between qubits. After teleportation, one needs to correct the local Pauli gate created by the bell measurement. A similar process can be used to apply quantum gates. However, the Pauli gates that arise during teleportation have to be corrected before the calculations can proceed, which necessitates subsequent adaptive operations.

With Clifford circuits, these subsequent operations can be omitted. Clifford circuits stabilize the Pauli group, which allows for simultaneous measurements and hence parallelization of the full Clifford circuit [26]. Consider a simple example of teleporting a single-qubit quantum state. A

Bell basis measurement projects two qubits on $\sum_{i\in\{0,1\}} \langle ii| P^{a,b} \otimes I$, where $P^{a,b} = Z^a X^b$ and $a, b \in \{0,1\}$ correspond to the four possible measurement outcomes.

By using one Bell-basis measurement, we can apply two sequential Clifford gates U_1 and U_2 on a quantum state $|\psi\rangle$, which gives:

$$\sum_{i,j\in\{0,1\}} \big[(\langle ii| (P^{a,b} \otimes I) \otimes I\big] U_1 \otimes I \otimes U_2 |\psi\rangle |jj\rangle = \sum_{i,j\in\{0,1\}} \langle i| P^{a,b} U_1 |\psi\rangle \langle i|j\rangle U_2 |j\rangle$$
$$= \sum_{i\in\{0,1\}} U_2 |i\rangle \langle i| P^{a,b} U_1 |\psi\rangle = U_2 P^{a,b} U_1 |\psi\rangle .$$

Note that besides projecting on a Bell state, an initial entangled Bell-state is required. U_2 is a Clifford gate, hence there exists a $P^{\hat{a},\hat{b}}$ such that $U_2 P^{a,b} U_1 |\psi\rangle = P^{\hat{a},\hat{b}} U_2 U_1 |\psi\rangle$, allowing the correction term to be pushed to the end of the circuit. Repeating the same argument for multiple Clifford unitaries gives the quantum state $...P_2^{a_2,b_2} U_2 P_1^{a_1,b_1} U_1 |\psi\rangle$. Due to the conjugation relation of Clifford and Pauli gates, all correction terms can be postponed to the end of the computation.

3.2.1 Clifford-ladder circuit

A similar argument holds when looking at *Clifford-ladder circuits*.

Definition 3.7 (Clifford-ladder circuit)**.** Let $\{U^i\}_{i=0}^{n-1}$ be a collection of n 2-qubit Clifford unitaries. A Clifford-ladder circuit C_{ladder} is a circuit of depth $\mathcal{O}(n)$ and width $\mathcal{O}(n)$ of the following form:

$$C_{ladder} = \prod_{i=0}^{n-1} U_{i,i+1}^{(i)}$$

where $U_{i,i+1}^{(i)}$ denotes that unitary $U^{(i)}$ is applied on qubits i and $i+1$.

Note that each 2-qubit Clifford unitary $U^{(i)}$ itself is of constant-depth.

The next lemma shows that any Clifford-ladder circuit has an equivalent LAQCC circuit. Figure 1 shows this mapping graphically. Each two-qubit unitary is parallelized using gate teleportation and with the Clifford commutation relations, the Pauli correction terms are pushed to the end of the computation.

Lemma 3.8. *Any Clifford-ladder circuit has an* LAQCC *implementation of depth* $\mathcal{O}(1)$ *and width* $\mathcal{O}(n)$.

Proof. Figure 1 shows the construction of a LAQCC circuit of width $\mathcal{O}(n)$ and depth $\mathcal{O}(1)$ implementing a Clifford-ladder circuit. The caps and cups denote Bell-state measurements and Bell-state creation, respectively. What remains to show is that an NC^1 circuit computes the Pauli-correction terms.

The i-th Bell measurement results in Pauli error $P_i = Z^{a_i} X^{b_i}$. A Clifford-ladder circuit of size n hence has an error vector $\big(a\, b\big)$ of length $2n$. The correction terms that have to be applied have the same form: we can label every corrective Pauli by an index j, such that $\hat{P}_j = Z^{\hat{a}_j} X^{\hat{b}_j}$. This gives a correction vector $\big(\hat{a}\, \hat{b}\big)$. Note that Pauli matrices anti-commute, hence reordering them will only incur a global phase. This implies a binary linear map $M : \big(a\, b\big) \mapsto \big(\hat{a}\, \hat{b}\big)$. As matrix vector multiplication is in NC^1, this error calculation is in NC^1 and Clifford-ladder circuits have an LAQCC implementation. □

Remark 3.9. *Constructing the binairy linear map* M *is not in* NC^1*, but it does follow directly from the quantum circuit. Instead, an* L *(logspace) precomputation gives the matrix associated to* M.

This result directly implies that in LAQCC we can apply two-qubit gate on any two any non-adjacent qubits.

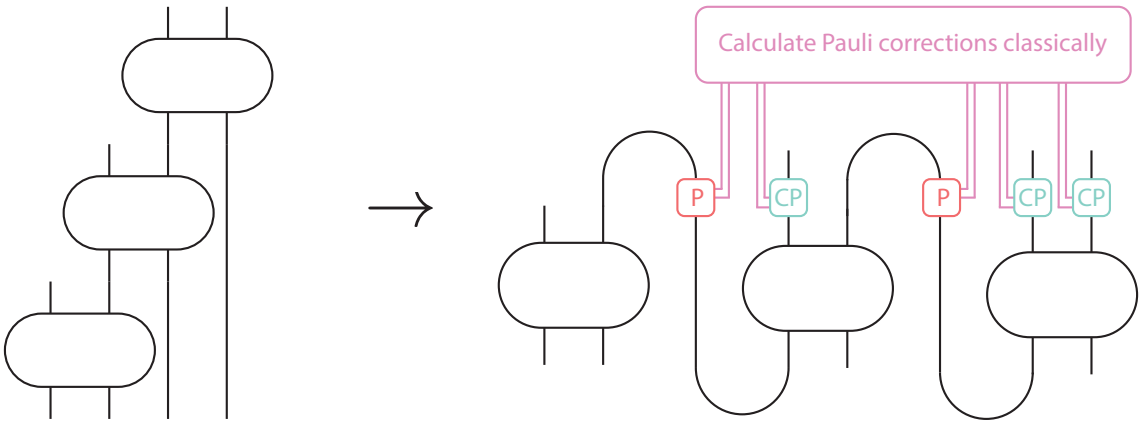

Figure 1: Graphical representation of Clifford-ladder circuit parallelization. Time flows upward and lines represent qubits and boxes quantum gates. A half circle represents either a Bell-state creation (ends pointing upwards) or a Bell-state measurement (ends pointing downwards). The Bell-state measurements can produce Pauli errors $P = Z^a X^b$, which are corrected by the boxes CP (corrective Pauli). The computations to determine how errors propagate are performed classically before the computations.

Corollary 3.10. *Any SWAP circuit needed to do an operation between non-adjacent qubits is a Clifford-ladder circuit and hence in* LAQCC.

This effectively removes the locality constraint in LAQCC *for applying a single* 2*-qubit gate on non-adjacent qubits.*

An example of a Clifford-ladder circuit is the creation of a GHZ state. We can parallelize this directly, for instance following the poor man's cat state approach of [54]. Figure 2 shows a LAQCC circuit using $2n-1$ qubits placed on a line that prepares an n-qubit GHZ state.

3.2.2 Clifford-grid circuit

Any Clifford unitary can be mapped to a linear-depth circuit given a linear nearest-neighbor architecture [29]. The most general representation of these circuits are so-called Clifford-grid circuits.

Definition 3.11 (Clifford-grid circuit)**.** Let n be the number of qubits. A Clifford-grid circuit of depth d is a circuit of the form

$$C_{grid} = \prod_{i=0}^{d} \bigotimes_{j=0}^{\frac{n}{2}} U_{i,j},$$

for Clifford unitaries $U_{i,j}$ and such that gate $U_{i,j}$ acts on qubits $2j$ and $2j+1$ if i is even, and $2j+1$ and $2j+2$ is i is odd.

The next lemma shows that Clifford-grid circuits also have an efficient LAQCC implementation.

Lemma 3.12. *Any Clifford-grid circuit of depth* $\mathcal{O}(n)$ *has an* LAQCC *implementation of depth* $\mathcal{O}(1)$ *and width* $\mathcal{O}(n^2)$.

Proof. Similar to the Clifford-ladder circuits, gate teleportation allows parallelization to obtain a LAQCC circuit. With a total of $\mathcal{O}(n^2)$ Clifford gates, this also requires $\mathcal{O}(n^2)$ qubits. Figure 3 illustrates the transformation.

Any Bell measurement in the circuit can incur a Pauli error, which has to be dealt with at the end of the circuit. The number of Pauli gates now scales with $\mathcal{O}(n^2)$. Similar to the Clifford-ladder

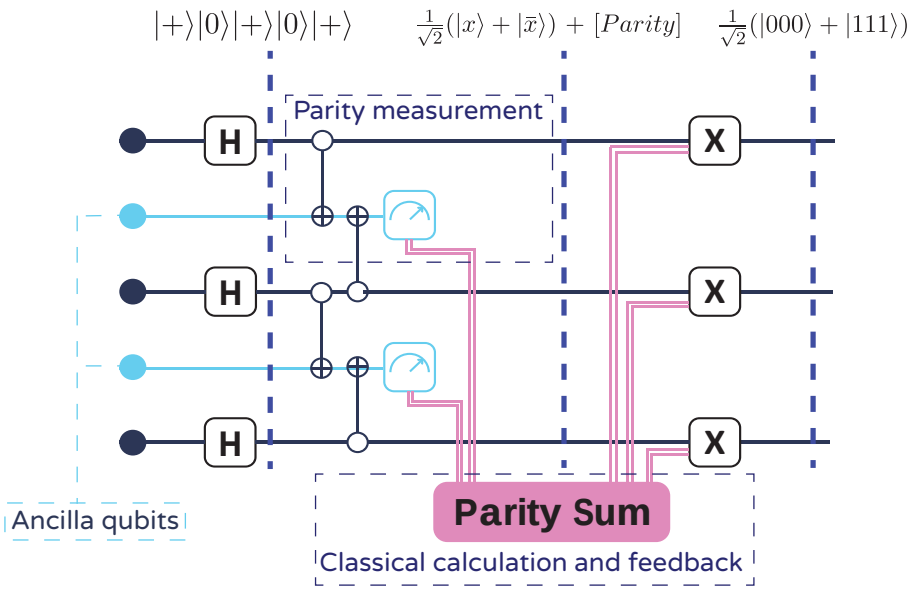

Figure 2: The quantum circuit to prepare the 3-qubit GHZ state. Double lines indicate classical information and dotted lines the quantum state at various points.

circuits, there now is a vector (ab) of length $\mathcal{O}(n^2)$ containing the information of the Pauli errors. The vector of correction terms, the vector $(\hat{a}\hat{b})$, has length $\mathcal{O}(n)$.

As these Pauli errors anti-commute, there again is a binary linear map $M : (ab) \mapsto (\hat{a}\hat{b})$. The corresponding matrix is rectangular and the error-correction calculations are in NC^1. □

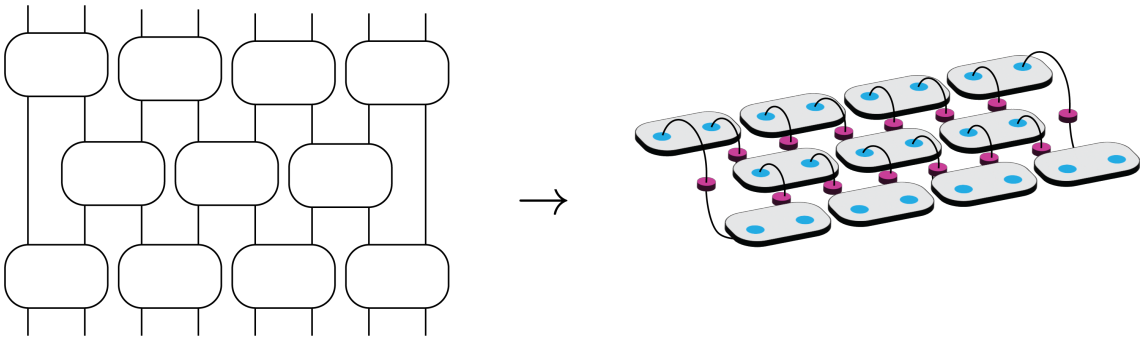

Figure 3: Graphical representation of Clifford-grid circuit parallelization. Every blue dot represents a qubit and all Clifford gates (boxes) are applied in parallel. The lines again represent Bell-state creations and Bell-state measurements, indicated by the pink boxes. The propagating Pauli errors can be corrected using the Bell-state measurement results.

Finding the matrix M for correcting a Clifford-grid circuit is more complex than for a Clifford-ladder circuit. An error occurring in the grid can have multiple paths contributing to a single output wire. For the final correction, the parity of each contributing error-path is needed. This computation is in $\oplus\mathsf{L} \subseteq \mathsf{NC}^2$ [2]. A precomputation again gives the matrix corresponding to the bilinear map M.

[2]This is not too surprising as simulating Clifford circuits classically is a complete problem for $\oplus\mathsf{L}$

3.3 Useful gates and routines with an LAQCC implementation

This section groups useful multi-qubit gates with an LAQCC implementation. The construction of W-states and Dicke states uses these gates, but their use might be of broader interest.

The tables give the action of the gates on computational basis states. Their effect on arbitrary states follows by linearity. The tables also give the width of the implementation and a reference to the implementation.

The first two gates directly follow from the Clifford-parallelization results described in Section 3.2.

Gate	Operation on basis states	Width	Implementation
Fanout_n	$\|x\rangle \|y_1\rangle \dots \|y_n\rangle \mapsto \|x\rangle \|y_1 \oplus x\rangle \dots \|y_n \oplus x\rangle$	$\mathcal{O}(n)$	Clifford-ladder circuit 3.8
$\text{Permutation}(\pi)_n$	$\|y_1\rangle \dots \|y_n\rangle \mapsto \|y_{\pi(1)}\rangle \dots \|y_{\pi(n)}\rangle$	$\mathcal{O}(n^2)$	Clifford-grid circuit 3.12

Table 1: Operations contained in LAQCC based on Clifford parallelization. Here $\pi \in S_n$ denotes a permutation of n elements.

Prior works extensively studied the fanout gate, for instance to construct a constant-depth OR_n function with one-sided error [24] and with an exact implementation [46], both assuming the fanout gate to be a native gate. The OR_n gate also implies two other gates, as the following table shows.

Gate	Operation on basis states	Width	Implementation
OR_n	$\|y_1\rangle \dots \|y_n\rangle \|x\rangle \mapsto \|y_1\rangle \dots \|y_n\rangle \|\text{OR}_n(y) \oplus x\rangle$	$\mathcal{O}(n \log(n))$	[46, Theorem 1]
AND_n	$\|y_1\rangle \dots \|y_n\rangle \|x\rangle \mapsto \|y_1\rangle \dots \|y_n\rangle \|\text{AND}_n(y) \oplus x\rangle$	$\mathcal{O}(n \log(n))$	negate input and output of OR_n
Equal_i	$\|j\rangle \|b\rangle \mapsto \begin{cases} \|j\rangle \|1 \oplus b\rangle & \text{if } \|j\rangle = \|i\rangle \\ \|j\rangle \|b\rangle & \text{else} \end{cases}$	$\mathcal{O}(n \log(n))$	negate part of input of AND_n

Table 2: Operations contained in LAQCC based on Fanout and local 1-qubit unitaries.

With these unbounded-fan-in OR and AND gates, all AC^0 circuits can be implemented. The next step is implementing LAQCC-type modular addition circuits, which gives circuits to check for equality and greater-than. These three gates take n-qubit quantum states as input. We introduce the indicator variable $\mathbb{1}_A$ for a Boolean expression A, which evaluates to 1 if A is true. Similarly, $|\mathbb{1}_A\rangle = |1\rangle$ if and only if A is true.

Gate	Operation on n-qubit integers $\|x\rangle, \|y\rangle$	Width	Implementation
Add_n	$\|x\rangle \|y\rangle \mapsto \|x\rangle \|y + x \bmod 2^n\rangle$	$\mathcal{O}(n^2)$	AC^0 circuit
Equality	$\|x\rangle \|y\rangle \|0\rangle \mapsto \|x\rangle \|y\rangle \|\mathbb{1}_{x=y}\rangle$	$\mathcal{O}(n^2)$	Appendix B.1
Greatherthan	$\|x\rangle \|y\rangle \|0\rangle \mapsto \|x\rangle \|y\rangle \|\mathbb{1}_{x>y}\rangle$	$\mathcal{O}(n^2)$	Appendix B.2

Table 3: Operations contained in LAQCC based on AC^0 circuits.

Hoyer and Spalek showed that fanout-gates imply efficient constant-depth implementations of for instance the quantum Fourier transform [24]. They use this constant-depth quantum Fourier transform to construct a constant-depth circuit for weighted counting. In particular, this circuit can be used to calculate the Hamming weight of an n-bit string, and to implement an "Exact t"-gate and a threshold gate. Appendix B.4 also explains how to modify the threshold gate to a weighted threshold gate.

Remark 3.13. *As the Threshold$_t$ gate is in* LAQCC*, any classical* TC_0 *circuit is in* LAQCC*.*

This section concludes not with a gate, but with a tool used for preparing uniform superpositions. This lemma extends Lemma 2.7 to preparing states instead of finding marked items.

Gate	Operation on n-qubit basis state $\|x\rangle$	Width	Implementation
QFT	$\|x\rangle \mapsto \frac{1}{\sqrt{2^{n-1}}} \sum_{j=0}^{2^{n-1}} e^{i2\pi \frac{x \cdot j}{2^n}} \|j\rangle$	$\mathcal{O}(n^3 \log(n))$	[24, Theorem 4.12]
Hammingweight	$\|x\rangle_n \|0\rangle_{\log(n)} \mapsto \|x\rangle_n \||x|\rangle_{\log(n)}$	$\mathcal{O}(n \log(n))$	[46, Lemma 4]
Exact_t	$\|x\rangle \|0\rangle \mapsto \|x\rangle \left\|\mathbb{1}_{\|x\|=t}\right\rangle$	$\mathcal{O}(n \log(n))$	Appendix B.3
Threshold_t	$\|x\rangle \|0\rangle \mapsto \|x\rangle \left\|\mathbb{1}_{\sum_i x_i \geq t}\right\rangle$	$\mathcal{O}(tn \log(n))$	Appendix B.4

Table 4: Quantum subroutines in LAQCC based on [24].

Lemma 3.14. *Given an n-qubit unitary U, that is implementable by a constant-depth circuit, a basis $\mathcal{C}$ and a partition of $\mathcal{C}$ in $\mathcal{G}$ and $\mathcal{B}$ such that $\frac{|\mathcal{G}|}{|\mathcal{C}|}$ is a known constant c. Suppose that U implements the map*

$$U : |y\rangle |b\rangle \mapsto \begin{cases} |y\rangle |b \oplus 1\rangle & \text{if } y \in \mathcal{G} \\ |y\rangle |b\rangle & \text{if } y \in \mathcal{B} \end{cases}.$$

Then there exists a LAQCC *circuit that prepares the state $\frac{1}{\sqrt{|\mathcal{G}|}} \sum_{y \in \mathcal{G}} |y\rangle$ by using U a constant number of times.*

Proof. Define $|\mathcal{G}\rangle = \frac{1}{\sqrt{|\mathcal{G}|}} \sum_{y \in \mathcal{G}} |y\rangle$ and $|\mathcal{B}\rangle = \frac{1}{\sqrt{|\mathcal{B}|}} \sum_{y \in \mathcal{B}} |y\rangle$. As $\mathcal{B}$ and $\mathcal{G}$ partition $\mathcal{C}$, it follows that $\langle \mathcal{G} | \mathcal{B} \rangle = 0$. Lemma 2.7 implies the existence of a circuit that prepares the desired state. Below, we explicitly construct the circuit.

First, prepare a uniform superposition $\sum_{i=0}^{2^n-1} |i\rangle$. Then, iteratively reflect over the state $|\mathcal{B}\rangle$ using U, and reflect over the uniform superposition state $\sum_{i=0}^{2^n-1} |i\rangle$. Both reflections have a LAQCC implementation and we only need to apply them a constant number of iterations.

To reflect over the uniform superposition, we have to implement the operation $2 |s\rangle \langle s| - I$, with $|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{2^n-1} |i\rangle$. To implement this operation, we first apply a layer of Hadamards, which implements a basis transformation mapping the uniform superposition state to the all zeroes state; Then apply the Exact_0-gate producing an output qubit that marks only the all zeroes-state and finally negate the output qubit and applies a Z-gate on it. Running this circuit in reverse, excluding the Z-gate, resets the output qubit and reverts the basis transformation. The last step of Lemma 3.14 requires a reflection using an R_Z-gate (rotational Z-gate) instead of the Z-gate. As the Exact_0-gate has an LAQCC implementation (see Table 4), this second inversion operation has a LAQCC implementation.

The total number of iterations is $\mathcal{O}(\sqrt{N/m})$, where $N = |\mathcal{C}|$ and $m = |\mathcal{G}|$. As their fraction is the constant c, it follows that $\mathcal{O}(\sqrt{c}) = \mathcal{O}(1)$ iterations are needed. □

3.4 Non-simulatability of LAQCC

Most of the power of LAQCC circuits seems to come from the classical intermediate calculations, which makes one wonder if these circuits are classically simulatable. Even if these circuits were indeed efficiently simulatable, they still have value as "fast" alternatives for state preparation. However, it is unlikely that all LAQCC circuits can be simulated efficiently by a classical simulator. Lemma 2.8 and the inclusion of the fan-out gate in LAQCC show that circuits consisting of commuting gates have an LAQCC implementation and in particular, the class of Instantaneous Quantum Polynomial-time (IQP) circuits, first introduced in [44], has equivalent LAQCC implementations.

Definition 3.15 (Definition 2 [32])**.** An IQP circuit on n qubits is a quantum circuit with the following structure: each gate in the circuit is diagonal in the Pauli-Z basis, the input state is $|+\rangle^{\otimes n}$, and the output is the result of a measurement in the Pauli-X basis on a specified set of output qubits.

Lemma 3.16. *Any* IQP *circuit has an* LAQCC *implementation.*

Proof. The following LAQCC circuit prepares the desired state: First prepare $|+\rangle^{\otimes n}$ by a single layer of Hadamard gates on all qubits. In this basis, all gates in with respect to the Pauli-Z basis commute, and hence by Lemma 2.8, we can parallelize all gates using poly(n) ancilla qubits. Next, we can again apply a layer of Hadamard gates and finally measure the desired qubits. □

Bremner, Jozsa, and Shepherd showed that efficient weak classical simulation of all possible IQP circuits up to small multiplicative error implies a collapse of the polynomial hierarchy [11]. Note that a circuit family is weakly simulatable if given the description of the circuit family, its output distribution can be sampled by purely classical means in poly(n) time.

Lemma 3.17 (Corollary 1 [11])**.** *If the output probability distributions generated by uniform families of* IQP *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c < \sqrt{2}$ *then the polynomial hierarchy would collapse to the third level, in particular,* $\mathsf{PH} = \Delta_3^p$.

Corollary 3.18. *If the output probability distributions generated by uniform families of* LAQCC *circuits could be weakly classically simulated to within multiplicative error* $1 \leq c < \sqrt{2}$ *then the polynomial hierarchy would collapse to the third level, in particular,* $\mathsf{PH} = \Delta_3^p$.

3.5 LAQCC containment in QNC^1

Let A be an LAQCC-circuit. We can write this circuit as a composition of unitary quantum layers U_i, measurements M_i and classical calculation layers C_i:

$$A = M_k U_k C_k \dots M_i U_i C_i \dots M_1 U_1 C_1,$$

for some constant k. Any unitary U_i is a QNC^0 circuit and any C_i is an NC^1-circuit. The measurements M_i can measure any subset of the qubits. By the principle of deferred measurements, we can always postpone them to the end of the circuit using $CNOT$ gates and fresh ancilla qubits [34, Section 4.4], which gives the following lemma.

Lemma 3.19. *For any* LAQCC*-circuit A there is a* QNC^1*-circuit B without intermediate measurements that outputs the same state as A.*

Proof. The LAQCC-circuit A contains classical computation layers C_i that use the intermediate measurement results as input. These measurements can be delayed until the end of the circuit by applying a CNOT from the qubit to a fresh ancilla qubit. This replaces the classical output wires by quantum wires.

Lemma A.1 shows that any NC^1-circuit can be run on a log-depth quantum circuit with $\mathcal{O}(\text{poly}(n))$ ancilla qubits. Hence, a QNC^1-circuit without topology constraints can take the role of the classical intermediate circuits C_i.

Now, let V_i be the quantum circuit implementing C_i. Then the quantum circuit

$$B = U_k V_k \dots U_1 V_1$$

is a quantum circuit of logarithmic depth simulating A. □

4 State preparation in LAQCC

In this section we consider what quantum states we can prepare using an LAQCC circuit beyond the stabilizer states and Clifford circuits discussed in the previous section. First, we show how to create a uniform superposition of computational basis states up to size q, where q is not a power of 2. We then use this procedure to create W-states, the uniform superposition over all n-bitstrings of Hamming-weight 1, and their generalization Dicke-(n, k) states, the uniform superposition over all n-bitstrings of Hamming-weight k.

4.1 Uniform superposition of size q

A simple Hadamard gate applied to n qubits prepares the uniform superposition $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle$. Preparing the state $\frac{1}{\sqrt{q}}\sum_{i=0}^{q-1}|i\rangle$, the superposition up to size q, is already harder for arbitrary q.

A simple probabilistic approach works as follows: 1) create a superposition $\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle$ with $n = \lceil \log_2(q) \rceil$ qubits; 2) mark the states $i < q$ using an ancilla qubit; 3) measure this ancilla qubit. Based on the measurement result, the desired superposition is found, which happens with probability at least one half.

The next theorem modifies this probabilistic approach to a protocol that deterministically prepares the uniform superposition modulo q in LAQCC.

Theorem 4.1. *There is a deterministic* LAQCC *circuit that prepares the uniform superposition of size q. This circuit requires $\mathcal{O}(\log_2(q)^2)$ qubits.*

Proof. Let $n = \lceil \log_2(q) \rceil$ and define $\mathcal{G} = \{i \mid 0 \leq i < q\}$ and $\mathcal{B} = \{i \mid q \leq i \leq 2^n - 1\}$. Construct the unitary

$$U_q : |y\rangle|b\rangle \mapsto \begin{cases} |y\rangle|b \oplus 1\rangle & \text{if } y < q \\ |y\rangle|b\rangle & \text{if } y \geq q \end{cases}.$$

The Greaterthan-gate of Table 3 implements the operator U_q, note that this gate requires $\mathcal{O}(n^2)$ qubits.

As $|\mathcal{G}|/2^n \geq 1/2$ and known, applying Lemma 3.14 with the sets $\mathcal{G}$ and $\mathcal{B}$ and the constant-depth implementation of U_q, gives a LAQCC algorithm that boosts the amplitude of $|\mathcal{G}\rangle$ to 1. □

Remark 4.2. *Note that in Lemma 3.14 it was implicitly assumed that $|\mathcal{G}| + |\mathcal{B}|$ is a power of two (allowing for a simple reflection over the uniform superposition state). This* LAQCC *implementation of creating a uniform superposition modulo any q removes this requirement.*

4.2 W-state in LAQCC

In this section we consider the W_n-state and how to prepare this state in LAQCC. The W_n-state is a uniform superposition over all n-qubit states with a single qubit in the $|1\rangle$-state and all others in the $|0\rangle$-state:

$$|W_n\rangle = \frac{1}{\sqrt{n}}\sum_i |e_i\rangle,$$

where $|e_i\rangle$ is the state with a one on the i-th position and zeroes elsewhere.

A first observation is that the W-state can be seen as a one-hot encoding of a uniform superposition over n elements. We can label the n states with non-zero amplitude of the W-state with an index. More precisely, we want to find circuits that implement the following map:

$$|i\rangle|0\rangle \mapsto |0\rangle|e_i\rangle, \tag{1}$$

with i an index and e_i the one-hot encoding of i. This index – which equals the position of the 1 – compresses the representation from n to $\log(n)$ bits. This compression naturally defines two operations:

$$\textbf{Uncompress}: \ |i\rangle_{\log(n)}|0\rangle_n \mapsto |i\rangle_{\log(n)}|e_i\rangle_n, \tag{2}$$

$$\textbf{Compress}: \ |i\rangle_{\log(n)}|e_i\rangle_n \mapsto |0\rangle_{\log(n)}|e_i\rangle_n. \tag{3}$$

Implementing both and combining them implements Mapping 1 giving an efficient W-state preparation protocol.

The **Compress** and **Uncompress** operations map between a one-hot and binary represenation of an integer i. We call the registers containing the binary representation index registers, and the register containing the one-hot representation the system register. The index registers serve as ancilla qubits and the W-state is prepared in the system register.

Lemma 4.3. *There exists an* LAQCC *circuit for any n implementing* ***Uncompress****, more specifically implementing the map:* $\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}|0\rangle_n \mapsto \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle|e_i\rangle_n$*. This circuit uses* $\mathcal{O}(n\log(n))$ *qubits placed in a grid pattern of size* $n\times(\log(n))$*.*

Proof. One column of the grid of length n consists of system qubits placed in a line. Adjacent to this line are $\log(n)$ index qubits. The left grid in Figure 4 shows the initial layout. The same figure also shows the steps to prepare the W-state in the system qubits.

$$\begin{aligned}\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}|0\rangle_{\log(n)}^{\otimes n-1}|0\rangle_n &\xrightarrow{(1)} \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}^{\otimes n}|0\rangle_n\\ &\xrightarrow{(2)} \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}^{\otimes n}|e_i\rangle_n\\ &\xrightarrow{(3)} \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle|0\rangle^{\otimes n-1}|e_i\rangle_n\end{aligned}$$

Step (1) uses fanout-gates to create a fully entangled state between the different index registers. Step (2) applies Equal_i gates in parallel from each index register to its corresponding system qubit to create the state $|e_i\rangle$ in the system register. Step (3) uses fanout-gates to disentangle and reset the index registers. Combined the **Uncompress** operations maps $\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}|0\rangle_{\log(n)}^{\otimes n-1}|0\rangle_n \mapsto \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle|0\rangle^{\otimes n-1}|e_i\rangle_n$ as required. □

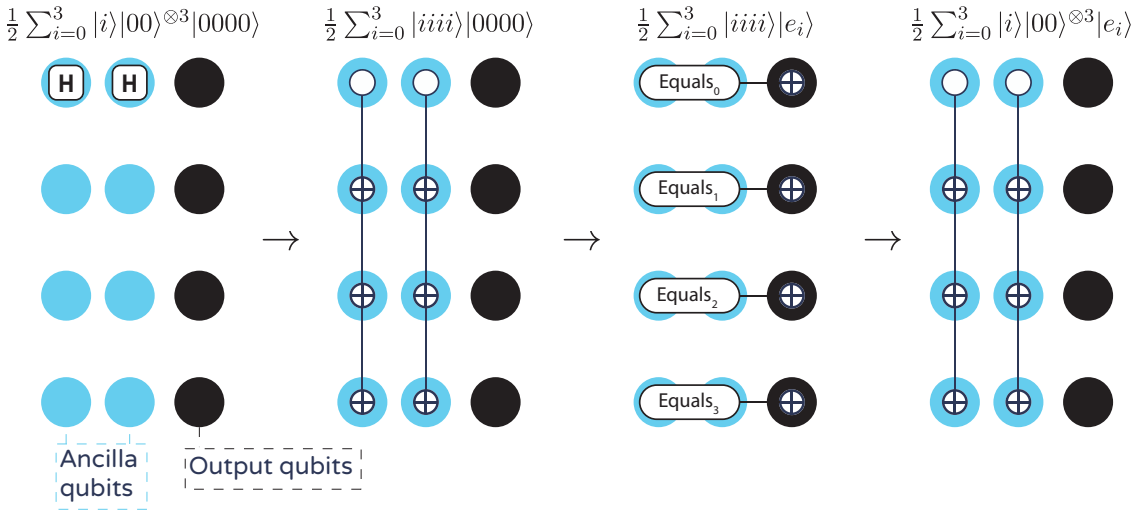

Figure 4: Circuit for the **Uncompress** operation for $n = 4$. Shown is a grid of 12 qubits: 8 blue index qubits, and 4 black system qubits. Each of the four grids represents a single timeslice in the **Uncompress** operation.

Lemma 4.4. *There exists an* LAQCC *circuit for any n implementing* ***Compress****, more specifically implementing the map:* $\frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle_{\log(n)}|e_i\rangle_n \mapsto \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|0\rangle|e_i\rangle_n$*. This circuit uses* $\mathcal{O}(n\log(n))$ *qubits placed in a grid pattern of size* $n\times(\log(n))$*.*

Proof. To implement **Compress**, the index registers are uncomputed using parallel $CNOT$-operations, controlled by the system register. These controlled gates commute for different indices in the system register and hence by Lemma 2.8 a parallel circuit for the uncomputation exists.

The **Compress** operation, also shown in Figure 5, consists of the operations:

$$\begin{aligned}
\frac{1}{\sqrt{n}}\sum_{i=0}^{n}|i\rangle_{\log(n)}|0\rangle_{\log(n)}^{\otimes n-1}|e_i\rangle_n &\xrightarrow{(1)} \frac{1}{n}\sum_{i,j=0}^{n}(-1)^{i\cdot j}|j\rangle_{\log(n)}|0\rangle_{\log(n)}^{\otimes n-1}|e_i\rangle_n \\
&\xrightarrow{(2)} \frac{1}{n}\sum_{i,j=0}^{n}(-1)^{i\cdot j}|j\rangle_{\log(n)}^{\otimes n}|e_i\rangle_n \\
&\xrightarrow{(3)} \frac{1}{n}\sum_{i,j=0}^{n}|j\rangle_{\log(n)}^{\otimes n}|e_i\rangle_n \\
&\xrightarrow{(4)} \frac{1}{n}\sum_{i,j=0}^{n}|j\rangle_{\log(n)}|0\rangle_{\log(n)}^{\otimes n-1}|e_i\rangle_n \\
&\xrightarrow{(5)} \frac{1}{\sqrt{n}}\sum_{i=0}^{n}|0\rangle_{\log(n)}^{\otimes n}|e_i\rangle_n
\end{aligned}$$

Step (1) applies Hadamard gates to the first index register, changing from the computational to the Hadamard basis, in which the NOT-operation is diagonal; Step (2) uses fanout-gates to create a fully entangled state in the index registers; Step (3) applies controlled-Z gates, controlled by the system qubit i and with targets the qubits in the i-th index register corresponding to the ones in the binary representation of i; Step (4) disentangles the index registers using fanout-gates; and, Step (5) applies Hadamard gates to clean the index register.

The controlled-Z gates in Step (3) apply phases that precisely cancel the phases already present, which disentangles the index registers from the system register. □

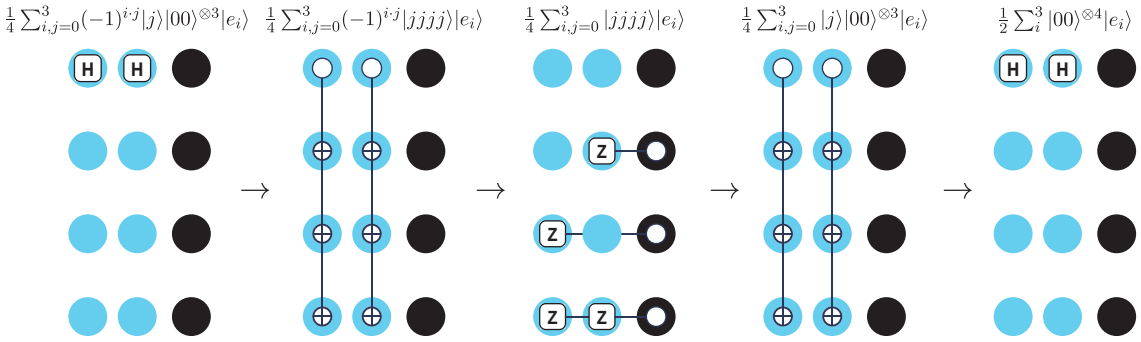

Figure 5: Circuit for the **Uncompress** operation for $n = 4$. Shown is a grid of 12 qubits: 8 blue index qubits, and 4 black system qubits. Each of the four grids represents a single timeslice in the **Compress** circuit.

Theorem 4.5. *There exists a circuit in* LAQCC *that prepares the* $|W_n\rangle$ *state. This circuit requires* $\mathcal{O}(n\log(n))$ *qubits placed in a grid of size size* $n \times (\log(n))$.

Proof. The circuit combines the circuits of Theorem 4.1, Lemma 4.3 and Lemma 4.4. It consists

of three steps:

$$
\begin{aligned}
|0\rangle_{\log(n)}^{\otimes n} |0\rangle_n &\xrightarrow{(1)} \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle |0\rangle^{\otimes n-1} |0\rangle \\
&\xrightarrow{(2)} \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle |0\rangle^{\otimes n-1} |e_i\rangle \\
&\xrightarrow{(3)} \frac{1}{\sqrt{n}} \sum_{i=0}^{n} |0\rangle^{\otimes n} |e_i\rangle
\end{aligned}
$$

Step one prepares the uniform superposition over indices, this can be done either by applying a layer of Hadamard gates, if n is a power of 2, requiring $\mathcal{O}(log(n))$ qubits, or using Theorem 4.1 if n is not a power of 2 requiring $\mathcal{O}(log(n)^2)$ qubits.; Step (2) is by Lemma 4.3 and requires $\mathcal{O}(n \log(n))$ qubits; and, Step(3) is by Lemma 4.4 and requires $\mathcal{O}(n \log(n))$ qubits. □

4.3 Dicke states for small k

In this section we generalize our method of preparing the $|W\rangle$-state in Theorem 4.5 to a more general set of states, Dicke states. A Dicke-(n, k) state is the uniform superposition over bitstrings of Hamming weight k and length n (which we again assume to be a power of 2 for simplicity):

$$
|D_k^n\rangle = \binom{n}{k}^{-1/2} \sum_{x \in \{0,1\}^n : |x| = k} |x\rangle . \tag{4}
$$

For $k = 1$, this state is precisely the W-state. There exists an efficient deterministic method to prepare a $|D_k^n\rangle$ state that requires a circuit of width $\mathcal{O}(n)$ and depth $\mathcal{O}(n)$, independent of k [5]. This methods starts from the $|1\rangle^{\otimes k} |0\rangle^{\otimes k-n}$ state and relies on a recursive formula for the Dicke state

$$
|D_k^n\rangle = \alpha_{k,n} \left|D_k^{n-1}\right\rangle \otimes |0\rangle + \beta_{k,n} \left|D_{k-1}^{n-1}\right\rangle \otimes |1\rangle .
$$

This relation implies a protocol that is inherently sequential, which is unsuited for an LAQCC implementation.

Instead, we present a LAQCC approach similar to the W-state preparation protocol. We apply the **Uncompress** operation of the W-state in parallel to put k ones into the bitstring. This method allows for the preparation of Dicke states with $k = \mathcal{O}(\sqrt{n})$, using $\mathcal{O}(n^2 \log(n)^3)$ qubits. The bound on k comes from the fact that using the **Uncompress** operation in parallel might cause overlaps to where the 1's are put into the system register. Having two ones in the same system qubit in effects negates the of the **Uncompress** operation. This gives a similar derivation as the birthday paradox which implies that the overlaps don't not happen to often for $k = \mathcal{O}(\sqrt{n})$. By using Lemma 3.14 boosts the amplitudes, and makes the protocol deterministic.

Again, consider two groups of qubits: Index registers with $\log(n)$ qubits each; and, system registers of n qubits each. Contrary to the W-state, the Dicke state requires multiple system registers during the preparation. The state is prepared in only one system register. Denote the index registers with subscripts i_1 up to i_k and the system registers with s_1 up to s_n. For clarity, these indices may be omitted if it is clear from the context.

The algorithm consists of four steps:

1. **Filling**: $|0\rangle_{i_1} \dots |0\rangle_{i_k} |0\rangle_{s_1} \to \frac{1}{n^{k/2}} \sum_{j_1,\dots,j_k=0}^{n-1} |j_1\rangle_{i_1} \dots |j_k\rangle_{i_k} |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1}$
2. **Filtering**: $\to \sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle$
3. **Ordering**: $\to \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1 < \dots < j_k}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle$

4. **Cleaning**: $\to \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1<\cdots<j_k}^{n-1} |0\rangle \dots |0\rangle |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle$

Note that after **Filling** there is a multiplicity in states. First, **Filtering** removes those states in which different indices j_l are the same, resulting in an incorrect state in the s_1 register. Second, **Ordering** removes the multiplicity from having multiple permutations of the index registers creating the same state in the s_1 register, by forcing a choice of ordering on the indices. These two steps give a unique pairing between index registers and system registers allowing the opperation **Cleaning**.

We will now proof that these four steps are achievable in LAQCC and explicitly visualize the corresponding circuits for $n = 4$ and $k = 2$.

Lemma 4.6. *There exists a circuit in* LAQCC *implementing* ***Filling****. More precisely implementing the map* $|0\rangle_{i_1} \dots |0\rangle_{i_k} |0\rangle_{s_1} \to \frac{1}{n^{k/2}} \sum_{j_1,\dots,j_k=0}^{n-1} |j_1\rangle_{i_1} \dots |j_k\rangle_{i_k} |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle_{s_1}$*. This circuit requires* $\mathcal{O}(kn \log(n))$ *qubits.*

Proof. To achieve a circuit implementing **Filling** we use **Uncompress** from Lemma 4.3 k times in parallel. Note that two **Uncompress** operations commute, hence by Lemma 2.8 k **Uncompress** operations can be implemented in parallel. Each of these parallel operations require an index register, a system register and $\mathcal{O}(n \log(n))$ extra ancilla qubits.

The correpsonding circuit consists of five steps:

$$
\begin{aligned}
|0\rangle_{i_1} \dots |0\rangle_{i_k} |0\rangle_{s_1} \dots |0\rangle_{s_k} &\xrightarrow{(1)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} |l\rangle_{s_1} |0\rangle_{s_2} \dots |0\rangle_{s_k} \\
&\xrightarrow{(2)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} |l\rangle_{s_1} |l\rangle_{s_2} \dots |l\rangle_{s_k} \\
&\xrightarrow{(3)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} (-1)^{(2^{j_1}+\cdots+2^{j_k})\cdot l} |l\rangle_{s_1} |l\rangle_{s_2} \dots |l\rangle_{s_k} \\
&\xrightarrow{(4)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} (-1)^{(2^{j_1}+\cdots+2^{j_k})\cdot l} |l\rangle_{s_1} |0\rangle_{s_2} \dots |0\rangle_{s_k} \\
&\xrightarrow{(5)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle_{s_1} |0\rangle_{s_2} \dots |0\rangle_{s_k}
\end{aligned}
$$

Step (1) brings all index registers in a uniform superposition of size n, use Theorem 4.1 if required, and one system register in a uniform superposition size 2^n; Step (2) uses fan-out gates to create entangled copies of the system register; Step (3) applies a phase flip between every pair of index and system register using **Uncompress** of Lemma 4.3, except instead of applying not gates to the system registers, apply phase gates; Step (4) uses fan-out gates to disentangle and uncompute all but one of the system registers; Step (5) applies Hadamard gates on the system register to obtain the one-hot representation of the index registers. Step(3), the step that requires most qubits, requires $\mathcal{O}(n \log(n))$ qubits for every pair of index and system register, of which there are k, resulting in the requirement of $\mathcal{O}(kn \log(n))$ qubits. □

Figure 6 shows these five steps graphically. Ancilla qubits are omitted for clarity. Note that some of the j_i in the index registers may intersect. The next Filtering step takes care of that.

Lemma 4.7. *There exists a circuit in* LAQCC *implementing* ***Filtering****. More precisely implementing the map* $\frac{1}{n^{k/2}} \sum_{j_1,\dots,j_k=0}^{n-1} |j_1\rangle_{i_1} \dots |j_k\rangle_{i_k} |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle_{s_1} \to \sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle$*. This circuit requires* $\mathcal{O}(kn \log(n))$ *qubits.*

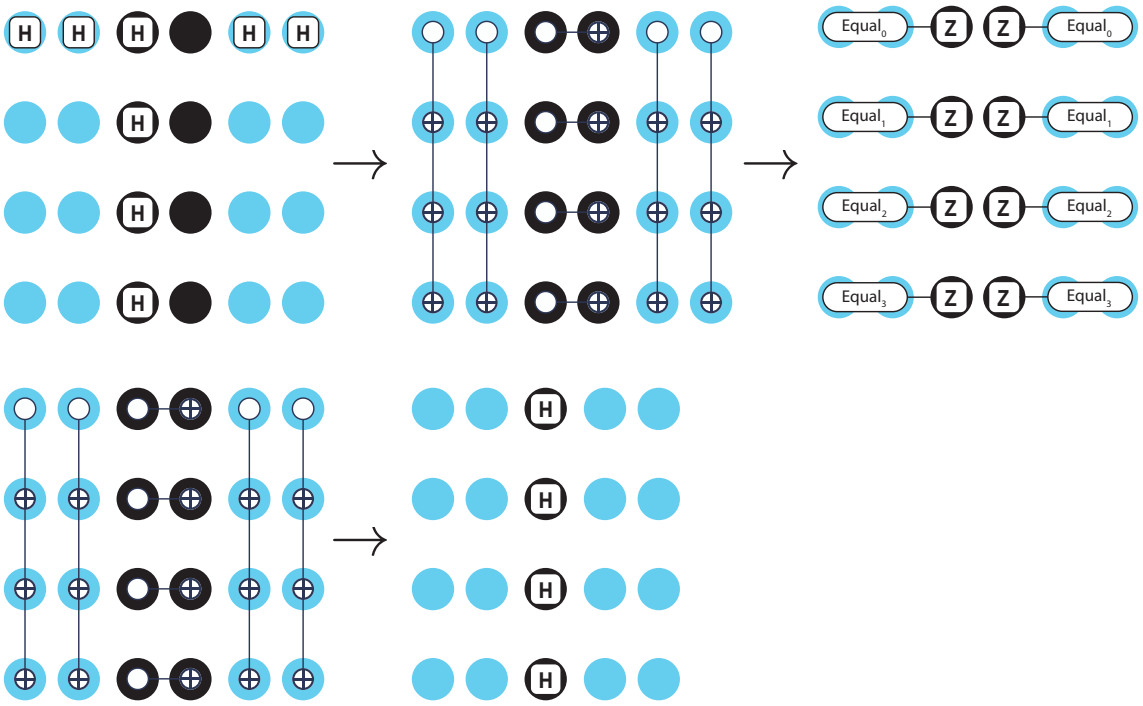

Figure 6: Circuit to implement **Filling**, $|0\rangle_{i_1} \dots |0\rangle_{i_k} |0\rangle_{s_1} \to \frac{1}{n^{k/2}} \sum_{j_1,\dots,j_k=0}^{n-1} |j_1\rangle_{i_1} \dots |j_k\rangle_{i_k} |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1}$. This circuit requires $\mathcal{O}(kn\log(n))$, for $n = 4$ and $k = 2$. A grid of 24 qubits is shown: 16 blue index qubits and 8 black system qubits. Each of the five grids represents a single timeslice in the circuit.

Proof. First note that the state produced by the Filling step,

$$\frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1},$$

contains states in which some of the indices j_l overlap. Let $|\psi\rangle = \sum_{j_1 \neq \dots \neq j_k} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle$, be the state in which none of the indices overlap, the desired output state. Then we can write

$$\frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1} = \alpha |\psi\rangle + \beta |\psi^\perp\rangle,$$

with $|\psi^\perp\rangle$ containing the states in which at least two of the indices j_l ovelap. Note that $\langle\psi|\psi^\perp\rangle = 0$. α can be exactly calculated by counting the number of quantum states with distinct j_i's, which gives $|\alpha|^2 = \frac{n!}{(n-k)!n^k}$. Lemma A.2 gives a lower bound on $|\alpha|^2$:

$$|\alpha|^2 = \frac{n!}{(n-k)!n^k} > e^{\frac{-2k^2}{n}},$$

which is at least constant for $k = \mathcal{O}(\sqrt{n})$.

The state $|\psi^\perp\rangle$ is a superposition of states in which the system register state has Hamming weight less than k, because at least two of the j_i's are the same causing a cancellation in the system register. We can use this to create a unitary U_{flag} that flags $|\psi^\perp\rangle$. We implement this in two

steps:

$$\frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1} \, |0\rangle_{log(k)} \, |0\rangle$$

$$\xrightarrow{(1)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1} \, ||e_{j_1} \oplus \dots \oplus e_{j_k}|\rangle \, |0\rangle$$

$$\xrightarrow{(1)} \frac{1}{n^{k/2}} \sum_{j_1 \dots j_k=0}^{n-1} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1} \, |0\rangle \left| \mathbb{1}_{|e_{j_1} \oplus \dots \oplus e_{j_k}|==k} \right\rangle = \alpha \, |\psi\rangle \, |1\rangle + \beta \left|\psi^{\perp}\right\rangle |0\rangle$$

Where $||$ denotes the Hamming weight of the bit string. Step (1) is achieved by applying a Hammingweight gate from Table 4, this requires $\mathcal{O}(n \log(n))$ qubits. Step(2) is achieved by applying an Exact_k gate, this requires $\mathcal{O}(\log(n))^2$ qubits. Step (2) also undoes the Hammingweight calculation of step (1) requiring $\mathcal{O}(n \log(n))$ qubits.

Now we can use Lemma 3.14 to amplify α to 1, with the oracle U_{flag}, and **Filling** the unitary that creates starting state $|\psi\rangle$. This produces the state

$$\sqrt{\frac{(n-k)!}{(n)!}} \sum_{j_1 \neq \dots \neq j_k} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \, .$$

using $\mathcal{O}(kn \log(n))$ qubits. □

To uncompute the index registers, we have to know which one in the system register corresponds to which index register, as any permutation of the index registers results in the same state in the system register. The **Ordering** step imposes an ordering on the index registers, thereby removing the redundancy in the ordering.

Lemma 4.8. *There exists a circuit in* LAQCC *implementing* ***Ordering****. More precisely implementing the map* $\sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k}^{n-1} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \to \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1 < \dots < j_k}^{n-1} |j_1\rangle \dots |j_k\rangle \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle$*. This circuit requires* $\mathcal{O}(k^2 \log(n)^2)$ *qubits.*

Proof. The first step of the LAQCC circuit that implements **Ordering** is to evaluate a Greaterthan-gate on all pairs of index registers, which requires k copies of each index register. We require k extra qubits per index register to store the outcome of the Greaterthan-gates. The copies of the index registers are created by doing a fan-out gate. Note that the distribution of the index registers should be set up in such a way that every possible pair can be compared by a Greaterthan-gate. We will denote the number of copies needed for every register, not how they are distributed.

$$\sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k} |j_1\rangle^{\otimes k} \, |0\rangle^{\otimes k} \dots |j_k\rangle^{\otimes k} \, |0\rangle^{\otimes k} \, |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \xrightarrow{(1)}$$

$$\sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k} \left[\, |j_1\rangle^{\otimes k} \, |\mathbb{1}_{j_1 > j_2}\rangle \dots |\mathbb{1}_{j_1 > j_k}\rangle \, \right] \dots \left[\, |j_k\rangle^{\otimes k} \, |\mathbb{1}_{j_k > j_1}\rangle \dots \left|\mathbb{1}_{j_k > j_{k-1}}\right\rangle \, \right] |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \, .$$

Each $\mathbb{1}_{j_k > j_{k'}}$ is an indicator variable that evaluates to one if and only if $j_k > j_{k'}$. This step requires $\mathcal{O}(k^2 \log(n)^2)$ qubits to perform. Next, we compute and measure the Hamming weight of the ancilla qubits $|\mathbb{1}_{j_1 > j_2}\rangle \dots |\mathbb{1}_{j_1 > j_k}\rangle$, using the gate from Table 4. By the Filtering step, all index registers are distinct, and hence, all measurement results are distinct. The measurement results, therefore, impose an ordering on the registers.

$$\xrightarrow{\text{(Hammingweight)}} \sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \cdots \neq j_k} \left[|j_1\rangle \, |\mathbb{1}_{j_1>j_2} + \cdots + \mathbb{1}_{j_1>j_k}\rangle \right] \\ \ldots \left[|j_k\rangle \, \left|\mathbb{1}_{j_k>j_1} + \cdots + \mathbb{1}_{j_k>j_{k-1}}\right\rangle \right] |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle$$

$$\xrightarrow{(measure)} \binom{n}{k}^{-1/2} \sum_{j_1<\cdots<j_k} \left[|j_1\rangle \, |0\rangle \right] \ldots \left[|j_k\rangle \, |k\rangle \right] |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle$$

This step costs $\mathcal{O}(k^2 \log(k))$ qubits. Assume without loss of generality that the measurement outcomes impose the ordering $j_1 < \cdots < j_k$. Otherwise, a permutation of the index registers achieves the same ordering, using the Permutation gate from Table 1.

Uncomputing the Hamming weights and the Greaterthan-gates gives the state

$$\binom{n}{k}^{-1/2} \sum_{j_1<\cdots<j_k} \left[|j_1\rangle \ldots |j_k\rangle \right] |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle \, .$$

□

The next Cleaning step will clean the index registers. The **Cleaning** step cleans the index registers for the Dicke state in a similar fashion as in the W-state protocol. In the cleaning process, we have to take the added ordering of the index registers into account. Suppose the l-th qubit of the system register is a 1. If this is the first 1 in the system register, it belongs to index register j_1, and if it is the m-th 1 it belongs to index register j_m. Computing the Hamming weight of the first $l-1$ qubits gives precisely this information. Combined, this shows that if the l-th qubit is a 1 and the Hamming weight of the first $l-1$ qubits equals m, then the l-th qubit should uncompute the $m+1$-th index register j_{m+1}.

Lemma 4.9. *There exists a circuit in* LAQCC *implementing* ***Cleaning****. More precisely implementing the map* $\frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1<\cdots<j_k}^{n-1} |j_1\rangle \ldots |j_k\rangle \, |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle \to \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1<\cdots<j_k}^{n-1} |0\rangle \ldots |0\rangle \, |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle$. *This circuit requires* $\mathcal{O}(n^2 \log(n))$ *qubits.*

Proof. The first step, as described above, is to aquire the Hamming weight from all the substrings of the system register. This requires n copies of the system register as well as a $\log(k)$ register to store the hamming weight value, which are created using the fanout gate.

$$\binom{n}{k}^{-1/2} \sum_{j_1<\cdots<j_k} |j_1\rangle \ldots |j_k\rangle \, |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle \, |0\rangle_n^{\otimes n-1} \, |0\rangle_{\log(n)}^{\otimes n} \xrightarrow{(1)}$$

$$\binom{n}{k}^{-1/2} \sum_{j_1<\cdots<j_k} |j_1\rangle \ldots |j_k\rangle \, |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle^{\otimes n} \, |0\rangle_{\log(n)}^{\otimes n} \xrightarrow{(2)}$$

$$\binom{n}{k}^{-1/2} \sum_{j_1<\cdots<j_k} |j_1\rangle \ldots |j_k\rangle \, |e_{j_1} \oplus \cdots \oplus e_{j_k}\rangle^{\otimes n} \bigotimes_{l=0}^{n-1} \left| |(e_{j_1} \oplus \cdots \oplus e_{j_k})_{[l,n]}| \right\rangle ,$$

where $|(e_{j_1} \oplus \cdots \oplus e_{j_k})_{[l,n]}|$ denotes the Hamming weight of the substring consisting of qubits l up until n of the system register. Step (1) copies the system qubits using fan-out gates; Step (2) computes the Hamming weight of all the qubits 1 up until $j-1$ using the Hammingweight-gate shown in Table 4; Step (3) clean the copies of the system register by applying fan-out. Step(3) is not shown in the equations but Figure 7 shows graphically how to compute the Hamming weight in parallel for $n = 4$ including Step(3). Note that at the end of the calculation, it is convenient to teleport the Hamming weight registers next to the system register. There are now n new registers containing the information of the Hamming weight, we will refer to them as the Hamming weight registers. This step requires $\mathcal{O}(n^2 \log(n))$ qubits.

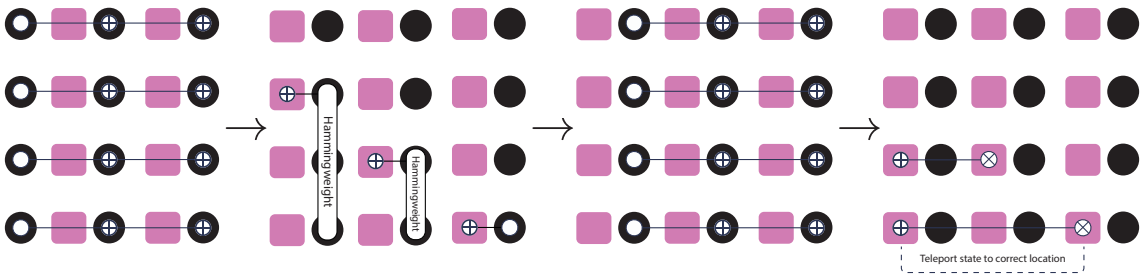

Figure 7: Circuit to implement of the Hamming weight calculation of all qubit strings l to $n-1$ in four steps in parallel. The black dots represent system qubits, the pink squares represent $\log(k)$ qubit registers that can count the Hamming weight up until k.

The last step that remains is to clean the k index registers. Cleaning the k index registers follows similar steps as the **Compress** method in the W-state protocol, with the added Hamming weight information taken into account. This step requires k copies of the system registers well as k copies of the Hamming weight registers. Every index register is paired with one copy of the system register and a copy of the n Hamming weight registers. Cleaning the j-th register consists of five steps, similar to the **Compress** method of the W-state: Step (1) applies Hadamard gates to bring the index register to phase space, in which $CNOT$-gates are diagonalized; Step (2) copies the index register; Step (3) uses the information in the Hamming weight and system register to apply the phases to the correct index register qubits; Step (4) cleans the index register copies; and, Step (5) applies Hadamard gates to reset the index register qubits to the $|0\rangle$ state

Figure 8 shows the steps taken to clean a single index register j. The black dots represent the qubits in the system register and the upper row of blue dots represent the qubits in index register j. The pink squares represent the ancilla Hamming weight register, where each square represents a group of $\log(k)$ qubits. This step requires $\mathcal{O}(nk\log(k)\log(n))$ qubits. At the end of the cleanup the state is as desired:

$$\frac{1}{\sqrt{\binom{n}{k}}}\sum_{j_1<\cdots<j_k}^{n-1}|0\rangle\ldots|0\rangle\,|e_{j_1}\oplus\cdots\oplus e_{j_k}\rangle\,.$$

Note that ancilla qubits used in one step can be reused in a later one, as they are cleaned at the end of every step. We are only interested in the most expensive step in the qubit cost count. The **Cleaning** step thus requires $\mathcal{O}(n^2\log(n))$ qubits. □

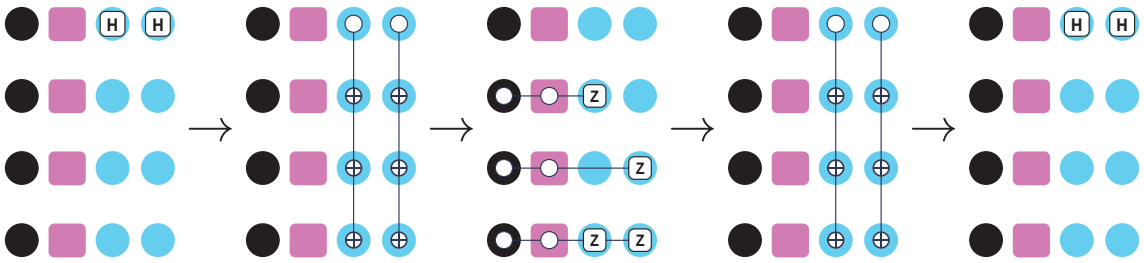

Figure 8: Circuit to clean index register j. The black dots represent qubits in the system register and the blue dots the index register and its copies. The pink squares represent the ancilla Hamming weight register and its copies. Each pink square represents a group of $\log(k)$ qubits. Each of the five grids represents a single timeslice in the circuit.

Theorem 4.10. *For any n and $k=\mathcal{O}(\sqrt{n})$ there exists a* LAQCC *circuit preparing the Dicke-(n,k) state, $|D_k^n\rangle$, using $\mathcal{O}(n^2\log(n))$ qubits.*

Proof. The circuit combines the circuits resulting from Lemmas 4.6, 4.7, 4.8 and 4.9. It consists of four steps:

$$
\begin{aligned}
|0\rangle_{i_1} \dots |0\rangle_{i_k} |0\rangle_{s_1} &\xrightarrow{(1)} \frac{1}{n^{k/2}} \sum_{j_1,\dots,j_k=0}^{n-1} |j_1\rangle_{i_1} \dots |j_k\rangle_{i_k} |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle_{s_1} \\
&\xrightarrow{(2)} \sqrt{\frac{(n-k)!}{n!}} \sum_{j_1 \neq \dots \neq j_k}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \\
&\xrightarrow{(3)} \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1 < \dots < j_k}^{n-1} |j_1\rangle \dots |j_k\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle \\
&\xrightarrow{(4)} \frac{1}{\sqrt{\binom{n}{k}}} \sum_{j_1 < \dots < j_k}^{n-1} |0\rangle \dots |0\rangle |e_{j_1} \oplus \dots \oplus e_{j_k}\rangle
\end{aligned}
$$

Step(1) implements **Filling** using Lemma 4.6 requiring $\mathcal{O}(kn\log(n))$ qubits; Step(2) implements **Filtering** using Lemma 4.7 requiring $\mathcal{O}(kn\log(n))$ qubits; Step(3) implements **Ordering** using Lemma 4.8 requiring $\mathcal{O}(k^2\log(n)^2)$ qubits; Step(4) implements **Cleaning** using Lemma 4.9 requiring $\mathcal{O}(n^2\log(n))$ qubits. After every step ancilla qubits are cleaned, so that they can be reused. As $k = \mathcal{O}(\sqrt{n})$ the largest amount of qubits required for a step is Step(4) requiring $\mathcal{O}(n^2\log(n))$ qubits. □

Bärtschi and Eidenbenz posed a conjecture on the optimal depth of quantum circuits that prepare the Dicke-(n,k) state[6]. They give an algorithm for generating Dicke-(n,k) states in depth $\mathcal{O}(k\log(\frac{n}{k}))$, given all-to-all connectivity, and conjecture that this scaling is optimal when k is constant. Our result shows that there is a LAQCC implementation in this regime, when one has access to intermediate measurements and feed forward. This does not disprove their conjecture. However the circuits shown here are also accessible in QNC^1 by Lemma 3.19, giving "pure" quantum circuits with depth $\mathcal{O}(\log(n))$ for $k = \mathcal{O}(\sqrt{n})$ achieving better scaling when $k = \omega(1)$.

4.4 Dicke states for all k using log-depth quantum circuits

The previous section gave a constant-depth protocol to prepare the Dicke-(n,k) state for $k = \mathcal{O}(\sqrt{n})$. We developed a different method for creating Dicke-(n,k) states which requires logarithmic (in n independend of k) quantum depth to prepare Dicke-(n,k), but works for arbitrary k. We first define what we mean with logarithmic quantum depth:

Notation 4.11. We let $\mathsf{LAQCC}-\mathsf{LOG}$ refer to the instance $\mathsf{LAQCC}(\mathsf{QNC}^0, \mathsf{NC}^1, \mathcal{O}(\log(n)))$, similar to LAQCC except that we allow for a logarithmic number of alterations between quantum and classical calculations. This results in a circuit of logarithmic quantum depth.

In this section we show a $\mathsf{LAQCC}-\mathsf{LOG}$ circuit that creates the Dicke-(n,k) state.

One way of studying the creation of Dicke states is by looking at efficient algorithms that convert numbers from one representation to another. An example of this is the **Uncompress-Compress** method in the W-state protocol, that converts numbers from a binary representation to a one-hot representation. Dicke states are a generalization of the W-state, hence the one-hot representation no longer suffices for preparing the state. Instead, we use a construction based on number conversion between the combinatorial representation and the factoradic representation. Below we introduce both circuits and present quantum circuits that map between the two. Theorem 4.21 proves that a log-depth quantum circuit with additional log-depth classical computations can prepare the Dicke-(n,k) state for any k.

4.4.1 Combinatorial number system

An interesting result showed that any integer $m \geq 0$ can be written as a sum of k binomial coefficients [7]. For fixed k, this is even unique as the next lemma shows.

Lemma 4.12 ([7]). *For all integers $m \geq 0$ and $k \geq 1$, there exists a unique decreasing sequence of integers $c_k, c_{k-1} \dots c_1$ with $c_j > c_{j-1}$ and $c_1 \geq 0$ such that*

$$m = \binom{c_k}{k} + \binom{c_{k-1}}{k-1} \dots \binom{c_1}{1} = \sum_{i=1}^{k} \binom{c_i}{i}.$$

This lemma allows for the definition of the combinatorial number representation:

Definition 4.13. Let $k \in \mathbb{N}$ be a constant. Any integer $m \in \mathbb{N}$ can be represent by a unique string of numbers $(c_k, c_{k-1} \dots, c_1)$, such that $c_k > c_{k-1} \cdots > c_1 \geq 0$ and $c_k \leq m$. This string is given by the unique decreasing sequence of Lemma 4.12. We call this string the *index representation* denoted by $m^{indx(k)}$.

The bit string of k ones at indices $(c_k, \dots, c_1)$ is the m-th bit string with k ones in the lexicographical order. This bit string is called the *combinatatorial representation*. We denote the m-th bit string with k ones as $m^{comb(k)}$.

The W-state protocol used the conversion between the binary representation of a number m and its combinatorial representation $m^{comb(1)}$. A generalized number conversion is precisely the protocol needed to prepare Dicke states.

A sketch of the protocol would be as follows: given positive integers k and n: Create a superposition state

$$\binom{n}{k}^{-\frac{1}{2}} \sum_{i=0}^{\binom{n}{k}-1} |i\rangle |0\rangle ;$$

Use number conversion to go from label i to $i^{comb(k)}$

$$\binom{n}{k}^{-\frac{1}{2}} \sum_{i=0}^{\binom{n}{k}-1} |i\rangle \left|i^{comb(k)}\right\rangle ;$$

Use number conversion from $i^{comb(k)}$ to i to clean up the label register

$$\binom{n}{k}^{-\frac{1}{2}} \sum_{i=0}^{\binom{n}{k}-1} |0\rangle \left|i^{comb(k)}\right\rangle = |D_k^n\rangle .$$

The conversion map from the combinatorial representation to the binary representation is given by Lemma 4.12. This calculation requires iterative multiplication and addition, both of which are in TC^0, hence this calculation is in TC^0.

The converse mapping, from binary to combinatorial representation for given k, can be achieved by a greedy iterative algorithm: On input m, find the biggest c_k such that $m \geq \binom{c_k}{k}$ and subtract this from m: $\tilde{m} = m - \binom{c_k}{k}$. This gives c_k and a residual $\tilde{m}$. Repeat this process for $\tilde{m}$: Find the largest c_j such that $\tilde{m} \geq \binom{c_j}{j}$ and update residual $\tilde{m} = \tilde{m} - \binom{c_j}{j}$, until all c_j are found.

This greedy algorithm is inherently linear in k as it requires all previously found $\{c_i\}_{i=j}^{k}$ to find c_{j-1}. Hence, it is not immediately obvious if and how to achieve this mapping in constant or even logarithmic depth.

4.4.2 Mapping between factoradic representation and combinatorial number system

A number representation closely related to the combinatorial number representation is the *factoradic representation*. This number system uses factorials instead of binomials to represent numbers.

Definition 4.14. A sequence $y = (y_{n-1}, y_{n-2}, \ldots, y_0)$ of integers, such that $j \geq y_j \geq 0$ is called a *factoradic*, or more explicitly an *n-factoradic*. The elements of an n-factoradic is called an *n-digit*. An n-factoradic y can represent a number m between 0 and $n! - 1$, in the following way

$$m = \sum_{j}^{n} y_j \cdot j!. \tag{5}$$

For a given $m \in \{0, \ldots, n!-1\}$, we call the n-factoradic y obeying the equality above, the *factoradic representation of* m. Denote $\mathrm{Fact}(n)$ as the set of all n-factoradics.

The following lemma shows that Equation 5 is a bijection, showing that the factoradic representation is unique.

Lemma 4.15. *For $k \geq 0$ it holds that:*

$$\sum_{i=0}^{k} i \cdot i! = (k+1)! - 1.$$

Proof. Proof by induction.
BASE STEP: Let k be 0:

$$0 \cdot 0! = 1! - 1$$

INDUCTION STEP: Assume the lemma holds for some j, then

$$\sum_{i=0}^{j+1} i \cdot i! = (j+1) \cdot (j+1)! + \sum_{i=0}^{j} i \cdot i = (j+1) \cdot (j+1)! + (j+1)! - 1 = (j+2)! - 1,$$

which completes the proof. □

This identity allows for using factorials as a base for a number system. The next lemma gives a log-space algorithm to convert a factoradic representation to its combinatorial representation.

Lemma 4.16. *There is a logspace algorithm $\mathcal{A}$ that, given $k \in \{0, \ldots, n\}$, and a uniformly random n-factoradic, outputs a uniformly random n-bit string of Hamming weight k.*

Proof. The algorithm $\mathcal{A}$ is given k and an n-factoradic $y = (y_{n-1}, \ldots, y_0)$. It will then output an n-bit string $y^{comb(k)} = y_{n-1}^{comb(k)} \ldots y_0^{comb(k)} \in \{0,1\}^n$ of Hamming weight k, one bit at a time, from left to right, according to the following rule. Let $H_{>n-j} = \sum_{i=n-j+1}^{n-1} y_i^{comb(k)}$ be the Hamming weight of the bits produced before we reach bit $n-j$. Then $y_{n-j}^{comb(k)}$ is given by:

$$(\mathcal{A}(y))_{n-j} = y_{n-j}^{comb(k)} = \begin{cases} 1 & \text{if } y_{n-j} < k - H_{>n-j} \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

This conversion requires comparing an n-digit with a constant and the Hamming weight of a bitstring. The only information that $\mathcal{A}$ needs to remember, as it goes from bit $n-j+1$ to bit $n-j$, is the Hamming weight $H_{>n-j}$ of the bits it produced so far, and this can be stored in logarithmic space.

Now note that the number of factoradic n-digit strings that map to the same combinatorial bit string is always $k!(n-k)!$: Let $y^{comb(k)} \in \{0,1\}^n$ have Hamming weight k. For any bit position $y_{n-j}^{comb(k)}$, there are $n-j+1-(k-H_{>n-j})$ possible choices for the n-digit $y_{n-j} \in \{0, \ldots, n-j\}$ that set $y_{n-j}^{comb(k)} = 0$. For the leftmost index $n-j$ such that $y_{n-j}^{comb(k)} = 0$, it holds that $H_{>n-j} = j-1$, and then there are $n-k$ possible n-digits y_{n-j} that set $y_{n-j}^{comb(k)} = 0$. Then, for the second index $n-j$ such that $y_{n-j}^{comb(k)} = 0$ it holds that $H_{>n-j} = j-2$, hence there are $n-k-1$ possible n-digits

y_{n-j} causing $y_{n-j}^{comb(k)} = 0$. And so forth. This results in $(n-k)!$ different possible choices for the $(n-k)$-many n-digits where $y^{comb(k)} = 0$.

Similarly, for the leftmost position $n-j$ where $y_{n-j}^{comb(k)} = 1$, there are k possible choices for the n-digit y_{n-j} that cause $y_{n-j}^{comb(k)} = 1$. The second leftmost position $n-j$ gives $k-1$ possible choices, and so forth, for a total of $k!$ possible settings of the k-many n-digits where $y^{comb(k)} = 0$.

Combined, we conclude that, for every n-bit string $y^{comb(k)} \in \{0,1\}^n$ of Hamming weight k, there are exactly (the same number of) $k!(n-k)!$ n-factoradics y such that $\mathcal{A}(y) = y^{comb(k)}$. Hence, a uniformly random n-factoradic is mapped by $\mathcal{A}$ to a uniformly random n-bit string of Hamming weight k, as claimed. □

This lemma gives a logspace algorithm to convert a uniformly random n-factoradic to a uniformly random n-bit string of Hamming weight k, for any k. It is well known that logspace is contained in TC^1, allowing this calculation to be performed in parallel log-depth when one has access to threshold gates [25]. As we saw in Section 3.3, we can compute a threshold gate in LAQCC. Hence, an $\mathsf{LAQCC}-\mathsf{LOG}$ can perform any TC^1 calculation. We conclude:

Corollary 4.17. *The following map can be implemented in* $\mathsf{LAQCC}-\mathsf{LOG}$.

$$\frac{1}{\sqrt{n!}} \sum_{y \in Fact(n)} |y\rangle |0\rangle \to \frac{1}{\sqrt{n!}} \sum_{y \in Fact(n)} |y\rangle |\mathcal{A}(y)\rangle .$$

In the next lemma, we show that a TC^0 circuit can implement the inverse of $\mathcal{A}$.

Lemma 4.18. *There exists a* TC^0 *algorithm which, when given an n-bit string $y^{comb(k)}$ of Hamming weight k, a uniformly-random k-factoradic, and a uniformly-random $(n-k)$-factoradic, outputs a uniformly random n-factoradic y among those such that $\mathcal{A}(y) = y^{comb(k)}$.*

Proof. The conversion can be done in parallel, generating an n-digit for every bit in $y^{comb(k)} = y_{n-1} \dots y_0 \in \{0,1\}^n$. Recall that we are given as input a uniformly-random k-factoradic $O_{k-1} \dots O_0$ and a uniformly-random $(n-k)$-factoradic $Z_{n-k-1} \dots Z_0$.

For every bit position $n-j$, for $1 \le j \le n$, calculate the Hamming weight of the bits from $n-j+1$ to $n-1$: $H_{>n-j} = \sum_{i=j+1}^{n-1} y_i^{comb(k)}$. Recall that iterated addition is in TC^0 [53].

If $y_{n-j}^{comb(k)} = 1$, set $y_{n-j} = O_{k-H_{>n-j}}$. This gives us a uniform random n-digit between 0 and $k - H_{>n-j} - 1$. If $y_{n-j}^{comb(k)} = 0$, set $y_{n-j} = k - H_{>n-j} + Z_{n-k-H_{>n-j}}$. Note that this gives us a uniform random n-digit between $k - H_{>n-j}$ and $n-j$. By construction, it now follows that $\mathcal{A}(y) = y^{comb(k)}$. Computing each n-digit in this way requires summation and indexing, both of which are in $\mathsf{AC}^0 \subseteq \mathsf{TC}^0$ [53]. □

Remark 4.19. *The above algorithm establishes a bijection $(y^{comb(k)}, Z, O) \leftrightarrow y$ between triples $(y^{comb(k)}, Z, O)$ with $y^{comb(k)} \in \{0,1\}^n$ of Hamming weight k, $Z \in Fact(n-k)$ and $O \in Fact(k)$ and n-factoradics $y \in Fact(n)$. Let $(\mathcal{A}(y), \mathcal{Z}(y), \mathcal{O}(y))$ be the image of an n-factoradic y under this bijection. The previous lemma shows that one can compute y from $(y^{comb(k)}, Z, O)$ in* TC^0. *It is not hard to see that the map $(\mathcal{A}(y), y) \mapsto (\mathcal{A}(y), y, \mathcal{Z}(y), \mathcal{O}(y))$ is also in* TC^0. *Indeed, to find $\mathcal{Z}(y)$ and $\mathcal{O}(y)$, we need only invert the two defining equalities $y_{n-j} = O_{k-H_{>n-j}}$ and $y_{n-j} = k - H_{>n-j} + Z_{n-k-H_{>n-j}}$.*

Corollary 4.20. *The following map can be implemented in* LAQCC.

$$\binom{n}{k}^{-\frac{1}{2}} \sum_{y^{comb(k)}} |0\rangle \left| y^{comb(k)} \right\rangle \to \frac{1}{\sqrt{n!}} \sum_{y \in Fact(n)} |y\rangle |\mathcal{A}(y)\rangle$$

where $y^{comb(k)}$ ranges over all n-bit strings of Hamming weight k.

Proof. The transformation consists of three steps:

$$
\begin{aligned}
& \binom{n}{k}^{-\frac{1}{2}} \sum_{y^{comb(k)}} \left| y^{comb(k)} \right\rangle |0\rangle |0\rangle |0\rangle \\
\xrightarrow{(1)} \; & \binom{n}{k}^{-\frac{1}{2}} \sum_{y^{comb(k)}} \left| y^{comb(k)} \right\rangle \left(\bigotimes_{j=0}^{n-k-1} \sum_{i=0}^{j} |i\rangle \right) \left(\bigotimes_{j=0}^{k-1} \sum_{i=0}^{j} |i\rangle \right) |0\rangle \\
= \; & \frac{1}{\sqrt{n!}} \sum_{y^{comb(k)}} \left| y^{comb(k)} \right\rangle \left(\sum_{Z \in \mathrm{Fact}(n-k)} |Z\rangle \right) \left(\sum_{O \in \mathrm{Fact}(k)} |O\rangle \right) |0\rangle \\
\xrightarrow{(2)} \; & \frac{1}{\sqrt{n!}} \sum_{y \in \mathrm{Fact}(n)} |\mathcal{A}(y)\rangle \left| \hat{Z}(y) \right\rangle \left| \hat{O}(y) \right\rangle |y\rangle \\
\xrightarrow{(3)} \; & \frac{1}{\sqrt{n!}} \sum_{y \in \mathrm{Fact}(n)} |\mathcal{A}(y)\rangle |0\rangle |0\rangle |y\rangle
\end{aligned}
$$

Step (1) prepares a uniform superposition over all n-factoradics using Theorem 4.1. Step (2) is Lemma 4.18, and Step (3) follows from Remark 4.19. In the above steps we implicitly used that the inverse of the used LAQCC operations are also LAQCC operations. Even though it is unclear if this inverse-property holds in general, it does hold for the considered LAQCC operations. The measurement steps, which might not be reversible, in this algorithm are used to implement fan-out gates. The inverse of a fan-out gate is the fan-out gate itself and hence is contained in LAQCC. □

Theorem 4.21. *There exists a* LAQCC$-$LOG*-circuit for preparing Dicke-(n,k) states, for any natural numbers n and $k \leq n$, it use $\mathcal{O}(poly(n))$ qubits.*

Proof. The circuit combines the circuits resulting from Lemma 4.16 and Lemma 4.18.

It consists of three steps:

$$
\begin{aligned}
|0\rangle^{\otimes n \log(n)} |0\rangle^{\otimes n} \xrightarrow{(1)} & \frac{1}{\sqrt{n!}} \left(\bigotimes_{j=0}^{n-1} \sum_{i=0}^{j} |i\rangle \right) |0\rangle^{\otimes n} = \sum_{y \in \mathrm{Fact}(n)} |y\rangle |0\rangle \\
\xrightarrow{(2)} & \frac{1}{\sqrt{n!}} \sum_{y \in \mathrm{Fact}(n)} |y\rangle |\mathcal{A}(y)\rangle \\
\xrightarrow{(3)} & \binom{n}{k}^{-\frac{1}{2}} \sum_{y \in \mathrm{Fact}(n)} |0\rangle |\mathcal{A}(y)\rangle = |D_k^n\rangle .
\end{aligned}
$$

Step (1) prepares a uniform superposition over all n-factoradics using Theorem 4.1; Step (2) is by Corollary 4.17; and, Step (3) reverses the algorithm of Corollary 4.20. □

5 Complexity results for LAQCC$(\mathcal{Q}, \mathcal{C}, d)$

Up until now we have considered the class LAQCC. In this section, we investigate the state-preparation complexity of LAQCC$(\mathcal{Q}, \mathcal{C}, d)$ when we increase the power of the quantum circuits to polynomial depth, and we allow for unbounded classical computational power.

Notation 5.1. The class LAQCC* is the instantiation LAQCC(QPoly(n), ALL, poly(n)): The class of polynomially many alternating polynomial-sized quantum circuits and arbitrary powerful classical computations, together with feed-forward of the classical information to future quantum operations. The quantum computations are restricted to all single-qubit gates and the two-qubit CNOT gate.

Recall the definition of State-classes:

Definition 3.4. Let $\mathcal{H}_n$ be a Hilbert space on n qubits, then define

$$\mathsf{StateX}_{n,\varepsilon} = \{|\psi\rangle \in \mathcal{H}_n \mid \exists\, \mathsf{X}\text{-circuit } A : \langle A\,|0\rangle^{\otimes n}, |\psi\rangle\rangle \geq 1-\varepsilon\}.$$

This is the subset of n-qubit states $|\psi\rangle$ such that there exists a circuit corresponding to the class X that prepares a quantum state that has inner product at least $1-\varepsilon$ with $|\psi\rangle$.

Define $\mathsf{StateX}_{\varepsilon} = \bigcup_{n\in\mathbb{N}} \mathsf{StateX}_{n,\varepsilon}$.

From which we directly have a definition for the class $\mathsf{StateLAQCC}^*_{\varepsilon}$ for any $\varepsilon > 0$.

Remark 5.2. *Note that for any non-zero ε, we can restrict ourselves to finite universal gate-sets. The Solovay-Kitaev theorem [27, 35] says that any multi-qubit unitary can be approximated to within precision δ by a quantum circuit with size depending on δ. Therefore, with a finite universal gate-set, any* LAQCC* *circuit with a continuous gate-set can be approximated by an* LAQCC* *circuit with gates from the finite set.*

To upper bound the state preparation complexity of LAQCC*, we compare it to the class PostQPoly and its circuit variant.

Definition 3.6. The class PostQPoly consists of all polynomial-sized quantum circuits with one extra qubit, where the outcome state is considered conditional on the extra qubit being in the one state. If the extra qubit is in the zero state, the output state may be anything.

The class $\mathsf{StatePostQPoly}_{n,\varepsilon}$ consists of all n-qubit states $|\psi\rangle$ for which a polynomial-sized quantum circuit exists that prepares a state that, conditional on the extra qubit being one, has inner product at least $1-\varepsilon$ with $|\psi\rangle$.

Next, we prove $\mathsf{StateLAQCC}^*_{\varepsilon} \subseteq \mathsf{StatePostQPoly}_{\varepsilon}$. Section 3.5 describes how any LAQCC circuit decomposes in alternating unitaries, measurements and classical calculation layers. A similar decomposition follows for LAQCC* circuits: Any LAQCC* can be written as

$$\Pi_{i=0}^{\mathrm{poly}(n)} M_i U_i(y_i) C_i(x_i)\,|0\rangle^{\otimes \mathrm{poly}(n)}\,, \tag{7}$$

where each $x_i \in \{0,1\}^*$ denotes the measurement outcome of the i-th measurement layer and $y_i \in \{0,1\}^*$ the output bitstring of the i-th classical calculation layer. Note, all x_i and y_i have length at most polynomial in n. The U_i's denote unitary operations that represent a polynomial-deep quantum circuit consisting of single qubit gates and the two-qubit CNOT gate. The M_i's denote a measurement of a subset of the qubits, and the C_i's represent unbounded classical computations, with input x_i.

Theorem 5.3. *It holds that* $\mathsf{StateLAQCC}^*_{\varepsilon} \subseteq \mathsf{StatePostQPoly}_{\varepsilon}$.

Proof. Fix $\varepsilon > 0$ and a positive integer n and let $|\psi\rangle \in \mathsf{StateLAQCC}^*_{\varepsilon}$. By definition, there exists an LAQCC* circuit $A = \Pi_{i=0}^{\mathrm{poly}(n)} M_i U_i(y_i) C_i(x_i)$, which prepares a state $|\phi\rangle$ with inner product at least $1-\varepsilon$ with $|\psi\rangle$.

Then consider the following PostQPoly-circuit: Let $B = \Pi_{i=0}^{\mathrm{poly}(n)} \mathrm{Equal}_{x_i}(x_i) U_i(y_i)\,|0\rangle^{\otimes \mathrm{poly}(n)}$, where the y_i are hardwired. The Equal_{x_i} gate replaces the measurement layer M_i, by checking if the subset of qubits that would be measured are in $|x_i\rangle$ computational basis state. It stores the output in an ancilla qubit. As a last step, apply an $\mathrm{AND}_{\mathrm{poly}(n)}$-gate on the ancilla qubits, which hold the outputs of the Equal_{x_i} gates, and store the result in an ancilla qubit. Conditional on this last ancilla qubit being one, the circuit prepares the state $|\phi\rangle$. □

Figure 9 gives a schematic overview of the proof and the translation of an LAQCC* circuit in a PostQPoly-circuit.

An interesting direction for future work is to extend the inclusion proved above to a true separation or an oracular separation. One approach is to use a similar oracle as used in [3] to

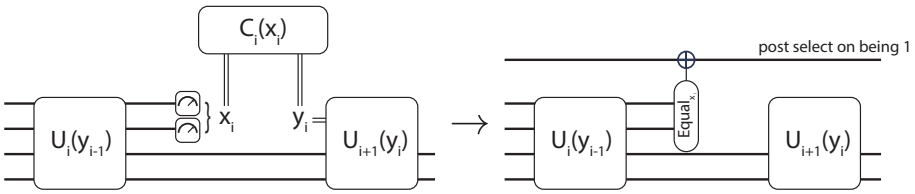

Figure 9: Schematic idea of transforming an LAQCC^* circuit for generating $|\psi\rangle$ into a $\mathsf{PostQPoly}$ circuit.

separate QMA and QCMA with respect to an oracle and use a counting argument to argue that $\mathsf{StateLAQCC}^{*\mathcal{O}}_{\varepsilon} \neq \mathsf{StatePostQPoly}^{\mathcal{O}}_{\varepsilon}$, for some oracle $\mathcal{O}$ and $\varepsilon = 1 - \frac{1}{\mathrm{poly}(n)}$. The LAQCC model allows for a constant number, more than one, of rounds of measurements and corrections. This was required for our three new state generation protocols. However other models considered only one round of measurements and corrections, for instance in the paper [39]. One may wonder if there is a hierarchy of model power allowing one or multiple measurements, and if there is a way to reduce the number of measurements rounds. A starting effort towards classifying types of states based on such a hierarchy can be found in [49]. It would be interesting to see a more extensive complexity theoretic analysis comparing models with different number of allowed rounds.

Acknowledgements

We want to thank Jonas Helsen, Joris Kattemölle, Ido Niesen, Kareljan Schoutens, Florian Speelman, Dyon van Vreumingen and Jordi Weggemans for insight full discussions. Furthermore, we would like to thank Georgios Styliaris for first mentioning how to parallelize Clifford ladder circuits. HB and MF were supported by the Dutch Ministry of Economic Affairs and Climate Policy (EZK), as part of the Quantum Delta NL programme. NN was supported by the quantum application project of TNO. This work was supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037).

References

[1] Scott Aaronson. Multilinear formulas and skepticism of quantum computing. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 118–127, New York, NY, USA, 2004. Association for Computing Machinery.

[2] Scott Aaronson, Yosi Atia, and Leonard Susskind. On the hardness of detecting macroscopic superpositions, 2020.

[3] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 115–128, 2007.

[4] Dave Bacon, Isaac L. Chuang, and Aram W. Harrow. Efficient quantum circuits for schur and clebsch-gordan transforms. *Phys. Rev. Lett.*, 97:170502, 10 2006.

[5] Andreas Bärtschi and Stephan Eidenbenz. Deterministic preparation of dicke states. In *International Symposium on Fundamentals of Computation Theory*, pages 126–139. Springer, 2019.

[6] Andreas Bärtschi and Stephan Eidenbenz. Short-depth circuits for dicke state preparation, 2022.

[7] Edwin F. Beckenbach. *Applied combinatorial mathematics*. New York, J. Wiley, 1964.

[8] Alla V. Bezvershenko, Catalin-Mihai Halati, Ameneh Sheikhan, Corinna Kollath, and Achim Rosch. Dicke transition in open many-body systems determined by fluctuation effects. *Phys. Rev. Lett.*, 127:173606, 10 2021.

[9] Tobias Brandes. Excited-state quantum phase transitions in dicke superradiance models. *Phys. Rev. E*, 88:032133, 9 2013.

[10] Sebastian Brandhofer, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens. Benchmarking the performance of portfolio optimization with QAOA. *Quantum Information Processing*, 22(1), 12 2022.

[11] Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2126):459–472, August 2010.

[12] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum*, 1:8, April 2017.

[13] Sean Clark, Richard Jozsa, and Noah Linden. Generalised clifford groups and simulation of associated quantum circuits. *arXiv preprint quant-ph/0701103*, 2007.

[14] Jeremy Cook, Stephan Eidenbenz, and Andreas Bärtschi. The quantum alternating operator ansatz on maximum k-vertex cover. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 83–92. IEEE, 2020.

[15] R. H. Dicke. Coherence in spontaneous radiation processes. *Phys. Rev.*, 93:99–110, 1 1954.

[16] W. Dür, G. Vidal, and J. I. Cirac. Three qubits can be entangled in two inequivalent ways. *Phys. Rev. A*, 62:062314, 11 2000.

[17] Guillermo García-Pérez, Oskari Kerppo, Matteo A. C. Rossi, and Sabrina Maniscalco. Experimentally accessible non-separability criteria for multipartite entanglement structure detection, 2021.

[18] Daniel Gottesman. The Heisenberg Representation of Quantum Computers. *arXiv e-prints*, pages quant–ph/9807006, 7 1998.

[19] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, 1999.

[20] Frederic Green, Steven Homer, Cristopher Moore, and Christopher Pollett. Counting, fanout and the complexity of quantum *ACC*. *Quantum Info. Comput.*, 2(1):35–65, 12 2002.

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

[22] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, February 2019.

[23] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek al kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt. Scalable multiparticle entanglement of trapped ions. *Nature*, 438(7068):643–646, 12 2005.

[24] Peter Høyer and Robert Špalek. Quantum fan-out is powerful. *Theory of Computing*, 1(5):81–103, 2005.

[25] David S. Johnson. A catalog of complexity classes. In *Algorithms and Complexity*, pages 67–161. Elsevier, 1990.

[26] Richard Jozsa. An introduction to measurement based quantum computation. *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment*, 199:137–158, 2006.

[27] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, December 1997.

[28] G. L. Long. Grover algorithm with zero theoretical failure rate. *Phys. Rev. A*, 64:022307, 7 2001.

[29] Dmitri Maslov and Martin Roetteler. Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory*, 64(7):4729–4738, 2018.

[30] Tony Metger and Henry S. Yuen. stateqip = statepspace. *ArXiv*, abs/2301.07730, 2023.

[31] Cristopher Moore. Quantum circuits: Fanout, parity, and counting. *arXiv preprint arXiv:quant-ph/9903046*, 1999.

[32] Yoshifumi Nakata and Mio Murao. Diagonal quantum circuits: Their computational power and applications. *The European Physical Journal Plus*, 129(7), jul 2014.

[33] Matthew Neeley, Radoslaw C. Bialczak, M. Lenander, E. Lucero, Matteo Mariantoni, A. D. O'Connell, D. Sank, H. Wang, M. Weides, J. Wenner, Y. Yin, T. Yamamoto, A. N. Cleland, and John M. Martinis. Generation of three-qubit entangled states using superconducting phase qubits. *Nature*, 467(7315):570–573, September 2010.

[34] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[35] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, June 2012.

[36] Yingkai Ouyang. Permutation-invariant quantum codes. *Physical Review A*, 90(6):062317, 2014.

[37] S K Özdemir, J Shimamura, and N Imoto. A necessary and sufficient condition to play games in quantum mechanical settings. *New Journal of Physics*, 9(2):43–43, February 2007.

[38] Paul Pham and Krysta M. Svore. A 2d nearest-neighbor quantum architecture for factoring. *Quantum Information and Computation*, 13:937–962, 7 2013.

[39] Lorenzo Piroli, Georgios Styliaris, and J. Ignacio Cirac. Quantum circuits assisted by local operations and classical communication: Transformations and phases of matter. *Phys. Rev. Lett.*, 127:220503, 11 2021.

[40] Martin Plesch and Vladimír Bužek. Efficient compression of quantum information. *Phys. Rev. A*, 81:032317, 3 2010.

[41] Robert Prevedel, Gunther Cronenberg, Mark S Tame, Mauro Paternostro, Philip Walther, Mu-Seong Kim, and Anton Zeilinger. Experimental realization of dicke states of up to six qubits for multiparty quantum networking. *Physical review letters*, 103(2):020503, 2009.

[42] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001.

[43] Gregory Rosenthal and Henry S. Yuen. Interactive Proofs for Synthesizing Quantum States and Unitaries. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 112:1–112:4, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[44] Dan J. Shepherd and Michael J. Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465:1413 – 1439, 2009.

[45] Leonard Susskind. Three lectures on complexity and black holes, 2018.

[46] Yasuhiro Takahashi and Seiichiro Tani. Collapse of the hierarchy of constant-depth exact quantum circuits. In *2013 IEEE Conference on Computational Complexity*, pages 168–178, 2013.

[47] Nathanan Tantivasadakarn, Ryan Thorngren, Ashvin Vishwanath, and Ruben Verresen. Long-range entanglement from measuring symmetry-protected topological phases. *arXiv preprint arXiv:2112.01519*, 2021.

[48] Nathanan Tantivasadakarn, Ruben Verresen, and Ashvin Vishwanath. The shortest route to non-abelian topological order on a quantum processor, 2022.

[49] Nathanan Tantivasadakarn, Ashvin Vishwanath, and Ruben Verresen. Hierarchy of topological order from finite-depth unitaries, measurement, and feedforward. *PRX Quantum*, 4(2), jun 2023.

[50] Géza Tóth. Multipartite entanglement and high-precision metrology. *Phys. Rev. A*, 85:022322, 2 2012.

[51] Oleksandr Tsyplyatyev, Jan von Delft, and Daniel Loss. Simplified derivation of the bethe-ansatz equations for the dicke model. *Phys. Rev. B*, 82:092203, 9 2010.

[52] John S Van Dyke, George S Barron, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. Preparing bethe ansatz eigenstates on a quantum computer. *PRX Quantum*, 2(4):040329, 2021.

[53] Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer, 1999.

[54] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, page 515–526. ACM, 6 2019.

A Useful lemmas

This section gives two lemmas. The first lemma upper bounds the computational power of LAQCC. The second lemma helps in preparing Dicke states for $k \in \mathcal{O}(\sqrt{n})$.

Lemma A.1. *Let $\Pi = (\Pi_{yes}, \Pi_{no})$ be a decision problem in NC^1. Then there is a uniform log-depth quantum circuit that decides on Π.*

Proof. Let B be the uniform Boolean circuit of logarithmic depth deciding on Π. As $\Pi \in \mathsf{NC}^1$, such a circuit exists.

For fixed input size n, write B as a Boolean tree of depth $\mathcal{O}(\log(n))$, with at its leaves the n input bits x_i and as root an output bit. This Boolean tree directly translates in a classical circuit using layers of AND, OR and NOT gates.

Each of these gates has a direct quantum equivalent gate, provided that we use ancilla qubits: First replace all OR gates by AND and NOT gates. Then replace all AND gates by Tofolli gates, which has three inputs. The third input is a clean ancilla qubit and will store the AND of the other two inputs. Finally, replace all NOT gates by X-gates. □

Lemma A.2. *Let $n, k \in \mathbb{N}$ and $k < \frac{n}{2}$ then:*

$$\frac{n!}{n^k(n-k)!} > e^{-\frac{2k^2}{n}}$$

Proof. The result follows by a simple computation

$$\begin{aligned}
\frac{n!}{n^k(n-k)!} &= e^{\sum_{i=1}^{k} \log(1-\frac{i}{n})} \\
&> e^{\sum_{i=1}^{k} \frac{-i}{n-i}} \\
&> e^{\sum_{i=1}^{k} \frac{-i}{n-k}} \\
&> e^{-\frac{k^2}{n-k}} \\
&> e^{-\frac{2k^2}{n}},
\end{aligned}$$

where we use that $\log(1+x) \geq \frac{x}{1+x}$ for $x > -1$.

□

B Gate implementations

B.1 Equality-gate

Define the Equality gate on two n-qubit computational basis states as

$$\text{Equality} : |x\rangle |y\rangle |0\rangle \mapsto |x\rangle |y\rangle |\mathbb{1}_{x=y}\rangle .$$

This gate is implemented in three steps: (1) subtract the first register from the second using a subtraction circuit; (2) apply Equal_0 on the second register and store the result in the third register; (3) add the first register to the second, undoing the subtraction computation:

$$\begin{aligned}
|x\rangle |y\rangle |0\rangle &\xrightarrow[(1)]{} |x\rangle |y-x\rangle |0\rangle \\
&\xrightarrow[(2)]{} |x\rangle |y-x\rangle |\mathbb{1}_{x=y}\rangle \\
&\xrightarrow[(3)]{} |x\rangle |y\rangle |\mathbb{1}_{x=y}\rangle
\end{aligned}$$

Addition and subtraction both have width $\mathcal{O}(n^2)$, which, as a result, the Equality-gate also has.

B.2 Greaterthan-gate

Define the Greaterthan gate on two n-qubit computational basis states as

$$\text{Greaterthan} : |x\rangle |y\rangle |0\rangle \mapsto |x\rangle |y\rangle |\mathbb{1}_{x>y}\rangle .$$

This gate is implemented in four step: (1) Add an extra clean qubit to the second register and interpret this as an $n+1$-qubit register with most significant bit zero; (2) subtract the first register from the second. The subtraction is modulo 2^{n+1}; (3) apply a CNOT-gate from most significant

bit of the second register to the third register; (4) add the first register to the second, undoing the subtraction computation:

$$\begin{aligned}
|x\rangle|y\rangle|0\rangle &\xrightarrow[(1)]{} |x\rangle|0y\rangle|0\rangle \\
&\xrightarrow[(2)]{} |x\rangle\left|y - x \bmod 2^{n+1}\right\rangle|0\rangle \\
&\xrightarrow[(2)]{} |x\rangle\left|y - x \bmod 2^{n+1}\right\rangle|\mathbb{1}_{x>y}\rangle \\
&\xrightarrow[(3)]{} |x\rangle|0\rangle|y\rangle|\mathbb{1}_{x>y}\rangle
\end{aligned}$$

This construction works, as after step (2), the most significant bit of the second register is one, precisely if x is larger than y.

Addition and subtraction both have width $\mathcal{O}(n^2)$, which, as a result, the Greaterthan-gate also has.

B.3 Exact$_t$-gate

Define the Exact$_t$ gate on an n-qubit computational basis state as

$$\text{Exact}_t : |x\rangle|0\rangle \mapsto |x\rangle\left|\mathbb{1}_{|x|=t}\right\rangle.$$

Here, $|x|$ denotes the Hamming weight of the n-bit string x.

This gate follows by combining the Hammingweight-gate and the Equality-gate: First, compute the Hamming weight of x in an ancilla register and then apply the Equality gate to check that this ancilla register equals t.

Another approach is to modify the circuit for OR slightly. In the OR-reduction step, add a gate $R_Z(-\varphi t)$, which adjusts the angle to be zero precisely if $|x| = t$ (see Theorem 4.6 of [24]). Then apply the circuit for OR and negate the output. The circuit for OR evaluates to zero, precisely if the input had Hamming weight t.

B.4 Threshold$_t$-gate

Define the Threshold$_t$ gate on an n-qubit computational basis state as

$$\text{Threshold}_t : |x\rangle|0\rangle \mapsto |x\rangle\left|\mathbb{1}_{|x|\geq t}\right\rangle.$$

Taking the OR over the outputs of Exact$_j$-gates for all $j \geq t$, gives the Threshold$_t$-gate. An improved implementation with better scaling in t is given in Theorem 2 of [46].

A weighted threshold gate uses weights w_i and evaluates to one precisely if $\sum_i w_i x_i \geq t$. Assume without loss of generality that $\sum_i w_i x_i$ evaluates to an integer. Otherwise, we can use the same ideas, but up to some precision.

Use the same OR-reduction as for the normal threshold gate. Instead of rotations $R_Z(\varphi)$ controlled by x_i, we use rotations $R_Z(w_i\varphi)$ controlled by x_i. This implements the weighted threshold gate.