PAPER • OPEN ACCESS

Compilation of algorithm-specific graph states for quantum circuits

To cite this article: Madhav Krishnan Vijayan *et al* 2024 *Quantum Sci. Technol.* **9** 025005

IOP Publishing *Quantum Sci. Technol.* **9** (2024) 025005 https://doi.org/10.1088/2058-9565/ad1f39

Quantum Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
14 February 2023

REVISED
16 August 2023

ACCEPTED FOR PUBLICATION
16 January 2024

PUBLISHED
12 February 2024

PAPER

Compilation of algorithm-specific graph states for quantum circuits

Madhav Krishnan Vijayan[1,*], **Alexandru Paler**[2], **Jason Gavriel**[1,3], **Casey R Myers**[4,5], **Peter P Rohde**[1] **and Simon J Devitt**[1]

[1] Centre for Quantum Software and Information, University of Technology Sydney, Sydney NSW 2007, Australia
[2] Aalto University, 02150 Espoo, Finland
[3] Centre for Quantum Computation and Communication Technology, Sydney, Australia
[4] School of Computing and Information Systems, Faculty of Engineering and Information Technology, The University of Melbourne, Melbourne VIC 3010, Australia
[5] Silicon Quantum Computing Pty Ltd, Level 2, Newton Building, UNSW Sydney, Kensington NSW 2052, Australia
[*] Author to whom any correspondence should be addressed.

E-mail: madhavkrishnan.vijayan@uts.edu.au

Keywords: graph states, quantum circuit compilation, measurement based quantum computing, ICM

Abstract

We present a quantum circuit compiler that prepares an algorithm-specific graph state from quantum circuits described in high level languages, such as Cirq and Q#. The computation can then be implemented using a series of non-Pauli measurements on this graph state. By compiling the graph state directly instead of starting with a standard lattice cluster state and preparing it over the course of the computation, we are able to better understand the resource costs involved and eliminate wasteful Pauli measurements on the actual quantum device. Access to this algorithm-specific graph state also allows for optimisation over locally equivalent graph states to implement the same quantum circuit. The compiler presented here finds ready application in measurement based quantum computing, NISQ devices and logical level compilation for fault tolerant implementations.

1. Introduction

The circuit model is ubiquitous in the field of quantum computing and is used to represent and analyse implementations of various quantum algorithms. However this is not a unique choice and other models [1] also exist for instance that of measurement based quantum computing (MBQC) [2, 3]. In MBQC, one starts with a standard square lattice cluster state, which is a certain type of quantum graph state, and performs single qubit measurements to teleport the logical quantum state through the lattice. Gates are applied to the state by the choice of measurement basis and corrections due to the probabilistic nature of measurement outcomes are handled by dynamically adapting the basis of future measurements based on past outcomes [4] as well as classically tracking the Pauli frame [5]. Pauli basis measurements are sufficient to perform Clifford operations, however to perform non-Clifford operations, additional rotated basis measurements are required. There have been recent efforts to build concrete methods for algorithm development in MBQC like software frameworks such as [6, 7].

We present a compiler which employs an alternate workflow where instead of starting with the lattice cluster state, we prepare an algorithm-specific graph state. The latter is essentially a local Clifford equivalent graph state of the system after all the Clifford gates in the circuit have been applied. Once the algorithm-specific graph state is prepared, the computation is carried out by performing non-Pauli measurements on the qubits as before. In this manner we classically perform the Clifford part of the computation efficiently [8, 9] and leave only the non-Clifford part to the quantum device. Specifying both an algorithm-specific graph state and a measurement procedure completely determines the quantum computation leaving us free to discard the circuit model completely and treat the computation as an instruction set to prepare a graph state and perform a sequence of measurements. This is conceptually similar to stabilizer simulators that use graph states as an underlying representation [10], however we demonstrate a compilation framework that accommodates arbitrary circuits and not just Clifford circuits. It is worth highlighting here that since we only simulate the Clifford part of the circuit, we are able to *compile* the arbitrary circuit with memory quadratic in the number of qubits in the algorithm-specific graph state. The full *simulation* of the computation including the non-Pauli measurements on this graph state would of course require exponential memory resources.

In this framework, questions of optimal implementations can be addressed in terms of graph optimisation in local Clifford orbits (see section 6.1) opening new avenues for physical architecture designs. The algorithm-specific graph state also naturally represents the connectivity requirements for implementing a particular algorithm allowing us to determine the minimum device resources needed. While from the perspective of compilation we require knowledge of both the algorithm-specific graph state as well the measurement procedure to implement the computation, for the purposes of algorithm benchmarking and quantum resource estimation, it sufficient to know the graph state and the number of non-Pauli measurements required. Another application of algorithm-specific graphs is for logical level compilation for a fault tolerant error correction code. By encoding the computation in a graph state, we dispense with the requirement of performing complex state evolution in an error corrected code and only require the generation of the graph state followed by logical qubit measurements.

To perform all the Clifford operations first, we compile a given quantum circuit described in high level languages, such as Cirq and Q#, in the so called ICM (Initialisation-CNOT-Measurement) form [11, 12], the details of which are given in section 5.1. The variation of the ICM form we use allows for any circuit to be implemented by a series of Clifford gates followed by non-Pauli measurements. The quantum state prepared by the circuit prior to measurement is a stabilizer state and hence can alternatively be generated from an algorithm-specific graph state by using appropriate local Clifford corrections [13]. To determine this algorithm-specific graph state, we simulate the Clifford gates directly using well known stabilizer simulation techniques [9, 14] and we employ a graph conversion algorithm which computes a locally equivalent graph state given any stabilizer state (see section 4). It is worth pointing out here that there exists an interesting parallel to our techniques in the literature of circuit optimisation using the ZX calculus [15]. The authors of [16] for instance are able to remove nodes in the ZX diagram which represent Clifford gates by appropriate rewrite rules. Determining whether there is an exact mathematical correspondence between these two techniques is left to future work.

This paper is structured as follows: In section 2 we detail some preliminaries about the stabilizer formalism, graph states and their representation as a tableau. Section 3 gives a brief introduction to measurement based quantum computing. In section 4 we describe the stabilizer to graph state conversion and present an example. Then in section 5.1 we review the ICM formalism and show how to decompose an arbitrary quantum circuit into a Clifford block initialisation followed by non-Pauli measurements. We explicitly show this workflow for the controlled-$V^\dagger$ and the Toffolli gates in section 5. In section 6.1 we discuss the implications of the local equivalence of graph states for resource optimisation and in section 6.2 we comment on the utility of the compiler for optical quantum computing, NISQ [17] devices and fault tolerant implementations. A technical overview of our implementation of the compiler is given in section 7. Finally, we conclude and discuss future work in section 8.

Our implementation of the compiler can be found at https://github.com/QSI-BAQS/Jabalizer. The version of the compiler used in this paper and all the assoiciated data can be found at [18].

2. Preliminaries

Graph states are at the foundation of our compiler, and are an important subclass of stabilizer states widely used in measurement based quantum computing [2]. Stabilizer states and graph states are related concepts and, in the following, we offer a brief review.

2.1. Stabilizer formalism

Stabilizer states which were initially introduced as a formal description of quantum error correction codes [19–21] have been extensively studied and find wide application [20, 22, 23]. We will now give a brief review of the mathematical formalism of stabilizer states, their representation and connection to graph states. The interested reader is referred to section 10.5.1 of [24] for a through treatment.

2.1.1. Stabilizer states

Operations which map the set of stabilizer states to itself constitute the Clifford group of operations and the well known Gottesman–Knill theorem proves that all such operations can be efficiently simulated on a classical computer [8].

The Pauli group $\mathcal{G}_n$ of n qubits is the set of all n-fold tensor product combinations of $\{\pm 1, \pm i,\} \times \{I, X, Y, Z\}$, where I is the single qubit identity operator and X, Y, Z are the single qubit Pauli operators, respectively. We say that a commuting subgroup $\mathcal{S}_n$ of $\mathcal{G}_n$ stabilises a vector space $\mathcal{V}$ if for all elements $s \in \mathcal{S}_n$ and $v \in \mathcal{V}$, $sv = v$. These stabilizer groups will consist of Pauli group elements with coefficients $+1$ and -1 only since they must satisfy $s^2 = I^{\otimes n}$. A concise way of representing these subgroups is by specifying a set of generators of the subgroup $S = \{s_1, s_2, \ldots, s_k\}$ such that any element of the group can be represented as products of these generators. For a 2^n dimensional system, specifying n independent generators uniquely specifies a stabilizer state. The set of operations that map the set of all stabilizer states to itself is called the Clifford group of operations. If we only use Clifford operations and measurements in our quantum circuit, we can completely describe the evolution of the state vector classically by keeping track of the stabilizer generators of the state and their evolution under Clifford operations [9, 10].

2.1.2. Evolving stabilizer states under Clifford operations

State evolution of stabilizer states can be performed in the Heisenberg picture by conjugating the stabilizers with the unitary operations acting upon the state since for the evolution

$$|\psi'\rangle = U|\psi\rangle,$$

we can equivalently write,

$$|\psi'\rangle = Us_i|\psi\rangle = Us_i U^\dagger\, U|\psi\rangle = s_i'|\psi'\rangle,$$

where s_i' stabilizes the evolved state $|\psi'\rangle$. The efficiency of classically simulating stabilizer circuits in this way was improved upon by [9] using the so-called CHP approach which tracks both stabilizers and anti-stabilizers (anti-stabilizers of a state along with the stabilizers span the entire Pauli group $\mathcal{G}_n$), improving the performance of measurements within this model. We will confine our discussion to using just the stabilizer generators for simplicity since both methods are conceptually equivalent.

2.2. Tableau representation

Using the relations $X^2 = Z^2 = I$ and $ZX = iY$, any stabilizer $s \in \mathcal{S}_n$ can be thought of as tensor products of terms $X^x Z^z$ and a phase $(-1)^p$, where $x, z, p \in \{0, 1\}$ are binary variables. This allows us to represent the generators S as rows in a $n \times (2n+1)$ binary matrix

$$\begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} & z_{11} & z_{12} & \ldots & z_{1n} & p_1 \\ x_{21} & x_{22} & \ldots & x_{2n} & z_{21} & z_{22} & \ldots & z_{2n} & p_2 \\ & & & & \vdots & & & & \\ x_{n1} & x_{n2} & \ldots & x_{nn} & z_{n1} & z_{n2} & \ldots & z_{nn} & p_n \end{bmatrix}. \tag{1}$$

Such a representation is known as a stabilizer tableau. For a stabilizer $s \in S$ we refer to its corresponding row in the tableau, not including the final phase column, as $r(s)$. The tableau representation has the useful property that for two stabilizers $s_i, s_j \in S$, $r(s_i) + r(s_j) = r(s_i s_j)$, where the addition is modulo 2. This means that we can replace any row in the tableau with its sum with another row if we correct for the phases, since if s_i and s_j are a valid stabilizer, so too is $s_i s_j$. The phase of the product $s_i s_j$ can be efficiently computed by taking the product of the stabilizers term by term and multiplying together all the resulting phases.

2.3. Graph states

Given a mathematical graph defined through a vertex set V and an undirected, unweighted edge set E, the graph state is prepared by initialising qubits in the $|+\rangle$ state for each vertex and applying a controlled-Z operation between qubits with a shared edge. Equivalently, they are defined through a canonical stabilizer set of the form

$$s_i = X_i \prod_{j \in n_i} Z_j \tag{2}$$

for all vertex labels i, where n_i is the edge neighbourhood of the ith vertex in the graph. For a graph state, with a diagonal X-block (i.e. identity matrix), the Z-block in the tableau representation shown in equation (1) will correspond to the adjacency matrix of the abstract classical graph. Thus for a graph state the tableau will be of the form $[I_n|A|p]$, where A is the graph adjacency matrix and p is a column vector of phases which are by convention all $+1$.

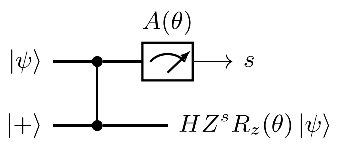


Figure 1. Gate teleportation circuit: Here $R_z(\theta) = e^{-i\theta Z/2}$ and measuring the observable $A(\theta) = R_z^\dagger(\theta)XR_z(\theta)$ on the first qubit has the effect of teleporting the state $R_z(\theta)|\psi\rangle$ to the next site with the addition of a Hadamard gate and Pauli Z correction depending on the measurement outcome.

3. Measurement based quantum computing

This section is a brief introduction to measurement based quantum computing. It is not our intention to be comprehensive and the reader is referred to [4, 25, 26] for further details. We will confine our discussion to elements that have a direct relevance to algorithm-specific graph compilation.

In MBQC the primary computational resource is a cluster state which is a special kind of graph state. The edges in a cluster state are confined to be between nearest neighbour nodes and the nodes themselves are arranged in a regular lattice structure. We will focus on the canonical example of the 2D square lattice in which MBQC is known to be universal. Note that other lattice structures are possible and 3D lattices in particular allow for fault tolerant topological codes [27]. Once the cluster state is prepared the entire computation is carried out by a series of single qubit measurement rounds, with the measured observable for each qubit depending on the measurement outcomes from previous rounds. Once all the measurements are done, the result of the computation can be determined by classical post processing. We now look at the details of how a universal set of gates (and thus any quantum circuit) can be implemented using MBQC.

3.1. Gate implementation

The primary driver of circuit simulation on a cluster state is a teleportation protocol which allows us to perform single qubit rotations. An arbitrary single qubit state can be encoded into a linear cluster using the Controlled-Z operation as shown in figure 1. Measuring the observable $A(\theta) = R_z^\dagger(\theta)XR_z(\theta)$, where $R_z(\theta) = e^{-i\theta Z/2}$, on the first qubit has the effect of teleporting the state $|\psi\rangle$ to the next site with the addition of a Hadamard gate and Pauli Z correction depending on the measurement outcome. Note that it is often convenient to represent $A(\theta)$ as the z-rotated X observable $\cos\theta X - \sin\theta Y$. Repeating the gate teleportation circuit in figure 1 twice with measurements angles θ_1 and θ_2 and outcomes s_1 and s_2, respectively, leads to the output state

$$\begin{aligned} |\psi_{\text{out}}\rangle &= HZ^{s_2}R_z(\theta_2)\,HZ^{s_1}R_z(\theta_1)\,|\psi\rangle \\ &= X^{s_2}R_x(\theta_2)\,Z^{s_1}R_z(\theta_1)\,|\psi\rangle \\ &= X^{s_2}Z^{s_1}R_x\left((-1)^{s_1}\theta_2\right)R_z(\theta_1)\,|\psi\rangle, \end{aligned} \tag{3}$$

where in the second line we have commuted the Hadamard operator on the left through to meet the one on the right which converts the Z correction and rotation to an X correction and rotation with $R_x(\theta) = HR_z(\theta)H$. In the last line we commute the Z correction to the left using the relation $e^{i\theta X}Z = Ze^{-i\theta X}$. It is straightforward to see that repeating this procedure we are able to implement a series of alternating X and Z rotations, three of which is sufficient to implement any single qubit unitary gate.

There are two unwanted effects due to the measurement dependent Pauli corrections: (1) there is a Pauli operator of the form $X^{f_x(\vec{s})}Z^{f_z(\vec{s})}$, where $f_x(\vec{s})$ and $f_z(\vec{s})$ are functions of the binary measurement outcome vector $\vec{s}$ (here $\vec{s} = (s_1, s_2)^T$); and 2) The R_x and R_z rotation angles are flipped from θ_i to $-\theta_i$ if an odd number of Z or X operators are commuted through it, respectively.

We do not need to do any active correction for the first effect since at the end of the circuit we are going to measure in the computational Z basis. The presence of a Z operator has no effect on the Z measurement and the presence of an X operator induces a relabeling between 0 and 1, which can be accounted for in post-processing as long as we keep track of all the measurement outcomes.

As for the second effect, as seen in equation (3), whether $R_x(\theta)$ or $R_x(-\theta)$ is implemented depends on the previous measurement outcome s_1 and in general all previous measurement outcomes for an arbitrary long sequence of gates. Since we can determine beforehand if there is a flip we will measure the observable $A(-\theta)$ instead of $A(\theta)$ to compensate for it. The general single qubit unitary in Euler normal form $U(\zeta,\eta,\xi)$, where $U(\zeta,\eta,\xi) = e^{-\zeta X/2}e^{-\eta Z/2}e^{-\xi X/2}$, is implemented using a 5 qubit linear cluster as shown in figure 2(a). CNOT gates can be implemented using a 15 qubit cluster state as shown in figure 2(b).

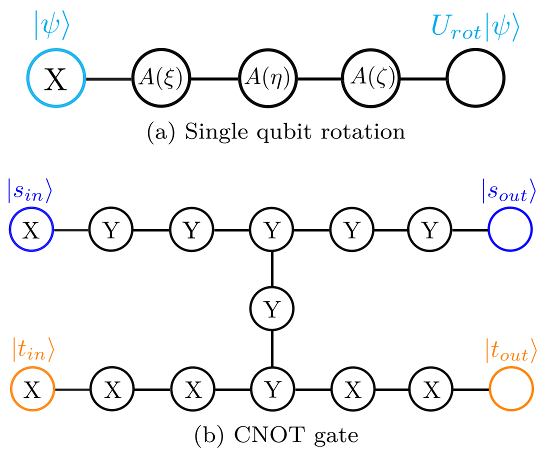


Figure 2. (a) An arbitrary single qubit unitary in the Euler representation $U_{\mathrm{rot}} = e^{-i\frac{\zeta}{2}Z}e^{-i\frac{\eta}{2}X}e^{-i\frac{\xi}{2}Z}$ is implemented by the linear cluster shown above by measuring the first qubit in Pauli X basis and the other three qubits in the $A(\theta) = \cos\theta X - \sin\theta Y$ basis, with θ being ξ, η and ζ, respectively, for the second, third and fourth qubits from the left. Based on previous measurement outcomes, $A(-\theta)$ could be measured instead of $A(\theta)$. (b) A CNOT gate can be implemented between a source qubit and a target qubit using the 15 qubit cluster state shown and simultaneous Pauli measurements as indicated on each qubit.

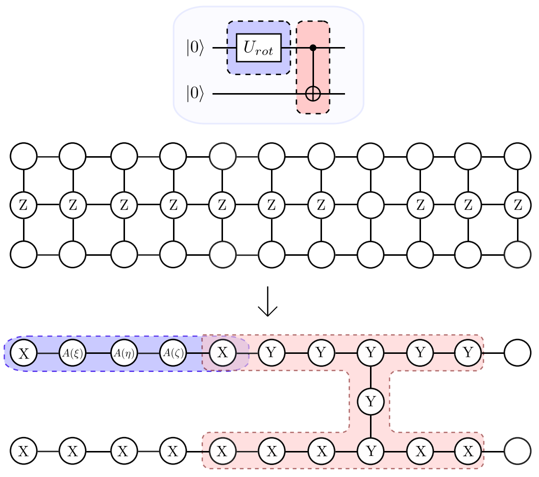


Figure 3. The logical circuit is transpiled for MBQC by (1) starting with a lattice cluster state of sufficient size. (2) Performing Pauli Z measurements to disconnect qubits and produce the required topology. (3) All Pauli measurements are performed in the first round of measurement (implements all Clifford gates). (4) Adaptive measurements are performed on the remaining qubits over several rounds until the computation is finished. (5) Post processing of the measurement results recovers the output of the logical circuit.

Finally, the preparation of the state $|\psi\rangle$ itself can be incorporated into the protocol by starting with a suitable single qubit rotation. Since arbitrary single qubit gates and CNOT gates together form a universal set of gates [28, 29], any quantum circuit can be simulated on a 2D cluster state.

3.2. Circuit transpilation

A high level circuit, described in terms of gates acting on qubits, can be implemented using a 2D square lattice cluster state of sufficient size. In the following, we describe the implementation method.

Assuming that a circuit consists of single qubit rotations and nearest neighbour CNOTs (which any circuit can be implemented as) the first step is to etch out the circuit topology out of the lattice cluster state. This is done using Pauli Z measurements which disconnect the measured node from the cluster as shown in figure 3. Once we have this topology which will support the gates what remains is to measure the remaining qubits in an adaptive fashion to simulate the gate operations.

One could in principle implement the gates in the same order as they are done in the circuit description. However, it turns out this is not necessary, or indeed even preferable [26]. The basis in which to measure a particular qubit k depends on the measurement outcomes of a set of qubits b_k which is called the backward cone of the qubit k. This implies that the qubits in the cluster state can be partitioned into disjoint sets $\{Q_t\}$ determining different measurement rounds. The measurement of the set Q_0 which contains qubits who's measurement basis does not depend on the outcome of any other measurement is conducted first. The results of this measurement round is used to determine the basis to measure qubits in Q_1, and so on until the computation is finished.

The set Q_0 contains all the qubits on which Pauli measurements are to be performed. The reason for this is that the adaptive nature of the measurement basis is governed by the presence of a Pauli operator $\sigma_i \in \{X, Y, Z\}$ on the qubit to be measured. The measured observable σ_j is transformed by this Pauli operator as $\sigma_i \sigma_j \sigma_i = (-1)^{\delta_{ij}} \sigma_j$. In other words, we either measure σ_j or $-\sigma_j$ which is just a relabelling of the measurement outcomes. All Clifford operations can be implemented in the MBQC scheme with just Pauli measurements, which means that the first round of measurement Q_0 implements all Clifford gates in the circuit. Note that this does not depend on the temporal ordering of these gates in the actual quantum circuit; in fact even the final computational basis state measurements at the end of the circuit are implemented in this first round. This does not mean that the output of the quantum circuit is known after the first round, the measurement outcomes of all the qubits are required to reconstruct the output of the logical circuit. As a consequence of this measurement based implementation of the Clifford gates, there will be Pauli corrections scattered throughout the circuit which have to be propagated either to the end of the circuit to modify the final measurement outcomes or to the start of the circuit to initialise the Pauli frame which tracks the accumulation of Pauli operators on the qubits.

Since the set of stabilizer states is invariant under Pauli measurements, after the first round of measurement we are left with a stabilizer state $|\psi_S\rangle$ which is local Clifford equivalent to some graph state(s) [13, 30]. These Clifford equivalent set of graph states are referred to as algorithm-specific graph states and can be considered the fundamental resource required to implement the quantum logic circuit.

3.3. Algorithm based graph compilation

Instead of preparing the algorithm-specific graph state through measurement as described in section 3.2, we could instead efficiently compute it classically and prepare it directly. This is a consequence of the fact that all the operations that generate the algorithm-specific graph state are Clifford operations on stabilizer states and the Gottesman–Knill theorem [8]. In this manner we are not performing any classically simulatable operations on a quantum device but saving those resources for the genuinely quantum part.

The most obvious way to generate an algorithm-specific graph is to start with a stabilizer description of a cluster state and following the procedure shown in figure 3 etch out the required circuit topology with Z basis measurements and perform all the Pauli basis measurements implementing the measurement round Q_0. The obtained stabilizer state can be converted to a graph state using the procedure explained in section 4.

Alternatively, we can go one step further and not encode the Clifford gates into the cluster at all but simulate them as unitary gates at the logic circuit level before we encode the circuit into a graph state. To do this we recast the circuit in a form called the ICM (Initialise-CNOT-Measurement) form as described in section 5.1 which allows all Clifford gates to be implemented first at the logical circuit level. Our method is described in section 5.

4. Stabilizer state to Graph state conversion

Every stabilizer state can be converted to a graph state using local Clifford operations. This means that given any stabilizer tableau, we can perform a process akin to Gaussian elimination augmented with local Clifford operations to obtain a corresponding graph state [13]. Note that this graph state is not unique and there could be multiple graph states which are local Clifford equivalent to a given stabilizer state.

4.1. Graph conversion algorithm

We now present an algorithm to convert any stabilizer state into a graph state (a pseudo-code is given in appendix B). The algorithm uses the fact that modulo 2 addition of the rows of the tableau corresponds to multiplication of the corresponding stabilizers, up to phase correction. i.e. for a stabilizer set $S = \{s_1, s_2, \ldots, s_n\}$, we can replace the row corresponding to s_j in the tableau with *rowsum*(i,j), where *rowsum* is a function that does modulo 2 addition for the X and Z blocks of the two rows and computes the appropriate phase for the final column. An efficient way to compute this phase from the tableau is given in [9]. To convert an arbitrary stabilizer to a graph state, we need to transform the X block into an identity matrix, and the Z block will automatically be symmetric to ensure that all the stabilizers commute [13]. The full algorithm is as follows —

Remove zero columns for the X block:

(a) If there is a zero column in the X-block, we perform a Hadamard operation by swapping that X column with its corresponding Z column since this Z column has to be non-zero. This is because a valid stabilizer state set cannot have both the Z and X column be all zeros for a qubit. That would imply that there is only an identity stabilising that qubit for all the stabilizers in the set which would leave the state of the qubit unspecified.

Make the X block lower triangular:

(b) Iterating i from 1 to n , if X_{ii}[6] $= 1$, we can set all X_{ji} with $j > i$ to equal 0 by replacing all rows j such that $X_{ji} = 1$ with $rowsum(i,j)$. We do this for all rows $j > i$.
(c) If for some i, $X_{ii} = 0$, but there exists some $j > i$ such that $X_{ji} = 1$, we swap rows i and j and perform the logic in step b).
(d) If there is no $j \geq i$ such that $X_{ji} = 1$, there must exist some $Z_{ji} = 1$, since we know that for every stabilizer state the X block can be made full rank using local operations and *rowsums*, the only freedom left is to bring it from the Z block. We perform a Hadamard operation swapping the i^{th} column of the X and Z block to achieve this and perform the logic in step b) or c) depending on whether X_{ii} is 0 or 1.

Make the X block diagonal:

(e) For i from n to 1, if any $X_{ji} = 1$, replace row j with $rowsum(i,j)$.

Make Z block diagonal zero:

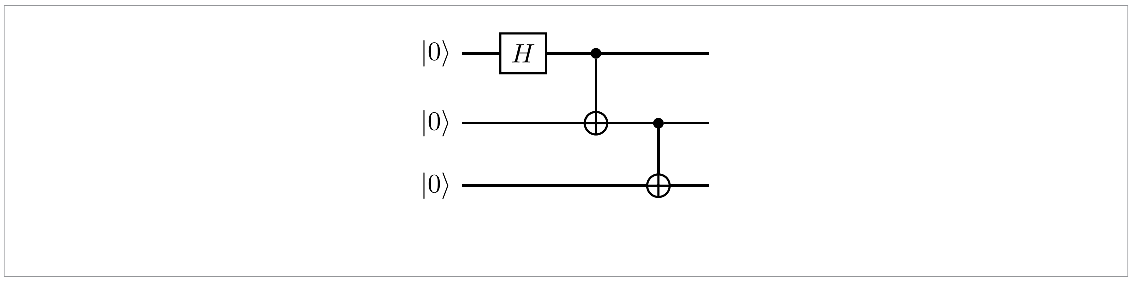
(f) For all i such that $Z_{ii} = 1$, we perform the phase gate transformation on qubit i which performs the transformation $XZ \to X$, leaving us with $Z_{ii} = 0$.

Make all phases positive:

(g) For all rows with a negative phase we apply a Z gate transformation which takes $X \to -X$.

4.2. Clifford circuit graph evolution example: GHZ

If we start with a Clifford circuit, we can apply the conversion algorithm directly. For example, taking the three qubit GHZ generation circuit, the steps for converting the GHZ state to a graph state are shown in table 1. The GHZ circuit,



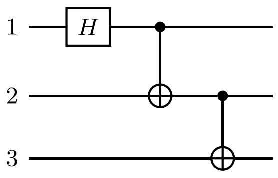
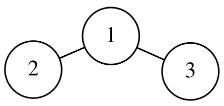
prepares the state $|\psi\rangle = |000\rangle + |111\rangle$ (we assume implicit normalisation throughout this paper for simplicity). Starting from the canonical stabilizer for the all zero state, the circuit does the following stabilizer transformation,

$$\begin{array}{ccc} Z & I & I \\ I & Z & I \\ I & I & Z \end{array} \to \begin{array}{ccc} X & X & X \\ Z & Z & I \\ I & Z & Z \end{array}.$$

As shown in the first two rows of table 1. The remaining rows of table 1 describes the conversion of this stabilizer state to a graph state through the application of Clifford gates (*rowsums* do not correspond to physical operations). Since this conversion process is invertible we can instead view this in the opposite direction—beginning with the algorithm-specific graph state in the final row of table 1 and applying the inverse of the local transformation rules (in this case Hadamard gates on qubits 3 and 2) generates the output of the Clifford circuit.

[6] Here the first subscript refers to the row and second the column.

Table 1. Conversion of a 3 qubit GHZ state to a graph state. The action of a Hadamard on a qubit is to swap Zs and Xs for the column corresponding to the qubit. $rowsum(i \rightarrow j)$ replaces s_j with $s_i s_j$.

Operation	Stabilizer	State
	$\begin{matrix} Z & I & I \\ I & Z & I \\ I & I & Z \end{matrix}$	$\|000\rangle$
	$\begin{matrix} X & X & X \\ Z & Z & I \\ I & Z & Z \end{matrix}$	$\|000\rangle + \|111\rangle$
H on 2	$\begin{matrix} X & Z & X \\ Z & X & I \\ I & X & Z \end{matrix}$	$\|0+0\rangle + \|1-1\rangle$
$rowsum(2 \rightarrow 3)$	$\begin{matrix} X & Z & X \\ Z & X & I \\ Z & I & Z \end{matrix}$	$\|0+0\rangle + \|1-1\rangle$
H on 3	$\begin{matrix} X & Z & Z \\ Z & X & I \\ Z & I & X \end{matrix}$	$\|0++\rangle + \|1--\rangle$
Final Graph		

5. A compiler for algorithm-specific graph states

We introduce a circuit compiler formalism which allows us to systematically decompose any given circuit using T-state injection [24] and a variation of it using T gate teleportation which will form part of our compiler pipeline.

In this section, we show that the ability to perform all Clifford operations simultaneously in the first round of measurement in the MBQC model has a corresponding analogue in the circuit model. Here, instead of initialising a standard cluster state and performing Pauli measurements we implement non-Clifford gates using gate teleportation as shown in figure 6, which allows all Clifford gates to be implemented first.

We will also provide a step-by-step derivation of an algorithm-specific graph state for some typical quantum gates. The basic principles demonstrated can be extended to arbitrary quantum circuits.

In the MBQC model, the Clifford gates could be implemented first because of the invariance of Pauli measurement bases due to Pauli corrections as explained in section 3.2. The corresponding property that allows this in the circuit model is that Pauli corrections due to the teleported non-Clifford gates when commuted through a Clifford gate do not change the Clifford gate but only the correction. This allows us to track the corrections and propagate them to the end of the circuit directly before subsequent measurements. This will impose a temporal ordering on the measurements that implement the teleported gates since the Pauli corrections due to previous measurements will determine the measurement bases for current measurements.

5.1. ICM formalism

The ICM form of a quantum circuit (figure 4) was introduced in [11] for logical level QEC compilation. It is a decomposition of an arbitrary circuit into three layers — (1) an initialisation layer preparing all qubits into one of 4 states, (2) an array of CNOTs and (3) a staggered measurement block in the Z and X basis with the basis choice of subsequent blocks depending on the measurement outcomes of previous ones.

The ICM formalism can be understood to be systematic state injection with selective measurement corrections applied through the basis choice in the measurement blocks. While the state injection circuit in figure 5 can be verified directly, it can also be understood from the basic MBQC gate teleportation circuit in

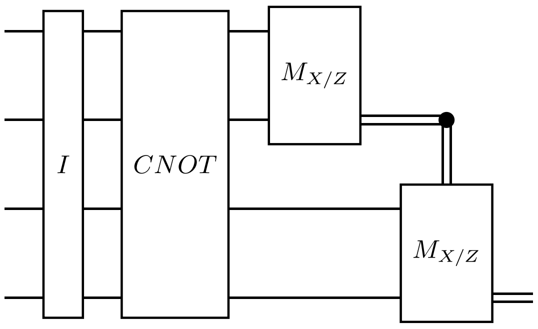


Figure 4. ICM form of a quantum circuit. The I block initialises each qubit in either the state $|0\rangle$ or the states $|0\rangle + e^{ik\pi}|1\rangle$, with $k \in \{0, 1/2, 1/4\}$. The CNOT block is an array of CNOTs. The measurement blocks performs measurement in the X or Z basis and subsequent measurements bases are decided based on the outcome.

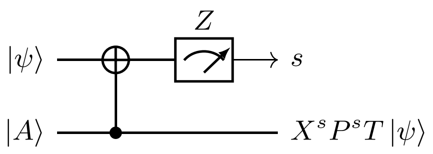
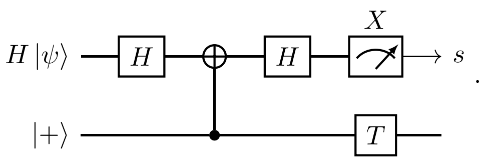


Figure 5. T gate implementation using state injection in the ICM formalism. Here the injected state $|A\rangle = T|+\rangle$. The X correction is propagated forward to measurement while the P gate correction is implemented by a similar state injection of $|Y\rangle = P|+\rangle$ instead of $|A\rangle$.

figure 1. Using the fact that $|A\rangle = T|+\rangle$ and that Z measurement is equivalent to a Hadamard gate followed by an X measurement, the circuit in figure 5 can be redrawn as,



Here we have commuted the T gate through the control of the CNOT gate. Using the fact that conjugating the target of a CNOT gate with a Hadamard induces a CZ gate, we can see this circuit is equivalent to the gate teleportation circuit in figure 1 with an input state $H|\psi\rangle$, $U_Z(\theta) = \mathbb{I}$ (i.e. $A(\theta = 0) = X$) and a T gate acting on the output. Hence, the output of this circuit must be $THZ^sH|\psi\rangle = TX^s|\psi\rangle$. When the measurement outcome is 0, we have a T gate applied as desired but when the outcome is 1 there is an X operator acting before T which we need to commute to the outside using the equivalence (up to global phase) $TX = XT^\dagger$. This gate can be corrected to a T gate by applying the operator PX hence confirming the circuit output in figure 5.

The CNOT and measurement blocks of the ICM form are Clifford operations and Pauli measurements, but the initialisation block requires the preparation of non-Clifford states for state injection. For our purpose we modify this scheme as shown in [12] known as inverse ICM, such that the initialisation block only prepares stabilizer states, but as a trade-off the measurements are now non-Pauli measurements. This is achieved using the gate teleportation circuit shown in figure 6. This circuit can be derived from the gate teleportation circuit in figure 1 by absorbing the Hadamard gate from the output to the right of the CZ gate and the Hadamard from the input $|+\rangle$ state to the left of it and setting $U_Z(\theta) = T$.

5.2. Controlled-$V^\dagger$ graph states

Consider the Clifford $+$ T gate decomposition of the controlled-$V^\dagger$ gate show in figure 7. The $V^\dagger$ gate is one of the square roots of the X gate and is defined as,

$$V^\dagger = \frac{1}{2}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix}. \tag{4}$$

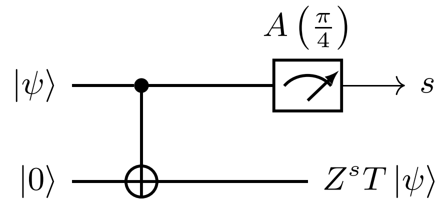


Figure 6. T gate teleportation in the inverse ICM formalism where the measured observable is $A\left(\frac{\pi}{4}\right) = T^{\dagger}XT$. $T^{\dagger}$ can be similarly implemented by measuring the observable $A\left(-\frac{\pi}{4}\right) = TXT^{\dagger}$. Using this subcircuit we are able to decompose any Clifford + T circuit into a stabilizer state initialisation followed by non-stabilizer measurements.

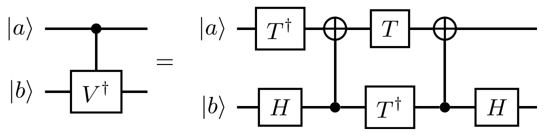


Figure 7. Controlled-$V^{\dagger}$ decomposition using Clifford + T gates.

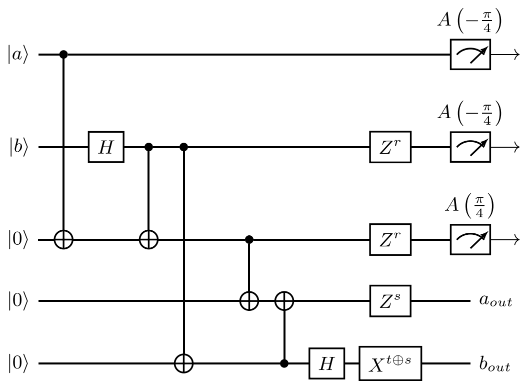


Figure 8. Teleported T gate implementation of the controlled-$V^{\dagger}$ gate with the Pauli corrections propagated to the end.

The T and $T^{\dagger}$ gates are implemented using the teleportation algorithm in figure 6 with the Pauli corrections propagated to the end of the circuit, as shown in figure 8. In this example, all the non-stabilizer measurements can be performed simultaneously since the only Pauli corrections for the measured qubits are Z corrections that changes the measurement of $A(\theta) \to ZA(\theta)Z = -A(\theta)$, which is a classical relabelling of the measurement outcomes. However, this is not the case in general, for example, if we wanted to now implement a T gate on the qubit b_{out}, the presence or absence of the X correction will decide the measurement basis since $XA(\theta)X = A(-\theta)$. Note that in the state injection picture of the ICM form, measurement outcomes determined whether T gates or $T^{\dagger}$ gates were implemented due to the Pauli X correction, we can see the same holds true here. We will not apply unitary corrections for these Pauli gates but rather modify the measurement basis to account for them. For example if we know based on previous measurement that an X gate correction acts on a qubit before an $A(\theta)$ basis measurement, we will instead perform an $A(-\theta)$ basis measurement. These Pauli corrections thus give a time ordered measurement pattern with feed-forward which can be determined at compilation time using Pauli tracking [5].

For inputs $|ab\rangle = |00\rangle$, the state of the system before the measurements are performed, ignoring the Pauli corrections is $|0000+\rangle + |0110-\rangle$, where $|\pm\rangle = |0\rangle \pm |1\rangle$. This is locally equivalent to the graph state shown in figure 9 (see appendix A for a detailed derivation). We can equivalently start with the graph state and apply the local corrections to generate the same output using the circuit given in figure 10. The intuition from the graph in figure 9 is that qubits 1 and 4 should factor out in the pre-measurement state—indeed, this is true for the state $|0000+\rangle + |0110-\rangle$.

To perform the controlled-$V^{\dagger}$ gate we implement the measurements in the $A\left(-\frac{\pi}{4}\right)$ basis on qubits 1, 2 and $A\left(\frac{\pi}{4}\right)$ on qubit 3. For ease of calculation, we apply the local corrections explicitly rather than absorbing them into the measurement basis and also assume the measurement outcomes $r = t = s = 0$. This can always be achieved in practice by measuring the qubits 2 and 3 in the basis of $-A\left(-\frac{\pi}{4}\right)$ and $-A\left(\frac{\pi}{4}\right)$, respectively, if $r = 1$ and applying a X correction to qubit 5 if $t \oplus s = 1$.

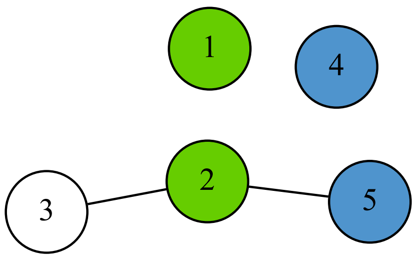


Figure 9. Local Clifford equivalent graph state before measurement for the controlled-$V^\dagger$ decomposition in figure 8 for inputs $|ab\rangle = |00\rangle$. The input[7] and output qubits are shown as green and blue, respectively. The circuit is implemented by applying local Clifford gates and measuring all the qubits except the output qubits.

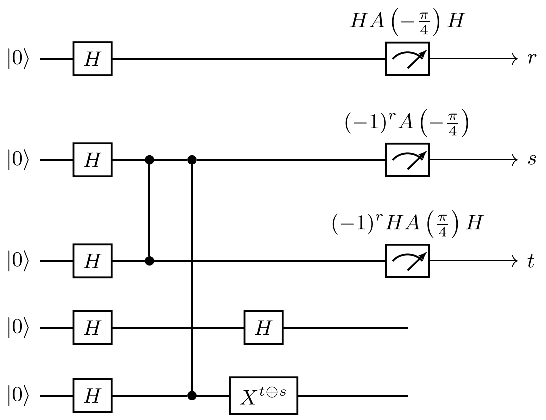

$HA\left(-\frac{\pi}{4}\right)H$
$(-1)^r A\left(-\frac{\pi}{4}\right)$
$(-1)^r HA\left(\frac{\pi}{4}\right)H$
$|0\rangle$
H
r
s
t
$X^{t\oplus s}$

Figure 10. Compiled circuit: The Hadamard and the CZ gates prepare the graph state given in figure 9 and then the local corrections are applied to generate the same output as the circuit in figure 8. In practise the local corrections can be absorbed into the measurement basis choice as shown for the first 3 qubits. The remaining corrections are also similarly propagated forward until measurement.

The operators $A\left(\pm\frac{\pi}{4}\right)$ can be written in their spectral decomposition as,

$$A\left(\tfrac{\pi}{4}\right) = |A^*\rangle\langle A^*| - |\overline{A}^*\rangle\langle\overline{A}^*|, \tag{5}$$
$$A\left(-\tfrac{\pi}{4}\right) = |A\rangle\langle A| - |\overline{A}\rangle\langle\overline{A}| \tag{6}$$

where, $|A\rangle = T|+\rangle$, $|\overline{A}\rangle = T|-\rangle$ and $|A^*\rangle$, $|\overline{A}^*\rangle$ are similarly defined with $T^\dagger$ instead of T. Measuring $A\left(\pm\frac{\pi}{4}\right)$ as 0, projects the computational basis states as,

$$|A^*\rangle\langle A^*|0\rangle = |A^*\rangle, \qquad |A^*\rangle\langle A^*|1\rangle = e^{i\pi/4}|A^*\rangle, \tag{7}$$
$$|A\rangle\langle A|0\rangle = |A\rangle, \qquad |A\rangle\langle A|1\rangle = e^{-i\pi/4}|A\rangle. \tag{8}$$

The measurement projects the state as,

$$|A\rangle\langle A|\otimes|A\rangle\langle A|\otimes|A^*\rangle\langle A^*|\left(|0000+\rangle+|0110-\rangle\right) = |AAA^*\rangle\left(|0+\rangle+|0-\rangle\right) = |AAA^*00\rangle. \tag{9}$$

giving the output state in the last two qubits as,

$$|a_{\mathrm{out}}b_{\mathrm{out}}\rangle = |00\rangle. \tag{10}$$

This is of course the output of a controlled-$V^\dagger$ acting on the input $|ab\rangle = |00\rangle$. A less trivial example is given by choosing the input to be $|ab\rangle = |+0\rangle$. This is equivalent to applying a Hadamard gate to the first qubit at the beginning of the circuit in figure 8. Since this only initialises a different stabilizer state and all Pauli corrections are introduced after this point, the measurement sequence and how future measurements are

[7] Note that input here refers to the labelling of the circuit qubits; the input state is not encoded into the green nodes, rather different inputs lead to different graph structures.

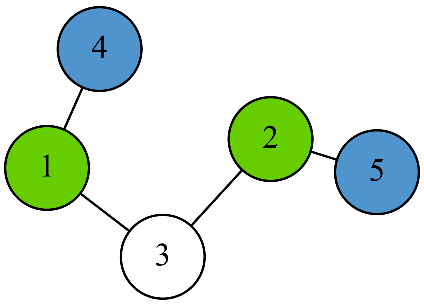


Figure 11. Locally equivalent graph state before measurement produced by the circuit in figure 8 for inputs $|ab\rangle = |+0\rangle$. Input and output nodes are shown in green and blue, respectively.

influenced by previous measurement outcomes remain unchanged. In this case, the stabilizer state produced before measurement is given by,

$$|\psi\rangle = |0000+\rangle + |0110-\rangle + |1011+\rangle + |1101-\rangle \tag{11}$$

which can be converted to the graph state shown in figure 11 (see appendix A for a full derivation). The post measurement state in this case will be,

$$\begin{aligned}&|A\rangle\langle A| \otimes |A\rangle\langle A| \otimes |A^*\rangle\langle A^*||\psi\rangle\\ &= |AAA^*\rangle\left(|0+\rangle + |0-\rangle + |1+\rangle + e^{-i\pi/2}|1-\rangle\right)\\ &= |AAA^*\rangle\left(|00\rangle + (1-i)\,|10\rangle + (1+i)\,|11\rangle\right).\end{aligned}$$

Normalising the state above and tracing out the measured qubits, we get the output state as,

$$|a_{\text{out}}b_{\text{out}}\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1-i}{2\sqrt{2}}|10\rangle + \frac{1+i}{2\sqrt{2}}|11\rangle. \tag{12}$$

The $V^\dagger$ gate acts on the computational basis states as,

$$\begin{aligned}|0\rangle &\xrightarrow{V^\dagger} \frac{1-i}{2}|0\rangle + \frac{1+i}{2}|1\rangle\\ |1\rangle &\xrightarrow{V^\dagger} \frac{1+i}{2}|0\rangle + \frac{1-i}{2}|1\rangle.\end{aligned}$$

The action of the controlled-$V^\dagger$ on the initial state $|+0\rangle$ can be seen to be,

$$\begin{aligned}|+0\rangle &\xrightarrow{CV^\dagger} \frac{1}{\sqrt{2}}\left(|00\rangle + |1\rangle\left(\frac{1-i}{2}|0\rangle + \frac{1+i}{2}|1\rangle\right)\right),\\ &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1-i}{2\sqrt{2}}|10\rangle + \frac{1+i}{2\sqrt{2}}|11\rangle\end{aligned}$$

as expected.

5.3. Toffoli gate graph states

We describe the decomposition of a Toffoli gate. We start with the seven T gate decomposition shown in figure 12. We apply the same procedure as before and teleport the T gates using the circuit in figure 6. The graph states produced along with the local corrections and qubits to be measured are given in table 2. From the generated graphs it is clear that one can only generate entanglement between all the output qubits if both the controls of the Toffolli gate are active (not in the $|0\rangle$ state).

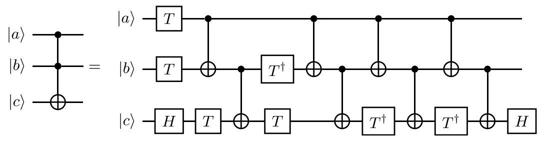
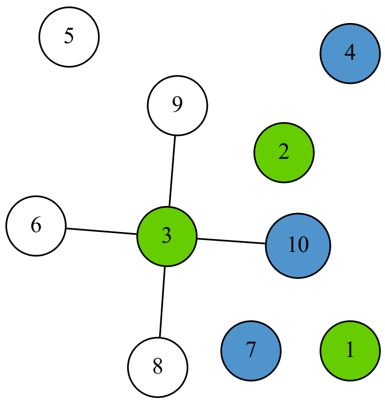
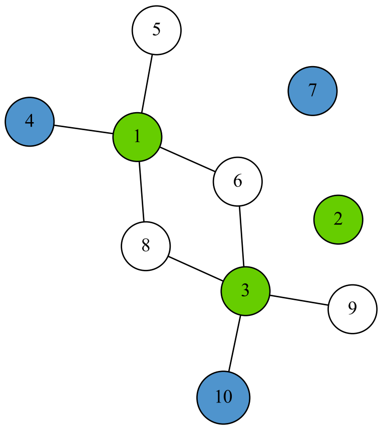
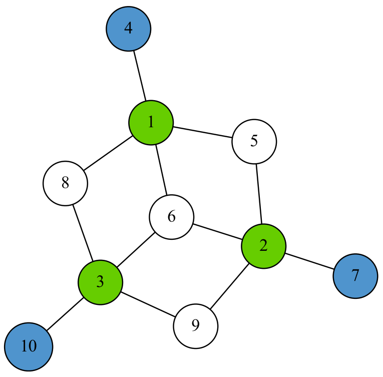


Figure 12. Toffoli gate decomposition using Clifford + T gates.

Table 2. Toffoli gate implementation for different inputs. The input and output qubits are shown in green and blue respectively. The circuit is implemented by generating the graph state, applying the local corrections and measuring all the qubits except the output qubits in the $A\left(\pm\frac{\pi}{4}\right)$ basis. Pauli corrections to the measurements are assumed to be classically tracked through the circuit to determine the measurement sequence.

Input	Graph	Local Corrections
$\vert 000\rangle$		$H_1H_2H_4H_5H_6H_7H_8H_9$
$\vert +00\rangle$		$H_2H_4H_5H_6H_7H_8H_9$
$\vert ++0\rangle$		$H_4H_5H_6H_7H_8H_9$

6. Discussion

It is important to note that the graph states produced by our algorithm are not unique. There can be many graph states that are equivalent under local Clifford operations [13] and hence locally equivalent to the output stabilizer state of the circuit. This fact can be exploited to optimise over different graph state implementations. We now discuss this in more detail as well as consider the hardware compatibility of our approach.

6.1. Resource optimisation

It has been shown that transformation between graph states using local Clifford operations is completely characterised by repeated application of the local complementation operations [13]. In graph theoretic terms for a vertex k with neighbourhood n_k, the local complementation will invert the edges (remove if there is one, add if there is not) between all vertices in n_k. Physically this operation on vertex k is performed by the operation $\sqrt{-iX_k}\prod_{j\in n_k}\sqrt{iZ_j}$, where $\sqrt{-iX_k} = e^{-i\frac{\pi}{4}X_k}$ is the square root of X operation acting on qubit k and $\sqrt{iZ} = e^{i\frac{\pi}{4}Z_j}$ is the square root of Z operation acting on qubit j. If two graph states $|G\rangle$ and $|G'\rangle$ are equivalent under local Clifford transformation there exist some finite set of local complementation operations $\{l_i\}$ such that,

$$l_n \ldots l_2 l_1 |G\rangle = |G'\rangle. \tag{13}$$

Whether two graph states are indeed equivalent under local Clifford operations can be determined in polynomial time [31].

This gives us an additional degree of freedom to optimise our algorithm-specific graph state. Depending on the physical resources it might be desirable to reduce the total edge count, degree (number of edges per node) or prepare an initial state with a simple topology such as a linear graph state. All these problems are well studied in classical graph theory and find ready application here [32–34].

6.2. Applications and open questions

MBQC based circuit execution is well suited to many optical quantum computing architectures [35, 36] as well as NISQ devices [37]. However, the compilation workflow presented here is not confined to any particular device architecture as one could use it as the logical level representation for any quantum device. An attractive application would be in the execution of fault tolerant quantum circuits using a suitable error corrected code such as the surface code [38–40]. The nodes of the algorithm-specific graph will now be logical qubits in a protected subspace. This could offer several simplifications during run time as the only gates that need to be implemented are the ones that generate the graph state (H and CZ) and gates to implement fault tolerant rotated basis measurements dispensing with the need for lattice surgery (outside graph state generation) and dynamical ancillary routing operations [41]. This makes exact determination of the space-time volume of executing quantum algorithms in a fault tolerant manner much more tractable than it is presently.

It is a reasonable assertion that the edge structure of algorithm-specific graphs that implements a quantum circuit upper-bounds the entanglement required for the computation. It is an open question whether this intuition can be quantified and if further connections exist between graph properties and the computational complexity of the algorithm being implemented similar to those known for regular universal cluster states [42]. If such connections exist, graph compilation could be used for classification and characterisation of different quantum algorithms—for example do classes of algorithm share common compiled graph properties? Such a finding could have significant implications for future designed-for-purpose quantum devices targeted at specific computational tasks similar to how GPUs work in classical computers.

The graph compilation framework we have presented here is input specific as can be readily noted from table 2. But clearly, there must exist graph structures that implement desired unitaries on arbitrary inputs like in the cluster state model but using only non-Pauli measurements. One way to see this is to note that in a cluster state computation, once all the Pauli measurements are performed, the state of the system will be local equivalent to a graph state. This state can be used to perform the computation with only non-Pauli measurements. A natural question to ask is how to synthesise such graph states for a target unitary operation. A related research direction is the optimisation of parameterised quantum circuits. Numerous applications exist which rely on the optimisation of circuit parameters using iterative classical-quantum processing [43, 44] to achieve a target quantum process such as variation quantum eigensolvers [45] and quantum machine learning [46] to name two. Using edge structure and measurement angles as parameters these problems can be recast in the graph compilation framework and it is an open question if specific graph structures provide advantages to classes of optimisation tasks.

7. Technical overview

In this section we describe the technical details of how different parts of the compiler pipeline are built. The ICM part of the pipeline has been built using Cirq, so Q# circuits are first translated into Cirq circuits. Nevertheless, our compiler is output agnostic and can generate OpenQasm circuits, too.

7.1. Q# to Cirq conversion

In order to support high-level Q# algorithms [47], an intermediate component is needed to extract the circuit representation. Q# is a functional language which is suited for hybrid quantum–classical algorithms and often exhibits non-trivial branching.

The Q# host program uses a simulator backend to perform compilation into a qubit register and a sequence of high-level operations. The choice of simulator backend depends on the nature of the experiment being run, but all are supported through the same translation extension. These operations are decomposed into primitive operations including `X, Y, Z, H, S, T, CX, R, Measure` gates. In the case of state space simulator backends, operations are sometimes only decomposed into `SWAP, CCNOT, CCX, CCZ` gates.

To yield a circuit representation, an extension attaches a hook to the simulator before the experiment is run. During execution, a call is sent to the extension during each operation for parsing. If real-time operation is desired, the gates are broken down and streamed to the next stage of the pipeline. Otherwise, the circuit representation of the experiment from start to finish is stored in an output file. As our desired output format is Cirq, the stream of operations is converted into an initialisation of a Cirq qubit register and a sequence of moments. A standalone module implementing this part of the pipeline can be found at https://github.com/QSI-BAQS/Trace2Cirq.git.

7.2. ICM compilation

To perform an ICM decomposition, the compiler has to take a given Clifford + T gate circuit and teleport all T and $T^\dagger$ gates using ancillas, as described in section 5.1. The Clifford + T decomposition can be performed using Cirq's native decompose protocols and the main technical challenge left is when the compiler encounters a T gate it needs to determine which wire this gate must act on depending on previous gate teleportations. For example, in the controlled-$V^\dagger$ decomposition in figure 8, when the compiler decomposes the T gate, it needs to know that the decomposition is applied to the second wire if the $T^\dagger$ gate has already been decomposed. To do this the compiler adds an `op_id` flag to every operation to keep track of the order of the operations and the class `SplitQubit` is used to track teleported qubits. When a gate is teleported, a new `SplitQubit` instance is created for the ancilla which inherits the `op_id` of the operation that generated it and a reference to this new wire is stored in the original one. Comparing a gate's `op_id` with the those of the wire and it is children, the compiler is able to determine the correct wire to apply the decomposition on.

7.3. Stabilizer simulation

The stabilizer simulator has been designed with a top level data structure which uses an integer array to track the tableau defined in section 2.2 and a backend which makes calls to a CHP simulator. This gives front-end access to the conceptually simpler tableau description while still using the more efficient CHP formalism for actual computation. The stabilizer backend we use is STIM [14].

This part of the pipeline receives the inverse ICM decomposed Cirq circuit and computes the output stabilizer state produced by the circuit before the measurement sequence is applied. The graph conversion algorithm described in section 4.1 is now applied to the stabilizer state and the compiler returns the appropriate graph state and a list of local operations that convert it back to the output stabilizer state.

8. Conclusion

We have presented a compiler to generate an algorithm-specific graph state based on a circuit description. It was shown that once the circuit is decomposed into a Clifford + T gate circuit using standard quantum circuit libraries we can use a teleported T gate implementation of the circuit. This allows the circuit to have a Clifford gate initialisation followed by non-Pauli measurements. The Clifford initialisation is simulated efficiently and an algorithm-specific graph state is extracted. Hardware level implementation of any circuit is now possible by generating this graph state, applying the local corrections and performing non-stabilizer measurements supplemented with tracking the Pauli frame. Our implementation is built for using Cirq circuits as input, but also accepts Q# circuits which are internally converted to Cirq circuits.

One of the main advantages of implementing the circuit in this way is that we are able to separate classical and quantum resources needed for a given quantum algorithm because the Clifford part of the circuit is classically simulated. The graph structure gives us an intuitive understanding of the entanglement structure that the algorithm utilised and how different qubits are connected and if some are superfluous. This leads naturally to questions of optimisation since the generated graph state is not unique. Depending on the physical hardware, different optimisations could be desirable such as reducing edge count, node degree etc similarly to the co-design approach described in [48].

It is an open question whether the algorithm-specific graph state generated using our method is the same as the one generated by the traditional circuit etching method on the 2D cluster state, and is the subject of future work.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://zenodo.org/doi/10.5281/zenodo.10559703.

Acknowledgments

The authors would like to thank the developers of OpenQasm.jl [49] and Graphs.jl [50] for making their code available on a free and open-source basis. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This research was developed with funding from the Defense Advanced Research Projects Agency [under the Quantum Benchmarking (QB) program under Award Nos. HR00112230007 and HR001121S0026 contracts].

Appendix A. Graph conversion of controlled-$V^\dagger$

The evolution of the stabilizer state and the subsequent conversion to a graph state of the pre-measurement state of the circuit in figure 8 is shown in tables 3 and 4.

Table 3. Gate transformations of the controlled-$V^\dagger$ decomposition give in figure 8 excluding final measurements for inputs $|ab\rangle = |00\rangle$.

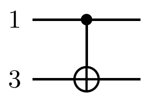
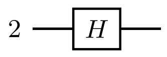
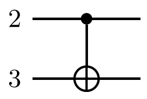
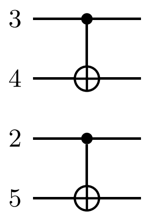
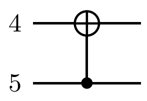
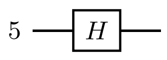
Operation	Stabilizer	State
	$\begin{matrix} Z & I & I & I & I \\ I & Z & I & I & I \\ I & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{matrix}$	$\|00000\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & Z & I & I & I \\ Z & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{matrix}$	$\|00000\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & X & I & I & I \\ Z & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{matrix}$	$\|00000\rangle + \|01000\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & X & X & I & I \\ Z & Z & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{matrix}$	$\|00000\rangle + \|01100\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & X & X & X & X \\ Z & Z & Z & I & I \\ I & I & Z & Z & I \\ I & Z & I & I & Z \end{matrix}$	$\|00000\rangle + \|01111\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & X & X & I & X \\ Z & Z & Z & I & I \\ I & I & Z & Z & Z \\ I & Z & I & I & Z \end{matrix}$	$\|00000\rangle + \|01101\rangle$
	$\begin{matrix} Z & I & I & I & I \\ I & X & X & I & Z \\ Z & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|0000+\rangle + \|0110-\rangle$

Table 4. Graph conversion of the final state in table 3. Input qubits are coloured green and output qubits are coloured blue.

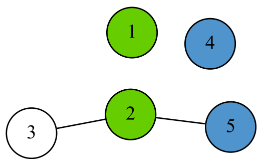
Operation	Stabilizer	State
	$\begin{matrix} Z & I & I & I & I \\ I & X & X & I & Z \\ Z & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|0000+\rangle + \|0110-\rangle$
$1\ -\boxed{H}-$	$\begin{matrix} X & I & I & I & I \\ I & X & X & I & Z \\ X & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|+000+\rangle + \|+110-\rangle$
$rowsum(1 \rightarrow 3)$	$\begin{matrix} X & I & I & I & I \\ I & X & X & I & Z \\ I & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|+000+\rangle + \|+110-\rangle$
$3\ -\boxed{H}-$	$\begin{matrix} X & I & I & I & I \\ I & X & Z & I & Z \\ I & Z & X & I & I \\ I & I & X & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|+0+0+\rangle + \|+1-0-\rangle$
$rowsum(3 \rightarrow 4)$	$\begin{matrix} X & I & I & I & I \\ I & X & Z & I & Z \\ I & Z & X & I & I \\ I & Z & I & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|+0+0+\rangle + \|+1-0-\rangle$
$4\ -\boxed{H}-$	$\begin{matrix} X & I & I & I & I \\ I & X & Z & I & Z \\ I & Z & X & I & I \\ I & Z & I & X & X \\ I & Z & I & I & X \end{matrix}$	$\|+0++\rangle + \|+1-+-\rangle$
$rowsum(5 \rightarrow 4)$	$\begin{matrix} X & I & I & I & I \\ I & X & Z & I & Z \\ I & Z & X & I & I \\ I & I & I & X & I \\ I & Z & I & I & X \end{matrix}$	$\|+0++\rangle + \|+1-+-\rangle$
Final Graph		

Table 5. Gate transformations of the controlled-$V^\dagger$ decomposition give in figure 8 excluding final measurements for inputs $|ab\rangle = |+0\rangle$.

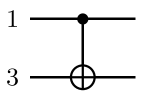
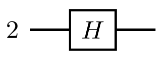
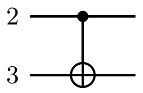
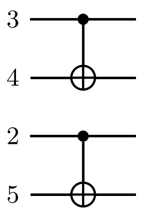
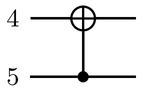
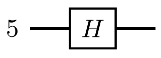
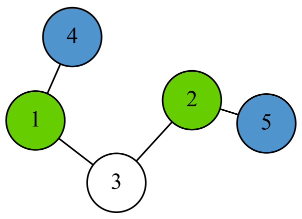
Operation	Stabilizer	State
	$\begin{array}{ccccc} X & I & I & I & I \\ I & Z & I & I & I \\ I & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{array}$	$|+0000\rangle$
	$\begin{array}{ccccc} X & I & X & I & I \\ I & Z & I & I & I \\ Z & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{array}$	$|00000\rangle + |10100\rangle$
	$\begin{array}{ccccc} X & I & X & I & I \\ I & X & I & I & I \\ Z & I & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{array}$	$|00000\rangle + |01000\rangle\ |10100\rangle + |11100\rangle$
	$\begin{array}{ccccc} X & I & X & I & I \\ I & X & X & I & I \\ Z & Z & Z & I & I \\ I & I & I & Z & I \\ I & I & I & I & Z \end{array}$	$|00000\rangle + |01100\rangle\ |10100\rangle + |11000\rangle$
	$\begin{array}{ccccc} X & I & X & X & I \\ I & X & X & X & X \\ Z & Z & Z & I & I \\ I & I & Z & Z & I \\ I & Z & I & I & Z \end{array}$	$|00000\rangle + |01111\rangle\ |10110\rangle + |11001\rangle$
	$\begin{array}{ccccc} X & I & X & X & I \\ I & X & X & I & X \\ Z & Z & Z & I & I \\ I & I & Z & Z & Z \\ I & Z & I & I & Z \end{array}$	$|00000\rangle + |01101\rangle\ |10110\rangle + |11011\rangle$
	$\begin{array}{ccccc} X & I & X & X & I \\ I & X & X & I & Z \\ Z & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{array}$	$|0000+\rangle + |0110-\rangle\ |1011+\rangle + |1101-\rangle$

Table 6. Graph conversion of the final state in table 5.

Operation	Stabilizer	State
	$\begin{matrix} X & I & X & X & I \\ I & X & X & I & Z \\ Z & Z & Z & I & I \\ I & I & Z & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|0000+\rangle + \|0110-\rangle\ \|1011+\rangle + \|1101-\rangle$
3 —$\boxed{H}$—	$\begin{matrix} X & I & Z & X & I \\ I & X & Z & I & Z \\ Z & Z & X & I & I \\ I & I & X & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|00+0+\rangle + \|01-0-\rangle\ \|10-1+\rangle + \|11+1-\rangle$
rowsum$(3 \rightarrow 4)$	$\begin{matrix} X & I & Z & X & I \\ I & X & Z & I & Z \\ Z & Z & X & I & I \\ Z & Z & I & Z & X \\ I & Z & I & I & X \end{matrix}$	$\|00+0+\rangle + \|01-0-\rangle\ \|10-1+\rangle + \|11+1-\rangle$
4 —$\boxed{H}$—	$\begin{matrix} X & I & Z & Z & I \\ I & X & Z & I & Z \\ Z & Z & X & I & I \\ Z & Z & I & X & X \\ I & Z & I & I & X \end{matrix}$	$\|00+++\rangle + \|01-+-\rangle\ \|10--+\rangle + \|11+--\rangle$
rowsum$(5 \rightarrow 4)$	$\begin{matrix} X & I & Z & Z & I \\ I & X & Z & I & Z \\ Z & Z & X & I & I \\ Z & I & I & X & I \\ I & Z & I & I & X \end{matrix}$	$\|00+++\rangle + \|01-+-\rangle\ \|10--+\rangle + \|11+--\rangle$
Final Graph		

Appendix B. Graph conversion algorithm

Algorithm 1. The graph conversion algorithm, with $O(n^3)$ runtime.

```
input  : Tableau $X$, $Z$, $P$.
output : Adjacency matrix $A$.
begin
    /* Make $X$-block full rank, $O(n^3)$ */
    for $i \leftarrow 1$ to $n$ do
        /* How many $X$'s are in the lower-triangular part of the $i$th column? */
        $\sigma = \sum_{k=i}^{n} X_{k,i}$
        /* If there aren't any, pull them over from the $Z$-block */
        if $\sigma = 0$ then
            H($i$)
        /* Perform any necessary row-swap to bring a leading $X$ onto the diagonal */
        for $j \leftarrow i$ to $n$ do
            if $X_{j,i} = 1$ then
                rowswap($i \leftrightarrow j$)
                break
        /* Eliminate $X$'s below the diagonal */
        for $j \leftarrow i+1$ to $n$ do
            if $X_{j,i} = 1$ then
                rowadd($i \rightarrow j$)
    /* Diagonalize $X$-block, $O(n^3)$ */
    for $i \leftarrow n-1$ to $1$ do
        for $j \leftarrow n$ to $i+1$ do
            if $X_{i,j} = 0$ then
                rowadd($j \rightarrow i$)
    /* Make $Z$-block diagonals zero and correct phases, $O(n)$ */
    for $i \leftarrow 1$ to $n$ do
        if $Z_{i,i} = 1$ then
            P$^\dagger$($i$)
        if $P_i = -1$ then
            Z($i$)
    /* $Z$-block contains adjacency matrix */
    return $A \leftarrow Z$
```

ORCID iDs

Madhav Krishnan Vijayan https://orcid.org/0000-0002-7863-7437
Alexandru Paler https://orcid.org/0000-0002-1536-8858

References

[1] Wang D-S 2021 A comparative study of universal quantum computing models: toward a physical unification *Quantum Eng.* **3** e85
[2] Raussendorf R and Briegel H J 2001 A one-way quantum computer *Phys. Rev. Lett.* **86** 5188
[3] Nielsen M A 2006 Cluster-state quantum computation *Rep. Math. Phys.* **57** 147
[4] Raussendorf R, Browne D E and Briegel H J 2003 Measurement-based quantum computation on cluster states *Phys. Rev.* A **68** 022312
[5] Paler A, Devitt S, Nemoto K and Polian I 2014 Software-based Pauli tracking in fault-tolerant quantum circuits *2014 Design, Automation Test in Europe Conf. Exhibition (DATE)* pp 1–4
[6] Evans A, Omonije S, Soulé R and Rand R 2022 MCBeth: a measurement based quantum programming language (arXiv:2204.10784)
[7] Zhang H, Wu A, Wang Y, Li G, Shapourian H, Shabani A and Ding Y 2022 A compilation framework for photonic one-way quantum computation (arXiv:2209.01545)
[8] Gottesman D 1998 The Heisenberg representation of quantum computers *Int. Conf. on Group Theoretic Methods in Physics* p 9807006
[9] Aaronson S and Gottesman D 2004 Improved simulation of stabilizer circuits *Phys. Rev.* A **70** 052328
[10] Anders S and Briegel H J 2006 Fast simulation of stabilizer circuits using a graph-state representation *Phys. Rev.* A **73** 022334
[11] Paler A, Polian I, Nemoto K and Devitt S J 2017 Fault-tolerant, high-level quantum circuits: form, compilation and description *Quantum Sci. Technol.* **2** 025003
[12] Herr D, Nori F and Devitt S J 2017 Lattice surgery translation for quantum computation *New J. Phys.* **19** 013034
[13] Van den Nest M, Dehaene J and De Moor B 2004 Graphical description of the action of local Clifford transformations on graph states *Phys. Rev.* A **69** 022316
[14] Gidney C 2021 Stim: a fast stabilizer circuit simulator *Quantum* **5** 497
[15] Coecke B and Duncan R 2011 Interacting quantum observables: categorical algebra and diagrammatics *New J. Phys.* **13** 043016
[16] Backens M, Miller-Bakewell H, de Felice G, Lobski L and van de Wetering J 2021 There and back again: a circuit extraction tale *Quantum* **5** 421
[17] Lau J W Z, Lim K H, Shrotriya H and Kwek L C 2022 NISQ computing: where are we and where do we go? *AAPPS Bull.* **32** 27
[18] Vijayan M K 2023 Jabalizer v0.1.1 *Zenodo* (available at: https://zenodo.org/doi/10.5281/zenodo.10559703)

[19] Bennett C H, DiVincenzo D P, Smolin J A and Wootters W K 1996 Mixed-state entanglement and quantum error correction *Phys. Rev.* A **54** 3824
[20] Calderbank A R, Rains E M, Shor P W and Sloane N J A 1997 Quantum error correction and orthogonal geometry *Phys. Rev. Lett.* **78** 405
[21] Gottesman D 1996 Class of quantum error-correcting codes saturating the quantum Hamming bound *Phys. Rev.* A **54** 1862
[22] Gottesman D 2009 *An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation* (American Mathematical Society) (available at: https://mathscinet.ams.org/mathscinet/relay-station?mr=2762145)
[23] Bombín H 2014 Structure of 2D topological stabilizer codes *Commun. Math. Phys.* **327** 387
[24] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge University Press)
[25] Raussendorf R and Wei T-C 2012 Quantum computation by local measurement *Annu. Rev. Condens. Matter Phys.* **3** 239
[26] Raussendorf R and Briegel H 2002 Computational model underlying the one-way quantum computer *Quantum Inf. Comput.* **2** 443
[27] Raussendorf R, Harrington J and Goyal K 2007 Topological fault-tolerance in cluster state quantum computation *New J. Phys.* **9** 199
[28] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H 1995 Elementary gates for quantum computation *Phys. Rev.* A **52** 3457
[29] Liu Y, Long G L and Sun Y 2008 Analytic one-bit and CNOT gate constructions of general n-qubit controlled gates *Int. J. Quantum Inf.* **06** 447
[30] Schlingemann D 2001 Stabilizer codes can be realized as graph codes (arXiv:quant-ph/0111080)
[31] Van den Nest M, Dehaene J and De Moor B 2004 Efficient algorithm to recognize the local Clifford equivalence of graph states *Phys. Rev.* A **70** 034302
[32] West D B 2001 *Introduction to Graph Theory* vol 2 (Prentice hall Upper Saddle River)
[33] Adcock J C, Morley-Short S, Dahlberg A and Silverstone J W 2020 Mapping graph state orbits under local complementation *Quantum* **4** 305
[34] Høyer P, Mhalla M and Perdrix S 2006 Resources required for preparing graph states *Algorithms Andcomputation (Lecture Notes in Computer Science)* (Springer) pp 638–49
[35] Bartolucci S *et al* 2021 Fusion-based quantum computation (arXiv:2101.09310)
[36] Huang J, Chen X Li X and Wang J 2023 Chip-based photonic graph states *AAPPS Bull.* **33** 14
[37] Ferguson R, Dellantonio L, Balushi A A, Jansen K, Dür W and Muschik C 2021 Measurement-based variational quantum eigensolver *Phys. Rev. Lett.* **126** 220501
[38] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 Surface codes: towards practical large-scale quantum computation *Phys. Rev.* A **86** 032324
[39] Horsman C, Fowler A G, Devitt S and Meter R V 2012 Surface code quantum computing by lattice surgery *New J. Phys.* **14** 123011
[40] Bombin H 2010 Topological order with a twist: ising anyons from an Abelian model *Phys. Rev. Lett.* **105** 030403
[41] Litinski D 2019 A game of surface codes: large-scale quantum computing with lattice surgery *Quantum* **3** 128
[42] Ghosh S, Deshpande A, Hangleiter D, Gorshkov A V and Fefferman B 2023 Complexity phase transitions generated by entanglement *Phys. Rev. Lett.* **131** 030601
[43] Lu D *et al* 2017 Enhancing quantum control by bootstrapping a quantum processor of 12 qubits *npj Quantum Inf.* **3** 45
[44] Li J, Yang X Peng X and Sun C-P 2017 Hybrid quantum-classical approach to quantum optimal control *Phys. Rev. Lett.* **118** 150503
[45] Tilly J *et al* 2022 The variational quantum eigensolver: a review of methods and best practices *Phys. Rep.* **986** 1–128
[46] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 Parameterized quantum circuits as machine learning models *Quantum Sci. Technol.* **4** 043001
[47] Heim B, Soeken M, Marshall S, Granade C, Roetteler M, Geller A, Troyer M and Svore K 2020 Quantum programming languages *Nat. Rev. Phys.* **2** 709
[48] Li G, Wu A, Shi Y, Javadi-Abhari A, Ding Y and Xie Y 2021 On the co-design of quantum software and hardware *Proc. 8th Annual ACM Int. Conf. on Nanoscale Computing and Communication (NANOCOM '21)* (Association for Computing Machinery) pp 1–7
[49] Luo X-Z, Liu J-G, Zhang P and Wang L 2020 Yao.jl: extensible, efficient framework for quantum algorithm design *Quantum* **4** 341
[50] Fairbanks J, Besançon M, Simon S, Hoffiman J, Eubank N and Karpinski S 2021 JuliaGraphs/Graphs.jl: an optimized graphs package for the Julia programming language (available at: https://github.com/JuliaGraphs/Graphs.jl/)