Fusion-based Foliated Quantum Codes

M. Gorman* and T. Farrelly
School of Mathematics and Physics, The University of Queensland, QLD 4072, Australia

We present an algorithm that generalizes the construction of arbitrary foliated CSS codes by utilizing fusions of resource states. An example application of the algorithm is shown to produce a fusion network that returns the foliated Steane code. We also go beyond the algorithm to improve the number of qubits and measurements required to implement this fusion network.

Introduction. To realize useful, large-scale quantum computers, it is necessary to suppress errors [1]. Although passive measures, like reducing noise, slows down the accumulation of errors, the inevitability of imperfect hardware and a noisy environment necessitates the employment of active error correction measures. As such, it is important to produce fault-tolerant quantum error correction schemes that can easily be leveraged in practical applications.

In this paper, we provide an overview of measurement-based quantum computation, fusion-based quantum computation, and foliated quantum codes. Also, we construct two different implementations of the foliated Steane code using fusion measurements. We then generalize this method, allowing for the production of arbitrary foliated CSS codes using fusions of resource states.

Measurement-based Quantum Computing. The measurement of a qubit is an irreversible procedure that destructively decoheres the qubit's state [2]. Because quantum devices are necessarily coupled to a noisy environment that will effectively measure a system's qubits, without quantum error correction procedures, the qubits' states will decohere and the encoded information will be lost. As such, quantum decoherence has been viewed as a hurdle in building useful quantum devices.

Despite this, measurements have found applications in quantum computing beyond just the final readout of results. In formulating a model called *measurement-based quantum computing* (MBQC), H. J. Briegel *et al.* showed that universal quantum computations can be performed solely through measurements on select qubits in particular bases [3]. It is worth noting that information is not lost as a result of these measurements because some qubits remain unmeasured, and because the measurement outcomes are recordable, the importance of which is established later.

MBQC differs from the circuit-based model of quantum computing, where a state is evolved by unitary operations and measured at the circuit's end. A simple demonstration of this can be found in the fact that MBQC has no clear classical analogue [2] while the circuit-based model was built as a quantum generalization of classical computing [4].

There are two constants in the design of a MBQC system. (1) The non-input qubits, auxiliary to the computation, are prepared in the $+1$ eigenstate of Pauli X, $|+\rangle$. And (2) the qubits are entangled with one another through controlled-phase gates to generate a *cluster state*; an entangled state with many qubits [2–4]. Fig. 1 serves as a pedagogical example of such a design. The circuit corresponds to a single input qubit and a single auxiliary qubit entangled by a controlled-phase gate to form a cluster state. The input qubit is then measured in the X basis, and the auxiliary qubit is in the state of the input qubit, up to some known unitaries.

Because useful MBQC requires many qubits, it is often simplest to consider a *non-circuit* schematic of the computations, and in calculations to consider the logical operators and stabilizers in the *Heisenberg picture*; the operators are evolved by measurements in time, but the state vectors remain unchanged. To illustrate the convenience of the non-circuit diagrams for MBQC, in Fig. 2 we provide an example in this picture that performs quantum teleportation [2].

Furthermore, the example provided in Fig. 2 is the identity in Fig. 1 applied twice. Hence, the unmeasured qubit is in a state $X^{m_2}Z^{m_1}|\psi\rangle$ where $|\psi\rangle$ is the original input state, $m_{i\in\{1,2\}}$ are the measurement outcomes, and X and Z are Pauli operators. These measurement dependent Pauli operators acting on the input state are called *Pauli biproducts* [2, 3]. To see the importance of Pauli biproducts, consider two single-qubit unitary operations on a general qubit state. The state outputted will be $U_2U_1|\psi\rangle$ where $U_{i\in\{1,2\}}$ are the unitary operations, and $|\psi\rangle$ is the original state. But, if these operations were implemented using MBQC, the state outputted would

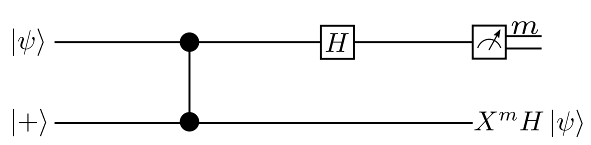

FIG. 1. A circuit identity for the smallest possible example of MBQC. Initially we consider the first qubit to be in an arbitrary state, $|\psi\rangle$, and the second qubit to be in the $+1$ eigenstate of Pauli X, $|+\rangle$. A controlled-phase gate is applied across both qubits, and a Hadamard, H, is then applied to the first qubit. The first qubit is then measured in the Pauli Z basis with outcome $m \in \{0,1\}$. When this measurement is combined with the earlier Hadamard gate, it is equivalent to a measurement in the X basis. After this procedure, the second qubit is known to be in the state $X^mH|\psi\rangle$. Adapted from [4].

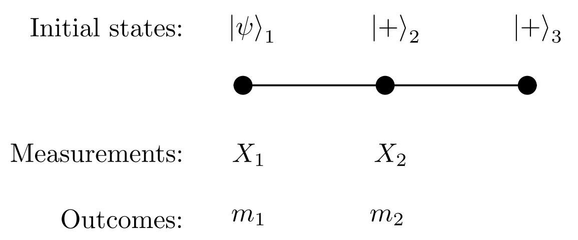

FIG. 2. Quantum teleportation using MBQC. The back circles are the qubits, and the black lines joining them are controlled-phase gates. The first qubit is in an arbitrary state, $|\psi\rangle$, while the other two qubits are prepared in the state $|+\rangle$. The measurements X_1 and X_2 are performed with outcomes m_1 and m_2 respectively. Using the identity in Fig. 1, it is clear that after the measurements the third qubit will be in the state $(X^{m_2}H)(X^{m_1}H)|\psi\rangle = X^{m_2}Z^{m_1}|\psi\rangle$, where we have used $HX = ZH$ and the fact that H is involutory. Adapted from [2].

be $P_2U_2P_1U_1|\psi\rangle$ where $P_{i\in\{1,2\}}$ are known Pauli biproducts [2]. Because the Pauli biproducts don't necessarily commute with the unitarizes, the output state in MBQC is not necessarily the desired result, and so Pauli biproducts present a challenge. However, because the Pauli biproducts are known from the measurement outcomes, by keeping track of them and modifying the unitary operations accordingly, the desired output state can be found; in a process called *adaptive measurements* [2, 4]. More specifically, say a Pauli biproduct, P, and a unitary, U, have the commutation relation $UP = PU'$ where U' is an adapted version of the unitary, then $P_2U_2'P_1U_1|\psi\rangle = P_2P_1U_2U_1|\psi\rangle$ [2]. That is, by adapting the measurements, the output will be the desired state up to some known Pauli biproduct that can be used to accurately interpret the result. This is clearly generalized to larger computations; with more operations. Lastly, as a simple example, let U_1 be any unitary, set $U_2 = R_x(\theta)$ to be an x rotation by Euler angle θ. Let P_2 be any Pauli biproduct and say from the measurement outcomes used to apply U_1 it is known that $P_1 = Z$. Well, the output would be $P_2R_x(\theta)ZU_1|\psi\rangle$. Observing that $R_x(\theta)Z = ZR_x(-\theta)$ we can see that the Pauli biproduct has corrupted the desired result. If, however we were to set $U_2 = R_x(-\theta)$ because we know that $P_1 = Z$ and intend to have $R_x(\theta)$ applied to the state inputted, then the output would be $P_2R_x(-\theta)ZU_1|\psi\rangle = P_2ZR_x(\theta)U_1|\psi\rangle =$ (Known Pauli biporduct)(Desired opperation) $|\psi\rangle$. That is, by adapting the measurements, the desired operation has been applied to an arbitrary state up to some known Pauli biproducts that can be accounted for to find the intended result.

In the picture of MBQC, computations are often constructed using a square lattice shaped cluster state [2–4]. The cluster state must be two-dimensional in order to implement 2-qubit gates, which, along with a carefully selected discrete set of single qubit gates, are sufficient for universal quantum computation [2]. A scheme for general, adaptive single-qubit unitary operations is constructed using measurements on one-dimensional cluster states, as explained in [2].

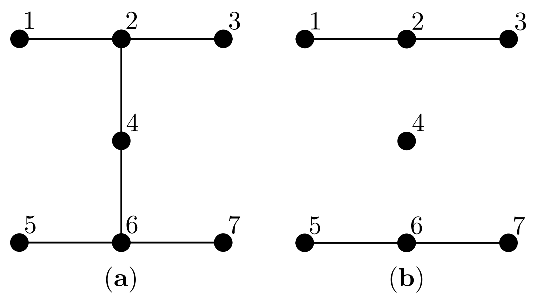

FIG. 3. Removal of a qubit from a cluster through a Z measurement. The back dots are qubits in the $|+\rangle$ state, and the black lines joining the qubits are controlled-phase gates. In (a) we present a cluster state. If the measurement Z_4 was performed, the stabilizers that remain will be those of the state presented in (b) and the state of the system will be consistent with qubits 2 and 6 having instead been prepared in the state $|(-1)^m\rangle$ where m was the outcome of the measurement, Z_4. Adapted from [2].

To implement a universal gate set, we also need a 2-qubit gate. However, because keeping all the qubits in the square lattice cluster state can cause unwanted entanglement between qubits, we must first consider how to remove qubits from the cluster. To do this, we perform a Z measurement on the desired qubit, which effectively severs the adjacent entangling, controlled-phase gates [2, 3]. Note that after the measurement, the state of the system will be consistent with the adjacent connected qubits having been prepared in the state $|(-1)^m\rangle$ where m was the outcome of the Z measurement [2]. A demonstration of this is provided in Fig. 3.

Empowered with the ability to remove qubits from larger cluster states, we can now consider the implementation of a controlled-phase gate in MBQC. One possible implementation is given in Fig. 4, where we can show that after the measurements, the logical operators of the circuit are consistent with those of a controlled-phase gate.

With reference to how the controlled-phase gate in Fig. 4 has input and output qubits, it should be clear that when these gates are implemented with the MBQC single-qubit unitaries, universal quantum computation is possible [3]. That is to say, in this construction, the input states can be viewed as travelling down the lattice with the desired operations applied to them as the measurements are performed.

To see some interesting features of MBQC we must first define a *Clifford operation*; a gate C that has $\forall P$ the property $PC = CP'$ where P and P' are elements of the Pauli group e.g. the Hadamard gate is a Clifford operation [2]. Using this fact, without considering adaptive measurements, Pauli biproducts can be commuted out in a protocol with just Clifford operations [2]. That

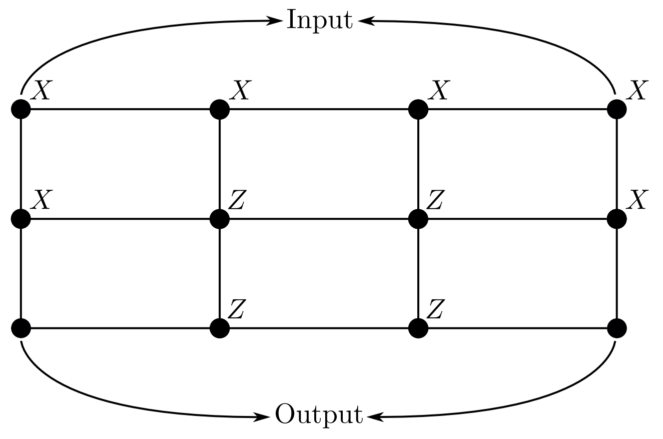

FIG. 4. Example of a controlled-phase gate, CZ, in MBQC. The back circles are qubits, the black lines joining them are controlled-phase gates, and the Xs and Zs are measurements. The qubits labelled input are in an arbitrary two-qubit state $|\psi\rangle$ and the remaining qubits are prepared in the $|+\rangle$ state. After the measurements are performed, the qubits labelled output are in the state $P_{1,2}CZ\,|\psi\rangle$ where $P_{1,2}$ are the measurement outcome dependent Pauli biproducts. This can be verified by seeing that the logical operators are consistent with a controlled-phase gate, CZ, after the measurements are performed. That is, after the measurements are performed, the two Z logical operators will be $\bar{Z}_1 = \pm Z_{\text{out }1}$ and $\bar{Z}_2 = \pm Z_{\text{out }2}$, and the two X logical operators will be $\bar{X}_1 = \pm X_{\text{out }1} Z_{\text{out }2}$ and $\bar{X}_2 = \pm Z_{\text{out }1} X_{\text{out }2}$ where out 1 and out 2 are each of the output qubits and the $\pm$ phase is the measurement outcome dependent biproduct. Note that because all the measurements are applied to disjoint sets of qubits, the measurements commute, and can thus be performed simultaneously. Adapted from [2].

is, because, by definition, Clifford gates do not need to be adaptive, and because all measurements in MBQC commute because they act on a disjoint set of qubits, the MBQC circuits with just Clifford operations have a constant depth irrespective of the number of measurements [2]. This presents a benefit over the circuit-based model of quantum computing. Explicitly, the time to measure a set of Clifford operations in MBQC is independent of the number of operations, while in a circuit model, the depth increases linearly with the number of gates [2]. Hence, the qubits spend less time in memory and so they are less prone to errors. However, this is not a significant benefit as classically simulating circuits of Clifford gates has the same time complexity [5]; because in the MBQC approach, it is necessary to account for the Pauli biproducts, which has logarithmic time complexity [2]. Another potential benefit of MBQC is that it offers a different implementation of a device's architecture. An example of where a MBQC approach varies from the circuit-based model of quantum computing is that the final measurement in a circuit can be performed before any computations in MBQC [2]. This is because, in MBQC, the measurements are designed to commute,

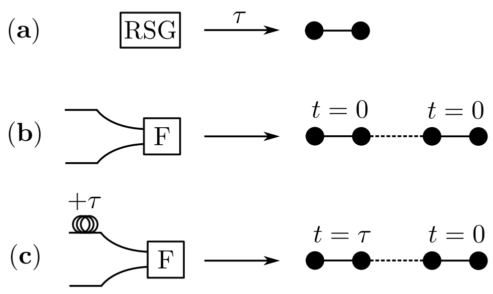

FIG. 5. Circuit components used in FBQC. (a) A resource state generator (RSG) which continuously produces resource states at a rate of $1/\tau$. In this example the RSG produces a 2-qubit resource state, but RSGs can produce arbitrary resource states. (b) A fusion device that takes in two cluster states and performs a reconfigurable fusion. In this example, the dashed line represents the fusion of two 2-qubit resource states both created by an RSG at the same time, t. An example of this fusion is provided in Fig. 6. (c) A fusion device with a time delay of $+\tau$ on one resource state. These time delays allow for the construction of large fusion networks using only a few RSGs, as seen in Fig. 7.

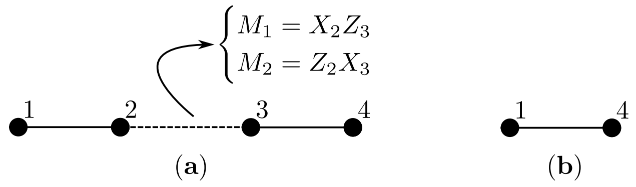

FIG. 6. An example of a fusion. (a) This fusion network is made from two resource states: one consisting of qubits 1 and 2, and the other with qubits 3 and 4. The fusion measurement is indicated by the dashed line connecting qubits 2 and 3. The fusion is performed by measuring $M_1 = X_2 Z_3$ and $M_2 = Z_2 X_3$. After the fusion, the stabilizers, up to some measurement dependent phase, will be consistent with the system being in the state given in (b). Note also that other fusion measurements exist, such as $M_1 = X_1 X_2$ and $M_2 = Z_1 Z_2$. Adapted from [6].

and so the outputs of the subsequent measurements in the computation can be interpreted as updating the final measurement. Although this example does not seem particularly useful, it is an interesting result that highlights how drastically the computation can change in MBQC, which, as previously stated, could potentially offer benefits in the implementation of quantum hardware.

Fusion-based Quantum Computing. *Fusion-based quantum computation* (FBQC) is a model of universal quantum computation built from resource states and fusion measurements [6].

A *resource state* is a type of cluster state that has a structure identical to the other cluster states in the system [6]. It can be viewed as a primitive from which larger entangled states can be built. These resource states are produced by a device called a *resource state generator* (RSG), which outputs an identical cluster state on a timescale, τ, Fig. 5a [6]. In linear optics, a RSG can

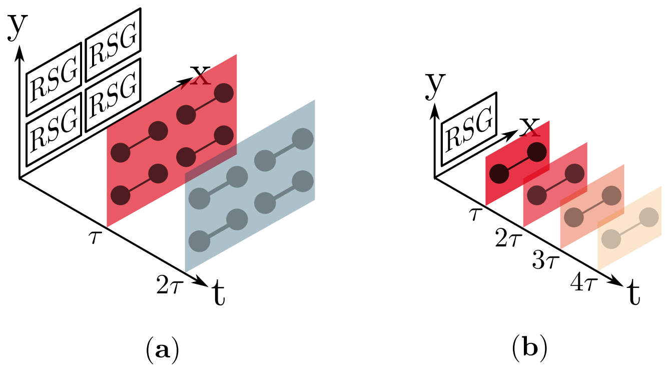

FIG. 7. Two constructions for producing 2-qubit resource states for a larger fusion network. (a) Two-dimensional sheets of resource states are made at a time. (b) A single resource state is made at a time, which can be used to produce the same fusion networks that (a) can produce. Note that schemes reliant on producing fewer resource states per time step are more error prone because their qubits will be required to wait in memory until enough time cycles have passed to yield sufficient resource states to produce the desired fusion network.

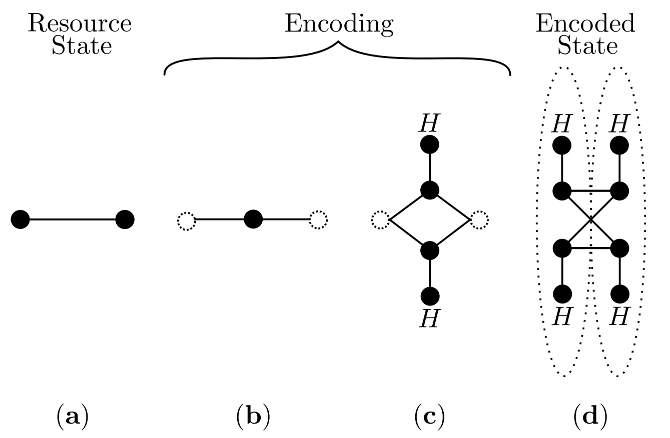

FIG. 8. The encoding of resource states to suppress photon loss and fusion failure errors - or their analogue in other architectures [6]. (a) An example 2-qubit resource state. The black circles are the qubits and the black line is a controlled-phase gate. (b) A single qubit with neighboring qubits, denoted by the dotted circles. The single qubit in (b) is encoded as a $(2, 2)$-Shor code in (c), where H represents a Hadamard on the respective qubits. Applying the encoding of (b) and (c) to the resource state in (a) gives the encoded state (d). In (d) the dotted ellipses show the locations of the unencoded qubits from the original resource state. Adapted from [6].

be implemented using fusions, or other projective measurements, of smaller, easier to produce states called *seed states* [6].

Fusions are entangling measurements that act on two different resource states. Fusions are used to build larger, more complicated states from the resource states available [6]. In physical implementations of FBQC, fusions are performed by *fusion devices*, Fig. 5b [6]. To see how fusion measurements are entangling, consider Fig. 6.

As stated previously, resource states and fusion measurements form the basis of FBQC. That is, by performing fusions on a set of resource states, in what is called a *fusion network*, one can construct the large-scale entangled states that are necessary for fault-tolerant universal quantum computation [6]. To achieve this, the authors of [6] consider crystal lattice-like codes [7, 8]. In this regime, the logical qubits can be introduced by taking Z measurements to create primal and dual boundaries in the syndrome graph [6]. The fault-tolerance comes from performing fusion measurements that belong to the stabilizers of a code [6]. And the universal gate set is implemented by modifying the bases of a selection of fusion measurements, and by using state injection and magic state distillation protocols [6]. This approach is favorable as the circuits - for fault-tolerance and universal logic - are of low quantum depth [6].

With reference to the fusion devices in Fig. 5b and Fig. 5c, we can see that fusions do not need to occur on qubits generated at the same time [6]. As such, another benefit of fusion networks is that they can be implemented using various architectures. For example, the resource states of a three-dimensional fusion network could be created all at once, one two-dimensional sheet at a time, or one resource state at a time, to name a few examples. A visualization of this feature is provided in Fig. 7.

It is worth noting that while fusion networks can fault-tolerantly encode logical qubits, the fusions themselves are susceptible to errors. While the exact error model is hardware dependent, it has been shown that encoding the qubits before fusions drastically improves the scheme's threshold [6]. An example of an encoded resource state is provided in Fig. 8. This encoding uses a $(2, 2)$-Shor code, Fig. 8c, where the 2s represent the distance of the X and Z logical operators.

Moreover, as mentioned earlier, fusion networks have been used [6] to construct codes not confined to the foliation of two-dimensional codes [7, 8]. While these unfoliatable codes have the potential to provide improved thresholds, they require more operations to prepare and hence are more prone to errors [7]. Furthermore, because these codes have not been compared to foliated quantum codes under a realistic noise model, the thresholds presented are not necessarily a true representation of their error correction performance. This is noted in [7] when the authors acknowledge that there is an architecture dependent optimization problem involving the code's achievable error tolerance and the difficulty of accurately preparing the code. For this reason, it is worthwhile also studying how to construct foliated quantum codes with fusions of resource states, as foliated codes have been the subject of more study and their performance is better understood.

Foliated Quantum Codes. A CSS code can be clusterized by taking the data qubits associated with each of

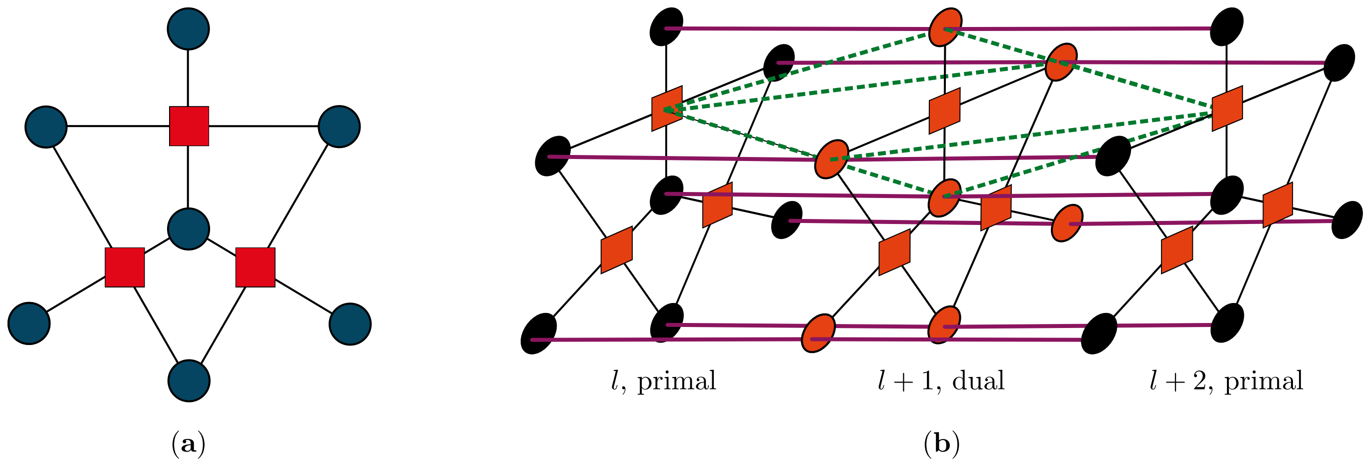

FIG. 9. A foliated Steane code. (a) A *progenitor cluster state* of the Steane code where the qubits are in an eigenstate of X. The black lines are controlled-phase gates, designed so that when the ancilla qubits, marked by red squares, are measured in the X basis, the data qubits, denoted by the blue circles, are projected onto an eigenstate of the Steane code's stabilizer generators, up to a phase dependent on the measurement outcomes. (b) A foliated Steane code with 3 layers. The purple lines are controlled-phase gates that connect the layered codes. By performing X measurements of the qubits marked with orange, the unmeasured qubits on the boundaries will be in a bell state encoded by the Steane code. Moreover, the green dotted lines show a parity check of the code, as detailed in [9, 10]. The encoding can be verified (in the Heisenberg picture) by considering how the stabilizers evolve with the measurements and by showing that the final stabilizers outputted give the Steane code's stabilizers and logical operators up to a measurement outcome dependent phase. Adapted from [9, 10].

the code's Z stabilizers and entangling them with a common ancilla qubit through controlled-phase gates [9, 10]. Then, by then measuring the ancilla qubits in the X basis, the cluster state will be in an eigenstate of the desired code's Z stabilizers [9, 10]. An example of a clusterized Steane code is given in Fig. 9a.

After clusterization, a code can then be *foliated*; taking multiple collections of a clusterized code and its clusterized dual code and joining corresponding data qubits, between the primal and dual graphs, with controlled-phase gates. By then measuring the ancilla qubits on the foliated code's boundaries and all the qubits in the foliated code's bulk, what remains is a bell state encoded by the desired code [9, 10]. An example of a foliated Steane code is given in Fig. 9b.

The foliation of codes is a generalisation of Raussendorf's three-dimensional cluster state, and so foliated codes inherit the potential for high thresholds, and fault-tolerant, universal quantum computation [9–11].

Fusion-based Foliated Quantum Codes. In this section we will construct a foliated Steane code by fusing resource states. We will also generalize this to foliate arbitrary CSS codes.

To produce a larger cluster state, we must first choose a resource state. In Fig. 10a and Fig. 10b we present one such choice of resource state. In Fig. 10a we show how the resource state can be reduced to a 3-qubit chain-like cluster state, which is important in entangling two other cluster states. Also, in Fig. 10b, we present a simple reinterpretation of the resource state to make a clearer connection to the ancilla qubits of the clusterized Steane

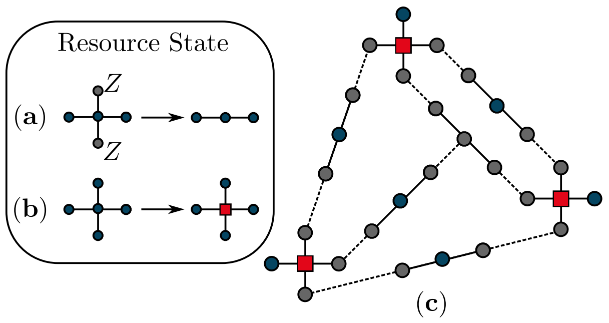

FIG. 10. Modification of a 5-qubit resource state through single-qubit measurements and reinterpretation of its qubits, and the clusterization of the Steane code through fusions of said resource state. The blue circles are unmeasured qubits, the grey circles are the measured qubits, the red squares are ancilla qubits, the Zs are measurements in the Z basis, and the black lines are controlled-phase gates. (a) By taking two Z measurements on any of the outer qubits of the resource state, a 3-qubit chain-like cluster state can be created. (b) By designating the center qubit of the resource state to be an ancilla qubit, we can perhaps more easily see how to construct a fusion network that produces a foliated quantum code. (c) A fusion network that produces the clusterized Steane code, Fig. 9a, using the 5-qubit resource state provided. This method requires 10 fusions, 10 single-qubit Z measurements, and 8 resource states for a total of 40 qubits. Note that this figure can be equivalently drawn two other ways and that we are left with 7 blue, unmeasured qubits representing the Steane Code, and 3 red, ancilla qubits used in the process of encoding, which is consistent with Fig. 9a.

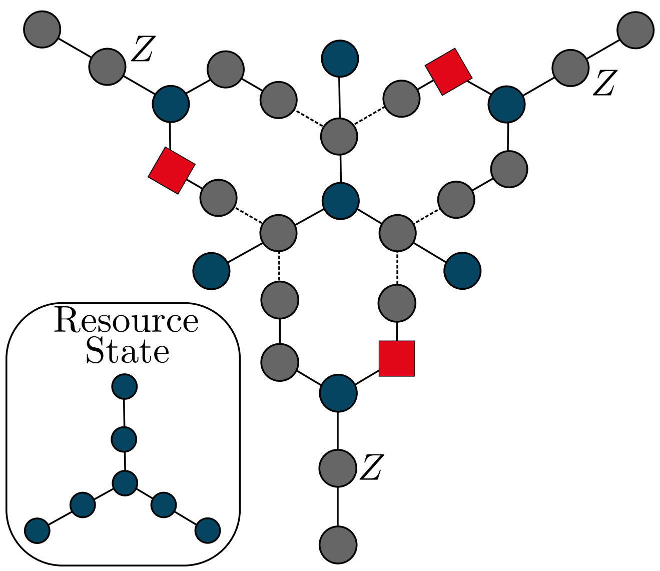

FIG. 11. An optimized fusion network that produces the Steane code, Fig. 9a, using the 7-qubit resource state shown in the figure. The dashed lines represent the fusions, and as in Fig. 10, the blue circles are unmeasured qubits, grey circles are measured qubits or qubits that are no longer part of the cluster, the red squares are a possible choice of locations for the ancilla qubits, the Zs are measurements in the Z basis, and the black lines are controlled-phase gates. This method requires 6 fusions, 3 single-qubit Z measurements, and 4 resource states for a total of 28 qubits, hence it is optimized with respect to Fig. 10c. Again, see that we are left with 7 blue, unmeasured qubits representing the Steane Code, and 3 red, ancilla qubits used in the process of encoding, which is consistent with Fig. 9a.

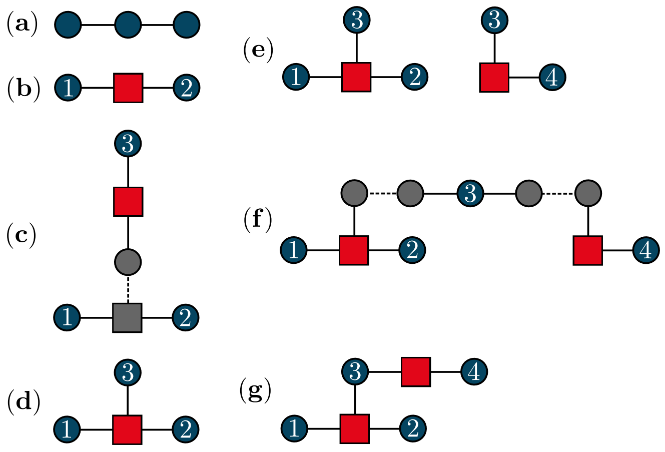

FIG. 12. Selected steps in the generalized algorithm for producing fusion-based foliated quantum codes. The squares are ancilla qubits, the circles are data qubits, and the black lines are controlled-phase gates. Qubits greyed out are measured in that step, and the red and blue qubits are unmeasured in that step. The dotted black lines are fusions. Qubits are labelled with a number i where Z_i is a measurement that appears in a stabilizer. (a) Our choice of resource state: the 3-qubit chain-like cluster state. (b) Reinterpretation of the resource state. (c) Fusing additional resource states to generate a larger cluster state. (d) A cluster state that represents one stabilizer. (e) Two cluster states that represent the two stabilizers in the code. (f) The fusion network to clusterize the code. (g) The clusterized code.

code in Fig. 9a. In Fig. 10c, we present one possible design for implementing the clusterized Steane code using the single resource state in Fig. 10a and Fig. 10b. To then implement the foliated Steane code, two of the 3-qubit chain-like cluster states in Fig. 10a must be fused to produce a 4-qubit chain-like cluster state, which can then be fused with the two corresponding data qubits of two separate clusterized Stenae codes. By repeating this for every data qubit on each clusterized code, the foliated Steane code is produced. That is, the foliated Steane code can be produced entirely by taking fusions of a single resource state.

It is worth noting that to produce the clusterized Steane code as we have in Fig. 10c, we are required to use 40 qubits, 10 fusions, and 10 single-qubit Z measurements. However, by using different resource states, we can optimize these numbers. For example, the construction of the Steane code provided in Fig. 11 only requires 28 qubits, 6 fusions, and 3 single-qubit Z measurements. Note that more qubits are necessary for error tolerant fusions, see Fig. 8.

To generalize our method of producing fusion-based foliated quantum codes to arbitrary CSS codes, we must again choose a resource state. The choice of resource state will dictate the efficiency of the code's construction with respect to the required number of qubits and measurements. The resource state we will use is the 3-qubit chain-like cluster state seen in Fig. 12a as it is the simplest case to consider. Perhaps a larger resource state, like the star-like cluster state in Fig. 10a and Fig. 10b, could be used to optimize our approach, but when constructing the resource states from seed states, the higher complexity of larger resource states could very well end up requiring more qubits and measurements than just using the 3-qubit chain-like resource state.

The generalized method for fusion-based foliated quantum codes is split into three algorithms, which are provided in the Appendix. In explaining each algorithm, we will use a toy example: a CSS code that has Z stabilizers $Z_1Z_2Z_3$ and Z_3Z_4.

Algorithm 1, in Appendix A, requires the code's Z stabilizers to be input. It selects one of these stabilizers and then takes the resource state, Fig. 12a, and denotes the center qubit to be an ancilla and assigns each of the two edge qubits with a number corresponding to two of the measurements in the selected stabilizer, say 1 and 2 from the stabilizer $Z_1Z_2Z_3$. The algorithm then checks to see if there are any more physical qubits required in that stabilizer, if not it skips the following and starts again with the next stabilizer. In our example case, we still need the

Z_3 measurement. So, the algorithm will fuse another resource state to the ancilla of the cluster state already created and denote the surviving qubit appropriately, Fig. 12c. Doing this will effectively append another qubit to the ancilla, Fig. 12d. Because we now have a complete stabilizer, the algorithm will now move to the next stabilizer and proceed similarly until all stabilizers have a corresponding cluster state, Fig 12e.

Algorithm 2, in Appendix B, will take the cluster states returned by Algorithm 1 and then fuse them to effectively create the clusterized code. The first step in doing this is to iterate over all qubits on all the cluster states to find qubits denoted by the same number. That is, the algorithm finds on what qubits the stabilizers overlap. In our example, the stabilizers only overlap on qubit 3. It then pairs the qubits denoted by the same numbers. The algorithm takes these pairs of qubits, and fuses them with the edge qubits of the resource state and denotes the center qubit of the resource state with the number associated with the qubits being fused, Fig. 12f. Once this algorithm is finished running, it returns the clusterized code, Fig. 12 g.

By repeating Algorithm 1 and Algorithm 2, a series of primal and dual clusterized CSS code layers can be produced. By inputting these clusterized codes into Algorithm 3, in Appendix C, the corresponding foliated code will be produced. The algorithm selects a primal clusterized code and a dual clusterized code. It then selects a non-ancilla qubit on one of the codes and iterates through the non-ancilla qubits of the other code until it finds one denoted with the same number. It then fuses these two qubits with the edges of a 4-qubit chain-like cluster state, as shown in Fig. 15. This is repeated until all the non-ancilla qubits have been fused with another qubit. The algorithm will then select a new primal clusterized state and repeat the aforementioned process on the dual clusterized state from before. Until there are no clusterized codes left, this is repeated by next selecting a dual code and then alternating: dual, primal, dual etc. If there is only a dual code clusterized state left and the previously fused cluster state was a dual, the remaining clusterized code is then fused to the first primal clusterized state used. This will yield a state with stabilizers equivalent to the foliated code up to a fusion measurement outcome dependent phase. Hence, the work in [9, 10] will also apply to the fusion-based foliated quantum, and so it too will have the properties of a foliated quantum code. Lastly, note that the 4-qubit chain-like cluster state is created by fusing two ends of two of the resource states.

To conclude, by showing an implementation of the foliated Steane code through fusions, we have given a proof of concept for our algorithm that uses fusions to construct generalized foliated CSS codes from a set of identical resource states. Although we note that this idea was mentioned in [6], it was not expounded upon to fruition because the authors focused on un-foliatable codes. While we acknowledge the potential of these un-foliatable codes, because they have not yet been studied under a realistic noise model and because they are more difficult to fault-tolerantly construct, we focused on constructing the better understood foliated quantum codes. That way, in a physical implementation, we can be more confident in our algorithm potentially offering the fault-tolerance with high thresholds afforded by foliated quantum codes. Moreover, we show that by carefully selecting resource states, the number of qubits, fusions, and single-qubit Z measurement can be minimized. Lastly, because our fusion-based construction of the Steane code requires few resources, it could serve as a near-term goal in experimental quantum error correction.

Further research on this topic could include comparing the error tolerance of foliated codes to crystal lattice codes [7, 8] using an error model that accounts for their preparation, using fusions to construct a code from seed states, optimizing said construction, and finding a general way of optimizing the resources necessary for implementing fusion-based foliated quantum codes.

---

* s4531762@uq.net.au

[1] J. Roffe, *Quantum Error Correction: An Introductory Guide.* Contemporary physics **60**, p.226-245 (2019).

[2] R. Jozsa, *An introduction to measurement based quantum computation.* arXiv: 0508124v2 (2005).

[3] H. J. Briegel, *et al.*, *Measurement-based quantum computation.* Nature Physics **5**, p.19–26 (2009).

[4] M. A. Nielsen, *Cluster-state quantum computation.* Reports on Mathematical Physics **57**(1), p.147-161 (2006).

[5] S. Aaronson, D. Gottesman, *Improved Simulation of Stabilizer Circuits.* Phys. Rev. A **70**(5), 052328 (2004).

[6] S. Bartolucci, *et al.*, *Fusion-based quantum computation.* arXiv: 2101.09310v1 (2021).

[7] N. Nickerson, H. Bombín, *Measurement based fault tolerance beyond foliation.* arXiv: 1810.09621v1 (2018).

[8] M. Newman, L. A. Castro, K. R. Brown, *Generating Fault-Tolerant Cluster States from Crystal Structures.* Quantum 4, **295** (2020).

[9] A. Bolt, G. Duclos-Cianci, D. Poulin, T. M. Stace, *Foliated Quantum Error-Correcting Codes.* Phys. Rev. Lett. **117**(7), 070501 (2016).

[10] A. Bolt, D. Poulin, T. M. Stace, *Decoding schemes for foliated sparse quantum error-correcting codes.* Phys. Rev. A **98**(6), 062302 (2018).

[11] R. Raussendorf, S. Bravyi, J. Harrington, *Long-range quantum entanglement in noisy cluster states.* Phys. Rev. A **71**, 062313 (2005).

Appendix A

```
Algorithm 1: Clusterizing stabilizers
Data: A CSS code's $Z$ stabilizer generators.
Result: Clusterized stabilizers.
while not all the $Z$ stabilizer generators have been considered do
    -Select a stabilizer from the data;
    -Take a 3-qubit chain-like cluster state, as seen in Fig. 12a, and set the middle qubit to be an ancilla and assign each of the remaining qubits a number corresponding to one of the qubits measured in the selected stabilizer, as seen in Fig. 12b;
    while not all physical qubits in the selected stabilizer have been considered do
        -Take another 3-qubit chain-like cluster state as seen in Fig. 12a and set the center qubit to be an ancilla and one of the remaining qubits to be another qubit measured in the stabilizer;
        -Fuse the unassigned qubit to the ancilla qubit of the original cluster state;
    end
end
```

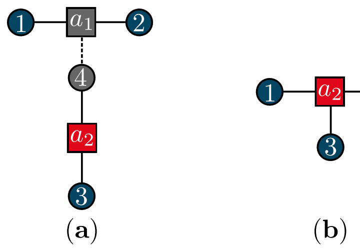

FIG. 13. Visualisation for the proof of Algorithm 1. The squares are ancilla qubits, the circles are data qubits, and the black lines are controlled-phase gates. Qubits greyed out are being measured, and the red and blue qubits are unmeasured in that step. The dotted black lines are fusions. Qubits are labelled with a number that corresponds to the subscripts of the Pauli matrices in the proof below. The fusion network in (a) returns the cluster state in (b).

Algorithm 1 converts a set of Z stabilizer generators into a set of *clusterized stabilizers*; a set of cluster states that when their ancilla qubits are measured in the X basis, their qubits are projected onto an eigenstate of the corresponding stabilizer.

Note that in the following, when we say *phase-entangled*, we mean two qubits entangled by a controlled-phase gate.

Proof. Throughout the proof, refer to Fig. 13. Consider a 3-qubit chain-like cluster state. Its stabilizers are

$$X_1 Z_{a_1},\quad Z_1 X_{a_1} Z_2,\quad Z_{a_1} X_2.$$

Now consider a second 3-qubit chain-like cluster state. It will have stabilizers

$$X_3 Z_{a_2},\quad Z_3 X_{a_2} Z_4,\quad Z_{a_2} X_4.$$

Take the following fusion measurements,

$$M_1 = X_{a_1} Z_4,\quad M_2 = Z_{a_1} X_4,$$

with outcomes m_1 and m_2 respectively. After the measurements the state's stabilizers will be

$$\begin{aligned}(Z_1 X_{a_1} Z_2)(Z_3 X_{a_2} Z_4) &= m_1 X_{a_2} Z_1 Z_2 Z_3,\\ (X_1 Z_{a_1})(Z_{a_2} X_4) &= m_2 X_1 Z_{a_2},\\ (Z_{a_1} X_2)(Z_{a_2} X_4) &= m_2 X_2 Z_{a_2},\\ &\quad X_3 Z_{a_2}.\end{aligned}$$

Observe that these stabilizers, up to a measurement outcome dependent phase, are equivalent to an ancilla qubit entangled with three qubits through controlled-phase gates. Hence, these fusions have effectively added a new data qubit to the ancilla qubit of the original cluster state. As such, by repeating this method, one can achieve a cluster state equivalent to an ancilla qubit phase-entangled with an arbitrary number of data qubits. So, by repeating this for every physical qubit in each Z stabilizer generator, a complete set of clusterized stabilizers is produced. With reference to the stabilizers, see that measuring $X_{a_{i\in\mathbb{N}}}$ does indeed give the code's stabilizers.

□

Appendix B

Algorithm 2: Clusterizing a CSS code

```
Data: The code's clusterized stabilizers.
Result: A clusterized CSS code.
-Select a non-ancilla qubit;
while there exists an instance of more than one
  qubit with the same assigned number do
    -Iterate through the other non-ancilla qubits
      to see if any have been assigned the same
      value as the slected qubit. If there is a qubit
      with a matching number, return it;
    if two qubits have been assigned the same
      number then
        -Fuse them to different ends of a 3-qubit
          chain-like cluster state, where the middle
          qubit of the a 3-qubit chain-like cluster is
          assigned the same number as the qubits
          being fused, as seen in Fig. 12f;
    else
        -Select a new non-ancilla qubit;
    end
end
```

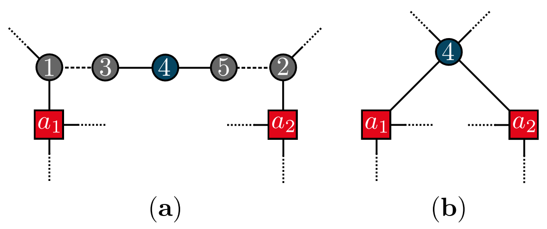

FIG. 14. Visualisation for the proof of Algorithm 2. The squares are ancilla qubits, the circles are data qubits, and the black lines are controlled-phase gates. Qubits greyed out are being measured, and the red and blue qubits are unmeasured in that step. The dotted black lines are fusions. The solid black lines that become dotted denote entanglement with an arbitrary number of other data qubits that make up the remainder of the clusterized stabilizer. Qubits are labelled with a number that corresponds to the subscripts of the Pauli matrices in the proof below. The fusion network in (a) returns the cluster state in (b).

Algorithm 2 converts a complete set of a code's clusterized stabilizers into the corresponding clusterized code.

Proof. Throughout the proof, refer to Fig. 14. Consider an arbitrary clusterized stabilizer. Two of its stabilizers will be

$$X_{a_1}Z_1P_1,$$
$$Z_{a_1}X_1P_3.$$

Now consider another arbitrary clusterized stabilizer. Its set of stabilizer will contain the elements

$$X_{a_2}Z_2P_2,$$
$$Z_{a_2}X_2P_4.$$

We only consider these four stabilizers because any other stabilizers either trivially follow from the symmetry of the problem or are unaffected by the fusions and are left unchanged. Note that P_1, P_2, P_3, and P_4 contain an arbitrary number of Z measurements that are independent of the fusion measurement. This is because $P_{i\in\{1,2,3,4\}}$ make up the remainder of the arbitrary stabilizer.

Now, also consider a 3-qubit chain-like cluster state, which has stabilizers

$$X_3Z_4,$$
$$Z_3X_4Z_5,$$
$$Z_4X_5.$$

Fusing the ends of this 3-qubit chain-like cluster state to the designated non-ancilla qubit of each of the two clusterized stabilizers gives the fusion measurements

$$M_1 = X_1Z_3,$$
$$M_2 = Z_1X_3,$$
$$M_3 = X_2Z_5,$$
$$M_4 = Z_2X_5,$$

with outcomes m_1, m_2, m_3, and m_4 respectively.

After taking measurements M_1 and M_2, the stabilizers of the system will be

$$\begin{aligned}(X_{a_1}Z_1P_1)(X_3Z_4) &= m_2X_{a_1}Z_4P_1,\\ &X_{a_2}Z_2P_2,\\ &X_2Z_{a_2}P_4,\\ (Z_{a_1}X_1P_3)(Z_3X_4Z_5) &= m_1X_4Z_{a_1}Z_5P_3,\\ &X_5Z_4.\end{aligned}$$

Then, after taking measurements M_3 and M_4, the stabilizers of the system will be

$$\begin{aligned}&m_2X_{a_1}Z_4P_1,\\ (X_{a_2}Z_2P_2)(X_5Z_4) &= m_4X_{a_2}Z_4P_2,\\ (X_2Z_{a_2}P_4)(m_1X_4Z_{a_1}Z_5P_3) &= m_1m_3X_4Z_{a_1}Z_{a_2}P_3P_4.\end{aligned}$$

Observe that these stabilizers, up to a measurement outcome dependent phase, are equivalent to having a single data qubit phase-entangled with two ancilla qubits. Note that this same procedure can be repeated to phase-entangle a single data qubit to an arbitrary number of ancilla qubits. Hence, by applying this method to the qubits that overlap in the code's Z stabilizer generators, and by iterating over all the data qubits on all the clusterized stabilizers, a cluster state with stabilizers equivalent to the clusterized code, up to a known phase, will be produced. Also note that the remaining stabilizers are just product of X and I. These remaining stabilizers necessarily commute with the Z-like stabilizers and the logical operators that we have already found, and commute with each other, hence they must be generators of the code's X-like stabilizers or the code's remaining logical operators [9].

□

Appendix C

Algorithm 3: Foliating a CSS code

```
Data: $N$ copies of a clusterized CSS code and
      $N \pm 1$ copies of the clusterized version of
      the selected CSS code's dual, for
      $2N \pm 1 \geq 3$ where the $\pm$ is the same as
      that in $N \pm 1$.
Result: A foliated CSS code.
-Select a clusterized primal code and designate it
  as the primary cluster;
while not all the data qubits on a boarder layer of
  the foliated code are entangled with a data qubit
  on another layer, or not all the data qubits in a
  bulk layer of the foliated code are entangled with
  qubits on two other layers do
    -Select a dual of the primary cluster's code
      and designate it as the secondary cluster;
    if no such clusters remain then
        -Set the first primary cluster to again be
          the primary cluster;
        -Set the one remaining clusterized dual
          code to be the secondary cluster;
    end
    -Select a non-ancilla qubit on the selected
      primary cluster;
    while not all the non-ancilla qubits on the
      selected primary cluster code are entangled
      with a qubit on the secondary cluster do
        -Iterate through the non-ancilla qubits on
          the secondary cluster and check if any
          have been assigned the same number as
          the selected qubit on the primary cluster;
        if two qubits have been assigned the same
          number then
            -Fuse two of the 3-qubit chain-like
              cluster states seen in Fig. 12a to
              produce a 4-qubit chain-like cluster
              state;
            -Fuse the two qubits assigned the same
              number to different ends of the 4-qubit
              chain-like cluster state.;
            -Label the two middle qubits of the
              4-qubit chain-like cluster state with
              the same number as the qubits being
              fused;
        end
    end
    -Set the primary cluster to be the secondary
      cluster
end
```

Algorithm 3 takes a group of primal and dual clusterized codes and returns the corresponding foliated code.

Proof. This proof is broken into two steps. The first step relates to Fig. 15a and Fig. 15b, while the second step relates to Fig. 15c and Fig. 15d.

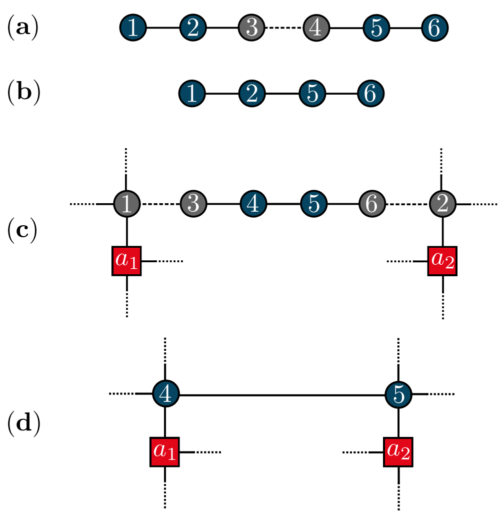

FIG. 15. Visualisation for the proof of Algorithm 3. The squares are ancilla qubits, the circles are data qubits, and the black lines are controlled-phase gates. Qubits greyed out are measured in that step, and the red and blue qubits are unmeasured in that step. The dotted black lines are fusions. The solid black lines that become dotted denote entanglement with an arbitrary number of other qubits to implement arbitrary clusterized codes. Qubits are labelled with a number that corresponds to the subscripts of the Pauli matrices in the proof below. The fusion network in (a) returns the cluster state in (b) and the fusion network in (c) returns the cluster state in (d).

First step: Produce a 4-qubit chain-like cluster state from two 3-qubit chain-like cluster states. Consider a 3-qubit chain-like cluster state, which has stabilizers

$$\begin{aligned} X_1Z_2&,\\ Z_1X_2Z_3&,\\ Z_2X_3&. \end{aligned}$$

Now consider another 3-qubit chain-like cluster state. Its stabilizers are

$$\begin{aligned} X_4Z_5&,\\ Z_4X_5Z_6&,\\ Z_5X_6&. \end{aligned}$$

Fuse one end from each of the above resource states. For example, consider the fusions

$$\begin{aligned} M_1 &= X_3Z_4,\\ M_2 &= Z_3X_4, \end{aligned}$$

with outcomes m_1 and m_2 respectively. After these measurements, the stabilizers will be

$$\begin{aligned} &X_1Z_2,\\ (Z_1X_2Z_3)(X_4Z_5) &= m_2X_2Z_1Z_5,\\ (Z_2X_3)(Z_4X_5Z_6) &= m_1X_5Z_2Z_6,\\ &X_6Z_5. \end{aligned}$$

Observe that these stabilizers are consistent with a 4-qubit chain-like cluster state, up to a phase. As required.

Second step: Fuse the clusterized codes into a foliated code. Consider an arbitrary clusterized code. Two of its stabilizers will be

$$\begin{aligned} &X_{a_1} Z_1 P_1, \\ &Z_{a_1} X_1 P_3. \end{aligned}$$

Now consider another arbitrary clusterized code. Its set of stabilizer will contain the elements

$$\begin{aligned} &X_{a_2} Z_2 P_2, \\ &Z_{a_2} X_2 P_4. \end{aligned}$$

Only these four stabilizers need to be considered, as discussed in Appendix B. Also, P_1, P_2, P_3, and P_4 each make up the remainder of the arbitrary Z measurement in the stabilizers. They are condensed into this one operator because they are independent of the fusion measurements.

Now, also consider a 4-qubit chain-like cluster state, which has stabilizers

$$\begin{aligned} &X_3 Z_4, \\ &Z_3 X_4 Z_5, \\ &\quad Z_4 X_5 Z_6, \\ &\qquad Z_5 X_6. \end{aligned}$$

Fusing the ends of this 4-qubit chain-like cluster state to each of the designated non-ancilla qubits of each of the two clusterized codes gives the fusion measurements

$$\begin{aligned} M_1 &= X_1 Z_3, \\ M_2 &= Z_1 X_3, \\ M_3 &= X_2 Z_6, \\ M_4 &= Z_2 X_6, \end{aligned}$$

with measurement outcomes m_1, m_2 m_3, and m_4 respectively. The stabilizers, after measuring M_1 and M_2, are

$$\begin{aligned} (X_{a_1} Z_1 P_1)(X_3 Z_4) &= m_2 X_{a_1} Z_4 P_1, \\ &\quad X_{a_2} Z_2 P_2, \\ &\quad X_2 Z_{a_2}, \\ (Z_{a_1} X_1 P_3)(X_4 Z_3 Z_5) &= m_1 X_4 Z_{a_1} Z_5 P_3, \\ &\quad X_5 Z_4 Z_6, \\ &\quad X_6 Z_5. \end{aligned}$$

Then, after measuring M_3 and M_4, these stabilizers are updated to

$$\begin{aligned} &m_2 X_{a_1} Z_4 P_1, \\ (X_{a_2} Z_2 P_2)(X_6 Z_5) &= m_4 X_{a_2} Z_5 P_2, \\ &m_1 X_4 Z_{a_1} Z_5 P_3, \\ (X_2 Z_{a_2} P_4)(X_5 Z_4 Z_6) &= m_3 X_5 Z_{a_2} Z_4 P_4. \end{aligned}$$

Observe that these stabilizers are equivalent to introducing a controlled-phase gate between the two designated qubits on the two sheets of the clusterized code, up to a measurement outcome dependant phase. Hence, by repeating this procedure for all the data qubits and selecting to fuse them with their corresponding qubit on the dual code, two layers of the foliated code are produced. Thus, by iterating over all the intended layers of the code, a foliated quantum code is produced, up to a known phase.

□