Optimal local unitary encoding circuits for the surface code

Oscar Higgott,[1,*] Matthew Wilson,[1,2] James Hefford,[1,2] James Dborin,[1,3] Farhan Hanif,[1] Simon Burton,[1] and Dan E. Browne[1]

[1] *Department of Physics and Astronomy, University College London, Gower Street, London WC1E 6BT, United Kingdom*
[2] *Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom*
[3] *London Centre for Nanotechnology, University College London, Gordon St., London WC1H 0AH, United Kingdom*

(Dated: February 4, 2020)

The surface code is a leading candidate quantum error correcting code, owing to its high threshold, and compatibility with existing experimental architectures. It is known that encoding a state in the surface code using local unitary operations requires time at least linear in the lattice size L, however the most efficient known method, introduced by Dennis *et al.* [1], has $O(L^2)$ time complexity. Here, we present an optimal local unitary encoding circuit for the planar surface code that uses exactly $2L$ time steps to encode a distance L planar code. We further show how an $O(L)$ complexity local unitary encoder for the toric code can be found by enforcing locality in the $O(\log L)$-depth non-local renormalisation encoder. We relate these techniques by providing an $O(L)$ local unitary circuit to convert between a toric code and a planar code, and also provide optimal encoders for the rectangular, rotated and 3D surface codes. Furthermore, using known mappings from surface codes, our circuits also imply optimal encoders for any 2D translationally invariant topological code, some 2D subsystem codes, as well as the 2D color code with and without boundaries. Our results may enable earlier experimental demonstrations of topological quantum error correcting protocols, and provide a tight upper bound on the time complexity of generating topological quantum order.

I. INTRODUCTION

One of the most promising error correcting codes for achieving fault-tolerant quantum computing is the surface code, owing to its high threshold and low weight check operators that are local in two dimensions [1, 2]. The stabilisers of the surface code are defined on the faces and sites of a $L \times L$ square lattice embedded on either a torus (the *toric* code) or a plane (the *planar* code). The toric code encodes two logical qubits, while the planar code encodes a single logical qubit.

An important component of any quantum error correction (QEC) code is its encoding circuit, which maps an initial product state of k qubits in arbitrary unknown states (along with $n-k$ ancillas) to the same state on k logical qubits encoded in a quantum code with n physical qubits. The encoding of logical states has been realised experimentally for the demonstration of small-scale QEC protocols using various codes [3–15], however one of the challenges of realising larger-scale experimental demonstrations of QEC protocols is the increasing complexity of the encoding circuits with larger system sizes, which has motivated the recent development of compiling techniques that reduce the number of noisy gates in unitary encoding circuits [16].

Encoding circuits can also be useful for implementing fermion-to-qubit mappings [17], an important component of quantum simulation algorithms, since some mappings introduce stabilisers in order to mitigate errors [18] or enforce locality in the transformed fermionic operators [19–22]. Local unitary encoding circuits provide a method to initialise and switch between mappings without the need for ancilla-based stabiliser measurements and feedback.

Despite significant progress in the development of quantum compilers, the best known local unitary encoding circuits for the surface code are far from optimal. Bravyi *et al.* [23] showed that any local unitary encoding circuit for the surface code must take time that is at least linear in the distance L, however the most efficient known local unitary encoding circuit for the surface code was introduced by Dennis *et al.* [1], who provide an encoding circuit that requires $\Omega(L^2)$ time to encode a distance L planar code. Aguado and Vidal [24] introduced a Renormalisation Group unitary encoding circuit for the toric code with $O(\log L)$ circuit depth, however their method requires non-local gates. Dropping the requirement of unitarity, encoders have been found that use stabiliser measurements [25, 26] or local dissipative evolution [27], and it has been shown that local dissipative evolution cannot be used to beat the $\Omega(L)$ lower bound for local unitary encoders [28]. If only the logical $|\bar{0}\rangle$ state is to be prepared, then stabiliser measurements [1] or adiabatic evolution [29] can be used, however encoding circuits by definition should be capable of encoding an arbitrary unknown input state.

In this work, we present local unitary encoding circuits for both the planar and toric code that take time linear in the lattice size, achieving the $\Omega(L)$ lower bound given by Bravyi *et al.* [23]. Furthermore, we provide encoding circuits for rectangular, rotated and 3D surface codes, as well as a circuit that encodes a toric code from a planar code.

* oscar.higgott.18@ucl.ac.uk

II. STABILISER CODES

An n-qubit Pauli operator $P = \alpha P_n$ where $P_n \in \{I, X, Y, Z\}^{\otimes n}$ is an n-fold tensor product of single qubit Pauli operators with the coefficient $\alpha \in \{\pm 1, \pm i\}$. The set of all n-qubit Pauli operators forms the n-qubit Pauli group $\mathcal{P}_n$. The *weight* $\mathrm{wt}(P)$ of a Pauli operator $P \in \mathcal{P}_n$ is the number of qubits on which it acts non-trivially. Any two Pauli operators commute if an even number of their tensor factors commute, and anti-commute otherwise.

Stabiliser codes [30] are defined in terms of a stabiliser group $\mathcal{S}$, which is an abelian subgroup of $\mathcal{P}_n$ that does not contain the element $-I$. Elements of a stabiliser group are called *stabilisers*. Since every stabiliser group is abelian and Pauli operators have the eigenvalues ± 1, there is a joint $+1$-eigenspace of every stabiliser group, which defines the stabiliser code.

The *check operators* of a stabiliser code are a set of generators of $\mathcal{S}$ and hence all measure $+1$ if the state is uncorrupted. Any check operator M that anticommutes with an error E will measure -1 (since $ME|\psi\rangle = -EM|\psi\rangle = -E|\psi\rangle$). The centraliser $C(\mathcal{S})$ of $\mathcal{S}$ in $\mathcal{P}_n$ is the set of Pauli operators which commute with every stabiliser. If an error $E \in C(\mathcal{S})$ occurs, it will be undetectable. If $E \in \mathcal{S}$, then it acts trivially on the codespace, and no correction is required. However if $E \in C(\mathcal{S}) \setminus \mathcal{S}$, then an undetectable logical error has occurred. The distance d of a stabiliser code is the smallest weight of any logical operator.

A stabiliser code is a Calderbank-Shor-Steane (CSS) code if there exists a generating set for the stabiliser group such that every generator is in $\{I, X\}^n \cup \{I, Z\}^n$.

III. THE SURFACE CODE

The surface code is a CSS code introduced by Kitaev [1, 2], which has check operators defined on a square lattice embedded in a two-dimensional surface. Each *site* check operator is a Pauli operator in $\{I, X\}^n$ which only acts non-trivially on the edges adjacent to a vertex of the lattice. Each *plaquette* check operator is a Pauli operator in $\{I, Z\}^n$ which only acts non-trivially on the edges adjacent to a face of the lattice. In the toric code, the square lattice is embedded in a torus, whereas in the planar code the lattice is embedded in a plane, without periodic boundary conditions (see Fig. 1). These site and plaquette operators together generate the stabiliser group of the code. While the toric code encodes two logical qubits, the surface code encodes a single logical qubit.

IV. ENCODING AN UNKNOWN STATE

We are interested in finding a unitary encoding circuit that maps a product state $|\phi_0\rangle \otimes \dots |\phi_{k-1}\rangle \otimes |0\rangle^{\otimes(n-k)}$

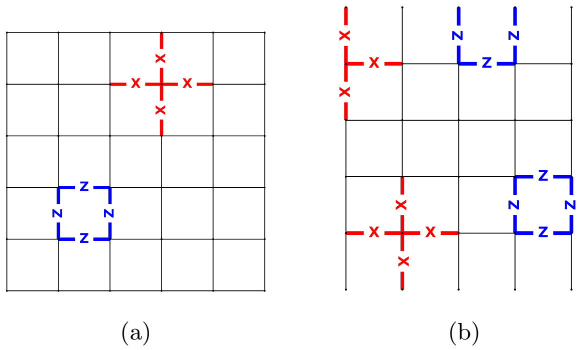


FIG. 1: The check operators for (a) the toric code and (b) the planar code. Opposite edges in (a) are identified and each edge corresponds to a qubit.

of k physical qubits in unknown states (along with ancillas) to the state of k logical qubits $|\bar{\phi}\rangle$ encoded in a stabiliser code with n physical qubits. Labelling the ancillas in the initial state $k, k+1, \dots, n-1$, we note that the initial product state is a $+1$-eigenstate of the stabilisers $Z_k, Z_{k+1}, \dots, Z_{n-1}$. Thus, we wish to find a unitary encoding circuit that maps the stabilisers $Z_k, Z_{k+1}, \dots, Z_{n-1}$ of the product state to a generating set for the stabiliser group $\mathcal{S}$ of the code. The circuit must also map the logical operators $Z_0, Z_1, \dots, Z_{k-1}$ and $X_0, X_1, \dots, X_{k-1}$ of the physical qubits to the corresponding logical operators $\bar{Z}_0, \bar{Z}_1, \dots, \bar{Z}_{k-1}$ and $\bar{X}_0, \bar{X}_1, \dots, \bar{X}_{k-1}$ of the encoded qubits (up to stabilisers).

Applying a unitary U to an eigenstate $|\psi\rangle$ of an operator S (with eigenvalue s) gives $US|\psi\rangle = sU|\psi\rangle = USU^\dagger U|\psi\rangle$: an eigenstate of S becomes an eigenstate of $USU^\dagger$. Therefore, we wish to find a unitary encoding circuit that, acting under conjugation, transforms the stabilisers and logicals of the initial product state into the stabilisers and logicals of the encoded state.

The CNOT gate, acting by conjugation, transforms Pauli X and Z operators as follows:

$$XI \leftrightarrow XX, \quad IZ \leftrightarrow ZZ, \tag{1}$$

and leaves ZI and IX invariant. Here $\sigma\sigma'$ for $\sigma, \sigma' \in \{I, Z, X\}$ denotes $\sigma_C \otimes \sigma_T$ with C and T the control and target qubit of the CNOT respectively. Since $Z = HXH$ and $X = HZH$, a Hadamard gate H transforms an eigenstate of Z into an eigenstate of X and vice versa. We will show how these relations can be used to generate unitary encoding circuits for the surface code using only CNOT and Hadamard gates.

As an example, consider the problem of generating the encoding circuit for the repetition code, which has stabilisers Z_0Z_1 and Z_1Z_2. We start in the product state $|\phi\rangle|0\rangle|0\rangle$ which has stabilisers Z_1 and Z_2. We first apply CNOT_{01} which transforms the stabiliser $Z_1 \to Z_0Z_1$ and leaves Z_2 invariant. Then applying CNOT_{12} transforms $Z_2 \to Z_1Z_2$ and leaves Z_0Z_1 invariant. We can also verify that the logical X undergoes the required transformation $X_0 \to \bar{X}_0 \coloneqq X_0 X_1 X_2$.

V. GENERAL ENCODING METHODS FOR STABILISER CODES

There exists a general method for generating an encoding circuit for any stabiliser code [30, 31], which we review in Appendix A. The specific structure of the output of this method means it can immediately be rearranged to depth $O(n)$. Using general routing procedures presented in [32–34] the output circuit could be adapted to a surface architecture with overhead $O(\sqrt{n})$, giving a circuit with depth $O(n\sqrt{n})$. This matches the scaling $O(\min(2n^2, 4nD\Delta))$ in depth for stabiliser circuits achieved in [35], where D and Δ are the diameter and degree respectively of the underlying architecture graph. Any stabiliser circuit has an equivalent skeleton circuit [36], and so can be implemented on a surface architecture with depth $O(n) = O(L^2)$, matching the previously best known scaling [1] for encoding the planar code. $O(n)$ is an optimal bound on the depth of the set of all stabiliser circuits [36], so we look beyond general methods and work with the specifics of the planar encoding circuit to improve on [1].

VI. OPTIMAL ENCODER FOR THE PLANAR CODE

Dennis *et al.* [1] showed how the methods outlined in section IV can be used to generate an encoding circuit for the planar surface code. The inductive step in their method requires $\Omega(L)$ time steps and encodes a distance $L+1$ planar code from a distance L code by turning smooth edges into rough edges and vice versa. As a result encoding a distance L planar code from an unencoded qubit requires $\Omega(L^2)$ time steps, which is quadratically slower than the lower bound given by Bravyi *et al.* [23].

However, here we present a local unitary encoding circuit for the planar code that requires only $2L$ time steps to encode a distance L planar code. The inductive step in our method, shown in Fig. 2 for $L = 4$, encodes a distance $L+2$ planar code from a distance L planar code using 4 time steps, and does not rotate the code. This inductive step can then be used recursively to encode an unencoded qubit into a distance L planar code using $2L$ time steps. If L is odd, the base case used is the distance 3 planar code, which can be encoded in 6 time steps. If L is even, a distance 4 planar code is used as a base case, which can be encoded in 8 time steps. Encoding circuits for the distance 3 and 4 planar codes are given in Appendix B. Our encoding circuit therefore matches the $\Omega(L)$ lower bound provided by Bravyi *et al.* [23].

We can also encode rectangular planar codes with height H and width W by first encoding a distance $\min(H, W)$ square planar code and then using a subset

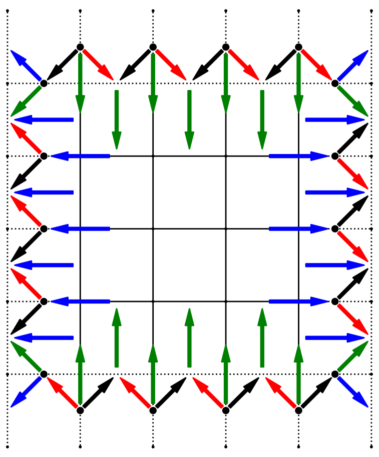


FIG. 2: Circuit to encode a distance 6 planar code from a distance 4 planar code. Each edge corresponds to a qubit. Each arrow denotes a CNOT gate, pointing from control to target. Filled black circles (centred on edges) denote Hadamard gates, which are applied at the beginning of the circuit. The colour of each CNOT gate (arrow) denotes the time step in which it is applied. The first, second, third and fourth time steps correspond to the blue, green, red and black CNOT gates respectively. Solid edges correspond to qubits originally encoded in the L=4 planar code, whereas dotted edges correspond to additional qubits that are encoded in the L=6 planar code.

of the gates in Fig. 2 (given explicitly in Appendix B) to either increase the width or the height as required. Increasing either the width or height by two requires three time steps, therefore encoding a $H \times W$ rectangular planar code from an unencoded qubit requires $2\min(H, W) + 3\left\lceil \frac{|H-W|}{2} \right\rceil$ time steps.

In Appendix B 2 we also provide an optimal encoder for the *rotated* surface code, which uses fewer physical qubits for a given distance L [26]. Our encoding circuit also uses an inductive step that increases the distance by two using four time steps, and therefore uses $2L + O(1)$ time steps to encode a distance L rotated surface code.

VII. LOCAL RENORMALISATION ENCODER FOR THE TORIC CODE

In this section we will describe an $O(L)$ encoder for the toric code based on the multi-scale entanglement renormalisation ansatz (MERA). The core of this method is to enforce locality in the Renormalisation Group (RG) encoder given by Aguado and Vidal [24]. The RG encoder starts from an $L = 2$ toric code and then uses an $O(1)$ depth inductive step which enlarges a distance 2^k code to a distance 2^{k+1} code, as shown in Fig. 3 for the first step ($k = 1$) (and reviewed in more detail in Appendix C).

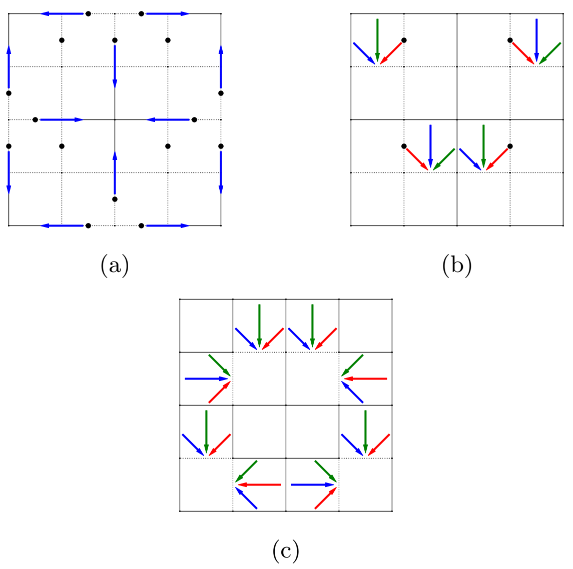


FIG. 3: Encoding a distance 4 toric code from a distance 2 toric code using the Renormalisation Group encoder of Aguado and Vidal [24]. Dashed edges, dashed edges with a node and solid edges correspond to decoupled ancillae in $|0\rangle$, in $|+\rangle$, and to qubits entangled with the original code respectively. Opposite edges are identified. Arrows denote CNOT operations from control to target qubits, and monochromatic gates in stages (b) and (c) may be executed in a single timestep.

The $L = 2$ base case toric code can be encoded using the method given by Gottesman in Ref. [30], as shown in Appendix C 1. While the RG encoder takes $O(\log L)$ time, it is non-local in it's original form.

In order to enforce locality in the RG encoder, we wish to find an equivalent circuit that implements an identical operation on the same input state, using quantum gates that act locally on the physical architecture corresponding to the final distance L toric code (here a gate is *local* if it acts only on qubits that belong to either the same site or plaquette). One approach to enforce locality in a quantum circuit is to insert SWAP gates into the circuit to move qubits adjacent to each other where necessary. Any time step of a quantum circuit can be made local on a $L \times L$ 2D nearest-neighbour (2DNN) grid architecture using at most $O(L)$ time steps, leading to at most a multiplicative $O(L)$ overhead from enforcing locality [32–34]. Placing an ancilla in the centre of each site and plaquette, we see that the connectivity graph of our physical architecture has a 2DNN grid as a subgraph. Therefore, using SWAP gates to enforce locality in the RG encoder immediately gives us a $O(L \log L)$ local unitary encoding circuit for the toric code which, while an improvement on the $O(L^2)$ encoder in Ref. [1], does not match the $\Omega(L)$ lower bound.

However, we can achieve $O(L)$ complexity by first noticing that all 'quantum circuit' qubits which are acted on non-trivially in the first k steps of the RG encoder can

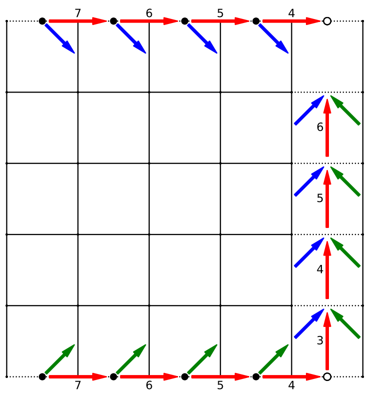
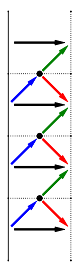


FIG. 4: Circuit to encode a distance 5 toric code from a distance 5 planar code. Solid edges correspond to qubits in the original planar code and dotted edges correspond to qubits added for the toric code. Opposite edges are identified. Arrows denote CNOT gates, and filled black circles denote Hadamard gates applied at the beginning of the circuit. Blue and green CNOT gates correspond to those applied in the first and second time step respectively. Red CNOTs are applied in the time step that they are numbered with. The hollow circles denote the unencoded qubit that is to be encoded into the toric code.

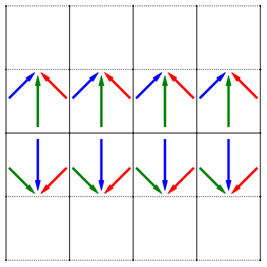
be mapped to physical qubits in a $2^{k+1} \times 2^{k+1}$ square region of the physical architecture. Therefore, the required operations in iteration k can all be applied within a $2^{k+1} \times 2^{k+1}$ region that also encloses the regions used in the previous steps. In Appendix C 2 we use this property to provide circuits for routing quantum information using SWAP gates (and no ancillas) that enforce locality in each of the $O(1)$ time steps in iteration k using $O(2^{k+1})$ time steps. This leads to a total complexity of $\sum_{k=1}^{\log_2(L)-1} O(2^{k+1}) = O(L)$ for encoding a distance L code, also achieving the lower bound given by Bravyi *et al.* [23]. In Appendix C 2 we provide a more detailed analysis to show that the total time complexity is $15L/2 - 6\log_2 L + 7 \sim O(L)$. Unlike the other encoders in this paper (which work for all L), the RG encoder clearly can only be applied when L is a power of 2.

VIII. ENCODING A TORIC CODE FROM A PLANAR CODE

While the method in section VI is only suitable for encoding planar codes, we will now show how we can encode a distance L toric code from a distance L planar code using only local unitary operations. Starting with a distance L planar code, $2(L-1)$ ancillas each in a $|0\rangle$ state, and an additional unencoded logical qubit, the circuit in Fig. 4 encodes a distance L toric code using $L + 2$ time steps. Therefore, encoding two unencoded qubits in a toric code can be achieved using $3L + 2$ time



(a)



(b)

FIG. 5: (a) Circuit to encode a 4×2 planar code from a four qubit repetition code (where adjacent qubits in the repetition code are stabilised by XX). Applied to a column of qubits corresponding to a surface code $\bar{Z}$, this encodes a layer in the yz-plane of a 3D surface code. (b) Circuit to encode the xz-plane of a 3D surface code once the yz-plane layers and a layer in the xy-plane have been encoded. Arrows denote CNOT gates pointing from control to target, and blue, green, red and black CNOT gates correspond to the first, second, third and fourth time steps respectively. Solid and dotted edges correspond to qubits that are initially entangled and in a product state respectively.

steps using the circuits given in this section and in section VI. Similarly, we can encode a planar code using the local RG encoder for the toric code, before applying the inverse of the circuit in Fig. 4.

IX. ENCODING A 3D SURFACE CODE

We will now show how the techniques developed to encode a 2D planar code can be used to encode a distance L 3D surface code using $O(L)$ time steps. We first encode a distance L planar code using the method given in section VI. This planar code now forms a single layer in the xy-plane of a 3D surface code (where the y-axis is defined to be aligned with a Z-logical in the original planar code). Using the circuit given in Fig. 5(a), we encode each column of qubits corresponding to a Z logical in the planar code into a layer of the 3D surface code in the yz-plane (which has the same stabiliser structure as a planar code if the rest of the x-axis is excluded). Since each layer in the yz-plane can be encoded in parallel, this stage can also be done in $O(L)$ time steps. If we encode each layer in the yz-plane such that the original planar code intersects the middle of each layer in the yz-plane, then each layer in the xz-plane now has the stabiliser structure shown in Fig. 5(b). Using the circuit in Fig. 5(b) repeatedly, all layers in the xz-plane can be encoded in parallel in $O(L)$ time steps. Therefore, a single unknown qubit can be encoded into a distance L 3D surface code in $O(L)$ time steps.

X. DISCUSSION

We have presented local unitary encoding circuits for the surface code that take time linear in the lattice size. Our results demonstrate that the $\Omega(L)$ lower bound given by Bravyi *et al.* [23] for generating topological quantum order is tight, and reduces the resource requirements for experimentally implementing some QEC protocols using the surface code. We have provided a new technique to encode the planar code in $O(L)$ time, as well as showing how an $O(L)$ local unitary encoding circuit for the toric code can be found by enforcing locality in the non-local RG encoder. We unify these two approaches by demonstrating how local $O(L)$-depth circuits can be used to convert between the planar and toric code, and generalise our method to rectangular, rotated and 3D surface codes.

Furthermore, using known local unitary mappings from one or more copies of the surface code, our results also imply the existence of optimal encoders for any 2D translationally invariant topological code, some 2D subsystem codes [37, 38], as well as the 2D color code with and without boundaries [39]. As an explicit example, the subsystem surface code with three-qubit check operators can be encoded from the toric code using the four time step quantum circuit given in Ref. [40].

Our techniques may enable earlier experimental demonstrations of some surface code error correction protocols, and may also be useful for encoding stabiliser-based fermion-to-qubit mappings such as the Verstraete-Cirac transform [20], which has a stabiliser structure similar to that of the rotated planar code [21]. Further work could also investigate optimal local unitary encoding circuits for codes on higher genus surfaces, such as punctured [41, 42] or hyperbolic surface codes [43].

ACKNOWLEDGMENTS

The authors would like to thank Mike Vasmer for informing us of the method for encoding stabiliser codes in Ref. [30], and Charlie Derby for helpful discussions on fermion-to-qubit mappings.

[1] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002).
[2] A. Y. Kitaev, Annals of Physics **303**, 2 (2003).
[3] J. Chiaverini, D. Leibfried, T. Schaetz, M. D. Barrett, R. Blakestad, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, *et al.*, Nature **432**, 602 (2004).

[4] C.-Y. Lu, W.-B. Gao, J. Zhang, X.-Q. Zhou, T. Yang, and J.-W. Pan, Proceedings of the National Academy of Sciences **105**, 11050 (2008).
[5] P. Schindler, J. T. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt, Science **332**, 1059 (2011).
[6] T. H. Taminiau, J. Cramer, T. van der Sar, V. V. Dobrovitski, and R. Hanson, Nature nanotechnology **9**, 171 (2014).
[7] G. Waldherr, Y. Wang, S. Zaiser, M. Jamali, T. Schulte-Herbrüggen, H. Abe, T. Ohshima, J. Isoya, J. Du, P. Neumann, *et al.*, Nature **506**, 204 (2014).
[8] D. Nigg, M. Mueller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, Science **345**, 302 (2014).
[9] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, *et al.*, Nature **519**, 66 (2015).
[10] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. Girvin, L. Jiang, *et al.*, Nature **536**, 441 (2016).
[11] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, Nature communications **7**, 11526 (2016).
[12] N. M. Linke, M. Gutierrez, K. A. Landsman, C. Figgatt, S. Debnath, K. R. Brown, and C. Monroe, Science advances **3**, e1701074 (2017).
[13] C. Vuillot, arXiv preprint arXiv:1705.08957 (2017).
[14] J. Roffe, D. Headley, N. Chancellor, D. Horsman, and V. Kendon, Quantum Science and Technology **3**, 035010 (2018).
[15] M. Gong, X. Yuan, S. Wang, Y. Wu, Y. Zhao, C. Zha, S. Li, Z. Zhang, Q. Zhao, Y. Liu, *et al.*, arXiv preprint arXiv:1907.04507 (2019).
[16] X. Xu, S. C. Benjamin, and X. Yuan, arXiv preprint arXiv:1911.05759 (2019).
[17] J. T. Seeley, M. J. Richard, and P. J. Love, The Journal of chemical physics **137**, 224109 (2012).
[18] Z. Jiang, J. McClean, R. Babbush, and H. Neven, Physical Review Applied **12**, 064041 (2019).
[19] S. B. Bravyi and A. Y. Kitaev, Annals of Physics **298**, 210 (2002).
[20] F. Verstraete and J. I. Cirac, Journal of Statistical Mechanics: Theory and Experiment **2005**, P09012 (2005).
[21] M. Steudtner and S. Wehner, Physical Review A **99**, 022308 (2019).
[22] V. Havlíček, M. Troyer, and J. D. Whitfield, Physical Review A **95**, 032332 (2017).
[23] S. Bravyi, M. B. Hastings, and F. Verstraete, Physical review letters **97**, 050401 (2006).
[24] M. Aguado and G. Vidal, Phys. Rev. Lett. **100**, 070404 (2008).
[25] J. Łodyga, P. Mazurek, A. Grudka, and M. Horodecki, Scientific reports **5**, 8975 (2015).
[26] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, New Journal of Physics **14**, 123011 (2012).
[27] J. Dengis, R. König, and F. Pastawski, New Journal of Physics **16**, 013023 (2014).
[28] R. König and F. Pastawski, Physical Review B **90**, 045101 (2014).
[29] A. Hamma and D. A. Lidar, Physical review letters **100**, 030502 (2008).
[30] D. Gottesman, arXiv preprint quant-ph/9705052 (1997).
[31] R. Cleve and D. Gottesman, Physical Review A **56**, 7682 (1997).
[32] D. Cheung, D. Maslov, and S. Severini, in *Workshop on Quantum Information Processing* (2007).
[33] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **469**, 20120686 (2013).
[34] S. Brierley, arXiv preprint arXiv:1507.04263 (2015).
[35] B. Wu, X. He, S. Yang, L. Shou, G. Tian, J. Zhang, and X. Sun, "Optimization of cnot circuits on topological superconducting processors," (2019), arXiv:1910.14478 [quant-ph].
[36] D. Maslov, Physical Review A **76** (2007), 10.1103/physreva.76.052310.
[37] B. Yoshida, Annals of Physics **326**, 15 (2011).
[38] H. Bombin, G. Duclos-Cianci, and D. Poulin, New Journal of Physics **14**, 073048 (2012).
[39] A. Kubica, B. Yoshida, and F. Pastawski, New Journal of Physics **17**, 083026 (2015).
[40] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, arXiv preprint arXiv:1207.1443 (2012).
[41] R. Raussendorf and J. Harrington, Physical review letters **98**, 190504 (2007).
[42] A. G. Fowler, A. M. Stephens, and P. Groszkowski, Physical Review A **80**, 052312 (2009).
[43] N. P. Breuckmann and B. M. Terhal, IEEE transactions on Information Theory **62**, 3731 (2016).

Appendix A: Procedure for Encoding a Stabiliser Code

1. Review of the General Method

We make use of a method [30, 31] for generating an encoding circuit for an arbitrary stabiliser code to find a base case $L = 2$ circuit of the toric code. We present the method here for completeness, giving the procedure in full and in the simplified case for which the code is CSS.

From a set of check operators one can produce a corresponding bimatrix

$$\left(L\middle|R\right)$$

Rows and columns represent check operators and qubits respectively. $L_{ij} = 1$ indicates that check operator i applies X to qubit j as opposed to the identity, similarly for the right hand side $R_{ij} = 1$ implies check operator i applies Z to qubit j. If both $L_{ij} = 1$ and $R_{ij} = 1$, then check operator i applies Y on qubit j.

A CSS code has check operators $P_n \in \{I, X\}^{\otimes n} \cup \{I, Z\}^{\otimes n}$, its corresponding bimatrix takes the form,

$$\left(\begin{array}{c|c} A & 0 \\ 0 & B \end{array}\right)$$

A and B have full row rank since they each represent an independent subset of the check operators. labelling the rank of A as r, the rank of B is $n - k - r$.

Via row addition, row swaps and column swaps, the left and right matrices of this simplified form can be taken to standard form [30] without changing the stabiliser group of the code. The standard form of the bimatrix is then

$$\left(\begin{array}{ccc|ccc} I & A_1 & A_2 & B & C_1 & C_2 \\ 0 & 0 & 0 & D & I & E \end{array}\right)$$

Where I,A_1,A_2 and D, I, E have (r), $(n-k-r)$, and (k) columns respectively. We may also represent the set of logical X operators as a bimatrix with each row representing the logical X for a particular encoded qubit,

$$\bar{X} = \left(\begin{array}{ccc|ccc} U_1 & U_2 & U_3 & V_1 & V_2 & V_3 \end{array}\right)$$

It is shown in [30] that the logical $\bar{X}$ operator can be taken to the form

$$\bar{X} = \left(\begin{array}{ccc|ccc} 0 & U_2 & I & V_1 & 0 & 0 \end{array}\right)$$

In the CSS case the check operator bimatrix reduces to

$$\left(\begin{array}{ccc|ccc} I & A_1 & A_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & D & I & E \end{array}\right)$$

and the logical X bimatrix to

$$\bar{X} = \left(\begin{array}{ccc|ccc} 0 & E^T & I & 0 & 0 & 0 \end{array}\right)$$

To produce a circuit which can encode state $|c_1 \dots c_k\rangle$ for any values of the c_i one should find a circuit which applies logical operators $\bar{X}_1^{c_1} \dots \bar{X}_k^{c_k}$ to the encoded $|0\rangle$ state $|\bar{0}\rangle \equiv \sum_{M \in S} M |0 \dots 0\rangle$. Letting F_c be row c of bimatrix F we denote the operator corresponding to F_c, with the operator on the m^{th} qubit replaced with identity, and then controlled by the m^{th} qubit as $F_{c(m)}$.

Since

$$\bar{X}_1^{c_1} \dots \bar{X}_k^{c_k} \sum_{M \in S} M |0 \dots 0\rangle = \sum_{M \in S} M \bar{X}_1^{c_1} \dots \bar{X}_k^{c_k} |0 \dots 0\rangle$$

the application of the X gates can be considered before applying the sum of stabiliser operations. Due to the I in the form of $\bar{X}$,

$$\bar{X}_{k(n)} |0_1 \dots 0_{n-k}\rangle |0_{n-k+1} \dots 0_{n-1} c_n\rangle = \bar{X}_k^{c_n} |0_1 \dots 0_n\rangle .$$

we see that independently of $|c_1 \dots c_k\rangle$ we can implement

$$\begin{aligned} \bar{X}_{1(n-k)} \dots \bar{X}_{k(n)} |0_1 \dots 0_{n-k}\rangle |c_{n-k+1} \dots c_n\rangle \\ = \bar{X}_1^{c_1} \dots \bar{X}_k^{c_k} |0_1 \dots 0_n\rangle \\ \equiv |0_1 \dots 0_r\rangle |Xc\rangle \end{aligned}$$

where in the last line it is emphasised that since $U_1 = 0$, $X_{i(j)}$ acts trivially on the first r qubits. Next to consider is $\sum_{M_j} M_j = (I + M_{n-k}) \dots (I + M_r) \dots (I + M_1)$.

We denote the right matrix of bimatrix M as RM. In standard form M_i always performs X on qubit i and it performs Z on qubit i when $RM_{ii} = 1$, giving

$$M_i |0 \dots 0_i \dots 0\rangle = RM_{ii} Z_i M_{i(i)} |0 \dots 1_i \dots 0\rangle$$

and so

$$\begin{aligned} (I + M_i) |0 \dots 0\rangle &= |0 \dots 0_i \dots 0\rangle + M_i |0 \dots 0_i \dots 0\rangle \\ &= RM_{ii} Z_i M_{i(i)} H_i |0 \dots 0\rangle \end{aligned}$$

or generally

$$\begin{aligned} &\prod_{i=1}^{r} (I + M_i)(|0_1 \dots 0_r\rangle |Xc\rangle) \\ = &\prod_{i=1}^{r} RM_{ii} Z_i M_{i(i)} H_i (|0_1 \dots 0_r\rangle |Xc\rangle) \end{aligned}$$

The remaining products

$$\prod_{i=r+1}^{n-k} (I + M_i) \tag{A1}$$

can be ignored since they consist only of σ_z operations and may be commuted to the front to act on $|0\rangle$ states.

Given initially some k qubits we wish to encode, and some additional $n-k$ auxiliary qubits, initialised in $|0\rangle$, a choice of generators for the stabiliser group is

$$\left(\begin{array}{cccccccc|cccccccc} 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \end{array}\right)$$

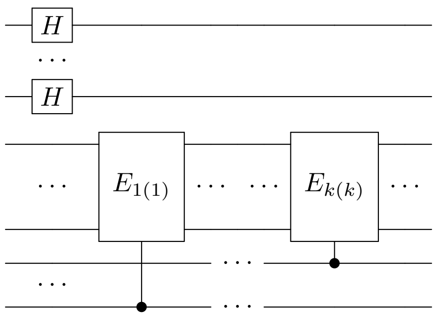
The general circuit which transforms the initial generator set to the standard form bimatrix is given by,

$$\prod_{i=1}^{r} M_{i(i)} H_i RM_{ii} Z_i \prod_{j=1}^{k} \bar{X}_{j(n-k+j)}$$

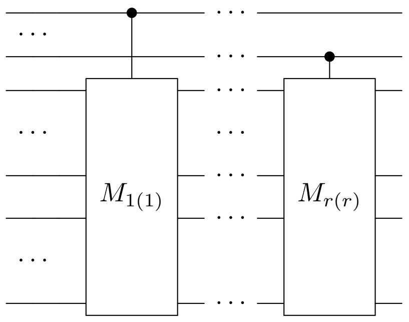
For CSS codes this reduces to

$$\prod_{i=1}^{r} M_{i(i)} H_i \prod_{j=1}^{k} \bar{X}_{j(n-k+j)}$$

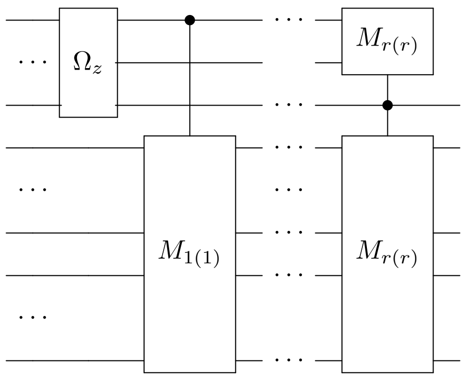
In the simplified case all gates are either initial H gates or $CNOT$'s. We may write the circuit in two stages, performing first the H gates and controlled $\bar{X}$ gates.



and in stage 2 the controlled X gates.



In the general case stage 1 is identical but stage 2 takes the form



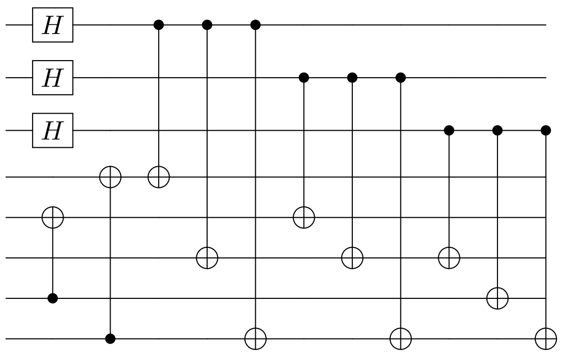
Where Ω_z consists of Z operations on some of the first r qubits and each $M_{i(i)}$ consists of controlled Z gates on some of the first r qubits and controlled Pauli gates on some of the following $n-r$ qubits. In the case of the $L=2$ toric code, with qubits labelled left to right and top to bottom the bimatrix is

$$\left(\begin{array}{cccccccc|cccccccc}
1&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
1&1&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1
\end{array}\right)$$

The standard form of this bimatrix is

$$\left(\begin{array}{cccccccc|cccccccc}
1&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&1&1&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&1&0&1&0&1&0\\
0&0&0&0&0&0&0&0&1&1&1&0&0&1&0&0
\end{array}\right)$$

The circuit which encodes the above stabiliser set is



It is important to have kept track of which column represents which qubit since column swaps are performed in bringing the matrix to standard form. Taking this into account gives the $L=2$ circuit on the toric architecture.

2. Depth of the General Method

Any stabiliser circuit has an equivalent skeleton circuit [36] which after routing on a surface architecture will have at worst $O(n)$ depth. The output of the general method for encoding a stabiliser code in fact already splits into layers of skeleton circuits. Stage 2 of the method applied to a CSS code has at worst $r(n-r)$ controlled Pauli gates CP_{ij} with i,j in $\{1\dots r\}$ and $\{r+1\dots n\}$ respectively, CP_{ij} is implemented before $CP_{i'j'}$ so long as $i<i'$. Stage 2 then takes the form of a skeleton circuit and as such the number of timesteps needed is $O(n)$ for surface or linear nearest neighbour architectures. Stage 1 has at most $k(n-k-r)$ gates and also takes the form of a skeleton circuit. In the worst case scenario stage 2 includes, in addition to the CP gates, controlled Z gates CZ with targets on the first r qubits. As noted in errata for [30], $i>j$ for any of the additional CZ_{ij} in stage 2. All CZ_{ij} can then be commuted to timesteps following all CP gates since each CP in a timestep following CZ_{ij} takes the form CP_{mn} with $n>m>i>j$. The circuit then splits into a layer of CP gates and a layer of CZ gates, each of which is a skeleton circuit, and so can be implemented in $O(n)$ timesteps on surface and linear nearest neighbour architectures.

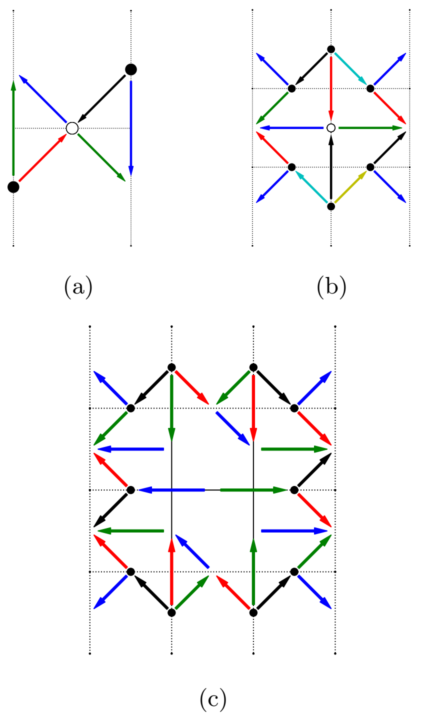


FIG. 6: Encoding circuits for the L=2, L=3 and L=4 planar codes. Each edge corresponds to a qubit, each arrow denotes a CNOT gate pointing from control to target, and each filled black circle denotes a Hadamard gate applied at the beginning of the circuit. The colour of each CNOT gate corresponds to the time step it is implemented in, with blue, green, red, black, cyan and yellow CNOT gates corresponding to the first, second, third, fourth, fifth and sixth time steps respectively. The hollow circle in each of (a) and (b) denotes the initial unencoded qubit. The circuit in (c) encodes an L=4 planar code from an L=2 planar code, with solid edges denoting qubits initially encoded in the L=2 code.

Appendix B: Additional planar encoding circuits

1. Planar base cases and rectangular code

In Fig. 6 we provide encoding circuits for the $L = 2$, $L = 3$ and $L = 4$ planar codes, requiring 4, 6 and 8 time steps respectively. These encoding circuits are used as base cases for the planar encoding circuits described in section VI. In Fig. 7 we provide encoding circuits that either increase the width or height of a planar code by two, using three time steps.

2. Rotated Surface Code

In Fig. 8 we demonstrate a circuit that encodes an $L = 7$ rotated surface code from a distance $L = 5$ rotated code. For a given distance L, the rotated surface code

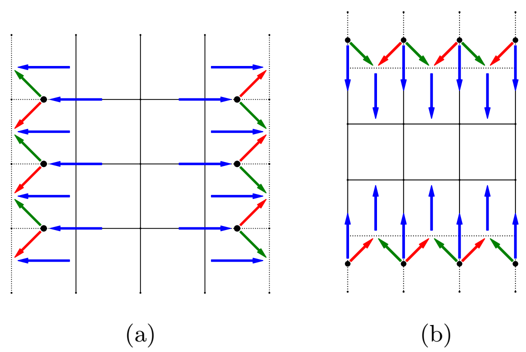


FIG. 7: (a) Circuit to increase the width of a planar code by two. (b) Circuit to increase the height of a planar code by two. Notation is the same as in Fig. 6.

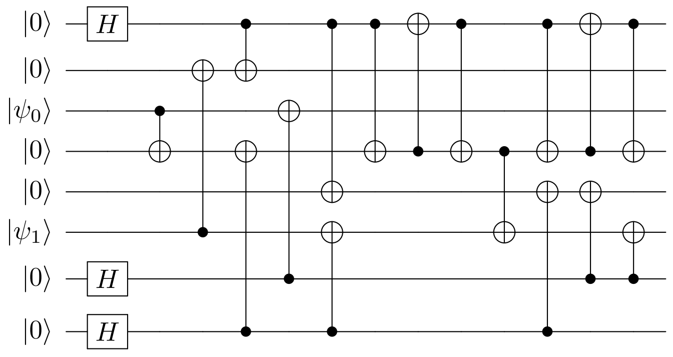
uses fewer physical qubits than the standard surface code to encode a logical qubit [26]. Considering a standard square lattice with qubits along the edges, a rotated code can be produced by removing qubits along the corners of the lattice boundary, leaving a diamond of qubits from the centre of the original lattice. The diagram in Fig. 8 shows the resultant code, rotated 45° compared to the original planar code, and with each qubit now denoted by a vertex rather than an edge. For a distance L code the rotated surface code requires L^2 qubits compared to $L^2 + (L-1)^2$ for the planar code.

The encoding circuit in Fig. 8 takes 4 steps to grow a rotated code from a distance $L = 5$ to $L = 7$. This is a fixed cost for any distance L to $L + 2$. To produce a distance $L = 2m$ code this circuit would be applied repeatedly $m + O(1)$ times to an $L = 2$ or $L = 3$ base case, requiring a circuit of total depth $2L + O(1)$.

Appendix C: Renormalisation Group encoder

1. Toric Code Encoder

Applying the Gottesman encoder to the toric code, as shown in Appendix A, and then enforcing locality using SWAP gates, gives the following encoding circuit for the $L = 2$ toric code that requires 10 time steps:



where the qubits are numbered $0 \ldots 7$ from top to bot-

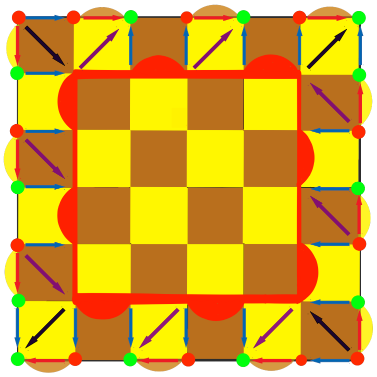


FIG. 8: Encoding circuit for the $L = 7$ rotated code from an $L = 5$ rotated surface code (shown as a red outline). The colour of each arrow denotes the time step the gate is applied in. The gates are applied in the order: blue, red, black, purple. The additional qubits are initialised in the $|+\rangle$ (red) or $|0\rangle$ (green state.) The yellow squares denote a Z stabiliser on the four corner qubits, and the brown squares represent an X operator on the four corner qubits. The rotated code has additional stabilizers between states on along the edges. In the $L = 5$ code these are shown as a red arch (with Z and X stabilisers on the vertical and horizontal edges respectively), and the yellow and brown arches in the $L = 7$ code edge are Z and X stabilizers between the two edge qubits.

tom. This circuit encodes the initial unknown qubit states $|\psi_0\rangle$ and $|\psi_1\rangle$ into logical states $|\bar{\psi_0}\rangle$ and $|\bar{\psi_1}\rangle$ of an $L = 2$ toric code with stabiliser group generators $X_0X_1X_2X_6$, $X_0X_1X_3X_7$, $X_2X_4X_5X_6$, $Z_0Z_2Z_3Z_4$, $Z_1Z_2Z_3Z_5$ and $Z_0Z_4Z_6Z_7$.

Equipped with an $L = 2$ base code emulated as the central core of a 4×4 planar grid, where the surrounding qubits are initially decoupled $+1$ Z eigenstates, one can apply the local routing methods of Appendix C 2 to obtain the initial configuration as depicted in Fig. 9. The ancillae qubits are then initialised as $|0\rangle$ or $|+\rangle$ eigenstates as depicted in Fig. 3(a) by means of Hadamard operations where necessary, before the circuit is implemented through the sequence of CNOT gates as depicted in Fig. 3(a)-(c).

By recursive application of Eq. (1), it is seen that the circuit forms the stabiliser structure of an $L = 4$ toric code on the planar architecture. Proceeding inductively, one can exploit the symmetry of a distance $L = 2^k$ toric code to embed it in the centre of a $2L \times 2L$ planar grid, "spread-out" the core qubits in time linear in the distance, and ultimately perform the $L = 2 \mapsto L = 4$ circuit on each 4×4 squarely-tesselated sub-grid.

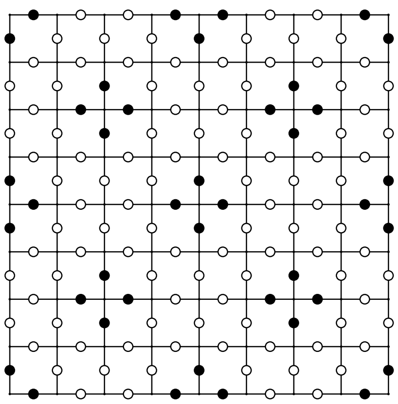


FIG. 9: Initial outwards spreading of qubits in a distance 4 toric code to prepare for the encoding of a distance 8 toric code. Solid black and unfilled nodes represent the routed qubits of the distance 4 code, and the ancillae respectively. One then executes the subroutine of FIG. 4 in each of the four 4x4 quadrants. This procedure generalises inductively for any targeted distance 2^k toric code.

2. Routing circuits for enforcing locality

To enforce locality in the Renormalisation Group encoder, which encodes a distance L toric code, one can use SWAP gates to "spread out" the qubits between iteration k and $k+1$, such that all of the $O(1)$ time steps in iteration $k+1$ are *almost local* on a $2^{k+1} \times 2^{k+1}$ region of the $L \times L$ torus. By *almost local*, we mean that the time step would be local if the $2^{k+1} \times 2^{k+1}$ region had periodic boundary conditions. Since at each iteration (until the final one) we use a region that is a subset of the torus, we in fact have a planar architecture (no periodic boundaries), and so it is not possible to simultaneously enforce locality in all of the $O(1)$ time steps in an iteration $k < \log L - 2$ of the RG encoder, which are collectively local on a toric architecture. Thus it is necessary to emulate a toric architecture on a planar one. In a time step in iteration k, this can be achieved by using $3(2^k - 1)$ time steps to move the top and bottom boundaries together (using SWAP gates) before applying any necessary gates which are now local (where the factor of three comes from the decomposition of a SWAP gate into 3 CNOT gates). Then $3(2^k-1)$ time steps are required to move the boundaries back to their original positions. The identical procedure can be applied simultaneously to the left and right boundaries. Thus there is an overhead of $3(2^{k+1} - 2)$ to emulate a toric architecture with a planar architecture. Starting from $L = 2$ and ending on a size L code gives an overall overhead to emulating the torus of $6(\sum_{i=1}^{\log_2(L)-2} 2^{i+1} - 2) = 6L - 12\log_2 L$, since from Fig. 3 it can be seen that opposite edges need be made adjacent two times per iteration to enforce locality in it. Additionally, the time steps within each iteration must

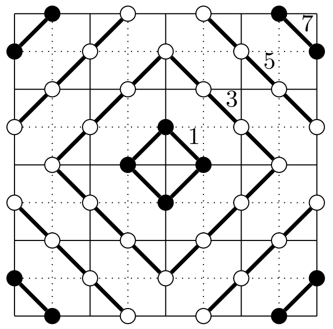


FIG. 10: $L = 4$ code showing the circles of the $M_{i,j}$ metric. Qubits from the previous iteration are in black and new ones in white.

be implemented. Noticing that the red CNOT gates in Fig. 3(b) can be applied simultaneously with the gates in Fig. 3(a), this can be done in 6 time steps, leading to an additional $6\log_2(L) - 6$ time steps in total in the RG encoder.

It is key to our routine to be able to "spread out" the qubits between each MERA step. We now show that this can be achieved in linear time by routing qubits through the planar grid. We firstly consider a single step of moving from a 2^k to a 2^{k+1} sized grid.

Our first observation is that while the qubits lie on the edges of our $2^k \times 2^k$ grid, one can subdivide this grid into one of dimensions $(2^{k+1}+1) \times (2^{k+1}+1)$, such that the qubits lie on corners of this new grid, labelled by their positions (i, j) with the centre of the grid identified with $(0, 0)$. Under the taxicab metric we can measure the distance of qubits from the centre as $M_{i,j} := |(i, j)| = |i| + |j|$ and one can check that qubits only ever lie at odd values of this metric, essentially forming a series of concentric circles with $M_{i,j} = 2n + 1, n \in \mathbb{N}$. See figure 10.

A general routing step requires enlarging these circles such that the initial radii R_I are mapped to final radii R_F in the following fashion:

$$\begin{array}{ccccc}
R_I & & R_F & & STEPS \\
2^k - 1 & \rightarrow & 2^{k+1} - 1 & & 2^{k-1} \\
2^k - 3 & \rightarrow & 2^{k+1} - 7 & & 2^{k-1} - 2 \\
2^k - 5 & \rightarrow & 2^{k+1} - 9 & & 2^{k-1} - 2 \\
2^k - 7 & \rightarrow & 2^{k+1} - 15 & & 2^{k-1} - 4 \\
& \vdots & & & \vdots \\
3 & \rightarrow & 7 & & 2
\end{array} \tag{C1}$$

Routing the qubits requires a series of SWAP gates to iteratively make the circles larger, e.g. $3 \rightarrow 5 \rightarrow 7$, the number of steps this requires is shown in table C1. At the initial time step, it is only possible to move the outermost circle ($R_I = 2^k - 1$) since all smaller circles are adjacent. One can check though, that the number of steps required to move these smaller circles is sufficiently small that it is possible to start moving them at a later time step. We provide a framework for the required steps in equation C2.

Thus all the qubits can be moved in 2^{k-1} steps. Each step requires (possibly) simultaneous SWAP gates, each of which can be decomposed into three CNOT gates. Thus the overall run time of each iteration is $3 \cdot 2^{k-1}$. To start from the $L = 2$ base code and enlarge to a desired $L = 2^m$ requires $\log_2(L) - 1$ iterations and thus the overall run time for the routing routine is given by the geometric series $\sum_{k=1}^{\log_2(L)-1} 3 \cdot 2^{k-1} = \frac{3}{2}(L - 2)$.

Combining this with the time to emulate a toric architecture with a planar architecture, and the 10 time steps required to encode the $L = 2$ base case using the Gottesman encoder (see Appendix A), the total number of time steps required for the local RG encoder is $15L/2 - 6\log_2 L + 7 \sim O(L)$, where L must be a power of 2.

$$\begin{array}{rrrrcr}
Timestep & & & & & \\
1. & 2^k - 1 \rightarrow 2^k + 1 & WAIT & WAIT & \dots & WAIT \\
2. & 2^k + 1 \rightarrow 2^k + 3 & 2^k - 3 \rightarrow 2^k - 1 & WAIT & \dots & WAIT \\
3. & 2^k + 3 \rightarrow 2^k + 5 & 2^k - 1 \rightarrow 2^k + 1 & 2^k - 5 \rightarrow 2^k - 3 & \dots & WAIT \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
2^{k-1} - 1. & 2^{k+1} - 5 \rightarrow 2^{k+1} - 3 & 2^{k+1} - 9 \rightarrow 2^{k+1} - 7 & 2^{k+1} - 13 \rightarrow 2^{k+1} - 11 & \dots & 3 \rightarrow 5 \\
2^{k-1}. & 2^{k+1} - 3 \rightarrow 2^{k+1} - 1 & DONE & 2^{k+1} - 11 \rightarrow 2^{k+1} - 9 & \dots & 5 \rightarrow 7
\end{array} \tag{C2}$$