Optimal local unitary encoding circuits for the surface code

(Dated: December 9, 2019)

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. By providing a $O(L)$ circuit to encode a toric code from a planar code, we show how a distance L toric code can be encoded using $3L+2$ time steps. We further show how an $O(L)$ complexity local unitary encoder for the surface code can also be found by enforcing locality in the $O(\log L)$-depth non-local renormalisation encoder.

I. INTRODUCTION

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 [2] 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 a logical operator:

$$d = \min_{e \in C(\mathcal{S}) \setminus \mathcal{S}} \mathrm{wt}(e). \tag{1}$$

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 either $\{I, X\}^n$ or $\{I, Z\}^n$.

III. THE SURFACE CODE

The surface code is a CSS code introduced by Kitaev [1, 3], 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.

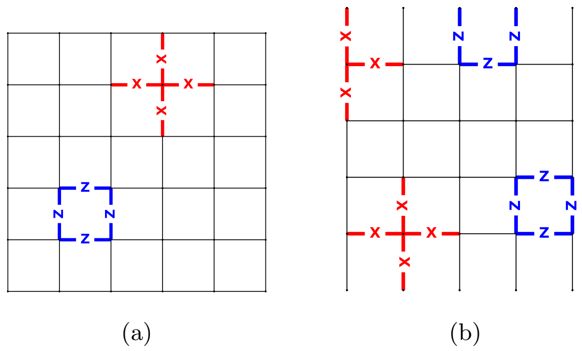


FIG. 1: The check operators for (a) the toric code and (b) the planar code. Opposite edges in (a) are identified.

IV. ENCODING AN UNKNOWN STATE

We are interested in finding a unitary encoding circuit that maps a product state $|\phi_0\rangle \otimes \ldots |\phi_{k-1}\rangle \otimes |0\rangle^{\otimes(n-k)}$ 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 $k, k+1, \ldots, n-1$, we note that the initial product state is a $+1$-eigenstate of the stabilisers $Z_k, Z_{k+1}, \ldots, Z_{n-1}$. Thus, we wish to find a unitary encoding circuit that maps the stabilisers $Z_k, Z_{k+1}, \ldots, Z_{n-1}$ 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 $Z_0, Z_1, \ldots, Z_{k-1}$

and $X_0, X_1, \ldots, X_{k-1}$ of the physical qubits to the corresponding logical operators $\bar{Z}_0, \bar{Z}_1, \ldots, \bar{Z}_{k-1}$ and $\bar{X}_0, \bar{X}_1, \ldots, \bar{X}_{k-1}$ of the encoded qubits (up to stabilisers).

Applying a unitary U to an eigenstate $|\psi\rangle$ of an operator S (with eigenvalue s), we find that $US|\psi\rangle = sU|\psi\rangle = USU^\dagger U|\psi\rangle$: 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:

$$XI \leftrightarrow XX, \quad IZ \leftrightarrow ZZ, \tag{2}$$

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 $Z_1 \rightarrow Z_0Z_1$ and leaves Z_2 invariant. Then applying CNOT$_{12}$ transforms $Z_2 \rightarrow Z_1Z_2$ and leaves Z_0Z_1 invariant. We can also verify that the logical X undergoes the required transformation $X_0 \rightarrow \bar{X}_0 := X_0X_1X_2$.

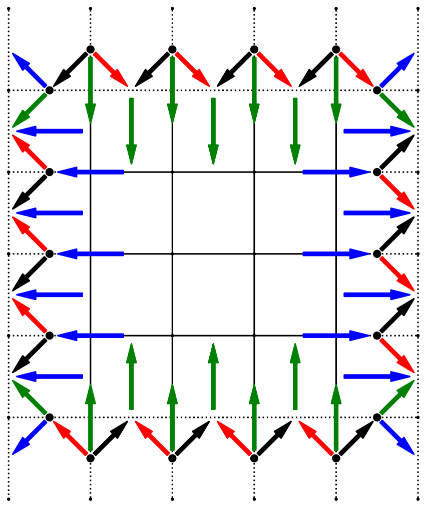


FIG. 2: Circuit to encode a distance 6 planar code from a distance 4 planar code. Each edge corresponds to a qubit. Each arrow denotes a CNOT gate, pointing from control to target. Filled black circles (centred on edges) denote Hadamard gates, which are applied at the beginning of the circuit. The colour 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.

V. ENCODING CIRCUIT 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 encodes a distance $L+1$ planar code from a distance L code by turning smooth edges into rough edges and vice versa. This inductive step requires $O(L)$ time steps (and in fact any local unitary circuit that rotates a planar by $\pi/2$ must take $\Omega(L)$ time, since "rotating" the logical operators $\bar{X}$ and $\bar{Z}$ requires $O(L)$ time using local gates). As a result encoding a distance L planar code from an unencoded qubit requires $O(L^2)$ time steps, which is quadratically slower than the lower bound given by Bravyi *et al.* [4].

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, 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 $O(L)$ lower bound provided by Bravyi *et al.* [4].

VI. ENCODING A TORIC CODE FROM A PLANAR CODE

While the method in Section V 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. 3 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 steps using the circuits given in this Section and in Section V.

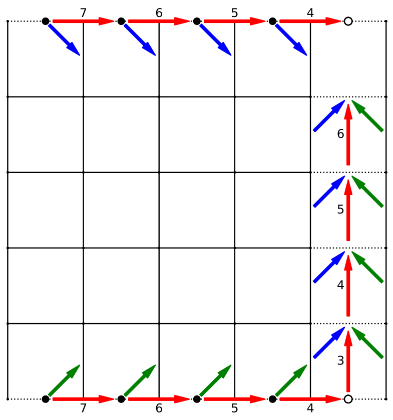


FIG. 3: Circuit to encode a distance 5 toric code from a distance 5 planar code. Solid edges correspond to qubits in the original planar code and dotted edges correspond to qubits added for the toric code. Opposite edges are identified. Arrows denote CNOT gates, and filled black circles denote Hadamard 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.

VII. RENORMALISATION ENCODER

In this section we will describe another route to an $O(L)$ encoder based on the multi-scale entanglement renormalisation ansatz (MERA). The core of this method is to start from an $L = 2$ toric code (sec. VII A) and reverse the order of the scheme outlined originally by Aguado and Vidal [5] as shown in figure INSERTNUMBER. Since each step enlarges a depth k code to a depth $2k$ code, this provides an $O(\log L)$ encoder but is non-local in its original form.

To enforce locality one can "spread out" the qubits between each iteration. This is achieved by routing the qubits through the grid which we outline in section VII C.

There is an added complication that each MERA step only works on a toric code. Since at each iteration (until the final one) we have just a subset of the torus, we in fact have a planar architecture. Thus it is necessary to emulate a toric architecture on a planar one. This can be achieved by performing 2^{k-1} SWAPs to move the top and bottom boundaries together before applying any necessary gates which are now local. Then 2^{k-1} SWAPs are required to move the the boundaries back to their original positions. The identical procedure can be applied to the left and right boundaries. Thus there is an overhead of 2^{k+1} to emulate the torus. Starting from $L = 2$ and ending on a size L code gives an overall overhead to emulating the torus of $\sum_{i=1}^{\log_2(L)-1} 2^{k+1} = 2(L-1)$

Combining this with the result of section VII C yields a run time of $7(L-1)/2 \sim O(L)$.

WHAT ABOUT TIME TO MAKE L=2?? and the time to do the MERA step - its constant depth but if we're being precise then...

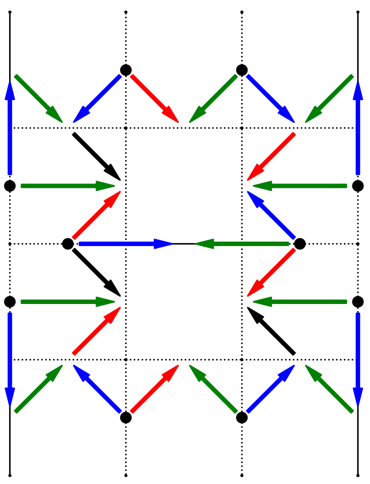


FIG. 4: Circuit to encode a distance 4 planar code from a distance 2 planar code. Gates of the same colour can all be performed in parallel (Figure 2).

A. Base Toric Code

From the general method in cite[Got] we present a simplification, for a class including the surface and toric codes, and from it to derive the encoding circuit of the $L = 2$ toric code from the stabiliser generators.

B. Generalisation to the surface code

The surface code is an alternative error correcting code that has a similar set of stabilisers, but with open boundary conditions as opposed to periodic boundary conditions. The stabiliser generators of the surface code are made of the plaquette and site operators as shown in Figure 1.

The renormalisation encoder continues in a similar way to the renormalisation encoder for the toric code outlined above. A distance L surface code is "spread" out into a distance $2L$, followed by two groups of local CNOT gates to complete the set of stabiliser generators. The first set of local CNOT gates acts to encode 4 distance $\frac{L}{2}$ surface codes within a grid of qubits that will define the distance L surface code. The second set of CNOT gates acts to "stitch" together multiple smaller surface codes, completing the set of stabiliser generators.

C. Routing

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 5.

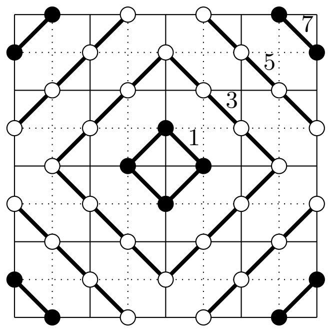


FIG. 5: $L = 4$ code showing the circles of the $M_{i,j}$ metric. Qubits from the previous iteration are in black and new ones in white.

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:

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

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 3. 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 appendix C.

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 $\sum_{k=1}^{\log_2(L)-1} 3 \cdot 2^{k-1} = \frac{3}{2}(L-1)$.

ACKNOWLEDGMENTS

Acknowledgements.

Appendix A: Gottesman Procedure

From a set of check operators one can produce a corresponding bi-matrix

$$\left(X \middle| Z\right)$$

where each row represents a check operator and each column represents a qubit. On the left, $Xij = 1$ indicates that check operator i applies a X operator to qubit j as opposed to the identity, similarly for the right hand side $Z_{ij} = 1$ implies check operator i applies Z to qubit j. If both $X_{ij} = 1$ and $Z_{ij} = 1$, then check operator i applies a Y operator on qubit j.

For stabiliser codes with check operators $P_n \in \{I, X\}^{\otimes n} \cup \{I, Z\}^{\otimes n}$ the corresponding bi-matrix takes the form,

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

Via separate Gaussian eliminations, the left and right matrices bi-matrices of this simplified form can be taken to standard form cite[] without changing the stabiliser group of the code. The standard form of the bi-matrix is then

$$\left(\begin{array}{ccc|ccc} I & A & B & 0 & 0 & 0 \\ 0 & 0 & 0 & C & I & D \end{array}\right)$$

Given initially some k qubits we wish to encode and some additional $n - k$ auxiliary qubits, initialised in $|0\rangle$, an initial choice of stabiliser generator set is

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

We define the gate, $A_{c(m)}$ to be the operator corresponding to row c of A with operator on m_{th} qubit removed and instead controlled by that m^{th} qubit. The

the general circuit which transforms the initial generator set to the standard form bi matrix is

.

In the case of the $L = 2$ toric code, with qubits labelled left to right and top to bottom as such,

the bi-matrix is

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

where we have labelled explicitly on top of each column the qubit in the original grid it represents. The standard form of this bi-matrix is

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

where we have kept track on the label of each qubit from above.

The collection of stabiliser generators implied by the standard form of the check matrix can be produced from the initialised setup using the circuit, which directly corresponds to this. We choose to switch the labels and positions of qubits 2 and 8 so that every gate is local on the toric grid. the resulting encoding procedure is given in figure[].

1. Gate Count of Simplified Gottesman Method

The dominant contribution to the scaling of the Gottesman method is the controlled stabiliser stage. For each m with $1 \leq m \leq r$, up to $n-r$ are gates are applied, giving a maximum $r(n-r)$ gates. For the restricted set of codes with all check operators $P_n \in \{I, X\}^{\otimes n} \cup \{I, Z\}^{\otimes n}$, r is the number of check operators of the form $\{I, X\}^{\otimes n}$. Writing $r(n-r) = r(n-r-k) + kr$, we observe that since k is constant, the dominant scaling in the gate count comes from the term $r(n-r-k)$ which is the number of X check operators multiplied by the number of Z check operators. In any code for which the number n_x of X and number n_z of Z operators both scale with $n_x + n_z$, (for example if they keep a constant ratio), then the scaling in the gate count of the Gottesman method is $o(n^2)$ where n is the number of stabiliser generators of the code.

2. Time Complexity of Simplified Gottesman Method

The final step in the Gottesman method is has at worst $r(n-r)$ gates, each one of them some controlled 2 gate operation $G(i,j)$ with i,j in $\{1\ldots r\}$ and $\{r+1\ldots n\}$ respectively. $G(i,j)$ is implemented before $G(i',j')$ so long as $i < i'$. This step has many gates which can be performed in parallel, in particular the circuit may be simply reshuffled into a form where $G(i,j)$ and $G(i',j')$ are in parallel so long as $i + j = i' + j'$. As such this reduces the number of time-steps from $r(n-r)$ to $n-r+r = n$. The time complexity of the Simplified Gottesman method is then $o(n) = o(L^2)$.

Assuming still that for all n, $P_n \in \{I, X\}^{\otimes n} \cup \{I, Z\}^{\otimes n}$, all gates in the controlled stabiliser stage are $CNOT$'s. Any circuit consisting only of $CNOT$ gates can be made local using a Steiner Tree algorithm which consists of implementing a $CNOT$ for every step of a Gaussian elimination procedure on an $n \times n$ matrix. $o(n^2) = o(L^4)$ new $CNOT$'s are required, it is unclear whether in general the gates in the encoding circuit may be made parallel so that the time complexity is less than $o(L^4)$.

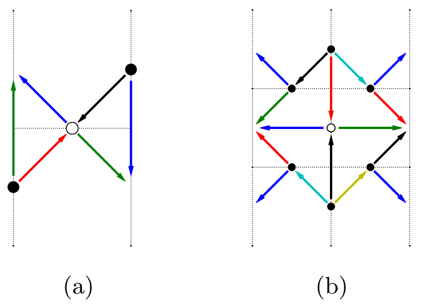
Appendix B: Planar encoding circuits for $L < 5$

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 V.

Appendix C: Sorting Routine

Timestep					
1.	$2^k-1 \to 2^k+1$	$WAIT$	$WAIT$	$\ldots$	$WAIT$
2.	$2^k+1 \to 2^k+3$	$2^k-3 \to 2^k-1$	$WAIT$	$\ldots$	$WAIT$
3.	$2^k+3 \to 2^k+5$	$2^k-1 \to 2^k+1$	$2^k-5 \to 2^k-3$	$\ldots$	$WAIT$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$2^{k-1}-1.$	$2^{k+1}-5 \to 2^{k+1}-3$	$2^{k+1}-9 \to 2^{k+1}-7$	$2^{k+1}-13 \to 2^{k+1}-11$	$\ldots$	$3 \to 5$
$2^{k-1}.$	$2^{k+1}-3 \to 2^{k+1}-1$	$DONE$	$2^{k+1}-11 \to 2^{k+1}-9$	$\ldots$	$5 \to 7$

[1] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, Journal of Mathematical Physics **43**, 4452 (2002).



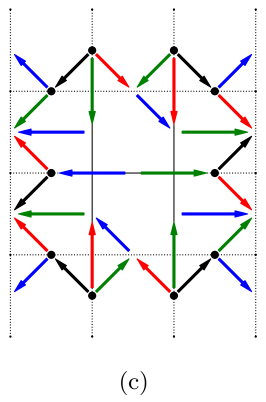


FIG. 6: Encoding circuits for the L=2, L=3 and L=4 planar codes. Each edge corresponds to a qubit, each arrow denotes a CNOT gate pointing from control to target, and each filled black circle denotes a Hadamard gate applied at the beginning of the circuit. The colour of each CNOT gate corresponds to 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.

[2] D. Gottesman, Stabilizer codes and quantum error correction, arXiv preprint quant-ph/9705052 (1997).

[3] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Annals of Physics **303**, 2 (2003).

[4] S. Bravyi, M. B. Hastings, and F. Verstraete, Lieb-robinson bounds and the generation of correlations and topological quantum order, Physical review letters **97**, 050401 (2006).

[5] M. Aguado and G. Vidal, Entanglement renormalization and topological order, Phys. Rev. Lett. **100**, 070404 (2008).