Survey of High threshold universal quantum computation on the surface code

Shivaprasad U Hulyal and Shashank Ravi
Indian Institute of Technology, Madras
(Dated: May 16, 2022)

This article presents quantum computation on surface code which features a two dimensional nearest-neighbor coupled lattice of qubits, a threshold error rate approaching 1%. This makes the scheme one of the best and most practical quantum computing schemes devised to date. Our primary article is of Austin G. Fowler et. al 2009[2]. We restrict the discussion to direct manipulation of the surface code using the stabilizer formalism. The error threshold rates are calculated using the Minimum Weight Perfect Matching (MPWM) algorithm. This was the best known threshold rate at the time the paper was published.

Keywords: Surface Code, Stabilizer Formalism, Quantum Error Correction

I. INTRODUCTION

Quantum computation involves manipulation using qubits, which are two-level quantum systems. The qubits are governed by the set of Pauli operators[2] whose algebra is

$$\begin{aligned} \hat{X}^2 = -\hat{Y}^2 = \hat{Z}^2 = \hat{I}^2 \\ \hat{X}\hat{Z} = -\hat{Z}\hat{X} \\ [\hat{X}, \hat{Y}] = \hat{X}\hat{Y} - \hat{Y}\hat{X} = -2\hat{Z} \end{aligned} \tag{1}$$

In the surface code, physical qubits are entangled together with the help a sequence of qubit CNOT operations. Subsequent measurements of the entangled states provide a means for error correction and error detection. A set of physical qubits entangled in this way is used to define a logical qubit. This logical qubit has far better performance than the underlying physical qubits due to the entanglement and measurement.

Unlike the classical logic element, a qubit can exist in a superposition of its eigenstates, that is, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. A quantum state is furthermore relatively delicate, easily perturbed by interactions with the outside world. These interactions can be intentional, or can result from errors. These qubit errors can be modeled by introducing random X bit-flip and Z phase-flip operators in the evolution of the qubit state.[2]

$$\begin{aligned} [\hat{X}_a\hat{X}_b\hat{X}_c\hat{X}_d, \hat{Z}_a\hat{Z}_b\hat{Z}_e\hat{Z}_f] = \\ (\hat{X}_a\hat{Z}_a)(\hat{X}_b\hat{Z}_b)\hat{X}_c\hat{X}_d\hat{Z}_e\hat{Z}_f - \\ (\hat{Z}_a\hat{X}_a)(\hat{Z}_b\hat{X}_b)\hat{X}_c\hat{X}_d\hat{Z}_e\hat{Z}_f \\ = 0 \end{aligned} \tag{2}$$

We have used the fact that operators on different qubits always commute, so e.g. $[\hat{X}_a, \hat{Z}_b] = 0$. Measurements of these two-qubit operators are thus compatible, so a two qubit state can actually be a simultaneous eigenstate of both $\hat{X}_a\hat{X}_b$ and $\hat{Z}_a\hat{Z}_b$

II. STABILIZER FORMALISM

Generally, quantum states are represented as a superposition of the basis vectors. But in this case, it will be much more convenient to express this state as the unique simultaneous $+1$ eigenvector of the commuting operators $X \otimes X$ and $Z \otimes Z$. These operators are known as stabilizers. As explained in section III, the error correcting algorithm is largely based on manipulation of these stabilizer operators. Even though it does not specify an arbitrary quantum state, we will restrict our explanation to stabilizers that are tensor products of just the Pauli operators.

Unitary action on a quantum state is simply described by

$$U|\psi\rangle = UMU^\dagger U|\psi\rangle$$

That is, the new set of stabilizers will be $UMU^\dagger$.[3]

Whereas, as far as the measurements of these stabilizers are concerned, we note that there is always a nonzero probability of obtaining at least one of the two eigenstates. So, for example, a single qubit in an unknown state with stabilizer I and the subsequent measurement of the Z operator. We will write the stabilizer of a qubit after such a measurement as $\pm Z$. So extending this, we must also check if other nontrivial stabilizers are present, hence we work with an abelian generator group of stabilizers. If the group contains non-commuting stabilizers, the second and subsequent anticommuting stabilizers are multiplied by the first anticommuting stabilizer. This results in commuting stabilizers, and the first anticommuting stabilizer is swapped with the operator that is being measured. Again the state we have projected into, decides the sign (of the eigen-value.

III. SURFACE CODE

The surface codes evolved Alexei Kitaev's invention known as toric codes [4]. This arose from his efforts to use qubits distributed on the surface of a toroid in order to develop simple models for topological order. Extending the same topology on a two dimensional lattice structure, we describe the features of a surface code.

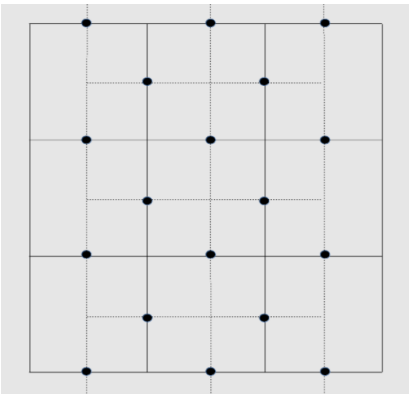

FIG. 1. Basic layout of surface code data qubits, each represented by a circle. A data qubit is located at the center of each edge of a square lattice.

Basic layout of the surface code is as shown in the above figure. Each data qubit is represented by a hollow circle and is placed at the center of each edge of the square "face". Whereas an ancilla qubit, is represented by a solid circle and is located at the corner of each of these faces (vertices)[6]. So, for each vertex v, there is a vertex operator X_v, which is the product of the Pauli X operators acting on the qubits sitting on the edges common to the vertex (which are four qubits in the interior of the lattice and three qubits if the vertex is sitting on the smooth boundary). For each face f, we again have a face operator Z_f, acting as a Pauli Z operator on the four qubits sitting on the edges of the face. There are only three operators if the face is located at the rough boundary. These operators again commute with each other and create a commuting/ abelian subgroup S. The code space C_S created by this group, i.e. the space of all states left invariant by all of these face operators and vertex operators, now turns out to be of dimension 2 and is thus encoding one logical qubit. The code formalised in this way is called the surface code.[2]

Now, there are logical Pauli operators as well. These operators are created by the chains marked in the figure above – Z_L logical operator is associated with the loop of Z operators and the X_L logical operator is associated with the chain of X operators. These operators commute with each other in S as their boundaries are zero i.e.they are open-ended and the fact that they anti-commute is depicted geometrically by the chain and the loop intersecting 'exactly once[5].

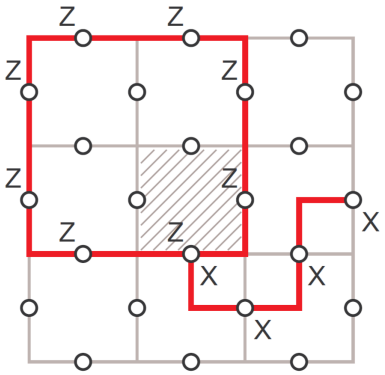

FIG. 2. Logical operators corresponding to the additional degree of freedom introduced by not enforcing the stabilizer associated with the shaded face

Thus we obtain a logical qubit and logical Pauli operators that act on this qubit.Now, to measure a face operator Z_f, we can place an additional measurement qubit in the center of the face and use this qubit as an ancilla in our measurement circuit.

IV. THRESHOLD RATE

The threshold error rate is obtained from four error rates in the simulations performed by the authors—

- p_i: Initialization error - This involves initialising in $|1\rangle$ instead of $|0\rangle$ with probability p
- p_r: Readout error - The eigenstate reported by the Z measurement device is incorrect with probability p
- p_m: Memory error - A- Application of an X, Y or Z gate, each with probability p/3, to an idle qubit.
- p_g: Two - qubit gate error - Application of any one of the 15 different possible nontrivial tensor product of I, X, Y and Z gates . Each of them, with probability p/15, after perfect application of the two-qubit gate.

Each syndrome is read after every iteration of the stabilizer. This value is checked against the previous iteration. If the values differ, then the syndrome change location in time and space is recorded.The syndromes are matched according the Minimum Weight Perfect Matching (detailed in next subsection) as shown in Fig.3

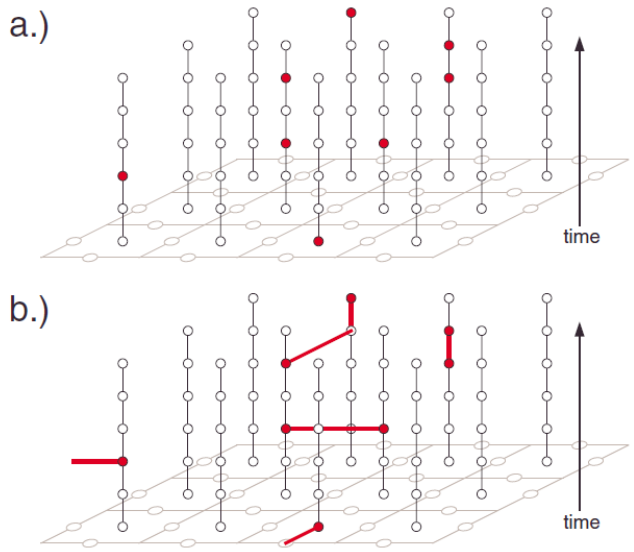

FIG. 3. (a) A red filled dot indicates that he reported syndrome is different from that in the previous time step. The three dimensional structure is just for data visualization and not a physical structure. (b) This shows the optimal matching from MWMPM highly likely to lead to a significant reduction of the number of errors if error correcting bit-flips are applied to the space like edges drawn with red line.

A. Minimum Weight Perfect Matching

The most widely used efficient decoder algorithm for the surface and toric codes code is based on Minimum Weight Perfect Matching (MWPM). MWPM is an problem from graph theory. Edmund's algorithm for solving MWPM (known as the Blossom algorithm) can be used to calculate the lowest weight error consistent with an error syndrome[1]. We create a graph whose vertexes correspond to the set of +1 charge quasi-particles in the error syndrome (recall we are only considering Z errors and vertex stabilizers here). We then create edges between each vertex and every other vertex, i.e. we create the complete graph on the vertices. We now assign a weight to every edge of the graph. The weight assigned is the minimum taxi-cab distance (i.e. the number of edges in the path) between the corresponding vertex operators on the lattice.

The lowest weight error correction operator for the syndrome now corresponds to the Minimum Weight Perfect Matching on the graph. In graph theory, a matching on a graph is a sub-graph where every vertex is connected to at most one edge. In other words we delete edges until no more than one edge is attached to each vertex. This has the effect of leaving vertices either paired or isolated. A perfect matching is a matching where every vertex is paired. Moreover, a minimal weight perfect matching is a perfect matching where the sum of the weights of the edges on the matching is minimized. Given the graph of the syndrome quasi-particles weighted by the taxi-cab length of the minimum path between them, the MWPM algorithm will return a matching which corresponds to the minimal weight correction operator corresponding to the syndrome. The MWPM algorithm returns the most likely error as a correction chain. It is therefore a very successful decoder[1]. It is not, however, optimal, since it can occur that the most likely error operator is not within the most likely equivalence class of errors.

B. Logical Errors

The MWPM is applied iteratively for error correction. In order to know if the simulation should continue or not, we need to determine whether the lattice suffered a logical error (and hence the encoded state has changed). A logical error corresponds to a chain of errors that starts on one boundary and ends on the opposite one[3]. A logical error is detected by repeating the readout cycle with all the error sources set to zero i.e., set $p_i = p_r = p_m = p_g = 0$, in other words have a "perfect readout". That is, there are no errors being simulated but there is a possibility of getting a logical error due to the matching algorithm's error correction method. A logical Z(X) error can be recognized by checking if the parity of Z(X) operators crossing a vertical (horizontal) line of qubits is odd as shown in Fig.4 If there is no logical error is detected, we revert the simulation state to what it was just before the perfect readout cycle of the stabilizers was executed and continue on.

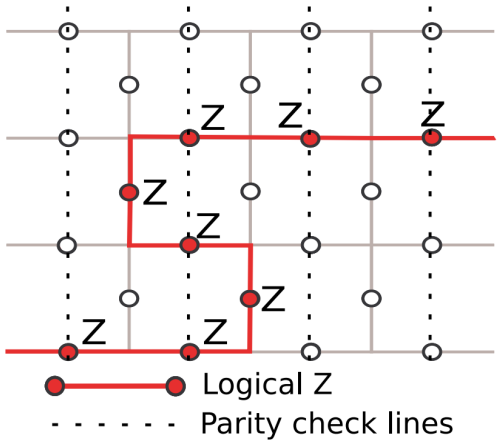

FIG. 4. Logical Z(X) error detection involves checking if the parity of Z(X) operators along any of the vertical horizontal lines of qubits is odd. If it is odd then a logical error has occured, else no error has occured The above figure is an example of logical Z error.

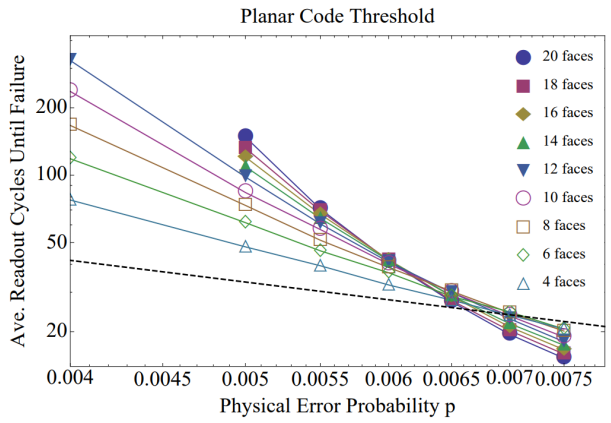

FIG. 5. A plot of average time until failure versus the physical error rate p. A threshold is observed at $p = 6.0 \times 10^{-3}$ where the curves of different lattice sizes cross. The case where no error correction is used (single qubit)is represented by a dashed line.

C. Threshold Results

During every run, the number of cycles it took to for a logical error to be detected is noted. The simulation is run for different lattice sizes and values of the physical error rate p. All of this data is then used to calculate the average number of steps until a logical error occurs for a given lattice size and p. The graph of Fig.5 shows the obtained results. The crossing of different simulations is observed at approximately $p = 6.0 \times 10^{-3}$, which is our numerical threshold. This means that, if the physical error rate is below this threshold value, the average number of readout cycles until failure can be increased arbitrarily by increasing the distance of the code (lattice size). This is better than the previous threshold schemes developed[7].

V. SURFACE CODE AND [n,k,d] CODES*

A distance-d surface code has one logical qubit and $n = d^2$ physical qubits located at sites of a square lattice of size $d \times d$ with open boundary conditions. Qubits and stabilizers are located at the edges and faces respectively. A stabilizer β_f located on a face f applies an X operator (black faces) or a Z operator (white faces) to each qubit on the boundary of that face f. Logical Pauli operators X_L and Z_L have support on the left and the top boundary, as shown in Fig.6. In 1950, Hamming introduced the [7,4] Hamming code. It encodes four data bits into seven bits by adding three parity bits. It can detect and correct single-bit errors. With the addition of an overall parity bit, it can also detect (but not correct) double-bit errors.Hence, in terms of surface codes, the [7,4] Hamming Code has a minimum distance of $d = 3$ and therefore, the surface code lattice has a minimum size of $n = d^2 = 9$.

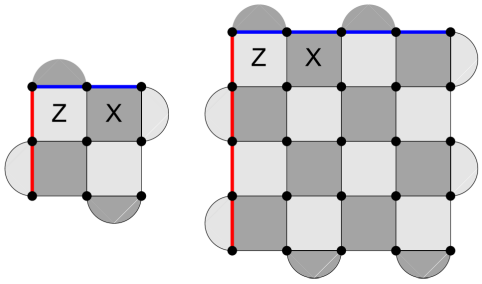

FIG. 6. Surface codes with distance $d = 3$ and $d = 5$

[1] *This section corresponds to the answers to the questions put forward during the term paper presentation.

REFERENCES

[1] Dan Browne. “The Toric Code”. In: (1996), pp. 21–44.

[2] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. “High-threshold universal quantum computation on the surface code 1”. In: (2009), pp. 1–14. DOI: 10.1103/PhysRevA.80.052312.

[3] Austin G. Fowler et al. “Surface codes: towards large-scale quantum computation”. In: *Phys. Rev. A* 86 (2012), p. 032324. URL: https://arxiv.org/ftp/arxiv/papers/1208/1208.0928.pdf.

[4] A. Yu Kitaev. “Fault-tolerant quantum computation by anyons”. In: *Annals of Physics* 303.1 (2003), pp. 2–30. ISSN: 00034916. DOI: 10.1016/S0003-4916(02)00018-0. arXiv: 9707021 [quant-ph].

[5] Argonne Quantum and Computing Tutorial. “Introduction to Quantum Error Correction. What is needed to make Quantum”. In: (2018).

[6] *Quantum Error Correction and Surface Codes*. https://leftasexercise.com/2019/04/08/quantum-error-correction-the-surface-code/. Accessed: 2022-04-30.

[7] Robert Raussendorf and Jim Harrington. “Fault-tolerant quantum computation with high threshold in two dimensions”. In: *Physical Review Letters* 98.19 (2007), pp. 1–4. ISSN: 00319007. DOI: 10.1103/PhysRevLett.98.190504. arXiv: 0610082 [quant-ph].