Review of the Planar Honeycomb Code

Neema Badihian
University of Southern California

(Dated: December 12, 2022)

A review of the current work by Gidney, Newman, and McEwen on the planar honeycomb code and their use of the software Stim.

I. INTRODUCTION

The planar honeycomb code is rather new but its roots are older than one might expect. It began when Alexei Kitaev created the honeycomb model in 2005[1]. In this model, he described a hexagonal lattice where each vertex represented a single qubit and edges representing Pauli operators. As recently as last year, this model was used by Hastings and Haah to create the first honeycomb code[2]. Their code involved logical observables but brilliantly, these observables are altered as the code is used. Other traits of this code that stand out are that it has weight two parity measurements and a degree three grid of qubits yet despite this, it yields an extremely strong memory, which is uncommon with such sparsity. [3] pursued the honeycomb code as an opportunity to test an in-house software they had developed known as Stim. As they applied the code with their software, they learned more about the honeycomb and managed to create new breakthroughs for it. As you continue to read, I will summarize the planar honeycomb code they used and the results of running Monte Carlo sampling on it in order to measure its abilities.

II. HONEYCOMB CODE

As with Kitaev's honeycomb model, the honeycomb code has a hexagonal lattice. Each edge represents two Pauli operators where each is applied to a separate qubit connected to the edge. This hexagonal pattern allows each qubit to have three edges connecting to it, each of a different Pauli operator. The plaquettes of this lattice represent six qubit stabilizers. Each plaquette is identified as a different Pauli depending on the edges surrounding it. A Y-Pauli plaquette would be made up of three X-Pauli edges and three Z-Pauli edges, as the X-Pauli and the Z-Pauli anti-commute to create the Y-Pauli. In Figure 1, the X-Pauli operators are represented by the color red, the Y-Pauli operators are represented by the color green, and the Z-Pauli operators are represented by the color blue. In the image, the edges are all black but they should be thought of as the color missing from either side of the edge. This means that when there is an edge with red on one side and green on the other, you should think of that edge as blue. Or in other words, an edge with a X-Pauli on one side and a Y-Pauli on the other side would be a Z-Pauli.

The honeycomb code goes through a series of rounds where a specific Pauli is measured on the edges each round. One round, all the X-Pauli edges might be measured. Another round, all the Y-Pauli edges might be measured. The edges are measured in a pattern. Hastings and Haah used a cyclic pattern going from X-Pauli measurements to Y-Pauli measurements to Z-Pauli measurements. This red-green-blue pattern was a defining characteristic of the periodic honeycomb code they identified but the work of [3] involves a different pattern; we will get to that. When two measurements have been performed on the edges and these two measurements are the same as the edges surrounding a plaquette type, the plaquettes of this type's stabilizers are recalculated. That means if X-Pauli edge measurements are done and then Y-Pauli measurements are done, the stabilizers of Z-Pauli plaquettes are recalculated.

The planar honeycomb code requires boundaries. A random section of the unbounded code is chosen and cut out. The boundaries of this finite code are made up of a specific color depending on the plaquettes it cuts though. If the boundary cuts through red and green plaquettes, a blue boundary is used. If the boundary cuts through red and blue plaquettes, a green boundary is used. And if it were to cut through blue and green plaquettes, a red boundary would be used. Everything outside the boundaries is no longer part of the code and we only use the qubits within our finite lattice. Edges that have been cut by a boundary now only contain single-qubit measurements because the qubit on the other end of the edge is outside the boundary.

Logical qubits are the product of anti-commuting observables. These observables must commute with the stabilizers as well as with the previous and next edge measurements. For this reason, the red-green-blue, or X-Y-Z, pattern used by Hastings and Haah does not work as eventually, the observables will anti-commute with an edge measurement at a boundary. The fact that the edges have been divided along the boundaries and left with only one qubit causes this anti-commuting to occur. To get around this, a different cycle is used. As seen in Figure 2, X-Y-Z edge measurements are done just like with the periodic honeycomb code but instead of repeating immediately, X-Z-Y edge measurements are done instead. We are left with an X-Y-Z-X-Z-Y pattern that repeats cyclically. The issue with this pattern is that it takes longer before some stabilizers are measured, which can cause

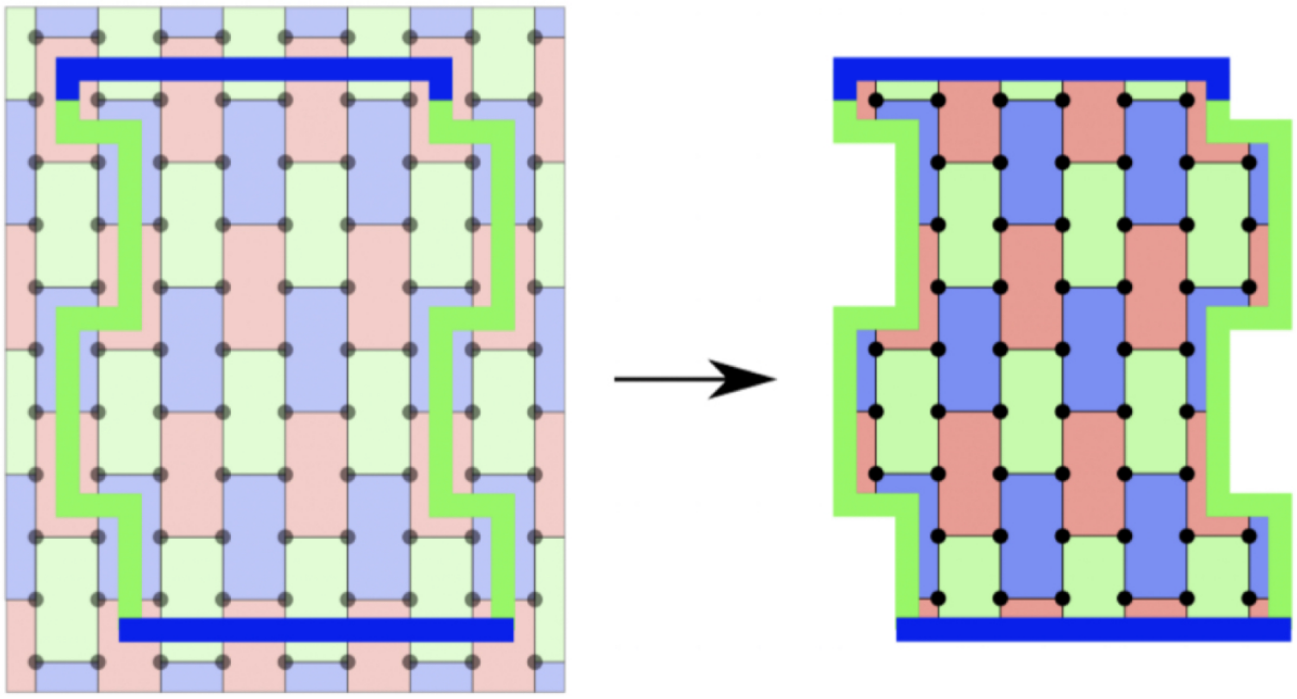

FIG. 1. Creating boundaries for the Planar Honeycomb Code

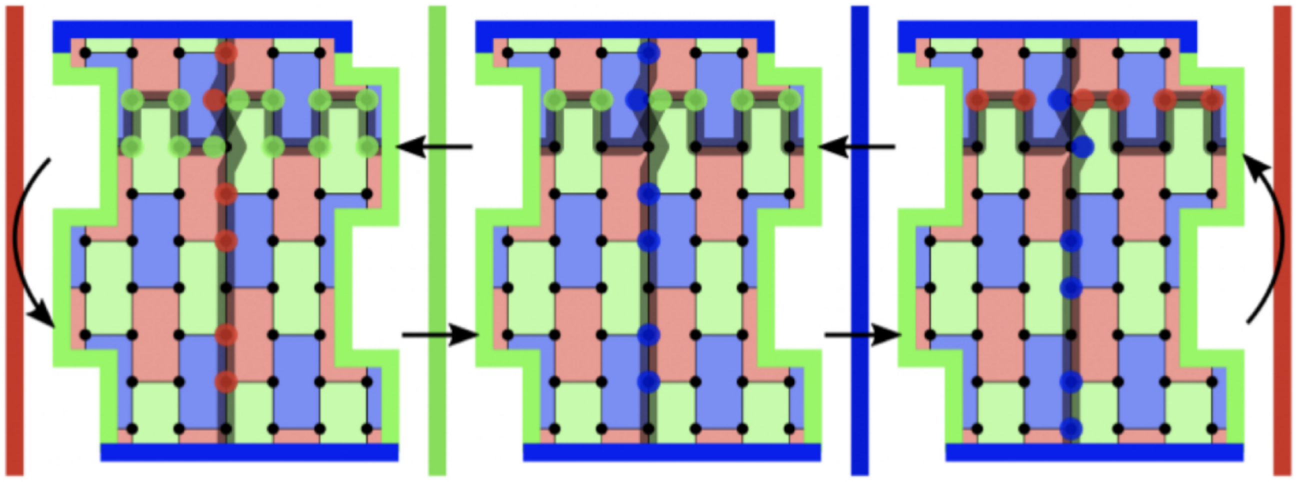

FIG. 2. Edge Measurements and the Transformation of the Code

more noise to grow unchecked. As mentioned before, these stabilizer measurements are done when the two previous edge measurements are the same as the edges surrounding a plaquette. In the cycle we have here, we would do stabilizer measurements in the following order:

$$Z \to X \to Y \to Y \to X \to Z$$

As you can see, X stabilizers only have two stabilizer measurements between them but Y and Z stabilizers alternate between being consecutive and having four measurements between them. This big gap in measurements can lead to error build up over time. According to [3], this results in "zig-zag patterns in the rate that detection events occur." In Figure 3, we can see that there is an issue with collecting plaquettes who lie on a boundary. While other plaquettes are collected twice per cycle, those on the boundary are only calculated once. Surprisingly, this does not seem to cause a noticeable difference.

Some edge measurements do not contribute to their surrounding stabilizers. For instance, a red edge has a

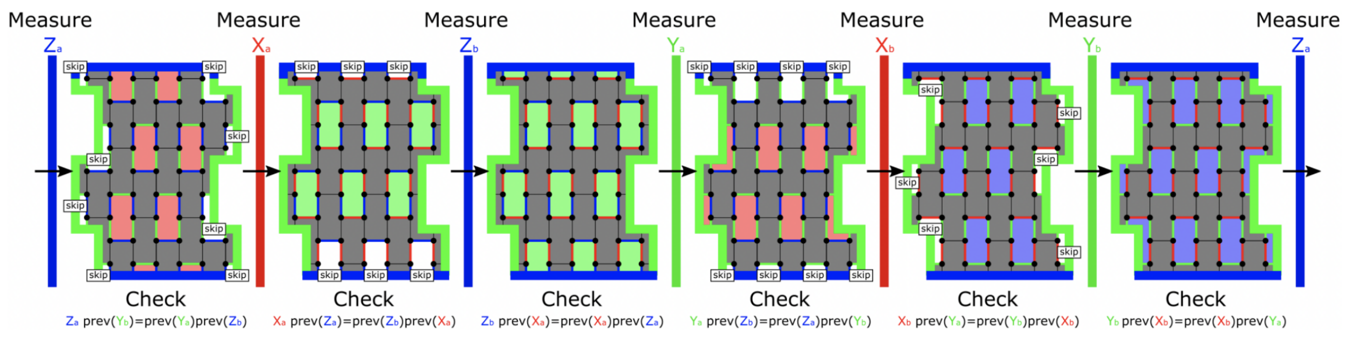
Measure
Za
Measure
Xa
Measure
Zb
Measure
Ya
Measure
Xb
Measure
Yb
Measure
Za
skip
Check
Za prev(Yb)=prev(Ya)prev(Zb)
Check
Xa prev(Za)=prev(Zb)prev(Xa)
Check
Zb prev(Xa)=prev(Xa)prev(Za)
Check
Ya prev(Zb)=prev(Za)prev(Yb)
Check
Xb prev(Ya)=prev(Yb)prev(Xb)
Check
Yb prev(Xb)=prev(Xb)prev(Ya)

FIG. 3. Edge Measurements and the Corresponding Stabilizer Calculations

blue plaquette and a green plaquette on either side. However, when a red edge measurement is done, it does not contribute to both the blue and green plaquettes. Instead, it only contributes to green plaquettes when the red edge measurement is next to a blue edge measurement. Similarly, it only contributes to blue plaquettes when the red edge measurement is next to a green edge measurement. This is because the red edge measurement appears twice on blue plaquettes when the red measurement is next to a blue measurement and is therefore canceled out.

The authors of this paper had previously worked on a honeycomb code where they chose the boundaries so that the ratio of the code's width to height was 2:3. However, the code they used in this paper was one with ratio 1:2. Thanks to the software they had developed, Stim, they were able to see that a 1:2 ratio performed better than a 2:3 ratio. Stim has the ability to find "the smallest undetectable set of 'graphlike' errors that cause a logical error[3]." A graphlike error is defined as an error that creates at most two detection events. A detection event is what occurs when a detector appears on a plaquette. If an X error were to occur on one qubit, two detectors would appear on the non-commuting plaquettes next to the qubit. With this ability of Stim, they noticed faster vertical movement for circuit error models SI1000 and SD6.

In Figure 4, you can see an example of an error mechanism found by Stim that moved faster than expected vertically. The X and Z square tiles in the image refer to the typical X-Pauli and Z-Pauli errors. The YM square tiles, however, refer to a Y-Pauli error as well as a classical measurement error. The circular items refer to detectors and the numbers within are the edge measurement layers that correspond. What is important to note here is that each square tile creates two detectors which cancel each other out. By having one error every two qubits, all detection events disappear. The graph below the honeycomb traces the error circuit through time and space.

III. SIMULATION

In order to quantify the performance of the planar honeycomb code, a simulation was performed using Monte Carlo sampling. Monte Carlo sampling, in its basic form, involves repeating a simulation with some level of randomness involved for unbiased results. The specific traits that were sought after were the threshold, the lambda factors, and the teraquop footprints. A teraquop footprint is the number of qubits required to achieve a failure rate as low as one in a trillion.

Three memory experiments were used for the Monte Carlo sampling with Stim. All three were identical in how they preserve their logical state but each prepared and measured differently. Intuitively, the H-type memory experiments prepared and measured horizontal observables while the V-type memory experiments prepared and measured vertical observables. The EPR-type memory experiments prepared "a perfect EPR pair between the logical qubit and a magically noiseless ancilla qubit[3]." The data from the EPR-pair experiments was not reported because of the distortion from magically noiseless boundaries and because of the unlikeliness of it being possible for real quantum computers.

The circuit error models used for this simulation were the SI1000, the S6, and the EM3 models. The SI1000 model stands for Superconducting-Inspired 1000 ns cycle. A defining characteristic of it is that its measurements are much noisier than its gates. The SD6 model stands for Standard Depolarizing 6 step cycle. It contains homogeneous component errors. Then there is the EM3 model which stands for Entangling-Measurement 3 step cycle. The EM3 model involves two-qubit entangling measurements. Some properties of circuit error models are shown in Figure 5.

There are a number of plots displayed in the original paper but rather than describe each one and show them all here, which would create a lot of clutter, I thought it would be better to discuss the results of the plots. One result that stood out was that the planar honeycomb code does not have a significantly lower threshold than the periodic honeycomb code. In fact, when using the SD6 model, it was slightly higher. Compared to the surface

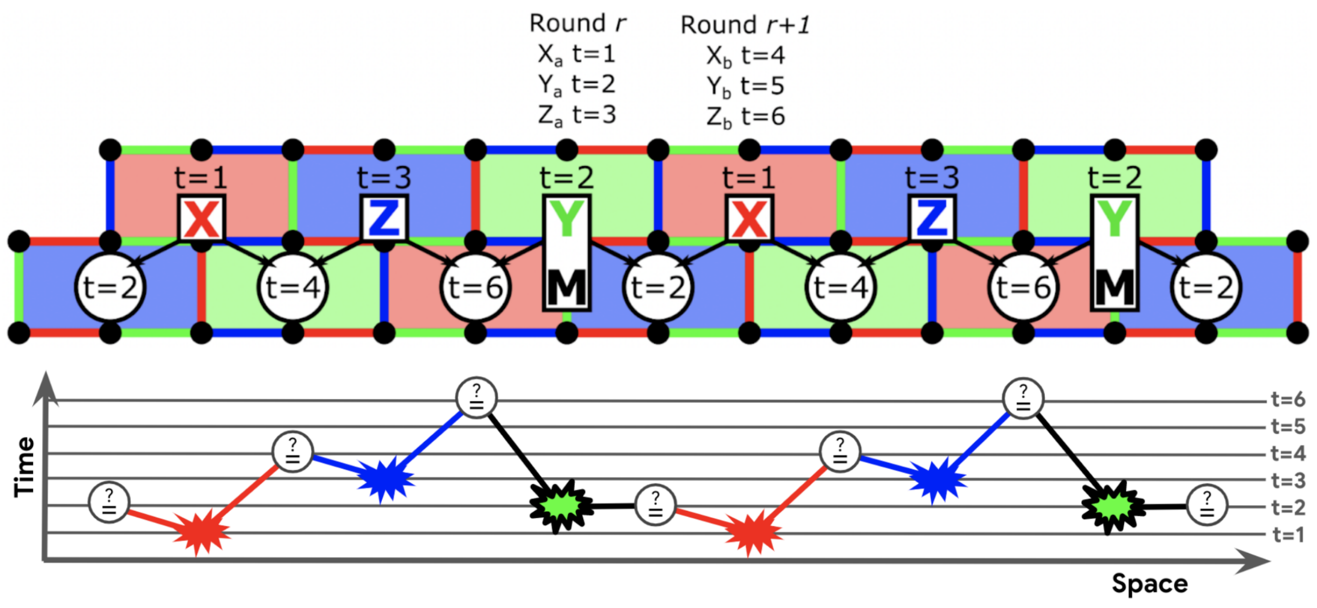

FIG. 4. Vertical Error Mechanism and its Movement through Time and Space

width » height	$h = 3$	$h = 6$	$h = 9$	$h = 12$	$h = 15$	$h = 18$	$h = 21$	$h = 24$	$h = 27$	$h = 30$	$h = 33$	$h = 36$	$h = 39$
EM3	1	2	3	4	5	6	7	8	9	10	11	12	13
SD6	1	3	4	6	7	9	10	12	13	15	16	18	19
SI1000	1	3	4	6	7	9	10	12	13	15	16	18	19
(sheared) EM3	1	2	3	4	5	6	7	8	9	10	11	12	13
(sheared) SD6	1	3	4	6	7	9	10	12	13	15	16	18	19
(sheared) SI1000	1	3	4	6	7	9	10	12	13	15	16	18	19
height » width	$w = 2$	$w = 3$	$w = 4$	$w = 5$	$w = 6$	$w = 7$	$w = 8$	$w = 9$	$w = 10$	$w = 11$	$w = 12$	$w = 13$	$w = 14$
EM3	1	1	2	2	3	3	4	4	5	5	6	6	7
SD6	1	2	3	4	5	6	7	8	9	10	11	12	13
SI1000	1	2	3	4	5	6	7	8	9	10	11	12	13
(sheared) EM3	1	N/A	2	N/A	3	N/A	4	N/A	5	N/A	6	N/A	7
(sheared) SD6	1	N/A	3	N/A	4	N/A	6	N/A	7	N/A	9	N/A	10
(sheared) SI1000	1	N/A	3	N/A	4	N/A	6	N/A	7	N/A	9	N/A	10

FIG. 5. Horizontal and vertical graphlike code distances of planar honeycomb patches under circuit noise.

code, the planar honeycomb code had half the threshold in the SD6 and SI1000 models but it had five to ten times higher of a threshold in the EM3 model.

The lambda factor plot shows how much the error rate decreases as the code distance increases. According to the results, lambda increases linearly with the error rate.

The teraquop footprint seems to have little change with the shift from periodic to planar honeycomb code. Using a physical error rate of 10^{-3}, the SD6 model costs about 7000 qubits, the SI1000 model costs about 50,000 qubits, and the EM3 model costs about 900 qubits. The SI1000 model is close to the threshold at a physical error rate of 10^{-3} so its teraquop footprint was much more costly.

IV. CONCLUSION

In the case of the SI1000 and SD6 models, the planar honeycomb code does not outperform the surface code. It does, however, significantly outperform the surface code with the EM3 model. Compared to the periodic honeycomb code, the differences in performance are negligible. Gidney, Newman, and McEwen make sure to warn that this paper should not be used as a comparison between the planar and periodic honeycomb codes, though. What this paper mostly shows is that the planar honeycomb code has potential to be used for real quantum computers. Experimental testing will need to be done for further insight into the code's capabilities.

[1] A. Kitaev, arXiv **arXiv:cond-mat/0506438**, 10.1016/arXiv.cond-mat/0506438 (2005).

[2] J. H. Matthew B. Hastings, arXiv **arXiv:2107.02194**, 10.22331/arXiv.2107.02194 (2021).

[3] M. M. Craig Gidney, Michael Newman, arXiv **arXiv:2202.11845**, 10.22331/arXiv.2202.11845 (2022).