Simulating Surface Codes using Qiskit

Preetham Tikkireddi[1*] **Atharva Vidwans**[2*]

[1,2] Department of Physics, University of Wisconsin-Madison
tikkireddi@wisc.edu, avidwans@wisc.edu,

Abstract

This paper presents an analysis of the fidelity of surface codes under different noise models using the Qiskit quantum computing platform. We compare the performance of surface codes with respect to different noise models. We discuss the challenges we faced during the implementation of surface codes on Qiskit. Despite being a well-known quantum error correction scheme, surface codes lack a good implementation in Qiskit. We also highlight the complexities involved in implementing surface codes, including the need for braiding for two-qubit logical gates and the additional challenges posed by one-qubit logical gates. We also address various caveats associated with surface codes, including the impact of initialization and readout errors on the stability of surface codes. Our findings demonstrate the need for developing new approaches to implement surface codes on Qiskit and further study the effects of noise on surface codes.

Introduction

Quantum computers hold the potential to solve problems that classical computers cannot solve in a feasible amount of time. However, quantum systems are inherently fragile and highly susceptible to errors in the presence of noise. This poses a significant challenge in the practical realization of quantum computers. Quantum error correction (QEC) is a promising approach to mitigate these errors, and surface codes are one of the most prominent QEC schemes.

Surface codes were first proposed in 1997 by Kitaev as a topological quantum error-correcting code that relies on the topological properties of a two-dimensional lattice. They are a promising candidate for quantum error correction because they have a high threshold error rate and can correct both single-qubit and multi-qubit errors. They are also highly scalable and only needs nearest neighbours connectivity unlike other schemes like Shor's code and color code.

In a surface code, logical qubits are encoded into a two-dimensional lattice of physical qubits. The code relies on the concept of stabilizer generators, which are measurements performed on the physical qubits to detect and correct errors. The stabilizer measurements are chosen in such a way that they commute with each other and can be efficiently implemented using Clifford gates.

[*]These authors contributed equally.

Despite the potential of surface codes for quantum error correction, their implementation on current quantum hardware is challenging. Implementing surface codes requires complex manipulation of multi-qubit states and topological braiding. Moreover, even though surface codes have been around for many years, no one has a good implementation of it on Qiskit that is available on Github. We hope that our study will enable others to learn about the challenges and some common misconceptions. We will also walk through the process of actually creating a logical qubit and logical gates in this paper. Since Qiskit is such a widely adopted package, we hope that future researcher can build on top of our software and have a much easier time implementing surface codes.

Background

Classical Error Correction

In classical computing a bit is the smallest unit of information which can have one of two values, '0' - off state and '1' - on state. Errors can occur due to noise from the environment, which can cause bit flips that can lead to data corruption or loss. To mitigate the impact of these errors, computer scientists have developed error correction codes that encode the information in redundant bits, allowing for the detection and correction of errors. These codes are used to protect important data in various areas, including digital communication and storage.

The simplest form of error correction involves adding redundancy to the original data by replicating it, and then decoding it by taking a majority vote. While this method is not very efficient, it can work well when the error rates are low. This scheme gives us an overall sense for error correction in classical computing.

First, the information is encoded into a higher dimensional space using additional physical bits, resulting in the creation of a logical bit. The logical bit is designed to possess increased robustness against the impact of errors induced by environmental noise or other factors that might lead to bit flip errors.

The encoding process transforms the information into a sequence of bits that can be transmitted or stored. This encoded information is then subject to noise that can induce errors in the sequence of bits. In order to recover the original information, the encoded state is inspected and corrected, if necessary. The correction process involves identifying errors in the encoded sequence, using the redundant bits to identify which bits have been affected, and correcting the errors. Finally,the decoding process retrieves the original information from the corrected encoded state.

However, as error rates increase, more sophisticated error correction codes are needed. Some of the popular classical error correction codes are Hamming codes, Low-Density Parity-Check (LDPC) and Turbo codes.

When designing an error correction code, scientists must balance efficiency, which refers to the ratio of encoded information to the resources needed to encode it, and resiliency to errors, which refers to the level of protection offered against errors. The optimal code depends on the specific requirements of the application and the level of noise in the environment.

Error correction and detection schemes are described using the $[n, k, d]$ scheme. It is a useful way to compare and evaluate different error-correcting codes based on their performance characteristics. The notation specifies three parameters of the code:

- n: the length of the codeword, which is the total number of bits used to encode the data.
- k: the number bits worth of information encoded. Or, the number of logical bits encoded.
- d: the minimum distance between any two codewords, which represents the minimum number of bit errors that the code can detect and correct.

Higher values of d, usually indicate a greater level of error-correcting capability, but may also require more complex decoding algorithms. Different codes may be better suited to different applications, depending on the specific requirements.

Challenges to Quantum Error Correction

Quantum error correction faces a number of challenges due to the unique properties of quantum systems:

- The no-cloning theorem for quantum states makes it impossible to arbitrarily duplicate data, requiring alternative ways of adding redundancy to the system.
- Qubits are susceptible to both bit-flips (X-errors) and phase-flips (Z-errors), requiring codes to detect both error types simultaneously, unlike in classical coding where only bit-flip errors need to be considered.
- The problem of wavefunction collapse presents a significant challenge in quantum error correction. Any measurements performed as part of the error correction procedure must be carefully chosen so as not to cause the wavefunction to collapse and erase the encoded information.
- No-cloning theorem also prevents us from being able to checkpoint our computation which would have allowed us to save the state in the middle of a computation so that we can return to it in case errors occur in the later steps.

Quantum error correction requires alternative ways of adding redundancy to the system due to the additional types or errors, the no-cloning theorem and the destructive nature of measurement in quantum computing. Therefore, directly translating classical codes to quantum codes is not possible.

Quantum Error Correction

Despite these challenges Peter Shor was able to design a groundbreaking solution by utilizing the concept of code concatenation. Shor utilized entanglement to encode quantum states across multiple qubits without duplicating the original state. Furthermore, by combining two distinct codes, Shor's code was able to detect and correct both bit-flip and phase-flip errors simultaneously.

Shor's code is a stabilizer code, which is a powerful tool for constructing and analyzing quantum error-correcting codes. While Shor's code represents a significant milestone in quantum error correction, it is not currently practical for existing quantum hardware due to scalability limitations and the requirement for performing two-qubit gates across many qubits. This is particularly challenging on popular hardware platforms like superconducting systems, which can only perform two-qubit gates on their nearest neighbouring qubits.

Despite these limitations, Shor's code remains a significant contribution to the field of quantum error correction and serves as a cornerstone for understanding the fundamentals of quantum error correction and the stabilizer formalism. The insights gained from Shor's work have paved the way for further developments in quantum error correction and the potential for quantum computers to achieve fault-tolerant quantum computation.

To compare quantum error detection schemes we follow a similar convention to classical $[n, k, d]$ scheme. We just add extra square brackets to denote that it is a quantum scheme, so it looks like this $[[n, k, d]]$.

Stabilizer Codes

Stabilizer codes are a class of quantum error-correcting codes. The basic idea of stabilizer codes is to encode the quantum information in a subspace of the Hilbert space spanned by states that are stabilized by a group of Pauli operators, called the stabilizer group.

The resultant logical state is then encoded in a higher dimensional subspace, called the code space, of the expanded Hilbert space. If an error occurs, the logical state is rotated to an orthogonal error space, which can be detected via a sequence of stabilizer measurements.

In the circuit of an $[[n, k, d]]$ stabilizer code, each of the stabilizers is measured using syndrome extraction circuits. The syndrome extraction circuit maps the logical state onto ancilla qubits and then measured. The stabilizer measurement returns '0' if the stabilizer commutes with an error E, and '1' if the stabilizer anti-commutes with the error. Therefore, if there are no errors the stabilizer measurement has no effect on the logical qubit.

The task of constructing a good stabilizer code involves finding stabilizers that anti-commute with the errors to be detected. In general, two Pauli operators will commute with one another if they intersect non-trivially on an even number of qubits, and anti-commute if they intersect non-trivially on an odd number of qubits. The results of the m stabilizer measurements are combined to give an m-bit syndrome, which allows us to deduce the best recovery operation to restore the logical state to the code-space.

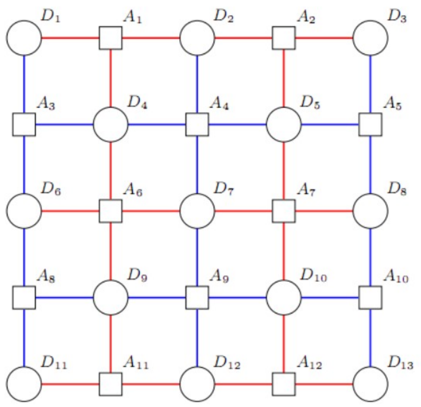

Figure (1) A distance-three surface code with parameters $[[13, 1, 3]]$.

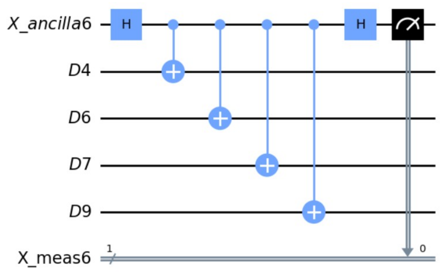

Figure (2) Geometry and quantum circuit for a measure-X qubit, which stabilizes $X_a X_b X_c X_d$.

Surface Codes

Surface codes are part of a broader family of topological codes. The surface code is a modular code that is built up by patching together repeated elements. This modular approach ensures that the surface code can be easily scaled in size while maintaining stabilizer commutativity.

While there exist numerous papers on surface codes, only a few resources provide comprehensive explanations on the topic. In particular, "Quantum Error Correction: An Introductory Guide" and "Surface codes: Towards practical large-scale quantum computation" provide good introductory and in-depth coverage of surface codes, respectively.

However, most existing GitHub repositories only implement 2D nXn surface codes without X-cut or Z-cut qubits, which poses significant limitations. For instance, such implementations can only support a single logical qubit on a 2D array, given that only two X edges and two Z edges can exist on such a surface. Moreover, the number of physical gates required to implement logical single qubit gates

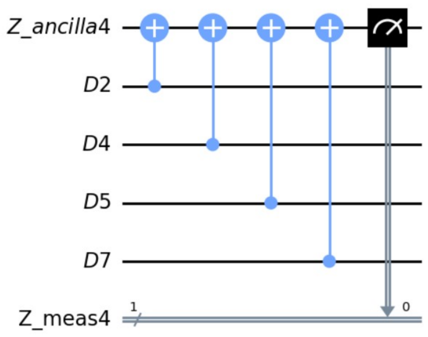

Figure (3) Geometry and quantum circuit for a measure-X qubit, which stabilizes $Z_a Z_b Z_c Z_d$.

increases linearly with the size of the surface code, making scalability a challenge.

To address these limitations, it is crucial to implement X-Cut or Z-Cut qubits, as explained in detail in the aforementioned papers. Doing so enables the creation of scalable surface codes capable of implementing all logical single qubit operations, as well as logical two qubit operations using braiding. This is not possible with uncut surface code implementations, making the adoption of X-Cut or Z-Cut qubits essential for the realization of practical surface code implementation.

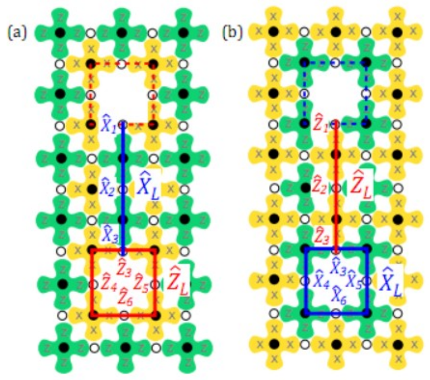

Figure (4) (a) A double Z-cut and (b) a double X-cut qubit, formed by turning off twomeasure- Z and two measure-X qubits, respectively. For the double Z-cut qubit the logical operators are $X_L = X_1\, X_2\, X_3$, a chain that links one Z-cut hole's internal X boundary to the other, and a $Z_L = Z_3\, Z_4\, Z_5\, Z_6$ loop that encloses the lower Z-cut hole. For the X-cut qubit, $Z_L = Z_1\, Z_2\, Z_3$, a chain that links the two internal X-cut holes' Z boundaries, and $X_L = X_3\, X_4\, X_5\, X_6$ loop that encloses the lower X-cut hole. X_L and X_L share one data qubit (qubit 3), so the operators anti-commute. Note that the loop operators can be defined to surround either of the two holes in the qubit. Figure from [2]

Logical qubit and operations on Cut-qubits

To initialize a logical qubit on an X-cut or Z-cut surface code qubit, we need to prepare it in the appropriate eigenstate. For an X-cut qubit, we can easily initialize it in the $|+_L\rangle$ or $|-_L\rangle$ which are the eigenstates of X_L. To do this, we start with a 2D array with no cuts and then create a double-X cut qubit by turning off two measure- X qubits to create two X-cut holes. We know the measurement outcomes of the upper measure- X qubit just before turning them off. If we define X_L as the loop surrounding the upper qubit hole, the initial state of the qubit corresponds to the measurement outcome of the upper measure- X qubit just before it is turned off. If the initial state is not the desired one, we can apply a Z_L to the logical qubit. Applying single qubit gates is a lot more scalable on these cut qubits as they only need a short chain of local gates as opposed to applying gates across the 2D qubit.

Initial Objective

In this research project, we aimed to apply surface codes to various quantum algorithms and compare their performance at different noise levels in order to determine the error thresholds using simulations. We initially assumed that we would find suitable implementations of surface codes on Qiskit, as surface codes are an active area of research with several published papers demonstrating their advantages on hardware.

However, our investigation of available surface code implementations on GitHub revealed that they were not sufficient for our purposes. The available implementations only created single logical qubits capable of implementing logical X and Z gates, which is inadequate for our needs. Moreover, none of the implementations had implemented logical two-qubit gates as this requires X-cut or Z-cut qubits, and implementing the two-qubit gates using braiding is a complex task.

To address these limitations, we decided to build on top of one of the existing GitHub repositories and make significant changes to improve its efficiency and realism. We managed to implement X-cut and Z-cut qubits, which had not been done previously, and developed logical single-qubit gates for them. Additionally, we started working on implementing the logical CNOT gate but were unable to complete it due to time constraints. However, we plan to complete it in the near future.

Our efforts have resulted in a more robust surface code implementation that is suitable for our research purposes. The development of logical two-qubit gates using braiding remains a complex task that requires further investigation, and we believe that our work represents an important step towards achieving this goal. Ultimately, our aim is to contribute to the growing body of research in the field of quantum computing by providing a reliable and efficient implementation of surface codes on Qiskit.

Our version of surface code

In our study, we successfully implemented X-cut and Z-cut qubits, which provide a significant improvement over existing surface code implementations. Most current surface codes rely on regular 2D surface codes that lack any holes or gaps, also known as grid fashion or qubits without cuts or holes. However, with these regular 2D surface codes, only a single logical qubit can be implemented, regardless of the dimension of the 2D surface. This limitation severely restricts the practicality of surface codes for large-scale quantum computing applications.

In contrast, our implementation of X-cut and Z-cut qubits allows for the creation of multiple logical qubits, making them much more scalable than grid fashion qubits. This scalability is a significant advantage for the implementation of surface codes for practical quantum computing applications. Furthermore, we demonstrate that our logical qubits outperform physical qubits, as shown in the subsequent section of this paper.

Our successful implementation of X-cut and Z-cut qubits highlights the potential for advancing quantum computing through the development of more efficient error-correction codes. As our research continues, we expect to further optimize the performance of these qubits and explore new avenues for advancing the field of quantum computing.

In our study, we have demonstrated successful implementation of the Pauli gates, namely X, Y, Z, and H gates on the logical qubit using the X-cut and Z-cut qubits. However, implementing the CNOT gate using these qubits seems to be a complex task.

Fidelity vs Error results

This study aims to analyze the impact of two types of errors on quantum computing. The first type of error, called the gate error, arises due to imperfections in the operations performed on the qubits. To model this error, we use the depolarizing noise, which has a probability of p_gate to replace the state of any qubit with a completely random state. The second type of error, the measurement error, occurs when the qubit state can flip from a 0 to a 1 or vice versa with probability p_meas immediately before measurement. It is important to note that thermal noise is not considered in our study. Therefore, to mitigate the effects of thermal noise on the logical qubit, we propose the use of dynamical decoupling.

We simulate the logical qubits for initial states $|0\rangle$, $|1\rangle$, $|+\rangle$, and $|-\rangle$. We conduct our tests on IBM Qiskit qasm simulator and introduce noise to simulate the working of actual quantum computers. We compare the performance of logical and physical qubits for 14 random error levels, where the gate and measurement errors range from 10^{-5} to 10^{-2}. We execute all tests for 10000 shots. The comparison between the logical and physical qubit error rates for different noise levels is shown in the graphs below. It is worth noting that we plot $1 - F$ on the y-axis instead of F where F is the fidelity, where lower values indicate better performance.

Our findings indicate that the performance of logical qubits is significantly affected by both types of errors, with higher error rates for gate errors compared to measurement errors. Moreover, the physical qubits demonstrate a higher error rate compared to logical qubits, particularly for higher error levels. These results emphasize the need for developing error-correcting codes to mitigate the impact of errors in quantum computing.

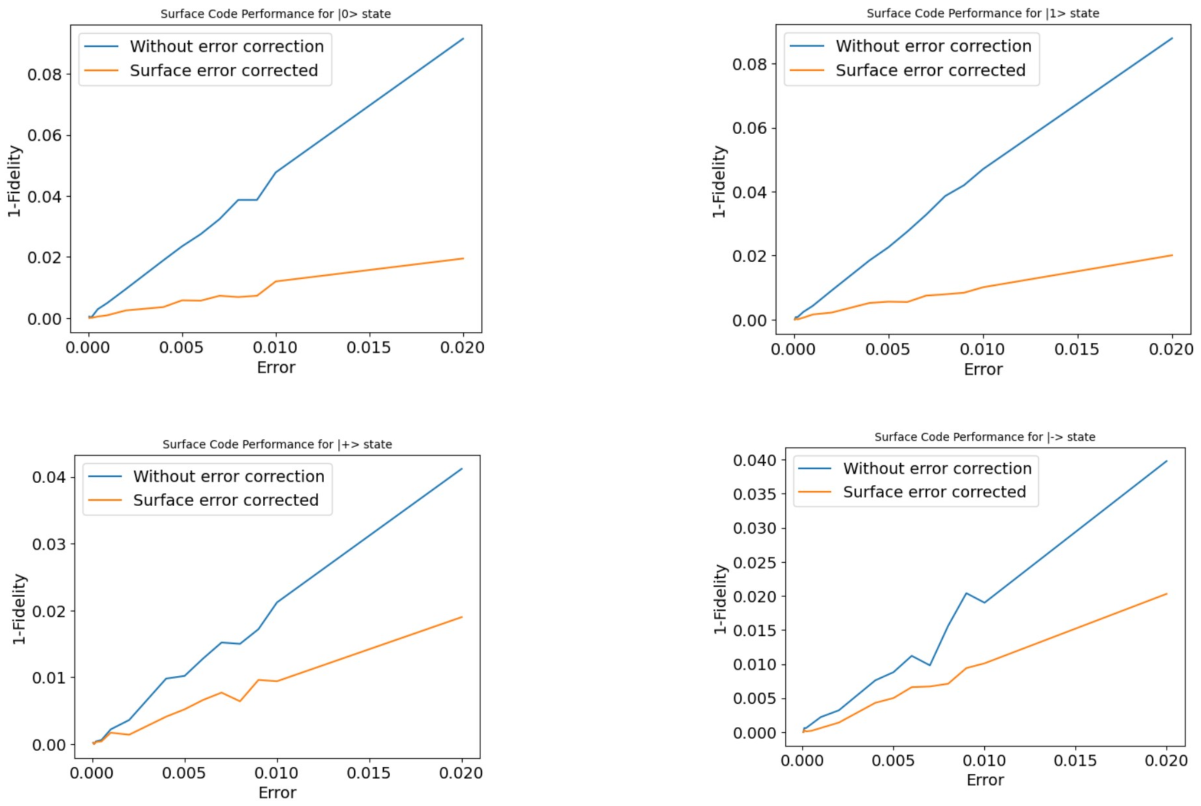

Figure (5) Comparing fidelity of logical and physical qubit at increasing error rates

Conclusion

Our research paper presents successful implementation of X-cut and Z-cut qubits, along with logical X, Y, Z, and H operations. Our logical qubits show consistent fidelity and low error rates, offering an advantage over physical qubits. Despite recent advancements in surface codes, there remains a lack of implementation or packaging of surface codes that can be applied in a scalable manner. Therefore, we strongly believe it will be a good opportunity for implementing scalable surface codes by using qubits with holes, enabling us to apply all single qubit and CNOT operations through braiding. We believe this approach holds significant potential for further advancing the field of quantum computing.

References

[1] Roffe, J. (2019). Quantum error correction: an introductory guide. Contemporary Physics, 60(3), 226-245.

[2] Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). Surface codes: Towards practical large-scale quantum computation. Physical Review A, 86(3), 032324.

[3] ”Suppressing quantum errors by scaling a surface code logical qubit.” Nature 614, no. 7949 (2023): 676-681.

[4] Zhao, Y., Ye, Y., Huang, H. L., Zhang, Y., Wu, D., Guan, H., ... & Pan, J. W. (2022). Realization of an error-correcting surface code with superconducting qubits. Physical Review Letters, 129(3), 030501.