An Introduction to Steane Encoding in Qiskit using the Bell Circuit

Sushant Ranjan
Department of Electrical and Computer Engineering, Spring 2023
Duke Graduate School
E-mail : sushant.ranjan@duke.edu

***Abstract*—Quantum computers are inherently noisy. Errors affect the computation all the time. As the quantum computers scale up the errors increase as well. Quantum Error Correcting codes play a crucial role in such a scenario. The [[7,1,3]] Steane code is one such code. It is one of the earlier codes that has been simulated and implemented experimentally many times over different architectures. In this report I present the simulation of a Steane Encoding in Qiskit and apply it to a Bell Circuit. This helps us visualize the error-correction cycle at different points in the circuit. I also explore different levels of error to see how it affects the outcome.**
All code used in this report can be found on GitHub: https://github.com/RanjanSushant/ECE621_Final_project

***Index Terms*—Steane Code, Error correction, Stabilizer Codes, Qiskit**

I. Introduction

Error correction plays an important role in computing, classical or quantum. It helps maintain the integrity of the data by keeping it error free or by recovering it once an error has occurred. Traditional classical codes cannot help correct various errors that occur in a quantum circuit. But we can use them as a basis to construct quantum error correcting codes.

Generally, a quantum error correcting code is defined by the parameters [[n,k,d]] where n is the number of physical qubits(unencoded qubits), k is the number of logical qubits(encoded qubits), and d is the distance between different codewords in the codespace. Given distance d, a code can correct (d-1)/2 errors. One category of quantum error correcting codes are the stabilizer codes. These codes are similar to classical linear codes. It combines two properties of Pauli algebra. First, Pauli operators have eigenvalue +1 or -1. Second, all Pauli operators except Identity anti-commute with each other. In a broad sense, the qubits are prepared in the +1 eigenstate of a Pauli operator(X or Z) and an error(phase flip or bit flip) changes it to -1 eigenstate. The number of stabilizers for a given quantum error correcting is equal to $2^{(n-k)}$. Usually, we use just the generators of the stabilizer group to denote all the stabilizers. Number of stabilizer generators for any given code is equal to n-k. It can be shown that if a code can correct X and Z type errors it can also correct Y type errors

II. About the Steane Code

Steane code is a type of stabilizer code. It is a [[7,1,3]] code i.e it encodes 7 physical qubits into 1 logical qubit with codewords that have a hamming distance of 3. This makes the number of generators to be 7-1=6. One possible set of generators is given in Fig 1.

Name	Operator
g_1	$I\ I\ I\ X\ X\ X\ X$
g_2	$I\ X\ X\ I\ I\ X\ X$
g_3	$X\ I\ X\ I\ X\ I\ X$
g_4	$I\ I\ I\ Z\ Z\ Z\ Z$
g_5	$I\ Z\ Z\ I\ I\ Z\ Z$
g_6	$Z\ I\ Z\ I\ Z\ I\ Z$

Fig. 1. Stabilizer generators for the Steane seven qubit code. The entries represent tensor products on the respective qubits; for example, ZIZIZIZ = $Z \otimes I \otimes Z \otimes I \otimes Z \otimes I \otimes Z = Z_1 Z_3 Z_5 Z_7$.

The Steane Code has a distance of 3 which means it can correct (3-1)/2 = 1 qubit error. This code is based on the classical Hamming code. Both X and Z checks have the same parity check matrix as can be seen from the stabilizer generators. This implies that similar circuits can be used for both X and Z type checks. The X stabilizers help detect Z type errors and vice-versa because they anti-commute with each other. Error on different qubits change different stabilizers which can be used to distinguish between different errors. These are called syndromes and they help us identify on which qubit the error has occurred. For example, reading from left to right in Fig 1, a Z error on qubit 1 will set off the g_3 stabilizer but an error on qubit 3 will set off both g_3 and g_2. Same is the case for other errors as well.

Apart from the above, the Steane code is also the smallest color code which is another set of quantum error correcting codes which I won't be covering here.

III. Circuit

In order to understand the implementation of Steane Code better, I decided to follow a step by step approach where I start with the ideal Bell state circuit and then keep updating it at each step.

A. Ideal Bell State

To begin with, we look at what an ideal bell state generation circuit looks like. As we can see it requires two qubits.

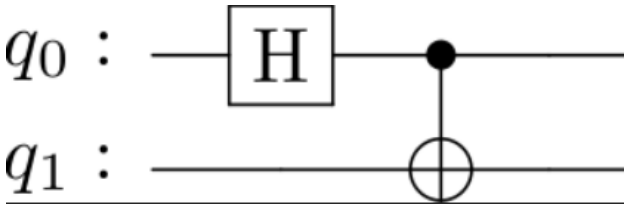

Fig. 2. Ideal bell state creation circuit

If we plot the possible output states as a histogram we get the result as shown in Fig. 3.

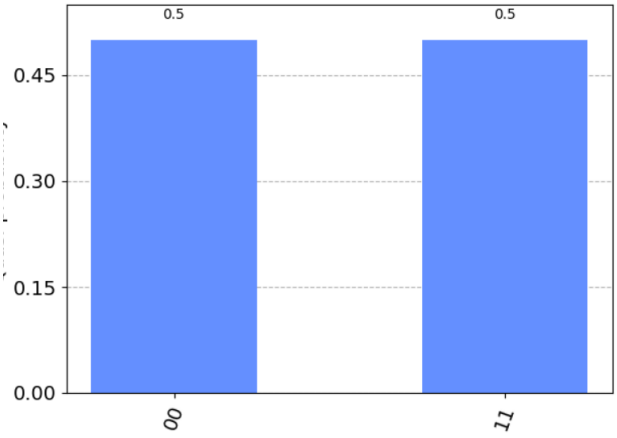

Fig. 3. Bell state outcomes and probabilites

B. Bell circuit with Error

In the next step we introduce random pauli error with certain probability into the circuit. The random error can be on both qubits or just.

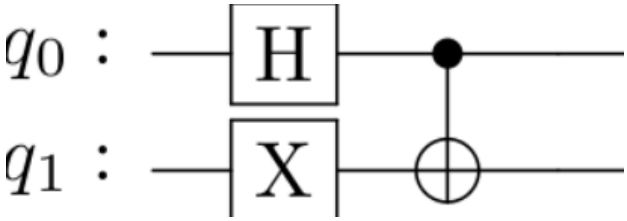

Fig. 4. Example of Bell circuit with randomized Pauli error(X in this case)

If we plot the hisotgram for this particular circuit we can see that the output probabilities are still the same but the states are not the ones that we want.

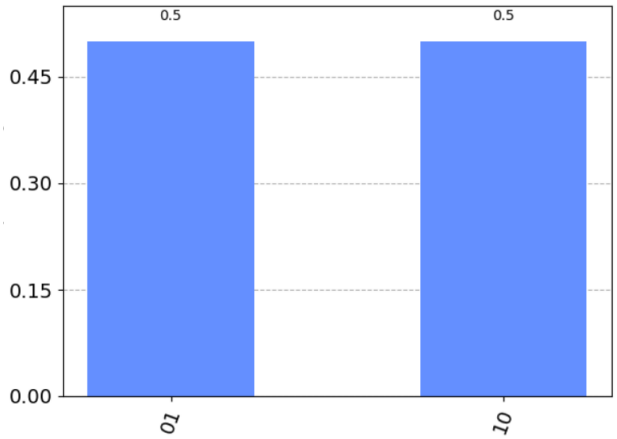

Fig. 5. Example of Bell circuit with randomized Pauli error

C. Fidelities

Before moving to the next step, I think this is a good point to discuss fidelity. Fidelity of a circuit can be thought of as the overlap between what state was expected and what was actually measured. In other words it tells us how close we are to the actual state that we want. We can check the fidelity of our error prone circuit by running it over multiple iterations and then taking the inner product between the expected and measured state. The resultant plot will tell us how the fidelity of the circuit changes as the probability of error changes.

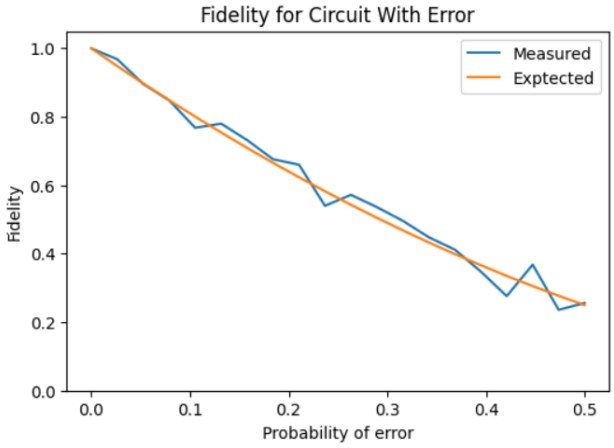

Fig. 6. Fidelity vs. Probability of error

As intuition suggests the fidelity of the circuit goes down as the probability of error increases.

D. Steane encoding

This is one the most important steps as here we generate the circuit for Steane code to be used later for correcting errors. As mentioned earlier, the Steane code encodes one logical qubit in seven physical qubits. Hence, we will two blocks of seven physical qubits to encode our two qubits required for the bell state. The two blocks will be exactly same but represnting two different logical qubits.

The seven physical qubits will be referred to as the data qubits. Since we cannot measure the data qubits directly as it will collapse any superposition, we need supporting bits or ancilla bits which keep track of the parity of various sets of the data qubits. The measured value of the ancilla at any point in the time evolution of the circuit tells us if any error has occurred. We can apply subsequent corrections to fix those errors.

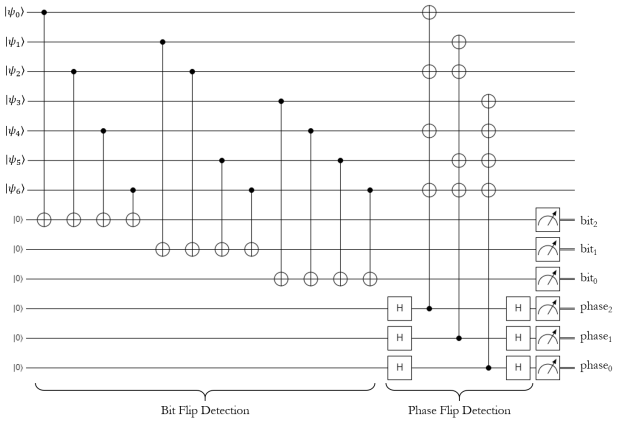

Fig. 7. Steane Encoding circuit (example)

Fig. 7 shows the resulting encoding circuit. It is important to note that this circuit is an example circuit which will be similar to what the code will generate but not exactly the same. This circuit helps us understand how a single block of Steane code is structured inlcuding the relation between the ancilla and the data qubits and the measurement of the ancillas. The ψ state denotes the data qubits. As evident from the circuit the ancilla are prepared in the 0 state.

E. Verifying the Steane Circuit

Before we move on to applying this error correction to our original bell state circuit we need to verify that the Steane code circuit we prepared is the correct one and that it generates correct logical states. We do this by verifying that it generates the correct logical 0 state.

The logical 0 state for Steane code is made up of a superposition of 8 states as shown.

$$\begin{aligned}|0_L\rangle = \frac{1}{\sqrt{8}}(&|0000000\rangle + |0001111\rangle + |0101101\rangle \\ &+ |0110011\rangle + |1101001\rangle + |1010101\rangle \\ &+ |1101111\rangle + |1110000\rangle)\end{aligned} \tag{1}$$

We can verify this by using the state_vector simulator of Qiskit and just printing the results.

F. Steane Encoded Bell Circuit

Up until what we have created are just two logical qubits using two Steane code blocks. In order to perform logical operations on these blocks we need to the block operator equivalents of the Pauli operators that we require, i.e. Hadamard and CNOT. When creating such logical operators we need to keep two things in mind. One, the operator doesn't cause the codeword state to fall out of the space. Secondly, All the parts of the block operator should introduce no more than the amount of error than the code can handle which in our case is 1. As we will see it is really easy to implement block level Hadamard and CNOTs.

1) Logical Hadamard: The logical Hadamard gate can be applied in a pretty straightforward manner by just applying Hadamards to all physical qubits of the logical qubit we want to apply the Hadamard to. This means there will be seven Hadamards applied

$$H_L = H_1 H_2 H_3 H_4 H_5 H_6 H_7$$

This logical Hadamard is fault-tolerant and it keeps the resulting state in the codespace as it just maps the X stabilizers to the Z and vice-versa.

2) Logical CNOT: As the CNOT is a qubit gate, the logical CNOT requires two steane blocks. Unlike some other quantum error correcting codes Steane gives us the capibility to perform the CNOT in a much easier way. This is called a transversal CNOT. Here, the qubits from block one act as control to the qubits of block two which are targets. The same qubit is connected via a CNOT and overall result is a logical CNOT

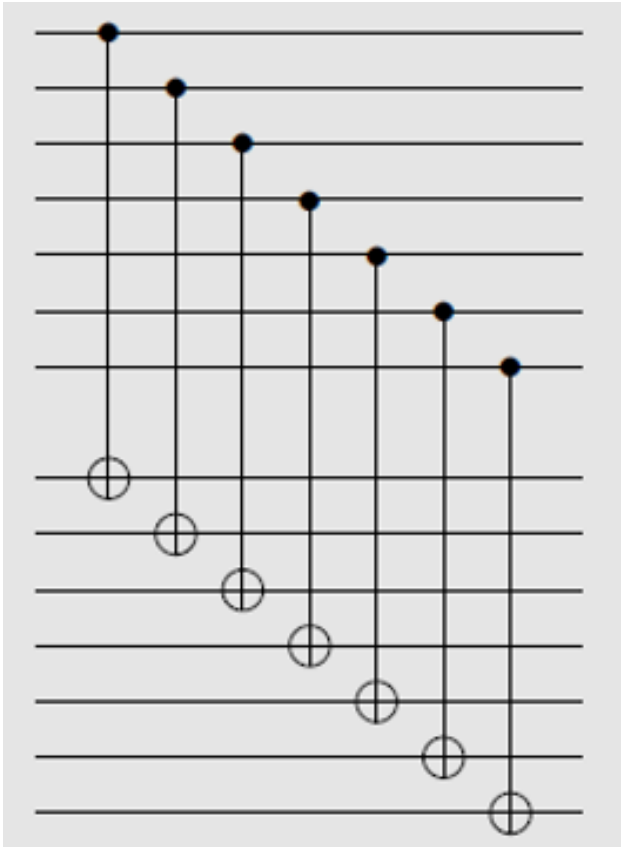

Fig. 8. Transversal CNOT

G. Measurements

Measuring the logical qubits can be done with the help of the ancilla. Measuring a 0 on the ancilla qubit corresponds to an eigenvalue of +1 which means no error occurred. Measuring a 1 on the ancilla corresponds to an eigenvalue of -1 which means an error has occurred.

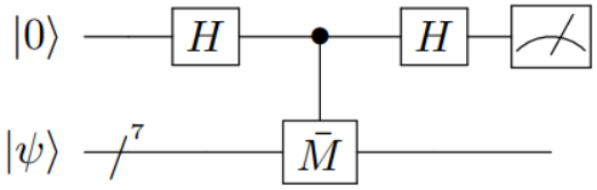

Fig. 9. Logical measurement. M is the basis in which you want to measure.

Once we measure the outcome we are done. Now we will see the results of the encoded circuit for different error levels.

IV. Results

We look at the encoded bell circuit by applied different levels of randomized error to the circuit. We plot the outcomes to observe how the introduction of error at different levels affected our outcome even when the probability of error is the same. The same circuit can be run for different error probabilities and many circuit identities and I am mentioning just one to highlight the difference.

1) No Error: To begin with, we start with no error in the encoded bell circuit. This is important because we want to make sure that even our encoded circuit behavior is similar to what we saw for our unencoded circuit at the very beginning. It should produce the same states with similar probabilities. Creating this circuit and executing it results in the plot below.

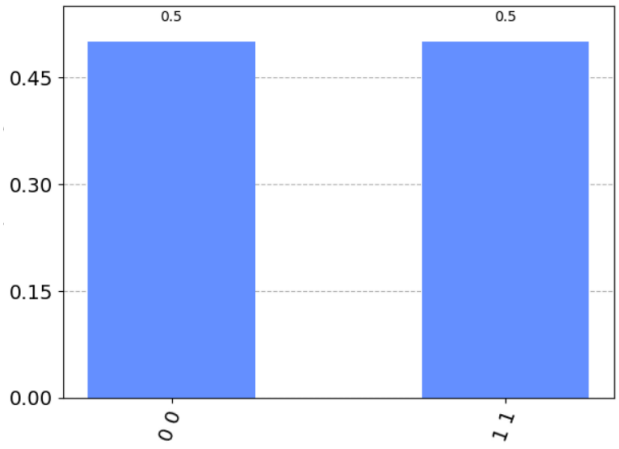

Fig. 10. Resulting states and probabilities for Steane encoded bell circuit

We observe that our encoded circuit still produces the same states proving that our logical operations worked as expected.

2) One error per logical qubit: Now we introduce errors into the circuit. We start with one error per logical qubit. We know that the Steane code can correct up to one error. We introduce this error at two places. First one after the logical hadamard and second one after the logical CNOT. For each error introduced we run steane error correction cycle once for each logical qubit. The steane error correction starts by initializing the ancilla qubits to 0 state. It then runs the steane we discussed in the previous section. Once this is done, it extracts the syndrome of the error correction cycle by measuring the X ancillas and Z ancillas separately. It applies the appropriate corrective gate depending on the value of the syndrome.

There is one more steane error correction added before any operation is applied to the logical qubits. This works towards initializing the quantum error correction cycle and also acts as a placeholder if we want to introduce any state preparation errors into the circuit.

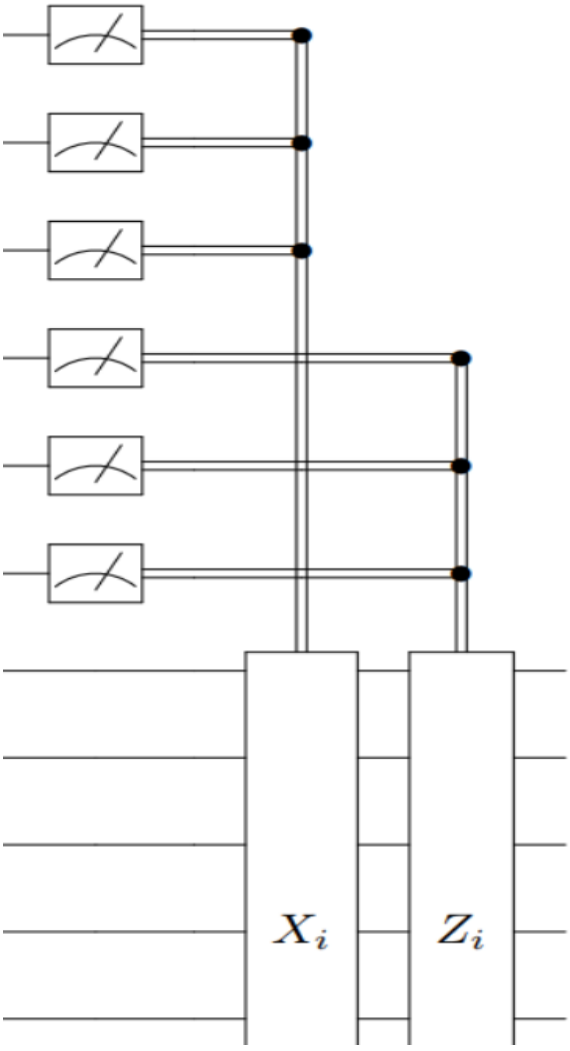

Fig. 11. Measurement dependent corrective gates

Once we run the error correction cycle for both the logical

qubits we again measure the outcomes to check if the error we introduced has been successfully fixed.

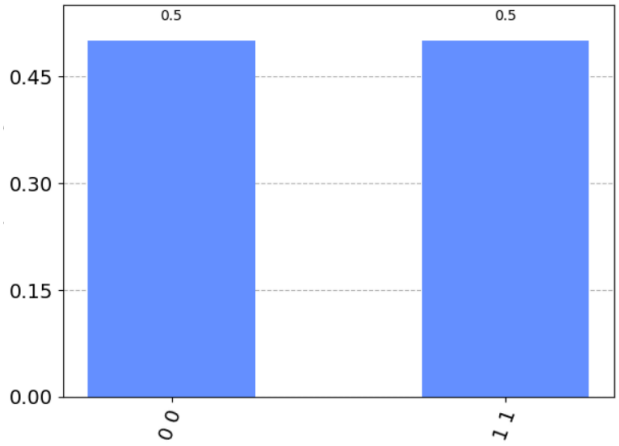

Fig. 12. Resulting states and probabilities for error corrected Steane encoded bell circuit

As we can observe from the histograms in Fig 12., they look quite similar to our encoded histogram from Fig. 10. Thus, we have successfully corrected the randomized error introduced in our logical qubits.

3) One error per physical qubit: Next, we introduce a randomized Pauli error to each physical qubit of the steane encoded block. Again the error is introduced at two places, after the logical Hadamard, and after the logical CNOT. It is important to note that here that when we say physical qubit error we are introducing error only in the data qubits.

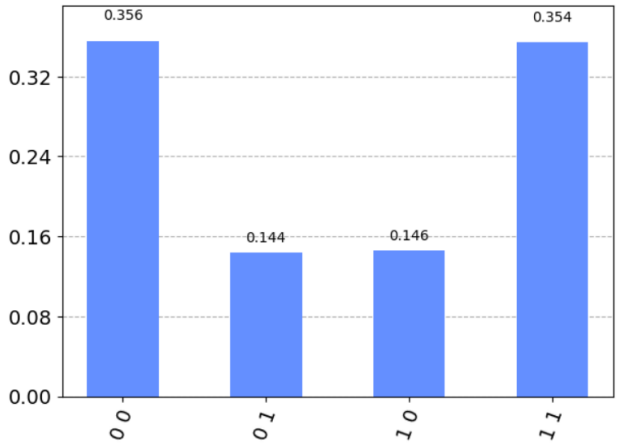

Fig. 13. Resulting states and probabilities for one error per physical qubit of Steance encoded bell circuit

Introducing the error and plotting the outcomes shows that the states are not entirely what we want but are instead spread with some probability of the measuring the wrong states. Steane code cannot successfully fix this level of errors in circuit. It might however fix some of the errors when the syndrome matches the real error. Otherwise, it will either not detect the error or introduce a logical error instead.

V. Discussion and Future work

As we see from the simulation results for the Steane encoded bell circuit that it indeed can fix one random error in a logical qubit successfully but fails beyond that. However, this is just the beginning. There are other error models that need to be simulated. We need to observe the effect of ancilla errors and how to handle them for a Steane circuit. We can also introduce errors during state preparation and measurement and see how it affects the outcomes. Steane code, and quantum error correcting codes in general, show promise but there is still a lot of work to be done.

References

[1] https://github.com/KnightShuffler/Steane-Code-in-Qiskit
[2] The Qiskit Documentation: $https://qiskit.org/documentation/index.html$
[3] Anderson et al, arXiv:2208.01863
[4] Quantum Computation and Quantum Information, 10^th anniversary edition, Michael A. Nielsen & Isaac L. Chuang