ECE 382V: Introduction to Quantum Computing Systems from a Software and Architecture Perspective
Lab 2

Sayam Sethi

November 2023

Contents

1 Introduction

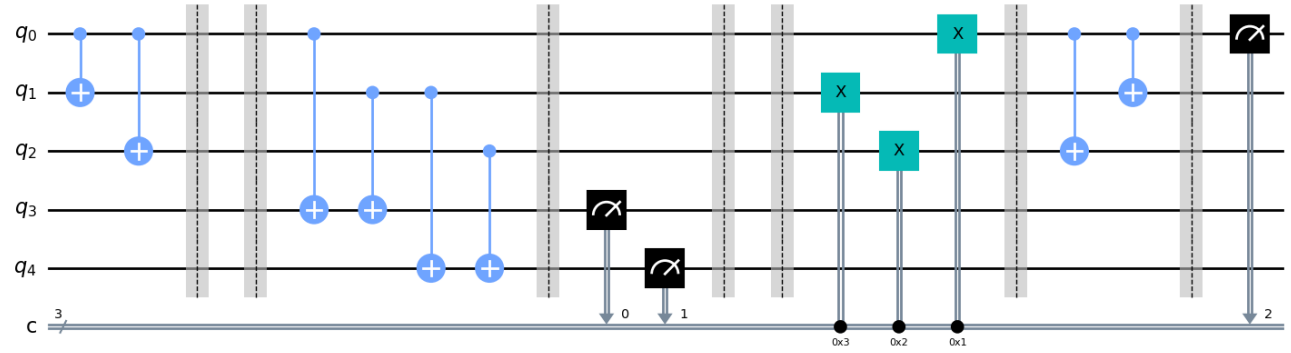
In this lab we implement and understand the different Quantum Error Correction techniques starting from the 3-qubit repetition code and moving onto the distance 3 surface codes. We aim to better comprehend the pros and cons of the different techniques and why they are used/not used anymore.

2 The Repetition Code

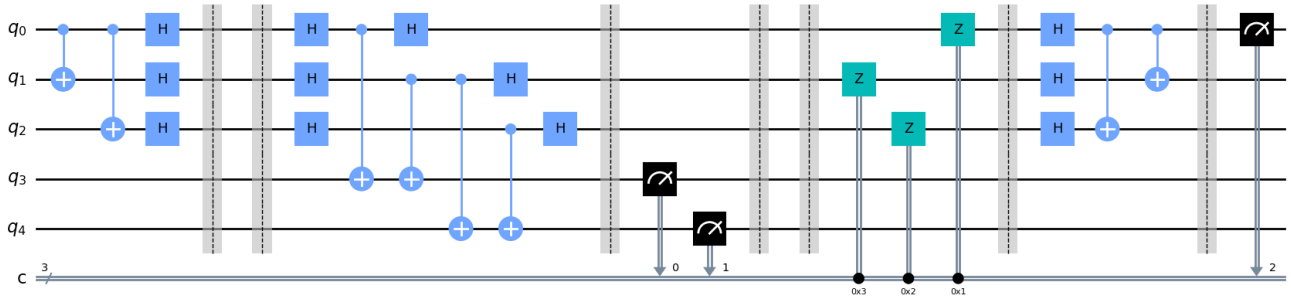
2.1 3-Qubit Repetition

2.1.1 Bit Flip and Phase Flip Errors

The error correction scheme used for handling bit flip and phase flip erorrs is almost the same, the only difference between the two is the logical states that the two qubits are entangled in. The bit-flip entanglement is in the standard basis and hence the logical states are $|000\rangle, |111\rangle$ however, the logical states for the phase flip are $|+++\rangle, |---\rangle$. The two circuits are shown in Figure 1.



(a) The error correction circuit for the bit flip error



(b) The error correction circuit for the phase flip error

Figure 1: The circuits for the bit flip and the phase flip repetition codes. The region between the first two barriers is where the gate operations are done and where the errors can be injected.

The decoder for both the circuits looks exactly the same as can be seen in Figure 1. The only difference is in the decoding applied for the respective value of the decoder, in the case of a bit flip error, the **X** gate is applied and in the case of the phase flip error, the **Z** gate is applied. On running the statevector simulations on this circuit, it can be seen that the errors injected are always corrected successfully, and the result of measuring q_0 is always 0.

2.1.2 Measurement Error

On running the circuits on a noisy simulation (IBM FakeKolkataV2), we notice that the errors are not always successfully corrected. The success rate is around 95% as can be seen from the following plots,

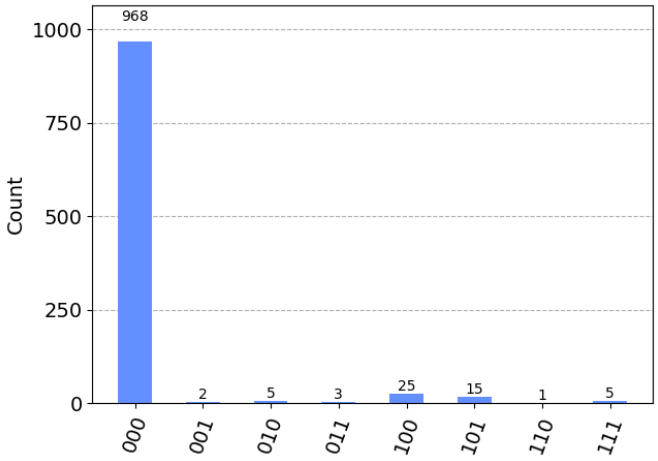
As can be seen from the plots (Figure 2), the decoding success rate decreases when the syndrome values read more 1's. However, the success rate is still quite high.

2.2 5-Qubit Repetition

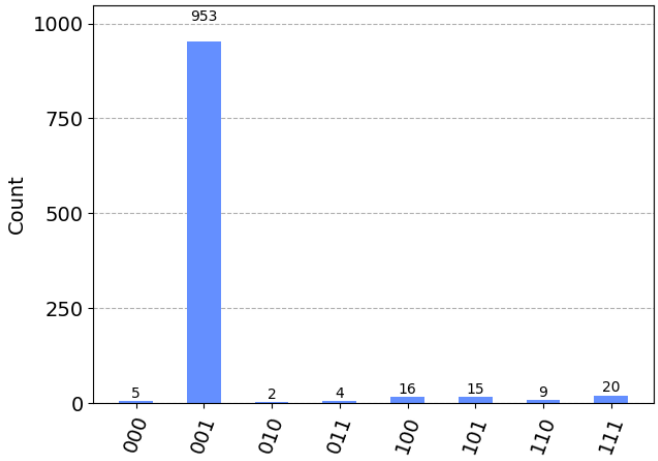
2.2.1 Bit Flip and Phase Flip Errors

The error correction scheme used for handling bit flip and phase flip erorrs is almost the same, the only difference between the two is the logical states that the two qubits are entangled in. The bit-flip entanglement is in the standard basis and hence the logical states are $|00000\rangle, |11111\rangle$ however, the logical states for the phase flip are $|++++\rangle, |----\rangle$. The two circuits are shown in Figure 3.

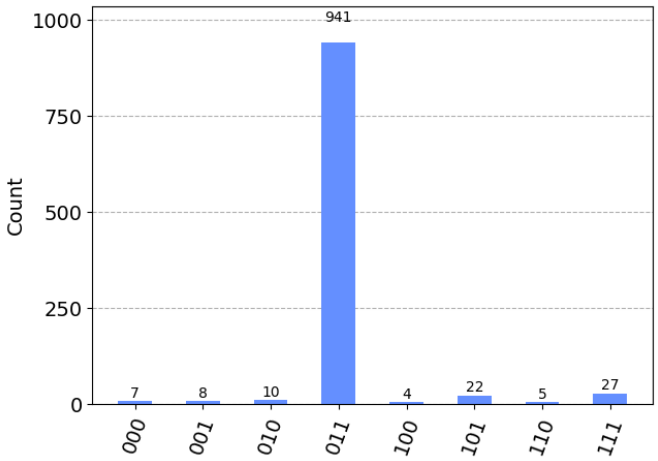
The decoder for both the circuits looks exactly the same as can be seen in Figure 3. The only difference is in the decoding applied for the respective value of the decoder, in the case of a bit flip error, the **X** gate is applied and in the case of the phase flip error, the **Z** gate is applied. On running the statevector



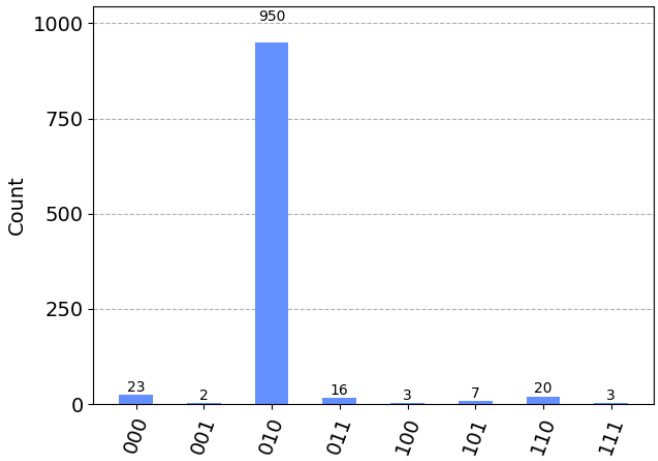
(a) Plot for the measurement with no errors



(b) Plot for the measurement with bit-flip on q_0



(c) Plot for the measurement with bit-flip on q_1



(d) Plot for the measurement with bit-flip on q_2

Figure 2: Plots for the different error introduced manually when running on a noisy simulator.

simulations on this circuit, it can be seen that the errors injected are always corrected successfully, and the result of measuring q_0 is always 0.

2.2.2 Measurement Error

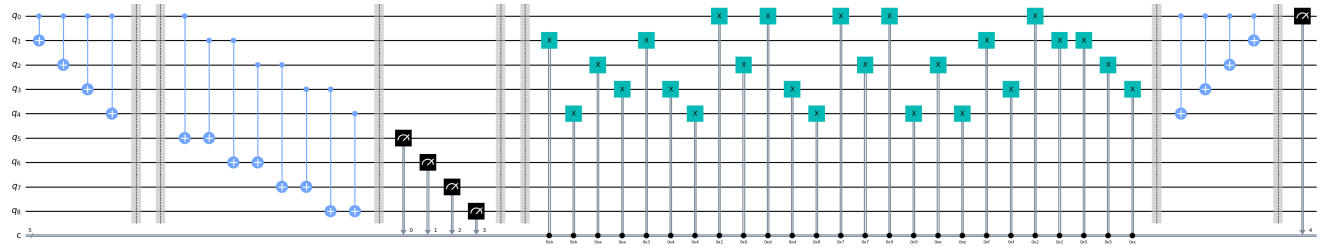
On running the circuits on a noisy simulation (IBM FakeKolkataV2), we notice that the errors are not always successfully corrected. The success rate varies significantly (between 77% and 88%) as can be seen from the following plots,

As can be seen from the plots (Figure 4), the decoding success rate decreases when the syndrome values read more 1's. However, compared to the success rate of the 3 qubit repetition code, the success rate is much lower due to the increased gate errors.

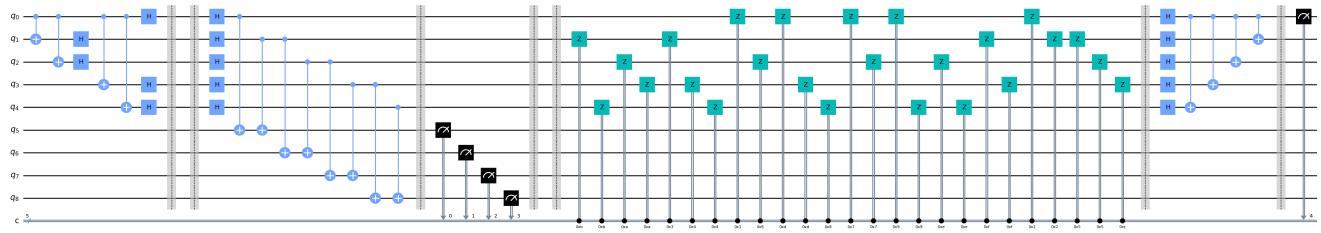
3 Surface Codes

3.1 Circuit Creation

The rotated surface code for distance three was implemented using the circuit show in Figure 5. Two rounds of syndrome extraction are done. The first one is used to entangle the data qubits and to obtain



(a) The error correction circuit for the bit flip error



(b) The error correction circuit for the phase flip error

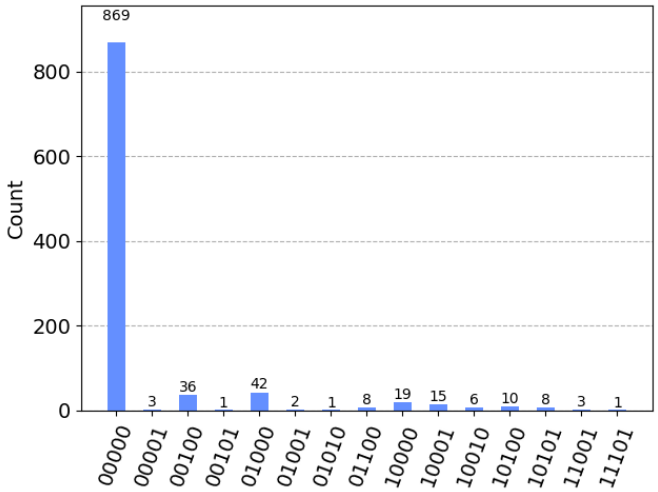
Figure 3: The circuits for the bit flip and the phase flip repetition codes. The region between the first two barriers is where the gate operations are done and where the errors can be injected.

the reference syndrome. The second round is used to compare the newly obtained syndrome with the reference syndrome. The `xor` between the two syndromes can then be used to determine the correction to be applied by using it as the input to the decoder. Finally, the correction is applied (not shown in the figure) and another round of syndrome extraction is done to see if the syndrome has been corrected (again, not shown in the figure). This surface code implementation is capable of correcting up to one bit-flip and one phase-flip error simultaneously on any of the data qubits.

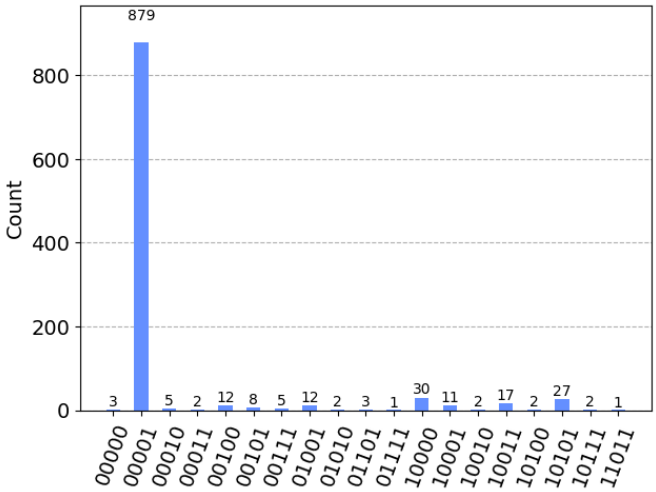
3.2 Decoder Implementation

A static decoder was implemented by using the `xor` values between the two syndrome results as the input to the decoder. If the decoder had already been mapped to some error, the error was updated. Since we only want to correct up to one bit flip and one phase flip errors, we can have at most 100 errors (10 possibilities for each: error on one of the 9 data qubits + no error). However, since we only have 8 bits for the syndrome, we can only store the decoding for up to 64 errors, thus overwriting 36 decoding. The decoding is shown below,

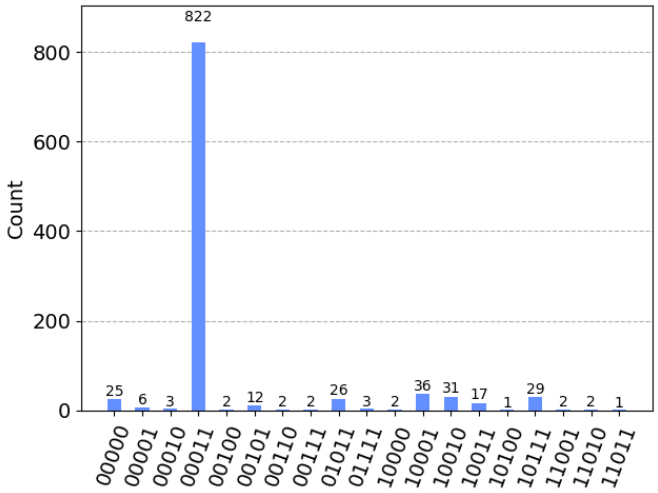
```
0 (frozenset({None}), frozenset({None}))
1 (frozenset({0}), frozenset({None}))
2 (frozenset({2}), frozenset({None}))
3 (frozenset({1}), frozenset({None}))
4 (frozenset({5}), frozenset({None}))
5 (frozenset({4}), frozenset({None}))
8 (frozenset({6}), frozenset({None}))
12 (frozenset({7}), frozenset({None}))
16 (frozenset({None}), frozenset({1}))
17 (frozenset({0}), frozenset({1}))
18 (frozenset({2}), frozenset({1}))
19 (frozenset({1}), frozenset({1}))
20 (frozenset({5}), frozenset({1}))
21 (frozenset({4}), frozenset({1}))
```



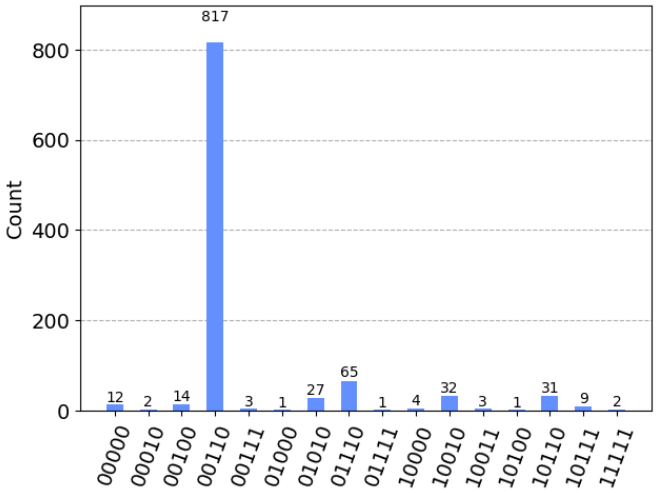
(a) Plot for the measurement with no errors



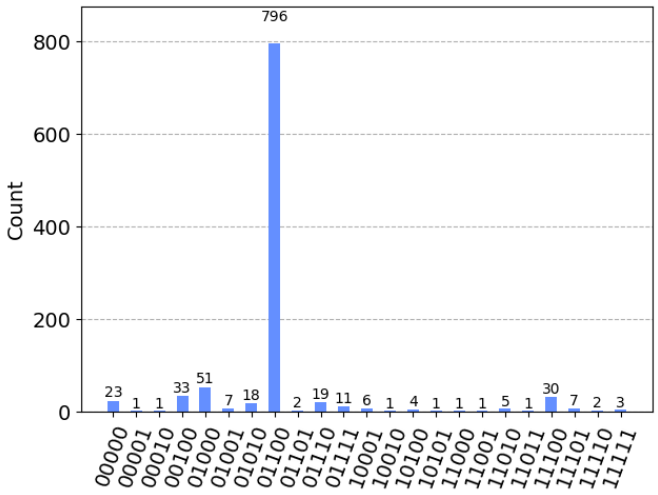
(b) Plot for the measurement with bit-flip on q_0



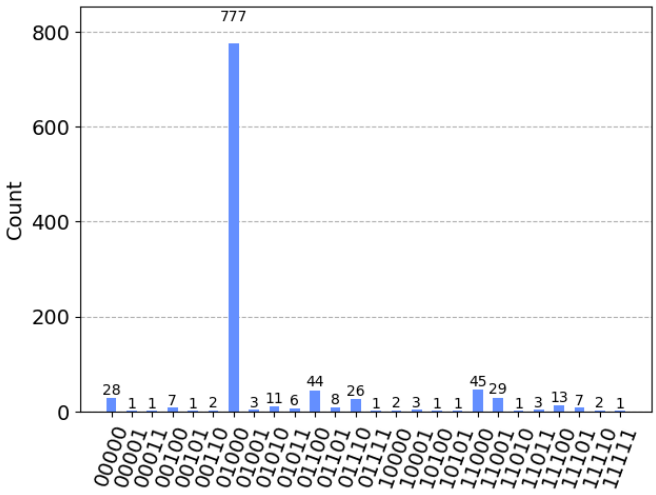
(c) Plot for the measurement with bit-flip on q_1



(d) Plot for the measurement with bit-flip on q_2



(e) Plot for the measurement with bit-flip on q_3



(f) Plot for the measurement with bit-flip on q_5

Figure 4: Plots for the different error introduced manually when running on a noisy simulator.

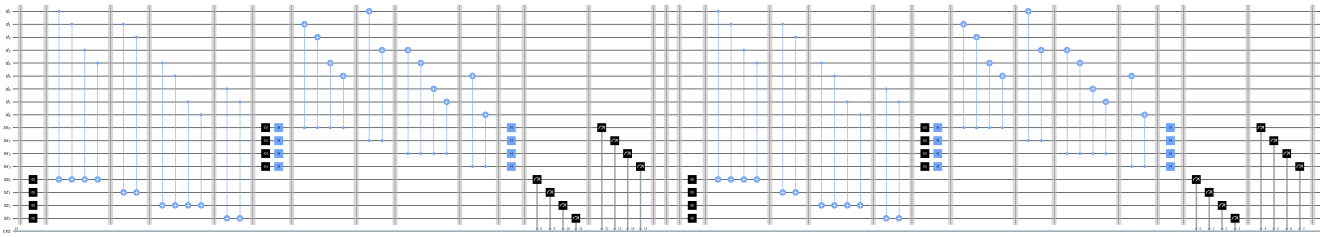


Figure 5: The circuit for the surface code. The region between the two syndrome extraction regions is where the gate operations are done and where the errors can be injected.

```
24 (frozenset({6}), frozenset({1}))
28 (frozenset({7}), frozenset({1}))
32 (frozenset({None}), frozenset({0}))
33 (frozenset({0}), frozenset({0}))
34 (frozenset({2}), frozenset({0}))
35 (frozenset({1}), frozenset({0}))
36 (frozenset({5}), frozenset({0}))
37 (frozenset({4}), frozenset({0}))
40 (frozenset({6}), frozenset({0}))
44 (frozenset({7}), frozenset({0}))
64 (frozenset({None}), frozenset({6}))
65 (frozenset({0}), frozenset({6}))
66 (frozenset({2}), frozenset({6}))
67 (frozenset({1}), frozenset({6}))
68 (frozenset({5}), frozenset({6}))
69 (frozenset({4}), frozenset({6}))
72 (frozenset({6}), frozenset({6}))
76 (frozenset({7}), frozenset({6}))
80 (frozenset({None}), frozenset({4}))
81 (frozenset({0}), frozenset({4}))
82 (frozenset({2}), frozenset({4}))
83 (frozenset({1}), frozenset({4}))
84 (frozenset({5}), frozenset({4}))
85 (frozenset({4}), frozenset({4}))
88 (frozenset({6}), frozenset({4}))
92 (frozenset({7}), frozenset({4}))
96 (frozenset({None}), frozenset({3}))
97 (frozenset({0}), frozenset({3}))
98 (frozenset({2}), frozenset({3}))
99 (frozenset({1}), frozenset({3}))
100 (frozenset({5}), frozenset({3}))
101 (frozenset({4}), frozenset({3}))
104 (frozenset({6}), frozenset({3}))
108 (frozenset({7}), frozenset({3}))
128 (frozenset({None}), frozenset({8}))
129 (frozenset({0}), frozenset({8}))
130 (frozenset({2}), frozenset({8}))
131 (frozenset({1}), frozenset({8}))
132 (frozenset({5}), frozenset({8}))
```

```
133 (frozenset({4}), frozenset({8}))
136 (frozenset({6}), frozenset({8}))
140 (frozenset({7}), frozenset({8}))
144 (frozenset({None}), frozenset({5}))
145 (frozenset({0}), frozenset({5}))
146 (frozenset({2}), frozenset({5}))
147 (frozenset({1}), frozenset({5}))
148 (frozenset({5}), frozenset({5}))
149 (frozenset({4}), frozenset({5}))
152 (frozenset({6}), frozenset({5}))
156 (frozenset({7}), frozenset({5}))
```

3.3 Error Correction

Now that we have the decoder, the error correction circuit is designed by performing controlled CNOT based on all possible initial syndromes and their `xor` values. Since this leads to a very large circuit, the circuit could not be shown. However, on running the circuit on all possible 100 error scenarios, it was observed that the syndrome is restored to what we started off with. This happens even when the error correction is different from the one that was actually injected. The reason for this is that the correction applied actually inserts even more errors and thus the syndrome is restored, however, the resultant circuit is actually erroneous. However, since this only happens on the boundary qubits, this can be corrected if we insert more syndrome qubits on the boundaries. However, since this wasn't the goal of the lab, this wasn't done.

3.4 Measurement Error

Since the circuit obtained after implementing the error correction was very big, it could not be run on the noisy simulator in finite time. Therefore, to simulate the measurement errors, the correction circuit was instead modified to read a different syndrome than what it should have been. This was done by changing the `xor` values between the two syndromes. The results are shown below,

```
Measure error on qubit None
[None] [None] 11010000 11010000 11010000
[None] [1] 11100000 11110000 11100000
[None] [2] 00100000 00110000 00100000
[None] [0] 11010000 11110000 11010000
[1] [None] 00010000 00010011 00010000
[1] [1] 11000000 11010011 11000000
[1] [2] 00000000 00010011 00000000
[1] [0] 01010000 01110011 01010000
[2] [None] 10100000 10100010 10100000
[2] [1] 11000000 11010010 11000000
[2] [2] 10100000 10110010 10100000
[2] [0] 01000000 01100010 01000000
[0] [None] 11100000 11100001 11100000
[0] [1] 11110000 11100001 11110000
[0] [2] 11010000 11000001 11010000
[0] [0] 00100000 00000001 00100000

Measure error on qubit 0
[None] [None] 10000000 10000000 10000001
```

```
[None] [1] 00000000 00010000 00000001
[None] [2] 00110000 00100000 00110001
[None] [0] 00000000 00100000 00000001
[1] [None] 01010000 01010011 01010001
[1] [1] 01100000 01110011 01100001
[1] [2] 01100000 01110011 01100001
[1] [0] 11000000 11100011 11000001
[2] [None] 00110000 00110010 00110001
[2] [1] 01100000 01110010 01100001
[2] [2] 01010000 01000010 01010001
[2] [0] 01110000 01010010 01110001
[0] [None] 00000000 00000001 00000001
[0] [1] 11000000 11010001 11000001
[0] [2] 00000000 00010001 00000001
[0] [0] 00110000 00010001 00110001

Measure error on qubit 1
[None] [None] 11110000 11110000 11110010
[None] [1] 10000000 10010000 10000010
[None] [2] 11110000 11100000 11110010
[None] [0] 01000000 01100000 01000010
[1] [None] 11110000 11110011 11110010
[1] [1] 10000000 10010011 10000010
[1] [2] 01110000 01100011 01110010
[1] [0] 11100000 11000011 11100010
[2] [None] 10000000 10000010 10000010
[2] [1] 11100000 11110010 11100010
[2] [2] 00000000 00010010 00000010
[2] [0] 10110000 10010010 10110010
[0] [None] 00110000 00110001 00110010
[0] [1] 00100000 00110001 00100010
[0] [2] 01110000 01100001 01110010
[0] [0] 01100000 01000001 01100010

Measure error on qubit 2
[None] [None] 10100000 10100000 10100100
[None] [1] 10100000 10110000 10100100
[None] [2] 00010000 00000000 00010100
[None] [0] 00010000 00110000 00010100
[1] [None] 11000000 11000011 11000011
[1] [1] 10010000 10000011 10000011
[1] [2] 11110000 11100011 11100011
[1] [0] 10010000 10110011 10110011
[2] [None] 11100000 11100010 11100010
[2] [1] 01100000 01110010 01110010
[2] [2] 00000000 00010010 00010010
[2] [0] 10000000 10100010 10100010
[0] [None] 11000000 11000001 11000100
[0] [1] 10010000 10000001 10010100
[0] [2] 11000000 11010001 11000100
```

```
[0] [0] 00110000 00010001 00110100

Measure error on qubit 3
[None] [None] 10110000 10110000 10111000
[None] [1] 11010000 11000000 11011000
[None] [2] 01000000 01010000 01001000
[None] [0] 00010000 00110000 00011000
[1] [None] 00100000 00100011 00100011
[1] [1] 10110000 10100011 10100011
[1] [2] 11100000 11110011 11110011
[1] [0] 01010000 01110011 01110011
[2] [None] 11100000 11100010 11100010
[2] [1] 11100000 11110010 11110010
[2] [2] 01100000 01110010 01110010
[2] [0] 10110000 10010010 10010010
[0] [None] 11010000 11010001 11010001
[0] [1] 10110000 10100001 10100001
[0] [2] 11110000 11100001 11100001
[0] [0] 00110000 00010001 00010001

Measure error on qubit 4
[None] [None] 10110000 10110000 10100000
[None] [1] 01110000 01100000 01100000
[None] [2] 10000000 10010000 10010000
[None] [0] 00110000 00010000 00010000
[1] [None] 01010000 01010011 01000000
[1] [1] 10100000 10110011 10110000
[1] [2] 10000000 10010011 10010000
[1] [0] 10000000 10100011 10100011
[2] [None] 11000000 11000010 11010000
[2] [1] 01010000 01000010 01000000
[2] [2] 10110000 10100010 10100000
[2] [0] 01110000 01010010 01010010
[0] [None] 10110000 10110001 10100000
[0] [1] 01000000 01010001 01010000
[0] [2] 11000000 11010001 11010000
[0] [0] 00100000 00000001 00000001

Measure error on qubit 5
[None] [None] 10110000 10110000 10010000
[None] [1] 00000000 00010000 00010000
[None] [2] 10010000 10000000 10000000
[None] [0] 00110000 00010000 00010000
[1] [None] 11100000 11100011 11000000
[1] [1] 10100000 10110011 10110011
[1] [2] 01010000 01000011 01000011
[1] [0] 01110000 01010011 01010000
[2] [None] 01110000 01110010 01010000
[2] [1] 01010000 01000010 01000010
[2] [2] 10010000 10000010 10000010
```

```
[2] [0] 00010000 00110010 00110000
[0] [None] 01100000 01100001 01000000
[0] [1] 01110000 01100001 01100001
[0] [2] 11100000 11110001 11110001
[0] [0] 00110000 00010001 00010000

Measure error on qubit 6
[None] [None] 10100000 10100000 11100000
[None] [1] 10000000 10010000 11000000
[None] [2] 10110000 10100000 11110000
[None] [0] 01100000 01000000 00100000
[1] [None] 10100000 10100011 11100000
[1] [1] 11110000 11100011 10110000
[1] [2] 01000000 01010011 00000000
[1] [0] 01000000 01100011 00000000
[2] [None] 10010000 10010010 11010000
[2] [1] 00000000 00010010 01000000
[2] [2] 01110000 01100010 00110000
[2] [0] 10010000 10110010 11010000
[0] [None] 11100000 11100001 10100000
[0] [1] 00100000 00110001 01100000
[0] [2] 11100000 11110001 10100000
[0] [0] 01000000 01100001 00000000

Measure error on qubit 7
[None] [None] 00100000 00100000 10100000
[None] [1] 01110000 01100000 11110000
[None] [2] 11000000 11010000 01000000
[None] [0] 00110000 00010000 00010000
[1] [None] 11000000 11000011 01000000
[1] [1] 00110000 00100011 10110000
[1] [2] 00100000 00110011 10100000
[1] [0] 00100000 00000011 00000011
[2] [None] 00000000 00000010 10000000
[2] [1] 01000000 01010010 11000000
[2] [2] 00000000 00010010 10000000
[2] [0] 01000000 01100010 01100010
[0] [None] 10010000 10010001 00010000
[0] [1] 00100000 00110001 10100000
[0] [2] 01110000 01100001 11110000
[0] [0] 01100000 01000001 01000001
```

The above output log can be read as, the first list is the bit flip error injected, the second list is the phase flip error injected, the first 8-bit input syndrome is the reference syndrome, the next 8 bit input is the syndrome obtained after the error was injected (without the measurement error), the last 8 bits are the syndrome obtained after the error correction was applied. It can be observed that the measurement error remains in the final syndrome for most of the corrections. It propagates even further when the incorrect correction was applied.

4 Conclusion

It can be seen that the error correction techniques are very helpful in mitigating the errors that occur because of noisy simulations. However, they are still tedious to implement and we still have a long way to go before we can implement successful error correction schemes in a real quantum system.