QC Architecture and Electronics (CESE4080) Homework 2

Tongyun Yang (5651794, tongyunyang@tudelft.nl)

2023.03.07

1 Exercise 1 Shor Code

1.1 Question

Implement the Shor code circuit in QX. Do not use parallel sections! Write the different parts of the circuits into separate sub-circuits. Add a display instruction at the end of each sub-circuit.

Answer: The implementation is shown in the code snippet below. Note that a .prepare sub-circuit is added to initialize all states of the qubits to $|0\rangle$.

To perform bit-flip, X gate applied; To perform phase-flip, Z gate applied.

```
Version 1.0

qubits 9

.prepare
    prep_z q[0:8]
    display

.init
    rx q[0], 1.3
    ry q[0], 0.3
    display

.phaseflip_encode
    CNOT q[0], q[3]
    CNOT q[0], q[6]
    H q[0]
    H q[3]
    H q[6]
    display

.bitflip_encode
    CNOT q[0], q[1]
    CNOT q[3], q[4]
    CNOT q[6], q[7]
    CNOT q[0], q[2]
    CNOT q[3], q[5]
    CNOT q[6], q[8]
    display

.bitflip_decode_and_correct
    CNOT q[0], q[1]
    CNOT q[3], q[4]
    CNOT q[6], q[7]
    CNOT q[0], q[2]
    CNOT q[3], q[5]
    CNOT q[6], q[8]
    Toffoli q[2], q[1], q[0]
    Toffoli q[5], q[4], q[3]
    Toffoli q[8], q[7], q[6]
    display

.phaseflip_decode_and_correct
    H q[0]
    H q[3]
    H q[6]
    CNOT q[0], q[3]
    CNOT q[0], q[6]
    Toffoli q[6], q[3], q[0]
    display
```

After the .init sub-circuit, the state of q[0] is shown in figure 1. It is observed that there is around 62.78% chance of measuring $|0\rangle$, and around 37.22% chance of measuring $|0\rangle$.

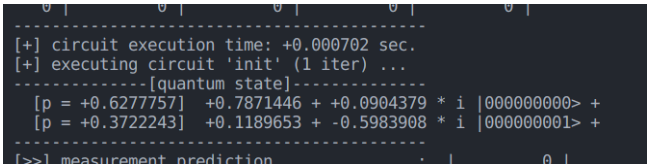


Figure 1: State of q[0] after .init

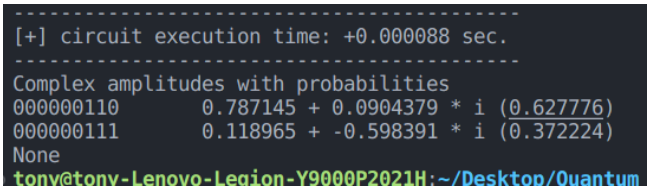


Figure 2: Final state of q[0]

1.2 Question

Modify the inject_error section to inject one bit flip on qubit q_0. Run the circuit. Check if the final state of the first qubit (after correction and decoding) is the same as its initial state. What do you observe?

Answer: After the modification, the .inject_error sub-circuit is shown in the code snippet below.

```
.inject_error
    X q[0]
    display
```

Running the circuit and checking the final state of q[0], as shown in figure 2. It is observed that the final state is exaclty the same as the initial state (both bit-wise and phase-wise).

1.3 Question

Modify the inject_error section. Save the modified circuit into a file named shor_9q_code_2.qc. Run the circuit with these errors and check if the final state of the first qubit (after correction and decoding) is the same as its initial state. What do you observe? Why?

Answer: After the modification, the .inject_error sub-circuit is shown in the code snippet below.

```
.inject_error
    X q[0]
    Z q[0]
    X q[1]
    display
```

The initial state and the final state is shown in figure 5a and figure 5b. The init state is different from the final state. Though the possibility of measuring $|0\rangle$ and $|1\rangle$ remains unchanged under two situations, the phase of the final state is flipped.

Shor code is noted as [9, 1, 3], meaning that there are 9 physical qubits, 1 logical qubits and the structure of code distance 3, hence at most one error could occur.

The official reason of this situation is due to code distance, however, I did some analysis myself, and was hoping that it could get reviewed as well.

The reasons of this situation are as follows:

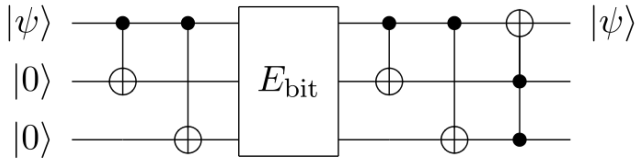
In a word, among (q0,q1,q2) (q3,q4,q5) (q6,q7,q8) all three "groups", one bit-flip could occur on each, and one phase-flip could occur in total. Two errors also must not happen on the same qubit. Analysis in detail are given below.

Each group consists of three "Bit flip code" as figure 3a and together forming a "Sign flip code" as figure 3b, and in each of these "code" only one error on one qubit could occur at a time. (The proof of this will not be given since it is very simple.) Then according to the functionality of the Hadamard gate, it basically turns bit-flips into sign-flips and vice versa.

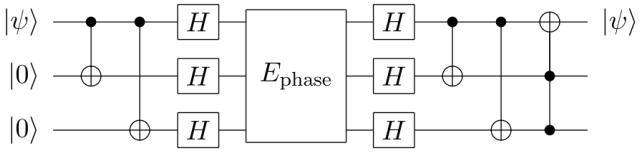
If a bit-flip and phase-flip occur at the same time on different qubits (in each group), they would be transformed into a phase-flip and a bit-flip. Due to the fact that "Bit flip code" ignores phase-flip, it could still function well. However, if a bit-flip and a phase-flip occur on the same qubit, things would go wrong. Because of the "Bit flip code" ignoring the phase, the correction of the bit-flip would apply an X gate to the flipped phase leading to an "exchange of amplitudes", and when the "Sign flip code" applies the correction, the "wrong" phase is flipped (e.g. if it was meant to correct the phase-flip on $|1\rangle$, it is now flipping the phase on $|0\rangle$), leading a result that amplitudes on both $|0\rangle$ and $|1\rangle$ to be flipped.

1.4 Question

To trace the curve, we execute the circuit 10 times using each of the following error probabilities: 0.001, 0.005, 0.01, 0.05 and 0.1. Note the number of logical bit-flip errors for each case and trace the



(a) Bit-flip code



(b) Phase-flip code

Figure 3: Bit-flip code VS. Phase-flip code

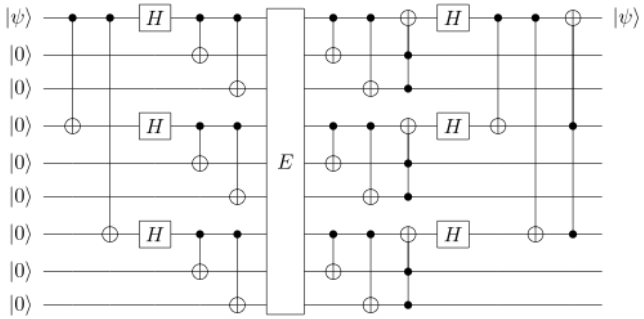
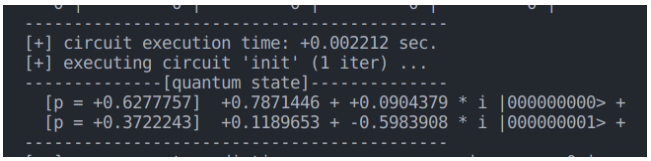
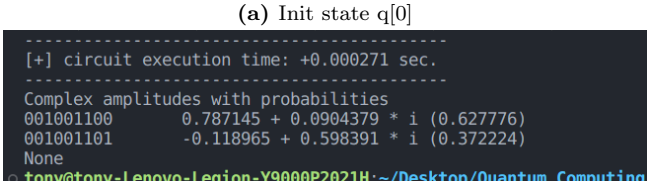


Figure 4: Shor code



(a) Init state q[0]



(b) Final state q[0]

Figure 5: Init state VS. Final state

curve of the physical error probability vs. logical bit flip errors. Provide the result as a graph.

Answer: The code used for testing physical error probability is shown in the code snippet below. It is also provided in the .zip file with the name "shor_9q_code_3.qc".

```
Version 1.0

qubits 9

.prepare
    prep_z q[0:8]
    display

.init
    X q[0]
    display

.phaseflip_encode
    CNOT q[0], q[3]
    CNOT q[0], q[6]
    H q[0]
    H q[3]
    H q[6]
    display

.bitflip_encode
    CNOT q[0], q[1]
    CNOT q[3], q[4]
    CNOT q[6], q[7]
    CNOT q[0], q[2]
    CNOT q[3], q[5]
    CNOT q[6], q[8]
    display

.bitflip_decode_and_correct
    CNOT q[0], q[1]
    CNOT q[3], q[4]
    CNOT q[6], q[7]
    CNOT q[0], q[2]
    CNOT q[3], q[5]
    CNOT q[6], q[8]
    Toffoli q[2], q[1], q[0]
    Toffoli q[5], q[4], q[3]
    Toffoli q[8], q[7], q[6]
    display

.phaseflip_decode_and_correct
    H q[0]
    H q[3]
    H q[6]
    CNOT q[0], q[3]
    CNOT q[0], q[6]
    Toffoli q[6], q[3], q[0]
    display

error_model depolarizing_channel, 0.1
```

The table of data is presented in the .zip file called "Error-Test.txt", and the graph is provided below. As a logical bit-flip is defined "given an input state $|1\rangle$ encoded into a 9-qubit logical state, when a logical bit-flip error happens, we obtain a $|0\rangle$ output state at the end of the circuit after decoding.", hence, in this case, we only focus on the state of qubit 0. The graph presenting the data is as shown in figure 6. It is observed that as the probability of physical errors rise, the probability of logical errors occurring rises as well.

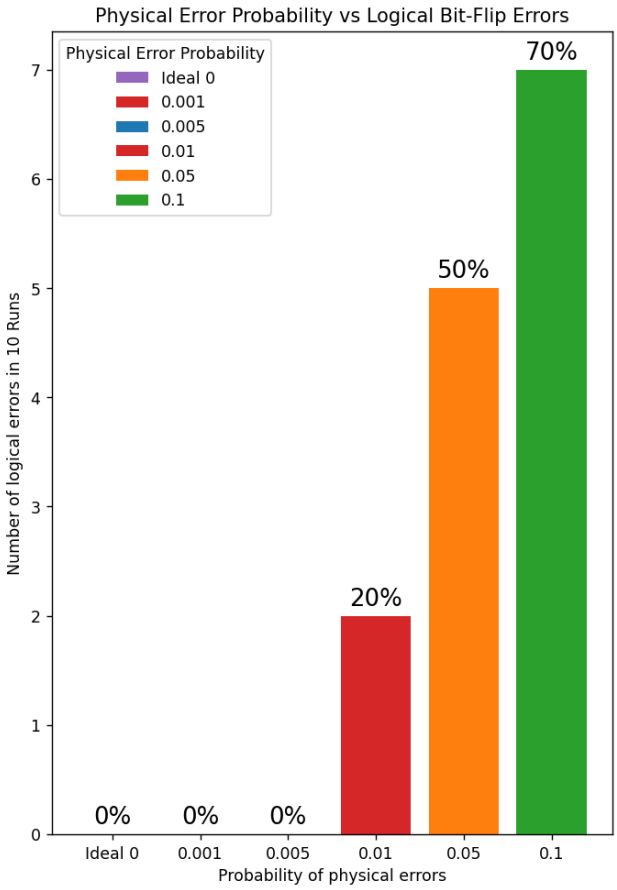


Figure 6: Bar chart: Physical Error Probability VS. Logical Bit-Flip Errors

2 Exercise 2 Steane code

2.1 Question

Write the Encoding sub-circuits. We encode qubit q_6 in QX into the $|0\rangle$ logical state using the circuit Steane code. Implement the circuit and Display the state after encoding.

Answer: The implementation is shown in the code snippet below. Note that a .prepare sub-circuit is added to initialize all states of the qubits to $|0\rangle$.

```
Version 1.0

qubits 7

.prepare
    prep_z q[0:6]
    display

.encode
    H q[0]
    H q[1]
    H q[2]
    CNOT q[6], q[5]
    CNOT q[6], q[4]
    CNOT q[0], q[3]
    CNOT q[0], q[5]
    CNOT q[0], q[6]
    CNOT q[1], q[3]
    CNOT q[1], q[4]
    CNOT q[1], q[6]
    CNOT q[2], q[3]
    CNOT q[2], q[4]
    CNOT q[2], q[5]
    display
```

The final state is shown in figure 7. This is the $|0\rangle$ logic qubit.

2.2 Question

Now we will build the error detection circuit knowing that the following stabilizer generators are used in Steane code to detect bit-flip and phase-flip errors: Write the error detection circuit based on such stabilisers and display the error syndromes (no errors injected). Use as many ancillas as stabilisers

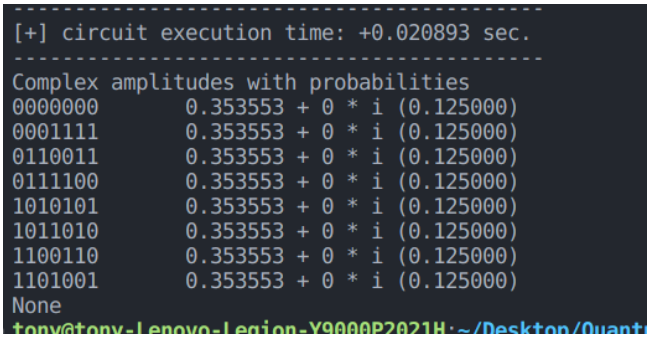


Figure 7: Final state of .encode sub-circuit

and respect the order of the stabilisers (the bottom most ancilla of your circuit. i.e. the qubits with highest number, should measure g1 and so on). Name this sub-circuit .detect_error.

Answer: The implementation is shown in the code snippet below.

```
Version 0.5

qubits 13

.prepare
    prep_z q[0:12]
    display

.encode
    H q[0]
    H q[1]
    H q[2]
    CNOT q[6], q[5]
    CNOT q[6], q[4]
    CNOT q[0], q[3]
    CNOT q[0], q[5]
    CNOT q[0], q[6]
    CNOT q[1], q[3]
    CNOT q[1], q[4]
    CNOT q[1], q[6]
    CNOT q[2], q[3]
    CNOT q[2], q[4]
    CNOT q[2], q[5]
    display

.detect_error
    CNOT q[0], q[12]
    CNOT q[1], q[12]
    CNOT q[2], q[12]
    CNOT q[3], q[12]

    CNOT q[0], q[11]
    CNOT q[1], q[11]
    CNOT q[4], q[11]
    CNOT q[5], q[11]

    CNOT q[0], q[10]
    CNOT q[2], q[10]
    CNOT q[4], q[10]
    CNOT q[6], q[10]
    display

    H q[7]
    CNOT q[7], q[0]
    CNOT q[7], q[1]
    CNOT q[7], q[2]
    CNOT q[7], q[3]
    H q[7]

    H q[8]
    CNOT q[8], q[0]
    CNOT q[8], q[1]
    CNOT q[8], q[4]
    CNOT q[8], q[5]
    H q[8]

    H q[9]
    CNOT q[9], q[0]
    CNOT q[9], q[2]
    CNOT q[9], q[4]
    CNOT q[9], q[6]
    H q[9]
    display

    measure q[7]
    measure q[8]
    measure q[9]
    measure q[10]
    measure q[11]
    measure q[12]
```

Qubit 7 detects phase-flip on qubit 6, 4, 2 and 0. Qubit 8 detects phase-flip on qubit 0, 1, 4 and 5. Qubit 9 detects phase-flip on qubit 0, 1, 2 and 3.

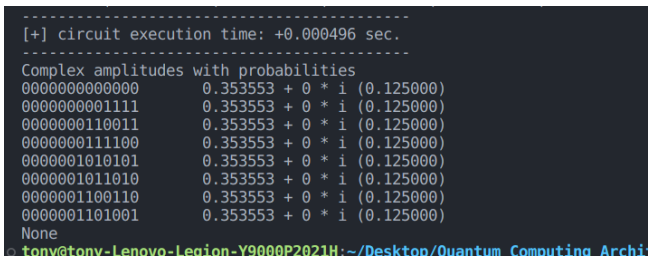


Figure 8: Syndromes when no error

	[g1, g2, g3, g4, g5, g6]
No error	[0, 0, 0, 0, 0, 0]
q0 bit-flip	[1, 1, 1, 0, 0, 0]
q1 bit-flip	[1, 1, 0, 0, 0, 0]
q2 bit-flip	[1, 0, 1, 0, 0, 0]
q3 bit-flip	[1, 0, 0, 0, 0, 0]
q4 bit-flip	[0, 1, 1, 0, 0, 0]
q5 bit-flip	[0, 1, 0, 0, 0, 0]
q6 bit-flip	[0, 0, 1, 0, 0, 0]
q0 phase-flip	[0, 0, 0, 1, 1, 1]
q1 phase-flip	[0, 0, 0, 0, 1, 1]
q2 phase-flip	[0, 0, 0, 1, 0, 1]
q3 phase-flip	[0, 0, 0, 0, 0, 1]
q4 phase-flip	[0, 0, 0, 1, 1, 0]
q5 phase-flip	[0, 0, 0, 0, 1, 0]
q6 phase-flip	[0, 0, 0, 1, 0, 0]

Table 1: Error Syndromes

Qubit 12, 11 and 10 detects bit-flip on qubits mentioned above in the same order. (12 corresponding to 9, 11 to 8, and 10 to 9)

The error syndromes are displayed as shown in figure 8. The syndrome is $|000_000\rangle$ when no errors are occurring.

2.3 Question

Next, we are going to inject some errors in our system (after encoding and before detection) and see how the error syndromes change. Name this sub-circuit .inject_error. Inject single bit-flip (phase-flip) errors and write down the error syndromes.

Answer: The errors are as shown in table 1.

```
[+] circuit execution time: +0.000419 sec.
-------------------------------------------
Complex amplitudes with probabilities
00000000010110       0.353553 + 0 * i (0.125000)
00000000011001       0.353553 + 0 * i (0.125000)
00000000100101       0.353553 + 0 * i (0.125000)
00000000101010       0.353553 + 0 * i (0.125000)
00000001000011       0.353553 + 0 * i (0.125000)
00000001001100       0.353553 + 0 * i (0.125000)
00000001110000       0.353553 + 0 * i (0.125000)
00000001111111       0.353553 + 0 * i (0.125000)
None
```

Figure 9: Final state of top qubit (encode circuit A)

3 Exercise 3 Fault-tolerant Bell pair circuit

3.1 Question

Follow the process, implement the encoding process. Name these sub-circuits .encode_qB and .encode_qA. Display the two logical states (by only doing either encode A or encode B).

Answer: Figure 9 shows the the final state of the top qubit, it is a logical $|1\rangle$. Figure 10 shows the final state of the bottom qubit, it is a logical $|0\rangle$.

The implementation is as shown in the code snippet below.

```
Version 1.0

qubits 14

.prepare
    prep_z q[0:13]
    map q[0], q0
    map q[1], q1
    map q[2], q2
    map q[3], q3
    map q[4], q4
    map q[5], q5
    map q[6], q6
    map q[7], q7
    map q[8], q8
    map q[9], q9
    map q[10], q10
    map q[11], q11
    map q[12], q12
    map q[13], q13


.encode_qA
    H q0
    H q1
    H q2
    CNOT q6, q5
    CNOT q6, q4
    CNOT q0, q3
    CNOT q0, q5
    CNOT q0, q6
    CNOT q1, q3
    CNOT q1, q4
    CNOT q1, q6
    CNOT q2, q3
    CNOT q2, q4
    CNOT q2, q5
    X q0
    X q1
    X q2
    X q3
    X q4
    X q5
    X q6
    display

.encode_qB
    H q7
    H q8
    H q9
    CNOT q13, q12
    CNOT q13, q11
    CNOT q7, q10
    CNOT q7, q12
    CNOT q7, q13
    CNOT q8, q10
    CNOT q8, q11
    CNOT q8, q13
    CNOT q9, q10
    CNOT q9, q11
    CNOT q9, q12
    display
```

3.2 Question

Before doing the logical H, we are going to perform a logical CNOT between both logical qubits to check that it works properly. Name this sub-circuit .logical_CNOT.

```
[+] circuit execution time: +0.000814 sec.
-----------------------------------------
Complex amplitudes with probabilities
00000000000000       0.353553 + 0 * i (0.125000)
00011110000000       0.353553 + 0 * i (0.125000)
01100110000000       0.353553 + 0 * i (0.125000)
01111000000000       0.353553 + 0 * i (0.125000)
10101010000000       0.353553 + 0 * i (0.125000)
10110100000000       0.353553 + 0 * i (0.125000)
11001100000000       0.353553 + 0 * i (0.125000)
11010010000000       0.353553 + 0 * i (0.125000)
None
```

Figure 10: Final state of top qubit (encode circuit B)

Answer: The implementation of this circuit is by adding the code as shown below.

```
.logical_CNOT
    CNOT q0, q7
    CNOT q1 , q8
    CNOT q2 , q9
    CNOT q3 , q10
    CNOT q4 , q11
    CNOT q5 , q12
    CNOT q6 , q13
```

3.3 Question

Add the following sub-circuits: Logical H, Decoding and Measurement.

Answer: The implementation of the .logical_H circuit is adding the code below onto what we have derived till now.

```
.logical_H
    H q0
    H q1
    H q2
    H q3
    H q4
    H q5
    H q6
```

The implementation of the .decode circuit is adding the code below. It is applying the same process as encoding, however, the sequence of all the operations are reversed.

```
.decoding
    X q6
    X q5
    X q4
    X q3
    X q2
    X q1
    X q0
    CNOT q2, q5
    CNOT q2, q4
    CNOT q2, q3
    CNOT q1, q6
    CNOT q1, q4
    CNOT q1, q3
    CNOT q0, q6
    CNOT q0, q5
    CNOT q0, q3
    CNOT q6, q4
    CNOT q6, q5
    H q2
    H q1
    H q0

    CNOT q9, q12
    CNOT q9, q11
    CNOT q9, q10
    CNOT q8, q13
    CNOT q8, q11
    CNOT q8, q10
    CNOT q7, q13
    CNOT q7, q12
    CNOT q7, q10
    CNOT q13, q11
    CNOT q13, q12
    H q9
    H q8
    H q7
```

Measure qA and qB is measuring q6 and q13. With the code below added, that is realized.

```
.measurement
    Measure q6
    Measure q13
```

The measurement output should show that either qA or qB should be state $|1\rangle$, and the other one ending up in the state $|0\rangle$. As shown in figure 12 is the result get when simulating the circuit. It confirms the assumption mentioned earlier. This could also be seen from a normal bell-pair as shown

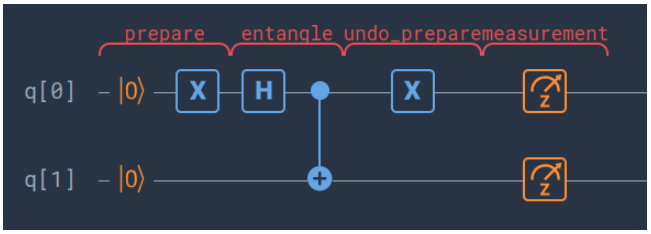


Figure 11: Same process applied on a normal bell-pair circuit

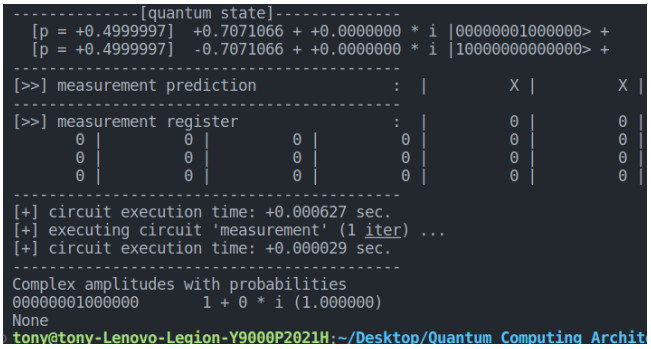


Figure 12: The final state of qA and qB after the decoding

in figure 11, we apply the exact same process as done for this FT bell-pair to it. The expression the normal bell-pair delivers is $\frac{1}{\sqrt{2}}|10\rangle - \frac{1}{\sqrt{2}}|01\rangle$, which is exactly the same case as we are getting in figure 12.

3.4 Question

Create a sub-circuit called inject_error (after the encoding sub-circuits and before the logical H section) and display the quantum state after decoding and before measurement. Before starting injecting errors, write down the binary register of the observed quantum state (the state of all the qubits after decoding).

Answer: The binary of observed quantum state is shown in figure 12. The state is expressed by $-0.7071066\,|1000000\_0000000\rangle + 0.7071066\,|0000000\_1000000\rangle$.

Implementing a logical bit-flip on qB, add the code below to the currently existing code.

```
.inject_error
    X q7
    X q8
    X q9
    X q10
    X q11
    X q12
    X q13
```

The binary observed quantum state now is either $|0000000\_0000000\rangle$ or $|1000000\_1000000\rangle$. The state did change, and the reason behind this could be achieved by experimenting on the normal bell-pair with similar processes. As shown in figure 13, the final entangled state is expressed by $\frac{1}{\sqrt{2}}|11\rangle - \frac{1}{\sqrt{2}}|00\rangle$.

Removing the previous error injection and injecting a new error of a single bit-flip in the top qubit of the logical qubit q_a. The implementation is shown as below.

```
.inject_error
    X q0
```

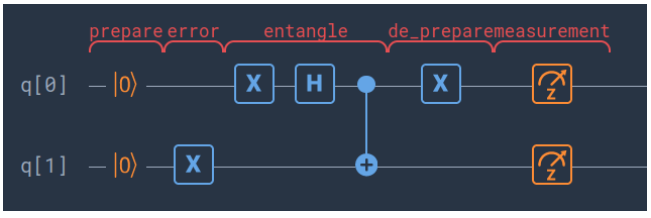


Figure 13: Bit-flip on qB, in a normal bell-pair circuit

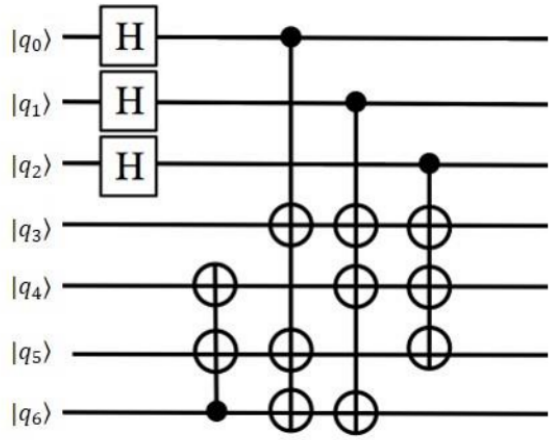


Figure 14: Steane code structure provided in Ex.2

The state did change again. The binary observed quantum state now is $0.7071066\,|1000000\_0000001\rangle - 0.7071066\,|0000000\_1000001\rangle$, note that a bit-flip occurs. First of all, most of the bits remain the same as they are when no errors exist, namely either $|1000000\_0000000\rangle$ or $|0000000\_1000000\rangle$. The state of the top physical qubit is constantly 1, this is because when a bit-flip is applied, and it passes through the Hadamard gate, the bit-flip turns into a phase-flip and does not propagates from control qubit to target qubit. Hence, you can observe a phase-flip in the final result, and a constant bit-flip that does not propagates. This could be seen in figure 14, which is the structure of the steane code given in exercise 2.

Inject a single phase-flip error in the top qubit of logical qubit qA. The implementation is shown as below.

```
.inject_error
    Z q0
```

The state did change again. The binary observed quantum state now is $0.7071066\,|1011000\_0011000\rangle - 0.7071066\,|0011000\_1011000\rangle$, note that there is a bit-flip. Similar to the situation where you only apply a single bit-flip on the top qubit, applying phase-flip also would leave q13 and a6 being either $|1\rangle$ or $|0\rangle$. As for the rest of the qubits, due to the fact that phase-flip passes through a Hadamard gate would result in bit-flip, hence it propagates to other qubits. It also leads to a bit-flip on the data qubit.

This assignment is done with the help of information found on [1], [2] and [3].

References

[1] "Steane's error correction code," (Date last accessed 08-03-2023). [Online]. Available: https://stem.mitre.org/quantum/error-correction-codes/steane-ecc.html#interpreting-the-syndrome-measurement

[2] Y. S. Weinstein, "Syndrome measurement strategies for the [[7,1,3]] code," Ph.D. dissertation, Quantum Information Science Group, Mitre, 200 Forrestal Rd, Princeton, 2015.

[3] "A short introduction to stabilizer codes," (Date last accessed 08-03-2023). [Online]. Available: https://people.engr.tamu.edu/andreas-klappenecker/689/stabilizer.pdf