Quantum Computing Architecture & Electronics Homework 2

Héctor Calero (5844460)

March 2023

1 Exercise 1: Shor code (3.5 Points)

1.1 Implement the Shor code circuit in QX. [0.5 points]

First we shall implement the 9 qubit Shor code in our QX simulator (shor_9q_code.qc):

```
# Shor 9 qubit circuit
version 0.5
qubits 9

.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[0],q[2]
cnot q[3],q[4]
cnot q[3],q[5]
cnot q[6],q[7]
cnot q[6],q[8]
display

.inject_error
x q[3]
#x q[0]
display

.bitflip_decode_and_correct
cnot q[0],q[1]
cnot q[0],q[2]
cnot q[3],q[4]
cnot q[3],q[5]
cnot q[6],q[7]
cnot q[6],q[8]
toffoli q[1],q[2],q[0]
toffoli q[4],q[5],q[3]
toffoli q[7],q[8],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[3],q[6],q[0]
display
```

The resulting state after running the circuit is the following:

```
Complex amplitudes with probabilities
000110000       0.787145 + 0.0904379 * i (0.627776)
000110001       0.118965 + -0.598391 * i (0.372224)
```

Where the rightmost digit represents q_0 and so on. Note that this state can be expressed as:

$$|00011000\rangle \otimes ((0.787145 + 0.0904379i)|0\rangle + (0.118965 - 0.598391i)|1\rangle)$$

. If we run the circuit without error injection, we get the state

$$|00000000\rangle \otimes ((0.787145 + 0.0904379i)|0\rangle + (0.118965 - 0.598391i)|1\rangle)$$

which means that the information encoded in q_0 does not change and the error correction is working.

1.2 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? [0.5 points]

Now we inject a bit flip on q_0. The outcome of the simulation now is:

```
-----------------------------------------
Complex amplitudes with probabilities
000000110       0.787145 + 0.0904379 * i (0.627776)
000000111       0.118965 + -0.598391 * i (0.372224)
```

Again we can write this as:

$$|00000011\rangle \otimes ((0.787145 + 0.0904379i)|0\rangle + (0.118965 - 0.598391i)|1\rangle)$$

As we can see, the final state of q_0 remains unchanged, which indicates that the circuit is indeed performing error correction effectively. Moreover, we can distinguish where the bit flip happened by looking at the state of the rest of the system (measuring the rest of the qubits), since the error syndromes are different for each case.

1.3 Modify the inject_error section to inject the following errors:

- **Bit-flip on qubit q_0**
- **Phase-flip on qubit q_0**
- **Bit-flip on qubit q_1**

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? [0.5 points]

The shor_9q_code _2.qc is exactly the same as the previous one but with slight changes in the injection of the error subcircuit:

```
# Shor 9 qubit circuit 2
.inject_error
x q[0]
z q[0]
x q[1]
display
```

Now the output state reads:

```
-----------------------------------------
Complex amplitudes with probabilities
001001100       0.787145 + 0.0904379 * i (0.627776)
001001101       -0.118965 + 0.598391 * i (0.372224)
```

Even though the probabilities of measuring q_0 in state 0 or 1 are the same as before, now the final state of qubit q_0 has changed to $(0.787145 + 0.0904379i)|0\rangle + (0.118965 - 0.598391i)|1\rangle$.

The reason why this happened is simple. Our circuit is designed to detect and correct single-qubit errors (bit-flips, phase-flips or combinations of both). However, we are injecting error on two different qubits at the same time. Our error correction scheme then cannot always repair the error and, in fact, can translate it into a complete logical operation. In this particular example, the effect of the error injection plus the error correction turns into a logical Z gate, and that is why there exists a phase-flip between our final decoded state and the original state.

1.4 To trace the curve of the logical error vs the the physical error, 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 curve of the physical error probability vs. logical bit flip errors. Provide the result as a graph. [2 points]

For this question, we built a new code shor_9q_code_3.qc which simply initializes $q[0]$ as 1, does not manually inject any error and includes the line of code: error_model depolarizing_channel, 0.001 (or whatever value of the physical error). We ran the circuit 10 times for each of the error probabilities that were asked and we counted the number of times the outcome state of qubit q_0 was 1 (logical bit-flip). The results are presented in the following graph:

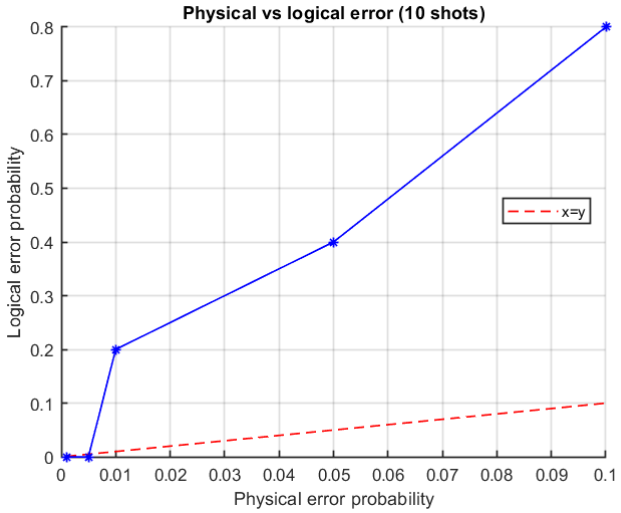

Figure 1: Physical error of the 9 qubit Shor algorithm as a function of the physical error. In red it is shown the treshold for which the physical and logical error probabilities are the same.

Even though it is true that the number of samples taken is not sufficiently large so as to obtain the precise curve of the error, we can more or less have an insight of the effect of the physical error on the logical bit-flip error. Firstly we note that for both of the error probabilities 0.001 and 0.005, not a single logical bit-flip was detected. This is obviously due to the fact that single qubit errors were unlikely, and even when they happened, they were corrected by the circuit itself. The probability of having two or three qubit errors in the circuit for these cases is almost negligible. As the error probability increases, the probability that two or more qubits have errors increases until the point that for a physical error of 0.1 we measured an 80% of logical errors.

The moral behind this experiment is that single qubit error correction codes such as the 9 qubit Shor circuit are only reliable when we can assume that the probability of more than one error occurring at a time is almost 0. If it is not the case, our circuit will indeed introduce more error than if we were treating the qubit independently rather than encoding it in a logical qubit.

2 Exercise 2: 7-qubit Steane code (2.5 Points)

2.1 Write the Encoding sub-circuits. We encode qubit q_6 in QX into the $|0\rangle$ logical state using the circuit shown above. Name this sub-circuit .encode. Display the state after encoding. [0.5 points].

The code we used to encode q_6 into the logical $|0\rangle$ is the following:

```
# Steane 7 qubit circuit
version 0.5
qubits 13

.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
```

And the state of the (7 physical) qubits after encoding looks like this:

```
----------------------------------------
Complex amplitudes with probabilities
0000000       0.353553 + 0 * i (0.125000)
0001111       0.353553 + 0 * i (0.125000)
0110011       0.353553 + 0 * i (0.125000)
0111100       0.353553 + 0 * i (0.125000)
1010101       0.353553 + 0 * i (0.125000)
1011010       0.353553 + 0 * i (0.125000)
1100110       0.353553 + 0 * i (0.125000)
1101001       0.353553 + 0 * i (0.125000)
```

2.2 Write the error detection circuit based on such stabilisers and display the error syndromes (no errors injected). Use as many ancillas as stabilisers 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. [1 point] The error detection code can be implemented like:

```
.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]

h q[9]
h q[8]
h q[7]

```

```
cnot q[9],q[0]
cnot q[9],q[1]
cnot q[9],q[2]
cnot q[9],q[3]

cnot q[8],q[0]
cnot q[8],q[1]
cnot q[8],q[4]
cnot q[8],q[5]

cnot q[7],q[6]
cnot q[7],q[4]
cnot q[7],q[2]
cnot q[7],q[0]

h q[9]
h q[8]
h q[7]

display
```

Here we simply used CNOTS with the physical qubits as the control states and the ancillas as target for the bit flip stabilizers. For the phase-flip error detectors we used CNOTS with the ancilla qubits as control and the physical qubits as target. Moreover, we need to wrap these with Hadamard gates so the measurement can be performed in the Z basis.

The total state after running the circuit with no error injection is:

```
----------------------------------------
Complex amplitudes with probabilities
0000000000000       0.353553 + 0 * i (0.125000)
0000000001111       0.353553 + 0 * i (0.125000)
0000000110011       0.353553 + 0 * i (0.125000)
0000000111100       0.353553 + 0 * i (0.125000)
0000001010101       0.353553 + 0 * i (0.125000)
0000001011010       0.353553 + 0 * i (0.125000)
0000001100110       0.353553 + 0 * i (0.125000)
0000001101001       0.353553 + 0 * i (0.125000)
```

And therefore, since the first digits appearing (corresponding to the ancilla qubits) are all 0, we know that the error syndrom for this case is: 000000.

2.3 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. [1 point]

The sub-circuit .inject_error is simply:

```
.inject_error
#x q[0]
display
```

In the commented line we can decide which operations to apply. We will perform bit-flip and phase-flip gates on all 7 qubits in order to determine all error syndromes:

	Error syndrome (6bits: bit-flip+phase-flip) order of forming the syndrome: [g1 g2 g3 g4 g5 g6]
No error	000000
Bit-flip q0	111000
Bit-flip q1	110000
Bit-flip q2	101000
Bit-flip q3	100000
Bit-flip q4	011000
Bit-flip q5	010000
Bit-flip q6	001000
Phase-flip q0	000111
Phase-flip q1	000110
Phase-flip q2	000101
Phase-flip q3	000100
Phase-flip q4	000011
Phase-flip q5	000010
Phase-flip q6	000001

There are two important conclusions that we can draw. the first one is that the errors syndrome do not repeat. This means that we can unequivocally diagnose single flip errors. Also bit-flip errors affect only to the three first stabilizers and phase-flip to the last three, so we can also identify combinations of single phase-flips and bit-flips simultaneously.

3 Exercise 3: Fault-tolerant Bell pair circuit (4 Points)

3.1 Encoding: We encode physical qubit q_B (bottom qubit) into the logical $|0\rangle$ state using the 7-qubit Steane code of Exercise 2. We also encode physical qubit q_A (top qubit) into logical $|1\rangle$state, but by first encoding it into the logical $|0\rangle$ state and then performing a logical X operation. Name these sub-circuits .encode_qB and .encode_qA. Display the two logical states (by only doing either encode A or encode B). [0.5 points]

The encoding of qubit 1 yields the 7 qubit state:

```
-----------------------------------------
Complex amplitudes with probabilities
0010110       0.353553 + 0 * i (0.125000)
0011001       0.353553 + 0 * i (0.125000)
0100101       0.353553 + 0 * i (0.125000)
0101010       0.353553 + 0 * i (0.125000)
1000011       0.353553 + 0 * i (0.125000)
1001100       0.353553 + 0 * i (0.125000)
1110000       0.353553 + 0 * i (0.125000)
1111111       0.353553 + 0 * i (0.125000)
```

And similarly for qubit B:

```
-----------------------------------------
Complex amplitudes with probabilities
0000000       0.353553 + 0 * i (0.125000)
0001111       0.353553 + 0 * i (0.125000)
0110011       0.353553 + 0 * i (0.125000)
0111100       0.353553 + 0 * i (0.125000)
1010101       0.353553 + 0 * i (0.125000)
1011010       0.353553 + 0 * i (0.125000)
1100110       0.353553 + 0 * i (0.125000)
1101001       0.353553 + 0 * i (0.125000)
```

It is evident that the state of qubit A is the result of applying an X gate on all physical qubits after encoding.

3.2 Logical CNOT: 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. [0.5 points]

The .logical_CNOT code reads:

```
.logical_CNOT
cnot q[0],q[7]
cnot q[1],q[8]
cnot q[2],q[9]
cnot q[3],q[10]
cnot q[4],q[11]
cnot q[5],q[12]
cnot q[6],q[13]
display
```

Which is the result of applying 1-to-1 cnots between the qubits encoding qA and qB.

3.3 Add the following sub-circuits:

- Logical H: perform a logical Hadamard on the top logical qubit before the CNOT. Name this subcircuit .logical_H. [0.5 points]
- Decoding: We add the last sub-circuit at the end named .decoding which performs decoding both logical states to get back their physical qubits. Write the corresponding decoding sub-circuits. What is the physical state after decoding? [0.5 points]
- Measurement: Measure q_A and q_B after decoding. What should be the measurement outcome? [0.5 points]

The logical Hadamard is performed by applying Hadamard gates to all qubits (it is a transversal logical gate). In our case we apply the logical Hadamard to q_A.

```
.logical_H #Acting on qA
h q[0]
h q[1]
h q[2]
h q[3]
h q[4]
h q[5]
h q[6]
display
```

The decoding circuit is straightforward, since we only need to apply the same operations of the encoding circuit but in the inverse order.

```
.decoding_qA
cnot q[2],q[3]
cnot q[2],q[4]
cnot q[2],q[5]
cnot q[1],q[3]
cnot q[1],q[4]
cnot q[1],q[6]
cnot q[0],q[3]
cnot q[0],q[5]
cnot q[0],q[6]
cnot q[6],q[5]
cnot q[6],q[4]
h q[0]
h q[1]
h q[2]

.decoding_qB
cnot q[9],q[10]
cnot q[9],q[11]
cnot q[9],q[12]
cnot q[8],q[10]
cnot q[8],q[11]
cnot q[8],q[13]
cnot q[7],q[10]
cnot q[7],q[12]
```

```
cnot q[7],q[13]
cnot q[13],q[12]
cnot q[13],q[11]
h q[7]
h q[8]
h q[9]
```

We know that the information is encoded in qubits q_0 and q_7, so in the no error case it is only needed to measure these states (the rest will be 0). Since the state of this two qubits will be the Bell state $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$, the two possible measurement outcomes will be $|00\rangle$ and $|11\rangle$, both with 50% probability. Indeed this is precisely what we observe when we measure.

3.4 Error injection: 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). Now do the following:

The state of the observed quantum state after decoding (with no errors) is the following:

```
-----------------------------------------
Complex amplitudes with probabilities
00000000000000       0.707107 + 0 * i (0.500000)
10000001000000       -0.707107 + 0 * i (0.500000)
```

If we recall that we are encoding our information in qubit q_6 and q_13 we can rewrite it as:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \otimes |000000000000\rangle_{ancillas}$$

Therefore we recover indeed the expression for a Bell state.

- Inject a logical bit-flip error on logical qubit qB. What is the physical quantum state now? Write down the binary register of the observed quantum state. Did it change? Explain why you got that outcome. [0.5 points]

The output state that we get is:

```
-----------------------------------------
Complex amplitudes with probabilities
00000001000000       -0.707107 + 0 * i (0.500000)
10000000000000       0.707107 + 0 * i (0.500000)
```

This is $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \otimes |000000000000\rangle_{ancillas}$. As we can see, the state has changed. This was indeed expected, since the global logical operation that we are performing is applying a Hadamard and a CNOT to the initial state $|11\rangle$. The outcome of this operation is precisely $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ (up to an irrelevant global phase).

- Remove the logical error previously injected and inject a single bit-flip in the top qubit of the logical qubit qA. What is the physical quantum state now? Write down the binary register of the observed quantum state. Did it change? Explain why you got that outcome. [0.5 points]

  The state after a bit-flip on qubit q_0 is:

  ```
  -----------------------------------------
  Complex amplitudes with probabilities
  00000000000001       0.707107 + 0 * i (0.500000)
  10000001000001       -0.707107 + 0 * i (0.500000)
  ```

  At first sight we may think that the state has changed, but indeed the information encoded remains the same:

  $$|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \otimes |000000000001\rangle_{ancillas}$$

This shows how the 7 qubit encoding protects the information, and prevents that a single bit error completely changes the state we wanted to encode (as it would happen if we performed a logical operation as before).

- Inject a single phase-flip error in the top qubit of logical qubit qA. What is the physical quantum state now? Write down the binary register of the observed quantum state. Did it change? Explain why you got that outcome. [0.5 points]

  Now the state reads:

```
1 ----------------------------------------
2 Complex amplitudes with probabilities
3 00110000011000        -0.707107 + 0 * i (0.500000)
4 10110001011000        0.707107 + 0 * i (0.500000)
```

Again it may seem that the state has changed, but in fact it can be rewritten as:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right) \otimes -|011000011000\rangle_{ancillas}$$

And therefore it is easy to see that we have done anything but added an irrelevant global phase. It is nevertheless interesting to note how a single phase-flip error on the uppermost qubit propagates into bit-flip changes into some of the other ancilla qubits. Still, the circuit is designed so as to these effects do not alter the final state of the encoded qubit.