Quantum Computing Architecture & Electronics Homework 3

Héctor Calero (5844460)

March 2023

1 Exercise 1: Ninja Star (5 Points)

1.1 Write the quantum circuit for the Ninja Star using the quantum code syntax. Each of the following subcircuits should end with a display instruction. Save the file as Ninja_star.qc.

- **Initialisation:** In this section initialise all the physical qubits to the $|0\rangle$ state. Name this subcircuit .init. E.g. use the prep_z q[0] command to initialize q[0] to the $|0\rangle$ state. **[0.5 points]**
- **Surface code cycle:** Write the circuit for the surface code cycle of the above Ninja Star (do not copy the circuit shown in the lecture because it is different!). The order of the qubits from top to bottom must be: D1, D2, D3, D4, D5, D6, D7, D8, D9, X1, Z2, X2, Z1,Z3, X3, Z4, X4. Name this subcircuit surface_code_cycle. **[0.5 points]**
- **Syndrome measurement:** Measure all the ancilla qubits after the surface code cycle. Name this subcircuit syndromes_code. Add display_binary at the end of the section. **[0.5 points]**

First we shall implement the 17 ninja Star code in our QX simulator (Ninja_Star.qc). In order to simplify the notation we map the physical qubits (q[0], q[1]...) to their corresponding names in the problem notation (D1, D2, ...).

```
# Ninja Star circuit

version 0.5
qubits 17
map q[0],D1
map q[1],D2
map q[2],D3
map q[3],D4
map q[4],D5
map q[5],D6
map q[6],D7
map q[7],D8
map q[8],D9
map q[9],X1
map q[10],Z2
map q[11],X2
map q[12],Z1
map q[13],Z3
map q[14],X3
map q[15],Z4
map q[16],X4

.init
prep_z q[0:16]

.surface_code_cycle
h X1
h X2
h X3
h X4

cnot D1,Z1
cnot X2,D2
cnot D3,Z2
cnot X3,D6
cnot D5,Z3
cnot X4,D8

cnot X2,D1
cnot D2,Z2
cnot X3,D5
cnot D4,Z3
cnot X4,D7
cnot D6,Z4

cnot X1,D3
cnot D4,Z1
cnot X2,D5
cnot X3,D9
cnot D8,Z3
cnot D6,Z2

cnot X1,D2
cnot X2,D4
cnot D5,Z2
cnot X3,D8
cnot D7,Z3
cnot D9,Z4

h X1
h X2
h X3
h X4
display

.syndromes_code
measure q[9:16]
display_binary
```

1.2 Run the circuit several times and display the output of the syndrome measurements. What do you observe? Do you see a pattern in the error syndromes? Why do you observe that? [0.5 points]

The results that we obtain seem a little arbitrary. In fact, the final simulated state (even before the measurements) change for different runs of the same simulation. This is an indicator that something in the QX simulator is not working correctly.

```
--------------[quantum state]--------------
  [p = +0.0351562]  +0.1875000 + +0.0000000 * i |00000000000000000> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000000000100> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000000110100> +
  [p = +0.0156250]  +0.1250000 + +0.0000000 * i |00000000001000000> +
  [p = +0.0156250]  +0.1250000 + +0.0000000 * i |00000000010000000> +2224)
                    .
                    .
                    .
```

```
--------------[quantum state]--------------
  [p = +0.0156250]  +0.1250000 + +0.0000000 * i |00000000000000000> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000000000110> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000000011001> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000011000000> +
  [p = +0.0039062]  +0.0625000 + +0.0000000 * i |00000000100000010>
                    .
                    .
                    .
```

The results shown above are partial results of two consecutive runs of the Ninja.qc code. The results show that both states and amplitudes before the measurement change, which should not happen.

Therefore it is impossible to detect any pattern, but we expect that not all possible error syndromes occur.

1.3 Now we are going to try to initialise our Ninja Star to the $|+\rangle$ state. To this purpose, initialise all the data qubits to the $|+\rangle$ state and the ancillas remain in the $|0\rangle$ state. Run the circuit several times and display the output of the syndrome measurements. What do you observe? Do you see a pattern in the error syndromes? Why do you observe that? [1 point]:

We now perform slight changes to introduce Hadamard gates in all data qubits in the initialization of our code, which now reads:

```
.init
prep_z q[0:16]

h D1
h D2
h D3
h D4
h D5
h D6
h D7
h D8
h D9
display
```

This time, the measurement outputs seem to be completely random. This is, we obtain all possible combinations of error syndromes.

The reason why this is now the case is simple. Our state in the data qubits after the initialization is a maximal superposition of all possible states: $|+++++++++\rangle = |000000000\rangle + |000000001\rangle + \cdots + |111111111\rangle$ (neglecting the normalization constant). Then, this can be understood as a superposition of all possible errors acting on our initial logical $|0\rangle$, and therefore it is reasonable that all possible combinations of error syndromes can also be measured.

1.4 We are not going to implement in QX a decoder that corrects for those 'initialisation' errors but we can derive the look up tables for detecting bit-flip and phase-flip errors separately. Write the two look up tables for 10 possible combinations of X ancillas and Z ancillas firing (out of 16 in total) and give two possible errors that lead to that error syndromes. Use the following format. [2 points]

X-Ancilla(s) firing	Z error in qubit(s)	Z-Ancilla(s) firing	X error in qubit(s)
No ancilla firing	No error/D1,D2,D3	No ancilla firing	No error/D2,D6,D9
X1	D3/D1,D2	Z1	D1/D4,D7
X2	D1/D4	Z2	D2/D3
X3	D6/D9	Z3	D7/D8
X4	D7/D8,D9	Z4	D9/D3,D6
X1,X2	D2/D1,D3	Z2,Z3	D5/D3,D7
X1,X3	D3,D6/D2,D5	Z2,Z4	D6/D3,D9
X1,X4	D3,D7/D1,D2,D8,D9	Z1,Z2	D1,D2/D4,D5
X2,X3	D5/D1,D9	Z1,Z4	D1,D9/D4,D7,D3,D6
X3,X4	D8/D7,D9	Z1,Z3	D4/ D1,D7
X1,X2,X3	D3,D5/D2,D6	Z2,Z3,Z4	D5,D9/D6,D8

As we can see from the selected cases, there is usually a possible error that involves less X or Z errors in the data qubits. If our error probability is low enough, we can assume that the probability of many errors occurring at the same time is very small. In this cases, we can unequivocally detect errors with a high certainty by choosing the most plausible scenarios.

2 Exercise 2: Scheduling and mapping (5 Points)

2.1

a) Draw the quantum interaction graph of the circuit in the Figure 2.1. Then find a good placement for all the 9 qubits in a 2D grid (assume the grid size is 3 $\times$ 3, that is, 9 possible locations), that minimises the communication overhead. Note that the communication overhead is the sum of each weight of the edges of the interaction graph multiplied by its Manhattan distance. You can compute the Manhattan distance as follows where a_i and $b_i$are the (x, y) coordinates of the qubits in the 2D lattice: d(a, b) = $\sum_i |a_i b_i|$. Comment on the advantages or disadvantages of this initial placement method.[0.5 points]

The quantum interaction graph can be simply drawn by connecting the qubits which share a CNOT gate. Since there is no more than one CNOT per pair of qubits, the weights of the edges are all 1. (We will encode our state in q0).

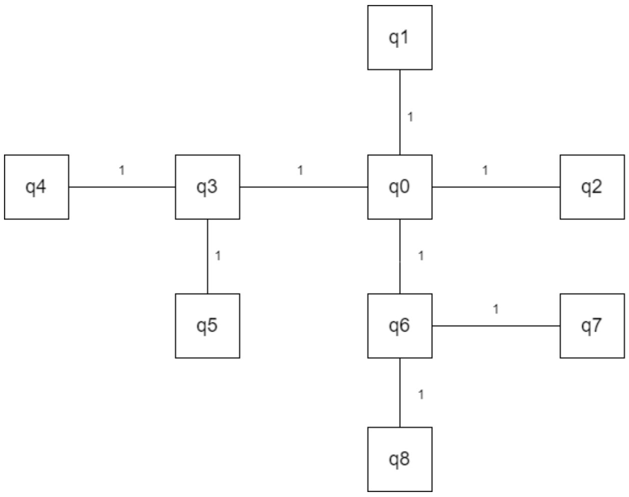

Figure 1: Quantum interaction graph of the 9-qubit Shor error correction encoding circuit.

From this graph it is immeadiate to realize that the qubit with most connections is q0, followed by q3 and q6. Therefore, a convenient placement for the qubits in a 2D grid would be to place q0 in the center and q3 and q6 on the edges (not the vertex). The rest of the qubits are simply placed next to the qubits they edges in common. An example of this placement could be the following:

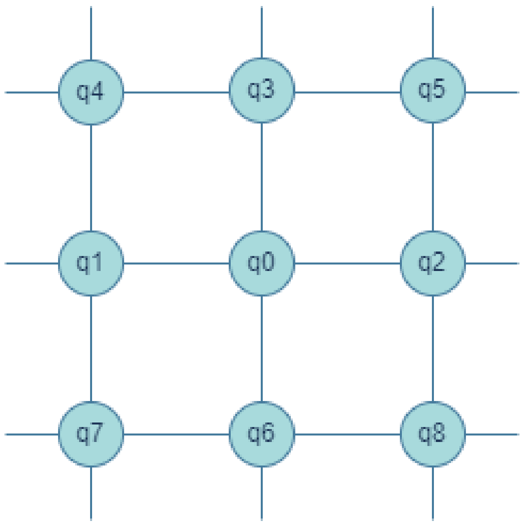

Figure 2: Placement of the 9 qubits on a 2D grid which minimizes the communication overhead.

It can be seen that the Manhattan is 1 in all cases, since all qubits which must have connectivity are adjacent. Therefore the communication overhead is simply the sum of the weights of the edges which is 8. Since there are 8 CNOTS in the circuit, this is the minumum value that we can obtain. The advantage of this placement is , of course, that the CNOTS can be implemented directly between neighbours, without needing additional qubits.

b) Write the quantum circuit shown in Figure 2.1 using the quantum code syntax. Save the file as Shor_encoding.qc. End the circuit with the display function. [0.3 points]

The code we used to encode q_0 into the logical $|0\rangle$ is the following:

```
# Shor encoding circuit
version 0.5
qubits 9

.init
prep_z q[0:9]
display

.encoding
cnot q[0],q[3]
cnot q[0],q[6]
h q[0]
h q[3]
h q[6]

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
```

And the state of the (9 physical) qubits after encoding looks like this:

```
-------------------------------------------
Complex amplitudes with probabilities
000000000       0.353553 + 0 * i (0.125000)
000000111       0.353553 + 0 * i (0.125000)
000111000       0.353553 + 0 * i (0.125000)
000111111       0.353553 + 0 * i (0.125000)
111000000       0.353553 + 0 * i (0.125000)
111000111       0.353553 + 0 * i (0.125000)
111111000       0.353553 + 0 * i (0.125000)
111111111       0.353553 + 0 * i (0.125000)
```

2.2 Draw the quantum instruction dependency graph (QIDG) for the circuit in the Figure 2.1. [1 point]

The QIDG is the following, where C_{ij} represents a CNOT with control on qubit qi and target qj. Note that in principle we do not explicitely impose any dependency between CNOTS in the last level, since this constraint will depend on the specifications of our hardware.

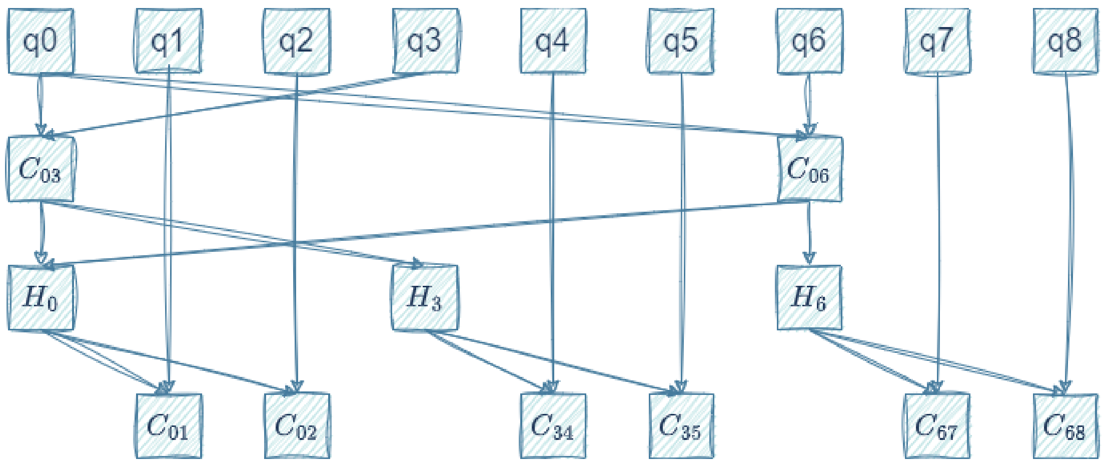

Figure 3: Quantum instruction dependency graph for the 9-qubit Shor error correction encoding circuit
.

2.3 Assume that the circuit in the Figure 2.1 is a logical circuit in which each qubit is a planar-based logical qubit (surface code). Based on the QIDG, schedule this logical circuit using: 1) first as soon as possible (ASAP) algorithm and 2) then as late as possible (ALAP). For simplicity, we assume that all the gates have the same execution time. Remember: in planar surface code only single-control single-target CNOT gates are allowed. Write both scheduled circuits using the quantum code syntax. Save these files as Shor_planar_ASAP.qc and Shor_planar_ALAP.qc, respectively. [2x0.8 points]

The ASAP circuit instructions are:

- 1) Initialize q[0:8]
- 2) C_{03}
- 3) C_{06}, H_3
- 4) H_0, H_6, C_{34}
- 5) C_{01}, C_{35}, C_{67}
- 6) C_{02}, C_{68}

And its corresponding QX code:

```
# Shor Planar ASAP
version 1.0
qubits 9

.step1
prep_z q[0:8]

.step2
cnot q[0],q[3]

.step3
{cnot q[0],q[6] | h q[3]}

.step4
```

```
{ h q[0] | h q[6]| cnot q[3],q[4]}

.step5
{cnot q[0],q[1] | cnot q[3],q[5] | cnot q[6],q[7]}

.step6
{cnot q[0],q[2] | cnot q[6],q[8]}

display
```

For the ALAP scheduling:

- 1) Initialize q0, q3
- 2) C_{03}, initialize q6
- 3) C_{06}
- 4) H_0, H_3, H_6, initialize q1, q4, q7
- 5) C_{01}, C_{34}, C_{67}, initialize q2, q5, q8
- 6) C_{02}, C_{35}, C_{68}

And its code:

```
# Shor Planar ALAP
version 0.5
qubits 9

.step1
{prep_z q[0]| prep_z q[3]}

.step2
{cnot q[0],q[3] | prep_z q[6]}

.step3
cnot q[0],q[6]

.step4
{ h q[0] |h q[3]|  h q[6]| prep_z q[1] | prep_z q[4] | prep_z q[7] }

.step5
{cnot q[0],q[1] | cnot q[3],q[4] | cnot q[6],q[7] | prep_z q[2] | prep_z q[5] | prep_z q
    [8]}

.step6
{cnot q[0],q[2] |  cnot q[3],q[5] | cnot q[6],q[8]}

display
```

As we can see, both cases yield a total of 6 steps.

2.4 Assume now that the circuit in the Figure 2.1 is a logical circuit in which each qubit is a defect-based logical qubit (surface code). Based on the QIDG, schedule this logical circuit using: 1) first as soon as possible (ASAP) algorithm and 2) then as late as possible (ALAP). Again, assume that all the gates have same execution time. Remember: in defect-based surface code single-control multi-target CNOT gates are allowed.Write both scheduled circuits using the quantum code syntax and save them as Shor_defect_ASAP.qc and Shor_defect_ALAP.qc, respectively. [2x0.8 points]

The ASAP scheduling in this case yields:

- 1) Initialize q[0:8]
- 2) C_{03}, C_{06}
- 3) H_0, H_3, H_6

- 4) C_{01},C_{02}, C_{34}, C_{35}, C_{67}, C_{68}

And its corresponding QX code:

```
# Shor Defect ASAP
version 0.5
qubits 9

.step1
prep_z q[0:8]

.step2
{cnot q[0],q[3] |cnot q[0],q[6]}

.step3
{ h q[0]| h q[3] | h q[6]}

.step4
{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
```

For the ALAP case, the instructions are:

- 1) Initialize q0, q3, q6
- 2) C_{03}, C_{06}
- 3) H_0, H_3, H_6, initialize q1, q2, q4, q5, q7, q8
- 4) C_{01},C_{02}, C_{34}, C_{35}, C_{67}, C_{68}

And the code is:

```
# Shor Defect ALAP
version 0.5
qubits 9

.step1
{prep_z q[0] | prep_z q[3] | prep_z q[6]}

.step2
{cnot q[0],q[3] |cnot q[0],q[6]}

.step3
{ h q[0]| h q[3] | h q[6] | prep_z q[1] | prep_z q[2] | prep_z q[4] | prep_z q[5] |
    prep_z q[7] | prep_z q[8]}

.step4
{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
```

Now the total number of steps is reduced to 4. Just to double ckeck that our instructions are correct, we can run all these circuits and prove that the outcome is indeed the same as in the Shor_encoding.qc

2.5 [BONUS] Write the quantum circuit shown in the second figure including the encoding part (as shown in the first figure) using the quantum code syntax. Save the file as Shor_serial.qc. Create two sub-circuits called .encoding and .detection. Both sub-circuits should end with a display instruction. Run the circuit and check if any ancilla fires. [0.5 points]

The Shor_serial.qc code reads:

```
# Shor serial circuit
version 0.5
qubits 17

.init
prep_z q[0:9]

.encoding
cnot q[0],q[3]
cnot q[0],q[6]
h q[0]
h q[3]
h q[6]

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

.detection
cnot q[0],q[9]
cnot q[1],q[9]
cnot q[1],q[10]
cnot q[2],q[10]
cnot q[3],q[11]
cnot q[4],q[11]
cnot q[4],q[12]
cnot q[5],q[12]
cnot q[6],q[13]
cnot q[7],q[13]
cnot q[7],q[14]
cnot q[8],q[14]

h q[0:8]
cnot q[0],q[15]
cnot q[1],q[15]
cnot q[2],q[15]
cnot q[3],q[15]
cnot q[4],q[15]
cnot q[5],q[15]
cnot q[3],q[16]
cnot q[4],q[16]
cnot q[5],q[16]
cnot q[6],q[16]
cnot q[7],q[16]
cnot q[8],q[16]
h q[0:8]

measure q[9:16]
display
```

The output state is the following:

```
-------------------------------------------
Complex amplitudes with probabilities
00000000000000000       0.353553 + 0 * i (0.125000)
00000000000000111       0.353553 + 0 * i (0.125000)
00000000000111000       0.353553 + 0 * i (0.125000)
00000000000111111       0.353553 + 0 * i (0.125000)
00000000111000000       0.353553 + 0 * i (0.125000)
```

```
00000000111000111        0.353553 + 0 * i (0.125000)
00000000111111000        0.353553 + 0 * i (0.125000)
00000000111111111        0.353553 + 0 * i (0.125000)
```

From here we see that all the ancilla qubits are in the state $|0\rangle$, so none of them will trigger.