QC Architecture and Electronics (CESE4080) Homework 3

Tongyun Yang (5651794, tongyunyang@tudelft.nl)

2023.03.12

1 Exercise 1 Ninja Star

1.1 Question

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

Answer:

Initialization The code preparing all qubits to $|0\rangle$ state, arranging them in the order as shown figure 1, and mapping them each to their corresponding name (e.g. Data qubit 1 = D1, Z ancilla qubit 1 = Z1, etc.) is shown below.

```
Version 0.5

qubits 17

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

Surface code cycle The code executing the surface code cycle is shown below, it is added on top of the former code. Personally, I prefer to write the code using bundles, however, it is interchangeable, so both answers are provided.

```
.surface_code_cycle
    {H X1 | H X2 | H X3 | H X4}
```

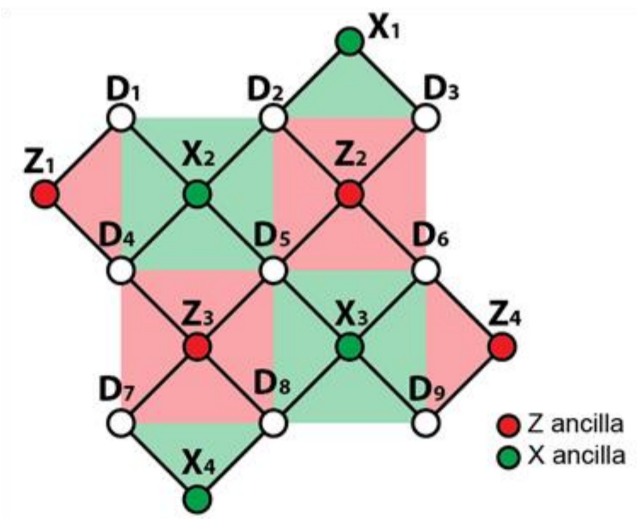


Figure 1: Ninja star structure for Exercise 1

```

    {CNOT D3, Z2 | CNOT X2, D2 | CNOT D1, Z1 | CNOT D5, Z3 | CNOT X3, D6 | CNOT X4, D8}
    #CNOT D3, Z2
    #CNOT X2, D2
    #CNOT D1, Z1
    #CNOT D5, Z3
    #CNOT X3, D6
    #CNOT X4, D8

    {CNOT D2, Z2 | CNOT X2, D1 | CNOT D4, Z3 | CNOT X3, D5 | CNOT D6, Z4 | CNOT X4, D7}
    #CNOT D2, Z2
    #CNOT X2, D1
    #CNOT D4, Z3
    #CNOT X3, D5
    #CNOT D6, Z4
    #CNOT X4, D7

    {CNOT X1, D3 | CNOT D6, Z2 | CNOT X2, D5 | CNOT D4, Z1 | CNOT D8, Z3 | CNOT X3, D9}
    #CNOT X1, D3
    #CNOT D6, Z2
    #CNOT X2, D5
    #CNOT D4, Z1
    #CNOT D8, Z3
    #CNOT X3, D9

    {CNOT X1, D2 | CNOT D5, Z2 | CNOT X2, D4 | CNOT D7, Z3 | CNOT X3, D8 | CNOT D9, Z4}
    #CNOT X1, D2
    #CNOT D5, Z2
    #CNOT X2, D4
    #CNOT D7, Z3
    #CNOT X3, D8
    #CNOT D9, Z4

    {H X1 | H X2 | H X3 | H X4}
    display
```

Syndrome measurement The code measuring the ancilla qubits and displaying the measurement registers is shown below. The code is added on top of the former existing code.

```
.syndromes_code
    {Measure X1 | Measure Z2 | Measure X2 | Measure Z1 | Measure Z3 | Measure X3 | Measure Z4 | Measure X4}
    display
    display_binary
```

1.2 Question

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?

Answer: In table 1, the syndromes are measured. Figure 2 shows the circuit in detail. It is observed that all X ancilla qubits which detects phase-flip could potentially be firing, while all Z ancilla qubits which detects bit-flip remains state $|0\rangle$. The analysis is as follows. First, it is observed that X3, X2 and Z2 (namely q[14], q[11] and q[10]) are entangled. Moreover, X2(q[11]) is entangled with D2(q[1]), D5(q[4]) and X3(q[14]); Z2(q[10]) is entangled with D5(q[4]), D6(q[5]) and X3(q[14]). Ignoring other qubits and only considering q[14], q[11], q[10] q[5], q[4] and q[1]. The expression before applying the second Hadamard gate could be written as: $\frac{1}{2}|000000\rangle + \frac{1}{2}|010011\rangle + \frac{1}{2}|100110\rangle + \frac{1}{2}|11010\rangle$. Hence after applying the Hadamard gate, Z2(q[10]) remains state $|0\rangle$. Next it is observed that Z4(q[15]), X3(q[14]), D9(q[8]) and D6(q[5]) are entangled, with the same method above applied, it could be found that the expression before applying the second Hadamard gate could be written as: $\frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|0111\rangle$. Hence after applying the Hadamard gate, Z4(q[15]) remains state $|0\rangle$. After that, it is also observed that X4(q[16]), Z3(q[13]), D8(q[7]) and D7(q[6]) are entangled, with the same approach the expression is: $\frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|1011\rangle$.

Above all are the reasons why Z ancilla qubits do not get fired, only X ancilla quits get fired, which is also the phenomenon observed. If data qubits are initialized in the Z-axis, only X ancilla qubits get fired.

1.3 Question

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?

Answer: Applying a Hadamard gate to q[0] to q[8] in the .init sub-circuit would do the job of initializing the Ninja star to $|+\rangle$ state.

```
.init
    prep_z q[0:16]
    H q[0:8]
```

	[X4, Z4, X3, Z3, Z1, X2, Z2, X1]
Run 1	[1,0,0,0,0,0,0,0]
Run 2	[0,0,0,0,0,0,0,0]
Run 3	[0,0,0,0,0,0,0,1]
Run 4	[0,0,0,0,0,1,0,1]
Run 5	[0,0,0,0,0,1,0,0]
Run 6	[1,0,1,0,0,0,0,0]
Run 7	[0,0,1,0,0,1,0,0]
Run 8	[0,0,1,0,0,1,0,1]
Run 9	[1,0,1,0,0,0,0,0]
Run 10	[1,0,0,0,0,0,0,0]
Run 11	[0,0,0,0,0,0,0,0]
Run 12	[0,0,1,0,0,0,0,0]
Run 13	[0,0,1,0,0,1,0,0]
Run 14	[1,0,1,0,0,1,0,1]
Run 15	[0,0,0,0,0,0,0,0]
Run 16	[1,0,1,0,0,1,0,0]

Table 1: Syndrome measurement after 16 runs

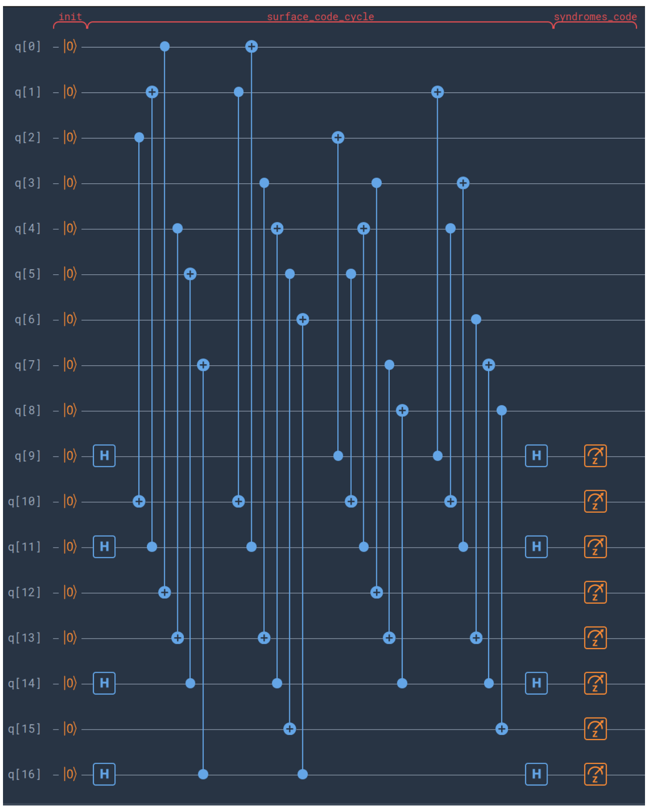


Figure 2: Ninja star one surface code cycle

	[X4, Z4, X3, Z3, Z1, X2, Z2, X1]
Run 1	[0,1,0,0,0,0,0,0]
Run 2	[0,1,0,0,0,0,1,0]
Run 3	[0,0,0,1,1,0,1,0]
Run 4	[0,0,0,0,0,0,0,0]
Run 5	[0,1,0,1,1,0,0,0]
Run 6	[0,0,0,1,1,0,0,0]
Run 7	[0,0,0,0,1,0,1,0]
Run 8	[0,1,0,0,0,0,1,0]
Run 9	[0,1,0,0,0,0,1,0]
Run 10	[0,1,0,1,1,0,1,0]
Run 11	[0,1,0,1,1,0,0,0]
Run 12	[0,0,0,1,0,0,0,0]
Run 13	[0,0,0,0,0,0,1,0]
Run 14	[0,1,0,0,0,0,1,0]
Run 15	[0,0,0,1,0,0,1,0]
Run 16	[0,1,0,0,0,0,0,0]

Table 2: Syndrome measurement after 16 runs

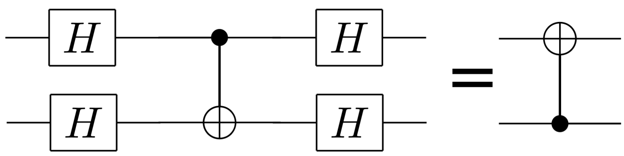


Figure 3: Equivalent logical operations

```
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
display
```

It is observed that only Z-errors ancilla qubits (namely ancilla qubits for measure bit-flip errors) are fired. This is because the data qubits are initialized in the X-axis, and the analysis follows a similar pattern as done in the previous exercise. However, this time in a different direction, the data qubits would propagate to the anicilla qubits (it was the other way around in the previous question). In addition, with some equivalent logical operations, like shown in figure 3, the circuit leads to only firing Z-errors ancilla qubits. **Some more analysis in detial are provided as follows.** By applying the equivalent logical operation, it is seen that X-errors ancilla qubits are all not firing, because state $|0\rangle$ would not propagate. As for the Z-erros ancilla qubits, they could potentially be in state $|1\rangle$ because of the same analysis done in the last question, only with data qubits and ancilla qubits reversed.

1.4 Question

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.

Answer: The table of the syndromes and the possible combination of error qubits are listed in

X-An. firring	Z-err. in q[?]	Z-An. firring	X-err. in q[?]
Null	Null/D1,D2,D3	Null	Null/D2,D6,D9
X1	D3/D2,D1	Z1	D1/D4,D7
X2	D1/D4	Z2	D3/D2
X3	D6/D9	Z3	D8/D7
X4	D7/D8,D9	Z4	D9/D6,D3
X1,X2	D3,D1/D3,D4	Z1,Z2	D1,D3/D1,D2
X1,X3	D3,D6/D3,D9	Z1,Z3	D1,D8/D1,D7
X1,X4	D3,D7/D3,D8,D9	Z1,Z4	D1,D9/D1,D6,D2
X2,X3	D1,D6/D1,D9	Z2,Z3	D3,D8/D3,D7
X2,X4	D1,D7/D4,D7	Z2,Z4	D3,D9/D2,D9
X3,X4	D6,D7/D9,D7	Z3,Z4	D8,D9/D8,D6,D3

Table 3: Possible Syndrome list (Note that the possibility is not fully explored)

table 3. Note that Note that only two possibilities are given for each syndrome, while there are more possible combinations of error qubits. Also note that this list is based on all data qubits initialized to state $|0\rangle$.

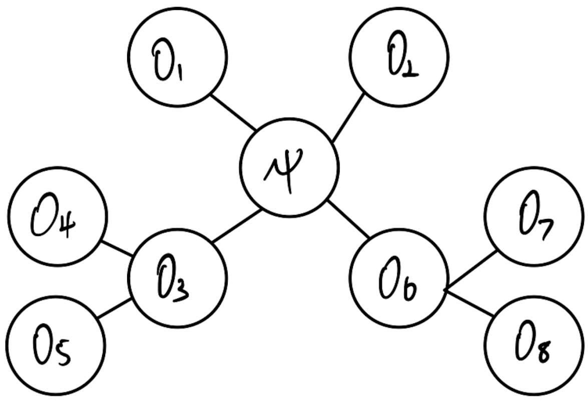


Figure 4: Quantum interaction graph

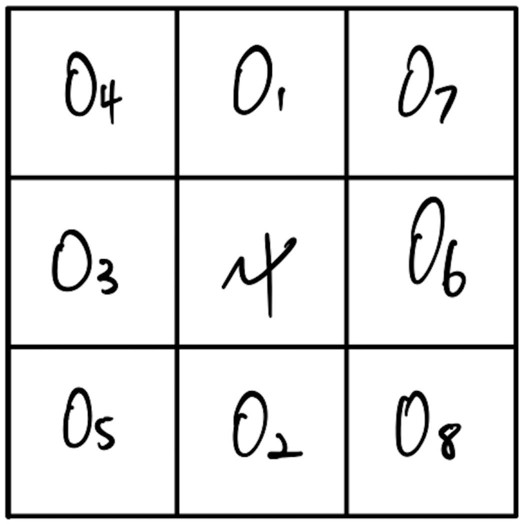


Figure 5: Qubit placement in 2D grid

2 Exercise 2 Scheduling and mapping

2.1 Question

This question consists of two parts, a) and b).

2.1.1 Question Draw the quantum interaction graph of the circuit. Then find a good placement for all the 9 qubits in a 2D grid (assume the grid size is 3×3, 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 provided. Comment on the advantages or disadvantages of this initial placement method.

Answer: The quantum interaction graph is shown in figure 4, and the placement of them on on a 2D grid of size 3x3 that minimizes the overhead is shown in figure 5. The manhattan distance of the figure 5 is 8. The advantage of the placement as shown in the figure is that it has the least Manhattan distance, and it is also provable to be be having the least Manhattan distance. The disadvantage is that this placing does not consider the frequency of interactions between qubits, in other words, it does not consider "weight". E.g. o_7 and o_5 might be interacting with each other most often among the others, however, they are placed very far apart.

2.1.2 Question Write the quantum circuit using the quantum code syntax. Save the file as Shor_encoding.qc. End the circuit with the display function.

Answer: The code is shown below.

```
Version 0.5

qubits 9

.init
    prep_z q[0:8]
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
    CNOT psi, o3
    CNOT psi, o6
    {H psi | H o3 | H o6}
    {CNOT psi, o1 | CNOT o3, o4 | CNOT o6, o7}
    {CNOT psi, o2 | CNOT o3, o5 | CNOT o6, o8}
    display
```

2.2 Question

Draw the quantum instruction dependency graph (QIDG) for the circuit.

Answer: First, we rearrange the code above a bit by removing the bundles and giving each line a name. Note that the command 'prep_z' for each qubit occupies a slot as well. (This is slightly different from what has been given in the lectures, but I find it clearer.)

```
Version 0.5

qubits 9

.init
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
G1:     prep_z psi
G2:     prep_z o1
G3:     prep_z o2
G4:     prep_z o3
G5:     prep_z o4
G6:     prep_z o5
G7:     prep_z o6
G8:     prep_z o7
G9:     prep_z o8
G10:    CNOT psi, o3
G11:    CNOT psi, o6
G12:    H psi
G13:    H o3
G14:    H o6
G15:    CNOT psi, o1
G16:    CNOT o3, o4
G17:    CNOT o6, o7
G18:    CNOT psi, o2
G19:    CNOT o3, o5
G20:    CNOT o6, o8
        display
```

The drawing of the QIDG is shown in figure 6.

2.3 Question

Assume that the circuit 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.

Answer: Planar ASAP code.

```
Version 0.5

qubits 9

.init
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
```

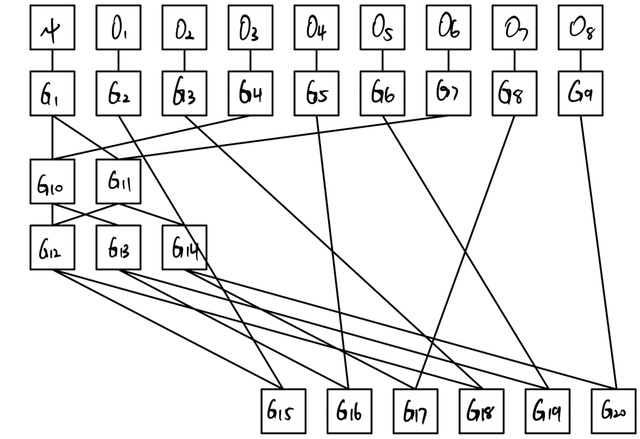


Figure 6: QIDG

```
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
    prep_z q[0:8]
    CNOT psi, o3
    {CNOT psi, o6 | H o3}
    {H psi | H o6 | CNOT o3, o4}
    {CNOT psi, o1 | CNOT o6, o7 | CNOT o3, o5}
    {CNOT psi, o2 | CNOT o6, o8}
    display
```

Planar ALAP code.

```
Version 0.5

qubits 9

.init
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
    {prep_z psi | prep_z o3}
    {CNOT psi, o3 | prep_z o6}
    CNOT psi, o6
    {H psi | H o6 | H o3 | prep_z o1 | prep_z o7 | prep_z o4}
    {CNOT psi, o1 | CNOT o6, o7 | CNOT o3, o4 | prep_z o2 | prep_z o8 | prep_z o5}
    {CNOT psi, o2 | CNOT o6, o8 | CNOT o3, o5}
    display
```

2.4 Question

Repeat question 2.3, but with defect-based logical qubit.

Answer: Defect ASAP code.

```
Version 0.5

qubits 9

.init
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
    prep_z q[0:8]
    {CNOT psi, o3 | CNOT psi, o6}
    {H psi | H o3 | H o6}
    {CNOT psi, o1 | CNOT o3, o4 | CNOT o6, o7 | CNOT psi, o2 | CNOT o3, o5 | CNOT o6, o8}
    display
```

Defect ALAP code.

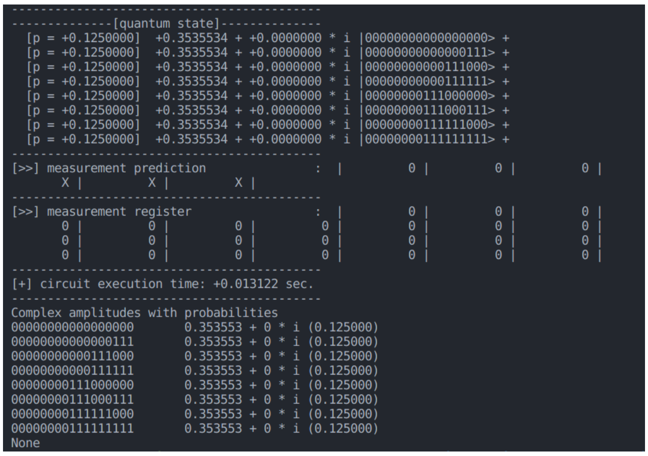
```
Version 0.5

qubits 9

.init
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8
    display

.entangle
    {prep_z psi | prep_z o3 | prep_z o6}
    {CNOT psi, o3 | CNOT psi, o6}
    {H psi | H o3 | H o6 | prep_z o1 | prep_z o4 | prep_z o7 | prep_z o2 | prep_z o5 | prep_z o8}
    {CNOT psi, o1 | CNOT o3, o4 | CNOT o6, o7 | CNOT psi, o2 | CNOT o3, o5 | CNOT o6, o8}
    display
```

2.5 Question

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.

Answer: The code of Shor_serial.qc is shown below.

```
Version 0.5

qubits 17

.init
    prep_z q[0:16]
    map q[0], psi
    map q[1], o1
    map q[2], o2
    map q[3], o3
    map q[4], o4
    map q[5], o5
    map q[6], o6
    map q[7], o7
    map q[8], o8

    map q[9], a1
    map q[10], a2
    map q[11], a3
    map q[12], a4
    map q[13], a5
    map q[14], a6
    map q[15], a7
    map q[16], a8
    display

.encode
    CNOT psi, o3
    CNOT psi, o6
    {H psi | H o3 | H o6}
    {CNOT psi, o1 | CNOT o3, o4 | CNOT o6, o7}
    {CNOT psi, o2 | CNOT o3, o5 | CNOT o6, o8}
    display

.detection
    CNOT psi, a1
    CNOT o1, a1
    CNOT o1, a2
    CNOT o2, a2
    CNOT o3, a3
    CNOT o4, a3
    CNOT o4, a4
    CNOT o5, a4
    CNOT o6, a5
    CNOT o7, a5
    CNOT o7, a6
    CNOT o8, a6

    {H psi | H o1 | H o2 | H o3 | H o4 | H o5 | H o6 | H o7 | H o8}
    CNOT psi, a7
    CNOT o1, a7
    CNOT o2, a7
    CNOT o3, a7
    CNOT o4, a7
    CNOT o5, a7
    CNOT o3, a8
    CNOT o4, a8
    CNOT o5, a8
    CNOT o6, a8
    CNOT o7, a8
    CNOT o8, a8
    {H psi | H o1 | H o2 | H o3 | H o4 | H o5 | H o6 | H o7 | H o8}
    display

    {Measure a1 | Measure a2 | Measure a3 | Measure a4 | Measure a5 | Measure a6 | Measure a7 | Measure a8}
```



```
-------------------------------------------
-------------[quantum state]-------------
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000000000000> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000000000111> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000000111000> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000000111111> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000111000000> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000111000111> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000111111000> +
  [p = +0.1250000]  +0.3535534 + +0.0000000 * i |0000000111111111> +
-------------------------------------------
[>>] measurement prediction              :  |         0 |         0 |         0 |
       X |         X |         X |
-------------------------------------------
[>>] measurement register                :  |         0 |         0 |         0 |
       0 |         0 |         0 |         0 |         0 |         0 |         0 |
       0 |         0 |         0 |         0 |         0 |         0 |         0 |
       0 |         0 |         0 |         0 |         0 |         0 |         0 |
-------------------------------------------
[+] circuit execution time: +0.013122 sec.
-------------------------------------------
Complex amplitudes with probabilities
0000000000000000       0.353553 + 0 * i (0.125000)
0000000000000111       0.353553 + 0 * i (0.125000)
0000000000111000       0.353553 + 0 * i (0.125000)
0000000000111111       0.353553 + 0 * i (0.125000)
0000000111000000       0.353553 + 0 * i (0.125000)
0000000111000111       0.353553 + 0 * i (0.125000)
0000000111111000       0.353553 + 0 * i (0.125000)
0000000111111111       0.353553 + 0 * i (0.125000)
None
```

Figure 7: Result after running Shor_serial.qc

```
display
```

No ancilla fires after several runs, it could be seen in figure 7.
This assignment is completed with the help of [1].

References

[1] L. Riesebos, “Pauli frames for quantum computer architectures,” Ph.D. dissertation, TUDelft, 2016.