

Relazione Home Assignment 8

Daniele Sacco (5616921), Daniele Scaffai (5658260), Lorenzo Vaccarecci (5462843)

Obiettivo

L'obiettivo di questo Home Assignment è l'implementazione del protocollo degli errori di Shor, in particolare gli errori di tipo bit flip, phase flip e small error.

Bit flip

Svolgimento

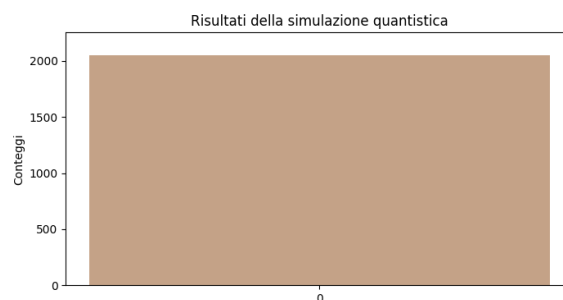
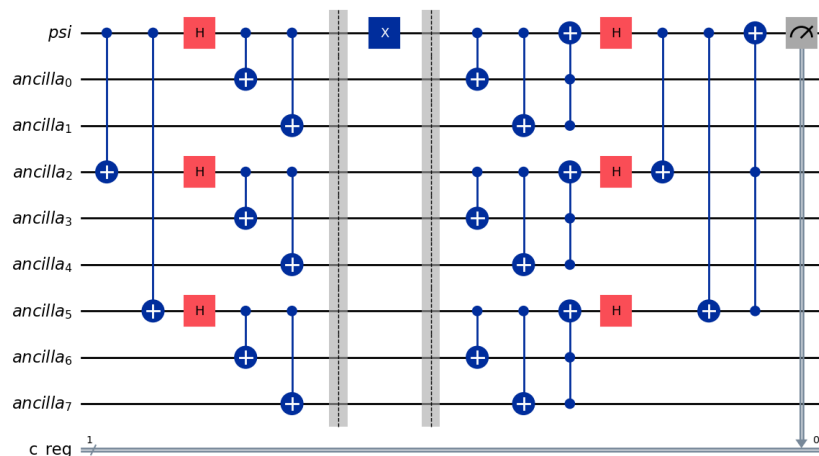
Abbiamo riprodotto, usando qiskit, il circuito che è stato fornito dal docente. Tra le due barriere viene applicata la porta di Pauli X per simulare il bit flip.

```
1 psi = QuantumRegister(1, 'psi')
2 ancilla = AncillaRegister(8, 'ancilla')
3 c_reg = ClassicalRegister(1, 'c_reg')
4 qc_x_gate = QuantumCircuit(psi, ancilla, c_reg)
5
6 qc_x_gate.cx(psi[0], ancilla[2])
7 qc_x_gate.cx(psi[0], ancilla[5])
8
9 qc_x_gate.h(psi[0])
10 qc_x_gate.h(ancilla[2])
11 qc_x_gate.h(ancilla[5])
12
13 qc_x_gate.cx(psi[0], ancilla[0])
14 qc_x_gate.cx(ancilla[2], ancilla[3])
15 qc_x_gate.cx(ancilla[5], ancilla[6])
16
17 qc_x_gate.cx(psi[0], ancilla[1])
18 qc_x_gate.cx(ancilla[2], ancilla[4])
19 qc_x_gate.cx(ancilla[5], ancilla[7])
20
21 qc_x_gate.barrier()
22 qc_x_gate.x(psi[0])
23 qc_x_gate.barrier()
24
25 qc_x_gate.cx(psi[0], ancilla[0])
26 qc_x_gate.cx(ancilla[2], ancilla[3])
27 qc_x_gate.cx(ancilla[5], ancilla[6])
28
29 qc_x_gate.cx(psi[0], ancilla[1])
30 qc_x_gate.cx(ancilla[2], ancilla[4])
31 qc_x_gate.cx(ancilla[5], ancilla[7])
32
33 qc_x_gate.ccx(ancilla[1], ancilla[0], psi[0])
34 qc_x_gate.ccx(ancilla[4], ancilla[3], ancilla[2])
35 qc_x_gate.ccx(ancilla[7], ancilla[6], ancilla[5])
36
37 qc_x_gate.h(psi[0])
38 qc_x_gate.h(ancilla[2])
39 qc_x_gate.h(ancilla[5])
40
```

```

41 qc_x_gate.cx(psi[0], ancilla[2])
42 qc_x_gate.cx(psi[0], ancilla[5])
43 qc_x_gate.ccx(ancilla[5], ancilla[2], psi[0])
44
45 qc_x_gate.measure(psi[0], c_reg[0])
46 qc_x_gate.draw(output='mpl')
47
48 simulator = AerSimulator()
49
50 job = simulator.run(qc_x_gate, shots=2048)
51 result = job.result()
52
53 counts = result.get_counts(qc_x_gate)
54 print(counts)
55
56 labels = list(counts.keys())
57 values = list(counts.values())
58
59 fig, ax = plt.subplots(figsize=(8, 4))
60
61 bars = ax.bar(labels, values, color='#c4a287')
62
63 ax.set_ylabel('Conteggi')
64 ax.set_title('Risultati della simulazione quantistica')
65 ax.set_ylim(0, max(values) * 1.1)
66
67 plt.show()

```



Phase flip

Svolgimento

Abbiamo riprodotto, usando qiskit, il circuito che è stato fornito dal docente. Tra le due barriere viene applicata la porta di Pauli Z per simulare il phase flip.

```

1 psi = QuantumRegister(1, 'psi')

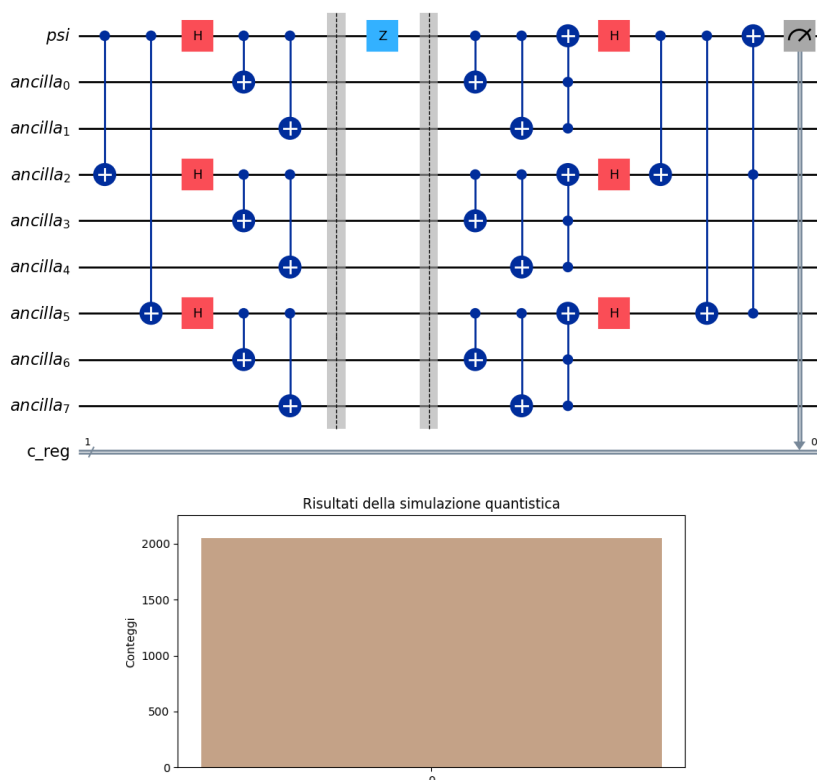
```

```

2 ancilla = AncillaRegister(8, 'ancilla')
3 c_reg = ClassicalRegister(1, 'c_reg')
4 qc_z_gate = QuantumCircuit(psi, ancilla, c_reg)
5
6 qc_z_gate.cx(psi[0], ancilla[2])
7 qc_z_gate.cx(psi[0], ancilla[5])
8
9 qc_z_gate.h(psi[0])
10 qc_z_gate.h(ancilla[2])
11 qc_z_gate.h(ancilla[5])
12
13 qc_z_gate.cx(psi[0], ancilla[0])
14 qc_z_gate.cx(ancilla[2], ancilla[3])
15 qc_z_gate.cx(ancilla[5], ancilla[6])
16
17 qc_z_gate.cx(psi[0], ancilla[1])
18 qc_z_gate.cx(ancilla[2], ancilla[4])
19 qc_z_gate.cx(ancilla[5], ancilla[7])
20
21 qc_z_gate.barrier()
22 qc_z_gate.z(psi[0])
23 qc_z_gate.barrier()
24
25 qc_z_gate.cx(psi[0], ancilla[0])
26 qc_z_gate.cx(ancilla[2], ancilla[3])
27 qc_z_gate.cx(ancilla[5], ancilla[6])
28
29 qc_z_gate.cx(psi[0], ancilla[1])
30 qc_z_gate.cx(ancilla[2], ancilla[4])
31 qc_z_gate.cx(ancilla[5], ancilla[7])
32
33 qc_z_gate.ccx(ancilla[1], ancilla[0], psi[0])
34 qc_z_gate.ccx(ancilla[4], ancilla[3], ancilla[2])
35 qc_z_gate.ccx(ancilla[7], ancilla[6], ancilla[5])
36
37 qc_z_gate.h(psi[0])
38 qc_z_gate.h(ancilla[2])
39 qc_z_gate.h(ancilla[5])
40
41 qc_z_gate.cx(psi[0], ancilla[2])
42 qc_z_gate.cx(psi[0], ancilla[5])
43 qc_z_gate.ccx(ancilla[5], ancilla[2], psi[0])
44
45 qc_z_gate.measure(psi[0], c_reg[0])
46
47 qc_z_gate.draw(output='mpl')
48
49 simulator = AerSimulator()
50
51 job = simulator.run(qc_z_gate, shots=2048)
52 result = job.result()
53
54 counts = result.get_counts(qc_z_gate)
55 print(counts)
56
57 labels = list(counts.keys())
58 values = list(counts.values())
59
60 fig, ax = plt.subplots(figsize=(8, 4))
61
62 bars = ax.bar(labels, values, color='#c4a287')
63
64 ax.set_ylabel('Conteggi')
65 ax.set_title('Risultati della simulazione quantistica')
66 ax.set_ylim(0, max(values) * 1.1)

```

```
67
68 plt.show()
```



Small error

Svolgimento

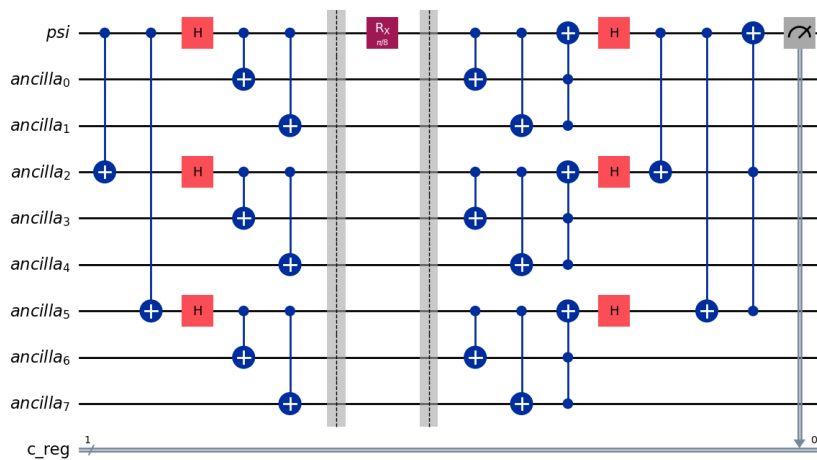
Abbiamo riprodotto, usando qiskit, il circuito che è stato fornito dal docente. Tra le due barriere viene applicata una piccola rotazione sull'asse delle x a $|\psi\rangle$ per simulare uno small error.

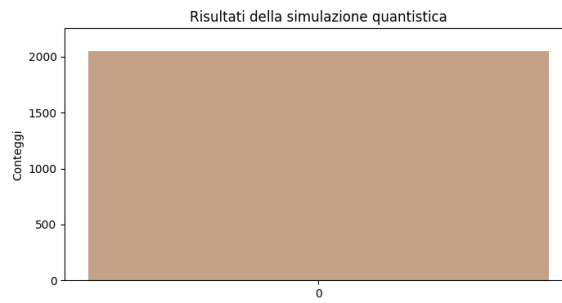
```
1 psi = QuantumRegister(1, 'psi')
2 ancilla = AncillaRegister(8, 'ancilla')
3 c_reg = ClassicalRegister(1, 'c_reg')
4 qc_rx_gate = QuantumCircuit(psi, ancilla, c_reg)
5
6 qc_rx_gate.cx(psi[0], ancilla[2])
7 qc_rx_gate.cx(psi[0], ancilla[5])
8
9 qc_rx_gate.h(psi[0])
10 qc_rx_gate.h(ancilla[2])
11 qc_rx_gate.h(ancilla[5])
12
13 qc_rx_gate.cx(psi[0], ancilla[0])
14 qc_rx_gate.cx(ancilla[2], ancilla[3])
15 qc_rx_gate.cx(ancilla[5], ancilla[6])
16
17 qc_rx_gate.cx(psi[0], ancilla[1])
18 qc_rx_gate.cx(ancilla[2], ancilla[4])
19 qc_rx_gate.cx(ancilla[5], ancilla[7])
20
21 qc_rx_gate.barrier()
22 qc_rx_gate.rx(np.pi/8, psi[0])
23 qc_rx_gate.barrier()
24
25 qc_rx_gate.cx(psi[0], ancilla[0])
26 qc_rx_gate.cx(ancilla[2], ancilla[3])
```

```

27 qc_rx_gate.cx(ancilla[5], ancilla[6])
28
29 qc_rx_gate.cx(psi[0], ancilla[1])
30 qc_rx_gate.cx(ancilla[2], ancilla[4])
31 qc_rx_gate.cx(ancilla[5], ancilla[7])
32
33 qc_rx_gate.ccx(ancilla[1], ancilla[0], psi[0])
34 qc_rx_gate.ccx(ancilla[4], ancilla[3], ancilla[2])
35 qc_rx_gate.ccx(ancilla[7], ancilla[6], ancilla[5])
36
37 qc_rx_gate.h(psi[0])
38 qc_rx_gate.h(ancilla[2])
39 qc_rx_gate.h(ancilla[5])
40
41 qc_rx_gate.cx(psi[0], ancilla[2])
42 qc_rx_gate.cx(psi[0], ancilla[5])
43 qc_rx_gate.ccx(ancilla[5], ancilla[2], psi[0])
44
45 qc_rx_gate.measure(psi[0], c_reg[0])
46
47 qc_rx_gate.draw(output='mpl')
48
49 simulator = AerSimulator()
50
51 job = simulator.run(qc_rx_gate, shots=2048)
52 result = job.result()
53
54 counts = result.get_counts(qc_rx_gate)
55 print(counts)
56
57 labels = list(counts.keys())
58 values = list(counts.values())
59
60 fig, ax = plt.subplots(figsize=(8, 4))
61
62 bars = ax.bar(labels, values, color='#c4a287')
63
64 ax.set_ylabel('Conteggi')
65 ax.set_title('Risultati della simulazione quantistica')
66 ax.set_ylim(0, max(values) * 1.1)
67
68 plt.show()

```





Risultati

E' stato aggiunto un misuratore alla fine di ogni circuito e sono state eseguite 2048 simulazioni per ciascuno. La misura attesa per ogni tipo di errore è 0, perchè è lo stato iniziale di $|\psi\rangle$.

Conclusioni

I risultati ottenuti corrispondono a quelli attesi, quindi possiamo concludere che l'implementazione del protocollo di correzione degli errori di Shor è corretta.