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.

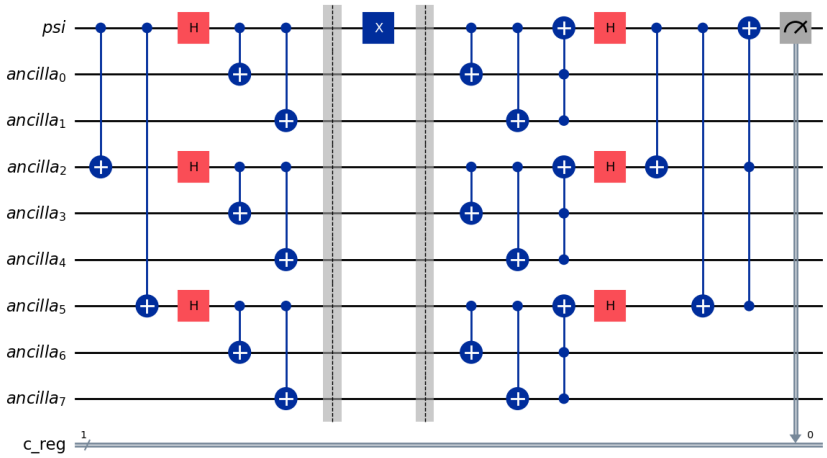
Bit flip

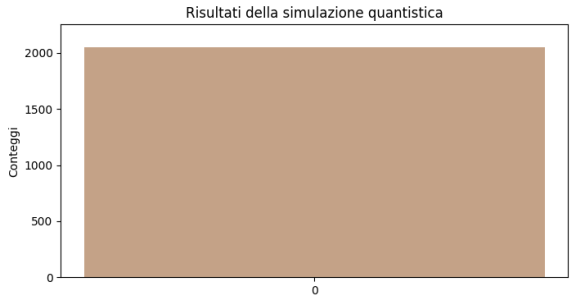
Svolgimento

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.

```
psi = QuantumRegister(1, 'psi')
ancilla = AncillaRegister(8, 'ancilla')
c_reg = ClassicalRegister(1, 'c_reg')
qc_x_gate = QuantumCircuit(psi, ancilla, c_reg)

qc_x_gate.cx(psi[0],ancilla[2])
qc_x_gate.cx(psi[0],ancilla[5])

qc_x_gate.h(psi[0])
qc_x_gate.h(ancilla[2])
qc_x_gate.h(ancilla[5])

qc_x_gate.cx(psi[0],ancilla[0])
qc_x_gate.cx(ancilla[2],ancilla[3])
qc_x_gate.cx(ancilla[5],ancilla[6])

qc_x_gate.cx(psi[0],ancilla[1])
qc_x_gate.cx(ancilla[2],ancilla[4])
qc_x_gate.cx(ancilla[5],ancilla[7])

qc_x_gate.barrier()
qc_x_gate.x(psi[0])
qc_x_gate.barrier()

qc_x_gate.cx(psi[0],ancilla[0])
qc_x_gate.cx(ancilla[2],ancilla[3])
qc_x_gate.cx(ancilla[5],ancilla[6])

qc_x_gate.cx(psi[0],ancilla[1])
qc_x_gate.cx(ancilla[2],ancilla[4])
qc_x_gate.cx(ancilla[5],ancilla[7])

qc_x_gate.ccx(ancilla[1],ancilla[0],psi[0])
qc_x_gate.ccx(ancilla[4],ancilla[3],ancilla[2])
qc_x_gate.ccx(ancilla[7],ancilla[6],ancilla[5])

qc_x_gate.h(psi[0])
qc_x_gate.h(ancilla[2])
qc_x_gate.h(ancilla[5])

```

```
qc_x_gate.cx(psi[0],ancilla[2])
qc_x_gate.cx(psi[0],ancilla[5])
qc_x_gate.ccx(ancilla[5],ancilla[2],psi[0])

qc_x_gate.measure(psi[0], c_reg[0])
qc_x_gate.draw(output='mpl')

simulator = AerSimulator()

job = simulator.run(qc_x_gate, shots=2048)
result = job.result()

counts = result.get_counts(qc_x_gate)
print(counts)

labels = list(counts.keys())
values = list(counts.values())

fig, ax = plt.subplots(figsize=(8, 4))

bars = ax.bar(labels, values, color='#c4a287')

ax.set_ylabel('Conteggi')
ax.set_title('Risultati della simulazione quantistica')
ax.set_ylim(0, max(values) * 1.1)

plt.show()
```





Phase flip

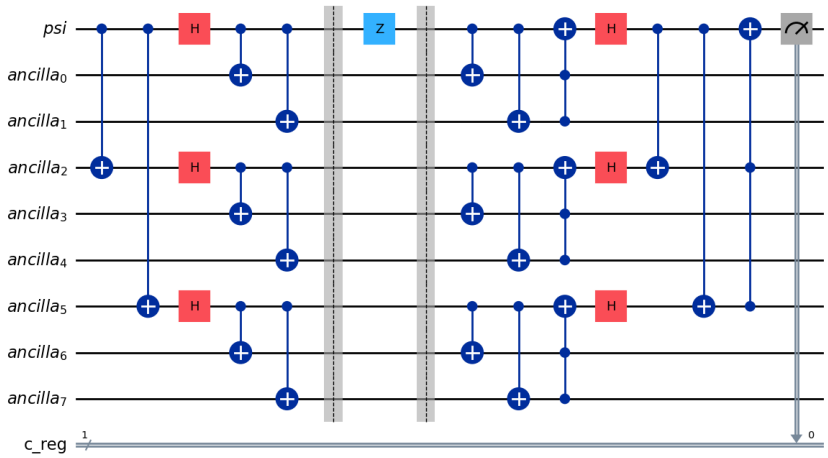
Svolgimento

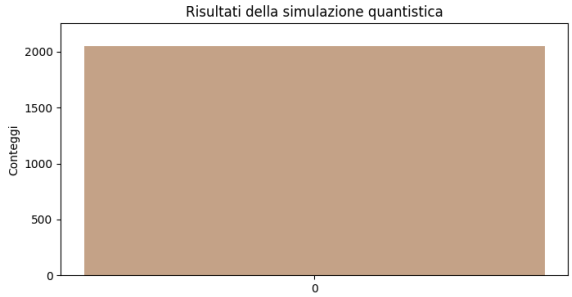
Abbiamo riprodotto, usando qiskit, il circuito che è stato fornito dal docente. Tra le due barriere viene applicata la porta di Pauli Z per simulare il phase flip.

```
psi = QuantumRegister(1, 'psi')
```

```
ancilla = AncillaRegister(8, 'ancilla')
c_reg = ClassicalRegister(1, 'c_reg')
qc_z_gate = QuantumCircuit(psi, ancilla, c_reg)

qc_z_gate.cx(psi[0],ancilla[2])
qc_z_gate.cx(psi[0],ancilla[5])

qc_z_gate.h(psi[0])
qc_z_gate.h(ancilla[2])
qc_z_gate.h(ancilla[5])

qc_z_gate.cx(psi[0],ancilla[0])
qc_z_gate.cx(ancilla[2],ancilla[3])
qc_z_gate.cx(ancilla[5],ancilla[6])

qc_z_gate.cx(psi[0],ancilla[1])
qc_z_gate.cx(ancilla[2],ancilla[4])
qc_z_gate.cx(ancilla[5],ancilla[7])

qc_z_gate.barrier()
qc_z_gate.z(psi[0])
qc_z_gate.barrier()

qc_z_gate.cx(psi[0],ancilla[0])
qc_z_gate.cx(ancilla[2],ancilla[3])
qc_z_gate.cx(ancilla[5],ancilla[6])

qc_z_gate.cx(psi[0],ancilla[1])
qc_z_gate.cx(ancilla[2],ancilla[4])
qc_z_gate.cx(ancilla[5],ancilla[7])

qc_z_gate.ccx(ancilla[1],ancilla[0],psi[0])
qc_z_gate.ccx(ancilla[4],ancilla[3],ancilla[2])
qc_z_gate.ccx(ancilla[7],ancilla[6],ancilla[5])

qc_z_gate.h(psi[0])
qc_z_gate.h(ancilla[2])
qc_z_gate.h(ancilla[5])

qc_z_gate.cx(psi[0],ancilla[2])
qc_z_gate.cx(psi[0],ancilla[5])
qc_z_gate.ccx(ancilla[5],ancilla[2],psi[0])

qc_z_gate.measure(psi[0], c_reg[0])

qc_z_gate.draw(output='mpl')

simulator = AerSimulator()

job = simulator.run(qc_z_gate, shots=2048)
result = job.result()

counts = result.get_counts(qc_z_gate)
print(counts)

labels = list(counts.keys())
values = list(counts.values())

fig, ax = plt.subplots(figsize=(8, 4))

bars = ax.bar(labels, values, color='#c4a287')

ax.set_ylabel('Conteggi')
ax.set_title('Risultati della simulazione quantistica')
ax.set_ylim(0, max(values) * 1.1)
```

```
plt.show()
```





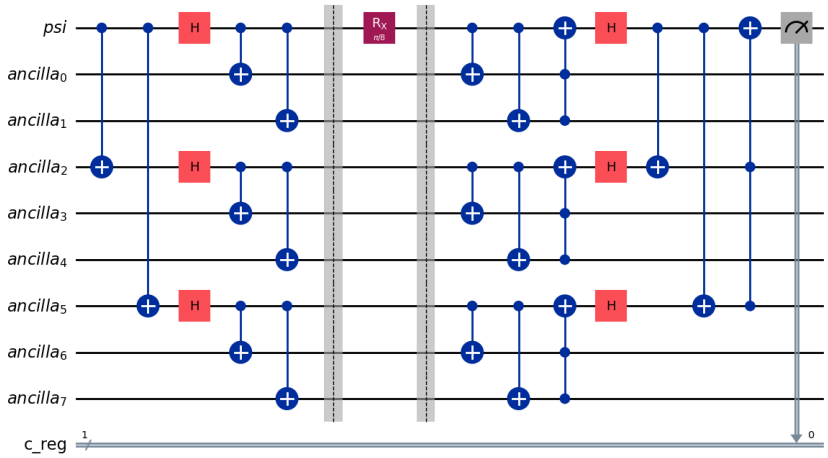
Small error

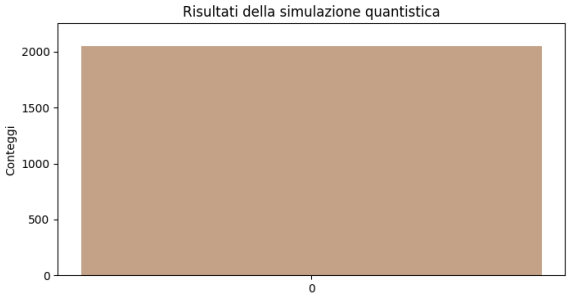
Svolgimento

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.

```
psi = QuantumRegister(1, 'psi')
ancilla = AncillaRegister(8, 'ancilla')
c_reg = ClassicalRegister(1, 'c_reg')
qc_rx_gate = QuantumCircuit(psi, ancilla, c_reg)

qc_rx_gate.cx(psi[0],ancilla[2])
qc_rx_gate.cx(psi[0],ancilla[5])

qc_rx_gate.h(psi[0])
qc_rx_gate.h(ancilla[2])
qc_rx_gate.h(ancilla[5])

qc_rx_gate.cx(psi[0],ancilla[0])
qc_rx_gate.cx(ancilla[2],ancilla[3])
qc_rx_gate.cx(ancilla[5],ancilla[6])

qc_rx_gate.cx(psi[0],ancilla[1])
qc_rx_gate.cx(ancilla[2],ancilla[4])
qc_rx_gate.cx(ancilla[5],ancilla[7])

qc_rx_gate.barrier()
qc_rx_gate.rx(np.pi/8, psi[0])
qc_rx_gate.barrier()

qc_rx_gate.cx(psi[0],ancilla[0])
qc_rx_gate.cx(ancilla[2],ancilla[3])
```

```
qc_rx_gate.cx(ancilla[5],ancilla[6])

qc_rx_gate.cx(psi[0],ancilla[1])
qc_rx_gate.cx(ancilla[2],ancilla[4])
qc_rx_gate.cx(ancilla[5],ancilla[7])

qc_rx_gate.ccx(ancilla[1],ancilla[0],psi[0])
qc_rx_gate.ccx(ancilla[4],ancilla[3],ancilla[2])
qc_rx_gate.ccx(ancilla[7],ancilla[6],ancilla[5])

qc_rx_gate.h(psi[0])
qc_rx_gate.h(ancilla[2])
qc_rx_gate.h(ancilla[5])

qc_rx_gate.cx(psi[0],ancilla[2])
qc_rx_gate.cx(psi[0],ancilla[5])
qc_rx_gate.ccx(ancilla[5],ancilla[2],psi[0])

qc_rx_gate.measure(psi[0], c_reg[0])

qc_rx_gate.draw(output='mpl')

simulator = AerSimulator()

job = simulator.run(qc_rx_gate, shots=2048)
result = job.result()

counts = result.get_counts(qc_rx_gate)
print(counts)

labels = list(counts.keys())
values = list(counts.values())

fig, ax = plt.subplots(figsize=(8, 4))

bars = ax.bar(labels, values, color='#c4a287')

ax.set_ylabel('Conteggi')
ax.set_title('Risultati della simulazione quantistica')
ax.set_ylim(0, max(values) * 1.1)

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.