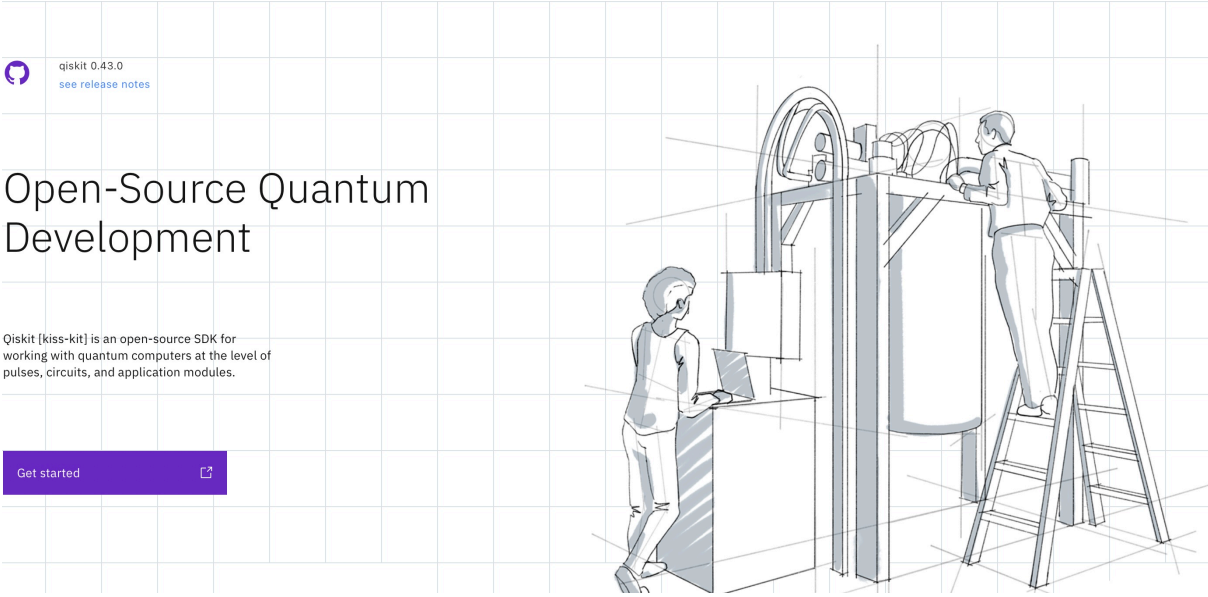
Fondamenti di Computazione Quantistica

Superdense Coding

Innanzitutto, il *superdense coding* è un protocollo quantistico grazie a cui si possono inviare un certo numero di bit logici, codificati con un minor numero di qubit.



Come funziona? 🛠️

Assumiamo di avere i classi due attori **A** e **B** (Alice e Bob) e che Alice voglia mandare, per esempio, 2 bit d'informazione logica a Bob utilizzando un unico *qubit*. Precondizione: i due attori devono condividere uno stato *entangled* così "costruito":

$$|\beta_{00}\rangle = 1/\sqrt{2} * (|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B) = (|00\rangle + |11\rangle)/\sqrt{2}$$

Il qubit con pedice A appartiene ad Alice e quello con $_B$ a Bob.

Di seguito le possibili combinazioni:

1. Alice invia a Bob ***00***: A non dovrà applicare nessuna porta (si applica la porta *Identità* per chiarezza), quindi Bob riceverà sempre lo stesso stato.

2. Alice invia a Bob ***10*** ed applicherà la porta *Z*, ottenendo lo stato

$$|\beta_{00}\rangle \; z \rightarrow 1/\sqrt{2} * (|0\rangle_B \otimes |0\rangle_B - |1\rangle_B \otimes |1\rangle_B)$$

3. Alice invia a Bob ***01*** ed applicherà la porta *X*, ottenendo lo stato

$$|\beta_{00}\rangle \; x \rightarrow 1/\sqrt{2} * (|1\rangle_B \otimes |0\rangle_B + |0\rangle_B \otimes |1\rangle_B)$$

4. Alice invia a Bob 11 ed applicherà la porta *iY*, ottenendo lo stato

$$|\beta_{00}\rangle \; iY \rightarrow 1/\sqrt{2} * (|0\rangle_B \otimes |1\rangle_B - |1\rangle_B \otimes |0\rangle_B)$$

Adesso Bob è in possesso di entrambi i qubit (il suo e quello di A), ma siccome ci sono quattro possibili stati inviati da Alice, Bob è costretto a misurare per conoscere con certezza il messaggio orinale.

Si osservi che le quattro possibilità sono gli stati ortogonali di Bell, quindi Bob è in grado di misurare con probabilità del 100% (si farà una simulazione apposita). Adesso Bob ha la possibilità di “decodificare” il messaggio ricevuto da Alice, applicandogli una porta CNOT in sequenza su i suoi due qubit e una porta di Hadamard al primo qubit.

$$I \rightarrow 1/\sqrt{2} * (|0\rangle_B \otimes |0\rangle_B + |1\rangle_B \otimes |1\rangle_B) \; CNOT \rightarrow 1/\sqrt{2} * (|0\rangle_B + |1\rangle_B) \otimes |0\rangle_B \; H \rightarrow |00\rangle_B$$

$$X \rightarrow 1/\sqrt{2} * (|1\rangle_B \otimes |0\rangle_B + |0\rangle_B \otimes |1\rangle_B) \; CNOT \rightarrow 1/\sqrt{2} * (|1\rangle_B + |0\rangle_B) \otimes |1\rangle_B \; H \rightarrow |01\rangle_B$$

$$Z \rightarrow 1/\sqrt{2} * (|0\rangle_B \otimes |0\rangle_B + |1\rangle_B \otimes |1\rangle_B) \; CNOT \rightarrow 1/\sqrt{2} * (|0\rangle_B + |1\rangle_B) \otimes |0\rangle_B \; H \rightarrow |10\rangle_B$$

$$iY \rightarrow 1/\sqrt{2} * (|0\rangle_B \otimes |1\rangle_B + |1\rangle_B \otimes |0\rangle_B) \; CNOT \rightarrow 1/\sqrt{2} * (|0\rangle_B + |1\rangle_B) \otimes |1\rangle_B \; H \rightarrow |11\rangle_B$$

Misurando in base canonica Bob ottiene i bit logici inviati originariamente da Alice.

S.D.C vs Teletrasporto Quantistico 🔭

Spiegata brevemente, la differenza sta nel fatto che il teletrasporto quantistico è un processo in cui lo stato di un qubit può essere "trasmesso" mediante l'uso dei bit logici e dello stato *entangled* mentre il *superdense coding* è un algoritmo che trasmette due bit logici usando un unico qubit.

Codice IBM Quantum Lab 👨🏻‍💻

Di seguito una suddivisione delle funzioni che evidenzia i passaggi logici della creazione, della trasmissione e della decodifica. Inoltre, per rendere più chiari i quattro casi (i messaggi che Bob riceve da Alice) si è deciso di ripetere l'esperimento per ognuno di essi.

Funzioni d'appoggio...

...che verranno utilizzate come ausilio dei vari esperimenti. La prima, *createEntangled(),* serve appunto a creare uno stato *entangled* (come descritto dal nome) condiviso tra A e B. Quindi creiamo un circuito con 2 qubit, gli applichiamo una porta di *Hadamard* e in seguito una porta *CNOT*:

[2]:
```
def createEntangled():
    es = QuantumCircuit(2)
    es.h(1)
    es.cx(1, 0)
    return es
```

La *decode()* fa l'esatto contrario della funzione soprastante, ossia applica una porta *CNOT* e successivamente una porta di *Hadamard* per distruggere l'*entanglement*:

[3]:
```
def decode(de):
    de.cx(1,0)
    de.h(1)
    return de
```

La *encode()* decide quale porta applicare, a seconda del messaggio inviato da Alice:

[4]:
```
def encode(sdc,msg):
    if msg == "00":
        sdc.i(1)
        return sdc
    elif msg == "01":
        sdc.x(1)
        return sdc
    elif msg == "10":
        sdc.z(1)
        return sdc
    else :
        sdc.y(1)
        return sdc
```

La funzione *misurazioniSim()* verrà eseguita alla fine per effettuare le misurazioni sul simulatore e raccogliere i dati dell'esperimento:

[5]:
```
def misurazioniSim(sdc):
    job = execute(sdc, backend = deviceSim)
    device_result = job.result().get_counts()
    return device_result;
```

La sua corrispettiva, *misurazioniReal()*, invece verrà eseguita per effettuare delle misurazione su un reale computer quantistico e raccoglierne i relativi dati:

[6]:
```
def misurazioniReal(sdc):
    job = execute(sdc,backend = deviceReal,shots = 1024)
    print(job.job_id())
    job_monitor(job)
    device_result = job.result().get_counts()
    return device_result
```

Misurazioni reali o simulate?

Per ogni caso descritto in precedenza ho effettuato misurazioni sia sul simulatore che su un computer quantistico "in carne ed ossa" (gentilmente fornito da IBM), riporto successivamente due tipi di istogrammi, uno verde che rappresenta i dati ricevuti dalla simulazione e uno blu che raffigura i dati del computer quantistico.

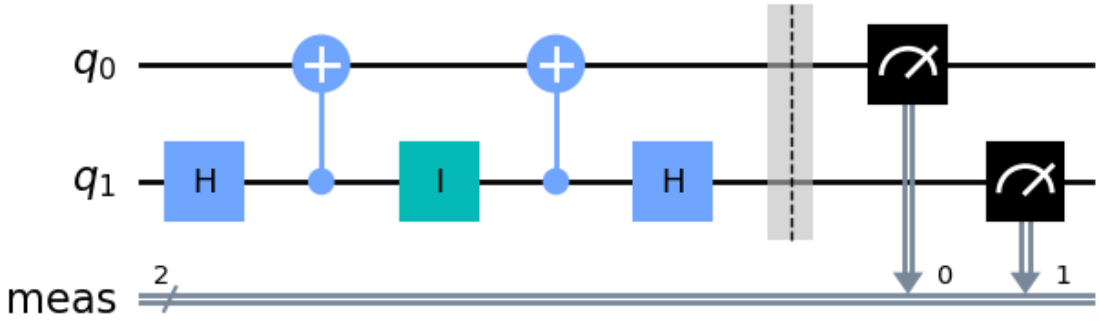
Si può facilmente notare che i vari istogramma blu sono diversi dai loro corrispettivi verdi; non a caso, sugli istogrammi blu nonostante ci sia un picco sulla decodifica corretta (l'unica presente negli istogrammi verdi), in alcuni casi vengono generate decodifiche errate, questo è dovuto al fatto che il computer quantistico è soggetto a interferenze e non sempre é in grado di fornire risultati esatti: di conseguenza ci si affida alla teoria della probabilità per poter dire qual é il risultato, e, nel tipico stile degli algoritmi randomizzati, si esegue il programma per più volte (nel caso di questo esperimento 1024).

Porta identità…

…ossia il caso 1 nel capitolo "Funzionamento". Infatti, dopo aver creato lo stato *entangled*, Alice invia il messaggio "00" e quindi *encode()* applicherà la porta **identità**, tramite il comando ***sdc.i***.

[7]:
```
#Alice vuole mandare i bit 00 a Bob
sdc = createEntangled()
sdc = encode(sdc,"00") #Alice invia
sdc = decode(sdc) #Bob riceve
sdc.measure_all()
sdc.draw('mpl')
```

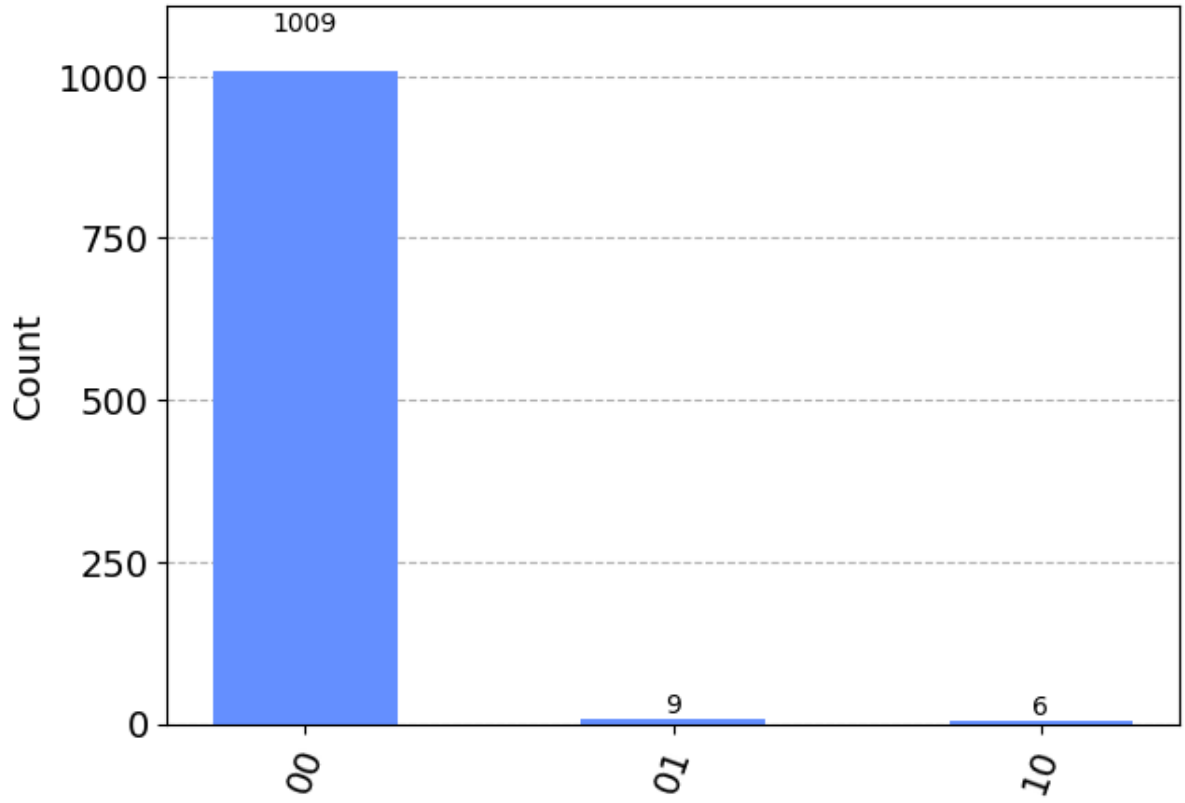
Di seguito il disegno del circuito:



L'istogramma restituito dall'esecuzione su hardware quantistico:

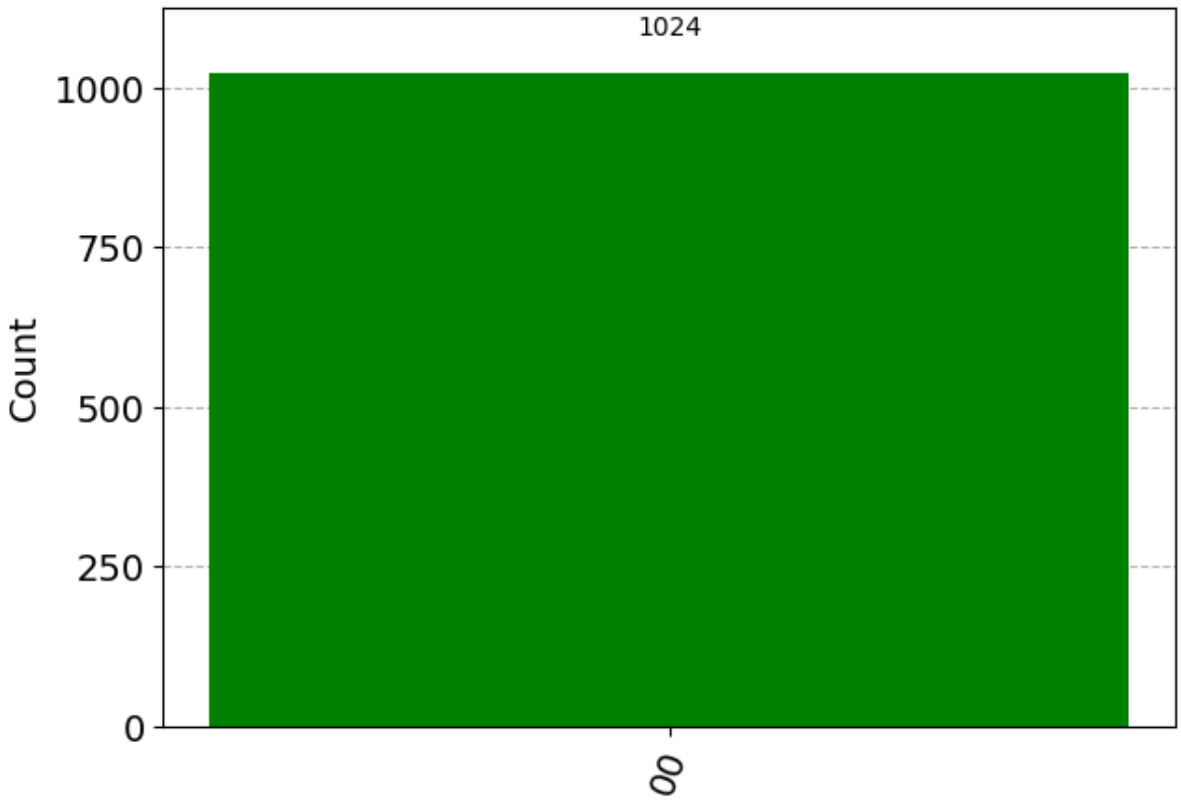
```
[8]:  device_result = misurazioniReal(sdc)
      plot_histogram(device_result)
```

```
chji2flnopt07g1llim0
Job Status: job has successfully run
```



…vs l'istogramma restituito dell'esecuzione sul simulatore:

```
[9]: device_result = misurazioniSim(sdc)
     plot_histogram(device_result, color='green')
```

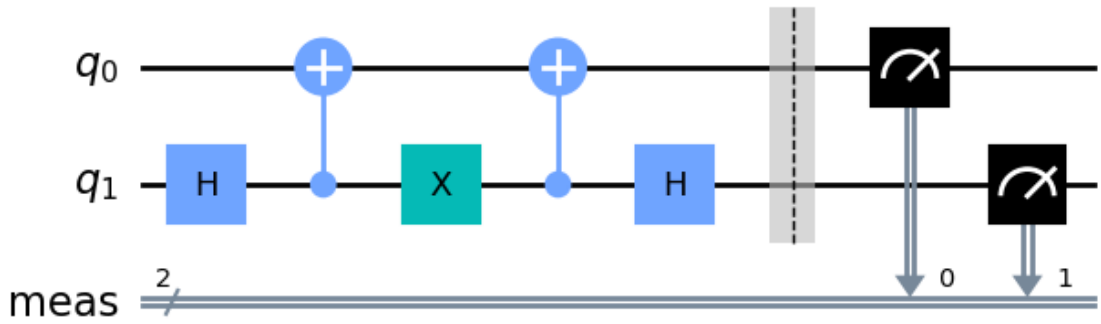


Porta X…

…ossia il caso 2 della sezione "Funzionamento". Infatti, dopo aver creato lo stato *entangled*, Alice invia il messaggio "01" e quindi *encode()* applicherà la porta **X**, tramite il comando ***sdc.x***.

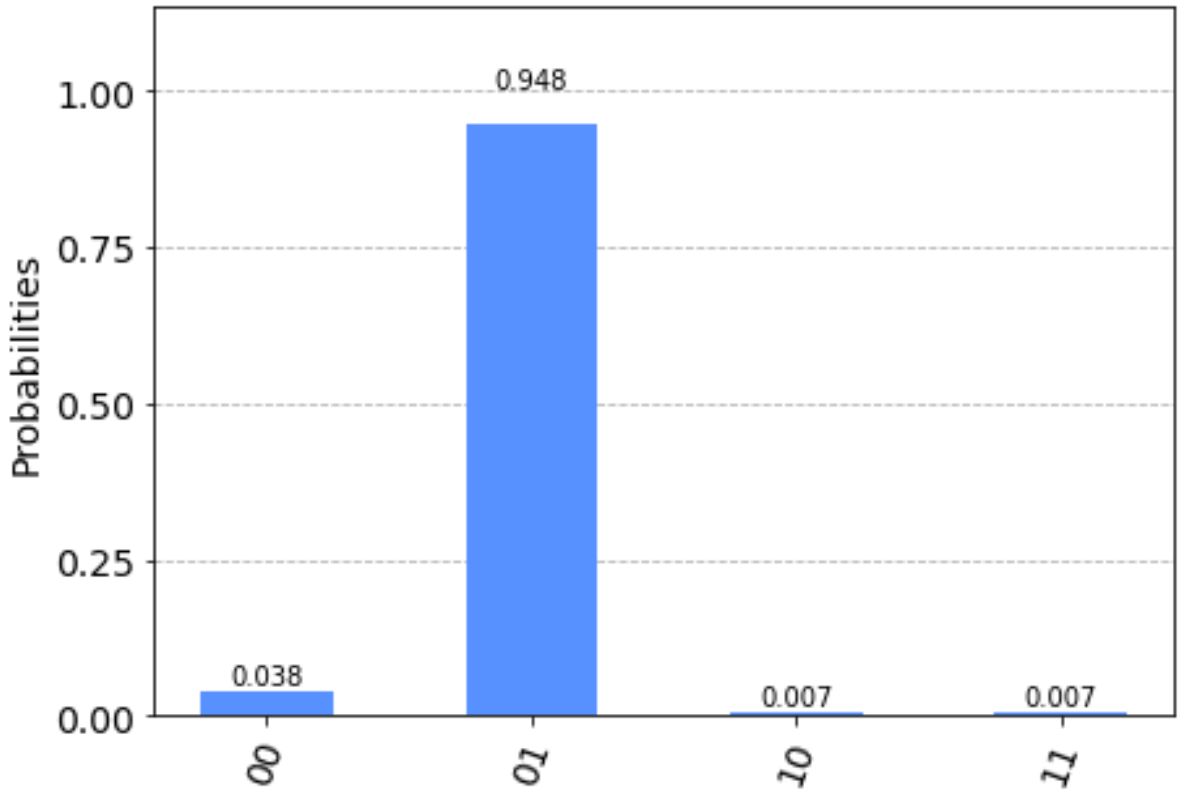
Di seguito il disegno del circuito:

```
[10]: #Alice vuole mandare i bit 01 a Bob
      sdc = createEntangled()
      sdc = encode(sdc,"01") #Alice invia
      sdc = decode(sdc) #Bob riceve
      sdc.measure_all()
      sdc.draw('mpl')
```



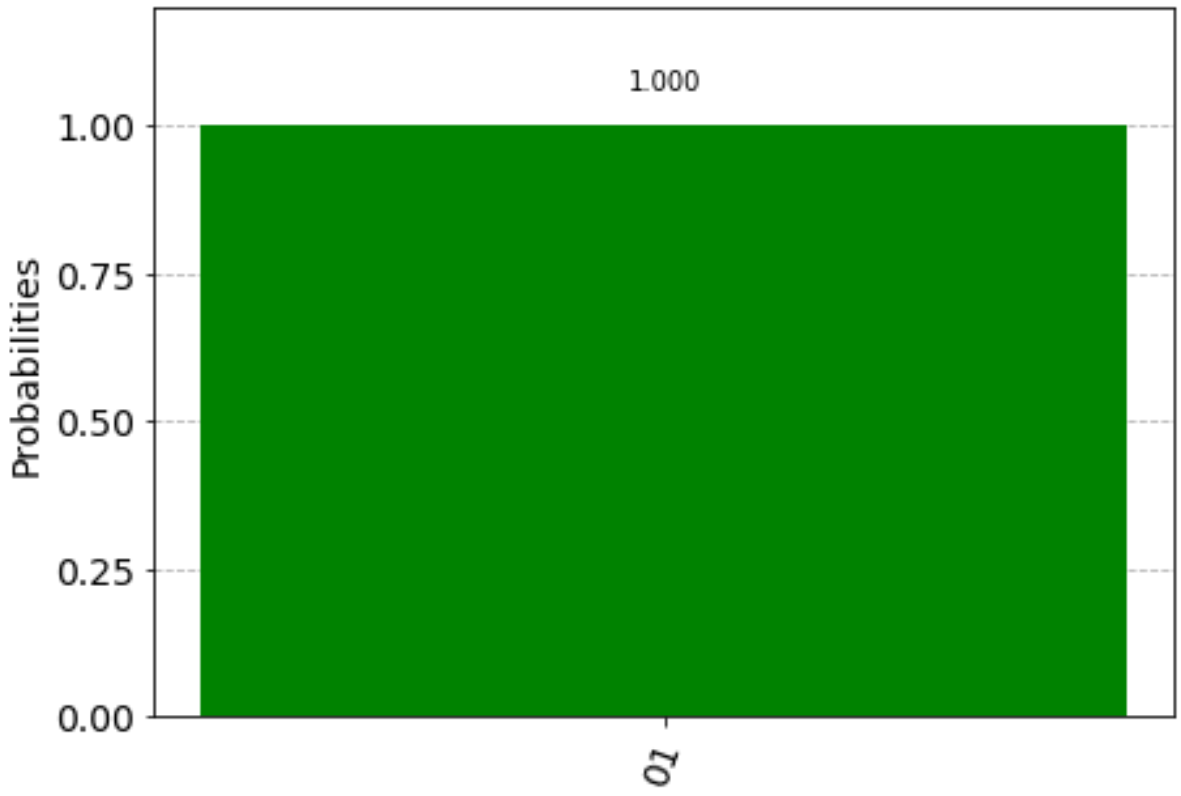
L’istogramma restituito dell’esecuzione su hardware quantistico:

```
[11]: device_result = misurazioniReal(sdc)
      plot_histogram(device_result)
```



…vs l’istogramma derivato dall’esecuzione sul simulatore:

```
[*]: device_result = misurazioniSim(sdc)
     plot_histogram(device_result, color='green')
```

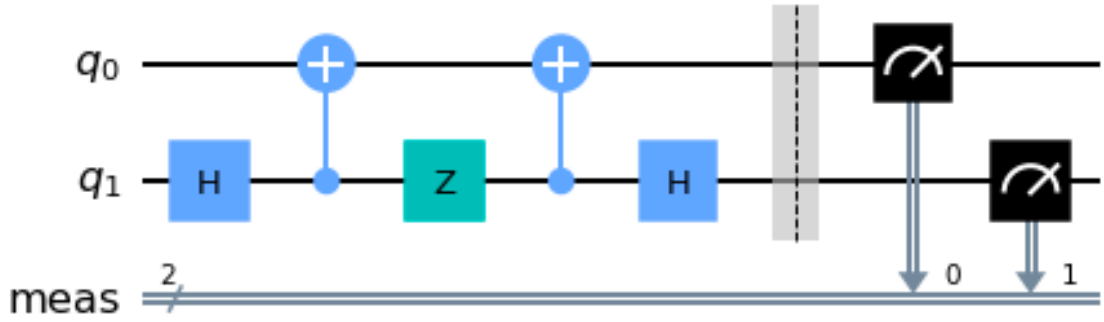


Porta Z…

…ossia il caso 3 della sezione “Funzionamento”. Infatti, dopo aver creato lo stato *entangled*, Alice invia il messaggio “10” e quindi *encode()* applicherà la porta **Z**, tramite il comando ***sdc.z***.

Di seguito il disegno del circuito:

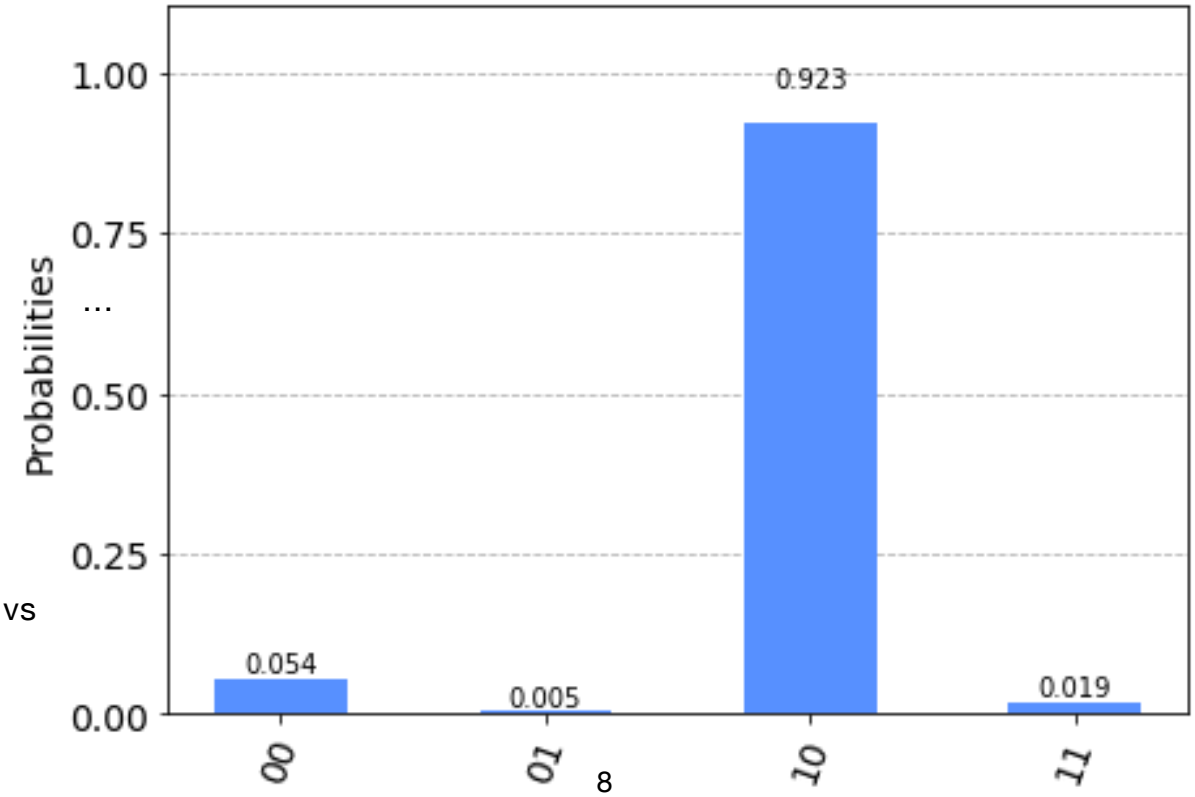
```
[*]: #Alice vuole mandare i bit 10 a Bob
     sdc = createEntangled()
     sdc = encode(sdc,"10") #Alice invia
     sdc = decode(sdc) #Bob riceve
     sdc.measure_all()
     sdc.draw('mpl')
```



L’istogramma restituito dell’esecuzione su hardware quantistico:

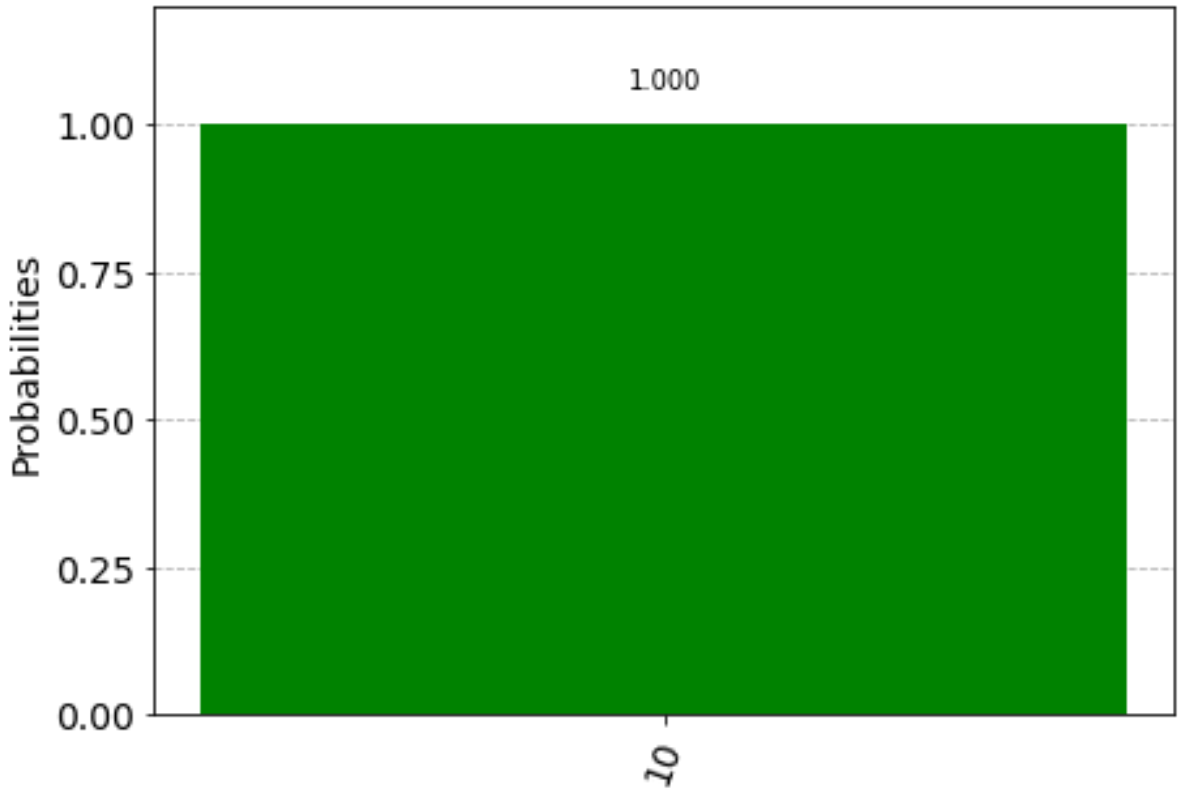
```
[*]: #Alice vuole mandare i bit 10 a Bob
     sdc = createEntangled()
     sdc = encode(sdc,"10") #Alice invia
     sdc = decode(sdc) #Bob riceve
     sdc.measure_all()
     sdc.draw('mpl')
```

vs



… vs l'istogramma derivato dall'esecuzione sul simulatore:

```
[*]: device_result = misurazioniSim(sdc)
     plot_histogram(device_result, color='green')
```

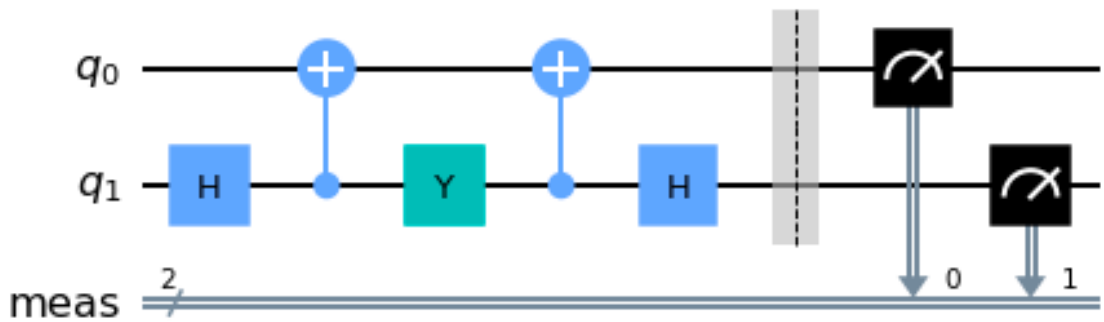


Porta iY…

…ossia il caso 4 della sezione "Funzionamento". Infatti, dopo aver creato lo stato *entangled*, Alice invia il messaggio "11" e quindi *encode()* applicherà la porta **iY**, tramite il comando ***sdc.y***.

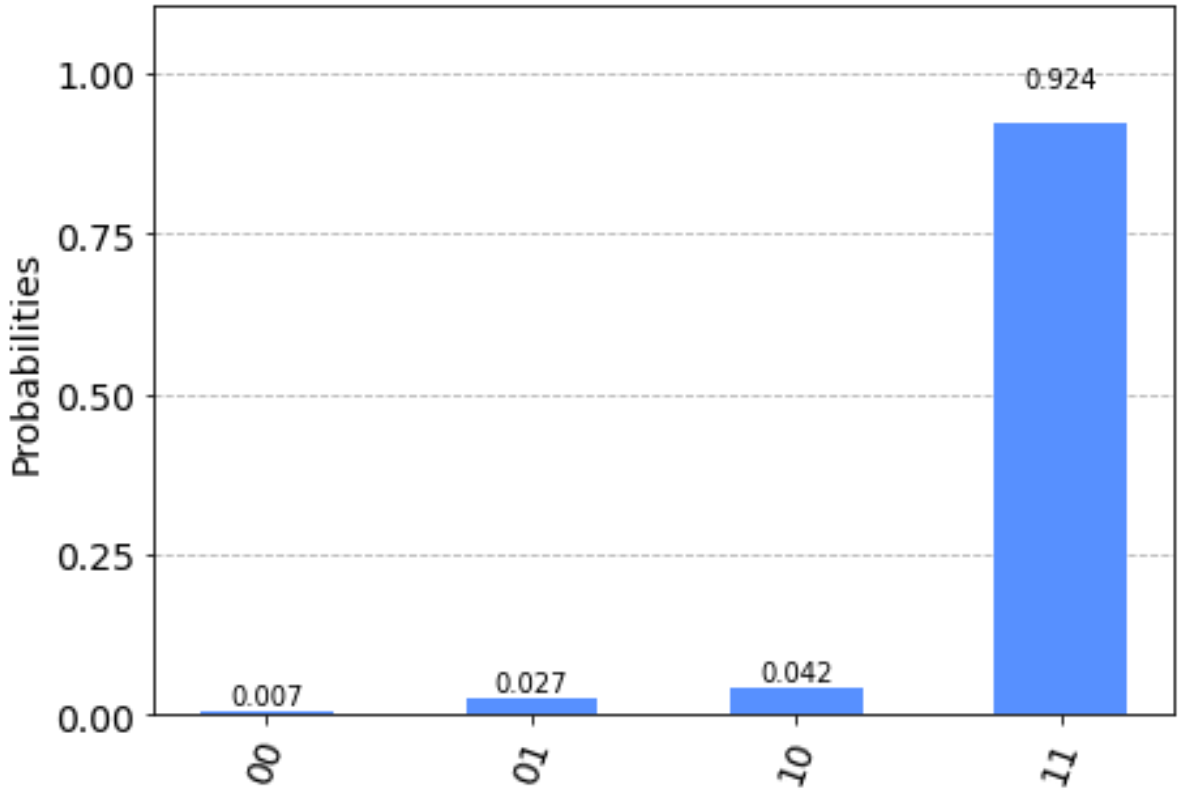
Di seguito il disegno del circuito:

```
[*]: #Alice vuole mandare i bit 11 a Bob
     sdc = createEntangled()
     sdc = encode(sdc,"11") #Alice invia
     sdc = decode(sdc) #Bob riceve
     sdc.measure_all()
     sdc.draw('mpl')
```



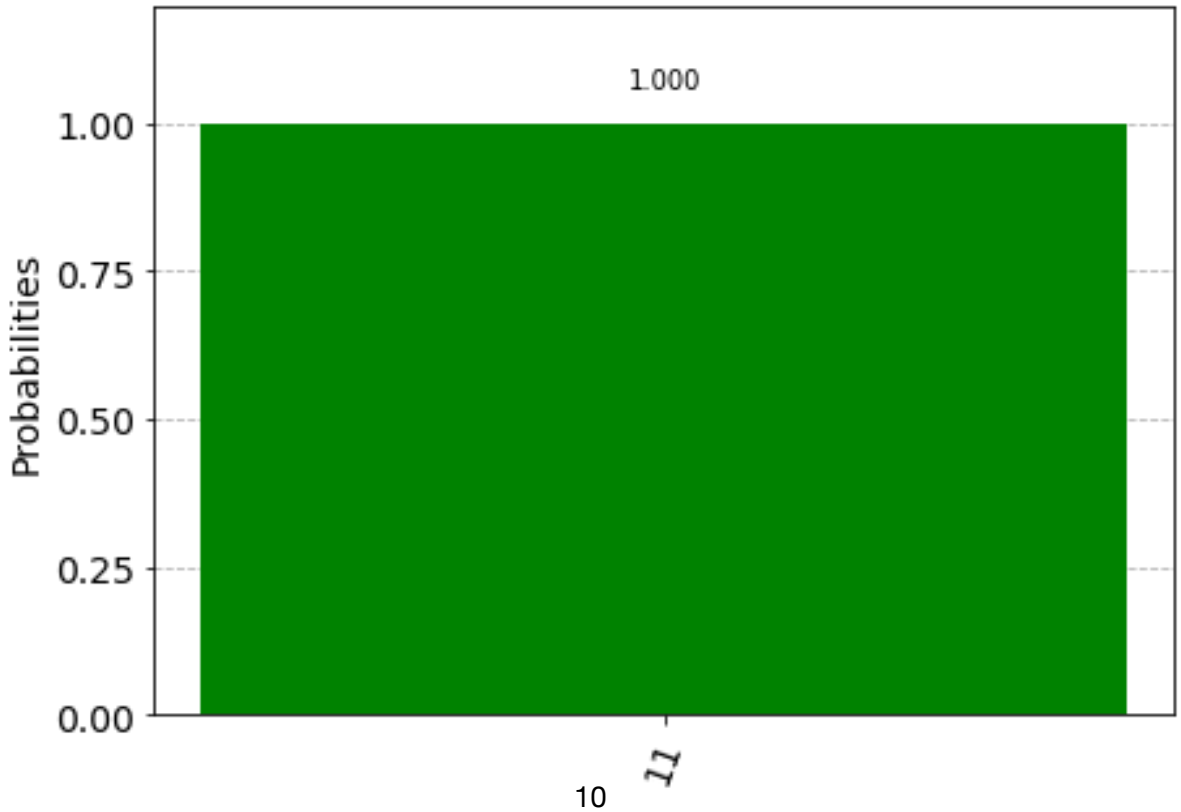
L’istogramma restituito dell’esecuzione su hardware quantistico:

```
[*]: device_result = misurazioniReal(sdc)
     plot_histogram(device_result)
```



…vs l’istogramma derivato dall’esecuzione sul simulatore:

```
[*]: device_result = misurazioniSim(sdc)
     plot_histogram(device_result, color='green')
```



Considerazioni finali 🧠

Gli istogrammi dei dati raccolti coincidono coi dati attesi dal protocollo. Inoltre, gli istogrammi dei dati simulati hanno dato al 100% risultati corretti e conformi alla teoria. Infine, come da aspettazione, i dati raccolti dall'hardware quantistico sono stati soggetti a interferenze e non hanno sempre dato risultati ineccepibili: comunque si è ottenuta una percentuale di correttezza maggiore del 90% dei casi, quindi nel complesso il *superdense coding* è riuscito con ottimi risultati.

L'unica nota negativa è il fatto dei tempi di attesa sul portale IBM online, che la maggior parte delle volte porta a dover aspettare una coda nell'ordine delle ore (in questo caso quasi 5) per un lavoro nell'ordine dei secondi (in questo caso 3).