Quantum Entanglement

Quantum entanglement is a quantum mechanical phenomenon in which the quantum states of two or more objects have to be described with reference to each other, even though the individual objects may be spatially separated.
For example, it is possible to prepare two particles in a single quantum state such that when one is observed to be spin-up, the other one will always be observed to be spin-down and vice versa, this despite the fact that it is impossible to predict, according to quantum mechanics, which set of measurements will be observed.

Quantum entanglement has applications in the emerging technologies of quantum computing and quantum cryptography, and has been used to realize quantum teleportation experimentally.

Bell State or Entanglement c

The Bell states are four specific maximally entangled quantum states of two qubits. They are in a superposition of 0 and 1--that is, a linear combination of the two states. Their entanglement means the following:

The qubit held by Alice (subscript "A") can be 0 as well as 1. If Alice measured her qubit in the standard basis, the outcome would be perfectly random, either possibility 0 or 1 having probability 1/2. But if Bob (subscript "B") then measured his qubit, the outcome would be the same as the one Alice got. So, if Bob measured, he would also get a random outcome on first sight, but if Alice and Bob communicated, they would find out that, although their outcomes seemed random, they are perfectly correlated.

$$\textbf{For initial state } \{00\}, |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |$$

$$\textbf{For initial state } \{10\}, |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |$$

$$\textbf{For initial state } \{01\}, |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |$$

$$\textbf{For initial state } \{11\}, |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |$$

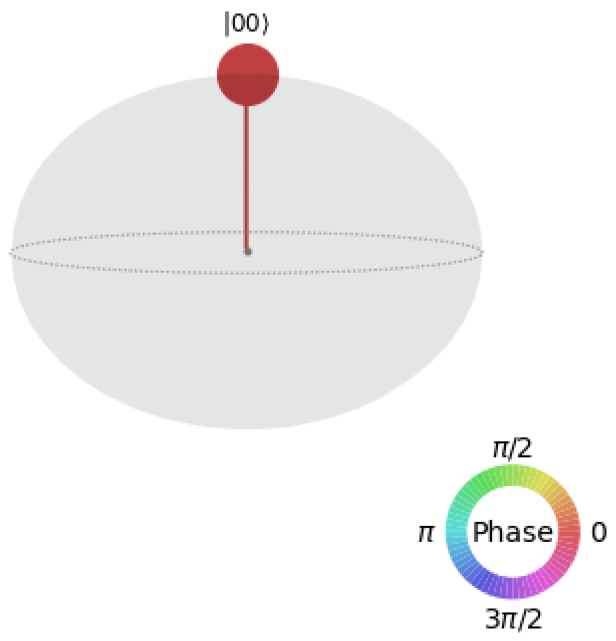
In [1]:

```
%matplotlib inline
# Importing standard Qiskit libraries and configuring accoun
t
from qiskit import QuantumCircuit, execute, Aer, IBMQ, Quant
umRegister, ClassicalRegister
from qiskit.tools.monitor import job_monitor
from qiskit.visualization import plot_bloch_multivector
from qiskit.visualization import plot_state_qsphere
from math import pi
from qiskit.quantum_info import Statevector
from IPython.core.display import Image, display
import numpy as np
```

lets initialize the qubits to {00} with statevector

In [2]:
```
sv = Statevector.from_label('00')
plot_state_qsphere(sv.data)
```

Out[2]:

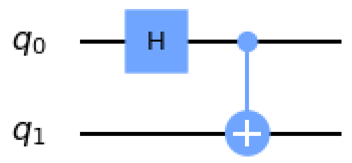


lets define our circuit of entanglement

In [3]:
```
entanglement1 = QuantumCircuit(2)
```

```
entanglement1.h(0)
entanglement1.cx(0,1)
entanglement1.draw('mpl')
```

Out[3]:

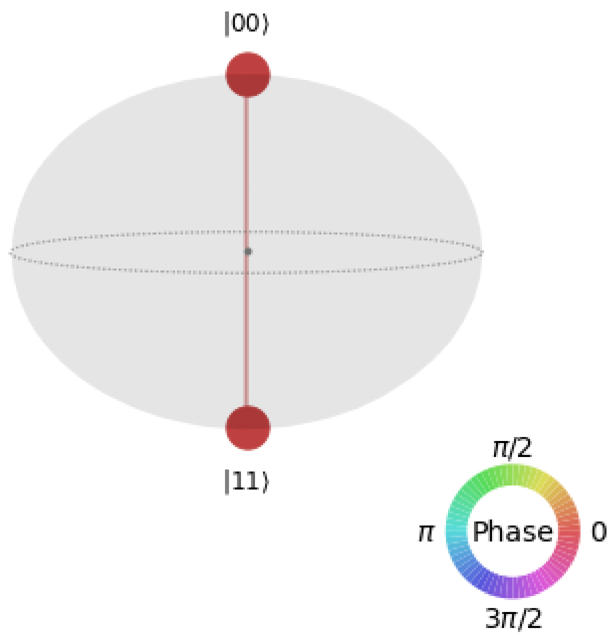


In [4]:

```
new_sv = sv.evolve(entanglement1)
print(new_sv)
plot_state_qsphere(new_sv.data)
```

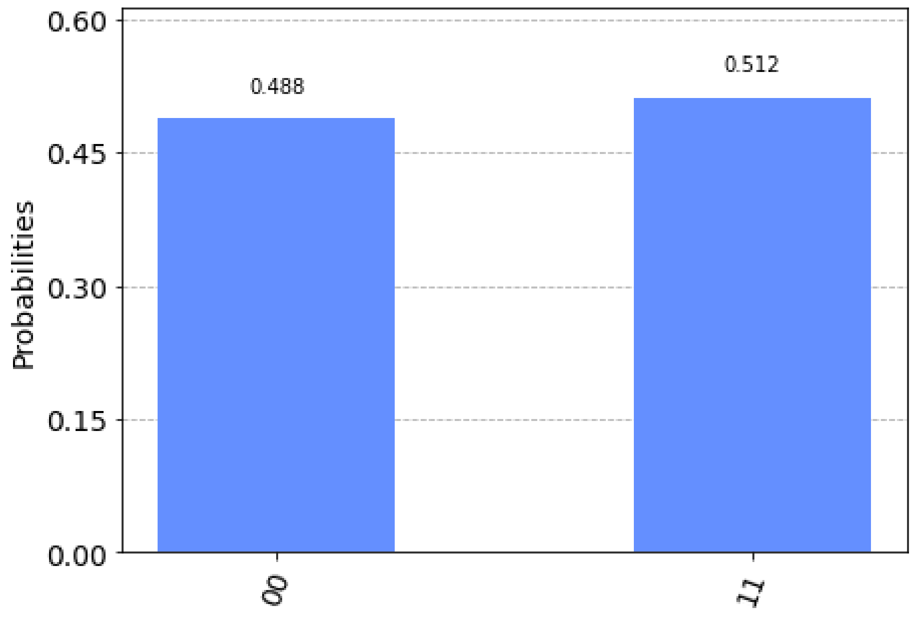
```
Statevector([0.70710678+0.j, 0.        +0.j, 0.        +0.
j,
             0.70710678+0.j],
            dims=(2, 2))
```

Out[4]:



In [5]:
```
counts = new_sv.sample_counts(shots=1000)

from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

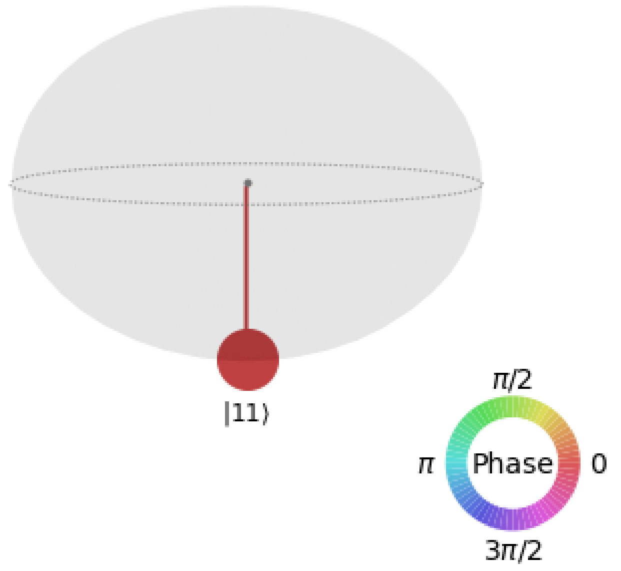
Out[5]:



lets initialize the qubits to {11} with statevector for 4^{th} Bell state

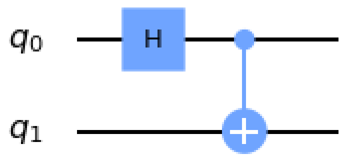
In [6]:
```
sv = Statevector.from_label('11')
plot_state_qsphere(sv.data)
```

Out[6]:



In [7]:
```
entanglement4 = QuantumCircuit(2)
entanglement4.h(0)
entanglement4.cx(0,1)
entanglement4.draw('mpl')
```

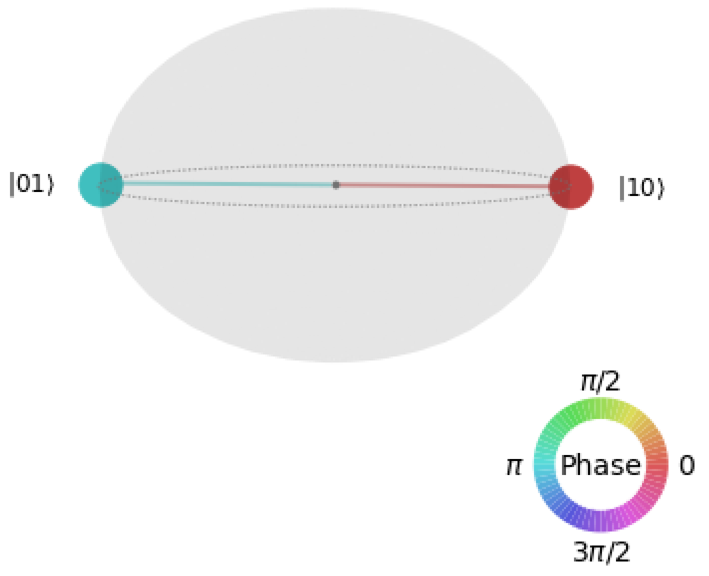
Out[7]:



In [8]:
```
new_sv = sv.evolve(entanglement4)
print(new_sv)
plot_state_qsphere(new_sv.data)
```

```
Statevector([ 0.        +0.j, -0.70710678+0.j,  0.70710678+
0.j,
              0.        +0.j],
            dims=(2, 2))
```

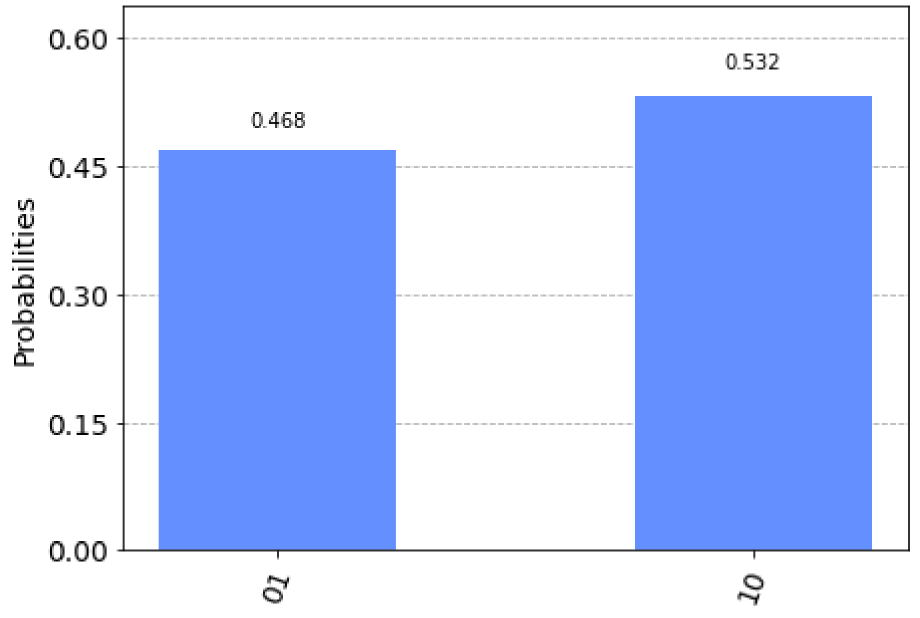
Out[8]:



In [9]:
```
counts = new_sv.sample_counts(shots=1000)
```

```
from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

Out[9]:

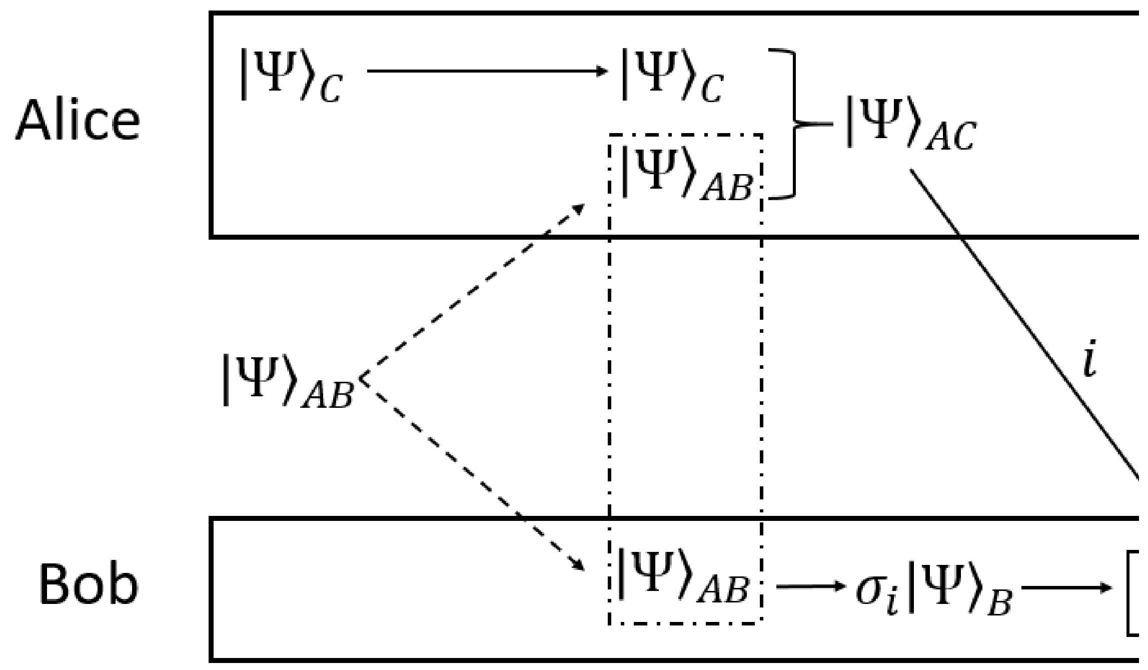


Quantum Teleportation

In quantum teleportation, the properties of quantum entanglement are used to send a spin state (qubit) between observers without physically moving the involved particle. The particles themselves are not really teleported, but the state of one particle is destroyed on one side and extracted on the other side, so the information that the state encodes is communicated. The process is not instantaneous, because information must be communicated classically between observers as part of the process. The usefulness of quantum teleportation lies in its ability to send quantum information arbitrarily far distances without exposing quantum states to thermal decoherence from the environment or other adverse effects.

Alice wants to send quantum information to Bob. Specifically, suppose she wants to send the qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. This entails passing on information about α and β to Bob. By taking advantage of two classical bits and an entangled qubit pair, Alice can transfer her state |ψ⟩ to Bob. We call this teleportation because, at the end, Bob will have |ψ⟩ and Alice won't anymore.

In [10]:
```
display(Image('https://ds055uzetaobb.cloudfront.net/brioche/uploads/ZmGbUHYGb7-teleport.png', width=800, unconfined=True))
```



Lets considered Alice qubit be

$$\sqrt{0.70}|0\rangle + \sqrt{0.30}|1\rangle$$

which is teleported to Bob

step 1: Initialization of Alice qubit to given state vec

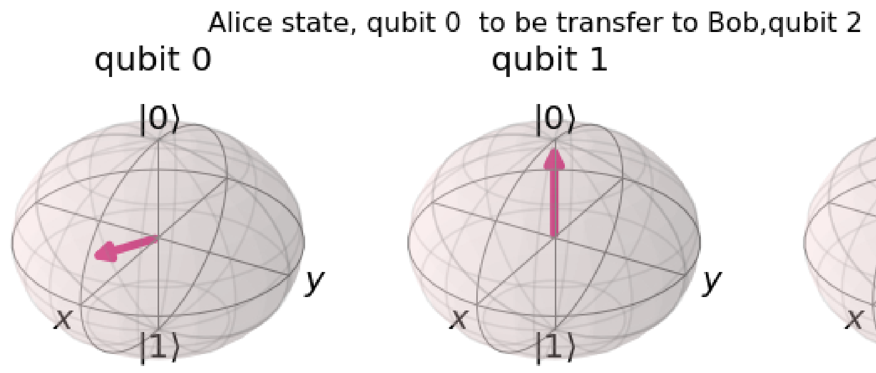
In [11]:
```
qreg_q = QuantumRegister(3, 'q')
creg= ClassicalRegister(1,'c')
creg_x = ClassicalRegister(1,'creg_x') # first classical register
creg_z = ClassicalRegister(1,'creg_z') # two classical register
Teleportation = QuantumCircuit(qreg_q,creg, creg_x,creg_z)
state_vector=[np.sqrt(0.7)*complex(1,0),np.sqrt(0.3)*complex(1,0)]
Teleportation.initialize(state_vector,(qreg_q[0]))
Teleportation.barrier()
Teleportation.draw()
```

Out[11]:
```
        ┌─────────────────────────────┐ ░ 
   q_0: ┤ initialize(0.83666,0.54772) ├─░─
        └─────────────────────────────┘ ░ 
   q_1: ────────────────────────────────░─
                                        ░ 
   q_2: ────────────────────────────────░─
                                        ░ 
   c: 1/══════════════════════════════════

creg_x: 1/══════════════════════════════════
```

```
creg_z: 1/═══════════════════════════════════════
```

In [12]:
```python
copy1 = Teleportation.copy()
state = Statevector.from_instruction(copy1)
plot_bloch_multivector(state, title=" Alice state, qubit 0
 to be transfer to Bob,qubit 2")
```

Out[12]:



We apply cx and Hadamard gate to q1 and q2 for Entanglement

In [13]:
```python
Teleportation.h(1)
Teleportation.cx(1,2)
Teleportation.barrier()
Teleportation.cx(0,1)
Teleportation.h(0)
Teleportation.draw()
```

Out[13]:

```
q_0: ┤ initialize(0.83666,0.54772) ├░───────
                                    ░  ┌───┐
```

```
     q_1: ──────────────────────────────────────────────░─┤ H ├──■
                                                        ░ └───┘  │
     q_2: ──────────────────────────────────────────────░───────┤ X
                                                        ░       └──
       c: 1/═══════════════════════════════════════════════════════

  creg_x: 1/═══════════════════════════════════════════════════════

  creg_z: 1/═══════════════════════════════════════════════════════
```

In [14]:
```
Teleportation.barrier()
```

Out[14]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x710962bb
e0>
```

Step 2: Applying a Bell measurement on Alice's qu

In [15]:
```
Teleportation.measure(0,2)
Teleportation.measure(1,1)
Teleportation.barrier()
Teleportation.draw()
```

Out[15]:
```
          ┌─────────────────────────────┐ ░
     q_0: ┤ initialize(0.83666,0.54772) ├─░────────────────
          └─────────────────────────────┘ ░ ┌───┐
     q_1: ────────────────────────────────░─┤ H ├──■──
                                          ░ └───┘┌─┴─┐
     q_2: ────────────────────────────────░──────┤ X ├
                                          ░      └───┘
       c: 1/═══════════════════════════════════════════
```

```
creg_x: 1/═══════════════════════════════════════════════════

creg_z: 1/═══════════════════════════════════════════════════
```

Step 3: Applying classically controlled operations c

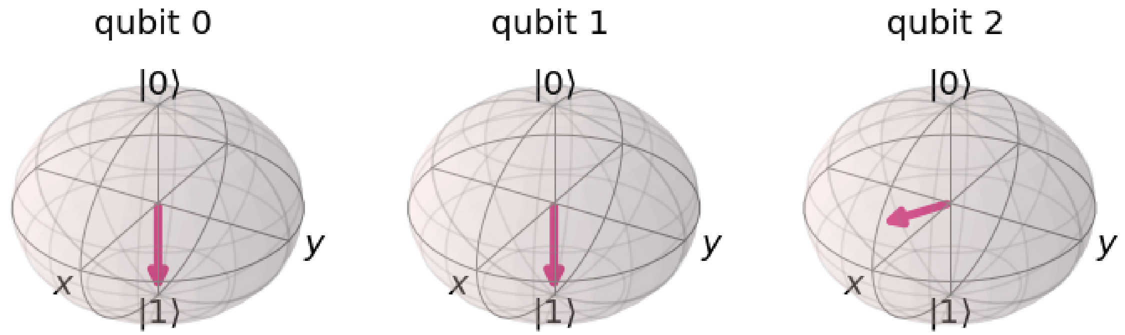
In [16]:
```
Teleportation.x(2).c_if(creg_x,1)
Teleportation.z(2).c_if(creg_z,1)
Teleportation.draw()
```

Out[16]:
```
        ┌─────────────────────────────┐ ░
   q_0: ┤ initialize(0.83666,0.54772) ├─░──────────────
        └─────────────────────────────┘ ░ ┌───┐
   q_1: ────────────────────────────────░─┤ H ├──■───
                                        ░ └───┘┌─┴─┐
   q_2: ────────────────────────────────░──────┤ X ├
                                        ░      └───┘
   c: 1/═════════════════════════════════════════════

creg_x: 1/═══════════════════════════════════════════

creg_z: 1/═══════════════════════════════════════════

«
«     q_0: ────────────────
«
«     q_1: ────────────────
«          ┌───┐ ┌───┐
«     q_2: ┤ X ├─┤ Z ├─
«                   ┌───┐  ┌───┐
«         c: 1/═════╪═════════╪══════
«                ┌───┴───┐    ║
«creg_x: 1/═╡ = 1 ╞════╪══════
«                └───────┘┌───┴───┐
«creg_z: 1/═══════════╡ = 1 ╞═
«                         └───────┘
```

Step 4: Measuring q2 will determite the Quantum t

In [17]:
```python
copy2 = Teleportation.copy()
# copy2.measure(2,creg)
statevector= Aer.get_backend('statevector_simulator')
psi= execute(copy2,statevector).result().get_statevector()
plot_bloch_multivector(psi)
```

Out[17]:



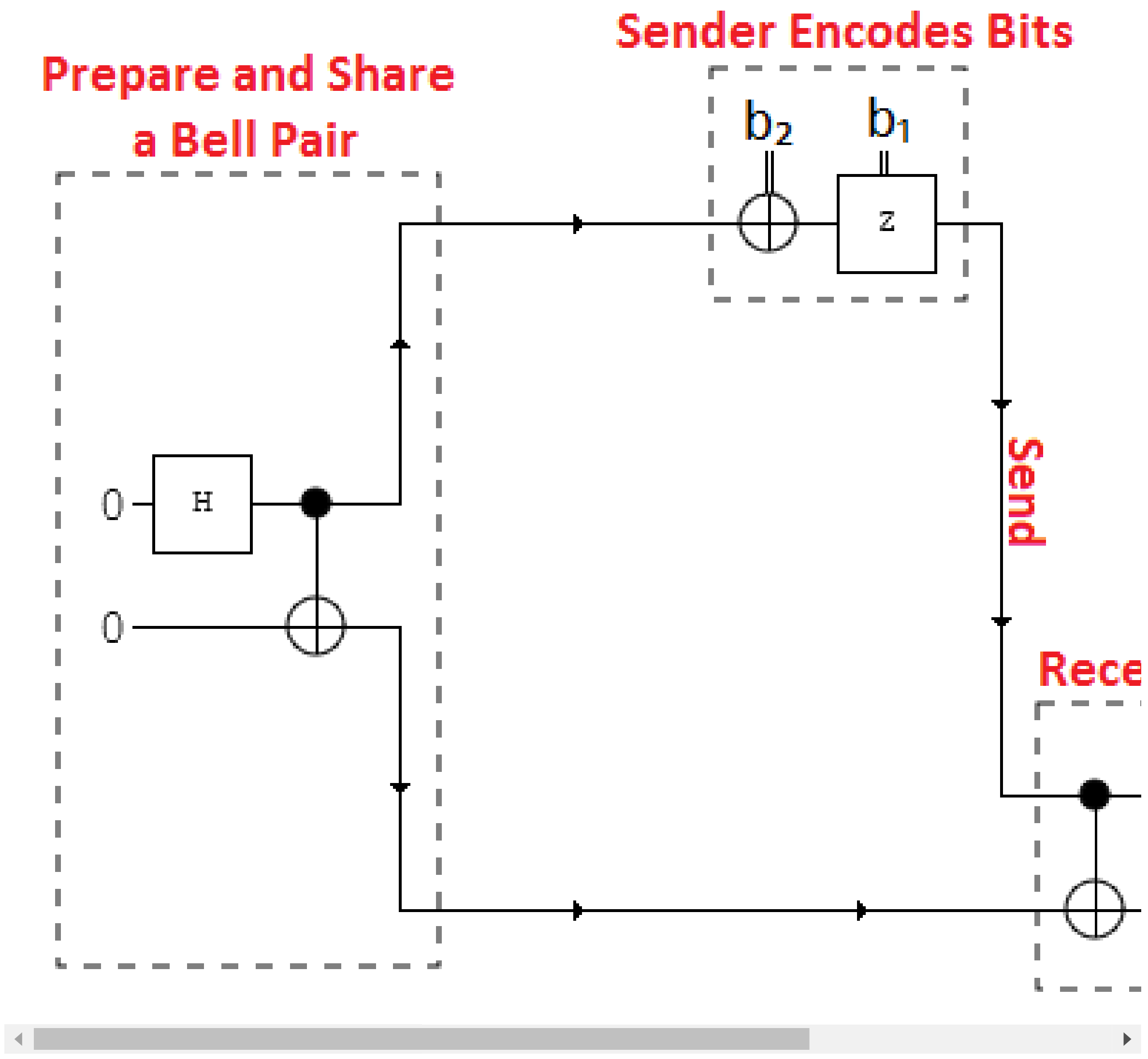
Here the qubit 2 is measured just like the Alice state by the bob which verify the Quantum Teleportation.

Superdensed Coding

In quantum information theory, superdense coding (or dense coding) is a quantum communication protocol to transmit two classical bits of information (i.e., either 00, 01, 10 or 11) from a sender (often called Alice) to a receiver (often called Bob), by sending only one qubit from Alice to Bob, under the assumption of Alice and Bob pre-sharing an entangled state.

Teleportation	Superdense coding
Transmit one qubit using two classical bits	Transmit two classical bits using one qubit

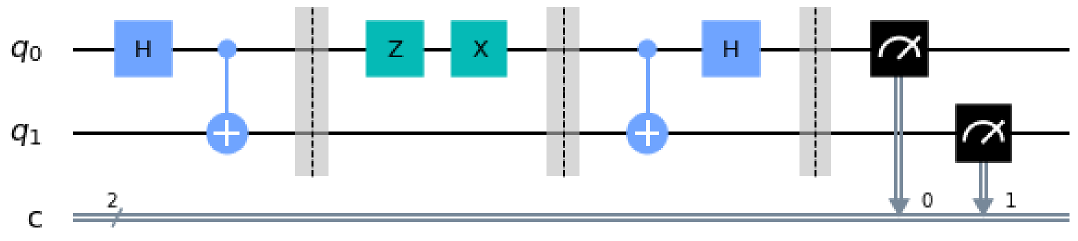
In [18]:
```
display(Image('https://upload.wikimedia.org/wikipedia/commons/b/b7/Superdense_coding.png', width=800, unconfined=True))
```



Suppose Alice wants to send two classical bits of information (00, 01, 10, or 11) to Bob using qubits (instead of classical bits). To do this, an entangled state (e.g. a Bell state) is prepared using a Bell circuit or gate by Charlie, a third person. Charlie then sends one of these qubits (in the Bell state) to Alice and the other to Bob. Once Alice obtains her qubit in the entangled state, she applies a certain quantum gate to her qubit depending on which two-bit message (00, 01, 10 or 11) she wants to send to Bob. Her entangled qubit is then sent to Bob who, after applying the appropriate quantum gate and making a measurement, can retrieve the classical two-bit message. Observe that Alice does not need to communicate to Bob which gate to apply in order to obtain the correct classical bits from his projective measurement.

In [25]:
```
qcs=QuantumCircuit(2,2)
qcs.h(0)
qcs.cx(0,1)
qcs.barrier()

#qcs.x(0)

#qcs.z(0)

qcs.z(0)
qcs.x(0)

qcs.barrier()

qcs.cx(0,1)
qcs.h(0)
qcs.barrier()

qcs.measure(0,0)
qcs.measure(1,1)
qcs.draw('mpl')
```

Out[25]:

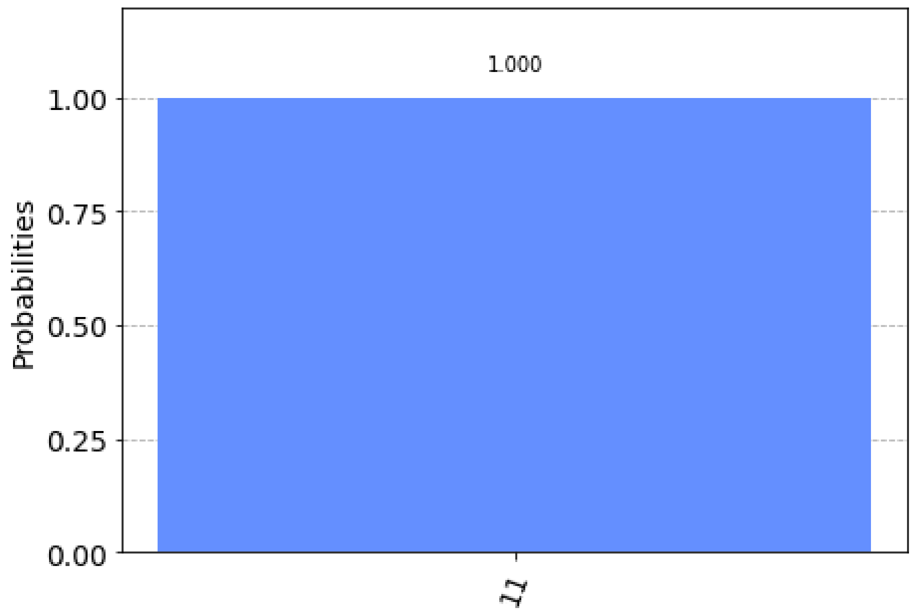


Here if we dont use any gates between 2nd and 3rd barrier we get 00 as output. If we use x gate we get 10 as a output. If we use z gate we get 01 as a output. If we use ZX gate we get 11 as a output.

In [26]:
```
backend = Aer.get_backend('qasm_simulator')
job_sim = execute(qcs, backend, shots=1024)
sim_result = job_sim.result()

measurement_result = sim_result.get_counts(qcs)
print(measurement_result)
plot_histogram(measurement_result)
```

```
{'11': 1024}
```

Out[26]:



Here Bob gets classical bits of information (i.e., either 00, 01, 10 or 11) from a sender (often called Alice)

```
In [ ]:
```