Name: Gudur Krishna Chaitanya Roll no: 102117049 email used for
gchaitanya_be21@thapar.edu

Superdense Coding:

1.Entanglement: The process begins with the creation of an entangled pair of qubits, typically through the use of a Bell state. Each qubit is sent to one of the parties, Alice and Bob.

2.Encoding: Alice possesses the qubit she wants to send to Bob and wishes to convey two classical bits of information to him. By applying quantum gates to her qubit and exploiting the entanglement, Alice can encode the two classical bits into her qubit's state.

3.Transmission: Alice sends her qubit to Bob through a quantum communication channel. This qubit carries the encoded information due to the entanglement and the operations Alice applied.

4.Decoding: Upon receiving the qubit from Alice, Bob performs specific quantum operations on his qubit, which, combined with measurements, allow him to decode the two classical bits encoded by Alice. This decoding process extracts the classical information without directly measuring Alice's qubit state, preserving the quantum information encoded in the qubit.

00 : I 01 : X 10 : Z 11 : ZX

```
from qiskit_ibm_provider import IBMProvider

provider = IBMProvider(token='f55702335547d565b44eb80fd6708f3b82b4d0147236062

active_account = provider.active_account()

print("Active Account Details:")

print(active_account)
```

```
<ipython-input-4-2dda43cd1c73>:1: DeprecationWarning: The package qiskit_ib
  from qiskit_ibm_provider import IBMProvider
Active Account Details:
{'channel': 'ibm_quantum', 'token': 'f55702335547d565b44eb80fd6708f3b82b4d0
```

```
from qiskit import QuantumCircuit
from qiskit import transpile
from qiskit.visualization import plot_histogram
import qiskit_aer

```

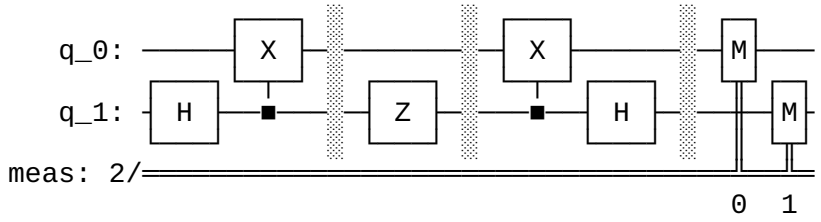
```
def create_bell_pair():
    qc = QuantumCircuit(2)
    qc.h(1)
```

```
    qc.h(1)
    qc.cx(1, 0)
    return qc
```

```
def encode_message(qc, qubit, msg):
    if len(msg) != 2 or not set(msg).issubset({"0","1"}):
        raise ValueError(f"message '{msg}' is invalid")
    if msg[1] == "1":
        qc.x(qubit)
    if msg[0] == "1":
        qc.z(qubit)
    return qc
```

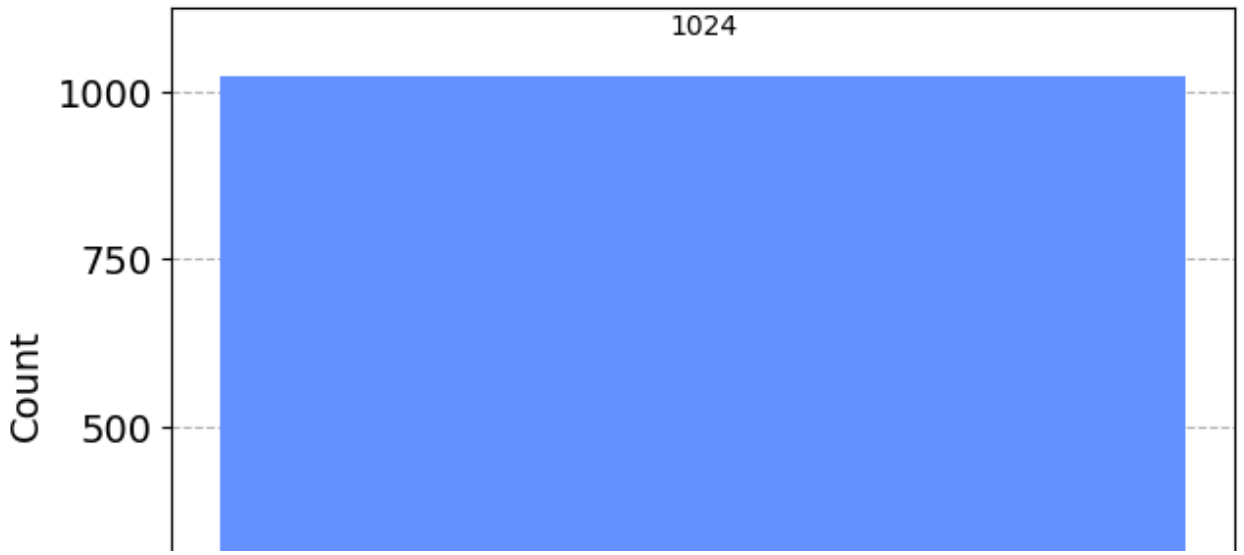
```
def decode_message(qc):
    qc.cx(1, 0)
    qc.h(1)
    return qc
```

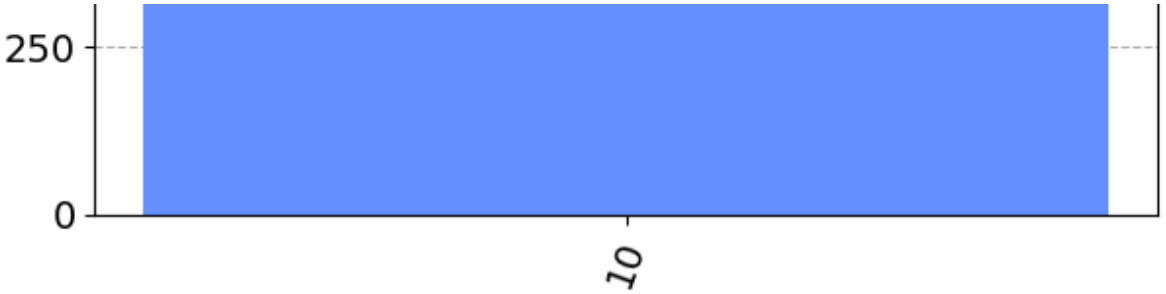
```
qc = create_bell_pair()
qc.barrier()
message = '10'
qc = encode_message(qc, 1, message)
qc.barrier()
qc = decode_message(qc)
qc.measure_all()
qc.draw(output='text',style='bw')
```




```
aer_sim = qiskit_aer.Aer.get_backend('aer_simulator')
result = aer_sim.run(qc).result()
counts = result.get_counts(qc)
plot_histogram(counts)

```




250
0
10