CSBB 311 : QUANTUM COMPUTING

LAB ASSIGNMENT 9 : SuperDense Coding

Submitted By:

Name: KARTIK MITTAL

Roll No: 221210056

Branch: CSE

Semester: 5th Sem

Group: 2

Submitted To: Dr. VS Pandey

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2024

Theory -

1. Introduction to Superdense Coding

- Superdense coding is a quantum communication protocol that allows the transmission of two classical bits of information using a single quantum bit (qubit) with the help of quantum entanglement. It is a fundamental demonstration of quantum information theory, showcasing the unique advantages of quantum communication over classical methods.

2. Key Elements of Superdense Coding

- **Entanglement**: The protocol relies on a shared entangled state between the sender (Alice) and the receiver (Bob), typically a Bell state.
- **Quantum Operations**: Alice performs specific quantum operations (Pauli operators) on her qubit to encode two classical bits of information.
- **Measurement**: Bob decodes the message by performing a joint measurement on both qubits to retrieve the classical information.

3. Workflow of Superdense Coding

- **Entanglement Sharing**: Alice and Bob begin by sharing a pair of entangled qubits. Bob retains one qubit, and Alice takes the other.
- **Encoding the Message**: To send two classical bits, Alice applies a quantum operation based on the message she wishes to encode.
    - `00`: No operation (Identity).
    - `01`: Apply XXX (bit-flip).
    - `10`: Apply ZZZ (phase-flip).
    - `11`: Apply $X \cdot ZX \cdot ZX \cdot Z$ (bit-flip and phase-flip).
- **Transmission**: Alice sends her qubit to Bob after encoding.
- **Decoding the Message**: Upon receiving Alice's qubit, Bob performs a joint measurement (Bell-state measurement) on both qubits to determine the original two-bit message.

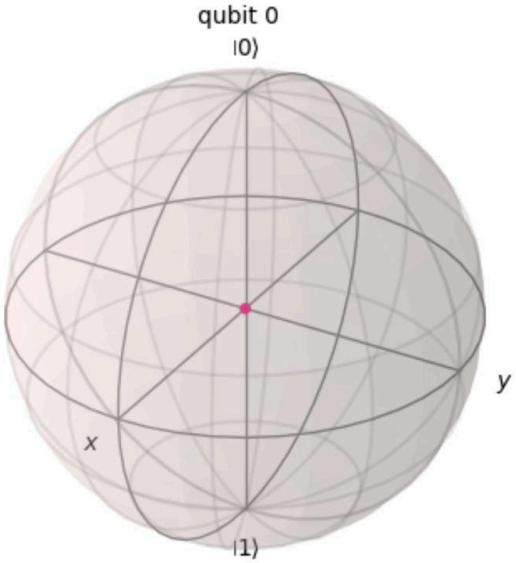
Code (SuperDense Coding) -

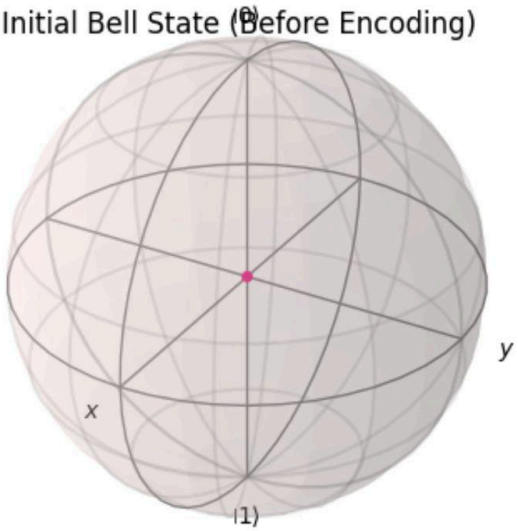
```python
from qiskit import QuantumCircuit
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram, plot_bloch_multivector
import matplotlib.pyplot as plt

# Step 1: Create a Bell State (entangled state)
def create_bell_pair():
    qc = QuantumCircuit(2)
    qc.h(0) # Apply Hadamard gate to qubit 0
    qc.cx(0, 1) # Apply CNOT gate (control: qubit 0, target: qubit 1)
    return qc

# Step 2: Encode the message (classical bits) onto the Bell pair
def encode_message(qc, message):
    if message == "00":
        pass # Do nothing
    elif message == "01":
        qc.x(0) # Apply X gate
    elif message == "10":
        qc.z(0) # Apply Z gate
    elif message == "11":
        qc.x(0)
        qc.z(0)
    else:
        raise ValueError("Message must be one of '00', '01', '10', or '11'")

# Step 3: Decode the message by reversing the entanglement
def decode_message(qc):
    qc.cx(0, 1) # Apply CNOT gate (control: qubit 0, target: qubit 1)
    qc.h(0) # Apply Hadamard gate to qubit 0
```

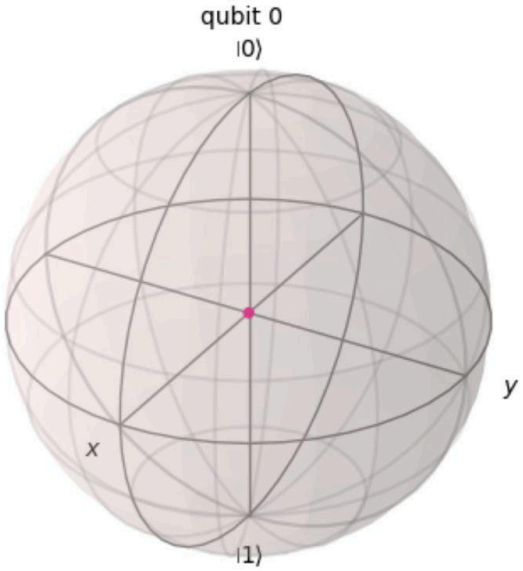
```python
# Main function to demonstrate superdense coding
def superdense_coding(message):
    if message not in ["00", "01", "10", "11"]:
        raise ValueError("Message must be one of '00', '01', '10', or '11'")

    # Step 1: Create a Bell pair
    qc = create_bell_pair()

    # Visualize the initial state of the qubits in the Bell state
    print("Initial Bell state (before encoding):")
    backend = Aer.get_backend('statevector_simulator')
    job = backend.run(qc)
    result = job.result()
    statevector = result.get_statevector(qc)
    plot_bloch_multivector(statevector)
    plt.title("Initial Bell State (Before Encoding)")
    plt.show()

    # Step 2: Encode the message
    encode_message(qc, message)

    # Visualize the state after encoding the classical message
    job = backend.run(qc)
    result = job.result()
    statevector = result.get_statevector(qc)
    plot_bloch_multivector(statevector)
    plt.title(f"State After Encoding Message {message}")
    plt.show()
```

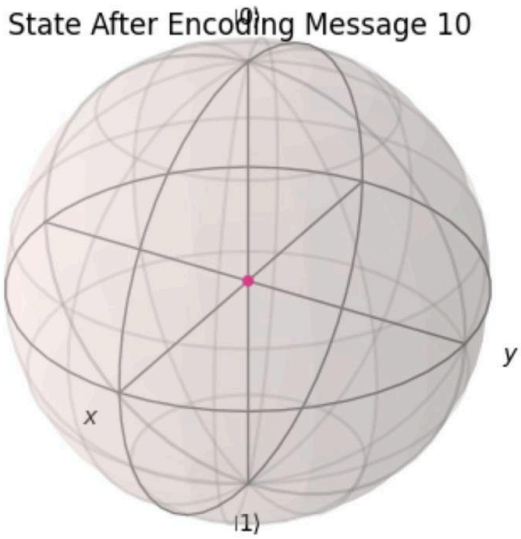
```
    # Step 3: Decode the message
    decode_message(qc)

    # Visualize the state after decoding (before measurement)
    job = backend.run(qc)
    result = job.result()
    statevector = result.get_statevector(qc)
    plot_bloch_multivector(statevector)
    plt.title(f"State After Decoding Message {message}")
    plt.show()

    # Step 4: Measure the qubits
    qc.measure_all()

    # Simulate the circuit and get the results
    simulator = Aer.get_backend('qasm_simulator')
    job = simulator.run(qc, shots=1024)
    result = job.result()
    counts = result.get_counts(qc)

    return qc, counts

# Test the superdense coding protocol
message = "10" # Replace with "00", "01", "10", or "11"
qc, counts = superdense_coding(message)

# Display the quantum circuit and the result
print(f"Message sent: {message}")
print(f"Measurement result: {counts}")
qc.draw("mpl")

# Plot histogram of results
plot_histogram(counts)
plt.show()
```

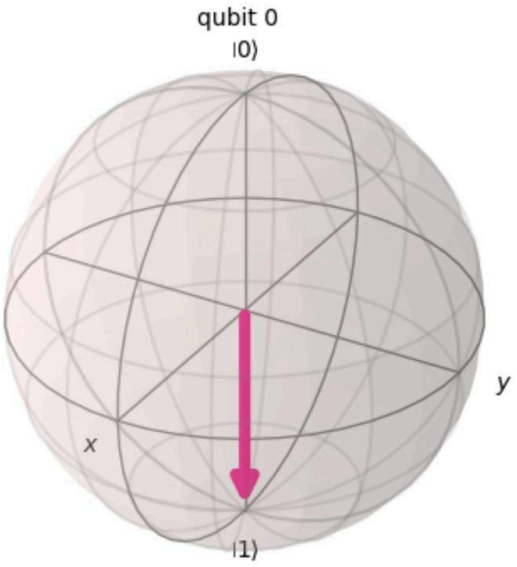
Output -

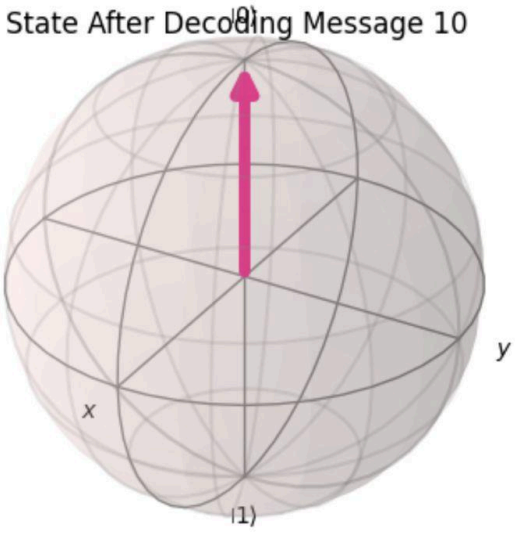

qubit 0
|0⟩
y
x
|1⟩

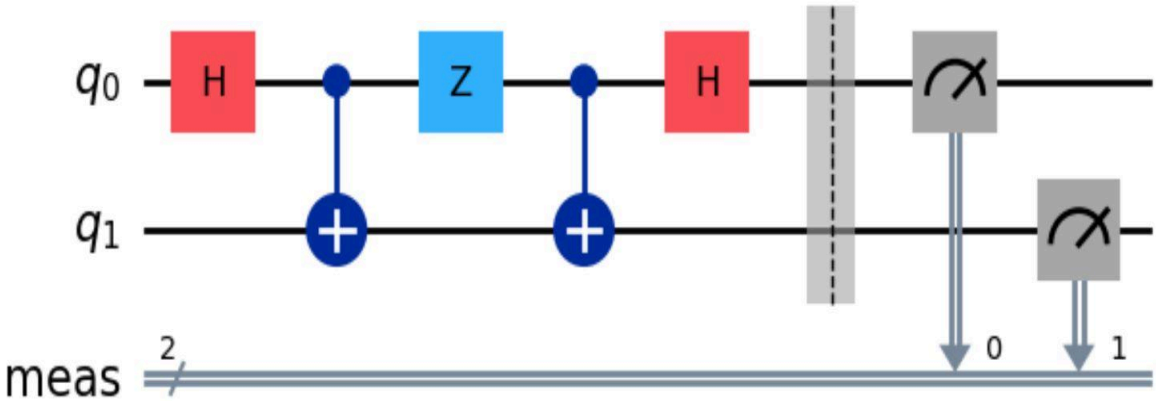

Initial Bell State (Before Encoding)
|0⟩
y
x
|1⟩

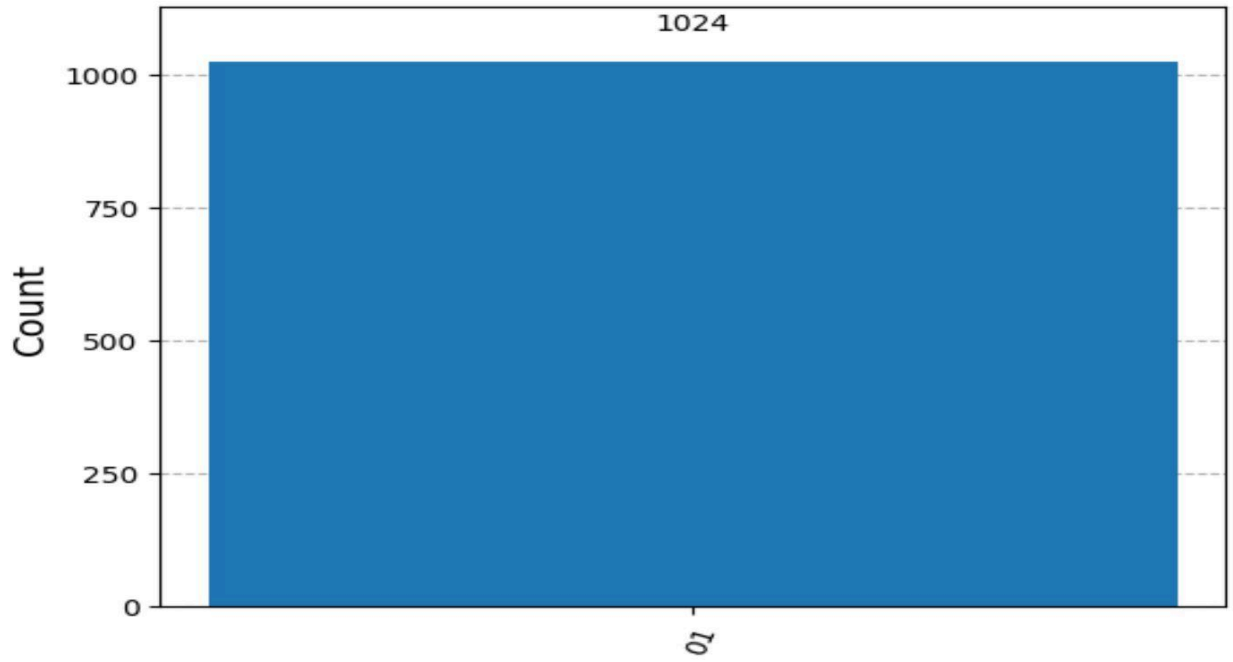

qubit 0
|0⟩
y
x
|1⟩


State After Encoding Message 10
|0⟩
y
x
|1⟩


qubit 0
|0⟩
y
x
|1⟩


State After Decoding Message 10
|0⟩
y
x
|1⟩


q_0
q_1
H
Z
H
meas
2
0
1



Conclusion -

- **Efficiency-Scalability Trade-off:** Superdense coding demonstrates an efficiency-scalability trade-off. While it enables the transmission of two classical bits using a single qubit, the protocol depends on the availability of high-quality entangled states and reliable quantum channels.
- **Algorithmic Significance**: Superdense coding is a foundational protocol in quantum communication, showcasing the power of entanglement to enhance classical information transfer. It enables efficient use of quantum resources and forms the basis for advanced communication techniques, playing a critical role in the development of secure quantum networks and distributed quantum systems.