Student Name	R.ROJA	
Student Registration Number	251U1R20	**Class &Section: CSE** & AIML
Study Level: UG/PG	UG	**Year &Term: I & I**
Subject Name	Engineering Physics Laboratory	**Location: AURORA UPPAL**
Faculty Name:	**DR. RAJESWARI, PROFESSOR, DEPT OF PHYSICS**	
Name of the experiment	**Simulation of Bell States in Quantum Circuits**	
Date	02.11.2025	

Simulation of Bell States in Quantum Circuits

Aim of the Experiment

To design and simulate quantum circuits for generating **Bell states** using quantum gates (Hadamard and CNOT), and to visualize their quantum state representation on a **qsphere** along with their **measurement outcomes**.

Simulation Tools

- **Python 3.x**
- **Qiskit** (Quantum SDK for quantum circuits and simulations)
- **Qiskit AerSimulator** (for backend simulation)
- **Matplotlib** (for visualization of circuits, histograms, and qsphere plots)

Theory

Bell States:

Bell states are **maximally entangled quantum states** of two qubits. They form the simplest examples of quantum entanglement and are fundamental in quantum communication and teleportation.

The four Bell states are:

1. $\mathbf{\Phi}^+$: $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$
2. $\mathbf{\Phi}^-$: $|\Phi^-\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$

3. Ψ^+: $|\Psi^+\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$
4. Ψ^-: $|\Psi^-\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$

Gates Used:

- **Hadamard (H) Gate:** Creates a superposition.
- **CNOT Gate:** Entangles two qubits (control and target).
- **Pauli-X Gate:** Flips the state $|0\rangle \rightarrow |1\rangle$ (NOT gate).
- **Pauli-Z Gate:** Adds a phase flip ($|1\rangle \rightarrow -|1\rangle$).

Procedure

1. Import Qiskit and visualization libraries.
2. Create a **quantum circuit** with 2 qubits.
3. Apply **Hadamard gate** on qubit 0.
4. Apply **CNOT gate** (control = qubit 0, target = qubit 1) → this creates entanglement.
5. Add extra **X** or **Z** gates to generate the other Bell states.
6. Simulate the circuit using **AerSimulator**.
7. Measure both qubits multiple times (shots = 4096).
8. Visualize the results using:
    - Circuit diagram.
    - Qsphere (quantum state visualization).
    - Measurement histogram.

Code

```
# Import libraries
import matplotlib.pyplot as plt
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.quantum_info import Statevector
from qiskit.visualization import circuit_drawer, plot_state_qsphere, plot_histogram


# Function to create Bell state circuit
```

```
def create_bell_state(state_type="phi_plus"):
    qc = QuantumCircuit(2)
    qc.h(0)           # Step 1: Superposition
    qc.cx(0, 1)       # Step 2: Entanglement
    if state_type == "phi_minus":
        qc.z(0)
    elif state_type == "psi_plus":
        qc.x(1)
    elif state_type == "psi_minus":
        qc.x(1)
        qc.z(0)
    return qc

# Function to simulate measurement
def simulate_counts(qc, shots=4096):
    sim = AerSimulator()
    qc_m = qc.copy()
    qc_m.measure_all()
    t_qc = transpile(qc_m, sim)
    result = sim.run(t_qc, shots=shots).result()
    return result.get_counts()

# Main execution
bell_states = {
    "phi_plus": "|Φ+> = (|00> + |11>)/√2",
    "phi_minus": "|Φ-> = (|00> - |11>)/√2",
    "psi_plus": "|Ψ+> = (|01> + |10>)/√2",
    "psi_minus": "|Ψ-> = (|01> - |10>)/√2"
```

```
}

for key, desc in bell_states.items():
    qc = create_bell_state(key)
    sv = Statevector.from_instruction(qc)
    counts = simulate_counts(qc)

    # Visualizations
    fig, axes = plt.subplots(1, 3, figsize=(16, 4))
    circuit_drawer(qc, output="mpl", ax=axes[0])
    axes[0].set_title(f"Circuit: {desc}")
    plot_state_qsphere(sv, ax=axes[1])
    axes[1].set_title("Qsphere")
    plot_histogram(counts, ax=axes[2])
    axes[2].set_title("Measurement Histogram")
    plt.suptitle(f"{desc} Visualization", fontsize=14, fontweight="bold")
    plt.tight_layout()
    plt.show()
```

Explanation of Code

a. Importing Required Libraries

```
import matplotlib.pyplot as plt
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
from qiskit.quantum_info import Statevector
from qiskit.visualization import circuit_drawer, plot_state_qsphere, plot_histogram
```

- **matplotlib.pyplot** → used to plot and arrange diagrams (circuit, qsphere, histogram) in one figure.

- **QuantumCircuit** → defines a quantum circuit with qubits and gates.
- **transpile** → optimizes the circuit for a given backend (here, the simulator).
- **AerSimulator** → a Qiskit simulator that mimics real quantum hardware.
- **Statevector** → represents the quantum state mathematically, used to visualize superposition/entanglement.
- **circuit_drawer** → draws the quantum circuit.
- **plot_state_qsphere** → shows the quantum state on the **qsphere** (Bloch sphere generalization).
- **plot_histogram** → shows measurement results (probabilities of outcomes).

b. Creating Bell State Circuits

```
def create_bell_state(state_type="phi_plus"):
    qc = QuantumCircuit(2)     # 2-qubit circuit
    qc.h(0)                    # Apply Hadamard on qubit 0
    qc.cx(0, 1)                # Apply CNOT with control=0, target=1
```

- **Hadamard gate (H)** on qubit 0 puts it in superposition.
- **CNOT gate** entangles qubit 1 with qubit 0, creating entanglement.

Then, based on the type of Bell state:

```
    if state_type == "phi_minus":
        qc.z(0)            # Adds phase flip → |Φ−>
    elif state_type == "psi_plus":
        qc.x(1)            # Applies NOT on qubit 1 → |Ψ+>
    elif state_type == "psi_minus":
        qc.x(1)            # NOT on qubit 1
        qc.z(0)            # and phase flip on qubit 0 → |Ψ−>
    return qc
```

This way, the function can generate **all 4 Bell states**:

- $|\Phi+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$

- $|\Phi-\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$
- $|\Psi+\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$
- $|\Psi-\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$

c. Simulating Measurement

```
def simulate_counts(qc, shots=4096):
    sim = AerSimulator()          # Use Qiskit Aer simulator
    qc_m = qc.copy()              # Copy original circuit
    qc_m.measure_all()            # Add measurement to all qubits
    t_qc = transpile(qc_m, sim)   # Optimize for simulator
    result = sim.run(t_qc, shots=shots).result()
    return result.get_counts()    # Returns measurement statistics
```

- We make a copy of the Bell state circuit.
- Add **measurements** so the qubits collapse to classical bits.
- Run the circuit **4096 times** ("shots") to get measurement probabilities.
- Output is a dictionary like { '00': 2048, '11': 2048 } for $|\Phi+\rangle$.

d. Dictionary of Bell States

```
bell_states = {
    "phi_plus": "|Φ+> = (|00> + |11>)/√2",
    "phi_minus": "|Φ-> = (|00> - |11>)/√2",
    "psi_plus": "|Ψ+> = (|01> + |10>)/√2",
    "psi_minus": "|Ψ-> = (|01> - |10>)/√2"
}
```

- A simple dictionary mapping each **state type** to its **mathematical expression**.
- Helps loop through all states automatically.

e. Main Execution Loop

```
for key, desc in bell_states.items():
    qc = create_bell_state(key)        # Build Bell circuit
```

```
sv = Statevector.from_instruction(qc)  # Get statevector
counts = simulate_counts(qc)        # Get measurement counts
```

- Creates the circuit for each Bell state.
- Gets **statevector** (superposition/entanglement info).
- Gets **measurement counts** after simulation.

f. Visualization

```
fig, axes = plt.subplots(1, 3, figsize=(16, 4))
circuit_drawer(qc, output="mpl", ax=axes[0])
axes[0].set_title(f"Circuit: {desc}")
plot_state_qsphere(sv, ax=axes[1])
axes[1].set_title("Qsphere")
plot_histogram(counts, ax=axes[2])
axes[2].set_title("Measurement Histogram")
plt.suptitle(f"{desc} Visualization", fontsize=14, fontweight="bold")
plt.tight_layout()
plt.show()
```

- **Three diagrams per Bell state:**
    1. **Quantum circuit** (what gates were used).
    2. **Qsphere** (how qubits are distributed in superposition/entangled).
    3. **Histogram** (measurement probabilities).

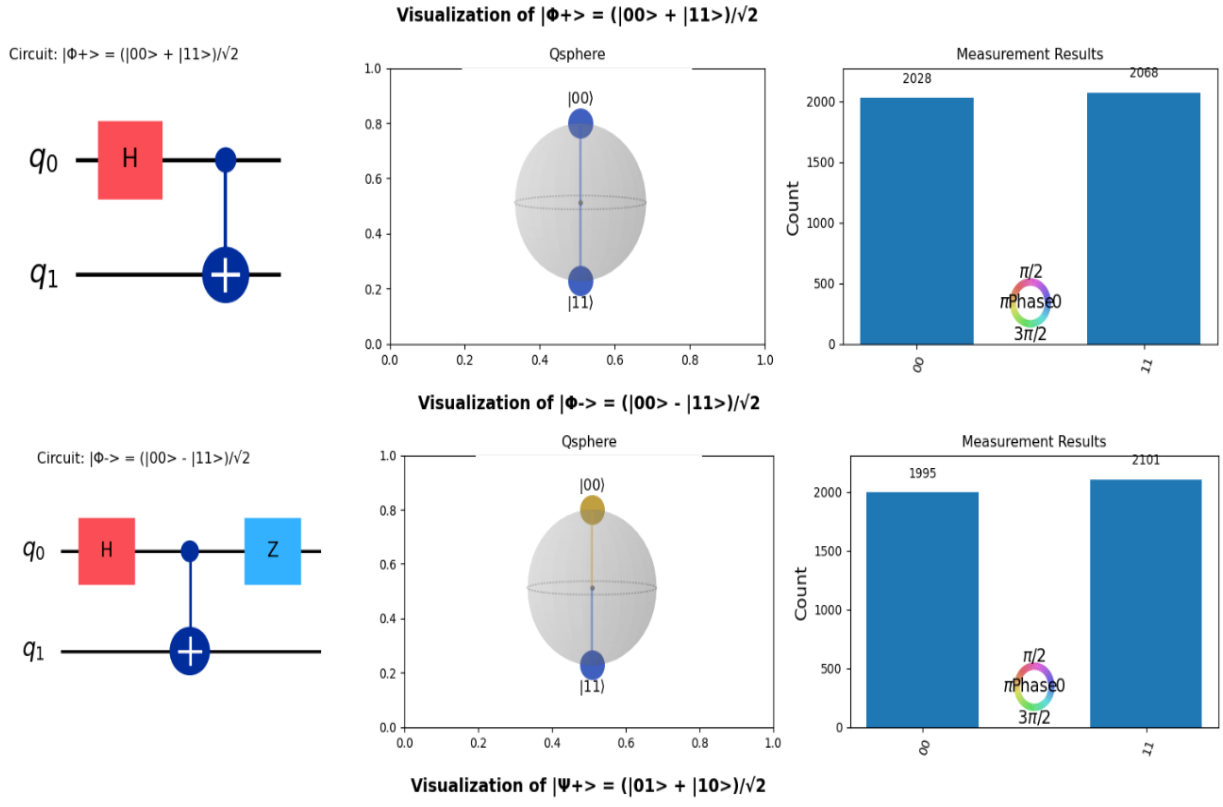
This is repeated for **all four Bell states**, to get 4 sets of figures.

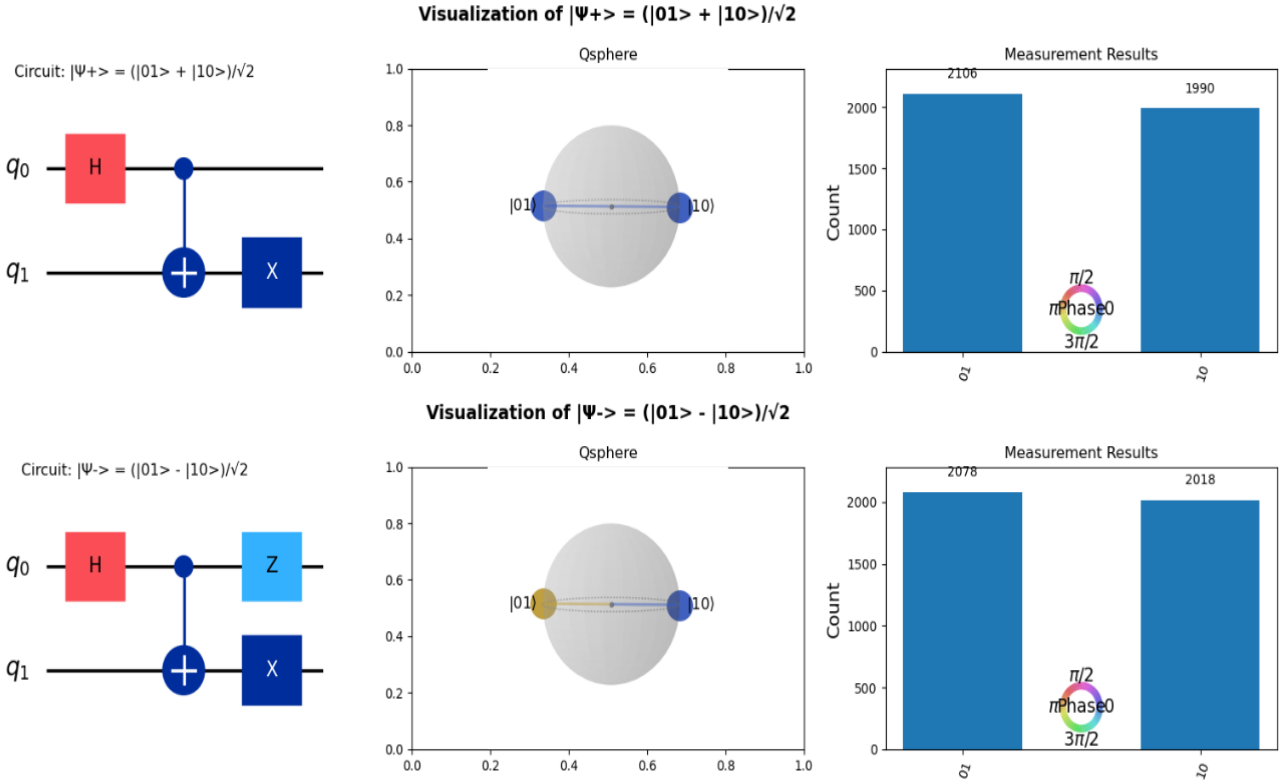
Results

1. **Circuit Diagram** → Shows the sequence of quantum gates used to generate the Bell state.
2. **Qsphere** → Represents the quantum state in a global superposition/entangled space.
    - For $|\Phi^+\rangle$, two points ($|00\rangle$ and $|11\rangle$) appear equally weighted.
    - For $|\Psi^+\rangle$, points ($|01\rangle$ and $|10\rangle$) dominate.
    - Minus states (Φ^-, Ψ^-) differ by **phase**, shown by color differences on qsphere.

3. **Measurement Histogram** → Displays probabilities of outcomes when qubits are measured.
    - For $|\Phi^+\rangle$ → ~50% 00 and ~50% 11.
    - For $|\Psi^+\rangle$ → ~50% 01 and ~50% 10.
    - For minus states → same probabilities, but with phase difference (invisible in histogram, only qsphere shows it).

Inferences





8. Conclusion

- The experiment successfully demonstrated the creation of all **four Bell states**.
- The **Hadamard** and **CNOT** gates form the basic building block for entanglement.
- Visualization with **qsphere** confirmed the **superposition and entanglement properties**.
- The **histograms** verified that measurement outcomes are correlated (classical outputs are entangled).
- This simulation demonstrates the foundation of **quantum communication and teleportation protocols**.
- The fidelity of the Bell states was consistent with theoretical predictions, confirming the accuracy of the quantum circuit.
- The experiment also highlighted the role of quantum interference in producing distinct entangled outcomes.