```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


from qiskit import QuantumCircuit, transpile, assemble, Aer, execute
from qiskit.visualization import plot_bloch_multivector, plot_histogram
from qiskit.extensions import Initialize
import numpy as np
qubits = 2
bell_circuits = []
bell_A = QuantumCircuit(qubits, qubits)
bell_A.h(0)
bell_A.cx(0, 1)
bell_A.measure_all()
bell_circuits.append(bell_A)
bell_B = QuantumCircuit(qubits, qubits)
bell_B.x(0)
bell_B.h(0)
bell_B.x(1)
bell_B.cx(0, 1)
bell_B.measure_all()
bell_circuits.append(bell_B)
bell_C = QuantumCircuit(qubits, qubits)
bell_C.x(0)
bell_C.h(0)
bell_C.cx(0, 1)
bell_C.measure_all()
bell_circuits.append(bell_C)
bell_D = QuantumCircuit(qubits, qubits)
bell_D.x(0)
bell_D.h(0)
bell_D.x(1)
bell_D.cx(0, 1)
bell_D.measure_all()
bell_circuits.append(bell_D)
simulator = Aer.get_backend('statevector_simulator')
results = []
for circuit in bell_circuits:
    transpiled_circuit = transpile(circuit, simulator)
    job = execute(transpiled_circuit, simulator)
    result = job.result()
    statevector = result.get_statevector()
    results.append(statevector)
for i, statevector in enumerate(results):
    print(f"Bell State {chr(65+i)}:")
    print(statevector)
for i, statevector in enumerate(results):
    plot_bloch_multivector(statevector)
for i, circuit in enumerate(bell_circuits):
    job = execute(circuit, simulator, shots=1024)
    result = job.result()
    counts = result.get_counts()
    print(f"Measurement outcomes for Bell State {chr(65+i)}:")
    print(counts)
    plot_histogram(counts)
for i, circuit in enumerate(bell_circuits):
    print(f"Circuit for Bell State {chr(65+i)}:")
```

```
    print(circuit)

# In[ ]:
```