

**TASK 5: CNOT Gate and Quantum Teleportation**

**Aim:** To simulate a CNOT gate and implement a simplified quantum teleportation protocol using Qiskit.

**Algorithm for CNOT Gate Implementation:**

1. Initialize a quantum circuit with 2 qubits and 2 classical bits.
2. Prepare input states (e.g., test all possible combinations:  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ ,  $|11\rangle$ ).
3. Apply CNOT gate (control qubit =  $q_0$ , target qubit =  $q_1$ ).
4. Measure the qubits and store results in classical bits.
5. Simulate the circuit using Qiskit's Aer simulator.
6. Plot the measurement outcomes.

**Mathematical Model for Quantum Teleportation:**

1. Entanglement (shared Bell pair)
2. Classical communication (2 bits)
3. Quantum operations (CNOT, Hadamard, measurements)

**Algorithm for Quantum Teleportation Implementation:**

1. Initialize 3-qubit circuit (Alice's  $q_0$ , shared  $q_1$ , Bob's  $q_2$ ) + 2 classical bits
2. Prepare Alice's qubit (e.g.,  $|1\rangle$  via X gate)
3. Create Bell pair between  $q_1$  &  $q_2$  (H + CNOT)
4. Teleportation protocol o CNOT( $q_0$ ,  $q_1$ ) o H( $q_0$ ) o Measure  $q_0$  &  $q_1 \rightarrow$  store in classical bits
5. Bob's corrections o Apply X if  $c_1=1$  o Apply Z if  $c_0=1$
6. Verify by measuring Bob's qubit

```
from qiskit import QuantumCircuit
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt
def cnot_circuit(input_state):
    """
    Creates and simulates a CNOT circuit for a given input
    state.
    Args:
        input_state (str): '00', '01', '10', or '11'
    """
    qc = QuantumCircuit(2, 2) # 2 qubits, 2 classical bits
    # Prepare input state
    if input_state[0] == '1':
        qc.x(0) # Set q0 to |1>
    if input_state[1] == '1':
        qc.x(1) # Set q1 to |1>
    # Apply CNOT (q0=control, q1=target)
```

```

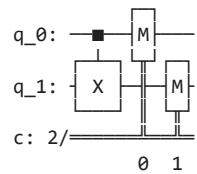
qc.cx(0, 1)
# Measure qubits
qc.measure([0, 1], [0, 1])
# Simulate
simulator = Aer.get_backend('qasm_simulator')
result = simulator.run(qc, shots=1000).result()
counts = result.get_counts(qc)
# Plot results
print(f"\nCNOT Gate Test | Input: |{input_state}>")
print("Circuit Diagram:")
print(qc.draw(output='text'))
plot_histogram(counts)
plt.show()
# Test all possible inputs
for state in ['00', '01', '10', '11']:
    cnot_circuit(state)
from qiskit import QuantumCircuit
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt
# Create circuit
qc = QuantumCircuit(3, 2) # 3 qubits, 2 classical bits
# Step 1: Prepare Alice's state (|1> for demo)
qc.x(0) # Comment out to teleport |0>
qc.barrier()
# Step 2: Create Bell pair (q1 & q2)
qc.h(1)
qc.cx(1, 2)
qc.barrier()
# Step 3: Teleportation protocol
qc.cx(0, 1)
qc.h(0)
qc.barrier()
# Step 4: Measure Alice's qubits
qc.measure([0,1], [0,1])
qc.barrier()
# Step 5: Bob's corrections
qc.cx(1, 2) # X if c1=1
qc.cz(0, 2) # Z if c0=1
# Step 6: Measure Bob's qubit
qc.measure(2, 0) # Overwrite c0 for verification
# Draw circuit
print("Teleportation Circuit:")
print(qc.draw(output='text'))
# Simulate
simulator = Aer.get_backend('qasm_simulator')
result = simulator.run(qc, shots=1000).result()
counts = result.get_counts(qc)
# Results
print("\nMeasurement results:")
print(counts)

```

```
plot_histogram(counts)
plt.show()
```

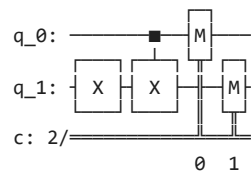
CNOT Gate Test | Input:  $|00\rangle$

Circuit Diagram:



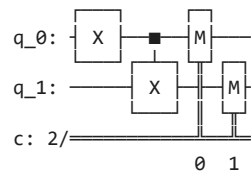
CNOT Gate Test | Input:  $|01\rangle$

Circuit Diagram:



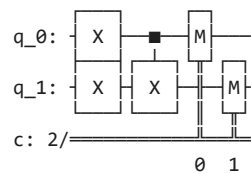
CNOT Gate Test | Input:  $|10\rangle$

Circuit Diagram:

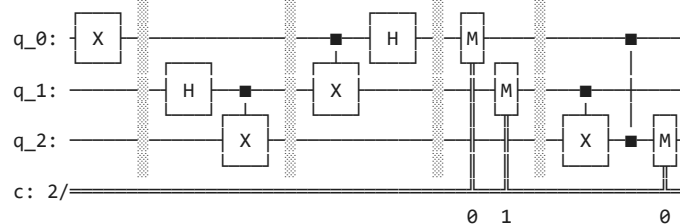


CNOT Gate Test | Input:  $|11\rangle$

Circuit Diagram:



Teleportation Circuit:



Measurement results:

```
{'11': 505, '01': 495}
```

**Result:**

This work illustrates the implementation, simulation, and verification of the CNOT gate using Qiskit, followed by the construction of a complete quantum teleportation protocol. The protocol is validated through simulation, confirming the accurate transfer of an arbitrary quantum state using entanglement and classical communication.