# Date:

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

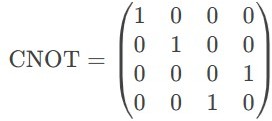
**Aim:** To simulate a CNOT gate and implement a simplified quantum teleportation protocol using Qiskit.

# Mathematical Model of the CNOT Gate

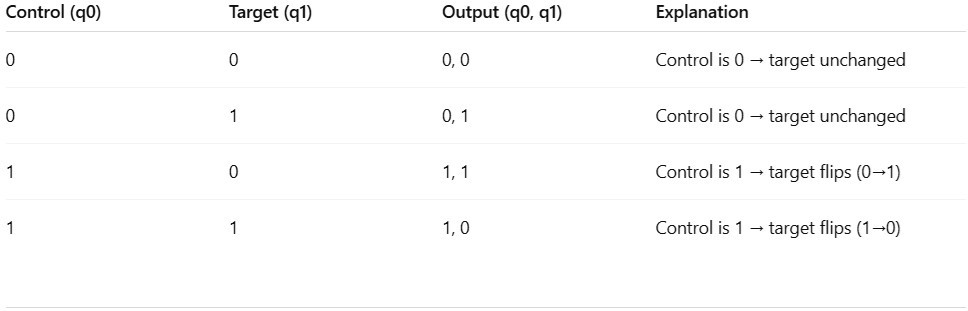
* The **CNOT (Controlled-NOT)** gate is a two-qubit quantum gate that flips the target qubit if and only if the control qubit is in state ∣1⟩.
* Computational basis ordering: ∣00⟩,∣01⟩,∣10⟩,∣11⟩ with first qubit = control(q0), second qubit = target(q1).

# Matrix Representation

The CNOT gate is represented by the following unitary matrix



# CNOT Gate Truth Table

****

* 1. **Effect on Basis States**
     + If **control = |0**⟩, target remains unchanged.
     + If **control = |1**⟩, target flips (X-gate applied).

# 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⟩, |01⟩, |10⟩, |11⟩).
3. **Apply CNOT gate** (control qubit = q0, target qubit = q1).
4. **Measure** the qubits and store results in classical bits.
5. **Simulate** the circuit using Qiskit’s Aer simulator.
6. **Plot** the measurement outcomes.

# Program

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)

# 4. Mathematical Model for Quantum Teleportation

Quantum teleportation enables transferring an unknown quantum state from Alice to Bob using:

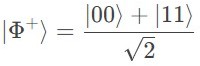
1. **Entanglement** (shared Bell pair)
2. **Classical communication** (2 bits)
3. **Quantum operations** (CNOT, Hadamard, measurements)

# Initial Setup

* + - Alice has qubit

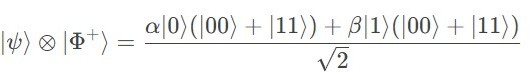


* + - Alice and Bob share an entangled Bell pair

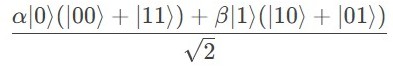


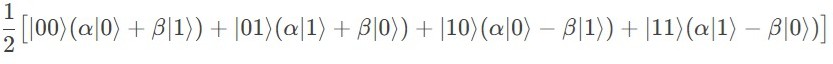
* 1. **Step-by-Step State Evolution**

1. **Combined state**

****

1. **Alice applies CNOT (q0 → q1)**

****

1. **Alice applies Hadamard to q0**
2. **Alice measures q0 & q1** → gets one of 4 classical outcomes (00, 01, 10, 11)

# Bob applies corrections

* + **00**: Do nothing
  + **01**: Apply X gate
  + **10**: Apply Z gate
  + **11**: Apply X then Z Final state at Bob's qubit

# Algorithm for Quantum Teleportation Implementation

* 1. **Initialize** 3-qubit circuit (Alice's q0, shared q1, Bob's q2) + 2 classical bits
  2. **Prepare** Alice's qubit (e.g., |1⟩ via X gate)
  3. **Create Bell pair** between q1 & q2 (H + CNOT)

# Teleportation protocol

* + - CNOT(q0, q1)
    - H(q0)
    - Measure q0 & q1 → store in classical bits

# Bob's corrections

* + - Apply X if c1=1
    - Apply Z if c0=1
  1. **Verify** by measuring Bob's qubit

# Program for Quantum Teleportation Implementation

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()

1. **Outputs:**

