CSBB311: QUANTUM COMPUTING

ASSIGNMENT 2 :- Quantum Measurement Using Qiskit

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

Code:-

```
# Import necessary libraries
from qiskit import QuantumCircuit, transpile, assemble
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_bloch_vector, plot_histogram
from qiskit.visualization import plot_bloch_multivector
from qiskit.quantum_info import Statevector
import matplotlib.pyplot as plt
import numpy as np

# Function to display a quantum circuit
def show_circuit(qc):
    plt.clf()  # Clear any previous figures
    qc.draw(output='mpl')
    plt.show()

# Function to show the Bloch sphere representation of a qubit's state
def show_bloch_vector(qc):
    plt.clf()  # Clear any previous Bloch sphere
    # Check if the circuit has more than one qubit, and only show Bloch sphere for single qubit circuits
    if qc.num_qubits == 1:
        simulator = AerSimulator()  # Use AerSimulator instead of Aer
        circ = transpile(qc, simulator)
        result = simulator.run(circ).result()
        state = result.get_statevector()
        plot_bloch_multivector(state)
        plt.show()
    else:
        print("Bloch sphere visualization is only for single-qubit circuits.")

# Explain and show the working of each gate with simulation
def explain_gate(qc, description):
    print(description)
    show_circuit(qc)
    show_bloch_vector(qc)

# 1. Pauli-X (NOT) Gate
description_x = "Pauli-X (NOT) Gate: Flips the qubit state from |0⟩ to |1⟩ or vice versa."
qc_x = QuantumCircuit(1)
qc_x.x(0)  # Apply X gate
explain_gate(qc_x, description_x)

# 2. Pauli-Y Gate
description_y = "Pauli-Y Gate: Rotates the qubit around the Y-axis of the Bloch sphere by π radians."
qc_y = QuantumCircuit(1)
qc_y.y(0)  # Apply Y gate
explain_gate(qc_y, description_y)

# 3. Pauli-Z Gate
description_z = "Pauli-Z Gate: Applies a phase flip to the |1⟩ state, leaving |0⟩ unchanged."
qc_z = QuantumCircuit(1)
qc_z.z(0)  # Apply Z gate
explain_gate(qc_z, description_z)

# 4. Hadamard Gate
description_h = "Hadamard Gate: Puts the qubit into a superposition, equally likely to be measured as |0⟩ or |1⟩."
qc_h = QuantumCircuit(1)
qc_h.h(0)  # Apply H gate
explain_gate(qc_h, description_h)
```

```python
# 6. T Gate
description_t = "T Gate: Adds a phase of π/4 to the |1⟩ state."
qc_t = QuantumCircuit(1)
qc_t.t(0)  # Apply T gate
explain_gate(qc_t, description_t)

# 7. Controlled-NOT (CNOT) Gate (2 qubits: control and target)
description_cnot = "CNOT Gate: Flips the target qubit if the control qubit is |1⟩."
qc_cnot = QuantumCircuit(2)
qc_cnot.cx(0, 1)  # Apply CNOT gate
show_circuit(qc_cnot)  # Only show the circuit, since Bloch vector visualization is not for multiple qubits

# 8. SWAP Gate (2 qubits)
description_swap = "SWAP Gate: Swaps the states of two qubits."
qc_swap = QuantumCircuit(2)
qc_swap.swap(0, 1)  # Apply SWAP gate
show_circuit(qc_swap)  # Only show the circuit for multi-qubit gates

# 9. Toffoli (CCNOT) Gate (3 qubits: 2 control, 1 target)
description_toffoli = "Toffoli (CCNOT) Gate: Flips the target qubit if both control qubits are in the |1⟩ state."
qc_toffoli = QuantumCircuit(3)
qc_toffoli.ccx(0, 1, 2)  # Apply Toffoli gate
show_circuit(qc_toffoli)  # Only show the circuit for multi-qubit gates

# 10. Rotation-X Gate (Rx)
description_rx = "Rotation-X Gate: Rotates the qubit around the X-axis by a given angle (here, π/2)."
qc_rx = QuantumCircuit(1)
qc_rx.rx(np.pi / 2, 0)  # Rotate around X-axis by π/2
explain_gate(qc_rx, description_rx)

# 11. Rotation-Y Gate (Ry)
description_ry = "Rotation-Y Gate: Rotates the qubit around the Y-axis by a given angle (here, π/2)."
qc_ry = QuantumCircuit(1)
qc_ry.ry(np.pi / 2, 0)  # Rotate around Y-axis by π/2
explain_gate(qc_ry, description_ry)

# 12. Rotation-Z Gate (Rz)
description_rz = "Rotation-Z Gate: Rotates the qubit around the Z-axis by a given angle (here, π/2)."
qc_rz = QuantumCircuit(1)
qc_rz.rz(np.pi / 2, 0)  # Rotate around Z-axis by π/2
explain_gate(qc_rz, description_rz)
```

Output