SCS 43XX

Quantum Mechanics in Computing

University of Colombo, School of Computing

Labsheet 02

Circuits in Qiskit

In this tutorial, we'll explore setting up quantum circuits.

Introduction

This labsheet covers the basics of quantum circuits using Qiskit. You will learn how to create quantum circuits, apply quantum gates, create entanglement, and measure qubits. The hands-on tasks will help solidify your understanding. The goal is to familiarize you with the fundamental building blocks of quantum circuits.

1 Setting Up Qiskit

Before starting, ensure you have Qiskit installed. If not, install it using:

```
pip install qiskit
```

Then, import the necessary modules for quantum circuit creation, simulation, and visualization:

```
from qiskit import QuantumCircuit, Aer, transpile, assemble, execute
from qiskit.visualization import plot_histogram
```

The `Aer` module allows you to simulate quantum circuits without needing real quantum hardware. To set up the quantum simulator, we can use the `qasm_simulator` backend for running circuits and measuring qubits.

```
# Setup the simulator
backend = Aer.get_backend('qasm_simulator')

# Example: Create a simple quantum circuit
qc = QuantumCircuit(1, 1)  # 1 qubit, 1 classical bit
qc.h(0)  # Apply Hadamard gate to qubit 0
qc.measure(0, 0)  # Measure the qubit into the classical bit
```

Once the circuit is created, we can execute it on the simulator. The next step is to transpile the circuit and then assemble it for execution.

```
# Transpile the circuit for optimization
compiled_circuit = transpile(qc, backend)

# Assemble the circuit for execution
job = execute(compiled_circuit, backend, shots=1024)

# Get the result and display the histogram of the measurement outcomes
result = job.result()
counts = result.get_counts()
plot_histogram(counts)
```

This will run the quantum circuit and plot a histogram of the measurement results.

2 Creating a Simple Quantum Circuit

A quantum circuit consists of qubits (quantum bits) and classical bits. Classical bits can be used to store measurement outcomes. Here's how you create a quantum circuit with 1 qubit and 1 classical bit:

```
qc = QuantumCircuit(1, 1)  # 1 qubit, 1 classical bit
qc.measure(0, 0)  # Measure qubit 0 into classical bit 0
qc.draw()
```

The `qc.draw()` command generates a diagram of the circuit. In this case, the diagram will show one qubit, one classical bit, and the measurement operation. The state of the qubit will be measured and stored in the classical bit.

3 Quantum Gates

Quantum gates manipulate qubits. Below are some basic quantum gates and their operations:

3.1 Hadamard Gate (H)

The Hadamard gate is one of the most commonly used gates. It puts a qubit into a superposition of states, meaning the qubit is in a combination of the $|0\rangle$ and $|1\rangle$ states.

Example:

```
qc = QuantumCircuit(1, 1)
qc.h(0)  # Apply Hadamard gate to qubit 0
qc.measure(0, 0)
qc.draw()
```

This circuit applies a Hadamard gate to qubit 0 and measures it.

3.2 Pauli Gates (X, Y, Z)

Pauli gates are fundamental in quantum computing:

- The X gate is the quantum equivalent of a classical NOT gate (also known as a "bit flip"). - The Y and Z gates perform rotations around the Y and Z axes, respectively.

Example:

```
qc = QuantumCircuit(1, 1)
qc.x(0)  # Apply Pauli-X gate (bit flip)
qc.measure(0, 0)
qc.draw()
```

3.3 Controlled NOT Gate (CNOT)

The CNOT gate is a two-qubit gate that applies a NOT (X) gate to the target qubit only if the control qubit is in the $|1\rangle$ state.

Example:

```
qc = QuantumCircuit(2, 2)
qc.cx(0, 1)  # Apply CNOT with qubit 0 as control and qubit 1 as target
qc.measure([0, 1], [0, 1])
qc.draw()
```

4 Measuring Qubits

After applying quantum gates, you need to measure the qubits to get classical information. Measurement collapses the qubit's quantum state into one of the basis states.

Example of measuring a qubit:

```
backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1024)
result = job.result()
counts = result.get_counts()
plot_histogram(counts)
```

The `plot_histogram(counts)` function will show the distribution of measurement outcomes, representing the probability of measuring each state (0 or 1).

5 Tasks

5.1 Task 1: Create a Quantum Coin Flip

Create a 1-qubit circuit:

1. Apply a Hadamard gate to the qubit.
2. Measure the qubit.

This will simulate a quantum coin flip. Observe the measurement results after running the circuit.

5.2 Task 2: Apply a Pauli-X Gate

Create a 1-qubit circuit:

1. Initialize the qubit in state $|0\rangle$.
2. Apply a Pauli-X gate to flip the state to $|1\rangle$.
3. Measure the qubit.

This will help you understand how the Pauli-X gate works.

5.3 Task 3: Create Superposition with Multiple Qubits

Create a 2-qubit circuit:

1. Apply a Hadamard gate to the first qubit.
2. Measure both qubits.

This will create a simple superposition between two qubits. Observe the results after measurement.

5.4 Task 4: Entanglement with Two Qubits

Create a 2-qubit circuit:

1. Apply a Hadamard gate to the first qubit.
2. Apply a CNOT gate with qubit 0 as control and qubit 1 as target.
3. Measure both qubits.

This will create a Bell state, which is an entangled state between the two qubits.

5.5 Task 5: Simple Measurement

Create a 1-qubit circuit:

1. Initialize the qubit in state $|0\rangle$.
2. Apply a Hadamard gate.
3. Measure the qubit.

This simple task will show how the measurement affects the qubit state.

5.6 Task 6: Implement a Quantum NOT Gate

Create a 1-qubit circuit:

1. Apply a Pauli-X gate to flip the qubit from state $|0\rangle$ to $|1\rangle$.
2. Measure the qubit.

This will help reinforce your understanding of the Pauli-X (NOT) gate.

5.7 Task 7: Superposition and Measurement with Three Qubits

Create a 3-qubit circuit:

1. Apply a Hadamard gate to each of the 3 qubits.
2. Measure all qubits.

This will create a superposition state across all three qubits. Measure the qubits and observe the distribution of outcomes.

5.8 Task 8: Entanglement with Three Qubits

Create a 3-qubit circuit:

1. Apply a Hadamard gate to the first qubit.
2. Apply CNOT gates between qubit 1 (control) and qubit 2 (target), then between qubit 1 (control) and qubit 3 (target).
3. Measure all qubits.

This will generate a 3-qubit entangled state.

5.9 Task 9: Quantum XOR Operation with Four Qubits

Create a 4-qubit circuit:

1. Apply a Hadamard gate to the first qubit.
2. Apply CNOT gates in a chain (qubit 1 controls qubit 2, qubit 2 controls qubit 3, etc.).
3. Measure all qubits.

This will simulate a quantum XOR operation over 4 qubits.