

# 2023 부산시 양자컴퓨 팅 개발자 교육 프로그램

## Lecture 4

Inho Choi

Qiskit Advocate

# Syllabus

- Lecture 1: 게이트와 양자 회로 기본 작성법
  - Single qubit gate      Lecture 1
  - Multiple qubit gate      Lecture 2
  - Multiple qubit gate Notebook Demonstration
  - Barriers and Properties of Quantum Circuit
- Lecture 2: 양자 회로의 측정과 OpenQasm
- Notebook Demonstration
- Lecture 3: 양자 백엔드에 양자회로 실행하기
- Lecture 4: 양자 회로 및 회로의 실행 결과 시각화 및 해석
- Lecture 5: 유용한 기능들

} Lecture 3

} Lecture 4

} Lecture 5

- Lecture 1: 게이트와 양자 회로 기본 작성법
  - Single qubit gate
  - Multiple qubit gate
  - Multiple qubit gate Notebook Demonstration
  - Barriers and Properties of Quantum Circuit
- Lecture 2: 양자 회로의 측정과 OpenQasm
  - **Notebook Demonstration**
- **Lecture 3: 양자 백엔드에 양자회로 실행하기**
- Lecture 4: 양자 회로 및 회로의 실행 결과 시각화 및 해석
- Lecture 5: 유용한 기능들

## Lecture 2 - 양자 회로의 측정과 OpenQasm

1. 양자 회로의 측정과 비단일 연산자
2. 양자 회로의 예제
3. OpenQasm

### 1. 양자 회로의 측정과 비단일 연산자 (non-unitary operator)

양자 회로에 단일 큐비트 게이트와 다중 큐비트 게이트와 같이 양자 회로에 직접 큐비트에 작용을 시키는 단일 연산자 (unitary operation) 연산자에도 접근을 할 수 있습니다. 비단일 연산자는 주로 아래와 같이 있습니다.

- 측정
- 큐비트의 초기화
- 고전적 조건부 연산자

```
In [1]: from qiskit import * # qiskit 라이브러리 모두 불러옵니다.
from qiskit.quantum_info import Statevector # 양자 상태벡터를 계산하기 위해 qiskit.quantum_info 모듈을 불러옵니다.
from qiskit.visualization import plot_bloch_multivector
```

측정

양자 컴퓨터를 측정하기 위해서 모든 정보를 접근할 필요는 없습니다. 양자 상태는 표준 기준으로 내려가 됩니다. Qiskit에서는 두가지의 방식 측정할 수 있으며 양자 비트를 고전 비트로 측정한다 라고 표현할 수 있습니다.

- `QuantumCircuit.measure(qubit, cbit)`
- `QuantumCircuit.measure_all`

무슨은 예를 들어 1개의 큐비트를 가진 양자 회로를 준비하고 상태가 0인 큐비트를 측정해봅시다.

```
In [2]: qc = QuantumCircuit(1,1) # 1개의 큐비트와 1개의 고전 비트를 가진 회로를 생성합니다.
qc.measure(0,0) # 양자 회로에 0번 큐비트를 측정합니다. 결과는 0번을 가진 고전 비트에 측정값을 저장합니다.
qc.draw('mpl') # 양자 회로를 그림합니다.
```

Out [2]:



## Lecture 3 - 양자 회로의 실행과 백엔드

1. 양자 회로의 실행
2. 백엔드

### 1. 양자 회로의 실행

큐비드와 연산자를 추가하여 양자 회로를 만들었다면 시뮬레이션이나 실제 양자 컴퓨터에 실행에 작업을 보내 실행시켜 결과값을 얻을 수 있습니다. 우선은 마지막 양자 회로를 만들어 양자 회로를 실행해 봅시다.

```
> from qiskit import * # qiskit 라이브러리 모두 불러옵니다.
from qiskit.visualization import plot_histogram, plot_circuit_layout # plot_histogram, plot_circuit_layout 모듈을 불러옵니다.
```

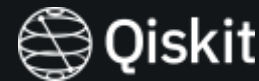
```
qc=QuantumCircuit(3,3) # 3개의 큐비트와 3개의 고전 비트를 가진 양자 회로를 만듭니다.
qc.h(0) # 0번 큐비트에 Hadamard 게이트를 적용합니다.
qc.cx(0,1) # 0번 큐비트를 control, 1번 큐비트를 target으로 지정해 CX 게이트를 적용합니다.
qc.cx(1,2) # 1번 큐비트를 control, 2번 큐비트를 target으로 지정해 CX 게이트를 적용합니다.
qc.measure([0,1,2],[0,1,2]) # 0,1,2번 큐비트를 0,1,2번 고전 비트에 측정합니다.
qc.draw(output='mpl') # 양자 회로를 그림합니다.
```



Github: Qiskit-Dev-Cert-lectures, 3 양자 회로의 실행과 백엔드

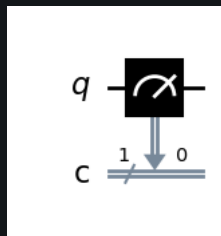
Github: Qiskit-Dev-Cert-lectures, 2 양자 회로의 측정과 OpenQasm

# Quantum Circuit and Measurement

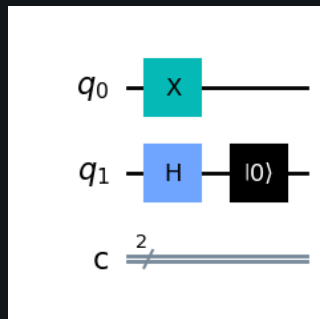


1. Non-unitary Operator
2. Quantum Circuit and Register
3. OpenQasm

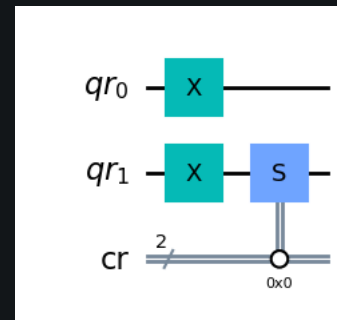
# Non-unitary Operator



Measure



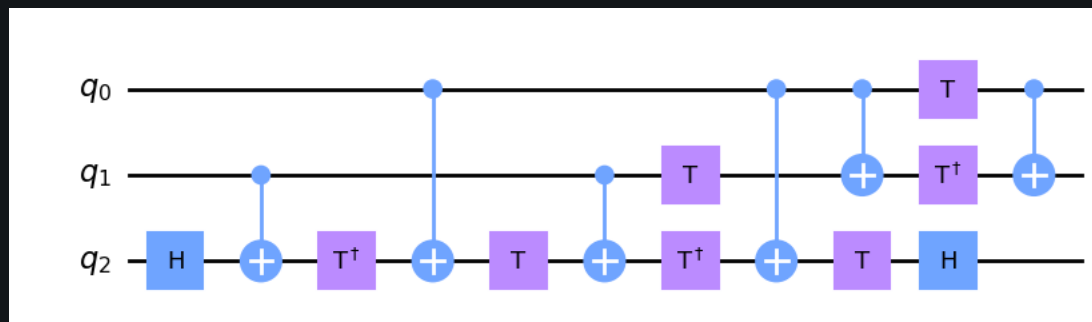
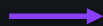
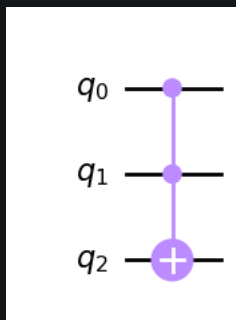
Initialization

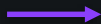
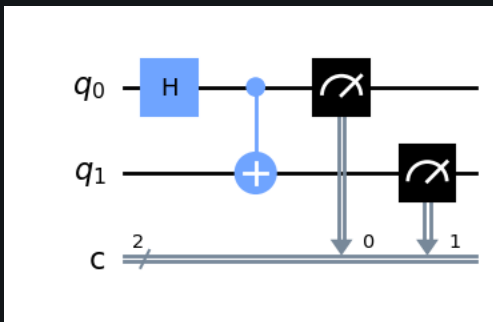


Classical conditional operator

# Quantum Circuit and Register

1. Quantum Register
2. Classical Register
3. Decompose

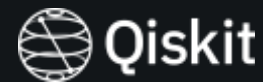




```
OPENQASM 2.0;  
include "qelib1.inc";  
qreg q[2];  
creg c[2];  
h q[0];  
cx q[0],q[1];  
measure q[0] -> c[0];  
measure q[1] -> c[1];
```



# Quantum Backend and Executing Quantum Circuit



1. Real Backend
2. Transpile
3. Simulation

# QnA