응용물리연구실심화실습3 (PHY3075)

2024년 11월 1주차

응용물리학과 2022006971 이민성

Quantum circuits

번역

- 소개

  이번 강의에서는 **양자 회로 모델(quantum circuit model) 계산**을 소개합니다. 이는 양자 계산을 설명하는 표준적인 방식이며, 이 강의 시리즈 전반에 걸쳐 사용될 것입니다.

  또한, 몇 가지 중요한 수학적 개념도 함께 다룰 예정입니다. 여기에는 **벡터 간의 내적(inner product)**, **직교성(orthogonality)** 및 **직교 정규성(orthonormality)**, 그리고 **투영(projections)** 및 투영 측정(projective measurements)의 개념이 포함됩니다. 투영 측정은 표준 기저 측정을 일반화한 것입니다.

  이러한 개념을 통해 양자 정보의 근본적인 한계에 대해 다룰 것입니다. 여기에는 양자 상태 복제 불가능성(no-cloning theorem)과 비직교 양자 상태를 완벽히 구별할 수 없는 원리(impossibility to perfectly discriminate non-orthogonal quantum states)가 포함됩니다.

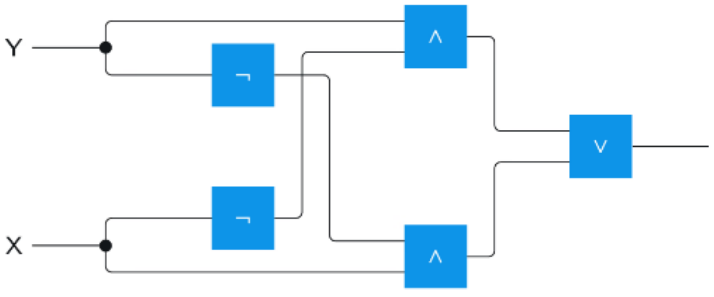
- 회로

  컴퓨터 과학에서 회로(circuit)란 정보를 **게이트(gate)** 네트워크를 통해 와이어(wire)가 운반하고, 게이트는 와이어가 운반하는 정보를 변환하는 연산을 나타내는 계산 모델을 의미합니다. 양자 회로는 이 일반적인 개념을 기반으로 한 계산 모델의 한 예에 불과합니다.

  "회로(circuit)"라는 단어는 종종 순환 경로를 의미하지만, 계산 모델로서의 회로에서는 순환 경로가 허용되지 않는 경우가 대부분입니다. 즉, 계산 모델로서 회로를 생각할 때는 주로 비순환 회로(acyclic circuit)를 다룹니다. 양자 회로 또한 이 패턴을 따릅니다. 양자 회로를 원하는 만큼 반복 실행할 수는 있지만, 양자 회로 자체는 피드백 루프를 포함할 수 없는 유한한 연산의 순서를 나타냅니다.

  불리언 회로

  다음은 (고전적인) 불리언 회로의 예입니다. 여기서 와이어는 이진 값을 전달하며, 게이트는 불리언 논리 연산을 나타냅니다.



  와이어를 따라 정보의 흐름은 왼쪽에서 오른쪽으로 진행됩니다. 그림의 왼쪽에 있는 **X**와 **Y**는 입력 비트로, 각각 우리가 선택할 수 있는 이진 값으로 설정할 수 있습니다. 오른쪽에 있는 와이어는 출력입니다. 중간 와이어는 왼쪽에서 오른쪽으로 평가되는 게이트에 의해 결정되는 값을 전달합니다.

게이트는 **AND** 게이트(∧로 표시), **OR** 게이트(∨로 표시), **NOT** 게이트(¬로 표시)입니다. 이들 게이트가 계산하는 함수는 많은 독자들에게 익숙할 가능성이 있지만, 여기서는 값의 표로 나타냅니다.

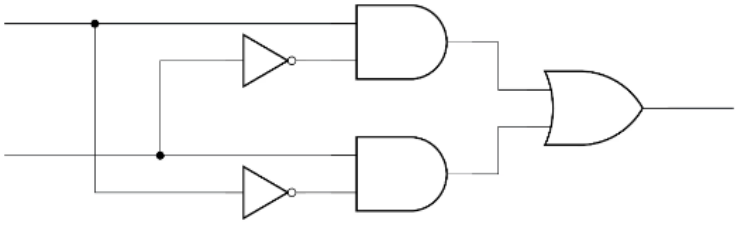
a	$\neg a$
0	1
1	0

ab	$a \wedge b$
00	0
01	0
10	0
11	1

ab	$a \vee b$
00	0
01	1
10	1
11	1

X와 **Y**의 이름 오른쪽에 있는 두 개의 작은 원은 **팬아웃(fanout) 작업**을 나타냅니다. 팬아웃은 단순히 와이어에 전달되는 값을 복사하여 여러 게이트에 입력될 수 있도록 하는 연산입니다. 고전적인 설정에서 팬아웃 작업은 항상 게이트로 간주되지 않으며, 때때로 "무료"인 것으로 처리되기도 합니다. 하지만 우리가 고전적인 불리언 회로가 어떻게 동등한 양자 회로로 변환될 수 있는지 논의할 때, 팬아웃 작업을 명시적으로 게이트로 분류하고 이를 정확히 처리해야 합니다.

다음은 전기 공학에서 더 일반적으로 사용되는 스타일로 그려진 동일한 회로로, **AND**, **OR**, **NOT** 게이트의 전통적인 기호를 사용합니다.



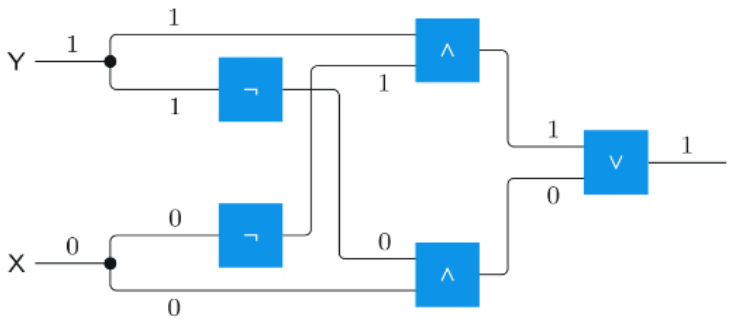
우리는 이 스타일이나 특정 게이트 기호를 더 이상 사용하지 않지만, 양자 회로에서는 게이트를 나타내기 위해 다른 기호를 사용하며, 이러한 기호는 우리가 이를 만날 때 설명할 것입니다.

이 예시에서 특정 회로는 배타적 OR(Exclusive OR, XOR)를 계산하는데, 이는 기호 ⊕로 나타냅니다.

ab	$a \oplus b$
00	0
01	1
10	1
11	0

다음 다이어그램에서는 입력에 대한 한 가지 선택을 고려합니다: **X = 0** 및 **Y = 1**.

각 와이어는 해당 와이어가 전달하는 값으로 레이블이 지정되어 있으므로, 연산을 따라갈 수 있습니다. 이 경우 출력 값은 **1**이며, 이는 XOR의 올바른 값입니다: **0 ⊕ 1 = 1**.

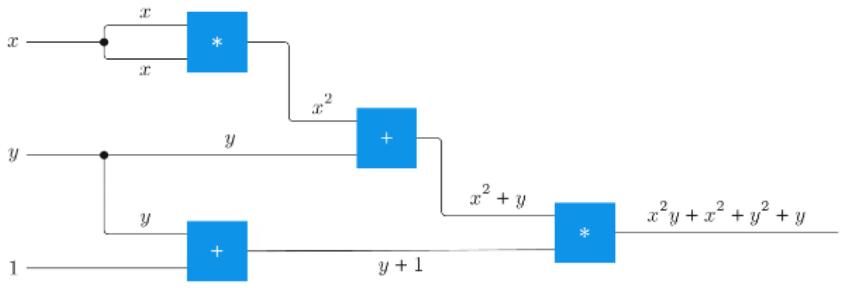


다른 세 가지 가능한 입력 설정은 비슷한 방식으로 확인할 수 있습니다.

다른 유형의 회로

앞서 언급했듯이, 컴퓨터 과학에서 회로의 개념은 매우 일반적입니다. 예를 들어, 와이어가 **0**과 **1** 이외의 값을 전달하는 회로도 연구되며, 서로 다른 연산을 나타내는 게이트들도 다루어집니다.

산술 회로(arithmetic circuits)에서는 와이어가 **정수** 값을 전달하고, 게이트는 덧셈이나 곱셈과 같은 **산술 연산**을 나타낼 수 있습니다. 다음 그림은 두 개의 변수 입력 값(**x**와 **y**)과 세 번째 입력이 **1**로 설정된 값을 가지는 산술 회로를 나타냅니다. 와이어가 전달하는 값은 **x**와 **y** 값의 함수로서 그림에 표시됩니다.



우리는 또한 게이트가 확률적 연산을 나타내는, 즉 **랜덤성**을 포함하는 회로도 고려할 수 있습니다.

양자 회로

양자 회로 모델에서, 와이어는 큐비트(qubits)를 나타내며, 게이트는 이 큐비트들에 작용하는 연산을 나타냅니다. 지금은 우리가 지금까지 접한 연산, 즉 유니터리 연산(unitary operations)과 표준 기저 측정(standard basis measurements)에 집중할 것입니다. 다른 종류의 양자 연산과 측정에 대해 배우게 되면, 그에 맞게 모델을 확장할 것입니다.

다음은 간단한 양자 회로의 예입니다:

이 회로에서는 **X**라는 이름의 단일 큐비트가 수평선으로 나타내어지며, 이 큐비트에 대한 유니터리 연산을 나타내는 일련의 게이트들이 있습니다. 위의 예들처럼, 정보의 흐름은 왼쪽에서 오른쪽으로 진행됩니다. 따라서 첫 번째 연산은 **Hadamard** 연산, 두 번째는 **S** 연산, 세 번째는 또 다른 **Hadamard** 연산, 마지막 연산은 **T** 연산입니다. 따라서 전체 회로를 적용하면 이 연산들의 조합인 **THSH**가 큐비트 **X**에 적용됩니다.

때때로 우리는 회로에 대한 입력 또는 출력 상태를 명시적으로 나타내고자 합니다. 예를 들어, **THSH** 연산을 상태 |0⟩에 적용하면 상태 (1 + i)/2 |0⟩ + 1/2 |1⟩을 얻습니다. 이를 다음과 같이 나타낼 수 있습니다:

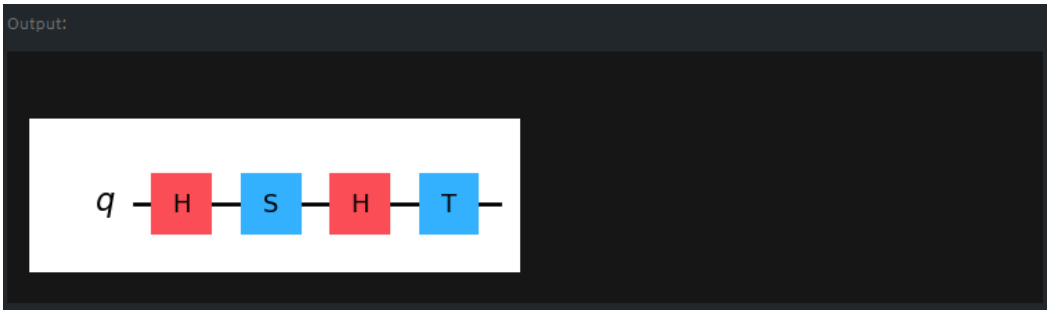
양자 회로는 종종 모든 큐비트가 **|0⟩** 상태로 초기화됩니다. 이번 경우처럼 말이죠. 그러나 입력 큐비트를 다른 상태로 설정해야 하는 경우도 있습니다.

이제 Qiskit에서 이 회로를 어떻게 지정할 수 있는지, 현재 섹션에 필요한 **import**부터 살펴보겠습니다.

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram
```

먼저 왼쪽에서 오른쪽으로 순차적으로 게이트를 추가하여 다음과 같이 회로를 만들 수 있습니다.

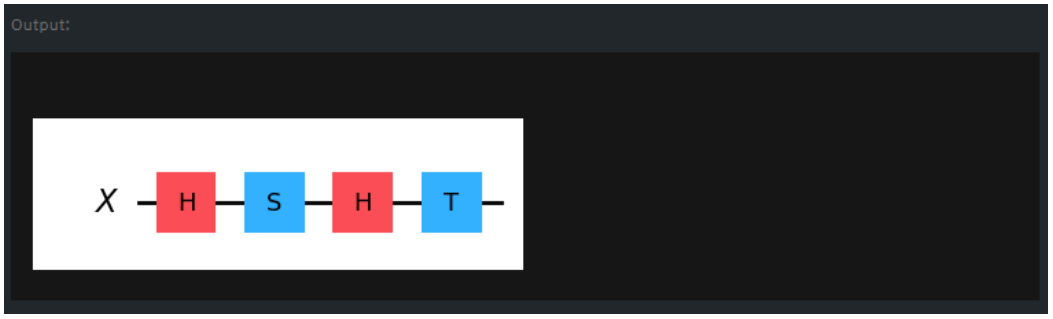
```
circuit = QuantumCircuit(1)

circuit.h(0)
circuit.s(0)
circuit.h(0)
circuit.t(0)

display(circuit.draw())
```



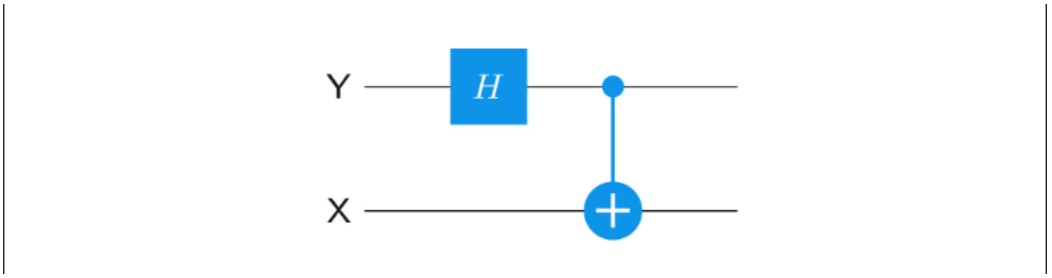
Qiskit에서 큐비트의 기본 이름은 **q0**, **q1**, **q2** 등이며, 우리 예시처럼 큐비트가 하나일 경우 기본 이름은 **q**입니다. 만약 우리가 원하는 이름을 선택하고 싶다면, **QuantumRegister** 클래스를 사용하여 다음과 같이 할 수 있습니다 :

```
X = QuantumRegister(1, "X")
circuit = QuantumCircuit(X)

circuit.h(X)
circuit.s(X)
circuit.h(X)
circuit.t(X)
```

```
display(circuit.draw())
```



다음은 큐비트 두 개를 사용한 양자 회로의 또 다른 예시입니다 :



항상 그렇듯이, **H**로 표시된 게이트는 **Hadamard** 연산을 의미하며, 두 번째 게이트는 **두 큐비트 게이트**입니다: 이는 **제어-NOT** 연산으로, 실선 원은 제어 큐비트를 나타내고, ⊕ 기호를 닮은 원은 타겟 큐비트를 나타냅니다.

이 회로를 더 자세히 살펴보고 그것이 무엇을 하는지 설명하기 전에, 양자 회로에서 큐비트가 어떻게 정렬되는지 명확히 해야 합니다. 이는 앞서 언급한 Qiskit이 사용하는 큐비트 명명 및 정렬 규칙과 관련이 있습니다.

Qiskit의 큐비트 정렬 규칙

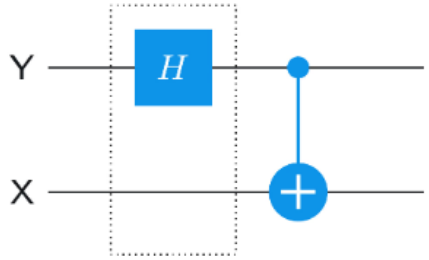
Qiskit에서는 회로 다이어그램에서 가장 위에 있는 큐비트가 인덱스 **0**을 가지며, 이는 큐비트 튜플의 가장 오른쪽 위치(또는 이 튜플에 해당하는 문자열, 카르테시안 곱, 텐서 곱의 위치)에 대응합니다. 두 번째로 위에 있는 큐비트는 인덱스 **1**을 가지며, 이는 튜플에서 두 번째로 오른쪽에 위치합니다. 이렇게 계속 내려가면서 가장 아래에 있는 큐비트는 가장 큰 인덱스를 가지며, 튜플에서 가장 왼쪽 위치에 대응합니다. 특히, Qiskit에서 **n**개의 큐비트를 가진 회로에 대한 기본 이름은 **$(q_{n-1}, \ldots, q_0)$**로 나타내어지며, 이때 **q_0**는 회로 다이어그램에서 가장 위에 있는 큐비트, **q_{n-1}**은 가장 아래에 있는 큐비트에 해당합니다.

이 정렬 규칙에 대한 추가 정보는 Qiskit 문서의 **Bit-ordering** 페이지에서 찾을 수 있습니다.

비록 Qiskit에서 사용하는 기본 이름인 **$q_{n-1}, \ldots, q_0$**는 자주 벗어날 수 있지만, 이 과정에서는 회로 다이어그램을 해석할 때 이 정렬 규칙을 따를 것입니다.

따라서 위의 회로에 대한 해석은 **(X, Y)**라는 두 큐비트에 대한 연산을 설명한다는 것입니다. 만약 회로의 입력이 양자 상태 **$|\psi\rangle\ |\phi\rangle$**라면, 이는 하위 큐비트 **X**가 상태 **$|\psi\rangle$**에서 시작하고, 상위 큐비트 **Y**는 상태 **$|\phi\rangle$**에서 시작한다는 의미입니다. 회로가 무엇을 하는지 이해하려면, 그 연산을 왼쪽에서 오른쪽으로 따라가면 됩니다.

첫 번째 연산은 **Y**에 대한 **Hadamard** 연산입니다.

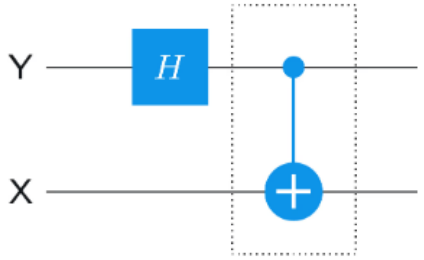


이렇게 단일 큐비트에 게이트를 적용할 때, 다른 큐비트에는 아무 일도 일어나지 않으며 — 아무 일이 일어나지 않는 것은 항등 연산이 수행되는 것과 동일합니다. 우리 회로에는 다른 큐비트 **X**만 있으므로, 위 그림에서 점선 사각형은 이 연산을 나타냅니다:

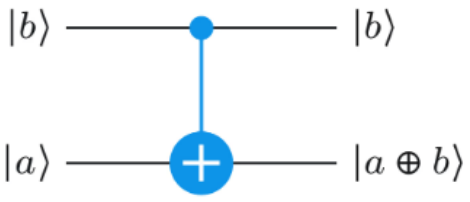
$$\mathbb{I} \otimes H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

항등 행렬은 텐서 곱의 왼쪽에 있고, **H**는 오른쪽에 있습니다. 이는 Qiskit의 정렬 규칙과 일치합니다.

두 번째 연산은 **제어-NOT** 연산으로, **Y**는 제어 큐비트이고 **X**는 타겟 큐비트입니다.



표준 기준 상태에서 제어되지 않는 게이트의 동작은 다음과 같습니다:



큐비트를 다음과 같이 정렬한다고 가정하면 (X,Y), 제어-NOT 게이트의 행렬 표현은 다음과 같습니다:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

전체 회로의 단일 작동을 다음과 같이 호출합니다. U는 연산의 구성입니다:

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}.$$

특히 Bell states에 대한 표기를 상기해 보세요,

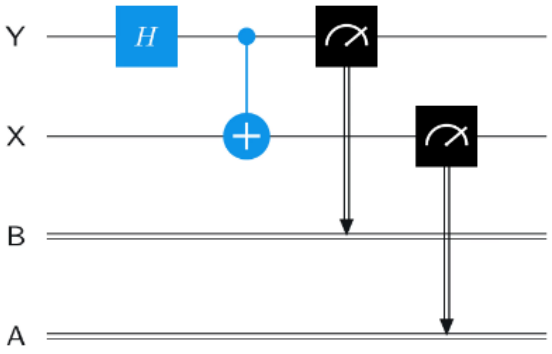
$$\begin{aligned} |\phi^+\rangle &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \\ |\phi^-\rangle &= \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \\ |\psi^+\rangle &= \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \\ |\psi^-\rangle &= \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle, \end{aligned}$$

우리는 얻습니다.

$$\begin{aligned} U|00\rangle &= |\phi^+\rangle \\ U|01\rangle &= |\phi^-\rangle \\ U|10\rangle &= |\psi^+\rangle \\ U|11\rangle &= -|\psi^-\rangle. \end{aligned}$$

따라서 이 회로는 두 개의 큐비트를 |00⟩로 초기화한 상태에서 **|ϕ+⟩** 상태를 생성할 수 있는 방법을 제공합니다. 일반적으로 이 회로는 표준 기저를 벨 기저로 변환하는 방법을 제공합니다. (마지막 상태인 −|ψ−⟩에서의 **−1** 위상 인자는 원한다면 제거할 수 있지만, 이를 위해서는 회로를 변경해야 합니다. 예를 들어, 제어-Z 게이트를 처음에 추가하거나 스왑 게이트를 마지막에 추가하여 다른 세 표준 기저 상태에 대한 회로의 작용에는 영향을 미치지 않으면서 마이너스 기호를 제거할 수 있습니다.)

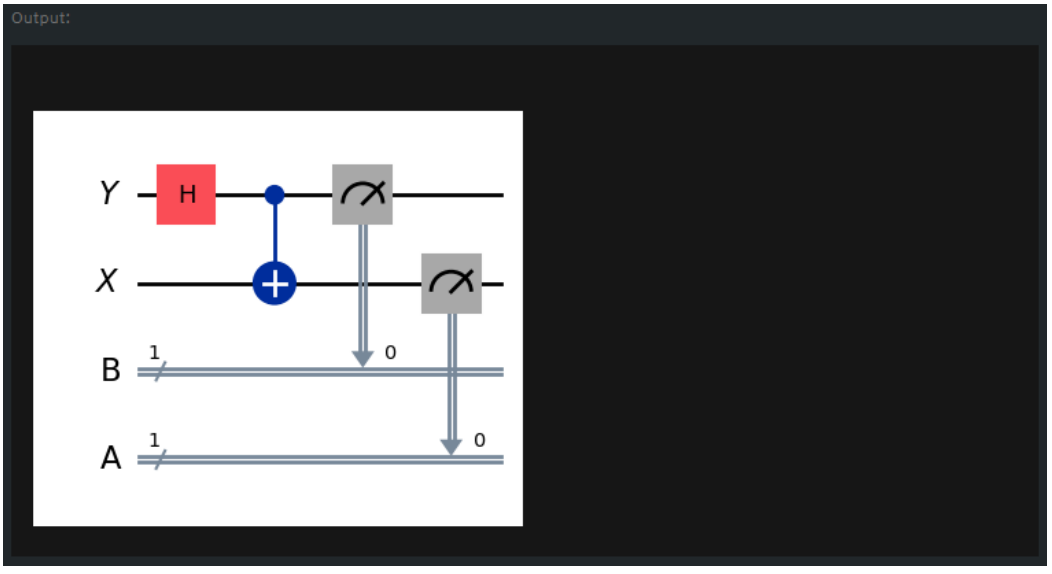
일반적으로, 양자 회로는 여러 개의 큐비트 와이어를 포함할 수 있습니다. 또한, 이 예제처럼 이중선으로 표시된 클래식 비트 와이어를 포함할 수도 있습니다.



이 회로에서는 이전 예제와 마찬가지로 두 개의 큐비트 **X**와 **Y**에 Hadamard 게이트와 제어-NOT 게이트가 적용됩니다. 또한 두 개의 클래식 비트 **A**와 **B**가 있으며, 두 개의 측정 게이트도 있습니다. 측정 게이트는 표준 기저 측정을 나타냅니다: 큐비트는 측정 후 상태로 변하고, 측정 결과는 화살표가 가리키는 클래식 비트에 덮어씌워집니다.

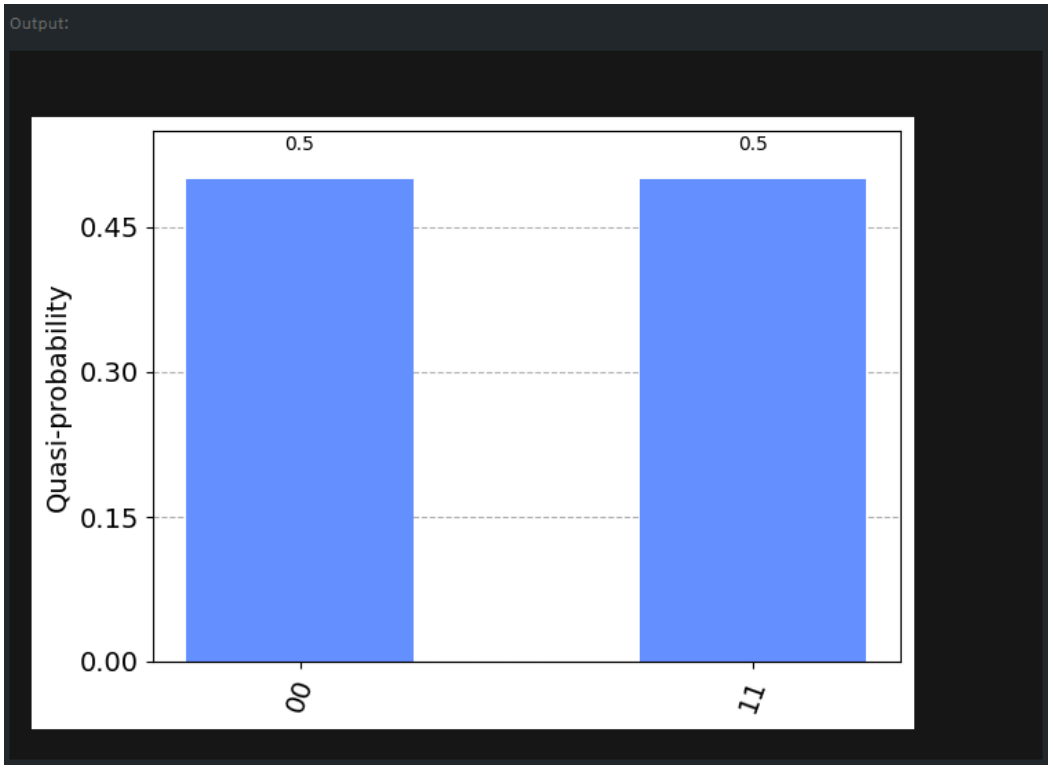
다음은 이 회로를 Qiskit을 사용하여 구현한 예입니다:

```
X = QuantumRegister(1, "X")
Y = QuantumRegister(1, "Y")
A = ClassicalRegister(1, "A")
B = ClassicalRegister(1, "B")

circuit = QuantumCircuit(Y, X, B, A)
circuit.h(Y)
circuit.cx(Y, X)
circuit.measure(Y, B)
circuit.measure(X, A)

display(circuit.draw())
```



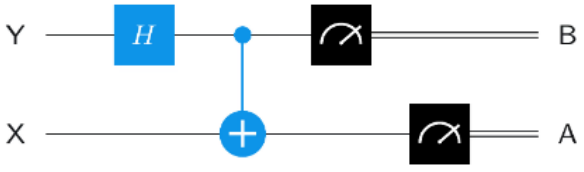
Sampler primitive를 사용하여 회로를 시뮬레이션할 수 있습니다.

```
results = Sampler().run(circuit).result()
statistics = results.quasi_dists[0].binary_probabilities()
display(plot_histogram(statistics))
```



때때로 측정을 큐비트를 입력으로 받아 클래식 비트를 출력하는 게이트로 묘사하는 것이 편리할 때가 있습니다 (즉, 큐비트를 측정 후 상태로 출력하고 결과를 별도의 클래식 비트에 기록하는 대신). 이는 측정된 큐비트가 폐기되었으며 이후에는 무시해도 된다는 의미입니다.

예를 들어, 다음 회로 다이어그램은 이전 다이어그램과 동일한 과정을 나타내지만, 측정 후 **X**와 **Y**를 무시하는 경우입니다:

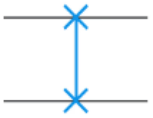


이 시리즈가 진행됨에 따라, 우리는 위의 간단한 예제보다 훨씬 더 복잡한 양자 회로의 많은 예시를 보게 될 것입니다. 다음은 일반적인 게이트에 대한 몇 가지 기호입니다:

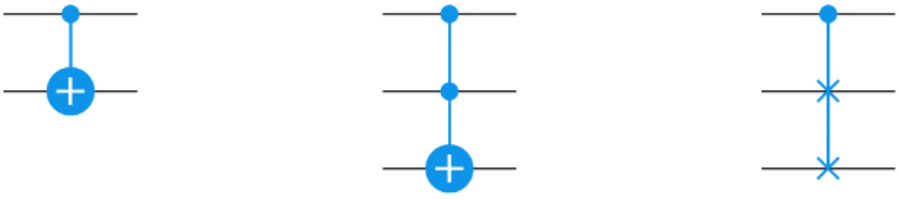
단일 큐비트 게이트는 일반적으로 연산을 나타내는 글자가 있는 사각형으로 표시됩니다, 예를 들어 이렇게:

Not 게이트(또는 X 게이트라고도 불림)는 때때로 더하기 기호 주위에 원으로 표시되기도 합니다:

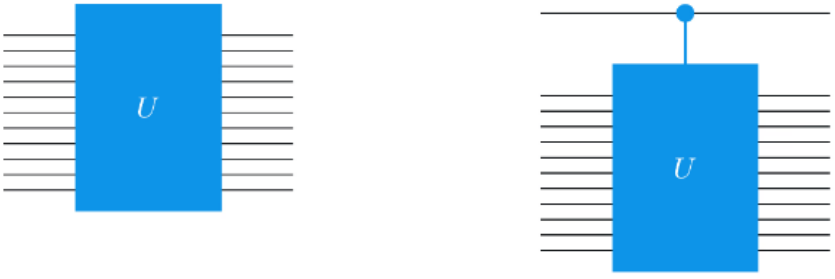
스왑 게이트는 다음과 같이 표시됩니다



제어 게이트, 즉 제어 유니터리 연산을 설명하는 게이트는 채워진 원(제어를 나타냄)과 수직선으로 연결된 연산으로 표시됩니다. 예를 들어, 제어-비트 반전 게이트, 제어-제어-비트 반전 게이트(또는 Toffoli 게이트), 제어-스왑(프레드킨) 게이트는 다음과 같이 표시됩니다:



여러 큐비트에 대한 임의의 유니터리 연산은 게이트로 볼 수 있습니다. 이들은 유니터리 연산의 이름이 적힌 사각형으로 표시됩니다. 예를 들어, 여기에는 (명시되지 않은) 유니터리 연산 U를 게이트로 나타낸 것과 이 게이트의 제어 버전이 있습니다:



정리

이번 강의에서는 **양자 회로 모델**과 관련된 개념들을 소개합니다. 양자 회로는 **유니터리 연산**과 **표준 기저 측정**을 사용하여 양자 계산을 모델링하는 방법입니다. 주요 개념은 **내적, 직교성, 직교 정규성**과 **투영 측정**입니다. 이론적으로는 **양자 상태 복제 불가능성**과 **비직교 양자 상태 구별 불가능성**을 다룹니다.

회로 모델

- **고전적인 회로**는 정보를 **게이트**를 통해 변환하는 모델입니다.
- **불리언 회로**는 이진 값을 처리하며, **AND**, **OR**, **NOT** 게이트를 사용합니다.
- **산술 회로**는 정수 값을 처리하며, 덧셈과 곱셈을 수행할 수 있습니다.

양자 회로

- 양자 회로에서는 **큐비트**를 사용하고, 큐비트에 작용하는 연산은 **유니터리 연산**입니다.

- 예시: **Hadamard** 연산, **S** 연산, **T** 연산 등의 순차적 연산을 포함한 양자 회로.
- 양자 회로는 주로 큐비트가 **|0⟩** 상태로 초기화되며, 입력 큐비트를 다른 상태로 설정할 수도 있습니다.

Qiskit 코드 예시

- 양자 회로를 Qiskit으로 작성하는 방법을 설명합니다:

```
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.h(0)
circuit.s(0)
circuit.h(0)
circuit.t(0)
display(circuit.draw())
```

이 강의는 양자 회로 모델을 통해 양자 계산의 기본 원리와 Qiskit을 활용한 양자 회로 구현 방법을 다룹니다.