2 큐비트 레지스터에서 효율적인 비유니터리 연산자 양자 회로 설계 연구

오지수, 박영훈*
숙명여자대학교

lucy27@sookmyung.ac.kr, *yh.park@sookmyung.ac.kr

Design of an Efficient Two-Qubit Quantum Circuit for a Non-Unitary Operator

Oh Jisoo, Park Younghoon*
Sookmyung Women's University

요 약

본 논문은 2-큐비트 환경에서 비유니터리 연산자를 효율적으로 구현하는 방법을 제안한다. 기존의 비유니터리 연산자의 회로는 비효율적이거나 많은 버려지는 보조 큐비트가 발생하였다. 본 논문에서는 2-큐비트 환경을 위한 비유니터리 연산자의 양자 회로 제안 기술을 제안한다. 기존 방식보다 버려지는 큐비트의 수를 줄이고, 양자 회로를 좀더 간소화하였다. 또한, 임의 수의 큐비트를 위한 비유니터리 연산자는 2-큐비트 비유니터리행렬로 분해할 수 있으므로 본 논문의 기술을 이용하여 임의의 비유니터리행렬도 효율적으로 구현할 수 있을 것으로 기대한다.

Ⅰ. 서 론

양자 컴퓨터는 고전 컴퓨터의 비트 대신 큐비트를 기반으로 중첩과 얽힘와 같은 양자역학적 성질을 활용하여 고전적인 계산 방법으로는 실행이 어려운 높은 수준의 계산 역량을 제공한다. 암호 해독, 양자 시뮬레이션 등 다양한 응용 분야에서 그 잠재력이 대두되고 있다. 최근 양자 알고리즘 설계, 오류 보정 기법, 하드웨어 아키텍처 등 다양한 측면에서 활발한 연구가 진행되고 있지만, 아직까지 하드웨어의 물리적 제약으로 인해 회로를 효율적으로 설계하는 문제는 여전히 핵심적인 과제로 남아있다.

양자 회로의 모든 기본 연산은 양자 역학의 유니터리성에 의해 유니터리 행렬로 표현되어야 한다. 이러한 연산은 폐쇄된 계에서의 순수한 양자 상태의 진화에는 적합하지만, 계산 중간에 측정을 수행하거나, 실제 양자 시스템에서 발생하는 양자 결맞음과 노이즈 현상 등을 표현하기에는 한계가 있다[1]. 또한, 기존의 유니터리 연산자로만 구성된 양자회로에서 효율성을 위하여 비유니터리 연산자를 사용할 수도 있다. 따라서, 비유니터리 연산자가 현실적인 양자 회로 설계와 시뮬레이션에서 중요한 대안으로 제시되고 있다.

비유니터리 연산자를 구현하기 위한 기술은 꾸준히 제안되어왔으나, 효율성이나 버려지는 보조큐비트의 수 문제가 발생할 수 있다[1,2,3]. 본 논문에서는 2-큐비트 환경에서 효율적으로 동작할 수 있는 양자회로의 구현 방법을 제안한다.큐비트가 여러 개인 비유니터리 연산자도 2-큐비트짜리 비유니터리 연산으로 분해할 수 있기 때문에, 본 논문의 기술을 이용하여 임의의 비유니터리 연산자도 효율적으로 구현할 수 있을 것으로 보인다.

Ⅱ. 비유니터리 행렬

정사각행렬 A에 대하여, A의 complex conjugate 를 $A^{\dagger}$라고 하자. $AA^{\dagger} = A^{\dagger}A = I$일 때, 정사각행렬 A을 유니터리 행렬이라고 하며, 이를 만족하지 않는 행렬을 비유니터리 행렬이라고 한다. 양자회로로 비유니터리행렬을 구현하는 방법은 많이 제안되었지만, 공통적으로 SVD (Singular Vector Decomposition)를 이용한 분해로 시작한다. A가 비유니터리행렬일 때, $A = UDV$를 만족하는 두 유니터리행렬 U, V와 대각행렬 D를 찾는다. U와 V는 유니터리 연산자이므로 양자회로로 구현 가능하며, D는 비유니터리행렬이기 때문에 특수한 처리가 필요하다.

D를 구현하기 위하여 여러 가지 방법이 제안되고 있지만, 가장 널리 쓰이는 방법은 제어연산자들의 곱으로 분해하고, 각각을 따로따로 구현하는 방법이다. 이 때, 제어연산자의 연산 파트는 $N(a) = \begin{bmatrix} 1 & 0 \\ 0 & a \end{bmatrix}$ (단, $|a| \leq 1$)형태이며, $N(a)$는 다음 그림 1 과 같이 구현한다.

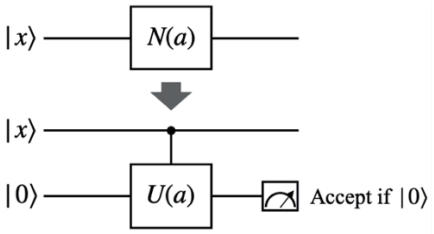

그림 1. 1-큐비트 비유니터리연산자 구현 예

여기서, $U(a) = \begin{bmatrix} a & \sqrt{1-a^2} \\ -\sqrt{1-a^2} & a \end{bmatrix}$로, 유니터리 연산자이다. $|x\rangle = \alpha|0\rangle + \beta|1\rangle$일 때, 보조 큐비트에 $|0\rangle$을 입력하면 전체 입력은 $\alpha|00\rangle + \beta|10\rangle$이 된다. 그러면 출력으로는 $\alpha|00\rangle + a\beta|10\rangle - \beta\sqrt{1-a^2}|11\rangle$이 계산되는데, $\alpha|00\rangle + a\beta|10\rangle$만이 우리가 원하는 결과이다. 따라서 두 번째 큐비트를 관측하여 $|0\rangle$이면 연산을 계속 진행하고, $|1\rangle$이면 실패한 것이므로 다시 수행한다. 이 때, 하나의 비유니터리 연산자를 수행하기 위해서는 보조 큐비트 하나가 버려지게 된다.

Ⅲ. 기존의 2-큐비트 비유니터리 행렬 계산 방식

2-큐비트를 처리하기 위한 비유니터리 행렬은 4×4 행렬이며, 이를 SVD 를 이용해서 두 개의 유니터리 행렬과 하나의 대각행렬의 곱으로 표현할 수 있다. 이 중, 유니터리 행렬은 양자 회로로 구현이 가능하며, 대각행렬이

비유니터리행렬이기 때문에 이를 구현하기 위해서는 특별한 방법이 필요하다. 비유니터리인 대각행렬을 구하기 위한 기술들은 몇 차례 제안되어 왔다.

$D(d_1, d_2, d_3, d_4)$를 대각성분이 d_1, d_2, d_3, d_4인 비유니터리 대각행렬이라고 하자. 어떤 입력 2-큐비트 $|x\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$에 대하여(단, α, β, γ, δ는 절댓값의 제곱의 합이 1인 복소수들), $D(d_1, d_2, d_3, d_4)|x\rangle$은 다음과 같이 계산된다:

$$\frac{d_1\alpha|00\rangle + d_2\beta|01\rangle + d_3\gamma|10\rangle + d_4\delta|11\rangle}{\sqrt{|d_1\alpha|^2 + |d_2\beta|^2 + |d_3\gamma|^2 + |d_4\delta|^2}}$$

즉, 00, 01, 10, 11이 관측될 확률의 비율이 $|\alpha|^2 : |\beta|^2 : |\gamma|^2 : |\delta|^2$ 이었던 양자 상태가 $|d_1\alpha|^2 : |d_2\beta|^2 : |d_3\gamma|^2 : |d_4\delta|^2$로 업데이트가 되는 것이다. 이를 위한 주요 연구로 Terashima등이 제안했던 방식과 Lin 등이 제안했던 방식이 있다.

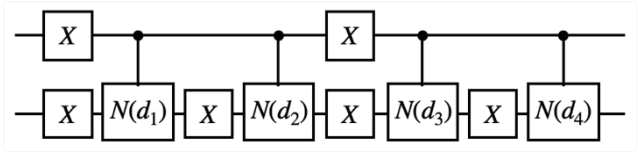

그림 2. Terashima 등이 제안한 비유니터리연산자 회로

우선, Terashima[1]등이 제안했던 방식은 대각행렬을 다음과 같이 곱의 형태로 분해하는 것으로 시작된다:

$$D(d_1, 1, 1, 1) \cdot D(1, d_2, 1, 1) \cdot D(1, 1, d_3, 1) \cdot D(1, 1, 1, d_4)$$

각각의 행렬은 controlled-non-unitary 연산자로 생각할 수 있고, 이를 이용하여 그림 2 와 같이 회로를 구성할 수 있다. 여기서 $N(d_i) = \begin{bmatrix} 1 & 0 \\ 0 & d_i \end{bmatrix}$으로, 역시 비유니터리연산자이다. 이를 구현하기 위하여 보조 큐비트를 사용한다.

Lin[3] 등의 방식은 다음과 같다. $D(d_1, d_2, d_3, d_4)$을 d_1, d_2, d_3, d_4 중 절댓값이 가장 큰 값으로 나눠주고, 그 행렬을 P라고 하자. 그러면 대각 성분 중 하나는 1이고, 나머지 대각성분들은 모두 절댓값이 1 이하가 된다. 대각행렬을 곱했을 때, 각 양자상태 계수의 비율만 원하는 대로 나오면 되기 때문에 $D(d_1, d_2, d_3, d_4)$ 대신 P를 곱해도 무방하다. 그런데 P는 비유니터리 행렬이므로 보조 큐비트 하나를 더 도입하여 유니터리 행렬이 되게끔 만들어준다. 그러면 큐비트가 총 3 개 필요하기 때문에 8×8 유니터리 행렬 $U_P = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}$를 만들어준다. 이 때, 유니터리 행렬로 만들어주기 위해서는 $QQ^\dagger = RR^\dagger = I - PP^\dagger$, $SS^\dagger = PP^\dagger$, $PR^\dagger + QS^\dagger = P^\dagger Q + R^\dagger S = 0$을 만족하는 Q, R, S를 찾아야 한다.

Ⅳ. 제안 방식

본 섹션에서는 기존 방식들 대비 조금 더 효율적으로 2-큐비트 비유니터리 행렬을 계산하는 방법을 제안한다. 본 기술은 Terashima 등이 제안한, 네 개의 곱으로 나타내는 방법 대신 두 개의 곱으로 나타내는 방법을 제안한다. 우선, SVD 를 이용하여 비유니터리 행렬로부터 대각행렬을 추출하며, 그 대각행렬을 $D(d_1, d_2, d_3, d_4)$라고 하자. 그리고, 이 대각행렬에서 d_1, d_2, d_3, d_4 중 절댓값이 가장 큰 것으로 나눈 결과를 $D(d_1', d_2', d_3', d_4')$라고 하자. 그러면, d_1', d_2', d_3', d_4' 중 하나는 1이고, 나머지 셋은 절댓값이 1 이하가 된다. 그러면, $D(d_1', d_2', d_3', d_4')$를 회로로 구현하면 되고, 본 논문에서 $D(d_1', d_2', d_3', d_4')$는 다음 수식을 이용하여 구현한다:

$$D(d_1', d_2', d_3', d_4') = D(d_1', d_2', 1, 1) \cdot D(1, 1, d_3', d_4')$$

여기서, $D(d_1', d_2', 1, 1)$는 첫 번째 큐비트가 $|0\rangle$일 때 두 번째 큐비트에 $D(d_1', d_2')$를 적용하는 연산자이고, 반대로 $D(1, 1, d_3', d_4')$는 첫 번째 큐비트가 $|1\rangle$일 때 두 번째 큐비트에 $D(d_3', d_4')$를 적용하는 연산자이다. 따라서, 다음 그림 3 과 같이 회로로 구성할 수 있다.

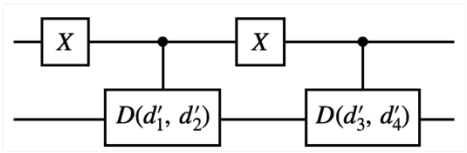

그림 3. 제안 비유니터리 연산자 회로

이제, $D(d_i', d_j')$를 구현할 수 있으면 된다. 이는 $|0\rangle$이 입력으로 들어가는 보조큐비트를 추가하고, $D(d_i', d_j')$를 대신할 수 있는 4×4 행렬을 구성한 뒤, 이를 통과시킨 결과 보조큐비트가 $|0\rangle$이면 결과를 받아들이고, $|1\rangle$이면 거절하는 방식으로 구현하면 된다. $D = D(d_i', d_j')$이라 할 때, 다음과 같이 U_D를 정의하자:

$$U_D = \begin{bmatrix} d_i' & \sqrt{1 - d_i'^2} & 0 & 0 \\ -\sqrt{1 - d_i'^2} & d_i' & 0 & 0 \\ 0 & 0 & d_j' & \sqrt{1 - d_j'^2} \\ 0 & 0 & -\sqrt{1 - d_j'^2} & d_j' \end{bmatrix}$$

그러면 U_D는 유니터리행렬이며, $D = D(d_i', d_j')$에 대한 회로는 다음과 같이 구성하면 된다:

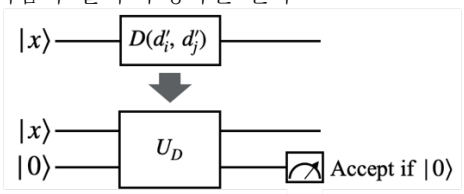

그림 4. 대각 1-큐비트 비유니터리행렬 구현

위의 구조가 가능한 이유는 $|x\rangle = \alpha|0\rangle + \beta|1\rangle$일 때, 보조 큐비트 $|0\rangle$을 추가하면 변형된 회로는 $\alpha|00\rangle + \beta|10\rangle$이 되며, 여기에 U_D를 적용하면 $d_i'\alpha|00\rangle - \alpha\sqrt{1 - d_i'^2}|01\rangle + d_j'\beta|10\rangle - \beta\sqrt{1 - d_j'^2}|11\rangle$이 된다. 이 때, 두 번째 비트를 관측하여 $|0\rangle$이 나오면 원하는 결과가 나오게 되기 때문이다.

Ⅴ. 결론 및 향후연구

본 논문에서는 2-큐비트 환경을 위한 비유니터리 연산자의 효율적인 구현 방법을 제안하였다. 기존의 구현 방식 보다 좀더 단순하게 회로를 구성할 수 있었으며, 필요한 보조 큐비트의 수도 다소 감소시킬 수 있었다. 향후에는 비유니터리 연산자의 성공 확률과 깊이, 연산자 수, 보조 큐비트 등 다양한 측정 수단들을 활용하여 기존 비유니터리 연산자 구현 기술과 비교분석을 하고자 한다.

참 고 문 헌

[1] Terashima H. and Masahito U. "Nonunitary Quantum Circuit." International Journal of Quantum Information 03 (2003): 633-647.

[2] Zylberman J., Nzongani U., Simonetto A., and Debbasch F.. "Efficient Quantum Circuits for Non-Unitary and Unitary Diagonal Operators with Space-Time-Accuracy trade-offs." 2024. hal-04554614

[3] Lin S., Dilip R., Green A., Smith A., and Pollmann F., "Real- and Imaginary-Time Evolution with Compressed Quantum Circuits," PRX Quantum, vol. 2, issue. 1, 2021