

중간회로측정과 큐비트 재사용의 효율성에 관한 연구

민상원, 이정수*

부산외국어대학교, *부산외국어대학교

alstkddnjs98@gmail.com, *rhee@bufs.ac.kr

A Study on the efficiency of Mid-circuit measurement and qubit reuse

Sangwon Min, Jungsoo Rhee*

Busan University of Foreign Studies., *Busan University of Foreign Studies.

요약

양자 회로를 구현할 때에는 사용하는 게이트의 수와 회로의 깊이, 사용하는 큐비트의 수 등을 고려해야 한다. 본 논문에서는 이전에 구현한 양자 RCA(Ripple Carry Adder)회로 상에서 큐비트 재사용과 중간회로측정을 사용했을 때 얼마나 효율성이 증가되는지에 대해 연구하였다. 그 결과 회로의 depth와 size는 증가했지만 보조큐비트의 수가 감소되었다.

I. 서론

양자컴퓨터는 양자역학의 원리를 기반으로 작동하는 컴퓨팅 기술로, 기존의 고전적 컴퓨터와는 근본적으로 다른 계산 방식을 사용한다. 고전적 컴퓨터가 이진수 비트를 사용하여 데이터를 처리하는 반면, 양자컴퓨터는 큐비트를 사용하여 0과 1의 중첩 상태를 동시에 표현할 수 있다. 이로 인해 양자컴퓨터는 특정 문제에서 고전적 컴퓨터에 비해 지수적인 속도 향상을 제공할 수 있으며, 특히 복잡한 최적화 문제, 암호 해독, 분자 시뮬레이션 등에서 강력한 성능을 발휘할 것으로 기대된다.[1]

양자컴퓨터에 작업을 하기 위해선 고전 컴퓨터와 같이 코딩을 해야 하는데, 다양한 양자 컴퓨팅 플랫폼이 양자 코딩을 위한 QDK(Quantum Development Kit)를 지원하고 있다. 이를 통해 개인이 로컬에서 직접 양자 코딩을 작성하고 시뮬레이터 환경을 통해 결과를 예측할 수 있다. [2][3] 또한 개발자는 고전 컴퓨터에서와 같이 코딩을 할 때 최소한의 비용으로 회로를 구축하는 것을 고려해야 한다. IBM은 이를 위해 중간회로 측정과 큐비트 재사용 기능을 소개하였다.[4]

본 연구는 IBM Qiskit을 활용하여 기존의 양자 RCA 회로에 중간회로 측정과 큐비트 재사용을 적용하여 보조큐비트의 수를 줄이는 강화된 양자 RCA 회로로 구현하는 과정을 보여준다.[5]

II. 본론

2-1. 양자 회로의 복잡성

양자 회로의 구성이 얼마나 효율적인지 나타낸 값을 양자회로의 복잡성이라고 한다. 이 복잡성은 회로상에 사용된 게이트 수(size), 연속적인 양자 게이트의 연산 횟수(depth)[6], 그리고 회로의 사용에 필요한 큐비트의 수에 의해 판단된다. [7]

2-2. 중간회로측정과 큐비트 재사용

IBM은 측정을 회로 마지막에 하는 것이 아닌 회로 중간에 할 수 있는 기능을 소개하였다. [4] 이를 통해 회로 중간에 큐비트의 상태를 측정하여 오류 수정을 용이하게 하는 등의 이득이 있다고 주장하였다. 또한, 회로상에서 한번 사용한 후 더 이상 사용하지 않는 큐비트를 다시 사용하여 보조

큐비트(Ancilla qubit) 수를 최소화 할 수 있는 큐비트 재사용 기능또한 소개하였다.

2-3. 중간회로측정과 큐비트 재사용 활용

그림 1에 의하면 입력받은 값의 크기(num_qubits)에 따라 보조큐비트(Cin, Cout, S)의 개수도 증가됨을 볼 수 있다. 이렇게 할당된 보조큐비트들은 각 단계가 실행될때만 한번 사용되고 다음 단계가 되면 사용되지 않는다. 예를 들어, I번째 보조큐비트들은 I+1번째가 되면 무기한 대기상태가 된다. 이러한 보조큐비트의 과도한 사용은 비용의 증가 뿐만 아니라 양자 회로의 오류 발생률을 증가시킬 수 있다.

```
def adder_circuit(num_qubits, num1, num2):
    A = QuantumRegister(num_qubits, 'a')
    B = QuantumRegister(num_qubits, 'b')
    Cin = QuantumRegister(num_qubits+1, 'cin')
    Cout = QuantumRegister(num_qubits, 'cout')
    S = QuantumRegister(num_qubits, 's')
    result = ClassicalRegister(num_qubits+1, 'result')

    qc = QuantumCircuit(A, B, Cin, Cout, S, result)
    bin_num1 = bin(num1)[2:].zfill(num_qubits)
    bin_num2 = bin(num2)[2:].zfill(num_qubits)
    for i in range(num_qubits):
        if bin_num1[num_qubits - 1 - i] == '1':
            qc.x(A[i])
        if bin_num2[num_qubits - 1 - i] == '1':
            qc.x(B[i])

    for i in range(num_qubits):
        qc.cx(A[i], S[i])
        qc.cx(B[i], S[i])
        qc.cx(Cin[i], S[i])
        qc.ccx(A[i], B[i], Cout[i])
        qc.ccx(B[i], Cin[i], Cout[i])
        qc.ccx(A[i], Cin[i], Cout[i])
        qc.cx(Cout[i], Cin[i+1])
        qc.measure(S[i], result[i])
        if i+1 == num_qubits:
            qc.measure(Cout[i], result[num_qubits])
    return qc
```

그림 1 기존의 양자 RCA 회로

```
def adder_circuit(num_qubits, num1, num2):
    A = QuantumRegister(num_qubits, 'a')
    B = QuantumRegister(num_qubits, 'b')
    Cin = QuantumRegister(1, 'cin')
    Cout = QuantumRegister(1, 'cout')
    S = QuantumRegister(1, 's')
    result = ClassicalRegister(num_qubits + 1, 'result')

    qc = QuantumCircuit(A, B, Cin, Cout, S, result)
    bin_num1 = bin(num1)[2:].zfill(num_qubits)
    bin_num2 = bin(num2)[2:].zfill(num_qubits)
    for i in range(num_qubits):
        if bin_num1[num_qubits - 1 - i] == '1':
            qc.x(A[i])
        if bin_num2[num_qubits - 1 - i] == '1':
            qc.x(B[i])

    for i in range(num_qubits):
        qc.cx(A[i], S[0])
        qc.cx(B[i], S[0])
        qc.cx(Cin[0], S[0])
        qc.ccx(A[i], B[i], Cout[0])
        qc.ccx(B[i], Cin[0], Cout[0])
        qc.ccx(A[i], Cin[0], Cout[0])
        qc.reset(Cin[0])
        if i < num_qubits - 1:
            qc.cx(Cout[0], Cin[0])
            qc.reset(Cout[0])
        qc.measure(S[0], result[i])
        qc.reset(S[0])
    qc.cx(Cout[0], S[0])
    qc.measure(S[0], result[num_qubits])
    print(qc)
    return qc
```

그림 2 중간회로측정과 큐비트 재사용 적용 후 RCA 회로

적용	depth	size	CX gate	CCX gate	Ancilla
전	$4n+2$	$7n$	$4n$	$3n$	$3n+1$
후	$6n$	$10n-1$	$4n$	$3n$	3

표 1 큐비트 재사용 적용 전후 양자회로 비용 비교

중간회로측정과 큐비트 재사용을 적용한 그림2에 의하면 보조큐비트의 수가 입력값의 크기와 상관없이 고정(Cin, Cout, S 각 1개씩, 총 3개)된 것을 볼 수 있다. 또한 큐비트는 처음 할당되면 기본적으로 0 상태로 초기화 되는데, 각 단계가 끝날 때 마다 reset 함수를 통해 Cin, Cout, S 보조큐비트들을 0으로 초기화시켜 처음 할당받을 때와 똑같은 환경으로 만들어준다. 이를 통해 1개의 큐비트를 여러번 재사용 할 수 있다. 결과를 저장하는 S 큐비트는 중간회로측정 기능에 의해 회로 마지막에 한번만 측정되는 것이 아닌 회로 중간에 여러번 측정되는 모습을 보여준다

표 1에 의하면 기존의 코드보다 $3n$ 보조큐비트가 감소되었다. 다만, 깊이가 $4n+2$ 에서 $6n$ 으로 증가되었고, cx gate와 ccx gate의 수는 바뀌지 않았지만 reset으로 인해 size가 $7n$ 에서 $10n-1$ 로 증가하였다.

III. 결론

본 논문에서는 중간회로측정과 큐비트 재사용의 활용이 얼마나 양자회로 복잡성에 효과적인지 양자 RCA 회로를 통해 알아보았다. reset 함수의 추가로 인해 양자회로의 depth와 size는 증가하였지만 보조큐비트의 수가 크게 줄어들었음을 알 수 있다.

ACKNOWLEDGMENT

“본 연구는 과학기술정보통신부 및 정보통신기획평가원의 ICT혁신인재 4.0 사업의 연구결과로 수행되었음” (IITP-2024-2020-0-01825)

참 고 문 헌

- [1] Charles Bennett, Ethan Bernstein, Gilles Brassard, Umesh Vazirani, Strengths and weaknesses of quantum computing, SIAM J. Comput., 26 (1997), 1510 - 1523
- [2] Microsoft, “Azure Quantum documentation” 2019, (<https://learn.microsoft.com/en-us/azure/quantum/>)
- [3] IBM qiskit, “Qiskit SDK”, 2020, (<https://github.com/Qiskit/qiskit>)
- [4] IBM qiskit, “How to measure and reset a qubit in the middle of a circuit execution”, 2021, (<https://www.ibm.com/quantum/blog/quantum-mid-circuit-measurement>)
- [5] S.W. Min, J.S. Rhee, “Introduction to Quantum Circuit Implementation Using IBM Qiskit,” The Journal of Korean Institute of Communications and Information Sciences, vol.49, no.7, pp. 1030-1038, Jul. 2024.
- [6] IBM Quantum Documentation, “Qiskit SDK - Circuit properties”, 2020, (<https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit#depth>)
- [7] Y. Takahashi, S. Tani, and N. Kunihiro, “Quantum addition circuits and unbounded fan-out,” Quantum Inf. and Computation, vol.10, no. 9, pp. 872-890, Sep. 2010.