특 집

양자 전산의 기초와 응용

김 명 식
서강대학교 물리학과

1. 왜 전자계산에 양자물리가 필요한가?

사람들은 언제나 더욱 더 작은 그리고 더욱 더 빠른 컴퓨터를 원한다. 지금까지는 컴퓨터의 발전이 우리의 기대를 충족시키는데 큰 문제가 없었다. 하지만 약 2020년 이후가 되면 상황이 다르게 될 것이라 예측한다[1]. Intel의 설립자인 Gordon Moore는 마이크로 칩에 집적되는 트랜지스터의 개수는 2년마다 배로 증가한다고 하는 Moore의 법칙을 말하였다. Robert Kyese는 한 비트의 정보를 저장하는데 필요한 전자와 같은 단위 매체의 수를 시간의 흐름에 따라 분석하였다[2]. 그 분석을 보면 약 2020년 후에는 1개의 원자에 1비트의 정보를 저장하여야 한다. 칩의 고집적화로 양자효과는 피할 수 없게 된다.

집적화에 필수적으로 따르는 또 하나의 문제는 열이다. 퍼스널 컴퓨터의 내부를 보면 컴퓨터 동작시 발생하는 열을 식히기 위해 냉각팬이 붙어있는 것을 볼 수 있다. 게다가 수퍼 컴퓨터의 경우에는 열을 식히기 위해 수냉식 냉각기를 따로 두고 있다. 열은 칩의 집적화뿐만 아니라 클럭 주파수의 문제이기도 하여 집적화가 이루어 질수록, 클럭 주파수가 커질수록 열 방출이 많아진다. 열 방출은 클럭 주파수의 제곱에 비례하여 늘어난다. 클럭 주파수는 2020년에 40 GHz가 될 것으로 예상되며, 이 때에는 칩으로 연산을 할 때 하나의 연산을 하기가 무섭게 칩이 녹아버리게 된다.

1970년대부터 몇명의 과학자들이 물리학과 계산과의 관계에 관심을 가지기 시작하였다. 그들의 연구는 주로 열역학과 계산과정 사이의 관계를 이해하는 것이었다[3,4]. '계산을 하는데 얼마나 열을 방출해야 하는가, 열방출의 최소한도가 존재하는가, 그리고 계산속도의 한계는 있는가?'와 같은 것이었다. IBM Thomas J. Watson 연구소의 Charles Bennett는 비가역적 논리연산은 필수적으로 열방출을 해야만 한다는 결과를 발표하였다[5]. 예를 들어 표 1에서 보는 것 처럼, AND 게이트를 보면 입력값은 2개인데 출력값은 1개이다. 만약 출력이 1이라면 입력값이 1과 1이라는 것을 알 수가 있지만 출력이 0이라면 입력값을 알 수가 없다. 결국 정보를 계산 중에 잃게 되어 그 계산과정은 비가역적이 된다. 사실 출력의 수가 입력의 수보다 작으면 출력으로부터 입력값을 알 수 없기 때문에 정보를 잃게 된다. 우리가 그러한 논리적 비가역 게이트를 쓴다면 필수적으로 에너지는 방출되어야 한다. 다른 여러가지 연산에서 자주 쓰이지만 한 비트(bit)의 메모리를 지우는 것도 에너지를 방출해야 한다. 비록 한 비트의 메모리를 지우지만 사실상 그 이전의 모든 정보를 지우는 것이다.

표 1. *AND 게이트*

입 력		출 력
0	0	0
0	1	0
1	0	0
1	1	1

현재는 컴퓨터에서 1과 0을 표시하는데 전자다발을 쓰지만 한개의 전자가 있고 없음에 따라 1과 0이 구별이 된다면 불확정성의 원리 등 양자적 효과를 무시할 수 없게 된다. 1과 0을 확실히 구분하는 고전적인 알고리즘으로 계산을 하려면 이 양자효과들은 아주 큰 걸림돌이 될 수 밖에 없다. 하지만 놀랍게도 양자 이론을 이용하여 작동하는 '양자 컴퓨터'는 가역적인 연산을 통하여 열역학 제 2 법칙에 따라 열방출이 없게 할 수 있을 뿐만 아니라[6], 원자 단위에서의 연산에서 앞의 현상들을 긍정적으로 수용하게 된다.

이 논문에서는 편의상 현재 널리 통용되고 있는 컴퓨터를 고전 물리에 근거하고 있다고 해서 고전 전산이라 하고, 새로이 양자물리에 근거하는 전산 이론을 양자 전산이라 부르겠다.

2. 투링 기계(Turing machine)

1.1 고전 투링 기계

*1999*년 3월 New York Times가 뽑은 20세기를 움직인 20인 가운데 영국의 수학자 Alan Turing이 끼어 있는 것을 보고 문외한들은 깜짝 놀라지 않을 수 없었다. 하지만 그가 오늘날 세

계를 움직이는 기계 '컴퓨터'의 알고리즘의 초석을 놓은 사람임을 알면 '아! 그렇구나'할 것이다.

컴퓨터가 할 수 있는 일은 무엇인가? 그리고 할 수 없는 일은 무엇인가? 이 질문에 관하여, 컴퓨터의 기종이나 구현방법에 무관하게 적용할 수 있는 이론이 바로 Turing의 이론이다. 1900년 독일의 수학자 Hilbert는 파리에서 열린 국제 수학회의에서 23가지의 도전해 볼 만한 문제를 제시하였다[7]. 그 중 마지막 문제인 23번째가 "어떠한 수학적인 문제라도 그 문제의 참, 거짓을 판별할 수 있는 물리적 절차가 존재하는가?"라는 문제이다. 이는 *Entscheidungsproblem*이라는 이름으로 널리 알려지게 되었다. Turing은 이 문제를 연구하다, 투링 기계(Turing machine)라는 고전적 전산의 가장 기본적인 모델을 만들었다. Turing은 계산이라고 간주할 수 있는 어떤 과정을 모형화 할 수 있는 시스템을 개발하였는데 이것은 현재의 컴퓨터가 개발되기 훨씬 이전이었다. Turing은 연필과 종이를 가진 사람에 의해서 수행되는 계산을 상상했다[8]. Turing은 종이가 구역들로 나뉘어져 있고, 각 구역은 하나의 기호를 기록하는데 필요한 종이의 범위라고 생각하였다. 그리고 이 상황에서 사람은 어떤 특정한 시간에 종이의 제한된 부분에만 신경을 집중할 수 있다고 판단했다. Turing은 또한 어떤 계산 과정도 단지 유한한 개수의 기호들만 포함할 수 있는 것으로 결론을 내렸다. 왜냐하면 각 기호는 공간의 일정한 범위 내에 기록될 수 있어야만 하기에, 기호들의 개수가 점점 많아지게 되면 서로 다른 기호들을 구별하는 특징들은 계속하여 줄어들 수 밖에 없고, 그렇게 되면 어떤 시점에서 인간은 상이한 기호들간의 구별을 못하기 시작하고, 그 결과 유효한 기호들의 개수는 제한되게 될 것이라고 보았기 때문이다.

종이의 특정한 구역을 고려할 때, 사람은 그 구역의 내용을 고치거나 또는 다른 구역으로의 이동을 할 수 있을 것이라고 Turing은 생각하였다. 어떤 동작이 일어날 것인가 하는 점과 그 동작의 세부사항은 그 구역의 현재 기호와 그 사람의 마음 상태에 달려 있을 것이다. 기호들의 개수에서 처럼, 인간은 단지 유한개의 분별가능한 마음의 상태를 가질 수 있을 것이다. 또한, 사람은 계산을 시작할 때 특별한 초기 상태에 있고 계산이 완료되었을 때 정지상태라고 정하는 상태에 있을 것이다. 종이의 한계가 그 모델의 능력을 제한하는 것을 막기 위해서, Turing은 계산에 이용 가능한 종이의 양을 무제한으로 하도록 제안하였다.

투링 기계를 구체적으로 형상화하면 다음과 같다. 투링 기계는 테이프와 헤드로 구성된다. 무한히 긴 테이프는 단위간격으로 구역이 정해져 있고 그 구역에는 알파벳같은 정보가 쓰여있다. 그리고 헤드는 테이프상의 정보와 헤드의 현재 상태를 근거로 좌우로 움직이거나 또는 정보를 테이프에 읽거나 쓰거나 할 수 있다. 투링 기계의 작동절차는 다음과 같다.

1. 시작
2. 테이프를 읽는다.
3. 입력정보와 헤드의 위치에 따라 출력을 테이프에 기입한다.
4. 출력기입후 헤드가 좌, 우로 움직이거나 혹은 정지한다.
5. 헤드가 정지의 위치에 갈 때까지 이 작업을 계속한다.

아무도 Turing의 계산모델보다 더 강력하고 널리 인정받는 계산 모델을 제안할 수 없었다는 점은 의미심장하다. 투링 기계는 오늘날의 컴퓨터보다 실제로 더 일반적이다. 왜냐하면 투링 기계는 저장공간의 부족으로 제한받지 않지만, 실제 모델에서는 결국 저장공간의 한계가 있기 때문이다. 투링 기계는 원래 연필과 종이를 가지고 계산하는 사람을 모델로 하였지만 전자계산 장비의 출현으로, 이 기계는 자기 테이프상에 읽고 쓰는 전기 전자기계에 자리를 내주었다. 이른바 우리가 말하는 컴퓨터이다.

결국 문제를 해결할 수 있는 알고리즘만 있으면 컴퓨터를 이용하여 유한시간 내에 문제를 풀 수 있다는 것을 투링 기계의 원리를 통해 알 수 있다. 하지만 Turing은 투링 기계의 개념을 제공하였지 실제 무엇을 이용하여 구현해야 하는가에 대해서는 제약을 두지 않았다. 결국 사람이든, 전자컴퓨터이든, 아니면 다른 어떠한 것으로 구현된 것이든 투링 기계의 개념만 따르고 알고리즘만 있으면 유한 시간내에 문제를 해결할 수 있다는 결론을 내릴 수 있다.

2.2 양자 투링 기계(Quantum Turing Machine)

IBM Thomas J. Watson 연구소의 Charles Bennett는 1973년 고전적 가역 투링 기계가 가능함을 보였다[9]. 투링 기계가 가역적이라면 한 시점에서 과거 연산의 정보를 잊어버리지 않고 연산을 가역적으로 수행할 수 있음을 의미하는 것이다. 이렇게 하면 회로에서 발생하는 열이 없는 투링 기계를 만들 수 있게 된다. 이를 바탕으로 Paul Benioff는 1980년에 처음으로 양자적 개념의 투링 기계를 제안하였다[10]. 고립된 양자 시스템은 동시에 여러 갈래로의 상태가 가능함을 Benioff는 관심을 갖고 주목했다. 그리고 고전적으로 가역적인 투링 기계와 비슷하게 작동하는 양자 시스템을 만들 수 있음을 알 수 있었는데 이를 제안한 것이 Benioff가 한 일이다. 하지만 Benioff 기계는 진정한 의미의 양자컴퓨터가 아니다. 왜냐하면 계산중간에는 중첩된 상태에 있다 하더라도 결국에는 고전적 상태 중의 하나의

결과가 나오기 때문이다.[1] 1982년 Caltech의 Richard Feynman은 고전 투링 기계로는 지수 함수적인 시간 증가없이 양자 현상을 시늉(simulation)할 수 없음을 밝혔다[11]. 하지만 아쉽게도 정확한 모델을 제시하지는 못하였다.

양자컴퓨터의 진가를 알 수 있는 진정한 모델은 1985년 영국 옥스포드 대학의 David Deutsch가 양자 투링 기계의 이론을 제시하면서 시작되었다[12]. 이 모델에서는 읽기, 쓰기 연산 모두가 양자적 상호작용에 의해 수행되고, 테이프는 양자적인 상태를 가지고 있다[13]. 고전적 투링 기계가 0, 1만을 저장하는데 비해 양자 투링기계는 이들의 중첩된 상태를 저장할 수 있다. 그래서 고전적으로는 한 번의 결과가 나오는 과정 동안에 양자 투링기계는 여러개의 결과를 동시에 구현할 수 있다. 이를 양자 병렬성(quantum parallelism)이라고 한다[14].

3. 양자전산의 기초

3.1 큐비트(qubit)

고전적인 컴퓨터에 메모리가 필요하듯이 양자 컴퓨터에도 정보를 담을 수 있는 메모리가 필요하다. 이를 양자 비트 혹은 큐비트(quantum bit-qubit)라 부른다. 고전 컴퓨터에서는 비트를 단순히 1과 0으로 표현하는데 반해 양자 컴퓨터의 큐비트는 양자적 순수상태인 $|1\rangle$과 $|0\rangle$으로 2진수를 표현한다. 이는 물론 다음과 같이 $|1\rangle$과 $|0\rangle$이 중첩된 상태도 표현할 수 있는 복합적인 두 준위 시스템이다:

$$|\Psi\rangle = c_0|0\rangle + c_1|1\rangle. \quad (1)$$

어떠한 두 준위 시스템도 큐비트가 될 수 있다. 예를 들어 1/2 스핀 입자나 두 준위 원자의 들뜸상태와 바닥상태 같은 것도 될 수 있다. 전자처럼 1/2 스핀입자의 경우 $|0\rangle$과 $|1\rangle$은 각각 전자의 상향스핀과 하향스핀에 해당한다고 보면 되겠다.

큐비트는 무한개 상태의 중첩으로 되어 있다 하더라도 결국 측정하면 하나의 값밖에 나오지 않는다. 만약 큐비트가 식 (1)의 상태로 되어 있을 때 측정을 한다면 $|c_0|^2$의 확률로 0이 측정되고 $|c_1|^2$의 확률로 1이 측정된다. 또한 중요한 것은 측정후 큐비트의 상태는 $c_0|0\rangle+c_1|1\rangle$ 그대로가 아니라 $|0\rangle$이나 $|1\rangle$로 되어 버린다는 것이다.

[1]여기서 중첩상태란 Paul Dirac이 그의 책 *'The principles of Quantum Mechanics* (Oxford University Press, Oxford, 1958)'에서 밝히는 것 처럼, 양자 물리의 핵심에 해당하는 개념으로 양자 물리의 많은 역설들은 이에 근거한다. 양자적 중첩상태에 대해서는 다음 절에서 좀 더 자세히 다룬다.

3.2 레지스터

고전 컴퓨터처럼 양자 컴퓨터에서도 큐비트들의 조합으로 레지스터를 만들 수 있다. 예를 들어 2진수로 110인 6을 저장한다고 하면 세개 큐비트의 상태 $|1\rangle|1\rangle|0\rangle$으로 만든다(이것을 레지스터 $|6\rangle$으로 약기하기로 한다.). $a=2^{n-1}a_{n-1}+...+2_1^1+2^0a_0$일 때 레지스터 $|a\rangle$는 $|a_{n-1}\rangle|a_{n-2}\rangle...|a_1\rangle|a_0\rangle$ (여기서 a_i=0 혹은 1이다.)를 나타낸다.

두개의 레지스터 상태 $|a\rangle$와 $|b\rangle$는 서로 직교한다, 즉

$$\langle a|b\rangle = \langle a_0|b_0\rangle\langle a_1|b_{1>}\cdots\langle a_{n-1}|b_{n-1}\rangle = \delta_{a,b}. \quad (2)$$

a와 b가 다르면 각 계수들 중의 하나 이상은 무조건 다르기 때문에 전체적으로 0이다.

n-비트 레지스터에서 일반적으로 상태는 다음과 같이 표기된다

$$|\Psi\rangle = \sum_{x=0}^{2n-1} c_x|x\rangle. \quad (3)$$

여기서 이 레지스터는 여러가지 다른 값을 동시에 갖고 있다. 이러한 상황은 고전컴퓨터에서와 아주 다르다. 만약 이 레지스터가 $|\psi\rangle=1/\sqrt{2}(|0\rangle+|3\rangle)$, 즉 $1/\sqrt{2}(|0\rangle|0\rangle+|1\rangle|1\rangle)$로 초기화되어 있다고 하자. 이 레지스터를 측정하면 $|0\rangle$이나 $|3\rangle$의 값이 동등한 확률로 측정될 것이다. 다른 오류가 없다면 $|2\rangle$나 $|1\rangle$은 절대 측정되지 않는다.

3.3 게이트(gate)

고전 컴퓨터에서 정보 처리는 논리 게이트를 통해 이루어진다. 논리 게이트는 입력 비트의 상태를 진리표에 따라 다른 상태로 대치시킨다. 가장 간단한 예로서 NOT 게이트를 들 수 있는데 NOT 게이트는 0을 1로, 1을 0으로 반전시킨다. 양자컴퓨터에서의 게이트는 같은 진리표를 갖는 유니타리 연산으로 구현된다. 예를 들어 NOT에 해당되는 유니타리 연산은 다음과 같다

$$U_{NOT}|0\rangle = |1\rangle \quad U_{NOT}|1\rangle = |0\rangle. \quad (4)$$

고전 컴퓨터에는 없는 게이트를 양자 컴퓨터에서는 정의할 수 있다. 다음의 예를 보자

$$U_A|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle) \quad U_A|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle). \quad (5)$$

U_A는 고전적인 상태를 양자적으로 중첩된 상태로 만든다.[2] 고전 컴퓨터에서는 생각할 수 없는 개념이다. U_A는 유용한 역할을 한다. $|0>$으로 초기화 되어있는 n-비트 레지스터의 각 비트에 U_A를 취해보자. 그러면 결과는 다음과 같다

$$
\begin{aligned}
|\psi> &= U_A \otimes U_A \otimes \cdots \otimes U_A \,|00\cdots 0> \\
&= \frac{1}{\sqrt{2}}(|0>+|1>) \otimes 1\sqrt{2}(|0>+|1>) \\
&\quad \otimes \cdots \otimes 1\sqrt{2}(|0>+|1>) \\
&= \frac{1}{2^{n/2}}(|00\cdots 0> + |00\cdots 1> + \cdots + |11\cdots 1>) \\
&= \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x>. \qquad (6)
\end{aligned}
$$

n번의 U_A 연산을 취함으로써 2^n개의 서로 다른 수의 중첩을 얻었다. 이것은 양자 레지스터의 주목할 만한 성질 중의 하나이다. 반면에 고전 컴퓨터에서는 n번의 연산을 해도 각 연산의 특별한 한 개의 결과밖에 얻지 못한다. 이 성질이 양자 컴퓨터의 "양자적 병렬처리"를 가능케 한다. 이처럼 2^n개의 서로 다른 수로 초기화가 되어 있는 레지스터에 선형 유니타리 연산을 가해주면 우리는 결과로 2^n개의 서로 다른 값을 동시에 얻게된다. 물론 측정은 다음의 문제이다.

3.4 함수

다음의 함수를 생각하자[14]

$$f : (0, 1, \cdots, 2^{m-1}) \rightarrow (0, 1, \cdots, 2^{n-1}). \qquad (7)$$

고전 컴퓨터에서는 0, 1, ···, 2^{m-1}의 입력값을 각각 $f(0)$, $f(1)$, ···, $f(2^{m-1})$의 출력값으로 변화시킨다. 반면에 양자 컴퓨터에선 가역적인 연산을 해야하는 특성때문에 조금 다르다. 1 대 1 대응 함수인 경우만 $|x>$를 $f(|x>)$로 대응시킬 수 있다. 만약 f가 1 대 1 대응 함수가 아니라면 결과값으로 부터 입력값을 알 수 없기 때문에 비가역적이다. 즉 유니타리 연산이 아니다. f가 1 대 1 대응 함수인지의 여부에 상관없이 연산을 수행하고자 할 때에는 입력값을 기록해 두면 된다. 그래서 양자컴퓨터에서는 2 개의 레지스터를 사용한다. 하나는 입력값을 저장하고 다른 하나는 출력값을 저장한다. 입력값 x를 $|x>$로 나타내고 출력값 $y=f(x)$를 $|y>=|f(x)>$로 나타내자. 이렇게 하면 각각의 레지스터는 서로 직교한다($<x|x'>=\delta_{xx'}$, $<y|y'>=\delta_{yy'}$). 임의의 연산 U_f가 수행된다면 다음과 같다.

$$U_f\,|x>|0> = |x>|f(x)>. \qquad (8)$$

양자 컴퓨터에서의 연산은 가역적일 뿐만 아니라 또한 양자적이다. 따라서 우리는 $f(x)$를 연산할 때 입력값을 하나씩 따로 따로 계산하지 않을 수도 있다. 레지스터 하나에 가능한 모든 값을 중첩시켜놓고 U_f의 연산을 한번만 수행하면 우리는 $f(0)$, ··· $f(2^{m-1})$의 2^m개 출력값을 얻게된다.

$$
\begin{aligned}
|\psi> &= U_f\left(\frac{1}{2^{m/2}}\sum_{x=0}^{2^{m-1}}|x>\right)|0> \\
&= \frac{1}{2^{m/2}}\sum_{x=0}^{2^{m-1}}|x>|f(x)>. \qquad (9)
\end{aligned}
$$

하지만 측정하면 하나의 값만 나타나므로 $|\psi>$로부터 2^m개의 모든 값을 알아낼 수는 없다. 만약 측정값이 $|j>$였다면 두번째 레지스터는 반드시 $|f(j)>$의 값을 갖고 있는 것이므로 두 레지스터의 상태는 $|\psi>=|j>|f(j)>$인 것이다. 하지만 가령 $|\Psi>$의 주기를 측정해야 하는 문제의 경우 양자 전산은 아주 빠른 알고리즘을 제공할 수 있는데 이에 대한 자세한 서술은 4장에서 하겠다.

3.5 양자 네트워크(quantum network)

현재 사용되고 있는 고전 컴퓨터는 Boolean 논리에 기초한다. Boolean 논리를 구현하는데는 AND, OR, NOT 게이트 등이 사용되지만, 이들을 따로따로 만들어 쓰는 것 보다는 한개의 게이트를 만들어 그의 조합으로 다른 게이트들을 구현할 수 있다면 컴퓨터 칩의 제조를 훨씬 단순화 시킬 수 있다. 이렇게 그 게이트의 조합으로 다른 모든 게이트를 만들 수 있는 게이트를 보편게이트(Universal gate)라 한다. Boolean 논리에서는 NAND 게이트가 보편게이트가 된다. 우리가 쓰는 컴퓨터의 연산장치는 바로 이 NAND게이트들의 조합으로 이루어져 있다.

NAND 게이트는 비가역적이다. 보편게이트를 가역적으로 만들면 연산이 가역적이 된다. Toffoli는 고전적으로 가역적 게이트를 설계하였다[15]. Toffoli 게이트라고 불리는 이 게이트는 3개의 입력에 3개의 출력을 가진다. Toffoli 게이트 외에도 Fredkin 게이트 역시 고전적 가역 보편 게이트 중의 하나이다[16].

양자적인 보편게이트를 다루기 전에 알아 두어야 할 양자 게이트가 있는데, 줄여서 c-NOT 게이트라 부르는 controlled-

[2]물리적으로는 한 예를 들어, 다음을 생각할 수 있다. 바닥상태 $|0>$에 있는 두준위 원자 하나에 $\pi/2$ 펄스를 쬐어 주면, 이 원자는 바닥과 흥분상태 $|1>$의 중첩에 이르게 되는데 이때 $\pi/2$ 펄스가 바로 이 U_A의 역할을 하는 것이다.

표 2. c-not 게이트

입력		출력	
control	target	control	target
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

NOT 게이트가 그것이다. c-NOT 게이트는 '제어(control)'와 '표적(target)' 두 개의 입력과 두 개의 출력비트가 있다. 표 2에서 보는 것 처럼 제어 비트가 1일 때에는 표적 비트가 변하고 0일 때에는 표적 비트의 변화가 없다.

c-NOT 게이트의 형태를 살펴보면 가역적이며, 제어비트의 값을 바꾸지 않는다. 즉 원하는 제어비트의 값을 계속 유지시키며 연산과정을 수행할 수 있다. c-NOT 게이트는 양자 2 비트 게이트 중에서 가장 중요한 게이트 중의 하나이다. c-NOT 게이트는 가역적이기는 하지만 보편 게이트는 아니다[17].

고전전산은 2진수의 Boolean 연산에 근거하여 그에 해당하는 출력을 낼 수 있으면 되었다. 하지만 양자전산은 가역전산으로 유니타리 연산에 근거를 둔다. 그러므로 양자 보편 게이트는 그것으로 모든 유니타리 연산이
가능하여야 한다.

Deutsch는 이러한 보편 논리 게이트를 만들었다. 우리는 편의상 이를 D-게이트라 부른다[18]. D-게이트는 controlled - controlled 회전 게이트인데 세 개의 큐비트 입력과 세 개의 큐비트 출력으로 이루어진 것으로 처음 두개의 입력은 변화 없이 지나가며, 세 번째 입력은 처음 두 개의 입력이 모두 1일 때에만 θ 만큼 회전을 시켜주는 것이다. D-게이트는 8×8 행렬로 나타나고 3 비트 연산에 쓰일 수 있다. D-게이트는 3 비트 양자 보편 게이트의 기준이 된다.

고전 전산에서 보편 게이트가 여럿 있듯이 양자 전산에도 수많은 보편 게이트가 있다. 또 다른 양자 보편 게이트의 예는 Barenco의 게이트이다. Barenco는 2 비트 게이트로 보편 게이트를 만들 수 있음을 보였다[19]. 또한 DiVincenzo는 양자전산에 쓰이는 2 비트 게이트는 거의 대부분이 보편적이라는 것을 보였다.

Deutsch 게이트 나 Barenco 게이트 등은 보편적이기는 하지만 물리적으로 구현하는데는 어려운 점이 많이 있다. 특히 D-게이트는 3 비트 연산자 이어서 더욱 그 구현이 힘들다. 이에 Barenco 등은 물리적으로 이보다 쉽게 구현 가능한 c-NOT 게이트와 1 비트 유니타리 양자 게이트의 조합으로도 어떤 양자 계산이든지 할 수 있다는 것을 증명해 보였다[20].

4. Shor의 알고리즘

조선일보 99년 4월 29일자 신문에서는 영국의 Nature지 최근호를 인용하여 다음과 같이 양자 컴퓨터 소자 개발에 대한 기사를 다루고 있다.

"xsgt일본 NEC는 xkgt 꿈의 컴퓨터로 불리는 xkgt 양자컴퓨터의 고체회로 소자 개발에 세계 최초로 성공했다고 28일 발표했다. 이 소자를 사용한 양자 컴퓨터가 완성될 경우 현재 최고성능을 자랑하는 수퍼컴퓨터가 10조년 걸릴 200자리 암호해독을 1시간에 마칠 수 있는 능력을 지니게 된다.

양자 컴퓨터는 xkgt 마이크로 세계에서는 물리적 상태가 중첩될 수 있다는 양자역학의 원리를 응용해 계산능력을 획기적으로 증폭시킨 컴퓨터 개념. 현재의 컴퓨터가 정보처리를 하나씩 순차적으로 해나가는 방식인 데 비해 양자 컴퓨터는 초고속 병렬처리가 가능해, 예컨대 소자가 10개 있다면 2 (1214) 갈래의 계산을 할 수 있게 된다.

NEC의 고체회로 소자는 소자를 영하 272.15도(절대온도 1도) 이하로 냉각시켜 초전도상태로 만든 뒤 고속 전압펄스를 가하는 방식에 의해 개발됐다. 개발된 소자는 전자 파장을 자유자재로 조절할 수 있으며, 이런 소자를 여러개 병렬시킨 집적회로와 궁극적으로는 양자 컴퓨터 개발도 가능함을 실증했다고 NEC는 밝혔다."

양자 컴퓨터는 여러 가지 응용력을 가지고 있고 그 중에 가장 일반적으로 알려져 있는 분야는 인수분해(factoring)이다[21]. IBM Thomas J. Watson 연구소의 Peter Shor가 이 알고리즘을 발견하고 나서야 Deutsch의 양자 전산 이론이 더욱 더 각광을 받게 되었다고 해도 과언이 아니다.

다음을 생각해 보자. 617×461는 284437이 되는 계산은 쉽고 빠른 알고리즘에 의하지만, 284437이 소수 617과 461의 곱으로 이루어 졌다는 것을 알아내는 계산은 가장 느린 알고리즘 중의 하나이다. 현재까지 알려져 있는 여러 가지의 암호 체계 중에서 Vernam암호체계와[22] 같은 비밀키(secret key) 방식의 암호체계에서는 상관없지만, 전자결재와 같은 대중에게 개방된 시스템 안에서의 비밀보장을 위한 방법 중의 하나인 공개키(public key) 방식의 RSA 암호체계[3]에서는[23,24] 인수분해가 가

[3]RSA 암호체계는 R. Rivest, A. Shamir 그리고 L. Adleman에 의해 미국의 MIT에서 1978년에 창안된 공개키 방식의 암호체계이다. 본문에서 말한 바와 같이 두 소수의 곱으로 이루어 진 합성수를 인수분해 할 때, 지수 함수적인 시간이 필요하다는 것을 이용한 것이다. 처음 Rivest와 그의 동료들이 129자리의 수를 1977년에 인수분해하려고 하였을 때, 1,600여대의 컴퓨터 네트워크로 17년간의 노력이 필요하였다. 따라서 150자리 이상의 수를 이용하면 충분히 안전한 암호체계를 만들수 있다고 생각하였다.

장 중요한 비밀보장의 열쇠를 가지고 있게 된다. 공개키의 두 소수의 곱인 합성수 N값을 인수분해를 할 수만 있다면 나머지 비밀키 역시 쉽게 알 수 있기 때문이다.

N을 인수분해 하는 고전적인 방법은 기본적으로 $\sqrt{N}$ 보다 작은 수를 이용하는 것이다. 즉 1부터 $\sqrt{N}$ 까지의 숫자들로 N을 하나 하나 나누어가는 방법이다. 하지만 이러한 방법은 매우 시간이 많이 걸린다. 즉 $\sqrt{N} \approx 10^{L/2}$회의 나눗셈이 필요하며 N의 자리수인 L의 크기에 대해 지수 함수적인 소요시간 증가가 나타나게 된다. (초당 10^{10}회의 나눗셈을 한다고 가정하면 L=34일 때($10^{34}<N<10^{35}$), 일년의 시간이 걸린다고 볼 수 있다.) 이렇게 지수 함수적인 시간을 필요로 하는 알고리즘을 느린 알고리즘이라 정의하며, 지수함수를 포함하지 않는 다항 함수적인 소요시간을 필요로 하는 것을 빠른 알고리즘이라 한다. 이때 N을 인수분해 하는 빠른 알고리즘을 만들어내는 것이 불가능하다면 공개키 방식의 암호체계는 비밀보장이 가능하게 된다.

인수분해에 있어 일일이 나누는 것보다 조금 빠른 알고리즘이 존재하는데, 여기서 인수분해의 문제는 어떤 함수의 주기를 찾는 것과 관련되어 있다[25]. 특별히 합성수 N을 인수분해 할 경우, N의 인수가 아닌 임의의 a에 대해

$$f_{a,N}(x) = a^x \bmod N \tag{10}$$

을 만족하는 x의 반복을 통해 함수 $f_{a,N}(x)$의 주기 r을 찾아내는 것이다. 중요한 것은 특별한 두 가지 조건을 만족하는, 즉 짝수이고 r mod N은 -1이 아닌 r을 찾는 것이다. 그 결과 알아낸 r에 대해 $a^{r/2}\pm1$과 N의 최대공약수를 각각 구하면 그 두 가지 값이 인수가 된다. 최대 공약수를 구하는 것은 유클리드 소거법[26]에 의해 빠른 알고리즘으로 구할 수 있다.

간단한 예를 들어, N=21, a=2인 경우를 살펴보자. x에 0, 1, 2, ⋯, 를 대입하였을 때, 함수 $f_{a,N}(x)$의 값은 1, 2, 4, 8, 16, 11이 반복하여 나온다. 여기서 반복 주기는 r=6이며, 짝수이고, 또한 r mod N은 -1이 아니다. 따라서 $a^{r/2}$+1인 9와 $a^{r/2}$-1인 7 두 수와 21의 최대공약수를 각각 구하면 3, 7이 되고 우리가 구하려는 21의 두 인수가 된다.

이러한 방식의 인수분해의 모든 과정은 대부분 다항 함수적인 시간이 필요하지만 그 중에 주기를 구하는 것만은 L의 크기에 따라 지수 함수적인 시간을 필요로 하는 것이다. 하지만 양자 컴퓨터에서는 다른 방식의 접근을 가능하게 한다. Shor의 양자 알고리즘이 바로 그것이다[21]. 이것은 양자 병렬성을 이용하는 것이며 크게 세가지 단계로 나뉘어진다. 먼저 기억 레지스터를 선택하고, 그 안에 취할 수 있는 모든 가능한 정수형 태의 중첩 상태를 구현하고 양자 병렬성의 장점을 사용하여 서로 다른 연산을 한꺼번에 수행한다. 마지막으로 위에서 말한 것처럼 주기를 이용한 소인수분해를 하는 것이다. 이제 조금 더 자세히 이 과정에 대해 알아보자.

4.1 레지스터의 선택

함수 f를 대응하는 유니타리 연산자 U_f에 적용할 때 고려해야 할 점은 다음과 같다. 함수 $f_{a,N}(x)$의 예에서는 결과가 N보다 크지 못하기 때문에 출력 레지스터는 적어도 L 큐비트를 가져야 하며, 입력 레지스터를 $2L$ 큐비트를 가져야 한다.

입력과 출력 두 레지스터를 각각 $|0\rangle|0\rangle$으로 초기화 한다. 다음에 입력 레지스터의 각 큐비트에 유니타리 연산자인 U_A를 적용함으로 0부터 2^{2L}-1까지의 모든 가능한 중첩상태인 식 (6)을 만든다. 그리고 함수값 $f_{a,N}(x)$를 구하는 유니타리 연산자 $U_{f_{a,N}}$을 두 레지스터에 취하여 식 (9)를 구한다. 이제 결과 값을 측정하는 것이 남는데, 동시에 모든 값을 알아낼 수는 없고 한 번의 측정에 하나의 값만을 알 수 있다. 하지만 인수분해의 문제에서는 측정된 모든 값 하나 하나에 관심이 있는 것이 아니다. 필요한 것은 주기함수 $f_{a,N}$의 주기이므로, 어떻게 이 주기를 효과적으로 찾을 수 있는지를 알아내는 것이 중요하다.

위에서 본 바와 같은 N=21, a=2인 경우 두 레지스터의 상태를 연산자 $U_{f_{a,N}}$에 적용한 결과는 다음과 같다.

$$\frac{1}{256}(|0\rangle|1\rangle+|1\rangle|2\rangle+|2\rangle|4\rangle+|3\rangle|8\rangle+|4\rangle|16\rangle +|5\rangle|11\rangle+|6\rangle|1\rangle+|7\rangle|2\rangle+\cdots). \tag{11}$$

이제 두 번째 레지스터를 측정한다. 이것은 두 번째 레지스터의 양자상태를 붕괴(collapse)시키며 어떠한 결과 k를 얻게 된다. 이것은 a^x mod N=k인 어떤 값이 존재함을 의미하고 표 3에서와 같이 1, 2, 4, 8, 16, 11 여섯 가지 값 중의 하나가 된다.

표 3. N=21, a=2인 경우의 두 번째 레지스터에서 측정 가능한 결과들에 대한 첫번째 레지스터 상태와 오프셋

두번째 레지스터 측정 결과(k)	첫번째 레지스터 상태	오프셋 (l_k)
1	$\xi(\|0\rangle+\|6\rangle+\|12\rangle+\cdots)\|1\rangle$	0
2	$\xi(\|1\rangle+\|7\rangle+\|13\rangle+\cdots)\|2\rangle$	1
4	$\xi(\|2\rangle+\|8\rangle+\|14\rangle+\cdots)\|4\rangle$	2
8	$\xi(\|3\rangle+\|9\rangle+\|15\rangle+\cdots)\|8\rangle$	3
11	$\xi(\|5\rangle+\|11\rangle+\|17\rangle+\cdots)\|11\rangle$	5
16	$\xi(\|4\rangle+\|10\rangle+\|16\rangle+\cdots)\|16\rangle$	4

두 번째 레지스터의 상태가 $|k>$ 일때, 첫 번째 레지스터는 다음과 같이 일반적으로 표현할 수 있다.

$$|\psi> = N \sum_{j=0}^{2^{2L}/r} |jr + l_k>. \quad (12)$$

여기에서 N은 규격화 상수이며, r은 주기, l_k는 오프셋(offset)이다. 그 중에서 필요한 것은 r이지만 r을 효과적으로 구하는 데는 오프셋 l_k의 값이 두번째 레지스터 측정결과에 의존한다는 큰 문제가 있다. 그러나 이 문제는 고전적인 빠른 푸리에 변환의 양자적 등가인 띄엄띄엄 푸리에 변환을[27,28] 사용하여 없앨 수 있다.

5. 맺음말

불확정성의 원리, 진공에너지의 존재, 슈뢰딩거의 역설, EPR역설, 터널링 효과 등을 통하여 물리학자들에게 친숙한 양자 물리는 이제 이 논문에서 소개된 양자전산 뿐 아니라 양자암호, 양자전송 등 양자정보라는 새로운 분야를 형성하며 그 응용성을 시사하고 있다. 19세기 물리를 중심으로 20세기의 전자산업이 발전하였다면 이제 21세기에는 20세기 물리인 양자물리를 통해 산업의 발전이 있을 것으로 예상된다. 21세기, 반도체 산업의 10배 이상의 시장이 된다는 정보산업 분야에서 양자정보의 역할을 기대한다.

감사의 글

이 글은 서강대학교의 이 진형, 장 진각 그리고 광주과기원의 송 창인의 무한한 도움을 받은 결과이다. 또한 교육부 기초과학 연구소 프로그램(015-D00118)의 지원을 감사한다.

참고문헌

1. D. A. Patterson, *Scientific Am.* **September**, 48 (1995).
2. R. W. Kyese, *IBM J. Res. Developm.* **32**, 24 (1988).
3. C. H. Bennett, *IBM J. Res. Developm.* **32**, 16 (1988).
4. A. Peres, *Phys. Rev. A.* **32**, 3266 (1985).
5. C. Bennett, *Scientific Am.* **November**, 88 (1987).
6. R. Landauer, *Nature,* **335**, 779 (1988).
7. C. P. Williams, S. H. Clearwater, *"Explorations in Quantum Computing"* (Springer-Verlag, New York, 1998).
8. J. G. Brookshear, *"Theory of Computation : Formal Languages, Automata, and Complexity"* (Addison-Wesley, England, 1989).
9. C. Bennett, *IBM J. Res. Developm.* **17**, 525 (1973).
10. P. A. Benioff, *J. Statistical Phys.* **22**, 563 (1980); *Int. J. theor. Phys.* **21**, 177 (1982); *Phys. Rev. Lett.* **48**, 1581 (1982).
11. R. P. Feynman, *Found. Phys.* **16**, 507 (1986); *"Lectures on Computation"* (Addison-Wesley, Reading, 1996)
12. D. Deutsch, *Proc. R. Soc. Lond. A.* **400**, 97 (1985).
13. 여기서 양자 상태라 하면, 양자적 순수상태들의 중첩을 의미한다.
14. A. Barenco, *Contemp. Phys.* **37**, 375 (1996).
15. T. Toffoli, *Math. Systems Theory* **14**, 13 (1981).
16. E. Fredkin and T. Toffoli, *Int. J. theor. Phys.* **21**, 219 (1982).
17. D. Deutsch, A. Barenco, A. Ekert, *Proc. R. Soc. Lond. A.* **449**, 669 (1995).
18. D. Deutsch, *Proc. R. Soc. Lond. A.* **425**, 73 (1989).
19. A. Barenco, *Proc. R. Soc. Lond. A.* **449**, 679 (1995).
20. A. Barenco et al., *Phys. Rev. A.* **52**, 3457 (1995).
21. P. Shor, Proceedings 35th Annual Symposium on Foundations of Computer Science. (IEEE Computer Society Press, Los Alamitos, 1994) p.124
22. C. H. Bennett, G. Brassard and A. Ekert, *Scientific Am.* **October**, 50 (1992).
23. R. Rivest, A. Shamir and L. Adelman, *Communications of the ACM,* **21**, 120 (1978); MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212 (*Janury* 1979).
24. N. Koblitz, *"A course in number theory and cryptography"* (Springer-Verlag, New York, 1987).
25. G. L. Miller, *J. Computer Science* **13**, 300 (1976).
26. H. E. Rose, *"A Course In Number Theory"* (Oxford, Clarendon, 1994).
27. D. Coppetsmith, *"An approximate Fourier transformation useful in quantum factoring"* IBM Research Report No.RC19642 (1994).
28. J. G. Jang, C. I. Song and M. S. Kim, New Physics (submitted).

저자약력

성명: 김명식
근무처: 서강대학교 물리학과 부교수
최종학력: 1988년, 영국 임페리알 칼리지 물리학 박사
주요경력: 영국 블라켓 연구소 연구원 (1988-1991)
슬로바키아 과학원 방문 연구원 (1992-1995)
독일 막스-플랑크 연구소 홈볼트 펠로우 (1997-1998)
이태리 국제이론물리 연구소 연구원 (1995-현재)
E-mail : mshkim@ccs.sogang.ac.kr