学校代码 10487　　学号 012003000611

分 类 号　　密级

華中科技大學

本 科 学 位 论 文

量子计算中若干前沿课题的研究与探讨

学位申请人： 龚哲轩

学 科 专 业： 计算机科学与技术

指 导 教 师： 肖亮

副指导教师： 曹强

答 辩 日 期： 2006 年 9 月 17 日

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Engineering

Research and exploration of several frontier topics in quantum computation

Candidate : Gong Zhe-Xuan

Major : Computer Science

Supervisor : Xiao Liang

Co-Supervisor : Cao Qiang

Huazhong University of Science & Technology

Wuhan 430074, P. R. China

October, 2006

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，在 ____ 年解密后适用本授权书。

　　不保密 □。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘 要

几十年来，电子计算机的性能一直按照“摩尔定律”迅速发展，而集成电路的尺寸也不断缩小。但减小到原子尺度时，占据主导地位的量子效应使人们不再可能制造传统计算机。为了克服经典计算机将遭遇的“物理极限”,人们试图按照量子力学的原理在原子尺度上制造一种全新的量子计算机，并努力使其性能超越传统计算机。

本文首先介绍了量子计算机的发展历程，讲述了量子力学的基础知识与量子计算机工作的基本原理。接着详细叙述目前存在的一些重要量子算法，并给出了非常有用的Shor质因数分解算法的经典计算机程序模拟（基于Mathematica 5.2）。

此后，文章对量子计算领域中的若干前沿课题作了研究与探讨，以通俗的语言介绍了无相互作用测量，量子芝诺效应，量子反事实运算的理论定义与实验装置；明确指出了目前量子反事实运算定义中存在的问题并做了改进，然后利用量子程序设计语言对其进行了精确表述；提出了一种全新的基于量子逆算法的有效降错方案；分析与评述了当前的波粒二象性计算机模型。

关键词：量子计算机，量子算法，量子反事实运算，量子降错。

Abstract

During the past decades, the performance of electronic computers has undergone a rapid improvement in accordance with the ”Moore's Law”. Meanwhile, the size of integrated circuit is decreasing continuously. In atomic size, however, the dominant quantum effects prevent people from manufacturing traditional computers any longer. It is therefore natural that in attempt to overcome the physical limit of classical computer, people are trying to invent a brand new kind of computer called quantum computer, which is hoped to be superior than its traditional counterpart.

This thesis first introduces the history of quantum computer, the elementary knowledge of quantum mechanics as well as the fundamental principles of quantum computation. Then a detailed illustration of several important quantum algorithms are given, together with a classical simulation of the useful Shor's factoring algorithm via Mathematica 5.2.

After that, research work of several frontier topics in the field of quantum computation is presented. This involves brief introduction of interaction-free measurement, quantum zeno effect and counterfactual quantum computation (both theoretical definition and experimental set-up); Confliction in current definition of counterfactual quantum computation is pointed out and then solved well. Exact definition is also expressed using quantum programming language; A new effective error-suppression scheme for reversible quantum algorithms is invented; and the recently proposed model of duality computer is analyzed and commented.

Key words: Quantum computer, Quantum algorithm, Counterfactual quantum computation, Quantum error-suppression.

目 录

插图索引

1 量子计算机概述

Knowledge is limited; imagination encircles the world.

—— *Albert Einstein*

1.1 经典计算机的极限

我们目前所使用的计算机，代表了近年来技术进步的顶点，而这个技术进步萌芽于Charles Babbage（1791-1871）的早期思想，并且以德国工程师Konrad Zuse于1941年创造出第一台计算机为开端。现代电子计算机诞生于1946年，半个世纪以来，由于科技的不断进步，计算机的性能有了突飞猛进的发展。仅就计算速度而言，第一台电子管计算机ENIAC的速度只有5000B/s，而现代高性能计算机的运算速度已经高达万亿次（大约10^{12}B/s），半个世纪竟提高了10亿（10^9）倍！从这个角度看，飞速提高的计算机性能似乎足以满足社会进步与经济发展不断提出的需求。但事实上，计算机性能的改进却始终赶不上人类不断增长的信息处理的需求。当今人类社会进步与经济发展向计算机提出了几乎是无止境的需求，而且随着计算机的出现，以及因特网的普及更加大了这个需求。计算机性能的提高远远满足不了人类社会信息量的增长所提出的需求，目前计算机处理海量数据的能力非常薄弱。因此研制更高性能的计算机始终是信息领域的重要课题。

半个世纪中，尽管计算机的性能起了巨大的变化，但这个原理却没有改变，机器性能的提高主要靠的是缩小元器件（物理载体）的尺寸。大家知道，由于半导体工艺与技术的不断进步，几十年来，集成电路的尺寸一直按照所谓的”摩尔定律”（Moore Law），以每18个月缩小一倍的速度持续发展，从而使计算机的性能指标（如计算速度、存储密度等）取得了每18个月翻一番的巨大成就。目前集成电路的特征尺寸已经降到了几十个纳米的量级，这个趋势还在继续着，预计再经过20年左右，将要降到几个原子的大小，甚至更小。于是出现了一个新的问题，在原子的尺度上传统的物理定律不再适用，遵循的是全新的量子力学规律，我们不再可能制造出传统的计算机。也就是说，大约20年后传统计算机将达到它的”物理极限”。人们自然提出如下问题；能不能按照量子力学的原理在原子的尺度上制造一种全新的计算机器（量子计算机）？这种机器的性能是否能够超越传统的计算机？

1.2 量子计算机的定义

事实上，现在放在我们面前的高速现代化的计算机和它庞大的重达30吨的祖先

并没有什么本质的区别，而那台庞大的机器是由18000个真空管和500米的电线构成的！尽管计算机已经变的更加小巧而且一般来说在执行任务时已经快的多，但是计算机的任务却并没有改变：都是把二进制位（0和1）作为基本的信息单元进行编码处理，然后通过操纵这些二进制位穿过连续排列的布尔逻辑门产生计算结果。每个二进制位是通过一个经典的物理系统实现的。例如晶体管电路，晶体管截止（输出低电平）代表了二进制数0，而晶体管导通（输出高电平）代表了二进制数1。

量子计算设备的设想首先在20世纪80年代，由物理学家和计算机科学家，IBM Thomas J Watson研究中心的Charles H. Bennett，伊利诺伊州Argonne国家实验室的Paul A. Benioff，牛津大学的David Deutsch和加利福尼亚理工学院（Caltech）的Richard P. Feynman提出。

传统计算机和量子计算机之间的关键区别在于：在量子计算机中，基本信息单元叫做一个量子位或者qubit (quantum bit)，它通常由光子，电子或某些原子来承载。不同于传统二进制位的地方在于，qubit不仅能在相应于传统计算机位的逻辑状态0和1稳定存在，而且也能在相应于这些传统位的叠加状态存在。换句话说，qubit能作为单个的0或1存在，也可以同时既作为0也作为1（即所谓叠加态：$\alpha\mathbf{0}+\beta\mathbf{1}$），而且用组合系数$\alpha,\beta$代表了每种状态的可能性。这种现象看起来和人的直觉不符，因为在人类的日常生活中发生的现象遵循的是传统物理规律，而不是统治原子级世界的量子力学规律。qubit具有这种性质的直接原因是因为它遵循了量子力学的规律，而量子力学从本质上说完全不同于传统物理学。量子力学的基本原理将会在下节中简要介绍。

与经典计算机非常类似：量子计算机操纵着量子位(qubit)穿过一系列的量子门，每个门的转化对一个或两个qubit起作用。在连续应用这些门的时候，量子计算机能对一系列qubit应用复杂的转化（相当于做了很多的数学运算）。然后，qubit在输出端被测量，而测量结果就作为最终的计算结果。

定义 1.1： 量子计算机是一类遵循量子力学规律进行数学和逻辑运算、存储及处理信息的物理装置。当某个装置处理和计算的是量子信息，运行的是量子算法时，它就是量子计算机

1.3 量子力学的基本原理

量子力学是20世纪初发展起来的一门研究微观客体（如电子，光子，原子等）物理规律的学科。它的**基本观念是微观客体具有波粒二象性**，即：

1. 实物粒子（电子，原子等等）具有波的特性，能发生干涉、衍射和叠加。
2. 各种波（电磁波，机械波等等）具有粒子的特性，其能量可以一份一份地不连续分布。

从这一基本观念出发，人们对微观世界作了下面几条基本假设，通常认为它们的正确性已为大量实验所证明，而量子力学正是建立在它们基础之上的公理化体系[1]：

- **状态公设**：微观粒子的状态（量子态）可以完全地用希尔伯特空间（复线性空间）的一个单位长度的矢量$|\psi\rangle$ 描述($\langle\psi|\psi\rangle = 1$)，这个矢量称为态矢量。
- **算符公设**：所有力学量是希尔伯特空间的线性厄米算符，基本力学量坐标x与动量p满足对易关系$[x, p] = i\hbar$。
- **测量公设**：在某一量子态$|\psi\rangle$上对一个力学量算符$\hat{F}$进行测量，所得到的结果只能是该力学量的本征值之一f_n。($\hat{F}|\psi_n\rangle = f_n|\psi_n\rangle$) 测到本征值$f_n$的几率就是该量子态按照该力学量完备本征态$|\psi_n\rangle$展开时，该本征态前面的系数的模方，即

$$若|\psi\rangle = \sum_n c_n|\psi_n\rangle, 则p_n = |c_n|^2 \qquad (式\ 1.1)$$

- **演化公设**：在对量子力学体系没有进行测量的情况下，量子态的时间演化遵守薛定谔方程：

$$i\hbar\frac{\partial|\psi\rangle}{\partial t} = \hat{H}|\psi\rangle \qquad (式\ 1.2)$$

其中$\hat{H}$是体系的哈密顿量算符，它的本征值代表体系的能量。

由以上基本观念与基本假设出发，我们还可以得到一个非常重要的原理－态叠加原理：

- **态叠加原理**：如果$|\psi_1\rangle$和$|\psi_2\rangle$是微观体系可能处于的状态，那么$\alpha|\psi_1\rangle + \beta|\psi_2\rangle$也是体系可能处于的状态。其中$\alpha, \beta$是任意复数，但因态矢量为单位长度，要求

$$|\alpha|^2 + |\beta|^2 = 1 \qquad (式\ 1.3)$$

1.4 一个典型的量子位

我们以光为例，利用上述的量子力学基本知识来解释量子计算机处理信息的基本单位－量子位(qubit)的存在与特性：

经典物理认为光是电磁波，但由于光波具有粒子性（基本观念），存在像粒子一样的单个光子，我们用一个态矢量$|\psi\rangle$来描写(状态公设)：

光子通常存在两个垂直的极化方向：水平极化与垂直极化。这标志着光子可以存在于两个不同的，正交的状态。我们用$|0\rangle$和$|1\rangle$来表示，它们是极化矢量这个力学量的本征态。根据态叠加原理：

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad (式\ 1.4)$$

例如取$\alpha = \beta = 1/\sqrt{2}$,则光子可以处在$|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$的状态，它对应在45°倾斜极化的光子。

事实上，这个单光子的极化方向就表示了一个量子位，它有两个正交的本征态$|0\rangle$和$|1\rangle$，我们称为基(basis)，对应着经典位的0,1两个态。一个量子位可以处在两个基的任意线性叠加中，也就是二维空间上任一方向。

下面的著名装置[2]告诉我们

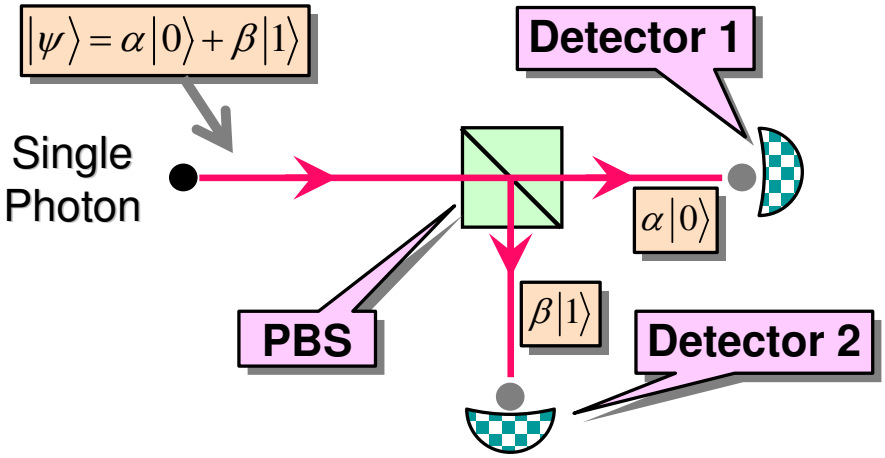


图 1-1 以单光子为例的量子位信息测量装置

光子通过极化分束器（Polarization Beam Splitter，可以将水平极化的光透射而让垂直极化的光反射)后，一部分向前行进，一部分向下传播。由于单光子是不可分割的最小整体,通常认为光子表现出了它波的特性而同时走两条路径。但是，一旦到达探测器Detector 1与Detector 2时，状态$|\psi\rangle$被测量。根据测量公设，Detector 1将有$|\alpha|^2$的几率探测到光子，Detector 2将有$|\beta|^2$的几率探测到光子，此时根据测量结果，我们可以知道光子实际走的路径。

遗憾的是，尽管一个量子位可以储存无穷多可能的状态，表达无穷多的信息，我们看到，最终的测量结果仍然的一个经典比特，只是得到0或1的概率不同。在第二章讲述量子算法时我将提到：我们总是希望最终测量结果是确定的经典比特值，量子位仅在计算过程中处于叠加态，最终输出大多接近基矢量(basis)。

1.5 量子计算机的威力

1.5.1 量子并行 (Quantum Parallelism)

量子比特叠加态的存在意味着我们可以对很多正交状态用一个逻辑门并行地操作，注意这种并行性不同于经典计算（需要多个逻辑门同时运算）。例如，我们需要进行操作

$$|x\rangle \xrightarrow{\hat{U}} (-1)^x|x\rangle \qquad \text{(式 1.5)}$$

如果系统的初态是N个基的叠加$|\psi\rangle = \frac{1}{\sqrt{N}}(|0\rangle + |1\rangle + \ldots + |N-1\rangle)$,经典计算机需要计算每个分量再相加。而量子计算机只需作一次操作

$$|\psi\rangle \xrightarrow{\hat{U}} \frac{1}{\sqrt{N}}(|0\rangle - |1\rangle + \ldots + (-1)^{N-1}|N-1\rangle) \quad \text{(式 1.6)}$$

表 1.1 量子并行的惊人威力

量子位的数目	可一次同时进行运算的正交状态数（并行度）
1	2 (2^1)
2	4 (2^2)
3	8 (2^3)
...	...
10	1048 (2^{10})
20	1048576 (2^{20})
30	1073741824 (2^{30})
...	...
300	2^{300} (大于宇宙中的总原子数)

1.5.2 量子纠缠 (Quantum Entanglement)

这是量子计算中使用的另一个量子物理学特征。当两个或多个粒子互相影响时，不可能独立描述任何一个量子的状态。即使当它们随后即被分开很远的距离，它们的行为表现的好像它们仍然是一个整体。因此我们称这些粒子是纠缠的。

例如两个粒子处在状态$|\psi_1\psi_2\rangle = |00\rangle + |11\rangle$上，我们无法独立描写粒子1或粒子2。更有趣的是，如果我们测量粒子1得到状态$|0\rangle$，那么粒子2的状态也迅速变为$|0\rangle$（测量公设）；如果我们粒子1得到状态$|1\rangle$，那么粒子2的状态也变为$|1\rangle$。

需要强调的是，粒子2状态的改变是瞬间的，无论两个粒子在地理上相距多么远，于是我们称两个粒子间的关联是**超空间**的，它是量子通信学的基础。同时我们将看到，在量子计算中，量子纠缠也被广泛应用于各种算法，产生经典算法无法达到的优越性。

1.5.3 可逆性 (Reversibility)

量子计算机的概念源于对可逆计算机的研究，而研究可逆计算机是为了克服计算机中的能耗问题。早在六七十年代，人们就发现，能耗会导致计算机芯片的发热，影响芯片的集成度，从而限制了计算机的运行速度。Landauer[3]最早考虑了这个问题，他考察了能耗的来源，指出：能耗产生于计算过程中的不可逆操作。例如，对两比待的异或操作，因为只有一比特的输出，这一过程损失了一个自由度，因此是不可

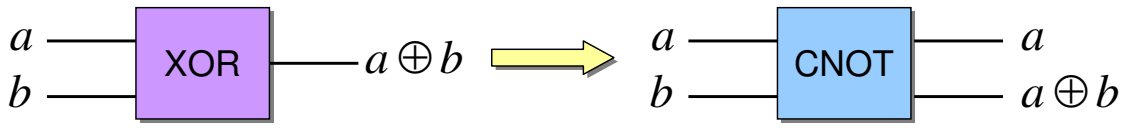


图 1-2 经典异或门到量子控制非门

逆的，按照热力学，必然会产生一定的热量。但这种不可逆性是不是不可避免的呢？事实上，只要对异或门的操作简单改进，即保留一个无用的比特，该操作就变为可逆的。因此物理原理并没有限制能耗的下限，消除能耗的关键是将不可逆操作改造为可逆操作,如图1-2。

量子力学中，根据演化公设（式1.2），在哈密顿算符不显含时间时，量子态总是作么正(Unitary)演化：

$$|\psi(t)\rangle = e^{i\hat{H}/t}|\psi 0\rangle = \hat{U}(t)|\psi(0)\rangle \quad \text{(式 1.7)}$$

么正演化U有两大特点：

1. 存在逆过程U^{-1}
2. 不改变态矢量的模长，即保持概率守恒：$UU^{\dagger} = U^{\dagger}U = 1$

所以说，量子计算的计算过程从原理上讲就是可逆的，而尽管经典计算机也可以修改电路得到对应的可逆计算机，但需要额外的开销与冗余。

2 量子算法基础

...algorithms are concepts that have existence apart from any programming language
—— *Donald Knuth*

2.1 量子图灵机

现代计算机算法的理论模型最早是由Turing与Church在1936年的两篇极为重要的论文中提出的[4,5]。经典计算机实际上就是一个**通用图灵机(Universal Turing Machine)**，它是计算机的抽象数学模型，由两部分构成：

1. 具有无限多个存储单元的记录带，每个存储单元内容的变化是有限的，通常用二进制的**"0"**和**"1"**来表示；
2. 一个具有有限内态的读写头，每步操作中读写头可以在记录带上左移或右移一格或不动。图灵机在操作中，读写头根据其内态和当前存储单元的内容，按既定的规则，改变其内态和存储单元的内容。并决定下一步读写头的移动方向。

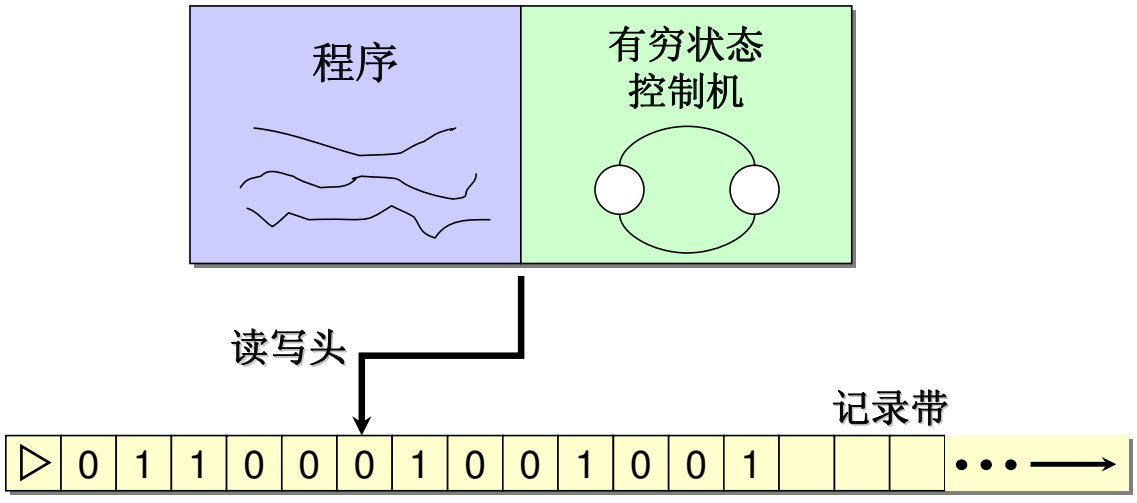


图 2-1 通用图灵机示意图

从这一组成看，经典图灵机有两个特点：

- 模型是不可逆的。例如，对于图灵机操作"写存储单元$\Longrightarrow$左移一格"，其逆就变成了"左移一格$\Longrightarrow$写存储单元"，该逆操作不再是一个有效的图灵机操作

- 操作是完全确定性的，用q代表当前读写头的状态，s代表当前存储单元内容，d取值为L, R, N,分别代表读写头左移、右移或不动，则在确定性算法中，当q, s给定时，下一步的状态q'，s'及读写头的运动d完全确定。

量子计算机也是建立在量子图灵机基础上。它的不同点在于：

- **对图灵机的操作是可逆的。除了对计算结果的测量外，体系做么正演化**
- **操作是非确定的。q, s, q', s'相应地变成了量子态，当q, s给定时，图灵机以一定的概率$|\psi(q, s, q, s', d)|^2$变换到状态q', s'，并且读写头实行运动d。概率幅函数$\psi(q, s, q, s', d)$ 为模长在$[0, 1]$间的复数，它完全决定了量子图灵机的性质（就是微观体系的状态波函数）。对叠加态操作时运算结果按概率幅叠加。**

2.2 量子逻辑门

量子图灵机仍然是一个抽象的数学模型，如何在物理上构造出量子计算机呢？理论上已证明[6]，量子图灵机可以等价为一个量子逻辑电路，因此可以通过一些量子逻辑门的组合来构成量子计算机。量子逻辑门按其输入比特的个数可分为单比特、二比特、及三比特逻辑门等。因为量子逻辑门是可逆的，所以其输入和输出比特数相等。量子逻辑门对输入比特进行一个确定的幺正变换，得到输出比特。Deutsch[7]最早考虑了用量子逻辑门来为造计算机的问题，他发现，几乎所有的三比特量子逻辑门都是通用逻辑门。通用逻辑门的含义是指，通过该逻辑门的级联，可以以任意精度逼近任何一个么正操作。后来不少人发展了Deutsch的结果，最后Deutsch[8]和Lloyd[9]各自独立地证明[11]，几乎所有的二比特量子逻辑门都是通用的，这里"几乎"是指，二比特通用量子逻辑门的集合是所有二比特逻辑门的集合的一个稠密子集。

实验上通常用一些具体的量子逻辑门来构造计算机。Barenco等人[10]证明，一个二比特的异或门（控制非门Controlled NOT)和对一比特进行任意操作的门可构成一个通用量子门集。相对来说，单比特逻辑门在实验上比较容易实现，现在的不少实验方案都集中于制造量子异或门。已有的用来实现量子异或门的方案包括：利用原子和光腔的相互作用(Cavity Quantum Electrodynamics)[11]；利用冷阱束缚离子[12]；以及利用电子或核自旋共振[13]。

图2-2给出了几种常用的量子逻辑门，原理上它们足以任何量子电路。其中Pauli-X门对应与经典的非(NOT)门。Controlled系列门的第一个量子位是控制位，运算过程中值不变，仅当其为1时才对第二个量子位进行操作。

Controlled-NOT是非常重要的二比特门，它可以将第一个量子位中的经典信息复制到第二个量子位(初值为0)中：

$$|x\rangle_1|0\rangle_2 \xrightarrow{CNOT} |x\rangle_1|x\rangle_2 \quad \text{其中} x = 0, 1 \qquad \text{(式 2.1)}$$

然而，当第一个量子位中是叠加态$\alpha|0\rangle_1 + \beta|1\rangle_1$时，能否将其内容复制到第二个

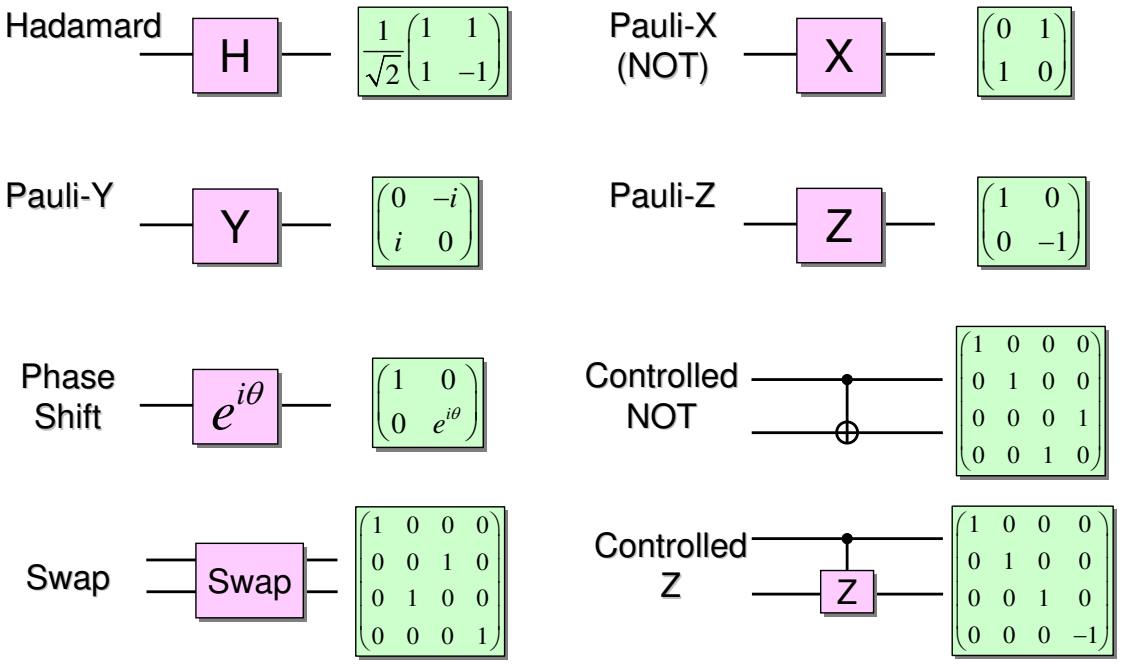


图 2-2 常用量子逻辑门的电路符号与运算矩阵

量子位中呢？**答案是否定的!**

$$\alpha|0\rangle_1+\beta|1\rangle_1|0\rangle_2 \xrightarrow{CNOT} \alpha|0\rangle_1|0\rangle_2+\beta|1\rangle_1|1\rangle_2 \quad \text{是纠缠态!} \tag{式 2.2}$$

事实上，从量子力学的基本原理出发，容易证明下面的著名定理：

定理 2.1：（量子态的不可克隆定理）不存在任何量子操作，可以将一个任意的量子位信息复制到另一个量子位中而不破坏原先的信息。

2.3 Deutsch-Jozsa 算法

下面介绍一个简单的量子算法，也是量子计算领域早期非常著名的算法[14]

- **算法：Deutch-Jozsa 问题**
- **输入：**给定一个黑盒了，它能作以下操作

$$|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y\oplus f(x)\rangle \quad \text{其中} x\in\{0,\ldots,2^n-1\}, f(x)\in\{0,1\} \tag{式 2.3}$$

  此外假定f(x)要么是**常值函数**（即对所有x都输出0或都输出1），要么是**平衡函数**（对精确一半的x取值输出0,对另一半输出1）
- **输出：**输出n位全0表示f(x)是常值函数，其余输出表示f(x)是平衡函数
- **运行时间：**一次U_f，总是成功

- **步骤:**

第一步　$|0\rangle^{\otimes n}|1\rangle$　初态$|\psi_0\rangle$　(式 2.4)

第二步　$\xrightarrow{H^{\otimes n}H} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$　制备叠加态$|\psi_1\rangle$　(式 2.5)

第三步　$\xrightarrow{U_f} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$　计算f(x)得$|\psi_2\rangle$　(式 2.6)

第四步　$\xrightarrow{H^{\otimes n}} \frac{1}{2^n}\sum_{z=0}^{2^n-1}\sum_{x=0}^{2^n-1}(-1)^{x\cdot z+f(x)}|z\rangle[\frac{|0\rangle-|1\rangle}{\sqrt{2}}]$　得$|\psi_3\rangle$　(式 2.7)

第五步　$\xrightarrow{Measurement} z$　测量$|\psi_3\rangle$前n位得到结果　(式 2.8)

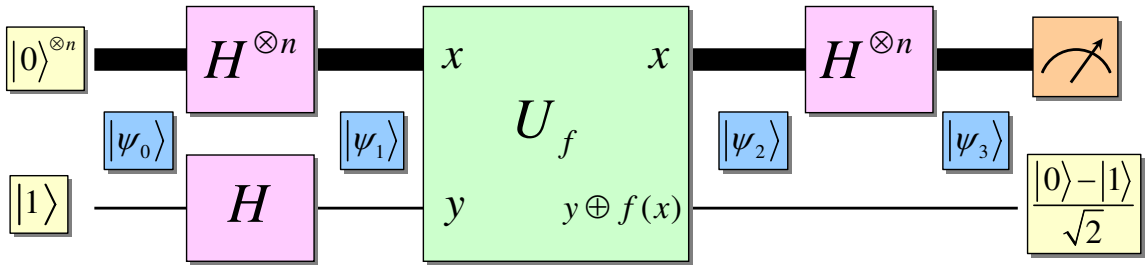


图 2-3 **Deutsch-Jozsa 算法的量子逻辑电路图**

Deutsch-Jozsa 算法的关键在于利用了量子并行性，即第三步中一次同时计算了2^n个不同x函数值$f(x)$，在第四步后，如果f(x)是常值函数，那么$|\psi_3\rangle$求和表达式中2^n个$|z\rangle=|0\rangle$的项前系数$(-1)^{x\cdot z+f(x)}$均相同，从而$|\psi_3\rangle=\pm|0\rangle^{\otimes n}\frac{|0\rangle-|1\rangle}{\sqrt{2}}$；如果f(x)是平衡函数，则$2^n$个$|z\rangle=|0\rangle$的项一半系数为1,一半为-1，刚好完全抵消，所以测量前n位必然得到的是全0外的其他结果。

容易推算，经典算法求解Deutsch-Jozsa问题至少需要运行$(2^{n-1}+1)$次黑盒子，而量子算法只须一次。可见对于此问题，量子算法的运算速度明显高于经典算法。

3 量子搜索算法

Quantum Mechanics Helps in Searching for a Needle in a Haystack

—— *Lev Grover*

3.1 引言

现实生活中许多计算问题都可归结为搜索问题，如最短路径问题、排序问题、图着色问题、数据库搜索问题及密码中的穷举攻击问题等均属于这类问题。对于N个元素的有序数据库的搜索问题，目前最快的经典算法可以达到$O(log_2N)$[15]。而对于无序数据库的搜索，只能达到$O(N)$，如果在N元数据库中搜索单个标记元素，在经典概率图灵机上平均需要$N/2$次查询操作。那么在量子计算机搜索同样的数据库，能否做得更快呢？

美国Bell实验室的Lev Grover 于1996年在美国ACM(Association for Computing Machinery)年会上提出[16]：量子计算机利用量子并行性可以在$O(\sqrt{N})$的时间复杂度内完成单标记态的无序数据库搜索。这是速度上的提升无疑是非常实用的。举个例子，原先在经典计算机上需要耗费1年时间$(3 \times 10^7 s)$做的搜索运算，在量子计算机上只需1个半小时就能完成!

3.2 Grover 算法

假设搜索空间有N个元素，那么可以用$n = log_2N$个经典或量子比特位来编号这些元素。这样只需在编号$0, 1, \ldots, N-1$中寻找一个事先被标记的元素编号。

为了判断所查询的元素是不是要找的元素，通常需要一个判断函数，它是一个黑盒子。我们将当前元素编号输入这个黑盒子，如果该元素是我们要找的元素，则黑盒子输出1, 否则输出0。至于黑盒子内部的判断过程，我们不知道，所以我们称其为先知(Oracle)，它的函数表达式如下：(ω为要找的元素编号)

$$\begin{cases} f(x) = 1, & x = \omega \\ f(x) = 0, & x \neq \omega \end{cases} \tag{式 3.1}$$

在量子搜索算法，我们需要让Oracle对一个叠加态$|\psi\rangle = (|0\rangle + |1\rangle + \ldots + |N\rangle$进行运算。为此，我们采用与Deutsch-Jozsa算法中(2.5-2.6)式中的方法，把f(x)的值存入

各项$|x\rangle$的相位中，具体作法引入一个额外的量子位$|q\rangle$并定义操作U_O

$$|x\rangle|q\rangle \xrightarrow{U_O} |x\rangle|q \oplus f(x)\rangle \qquad \text{置}|q\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}\text{，则} \tag{式 3.2}$$

$$|x\rangle(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \xrightarrow{U_O} (-1)^{f(x)}|x\rangle(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) \tag{式 3.3}$$

$$\because \quad |q\rangle\text{前后不变，可以略去不写} \quad |x\rangle \xrightarrow{U_O} (-1)^{f(x)}|x\rangle \tag{式 3.4}$$

利用上面的Oracle操作，可以定义Grover提出的量子搜索算法如下：

- **算法：Grover量子搜索**
- **输入：** Oracle操作U_O(式3.4)及量子位初态$|0\rangle^{\otimes n}$
- **输出：** 要找的元素编号ω
- **运行时间：** $O(\sqrt{2^n})$次迭代，成功概率$O(1)$
- **步骤:**

第一步 $|0\rangle^{\otimes n}$ 初态$|\psi_0\rangle$ (式 3.5)

第二步 $\xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$ 制备均匀叠加态$|\psi_1\rangle$ (式 3.6)

第三步 $\xrightarrow{U_O} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle$ 查询Oracle得$|\psi_2\rangle$ (式 3.7)

第四步 $\xrightarrow{U_{\psi_1}} (2|\psi\rangle\langle\psi| - I)\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle$ 绕$|\psi_1\rangle$翻转得$|\psi_3\rangle$ (式 3.8)

第五步 重复第二和第三步$R \approx \frac{\pi\sqrt{2^n}}{4}$次，得$|\psi_R\rangle \approx |\omega\rangle$ (式 3.9)

第六步 测量$|\psi_R\rangle$，得ω (式 3.10)

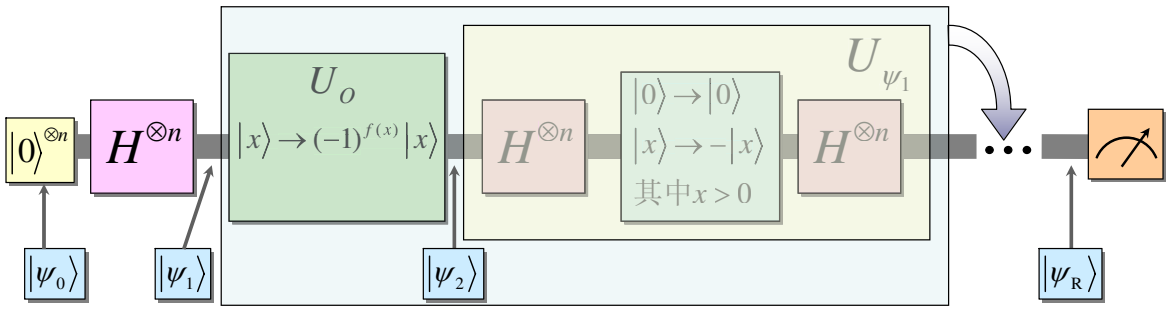


图 3-1 **Grover算法的量子逻辑电路图**

注意U_{ψ_1}操作在逻辑电路中是由三个操作相继完成的。Grover算法实际上是将初态(N个不同元素对应态的均匀叠加) 中目标元素对应的态所占的比例不断增加，直至

趋于100%，这可以由下面的示意图从看出：[17]

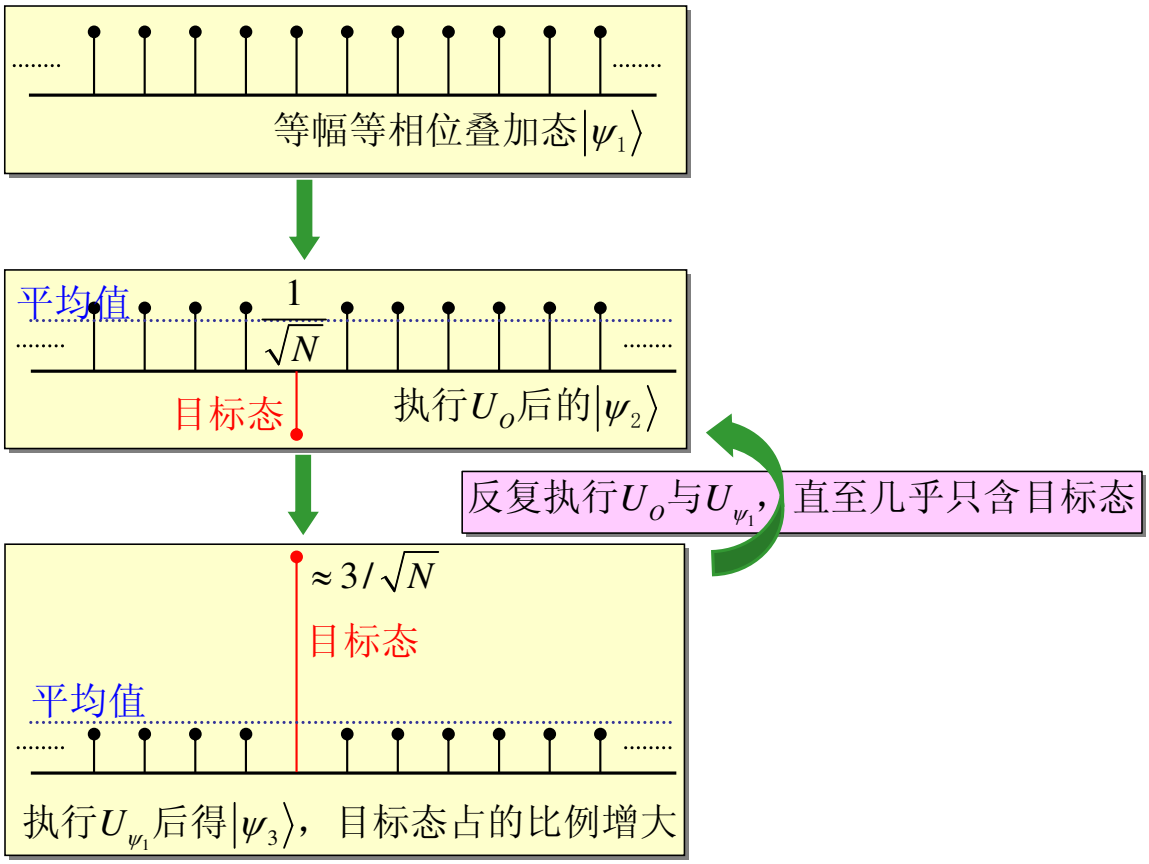


图 3-2 **Grover算法实现原理图解**

另外，也可以把$|\psi\rangle$中目标态与非目标态看成两部分，于是Grover算法也可以形象地看成二维态矢量空间中的旋转操作。每次迭代将$|\psi\rangle$向目标态所在方向旋转2θ角，由于

$$|\psi_1\rangle = \frac{1}{\sqrt{N}}|\text{目标态}\rangle| + \frac{N-1}{\sqrt{N}}|\text{非目标态}\rangle| \qquad \text{(式 3.11)}$$

$$\therefore \qquad \sin\theta = \frac{1}{\sqrt{N}} \qquad \text{(式 3.12)}$$

下面计算迭代次数R。从图3-3可知：（以下近似假设N很大）

$$(2R-1)\theta = \frac{\pi}{2} \qquad \text{(式 3.13)}$$

$$\frac{1}{\sqrt{N}} = \sin\theta \approx \theta \approx \frac{\pi}{2(2R-1)} \qquad \text{(式 3.14)}$$

$$R \approx \frac{\pi\sqrt{N}}{4} \sim \sqrt{N} \qquad \text{(式 3.15)}$$

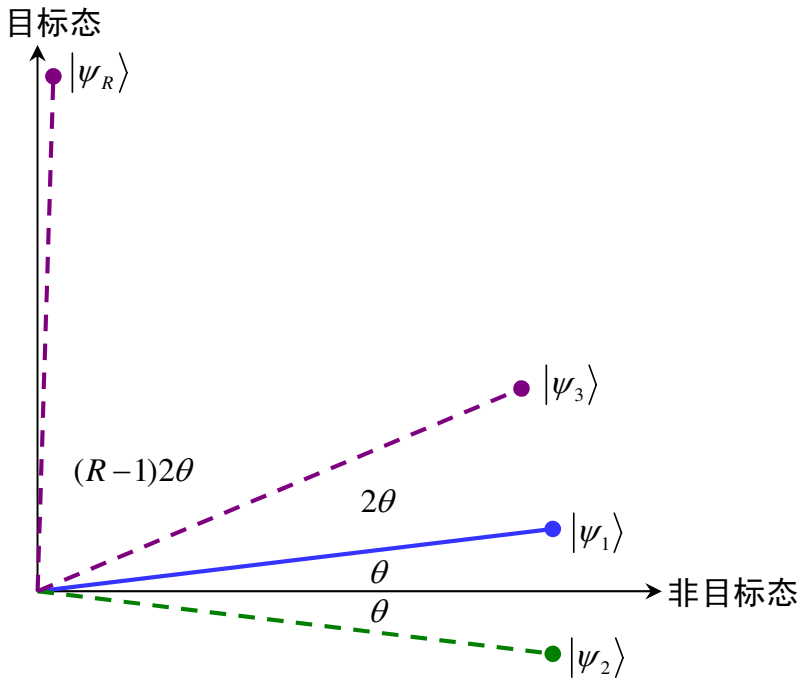


图 3-3 **Grover算法也相当二维态矢量空间中的旋转操作**

另外，注意到一般说来$(\pi\sqrt{N}/4)$并非整数，所以R次旋转不一定能精确输出目标态，所以Grover算法存在$O(1/N)$概率搜索失败[18]。

清华大学龙桂鲁教授提出[19]，将Grover搜索中的两个相位取反（180°旋转）操作（U_O中对目标态相位的取反，U_{ψ_1}中对所有非零态的相位取反,见图3-1）换成以下角度的旋转:

$$\phi = 2\arcsin[\sqrt{N}\sin(\frac{\pi}{4R+6})] \qquad \text{(式 3.16)}$$

则旋转R次（R可以任取，ϕ相应改变即可）后可以精确得到目标态，从而实现零失败率的量子搜索算法，其时间复杂度仍为$O(\sqrt{N})$。

3.3 波粒二象性计算机

Grover算法固然优于经典搜索算法，但是人们并不愿意满足于平方加速。那么我们能否做得更快呢？如果能，那将意味着什么呢？

美国麻省的Clay Mathematics Institute曾在千年之际提出七个世际难题，其中之一就是P与NP问题[20]。在经典计算理论中，一直无法有效解决的计算问题称为NP-Complete,也就是不能在多项式级时间复杂度内解决的计算问题，例如著名的背包问题，以及数据加密标准算法DES, AES等。NP-Complete问题间互相等价，n位无序数据库的搜索也是NP-Complete问题，如果能够实现$O(n)$或$O(log_2 N)$时间复杂度的量子搜

索算法，那么所有NP-Complete问题都迎刃而解，这将对计算领域与信息安全产生极其巨大的影响。

然而，美国加州理工学院的John Preskill教授早在1998年就证明[21]，在量子计算机上利用任何么正操作序列，都需要运行至少$O(\sqrt{N})$次Oracle运算才能获得搜索结果，换句话说，Grover算法已经是最佳的量子搜索算法，是不可以超越的。

多年以来，不断有人提出各种量子算法，企图超越Grover搜索算法，但最后都被证明是错误的。最近，清华大学龙桂鲁教授提出一种所谓的“波粒二象性计算机”[22]，指出它可以超越量子Grover搜索算法，在$O(1)$的时间复杂度内完成。其方案可以用如下的装置图（源于Mach-Zender干涉仪）来概述：

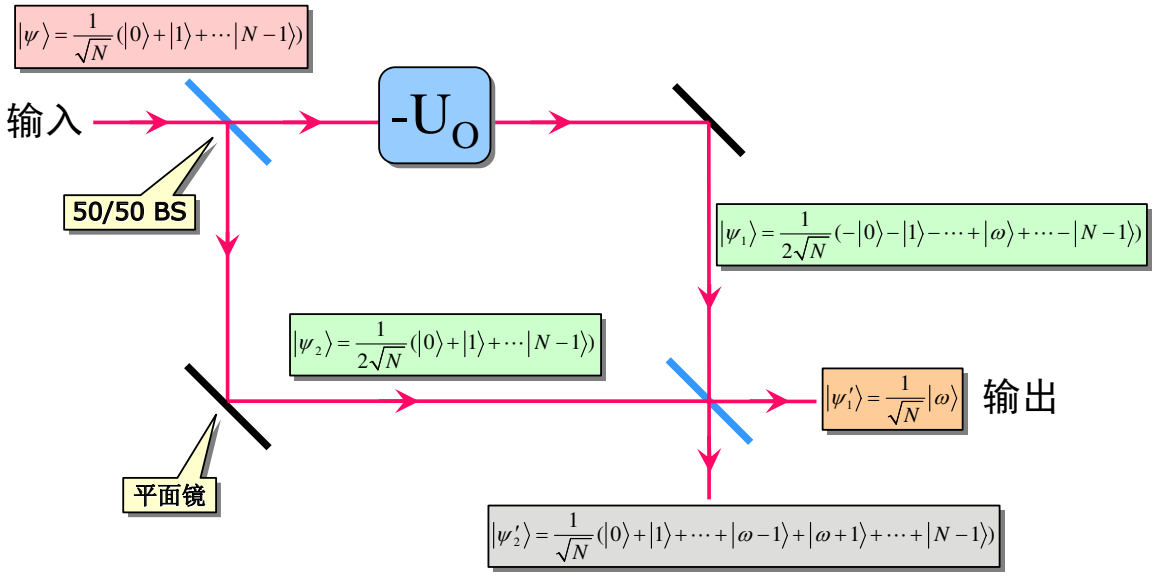


图 3-4 只需一次查询运算的搜索算法

这里采用了光子储存量子位信息。这个装置利用光子的波粒二象性这。50/50BS(Beam Splitter)的作用是将50%的光透射，50%的光反射，它相当于一个Hadamard门操作。如果只对上面的路径作Oracle运算U_O，并引入负相位，那么右边输出端是两条路径态矢量的相加结果，下边输出端则是两条路径态矢量相减的结果。可以看到右边输出端的$|\psi_1'\rangle$只含目标态，如果我们丢弃下边的$|\psi_2'\rangle$，只对右边输出端测量，是不是就一次性得到搜索结果了吗？

答案是否定的，原因是在右边的输出端只有$\langle\psi_1'|\psi_1'\rangle=1/N$的概率能测量到光子，其他时候我们根本得不到任何输出结果，所以需要反复运行多次装置，其消耗的总时间并不优于Grover量子搜索算法。

其实，这个所谓的“波粒二象性计算机”与我们前面讲述的量子计算机并没有实质性的区别，如果引入一个额外的量子位来表示光走的路径（向右用$|0\rangle$,向下用$|1\rangle$）我们可以把上面的装置用量子逻辑电路图表示：

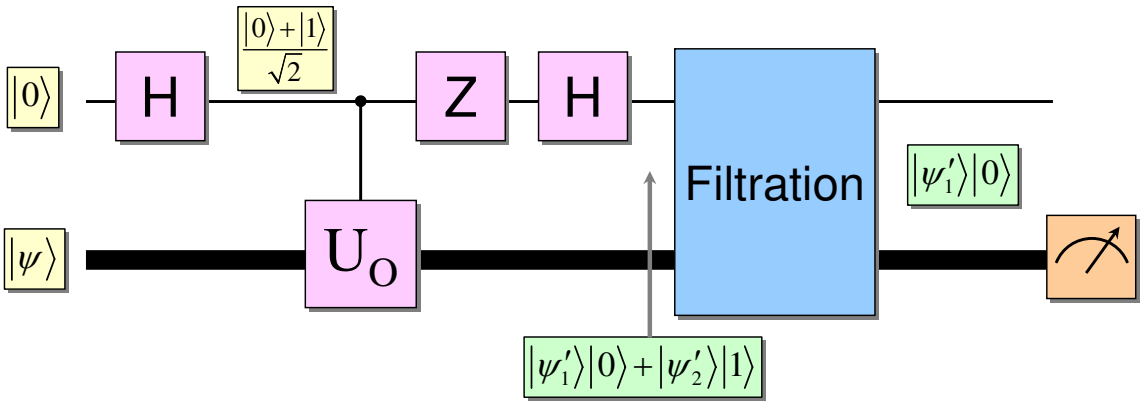


图 3-5 波粒二象性计算机的量子逻辑电路图

这里唯一前面未提到的操作是Filtration[1]（滤去），它的作用的吸收输出态$|\psi_1'\rangle|0\rangle + |\psi_2'\rangle|1\rangle$中含有$|1\rangle$的部分（即滤去下边输出端的光波），从而最终的输出只含目标态$|\omega\rangle$，这个Filtration操作不是么正的，但是只需对Preskill的证明加以修改，仍然可以证明，即使利用Filtration 的任意操作序列构成的算法，都不可能超越Grover搜索算法。这也说明量子计算机要想攻克NP-Complete问题也是较为困难的。下一章所论述的算法或许在此问题上更能给人以启发意义。

4 量子质因数分解算法

The problem of distinguishing prime numbers from composites, and of resolving composite numbers into their prime factors, is one of the most important and useful in all of arithmetic ..... The dignity of science seems to demand that every aid to the solution of such an elegant and celebrated problem be zealously cultivated.

—— *Carl Friedrich Gauss, As quoted by Donald Knuth*

If computers that you build are quantum,
Then spies everywhere will all want 'em.
Our codes will all fail,
And they'll read our email,
Till we get crypto that's quantum, and daunt 'em

—— *Peter Shor*

To read our E-mail, how mean
of the spies and their quantum machine;
be comforted though,
they do not yet know
how to factorize twelve or fifteen.

—— *Volker Strassen (著名经典算法学家)*

4.1 RSA体系的不安全性

目前广泛应用于金融、政府和国防等机构的RSA公开密钥密码体系，是由计算机学家Rivest, Shamir, Adleman三人于1978发明的[23]。RSA公钥密码体系的基本原理如下：

1. 找到两个大质数p和q（至少100-200位十进制数字），并计算$n = pq$
2. 随机选择一个小的奇数e,使其与(p-1)(q-1)互质
3. 计算$d = e^{-1} \mod (p-1)(q-1)$
4. 定义RSA公钥为整数对(e, n)，私钥为整数对(e, n)

它主要是利用两个大质数的乘积难以分解来作为安全性依据。因为从公钥(e,n)要得到私钥中的d就必须先分解n得到p,q。而长期以来，人们普遍认为，大数的质因子分解问题不存在经典的多项式算法（或者说有效算法），尽管尚未严格证明它是一个NP-Complete问题。但就目前经典计算机的计算能力而言，一个2000位以上的RSA公钥体系已经可以认为是绝对安全的。

然而，美国$AT\&T$实验室的Peter Shor于1994年在IEEE年会上提出了基于量子计算机的的大数质因子分解算法[24]，它可以多项式时间复杂度内得出计算结果。此算法引起巨大轰动，也掀起了研究量子计算机的高潮。如果量子计算机一旦投入实用，那么采用Shor算法可以在几分之一秒内实现1000位数的因子分解，这将使现有的公开密钥RSA体系无密可保!

4.2 量子傅里叶变换

Shor算法中的核心技术是量子傅里叶变换(Quantum Fourier Transformation)，它与经典的离散傅里叶变换(Discrete Fourier Transformation)[25]极其类似。

对正交基$|0\rangle,|1\rangle\ldots|N-1\rangle$的量子傅里叶变换定义下面的线性么正变换：

$$|x\rangle \xrightarrow{U_F} \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi ixy/N}|y\rangle \tag{式 4.1}$$

稍加运算，可以把量子傅里叶变换写成量子位的积形式：（以下$n=log_2N$）

$$|j_1j_2\ldots j_n\rangle \xrightarrow{U_F} \frac{(|0\rangle+e^{2\pi i0.j_n}|1\rangle)(|0\rangle+e^{2\pi i0.j_{n-1}j_n}|1\rangle)\ldots(|0\rangle+e^{2\pi i0.j_1j_2\ldots j_n}|1\rangle)}{2^{n/2}} \tag{式 4.2}$$

写成这种形式的好处是很容易设计量子逻辑电路，因为括号中的各项可以用相移门(Phase Shift Gate 见图2-2)R_k来实现

$$R_k=\begin{pmatrix}1 & 0\\ 0 & e^{2\pi i/2^k}\end{pmatrix} \tag{式 4.3}$$

容易计算知n位量子傅里叶变换电路中需要$O(n^2)$个逻辑门，而经典电路最少也需要$O(n2^n)$个逻辑门（使用FFT傅里叶变换），可见量子傅里叶变换相对经典算法有指数级加速。

4.3 求阶问题

我们先看一个称为求阶(Order Finding)的问题，它与大数的质因子分解问题可以互相转化。其定义为

定义 4.1：求阶问题：给定函数$f(x)=a^x \mod N$，a为与N互质且小于N，试求$f(x)$的周期。（a,x,N均为正整数）

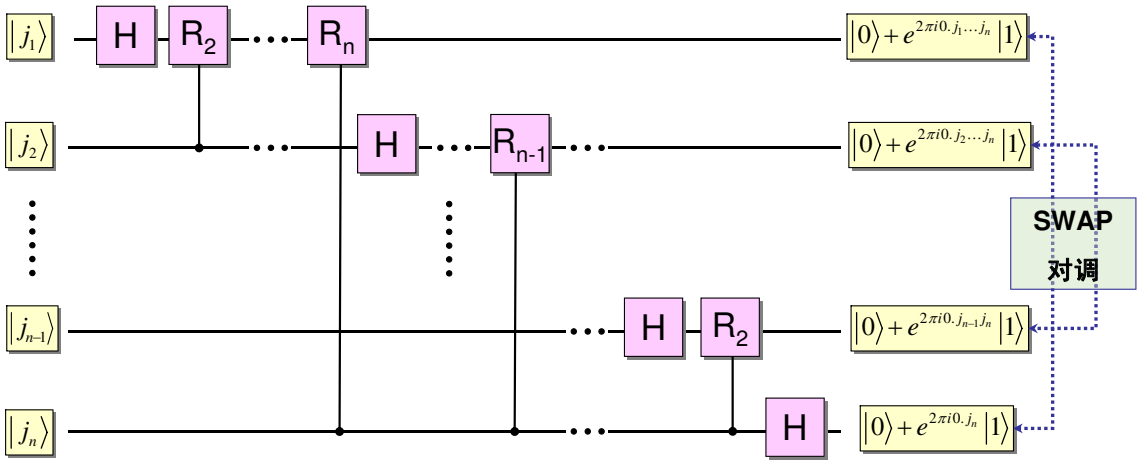


图 4-1 量子傅里叶变换逻辑电路图

数论上可以证明$f(x)$是周期函数，例如，令$N=21, a=2$，列表$f(x)$可知$f(x)$周期为6 量子求阶算法的步骤如下：（以下$n=\log_2 N$）

表 4.1 举例：f(x)是周期函数

f(0)=1	f(6)=1	f(12)=1
f(1)=2	f(7)=2	f(13)=2
f(2)=4	f(8)=4	f(14)=4
f(3)=8	f(9)=8	f(15)=8
f(4)=16	f(10)=16	f(16)=16
f(5)=11	f(11)=11	……

1. 初始化两个量子寄存器，一个$2n$位，用于保存x；另一个n位，用于保存$f(x)$：

$$|\psi_1\rangle|\psi_2\rangle = |0\rangle|0\rangle \qquad \text{(式 4.4)}$$

2. 然后对第一个寄存器作Hadamard门操作，使其成为N^2个态的均匀叠加（注：实际上只要用与N^2相近个数q的态就行，但要满足“光滑性”[26]）

$$|\psi_1\rangle|\psi_2\rangle = \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|x\rangle|0\rangle \qquad \text{(式 4.5)}$$

3. 计算$f(x)$，从而两个寄存器处于下面的纠缠态

$$|\psi_1\rangle|\psi_2\rangle = \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|x\rangle|f(x)\rangle \qquad \text{(式 4.6)}$$

4. 对第一个寄存器作量子傅里叶变换（见式4.1），两个寄存器里的纠缠态变为

$$|\psi_1\rangle|\psi_2\rangle = \frac{1}{\sqrt{qN}}\sum_{x=0}^{q-1}\sum_{x=0}^{N-1} e^{2\pi ixy/N}|x\rangle|f(x)\rangle \qquad \text{(式 4.7)}$$

5. 由于$f(x)$是周期函数，上面的求和式中有许多项可以合并，相消，或近似相消，最后得出只有y取以下值的项前系数明显不为0：

$$y = [k\frac{q}{T}] \quad (k = 0, 1, \ldots, T-1) \qquad \text{(式 4.8)}$$

6. 测量第一个寄存器，可以知道可能的样本值k，会发现k/q是T^{-1}的整数倍，从而容易推算出T

这个算法向来抽象难懂，于是，我们在经典计算机用Mathematica 5.2做了数值模拟（程序代码见附录）（注意，经典计算机做量子傅里叶变换需要指数时间的复杂度，所以非常慢，只能模拟很小的N以示意）

下面的程序模拟中取$N = 15$，$q = 243 \approx 15^2$，这是q的选择不唯一，程序里通过专门的函数取了其中最小的一个，已减少运行时间。

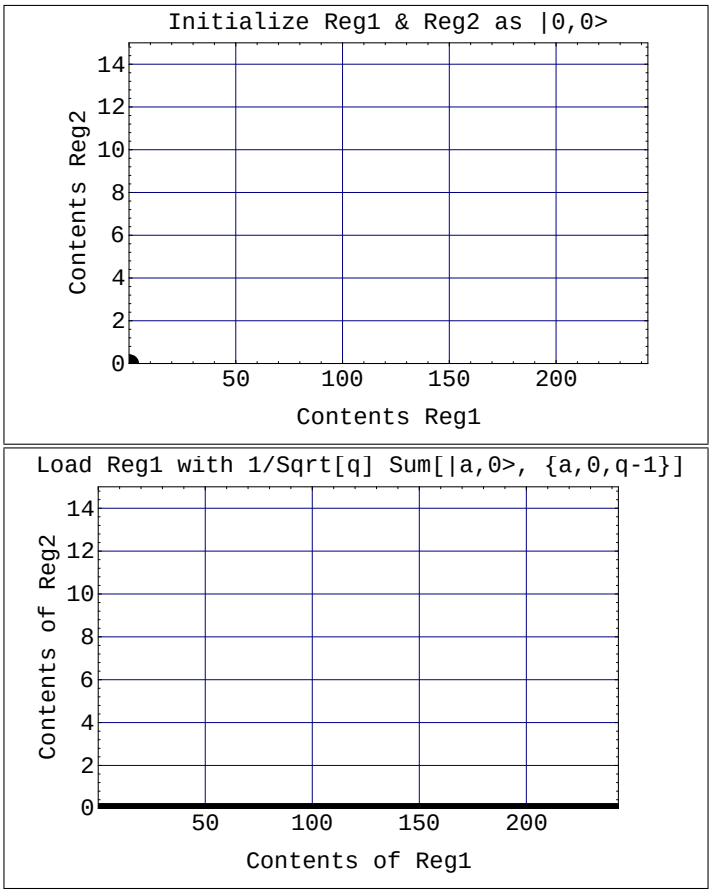


图 4-2 求阶算法第一二步时两寄存器的状态

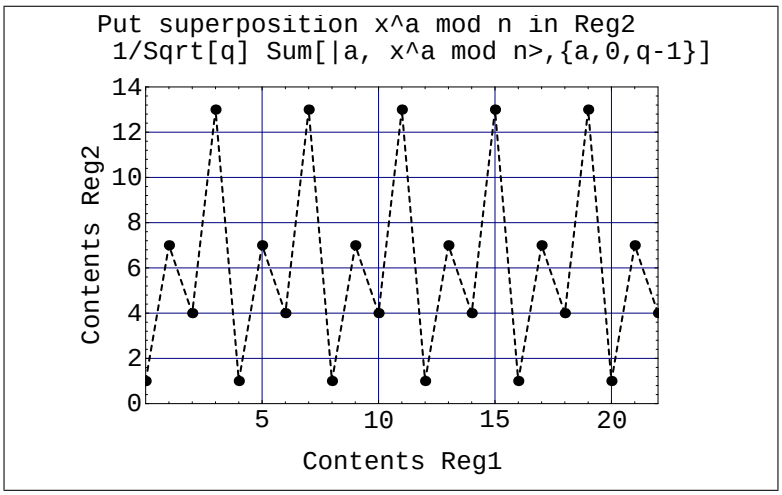


图 4-3 求阶算法第三步时两寄存器的状态

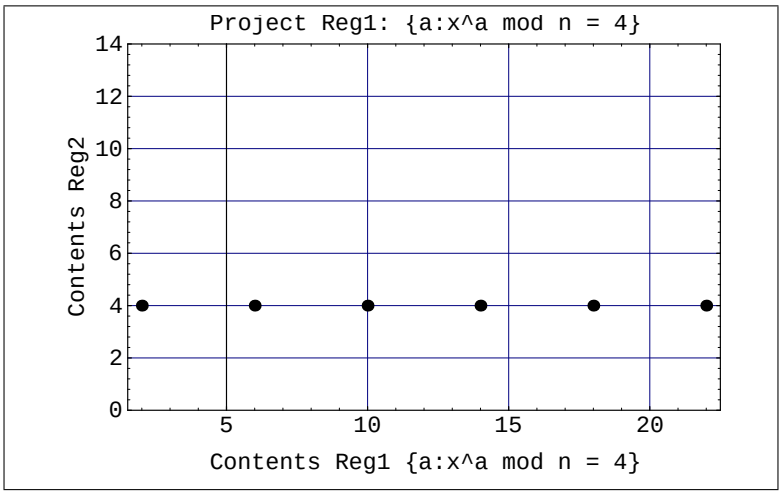


图 4-4 测量寄存器**2**后的一个可能状态

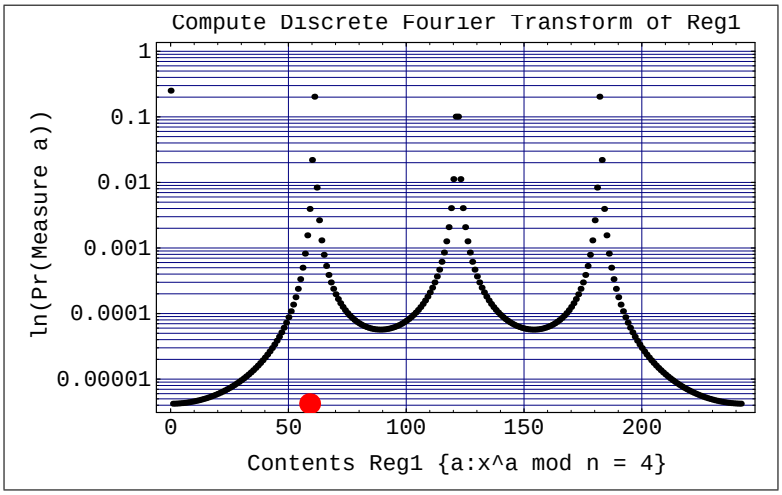
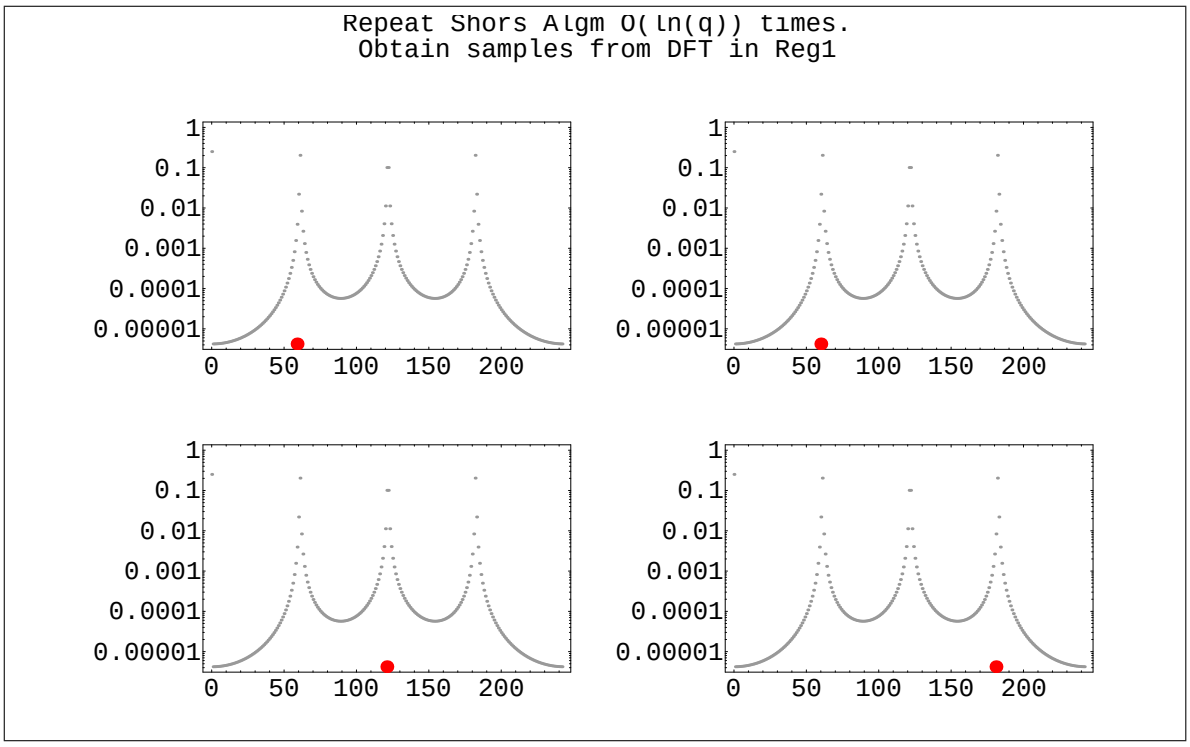


图 4-5 对寄存器**1**作傅里叶变换并测得一个可能**k**值




```
Found samples: {ket[182], ket[0], ket[0], ket[181], ket[0], ket[121],
      ket[182], ket[0], ket[182], ket[121], ket[59], ket[60]}
```

图 4-6 反复前两步以获得所有样本k值

从获得的k值看，可以发现:

$$\frac{k}{q} \approx \frac{0}{4}, \frac{1}{4}, \frac{2}{4}, \frac{3}{4} \qquad \text{(式 4.9)}$$

从而得知周期$T = 4$，注意实际程序中是用连分数算法(Continued fractions Algorithm[18]从k值集合中求出T来的，这里由于篇幅所限，不再叙述。

4.4 Shor 算法与经典模拟

下面基于求阶算法，给出Shor提出的大数质因子分解算法，简称Shor算法：

- **算法：Shor大数质因子分解**
- **输入：** 一个合数N
- **输出：** N的一个质因子
- **运行时间：** $O[(\log N)^3]$次迭代，成功概率$O(1)$
- **步骤:**

1. 如果N是偶数，返回2

2. 判断是否$N = a^b$，如果是的话返回a（用经典多项式算法即可）
3. 随机在$[1, N-1]$内选取一个整数x.如果$\gcd(x, N) > 1$则返回$\gcd(x, N)$
4. 用求阶算法寻找周期T
5. 如果T是偶数并且$x^{T/2} \neq -1 \mod N$，那么计算$\gcd(x^{T/2-1}, N)$和$\gcd(x^{T/2+1}, N)$，如果它们不是平凡的则返回它们，否则算法失败

以下是在经典计算机上模拟的结果（求阶算法部分见图4-2至4-6）

```
RunShorsAlgorithm[15]
Step 1: Picking a random integer x, 1 < x < 15, that is co-prime to 15.
>> Picked x == 7.
Step 2: Picking a "smooth" q, i.e. an integer of order O(n^2) ~ 225
that has small prime factors.
>> Picked q == 243.
Step 3: Computing the period of 7 mod 15.
Repeat steps 3(a)-3(g) O(log(q)) times.
                  ......
>> Period was r=4
Step 4: Obtain the factors of n from the period, r.
Computing the factors of n from
GCD[x^(r/2) - 1, n] and
GCD[x^(r/2) + 1, n]
In this case, x=7, r=4, n=15
GCD[49 - 1, 15] = 3
GCD[49 + 1, 15] = 5
>> So the factors of n=15 are 3 and 5.
```

图 4-7 **Shor算法经典程序模拟结果**

5 量子反事实运算

Quantum computer solves problem, without running

—— *Paul G. Kwiat*

5.1 Nature上的新闻

2006年2月，世界著名杂志《自然》上刊登了一篇由美国伊里偌斯州额班－香槟分校的Kwiat教授所在小组完成的文章[27]。文章从实验上证实，量子计算机在某种情形，可以在不运行的情况下告诉我们问题的正确运算结果（我们称其为量子反事实运算）。这一看似荒谬却反应物理事实的结论引起了学术界与新闻界的轰动。那么，让我们从头讲述这里面所包含的玄机与可能的应用前景。

5.2 Elizur-Vaidman 炸弹实验

量子反事实运算源于十几年前的一个有趣的假想实验，通常称为Elizur-Vaidman炸弹实验[28]。

假设现在有这样一个棘手的问题：在一个黑暗的箱子里，可能放有一个极其危险的炸弹，这个炸弹一旦遇到哪怕一个光子，就立即爆炸。而我们手头只有光学装置可用于探测炸弹的存在。那么有没有办法在不引爆炸弹的情况下知道箱子里有没有炸弹呢？

经典物理学告诉我们：不可能。但量子力学却给予我们奇特的力量－波粒二象性

Elizur与Vaidman二人用量子力学里最简单的单光子Mach-Zender干涉仪（参见图3-4）给出了可能的解答。

前面说过，两个BS(Beam Splitter)相当于Hadarmard操作。因此如果光路中没有炸弹，那么单光子必从最右边输出（两个Hadarmard操作等效于恒等操作），只有探测器1会检测到光子但如果光路中有炸弹，那么光子有50%机率被上边光路中的炸弹吸收并且炸弹爆炸。但另有50%机率光子走下边的光路，从而被第二个BS又分成两路。25%机率被探测器1检测到，25%机率被探测器2检测到。注意如果探测器检测到光子，那么可以肯定地推断，光路上必有炸弹（否则只有探测器1能检测到光子），又因为光子最终在输出端被探测到，说明它没有被炸弹吸收，炸弹也没有被吸收。也就是说，我们有25%机率成功地完成任务!

这个装置在1995年被Kwiat教授等用光学实验证实[29]。

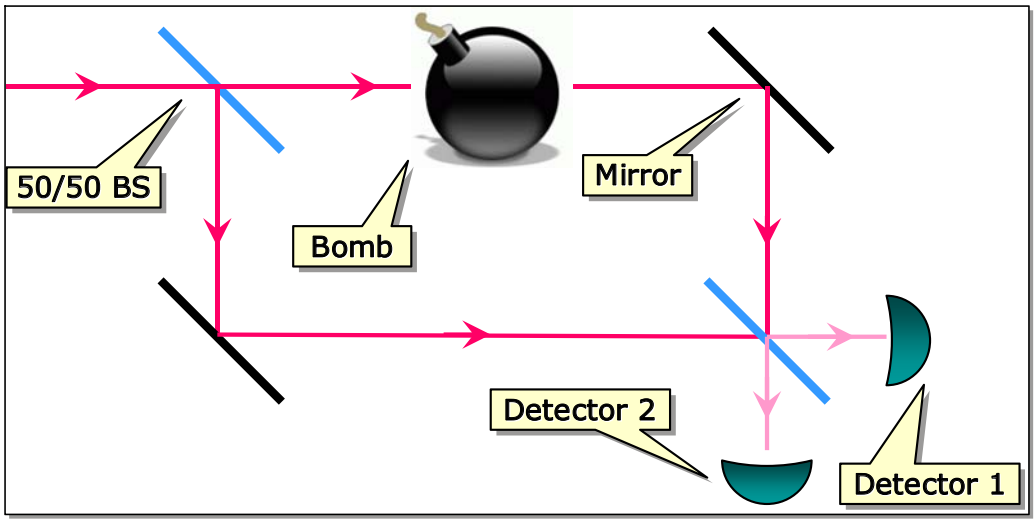


图 5-1 **Elizur-Vaidman 炸弹实验装置**

5.3 量子芝诺效应

但是，我们注定要有50%被炸弹炸掉吗？能否绝对安全地知道探测到炸弹的存在呢？

奇特的量子世界告诉我们：如果运用量子芝诺效应，这也是可能的！[30]

提到芝诺效应，人们自然会想起著名的芝诺佯谬。如果兔子与乌龟赛跑的过程中，我们不断地观察兔子有没有追上乌龟，那么兔子每次追到乌龟上次到的地点，乌龟已经又向前爬了一段距离。这样兔子永远追不上乌龟！

量子芝诺效应与其类似，说的是**如果在微观体系状态的演化过程中，不断地观察／测量这个状态，那么这个状态的演化将无法继续**[31]

在图5-3中，一个初态为水平极化的光子（用$|0\rangle$表示）进入了一个N周的光学循环，光每次循环要通过一个极化方向旋转器(Polarization Rotator)，它每次将光子的极化方向旋转小角度$\theta = 90°/N$，例如第一次时：

$$|0\rangle \longrightarrow \cos\theta|0\rangle + \sin\theta|1\rangle \tag{式 5.1}$$

如果光路中没有炸弹，则光子每次循环旋转θ角，N次后转成垂直极化的光子：

$$|0\rangle \longrightarrow \cos\theta|0\rangle+\sin\theta|1\rangle \longrightarrow \cos 2\theta|0\rangle+\sin 2\theta|1\rangle \longrightarrow \ldots \longrightarrow \cos N\theta|0\rangle+\sin N\theta|1\rangle = |1\rangle \tag{式 5.2}$$

但如果光路中有炸弹，并且炸弹只吸收垂直极化部分的光子（即$|1\rangle$前的系数）。那么光子总是被迫停留在水平极化状态。

$$|0\rangle \to \cos\theta|0\rangle+\sin\theta|1\rangle \to \cos\theta|0\rangle \to \cos\theta(\cos\theta|0\rangle+\sin\theta|1\rangle) \to \cos^2\theta|0\rangle \to \ldots \to \cos^N\theta|0\rangle \tag{式 5.3}$$

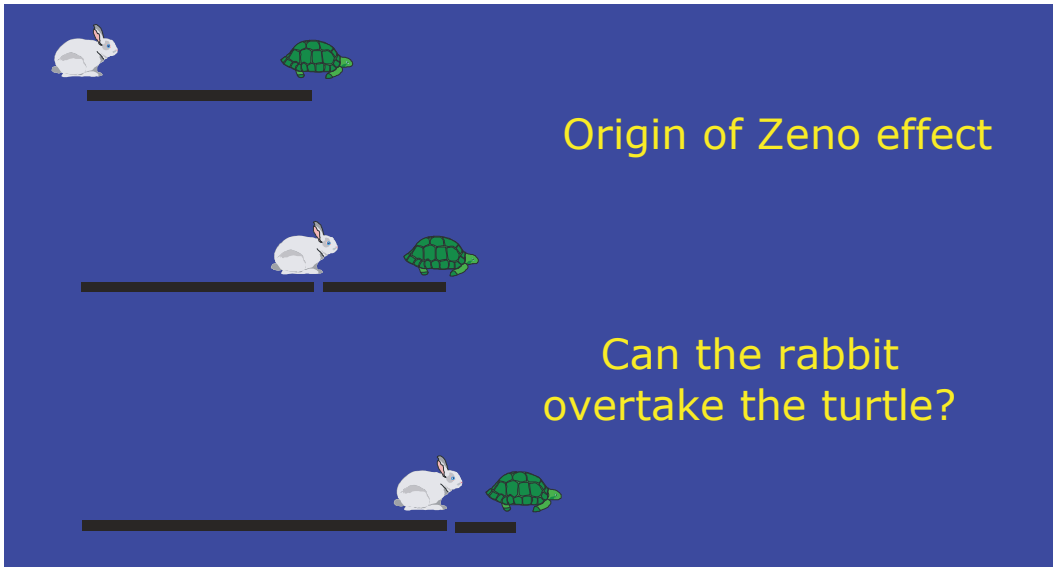


图 5-2 芝诺佯谬：兔子永远追不上乌龟

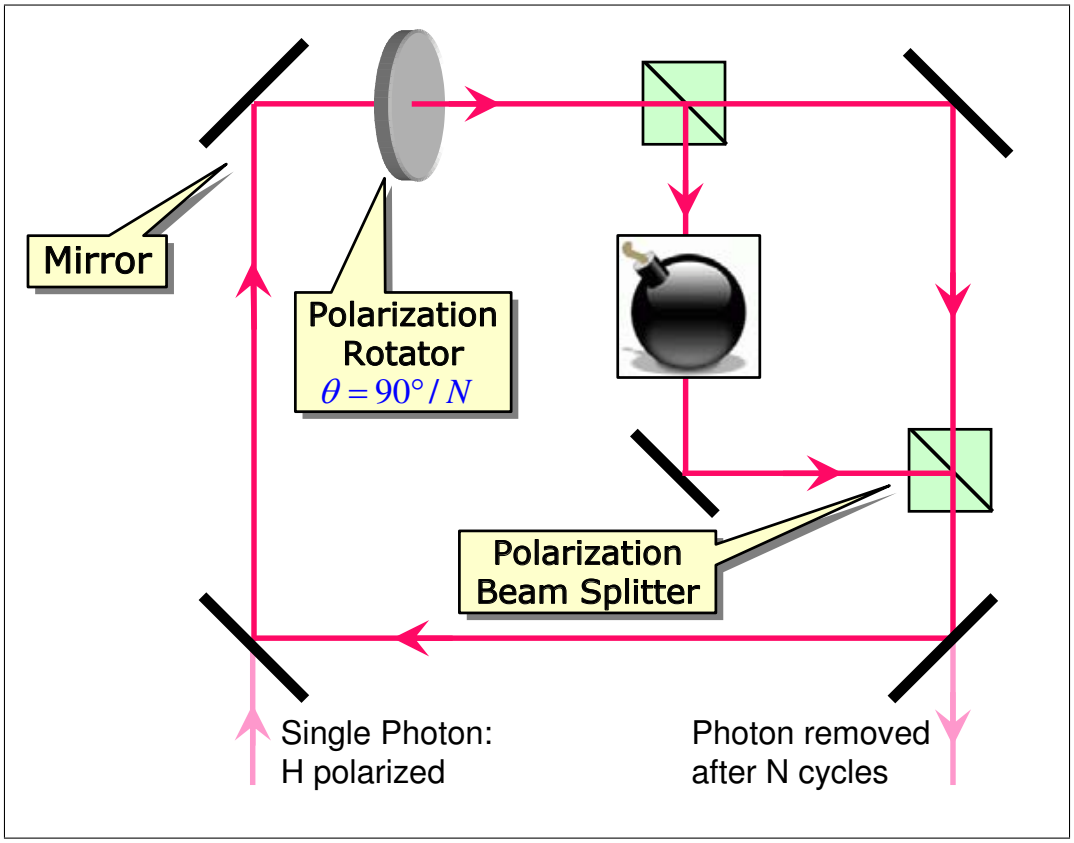


图 5-3 利用了量子芝诺效应的炸弹实验装置

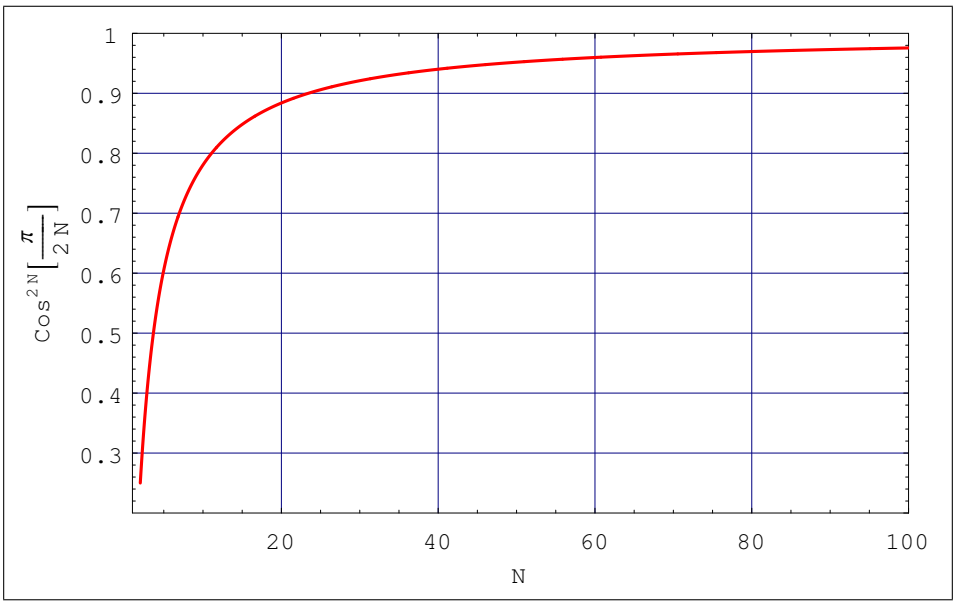


图 5-4 **循环次数N与安全探测到炸弹概率间的关系**

这样，在输出端能有$\cos^{2N}[90°/N]$的几率检测到水平极化的光子。这说明光路中一定有炸弹（否则输出光子一定是垂直极化的）。又因为光子能够顺利输出，说明炸弹没有吸收光子而被引爆。注意当$N \to \infty$时$\cos^{2N}[90°/N] \to 1$（见图5-4），所以我们只要增加循环次数（相应调整极化方向旋转器的角度θ）就能以接近1的概率在不引爆炸弹的情况下获知炸弹的存在与否。

5.4 量子反事实运算

既然能让炸弹不爆炸，那么也能让量子计算机不运行。方法很简单，我们把图5-3中的炸弹换成一个量子计算机(Quantum Computer, QC)，这个量子计算机计算某一问题，然后把输出结果（0或1）用光子的出射方向来表示。如果没有光子通过计算机，则计算机不运行。于是计算机输出1的情况下，一旦计算机运行，光子必然被吸收，而我们有$\cos^{2N}[90°/N]$的几率能在输出端检测到水平极化的光子，这说明计算机输出1且计算机没有运行，我们称这个几率为关于计算机输出的1**量子反事实运算概率(Counterfactuality)**，用p_1表示。

但是对于计算机输出0的情况，在输出端必然检测到垂直极化的光子，此时计算机已经运行了。所以**关于计算机输出1的量子反事实运算概率**为零，即$p_0 = 0$

英国剑桥大学的Mitchison与布里斯托大学的Jozsa于2000年提出了量子反事实运算的形式化理论定义。[32] 这个定义抽象难懂。在这里采用了最通俗的语言加以表述：[33]

定义 5.1： 当且仅当从整个量子计算机系统获得的输出同时满足下面两个条件时，可

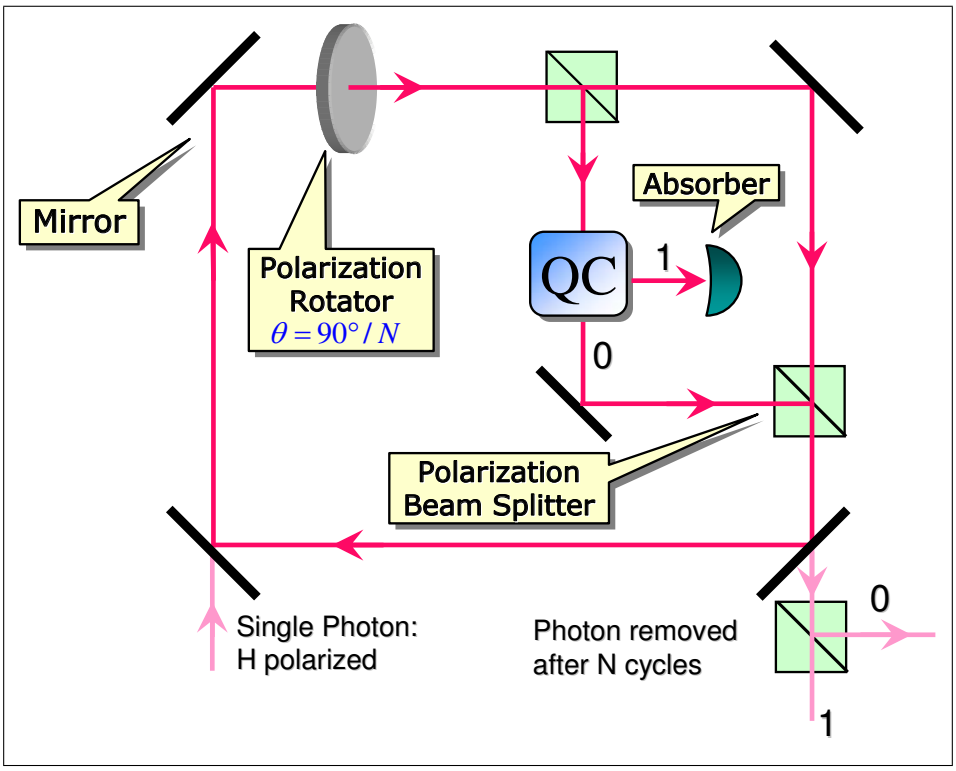


图 5-5 利用了量子芝诺效应的量子反事实运算装置

以在计算机实际上并没有运行的前提下得知计算机运算结果是r:

1. 如果计算机运算结果为r,产生该输出的所有可能的量子路径中不能存在有量子计算机。
2. 如果计算机运算结果为$1-r$，产生该输出的所有可能的量子路径的幅值之和为零。

这里的量子路径可以直观地理解为把光看成粒子，它可能走的路径。

以图5-5为例，它仅仅将图5-1的炸弹换成了量子计算机。我们前面已经分析过。当探测器2检测到光子时，我们知道计算机运算结果是1且计算机并没有运行。

1. 若计算机输出运算结果为1，则光子只有通过路径D才能通过探测器2，该路径不存在有量子计算机，条件1被满足。
2. 若计算机输出运算结果为0，则光子可以通过路径C, D到达探测器1，稍加计算可知这两条路径光的波振幅之和零，条件2被满足。

可见形式化定义是合理的。

5.5 定义中存在的问题

但是量子反事实运算的判断与定义并非这么简单。按照Mitchison与Jozsa的定义，

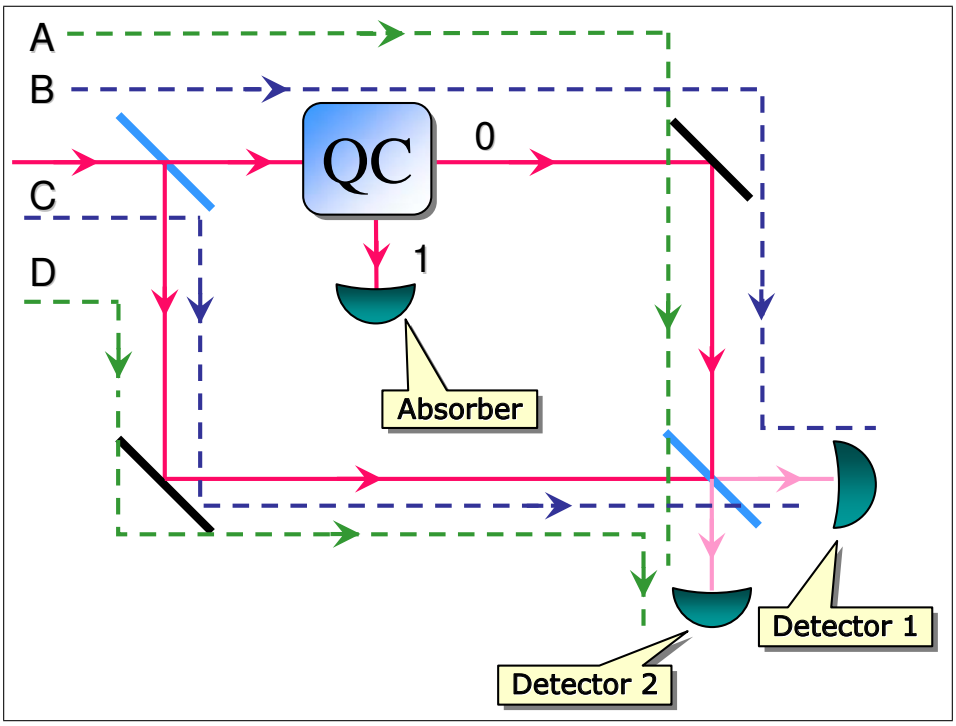


图 5-6 量子反事实运算定义解析

可以从数学上严格证明：[32]

$$p_0 + p_1 \leq 1 \qquad \text{(式 5.4)}$$

这意味着不可能在任意情况下都以趋近1的概率不让计算机运行而获得运算结果。因为即使是利用了量子芝诺效应（见图5-5），我们可以使$p_1 = \cos^{2N}[90°/N] \to 1$，但由于$p_0 = 0$所以满足5.4式。

然而令人惊奇的是，Kwiat组在Nature上发表的文章[27]采用了一种嵌套式的量子芝诺效应，居然能使：

$$p_0 \to 1 \quad \text{并且} \quad p_1 \to 1 \qquad \text{(式 5.5)}$$

也就是说，可以总让计算机不运行而知道运算结果!

尽管Nature文章的实验装置极为复杂，但基本原理容易解释。我们只需把图5-5的装置嵌入其中的量子计算机(QC)中，那么内循环输出运算结果为0（计算机已经运行）的光子被外循环吸收。这样如果最后输出有光子，那么无论对哪种运算结果，要么内循环计算机运行但外循环中光子根本不通过内循环，要么内循环计算机不运行但光子通过内循环。总之可以使p_0，p_1在足够多的循环次数下趋近于1，从而打破式5.4的限制!

于是引发了一个争论，Mitchison与Jozsa随后[34]指出，这个由他们的定义推导的限制是不可能被打破的，而Nature文章上的量子反事实运算在部分情况下并不满足它

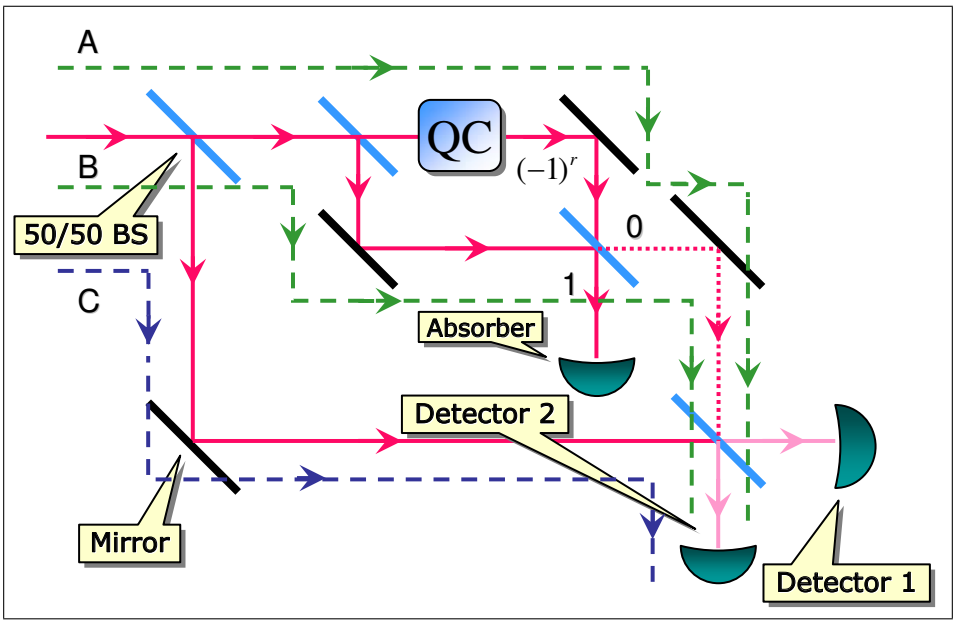


图 5-7 量子反事实运算定义的问题所在

们的理论定义，也就是说计算机并不是真正的没有运行。然后Kwiat组[35] 迅速回应，指出他们的方案没有问题，到是Mitchison与Jozsa的定义需要修改才严格正确，但他们也无法给出满意的修正定义。

经过仔细研究，我们可以把存在的矛盾集中概括于图5-7的装置：

这里的量子计算机在运算结果为1时把通过它的光子的相位取反，从而光子通过干涉仪后被Absorber吸收，因此它等价于图5-6的装置，也就是说当探测器2检测到光子时，我们知道计算机运算结果是1且计算机并没有运行。

然而按照Mitchison与Jozsa的定义，因为此时光子可以通过A, B, C三条路径到达探测器2，而由于路径2中有量子计算机，条件1并不成立。从而这个定义会给出不符合事实的错误判断。那么问题出在哪儿呢？

原来路径B, C在Absorber处的BS那儿相消干涉，光子根本不可能走点虚线的路径到达探测器2。所以尽管路径A中存在量子计算机，但一旦探测器2探测到光子，光子必然不可能走路径A，从而计算机实际上并未运行。

5.6 经过改进的定义

基于当前存在的矛盾，我们提出了改进的量子反事实运算定义，它适用于一切量子计算过程。[33]

定义 5.2： 当且仅当从整个量子计算机系统获得的输出同时满足下面两个条件时，可以在计算机实际上并没有运行的前提下得知计算机运算结果是r:

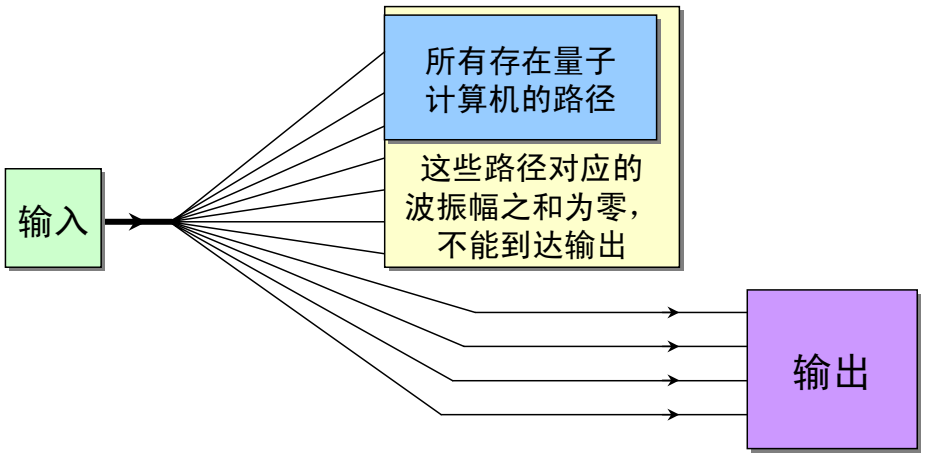


图 5-8 改进定义的直观图解

1. 如果计算机运算结果为r,那么存在一个幅值之和为零的量子路径集合，它包含所有存在有量子计算机的路径。
2. 如果计算机运算结果为$1-r$，产生该输出的所有可能的量子路径的幅值之和为零。

这一定义可以用下面的示意图清楚地表示出来。

除此以外，量子反事实运算的定义还可以用量子程序设计语言qGCL精确表述。qGCL(quantum Guarded Command Language)是由英国牛津大学的Paolo Zuliani提出的[36]，它是从著名经典计算机学家Dijkstra的Guarded Command Language(GCL)拓展形成的。其文法定义见图5-9:（采用扩充的Backus范式表述）

Zuliani于2005年将量子反事实运算的用qGCL进行了表述[37],但是Zuliani并没有充分理解量子反事实运算的物理含义，所以没有考虑量子干涉效应，其定义比Mitchison与Jozsa的定义更加不符合物理事实。这里给出了基于qGCL的量子反事实运算精确定义[33]（注：词法说明请参见[37])

⟨量子程序⟩ ::= ⟨量子语句⟩{;⟨量子语句⟩}

⟨量子语句⟩ ::= 态矢量 := ⟨么正操作⟩(态矢量) ‖

Fin[力学量本征基完备集](⟨力学量本征值⟩,⟨态矢量⟩) ‖

In(⟨力学量⟩) ‖ **skip** ‖ 变量 := 力学量本征值 ‖ ⟨循环语句⟩ ‖ ⟨条件语句⟩ ‖

⟨非可确定选择语句⟩ ‖ ⟨概率性选择语句⟩ ‖ ⟨局部语句块⟩ ‖

⟨循环语句⟩ ::= **while** ⟨条件⟩ **do** ⟨量子语句⟩ **od**

⟨条件⟩ ::= ⟨逻辑表达式⟩

⟨条件语句⟩ ::= **if** ⟨条件⟩ → ⟨量子语句⟩ **fi**

⟨非可确定选择语句⟩ ::= ⟨量子语句⟩ □ ⟨量子语句⟩

⟨概率性选择语句⟩ ::= $\langle$量子语句$\rangle_{p}$ ⊕ $\langle$量子语句$\rangle_{1-p}$

⟨局部语句块⟩ ::= **var** • ⟨量子语句⟩ **rav**

图 5-9 **qGCL的BNF文法定义**

先定义量子计算过程G，它是如下的一个有穷过程：

proc G (**value** $t \in \mathbb{B}$, **result** $h \in seq(\mathbb{B}^n)$, **result** $sw \in seq(\mathbb{B})$, **result** $m \in \mathbb{B}^n$) ≜

var $r \in \mathbb{B}$, $s \in \mathbb{B}$, $a \in \mathbb{B}^{n-2}$, $i \in \mathbb{B}^n$ •

$h, sw, m := \langle\ \rangle$; $r, s, a := \delta_0$;

while *cond* **do**

$r \otimes s \otimes a := \varepsilon(r \otimes s \otimes a)$;

Fin[Δ]$(i, r \otimes s \otimes a)$;

$h := h + i$; $sw := sw + s$;

if $s \to r := \mathrm{QC}_t(r)$; **fi**

od

Fin[Δ](m, ψ);

rav

当且仅当G满足下面两个条件时，量子计算机没有运行而可知道运算结果为r

1. $\exists hs \bullet (wp.\sum_{h} G(t,h,sw,m).[h \in hs \wedge sw \in 0^*] + wp.\sum_{h'} G(t,h',sw,m).[sw \notin 0^*]) \equiv 0$

2. $\sum_{h} G(1-t,h,sw,m) \equiv 0$

图 5-10 **qGCL表述的量子反事实运算定义**

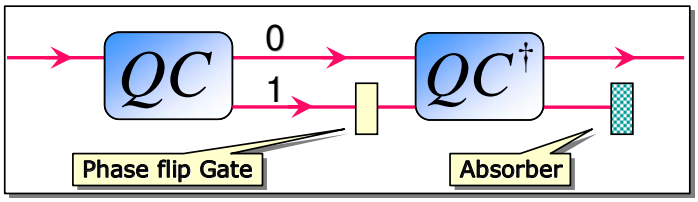


图 5-11 利用量子反事实运算来降低错误率

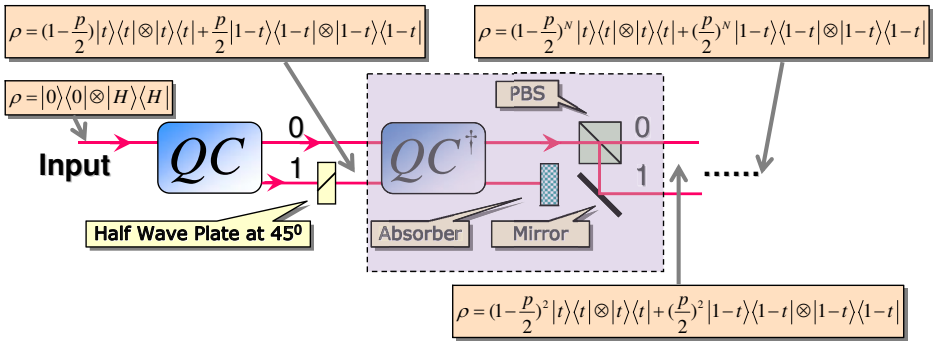


图 5-12 利用可逆性的降错方案装置

5.7 可逆计算机的有效降错方案

量子反事实运算固然神奇，但是人们更关心的是它的应用。一个自然的想法是它应该可以抑制错误的发生。因为既然计算机没有运行，那么它在运算过程中是否发生错误就不重要。那么这个应用是否可行呢？

Kwiat组在Nature上的文章中实现了一种错误抑制方案，他们将前面图5-5中的量子计算机换成下图5-11的装置。然后实现了一定的误差抑制效果。（其中$QC^{\dagger}$是量子计算机的逆算法，能将运算结果还原到初始状态）然而Mitchison与Jozsa随后[34]指出，这一方案的降错效果并不理想，甚至还远不如直接把计算机运行多次，然后取多数结果为正确结果的效果好。Kwiat组则回应说[35] 他们方案有独特的优点，可以直接把较准确的结果送到下一级的量子计算过程，而不需要反复运行计算机并通过投票机制选出正确结果。那么我们能否把两方面优点结合起来呢？既能非常有效的降低错误率，又能直接把结果送入下一步计算操作？

答案是可以[38]。需要用到的是量子计算机的威力之一——**可逆性**。只需将图5-11中的装置改成图5-12的装置，然后反复重复阴影部分的操作，可以把错误率降到趋近于零。

这个降错算法(Error-suppression scheme)的严格数学表述较复杂，可参见附录2中的文章[38]（已投往国际权威物理期刊*Physical Review A*，经清华大学龙桂鲁教授推荐，现已刊登在美国Cornell大学的预印本网站上）

事实上，这个算法也适用于经典计算机，但要求经典计算机的逻辑门都是可逆的（对于量子计算机，它先天具有可逆性）。下面是用经典计算机对该算法作的一个

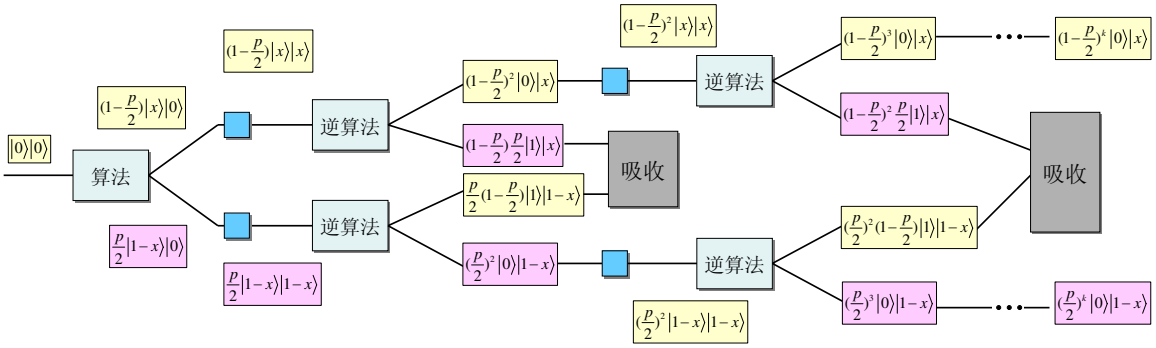


图 5-13 简化的降错方案装置

通俗描述：

这里做了最简单的假设，即经典算法输出的结果是单比特的。保存结果的比特初始化为0。如果算法运行正确（概率设为$(1-\frac{p}{2})$，那么输出x，错误(概率为$\frac{p}{2}$)，则输出$1-x$（图中红色框表示错误输出，下同）；对于逆算法，如果正确，则把x还原为0，$1-x$还原为1，否则相反。为了实现纠错，需要引入了一个额外的比特，它不参与任何算法，只是在中间某些地方存放中间结果（图中蓝色的逻辑门表示两个比特间的复制操作）。

我们看到，随着逆算法运行次数k的增多（实际通过循环来运行逆算法），错误输出的概率$\epsilon=(p/2)^k$以指数形式下降（见图5-14）

此外，如果算法输出的结果有N个比特，通过计算可以给出k次循环后的最终输出错误率：

$$\epsilon(N,p,k)=\frac{1}{1+\frac{[2^N(\frac{1}{p}-1)+1]^k}{2^N-1}} \qquad \text{(式 5.6)}$$

图5-15指出，这个降错方案会因算法规模的增大（N的增大）而效果更好。这使得这一方案非常实用。

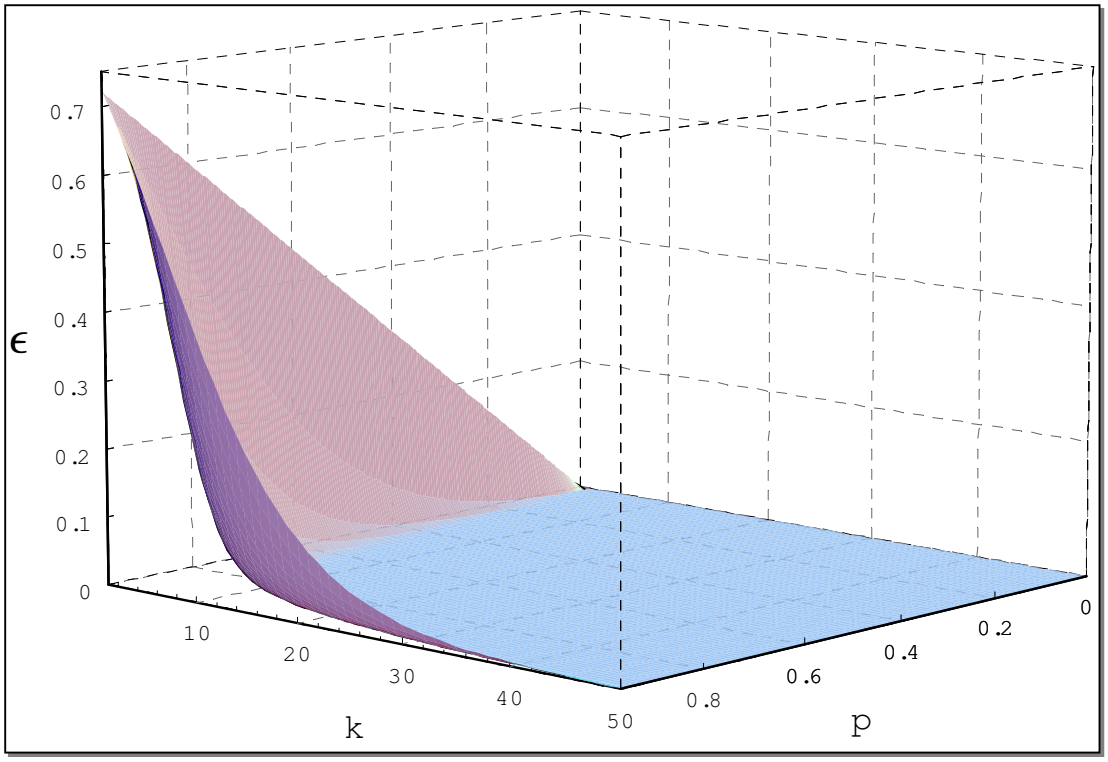


图 5-14 错误输出的概率以指数形式下降

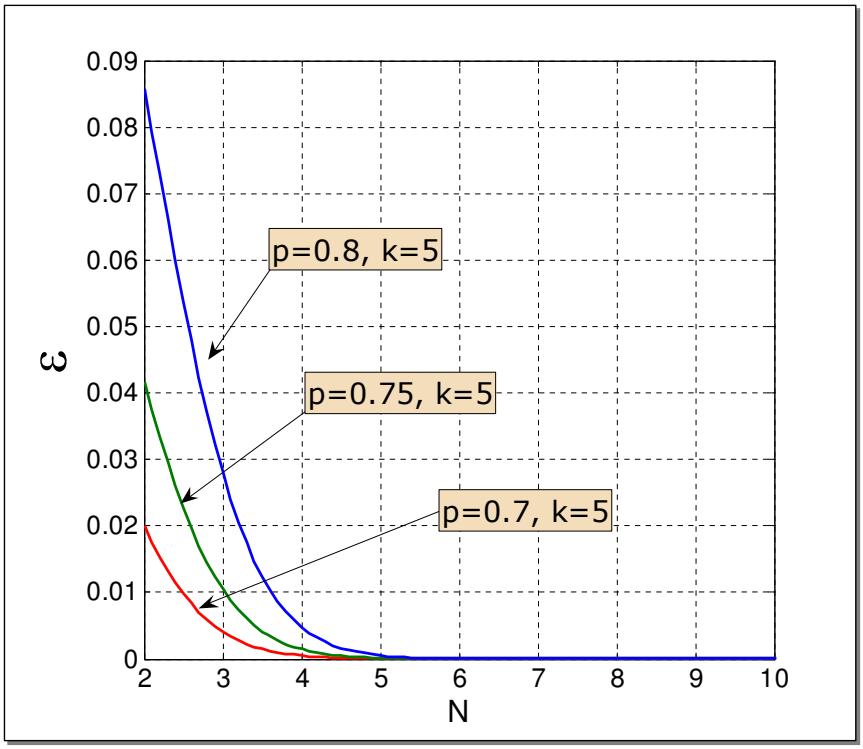


图 5-15 算法规模的增大使降错效果更好

6 总结与展望

6.1 全文总结

本文详尽地叙述了有关量子计算机与量子算法的一系列知识，以及我量子反事实运算这一前沿课题的研究成果。文章共6章24节，超过3万字。32幅插图全部为作者动手精心绘制的矢量图形。文章绝大部分内容出自归纳与创新，其中有原创性的工作包括：

- Shor算法的经典计算机程序模拟
- 波粒二象性计算机的分析与评论
- 量子反事实运算的定义改进
- 基于可逆计算机的量子降错方案

6.2 未来的工作

尽管量子计算机尚未走向实用化（目前实验室中最先进的装置只能处理十几个量子位，与经典计算机差距都甚远），其原因是量子系统过于精密脆弱，很容易受周围环境扰动而被破坏，丢失信息处理的能力。目前人们还在努力寻找好的物理系统（如超导电路，固态纳米材料，冷原子等）去实现量子位与量子位的操纵。一旦这些小规模的量子电路能够集成，那么如同经典计算机一样，量子计算机也将得以迅速发展。

迄今为止，量子计算领域公认最有用的算法仍然是Shor的质因数分解算法，其原因是它提供了大大优于经典计算机的算法执行速度（举个形象的例子，经典计算机分解一个大数需要计算$2^{100} \approx 10^{30}$次，而量子计算机只需要10^2次左右）。假如量子计算机能够真正实现，人们不仅要担心社会的信息安全是否会失去保障。幸运的是，量子世界在为我们带来攻破经典加密体系时，也为我们带来了量子保密通信体系。它将从物理原理上保证通信的无条件安全，即便量子计算机也无法攻破。而这一领域目前已经接近于实用化，其前景非常乐观，值得我们投入到对它的研究中去。

致 谢

本文的完成过程是一个艰辛但又充满收获的过程。这里面离不开很多老师的帮助。我首先要感谢华中科技大学光电国家实验室肖亮老师从我着手毕业设计一开始以来的强有力帮助，还有华中科技大学光电国家实验室曹强老师对我选择这样一个有意义课题的指导与宝贵意见。

另外，我还要感谢我的专业导师，长江学者吴颖教授一直以来在物理学上给予我的教导与指点，也要感谢我在今年暑假参加的中科院量子信息暑期学校，量子技术与基础国际会议（ICQFT'06），亚洲量子信息学术会议（AQIS'06）上：量子计算机创始人之一、英国Bristol大学计算机系Richard Jozsa教授；以色列Tel Axiv大学Lev Vaidman教授；美国Los Alamos国家实验室著名物理学家W. Zurek；美国国家科学院院士、Carniege-Mellon大学Robert Griffiths教授；美国Illinois大学Urbana-Champaign分校J. Barreiro 博士；英国物理学会院士、清华大学龙桂鲁教授；以及中国科技大学杜江峰教授等与我有启发意义的讨论与交流，以及对我工作的肯定与改进意见。

本文的工作受国家自然科学基金项目号10575040与90503010，及国家基础科学研究基金项目号2005CB724508的支持。

参考文献

[1] Sakurai J J. *Modern quantum mechanics*. Redwood City: Addison-Wesley, 1994.

[2] Dirac P M. *Principle of quantum mechanics*. London: Oxford Univ. Press, 1958.

[3] Landauer R. Irreversibility and heat generation in the computing process. *IBM J.Res.Dev.*, 1961, 5(183).

[4] Turing A M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.*, 1936, 42(230).

[5] Church A. An unsolvable problem of elementary number theory. *Am. J. Math.*, 1936, 58(345).

[6] Yao A C. *Procedings of the 34th Annual Symposium of Foundation of Computer Science*. Los Alamitos, CA: IEEE Computer Society, l993,352.

[7] Deutsch D. Quantum computational networks. *Proc. R. Soc. London A.*, 1989, 425(73).

[8] Deutsch D. Universality in quantum computation. *Proc. R. Soc. London A.*, 1995, 449(1937):631–633.

[9] Lloyd S. Almost Any Quantum Logic Gate is Universal. *Phys. Rev. Lett.*, 1995, 75:346–349.

[10] Barenco A. Elementary gates for quantum computation. *Phys. Rev. A.*, 1995, 52:3457 - 3467.

[11] Pellizzari T. Decoherence, Continuous Observation, and Quantum Computing: A Cavity QED Model. *Phys. Rev. Lett.*, 1995, 75:3788 - 3791.

[12] Cirac I, Zoller P. Quantum Computations with Cold Trapped Ions. *Phys. Rev. Lett.*, 1995, 74:4091 - 4094.

[13] Gershenfeld N, Chuang I. Bulk Spin-Resonance Quantum Computation. *Science*, 1997, 275:350.

[14] Deutsch D, Jozsa R. Rapid solution of problems by quantum computation. *Proc. R. Soc. London A.*, 1992, 439:553.

[15] Knuth D E. *The art of computer programming, Volume III*. Redwood City: Addison-Wesley, 1998.

[16] Grover L. *In Proc. 28th Annual ACM Symposium on the theory of Computation*. New York: ACM Press, 1996.

[17] Grover L. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 1997, 79:325–328.

[18] Nilsen M A, Chuang I L. *Quantum computation and quantum information*. Cambridge, England: Cambridge Uni. Press, 200.

[19] Long G L. Grover algorithm with zero theoretical failure rate. *Phys. Rev. A.*, 2001, 64:022307.

[20] P vs NP Problem. *Clay Mathematics Institute, http://www.claymath.org/millennium/P_vs_NP/.*

[21] Preskill J. *Lecture Notes for Physics 229: Quantum Information and Computation*. California Institute of Technology, l998.

[22] Long G L. The General Quantum Interference Principle and the Duality Computer. *Commun. Theor. Phys.*, 2005, 45:825–844.

[23] Rivest R L, Shamir A, Adleman L M. A method of obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 1978, 21(2):120–126.

[24] Shor P. Algorithms for quantum computation: discrete logithms and factoring. *In Proceedings, 35th Annual Symposium on Foundations of Computer Science*, IEEE Press, Los Alamitos, CA, 1996, pages 56–65.

[25] Apostol T M. *Mathematical Analysis*. Massachusetts: Addison-Wesley, 1974.

[26] Shor P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comp.*, 1997, 26(5):1484–1509.

[27] Hosten O, Rakher M T, Barreiro J T, et al. Counterfactual quantum computation through quantum interrogation. *Nature*, 2006, 439:949–952.

[28] Elitzur A C, Vaidman L. Quantum Mechanical Interation-free measurement. *Found. Phys.* , 1993, 23:987–997.

[29] Kwiat P, Weinfurter H, Herzog T, et al. Interaction-Free Measurement. *Phys. Rev. Lett.*, 1995, 74:4763–4766.

[30] Kwiat P G, White A, Mitchell J R, et al. High-Efficiency Quantum Interrogation Measurements via the Quantum Zeno Effect. *Phys. Rev. Lett.*, 1999, 83:4725 – 4728.

[31] Misra B, Sudarshan E C G. The Zeno's paradox in quantum theory. *J. Math. Phys.*, 1977, 18:756.

[32] Mitchison G, Jozsa R. Counterfactual quantum computation. *Proc. R. Soc. Lond. A* , 2001, 457:1175–1193.

[33] Gong Z X. A study on the definition and application of counterfactual quantum computation. *Poster Presentation at International Conference on Quantum Foundation and Technology*, 2006, No. 32.

[34] Mitchison G, Jozsa R. The limits of counterfactual quantum computation. *quant-ph/0606092*.

[35] Hosten O, Rakher M T, Barreiro J T, et al. Counterfactual computation revisited. *quant-ph/0607101*.

[36] Zuliani P. Quantum Programming: [PhD Dissertation]. St Cross College, University of Oxford, 2001.

[37] Zuliani P. On counterfactual computation. *Lecture Notes on Computer Science, Springer*, 2005, 3699:251–266.

[38] Gong Z X. Effective error-suppression scheme for reversible quantum computer. *arxiv.org/quant-ph /0608152*.

附录 1　Shor算法的经典计算机模拟程序

```
Needs["Statistics`DataManipulation`"]

Options[RunShorsAlgorithm] = {Period->QuantumMethod};

RunShorsAlgorithm[n_, opts___]:=
        Module[{method,x,q,r},
                $DefaultFont = {"Courier", 10};
                method = Period /. {opts} /. Options[RunShorsAlgorithm];
                x = PickRandomInteger[n];
                DescribeStep1[x, n];
                q = PickGoodSmoothq[n];
                DescribeStep2[q, n];
                DescribeStep3[x, q, n];
                r = Which[method===QuantumMethod, Period[x, n, q],
                          method===ClassicalMethod, PeriodViaClassicalAlgorithm[x, n,
q],
                          True, Message[RunShorsAlgorithm::method]; Abort[]
                         ];
                Print["Period was r=",r];
                If[EvenQ[r],
                   (DescribeStep13[x,r,n];
                    {GCD[x^(r/2) - 1, n], GCD[x^(r/2) + 1, n]}
                   ),
                   Message[RunShorsAlgorithm::odd, r]
                  ]
        ]

RunShorsAlgorithm::usage =
        "RunShorsAlgorithm[n] creates a simulation of how a \
        quantum computer would find the factors of n using Peter \
        Shor's quantum factoring algorithm. \
        You can give RunShorsAlgorithm various options that control \
        the output from the simulator. The period of x^a mod n can be \
        calculated in one of two \
        ways. If you set Period->QuantumMethod (the default) then \
        RunShorsAlgorithm performs a faithful quantum simulation of \
        the steps a quantum computer would take to compute the period \
        of x^a mod n. This can be slow because of the need to \
        compute a discrete Fourier transform. If you are only \
        interested in seeing what factors a quantum computer would \
        have found (not how it did it) then you can set \
        Period->ClassicalMethod. This returns the same result as the \
        quantum simulation but computes the period in a classical way.";

RunShorsAlgorithm::method =
        "The method for computing the period must be either \
        Period->QuantumMethod or Period->ClassicalMethod.";

RunShorsAlgorithm::odd =
        "The period was an odd number (r=`1`). Shor's quantum \
        factoring algorithm failed. Try again!";
```

1. Pick a Random Integer Co-prime to n

```
(* Picks a random integer, x, that is co-prime to n.
   Two integers are co-prime if GCD[x,n]==1.
*)
PickRandomInteger[n_]:=
        Module[{x},
                x = n;
                While[GCD[x,n]=!=1,
                      x = Random[Integer, {2,n-1}]; (* choose random integer < n *)
                     ];
                x
        ]
```

2. Pick a Smooth q

```
(* Pick a smooth q i.e. an integer in n^2 <= q <= 2 n^2 that
   has small prime factors. Deutsch has shown that if you pick
   q=2^L for some integer L (still with n^2 <= q=2^L <= 2 n^2),
   then you can compute the discrete Fourier transform (needed
   later in the algorithm) very efficiently with a simple quantum
   circuit. However, q=2^L is not necessarily the SMALLEST q that
   is usable. In our (classical) simulation the size of q affects
   the efficiency of the simulator. Hence I opted to go with Peter
   Shor's original specification of q as being a smooth number of
   order O(n^2). I actually generate a few such smooth q's and
   pick the smallest provided it is smaller than the least power
   of 2 greater than n^2 (i.e. a q of the form
   n^2 <= q=2^L << 2 n^2). This was I ensure I am using about as
   small a q as I can (which speeds up the computation of the
   discrete Fourier transform. Computing the DFT is the slowest
   step which dominates the time to factor
   n in this simulator. I really should speed it up).

   The interpretation of a "small" prime being one of order
   O(Log[n]) is mine. It might be a bit conservative.
*)
PickGoodSmoothq[n_]:=
        Module[{smallPrimes, maxExponents},
                smallPrimes   = Table[Prime[i],{i,1,Log[n]}];
                maxExponents = Map[Floor[Log[#, N[2 n^2]]]&, smallPrimes];
                Min[Join[Table[PickSmoothq[n, smallPrimes, maxExponents], {20}],
                         {2^Ceiling[N[Log[2, n^2]]]}
                        ]  (* i.e. the latter is a q s.t. n^2 <= q=2^L <= 2 n^2 *)
                  ]
        ]

PickSmoothq[n_, smallPrimes_, maxExponents_]:=
        Module[{q},
                q = 0;
                While[Not[n^2 <= q <= 2 n^2],
                          q = Guessq[smallPrimes, maxExponents]
                         ];
                q
```

```
        ]

Guessq[primes_, expts_]:=
        Apply[Times,
              MapThread[#1^Random[Integer,{0,#2}]&, {primes, expts}]]
```

3. Compute the Period of x^a mod n (Quantum Method)

```
 Period[x_, n_, q_]:=
        Module[{reg1, reg2, measureReg2, projectReg1, fourierReg1,
                samplesOfReg1, valuesOfReg1, r},

                  reg1 = Apply[Plus, Map[(N[1/Sqrt[q]] ket[#])&, Range[0, q-1]]];
                  DescribeStep4[reg1];
                  DescribeStep5[reg1,q,n];

                reg2 = Map[Mod[x^#[[2,1]], n]&, Apply[List, reg1]];
                  DescribeStep6[reg1, reg2, q, n];

                measureReg2 = SampleRegister[reg2];
                  Print["Step 3(d): Measuring the state of Reg2"];
                  Print["measureReg2 = ", measureReg2];
                  DescribeStep7[reg1, reg2, q, n, measureReg2];
                  Print["The measurement of Reg2 has a side effect on the state of
Reg1."];
                  Print["In fact the state of Reg1 is projected into:"];

                projectReg1 = Renormalize[Select[reg1,
                                          (Mod[x^#[[2,1]], n] ==
measureReg2)&]];
                  DescribeStep8[reg1, reg2, projectReg1, measureReg2, n];

              Print["\n"];
              Print["Step 3(e): Computing discrete Fourier transform of the contents of
Reg1."];
              Print["This may take several minutes!"];
              Print["(Remember you are watching a classical simulation of a quantum
algorithm)."];
                fourierReg1   = QuantumDFT[projectReg1, q] // Chop;
                  DescribeStep9[fourierReg1, measureReg2];


                samplesOfReg1 = Table[SampleRegister1[fourierReg1], {12}];
        Print["\n"];
        Print["Steps 3(f)&3(g): Sample from Fourier spectrum created in Reg1"];
        Print["This entails repeating the previous steps & measuring Reg1 each
time."];
        Print["Found samples: ", samplesOfReg1];
                valuesOfReg1 = DeleteCases[Sort[Union[Map[First, samplesOfReg1]]],
0];
                  DescribeStep10[fourierReg1, measureReg2, First[valuesOfReg1]];
                  DescribeStep11[fourierReg1, valuesOfReg1];
        Print["\nStep 4: Extract the period, r, from the given samples."];
                DescribeStep12[];
                r = ExtractPeriodFromContinuedFraction[valuesOfReg1, n, q]
]
```

```
NotAllIntegersQ[{}]:=True
NotAllIntegersQ[ans_]:=
        Not[Apply[And, Map[IntegerQ[#[[2]]]&, ans]]]


QuantumDFT[w_. ket[a_], q_]:=
        Apply[Plus,
              Map[(N[w/Sqrt[q] Exp[2 Pi I a #/q]] ket[#])&, Range[0,q-1]]]

QuantumDFT[superposition_, q_]:=
        Map[QuantumDFT[#,q]&, Expand[superposition]]

Renormalize[superposition_]:=
        Expand[N[1/(Sqrt[Length[superposition]] Factor[superposition][[1]])]
superposition]

SampleRegister1[superposition_]:=
        Module[{elems, probs},
                elems = Map[#[[2]]&, Apply[List, superposition]];
                probs = Map[Abs[#[[1]]]^2&, Apply[List, superposition]];
                BiasedSelect[elems, probs]
        ]

SampleRegister[contentsOfRegister_]:=
        Module[{counts, elems, probs},
                {counts, elems} = Transpose[Frequencies[contentsOfRegister]];
                probs = N[counts/Apply[Plus, counts]];
                BiasedSelect[elems, probs]
        ]
```

4. *Extract the Period from a Continued Fraction Expansion*

```
 Needs["NumberTheory`ContinuedFractions`"]

ExtractPeriodFromContinuedFraction[samplesOfReg1_, n_, q_]:=
        Module[{rValues, r},
         rValues = Map[ExtractPeriodFromOneSample[#,n,q]&, samplesOfReg1];
                r = MajorityOpinion[rValues];
                r
        ]

ExtractPeriodFromContinuedFraction::canCancel =
        "The quotient c/r = `1`/`2` can cancel further";

ExtractPeriodFromOneSample[k_, n_, q_]:=
        Module[{f, cPrime, r},
                f = ClosestFractionTo[k/q, n, q];
                cPrime = Numerator[f];
                r = Denominator[f];
                If[GCD[cPrime, r]==1,
                   r,
                   Message[ExtractPeriodFromContinuedFraction::canCancel, cPrime, r]
                  ]
        ]
```

```
ClosestFractionTo[f_, n_, q_]:=
        Module[{cfs, approximations},
                cfs = Table[Normal[ContinuedFraction[f, i]], {i, 1, n}];
                approximations = Select[cfs, Denominator[#]<=n&];
                Last[approximations]
        ]

(* Each time you approximate k/q by lambda/r you get a particular
   fraction lambda/r. Unfortunately, if lambda (the multiple of
   the inverse period) happens to share a common factor with the
   period, r, then Mathematica will cancel the common factor between
   lambda and r, converting lambda/r into lambda1/r1 (r1<r). Hence
   the estimated value for "r" (i.e. r1) will be too low. For this
   reason we repeat the algorithm a few times to obtain enough
   samples to be sure that we have guessed the correct period, r.
*)
MajorityOpinion[rValues_]:=
        Module[{freq},
                pairs = Frequencies[rValues];
                freq = Map[First, pairs];
                pos = Position[freq, Max[freq]][[1]];
                pairs[[pos]][[1,2]]
        ]
```

5. Utility: BiasedSelect

```
 (* The list of probabilities should sum to 1. The call to Partition
   constructs a set of probability intervals whose width is proportional
   to the probability with which the corresponding element in list
   is selected.
*)
BiasedSelect[list_, probabilities_]:=
        Module[{random},
                random  = Random[];
                Apply[Part[list, #]&,
                      Flatten[
                       Position[Map[InRangeQ[random, #]&,
                                    Partition[FoldList[Plus,0,probabilities],2,1]
                                   ],
                                True
                       ]
                      ]
                     ]
        ] (* /;CheckProbabilitiesQ[probabilities] *)

BiasedSelect::usage =
        "BiasedSelect[{e1,e2,...,en}, {p1,p2,...,pn}] returns element ei of \
        the first list with probability given in the second list pi.";

BiasedSelect::probabilityLeak =
        "You have a probability leak. The probabilities you specified do \
        not add up to 1.";

BiasedSelect::excess =
        "The probabilities you specified sum to greater than 1.";

CheckProbabilitiesQ[probabilities_]:=
```

```
        Module[{psum = Apply[Plus, probabilities]},
                Which[psum<1,  Message[BiasedSelect::probabilityLeak],
                      psum>1,  Message[BiasedSelect::excess],
                      psum==1, True
                     ]
        ]

InRangeQ[n_, {lb_, 1}]:=    lb <= n <= 1
InRangeQ[n_, {lb_, ub_}]:= lb <= n < ub
```

6. *Computing the Period of x^a mod n (Classical Method)*

This function is only used to provide a quick simulation of the input/output behavior of Shor's algorithm. It is a purely classical algorithm.

```
 PeriodOfSequence[sequence_] :=
  Module[{period, chunks}, period = 1;
    chunks = PartitionWithRmdr[sequence, period];
    While[ !PeriodicQ[chunks] && period <= Length[sequence],
     period++; chunks = PartitionWithRmdr[sequence, period]];
    If[ !PeriodicQ[chunks], Infinity, period]]

PeriodicQ[{{___}}] := False
PeriodicQ[{firstLists__, lastList_}] :=
  Module[{firstList}, firstList = First[{firstLists}];
    {firstList} === Union[{firstLists}] && Subsequence[lastList, firstList]]

Subsequence[{list__}, {list__, ___}] := True
Subsequence[_, _] := False

PartitionWithRmdr[list_, size_] :=
  Module[{rmdr}, rmdr = {Take[list, -Mod[Length[list], size]]};
    Join[Partition[list, size], If[rmdr === {{}}, {}, rmdr]]]
 (* This function computes the period of the sequence of
   values of x^a mod n directly. We use it simply to allow a
   fast (fake) simulation of Shor's quantum algorithm. In Shor's
   algorithm a quantum computer can compute the DFT very quickly.
   However, when we SIMULATE a quantum algorithm on a classical
   machine it appears to be much slower. Don't be fooled. The
   quantum algorithm for factoring an integer would, if run on a
   quantum computer, be much more efficient than any classical
   factoring algorithm, running on any classical computer. Indeed
   the inefficiency of simulating quantum computers on classical
   computers is inevitable. Otherwise you could achieve the same
   efficiency as a quantum computer simply by simulating a quantum
   computer on a classical machine. This is not so. There is no
   known way to simulate quantum physics on a classical machine
   without incurring an exponential slow down.
*)

PeriodViaClassicalAlgorithm[x_, n_, q_]:=
        Module[{reg1, reg2, lists, periods},
                reg1 = Apply[Plus, Map[(N[1/Sqrt[q]] ket[#])&, Range[0, q-1]]];
        Print["reg1 = ", Short[reg1, 8]];
                reg2 = Map[Mod[x^#[[2,1]], n]&, Apply[List, reg1]];
        Print["\nreg2 = ", reg2];
                PeriodOfSequence[reg2]
```

```
        ]

AllSameQ[first_, rest_]:=
        Apply[And, Map[first === #&, rest]]
```

7. Code for creating graphics

```
 Clear[PlotReg2VSReg1, PlotProbabilitiesVSContents]
 PlotReg2VSReg1[reg1_, reg2_, opts___]:=
        ListPlot[MapThread[{#1, #2}&, {reg1, reg2}],
                 Frame->True,
                 GridLines->Automatic,
                 opts
                ]

PlotProbabilitiesVSContents[w_. ket[i_] + kets_., opts___]:=
        Module[{probs, conts, grid, frame},
                probs = Probabilities[w ket[i] + kets];
                conts = Contents[w ket[i] + kets];
                ListPlot[MapThread[{#1, #2}&, {conts, probs}],
                        opts,
                        GridLines->Automatic,
                        Frame->True]
        ]

(* w_. ket[i_] + kets_. is a superposition of states of a quantum
   memory register. Probabilities[...] gives the probability with
   which each result would be obtained upon measuring the state of
   the register. Contents[...] gives the actual values that are
   stored in superposition in the register.
*)
Probabilities[w_. ket[i_] + kets_.]:=
        Module[{amplitudes},
                amplitudes = Map[#[[1]]&, Apply[List, w ket[i] + kets]];
                Abs[amplitudes/ Sqrt[Apply[Plus, Abs[amplitudes]^2]]]^2
        ]

Probabilities[superposition_List]:=
        Probabilities[Apply[Plus, superposition]]

Contents[w_. ket[i_] + kets_.]:=
        Map[#[[2,1]]&, Apply[List, w ket[i] + kets]]

Contents[superposition_List]:=
        Contents[Apply[Plus, superposition]]
```

Effective error-suppression scheme for reversible quantum computer

Zhe-Xuan Gong*

Department of Physics and Department of Computer Science,

Huazhong University of Science and Technology, Wuhan, 430074, China

(Dated: October 2, 2006)

We construct a new error-suppression scheme that makes use of the adjoint of reversible quantum algorithms. For decoherence induced errors such as depolarization, it is presented that provided the depolarization error probability is less than 1, our scheme can exponentially reduce the final output error rate to zero using a number of cycles, and the output state can be coherently sent to another stage of quantum computation process. Besides, experimental set-ups via optical approach have been proposed using Grover's search algorithm as an example. Some further discussion on the benefits and limitations of the scheme is given in the end.

PACS numbers: 03.67.Pp, 03.67.Lx, 89.20.Ff

I. INTRODUCTION

The goal of doing quantum computation and quantum information processing reliably in the presence of noise and decoherence has been pursued since the advent of quantum error correction, which was independently discovered by Shor [1] and Steane [2]. Later on, several different approaches to this goal have been studied. Error-avoiding codes [3] depend on existence of subspaces free of decoherence due to special symmetry properties, and bang-bang type control strategies [4], including the recent protocol using super-zeno effect [5], achieve the suppression of decoherence by suitably coupling the system strongly to an external system for short intervals.

Hosten, et al, in their recent paper [6], proposed a novel protocol for counterfactual computation using chained quantum zeno effect. They showed that in certain circumstances, their protocol could also eliminate errors induced by decoherence. However, Mitchison and Jozsa [7] argued that the actual benefit of this protocol seemed quite limited, in that one could resort to much simpler procedure of just running the computer for many times, which might even eliminate the errors more effectively in most situations. Reasonable as it is, Hosten, et al [8] then pointed out a key benefit of their protocol that truly outruns its rival. Based on their view, one of the potentially important aspects of any quantum computing protocol involves sending the output *coherently* to another stage of a quantum computer. The simple method of running the computer for many times cannot output an extremely pure answer easily, because it needs some sort of *majority voting* [9] schemes to yield the final answer, whereas the protocol using counterfactual quantum computation can make this benefit by cycling a single photon many times before sending it to the next processing stage with a low error probability.

Their interesting discussion therefore enlightens one to have a try of combining the profits of both protocols: the

*Electronic address: gongzhexuan@gmail.com

simplicity and efficiency of repeatedly running the computer and the coherent state transmission characteristic of error-suppression protocol with counterfactual computation. Here we propose a new error suppression scheme that may achieve this nirvana. We noted that the error suppression protocol introduced in [6] made use of the adjoint of Grover's search algorithm, which could undo the search process. Unlike their classical counterparts, many quantum computation processes are unitary, since quantum circuits are fundamentally reversible. One would then ask, naturally, that is it possible to take the advantage of the reversibility of quantum computers to help fighting against errors? The answer is yes.

II. A FIRST LOOK AT THE SIMPLEST CASE

First we'd like to show the big picture of our scheme in the simplest case. Consider the Grover's search algorithm (GSA) [10] for two database elements, which is apparently a unitary algorithm if there's no error:

$$|0\rangle \xrightarrow{GSA} |x\rangle \qquad |x\rangle \xrightarrow{GSA^\dagger} |0\rangle \tag{1}$$

where $x \in \{0, 1\}$ is the marked element. In our theoretical model, decoherence causes depolarization. Assume that, with probability $p \in [0, 1]$, the search algorithm becomes entangled with the environment, and outputs a mixed state:

$$|0\rangle\langle 0| \xrightarrow{GSA} (1-p)|x\rangle\langle x| + p\frac{I}{2} = (1-\frac{p}{2})|x\rangle\langle x| + \frac{p}{2}|1-x\rangle\langle 1-x| \tag{2}$$

$$|x\rangle\langle x| \xrightarrow{GSA^\dagger} (1-\frac{p}{2})|0\rangle\langle 0| + \frac{p}{2}|1\rangle\langle 1| \tag{3}$$

$$|1-x\rangle\langle 1-x| \xrightarrow{GSA^\dagger} \frac{p}{2}|0\rangle\langle 0| + (1-\frac{p}{2})|1\rangle\langle 1| \tag{4}$$

As a result, if we first run Grover's search algorithm, and then run the algorithm adjoint, we can get the original state $|0\rangle\langle 0|$ with a probability: $(1-\frac{p}{2})^2 + (\frac{p}{2})^2 \in [\frac{1}{2}, 1]$. Note that the first term $(1-\frac{p}{2})^2$ means that both algorithms are running correctly, while the second term $(\frac{p}{2})^2$ denotes that both have wrong outputs.

We could separate these two terms into orthogonal parts to reduce the depolarization error to $(\frac{p}{2})^2$ by adding an ancillary qubit that does not enter either algorithm. This will change the process into:

$$|0\rangle\langle 0| \otimes |0\rangle\langle 0| \xrightarrow{GSA} (1-\frac{p}{2})|x\rangle\langle x| \otimes |0\rangle\langle 0| + \frac{p}{2}|1-x\rangle\langle 1-x| \otimes |0\rangle\langle 0| \tag{5}$$

$$\xrightarrow{CNOT2} (1-\frac{p}{2})|x\rangle\langle x| \otimes |x\rangle\langle x| + \frac{p}{2}|1-x\rangle\langle 1-x| \otimes |1-x\rangle\langle 1-x| \tag{6}$$

$$\xrightarrow{GSA^\dagger} (1-\frac{p}{2})^2|0\rangle\langle 0| \otimes |x\rangle\langle x| + (\frac{p}{2})^2|0\rangle\langle 0| \otimes |1-x\rangle\langle 1-x| + (1-\frac{p}{2})\frac{p}{2}|1\rangle\langle 1| \otimes |x\rangle\langle x| + \frac{p}{2}(1-\frac{p}{2})|1\rangle\langle 1| \otimes |1-x\rangle\langle 1-x| \tag{7}$$

$$\xrightarrow{ABSORB} (1-\frac{p}{2})^2|0\rangle\langle 0| \otimes |x\rangle\langle x| + (\frac{p}{2})^2|0\rangle\langle 0| \otimes |1-x\rangle\langle 1-x| \tag{8}$$

$$\xrightarrow{CNOT1} (1-\frac{p}{2})^2|x\rangle\langle x| \otimes |x\rangle\langle x| + (\frac{p}{2})^2|1-x\rangle\langle 1-x| \otimes |1-x\rangle\langle 1-x| \tag{9}$$

where the operation CNOT1(2) means that the target qubit is 1(2), with the control qubit 2(1), and ABSORB is to terminate the amplitude of both qubits unless the first qubit is in state $|0\rangle$. This process can be carried out by the simple experimental set-up through optical approach in Fig. 1, which uses two different paths (upper and lower) as the first qubit of a single photon, and two orthogonal polarization directions (Horizontal and Vertical) as the ancillary qubit.

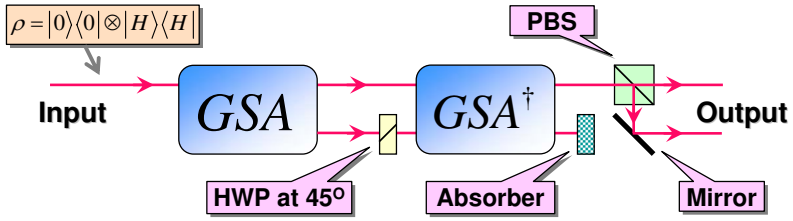


FIG. 1: **Experimental set-up for the simplest reversible quantum algorithm.** The half-wave plate (HWP) at 45° rotates the polarization of photon by 90° and polarizing beam splitter (PBS) transmits photon in $|H\rangle$ and reflects $|V\rangle$.

III. REDUCING THE ERROR RATE TO ZERO

Of courses, we are not to stop at the stage of just reducing the error probability from p to p^2. What we aim for is to cut the output error rate down to an arbitrarily small amount. To achieve it, we *only* need to run $GSA^\dagger$ again and again, *i.e.* repeating the process in Eq.(7)-(9):

$$\begin{aligned}
& (1-\frac{p}{2})^2|x\rangle\langle x| \otimes |x\rangle\langle x| + (\frac{p}{2})^2|1-x\rangle\langle 1-x| \otimes |1-x\rangle\langle 1-x| \\
\xrightarrow{GSA^\dagger} \quad & (1-\frac{p}{2})^3|0\rangle\langle 0| \otimes |x\rangle\langle x| + (\frac{p}{2})^3|0\rangle\langle 0| \otimes |1-x\rangle\langle 1-x| \\
& +(1-\frac{p}{2})^2\frac{p}{2}|1\rangle\langle 1| \otimes |x\rangle\langle x| + (\frac{p}{2})^2(1-\frac{p}{2})|1\rangle\langle 1| \otimes |1-x\rangle\langle 1-x| \qquad (10) \\
\xrightarrow{ABSORB} \quad & (1-\frac{p}{2})^3|0\rangle\langle 0| \otimes |x\rangle\langle x| + (\frac{p}{2})^3|0\rangle\langle 0| \otimes |1-x\rangle\langle 1-x| \qquad (11) \\
\xrightarrow{CNOT1} \quad & (1-\frac{p}{2})^3|x\rangle\langle x| \otimes |x\rangle\langle x| + (\frac{p}{2})^3|1-x\rangle\langle 1-x| \otimes |1-x\rangle\langle 1-x| \qquad (12) \\
\longrightarrow \quad & (1-\frac{p}{2})^k|x\rangle\langle x| \otimes |x\rangle\langle x| + (\frac{p}{2})^k|1-x\rangle\langle 1-x| \otimes |1-x\rangle\langle 1-x| \qquad (13)
\end{aligned}$$

Accordingly, we find that by running the quantum algorithm (or its adjoint) for a total of k times, we can reduce the probability of getting the wrong result to $(\frac{p}{2})^k$. And the output error rate $\frac{(\frac{p}{2})^k}{(\frac{p}{2})^k+(1-\frac{p}{2})^k}$ will become near to zero when $k \to \infty$ and $p < 1$.

IV. GENERAL CASES

Now let's consider a general reversible quantum computer: Suppose this computer has an output register consisted of N qubits that represents the binary result of the computation, which is initialized to $|0_1 0_2 \ldots 0_N\rangle$ at the beginning. We'd also like to assume that the output register is always in computational basis. (For Grover's search algorithm acting on more than two qubits, one could achieve this by simply replacing phase inversion operations with phase rotations of angles smaller than π [11]) Given that without decoherence and noise, the quantum algorithm (QA) will do the unitary transformation to the output register, and the adjoint algorithm will undo this process: (Here we ignore the extra qubits the computer will generally require for its input and programming)

$$|0_1 0_2 \ldots 0_N\rangle \xrightarrow{QA} |y_1 y_2 \ldots y_N\rangle \qquad |y_1 y_2 \ldots y_N\rangle \xrightarrow{QA^\dagger} |0_1 0_2 \ldots 0_N\rangle \qquad (14)$$

When decoherence causes depolarization with probability p, the algorithm will work as:

$$|0_1 0_2 \dots 0_N\rangle\langle 0_1 0_2 \dots 0_N| \xrightarrow{QA} (1-\frac{2^N-1}{2^N}p)|y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| + \frac{p}{2^N} \sum_{i_1 i_2 \dots i_N \neq y_1 y_2 \dots y_N} |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \tag{15}$$

Applying the error-suppression scheme above with the assistance of an ancillary register consisted of N qubits, we obtain:

$$\begin{aligned}
& |0_1 0_2 \dots 0_N\rangle\langle 0_1 0_2 \dots 0_N| \otimes |0_1 0_2 \dots 0_N\rangle\langle 0_1 0_2 \dots 0_N| \\
\xrightarrow{QA,CNOT2} \quad & (1-\frac{2^N-1}{2^N}p)|y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \otimes |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \\
& + \frac{p}{2^N} \sum_{i_1 i_2 \dots i_N \neq y_1 y_2 \dots y_N} |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \otimes |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \qquad (16) \\
\xrightarrow{QA^\dagger,ABSORB} \quad & (1-\frac{2^N-1}{2^N}p)^2 |0_1 0_2 \dots 0_N\rangle\langle 0_1 0_2 \dots 0_N| \otimes |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \\
& + (\frac{p}{2^N})^2 \sum_{i_1 i_2 \dots i_N \neq y_1 y_2 \dots y_N} |0_1 0_2 \dots 0_N\rangle\langle 0_1 0_2 \dots 0_N| \otimes |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \qquad (17) \\
\xrightarrow{CNOT1} \quad & (1-\frac{2^N-1}{2^N}p)^2 |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \otimes |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \\
& + (\frac{p}{2^N})^2 \sum_{i_1 i_2 \dots i_N \neq y_1 y_2 \dots y_N} |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \otimes |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \qquad (18) \\
\xrightarrow{\cdots\cdots} \quad & (1-\frac{2^N-1}{2^N}p)^k |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \otimes |y_1 y_2 \dots y_N\rangle\langle y_1 y_2 \dots y_N| \\
& + (\frac{p}{2^N})^k \sum_{i_1 i_2 \dots i_N \neq y_1 y_2 \dots y_N} |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \otimes |i_1 i_2 \dots i_N\rangle\langle i_1 i_2 \dots i_N| \qquad (19)
\end{aligned}$$

Here the operation CNOT1(2) is a group of CNOT gates working respectively on each target qubits in register 1(2) and ABSORB will absorb all the qubits unless the first register is in the state $|0_1 0_2 \dots 0_N\rangle$.

The final output error rate, after running the algorithm and its adjoint for k times altogether, can be written as

$$\epsilon(N,p,k) = \frac{(2^N-1)(\frac{p}{2^N})^k}{(2^N-1)(\frac{p}{2^N})^k + (1-\frac{2^N-1}{2^N}p)^k} = \frac{1}{1+\frac{[2^N(\frac{1}{p}-1)+1]^k}{2^N-1}} \tag{20}$$

Fig. 2 shows a possible set-up for a two-qubit (two-photon) reversible quantum algorithm. It makes use of optical cycles to conveniently repeat the error-suppression process in Eq.(17)-(18).

V. DISCUSSIONS

To check the effectiveness of our error-suppression scheme, we have plotted the function $\epsilon(N,p,k)$ with the case N=2 in Fig. 3. We can see that as the number of cycles increases, the final output error rate is decreasing to zero exponentially. Even for relatively large p, we only need to run the quantum computer for a few times to effectively eliminate the errors. (*e.g.* For $N=2, k=10$ and $p=0.5, \epsilon \approx 3\times 10^{-7}$).

Moreover, the efficiency of our scheme is not compromised by the scale of the reversible computer. Conversely, we show in Fig. 4 that when the number of qubits N increases, the error rate of our final output actually drops down with exponential speed.

On the other hand, however, it is necessary to mention that with $\epsilon > 0$, we always have a probability $\zeta = 1-(\frac{p}{2^N})^k - (1-\frac{2^N-1}{2^N}p)^k$ of failing to obtain a final output, which means that the photons (for optical set-up) are

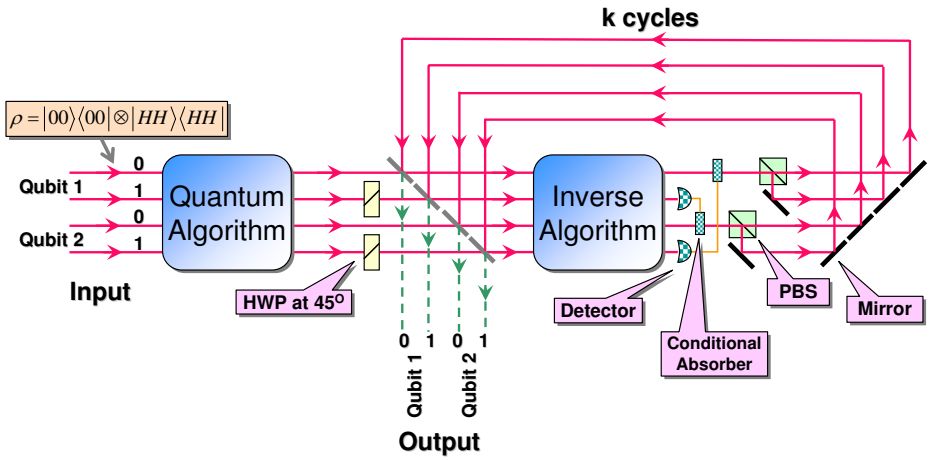


FIG. 2: **Experimental set-up of general error-suppression scheme for reversible quantum computer.** The mirrors in gray color are inserted after the first cycle and removed right before k^{th} cycles so as to get the final output. The conditional absorber takes effect only if the detector controlling it has detected a photon.

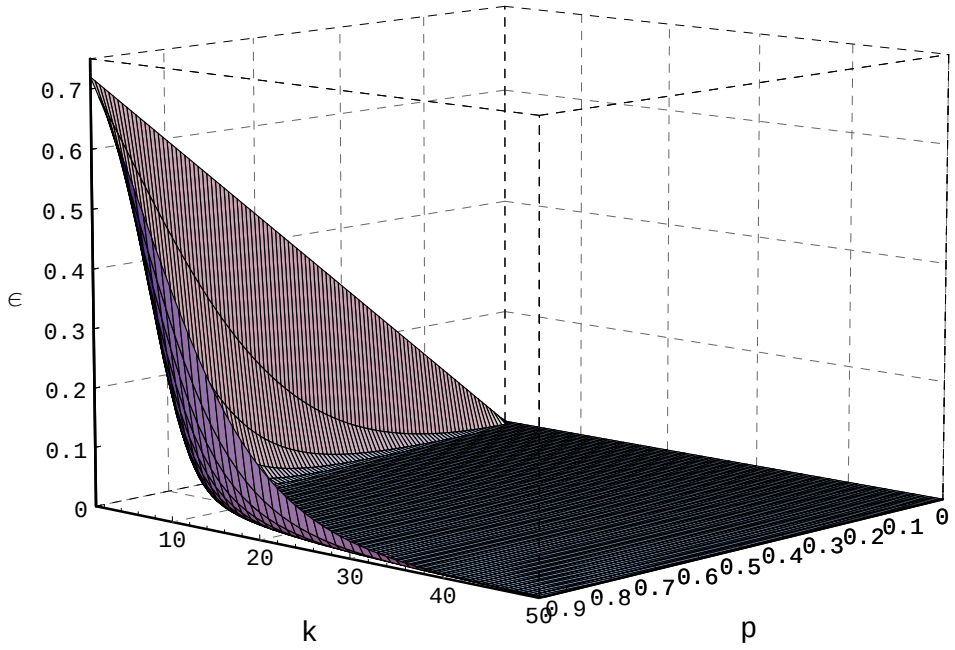


FIG. 3: **Final output error rate function $\epsilon(N, p, k)$ with the case N=2**

absorbed during the process of the scheme. Consequently, we have to run the whole algorithm for a second time or more until we obtain a result. (Note that the majority voting scheme has the same problem). Fortunately, this drawback does not put a high toll on our scheme, since for reasonable values (relatively small) of p and k, *e.g.* $p = 0.1, k = 5, N = 4$, we only need to run the whole algorithm 1.6 times on average while the output error rate is already below 3×10^{-10}. Fig. 5 gives more in detail.

Another interesting point is that for any $p \in (0, 1)$, our error-suppression scheme can give an extremely correct result after enough number of cycles, but if $p = 1$, that is, the quantum computer gives no information in its output (a

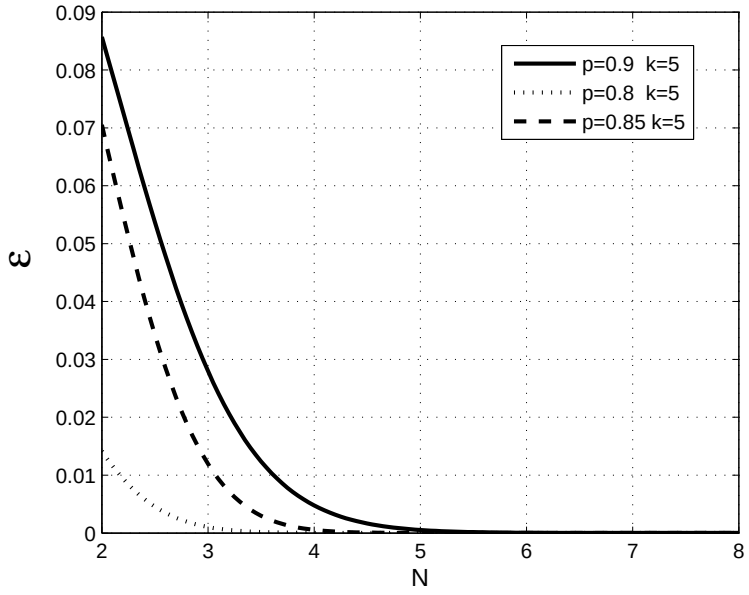


FIG. 4: **Relation between final output error rate ϵ and the number of qubits N**

completely mixed state, $I/2^N$, with the mutual information being zero), then it is natural to deduce that by whatever means, including our scheme, it is simply impossible to generate any useful information in the final output. Our scheme acts as an *information amplifier*, but it cannot produce any information from nil.

The process of suppressing errors step by step gradually is analogous to fault tolerant quantum logic using concatenated codes [12]-[13]. The fault tolerant quantum logic generally consists of three sub processes: encoding, syndrome measurement and recovery, which might be more complicated to be experimentally carried out compared to our scheme. Additionally, as the threshold theorem for fault-tolerant computation holds, each component gate of fault-tolerant logic should fail with a probability below the threshold p_{th}, the typical value of which is approximately 10^{-4} [9]. Our scheme puts no limit on the threshold of p, except for the extreme case of $p = 1$.

There are, nevertheless, a set of limitations for our error suppressing scheme. First, it is best applicable to quantum computers that are reversible. For quantum algorithms that involve measurement to yield final output, such as Shor's factoring algorithm, we need to take further measures to make it run in a totally reversible way (This can be achieved in principle, by using some extra registers and unitary operations); Second, we have to emphasize another premise, that the output of the quantum computer must be in computational basis, *i.e.* it does not allow superpositions (Note that within the quantum computer, there is no such limit.*e.g.* Grover's search algorithm). It is our hope that this error-suppression scheme can be of use to a wider scope of quantum computation processes, as well as stimulating further discourse on related topics.

Acknowledgments

The author would like to thank Professor Richard Jozsa and Professor Robert Griffiths for helpful discussion and suggestions during ICQFT'06 and AQIS'06, and to thank Professor Ying Wu for kind support and encouragement.

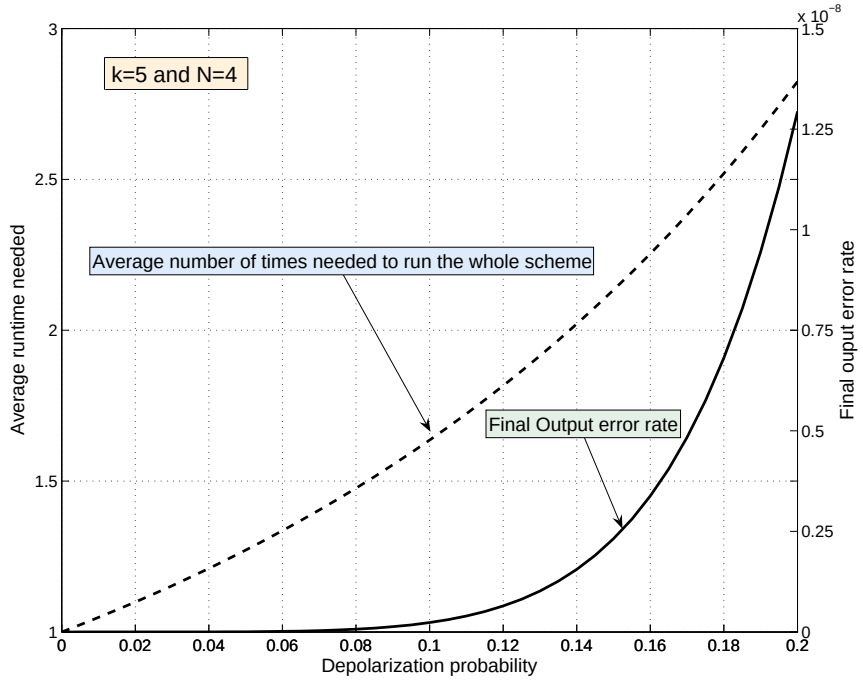


FIG. 5: **Average number of times needed to run the whole scheme / Corresponding final output error rate.**

The project is supported in part by National Natural Science Foundation of China under Grant Nos. 10575040 and 90503010, and National Fundamental Research Programme of China under Grant No 2005CB724508.

[1] P. W. Shor, *Phys. Rev. A* 52, R2493 (1995).

[2] A. M. Steane, *Phys. Rev. Lett.* 77, 793 (1996).

[3] P. Zanardi and M. Rasetti, *Phys. Rev. Lett.* 79, 3306 (1997).

[4] L. Viola and S. Lloyd, *Phys. Rev. A* 58, 2733 (1998).

[5] D. Dhar, L. K. Grover, and S. M. Roy, *Phys. Rev. Lett.* 96, 100405 (2006)

[6] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters and P. G. Kwiat, *Nature*, 439, 949-952 (2006)

[7] G. Mitchison and R. Jozsa, *quant-ph*/0606092

[8] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters and P. G. Kwiat, *quant-ph*/0607101

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* 426-495 (Cambridge Univ. Press, Cambridge, UK, 2000).

[10] L. K. Grover, *Phys. Rev. Lett.* 79, 325-328 (1997).

[11] G. L. Long, *Phys. Rev. A* 64, 022307 (2001)

[12] P. W. Shor, *In Proceedings, 37th Annual Symposium on Fundamentals of Computer Science*, pages 56-65, IEEE Press, Los Alamitos, CA, 1996

[13] J. Preskill, *Proc. R. Soc. London A*, 454, 385-410, 1998