Квантовые вычисления: учебное руководство

С. Л. Браунштейн
(Samuel L. Braunstein)[1]

Представьте себе компьютер, память которого экспоненциально больше, чем можно было бы ожидать, оценивая его явный физический размер; компьютер, который может оперировать одновременно с экспоненциально большим набором входных данных; компьютер, который проводит вычисления в туманном для большинства из нас гильбертовом пространстве.

Тогда Вы думаете о квантовом компьютере.

Чтобы понять, что такое квантовый компьютер, требуется всего лишь несколько относительно простых понятий квантовой механики. Тонкость состоит в том, чтобы научиться работать с этими положениями. Является ли такой компьютер неизбежностью, или построить его будет слишком сложно?

Настоящая работа знакомит с тем, как можно использовать квантовую механику для усовершенствования вычислений. Проблема, которая здесь рассматривается — факторизация большого числа, решение которой является экспоненциально сложной для обычного компьютера. В качестве вступления мы дадим обзор стандартных инструментов вычисления, универсальных гейтов и машин. Эти идеи впервые появились в теории классических компьютеров без диссипации, а затем были применены к квантовым компьютерам.

Схематически описывается модель квантового компьютера, а также некоторые тонкости в его программировании. Алгоритм Шора [1, 2] эффективной факторизации чисел на квантовом компьютере представлен в двух частях: квантовая процедура внутри алгоритма и классический алгоритм, который требует квантовую процедуру. Обсуждается

[1] Encyclopedia of Applied Physics, Update, WILEY-VCH, 1999.
Перевод В. В. Белокурова.

математическая структура в факторизации, которая делает алгоритм Шора возможным. В заключении дается общий взгляд на осуществимость и перспективы квантовых вычислений в ближайшие годы.

Начнем с описания поставленной задачи: факторизации числа N на простые множители (например, число 51688 может быть разложено как $2^3 \times 7 \times 13 \times 71$). Удобный способ оценить, как быстро конкретный алгоритм может решить задачу, состоит в том, чтобы выяснить, как число шагов, требуемых для выполнения алгоритма, растет с увеличением размера входных данных.

Для задачи факторизации входными данными является само число N, которое мы хотим факторизовать; следовательно, длина входа есть $\log N$. (Основание логарифма определяется нашей системой счисления. Так, основание 2 задает длину в двоичной системе; а основание 10 в десятичной). «Приемлемыми» алгоритмами являются те, в которых число шагов растет как некоторый полином небольшой степени от размера входных данных (со степенью, возможно, 2 или 3).

На обычных компьютерах самые лучшие известные алгоритмы факторизации выполняются за $O\left(\exp\left[(64/9)^{1/3}(\ln N)^{1/3}(\ln\ln N)^{2/3}\right]\right)$ шагов [3]. Таким образом, этот алгоритм растет экспоненциально с размером входных данных $\log N$. Например, в 1994 году 129-значное число (известное как $RSA129$ [3′]) было успешно факторизовано с использованием этого алгоритма на приблизительно 1600 рабочих станциях, распределенных по всему миру; полная факторизация заняла восемь месяцев. Используя этот результат для оценки множителя перед приведенной выше экспонентой, получим, что потребуется приблизительно $8 \cdot 10^5$ лет для факторизации теми же компьютерами 250-значного числа; аналогично, для 1000-значного числа потребуется 10^{25} лет (значительно больше, чем возраст вселенной). Трудность факторизации больших чисел является определяющим фактором для криптосистем с открытым ключом, таких, какие используются в банках. Там такие коды считаются надежными в силу сложности факторизации чисел с приблизительно 250 знаками.

Недавно был разработан алгоритм для факторизации чисел на квантовом компьютере, который реализуется за $O\left((\log N)^{2+\varepsilon}\right)$ шагов, где ε — некоторое малое число [1]. Он приблизительно квадратично зависит от размера входных данных, поэтому факторизация 1000-значного числа с помощью такого алгоритма потребует только несколько

миллионов шагов. Это означает, что криптосистемы с открытым ключом, основанные на факторизации, могут быть взломаны.

Чтобы дать представление о том, за счет чего происходит такое экспоненциальное улучшение, рассмотрим элементарный квантовомеханический эксперимент, который показывает, где могут быть заложены такие возможности [5]. Двухщелевой эксперимент дает прототип наблюдаемого квантовомеханического поведения: источник испускает фотоны, электроны или другие частицы, которые достигают пары щелей. Эти частицы претерпевают унитарную эволюцию, и в конце процесса измерения мы видим интерференционную картину, когда обе щели открыты, и которая полностью исчезает, если одна из щелей закрыта. В некотором смысле частицы проходят через обе щели параллельно. Если бы такая унитарная эволюция представляла бы вычисление (или некоторую операцию в рамках вычисления), тогда квантовая система выполняла бы вычисления параллельно. Квантовый параллелизм достигается бесплатно. Полученный на выходе этой системы результат представлял бы собой интерференцию результатов параллельных вычислений.

1. Вычисления на атомных расстояниях

Квантовые компьютеры будут проводить вычисления на атомных расстояниях [5, 6]. На рис. 1 приведены результаты полученных Кейесом в 1988 году [7] оценок изменения с годами числа примесей в основаниях биполярных транзисторов, требуемых для логических операций. Можно считать, что этот график показывает число электронов, необходимых для хранения одного бита информации. Экстраполяция графика означает, что в течение следующих двух десятилетий мы могли бы проводить вычисления на атомных расстояниях.

2. Обратимое вычисление

В чем состоят трудности попыток построить классическую вычислительную машину на таких малых расстояниях? Одна из наиболее крупных проблем программы миниатюризации обычных компьютеров связана с выделением теплоты.

Уже в 1961 году Ландауэр исследовал физические ограничения, налагаемые на вычисления диссипацией [8]. Удивительно, но ему удалось

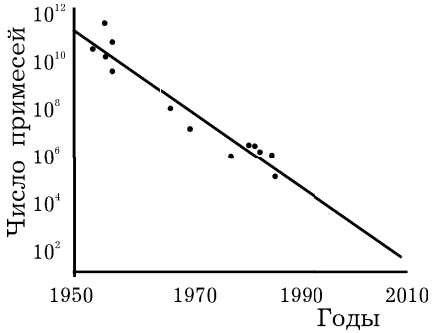

Рис. 1. График из работы [7], показывающий изменение с годами числа неоднородностей в биполярных транзисторах, используемых для выполнения логических операций.

показать, что практически все операции, требуемые для вычисления, могут быть проведены обратимым образом, и поэтому без диссипации теплоты! Первое условие для того, чтобы детерминированное устройство было обратимым, состоит в том, что входные и выходные данные должны единственным образом восстанавливаться друг из друга. Это называется логической обратимостью. Если в дополнение к логической обратимости устройство может реально действовать в обратном направлении по времени, тогда оно называется физически обратимым, и второй закон термодинамики гарантирует, что оно не рассеивает теплоту.

Работа по классическим обратимым вычислениям заложила основы для развития квантовомеханических компьютеров. На квантовом компьютере программы выполняются посредством унитарной эволюции входных данных, которые задаются состоянием системы. Так как унитарные операторы U обратимы, и $U^{-1} = U^{+}$, то на квантовом компьютере вычисления всегда можно обратить.

3. Классические универсальные машины и логические гейты

Рассмотрим теперь основные логические элементы, используемые в вычислении, и объясним, как обычные компьютеры могут быть использованы для любого «разумного» вычисления. Разумное вычисление — такое, которое может быть записано в терминах некоторого

A	B	AND	OR	XOR	NOT B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	1
1	1	1	1	0	0

Таблица 1. Определение действия некоторых элементарных логических гейтов. Каждая строка показывает два входных значения A и B и соответствующие выходные значения для гейтов AND, OR и XOR. Выход для NOT гейта показан только для входа B.

(возможно большого) булевского выражения, и любое булевское выражение может быть построено из фиксированного набора логических гейтов. Такой набор (например, AND (И), OR (ИЛИ) и NOT (НЕ)) называется универсальным. В действительности можно обойтись только двумя гейтами, такими как AND и NOT, или OR и NOT. Действуя альтернативным способом, мы можем заменить некоторые из этих примитивных гейтов другими, такими как исключающее ИЛИ (называется XOR); тогда AND и XOR образуют универсальный набор. Результаты действия этих гейтов приведены в таблице 1. Любое устройство, которое может смонтировать произвольные комбинации логических гейтов из универсального набора, является универсальным компьютером.

Какие из приведенных выше гейтов обратимы? Поскольку AND, OR и XOR — операции, отображающие много данных в одно, то в том виде, как они заданы, они не являются логически обратимыми. Прежде чем мы обсудим, как эти логические гейты могут быть сделаны обратимыми, мы рассмотрим некоторые нестандартные гейты, которые нам для этого потребуются.

4. FANOUT (разворачивание) и ERASE (стирание)

Хотя приведенные выше гейты достаточны для математического аппарата логики, они недостаточны для построения практической вычислительной машины. Для этого требуются еще гейты FANOUT и ERASE (рис. 2).

Вначале рассмотрим гейт FANOUT. Является ли он обратимым? Очевидно, никакая информация не разрушается, поэтому он, по крайней мере логически, обратим. Ландауэр показал, что он может быть так-

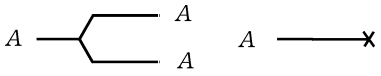

Рис. 2. Два нестандартных гейта, которые, в дополнение к универсальному набору, требуются для построения компьютера: (a) FANOUT гейт, который дублирует входные данные A и (b) ERASE гейт, который уничтожает его входные данные.

же и физически обратим [8]. Опишем простую модель для FANOUT, основанную на схеме Беннетта для обратимого измерения (рис. 3) [9]. Здесь темный шар используется для того, чтобы определить наличие или отсутствие второго (светлого) шара внутри ловушки. Ловушка состоит из набора отражателей и может рассматриваться как регистр с одним битом памяти. Если ловушка занята, то темный шар отражается и покидает ловушку в направлении M (при этом светлый шар продолжает двигаться вдоль своей первоначальной траектории); в противном случае он проходит без помех в направлении N. После того, как темный шар покинет ловушку, направление его движения используется для того, чтобы заселить или нет другую ловушку.

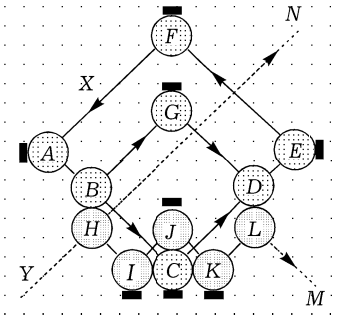

Рис. 3. Обратимое измерение наличия светлого шара в ловушке, состоящей из отражателей (темные прямоугольники) [9]. Темный шар входит в ловушку из Y. В отсутствии светлого шара в ловушке темный шар проследует по пути HN. При наличии светлого шара (в это время начинающего движение в X) темный шар отклонит светлый шар от его первоначальной траектории $ABCDEF$ на траекторию $ABGDEF$, а сам проследует по пути $HIJKLM$.

Давайте теперь рассмотрим операцию ERASE, которая требуется для периодической «чистки» памяти компьютера.

Один тип стирания может быть проведен обратимым образом. Если у нас есть продублированная копия некоторой информации, то мы можем стереть добавочные копии, т. е. провести операцию, обратную той, что совершает FANOUT гейт. Трудность возникает, когда мы хотим стереть имеющуюся последнюю копию, т. е. совершить так называемое примитивное стирание (примитивный ERASE).

Рассмотрим одиночный бит, представленный как пара равновероятных классических состояний некоторой частицы. Для стирания информации о состоянии частицы мы должны необратимым образом сжать фазовое пространство в два раза. Если позволить этому сжатому фазовому пространству адиабатически расширяться при температуре T до его первоначального размера, то можно получить количество работы, равное $k_B T \ln 2$ (где k_B — постоянная Больцмана). Основываясь на простых моделях и более общих аргументах относительно сжатия фазового пространства, Ландауэр сделал вывод о том, что стирание одного бита информации при температуре T требует диссипации по меньшей мере $k_B T \ln 2$ теплоты (результат, известный как принцип Ландауэра) [8].

5. Вычисление без ERASE

К счастью, гейт примитивный ERASE не является абсолютно необходимым в вычислениях. Чтобы понять, почему это так, рассмотрим, что требуется для вычисления произвольных функций, использующего обратимую логику (где примитивный ERASE запрещен). Ландауэр показал, как любая функция $f(a)$ может быть воспроизведена взаимно однозначно с ее аргументом (один к одному) в результате сохранения копии входных данных:

$$f : a \to (a,\ f(a)).$$

Круглые скобки означают здесь упорядоченный набор величин, в данном случае двух. Дополнительные «щели» будут добавлены (или удалены), как потребуется в нашем дальнейшем обсуждении.

Как этот трюк может быть использован для выполнения обратимой логики? Одно решение, известное как Тоффоли-гейт, показано на рис. 4 [8, 10, 11]. Выход этого гейта может быть разложен в различные гейты:

$$B \oplus (A.C) = \begin{cases} A.C, & \text{для} \quad B = 0 & \text{(AND)} \\ A \oplus B, & \text{для} \quad C = 1 & \text{(XOR)} \\ \bar{B}, & \text{для} \quad A = C = 1 & \text{(NOT)} \\ A, & \text{для} \quad B = 0,\ C = 1 & \text{(FANOUT)} \end{cases}$$

где $A.B$ изображает AND-гейт, $A \oplus B$ изображает XOR-гейт и $\bar{A}$ изображает NOT-гейт. Мы видим, что этот гейт является универсальным, поскольку он выполняет AND, XOR, NOT или FANOUT, в зависимости от того, что имеется на входе.

Комбинация многих таких гейтов может затем использоваться для любого вычисления и будет оставаться обратимой.

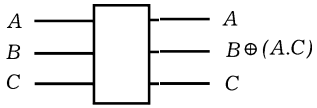

Рис. 4. Универсальный обратимый Тоффоли-гейт с тройным входом и тройным выходом. Этот гейт, очевидно, является обратимым, т. к. повторное его применение воспроизводит первоначальные входные данные.

Как было замечено Ландауэром, эта процедура приводит к прямой проблеме из-за отсутствия примитивного ERASE. Чем больше гейтов мы используем, тем больше «мусорных» битов мы накопим: в каждом гейте мы должны хранить входные биты для сохранения обратимости. Другими словами, компьютер, построенный из логически обратимых гейтов вместо обычных, логически необратимых гейтов, вел бы себя как

$$f\colon a \to (a, j(a), f(a)),$$

с большим числом дополнительных мусорных битов $j(a)$.

Беннетт решил эту проблему, показав, что мусорные биты могут быть обратимым образом стерты на промежуточных шагах с минимальными затратами времени и памяти [12, 13]. Идею решения Беннетта можно истолковать на языке следующей процедуры:

$$f\colon a \to (a,\ j(a),\ f(a)),$$
$$\text{FANOUT}\colon (a,\ j(a),\ f(a)) \to (a,\ j(a),\ f(a),\ f(a)),$$
$$f^{\dagger}\colon (a,\ j(a),\ f(a),\ f(a)) \to (a,\ f(a)),$$

где $f^{\dagger}$ обозначает возвращение к невычисленному f, как противоположное к вычислению f^{-1}. Сначала вычисляется f и при этом получаются

мусорные биты и искомый выходной результат. Затем применяется FANOUT-гейт для дублирования выходного результата. В конце мы возвращаемся к невычисленной исходной функции f, выполняя в обратную сторону операцию ее вычисления. Эта процедура удаляет мусорные биты и первоначальный выходной результат. Однако дубликат остается!

Это завершает наше обсуждение устройства классических обратимых компьютеров. Мы установили, что требование обратимости не является препятствием для логической конструкции вычислительных машин. Прежде, чем переносить эти идеи на квантовые системы, введем некоторые элементарные квантовомеханические понятия.

6. Элементарные квантовые понятия

Одну из простых квантовых систем, в которой имеется два уровня, представляет собой частица со спином $1/2$. Ее базисные состояния, спин вниз $|\downarrow\rangle$ и спин вверх $|\uparrow\rangle$, могут быть переобозначены для представления двоичных нуля и единицы, т. е., соответственно, $|0\rangle$ и $|1\rangle$. Состояние одной такой частицы описывается волновой функцией $\psi = \alpha|0\rangle + \beta|1\rangle$. Квадраты модуля комплексных коэффициентов $|\alpha|^2$ и $|\beta|^2$ задают вероятности найти частицу в соответствующих состояниях. Обобщая это на набор k частиц спина $1/2$, получаем, что теперь имеется 2^k базисных состояний (квантовомеханических векторов, которые образуют гильбертово пространство), соответствующих, скажем, 2^k возможным двоичным строкам длины k. Например, $|25\rangle = |11001\rangle = |\uparrow\uparrow\downarrow\downarrow\uparrow\rangle$ — одно из таких состояний для $k = 5$.

Размерность гильбертова пространства растет экспоненциально с увеличением k.

В самом общем смысле квантовые вычисления используют этот огромный объем, скрытый даже в самых малых системах.

7. Логические гейты для квантовых битов

В этом разделе мы опишем, как можно построить произвольные логические гейты для квантовых битов.

Мы начнем с рассмотрения различных однобитных операций и одной двубитной — XOR-операции. Их комбинации достаточны для по-

строения Тоффоли-гейта для квантовых битов, или, на самом деле, любой унитарной операции на конечном числе битов.

Начнем с одного квантового бита. Если представить состояния $|\downarrow\rangle$ и $|\uparrow\rangle$ (т. е. $|0\rangle$ и $|1\rangle$) как векторы $\binom{1}{0}$ и $\binom{0}{1}$, соответственно, то наиболее общему унитарному преобразованию отвечает матрица 2×2 вида

$$U_\theta \equiv \begin{pmatrix} e^{i(\delta+\sigma+\tau)}\cos(\theta/2) & e^{-i(\delta+\sigma-\tau)}\sin(\theta/2) \\ -e^{i(\delta-\sigma+\tau)}\sin(\theta/2) & e^{i(\delta-\sigma-\tau)}\cos(\theta/2) \end{pmatrix},$$

в которой обычно полагают $\delta = \sigma = \tau = 0$ [14]. Используя этот оператор, мы можем инвертировать биты:

$$U_\pi|0\rangle = -|1\rangle, \quad U_\pi|1\rangle = |0\rangle.$$

Происшедшее изменение знака означает появление фазового множителя, который не влияет на логическую операцию гейтов и может быть опущен, если мы захотим, сразу или на более позднем этапе. Такие однобитные вычисления изображены схематично как квантовая цепь на рис. 5 [14, 15].

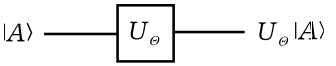

Рис. 5. Схематичная диаграмма квантового цикла для однобитного гейта. Линия изображает один квантовый бит (такой, как задает частица спина $^1/_2$). Первоначально этот бит имеет состояние, описываемое вектором $|A\rangle$; после того, как он пройдет через эту цепь, он выйдет в состоянии $U_\theta|A\rangle$.

Другой важный однобитный гейт — это $U_{-\pi/2}$, который отображает состояние со спином вниз в равную суперпозицию состояний со спином вниз и спином вверх

$$U_{-\pi/2}|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right).$$

Рассмотрим строку, задаваемую k частицами спина $^1/_2$, спины которых первоначально направлены вниз. Если мы применим наш гейт независимо к каждой частице, то получим суперпозицию всех возможных двоичных строк длины k:

$$|0\rangle \to \frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a\rangle,$$

где $q = 2^k$. Наш компьютер находится теперь в суперпозиции экспоненциально большого числа целых чисел a от 0 до $2^k - 1$. Предположим, что мы можем теперь построить унитарную операцию, которая отображает пару двоичных строк $|a; 0\rangle$ в пару $|a; f(a)\rangle$ для некоторой функции $f(a)$. Тогда такой унитарный оператор, действуя на суперпозицию состояний

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a; 0\rangle \to \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a; f(a)\rangle \,,$$

вычисляет функцию $f(a)$ параллельно экспоненциально большое число раз для различных входных значений a.

Чтобы понять, как такие унитарные операторы могут быть построены из нескольких элементарных операторов, рассмотрим XOR-гейт [14, 15].

Записывая двухчастичные базисные состояния как вектора

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

мы можем представить XOR гейт унитарным оператором

$$U_{\mathrm{XOR}} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Здесь первая частица действует как условный гейт для инвертирования состояния второй частицы. Легко проверить, что состояние второй частицы отвечает действию XOR-гейт, заданного в таблице 1. Квантовая цепь для XOR-гейта изображен на рис. 6.

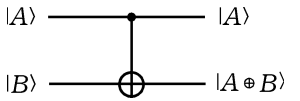

Рис. 6. Диаграмма квантового цикла для XOR-гейта. Низший бит $|B\rangle$ инвертируется всякий раз, когда верхний бит $|A\rangle$ является единицей.

Эта цепь эквивалентна элементарной команде: если ($|A\rangle = 1$), то $|B\rangle \to \mathrm{NOT}|B\rangle$, что можно понимать как пример программы квантового компьютера [22]. Скобки $|\ \rangle$ являются напоминанием того, что

мы имеем дело с квантовыми, а не классическими битами. XOR-гейт позволяет перемещать информацию, как показано на рис. 7.

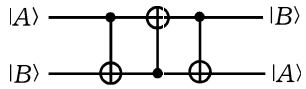

Рис. 7. Цикл для перестановки местами пары битов.

Как построить Тоффоли-гейт? Главная проблема с этим гейтом заключается в том, что он требует три бита на входе и три на выходе. Кажется, что это соответствует квантовому процессу рассеяния, включающему трехчастичные столкновения [16], требующие (возможно) непомерного контроля за частицами [5]. К счастью, Тоффоли-гейт может быть построен только из процессов двухчастичного рассеяния [15, 17, 18, 19, 20]. В частности, здесь мы показываем конструкцию, включающую XOR-гейт и некоторые однобитные гейты U_θ (рис. 8) [14].

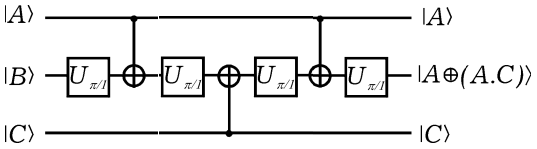

Рис. 8. Тоффоли-гейт, построенный из двубитных XOR-гейтов плюс некоторых однобитных гейтов [5, 14]. Эта цепь вводит некоторые дополнительные знаки в унитарной матрице U_{XOR}, которые могут быть удалены на более позднем этапе.

XOR-гейт не только достаточен для всех логических операций на квантовом компьютере, но он может быть использован для построения произвольных унитарных преобразований на любом конечном наборе битов. Рассматривались многочисленные предложения, как создать такие гейты [2, 6].

8. Модельный квантовый компьютер и квантовые коды

В этом разделе мы опишем простую модель квантового компьютера, основанного на классическом компьютере, обучающем машину управлять набором спинов. У этой модели есть некоторые внутренние ограничения, которые делают разработку алгоритмов на языке высоко-

го уровня довольно сложной. Мы обсудим некоторые из правил для написания такого кода квантового компьютера как языка высокого уровня и приведем пример.

Рассмотрим следующую модель действия квантового компьютера. Несколько тысяч частиц спина $^1/_2$ (или двухуровневых систем) первоначально находятся в некотором определенном состоянии, например, со всеми спинами вниз. Классическая машина берет отдельные спины или пары спинов и скрещивает их (производя элементарную однобитную операцию U_θ или двубитный XOR-гейт); см. рис. 9*a*, *b* и *c*. Эти этапы повторяются на разных парах спинов согласно предписаниям обычной компьютерной программы. Т. к. спины скрещиваются, то мы не должны выделять состояния спинов на промежуточных этапах.

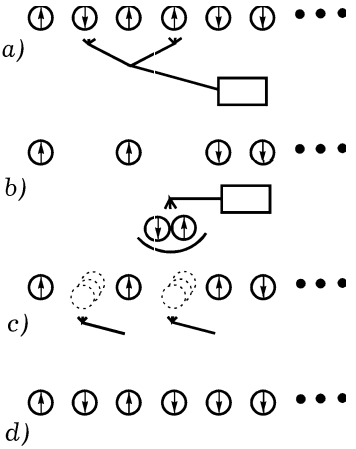

Рис. 9. Модельный квантовый компьютер в представлении Шора [21]. Первоначально все частицы имеют спины вниз. Этап *a*) классическая машина берет отдельные спины или пары спинов и на этапе *b*) производит подобранную однобитную или двубитную операцию; на этапе *c*) «скрещенные» частицы возвращаются на свои первоначальные места. Эти три этапа повторяются много раз в соответствии с командами, заданными обычным классическим компьютером. Когда этот цикл завершен, этап *d*) состоит в измерении состояния частиц (помещая их в некоторую частную двоичную строку); эта двоичная строка является результатом вычисления.

Мы должны сохранить квантовую суперпозицию неповрежденной. Более того, все, что может нарушить ориентацию спинов или прервать

унитарную эволюцию состояний, не должно интерферировать со спинами. Когда этот определенный цикл манипуляций завершен, ориентации спинов измеряются (рис. 9*d*).

Полученный набор измеренных ориентаций является итогом вычисления.

Если задана такая парадигма квантового компьютера, то как может выглядеть его язык высокого уровня (его компьютерный код)? Наиболее серьезная трудность, с которой приходится сталкиваться, состоит в том, что квантовая информация управляется обычным компьютером абсолютно слепым образом — без какого-либо доступа к значениям этой квантовой информации. Это означает, что программа не может использовать сокращения, обусловленные значением квантовой переменной (или регистром, или битом). Например, циклы должны итерироваться точно то же самое число раз независимо от значений квантовых переменных. Аналогично, операции условного перехода через большие куски программы должны быть разбиты на повторяющиеся условия для каждого шага. К тому же каждая команда, выполняемая с квантовыми битами, должна быть логически обратимой. Так, обычное присвоение значения переменной, такое как $|a\rangle = n$, незаконно, а, вместо этого, оно должно выполняться как приращение первоначально равной нулю переменной $|a\rangle = |a\rangle + n$.

Пример такой программы, которая могла бы выполняться на этой машине, мог бы выглядеть следующим образом [22]:

```
    do 10 k = 1, worstdiv
    |a⟩ = |a⟩ − n
    if (|a⟩ = 0)   |q⟩ = |q⟩ + 1
10  continue
    do 20 k = 1, worstdiv
    if (k > |q⟩)   |a⟩ = |a⟩ + n
20  continue
```

Этот фрагмент программы может быть использован для вычисления частного и остатка, помещенных в $|q\rangle$ и $|a\rangle$ соответственно, для деления $|a\rangle$ на n; постоянная `worstdiv` — число раз, которое в худшем случае должен пробегаться цикл. Здесь $|q\rangle$ первоначально равен нулю. Каждая команда здесь — либо обычная компьютерная команда, либо программа, включающая квантовые переменные. Первые являются прямыми командами для внешнего компьютера, в то время как

последние следует интерпретировать как последовательность манипуляций, которые должны совершаться с квантовыми битами. Так, как она написана, эта программа является необратимой (и также не очень эффективной), например, идентификатор 10 не дает описания того, какой путь должен бы быть использован, чтобы добраться до него. Она, однако, легко может быть переписана [22].

9. Квантовый параллелизм. Период последовательности

Теперь наших знаний достаточно, чтобы понять, как квантовый компьютер может выполнять логические операции и вычислять подобно обычному компьютеру. В этом разделе мы опишем алгоритм, использующий квантовый параллелизм, на который мы уже намекали: поиск периода длинной последовательности.

Рассмотрим последовательность

$$f(0),\ f(1),\ \ldots\ ,\ f(q-1),$$

где $q = 2^k$; чтобы найти ее период, мы используем квантовый параллелизм. Начнем с совокупности частиц, спины которых первоначально направлены вниз. Сгруппируем их в два набора (два квантовых регистра, или квантовые переменные):

$$|0;0\rangle = |\downarrow,\downarrow,\ldots\ ;\downarrow,\downarrow,\ldots\rangle\,,$$

при этом первый ряд содержит k битов, и число битов в другом достаточно для наших целей. (В действительности, требуются и другие регистры, но зная решение Беннетта задачи об уборке мусора, пока о них можно умолчать.)

Применив к каждому биту первого регистра однобитную операцию $U_{-\pi/2}$, получим суперпозицию всех возможных двоичных строк длины k в этом регистре:

$$\to \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;0\rangle\,.$$

Следующий шаг состоит в том, что вычисление функции $f(a)$ разбивается на ряд однобитных и двубитных унитарных операций. Последовательность операций конструируется таким образом, чтобы преобразовать состояние $|a;0\rangle$ в состояние $|a;f(a)\rangle$ для любого a. Число битов,

необходимых для второго регистра, должно быть, по крайней мере, достаточным для того, чтобы вместить самый длинный результат $f(a)$ для любого из этих вычислений. Когда эта последовательность операций применяется к нашей экспоненциально большой суперпозиции, а не к одному состоянию на входе, мы получаем

$$\to \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;\, f(a)\rangle \,.$$

Экспоненциально большое число вычислений проводится по существу бесплатно.

Конечный вычислительный шаг, также как и первый, опять является чисто квантовомеханическим. Рассмотрим дискретное «квантовое» преобразование Фурье первого регистра

$$|a\rangle \to \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi i a c/q} |c;\, f(a)\rangle \,.$$

Легко заметить, что оно обратимо в результате обратного преобразования (нетрудно проверить, что оно унитарно). Эффективный способ вычисления этого преобразования с помощью однобитных и двубитных гейтов был описан Копперсмитом (рис. 10) [23, 24, 6].

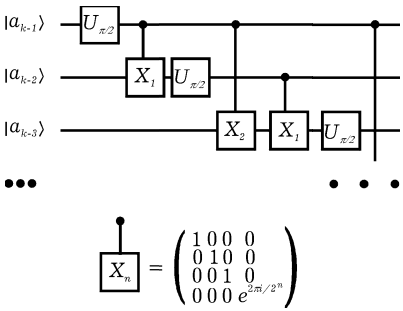

Рис. 10. Цепь для квантового преобразования Фурье переменной $|a_{k-1} \ldots a_1 a_0\rangle$, использующая метод Копперсмита быстрого преобразования Фурье [23, 24, 6]. Двубитные «X_n»-гейты, в свою очередь, могут быть разложены в некоторые однобитные и XOR-гейты [14].

Когда это квантовое преобразование Фурье применяется к нашей суперпозиции, мы получаем

$$\to \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} |c;\, f(a)\rangle \,.$$

Вычисление теперь завершено, и мы восстанавливаем результат на выходе квантового компьютера, измеряя состояние всех спинов в первом регистре (для первых k битов). В действительности, как только преобразование Фурье выполнено, второй регистр может быть даже отброшен [27].

Как будет выглядеть выход?

Предположим, что $f(a)$ имеет период r, т. е. $f(a+r) = f(a)$. Сумма по a приводит к конструктивной интерференции коэффициентов, только когда c/q кратно обратному периоду $1/r$ [25]. При всех других значениях c/q происходит в большей или меньшей степени деструктивная интерференция. Таким образом, распределение вероятности найти первый регистр с различными значениями схематично показано на рис. 11.

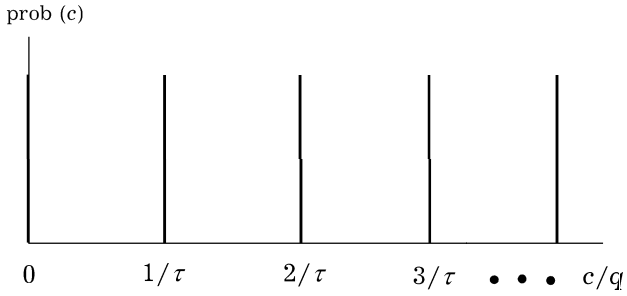

Рис. 11. График зависимости вероятности каждого результата относительно c/q. Конструктивная интерференция дает узкие пики при значениях c/q, кратных обратному периоду последовательности $1/r$.

Один полный цикл работы квантового компьютера дает случайное значение c/q, соответствующее одному из пиков вероятности каждого результата $\mathrm{prob}(c)$. Иначе говоря, мы получаем случайное значение, кратное обратному периоду. Для выделения самого периода нам нужно только повторить это квантовое вычисление, грубо говоря, $\log\log r/k$ раз для того, чтобы получить высокую вероятность, по крайней мере, одному из кратных быть взаимно простым с периодом r — тогда он од-

нозначно определяется [1]. Таким образом, этот алгоритм дает только вероятностный результат. К счастью, мы можем сделать эту вероятность настолько большой, насколько захотим.

Вся описанная выше работа может показаться несколько обескураживающей. Мы столкнулись с большими трудностями при конструировании квантового компьютера для поиска периода последовательности. Преимущество, однако, состоит в том, что последовательность вычисляется параллельно и является экспоненциально длинной — даже для малого количества битов, скажем $k = 140$, в первом регистре квантовый компьютер получал и сохранял больше результатов, чем число частиц во вселенной. Сейчас мы опишем простую структуру, которая существует в математической задаче факторизации и которая позволяет применить в этом случае описанный выше алгоритм квантовых вычислений.

10. Факторизация чисел

Мы хотим факторизовать число N. Достаточно найти хотя бы один множитель, так как затем мы можем свести задачу к более простой. Прежде всего, выберем число x.

С помощью алгоритма Евклида (см. приложение) можно эффективно вычислить общие множители у N и x, и редуцировать задачу. Поэтому предположим, что эти числа взаимно просты. Затем рассмотрим последовательность, образованную функцией $f(a) = x^a \pmod N$. Здесь $a \pmod b$ обозначает остаток от деления a на b. Он может пониматься как час, показываемый на циферблате с b делениями после того, как прошло a часов с тех пор, когда часы были поставлены на нулевой час (час b).

Последовательности $\{x^a\}$ и $\{x^a \pmod N\}$ выглядят, соответственно, как

$$1,\ x,\ \ldots,\ x^{r-1},\ x^r,\ x^{r+1},\ldots$$

$$\underbrace{1,\ x,\ \ldots,\ }_{r\text{ членов}},\ \underbrace{1,\ x,\ \ldots,\ }_{r\text{ членов}},\ \underbrace{1,\ x,\ \ldots,\ }_{r\text{ членов}},$$

Число r — это минимальная степень, для которой $x^r = 1 \pmod N$.

Пристальный взгляд на нижнюю последовательность обнаружит, что она имеет периодическую структуру с периодом r. Используя стандартные алгоритмы, этот период для длинных последовательностей по-

лучить весьма непросто. Однако описанным в предыдущем разделе алгоритмом для квантового компьютера он может быть вычислен эффективным образом. Эта возможность, как мы сейчас продемонстрируем, открывает новый способ найти множители числа N.

Давайте предположим, что описанным выше алгоритмом квантовых вычислений мы получили период r [26]. Если этот период четный, мы можем приступить к нашему алгоритму факторизации. Если нет, мы должны выбрать другое x и начать сначала. Случайным образом выбранное x приведет к подходящему четному периоду r в пятидесяти процентах случаев, поэтому понадобится не так много попыток [1, 2].

Приступим теперь к алгоритму факторизации. Выбрав x так, что последовательность $\{x^a \pmod N\}$ имеет четный период r, перепишем соотношение $x^r = 1 \pmod N$ как разность двух квадратов:

$$\left(x^{r/2}\right)^2 - 1 \equiv 0 \pmod N.$$

Выражая левую сторону как произведение суммы и разности, получим

$$\left(x^{r/2} + 1\right)\left(x^{r/2} - 1\right) \equiv 0 \pmod N.$$

Это просто означает, что произведение двух сомножителей слева кратно числу N, которое мы хотим факторизовать. Таким образом, или один, или другой сомножитель должен иметь общий множитель с N. Окончательный этап алгоритма состоит в вычислении наибольшего общего делителя каждого из этих сомножителей с N (эффективный классический алгоритм описан в приложении). Любой нетривиальный общий делитель является множителем, который мы искали. Таким образом, поиск будет завершен.

В качестве примера рассмотрим число $N = 91$. Выбирая $x = 3$, найдем, что последовательность $3^a \pmod 9)1$ имеет вид

a :	0,	1,	2,	3,	4,	5,	6,	7,	...
3^a :	1,	3,	9,	27,	81,	243,	243,	2187,	...
$3^a \pmod 9)1$:	1,	3,	9,	27,	81,	61,	1,	3,	...

Квантовый компьютер может вычислять период параллельно, однако здесь достаточно взглянуть невооруженным глазом, чтобы заметить, что последовательность имеет период $r = 6$ (так как он четный, можно приступать к алгоритму).

Переписывая соотношение $3^6 \equiv 1 \pmod 9 1$ как сказано выше, получим $28 \times 26 \equiv 0 \pmod 9 1$. Это означает, что либо НОД (28, 91), либо НОД (26, 91) являются нетривиальными делителями 91. В действительности, в этом случае оба члена дают различные множители, соответственно, 7 и 13. Это завершает разложение числа 91 на простые множители: $91 = 7 \times 13$.

11. Перспективы

Каковы перспективы квантовых вычислений? В этом разделе мы обсудим поиски других алгоритмов и опишем наибольшие трудности, возникающие при построении квантовых компьютеров.

В этой статье мы обсудили один алгоритм, приводящий к экспоненциальному ускорению по сравнению с обычными методами — эффективное вычисление периода длинной последовательности. Сегодня это — единственный алгоритм, обнаруживающий такое ускорение. Этот алгоритм был применен к традиционной задаче вычислительной математики — задаче факторизации только благодаря пониманию глубокой структуры, лежащей в основе этой проблемы. Это требование оказывается общим — квантовый параллелизм приведет к экспоненциальному ускорению только в тех задачах, структура которых позволяет избежать необходимости проверки экспоненциально большого числа решений [28, 29, 30, 31]. Таким образом, подход с применением грубой силы к некоторым сложнейшим вычислительным вопросам, известным как NP-полные задачи, не приведет к успеху и с использованием квантового параллелизма. Любой прогресс в решении таких задач требует обнаружения некоторой структуры, лежащей в их основе. Вместо этого, квантовые компьютеры, похоже, будут наиболее полезны для моделирования малых квантовых систем и управления ими [6].

Насколько трудно будет построить квантовый компьютер? Даже в рамках очевидно малых систем атомного размера квантовые вычисления проходят на огромном объеме гильбертова пространства. Квантовое вычисление подразумевает построение траектории от стандартного начального состояния к сложному конечному состоянию. Главная трудность состоит в том, чтобы держаться этой траектории. Покинуть ее — означает исчезнуть в гильбертовом пространстве. Наибольшая проблема — это сверхчувствительность к возмущениям, сдвигающим вычислительную траекторию случайным образом с ее направления. Такие

возмущения происходят от неконтролируемых связей с внешним шумом [32]. Слишком рано предсказывать тяжесть этой проблемы. Хотя кажется, что нет фундаментальных ограничений на то, как хорошо мы можем изолировать квантовую систему. В настоящее время некоторые реализации квантовых компьютеров рассматриваются теоретиками и экспериментаторами в разных странах [17, 18, 33, 34, 35, 36, 37]. Одна многообещающая схема включает ионные ловушки [34, 35] — следующее поколение образцов атомных часов. В течение следующих двух десятилетий обычные компьютеры достигнут атомных размеров; возможно, квантовые компьютеры достигнут этих размеров раньше.

12. Приложение

Здесь мы опишем алгоритм Евклида для нахождения наибольшего общего делителя (НОД) двух чисел $n_0 \geqslant n_1$ [38]. Алгоритм осуществляется как последовательное деление с остатком следующих чисел:

$$\begin{aligned} n_0 &= d_1 \times n_1 + n_2 \\ n_1 &= d_2 \times n_2 + n_3 \\ &\cdots \\ n_{m-2} &= d_{m-1} \times n_{m-1} + n_m \\ n_{m-1} &= d_m \times n_m + 0, \end{aligned}$$

где d_m — частные и $n_{m-1} \geqslant n_m$ на каждом шаге. Последний ненулевой остаток n_m дает ответ, т. е. $\text{НОД}(n_0, n_1) = n_m$. Например, последовательность делений

$$\begin{aligned} 91 &= 3 \times 28 + 7 \\ 28 &= 4 \times 7 + 0, \end{aligned}$$

дает $\text{НОД}(28, 91) = 7$ прямо за два шага. В худшем случае число шагов, требуемых для выполнения алгоритма Евклида, равно $O(\log \log n_1)$.

Литература

[1] P. W. Shor. In Proc. 35th Annual Symposium on the Foundations of Computer Science, edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos, California, 1994), p. 124.

[2] Более детальное описание алгоритма Шора можно найти в работе: A. Ekert and R. Jozsa. *Shor's quantum algorithm for factorizing numbers*, Rev. Mod. Phys. 1995, to appear.

[3] A. M. Odiyzko. *The future of integer factorization*, AT&T Bell Laboratories preprint 1995.

[3′] R. Rivest, A. Shamir and L. Adieman. *On digital signatures and public-key cryptosystems*, MIT Laboratory for Computer Science, preprint MIT/LCS/TR-212,1979.

[4] D. Atkins, M. Graff, A. K. Lenstra and P. C. Leyland. In Advances in Cryptology–ASIACRYPT'94, Eds. J. Pieprzyk and R. Safavi-Naini. *Lecture Notes* in Comp. Sci. 917 (Springer Verlag, Berlin, 1995), p. 263.

[5] D. P. DiVincenzo, presented at Quantum Computation 1994, Villa Gualino, Turin, Italy, October 1994, unpublished.

[6] D. P. DiVincenzo. *Quantum computation.* Science, to appear 1995.

[7] R. W. Keyes. IBM J. Res. Develop. 32, 24 (1988).

[8] The seminal paper in reversible computation: R. Landauer. IBM J. Res. Develop. 3, 183 (1961).

[9] This paper describes the history of reversible computation: C. H. Bennett. IBM J. Res. Develop. 32, 16 (1988).

[10] T. Toffoli. In Automata, Languages and Programming, Eds. J. W. de Bakker and J. van Leeuwen (Springer-Verlag, New York, 1980) p. 632.

[11] E. Fredkin and T. Toffoli. Int. J. Theor. Phys. 21, 219 (1982).

[12] C. H. Bennett. IBM J. Res. Develop. 17, 525 (1973).

[13] C. H. Bennett. SIAM J. Comput. 18, 766 (1989).

[14] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter. *Elementary gates for quantum computation*, submitted to Phys. Rev. A, 1995.

[15] D. P. DiVincenzo. Phys. Rev. A 51, 1015 (1995).

[16] D. Deutsch. Proc. Roy. Soc. Lond. A 425, 73 (1989).

[17] A. Barenco, D. Deutsch and A. Ekert. Phys. Rev. Lett. 74, 4083 (1995).

[18] T. Sleator and H. Weinfurter. Phys. Rev. Lett. 74, 4087 (1995).

[19] D. Deutsch, A. Barenco and A. Ekert. Proc. Roy. Soc. Lond. A 449, 669 (1995).

[20] S. Lloyd. *Almost any quantum logic gate is universal*, Los Alamos National Laboratory preprint.

[21] P. W. Shor, presented at Quantum Computation 1994, Villa Gualino, Turin, Italy, October 1994, unpublished.

[22] D. P. DiVincenzo, private communication and work presented at Quantum Computation 1995, Villa Gualino, Turin, Italy, June 1995, unpublished.

[23] D. Coppersmith. *An approximate Fourier transform useful in quantum factoring*, IBM Research Report RC19642 (1994).

[24] R. Cleve. *A note on computing Fourier transforms by quantum programs*, unpublished.

[25] Необходимо, чтобы дискретное преобразование Фурье обеспечило достаточное разрешение для выделения кратного обратному периоду из соотношения c/q. Это всегда возможно, если число битов k в первом квантовом регистре удовлетворяет неравенству $r^2 \leqslant q = 2^k$.

[26] Так как период r неизвестен, мы требуем $N^2 \leqslant q = 2^k$. Тогда преобразование Фурье обеспечивает на данном этапе вычислений достаточное разрешение [1,2].

[27] I. L. Chuang, R. Laflamme, P. Shor and W. H. Zurek. *Quantum computers, factoring and decoherence*, Report LA-UR-95-241 (1995).

[28] R. Jozsa. Proc. R. Soc. Lond. A 435, 563 (1991).

[29] D. Deutsch and R. Jozsa. Proc. R. Soc. Lond. A 439, 554 (1992).

[30] A. C.-C. Yao. *Quantum circuit complexity*, preprint.

[31] C. H. Bennett, E. Bernstein, G. Brassard and U. V. Vazirani. *Strengths and weaknesses of quantum computing*, preprint.

[32] W. G. Unruh. Phys. Rev. A 51, 992 (1995).

[33] S. Lloyd. Science 261, 1569 (1993).

[34] J. I. Cirac and P. Zoller. Phys. Rev. Lett. 74, 4091 (1995).

[35] T. Pellizzari, S. A. Gardiner, J. I. Cirac and P. Zoller. *Decoherence, continuous observation and quantum computing: a cavity QED model*, preprint.

[36] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi and H. J. Kimble. *Measurement of conditional phase shifts for quantum logic*, Caltech preprint.

[37] R. Hughes, presented at Quantum Computation 1995, Villa Gualino, Turin, Italy, June 1995, unpublished.

[38] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*, Oxford, Clarendon Press, 1979.