Національна академія наук України

ІНСТИТУТ ФІЗИКИ КОНДЕНСОВАНИХ СИСТЕМ

ICMP–03–04U

Т.Крохмальський

Квантові комп'ютери: основи і алгоритми
(короткий огляд)

ЛЬВІВ

УДК: 539.18

PACS: 03.65.Ca, 07.05.Bx, 02.70.Rw, 89.80.+h

Квантові комп'ютери: основи і алгоритми (короткий огляд)

Т.Крохмальський

Анотація. Цей огляд допоможе познайомитись із теоретичними засадами виконання квантових розрахунків за допомогою квантових процесорів, з суттєвим використанням їх квантових властивостей. В ньому розглядаються поняття розраховності функцій, складності алгоритмів, класичної та квантової машин Тюринга, квантові біти, квантові логічні елементи, квантові обчислювальні мережі. Вияснено, також, на яких ефектах грунтується надефективність квантових комп'ютерів в порівнянні з класичними. Приведено приклади ефективних квантових алгоритмів.

Quantum computers: basis and algorithms (short review)

T.Krokhmalskii

Abstract. This review will help to acquaint with the theoretical fundamentals of performing quantum computations using quantum processors which essentially exploit their quantum properties. The concepts of the function computability, the algorithm complexity, the classical and quantum Turing machine, the quantum bits, the quantum gates, the quantum networks are considered here. It is clarified what effects yield supereefficiency of the quantum computers in comparison with the classical ones. Several examples of effective quantum algorithms are given.

1. Вступ

В останньому десятилітті двадцятого віку в рамках квантової механіки почала розвиватися нова область досліджень, що отримала назву **квантова інформатика**. Вона включає такі розділи як квантові обчислення, квантова телепортація, квантова криптографія та інші. Ця нова область послуговується здебільшого теоретичними методами як через дороговизну експериментальних, так і через їх суттєву технічну обмеженість, оскільки тільки в останні роки вдалося збудувати експериментальні пристрої, які дозволили виконати перевірку деяких теоретичних передбачень.

Квантова інформатика почалася з кількох джерел: одним з них є проблема, що виникає із закону Мура [1], згідно з яким число електронних елементів в одиниці об'єму класичного комп'ютера подвоюється щороку, і, згідно оцінок 1995 року, в 2017 році один елемент (скажімо – транзистор) буде припадати на один атом. Тут виникають питання: де лежить класична границя і чи можуть мікроскопічні частинки виконувати функції електронних елементів комп'ютера?

Іншим джерелом є теорія абстрактних обчислювальних машин і проблема розраховності, де виникли питання: чи можна побудувати квантову машину Тюринга і яким буде клас функцій, розраховних нею? [2]

Ще одне джерело лежить в таких засадничих проблемах квантової механіки як парадокс Айнштайна-Подольського-Розена, парадокс шредінгерового кота, нерівності Белла та інше [3].

Тут не розглядаються всі складні проблеми квантової інформатики, а тільки робиться спроба познайомити читача, який знає основи квантової механіки, із засадами виконання квантових обчислень, тобто, з принципом дії квантового комп'ютера (точніше – квантового процесора), проект якого розвивається багатьма авторами на протязі останніх років.

З короткою історією виникнення і розвитку квантової інформатики, а також з її іншими розділами можна познайомитися за препринтом [4].

Детальніший виклад основ квантових обчислень можна знайти в оглядах [1, 5–8], а також в книзі [9].

Я не є спеціалістом з квантової інформатики, але, як автор, мушу визнати, що сприймаю її як поглиблене дослідження основ квантової механіки і дуже скептично ставлюся до проектів її фізичної реалізації.

2. Розраховність

«**Розраховним** називається число, яке можна як завгодно точно наблизити раціональним» [10].

Розрахунок (обчислення) здійснюється за допомогою *алгоритму*, що є послідовністю певних "механічних" дій, правильне виконання яких гарантує вірний результат не залежно від кваліфікації виконавця. Великі за об'ємом обчислення проводять на технічних пристроях, які називають *обчислювальними машинами* чи *комп'ютерами*. Теоретичною основою (математичною моделлю) обчислювальних машин, як технічних пристроїв (фізичних систем), є *абстрактні обчислювальні машини* (АОМ), що є певними математичними системами, в рамках яких описують і аналізують алгоритми. Вивчення АОМ дозволяє встановити також граничні можливості реальних обчислювальних машин. Історично першою АОМ, і до сьогодні найпопулярнішою, є машина Тюринга (1936). Але функціонально ближчою до сучасних комп'ютерів та інтуїтивно зрозумілішою є ідеалізована машина з необмеженими регістрами (МНР), вперше розглянута Шепердсоном і Стерджисом (1963), тому ми коротко зупинимося на її розгляді [11].

МНР складається з безмежного числа регістрів, що позначаються через R_1, R_2, R_3, ..., R_j, ..., кожен з яких (R_j) містить деяке натуральне число (r_j). МНР змінює вміст регістрів у відповідь на деякі *команди*, що відповідають найпростішим операціям над числами. Скінчений список команд утворює *програму*. Для МНР визначені такі чотири базові команди:

1. *Команда занулення* $Z(j)$: $r_j := 0$,
2. *Команда додавання одиниці* $S(j)$: $r_j := r_j + 1$,
3. *Команда переадресування* $T(j,m)$: $r_j := r_m$,
4. *Команда умовного переходу*:

$$J(j,m;q): \begin{cases} \text{якщо } r_j = r_m, & \text{переходь на команду за номером } q, \\ \text{якщо } r_j \neq r_m, & \text{виконуй наступну команду}, \\ \text{якщо } q > p, & \text{stop} \end{cases}$$

(p номер останньої команди програми P).

МНР здійснює обчислення згідно програми P починаючи з деякої початкової конфігурації $\{r_i = a_i\}$ (теоретично i може прямувати до ∞), і закінчує обчислення після скінченого числа кроків в кінцевій конфігурації $\{r_j = b_j\}$. Як правило, розглядаються скінчені початкові $\{a_i\}$ та кінцеві $\{b_j\}$ конфігурації $1 \leq i \leq n$ і $1 \leq j \leq m$. Кажуть,

що обчислення сходяться, якщо машина зупиняється і розходяться, якщо машина не зупиняється.

Далі будемо розглядати функції $\mathbb{Z}^n \xrightarrow{f} \mathbb{Z}$, де $\mathbb{Z}$ — множина невід'ємних цілих чисел. Відомо, що такі функції складають незлічену множину, в той час коли для МНР із чотирьох базисних команд можна збудувати злічену множину програм [11].

Як в такому випадку окреслити множину функцій, розраховних на МНР?

Легко показати, що розраховними є **основні (базові) функції:**

a) нуль-функція $\oslash$ ($\oslash(x) = 0$ для всіх x);

b) функція додавання одиниці $x + 1$;

c) функція проекції U_i^n: $U_i^n(x_1, x_2, \ldots, x_n) = x_i$; $n \geq 1$, $1 \leq i \leq n$.

Суперпозиція функцій. Нехай функції $f(y_1, \ldots, y_k)$ і $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$, де $\mathbf{x} = (x_1, \ldots, x_n)$, — є МНР розраховними функціями. Тоді і суперпозиція цих функцій $h(\mathbf{x}) \stackrel{def}{=} f(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$ також є МНР розраховною функцією.

Рекурсія. Нехай $\mathbf{x} = (x_1, \ldots, x_n)$, а $f(\mathbf{x})$ і $g(\mathbf{x}, y, z)$ є функціями, $(x_j, y, z \in \mathbb{Z})$. Тоді існує єдина функція $h(\mathbf{x}, y)$, що задовільняє рівняння рекурсії

$$h(\mathbf{x}, 0) = f(\mathbf{x}),$$

$$h(\mathbf{x}, y + 1) = g(\mathbf{x}, y, h(\mathbf{x}, y)).$$

Якщо $f(\mathbf{x})$ і $g(\mathbf{x}, y, z)$ — МНР розраховні функції, то і $h(\mathbf{x}, y)$ — також МНР розраховна функція.

Функції, отримані на основі базових функцій, суперпозиції і рекурсії, називаються **примітивно рекурсивними функціями** [11]. До них належать:

a) $x + y$; b) $x \cdot y$; c) x^y;

d) $x \dot{-} 1$ $(0 \dot{-} 1 = 0)$; e) $x \dot{-} y = \begin{cases} x - y, & x \geq y, \\ 0, & x \leq y; \end{cases}$

f) $\mathrm{sg}(x) = \begin{cases} 1, & x = 0, \\ 0, & x \neq 0; \end{cases}$ g) $\overline{\mathrm{sg}}(x) = \begin{cases} 0, & x = 0, \\ 1, & x \neq 0; \end{cases}$

h) $|x \dot{-} y|$; i) $x!$;

j) $\min(x, y)$; k) $\max(x, y)$;

l) $\mathrm{rm}(x, y)$ – залишок від ділення y на x $(\mathrm{rm}(0, y) \stackrel{def}{=} 0)$;

m) $\mathrm{qt}(x, y)$ – частка від ділення y на x $(\mathrm{qt}(0, y) \stackrel{def}{=} 0)$;

n) $\mathrm{div}(x, y) = \begin{cases} 1, & \text{якщо } x|y \ (x \text{ ділить } y), \\ 0, & \text{якщо } x \nmid y \ (x \text{ не ділить } y). \end{cases}$

o) Скінчена сума і скінчений добуток розраховних функцій також є розраховною функцією.

За допомогою *оператора мінімізації* μ клас примітивно рекурсивних функцій розширюється до класу **рекурсивних функцій**. Оператор мінімізації μ визначається так: для довільної розраховної функції $f(\mathbf{x}, y)$ оператор мінімізації $\mu y(f(\mathbf{x}, y) = 0)$ відбирає найменший розв'язок рівняння $f(\mathbf{x}, y) = 0$ відносно y. Іншими словами, функція $y(\mathbf{x})$ є неявно заданою.

Функції, визначені на всій множині $\mathbb{Z}$ називаються **тотальними**, в іншому випадку – **частинними** [11].

Класом $\mathcal{R}$ **частинно-рекурсивних функцій** називається найменший клас частинних функцій, що містить базисні функції $\oslash$, $x+1$, U_i^n і замкнутий відносно операцій суперпозиції, рекурсії і мінімізації.

Або, $\mathcal{R}$ – це клас частинних функцій, кожну з яких можна отримати з основних функцій за допомогою скінченого числа операцій суперпозиції, рекурсії і мінімізації [11].

Отже клас МНР розраховних функцій, означений вище в конструктивній манері, збігається з класом $\mathcal{R}$.

В 30-60-их роках були запропоновані інші АОМ і означені відповідні їм класи розраховних функцій, не обовязково означені конструктивно:

a) Гедель-Ербан-Кліні (1936). Загальнорекурсивні функції, визначені за допомогою числення рекурсивних рівнянь.

b) Черч (1936). λ–означені функції.

c) Гедель-Кліні (1936). μ–рекурсивні функції і частинно-рекурсивні функції.

d) Тюринг (1936). Функції, розраховні машиною Тюринга.

e) Пост (1943). Функції, визначені канонічними дедуктивними системами.

f) Марков (1951). Функції, що задаються нормальними алгорифмами над скінченим алфавітом.

g) Шепердсон-Стерджис (1963). МНР розраховні функції.

Як підсумок цих досліджень був отриманий **основний результат**:

> **Всі ці уточнення поняття розраховності приводять до одного й того ж класу розраховних функцій $\mathcal{C}(\mathcal{T})$ (класу Тюринга).**

Тобто, всі АОМ можуть розраховувати тільки функції з класу $\mathcal{C}(\mathcal{T})$. А реальні ОМ можуть розраховувати функції, що творять підклас в $\mathcal{C}(\mathcal{T})$.

Доведено, що клас $\mathcal{C}(\mathcal{T})$ утворює злічену множину, тобто, всі розраховні функції можна перенумерувати натуральними числами $f_i(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_n)$, $n \to \infty$.

Функція $F(y, \mathbf{x})$ називається **універсальною** функцією якогось класу $\{\varphi_j(\mathbf{x})\}$, якщо всі функції цього класу отримуються з неї покладанням $y = j$, тобто, $\varphi_j(\mathbf{x}) = F(j, \mathbf{x})$.

АОМ, яка розраховує універсальну функцію класу $\mathcal{C}(\mathcal{T})$ називається **універсальною** АОМ. Іншими словами, універсальна АОМ моделює всі інші АОМ.

Зрозуміло, що наведені означення не можуть претендувати на повноту і строгість, в тому сенсі, який прийнятий в цьому розділі математики. Однак, нашою метою є, радше, встановлення того факту, що не всі функції можуть бути алгоритмічно разраховними, і створення хоча б приблизного уявлення про клас розраховних функцій. Для докладнішого ознайомлення з проблемою розраховності функцій окрім [11] можна рекомендувати книгу [2].

3. Складність алгоритмів

Іншою характеристикою алгоритмів є ефективність, яка показує як алгоритм використовує основні ресурси: час, простір, точність. Під простором розуміють число конструктивних елементів ОМ, необхідних для виконання алгоритму.

Алгоритми, які отримують на вході тільки аргументи функції, котру вони розраховують, з послідовністю команд, що залежить тільки від функції і її аргументу, називаються **детермінованими**. Якщо алгоритм на вході крім аргументу функції отримує деяку випадкову (напр. числову) послідовність, яка впливає на хід виконання алгоритму і, як наслідок, – на результат, достовірність якого визначається імовірністю (напр. $P > 2/3$), називаються **імовірнісними** (або **рандомізованими**). До них, зокрема, належать Монте-Карло алгоритми.

Для різних функцій з класу $\mathcal{C}(\mathcal{T})$ використовуються різні алгоритми, які працюють з різною ефективністю, що визначається тільки характером функції.

Нехай L – довжина вхідного слова ОМ. Для числа X це буде $L = \log_k(X)$, де k – основа кодування, прийнята в даній ОМ, найчастіше $k = 2$.

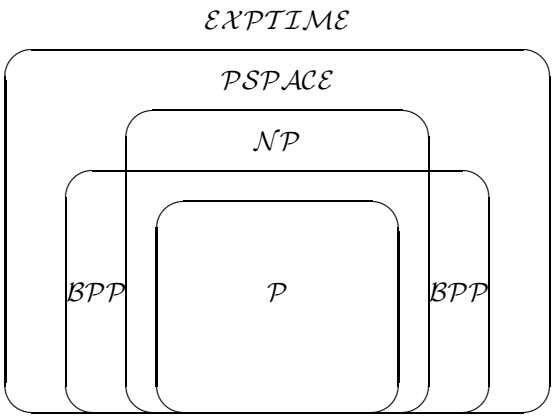


Рис. 1. Зв'язок між різними класами складності алгоритмів [12]

1. Алгоритми, які розраховують функції за час $t \approx cL^m$, належать до класу складності $\mathcal{P}$ (**поліномного**).

2. Алгоритми, які за поліномний час $t = cL^m$ встановлюють чи випадково запропонований розв'язок є істинним ($y = f(\mathbf{x})$), входять до класу $\mathcal{NP}$ (**недетермінованого поліномного**). Відомо, що $\mathcal{P} \subseteq \mathcal{NP}$, існує проблема $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$.

3. Імовірнісні алгоритми належать до класу $\mathcal{BPP}$ (**bounded probability polynomial**), якщо за поліномний час $t = cL^m$ вони дають правильну відповідь $y = f(\mathbf{x})$ з імовірністю $P > 2/3$.

4. $\mathcal{PSPACE}$ алгоритми вимагають поліномного простору, тобто, пам'яті і числа логічних елементів.

5. $\mathcal{EXPTIME}$ алгоритми вимагають експонентного часу виконання $t = c\exp(aL^r)$.

Співвідношення між різними класами складності алгоритмів ілюструє рисунок 1, взятий з [12]. Очевидно, що алгоритми всіх цих класів складності обчислюють функції із класу розраховних функцій $\mathcal{C}(\mathcal{T})$.

Приклади.

1. Додавання, множення і ділення двох чисел довжини L вимагає часу $t \simeq cL^2 \in \mathcal{P}$.

2. Знаходження найменшого спільного дільника (НСД) за алгоритмом Евкліда вимагає часу $t \simeq cL^3 \in \mathcal{P}$.

3. Розпізнавання простоти. Число x просте чи складене? Найкращий детермінований алгоритм розв'язує цю задачу за час $t \simeq cL^{a \log\log L}$. Імовірнісний алгоритм Соловея-Штрассена (1977) вима-

гає часу $t \simeq cL^3 \log \frac{1}{\varepsilon}$, де ε визначає імовірність правильної відповіді ($P = 1 - \varepsilon$) [13].

4. Факторизація. Нехай x – складене, знайти p і q такі, що $x = pq$. Найкращий імовірнісний алгоритм вимагає часу $t \simeq c2^{a\sqrt{L \log L}}$. (Є алгоритми з оцінкою часу $t \simeq c2^{2L^{1/3}(\log L)^{2/3}}$, яка не доведена) [13].

Нехай число x має 130 десяткових знаків, тобто, $L \approx 300$, тоді час факторизації за другим алгоритмом при швидкодії комп'ютера 10^{12} оп/сек рівний 10^{18}сек (42 дні). Якщо ж $L = 600$, то $t \simeq 10^{25}$сек.

Як вперше зауважив Файнман, моделювання квантових систем на класичних комп'ютерах вимагає експонентно великих об'ємів пам'яті і процесора, оскільки навіть система N дворівневих частинок має 2^N станів, а m–рівнева система має $2^{N \log_2 m}$ станів, а якщо поцесор не розпаралелений, то це тягне за собою експонентне зростання часу розрахунку.

Для ілюстрації трудності квантово-механічних задач приведемо оцінку з роботи [15]: «Для квантово-механічного розрахунку молекули метану необхідно провести обчислення за методом сіток в 10^{42} точках. Якщо вважати, що в кожній точці треба виконати всього 10 елементарних операцій, і припустити, що всі обчислення відбуваються при наднизькій температурі (T=3·10^{-3}K), то і при цьому розрахунок молекули метану вимагає витрат енергії, що виробляється на Землі приблизно за століття.»

Отже алгоритми (або задачі) можна поділити на легкі і трудні. Легкі розв'язуються з поліномним використанням ресурсів, а трудні — з експонентним. Треба зауважити, що ці властивості алгоритмів (задач) проявляються при довгих L вхідних словах (даних). Для практичних задач з коротким входом навіть теоретично експонентні алгоритми можуть бути цілком ефективними.

На закінчення цього розділу зауважимо, що в теорії алгоритмів не існує методів теоретичного доведення того, чи для даної задачі можна збудувати ефективний алгоритм. Це вирішується тільки шляхом практичного створення відповідного алгоритму [13].

Матеріал цього розділу, в основному, грунтується на книзі [13].

4. Алгебра Буля і класичні обчислювальні машини

Алфавіти і команди АОМ можна реалізувати на алгебрі Буля, а операції алгебри Буля і її змінні, в свою чергу, можна фізично реалізувати у вигляді механічних чи електронних пристроїв та їх станів, що творить грунт для практичної побудову обчислювальних машин, як технічних пристроїв [16].

Довільне натуральне число можна задати у двійковому зображенні:

$$\mathbf{x} = x_n 2^n + x_{n-1} 2^{n-1} + \ldots + x_1 2 + x_0, \tag{1}$$

яке символічно записується

$$\mathbf{x} = x_n x_{n-1} \cdots x_1 x_0, \tag{2}$$

де $x_j = \{0, 1\}$.

Ірраціональні числа зображаються

$$r = A_n 2^n + A_{n-1} 2^{n-1} + \ldots + A_1 2 + A_0 + \sum_{j=1}^{\infty} a_j 2^{-j}, \tag{3}$$

і записуються

$$r = A_n A_{n-1} \cdots A_1 A_0, a_1 a_2 \cdots a_j \cdots, \tag{4}$$

для раціональних чисел послідовність $\{a_j\}$ – періодична. Надалі обмежимося розглядом тільки натуральних чисел, оскільки всі інші (принаймні з достатньою точністю) можна виразити через них. Вирази (1)–(4) насправді є назвами чисел в двійковому зображенні, а не самими числами. В прийнятому позиційному записі назв чисел при основі k використовуються k відмінних між собою символів. Правила виконання арифметичних дій аль-Хорезмі (алгоритми), сформульовані для назв чисел в десятковому зображенні, чинні для дій з назвами чисел в довільному позиційному зображенні і приводять до назв чисел–результатів цих дій. Розглянемо прості дії з числами (1)–(2).

$$7 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 111, \quad 4 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 100; \tag{5}$$

a) додавання: $7 + 4 = 11 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1011$,

$$\begin{array}{rr} & 111 \\ + & 100 \\ \hline & 1011 \end{array} \tag{6}$$

b) віднімання: $7 - 4 = 3 = 1 \cdot 2^1 + 1 \cdot 2^0 = 11$,

$$\begin{array}{rr} & 111 \\ - & 100 \\ \hline & 011 \end{array} \tag{7}$$

с) множення: $7\times 4 = 28 = 1\cdot 2^4+1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+0\cdot 2^0 = 11100$,

$$\begin{array}{r} 111 \\ \times\quad 100 \\ \hline 000 \\ 000 \\ 111 \\ \hline 11100. \end{array} \qquad (8)$$

З прикладів (6)–(8) видно, що ці елементарні операції легко реалізувати фізично, якщо позиції в записі числа замінити дротиками рахівниці, коліщатками, магнітними кільцями чи транзисторами, а значення $0, 1$ — відповідно їхніми станами. Зміна цих станів при виконанні арифметичних операцій повинна відповідати алгоритмам (6)–(8) та подібним до них.

Хоча еволюція ОМ, як фізичної системи, при виконанні програми і підлягає фізичним законам, однак зміна (еволюція) виділених для зображення чисел станів описується не фізичними законами, а програмою, яку виконує ОМ.

При виконанні операцій (6)–(8) були використані такі правила:

$0+1=1+0=1$, $\quad 0+0=1+1=0$ — додавання за модулем 2;

$$0\cdot 0 = 0\cdot 1 = 1\cdot 0 = 0, \qquad 1\cdot 1 = 1.$$

Якщо числам 0 і 1 поставити у відповідність булеві змінні **O** і **I**, а операціям $+$ та $\cdot$ співставити логічні операції OR та AND, то можна виявити повну відповідність:

$$\begin{array}{lcl} 0\cdot 0 = 0 & \leftrightarrow & \mathbf{O}\ \text{AND}\ \mathbf{O} = \mathbf{O}; \\ 0\cdot 1 = 0 & \leftrightarrow & \mathbf{O}\ \text{AND}\ \mathbf{I} = \mathbf{O}; \\ 1\cdot 0 = 0 & \leftrightarrow & \mathbf{I}\ \text{AND}\ \mathbf{O} = \mathbf{O}; \\ 1\cdot 1 = 1 & \leftrightarrow & \mathbf{I}\ \text{AND}\ \mathbf{I} = \mathbf{I}; \end{array} \qquad (9)$$

$$\begin{array}{lcl} 0+0 = 0 & \leftrightarrow & \mathbf{O}\ \text{OR}\ \mathbf{O} = \mathbf{O}; \\ 0+1 = 1 & \leftrightarrow & \mathbf{O}\ \text{OR}\ \mathbf{I} = \mathbf{I}; \\ 1+0 = 1 & \leftrightarrow & \mathbf{I}\ \text{OR}\ \mathbf{O} = \mathbf{I}; \\ 1+1 = 0 & \leftrightarrow & \mathbf{I}\ \text{OR}\ \mathbf{I} = \mathbf{I}; \end{array} \qquad (10)$$

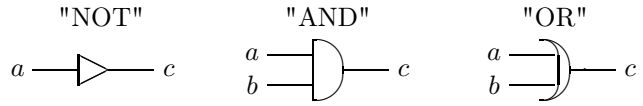
за винятком останнього рядка у (10), тому замість операції OR в комп'ютерах використовується операція "виключного" OR (XOR), яка визначена такими правилами:

$$\begin{array}{lcl} 0+0 = 0 & \leftrightarrow & \mathbf{O}\ \text{XOR}\ \mathbf{O} = \mathbf{O}; \\ 0+1 = 1 & \leftrightarrow & \mathbf{O}\ \text{XOR}\ \mathbf{I} = \mathbf{I}; \\ 1+0 = 1 & \leftrightarrow & \mathbf{I}\ \text{XOR}\ \mathbf{O} = \mathbf{I}; \\ 1+1 = 0 & \leftrightarrow & \mathbf{I}\ \text{XOR}\ \mathbf{I} = \mathbf{O}. \end{array} \qquad (11)$$

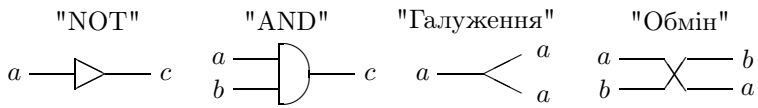
Окрім наведених в алгебрі Буля є ще операція запереченняи NOT з властивостями:

$$\text{NOT}\ \ \mathbf{I} = \mathbf{O}, \quad \text{NOT}\ \ \mathbf{O} = \mathbf{I}. \qquad (12)$$

В електроніці операції алгебри Буля реалізовуються у вигляді технічних пристроїв, що називаються логічними елементами або вентилями, які прийнято позначати так:



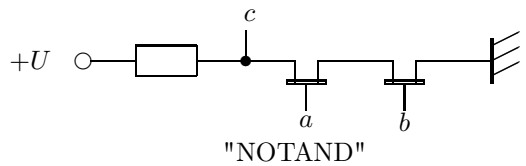
Доведено, що із скінченого числа елементів "NOT", "AND", "OR" чи "XOR" можна збудувати скінчену універсальну обчислювальну машину [14]. Класичну обчислювальну машину можна збудувати і з меншого числа елементів [14]:



.

Тут лінії означають провідники, якими подаються стандартні напруги.

Більше того, виявляється [14], що ОМ можна збудувати з одного логічного елемента "NOTAND", який фізично можна реалізувати таким чином:



Дія цього логічного елемента описується такою таблицею

a	0	0	1	1
b	0	1	0	1
c	1	1	1	0

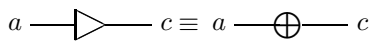
де 1 означає наявність напруги, а 0 – її відсутність. В точці c, з'єднаній через резистор з клемою, на яку подано напругу $+U$, завжди буде напруга за винятком випадку коли в обидві точки a і b одночасно подано напругу, тобто, коли одночасно замкнуто відповідні контакти.

5. Зворотні обчислювальні машини

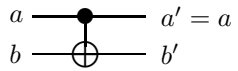
Легко помітити, що одному й тому ж вихідному стану логічних елементів "AND", "OR" чи "XOR" відповідають декілька вхідних, що свідчить про математичну незворотність цих логічних елементів. Класична обчислювальна машина, складена з незворотних елементів, є незворотною обчислювальною машиною, тобто, запущена в зворотному напрямі така ОМ не прийде до початкових даних навіть для детермінованих алгоритмів. З цією незворотністю пов'язувалось розсіювання тепла в процесі роботи обчислювальної машини. Однак Клод Беннет (C.Bennet) на початку 80-х років показав, що із зворотних елементів можна збудувати зворотну ОМ, яка для детермінованих алгоритмів при зворотному виконанні програми приходить від результатів до початкових значень, але розсіювання тепла неминуче залишається. Це відбувається при виконанні операції занулення, яка є складовою частиною процесу обчислення. Тобто, як фізична система, класична ОМ є незворотною, іншими словами, не можна виконати математичний розрахунок не збільшивши ентропії в системі ОМ+оточення [14].

В роботах Toffoli (1981) було запропоновано набір зворотних логічних елементів:

"NOT"

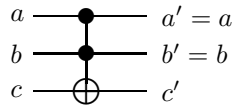


"XOR"



a	0	0	1	1
b	0	1	0	1
a'	0	0	1	1
b'	0	1	1	0

"Toffoli gate"



Останній логічний елемент діє так, що c' =NOT c тільки тоді, коли $a = b = 1$, в інших випадках $c' = c$. Виявляється, що збудувати зворотну ОМ без елемента з трьома лініями ("Toffoli gate") **неможливо**, на противагу до незворотних машин. І навпаки, одного логічного елемента "Toffoli gate" достатньо для побудови зворотної ОМ, з цієї причини він називається **універсальним** логічним елементом.

Зрозуміло, що в математичному сенсі (як реалізація машини Тюринга), зворотна ОМ цілком еквівалентна незворотній ОМ.

6. Квантові розрахунки

Замінюючи логічні елементи в схемі зворотної ОМ унітарними операторами, а її стани — станами дворівневих квантових систем, можна спробувати збудувати відповідну квантову ОМ, що і було зроблено Беньовим (P.Benioff, 1980) і Файнманом (R.Feynman, 1981). Виявилось, що така машина насправді реалізує зворотну **класичну** машину Тюринга на квантових елементах. До того ж тут додаються проблеми приготування початкового стану і зчитування результату, а також — проблеми фізичної реалізації логічних елементів.

Принципово квантову машину Тюринга винайшов Девід Дойч (David Deutsch, 1985) з Оксфорда [17]. Він запропонував зовсім інший підхід до поняття розраховності.

Розглянуте нами раніше поняття розраховності грунтується на тезі Черча-Тюринга:

"Всяка функція, яка інтуїтивно може вважатися розраховною, може бути розрахована на універсальній машині Тюринга." (Цю тезу ми непрямо формулювали вище.)

Д.Дойч запропонував поняття **"функція, яка інтуїтивно може**

вважатися розраховною" замінити поняттям **"функції, які можуть бути в принципі розраховані фізичними системами"**.

Крім того він означує поняття досконалого уподібнення (perfect simulation):

Обчислювальна машина $\mathcal{M}$ здатна досконало уподібнювати фізичну систему S, відносно деякого позначування (labeling) її входів і виходів, якщо існує програма $\pi(S)$ для $\mathcal{M}$, що робить $\mathcal{M}$ обчислювально еквівалентною до S в цьому ж позначуванні.

Тезу Черча-Тюринга Д.Дойч пропонує замінити **принципом** Черча-Тюринга, який у фізичній версії звучить так:

> **Кожна скінчена реалізовна фізична система може бути досконало уподібнена моделлю універсальної обчислювальної машини, яка діє скінченим чином.**

Д.Дойч стверджує, що така ОМ не може бути реалізована в класичній фізиці через неперервність станів класичних фізичних систем. Успіх моделювання класичних систем він пов'язує із успішною дискретизацією моделей таких систем. Далі Д.Дойч стверджує: якщо вдасться побудувати таку ОМ (як фізичну систему), тоді досконале уподібнення певного класу (чи усіх) фізичних систем буде формулюванням властивостей цих систем на мові властивостей тієї фізичної системи, яка реалізує машину. А програма такої машини буде символьним формулюванням цих властивостей. Д.Дойч запропонував теоретичну схему квантової машини Тюринга і показав, що вона має властивості, які не може відтворити жодна класична машина Тюринга, проте:

> **Квантова машина Тюринга $\mathcal{QT}$, (а значить і квантовий процесор) не розширює класу розраховних функцій, тобто $\mathcal{C}(\mathcal{QT}) = \mathcal{C}(\mathcal{T})$.**

А надефективність квантового процесора полягає в тому, що деякі задачі, які є трудними для класичного комп'ютера, тобто вимагають експонентного часу, розв'язуються на квантових процесорах за поліномний час.

7. Квантові біти, квантові логічні елементи і квантові мережі

В цьому розділі розглянемо базові структурні елементи і засади функціонування квантового процесора в сенсі узгодженості з відомими принципами квантової механіки. Фізична реалізація моделей квантового комп'ютера є окремою важливою задачею, якої ми тут не будемо торкатися.

Стани квантового комп'ютера формуються із станів окремих (в певному сенсі не взаємодіючих між собою) двостанових квантових систем, званих квантовими бітами (qubits), які далі будемо скорочено називати квабітами. Для опису станів квабітів в гільбертовому просторі використовують спінорне зображення

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{13}$$

Низка n–квабітів утворює n–квабітовий регістр, стани якого лежать в гільбертовому просторі, що є тензорним добутком n гільбертових просторів окремих квабітів

$$\mathcal{H}^{[n]} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \ldots \otimes \mathcal{H}_n. \tag{14}$$

Відповідні спінорні базиси також є тензорним добутком, наприклад

$$|00\rangle = |0\rangle|0\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \tag{15}$$

$$|01\rangle = |0\rangle|1\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}0\\1\end{bmatrix}\\0\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \tag{16}$$

$$|10\rangle = |1\rangle|0\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\begin{bmatrix}1\\0\end{bmatrix}\\1\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \tag{17}$$

$$|11\rangle = |1\rangle|1\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\begin{bmatrix}0\\1\end{bmatrix}\\1\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}. \tag{18}$$

Зрозуміло, що n–квабітовий регістр може зображати всі натуральні числа від 0 ($|0\ldots0\rangle_n$) до 2^n-1 ($|1\ldots1\rangle_n$) (як і відповідний класичний регістр).

$$\mathbf{x}\leftrightarrow|\mathbf{x}\rangle\equiv|x_{n-1}x_{n-2}\ldots x_2x_1x_0\rangle$$
$$\equiv|x_{n-1}\rangle\otimes|x_{n-2}\rangle\otimes\ldots\otimes|x_2\rangle\otimes|x_1\rangle\otimes|x_0\rangle,\quad x_j=\{0,1\}. \tag{19}$$

Арифметичним операціям з цими числами (за mod 2^n) відповідають зміни станів окремих квабітів квантового регістру, які здійснюються під впливом певних фізичних чинників (як правило – електромагнітних полів). Такі фізичні пристрої називаються квантовими логічними елементами (quantum gates), для скорочення будемо позначати їх КЛЕ. КЛЕ реалізують ширший клас операцій ніж операції алгебри Буля. Дія КЛЕ на квантовий регістр в гільбертовому просторі його станів описується унітарними операторами.

Розгляньмо спочатку найпростіші КЛЕ: одноквабітові перетворення Адамара $\hat{H}$ і зміни фази $\hat{\Phi}$, унітарні оператори яких в спінорному базисі зображаються матрицями 2×2

$$\hat{H}=\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix},\quad\hat{\Phi}(\varphi)=\begin{bmatrix}1&0\\0&\mathrm{e}^{i\varphi}\end{bmatrix}. \tag{20}$$

З означення перетворення Адамара легко встановити його дію

$$\hat{H}|0\rangle=\frac{1}{\sqrt{2}}\left(|0\rangle+|1\rangle\right),\quad\hat{H}|1\rangle=\frac{1}{\sqrt{2}}\left(|0\rangle-|1\rangle\right), \tag{21}$$

або

$$\hat{H}|x\rangle=\frac{1}{\sqrt{2}}\left(|0\rangle+(-1)^x|1\rangle\right)=\frac{1}{\sqrt{2}}\left(|0\rangle+\mathrm{e}^{i\pi x}|1\rangle\right)=\frac{1}{\sqrt{2}}\sum_{y=0}^{1}\mathrm{e}^{i\pi xy}|y\rangle, \tag{22}$$

де $x,y=\{0,1\}$. Розгляньмо тепер одночасну дію операторів Адамара на кожен квабіт квантового регістру, що перебуває в стані $|0\ldots0\rangle_n$

$$\hat{H}^{[n]}|0\ldots0\rangle_n=\frac{1}{2^{\frac{n}{2}}}\left(|0\rangle+|1\rangle\right)\left(|0\rangle+|1\rangle\right)\left(|0\rangle+|1\rangle\right)\ldots\left(|0\rangle+|1\rangle\right)$$
$$=\frac{1}{2^{\frac{n}{2}}}\sum_{x_{n-1}=0}^{1}\sum_{x_{n-2}=0}^{1}\cdots\sum_{x_2=0}^{1}\sum_{x_1=0}^{1}\sum_{x_0=0}^{1}|x_{n-1}x_{n-2}\ldots x_2x_1x_0\rangle$$
$$=\frac{1}{2^{\frac{n}{2}}}\sum_{\mathbf{x}}|\mathbf{x}\rangle. \tag{23}$$

Як бачимо, дія n одноквабітових операторів Адамара на кожен з квабітів n–квабітового регістру перевела останній в стан, що є суперпозицією 2^n станів, кожен з яких визначає натуральне число з області $0...2^n-1$. Тобто, такий стан відображає одночасно всі числа з області $0...2^n-1$ з однаковою амплітудою.

Ця властивість квантових регістрів називається квантовим паралелізмом і є найважливішою з тих, що визначають надефективність квантових процесорів.

Класичні комп'ютери такою властивістю не володіють, оскільки кожен їхній стан визначає тільки одне число.

Легко перевірити, що одночасна дія n операторів Адамара на всі квабіти регістру, який перебуває в стані, що визначає число $\mathbf{y}=y_{n-1}y_{n-2}\ldots y_2y_1y_0$, $\quad(y_j=\{0,1\})$, переводить його в стан, що задає "суперпозицію" чисел

$$\hat{H}^{[n]}|\mathbf{y}\rangle=\frac{1}{2^{\frac{n}{2}}}\sum_{x_{n-1},\ldots,x_0=0}^{1}(-1)^{\mathbf{x}\cdot\mathbf{y}}|\mathbf{x}\rangle, \tag{24}$$

$$\mathbf{y}\cdot\mathbf{x}\equiv(y_{n-1}x_{n-1}+y_{n-2}x_{n-2}+\ldots+y_2x_2+y_1x_1+y_0x_0)\bmod2. \tag{25}$$

На перший погляд це перетворення видається перетворенням Фур'є, але ця відповідність існує тільки, коли $\mathbf{y}=0$, оскільки при перетворенні Фур'є в показник експоненти входить множення $\mathbf{xy}$ mod 2^n, в той час коли множення (25) швидше нагадує скалярний добуток за mod 2.

Дія оператора зміни фази $\hat{\Phi}(\varphi)$ є дещо простішою

$$\hat{\Phi}(\varphi)|x\rangle=\mathrm{e}^{i\varphi x}|x\rangle,\qquad x=\{0,1\}. \tag{26}$$

Операторів Адамара і операторів зміни фаз досить, щоб збудувати довільний унітарний одноквабітовий оператор. Мережа, збудована з квабітів і КЛЕ, називається квантовою мережею (quantum network) і зображається схемами з такими позначеннями: світова лінія квабіта позначається горизонтальною лінією, а дія КЛЕ на цей квабіт — певним знаком на цій лінії. Наприклад, дія оператора Адамара $\hat{H}$ на квабіт $|x\rangle$ зображається графічно так:

$$\hat{H}|x\rangle\leftrightarrow|x\rangle\text{—}\boxed{\mathrm{H}}\text{—}\ \frac{1}{\sqrt{2}}(|0\rangle+(-1)^x|1\rangle)$$
$$\equiv\frac{1}{\sqrt{2}}(|0\rangle+\mathrm{e}^{i\pi x}|1\rangle)=\frac{1}{\sqrt{2}}\sum_y\mathrm{e}^{i\pi xy}|y\rangle, \tag{27}$$

а дія оператора зміни фази —

$$\hat{\Phi}(\varphi)|x\rangle \quad \leftrightarrow \quad |x\rangle \xrightarrow{\hat{\Phi}(\varphi)} \mathrm{e}^{i\varphi x}|x\rangle, \tag{28}$$

Найзагальніший одноквабітовий унітарний оператор залежить від чотирьох параметрів і його матриця має вигляд (A.Barenco, 1995) [18]

$$\hat{U} = \begin{bmatrix} \mathrm{e}^{i(\delta+\frac{\alpha}{2}+\frac{\beta}{2})}\cos(\theta) & \mathrm{e}^{i(\delta+\frac{\alpha}{2}-\frac{\beta}{2})}\sin(\theta) \\ -\mathrm{e}^{i(\delta-\frac{\alpha}{2}+\frac{\beta}{2})}\sin(\theta) & \mathrm{e}^{i(\delta-\frac{\alpha}{2}-\frac{\beta}{2})}\cos(\theta) \end{bmatrix}, \tag{29}$$

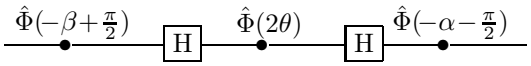
яку можна факторизувати

$$\hat{U} = \mathrm{e}^{i(\delta+\frac{\alpha}{2}+\frac{\beta}{2})} \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-i\alpha} \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{-i\beta} \end{bmatrix}. \tag{30}$$

Після нескладних перетворень отримаємо

$$\hat{U} = \mathrm{e}^{i(\delta+\frac{\alpha}{2}+\frac{\beta}{2}-\theta)}\hat{\Phi}(-\alpha-\frac{\pi}{2})\hat{H}\hat{\Phi}(2\theta)\hat{H}\hat{\Phi}(-\beta+\frac{\pi}{2}). \tag{31}$$

З точністю до загального фазового множника квантова мережа такого перетворення має вигляд

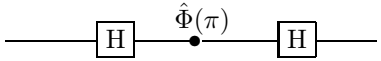


.

Як видно з (29), при $-\alpha = \beta = \delta = \theta = \frac{\pi}{2}$ оператор $\hat{U}$ стає оператором заперечення, тобто реалізує операцію NOT алгебри Буля

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \sigma_x|0\rangle = |1\rangle, \qquad \sigma_x|1\rangle = |0\rangle, \tag{32}$$

а його квантова мережа спрощується до

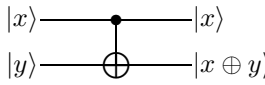


.

Дія двоквабітових КЛЕ описується унітарними операторами в гільбертовому просторі, які в базисі $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ зображаються матрицями 4×4. Зокрема, матриця оператора контрольовного НЕ (XOR або CNOT) має вигляд

$$\mathrm{XOR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{33}$$

а його графічне зображення таке



.

Оператор XOR перекидає (заперечує) стан другого квабіта тільки тоді, коли перший квабіт перебуває в стані $|1\rangle$, тобто, реалізує додавання за модулем 2, як і в класичній зворотній ОМ. Варто звернути увагу, що

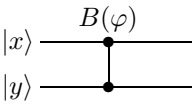
$$\mathrm{XOR}|x\rangle|0\rangle = |x\rangle|x\rangle, \tag{34}$$

і може видатися, що порушується теорема про відсутність клонування, але ніякого клонування немає, бо (34) вірно тільки для відомих станів $x = 0$ або $x = 1$.

Матриця іншого КЛЕ – контрольовного зсуву фази – має схожу загальну структуру

$$B(\varphi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{i\varphi} \end{bmatrix}, \tag{35}$$

а його квантова мережа має вигляд



$\mathrm{e}^{i\varphi xy}|x\rangle|y\rangle$.

Матриці (33) і (35) мають структуру, характерну для контрольовних операторів: на діагоналі матриці всюди стоять одиниці за винятком двох елементів справа внизу, де розташована 2×2 матриця оператора, який змінює стан останнього (згори) квабіта, при умові, що всі інші квабіти перебувають в станах, які відповідають 1

$$\mathrm{CU}_{[n\times n]} = \begin{bmatrix} I_{[n-2\times n-2]} & O \\ O & U_{[2\times 2]} \end{bmatrix}. \tag{36}$$

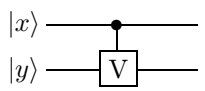
Введімо одноквабітовий оператор

$$\mathrm{V} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \tag{37}$$

де i уявна одиниця, і відповідний йому двоквабітовий контрольовний оператор

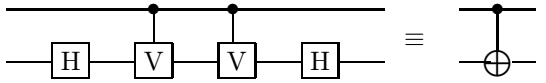
$$\mathrm{CV} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = B(\frac{\pi}{2}), \tag{38}$$

що є частковим випадком оператора $B(\varphi)$ і так зображається графічно



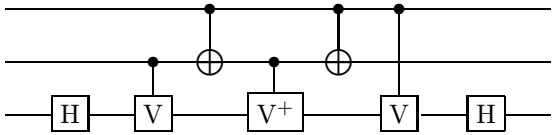
$i^{xy}|x\rangle|y\rangle$.

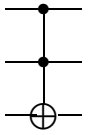
З цього оператора і оператора Адамара можна збудувати квантову мережу, яка реалізовує оператор XOR



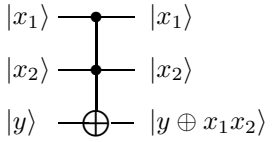
$\equiv$.

В аналогічний спосіб з цих двох операторів можна збудувати КЛЕ — квантовий відповідник "Toffoli gate" з трьома входами і виходами

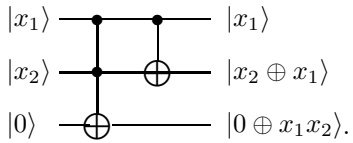


$\equiv$  .

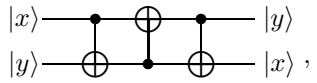
Останній КЛЕ виконує таку ж операцію як в класичному випадку "Toffoli gate"



і разом з КЛЕ "XOR" дозволяє збудувати суматор з переносом у вищий розряд



Часто вживаний при побудові квантових мереж КЛЕ "SWAP", який здійснює обмін вмістом двох квабітів, так пов'язаний з КЛЕ "XOR":



звідки можна легко отримати матрицю оператора SWAP в спінорному базисі

$$\mathrm{SWAP}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}. \tag{39}$$

Оператор іншого часто вживаного КЛЕ $\sqrt{\mathrm{XOR}}$ задається матрицею

$$\sqrt{\mathrm{XOR}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix}. \tag{40}$$

В роботі А.Баренцо та ін. (A.Barenco et al, 1995) [18] доведено, що з двох одноквабітових операторів $\hat{H}$ і $\hat{\Phi}(\varphi)$ та з одного двоквабітового (наприклад з XOR, чи іншого оператора) можна точно побудувати довільний унітарний оператор, що діє в гільбертовому просторі $\mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$ і використати при цьому $4^n n^3$ цих базових операторів (в цій же статті зазначено, що в неопублікованій роботі E.Knill отримано значення $4^n n$). А це означає, що з певним чином дібраних базових КЛЕ можна збудувати квантову мережу, яка реалізує унітарний оператор, котрий виконує відповідні арифметичні розрахунки.

В роботі [19] дано такі означення:

Набір 1-квабітових КЛЕ A_i та 2-квабітових КЛЕ B_j називається **універсальним**, якщо для довільного $n \geq 2$ кожен $n-$квабітовий КЛЕ може бути з довільною точністю наближений $n-$квабітовою мережею збудованою з A_i та B_j.

Набір 1-квабітових КЛЕ A_i та 2-квабітових КЛЕ B_j називається **точно універсальним**, якщо для довільного $n \geq 2$ кожен $n-$квабітовий КЛЕ може бути точно відтворений $n-$квабітовою мережею збудованою з A_i та B_j.

Двоквабітовий КЛЕ називається *примітивним*, якщо результат його дії на тензорний добуток двох векторів можна зобразити тензорним добутком двох інших векторів. Двоквабітовий КЛЕ називається *імпримітивним*, якщо він не є примітивним.

В цій же роботі [19] доведено теорему.

Нехай задано 2-квабітовий КЛЕ V. Тоді наступні твердження є еквівалентними

(i) Набір всіх 1-квабітових КЛЕ A разом з V є універсальним

(ii) Набір всіх 1-квабітових КЛЕ A разом з V є точно універсальним

(iii) V є імпримітивним.

Варто звернути увагу, що в класичній зворотній ОМ універсальним був трибітовий "Toffoli gate", в той час коли у квантовому випадку універсальний набір обмежується двоквабітовими КЛЕ. Успіх побудови реального квантового процесора тісно повязаний з таким вибором універсального набору КЛЕ, котрий вдасться реалізувати фізично у вигляді послідовності переходів у фізичній системі при збереженні її квантової когерентності. Докладніше універсальні набори КЛЕ розглянуто в [5, 18].

Отже квантовий процесор — це квантовий регістр + скінчена мережа квантових логічних елементів (КЛЕ), що діючи на квантовий регістр змінює його стани.

Квантове обчислення — це: ініціалізація початкового стану квантового регістру + унітарна еволюція квантового регістру під керуванням квантової мережі за скінчений проміжок часу від початкового стану квантового регістру ("входу") до його кінцевого стану ("виходу") + відчитання кінцевого стану квантового регістру.

Під унітарною еволюцією тут розуміється не власна еволюція квантового регістру як ізольованої фізичної системи, що описується деяким гамільтоніаном, а унітарна еволюція під дією зовнішніх чинників, які реалізують КЛЕ з квантової мережі, котра відповідає програмі розрахунку визначеної функції.

Залишаються, звичайно, проблеми приготування початкового стану і "відчитання" кінцевого стану. Зрозуміло, що при одному й тому ж початковому стані унітарна еволюція квантової мережі буде приводити до того ж самого кінцевого стану (при збереженні квантової когерентності, тобто, за відсутності впливів оточення на роботу

квантового процесора). Але оскільки "відчитати" кінцевий стан можна тільки здійснюючи вимірювання, тобто, проектування на власні стани деякого оператора, то дістати результати розрахунку можна із статистичного аналізу розподілу результатів вимірюваннь, отриманих при багатократному повторенні роботи квантового процесора. Достовірною прийнято вважати відповідь, знайдену з імовірністю $P \geq 2/3$.

8. Квантові алгоритми

Розрахунок функції $f(x)$ квантовим процесором як відображення $\{0,1\}^{[n]} \xrightarrow{f} \{0,1\}^{[m]}$ описується дією унітарного оператора в гільбертовому просторі $\mathcal{H}^{[n+m]}$, в якому квантовий регістр може перебувати в 2^{n+m} станах. Зазвичай, для аналізу запропонованих квантових алгоритмів, зручно розділити квантовий регістр на два: регістр аргументу $|x\rangle_n$ і регістр значень функції $|y\rangle_m$. Тоді розрахунок функції можна зобразити

$$|x\rangle_n|y\rangle_m \stackrel{f(x)}{\rightarrow} |x\rangle_n|y \oplus f(x) \bmod 2^m\rangle_m \tag{41}$$

або

$$|\mathbf{x}\rangle|0\rangle \stackrel{f(\mathbf{x})}{\rightarrow} |\mathbf{x}\rangle|f(\mathbf{x})\rangle. \tag{42}$$

На сьогодні опубліковано значну кількість проектів квантових мереж для ефективного (поліномного) розрахунку тих чи інших розраховних функцій. Квантова надефективність може проявлятися не обов'язково в розрахунку самих функцій, а в дослідженні деяких їхніх властивостей. Тому, здебільшого, далі будемо вважати, що ми володіємо квантовою (під)мережею – "оракулом" (oracle), яка ефективно розраховує деяку функцію $f(\mathbf{x})$. Задача полягає в тому, щоби добудувати до неї нову квантову мережу, яка б дозволила ефективно (в поліномний час чи з поліномним числом КЛЕ, що те саме) провести дослідження властивостей функції $f(\mathbf{x})$, котре на класичному комп'ютері обходиться експонентними затратами.

Розгляньмо далі декілька прикладів найпопулярніших квантових алгоритмів, які ілюструють надефективність квантових процесорів.

8.1. Квантове перетворення Фур'є [5]

Згадаймо, що число $\mathbf{x}$ в двійковому зображенні за модулем 2^n має вигляд

$$\mathbf{x} = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_2 2^2 + x_1 2^1 + x_0 2^0 = \sum_{j=0}^{n-1} x_j 2^j, \quad (43)$$

тоді добуток двох чисел (за модулем 2^n) можна записати

$$\mathbf{xy} = \sum_{j=0}^{n-1} x_j 2^j y_0 + 2\sum_{j=0}^{n-2} x_j 2^j y_1 + 2^2 \sum_{j=0}^{n-3} x_j 2^j y_2 + \ldots$$
$$+2^{n-3}\big(x_2 2^2 + x_1 2 + x_0\big)\, y_{n-3} + 2^{n-2}(x_1 2 + x_0)\, y_{n-2} + 2^{n-1} x_0 y_{n-1}. \quad (44)$$

Не складає труднощів і отримання частки від ділення останнього виразу на 2^n (знову ж таки, за модулем 2^n)

$$\begin{aligned}\frac{\mathbf{xy}}{2^n} &= \Big(\frac{x_{n-1}}{2} + \frac{x_{n-2}}{2^2} + \ldots + \frac{x_2}{2^{n-2}} + \frac{x_1}{2^{n-1}} + \frac{x_0}{2^n}\Big)\, y_0 \\ &+ \Big(\frac{x_{n-2}}{2} + \frac{x_{n-3}}{2^2} + \ldots + \frac{x_2}{2^{n-3}} + \frac{x_1}{2^{n-2}} + \frac{x_0}{2^{n-1}}\Big)\, y_1 \\ &+ \Big(\frac{x_{n-3}}{2} + \frac{x_{n-4}}{2^2} + \ldots + \frac{x_2}{2^{n-4}} + \frac{x_1}{2^{n-3}} + \frac{x_0}{2^{n-2}}\Big)\, y_2 + \ldots \\ &+ \Big(\frac{x_2}{2} + \frac{x_1}{2^2} + \frac{x_0}{2^3}\Big)\, y_{n-3} + \Big(\frac{x_1}{2} + \frac{x_0}{2^2}\Big)\, y_{n-2} + \frac{x_0}{2^2} y_{n-1}.\end{aligned} \quad (45)$$

На Рис. 2 зображено квантову мережу для перетворення Фур'є чотирирозрядного двійкового числа, далі стане зрозумілим, як розбудувати цю мережу для n-розрядного числа. Перш ніж ознайомитися з дією квантової мережі для виконання квантового перетворення Фур'є пригадаємо як діють її окремі елементи: оператор Адамара діє на окремий квабіт

$$\hat{H}|x_j\rangle = |0\rangle + \mathrm{e}^{i\pi x_j}|1\rangle, \quad (46)$$

а оператор зміни фази в цій мережі — на два квабіти

$$\hat{B}(\varphi)|x_k\rangle|x_j\rangle = \mathrm{e}^{i\varphi x_k x_j}|x_k\rangle|x_j\rangle. \quad (47)$$

(Тут і далі в цьому розділі множники $1/\sqrt{2}$, що виникають при перетвореннях Адамара, будемо упускати.) Позначивши індексами біля

значків операторів КЛЕ номери квабітів, на які вони діють, еволюцію квантової мережі можна записати так

$$\begin{aligned}|\mathbf{x}\rangle \equiv |x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle &\xrightarrow{H_3} \big(|0\rangle + \mathrm{e}^{i\pi x_3}|1\rangle\big)\,|x_2\rangle|x_1\rangle|x_0\rangle \xrightarrow{B_{32}(\frac{\pi}{2})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}\right)}|1\rangle\Big)\,|x_2\rangle|x_1\rangle|x_0\rangle \xrightarrow{H_2} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}\right)}|1\rangle\Big)\,\big(|0\rangle + \mathrm{e}^{i\pi x_2}|1\rangle\big)\,|x_1\rangle|x_0\rangle \xrightarrow{B_{31}(\frac{\pi}{4})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}\right)}|1\rangle\Big)\,\big(|0\rangle + \mathrm{e}^{i\pi x_2}|1\rangle\big)\,|x_1\rangle|x_0\rangle \xrightarrow{B_{21}(\frac{\pi}{2})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}\right)}|1\rangle\Big)\,|x_1\rangle|x_0\rangle \xrightarrow{H_1} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}\right)}|1\rangle\Big) \\ &\big(|0\rangle + \mathrm{e}^{i\pi x_1}|1\rangle\big)\,|x_0\rangle \xrightarrow{B_{30}(\frac{\pi}{8})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}\right)}|1\rangle\Big) \\ &\big(|0\rangle + \mathrm{e}^{i\pi x_1}|1\rangle\big)\,|x_0\rangle \xrightarrow{B_{20}(\frac{\pi}{4})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}+\frac{x_0}{4}\right)}|1\rangle\Big) \\ &\big(|0\rangle + \mathrm{e}^{i\pi x_1}|1\rangle\big)\,|x_0\rangle \xrightarrow{B_{10}(\frac{\pi}{2})} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}+\frac{x_0}{4}\right)}|1\rangle\Big) \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_1+\frac{x_0}{2}\right)}|1\rangle\Big)\,|x_0\rangle \xrightarrow{H_0} \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8}\right)}|1\rangle\Big)\,\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}+\frac{x_0}{4}\right)}|1\rangle\Big) \\ &\Big(|0\rangle + \mathrm{e}^{i\pi\left(x_1+\frac{x_0}{2}\right)}|1\rangle\Big)\,\big(|0\rangle + \mathrm{e}^{i\pi x_0}|1\rangle\big) \\ &= \sum_{y_0} \mathrm{e}^{i\pi\left(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8}\right)y_0}|y_0\rangle \sum_{y_1} \mathrm{e}^{i\pi\left(x_2+\frac{x_1}{2}+\frac{x_0}{4}\right)y_1}|y_1\rangle \\ &\sum_{y_2} \mathrm{e}^{i\pi\left(x_1+\frac{x_0}{2}\right)y_2}|y_2\rangle \sum_{y_3} \mathrm{e}^{i\pi x_0 y_3}|y_3\rangle \\ &= \sum_{\mathbf{y}} \mathrm{e}^{i2\pi\frac{\mathbf{xy}}{2^4}}|\mathbf{y}\rangle.\end{aligned} \quad (48)$$

Варто відзначити, що порядок позицій розрядів числа-результату є зворотним в порівнянні з числом-аргументом.

З Рис. 2 і з перетворень (48) легко зрозуміти як розширити мережу для квантового перетворення Фур'є n-розрядного числа і вста-

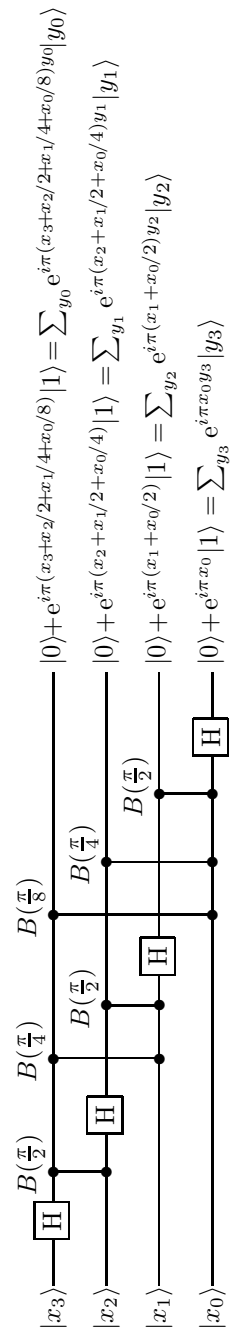


Рис. 2. Квантова мережа перетворення Фур'є чотириквабітового регістра

новити, що така мережа буде мати $n(n+1)/2$ КЛЕ.

Допустимо, що ми перевели квантовий регістр в суперпозиційний стан з амплітудами, які є значеннями деякої функції $f(\mathbf{x})$, а потім здійснили квантове перетворення Фур'є регістру аргументу

$$\sum_{\mathbf{x}} f(\mathbf{x})|\mathbf{x}\rangle \stackrel{QFT}{\longrightarrow} \sum_{\mathbf{x}} f(\mathbf{x}) \frac{1}{2^{n/2}} \sum_{\mathbf{y}} \exp(i2\pi \frac{\mathbf{xy}}{2^n})|\mathbf{y}\rangle = \sum_{\mathbf{y}} \tilde{f}(\mathbf{y})|\mathbf{y}\rangle, \quad (49)$$

тоді нові амплітуди станів регістра будуть фур'є-трансформантами функції $f(\mathbf{x})$

$$\tilde{f}(\mathbf{y}) = \frac{1}{2^{n/2}} \sum_{\mathbf{x}} f(\mathbf{x}) \exp(i2\pi \frac{\mathbf{xy}}{2^n}). \quad (50)$$

Звичайно, відчитання цих амплітуд є непростою проблемою, але у випадках, коли перетворення Фур'є є проміжною операцією в розрахунках, переваги квантового алгоритму є безперечними. Нагадаємо, що класичний алгоритм виконує це перетворення за $s = 2^{2L}$ кроків ($L = \log_2 N$). Для випадку, коли $N = 2^n$ запропоновано класичний алгоритм швидкого перетворення Фур'є, який виконується за $s = 2^n n$ кроків ($L = n$), тобто навіть цей алгоритм є експонентно трудним, а, як ми вказували раніше, для квантового алгоритму $s = n(n+1)/2$.

8.2. Задача Дойча-Джозси

Задача Дойча була історично першою, на якій було показано, що експонентно трудні для класичних комп'ютерів задачі, на квантових процесорах можуть бути розв'язані з поліномними затратами. Задача полягає в тому, щоб встановити до якого класу належить функція — "constant" чи "balanced". В першому варіанті вона була сформульована Дойчем для бінарного аргументу. Коли

$$f(0) = f(1) = 0 \;\text{ або }\; f(0) = f(1) = 1$$

функція є сталою ("constant"), а коли

$$f(0) = 0, f(1) = 1 \;\text{ або }\; f(0) = 1, f(1) = 0$$

функція є змінною ("balanced"). Пізніше Дойч і Джозса (R.Jozsa) узагальнили цю задачу на довільний аргумент $\mathbf{x}$ з області $0 \ldots N$, при тому, що сама функція приймає два значення $\{0, 1\}$. В цьому випадку функція вважається "constant", якщо для всіх значень аргументу вона приймає значення 0 або 1, і "balanced", якщо для однієї

половини значень аргументу вона приймає значення 0, а для іншої половини — значення 1. В класичному випадку для встановлення до якого класу належить така функція треба виконати 2^n ($n = \log_2 N$) обчислень її значень. За допомогою квантового процесора цю перевірку можна виконати при одноразовому розрахунку функції $f(\mathbf{x})$. Для цього будується регістр аргументу з n квабітів і регістр значень функції з одного квабіта, які приводяться в початковий стан

$$|0\ldots0\rangle_n|1\rangle, \tag{51}$$

після чого на кожен квабіт діють оператором Адамара, що приводить до такої суперпозиції

$$\sum_{\mathbf{x}}|\mathbf{x}\rangle(|0\rangle - |1\rangle). \tag{52}$$

Потім розраховують функцію $f(\mathbf{x})$

$$\sum_{\mathbf{x}}|\mathbf{x}\rangle(|f(\mathbf{x})\rangle - |1 + f(\mathbf{x})\rangle) = \sum_{\mathbf{x}}(-1)^{f(\mathbf{x})}|\mathbf{x}\rangle(|0\rangle - |1\rangle), \tag{53}$$

а тоді знову на кожен квабіт аргументу діють оператором Адамара (див. (24) і (25)) і отримують суперпозицію

$$\sum_{\mathbf{x},\,\mathbf{y}}(-1)^{f(\mathbf{x})+\mathbf{x}\cdot\mathbf{y}}|\mathbf{y}\rangle(|0\rangle - |1\rangle). \tag{54}$$

Нарешті проводять вимірювання значення числа $\mathbf{y}$, записаного в регістрі аргументу. Якщо $\mathbf{y} = 0$, то $f(\mathbf{x})$ є "constant", оскільки інакше амплітуда стану $|0\ldots0\rangle$ була б рівною нулю

$$\sum_{\mathbf{x}}(-1)^{f(\mathbf{x})} = 0, \tag{55}$$

якщо ж $\mathbf{y} \neq 0$, то $f(\mathbf{x})$ є "balanced", інакше амплітуда стану $|\mathbf{y}\rangle$ була б рівною нулю

$$\sum_{\mathbf{x}}(-1)^{\mathbf{x}\cdot\mathbf{y}} = 0. \tag{56}$$

Повторюючи попередні міркування, легко встановити, що для функції $f(\mathbf{x}) = \mathbf{a}\cdot\mathbf{x}$, вимірювання дає $\mathbf{y} = \mathbf{a}$.

8.3. Визначення періоду функції (задача D.Simona)

Нехай задана функція $f\colon \{0,1\}^n \xrightarrow{f} \{0,1\}^n$, про яку відомо, що вона має період $\mathbf{r}$, $f(\mathbf{x}+\mathbf{r}) = f(\mathbf{x})$, такий що $2^{n-1} < \mathbf{r} < 2^n - 1$ і необхідно знайти його значення. Побудуймо два регістри по n квабітів кожен, подіймо на кожен квабіт регістру аргументу оператором Адамара, після чого обчислімо функцію f:

$$|0\ldots0\rangle_n|0\ldots0\rangle_n \xrightarrow{H^{[n]}} \sum_{\mathbf{x}}|\mathbf{x}\rangle|0\ldots0\rangle_n \xrightarrow{f} \sum_{\mathbf{x}}|\mathbf{x}\rangle|f(\mathbf{x})\rangle, \tag{57}$$

далі виміряймо число, записане в регістрі значень функції, що дасть деяке $f(\mathbf{k})$, а обидва регістри перейдуть в стан

$$(|\mathbf{k}\rangle + |\mathbf{k}+\mathbf{r}\rangle)|f(\mathbf{k})\rangle. \tag{58}$$

Застосуймо $H^{[n]}$ до регістру аргументу

$$\begin{aligned}&\sum_{\mathbf{y}}\left[(-1)^{\mathbf{k}\cdot\mathbf{y}} + (-1)^{(\mathbf{k}+\mathbf{r})\cdot\mathbf{y}}\right]|\mathbf{y}\rangle|f(\mathbf{k})\rangle\\ &= \sum_{\mathbf{y}}(-1)^{\mathbf{k}\cdot\mathbf{y}}\left[1 + (-1)^{\mathbf{r}\cdot\mathbf{y}}\right]|\mathbf{y}\rangle|f(\mathbf{k})\rangle.\end{aligned} \tag{59}$$

Вимірювання числа $\mathbf{y}$, записаного в регістрі аргументу, дасть

$$\mathbf{y}\cdot\mathbf{r} = 0.$$

Повторивши вимірювання n разів отримаємо систему n лінійних рівнянь для $\mathbf{r}$, яку розв'яжемо на класичному комп'ютері.

8.4. Факторизація чисел (P.Shor, 1994)

Алгоритм Шора для факторизації чисел за допомогою квантового процесора був тим камінцем, що викликав лавину публікацій, присвячених розвитку проекту квантового процесора. Такий відгук повязаний з тим, що факторизація великих натуральних чисел на прості співмножники є головним трудним етапом в несанкціонованому дешифруванні текстів, зашифрованих за методом RSA (див. Додаток А), який є стандартом шифрування конфіденційної інформації в США і найпоширенішим методом шифрування інформації, що передається по всесвітній інформаційній мережі.

Нехай N – складене число ($N \leq 2^n$). Його дільники є серед найбільших спільних дільників чисел N і $a^{r/2} \pm 1$, де r період послідовності $a^j \bmod N$, $j = 0,\ldots,$ ($a^r \bmod N = 1$) (див. Додаток Б). Трудною задачею тут є визначення періоду r, бо якщо період відомий, то

знаходження найбільших спільних дільників чисел N і $a^{r/2} \pm 1$ за допомогою алгоритму Евкліда є задачею поліномної трудності для класичного процесора.

Збудуймо два регістри: регістр значень функції довжини n ($N \leq 2^n$) і регістр аргументу довжини m ($M = 2^m$), такої що $m \gg n$. Занулімо обидва регістри, потім перетвореннями Адамара переведімо регістр аргументу в суперпозиційний стан, після чого виберім число $a < N$ взаємно просте з N і розрахуймо функцію $a^{\mathbf{x}} \bmod N$

$$|0\ldots0\rangle_m|0\ldots0\rangle_n \xrightarrow{H^{[m]}} \sum_{\mathbf{x}} |\mathbf{x}\rangle|0\rangle_n \xrightarrow{a^{\mathbf{x}} \bmod N} \sum_{\mathbf{x}} |\mathbf{x}\rangle|a^{\mathbf{x}} \bmod N\rangle. \qquad (60)$$

Виміряймо значення функції

$$(|\mathbf{k}\rangle + |\mathbf{k}+\mathbf{r}\rangle + |\mathbf{k}+2\mathbf{r}\rangle + |\mathbf{k}+3\mathbf{r}\rangle + \cdots + |\mathbf{k}+l\mathbf{r}\rangle)|a^{\mathbf{k}}\rangle = \sum_{j=0}^{l} |\mathbf{k}+j\mathbf{r}\rangle|a^{\mathbf{k}}\rangle \qquad (61)$$

і здійснімо перетворення Фур'є регістру аргументу

$$\sum_{\mathbf{y}}\sum_{j=0}^{l} \exp\left[\frac{2\pi i(j\mathbf{r}+\mathbf{k})\mathbf{y}}{M}\right]|\mathbf{y}\rangle|a^{\mathbf{k}}\rangle = \sum_{\mathbf{y}} \frac{\exp\left[2\pi i\mathbf{r}\mathbf{y}l/M\right]-1}{\exp\left[2\pi i\mathbf{r}\mathbf{y}/M\right]-1} \exp\left[2\pi i\mathbf{k}\mathbf{y}/M\right]|\mathbf{y}\rangle|a^{\mathbf{k}}\rangle. \qquad (62)$$

Виміряймо $\mathbf{y}$, при цьому найбільш імовірним є результат, для якого виконується співвідношення $\mathbf{r}\mathbf{y}/M = p$, де p деяке ціле число. Оскільки $\mathbf{y}$ і M нам відомі, то побудуймо відношення $\mathbf{y}/M$ і замінімо його дробом $\frac{\mathbf{y}}{M} \approx \frac{D}{d}$, де $d < N$. Перевірмо чи $a^d \bmod N = 1$, якщо ні — то повторюймо алгоритм, якщо ж так — то запишімо $r_1 = d$. Після багатократного повторення отримаємо набір $r_1, r_2, \ldots$, мінімальне число з цього набору і буде шуканим періодом r. Застосування алгоритму Евкліда дасть значення співмножників числа N (див. Додаток Б та [4–9, 13]).

9. Прикінцеві зауваження

Раніше було відзначено, що властивість квантового регістра перебувати в суперпозиційному стані породжує квантовий паралелізм, котрий є однією з передумов надефективності квантового процесора.

Якщо приглянутись до приведених вище квантових алгоритмів, то можна помітити, що в процесі розрахунку регістри аргументу і значень функції перебувають в станах, які не можна зобразити у вигляді тензорного добутку. Такі стани називаються **заплутаними** (entangled), це дозволяє, проводячи вимірювання над одним регістром, фіксувати відповідний результат в іншому. Такий самий ефект спостерігається в парадоксі Айнштайна-Подольського-Розена. Це друга квантова властивість, яка дозволяє проводити квантові обчислення. Ці та інші квантові властивості квантовий процесор (як і будь-яка квантова система) зберігає доти доки зберігається квантова когерентність. Під впливом (класичного, тобто неконтрольованого, оточення) квантова система втрачає квантову когерентність, відбувається декогеренція (decoherence), яка руйнує квантовий процес, зокрема, процес квантового обчислення. Якщо декогеренція незначна і приводить до незначних похибок в процесі розрахунку, то цей процес можна стабілізувати і усунути похибки, застосувавши вже розроблені схеми квантової корекції похибок, подібно до того як робиться в класичних комп'ютерах. Такі схеми є досить складними і потребують окремого розгляду.

Ще складнішими є проблеми фізичної реалізації проектів квантових процесорів, тому їх теж тут не будемо розглядати, однак приведемо п'ять необхідних вимог, яким повинні задовольняти фізичні системи, що могли б бути використані як квантові процесори [9, 20]:

1) Необхідно виділити і зафіксувати в просторі набір дворівневих частинок-квабітів, на які можна було б в ході розрахунку діяти вибірково поодинці чи попарно і в такий спосіб організовувати їх квантову еволюцію, що відповідає виконуваному алгоритму. Повномасштабний квантовий процесор повинен складатися з $L > 10^3$ квабітів.

2) Необхідно забезпечити можливість приготування початкового стану $|0\ldots0\rangle_L$, тобто, ініціалізацію.

3) Час декогеренції повинен щонайменше в 10^4 перевищувати час однієї квантової операції (такту). Помилка при виконанні окремої квантової операції не повинна перевищувати 10^{-4}, що вважається прийнятним для роботи багатоквабітового процесора з використанням схем квантової корекції похибок.

4) Оскільки будь-яка унітарна квантова операція може бути зведена до сукупності одноквабітових і двоквабітових операцій, то при виборі фізичної системи суттєво, щоби між квабітами існували відповідні взаємодії для забезпечення виконання двоквабітових операцій. Імпульси, що керують операціями, повинні контролюватися з точ-

ністю не меншою ніж 10^{-4}.

5) Необхідно забезпечити високу надійність вимірювання квантової системи на виході.

Отже побудова квантового процесора зводиться до створення фізичної системи, що відповідає цим вимогам, і яка еволюціонує під дією квантової мережі, утвореної з одного з універсальних наборів КЛЕ. Такий фізичний квантовий процесор зреалізував би квантову машину Тюринга, що могла б ефективно розв'язувати деякі задачі, що є експонентно трудними для класичного комп'ютера.

10. Подяки

Я дякую Юрію Козицькому, який спонукав мене познайомитися з цією цікавою проблемою квантової механіки. Я глибоко вдячний також Олегу Держку, Володимиру Ткачуку, Юрію Криницькому і Тарасу Фітю за дискусії та критичні зауваження до рукопису. Іван Олександрович Вакарчук погодився обговорити цю тему на керованому ним семінарі "Філософія науки", за що я йому також щиро дякую.

11. Додаток А [13]

Система шифрування RSA була запропонована 1977 року Рональдом Райвестом, Аді Шаміром та Леонардом Адлеманом.

Вибирають два великі прості числа p і q. Для їх добутку $N = p\,q$ значення функції Ойлера дорівнює

$$\phi(N) = (p-1)(q-1).$$

Далі випадковим чином вибирають елемент e, що не перевищує значення $\phi(N)$ і є взаємно простим з $\phi(N)$. Після цього знаходять обернений елемент d

$$ed \mod \phi(N) = 1.$$

Тоді покладають e, N — відкритий ключ, d — закритий (таємний) ключ. Інформація зашифровується конфідентами, яким власник таємного ключа d надав відкритий ключ e, N. Шифрування здійснюється в кілька кроків: спочатку текст переводять в двійкову форму, тоді розділяють на блоки довжини m, такої щоб $2^m < N$. Тоді кожен з блоків вважають двійковим числом M_j (j – номер блоку), над яким здійснюють операцію $E(M) = M^e \mod N$. Зашифровану в такий спосіб інформацію відкрито надсилають власнику таємного

ключа, оскільки тільки він може її дешифрувати таким чином

$$E(M)^d \mod N = M,$$

тобто виконанням оберненої до шифрування операції. Несанкціонований доступ може отримати той, хто зуміє факторизувати число N, але це є дуже трудною задачею, оскільки числа p і q вибирають достатньо великими.

12. Додаток Б

Число x за модулем N тобто $x \mod N$ рівне залишку від ділення числа x на N, наприклад: $0 \bmod 7 = 0$, $1 \bmod 7 = 1$, $2 \bmod 7 = 2$, $\ldots, 7 \bmod 7 = 0$, $8 \bmod 7 = 1$, $9 \bmod 7 = 2$, $10 \bmod 7 = 3$ і т.д.

Розгляньмо процес факторизації числа $N = 21$ на прості множники 3 і 7. Для цього виберімо просте число менше від $N = 21$, наприклад $a = 5$, і побудуймо послідовність $a^j \mod N$, $j = 0, 1, 2, \ldots$. Отримаємо послідовність: $1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, \ldots$, звідси легко встановити, що $r = 6$, а також $a^{r/2} \bmod N = 6$, звідки знаходимо $a^{r/2} \bmod N + 1 = 7$, $a^{r/2} \bmod N - 1 = 5$. Співмножники чила 21 є серед найбільших спільних дільників чисел 21, 7 і 5, яким є число 7, а другий співмножник отримується діленням, що дає число 3.

Література

1. V.Vedral, M.B.Plenio. Basics of Quantum Computation. 1998. arXiv:quant-ph/9802065.
2. Ю.И.Манин. Вычислимое и невычислимое. М.: Советское радио. 1980. 128 с.
3. А.Боум. Квантовая механика. Основы и приложения. М.: Мир. 1990. 720с.
4. A.Steane. Quantum Computing. 1997. arXiv:quant-ph/9708022. (Є також російський переклад: Э.Стин. Квантовые вычисления. РХД. Москва-Ижевск. 2000. 111 с.)
5. D.Aharonov. Quantum Computation. 1998. arXiv:quant-ph/9812037.
6. A.Ekert, P.Hayden, H.Inamori. Basic Concepts in Quantum Computation. 2000. arXiv:quant-ph/0011013.

7. S.J.Lomonaco,Jr. A Rosetta Stone for Quantum Mechanics with an Introduction to Quantum Computation. 2000. arXiv:quant-ph/0007045.
8. P.W.Shor. Introduction to Quantum Algorithms. 2000. arXiv:quant-ph/0005003.
9. К.А.Валиев, А.А.Кокин. Квантовые компьютеры: надежды и реальность. РХД. Москва-Ижевск. 2000. 351 с.
10. Математическая энциклопедия. Т.1. Сов. энц. М.: 1982.
11. Н.Катленд. Вычислимость. Введение в теорию рекурсивных функций. М.: Мир. 1983. 256 с.
12. A.Ekert, C.Macchiavello. An Overview of Quantum Computing. In: Unconventional Models of Computation. Ed. by C.S.Calura, J.Casti, M.J.Dinneen. Springer-Verlag Singapore PteLtd. 1998. PP.19-44.
13. О.Вербіцький. Вступ до криптології. ВНТЛ. Львів. 1998. 247 с.
14. Р.Ф.Фейнман. Квантовомеханические ВМ. УФН. 1986. **149**, с. 671.
15. Р.П.Поплавский. Термодинамические модели информационных процессов. УФН. 1975. **115**, с.465-501.
16. Компьютеры. Справ. руков. в 3 томах. Под ред. Г.Хелмса. Т.1. Мир. М.: 1986. 413 с.
17. D.Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. Proc.Roy.Soc.London A. 1985. **400**, pp.97-117.
18. A.Barenco, C.H.Bennett, R.Cleve, D.P.DiVincenzo, N.Margolus, P.Shor. Elementary Gates for Quantum Computation. Phys. Rev. A. 1995. **52**, No 5. pp.3457-3467.
19. J.-L.Brylinski, R.Brylinski. Universal Quantum Gates. 2001. arXiv: quant-ph/0108062.
20. D.P.DiVincenzo, G.Burkard, D.Loss, E.V.Sukhorukov. Quantum Computation and Spin Electronics. 1999. arXiv:cond-mat/9911245.

Препринти Інституту фізики конденсованих систем НАН України розповсюджуються серед наукових та інформаційних установ. Вони також доступні по електронній комп'ютерній мережі на WWW-сервері інституту за адресою http://www.icmp.lviv.ua/

The preprints of the Institute for Condensed Matter Physics of the National Academy of Sciences of Ukraine are distributed to scientific and informational institutions. They also are available by computer network from Institute's WWW server (http://www.icmp.lviv.ua/)

Тарас Євстахійович Крохмальський

Квантові комп'ютери: основи і алгоритми (короткий огляд)

Роботу отримано 24 березня 2003 р.

Затверджено до друку Вченою радою ІФКС НАН України

Рекомендовано до друку семінаром відділу Статистичної теорії конденсованого стану

Виготовлено при ІФКС НАН України