

КВАНТОВІ КОМП'ЮТЕРИ: ОСНОВИ Й АЛГОРИТМИ (КОРОТКИЙ ОГЛЯД)

Т. Крохмальський

*Інститут фізики конденсованих систем Національної академії наук України
вул. Свенціцького, 1, Львів, 79011, Україна*

(Отримано 6 травня 2003 р.)

Цей огляд допоможе ознайомитись із теоретичними засадами виконання квантових розрахунків за допомогою квантових процесорів, із суттєвим використанням їхніх квантових властивостей. У ньому розглянуто поняття розраховності функцій, складності алгоритмів, класичної та квантової машин Тюринга, квантові біти, квантові логічні елементи, квантові обчислювальні мережі. Вияснено, на яких ефектах ґрунтується надфективність квантових комп'ютерів порівняно з класичними. Наведено приклади ефективних квантових алгоритмів.

Ключові слова: квантова інформатика, квантовий комп'ютер.

PACS number(s): 03.65.Ca, 07.05.Bx, 02.70.Rw, 89.80.+h

I. ВСТУП

В останньому десятилітті двадцятого століття в межах квантової механіки почала розвиватися нова галузь досліджень, що отримала назву **квантова інформатика**. Вона включає такі розділи, як квантові обчислення, квантова телепортація, квантова криптографія та інші. Ця нова галузь послуговується здебільшого теоретичними методами як через дороговизну експериментальних, так і через їх суттєву технічну обмеженість, оскільки тільки останніми роками вдалося збудувати експериментальні пристрої, які дали змогу 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 кількість електронних елементів в одиниці об'єму класичного комп'ютера подвоюється щороку, і, за оцінками 1995 року, в 2017 році один елемент (скажімо – транзистор) буде припадати на один атом. Тут виникають питання: де лежить класична межа і чи можуть мікроскопічні частинки виконувати функції електронних елементів комп'ютера?

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

Ще одне джерело лежить у таких засадничих проблемах квантової механіки, як парадокс Айнштайна–Подольського–Розена, парадокс шрединґерового kota, нерівності Белла та інше [3].

Ми не розглядаємо всіх складних проблем квантової інформатики, а тільки робимо спробу познайомити читача, який знає основи квантової механіки, із засадами виконання квантових обчислень, тобто з принципом дії квантового комп'ютера (точніше – квантового процесора), проєкт якого розвивають багато авторів протягом останніх років.

З короткою історією виникнення й розвитку квантової інформатики, а також з її іншими розділами

можна ознайомитися за препринтом [4].

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

Я не спеціаліст із квантової інформатики, але, як автор, мушу визнати, що сприймаю її як поглиблене дослідження основ квантової механіки і дуже скептично ставлюся до проєктів її фізичної реалізації.

II. 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 правильний результат не залежно від кваліфікації виконавця. Великі за обсягом 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 становлять собою певні математичні системи, у межах яких описують й аналізують алгоритми. Вивчення АОМ дає змогу 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 реєстрами (МНР), яку вперше розглянули Шепердсон і Стерджис (1963), тому ми коротко зупинимось на її аналізі [11].

МНР складається з безмежної кількості реєстрів, що позначаються через $R_1, R_2, R_3, \dots, R_j, \dots$, кожен з яких (R_j) містить деяке натуральне число (r_j). МНР змінює вміст реєстрів у відповідь на деякі *команди*, що відповідають найпростішим операціям над

числами. Скінченний 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 з програмою P , починаючи з деякої початкової конфігурації $\{r_i = a_i\}$ (теоретично i може прямувати до ∞), і закінчує обчислення після скінченного числа кроків у кінцевій конфігурації $\{r_j = b_j\}$. Як правило, розглядаються скінченні початкові $\{a_i\}$ та кінцеві $\{b_j\}$ конфігурації $1 \leq i \leq n$ і $1 \leq j \leq m$. Кажуть, що обчислення збігаються, якщо машина зупиняється, і розбігаються, 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 становлять незліченну множину, тоді як для МНР із чотирьох базисних команд можна збудувати зліченну множину програм [11].

Як у такому разі окреслити множину функцій, розраховних на МНР?

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

- a) нуль-функція \odot ($\odot(x) = 0$ для всіх x);
- b) функція додавання одиниці $x + 1$;
- c) функція проєкції U_i^n : $U_i^n(x_1, x_2, \dots, x_n) = x_i$; $n \geq 1, 1 \leq i \leq n$.

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

Рекурсія. Нехай $\mathbf{x} = (x_1, \dots, x_n)$, 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 задовольняє 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 ре-**

курсивними функціями [11]. До них належать:

- a) $x + y$;
- b) $x \cdot y$;
- c) x^y ;
- d) $x \dot{-} 1$ ($0 \dot{-} 1 = 0$);
- e) $x \dot{-} y = \begin{cases} x - y, & x \geq y, \\ 0, & x < y; \end{cases}$
- f) $\text{sg}(x) = \begin{cases} 1, & x = 0, \\ 0, & x \neq 0; \end{cases}$
- g) $\overline{\text{sg}}(x) = \begin{cases} 0, & x = 0, \\ 1, & x \neq 0; \end{cases}$
- h) $|x - y|$;
- i) $x!$;
- j) $\min(x, y)$;
- k) $\max(x, y)$;

l) $\text{gm}(x, y)$ — залишок від ділення y на x ($\text{gm}(0, y) \stackrel{\text{def}}{=} 0$);

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

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

o) Скінченна сума і скінченний 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 $f(\mathbf{x}, y) = 0$ щодо y . Інакше кажучи, 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 $\odot, x + 1, U_i^n$ і замкнутий стосовно операцій суперпозиції, рекурсії та мінімізації.

Або \mathcal{R} — це клас частинних функцій, кожен з яких можна отримати з основних функцій за допомогою скінченного числа операцій суперпозиції, рекурсії та мінімізації [11].

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

У 1930–60-их роках були запропоновані інші АОМ і означені відповідні їм класи розраховних функцій, не обов’язково 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**:

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

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

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

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 $\mathcal{C}(T)$, називається **універсальною АОМ**. Інакше кажучи, універсальна АОМ моделює всі інші АОМ.

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

III. СКЛАДНІСТЬ АЛГОРИТМІВ

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

Алгоритми, що отримують на вході тільки аргументи функції, яку вони розраховують, з послідовністю команд, що залежить тільки від функції та 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 ймовірністю (напр. $P > 2/3$), називаються **ймовірнісними** (або **рандомізованими**). До них, зокрема, належать Монте-Карло-алгоритми.

Для різних функцій із класу $\mathcal{C}(T)$ використовують 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 цій ОМ, найчастіше $k = 2$.

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. Ймовірнісні 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 ймовірністю $P > 2/3$.

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

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

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

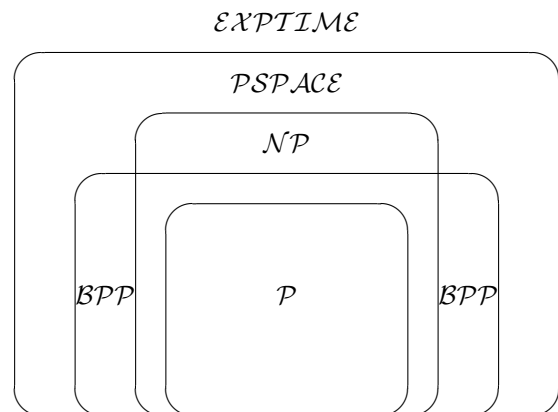


Рис. 1. Зв'язок між різними класами складності алгоритмів [12].

Приклади.

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

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

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) вимагає часу $t \simeq cL^3 \log \frac{1}{\varepsilon}$, де ε визначає ймовірність 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 дорівнює 10^{18} сек (42 дні). Якщо ж $L = 600$, то $t \simeq 10^{25}$ сек.

Як уперше зауважив Файнман [14], 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 – $2^{N \log_2 m}$ станів, а якщо процесор не розпаралелений, то це тягне за собою експонентне зростання часу розрахунку.

Для ілюстрації складності квантовомеханічних задач наведемо оцінку з праці [15]: “Для квантовомеханічного розрахунку молекули метану необхідно провести обчислення за методом сіток у 10^{42} точках. Якщо ж вважати, що в кожній точці треба виконати всього 10 елементарних операцій, і припустити, що всі обчислення відбуваються при наднизькій температурі ($T = 3 \cdot 10^{-3}$ 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 нема методів теоретичного доведення того, чи для цієї задачі можна збудувати ефективний алгоритм. Це вирішується тільки шляхом практичного створення відповідного алгоритму [13].

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

ІV. АЛГЕБРА БУЛЯ І КЛАСИЧНІ ОБЧИСЛЮВАЛЬНІ МАШИНИ

Алфавіти й команди АОМ можна реалізувати на алгебрі Буля, а операції алгебри Буля та її змінні, своєю чергою, можна фізично реалізувати у вигляді механічних чи електронних пристроїв та їх станів, що творить ґрунт для практичної побудови обчислювальних машин як технічних пристроїв [16].

Довільне натуральне число можна задати у двійковому зображенні:

$$x = x_n 2^n + x_{n-1} 2^{n-1} + \dots + x_1 2 + x_0, \quad (4.1)$$

яке символічно записуємо у вигляді:

$$x = x_n x_{n-1} \dots x_1 x_0, \quad (4.2)$$

де $x_j = \{0, 1\}$.

Ірраціональні числа зображаємо так:

$$r = A_n 2^n + \dots + A_1 2 + A_0 \sum_{j=1}^{\infty} a_j 2^{-j}, \quad (4.3)$$

і записуємо як

$$r = A_n A_{n-1} \dots A_1 A_0, a_1 a_2 \dots a_j \dots, \quad (4.4)$$

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 (4.1)–(4.4) насправді є назвами чисел у двійковому зображенні, а не самими числами. У прийнятому позиційному записі назв чисел при основі k використовуємо k відмінних між собою символів. Правила виконання арифметичних дій аль-Хорезмі (алгоритми), сформульовані для назв чисел у десятковому зображенні, чинні для дій з назвами чисел у довільному позиційному зображенні і приводять до назв чисел-результатів цих дій. Розгляньмо прості дії з числами (4.1)–(4.2).

$$7 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 111,$$

$$4 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 100; \quad (4.5)$$

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

$$\begin{array}{r} 111 \\ + 100 \\ \hline 1011 \end{array} \quad (4.6)$$

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

$$\begin{array}{r} 111 \\ - 100 \\ \hline 011 \end{array} \quad (4.7)$$

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

$$\begin{array}{r} 111 \\ \times 100 \\ \hline 000 \\ 000 \\ 111 \\ \hline 11100. \end{array} \quad (4.8)$$

З прикладів (4.6)–(4.8) видно, що ці елементарні операції легко реалізувати фізично, якщо позиції в записі числа замінити дробками рахівниці, коліщатками, магнетними кільцями чи транзисторами, а значення 0, 1 — відповідно їхніми станами. Зміна цих станів під час виконання арифметичних операцій повинна відповідати алгоритмам (4.6)–(4.8) та подібним до них.

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

При виконанні операцій (4.6)–(4.8) були використані такі правила:

$$\begin{aligned} 0 + 1 &= 1 + 0 = 1, \\ 0 + 0 &= 1 + 1 = 0 \quad \text{— додавання за модулем 2;} \\ 0 \cdot 0 &= 0 \cdot 1 = 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1. \end{aligned}$$

Якщо числом 0 і 1 поставити у відповідність булеві змінні **O** і **I**, а з операціями + та · зіставити логічні операції OR та AND, то можна виявити повну відповідність:

$$\begin{aligned} 0 \cdot 0 &= 0 \leftrightarrow \mathbf{O \text{ AND } O = O}; \\ 0 \cdot 1 &= 0 \leftrightarrow \mathbf{O \text{ AND } I = O}; \\ 1 \cdot 0 &= 0 \leftrightarrow \mathbf{I \text{ AND } O = O}; \\ 1 \cdot 1 &= 1 \leftrightarrow \mathbf{I \text{ AND } I = I}; \end{aligned} \quad (4.9)$$

$$\begin{aligned} 0 + 0 &= 0 \leftrightarrow \mathbf{O \text{ OR } O = O}; \\ 0 + 1 &= 1 \leftrightarrow \mathbf{O \text{ OR } I = I}; \\ 1 + 0 &= 1 \leftrightarrow \mathbf{I \text{ OR } O = I}; \\ 1 + 1 &= 0 \leftrightarrow \mathbf{I \text{ OR } I = I}; \end{aligned} \quad (4.10)$$

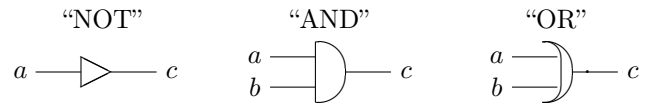
за винятком останнього рядка у (4.10), тому, замість операції OR, у комп'ютерах використовують операцію “виключного” OR (XOR), яка визначена такими правилами:

$$\begin{aligned} 0 + 0 &= 0 \leftrightarrow \mathbf{O \text{ XOR } O = O}; \\ 0 + 1 &= 1 \leftrightarrow \mathbf{O \text{ XOR } I = I}; \\ 1 + 0 &= 1 \leftrightarrow \mathbf{I \text{ XOR } O = I}; \\ 1 + 1 &= 0 \leftrightarrow \mathbf{I \text{ XOR } I = O}. \end{aligned} \quad (4.11)$$

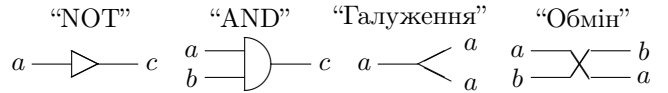
Окрім наведених операцій, наведених вище, в алгебрі Буля, є ще операція заперечення NOT з властивостями:

$$\text{NOT } \mathbf{I} = \mathbf{O}, \quad \text{NOT } \mathbf{O} = \mathbf{I}. \quad (4.12)$$

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

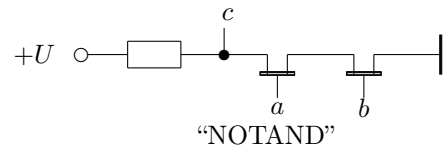


Доведено, що зі скінченної кількості елементів “NOT”, “AND”, “OR” чи “XOR” можна збудувати скінченну універсальну обчислювальну машину [17]. Класичну обчислювальну машину можна збудувати і з меншої кількості елементів [17]:



Тут лінії означають провідники, якими подають стандартні напруги.

Мало того, виявляється [17], що ОМ можна збудувати з одного логічного елемента “NOTAND”, який фізично можна реалізувати так:



Дія цього логічного елемента описується такою таблицею:

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

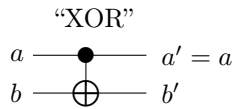
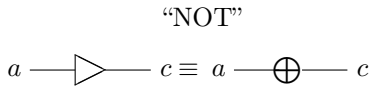
де 1 означає наявність напруги, а 0 — її відсутність. У точці c , з'єднаній через резистор з клемою, на яку подано напругу $+U$, завжди буде напруга, за винятком випадку, коли в обидві точки a і b одночасно подано напругу, тобто коли одночасно замкнено відповідні контакти.

V. ЗВОРОТНІ ОБЧИСЛЮВАЛЬНІ МАШИНИ

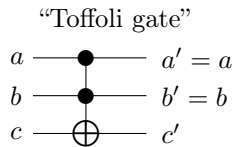
Легко помітити, що одному й тому ж вихідному стану логічних елементів “AND”, “OR” чи “XOR” відповідають декілька вхідних, що свідчить про математичну незворотність цих логічних елементів. Класична обчислювальна машина, складена з незворотних елементів, є незворотною обчислювальною машиною, тобто запущена у 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 (С. Bennet) (див. [4]) на початку 1980-х років показав, що зі зворотних елементів можна збудувати зворотну ОМ, яка для детермінованих алгоритмів при зворотному виконанні програми приходять 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, інакше кажучи, не можна виконати математичний розрахунок, не збільшивши ентропії в системі ОМ+оточення [17].

У працях Toffoli (1981) (див. [4]) було запропоновано набір зворотних логічних елементів:



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



Останній логічний елемент діє так, що $c' = \text{NOT } c$ 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.

VI. КВАНТОВІ РОЗРАХУНКИ

Замінюючи логічні елементи у схемі зворотної ОМ унітарними операторами, а її стани — станами дворівневих квантових систем, можна спробувати збудувати відповідну квантову ОМ, що і зробили Бенюв (P. Benioff, 1981) і Файнман (R. Feynman, 1982) (див. [4]). 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 [18]. Він

запропонував зовсім інший підхід до поняття розраховності.

Поняття розраховності, яке ми розглянули раніше, ґрунтується 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 означає поняття досконалого уподібнення (perfect simulation):

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

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

Кожна скінченна 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 у класичній фізиці через неперервність станів класичних фізичних систем. Успіх моделювання класичних систем він пов'язує з успішною дискретизацією моделей таких систем. Далі Д. Дойч стверджує: якщо вдасться побудувати таку ОМ (як фізичну систему), тоді досконале уподібнення певного класу (чи всіх) фізичних систем буде формулюванням властивостей цих систем мовою 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 QT (а значить і квантовий процесор) не розширює класу розраховних функцій, тобто $\mathcal{C}(QT) = \mathcal{C}(T)$.

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

VII. КВАНТОВІ БІТИ, КВАНТОВІ ЛОГІЧНІ ЕЛЕМЕНТИ І КВАНТОВІ МЕРЕЖІ

У цьому розділі розглянемо базові структурні елементи й 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 зі станів окремих (у певному сенсі не взаємодіючих між собою) двостанових квантових систем, званих квантовими бітами (qubits), які далі будемо скорочено називати квабітами. Для опису станів квабітів у гільбертовому просторі використовують спірне зображення

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (7.13)$$

Низка n -квабітів утворює n -квабітовий реєстр, стани якого лежать у гільбертовому просторі, що є тензорним добутком n гільбертових просторів окремих квабітів

$$\mathcal{H}^{[n]} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \dots \otimes \mathcal{H}_n. \quad (7.14)$$

Відповідні спірні базиси також є тензорним добутком, наприклад

$$|00\rangle = |0\rangle|0\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (7.15)$$

$$|01\rangle = |0\rangle|1\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad (7.16)$$

$$|10\rangle = |1\rangle|0\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad (7.17)$$

$$|11\rangle = |1\rangle|1\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad (7.18)$$

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

$$\begin{aligned} \mathbf{x} \leftrightarrow |\mathbf{x}\rangle &\equiv |x_{n-1}x_{n-2}\dots x_2x_1x_0\rangle \\ &\equiv |x_{n-1}\rangle \otimes \dots \otimes |x_1\rangle \otimes |x_0\rangle, \quad x_j = \{0, 1\}. \end{aligned} \quad (7.19)$$

Арифметичним операціям із цими числами (за mod 2^n) відповідають зміни станів окремих квабітів квантового реєстру, які здійснюються під впливом певних фізичних чинників (як правило — електромагнетних 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 реєстр у гільбертовому просторі його станів описується унітарними операторами.

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

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \hat{\Phi}(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}. \quad (7.20)$$

З означення перетворення Адамара легко встановити його дію

$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad \hat{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \quad (7.21)$$

або

$$\begin{aligned} \hat{H}|x\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi x}|1\rangle) \\ &= \frac{1}{\sqrt{2}} \sum_{y=0}^1 e^{i\pi xy}|y\rangle, \end{aligned} \quad (7.22)$$

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

$$\begin{aligned} \hat{H}^{[n]}|0\dots 0\rangle_n &= \frac{1}{2^{\frac{n}{2}}}(|0\rangle + |1\rangle) \dots (|0\rangle + |1\rangle) \\ &= \frac{1}{2^{\frac{n}{2}}} \sum_{x_{n-1}=0}^1 \dots \sum_{x_1=0}^1 \sum_{x_0=0}^1 |x_{n-1}x_{n-2}\dots x_2x_1x_0\rangle \\ &= \frac{1}{2^{\frac{n}{2}}} \sum_{\mathbf{x}} |\mathbf{x}\rangle. \end{aligned} \quad (7.23)$$

Як бачимо, дія n одноквабітових операторів Адамара на кожен із квабітів n -квабітового реєстру перевела останній у стан, що є суперпозицією 2^n станів, кожен з яких визначає натуральне число з відрізка $0\dots 2^n - 1$.

Тобто такий стан відображає одночасно всі числа з відрізка $0 \dots 2^n - 1$ з однаковою амплітудою.

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

Класичні комп'ютери такої властивості не мають, 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 $\mathbf{y} = y_{n-1}y_{n-2} \dots y_2y_1y_0$, ($y_j = \{0, 1\}$), переводить його в стан, що задає "суперпозицію" чисел

$$\hat{H}^{[n]}|\mathbf{y}\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x_{n-1}, \dots, x_0=0}^1 (-1)^{\mathbf{x} \cdot \mathbf{y}} |\mathbf{x}\rangle, \quad (7.24)$$

$$\mathbf{y} \cdot \mathbf{x} \equiv (y_{n-1}x_{n-1} + \dots + y_1x_1 + y_0x_0) \bmod 2. \quad (7.25)$$

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 $\mathbf{x}\mathbf{y} \bmod 2^n$, тоді як множення (7.25) швидше нагадує скалярний добуток за $\bmod 2$.

Слід указати на відносність суперпозиції: стани, задані як суперпозиція, в іншому базисі будуть мати зовсім інший вигляд.

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

$$\hat{\Phi}(\varphi)|x\rangle = e^{i\varphi x}|x\rangle, \quad x = \{0, 1\}. \quad (7.26)$$

Операторів Адамара й операторів зміни фаз достатньо, 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 — горизонтальна лінія, а дія КЛЕ на цей квабіт — певний знак на цій лінії. Наприклад, дія оператора Адамара \hat{H} на квабіт $|x\rangle$ зображається графічно так:

$$\begin{aligned} \hat{H}|x\rangle &\leftrightarrow |x\rangle \text{---} \boxed{\text{H}} \text{---} \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) \\ &\equiv \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi x}|1\rangle) = \frac{1}{\sqrt{2}} \sum_y e^{i\pi xy}|y\rangle, \end{aligned} \quad (7.27)$$

а дія оператора зміни фази —

$$\hat{\Phi}(\varphi)|x\rangle \leftrightarrow |x\rangle \text{---} \bullet \text{---} e^{i\varphi x}|x\rangle, \quad (7.28)$$

Найзагальніший одноквабітовий унітарний опера-

тор залежить від чотирьох параметрів, і його матриця має вигляд (А. Vargeno, 1995) [19]

$$\hat{U} = \begin{bmatrix} e^{i(\delta + \frac{\alpha}{2} + \frac{\beta}{2})} \cos(\theta) & e^{i(\delta + \frac{\alpha}{2} - \frac{\beta}{2})} \sin(\theta) \\ -e^{i(\delta - \frac{\alpha}{2} + \frac{\beta}{2})} \sin(\theta) & e^{i(\delta - \frac{\alpha}{2} - \frac{\beta}{2})} \cos(\theta) \end{bmatrix}, \quad (7.29)$$

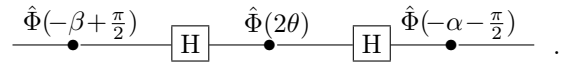
її можна факторизувати

$$\begin{aligned} \hat{U} &= e^{i(\delta + \frac{\alpha}{2} + \frac{\beta}{2})} \\ &\times \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\alpha} \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\beta} \end{bmatrix}. \end{aligned} \quad (7.30)$$

Після нескладних перетворень отримаємо

$$\hat{U} = e^{i(\delta + \frac{\alpha}{2} + \frac{\beta}{2} - \theta)} \hat{\Phi}\left(-\alpha - \frac{\pi}{2}\right) \hat{H} \hat{\Phi}(2\theta) \hat{H} \hat{\Phi}\left(-\beta + \frac{\pi}{2}\right). \quad (7.31)$$

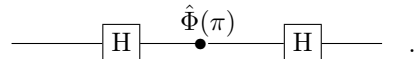
З точністю до загального фазового множника квантова мережа цього перетворення є такою:



Як видно з (7.29), при $-\alpha = \beta = \delta = \theta = \frac{\pi}{2}$ оператор \hat{U} стає оператором заперечення, тобто реалізує операцію NOT алгебри Буля

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_x|0\rangle = |1\rangle, \quad \sigma_x|1\rangle = |0\rangle, \quad (7.32)$$

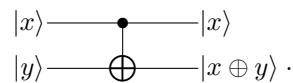
а його квантова мережа спрощується до



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 контрольованого НЕ (XOR або CNOT) має вигляд

$$\text{XOR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad (7.33)$$

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

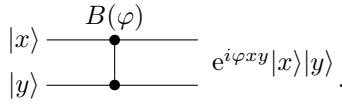
$$\text{XOR}|x\rangle|0\rangle = |x\rangle|x\rangle, \quad (7.34)$$

і може видатися, що порушується теорема про відсутність клонування, але ніякого клонування немає, бо (7.34) правильно тільки для відомих станів $x = 0$ або $x = 1$.

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

$$B(\varphi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}, \quad (7.35)$$

а його квантова мережа має вигляд



Матриці (7.33) і (7.35) мають структуру, характерну для контрольованих операторів: на діагоналі матриці всюди стоять одиниці, за винятком двох елементів справа внизу, де розташована 2×2 матриця оператора, який змінює стан останнього (згори) кубіта при умові, що всі інші кубіти перебувають у станах, які відповідають 1

$$CU_{[n \times n]} = \begin{bmatrix} I_{[n-2 \times n-2]} & O \\ O & U_{[2 \times 2]} \end{bmatrix}. \quad (7.36)$$

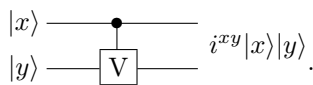
Уведемо однокубітовий оператор

$$V = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad (7.37)$$

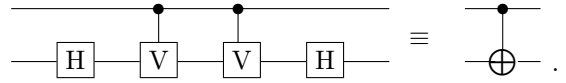
де i – уявна одиниця, і відповідний йому двокубітовий контрольований оператор

$$CV = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} = B\left(\frac{\pi}{2}\right), \quad (7.38)$$

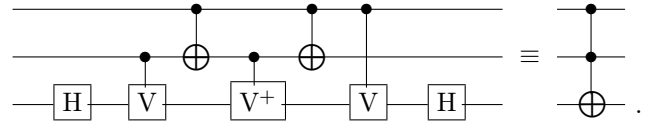
що є частковим випадком оператора $B(\varphi)$ і так зображається графічно:



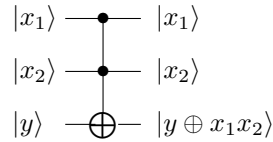
Із цього оператора й оператора Адамара можна збудувати квантову мережу, яка реалізує оператор XOR



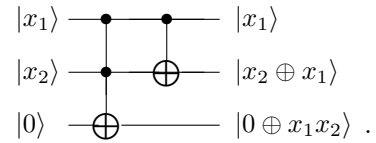
Аналогічно з цих двох операторів можна збудувати КЛЕ – квантовий відповідник “Toffoli gate” з трьома входами та виходами



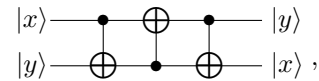
Останній КЛЕ виконує таку ж операцію, як у класичному випадку “Toffoli gate”,



і разом з КЛЕ “XOR” дає змогу збудувати суматор з переносом у вищий розряд



Часто вживаний при побудові квантових мереж КЛЕ “SWAP”, який здійснює обмін місцями двох кубітів, так пов'язаний з КЛЕ “XOR”:



звідки можна легко отримати матрицю оператора SWAP у спірному базисі

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (7.39)$$

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

$$\sqrt{\text{XOR}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix}. \quad (7.40)$$

У праці А. Баренцо *та ін.* (А. Varenco *et al.*, 1995) [19] доведено, що з двох однокубітових операторів \hat{H} і $\hat{F}(\varphi)$ та з одного двокубітового (наприклад, з XOR чи іншого оператора) можна точно побудувати довільний унітарний оператор, що діє в гільбертовому

просторі $\mathcal{H}_1 \otimes \dots \otimes \mathcal{H}_n$, і використати при цьому $4^n n^3$ цих базових операторів (у цій же статті зазначено, що в неопублікованій роботі Е. Knill отримано значення $4^n n$). А це вказує на те, що з певним чином дібраних базових КЛЕ можна збудувати квантову мережу, яка реалізує унітарний оператор, котрий виконує відповідні арифметичні розрахунки.

У праці [20] 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, *імпримітивним*, якщо він не є примітивним.

У цій же роботі [20] доведено теорему.

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

- (i) *набір усіх 1-квабітових КЛЕ A разом з V є універсальним;*
- (ii) *набір усіх 1-квабітових КЛЕ A разом з V є точно універсальним;*
- (iii) *V є імпримітивним.*

Варто звернути увагу, що у класичній зворотній ОМ універсальним був трибітовий “Toffoli gate”, тоді як у квантовому випадку універсальний набір обмежується двоквабітовими КЛЕ. Успіх побудови реального квантового процесора тісно пов’язаний з таким вибором універсального набору КЛЕ, який вдасться реалізувати фізично як послідовність переходів у фізичній системі при збереженні її квантової когерентності. Докладніше універсальні набори КЛЕ розглянуто в [5, 19].

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

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

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

Залишаються, звичайно, проблеми приготування початкового і “відчитання” кінцевого станів. Зрозуміло, що при одному й тому ж початковому стані унітарна еволюція квантової мережі приводитиме до того ж самого кінцевого стану (при збереженні квантової когерентності, тобто за відсутності впливів оточення на роботу квантового процесора). Але оскільки “відчитати” кінцевий стан можливо, тільки здійснюючи вимірювання, тобто проектування на власні стани деякого оператора, то дістати результати розрахунку можна зі статистичного аналізу розподілу результатів вимірювань, отриманих при багатократному повторенні роботи квантового процесора. Достовірно прийнято вважати відповідь, знайдену з імовірністю $P \geq 2/3$.

VIII. 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 $\mathcal{H}^{[n+m]}$, у якому квантовий реєстр може перебувати в 2^{n+m} станах. Зазвичай, для аналізу запропонованих квантових алгоритмів зручно розділити квантовий реєстр на два: реєстр аргументу $|x\rangle_n$ і реєстр значень функції $|y\rangle_m$. Тоді розрахунок функції можна зобразити так:

$$|x\rangle_n |y\rangle_m \xrightarrow{f(x)} |x\rangle_n |y \oplus f(x) \bmod 2^m\rangle_m \quad (8.41)$$

або

$$|\mathbf{x}\rangle |0\rangle \xrightarrow{f(\mathbf{x})} |\mathbf{x}\rangle |f(\mathbf{x})\rangle. \quad (8.42)$$

На сьогодні опубліковано багато проектів квантових мереж для ефективного (поліномного) розрахунку тих чи інших розраховних функцій. Квантова неадекватність може проявлятися не обов’язково в розрахунку самих функцій, а в дослідженні деяких їхніх властивостей. Тому далі здебільшого будемо вважати, що ми маємо квантову (під)мережу-“оракул” (oracle), яка ефективно розраховує деяку функцію $f(\mathbf{x})$. Задача полягає в тому, щоби побудувати до неї нову квантову мережу, яка б дала змогу ефективно (в поліномний час чи з поліномним числом КЛЕ, що те саме) вивчити властивості функції $f(\mathbf{x})$, що на класичному комп’ютері призводить до експонентних затрат.

Розглянемо далі декілька прикладів найпопулярніших квантових алгоритмів, які ілюструють неадекватність квантових процесорів.

A. Квантове перетворення Фур’є [5]

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

$$\mathbf{x} = x_{n-1}2^{n-1} + \dots + x_12^1 + x_02^0 = \sum_{j=0}^{n-1} x_j2^j, \quad (8.43)$$

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

$$\mathbf{xy} = \sum_{j=0}^{n-1} x_j2^j y_0 + 2 \sum_{j=0}^{n-2} x_j2^j y_1 + \dots + 2^{n-2}(x_12 + x_0) y_{n-2} + 2^{n-1}x_0 y_{n-1}. \quad (8.44)$$

Не становить труднощів й отримати частку від ділення останнього виразу на 2^n (знову ж, за модулем 2^n)

$$\begin{aligned} \frac{\mathbf{xy}}{2^n} &= \left(\frac{x_{n-1}}{2} + \frac{x_{n-2}}{2^2} + \dots + \frac{x_1}{2^{n-1}} + \frac{x_0}{2^n} \right) y_0 \left(\frac{x_{n-2}}{2} + \frac{x_{n-3}}{2^2} + \dots + \frac{x_1}{2^{n-2}} + \frac{x_0}{2^{n-1}} \right) y_1 \\ &+ \left(\frac{x_{n-3}}{2} + \frac{x_{n-4}}{2^2} + \dots + \frac{x_1}{2^{n-3}} + \frac{x_0}{2^{n-2}} \right) y_2 \dots + \left(\frac{x_1}{2} + \frac{x_0}{2^2} \right) y_{n-2} + \frac{x_0}{2^2} y_{n-1}. \end{aligned} \quad (8.45)$$

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, пригадаймо, як діють її окремі елементи: оператор Адамара діє на окремий квант

$$\hat{H}|x_j\rangle = |0\rangle + e^{i\pi x_j}|1\rangle, \quad (8.46)$$

а оператор зміни фази в цій мережі — на два кванти

$$\hat{B}(\varphi)|x_k\rangle|x_j\rangle = e^{i\varphi x_k x_j}|x_k\rangle|x_j\rangle. \quad (8.47)$$

(Тут і далі в цьому розділі множники $1/\sqrt{2}$, що виникають при перетвореннях Адамара, будемо упускати.) Позначивши індексами біля значків операторів КЛЕ номери квантів, 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} (|0\rangle + e^{i\pi x_3}|1\rangle)|x_2\rangle|x_1\rangle|x_0\rangle \xrightarrow{B_{32}(\frac{\pi}{2})} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2})}|1\rangle)|x_2\rangle|x_1\rangle|x_0\rangle \xrightarrow{H_2} \\ &(|0\rangle + e^{i\pi(x_3+\frac{x_2}{2})}|1\rangle)(|0\rangle + e^{i\pi x_2}|1\rangle)|x_1\rangle|x_0\rangle \xrightarrow{B_{31}(\frac{\pi}{4})} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4})}|1\rangle)(|0\rangle + e^{i\pi x_2}|1\rangle)|x_1\rangle|x_0\rangle \xrightarrow{B_{21}(\frac{\pi}{2})} \\ &(|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2})}|1\rangle)|x_1\rangle|x_0\rangle \xrightarrow{H_1} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2})}|1\rangle) \\ &\times (|0\rangle + e^{i\pi x_1}|1\rangle)|x_0\rangle \xrightarrow{B_{30}(\frac{\pi}{8})} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2})}|1\rangle) \\ &\times (|0\rangle + e^{i\pi x_1}|1\rangle)|x_0\rangle \xrightarrow{B_{20}(\frac{\pi}{4})} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2}+\frac{x_0}{4})}|1\rangle) \\ &\times (|0\rangle + e^{i\pi x_1}|1\rangle)|x_0\rangle \xrightarrow{B_{10}(\frac{\pi}{2})} (|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2}+\frac{x_0}{4})}|1\rangle)(|0\rangle + e^{i\pi(x_1+\frac{x_0}{2})}|1\rangle)|x_0\rangle \xrightarrow{H_0} \\ &(|0\rangle + e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8})}|1\rangle)(|0\rangle + e^{i\pi(x_2+\frac{x_1}{2}+\frac{x_0}{4})}|1\rangle)(|0\rangle + e^{i\pi(x_1+\frac{x_0}{2})}|1\rangle)(|0\rangle + e^{i\pi x_0}|1\rangle) \\ &= \sum_{y_0} e^{i\pi(x_3+\frac{x_2}{2}+\frac{x_1}{4}+\frac{x_0}{8})y_0}|y_0\rangle \sum_{y_1} e^{i\pi(x_2+\frac{x_1}{2}+\frac{x_0}{4})y_1}|y_1\rangle \sum_{y_2} e^{i\pi(x_1+\frac{x_0}{2})y_2}|y_2\rangle \sum_{y_3} e^{i\pi x_0 y_3}|y_3\rangle = \sum_{\mathbf{y}} e^{i2\pi \frac{\mathbf{xy}}{2^4}}|\mathbf{y}\rangle. \end{aligned} \quad (8.48)$$

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

З рис. 2 і з перетворень (8.48) легко зрозуміти, як розширити мережу для квантового перетворення Фур'є n -розрядного числа й установити, що така мережа матиме $n(n+1)/2$ КЛЕ.

Припустимо, що ми перевели квантовий реєстр у суперпозиційний стан з амплітудами, які є значеннями деякої функції $f(\mathbf{x})$, а потім здійснили квантове перетворення Фур'є реєстру аргументу

$$\sum_{\mathbf{x}} f(\mathbf{x})|\mathbf{x}\rangle \xrightarrow{QFT} \sum_{\mathbf{x}} f(\mathbf{x}) \frac{1}{2^{n/2}} \sum_{\mathbf{y}} \exp\left(i2\pi \frac{\mathbf{x}\mathbf{y}}{2^n}\right) |\mathbf{y}\rangle = \sum_{\mathbf{y}} \tilde{f}(\mathbf{y})|\mathbf{y}\rangle, \quad (8.49)$$

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

$$\tilde{f}(\mathbf{y}) = \frac{1}{2^{n/2}} \sum_{\mathbf{x}} f(\mathbf{x}) \exp\left(i2\pi \frac{\mathbf{x}\mathbf{y}}{2^n}\right). \quad (8.50)$$

Звичайно, відчитання цих амплітуд не проста проблема, але у випадках, коли перетворення Фур'є є проміжною операцією в розрахунках, переваги квантового алгоритму безперечні. 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 $s = n(n+1)/2$.

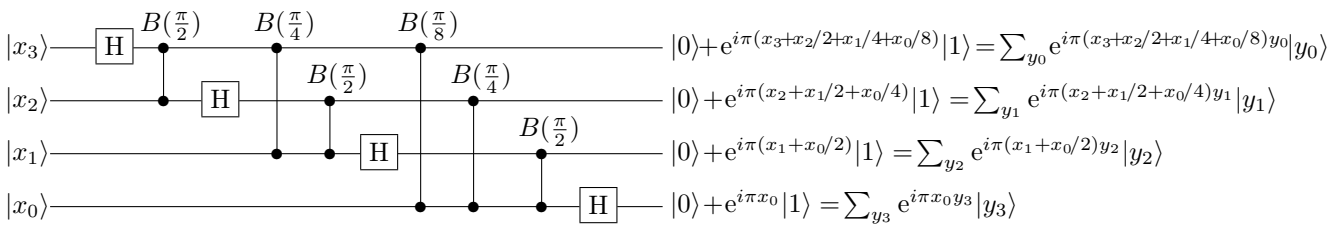


Рис. 2. Квантова мережа перетворення Фур'є чотириквбітового реєстру.

В. Задача Дойча–Джозси

Задача Дойча була історично першою, на якій показано, що експонентно трудні для класичних комп'ютерів задачі на квантових процесорах можна розв'язати з поліномними затратами. Задача полягає в тому, щоб установити, до якого класу належить функція — “constant” чи “balanced”. У першому варіанті її сформулював Дойч для бінарного аргументу. Коли

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

функція є сталою (“constant”), а коли

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

функція є змінною (“balanced”). Пізніше Д. Дойч і Р. Джозса (R. Jozsa) узагальнили цю задачу на довільний аргумент \mathbf{x} з відрізка $0 \dots N$, при тому, що сама функція приймає два значення $\{0, 1\}$. У цьому випадку функція вважається “constant”, якщо для всіх зна-

чень 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. У 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 реєстр аргументу з n квбітів і реєстр значень функції з одного квбіта, які приводяться в початковий стан

$$|0 \dots 0\rangle_n |1\rangle, \quad (8.51)$$

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

$$\sum_{\mathbf{x}} |\mathbf{x}\rangle (|0\rangle - |1\rangle). \quad (8.52)$$

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

$$\begin{aligned} & \sum_{\mathbf{x}} |\mathbf{x}\rangle (|f(\mathbf{x})\rangle - |1 + f(\mathbf{x})\rangle) \\ &= \sum_{\mathbf{x}} (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle (|0\rangle - |1\rangle), \end{aligned} \quad (8.53)$$

а тоді знову на кожен кватбіт аргументу діють оператором Адамара (див. (7.24) і (7.25)) і отримують суперпозицію

$$\sum_{\mathbf{x}, \mathbf{y}} (-1)^{f(\mathbf{x}) + \mathbf{x} \cdot \mathbf{y}} |\mathbf{y}\rangle (|0\rangle - |1\rangle). \quad (8.54)$$

Нарешті вимірюють значення числа \mathbf{y} , записаного в регістрі аргументу. Якщо $\mathbf{y} = 0$, то $f(\mathbf{x})$ є "constant", оскільки інакше амплітуда стану $|0 \dots 0\rangle$ дорівнювала б нулеві

$$\sum_{\mathbf{x}} (-1)^{f(\mathbf{x})} = 0, \quad (8.55)$$

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

$$\sum_{\mathbf{x}} (-1)^{\mathbf{x} \cdot \mathbf{y}} = 0. \quad (8.56)$$

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

С. Визначення періоду функції (задача D. Simona)

Нехай задана функція $f: \{0, 1\}^n \xrightarrow{f} \{0, 1\}^n$, 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 потрібно знайти його значення. Побудуємо два регістри по n кватбітів кожен, подіймо на кожен кватбіт регістру аргументу оператором Адамара, після чого обчислімо функцію f :

$$\begin{aligned} & |0 \dots 0\rangle_n |0 \dots 0\rangle_n \xrightarrow{H^{[n]}} \sum_{\mathbf{x}} |\mathbf{x}\rangle |0 \dots 0\rangle_n \xrightarrow{f} \\ & \xrightarrow{f} \sum_{\mathbf{x}} |\mathbf{x}\rangle |f(\mathbf{x})\rangle, \end{aligned} \quad (8.57)$$

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

$$(|\mathbf{k}\rangle + |\mathbf{k} + \mathbf{r}\rangle) |f(\mathbf{k})\rangle. \quad (8.58)$$

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

$$\begin{aligned} & \sum_{\mathbf{y}} \left[(-1)^{\mathbf{k} \cdot \mathbf{y}} + (-1)^{(\mathbf{k} + \mathbf{r}) \cdot \mathbf{y}} \right] |\mathbf{y}\rangle |f(\mathbf{k})\rangle \\ &= \sum_{\mathbf{y}} (-1)^{\mathbf{k} \cdot \mathbf{y}} [1 + (-1)^{\mathbf{r} \cdot \mathbf{y}}] |\mathbf{y}\rangle |f(\mathbf{k})\rangle. \end{aligned} \quad (8.59)$$

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.

D. 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 пов'язаний з тим, що факторизація великих натуральних чисел на прості співмножники є головним трудним етапом у несанкціонованому дешифруванні текстів, зашифрованих за методом RSA (див. додаток А), який є стандартом шифрування конфіденційної інформації в США та найпоширенішим методом шифрування інформації, що передається всесвітньою інформаційною мережею.

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 $a^j \bmod N$, $j = 0, \dots, (a^r \bmod N = 1)$ (див. додаток Б). Складною 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.

Збудуємо 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 переведемо регістр аргументу в суперпозиційний стан, після чого виберім число $a < N$ взаємно просте з N і розрахуємо функцію $a^x \bmod N$

$$\begin{aligned} & |0 \dots 0\rangle_m |0 \dots 0\rangle_n \xrightarrow{H^{[m]}} \sum_{\mathbf{x}} |\mathbf{x}\rangle |0\rangle_n \xrightarrow{a^x \bmod N} \\ & \xrightarrow{a^x \bmod N} \sum_{\mathbf{x}} |\mathbf{x}\rangle |a^x \bmod N\rangle. \end{aligned} \quad (8.60)$$

Виміримо значення функції

$$\begin{aligned} & (|\mathbf{k}\rangle + |\mathbf{k} + \mathbf{r}\rangle + |\mathbf{k} + 2\mathbf{r}\rangle + \dots + |\mathbf{k} + l\mathbf{r}\rangle) |a^{\mathbf{k}}\rangle \\ &= \sum_{j=0}^l |\mathbf{k} + j\mathbf{r}\rangle |a^{\mathbf{k}}\rangle \end{aligned} \quad (8.61)$$

і здійснено перетворення Фур'є регістру аргументу

$$\sum_{\mathbf{y}} \sum_{j=0}^l \exp \left[\frac{2\pi i(j\mathbf{r} + \mathbf{k})\mathbf{y}}{M} \right] |\mathbf{y}\rangle |a^{\mathbf{k}}\rangle \quad (8.62)$$

$$= \sum_{\mathbf{y}} \frac{\exp [2\pi i\mathbf{r}\mathbf{y}l/M] - 1}{\exp [2\pi i\mathbf{r}\mathbf{y}/M] - 1} \exp [2\pi i\mathbf{k}\mathbf{y}/M] |\mathbf{y}\rangle |a^{\mathbf{k}}\rangle.$$

Виміряймо \mathbf{y} , при цьому найімовірнішим 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 побудуємо 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, \dots , 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]).

ІХ. ПРИКІНЦЕВІ ЗАУВАЖЕННЯ

Раніше було відзначено, що властивість квантового регістру перебувати в суперпозиційному стані породжує квантовий паралелізм, який є однією з передумов надфективності квантового процесора. Якщо приглянутись до наведених вище квантових алгоритмів, то можна помітити, що в процесі розрахунку регістри аргументу і значень функції перебувають у станах, які не можна зобразити як тензорний добуток. Такі стани називаються **заплутаними** (entangled), це дає змогу, вимірюючи один регістр, фіксувати відповідний результат в іншому. Такий самий ефект спостерігаємо 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 невеликих похибок під час розрахунку, то цей процес можна стабілізувати й усунути їх, застосувавши вже розроблені схеми квантової корекції помилок, подібно до того, як робиться у класичних комп'ютерах. Такі схеми є досить складними й потребують окремого розгляду.

Ще складнішими є проблеми фізичної реалізації проєктів квантових процесорів, тому їх теж не будемо тут розглядати, однак наведемо п'ять необхідних вимог, що повинні задовольняти фізичні системи, які можна б використати як квантові процесори [9, 21]:

1). Неодмінно треба виділити й зафіксувати у просторі набір дворівневих частинок-квабітів, на які можна було під час розрахунку діяти вибірково поодинці

чи попарно й у такий спосіб організувати їх квантову еволюцію, що відповідає виконуваному алгоритмові. Повномасштабний квантовий процесор повинен складатися з $L > 10^3$ квабітів.

2). Слід забезпечити можливість приготування початкового стану $|0 \dots 0\rangle_L$, 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 була відповідна взаємодія для забезпечення виконання двоквабітових операцій. Імпульси, що керують операціями, повинні контролюватися з точністю не меншою ніж 10^{-4} .

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

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

Х. ПОДЯКИ

Я дякую Юрієві Козицькому, який спонукав мене ознайомитися з цією цікавою проблемою квантової механіки. Глибоко вдячний також Олегові Держкові, Володимирові Ткачукові, Юрієві Криницькому й Тарасові Фітю за дискусії та критичні зауваження до рукопису. Іван Олександрович Вакарчук погодився обговорити цю тему на семінарі “Філософія науки”, за що я йому також щиро дякую.

ХІ. ДОДАТОК А [13]

Систему шифрування RSA запропонували 1977 року Рональд Райвест, Аді Шамір та Леонард Адлеман.

Вибирають два великі прості числа p і q . Для їх добутку $N = pq$ 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

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

Тоді покладають e , N — відкритий ключ, d — закритий (таємний) ключ. Інформацію зашифровують конфіденти, яким власник таємного ключа d надав відкритий ключ e , N . Шифрування здійснюється кількома кроками: спочатку текст переводять у двійкову форму, тоді розділяють на блоки довжиною m , такою, щоб $2^m < N$. Тоді кожен з блоків уважають двійковим числом M_j (j — номер блоку), над яким здійснюють операцію $E(M) = M^e \bmod N$. Зашифровану в такий спосіб інформацію відкрито надсилають власникові таємного ключа, оскільки тільки він може її дешифрувати так:

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

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

ХІІ. ДОДАТОК Б

Число x за модулем N , тобто $x \bmod N$, дорівнює залишкові від ділення числа x на N , наприклад: $0 \bmod 7 = 0$, $1 \bmod 7 = 1$, $2 \bmod 7 = 2$, \dots , $7 \bmod 7 = 0$, $8 \bmod 7 = 1$, $9 \bmod 7 = 2$, $10 \bmod 7 = 3$ 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 $a^j \bmod N$, $j = 0, 1, 2, \dots$. Отримаємо послідовність: 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, \dots , 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 числа 21 є серед найбільших спільних дільників чисел 21, 7 і 5, яким є число 7, а другий співмножник отримуємо діленням, що дає число 3.

-
- [1] V. Vedral, M. B. Plenio, *Basics of Quantum Computation* arXiv:quant-ph/9802065 (1998).
- [2] Ю. И. Манин, *Вычислимое и невычислимое* (Советское радио, Москва, 1980).
- [3] А. Боум, *Квантовая механика. Основы и приложения* (Мир, Москва, 1990).
- [4] А. Steane, *Quantum Computing*, arXiv:quant-ph/9708022 (1997). (Є також російській переклад: Э. Стин, *Квантовые вычисления* (РХД, Москва–Ижевск, 2000).
- [5] D. Aharonov, *Quantum Computation* arXiv:quant-ph/9812037 (1998).
- [6] A. Ekert, P. Hayden, H. Inamori, *Basic Concepts in Quantum Computation* arXiv:quant-ph/0011013 (2000).
- [7] S. J. Lomonaco, Jr., *A Rosetta Stone for Quantum Mechanics with an Introduction to Quantum Computation* arXiv:quant-ph/0007045 (2000).
- [8] P. W. Shor, *Introduction to Quantum Algorithms* arXiv:quant-ph/0005003 (2000).
- [9] К. А. Валиев, А. А. Кокин, *Квантовые компьютеры: надежды и реальность* (РХД, Москва–Ижевск, 2000).
- [10] Математическая энциклопедия. Т. 1. (Советская энциклопедия, Москва, 1982).
- [11] Н. Катленд, *Вычислимость. Введение в теорию рекурсивных функций* (Мир, Москва, 1983).
- [12] A. Ekert, C. Macchiavello, In: *Unconventional Models of Computation*, edited by C. S. Calura, J. Casti, M. J. Dinneen, (Springer-Verlag Singapore, PteLtd, 1998). p. 19.
- [13] О. Вербицкий, *Вступ до криптології* (ВНТЛ, Львів, 1998).
- [14] R. Feynman, *Int. J. Theor. Phys.* **21**, 467(1982).
- [15] Р. П. Поплавский, *Усп. физ. наук* **115**, 465 (1975).
- [16] *Компьютеры. Справ. руков. в 3 томах*. Под ред. Г. Хелмса. Т. 1. (Мир, Москва, 1986).
- [17] Р. Ф. Фейнман, *Усп. физ. наук* **149**, 671 (1986).
- [18] D. Deutsch, *Proc. Roy. Soc. London A* **400**, 97 (1985).
- [19] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, *Phys. Rev. A* **52**, 3457 (1995).
- [20] J.-L. Brylinski, R. Brylinski, *Universal Quantum Gates* arXiv:quant-ph/0108062 (2001).
- [21] D. P. DiVincenzo, G. Burkard, D. Loss, E. V. Sukhorukov, *Quantum Computation and Spin Electronics* arXiv:cond-mat/9911245 (1999).

QUANTUM COMPUTERS: BASIS AND ALGORITHMS (A SHORT REVIEW)

T. Krokhmalkii

*Institute for Condensed Matter Physics of the National Academy of Sciences of Ukraine
1 Svientsitskii St., Lviv, UA-79011, Ukraine*

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