

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Е.В.Жаріков, К.І.Ліщук

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ КВАНТОВИХ ОБЧИСЛЕНЬ

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра
за освітньою програмою «Інженерія квантового програмного забезпечення»
спеціальності F2 Інженерія програмного забезпечення

Електронне мережеве навчальне видання

Київ
КПІ ім. ІГОРЯ СІКОРСЬКОГО
2026

УДК 004.382:004.42(075.8)
B19

Автори: *Жаріков Едуард В'ячеславович*, д-р техн. наук, проф.
Ліщук Катерина Ігорівна, канд. техн. наук, доц.

Рецензент: *Стеценко І.В.*, д-р техн. наук, професор,
професор кафедри інформатики та програмної інженерії

Відповідальний редактор: *Олійник Ю.О.*, канд. техн. наук, доц.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 11.09.2026 р.)
за поданням вченої ради факультету інформатики та обчислювальної техніки
(протокол № 1 від 27.08.2026 р.)*

Жаріков Е. В.
B19 Алгоритмічне забезпечення квантових обчислень [Електронний ресурс] : навч. посіб. для здобувачів ступеня магістра за освіт. програмою «Інженерія квантового програмного забезпечення» спец. F2 Інженерія програмного забезпечення / Е. В. Жаріков, К. І. Ліщук ; КПІ ім. Ігоря Сікорського. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2026. – 133 с.

Навчальний посібник охоплює теоретичний матеріал з дисципліни «Алгоритмічне забезпечення квантових обчислень» та призначений для здобувачів ступеня магістра спеціальності F2 Інженерія програмного забезпечення кафедри інформатики та програмної інженерії. У посібнику наведено систематизоване викладення основних понять квантових обчислень: кубіт, вентиль, оператор, суперпозиція, запутаність, телепортація, основи лінійної алгебри, а також основи проектування квантових алгоритмів і схем, що дозволить забезпечити набуття ключових фахових компетентностей, теоретичних знань і практичних навичок з проектування, програмування та застосування квантових алгоритмів. Посібник включає різні аспекти розробки та аналізу квантових схем для класичних до гібридних квантових алгоритмів. При створенні навчального посібника використані матеріали компаній AWS та IBM з орієнтацією на поглиблене вивчення технологій квантових обчислень студентами технічних спеціальностей.

Посібник може бути корисним студентам відповідних спеціальностей при вивченні дисциплін, пов'язаних із квантовими обчисленнями, фреймворками провідних постачальників квантових обчислень, а також при виконанні курсових робіт та магістерських дисертацій, у яких використовуються відповідні технології та підходи квантових обчислень. Оформлення посібника виконано згідно вимог до конструкторської документації.

УДК 004.382:004.42(075.8)

Реєстр. № 25/26-028. Обсяг 7 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
<https://kpi.ua>

Свідectво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

© Е. В. Жаріков, К. І. Ліщук, 2026
© КПІ ім. Ігоря Сікорського, 2026

ЗМІСТ

ВСТУП.....	5
1 ПРОБЛЕМИ СТВОРЕННЯ КВАНТОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	10
1.1 Вступ до квантових обчислень	10
1.2 Програмне забезпечення для симуляції квантових обчислень.....	22
2 ПОРІВНЯННЯ РОБОТИ КВАНТОВОГО КОМП'ЮТЕРА ТА КЛАСИЧНОГО КОМП'ЮТЕРА	24
2.1 Відмінності квантових та класичних обчислень, кубіти.....	26
2.2 Апаратне забезпечення для квантових обчислень	34
2.3 Приклад застосування квантових обчислень	36
2.4 Нотації представлення кубіта.....	40
3 МАТЕМАТИЧНІ ОСНОВИ КВАНТОВИХ ОБЧИСЛЕНЬ: ЛІНІЙНА АЛГЕБРА, ЙМОВІРНІСТЬ, МАТРИЦІ, КОМПЛЕКСНІ ЧИСЛА.....	45
3.1 Основи лінійної алгебри	45
3.2 Комплексні числа	48
4 ОСНОВНІ ПОНЯТТЯ КВАНТОВИХ ОБЧИСЛЕНЬ: КУБІТ, СУПЕРПОЗИЦІЯ, ЗАПЛУТАНІСТЬ, ТЕЛЕПОРТАЦІЯ	52
4.1 Модель квантових схем	69
4.2 Глобальна та відносна фаза	70
4.3 Квантові вентиля керованих операцій	71
4.4 Аналогія між булевими та квантовими вентилями.....	73
4.5 Квантова заплутаність.....	74
4.6 Оберненість (зворотність) та унітарність	76
5 КВАНТОВА ТЕЛЕПОРТАЦІЯ ТА НАДЩІЛЬНЕ КОДУВАННЯ.....	79
5.1 Квантова телепортація	79
5.2 Формальний опис квантової телепортації	84
5.3 Загальна схема квантових обчислень та квантовий паралелізм	88
5.4 Надщільне кодування.....	89
6 ПРОЕКТУВАННЯ КВАНТОВИХ АЛГОРИТМІВ І СХЕМ.....	93
6.1 Складність алгоритмів	93
6.2 Проектування квантових алгоритмів.....	98
6.3 Визначення поняття оракула.....	101
6.4 Підходи до оцінки функцій на прикладі алгоритмів Шора та Дойча ..	102
6.5 Від функції до оракула.....	107

6.6	Алгоритм Дойча.....	109
6.7	Алгоритм Дойча-Йожи	115
6.8	Гібридні квантові алгоритми, квантовий алгоритм наближеної оптимізації.....	123
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....		128

ВСТУП

Квантові обчислення представляють собою один із найбільш революційних напрямків розвитку інформаційних технологій XXI століття, що ґрунтується на фундаментальних принципах квантової механіки та відкриває безпрецедентні можливості для розв'язання складних обчислювальних задач. Актуальність квантових обчислень зумовлена як теоретичними досягненнями в галузі квантової інформатики, так і практичними потребами сучасного суспільства в обробці величезних масивів даних, криптографічному захисті інформації, моделюванні складних фізичних та хімічних систем, оптимізації логістичних процесів та розв'язанні задач штучного інтелекту.

Концептуальні основи квантових обчислень були закладені в 1980-х роках у роботах Річарда Фейнмана [1] та Пола Беніюффа [2, 3], де автори висловили ідеї про можливість використання квантових систем для ефективного моделювання квантових процесів. Фейнман у своїй піонерській роботі 1982 року показав, що класичні комп'ютери принципово неспроможні ефективно симулювати квантові системи через експоненційне зростання обчислювальної складності, і запропонував використовувати для цього квантові комп'ютери. Пізніше Девід Дойч формалізував концепцію універсального квантового комп'ютера [4], заклавши теоретичний фундамент для подальшого розвитку квантових алгоритмів. Розвиток квантових алгоритмів відбувся завдяки працям таких вчених, як Девід Дойч, Чарльз Беннетт, Деніел Бернштейн, Віджай Вазірані, Пітер Шор та Лов Гровер.

Справжній прорив у розумінні потенціалу квантових обчислень відбувся в 1994 році, коли Пітер Шор розробив квантовий алгоритм факторизації великих чисел [5], який демонструє експоненційне прискорення порівняно з найкращими відомими класичними алгоритмами. Алгоритм Шора має фундаментальне значення для криптографії, оскільки загрожує безпеці широко використовуваних криптосистем з відкритим ключем, таких як RSA, що базуються на складності задачі факторизації. Це відкриття стимулювало інтенсивні дослідження як у

напрямку створення квантових комп'ютерів, так і в розробці постквантової криптографії.

У 1996 році Лов Гровер запропонував квантовий алгоритм пошуку в неупорядкованій базі даних [6], який забезпечує квадратичне прискорення порівняно з класичними методами. Хоча це прискорення менш суттєве, ніж у випадку алгоритму Шора, алгоритм Гровера має широкий спектр застосувань, включаючи оптимізаційні задачі, пошук у базах даних, розв'язання *NP*-повних задач та криптоаналіз симетричних криптосистем. Важливість алгоритму Гровера також полягає в тому, що він є оптимальним для задачі неупорядкованого пошуку, що було доведено Беннетом та іншими дослідниками [7].

Сучасний стан розроблення квантових алгоритмів характеризується інтенсивним розвитком як фундаментальних, так і прикладних напрямків. Класичні квантові алгоритми, такі як алгоритми Дойча [8], Дойча-Йожи [9], Бернштейна-Вазірані [10] та Саймона [11], хоча й мають переважно теоретичне значення, але демонструють фундаментальні переваги квантових обчислень над класичними для певних класів задач. Ці алгоритми слугують важливими навчальними прикладами та ілюструють ключові квантові феномени, такі як квантовий паралелізм та інтерференція, які лежать в основі квантової обчислювальної переваги.

Останнє десятиліття ознаменувалося появою нових класів квантових алгоритмів, адаптованих до можливостей сучасних квантових пристроїв епохи зашумлених квантів (або квантових систем) середнього масштабу (англ. Noisy Intermediate-Scale Quantum, NISQ) [12]. Варіаційні квантові алгоритми, такі як алгоритм варіаційного квантового розв'язувача власних значень (англ. Variational Quantum Eigensolver, VQE) [13] та квантовий алгоритм наближеної оптимізації (англ. Quantum Approximate Optimization Algorithm, QAOA) [14], поєднують квантові та класичні обчислювальні ресурси для розв'язання оптимізаційних задач та знаходження власних значень гамільтоніанів. Ці

гібридні підходи демонструють стійкість до шуму та помилок, що робить їх перспективними для реалізації на сучасних квантових процесорах.

Квантове машинне навчання (англ. Quantum Machine Learning) представляє собою швидко зростаючу галузь, що досліджує можливості використання квантових обчислень для прискорення алгоритмів машинного навчання та розпізнавання образів [15]. Квантові нейронні мережі, квантові методи класифікації та кластеризації, квантові генеративні моделі відкривають нові перспективи для обробки великих даних та розв'язання задач штучного інтелекту. Хоча практична квантова перевага в цій галузі ще не продемонстрована переконливо, теоретичні дослідження вказують на потенційні експоненційні прискорення для певних класів задач.

Квантова криптографія та квантові комунікації становлять окремий важливий напрямок застосування квантових принципів [16]. Протоколи квантового розподілу ключів, такі як BB84 [17] та E91 [18], забезпечують теоретично абсолютну безпеку передачі інформації, гарантовану законами квантової механіки. Квантова телепортація [19] та надщільне кодування [20] демонструють унікальні можливості квантових систем для передачі інформації та створення розподілених квантових мереж.

Застосування квантових алгоритмів у сучасних задачах охоплює широкий спектр галузей. У хімії та матеріалознавстві квантові симулятори використовуються для моделювання молекулярних структур, хімічних реакцій та властивостей нових матеріалів [21]. Фармацевтичні компанії досліджують можливості квантових обчислень для прискорення процесу розробки нових ліків через точне моделювання взаємодії молекул. У фінансовій галузі квантові алгоритми застосовуються для оптимізації фінансових портфелів, оцінки ризиків, ціноутворення опціонів та виявлення шахрайства [22].

Логістика та оптимізація ланцюгів постачання представляють собою природну сферу застосування квантових алгоритмів, оскільки багато задач у цій галузі є *NP*-складними та важко розв'язуються класичними методами [23].

Квантові підходи до задачі комівояжера, маршрутизації транспорту, розподілу ресурсів та планування виробництва демонструють перспективи значного підвищення ефективності. Провідні компанії, такі як Volkswagen, Airbus та DHL, вже проводять експерименти з квантовими алгоритмами для оптимізації своїх виробничих процесів.

У галузі штучного інтелекту та обробки природної мови квантові алгоритми досліджуються для прискорення навчання нейронних мереж, покращення алгоритмів рекомендаційних систем та підвищення точності розпізнавання образів [24]. Квантові методи також знаходять застосування в біоінформатиці для аналізу геномних даних, вирівнювання послідовностей ДНК та моделювання білкових структур.

Сучасні квантові платформи, розроблені компаніями IBM, Amazon, Microsoft, IonQ, Rigetti, Google та іншими, надають дослідникам та розробникам доступ до реальних квантових процесорів через хмарні сервіси [25]. Це прискорює доступ до квантових обчислень та стимулює розвиток квантового програмного забезпечення. Фреймворки квантового програмування, такі як Qiskit [26], Amazon Braket [27], Q# [28], PennyLane [29], Cirq [30], D-Wave [31] та інші, забезпечують інструменти для проєктування, симуляції та виконання квантових алгоритмів.

Незважаючи на значний прогрес, квантові обчислення все ще стикаються з серйозними технічними викликами. Декогеренція, квантові помилки, обмежена кількість кубітів та їх зв'язність, складність масштабування квантових систем залишаються основними перешкодами на шляху до створення повномасштабних універсальних квантових комп'ютерів [32]. Розробка ефективних методів квантової корекції помилок [33] та створення відмовостійких квантових обчислень є критично важливими для реалізації повного потенціалу квантових технологій.

Цей навчальний посібник призначений для магістрів, які прагнуть опанувати фундаментальні принципи та практичні навички в галузі квантових

обчислень, зокрема для аналізу квантових алгоритмів. Посібник систематично викладає математичні основи квантової механіки, основні концепції квантової інформатики та детально розглядає класичні квантові алгоритми, які становлять фундамент сучасної квантової алгоритміки. Особлива увага приділяється практичним аспектам проєктування квантових схем та розумінню принципів роботи квантових алгоритмів.

Структура посібника побудована таким чином, щоб забезпечити послідовне засвоєння матеріалу від базових математичних концепцій до складних квантових алгоритмів. Читач познайомиться з варіантами реалізації квантового програмного забезпечення, зрозуміє фундаментальні відмінності між квантовими та класичними обчисленнями, опанує необхідний математичний апарат та детально вивчить ключові квантові алгоритми, що визначили розвиток галузі квантових обчислень.

1 ПРОБЛЕМИ СТВОРЕННЯ КВАНТОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Вступ до квантових обчислень

В останні десятиліття двадцятого століття вчені прагнули поєднати дві найвпливовіші та революційні теорії століття: теорію інформації та квантову механіку. Їхній успіх породив новий погляд на обчислення та інформацію – квантову теорію інформації. Це назавжди змінило уявлення про обчислення, інформацію та їхній зв'язок з фізикою, а також надихнуло на створення нових застосувань, включно з деякими абсолютно відмінними алгоритмами та протоколами.

Теорія інформації, що включає основи як комп'ютерних наук, так і комунікацій, настільки ефективно абстрагувалася від фізичного світу, що стало можливим обговорювати основні питання в галузі комп'ютерних наук та комунікацій, такі як ефективність алгоритму чи надійність комунікаційного протоколу, не розуміючи деталей фізичних пристроїв, що використовуються для обчислень чи комунікації. Ця здатність ігнорувати базову фізику виявилася надзвичайно корисною, і її успіх можна побачити у повсюдному поширенні обчислювальних та комунікаційних пристроїв навколо нас. Донедавна інформаційні науки міцно корінилися в класичній механіці. Наприклад, машина Тюрінга є класичною механічною моделлю, що діє за суто класичними механічними принципами.

Квантова механіка відігравала все більшу роль у розробленні нових та ефективніших обчислювальних пристроїв. Квантова механіка лежить в основі роботи традиційних, класичних комп'ютерів та комунікаційних пристроїв, від транзистора до лазера та найновіших апаратних досягнень, що збільшують швидкість і потужність та зменшують розмір комп'ютерних та комунікаційних компонентів. Донедавна вплив квантової механіки обмежувався сферою низькорівневої реалізації, він не впливав на те, як думали про обчислення чи комунікацію або як їх вивчали.

На початку 1980-х років кілька дослідників усвідомили, що квантова механіка має несподівані наслідки для обробки інформації. Чарльз Беннетт та Жиль Brassar, спираючись на ідеї Стівена Візнера, показали, як неklasичні властивості квантового вимірювання забезпечують доказово безпечний механізм для створення криптографічного ключа. Річард Фейнман, Пол Беніюфф та інші визнали, що певні квантові явища – явища, пов'язані з так званими заплутаними частинками, – неможливо ефективно симулювати за допомогою машини Тюрінга.

Це спостереження призвело до припущення, що, можливо, ці квантові явища можна використовувати для прискорення обчислень загалом. Така програма вимагала переосмислення інформаційно-теоретичної моделі, що лежить в основі обчислень, виводячи її за межі суто класичної сфери.

Ідею квантового комп'ютера вперше запропонував у 1981 році нобелівський лауреат Річард Фейнман, який зазначив, що точне та ефективне моделювання квантово-механічних систем буде неможливим на класичному комп'ютері, але новий тип машини, комп'ютер, сам «побудований з квантово-механічних елементів, що підкоряються квантово-механічним законам» [1], одного дня зможе виконувати ефективні симуляції квантових систем.

Класичні комп'ютери за своєю суттю не здатні симулювати таку систему з субекспоненційною часовою та просторовою складністю через експоненційне зростання обсягу даних, необхідних для повного представлення квантової системи. Квантові комп'ютери, з іншого боку, використовують унікальні, неklasичні властивості квантових систем, з яких вони побудовані, що дозволяє їм обробляти експоненційно великі обсяги даних лише за поліноміальний час. Елементна база, яка забезпечує виконання обчислень, зазнала стрімкого розвитку у другій половині двадцятого сторіччя (рис. 1.1). На заміну напівпровідників з'являються нові обчислювальні архітектури з відповідною елементною базою, які реалізують новий підхід до обчислень.

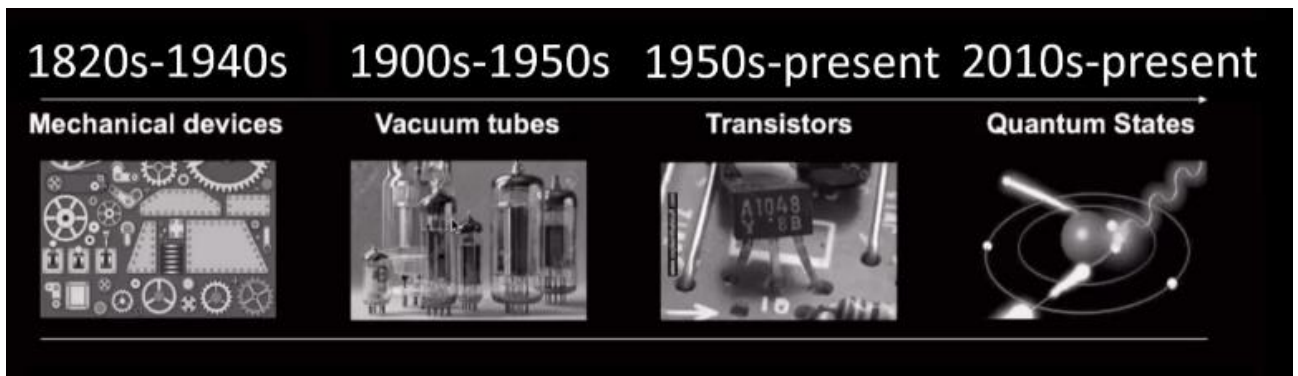


Рис. 1.1 - Еволюція елементної бази обчислень.

У подальшому розгляді матеріалу будемо використовувати таке визначення квантової обробки інформації. **Квантова обробка інформації** – галузь, що включає квантові обчислення, квантову криптографію, квантові комунікації та квантові ігри і досліджує наслідки використання квантової механіки замість класичної для моделювання інформації та її обробки.

Квантові обчислення не означають зміну фізичного субстрату, на якому виконуються обчислення, з класичного на квантовий, а змінюють саме поняття обчислення. Зміна починається на фундаментальному рівні: основною одиницею обчислення є вже не біт, а **квантовий біт** або **кубіт**. Перенесення обчислень на квантово-механічну основу призвело до відкриття швидших алгоритмів, нових криптографічних механізмів та вдосконалених комунікаційних протоколів. Квантові обчислення виконує квантовий комп'ютер, функціональність якого базується на: методі представлення кубітів (наноструктури, атоми, іони, фотони – структури, які можуть бути інтерпретовані як 0 та 1 та бути у стані **суперпозиції**) та методі, що забезпечує їх взаємодію з підтримкою квантових станів.

Аналітики вважають, що міне від 5 до 10 років, перш ніж стануть можливими справжні, корисні квантові обчислення, і більшість очікує, що їхній вплив буде величезним. Багато експериментальних квантових комп'ютерів, що існують сьогодні, базуються на принципах надпровідності та працюють при дуже низькій температурі (наприклад, 10 мілікельвінів, або близько до -273 градусів за Цельсієм). Це має певні практичні обмеження, яких не зустріти у класичних комп'ютерах, що працюють при кімнатній температурі.

Після золотої доби фізики у 1930-х роках настала так звана *друга квантова революція*, очолювана великим британським фізиком Полем Діраком. Він зробив глибокий внесок, знайшовши спосіб описати поведінку електрона в релятивістських термінах за допомогою свого великого рівняння. Дірак не тільки запропонував релятивістський опис електрона, але й надав базовий підхід для квантування інших властивостей оточуючого всесвіту, таких як маса, заряд, положення та енергія.

У наступні роки ці ідеї стали основою для квантування всіх субатомних сил у природі, таких як слабкі, сильні ядерні сили та електромагнетизм. Це приклади того, що зараз називається *квантовою теорією поля* (КТП): основи сучасної фізики.

Основи КТП прості, але такі ж дивні, як і теорія, від якої вона відгалужується:

- у субатомному світі поля охоплюють увесь всесвіт. Існує поле для кожної частинки в *стандартній моделі*: електронів, протонів, кварків, фотонів тощо;
- частинки, які усім відомі, є лише локалізованими коливаннями у своїх відповідних полях;
- поля можуть взаємодіяти одне з одним, щоб пояснити, як частинки створюються та знищуються.

Щоб зрозуміти взаємодію між частинками, розгляньте рисунок, що показує поле електрона (рис. 1.2). Електрон є локалізованим коливанням у цьому полі (вгорі). Якщо електрон випромінює фотон, то КТП стверджує, що частина енергії електрона створює інше коливання у фотонному полі (внизу), яке потім віддаляється.

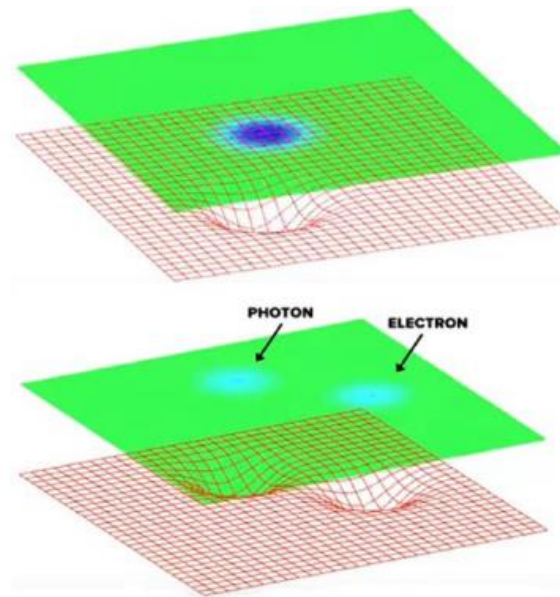


Рис. 1.2 – Базовий опис квантового поля електрона (зверху) та електрона, що випромінює фотон (знизу).

КТП бере ідеї з квантової механіки, такі як потік енергії у вигляді дискретних порцій (квантів), і поєднує їх з ідеєю полів Фарадея, які є неперервними, гладкими об'єктами, що коливаються в просторі. Це спадщина Фарадея: у нашому всесвіті немає частинок, є лише поля та коливання, які вони створюють. Отже, КТП здається простим поєднанням двох старих принципів у сучасному описі реальності. Тим не менш, математичний апарат настільки складний, що розв'язання більшості рівнянь у КТП вимагає потужностей суперкомп'ютерів.

Фраза «квантові обчислення» не є аналогічною фразам «ДНК-обчислення» чи «оптичні обчислення»: вони описують систему, на якій виконуються обчислення, не змінюючи власне поняття обчислення.

Класичні комп'ютери використовують квантову механіку, але вони обчислюють за допомогою бітів, а не кубітів. З цієї причини вони не вважаються квантовими комп'ютерами. Квантовий або класичний комп'ютер може бути, або не бути оптичним комп'ютером, залежно від того, чи використовуються оптичні пристрої для виконання обчислень.

Чи є комп'ютер квантовим, чи класичним, залежить від того, чи представлена та обробляється інформація квантовим чи класичним способом.

Поняття «квантові обчислення» за своїм характером ближче до «аналогових обчислень», оскільки обчислювальна модель для аналогових обчислень відрізняється від стандартної: допускається континуум значень, а не лише дискретний набір.

Ці дві моделі значно відрізняються тим, що аналогові обчислення не підтримують **заплутаність**, ключовий ресурс для квантових обчислень, а вимірювання реєстрів квантового комп'ютера можуть дати лише невеликий, дискретний набір значень.

Крім того, хоча кубіт може приймати континуум значень, у багатьох відношеннях кубіт більше схожий на біт з його двома дискретними значеннями, ніж на аналогове обчислення. Наприклад, як буде показано далі, з кубіта шляхом вимірювання можна отримати лише один біт інформації.

Галузь квантової обробки інформації повільно розвивалася у 1980-х та на початку 1990-х років, коли невелика група дослідників розробляла теорію квантової інформації та квантової обробки інформації. Девід Дойч розробив поняття квантово-механічної машини Тюрінга. Деніел Бернштейн, Віджай Вазірані та Ендрю Яо вдосконалили його модель і показали, що квантова машина Тюрінга може симулювати класичну машину Тюрінга, а отже, будь-яке класичне обчислення, з уповільненням не більше ніж у поліноміальний час.

Потім була визначена стандартна модель квантових схем, що призвело до розуміння квантової складності з точки зору набору базових квантових перетворень, які називаються **квантовими вентилями** (*також застосовуються інші назви – спінові матриці Паулі, гейти*). Ці вентиля є теоретичними конструкціями, які можуть мати, або не мати прямих аналогів у фізичних компонентах реального квантового комп'ютера.

На початку 1990-х років дослідники розробили перші справді квантові алгоритми. Незважаючи на ймовірнісну природу квантової механіки, перші квантові алгоритми, для яких можна було довести перевагу над класичними алгоритмами, дають правильну відповідь з упевненістю. Вони перевершують

класичні алгоритми, розв'язуючи за поліноміальний час з упевненістю задачу, яку можна розв'язати за поліноміальний час лише з високою ймовірністю за допомогою класичних методів.

Такий результат не має прямого практичного застосування та інтересу, оскільки неможливість створення ідеальної машини зводить будь-яку практичну машину, що виконує будь-який алгоритм, до розв'язання задачі лише з високою ймовірністю. Але такі результати мали високий теоретичний інтерес, оскільки вони вперше показали, що квантові обчислення теоретично потужніші за класичні для певних обчислювальних задач.

Дойч у 1985 році запропонував одночасно обчислювати одnobітну булеву функцію у двох точках, використовуючи суперпозицію нулів та одиниць, що відомо як **квантовий паралелізм**. На той час бракувало основи для створення нових алгоритмів, яка була закладена лише у 1989 році, коли Дойч опублікував статтю про квантові схеми, представивши квантові вентиля, що замінюють добре відомі класичні вентиля, такі як AND, OR, NOT.

У 1992 році галузь квантових алгоритмів значно розвинулась, коли Дойч та Йожа створили алгоритм для визначення, чи є булева функція збалансованою чи константною. Цей алгоритм стимулював розробку **алгоритмів на основі оракула**. Задана певна функція, і необхідно розкрити її приховану властивість. Функцію можна обчислювати скільки завгодно разів, але мета полягає в тому, щоб знайти властивість, зробивши якомога менше запитів на обчислення.

Бернштейн та Вазірані у 1993 році зазначили, що алгоритм Дойча-Йожі можна використовувати для знаходження конкретної булевої функції в наборі лінійних булевих функцій. Алгоритм Бернштейна-Вазірані швидший за свій класичний аналог, при цьому не використовується **заплутаність**. Потужність алгоритму забезпечується лише *квантовим паралелізмом*.

Саймон у 1994 році опублікував квантовий алгоритм, який експоненційно швидше за класичні алгоритми для визначення, чи є функція взаємно однозначною (один-до-одного) чи два-до-одного. У цьому випадку доводиться

розширювати тип функції, щоб мати вихід з багатьма бітами. Цей алгоритм використовує максимальну *заплутаність*. Задача Саймона має цікаве застосування для знаходження прихованої підгрупи спеціального класу груп.

Шор у 1994 році створив два знамениті квантові алгоритми для факторизації складених цілих чисел та обчислення дискретних логарифмів, які мають сильний вплив на злам криптографічних методів, що використовуються у повсякденному житті людини. Алгоритми Шора допомогли вивести квантові обчислення в центр уваги. З того часу ця галузь стрімко зростає. Алгоритм Шора також можна сформулювати як алгоритм на основі оракула з періодичною функцією [34]. Мета полягає в тому, щоб знайти період, обчисливши функцію якомога менше разів. Для знаходження періоду зазвичай використовується перетворення Фур'є, яке є корисним у класичних алгоритмах, незважаючи на складність $O(N \log N)$, де N – розмір даних. У квантовому випадку перетворення Фур'є можна реалізувати за допомогою $O(\log^2 N)$ універсальних вентилів, а основний вплив на роботу алгоритму робить *квантова суперпозиція*.

Алгоритм Шора став важливим відкриттям не тільки тому, що він забезпечує експоненційне прискорення порівняно з найшвидшими класичними алгоритмами, але й тому, що низка алгоритмів криптографії з відкритим ключем, включаючи широко використовуваний алгоритм RSA, залежать від того факту, що не існує відомого ефективного класичного алгоритму для розкладання цілих чисел на прості множники [35].

Шор продемонстрував, що реалізація повномасштабного квантового комп'ютера матиме потенціал для справді значного збільшення швидкості обчислень, водночас вказуючи на можливі наслідки такого зростання обчислювальної потужності для галузі кібербезпеки. З обох цих причин відкриття Шора викликало значний інтерес до розробки квантових алгоритмів та комп'ютерів, який зберігається і донині. Довгий час шукали класичне рішення за поліноміальний час, настільки, що світ почувався достатньо впевненим у відсутності такого рішення, що багато протоколів безпеки, включно з широко

використовуваним алгоритмом RSA, повністю базують свою безпеку на обчислювальній складності цієї задачі.

Невідомо, чи існує ефективне класичне рішення, тому результат Шора не доводить, що квантові комп'ютери можуть розв'язати задачу ефективніше, ніж класичний комп'ютер. Але навіть у малоймовірному випадку, якщо буде знайдено поліноміальний класичний алгоритм для цієї задачі, це буде свідченням елегантності та ефективності точки зору квантової теорії інформації, що квантовий алгоритм, незважаючи на всі неінтуїтивні аспекти квантової механіки, було легше знайти.

Хоча результат Шора викликав великий інтерес до цієї галузі, сумніви щодо його практичної значущості залишалися. Квантові системи є надзвичайно крихкими. Ключові властивості, такі як *квантова запутаність*, легко порушуються впливом навколишнього середовища, що призводить до декогеренції квантових станів.

Властивості квантової механіки, такі як неможливість надійного копіювання невідомого квантового стану, робили малоймовірним, що коли-небудь будуть знайдені ефективні методи виправлення помилок для квантових обчислень. З цих причин здавалося малоймовірним, що надійні квантові комп'ютери можуть бути побудовані.

На щастя, незважаючи на серйозні та поширені сумніви щодо того, чи зможе квантова обробка інформації коли-небудь стати практичною, сама теорія виявилася настільки привабливою, що дослідники продовжували її вивчати. В результаті, у 1996 році Шор та Роберт Калдербанк, а незалежно від них Ендрю Стін, винайшли спосіб обійти, здавалося б, непереборні проблеми квантової механіки для розробки методів *квантового виправлення помилок*. Сьогодні квантове виправлення помилок, мабуть, є найзрілішою областю квантової обробки інформації.

Гровер зосередився на неупорядкованих базах даних і в 1996 році створив квантовий алгоритм, який знаходить елемент квадратично швидше, ніж

класичний пошук. Алгоритм Гровера також можна сформулювати як алгоритм на основі *оракула з булевою функцією*, яка є константною, за винятком однієї точки в області визначення [6]. Мета полягає в тому, щоб знайти цю точку, обчисливши функцію якомога менше разів. Коли він записаний як алгоритм «чорної скриньки», стає зрозуміло, що алгоритм Гровера має широке застосування.

Історія квантових обчислень в роках:

- 1924, термін «квантова механіка» використаний М. Борном;
- 1925, формулювання матричної механіки Гейзенбергом, Борном, Йорданом;
- 1925-1927, копенгагенська інтерпретація;
- 1930, «Принципи квантової механіки» Дірака;
- 1935, ідеї Ейнштейна, Подольського та Розена;
- 1935, «Квантова заплутаність» та кіт Шредінгера, Шредінгер та Ейнштейн;
- 1947, «Моторшна дія на відстані» у листі А. Ейнштейна до М. Борна;
- 1976, спроба створити квантову теорію інформації;
- 1980, квантово-механічна модель машини Тюрінга запропонована Беніюффом;
- 1981, «Моделювання фізики за допомогою комп'ютерів», Фейнман;
- 1985, квантова машина Тюрінга запропонована Дойчем;
- 1992, алгоритм Дойча-Йожи;
- 1993, перша стаття про квантову телепортацію;
- 1994, алгоритм факторизації Шора (пор. шифрування RSA);
- 1996, алгоритм пошуку Гровера;
- 2004, перше п'ятифотонне заплутування, Китай;

- 2011, перший комерційно доступний квантовий комп'ютер (D-Wave);
- 2017, перша квантова телепортація незалежного однофотонного кубіта (14 км), Китай;
- 2018, Закон США про національну квантову ініціативу;
- 2019, квантова перевага Google;
- 2022, Нобелівська премія (Aspect, Clauser, Zeilinger) за порушення нерівності Белла;
- 2022, використано 433 кубіти, IBM;
- 2023, премія «Прорив» 2023 року (Беннет, Brassard, Шор, Дойч).

З кожним роком збільшується кількість галузей застосування квантових обчислень та технологій:

- квантове моделювання;
- криптографія;
- математика (факторизація, програма прихованих підгруп, дискретна логарифмічна задача);
- оптимізація;
- алгоритм пошуку;
- квантове машинне навчання;
- переваги квантового моделювання (покращене навчання з невеликою кількістю даних, швидша збіжність, менша кількість параметрів).

Наскільки практичними виявляться квантові обчислення та квантова інформація, досі невідомо. Немає відомих фундаментальних фізичних принципів, які б забороняли створення великомасштабних та надійних квантових комп'ютерів. Однак інженерні проблеми залишаються. Лабораторні експерименти продемонстрували квантові обчислення з десятками квантових бітів, що виконують сотні квантових операцій.

Теоретики та експериментатори по всьому світу досліджують багато перспективних підходів, але ще залишається невизначеність щодо того, як, коли, і чи взагалі буде побудований квантовий комп'ютер, здатний виконувати загальні квантові обчислення на сотнях/тисячах кубітів.

Нещодавно Microsoft представило новаторські досягнення в квантових обчисленнях, які базуються на ключовій ролі нульових мод Майорани [36]. Ці екзотичні частинки, які є власними античастинками, можуть бути сплетені для нелокального кодування інформації та будуть використовуватись в певних топологічних надпровідниках. Їхня притаманна стабільність робить їх дуже ймовірним кандидатом для створення топологічних кубітів, які стійкі до помилок навколишнього середовища, що забезпечує значну перевагу над звичайними кубітами.

Компанія демонструє створення та маніпулювання цими невловимими нульовими модами Майорани, охоплюючи теоретичні основи нульових мод Майорани, їх експериментальну перевірку та архітектурні стратегії, що досліджуються для побудови відмовостійкого квантового комп'ютера (рис. 1.3). Кінцева мета полягає в масштабуванні квантових комп'ютерів до архітектур з мільйонами кубітів, що дозволить розв'язувати складні задачі, які наразі недоступні класичним суперкомп'ютерам, значно зменшити витрати на виправлення помилок та отримати квантову перевагу.

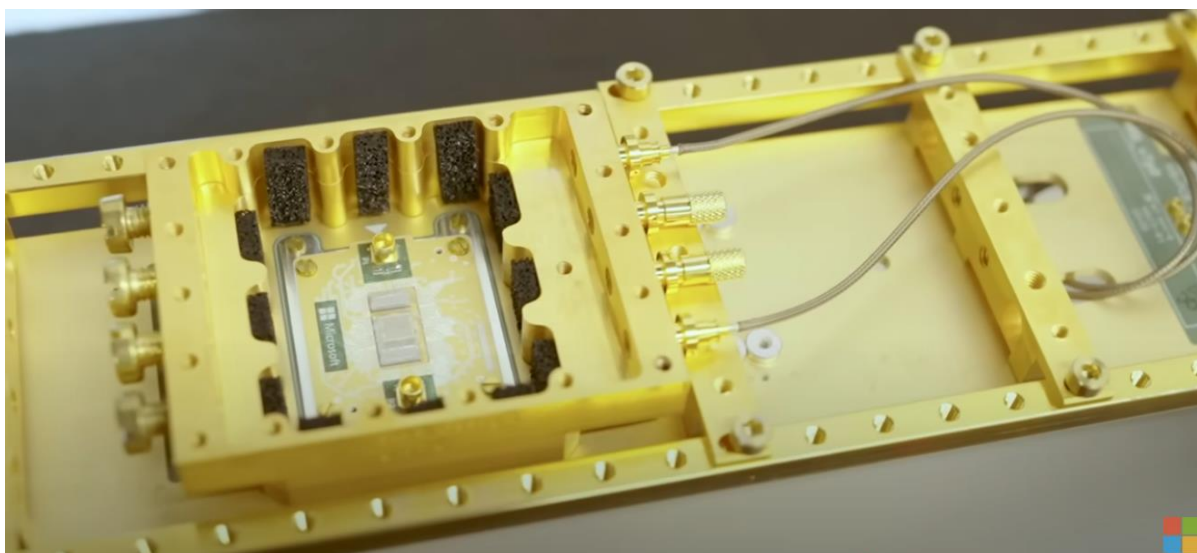


Рис. 1.3 – Реалізація технології Majorana (Microsoft) [35].

Це дійсно проривне досягнення дозволить прискорити використання квантових обчислень у таких галузях, як машинне навчання, розробка ліків та матеріалознавство.

Але для переходу від лабораторних зразків до стабільних топологічних кубітів потрібен прогрес у розробці *епітаксіальних* матеріалів та *гетероструктур* нанодротів/надпровідників, у виготовленні відтворюваних пристроїв, високоточному управлінні та зчитуванні, а також кріогенній електроніці. «Шлях до мільйона кубітів» підкреслює комплексну дорожню карту досліджень та розробок Microsoft на шляху до наступної ери квантових обчислень з використанням стійких до помилок квантових процесорів.

Квантові обчислення являють собою зміну парадигми в обчисленнях, прокладаючи шлях за межі можливостей, властивих класичним системам. Маючи глибоко вкорінені принципи квантової механіки, квантові обчислення пропонують нові методи розв'язання складних задач, які наразі є нерозв'язними для класичних комп'ютерів. Поява квантових алгоритмів, що використовують такі властивості, як *суперпозиція*, *заплутаність* та *інтерференція*, надає потенціал для експоненційного прискорення в певних обчислювальних завданнях.

1.2 Програмне забезпечення для симуляції квантових обчислень

Згідно з нещодавнім оглядом Гарвала та ін. [37] та підручником [38], існуючі мови програмування дотримуються або імперативної парадигми (наприклад, QCL [39], Silq [40] або Q# [28]), або функціональної парадигми (наприклад, QPL [41], Quipper [42] або LIQUi|> [43]). Хоча вони відрізняються своїми можливостями та ступенем абстракції, їхні квантові особливості зосереджені на генерації квантових схем, які в кінцевому підсумку застосовують списки операторів до квантових станів. Ті ж самі спостереження можна зробити і щодо комерційних підходів (наприклад, Amazon Braket, Qiskit або Ocean) до проблеми квантового програмування, де інструкції щодо генерації квантових схем вбудовані в мову-хост, зазвичай Python.

Хоча дослідження в галузі інженерії програмного забезпечення встановили багато методів, технік та процесів, спрямованих на системне створення високоякісних програмних артефактів, деякі з елементарних опцій розробки, такі як **налагодження (debugging)**, **трасування (tracing)** та деякі варіанти **тестування**, не можуть бути безпосередньо застосовані у значущий спосіб у квантовій сфері.

Враховуючи таке скорочення інженерних можливостей, перевірка якості (або коректності) квантових програм повинна більше зосереджуватися на інших методах, таких як формальна верифікація. Це, у свою чергу, вимагає засобів для належної формалізації квантових програм. Починаючи з основоположної роботи [26], проводилися дослідження численних підходів до наділення квантових мов програмування формальною семантикою, на основі якої може здійснюватися верифікація. Проте, поточний стан справ все ще відстає від класичного рівня зрілості.

Від інженерів-програмістів, що працюють над квантовими обчисленнями, не варто вимагати знання всіх тонкощів квантової фізики, але розумне усвідомлення основних принципів, методів та формалізмів є важливим для визначення ефективних рівнів абстракції. Ситуація схожа на ту, що була на зорі інженерії програмного забезпечення як дисципліни: тоді стала очевидною потреба у використанні структурованих інженерних підходів для створення програмного забезпечення, проте низькорівневі деталі залишалися вирішальними – як це втілено, наприклад, у таких дослідницьких темах, як створення ефективних компіляторів для виразних високорівневих мов, які, тим не менш, враховували дуже відмінні апаратні властивості тогочасних систем.

Автори [44] вважають, що погляд на квантові обчислення, заснований на **супероператорах**, є вдалою відправною точкою для виведення таких надійних, практичних та корисних абстракцій з використанням усталених методів комп'ютерних наук для квантового програмування та квантової інженерії програмного забезпечення.

2 ПОРІВНЯННЯ РОБОТИ КВАНТОВОГО КОМП'ЮТЕРА ТА КЛАСИЧНОГО КОМП'ЮТЕРА

Зіставлення *класичних* та *квантових* обчислень уособлює одну з найглибших відмінностей в еволюції обчислювальних технологій. Класичні обчислення, що лежать в основі більшості сучасних електронних пристроїв, базуються на *детермінованій поведінці* класичної фізики. На противагу цьому, квантові обчислення (які не є просто поступовим вдосконаленням класичних обчислень) використовують імовірнісні та неінтуїтивні принципи квантової механіки, забезпечуючи потенційно *експоненційне прискорення* для певних класів задач.

У класичних обчисленнях інформація обробляється за допомогою двійкових розрядів або *бітів*, які фізично представлені різними станами фізичної системи, зазвичай електронної, наприклад, рівнями напруги на транзисторі (рис. 2.1). Ці біти можуть існувати в одному з двох станів: 0 або 1. Схематично, класичні обчислення можна представити такою формулою: $\{0,1\}^n \rightarrow \{0,1\}^m, m \leq n$.

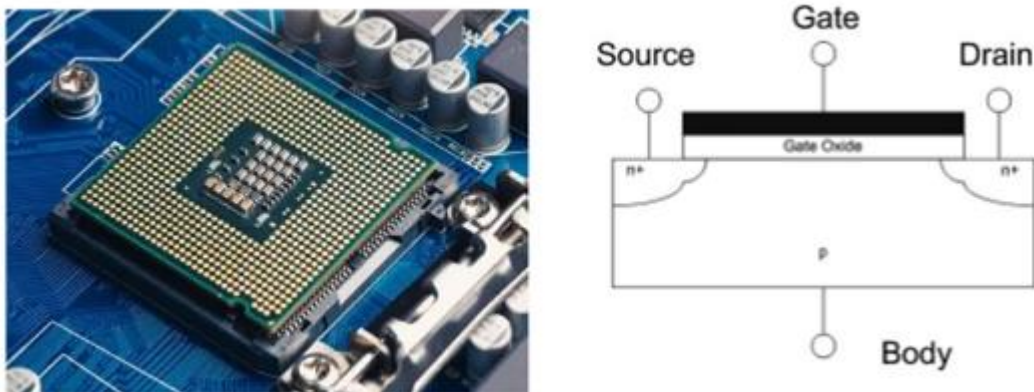


Рис. 2.1 – Елементна база класичних обчислень [45].

Класичні логічні вентиля, такі як AND (І), OR (АБО), NOT (НЕ/НІ), NAND (НЕ-І), NOR (НЕ-АБО), XOR (виключне АБО) та XNOR (виключне НЕ-АБО), обробляють сигнали/біти для виконання обчислень. Наприклад, вентиль NAND видає 1 лише тоді, коли один або обидва його входи дорівнюють 0 (рис. 2.2). Цей детермінований процес, заснований на булевій алгебрі, гарантує, що за

однакових вхідних даних класичний комп'ютер завжди видаватиме однакові вихідні дані.

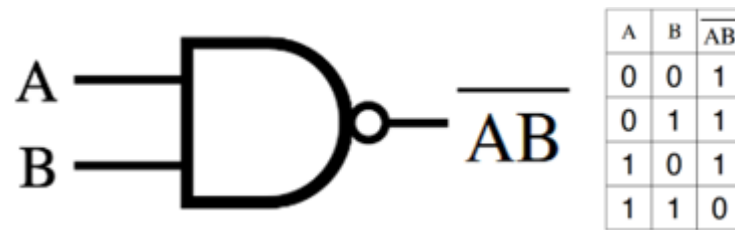


Рис. 2.2 – Таблиця істинності та вентиль для логічного елемента НЕ-І.

Основні відмінності квантових обчислень від класичних схематично показані на рис. 2.3. При цьому, стан комірки у класичних обчисленнях обмежений двома дискретними значеннями, а стан комірки (кубіта) у квантових обчисленнях може приймати неперервне значення (наприклад, у термінах координат на сфері Блоха) у процесі роботи схеми і тільки після зчитування (виміру, спостереження) опиняється у стані 0 або 1. Логіку виконання обчислень можна представити, для класичних обчислень, у вигляді алгоритмів або комбінаційних схем, у той час як для квантових обчислень логіку виконання представляють квантовими схемами (квантовими алгоритмами), які складаються з квантових вентилів (в літературі інколи називають гейтами).

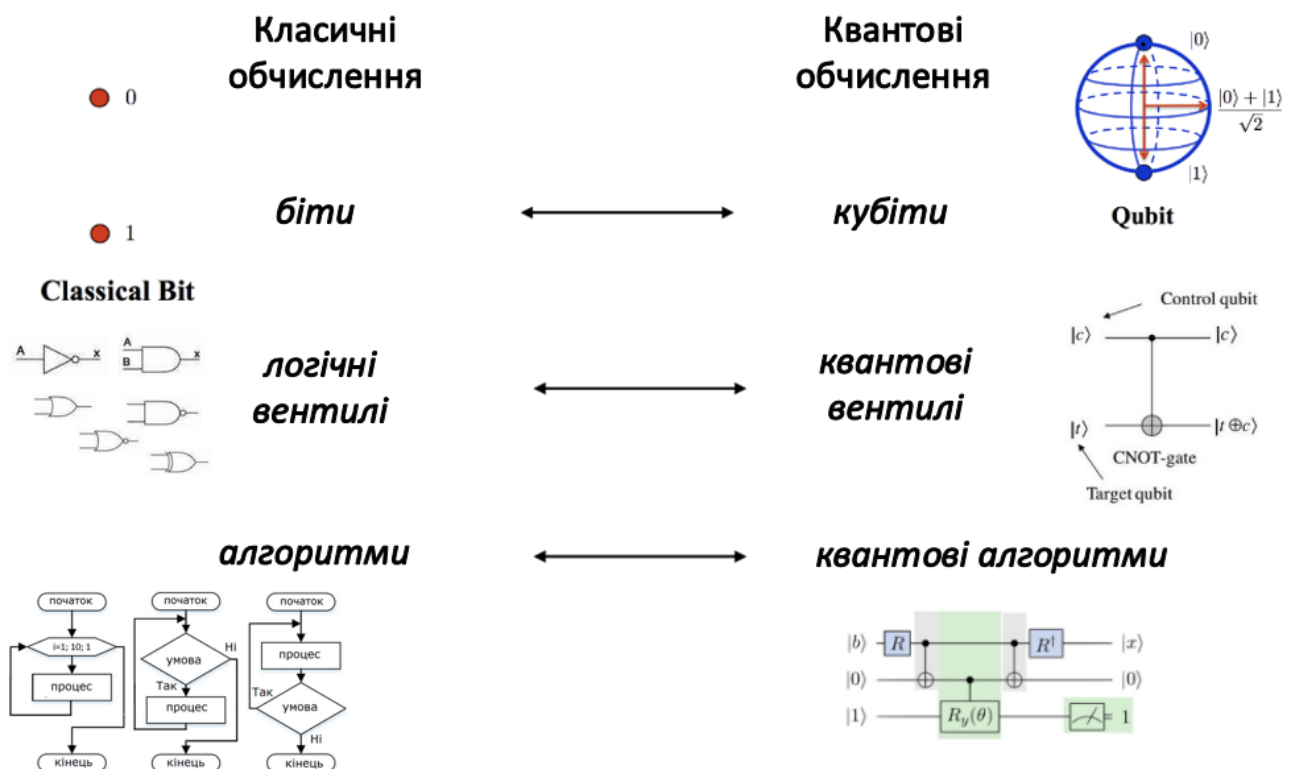


Рис. 2.3 – Схематичне порівняння квантових та класичних обчислень.

У будь-який момент виконання класичного алгоритму кожен біт перебуває в конкретному стані: 0 або 1. Як наслідок, байт у будь-який момент часу також перебуває в конкретному стані. Кожен з 8 бітів у байті є або 0, або 1. Розмір пам'яті комп'ютера виражається як кількість бітів, до яких може отримати доступ процесор. Обсяг пам'яті є одним з основних факторів, що впливають на якість та продуктивність класичних комп'ютерів. Кількість бітів є важливою характеристикою для опису довжини інструкцій процесора та точності чисел. (Наприклад, тип `long` у Java має 64 біти, тоді як `int` у Java – 32 біти.)

Представлення натурального числа, x_i – біт, $x_{n-1} \neq 0$. Двійкове представлення бітового рядку довжиною n : $\{0,1\}^n$. Наприклад, для 10010 це число $16+2=18$, яке можна отримати з використанням загальної формули $2^{n-1}x_{n-1} + \dots + 2x_1 + x_0$. Кожен біт у послідовності вказує, чи слід додавати відповідний ступінь двійки до десяткового числа. Коли біт дорівнює 1, відповідний степінь двійки додається, коли біт дорівнює 0, він не додається. Кажуть, що крайній правий біт послідовності має індекс 0, біт ліворуч від нього має індекс 1 і так далі. Загалом, біт з індексом i відповідає значенню 2^i .

Вважається, що для спеціалістів з інженерії програмного забезпечення, зацікавлених у галузі квантових обчислень, глибоке знання квантової фізики не є обов'язковою умовою для розуміння квантових алгоритмів, так само як не потрібно знати, як створити центральний процесор, щоб розробляти класичні алгоритми. Однак, все ж важливо бути знайомим з основними поняттями, що відрізняють квантово-механічні системи від класичних, щоб отримати краще інтуїтивне розуміння математики квантових обчислень, а також самих алгоритмів.

2.1 Відмінності квантових та класичних обчислень, кубіти

Перша відмінна риса квантової системи відома як *суперпозиція*, або, більш формально, *принцип суперпозиції квантової механіки*.

Замість того, щоб існувати в одному конкретному стані в кожен момент часу, квантова система насправді перебуває у всіх своїх можливих станах одночасно. Стосовно квантового комп'ютера це означає, що квантовий регістр існує в суперпозиції всіх можливих конфігурацій 0 і 1 одночасно, на відміну від класичної системи, регістр якої в будь-який момент часу містить лише одне значення. Лише коли стан *квантової системи* (регістра) спостерігається, тобто значення кубіту вимірюється, вона переходить (колапсує, англ. Collapses) в спостережуваний, визначений класичний стан.

Все ж можливо проводити обчислення, використовуючи таку, на перший погляд, некеровану систему, оскільки кожному з можливих станів квантової системи можна присвоїти ймовірності. Отже, квантова система є *ймовірнісною*: існує обчислювана ймовірність, що відповідає шансу спостереження будь-якого даного стану, якщо стан системи вимірюється. Квантове обчислення виконується шляхом збільшення ймовірності спостереження «правильного» стану до достатньо високого значення, щоб правильну відповідь можна було знайти з розумним ступенем впевненості.

Квантові системи також можуть демонструвати **другу відмінність** – *квантову запутаність*. Стан вважається заплутаним, якщо його неможливо розкласти на більш фундаментальні частини.

Іншими словами, два окремі елементи квантової системи (кубіти) є заплутаними, якщо одну частину неможливо описати, не враховуючи стан іншої. У квантовому комп'ютері ймовірність спостереження певної конфігурації двох кубітів може залежати від ймовірності спостереження іншої можливої конфігурації цих кубітів, і неможливо описати ймовірність спостереження однієї конфігурації, не враховуючи іншу.

Особливо цікавою властивістю квантової запутаності є те, що елементи квантової системи можуть бути заплутаними, навіть коли вони розділені значним простором. Точна фізика квантової запутаності залишається

невловимою навіть для професіоналів у цій галузі, але це не завадило їм застосовувати заплутаність у квантовій теорії інформації.

Квантова телепортація, третя відмінність, – важлива концепція в галузі квантової криптографії, яка базується на заплутаних квантових станах для достатньо точної передачі квантової інформації на відносно великі відстані. У 2025 році дослідження квантової телепортації показали прогрес у різних галузях: між віддаленими надпровідними чіпами та на відстані міста.

Демонстрація квантової телепортації між віддаленими надпровідними чіпами. Дослідники продемонстрували детерміновану телепортацію квантових станів та заплутаних кубітів між віддаленими надпровідними чіпами, з'єднаними 64-метровою кабельною шиною. Це досягнення, опубліковане в січні 2025 року, є значним для масштабування квантових обчислень через розподілені мережі [46].

Демонстрація квантової телепортації на відстані у великих містах. Показана життєздатність точної квантової телепортації на «великі відстані» та «відстані в межах міста» з використанням оптичних гібридних станів. Дослідники продемонстрували успішну телепортацію через оптоволоконні мережі в міському середовищі, хоча конкретні відстані не деталізовані в доступних джерелах.

Хоча абсолютні рекорди відстані з експериментів на основі супутників (понад 1200 км) були встановлені в попередні роки, у 2025 році дослідники більше зосередилися на практичній реалізації та інтеграції інфраструктури, ніж на чистих рекордах відстані. Акцент змістився на те, щоб квантова телепортація надійно працювала в реальних мережевих умовах та покращила основні квантові стани, що використовуються для телепортації.

Варто зазначити, що попередні рекорди квантової телепортації, такі як 143 кілометри, були встановлені в попередні роки (наприклад, у 2012 та 2015 роках) з використанням різних методів, часто з використанням вільного простору або волоконно-оптичних з'єднань [47, 48, 49].

Фундаментальними одиницями інформації в квантових обчисленнях є *кубіти*, які використовують *суперпозицію*, щоб існувати в кількох станах одночасно. Подібно до біта, кубіт може (після його вимірювання) містити значення 0 і 1. Але на відміну від біта, кубіт може також містити значення, що є *комбінаціями станів 0 і 1*. У такому випадку кубіт перебуває в так званому стані *суперпозиції*.

Хоча на перший погляд це може здатися нелогічним, саме це відбувається в природі з низкою елементарних частинок, і це безпосередньо пов'язано з основними ідеями квантової механіки. Той факт, що цей стан суперпозиції зустрічається в природі у елементарних частинок, є гарною ознакою того, що створення квантових комп'ютерів є реалістичним. Класичні комп'ютери ігнорують ці квантові ефекти, тому класичне обладнання не можна нескінченно зменшувати, не досягнувши меж, де починають впливати квантові ефекти.

Коли кубіт вимірюється, він повертає або 0, або 1, а не щось середнє. Зв'язок між станом суперпозиції кубіта та фактичним значенням при вимірюванні пояснюється далі в посібнику. Отже, суперпозиція пов'язана з тим, наскільки ймовірно, що даний кубіт при вимірюванні матиме значення 0 або 1.

Після вимірювання значення кубітів повертається до 0 або 1. Хоча ймовірність вимірювання кубіта з індексом 1 як 1 є високою, у наступному ілюстраційному прикладі результат вимірювання дорівнює 0 (рис. 2.4). Це зроблено, щоб проілюструвати, що зв'язок між ймовірностями та вимірюванням полягає не лише в округленні.

Наразі найважливішим є те, що, як наслідок, певна кількість кубітів може містити більше інформації, ніж та ж кількість класичних бітів, оскільки один кубіт містить інформацію, складнішу за 0 або 1. Це важливо для задач або алгоритмів, які теоретично вимагають експоненційно більше бітів для лінійно зростаючої складності (або зростаючої кількості вхідних даних).

Крім того, *квантова запутаність* дозволяє кубітам стати взаємопов'язаними таким чином, що стан одного (незалежно від того, наскільки він віддалений) миттєво впливає на стан іншого, що є основою для таких квантових протоколів, як *телепортація* та *виправлення помилок*. Класичні системи не мають таких можливостей, оскільки класична інформація є локалізованою.

Розглянемо **виправлення помилок**. Один з підходів до виправлення помилок покладається на надлишковість: зазвичай дані копіюються кілька разів, і будь-які невідповідності через помилки вирішуються шляхом голосування більшості.

Квантове виправлення помилок є набагато складнішим через **теорему про неможливість клонування**, яка забороняє створення ідентичних копій довільного невідомого квантового стану.

Квантове виправлення помилок включає підготовку заплутаних станів, які розподіляють квантову інформацію між кількома кубітами, дозволяючи виявляти та виправляти довільні помилки на окремих кубітах без прямого вимірювання квантового стану.

Ще один різкий контраст полягає в *обчислювальній складності*. Класичні комп'ютери виконують алгоритми розв'язання поліноміальних (P) задач, або NP -повних (недетерміновані поліноміальні повні) задач, за допомогою явних покрокових наборів інструкцій, що призводить до послідовної обробки.

Клас P охоплює задачі, розв'язувані детермінованою машиною Тюрінга за поліноміальний час, тоді як NP -повні представляють задачі, для яких розв'язки можна перевірити за поліноміальний час, але знаходження розв'язків не обов'язково відбувається за поліноміальний час.

Квантові обчислення розв'язують певні задачі з меншою складністю, використовуючи квантові алгоритми, такі як **алгоритм Шора**, який розкладає цілі числа на множники за поліноміальний час і має складність приблизно $\tilde{O}(n^2)$

або $O((\log N)^2 \log(\log N) \log(\log(\log N)))$, де n – кількість біт числа N , що є експоненційним покращенням порівняно з найкращими відомими класичними алгоритмами $O(e^{\log N^{\frac{1}{3}} (\log(\log N))^{\frac{2}{3}}})$.

Більше того, **алгоритм Гровера** виконує задачі неструктурованого пошуку квадратично швидше, ніж його класичні аналоги. При пошуку в невідсортованих базах даних алгоритм Гровера знаходить елемент за $O(\sqrt{N})$ операцій порівняно з $O(N)$ для класичних методів, де N – кількість елементів в наборі. Результатом роботи алгоритму на рис. 2.6 є гістограма з піком у позначеному стані $|11\rangle$, що демонструє розширені можливості пошуку.

```
from qiskit import Aer, QuantumCircuit, transpile, assemble
from qiskit.visualization import plot_histogram
nqubits = 2
grover_circuit = QuantumCircuit(nqubits)
# Ініціалізація кубітів для суперпозиції
grover_circuit.h(range(nqubits))
# Позначте стан |11> (оракул)
grover_circuit.cz(0, 1)
# Інверсія відносно середнього значення (оператор дифузії Гровера)
grover_circuit.h(range(nqubits))
grover_circuit.x(range(nqubits))
grover_circuit.cz(0, 1)
grover_circuit.x(range(nqubits))
grover_circuit.h(range(nqubits))
# Виміряти кубіти
grover_circuit.measure_all()
# Виконайте схему
backend = Aer.get_backend('qasm_simulator')
qobj = assemble(grover_circuit)
results = backend.run(qobj).result()
counts = results.get_counts()
# Вивести результати
plot_histogram(counts)
```

Рис. 2.6 – Ілюстрація роботи алгоритму Гровера.

Незважаючи на ці відмінності, квантові обчислення не є універсальною заміною класичним обчисленням. Скоріше, вони є взаємодоповнюючими. Класичні комп'ютери справляються з лінійними операціями, маючи зрілу технологію, що підтримує надійну, безпомилкову реалізацію великих обсягів обчислень, тоді як квантові комп'ютери створюються для конкретних завдань, де квантове прискорення дає значні переваги.

Квантові комп'ютери об'єднуються з класичними в *гібридних архітектурах*, де квантові процесори вирішують завдання, що підходять для квантових обчислень, передаючи результати класичним процесорам для подальшої обробки. Ця синергія знаходить своє відображення у варіаційних алгоритмах, де квантові та класичні обчислення розв'язують задачі оптимізації.

Сучасні квантові обчислення стикаються з такими проблемами, як *декогеренція* та *рівень помилок кубітів*. Сучасні квантові пристрої, хоча й демонструють *квантову перевагу* для певних задач, залишаються вразливими до впливу навколишнього середовища та обмежених ресурсів, що стримує їхню *відмовостійкість* та *масштабованість*. Ці обмеження підкреслюють необхідність продовження досліджень та експериментів у галузі квантового виправлення помилок, стратегій зменшення шуму та розробки надійного квантового апаратного забезпечення, що подовжує *час когерентності* та знижує *рівень помилок вентилів*.

Сформулюємо декілька важливих тез, які стосуються перспектив використання квантових обчислень:

- не слід думати, що квантові обчислення «виправляють» усе;
- квантові обчислення фундаментально відрізняються від класичних обчислень;
- квантові обчислення переважно підходять для складних задач;
- квантові обчислення та класичні комп'ютери повинні будуть працювати разом;
- апаратне забезпечення є складним і не входить до розгляду у цьому курсі;
- хоча апаратне забезпечення ще не готове до широкого використання та до масового виробництва, вже треба працювати над програмним забезпеченням завдяки квантовим симуляторам та раннім прототипам.

Потенційний успіх квантових обчислень залежить від різних факторів, які можна розділити на дві категорії:

- апаратне забезпечення (потрібне нове та складне апаратне забезпечення);
- програмне забезпечення (щоб використовувати можливості, що надаються квантовим апаратним забезпеченням, необхідно розробити спеціалізоване програмне забезпечення).

Очікується, що квантове програмне забезпечення, написане для симуляторів квантових комп'ютерів, також працюватиме на реальних квантових комп'ютерах, за умови, що основні квантові концепції будуть схожими.

2.2 Апаратне забезпечення для квантових обчислень

Слід зазначити, що порівнювати різні квантові комп'ютери є нетривіальною задачею. На перший погляд, кількість кубітів може здатися найважливішим критерієм, але це може вводити в оману. Однією з найбільших труднощів при створенні квантових комп'ютерів є збереження квантових станів якомога довше. Найменше збурення може зруйнувати квантові стани, і тому квантові комп'ютери схильні до помилок, які необхідно виправляти.

У звіті за 2025 рік зазначаються частки ринку квантових процесорів з різними принципами дії [50]. Більшість пропозицій використовують надпровідники та спіни електрона. Також все більше розвиваються платформи з використанням захоплених іонів, фотонів та нейтральних атомів (рис. 2.7).

Крім принципу дії, квантові процесори розрізняються за кількістю кубітів, що надаються для роботи квантових схем (рис. 2.8). Найбільшу частку ринку охоплюють процесори на основі *квантового відпалу* та надпровідників. Слід зазначити, що квантові процесори на основі відпалу – це вузькоспеціалізований інструмент, створений саме для *задач оптимізації*. Процесор призначений виключно для задач пошуку мінімуму енергії при розв'язанні великих, комбінаторних задач. Процесори на основі квантового відпалу не є універсальними, оскільки кубіти мають обмежену зв'язність між собою.

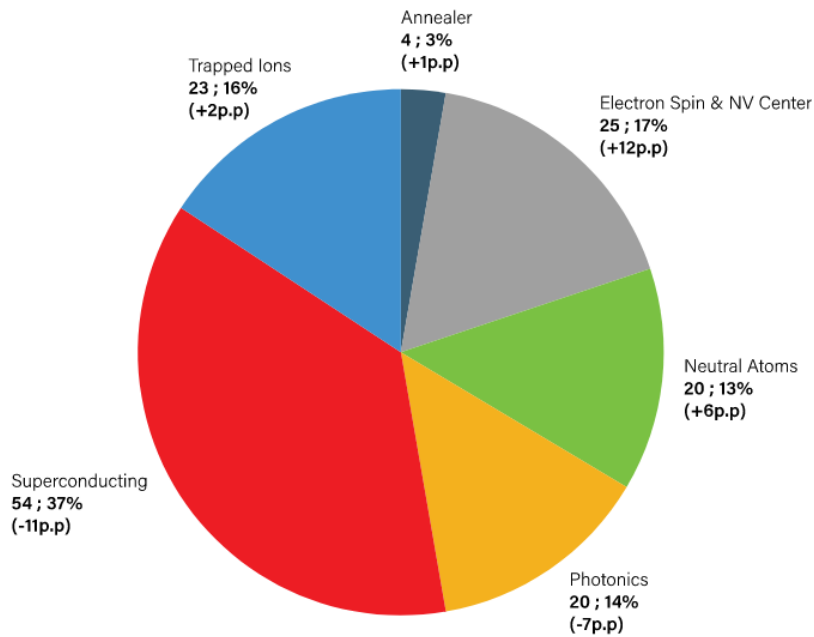


Рис. 2.7 – Доступність комерційних квантових процесорів на ринку [50].

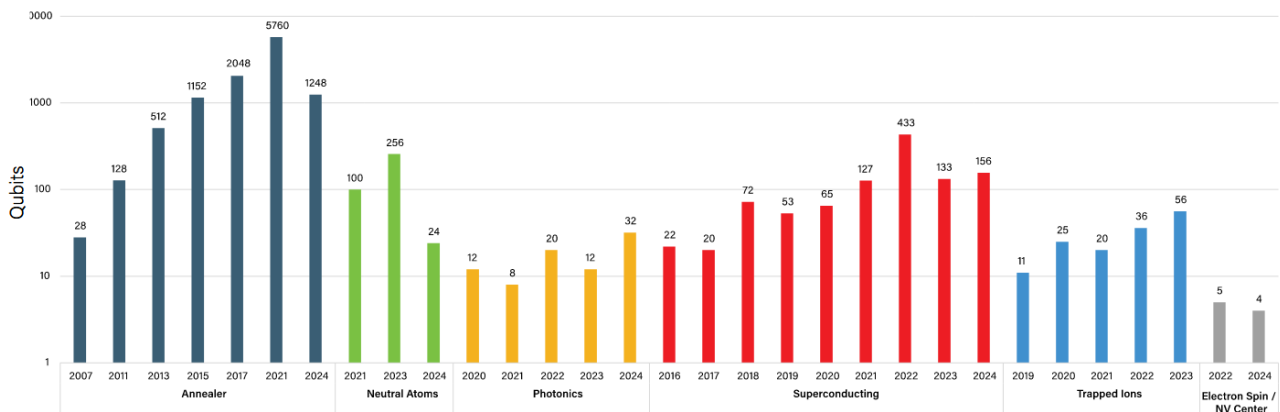


Рис. 2.8 – Кількість доступних кубітів в залежності від принципу дії (платформи) [50].

Оцінювання квантових процесорів здійснюється по таким ключовим показникам: кількість кубітів, когеренція, квантова точність, квантовий об'єм, швидкість затвору. Тестування продуктивності квантових процесорів виконується за допомогою: фізичних показників, наприклад, кількість кубітів та точність роботи вентилів кубітів; агрегованих показників, наприклад, квантовий об'єм, CLOPS та логічні кубіти; показників рівня додатків, наприклад, Q-Score та RACBEM [50].

Як вже було зазначено, квантові обчислення базуються на квантовій фізиці, з ними часто пов'язана певна «таємничість». Квантова фізика – не найпростіша частина фізики, а деякі її аспекти надзвичайно важко зрозуміти. Тому квантові

обчислення часто зображують як «таємничий» новий спосіб роботи з даними, що кардинально змінить світ.

Квантові обчислення мають величезний вплив на низку важливих питань в ІТ-індустрії. Тому важливо підготуватися до появи справжніх квантових комп'ютерів. Реальне квантове апаратне забезпечення з достатньою кількістю кубітів ще не доступне в широкому масштабі, розробники повинні усвідомлювати, що створення програмного забезпечення з використанням квантових обчислень також потребує часу. Завдяки квантовим симуляторам та раннім прототипам ніщо не заважає почати досліджувати квантові обчислення у проєктах. Це підвищить шанси на те, що програмне забезпечення буде готове до моменту появи апаратного забезпечення для використання у промислових масштабах.

2.3 Приклад застосування квантових обчислень

Хоча існують галузі, де квантові обчислення теоретично могли б призвести до величезних проривів, загально визнано, що квантові комп'ютери або квантові процесори можуть взяти на себе деякі завдання від класичних комп'ютерів, але вони не замінять класичні комп'ютери.

Проблеми, які можна вирішити за допомогою квантових обчислень, не відрізняються від проблем, які сьогодні вирішуються за допомогою класичних комп'ютерів. Однак, оскільки квантові обчислення використовують абсолютно інший базовий підхід, задачі можна розв'язувати абсолютно по-іншому.

Для певного набору задач за допомогою квантових обчислень можна досягти значного підвищення продуктивності. Як наслідок, квантові комп'ютери повинні бути здатні розв'язувати задачі, які сьогодні практично нерозв'язні, оскільки для їх розв'язання не вистачає обчислювальних ресурсів – наприклад, для симуляції хімічних реакцій, задач оптимізації або факторизації цілих чисел.

Квантові комп'ютери будуть найбільш корисними для розв'язання задач, які не можуть бути розв'язані класичними комп'ютерами за поліноміальний час, але можуть бути розв'язані квантовим комп'ютером за поліноміальний час.

Поширеним прикладом є *факторизація цілих чисел*, що є звичайною операцією в шифруванні (наприклад, у широко використовуваній криптосистемі RSA), або, точніше, у зламі шифрування. Основна ідея факторизації цілих чисел полягає в розкладанні числа на прості числа, які при множенні дають початкове число: наприклад, $15 = 3 \times 5$.

Задача ускладнюється, якщо числа стають більшими, як-от $146963 = 281 \times 523$. Чим більше число, яке необхідно розкласти на множники, тим довше триватиме пошук рішення. Це основа багатьох алгоритмів безпеки. Вони використовують ідею, що практично неможливо розкласти на множники число, що складається з 1024 бітів. Можна показати, що час, необхідний для розв'язання цієї задачі, має порядок $e^{\sqrt{(64/9)b(\log b)^2}}$, де b – кількість бітів у початковому числі. Зі збільшенням b час, необхідний для факторизації числа, зростає експоненційно. Час, необхідний для факторизації числа з b бітами класичним методом показано на рис. 2.9. Зауважте, що абсолютний час відрізняється в залежності від апаратного забезпечення обчислень. Навіть якщо використовуються найшвидші існуючі комп'ютери, додавання одного біта створює величезну різницю.

Кажуть, що ця задача є не поліноміальною, оскільки невідомо жодного класичного алгоритму, який міг би розв'язати її за поліноміальний час. Отже, зі збільшенням кількості бітів класичним комп'ютерам стає майже неможливо знайти розв'язок цієї задачі.

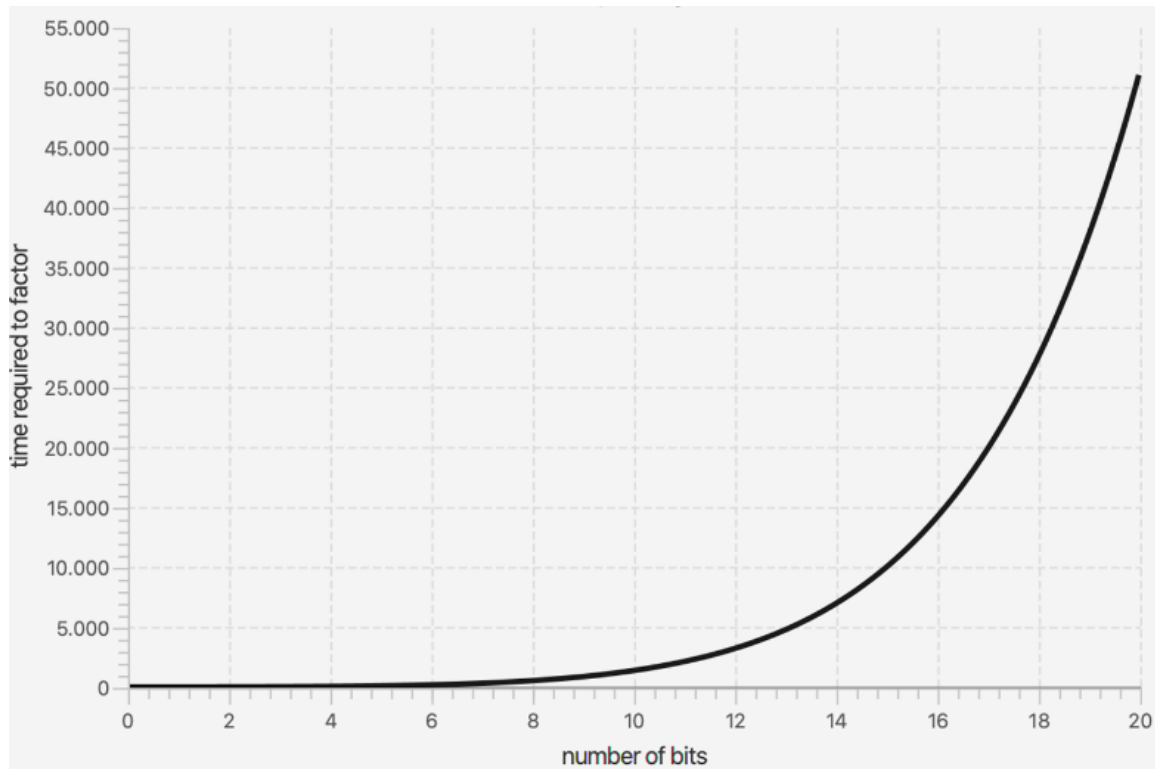


Рис. 2.9 – Час факторизації числа в залежності від кількості бітів [51].

Однак цю ж задачу можна розв'язати за допомогою квантового алгоритму за поліноміальний час. Використовуючи *алгоритм Шора*, час для розв'язання цієї задачі за допомогою квантового комп'ютера має порядок b^3 . Щоб показати, що це означає, накладаємо графік необхідного часу з використанням квантового алгоритму на квантовому комп'ютері на графік необхідного часу з використанням класичного алгоритму на класичному комп'ютері (рис. 2.10).

Такі задачі називають *поліноміальними в квантовому сенсі*. Саме для таких задач призначені квантові комп'ютери. Починаючи з певної кількості бітів, квантовий комп'ютер буде набагато швидшим за класичний. Більше того, чим більша кількість бітів, тим більша різниця. Це тому, що необхідний час для розв'язання задачі на класичному комп'ютері зростає експоненційно зі зростанням кількості бітів, тоді як те ж саме збільшення бітів спричинить лише поліноміальне зростання для квантового алгоритму (рис. 2.10).

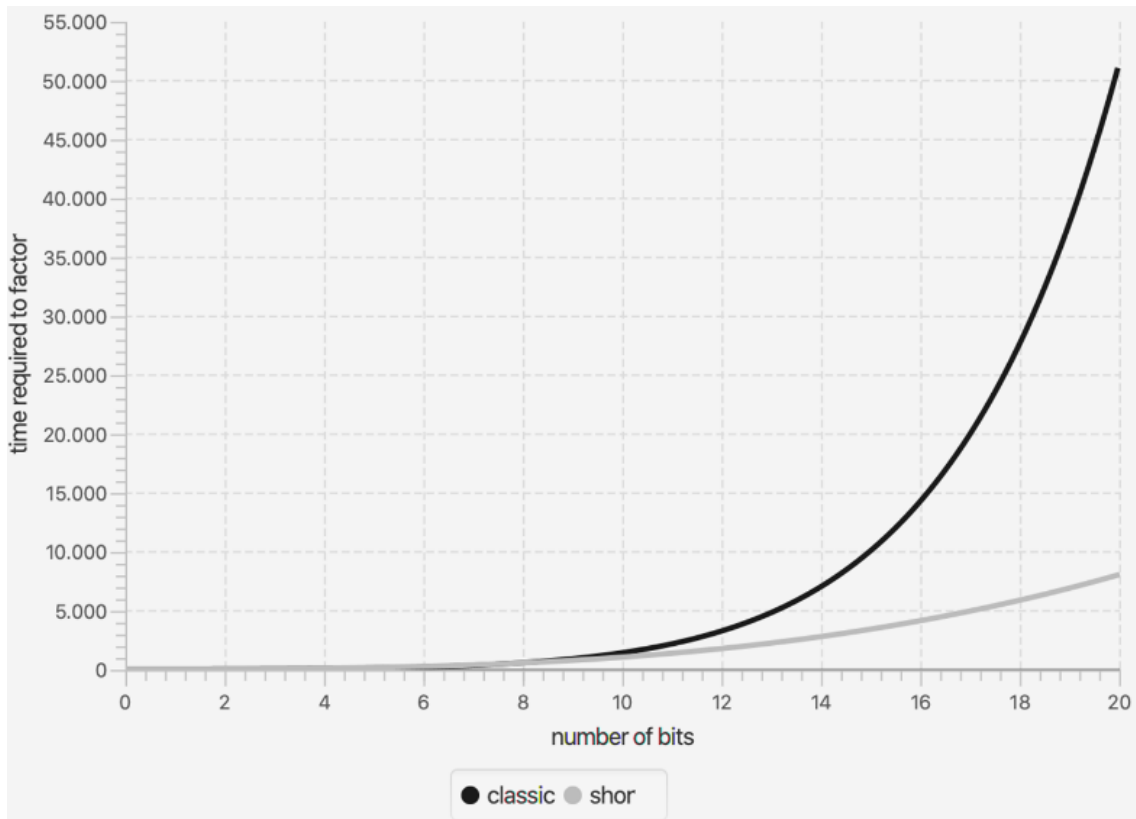


Рис. 2.10 – Порвняння часу факторизації числа класичним та квантовим алгоритмами [51].

Складність алгоритмів часто виражається як часова складність в термінах нотації великого O [52]. Загалом, алгоритми потребують більше часу для завершення, коли збільшується обсяг вхідних даних. Задачі часто поділяють на різні категорії, які вказують, наскільки складнішою стає задача при збільшенні вхідних даних.

Припустимо, що є n елементів вхідних даних. Якщо кожен елемент вимагає фіксованої кількості кроків обробки, загальний час для завершення алгоритму є лінійним відносно n . У цьому випадку кажуть, що алгоритм має лінійний час виконання. Багато алгоритмів є складнішими. Коли кількість вхідних елементів зростає, загальна кількість необхідних кроків може зростати як n^2 , або навіть як k -й ступінь n , n^k , для фіксованого значення k . У цьому випадку кажуть, що алгоритм має *поліноміальний час виконання*. Деякі алгоритми є ще складнішими, коли кількість вхідних елементів зростає.

Якщо невідомо жодного алгоритму, який міг би розв'язати задачу за поліноміальний час, то кажуть, що алгоритм має *неполіноміальний час*.

виконання. Алгоритми називають такими, що мають *експоненційний час виконання*, якщо вони вимагають експоненційно більше кроків при збільшенні n . Коли задача вимагає 2^n кроків, зрозуміло, що складність різко зростає, оскільки n знаходиться в показнику ступеня кількості кроків.

2.4 Нотації представлення кубіта

Хоча суперпозиція ще не обговорювалася детально, очевидно, що не (завжди) можливо ідентифікувати стан кубіта як єдиний – 0 або 1, що означає, що потрібна інша *нотація* для опису стану кубіта.

Існує кілька *нотацій* для кубітів, і залежно від випадку використання (показ стану схеми, пояснення роботи вентилів тощо), одна нотація може бути кращою за іншу. Розглянемо дві нотації на рис. 2.11: *нотація Дірака* та *векторна нотація*. При обговоренні суперпозиції у наступних розділах, ці нотації будуть використані та розширені. Наразі розглянемо лише базисні стани кубіта, які представляють значення 0 і 1.

$$\begin{array}{cc} |0\rangle & \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \text{а)} & \text{б)} \end{array}$$

Рис. 2.11 – Представлення кубіта у нотації а) Дірака та б) векторній нотації.

Для простого випадку, коли кубіт перебуває в одному зі своїх базисних станів, векторне представлення одного кубіта є простим. Представимо кубіт як вектор з двома елементами (рис. 2.12). Якщо кубіт містить значення 0, перший елемент у векторі дорівнює 1, а інший – 0.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Рис. 2.12 – Представлення кубітів у різних нотаціях.

У системі з більш ніж одним кубітом стан кубітів у нотації Дірака досягається шляхом конкатенації окремих кубітів. Два кубіти, кожен з яких має значення 0 – $|0\rangle|0\rangle$, можна описати так: $|00\rangle$.

Векторна нотація системи з кількома кубітами вимагає деяких векторних операцій. Результуючий вектор, що представляє систему з кількома кубітами, отримується шляхом *тензорного добутку* векторів кожного кубіта. Не обов'язково знати, як ці вектори отримуються. У системі (кубітовому регістрі), де перший кубіт дорівнює 0 і другий – 0, нотація кубітів є такою:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Система, де перший кубіт дорівнює 1, а другий – 0, нотація кубітів представлена на рис. 2.13.

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

Рис. 2.13 – Система з кубітами 0 та 1.

Загальна формула тензорного добутку двох двовимірних векторів має такий вигляд:

$$\begin{bmatrix} x \\ y \end{bmatrix} \otimes \begin{bmatrix} z \\ p \end{bmatrix} = \begin{bmatrix} xz \\ xp \\ yz \\ yp \end{bmatrix}.$$

Тензорний добуток двох матриць поєднує обидві матриці. Припустимо, що задана матриця **A** з k рядками та l стовпцем та матриця **B** з m рядками та n стовпцями. Тензорний добуток **A** та **B** – це нова матриця **C** з $k \cdot m$ рядками та $l \cdot n$. Кожен елемент матриці **A** замінюється в матриці, де заданий елемент **A** множиться на матрицю **B**, як показано у цій формулі:

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \otimes \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} a_{00}\mathbf{B} & a_{01}\mathbf{B} \\ a_{10}\mathbf{B} & a_{11}\mathbf{B} \end{bmatrix} = \begin{bmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{bmatrix}.$$

Це пояснює, чому розмірності нової матриці є добутком розмірностей вихідних матриць. Приклад тензорного добутку матриць **A** і **B** показаний на рис. 2.14.

$$\mathbf{A} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} 2 \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} \\ 3 \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 4 \\ -3 \\ 0 \\ 6 \end{bmatrix}$$

Рис. 2.14 - Приклад тензорного добутку матриць **A** і **B**.

Тензорні добутки достатньо поширені, оскільки вони є математичною абстракцією, яка описує взаємодію між двома квантовими системами: векторний простір, що описує одну квантову систему, тензоризовану векторним простором, що описує іншу, – це векторний простір, що складається з лінійних комбінацій усіх векторів у двох векторних просторах. Застосування тензорного добутку до квантових обчислень відбувається у квантових регістрах – великих квантових системах, що складаються з їхніх об'єднаних менших кубітів-аналогів.

Тензорний добуток – спосіб об'єднання векторних просторів у більші векторні простори, є дуже важливою операцією в квантових обчисленнях. Тензорний добуток між двома векторами $|u\rangle \otimes |v\rangle$ настільки поширений, що його зазвичай спрощують до $|u\rangle|v\rangle$ або $|uv\rangle$, а вектор, тензоризований самим собою n разів, позначається $|v\rangle^{\otimes n}$, де $\otimes n$ – ступінь тензоризації. Два вектор-стовпці $|u\rangle$ та $|v\rangle$ довжин m та n дають вектор-стовпець довжини $m \cdot n$ після тензоризації (рис. 2.15).

$$|u\rangle |v\rangle = |uv\rangle = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_m \end{bmatrix} \otimes \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_0 \cdot v_0 \\ u_0 \cdot v_1 \\ \vdots \\ u_0 \cdot v_n \\ u_1 \cdot v_0 \\ u_1 \cdot v_1 \\ \vdots \\ u_{m-1} \cdot v_n \\ u_m \cdot v_0 \\ u_m \cdot v_1 \\ \vdots \\ u_m \cdot v_n \end{bmatrix}$$

Рис. 2.15 – Загальний випадок тензорного добутку двох векторів-стовпців.

Існує ще один зручний зв'язок між *нотацією Дірака* та *векторною нотацією*. Якщо розглядати кубіти як біти, то біти в нотації Дірака відповідають цілому числу:

$$\begin{aligned} |00\rangle &= 0 & |10\rangle &= 2 \\ |01\rangle &= 1 & |11\rangle &= 3. \end{aligned}$$

Порівняймо цю нотацію з векторною нотацією з рис. 2.13. Єдиний елемент у цьому векторі, що має значення 1, знаходиться на позиції 2 (якщо припустити, що відлік починається з позиції 0). $|10\rangle$ відповідає десятковому значенню 2, і це збігається з другим елементом у відповідному векторі, знову ж таки, припускаючи, що відлік починається з позиції 0 (рис. 2.16). Отже, якщо прочитати біти в нотації Дірака як десяткове число, наприклад n , відповідний вектор складається з нулів, за винятком елемента на позиції n (починаючи з 0), який дорівнює 1.

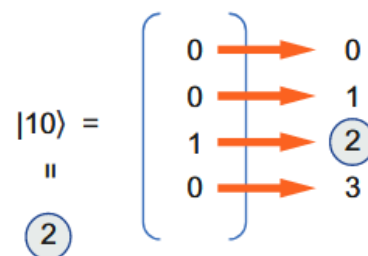


Рис. 2.16 – Залежність між десятковим значенням списку кубітів та позицією 1 у відповідному векторі ймовірностей.

Якщо додати ще один кубіт до системи, потрібно додати ще один тензорний добуток. Трьохкубітну систему, в якій перший кубіт дорівнює 1, другий – 0, а третій – 1, можна представити, як на рис. 2.17.

$$|101\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Рис. 2.17 – Трьохкубітна система $|101\rangle$.

Зауважте, що згаданий раніше зв'язок все ще діє для системи $|101\rangle$, де комбінація 101 є двійковим представленням цілого числа 5, і якщо порахувати перший рядок у векторі як рядок 0, елемент у рядку 5 дорівнює 1.

Розмір результуючого вектору швидко зростає зі збільшенням кількості бітів. Загалом, для n бітів результуючий вектор містить 2^n елементів. Може з'явитися питання, навіщо все так ускладнено? Навіщо потрібен вектор з вісьмома елементами, коли використовується лише три кубіти, і лише один з цих восьми елементів дорівнює 1? Відповідь криється у *суперпозиції*. На даний момент кубіти представлені в базисному стані. Якщо розглядати кубіти в стані суперпозиції, стане корисним і навіть необхідним представляти кубіти таким чином.

3 МАТЕМАТИЧНІ ОСНОВИ КВАНТОВИХ ОБЧИСЛЕНЬ: ЛІНІЙНА АЛГЕБРА, ЙМОВІРНІСТЬ, МАТРИЦІ, КОМПЛЕКСНІ ЧИСЛА

3.1 Основи лінійної алгебри

Матриця – це прямокутний масив чисел, символів або виразів, розташованих у рядках та стовпцях. Окремі значення матриці називаються її *елементами*. Горизонтальні та вертикальні лінії елементів у матриці називаються відповідно рядками та стовпцями. Розмір матриці визначається кількістю рядків та стовпців, які вона містить. Матриця з m рядками та n стовпцями називається матрицею розміру $m \cdot n$ (рис. 3.1). Матрицю можна використовувати для представлення лінійного відображення (англ. Linear map). Матрицю можна використовувати для представлення властивості математичного об'єкта.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Рис. 3.1 – Представлення матриці $\mathbf{A}$ розміром $m \times n$.

Матриця з одним рядком і n стовпцями називається вектором-рядком довжини n . Матриця з n рядками та одним стовпцем називається вектором-стовпцем довжини n . Вектор $\mathbf{A}$ у двовимірному просторі (також позначається, як $\vec{A}$) можна записати як матрицю-стовпчик (рис. 3.2).

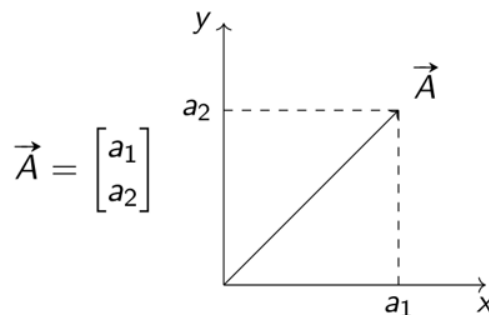


Рис. 3.2 – Графічне представлення вектору $\mathbf{A}$.

Вектор $\mathbf{B}$ у тривимірному просторі можна записати як матрицю-стовпчик (рис. 3.3).

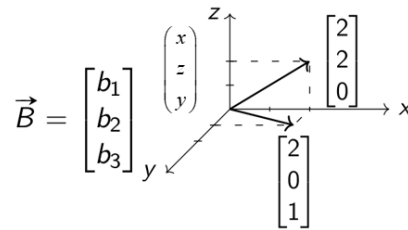


Рис. 3.3 – Представлення векторів у тривимірному просторі.

Довжина вектору представлена його нормою: для двовимірного вектору – $\|\mathbf{A}\| = \sqrt{a_1^2 + a_2^2}$, для тривимірного вектору – $\|\mathbf{B}\| = \sqrt{b_1^2 + b_2^2 + b_3^2}$, для n -вимірного вектору – $\|\mathbf{C}\| = \sqrt{c_1^2 + c_2^2 + \dots + c_n^2}$.

Для системи лінійних рівнянь, наприклад, другого порядку, застосовують лінійне перетворення виду:

$$\begin{cases} x_2 = ax_1 + by_1 \\ y_2 = cx_1 + dy_1 \end{cases} \quad \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}.$$

У двовимірному просторі це можна відобразити як перехід з точки з координатами (x_1, y_1) у точку (x_2, y_2) (рис. 3.4).

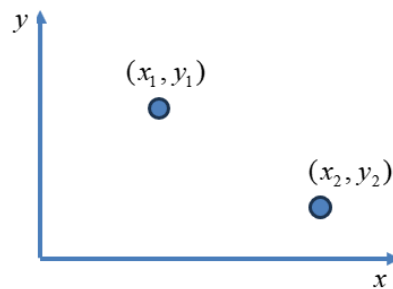


Рис. 3.4 – Двовимірний простір для лінійного перетворення.

Наприклад, для системи рівнянь

$$\begin{cases} 4x - 5y = 3 \\ 2x + y = 5 \end{cases},$$

можемо застосувати таке матричне представлення:

$$\begin{bmatrix} 4 & -5 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}.$$

Приклад обертання вектору з точки з координатами $(1;1)$ у точку з координатами $(0,7;0,7)$ можна представити рівнянням

$$\begin{bmatrix} 0,7 \\ 0,7 \end{bmatrix} = \begin{bmatrix} \cos(45) & -\sin(45) \\ \sin(45) & \cos(45) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Приклад множення матриці розміром 2×2 на матрицю розміром 2×1 показаний на рис. 3.5.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{bmatrix}$$

Рис. 3.5 – Множення матриці розміром 2×2 на матрицю розміром 2×1 .

Приклад множення матриці розміром 2×2 на матрицю розміром 2×2 показаний на рис. 3.6.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Рис. 3.6 – Множення двох матриць розміром 2×2 .

Приклад множення матриці розміром 3×3 на матрицю розміром 3×3 показаний на рис. 3.7.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix} = \begin{bmatrix} aj + bm + cp & ak + bn + cq & al + bo + cr \\ dj + em + fp & dk + en + fq & dl + eo + fr \\ gj + hm + ip & gk + hn + iq & gl + ho + ir \end{bmatrix}$$

Рис. 3.7 – Множення двох матриць розміром 3×3 .

Основне правило множення двох матриць полягає в тому, що *кількість стовпців першої матриці повинна дорівнювати кількості рядків другої матриці*. У результаті множення матриці $m \times n$ на матрицю $n \times r$ утвориться нова матриця розміром $m \times r$. Множення матриць не є комутативним, тобто $\mathbf{AB} \neq \mathbf{BA}$. Важлива послідовність, у якій стоять матриці у виразі.

Обернена матриця – це така квадратна матриця $\mathbf{A}^{-1}$, яка при множенні на початкову квадратну матрицю $\mathbf{A}$ дає одиничну матрицю $\mathbf{E}$ (матриця з одиницями на головній діагоналі та нулями в інших місцях, інколи може бути представлена позначенням $\mathbf{I}$), що позначається рівнянням $\mathbf{A}^{-1}\mathbf{A} = \mathbf{AA}^{-1} = \mathbf{E}$. Обернена матриця існує тільки для квадратних невироджених матриць, при $\det(\mathbf{A}) \neq 0$. Також $\mathbf{A}^{-1} = 1 / \mathbf{A}$. Наприклад,

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} -1/6 & 1/3 \\ 1/2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Приклади додавання та віднімання матриць показані на рис. 3.8, при цьому матриці повинні мати однакові розміри, інакше додавання та віднімання неможливі.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1-5 & 2-6 \\ 3-7 & 4-8 \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix} = -4 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Рис. 3.8 – Приклади додавання та віднімання матриць.

Знайдемо транспоновану матрицю.

Загальне правило транспонування матриці полягає в тому, щоб поміняти місцями рядки та стовпці матриці. Нехай $\mathbf{A}$ – матриця розміру $m \times n$. Транспонування $\mathbf{A}$, позначене як $\mathbf{A}^T$, – це матриця розміру $n \times m$. Загальний приклад транспонованої матриці $\mathbf{A}$ показаний на рис. 3.9.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Рис. 3.9 – Загальний приклад транспонованої матриці $\mathbf{A}$.

3.2 Комплексні числа

Комплексне число c – це число $a + bi$, де a та b – дійсні числа, а i – уявне число, яке визначається як $i^2 = -1$, $i = \sqrt{-1}$. a називається дійсною частиною c , а b – уявною частиною c . Цей запис називається *декартовим записом*. Існує також *полярний запис*. Комплексна площина показана на рис. 3.10.

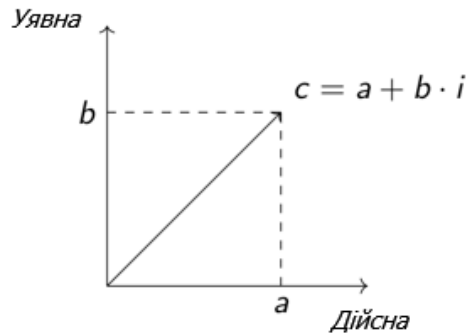


Рис. 3.10 – Комплексне число c на комплексній площині.

Комплексне число c можна представити як двовимірний вектор: $c = \begin{bmatrix} a \\ b \end{bmatrix}$.

Модуль комплексного числа $a + bi$ є також довжиною вектора $|c| = \sqrt{a^2 + b^2}$.

Аргумент комплексного числа $\arg(c)$ визначається як кут вектора з додатною дійсною віссю, який утворюється обертанням додатної осі x проти годинникової стрілки до прямої, що з'єднує початок координат $(0, 0)$ з точкою (x, y) .

Над комплексними числами визначені такі операції: додавання, віднімання, множення та ділення. Нехай маємо комплексні числа $c_1 = a_1 + b_1i$ та $c_2 = a_2 + b_2i$, тоді додавання, віднімання, множення та ділення визначається такими рівняннями:

$$\begin{aligned} c_1 + c_2 &= a_1 + a_2 + (b_1 + b_2)i, \\ c_1 - c_2 &= a_1 - a_2 + (b_1 - b_2)i, \\ c_1 c_2 &= (a_1 + b_1i)(a_2 + b_2i) = a_1 a_2 - b_1 b_2 + (a_1 b_2 + a_2 b_1)i, \\ \frac{c_1}{c_2} &= \frac{c_1}{c_2} \frac{c_2^*}{c_2^*} = \frac{c_1 c_2^*}{|c_2|^2}, \end{aligned}$$

де c^* – спряжене число, яке визначається, як $c^* = a - bi$, наприклад, $c = 3 + 2i, c^* = 3 - 2i, c = 5 - 4i, c^* = 5 + 4i$. Квадрат величини (англ. Squared magnitude) визначається, як $cc^* = (a + bi)(a - bi) = a^2 + b^2 = |c|^2$, наприклад $|3 + 2i|^2 = 3^2 + 2^2 = 13$. Спряжене число вектора – це вектор, дзеркально відображений відносно дійсної осі. Зауважте, що $|c|^2$ – дійсне число, оскільки

$|c|$ також є дійсним числом. Отже, ділення комплексних чисел можна переписати як множення, поділене на дійсне число, оскільки $|c_2|^2$ – дійсне число. Приклад

$$\text{ділення: } \frac{6+4i}{3+2i} = \frac{6+4i}{3+2i} \cdot \frac{3-2i}{3-2i} = \frac{(6+4i)(3-2i)}{3^2+2^2} = \frac{18-12i+12i-8i^2}{13} = 2.$$

Нехай є вектор $\mathbf{r}$ на комплексній площині (рис. 3.11) який представляє комплексне число c . Тоді мають місце такі вирази:

$$a = r \cos(\theta),$$

$$b = r \sin(\theta),$$

$$c = a + bi = r \cos(\theta) + r \sin(\theta)i = r(\cos(\theta) + i \sin(\theta)) = re^{i\theta},$$

$$\theta = \arg(c), r = \sqrt{a^2 + b^2}.$$

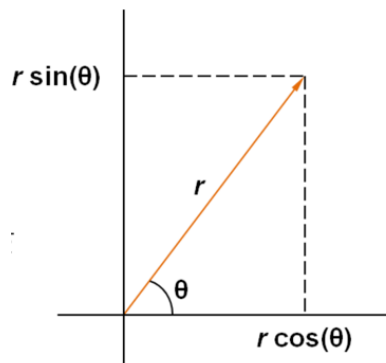


Рис. 3.11 – Вектор $\mathbf{r}$ на комплексній площині.

Наведемо приклади операцій над матрицями з комплексними елементами:

$$\begin{bmatrix} 1+2i & 2 \\ 3 & 2-5i \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 5 & 3 \end{bmatrix} = \begin{bmatrix} 3+2i & 6 \\ 8 & 5-5i \end{bmatrix},$$

$$\begin{bmatrix} 1+2i & 2 \\ 3 & 2-5i \end{bmatrix} - \begin{bmatrix} 2 & 4 \\ 5 & 3 \end{bmatrix} = \begin{bmatrix} -1+2i & -2 \\ -2 & -1-5i \end{bmatrix},$$

$$(1+2i) \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 2+4i \\ 5+10i \end{bmatrix}.$$

Комплексно спряжену матрицю отримуємо шляхом взяття спряженої матриці кожного елемента матриці.

Нехай $\mathbf{A}$ – матриця розміру $m \times n$, тоді комплексно спряжену матрицю для $\mathbf{A}$, позначену як $\mathbf{A}^*$, розміром $m \times n$ можна представити так:

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, \mathbf{A}^* = \begin{bmatrix} a^* & b^* & c^* \\ d^* & e^* & f^* \end{bmatrix}.$$

Приклади спряжених матриць:

$$\begin{aligned} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^* &= \begin{bmatrix} 1^* & 2^* \\ 3^* & 4^* \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \\ \begin{bmatrix} 1 & 2i & 3 \\ 4 & 5 & 6i \\ 7 & 8i & 9 \end{bmatrix}^* &= \begin{bmatrix} 1^* & 2i^* & 3^* \\ 4^* & 5^* & 6i^* \\ 7^* & 8i^* & 9^* \end{bmatrix} = \begin{bmatrix} 1 & -2i & 3 \\ 4 & 5 & -6i \\ 7 & -8i & 9 \end{bmatrix}, \\ \begin{bmatrix} 2+5i & i \\ 3 & 3-4i \end{bmatrix}^* &= \begin{bmatrix} 2-5i & -i \\ 3 & 3+4i \end{bmatrix}. \end{aligned}$$

Спряжена транспонована матриця отримується шляхом взяття спряженого кожного елемента матриці, а потім транспонування матриці. Нехай $\mathbf{A}$ – матриця розміру $m \times n$. Спряжена транспонована матриця для $\mathbf{A}$, позначена $\mathbf{A}^\dagger$, – це матриця розміру $n \times m$.

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, \mathbf{A}^\dagger = \begin{bmatrix} a^* & d^* \\ b^* & e^* \\ c^* & f^* \end{bmatrix}.$$

Приклади спряжених матриць:

$$\begin{aligned} \begin{bmatrix} 1i & 2 \\ 3 & 4i \\ 5i & 6 \end{bmatrix}^\dagger &= \begin{bmatrix} 1i^* & 3^* & 5i^* \\ 2^* & 4i^* & 6^* \end{bmatrix} = \begin{bmatrix} -i & 3 & i \\ 2 & -4i & 6 \end{bmatrix}, \\ \begin{bmatrix} 2+5i & i \\ 3 & 3-4i \end{bmatrix}^\dagger &= \begin{bmatrix} 2-5i & 3 \\ -i & 3+4i \end{bmatrix}. \end{aligned}$$

4 ОСНОВНІ ПОНЯТТЯ КВАНТОВИХ ОБЧИСЛЕНЬ: КУБІТ, СУПЕРПОЗИЦІЯ, ЗАПЛУТАНІСТЬ, ТЕЛЕПОРТАЦІЯ

Перш ніж заглиблюватися у формальну математику, що використовується для опису квантових алгоритмів, важливо спочатку ознайомитися з нотацією, яка зазвичай використовується для опису більш загальних квантово-механічних систем, а також станів квантових регістрів. Ця нотація відома як **нотація бракет** або **нотація Дірака**, названа на честь нобелівського лауреата Поля Дірака. *Нотація Дірака* – це спосіб опису векторів. Вектор-стовпець $|v\rangle$ називається «Кет- v ». Дуальний вектор для $|v\rangle$ має таке позначення Дірака: $\langle v| = \overline{v}^T = [\overline{v_0}, \overline{v_1}, \dots, \overline{v_n}]$, де $\overline{v}$ – це комплексно спряжений вектор з v , який також може бути позначений v^* у літературі з квантових обчислень. Цей дуальний вектор називається «Бра- v ».

Нотація Дірака – це зручний спосіб опису векторів у *гільбертовому просторі* $\mathbb{C}^n$, який є векторним простором, найбільш корисним для розгляду квантових систем. *Гільбертовий простір* – це векторний простір зі скалярним добутком та нормою, визначеною цим скалярним добутком.

Скалярний добуток векторного простору – це операція, яка присвоює скалярне значення кожній парі векторів u та v у векторному просторі, а скалярний добуток двох векторів у гільбертовому просторі $\mathbb{C}^n$ позначається за допомогою нотації Дірака $\langle u|v\rangle$. Таким чином, скалярний добуток $\langle u|v\rangle$ двох векторів у комплексному гільбертовому просторі задається скалярним добутком векторів v та $\overline{u}^T$:

$$\langle u|v\rangle = \overline{u}^T v = [\overline{u_0}, \overline{u_1}, \dots, \overline{u_n}] \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = \overline{u_0}v_0 + \overline{u_1}v_1 + \dots + \overline{u_n}v_n.$$

Для скінченновимірних векторних просторів, таких як векторний простір $\mathbb{C}^n$, гільбертові простори нічим не відрізняються від просторів скалярного

добутку. Формально, гільбертів простір має бути простором скалярного добутку, який також є повним метричним простором: скалярний добуток гільбертового простору H має індукувати норму таку, що для всіх наборів векторів у H , якщо $\sum_{i=0}^{\infty} \|v_i\|$ збігається абсолютно, то воно збігається до значення в H (кожна послідовність Коші збігається в H).

За визначенням, скалярний добуток задовольняє такі умови: $\langle v|v \rangle \geq 0, \langle v|v \rangle = 0$ тоді і тільки тоді, коли $|v\rangle = 0$; $\langle u|v \rangle = \overline{\langle v|u \rangle}$ для всіх $|u\rangle, |v\rangle$ у векторному просторі; $\langle u|\alpha_0 v + \alpha_1 w \rangle = \alpha_0 \langle u|v \rangle + \alpha_1 \langle u|w \rangle$.

У більш загальному випадку, скалярний добуток $|u\rangle \sum_i \alpha_i |v_i\rangle = \sum_i \alpha_i \langle u|v_i \rangle$ для всіх скалярів α_i та векторів $|u\rangle, |v\rangle$ у векторному просторі (це відомо як лінійність у другому аргументі). Отже, норма вектора $|v\rangle$ у гільбертовому просторі визначається за допомогою квадратного кореня зі скалярного добутку $|v\rangle$ на самого себе: $\|v\| = \sqrt{\langle v|v \rangle}$. Ця норма також відома як ℓ^2 -норма, 2-норма або евклідова норма над комплексними числами. Геометрично ця норма задає відстань від початку координат до точки $|v\rangle$, що впливає з теореми Піфагора.

Поряд зі скалярним добутком, можна також визначити *зовнішній добуток* двох векторів. Зовнішній добуток, що позначається $|v\rangle \langle u|$ або $|v\rangle \otimes \langle u|$, є тензорним або *кронекерівським* добутком $|v\rangle$ зі спряженою транспозицією $\langle u|$. Результатом є не скаляр, як у випадку зі скалярним добутком, а матриця:

$$|v\rangle \langle u| = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} \begin{bmatrix} \overline{u_0} & \overline{u_1} & \dots & \overline{u_m} \end{bmatrix} = \begin{bmatrix} \overline{v_0 u_0} & \overline{v_0 u_1} & \dots & \overline{v_0 u_m} \\ \overline{v_1 u_0} & \overline{v_1 u_1} & \dots & \overline{v_1 u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{v_n u_0} & \overline{v_n u_1} & \dots & \overline{v_n u_m} \end{bmatrix}.$$

Зовнішній добуток часто використовується для опису лінійного перетворення між векторними просторами. Замість використання «неелегантних» матриць перетворення, як зображено вище, лінійне

перетворення з гільбертового простору U в інший гільбертів простір V на векторі $|\mathbf{w}\rangle$ в U можна лаконічно та зручно описати: $(|\mathbf{v}\rangle\langle\mathbf{u}|)|\mathbf{w}\rangle = |\mathbf{v}\rangle\langle\mathbf{u}|\mathbf{w}\rangle = \langle\mathbf{u}|\mathbf{w}\rangle|\mathbf{v}\rangle$.

Класичний біт може бути представлений як число в системі числення з основою 2, яке приймає значення 1 або 0. Кубіти представлені подібним чином, оскільки вони також є числами в системі числення з основою 2 і приймають значення 1 або 0 при вимірюванні, а отже, переходять у класичний стан. На відміну від класичних бітів, у своєму «неколапсованому» квантовому стані (до вимірювання) кубіт перебуває в *суперпозиції* вимірюваних значень 1 і 0.

Найзручніший спосіб математичного представлення стану кубіта в будь-який момент часу – це двовимірний простір станів у $\mathbb{C}^2$ з ортонормальними базисними векторами $|1\rangle$ і $|0\rangle$.

Суперпозиція $|\psi\rangle$ кубіта представлена як лінійна комбінація цих базисних векторів: $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, де α_0 – комплексна скалярна амплітуда вимірювання 0, α_1 – амплітуда вимірювання значення 1.

Амплітуди можна розглядати як «**квантові ймовірності**», оскільки вони представляють ймовірність того, що заданий квантовий стан буде спостерігатися, коли суперпозиція згортається («колапсується»). Фундаментальна відмінність між ймовірностями станів у *класичних ймовірнісних алгоритмах* та амплітудами станів у *квантових алгоритмах* полягає в тому, що амплітуди представлені *комплексними числами*, тоді як традиційні ймовірності описуються *дійсними числами*. Комплексні числа необхідні для повного опису *суперпозиції* станів та *інтерференції* або *заплутаності*, властивих квантовим системам. З цього випливає: оскільки ймовірності класичної системи повинні дорівнювати 1, щоб ймовірності утворювали повний розподіл ймовірностей, квадрати абсолютних значень амплітуд станів у квантовій системі аналогічно повинні дорівнювати 1.

Звичайно, так само, як апаратне забезпечення, що лежить в основі бітів класичного комп'ютера, може змінюватися у відповідь на зміну напруги, квантові системи зазвичай не поведуться так ідеально.

Можна припустити, що ці напруги відповідають дискретним логічним значенням, і подібне припущення робиться щодо квантових векторів станів, яке спрощує міркування про квантові обчислювальні системи, що називається *умовою нормалізації*: $|\psi\rangle$ – одиничний вектор. Це означає, що $\| |\psi\rangle \| = \langle \psi | \psi \rangle = 1$. Оскільки $|0\rangle$ та $|1\rangle$ є ортонормальними, то $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Отже, ймовірність спостереження одного можливого стану з суперпозиції отримується шляхом зведення в квадрат абсолютного значення її амплітуди. Ймовірність того, що кубіт знаходиться у стані $|0\rangle$, дорівнює $|\alpha_0|^2$, а ймовірність того, що кубіт буде виміряно як $|1\rangle$, дорівнює $|\alpha_1|^2$, або $1 - |\alpha_0|^2$.

Зведемо скалярний добуток до його складових векторів:

$$\begin{aligned}
 1 &= \langle \psi | \psi \rangle \\
 &= (\overline{a_0} \langle 0 | + \overline{a_1} \langle 1 |) \cdot (a_0 |0\rangle + a_1 |1\rangle) \\
 &= (\overline{a_0} [\psi_{00} \ \psi_{01}] + \overline{a_1} [\psi_{10} \ \psi_{11}]) \cdot \left(a_0 \begin{bmatrix} \psi_{00} \\ \psi_{01} \end{bmatrix} + a_1 \begin{bmatrix} \psi_{10} \\ \psi_{11} \end{bmatrix} \right) \\
 &= [\overline{a_0 \psi_{00}} + \overline{a_1 \psi_{10}} \ \overline{a_0 \psi_{01}} + \overline{a_1 \psi_{11}}] \cdot \begin{bmatrix} a_0 \psi_{00} + a_1 \psi_{10} \\ a_0 \psi_{01} + a_1 \psi_{11} \end{bmatrix} \\
 &= \overline{a_0 \psi_{00}} a_0 \psi_{00} + \overline{a_1 \psi_{10}} a_0 \psi_{00} + \overline{a_0 \psi_{00}} a_1 \psi_{10} + \overline{a_1 \psi_{10}} a_1 \psi_{10} \\
 &\quad + \overline{a_0 \psi_{01}} a_0 \psi_{01} + \overline{a_1 \psi_{11}} a_0 \psi_{01} + \overline{a_0 \psi_{01}} a_1 \psi_{11} + \overline{a_1 \psi_{11}} a_1 \psi_{11} \\
 &= |a_0|^2 (|\psi_{00}|^2 + |\psi_{01}|^2) + |a_1|^2 (|\psi_{10}|^2 + |\psi_{11}|^2) \\
 &\quad + \overline{a_1} a_0 (\overline{\psi_{10}} \psi_{00} + \overline{\psi_{11}} \psi_{01}) + \overline{a_0} a_1 (\overline{\psi_{00}} \psi_{10} + \overline{\psi_{01}} \psi_{11}) \\
 &= |a_0|^2 + |a_1|^2 .
 \end{aligned}$$

Цей висновок є прикладом зручності нотації Дірака. Та сама еквівалентність стає зрозумілою за допомогою набагато меншої кількості кроків в нотації Дірака:

$$\begin{aligned}
1 &= \langle \psi | \psi \rangle \\
&= (\overline{a_0} \langle 0| + \overline{a_1} \langle 1|) \cdot (a_0 |0\rangle + a_1 |1\rangle) \\
&= |a_0|^2 \langle 0|0\rangle + |a_1|^2 \langle 1|1\rangle + \overline{a_1} a_0 \langle 1|0\rangle + \overline{a_0} a_1 \langle 0|1\rangle \\
&= |a_0|^2 + |a_1|^2 .
\end{aligned}$$

Вміст «бра» та «кетів» Дірака є мітками, які описують основні вектори. $|0\rangle$ та $|1\rangle$ можна перетворити на будь-які два вектори, що утворюють ортонормований базис у $\mathbb{C}^2$. Найпоширеніший базис, що використовується в квантових обчисленнях, називається обчислювальним базисом:

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

Але можна використовувати будь-який інший ортонормований базис. Наприклад, базисні вектори:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Це інший, але еквівалентний спосіб представлення кубіта:

$$\begin{aligned}
|\psi\rangle &= a_0 |0\rangle + a_1 |1\rangle \\
&= a_0 \frac{|+\rangle + |-\rangle}{\sqrt{2}} + a_1 \frac{|+\rangle - |-\rangle}{\sqrt{2}} \\
&= \frac{a_0 + a_1}{\sqrt{2}} |+\rangle + \frac{a_0 - a_1}{\sqrt{2}} |-\rangle .
\end{aligned}$$

Тут, замість вимірювання станів $|0\rangle$ та $|1\rangle$, кожен з відповідними ймовірностями $|\alpha_0|^2$ та $|\alpha_1|^2$, стани $|+\rangle$ та $|-\rangle$ будуть вимірюватися з ймовірностями $\frac{|\alpha_0 + \alpha_1|^2}{2}$ та $\frac{|\alpha_0 - \alpha_1|^2}{2}$. Обчислювальний базис, як правило, є найпростішою основою для обчислень та розуміння квантових алгоритмів.

Важко виконати будь-які корисні обчислення лише з одним кубітом. Як і класичні комп'ютери, квантові комп'ютери використовують *квантові регістри*, що складаються з кількох кубітів. У згорнутому стані квантові регістри є бітовими рядками, довжина яких визначає обсяг інформації, яку вони можуть

зберігати. У суперпозиції кожен кубіт у регістрі знаходиться у станах $|0\rangle$ та $|1\rangle$, і, отже, регістр з n кубітів знаходиться в суперпозиції всіх 2^n можливих бітових рядків, які можна представити за допомогою n бітів. Простір станів квантового регістра розміром n є лінійною комбінацією n базисних векторів, кожен з яких має довжину 2^n :

$$|\psi_n\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle,$$

де i – це десяткове ціле представлення числа довжини n у двовимірній системі числення. Отже, регістр із трьох кубітів матиме таке розкладання:

$$|\psi_2\rangle = \alpha_0 |000\rangle + \alpha_1 |001\rangle + \alpha_2 |010\rangle + \alpha_3 |011\rangle + \alpha_4 |100\rangle + \alpha_5 |101\rangle + \alpha_6 |110\rangle + \alpha_7 |111\rangle.$$

У матричній формі таке розкладання має таке представлення, не дуже зручне для виконання операцій:

$$|\psi_2\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_5 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_6 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_7 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Як згадувалося раніше, кожна можлива бітова конфігурація в квантовій суперпозиції позначається тензорним добутком відповідних кубітів. Розглянемо $|101\rangle$, бітовий рядок, що представляє ціле число 5:

$$|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0]^T.$$

Як згадувалося раніше, система з 2-ма кубітами представлена так:

$$|\psi_1\psi_1\rangle = \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} \otimes \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix}.$$

Це рівняння показує, що систему з двома кубітами можна описати (ймовірнісним) вектором з чотирма значеннями. Два кубіти можуть одночасно містити чотири значення. При вимірюванні для кожного кубіта залишається лише *одне* значення. Але усі обчислення в квантовому алгоритмі оперують *чотирма* значеннями. За ширшим тлумаченням, система з n кубітами відповідає вектору з 2^n елементами. Отже, очікується, що квантові комп'ютери зможуть розв'язувати експоненційні задачі: зі збільшенням кількості кубітів квантова система може працювати з експоненційно зростаючою кількістю значень.

Як і у випадку з окремими кубітами, квадрат абсолютного значення амплітуди, пов'язаної з заданим бітовим рядком, є ймовірністю спостереження цього бітового рядка після згортання регістра до класичного стану, а квадрати абсолютних значень амплітуд усіх 2^n можливих бітових конфігурацій n -бітового регістра додаються до одиниці:

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

На рис. 4.1 шість кубітів відповідають вектору з $64 = 2^6$ елементами. Після вимірювання один з цих елементів має значення 1, а всі інші елементи мають значення 0. За індексом елемента, який має значення 1, можна обчислити окремі значення для шести кубітів. Отже, починаємо з шести значень (кожне або 0, або 1) і закінчуємо шістьма значеннями (кожне або 0, або 1).

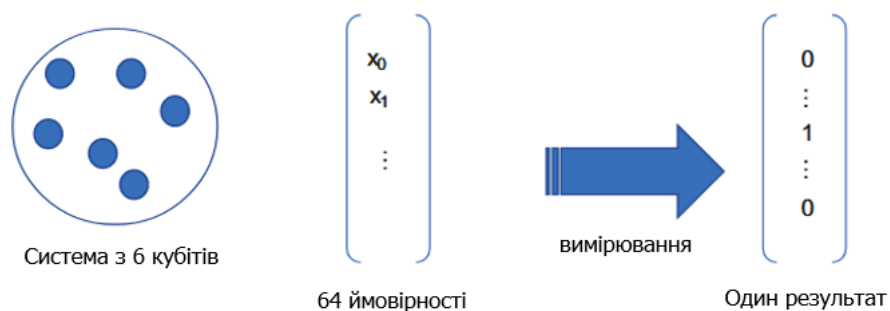


Рис. 4.1 – Приклад вимірювання системи з шести кубітів.

На перший погляд, через суперпозицію може не бути чіткої різниці між кількістю бітів. У системі може бути експоненційна кількість елементів, але як тільки вимірюють значення кубітів, відбувається повернення до класичного стану, де кожен біт має рівно одне чітко визначене значення. *Перевага суперпозиції* для квантових обчислень полягає у *можливості квантової системи виконувати обробку, поки кубіти перебувають у стані суперпозиції*.

Отже, операції, визначені квантовими алгоритмами, оброблюють не лише шість бітів, вони оброблюють 64 значення ймовірності. Один крок у квантовому алгоритмі на квантовому комп'ютері з шістьма кубітами – це зміна 64 значень. **Додавання одного кубіта подвоює обчислювальні можливості квантового комп'ютера.** Це пояснює термін *експоненційний*, який часто використовується в квантових обчисленнях: додавання n кубітів додає обчислювальну потужність, пропорційну 2^n , де n знаходиться в показнику ступеня цього рівняння.

Порівняймо класичний комп'ютер із шістьма бітами з квантовим комп'ютером із шістьма кубітами (рис. 4.2). Обидва комп'ютери мають одне значення, що складається з шести бітів, як вхідне значення для алгоритму, і після вимірювання вихідного значення алгоритму вони обидва знову зчитують значення з шести бітів. Обидва комп'ютери можуть обробити 64 можливі комбінації як вхідне значення. Ключова відмінність полягає в тому, що квантовий комп'ютер може обробити ці 64 комбінації одночасно.

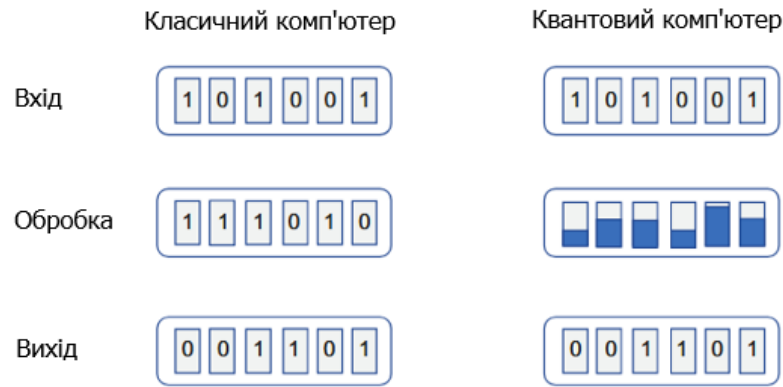


Рис. 4.2 – Порівняння квантового і класичного комп'ютерів.

Розуміючи, як працюють квантові регістри та чим вони відрізняються від класичних регістрів, тепер можна уявити, як стан квантового регістра може еволюціонувати з часом, змінюючись для досягнення певної кінцевої мети.

У класичних обчисленнях один із способів мислення про проектування та виконання алгоритмів – це використання універсальних машин Тюрінга. Квантові універсальні машини Тюрінга вперше були описані Девідом Дойчем у 1985 році [4], але проектування алгоритмів для квантових машин Тюрінга є ще складнішим та трудомісткішим, ніж для класичних машин Тюрінга. *Як стрічка квантової машини Тюрінга, так і її головка читання-запису існують у суперпозиціях експоненціального числа станів!*

Як і в класичній інформатиці, замість використання машини Тюрінга як обчислювальної моделі, операції на квантовому комп'ютері найчастіше описуються за допомогою **квантових схем**, що складаються з кубітів та квантових логічних вентилів, концепції, яку також ввів Дойч через кілька років після його специфікації квантового аналога машини Тюрінга [8].

У класичних обчисленнях двійкові значення, що зберігаються в регістрі, проходять через логічні вентиля, які за певного *двійкового входу* генерують певний *двійковий вихід*. Класичні логічні вентиля описуються за допомогою *булевої алгебри*. Квантові логічні вентиля діють подібним чином, оскільки *квантові логічні вентиля*, що застосовуються до *квантових регістрів*, **відображають одну квантову суперпозицію в іншу**, разом дозволяючи еволюцію системи до певного бажаного кінцевого стану – розв'язку задачі.

Як отримати перевагу від властивості експоненційного масштабування квантових обчислень, оскільки експоненційний ступінь застосовується лише під час обробки, а не під час вимірювання? Особливості написання квантових алгоритмів полягають у тому, щоб придумати операції, застосування яких призведе до вимірювання, що більше наближає до розв'язку задачі. Перед вимірюванням квантової системи виконується обробка у спосіб застосування квантових вентилів (рис. 4.3). Ці операції з квантовими вентилями змінюють стан вектора ймовірності.

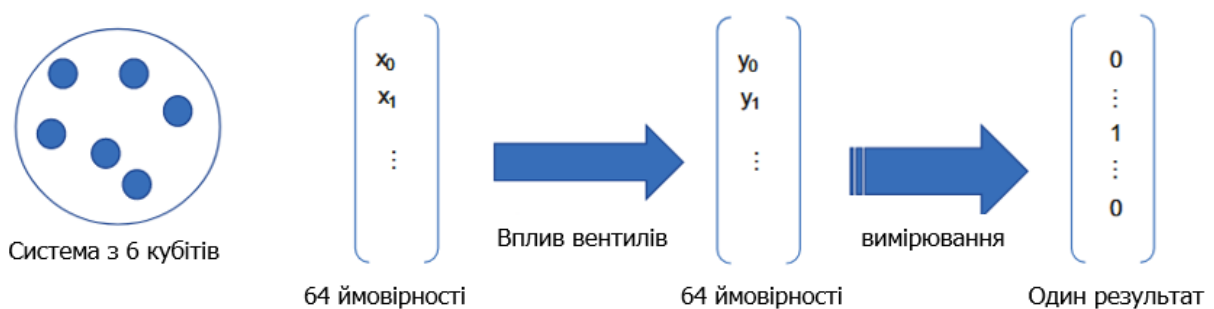


Рис. 4.3 – Схема роботи квантового комп'ютера.

Наприклад, серед 1000 чисел, одне є простим числом. Необхідно знайти індекс простого числа. Уявіть, що ви можете одночасно оброблювати ці числа таким чином, щоб усі вони стали 0, крім простого числа, яке стає 1. Таким чином, одне вимірювання покаже положення простого числа. Хоча для цієї задачі не існує простого квантового алгоритму, аналогія показує перевагу можливості обробляти велику кількість значень, навіть якщо результатом є одне значення.

Квантові *логічні вентиля* математично представлені як *матриці перетворення* або *лінійні оператори*, що застосовуються до *квантового регістра* у спосіб тензорного добутку матриці перетворення з матричним представленням регістра. Усі лінійні оператори, що відповідають квантовим логічним вентилям, повинні бути **унітарними**. Тобто, якщо комплексна матриця U є унітарною, то має бути вірним, вираз $U^{-1} = U^{\dagger}$, де $U^{\dagger}$ – спряжена транспонована матриця: $U^{\dagger} = \overline{U}^T$. Звідси випливає, що $UU^{\dagger} = U^{\dagger}U = I$. Унітарні оператори зберігають скалярний добуток двох векторів, геометрично зберігаючи довжини векторів та кут між ними: $\langle u | U^{\dagger}U | v \rangle = \langle u | I | v \rangle = \langle u | v \rangle$. Композиція двох унітарних

операторів також є унітарною. Для унітарних матриць перетворення U та V мають такий вигляд: $(UV)^\dagger = V^\dagger U^\dagger = V^{-1} U^{-1} = (UV)^{-1}$, оскільки добуток (як ермітово спряжених, так і звичайних обернених) дорівнює *оберненому* порядку добутку.

Унітарні перетворення, що виконуються над одним кубітом, можна візуалізувати як обертання та відбиття навколо вісей x , y та z **сфери Блоха** (рис. 4.4). Усі можливі лінійні комбінації $\alpha_0|0\rangle + \alpha_1|1\rangle$ в $\mathbb{C}^2$ відповідають усім точкам (θ, ϕ) на поверхні одиничної сфери, де $\alpha_0 = \cos(\theta/2)$ та $\alpha_1 = e^{i\phi} \sin(\theta/2)$:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle = \cos\frac{\theta}{2}|0\rangle + (\cos\phi + i\sin\phi)\sin\frac{\theta}{2}|1\rangle.$$

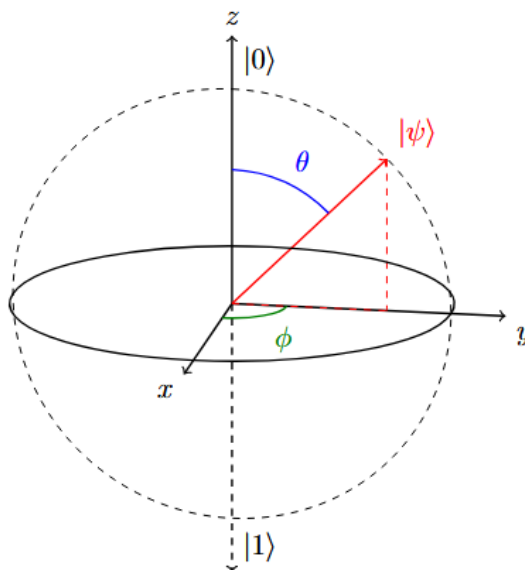


Рис. 4.4 – Представлення кубіта на сфері Блоха.

Хоча схеми полегшують розуміння потоку операцій у квантових алгоритмах та їх роботи для пошуку розв'язку, *сфера Блоха* інтуїтивно нагадує про базову реалізацію, яка зумовлює особливі властивості, що відрізняють *квантові обчислення* від *класичних обчислень* на основі взаємодії між вентилями (операторами) та кубітами.

У реальному квантовому комп'ютері «дроти» схеми можуть відповідати окремому електрону, що рухається в часі, а кожен «вентиль» – зміні в характері руху цього електрона. Оскільки квантові вентиля настільки тісно пов'язані з

їхніми базовими квантово-механічними системами, важливо знати, як ці вентиля впливають на квантову систему, щоб отримати результат.

Хоча будь-яке унітарне перетворення є допустимою елементарною операцією на квантовому комп'ютері, існує невелика кількість часто використовуваних *квантових логічних вентилів*, аналогічних класичним логічним вентилям за роллю, яку вони відіграють у галузі квантових обчислень. Фактично, будь-яке унітарне перетворення можна виконати за допомогою кількох вентилів. Одним із важливих прикладів є однокубітний оператор (вентиль) Адамара (англ. Hadamard gate).



Рис. 4.5 – Вентиль (оператор) Адамара.

Вентиль Адамара є однією з найважливіших концепцій квантових обчислень. Він переводить кубіт у стан *суперпозиції*. Після застосування вентиля Адамара до кубіта, який містить значення 0, існує 50% ймовірність того, що кубіт буде виміряно як 0, і 50% ймовірність того, що кубіт, після вимірювання, матиме значення 1. Часто званий «чесним підкиданням монети», оператор Адамара, застосований до кубіта зі значенням $|0\rangle$ або $|1\rangle$, індукує рівну суперпозицію станів $|0\rangle$ та $|1\rangle$, з однаковою ймовірністю, що кубіт знаходиться в стані $|1\rangle$ або $|0\rangle$ під час вимірювання. Дія оператора Адамара показана на рис. 4.6.

$$\text{---} \boxed{H} \text{---} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \langle 0| + \frac{|0\rangle - |1\rangle}{\sqrt{2}} \langle 1|$$

Рис. 4.6 – Представлення оператора Адамара у матричній формі.

Багато квантових алгоритмів починаються із застосування оператора Адамара до кожного кубіта в регістрі, що надає кожній з 2^n можливих побітових конфігурацій n кубітів однакову ймовірність 2^{-n} бути спостережуваною під час

вимірювання системи. Геометрично оператор Адамара виконує обертання на $\pi/2$ навколо осі y , а потім обертання навколо осі x на π радіан на сфері Блоха, що показано для різних базових станів (рис. 4.7).

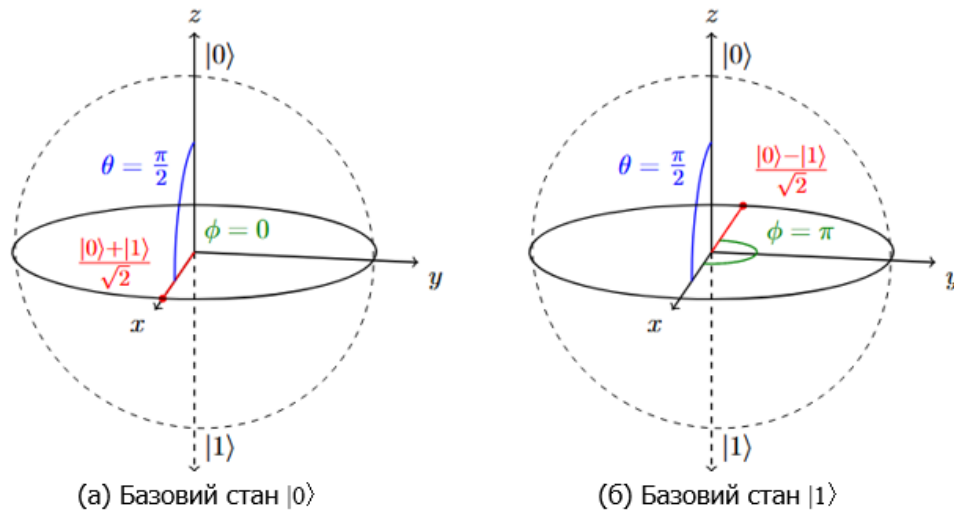


**Рис. 4.7 – Дія оператора Адамара на кубіт
(а) у базовому стані $|0\rangle$ та (б) у базовому стані $|1\rangle$.**

Різниця між амплітудами та простими ймовірностями стає очевидною, коли операції застосовуються до квантових систем, які можуть мати однакові розподіли ймовірностей, але різні амплітуди, з яких ці ймовірності виводяться.

Наприклад, розглянемо результати застосування перетворення Адамара до $|0\rangle$ та $|1\rangle$:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \langle 0|0\rangle + \frac{|0\rangle - |1\rangle}{\sqrt{2}} \langle 1|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$H|1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \langle 0|1\rangle + \frac{|0\rangle - |1\rangle}{\sqrt{2}} \langle 1|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} .$$

Отже, $H|0\rangle = |+\rangle, H|1\rangle = |-\rangle$. Розподіл ймовірностей двох результатів абсолютно однаковий, два стани відрізняються лише фазою $|1\rangle$:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} .$$

Однак, повторне застосування перетворення Адамара до кожного стану призведе до таких розподілів:

$$\begin{aligned}
H \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] &= \frac{|0\rangle + |1\rangle}{2} \langle 0|0\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|0\rangle + \frac{|0\rangle + |1\rangle}{2} \langle 0|1\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|1\rangle \\
&= \frac{|0\rangle + |1\rangle}{2} + \frac{|0\rangle - |1\rangle}{2} \\
&= |0\rangle, \\
H \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] &= \frac{|0\rangle + |1\rangle}{2} \langle 0|0\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|0\rangle - \frac{|0\rangle + |1\rangle}{2} \langle 0|1\rangle - \frac{|0\rangle - |1\rangle}{2} \langle 1|1\rangle \\
&= \frac{|0\rangle + |1\rangle}{2} - \frac{|0\rangle - |1\rangle}{2} \\
&= |1\rangle.
\end{aligned}$$

Коли одна й та сама серія операцій виконується на, здавалося б, лише незначно різних системах, ці дві системи можуть давати, і дають, різні результати. Отже, амплітуди є важливою відмінною рисою квантових комп'ютерів.

Розглянемо роботу оператора Адамара та вихід відповідного вентиля. Якщо вхід дорівнює $|0\rangle$, вихід буде $|+\rangle$. Якщо вхід дорівнює $|1\rangle$, вихід буде $|-\rangle$. Якщо вхід є суперпозицією $\alpha|0\rangle + \beta|1\rangle$, вихід буде суперпозицією $|+\rangle$ та $|-\rangle$ з однаковими амплітудами, $\alpha|+\rangle + \beta|-\rangle$, оскільки використовується властивість лінійності вентиля, тобто замість того, щоб вважати, що H є матрицею, будемо вважати, що H є лінійним оператором, і якщо H застосовується до лінійної комбінації векторів $|0\rangle$ та $|1\rangle$ з амплітудами α та β , результатом є лінійна комбінація $H|0\rangle$ та $H|1\rangle$ з однаковими амплітудами α та β . Для кубітів $|+\rangle$ та $|-\rangle$ дійсні такі перетворення: $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ та $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.

Застосування перетворення Адамара до кубіта $|0\rangle$ показано на рис. 4.8.

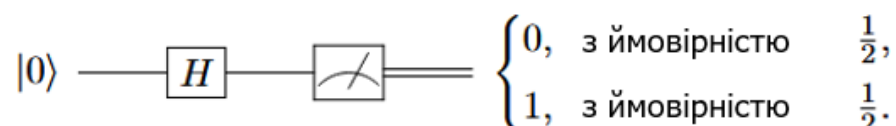


Рис. 4.8 – Застосування вентиля Адамара та вимірювання.

Можна уникнути цієї абстрактної точки зору, але коли матрицю множимо на суму векторів, необхідно розподілити множення по векторах. Перевірити, що множення H на вектори ортонормованого базису призводить до векторів норми

1, недостатньо, щоб показати, що H є унітарною матрицею. Також необхідно показати, що результуючі вектори є ортогональними, тобто показати, що $\langle - || + \rangle = 0$.

Три **вентилі Паулі**, названі на честь ще одного нобелівського лауреата, австрійського фізика Вольфганга Паулі, також є важливими однокубітними вентилями для квантових обчислень. Вентилі Паулі- X , Паулі- Y та Паулі- Z відповідають обертанням на π радіан навколо осей x , y та z відповідно на сфері Блоха. Результати використання вентилів Паулі- X та Паулі- Y , застосованих до кубіта $|0\rangle$, відрізняються лише фазою. **Принцип виключення** Паулі стверджує, що жодні дві частинки не можуть одночасно займати один і той самий квантовий стан. Результатами цього визначного дослідження є матриці спінів Паулі (X , Y , Z), які зазвичай позначаються грецькою літерою сигма (σ):

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Вентиль Паулі- X міняє місцями амплітуди $|0\rangle$ та $|1\rangle$, інвертує значення кубіта 0 на 1 і навпаки:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Схематично дію оператора Паулі- X можна представити, як на рис. 4.9.



Рис. 4.9 – Схема дії вентиля Паулі- X .

Код, який симулює дію оператора, представлений на рис. 4.10. Запуск цієї схеми призводить до появи кубіта в стані суперпозиції, що відображається в амплітудах ймовірностей стану: $[0.70710678+0.j, 0.70710678+0.j]$. Це вказує на однакову ймовірність вимірювання кубіта як у $|0\rangle$, так і в $|1\rangle$.

```

from qiskit import QuantumCircuit, Aer, execute
# Створення квантової схеми з одним кубітом
qc = QuantumCircuit(1)
# Застосування вентиля Адамара для створення суперпозиції
qc.h(0)
# Виконання схеми на симуляторі вектора станів
backend = Aer.get_backend('statevector_simulator')
job = execute(qc, backend)
result = job.result()
# Отримання та вивід результуючого вектору стану
statevector = result.get_statevector()
print(statevector)

```

Рис. 4.10 – Ілюстрація роботи вентиля Паулі-X за допомогою Qiskit.

Вентиль Паулі-Y, показаний на рис. 4.11, міняє місцями амплітуди $|0\rangle$ та $|1\rangle$, множить кожену амплітуду на i та інвертує амплітуду $|1\rangle$.

$$\text{---}[Y]\text{---} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i|1\rangle\langle 0| - i|0\rangle\langle 1|$$

Рис. 4.11 – Схема роботи вентиля Паулі-Y.

Вентиль Паулі-Z зводить нанівець амплітуду $|1\rangle$, залишаючи амплітуду $|0\rangle$ незмінною (рис. 4.12). Він еквівалентний вентилю з фазовим зсувом з $\theta = \pi$. Він не змінює значення, якщо вхід дорівнює $|0\rangle$, але змінює знак на вхідному сигналі $|1\rangle$.

$$\text{---}[Z]\text{---} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |1\rangle\langle 0| - |0\rangle\langle 1|$$

Рис. 4.12 – Схема роботи вентиля Паулі-Z.

Вентиль Паулі-Z, що змінює лише фазу системи, є окремим випадком більш загального вентиля з фазовим зсувом, який не змінює амплітуду $|0\rangle$, але змінює фазу $|1\rangle$ на коефіцієнт $e^{i\theta}$ для будь-якого значення θ (рис. 4.13).

$$\text{---}[R_\theta]\text{---} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} = |1\rangle\langle 0| + e^{i\theta}|0\rangle\langle 1|$$

Рис. 4.13 – Схема роботи вентиля з фазовим зсувом.

Інший окремий випадок вентиля з фазовим зсувом, де $\theta = \pi/2$, відомий як *фазовий вентиль*, що позначається як S , який змінює фазу $|1\rangle$ в i -й раз (рис. 4.14).

$$\text{---} \boxed{S} \text{---} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = |1\rangle \langle 0| + i |0\rangle \langle 1|$$

Рис. 4.14 – Схема роботи фазового вентиля.

Вентиль з фазовим зсувом, де $\theta = \pi/4$, називається вентиляем $\pi/8$, або T (рис. 4.15).

$$\text{---} \boxed{T} \text{---} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = |1\rangle \langle 0| + e^{i\pi/4} |0\rangle \langle 1|$$

Рис. 4.15 – Схема роботи вентиля з фазовим зсувом $\pi/8$.

Принцип множення матриць працює для будь-якого вентиля, у тому числі і нового. Це допомагає зрозуміти загальний принцип того, як вентиля пов'язані з матрицями. Загальна дія застосування вентиля до одного кубіта:

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \rightarrow \begin{bmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} q_{00}\alpha & q_{01}\beta \\ q_{10}\alpha & q_{11}\beta \end{bmatrix} = \begin{bmatrix} \alpha' \\ \beta' \end{bmatrix}.$$

Вентиль U замінює всі інші однокубітні вентиля. Загальне представлення вентиля:

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos \frac{\theta}{2} & -e^{i\lambda} \sin \frac{\theta}{2} \\ e^{i\phi} \sin \frac{\theta}{2} & e^{i(\lambda+\phi)} \cos \frac{\theta}{2} \end{bmatrix}.$$

Приклад. Як згенерувати суперпозицію так, щоб амплітуди $\alpha|0\rangle + \beta|1\rangle$ були різними та ненульовими? Наприклад, як згенерувати стан $\alpha|0\rangle + \beta|1\rangle$ такий, що $|\alpha|^2 = 25\%$ та $|\beta|^2 = 75\%$ перед вимірюванням? Відповідь полягає у використанні загального однокубітного вентиля. Після застосування $U(\theta, 0, 0)$ до $|0\rangle$, отримуємо:

$$U(\theta, 0, 0)|0\rangle = \cos \frac{\theta}{2}|0\rangle + \sin \frac{\theta}{2}|1\rangle.$$

Необхідно обрати $\theta = 2\pi/3$, оскільки $\cos^2(\pi/3) = 1/4$. Попередній приклад з використанням вентиля Адамара можна відтворити, взявши $\theta = \pi/2$, оскільки $H = U(\pi/2, 0, 0)$.

4.1 Модель квантових схем

Поєднуючи в собі стійкість для підтримки квантової когерентності та гнучкість для інтеграції нових вентилів та алгоритмів, модель квантових схем надає представлення можливостей квантових обчислень (рис. 4.16). Її комплексна структура постійно вдосконалюється, щоб відповідати значним експериментальним та теоретичним викликам та внескам, поступово просуваючись до більш повсюдних застосувань квантових обчислень.

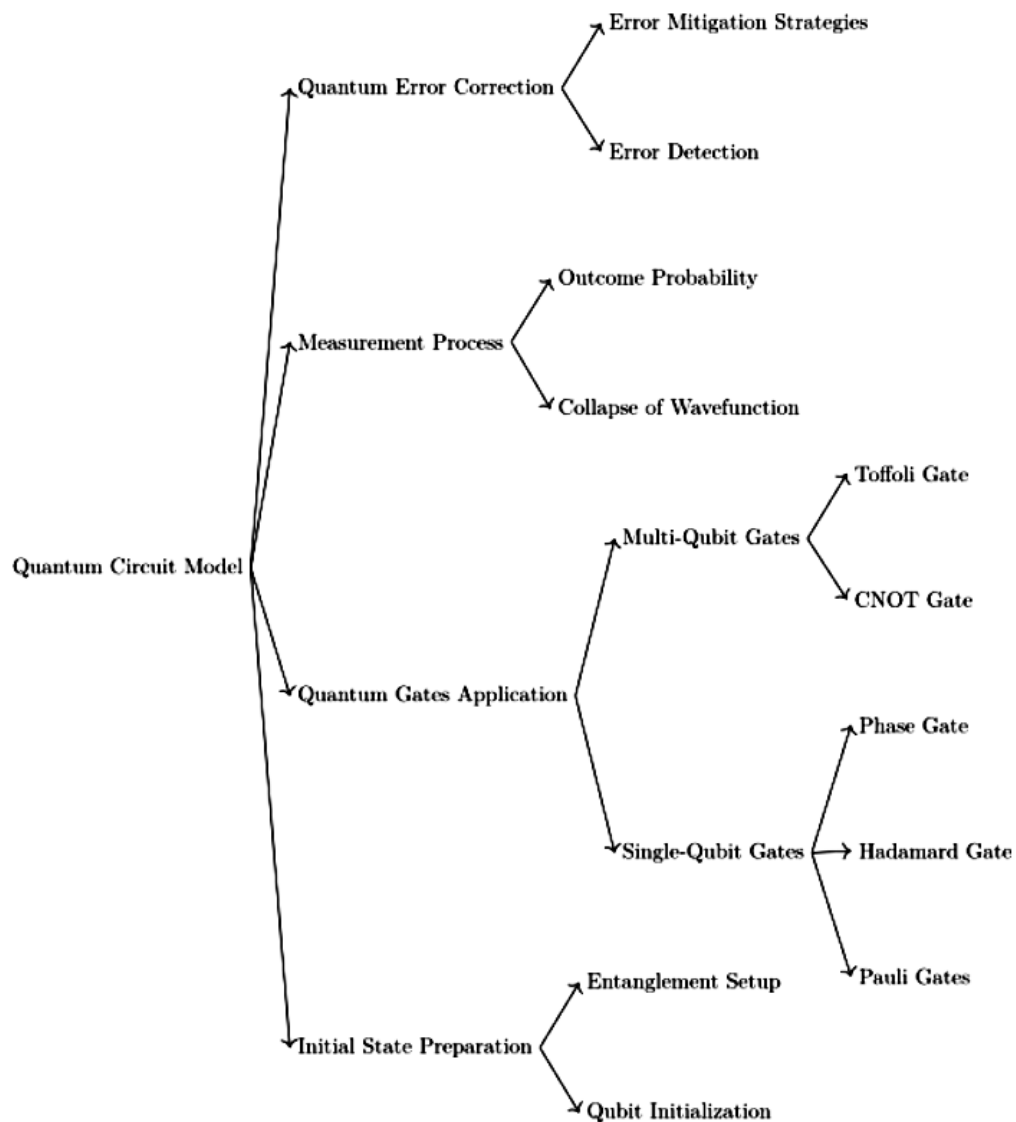


Рис. 4.16 – Модель квантових схем [53].

Побудова складних схем складається з таких етапів: ініціалізації кубітів, застосування вентилів для перетворення станів і завершення вимірюванням кубітів. Кожен етап вимагає точного контролю над кубітами для точного виконання алгоритму. Розуміння запутаності, суперпозиції та інтерференції є

основоположним, що дозволяє розробникам схем розробляти алгоритми, що використовують *квантовий паралелізм* і тунелювання.

Ключовим принципом у проєктуванні складних схем є модульний підхід. Розбиваючи алгоритми на компоненти, кожен з яких представляє підпрограму або квантову функцію, розробники забезпечують гнучкість, масштабованість та керуваність схем.

Кожен модуль відповідає за певну операцію, таку як блок перетворення Фур'є, підготовка станів або цикл виправлення помилок. Така модульна конструкція не лише спрощує представлення складних алгоритмів, але й полегшує налагодження та оптимізацію на різних квантових архітектурах.

Кожна квантова схема має еквівалентний алгебраїчний вираз. Наприклад, якщо A , B , C є однокубітними вентилями, схема, представлена на рис. 4.17, еквівалентна алгебраїчному виразу $|\psi\rangle = CBA|0\rangle$, де CBA унітарна матриця розміром 2×2 .

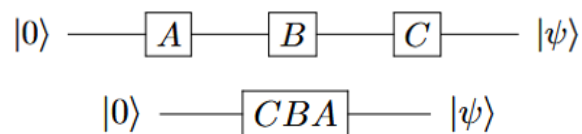


Рис. 4.17 – Приклад схеми з трьома вентилями.

Наприклад, такі схеми на рис. 4.18 є еквівалентними.

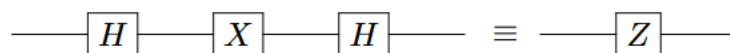


Рис. 4.18 – Еквівалентні схеми.

4.2 Глобальна та відносна фаза

Фаза – це термін, який широко використовується в квантовій механіці з різними значеннями залежно від контексту. У квантових обчисленнях використовуються два різні типи фаз: глобальний фазовий коефіцієнт (англ. Global phase factor) та відносний фазовий коефіцієнт (англ. Relative phase factor).

Глобальний фазовий коефіцієнт – це коли фаза однакова для обох амплітуд, наприклад, $e^{i\theta}(|\alpha\rangle + |\beta\rangle)$. Це означає, що ймовірність вимірювання α або β не

змінюється, оскільки для довільного фазового коефіцієнта $e^{i\theta}$ квадрат величини дорівнює 1, $|e^{i\theta}|^2 = 1$, таким чином, це не впливає на вимірювання двох станів. Це означає, що $e^{i\theta}|\psi\rangle = |\psi\rangle$, якщо $|\psi\rangle$ – вектор станів, а θ – дійсне число [54]. На жаль, з точки зору спостереження, ці стани ідентичні.

Два стани можуть відрізнятися *відносною фазою*. У цьому випадку стани $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ та $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ такі, що амплітуда $|1\rangle$ дорівнює $\frac{1}{\sqrt{2}}$ та $-\frac{1}{\sqrt{2}}$, тобто вони відрізняються відносною фазою -1 . Ці два стани не можна розглядати як ідентичні з точки зору спостереження, навіть якщо ймовірності спостереження двох станів однакові при роботі в обчислювальному базисі [55].

4.3 Квантові вентиля керованих операцій

Квантові обчислення також використовують керовані операції – багатокубітні операції, які змінюють стан кубіта на основі значень інших кубітів. Квантовий вентиль «керований-НІ» (або CNOT, у літературі також зустрічається позначення CX) змінює місцями амплітуди базисних станів $|0\rangle$ та $|1\rangle$ кубіта (рис. 4.19), що еквівалентно застосуванню вентиля Паулі-X, лише якщо керуючий кубіт має значення $|1\rangle$. Матричне представлення вентиля CX показано на рис. 4.20. Таблиця переходів вентиля CX показана на рис. 4.21.

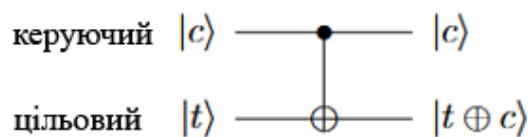


Рис. 4.19 – Квантовий вентиль «керований-НІ» (CNOT).

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Рис. 4.20 – Матричне представлення вентиля CNOT (CX).

Вхід		Вихід	
c	t	c	$c \oplus t$
$ 0\rangle$	$ 0\rangle$	$ 0\rangle$	$ 0\rangle$
$ 0\rangle$	$ 1\rangle$	$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$
$ 1\rangle$	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$

Рис. 4.21 – Таблиця переходів вентиля CX.

Але керовані операції не обмежуються застосуванням тільки вентиля Паулі-X, може бути керованою будь-яка унітарна операція (рис. 4.22).

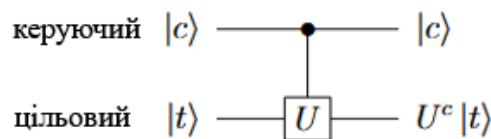


Рис. 4.22 – Узагальнена схема керованої унітарної операції.

Фактично, керовані операції можливі з будь-якою кількістю n керуючих кубітів та будь-яким унітарним оператором на k кубітах. *Вентиль Тоффолі* (англ. Toffoli) є найвідомішим з таких вентилів. Також відомий, як вентиль «керований-керований-НІ» (controlled-controlled-NOT, CCNOT, CCX), вентиль Тоффолі діє на три кубіти: як впливає з альтернативної назви, два керуючі кубіти та один цільовий (рис. 4.23). Якщо обидва керуючі кубіти встановлені, то амплітуди цільового кубіта змінюються місцями:

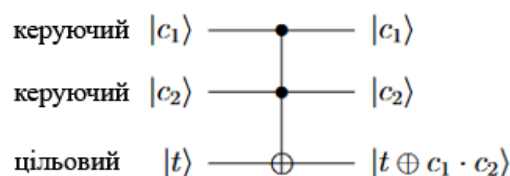


Рис. 4.23 – Вентиль Тоффолі.

Вентиль Тоффолі, спочатку розроблений Томмазо Тоффолі [56] як універсальний, зворотний (зворотний, той, результати якого можна якимось способом скасувати і повернути попередній стан) класичний логічний вентиль, є особливо важливим, оскільки залежно від вхідних даних вентиль може виконувати логічні операції I, XOR, NOT та FANOUT, що робить його універсальним для класичних обчислень. Оскільки квантові обчислення є зворотними, зворотний вентиль Тоффолі є допустимим перетворенням не лише у класичних схемах, але й у квантових схемах, що означає, що квантові

обчислення є принаймні такими ж потужними, як і класичні обчислення. Діраківське представлення 3-кубітного вентиля Тоффолі дещо громіздке, але матричне представлення досить просте (рис. 4.24).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Рис. 4.24 – Матричне представлення 3-кубітного вентиля Тоффолі.

Розуміючи кубіти, квантові регістри та квантові вентиля, які на них впливають, можна почати розуміти квантові алгоритми. Залишається ще одна математична формальність: як описати обчислювальну складність квантових систем. Зрештою, важко стверджувати, що квантовий алгоритм може розв'язати задачу швидше, ніж класичний алгоритм, не маючи можливості кількісно оцінити та порівняти обчислювальну складність. Складність алгоритмів розглянута у розділі 6.1.

4.4 Аналогія між булевими та квантовими вентилями

Логічні вентиля мають свої квантові аналоги, які можна побудувати за допомогою так званої множини Кліффорда (X, Y, Z, H, Toffoli та S). Наведемо декілька прикладів тотожностей, які корисно запам'ятати.

Логічний вентиль AND, еквівалентний квантовому вентилю Тоффолі, як показано на рис. 4.25.



Рис. 4.25 – Еквівалентність вентилів AND та Тоффолі.

Логічний вентиль XOR можна представити з 3-х кубітів та двох вентилів CNOT (рис. 4.26).

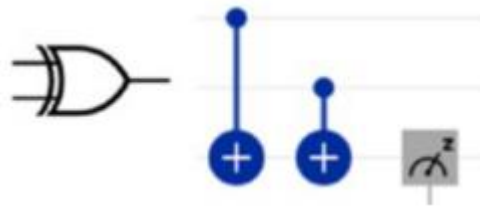


Рис. 4.26 – Еквівалентність вентилів XOR та CNOT.

Логічний вентиль OR можна представити з XOR та логічного оператора Тоффолі (рис. 4.27).



Рис. 4.27 – Еквівалентність вентиля OR та квантових вентилів.

Логічний вентиль NAND дуже важливий, оскільки будь-яку булеву функцію можна реалізувати за допомогою комбінації вентилів NAND. Ця властивість називається функціональною повнотою. NAND може бути представлений трьома кубітами, а в останньому кубіті знаходяться Тоффолі та NOT (рис. 4.28).



Рис. 4.28 – Еквівалентність вентиля NAND та квантових вентилів.

4.5 Квантова заплутаність

Два заплутані кубіти пов'язані один з одним у квантово-механічному сенсі. Це означає, що коли виконується дія з одним кубітом, відбудуться зміни і в характеристиках інших кубітів.

Щоб зрозуміти заплутаність двох кубітів, уявіть кубіти як дві кулі різних кольорів (наприклад, синього та зеленого). Помістимо дві кулі у два однакові

контейнери та перетасовуємо контейнери. У цей момент, перед відкриттям будь-якої коробки, існує така ж ймовірність, що коли обрано коробку, вона містить синю та зелену кулі. Після відкриття одного контейнера розташування кульок повністю визначено, і достеменно відомо, що містить інший контейнер, не відкриваючи його. Можна сказати, що під час «вимірювання» першої кулі відбувається також вимірювання другої кулі, щоб вона «згорнулася» до кольору, оскільки з самого початку елементи в двох контейнерах знаходилися в суперпозиціях двох кульок.

Це схоже на кота Шредінгера, оскільки після перетасовування контейнерів у кожному з них є кулька, яка є одночасно синьою та зеленою, але щойно контейнер буде відкрито, стає відомо, якого вона кольору, а також якого кольору кулька в іншому контейнері.

Рівняння $|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$ показує стан Белла, також відомий як ЕПР-пара (англ. EPR-pair), від, можливо, трьох найвідоміших дослідників у галузі квантової механіки: Ейнштейна, Подольського та Розена. Ці два кубіти у стані $|\psi\rangle$ є заплутаними, що випливає з рівняння, оскільки якщо виміряти перший кубіт і отримати 0, другий кубіт також повинен мати значення 0, оскільки єдиний стан, де перший кубіт дорівнює 0, це $|00\rangle$, і навпаки, якщо виміряти 1, інший кубіт дорівнює 1 [54].

Схема, показана на рис. 4.29, переводить систему у стан Белла.

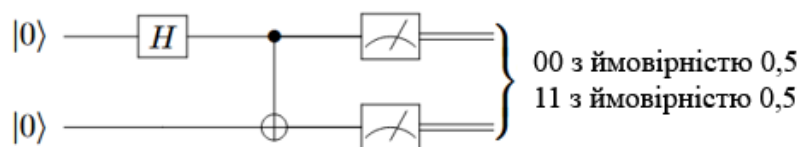


Рис. 4.29 – Схема, що переводить два кубіти у стан Белла.

Квантова заплутаність є ще одним наріжним каменем, коли два або більше кубітів стають пов'язаними таким чином, що квантовий стан одного кубіта неможливо повністю описати без урахування інших. Коли кубіти заплутані, вимірювання одного негайно впливає на стан інших, незалежно від відстані між ними, що ілюструє нелокальні кореляції і суперечать класичним уявленням.

Пара кубітів може бути заплутана у спосіб створення станів Белла, що ілюструє природу заплутаності. Чотири стани Белла представлені так:

$$\begin{aligned} |\Phi^+\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), & |\Phi^-\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \\ |\Psi^+\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), & |\Psi^-\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle). \end{aligned}$$

Кожен стан Белла втілює максимальну заплутаність для двох кубітів, де вимірювання дають корельовані результати. Код на рис. 4.30 створює стан Белла $|\Phi^+\rangle$ за допомогою квантової схеми.

```
from qiskit import QuantumCircuit, Aer, execute
# Створення квантової схеми з двома кубітами
qc = QuantumCircuit(2)
# Застосування вентиля Адамара до першого кубіта
qc.h(0)
# Заплутати кубіти за допомогою вентиля CNOT
qc.cx(0, 1)
# Виконання схеми на симуляторі вектора станів
backend = Aer.get_backend('statevector_simulator')
job = execute(qc, backend)
result = job.result()
# Отримання та виведення результуючого вектору стану
statevector = result.get_statevector()
print(statevector)
```

Рис. 4.30 – Створення стану Белла у Qiskit.

Після виконання, результуючий вектор станів підтверджує перехід схеми у стан заплутаності: $[0.70710678+0.j, 0. +0.j, 0. +0.j, 0.70710678+0.j]$. Ймовірності вказують на те, що вимірювання одного кубіта призводить до «колапсу» другого у відповідний стан, демонструючи їхню заплутаність.

4.6 Оборненість (зворотність) та унітарність

Квантові вентиля фізично реалізуються за допомогою властивостей квантової механіки, і тому вони повинні відповідати вимогам та обмеженням, пов'язаним з квантовою механікою.

Однією з ключових вимог до квантового вентиля є його *оборненість* (зворотність). Тобто, операція, яку реалізує вентиль, повинна мати можливість повернення кубіта(ів) у попередній стан, до впливу оператора (вентиля). Це

означає, що коли квантовий вентиль застосовується до заданого початкового стану, має існувати інший квантовий вентиль, який повертає результат до початкового стану кубіта.

У квантовій системі інформація не може просто зникнути. Інформація, яка була в системі до застосування певного квантового вентиля, повинна бути відновлюваною. Більшість вентилів, які розглянуті вище, є зворотними.

Однак, існують операції, які роблять квантову систему незворотною:

1. Вимірювання (англ. Measurement). Це найяскравіший приклад незворотної операції. Коли кубіт вимірюється, його квантовий стан переходить в класичний вимір, «колапсує» до одного з класичних станів (0 або 1). Вся інформація про суперпозицію та фази втрачається, і не має можливості повернути кубіт до його стану до вимірювання. Це фундаментально незворотній процес.
2. Операції, що призводять до втрати інформації. Хоча це не «вентилі» в строгому сенсі, взаємодія квантової системи з навколишнім середовищем, відома як декогеренція (англ. Decoherence), або втрата енергії (дисипація, англ. Dissipation) призводять до незворотної втрати квантової інформації. Ці процеси є однією з головних перешкод на шляху створення стабільних квантових комп'ютерів.
3. Неунітарні операції (англ. Non-unitary operations). Будь-яка операція, яка не є унітарною, є незворотною. Вимірювання є прикладом неунітарної операції. Унітарність є ключовою вимогою для зворотних квантових вентилів.

Покажемо зворотність вентилів/операторів на простому прикладі – вентилі Паулі-Х. Вентиль, який повертає систему до початкового стану після застосування вентиля Паулі-Х, є ще одним вентилям Паулі-Х. Це може бути доведено експериментально та формально. Експериментально це можна перевірити в системі Quirk [58]. На рис. 4.31 показано дію вентиля Паулі-Х. На

рис. 4.32 показано повторну дію вентиля Паулі-X та відповідні результати вимірювання кубіту.

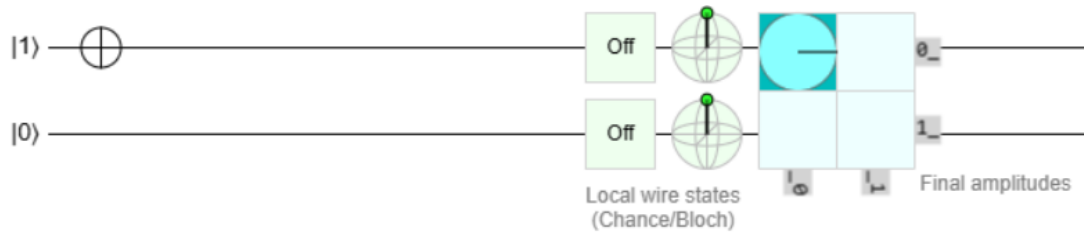


Рис. 4.31 – Дія вентиля Паулі-X в системі Quirk.

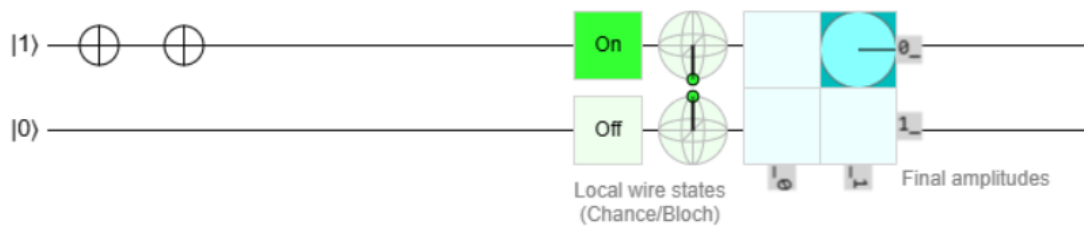


Рис. 4.32 – Дія двох вентилів Паулі-X в системі Quirk.

Формальне доведення представлено множенням матриці вентилів на вектор ймовірності кубіта:

$$\psi = \mathbf{XX} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}.$$

5 КВАНТОВА ТЕЛЕПОРТАЦІЯ ТА НАДЩІЛЬНЕ КОДУВАННЯ

Заплутаність дозволяє використовувати багато квантових протоколів, таких як квантова телепортація, надщільне кодування, і формує основу квантової криптографії.

5.1 Квантова телепортація

Квантова телепортація є алгоритмом або протоколом, який дозволяє передавати квантові стани з одного місця в інше без переміщення через проміжний простір. Це досягається у спосіб попереднього обміну заплутаним станом між відправником і одержувачем, використовуючи заплутаність і класичну комунікацію для передачі інформації про стан.

Як сказав фізик Річард Фейнман: «Якщо ви думаєте, що розумієте квантову механіку, ви її не розумієте». На щастя, в інженерії програмного забезпечення не є обов'язковим розуміти квантову механіку при її використанні у проєктах. Хоча корисно міркувати про квантово-механічні концепції, що лежать в основі квантових обчислень, не обов'язково розуміти їх при програмуванні квантових комп'ютерів. Так само, як розробники класичного програмного забезпечення не розуміють, як працює транзистор.

Квантова телепортація надсилає інформацію з кубіта, який утримує Аліса, до кубіта, який утримує Боб. Кубіт Аліси залишається в Аліси, тому фізично кубіт не передається. Однак у результаті передається деяка квантова інформація від Аліси до Боба. Звучить так, ніби це пов'язано з основною проблемою про неможливість клонувати кубіт, але якщо існує спосіб передачі його квантової інформації на відстань, це набагато наближає дослідження до створення квантової мережі.

Можна математично довести, що алгоритм телепортації дійсно телепортує інформацію від Аліси до Боба. Встановимо штучно початкові значення кубіту, який призначений для телепортації, і пізніше перевіримо, чи телепортація була успішною. Кубіт, який утримує Аліса, показано на рис. 5.1 ліворуч як кубіт q .

Передумовою для квантової телепортації є те, що Аліса та Боб мають спільну заплутану пару кубітів a та b , як показано на рис. 5.1 праворуч.



Рис. 5.1 – Ідея квантової телепортації кубіта q .

У цьому алгоритмі використовуємо три кубіти: q – кубіт для телепортації до Боба; a та b – заплутані кубіти. Застосуємо до кубіта a вентиль Адамара. Застосуємо вентиль CNOT до кубітів a та b . Кубіти q та a розташовані у Аліси, кубіт b розташований у Боба.

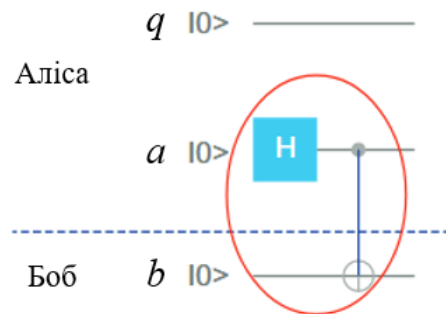


Рис. 5.2 – Перший крок алгоритму телепортації – створення заплутаності між кубітами Аліси та Боба.

На другому кроці алгоритму телепортації Аліса налаштовує взаємодію кубіту q зі своїм компонентом заплутаної пари кубітів – a .



Рис. 5.3 – Другий крок алгоритму квантової телепортації – організація взаємодії кубітів q та a .

Математично можна довести, що наступні кроки передадуть інформацію з кубіта q до кубіта b , який зберігається Бобом. Спочатку Аліса застосовує вентиль CNOT між своїм кубітом q та своєю половиною заплутаної пари. Далі вона застосовує вентиль Адамара до свого кубіта.

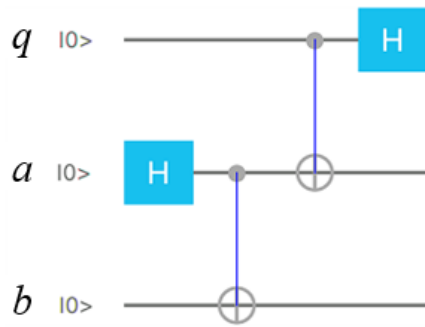


Рис. 5.4 – Реалізація другого кроку алгоритму квантової телепортації

Додаємо вентиль CNOT до кубітів q та a . Додаємо вентиль Адамара до кубіта q . На наступному кроці Аліса має виміряти свої два кубіти (рис. 5.5).

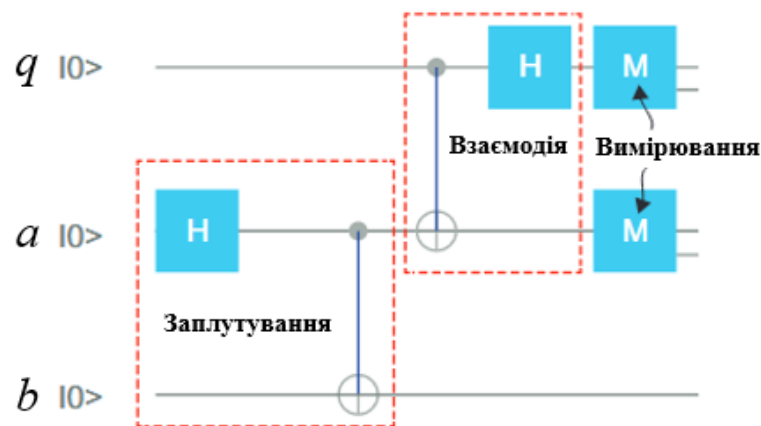


Рис. 5.5 – Вимірювання на боці Аліси в алгоритмі квантової телепортації.

Зрештою, на основі вимірювань, виконаних Алісою, Боб застосовує деякі операції до свого кубіту. Якщо вимірювання першого кубіта (вихідного кубіта, який необхідно телепортувати) дорівнює 1, Боб застосовує до свого кубіта вентиль Паулі-Х. Якщо вимірювання кубіта a дорівнює 1, Боб застосовує до свого кубіта вентиль Паулі-Z. Отже, залежно від вимірювань Аліси, Боб застосовує вентиль Паулі-Х та/або вентиль Паулі-Z (рис. 5.6).

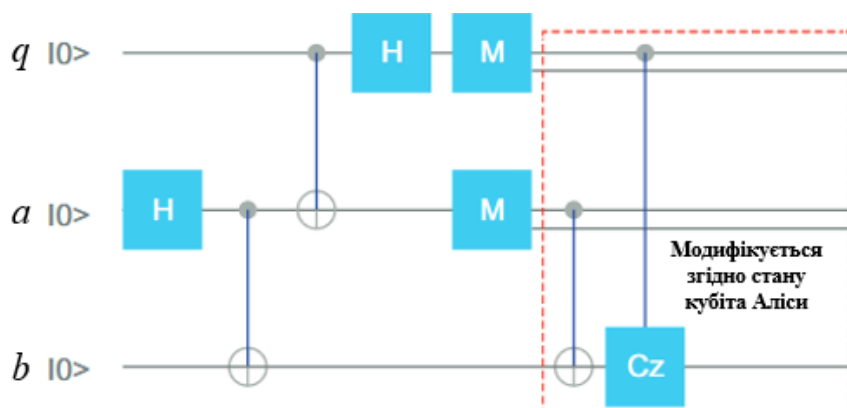


Рис. 5.6 – Реалізація другого кроку алгоритму квантової телепортації.

На рис. 5.7 показана схема алгоритму квантової телепортації з трьома кубітами:

1. Існує 50% ймовірність, що кубіт q (позначений $q[0]$) буде виміряний зі значенням 1, і 50% ймовірність, що він буде виміряний зі значенням 0.
2. Існує 50% ймовірність, що кубіт a (позначений $q[1]$) буде виміряний зі значенням 1, і 50% ймовірність, що він буде виміряний зі значенням 0.
3. Кубіт b (позначений $q[2]$) гарантовано буде виміряний зі значенням 0.

Остання частина є найважливішою. Спочатку q містив значення 0. В кінці схеми телепортації значення q не визначається, але значення b тепер дорівнює 0. Отже, інформація з кубіта q телепортується до кубіта b .

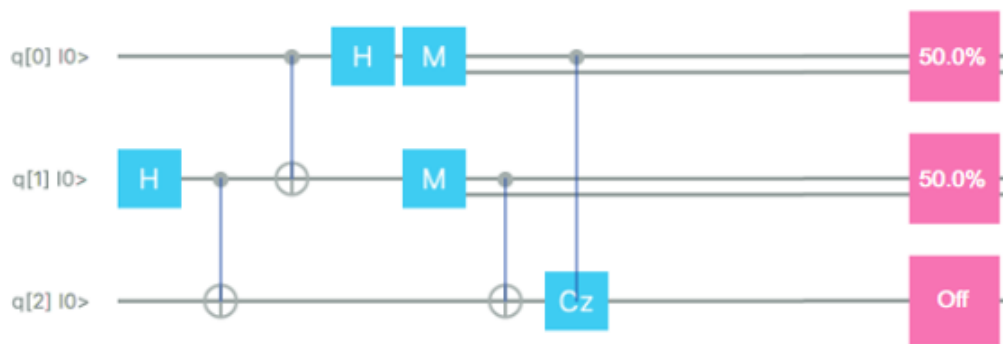


Рис. 5.7 – Схема алгоритму квантової телепортації.

На рис. 5.8 показано статистичні результати 1000 запусків схеми квантової телепортації. Вісь x показує потенційні результати (комбінації значень кубітів), вісь y – кількість вимірювань комбінації кубітів.



Рис. 5.8 – Кількість вимірювань для комбінацій кубітів 000, 001, 010, 011.

Перша комбінація (000) відповідає випадку, коли всі три кубіти вимірюються зі значенням 0. Друга комбінація (001) стосується випадку, коли перший кубіт ($q[0]$ або q) вимірюється значенням 1, а інші кубіти вимірюються зі значенням 0.

З 1000 запусків:

- близько 250 мали результат 000 (отже, всі кубіти були виміряні зі значенням 0).
- близько 250 мали результат 001, що означає, що q було виміряно зі значенням 1, а a та b були виміряні зі значенням 0.
- близько 250 запусків мали результат 010, що означає, що q було виміряно зі значенням 0, a було виміряно зі значенням 1, а b було виміряно зі значенням 0.
- близько 250 запусків мали результат 011, що означає, що q було виміряно зі значенням 1, a було виміряно зі значенням 1, а b було виміряно зі значенням 0.

Важливо зіставити кубіти на рис. 5.7 з результатами на рис. 5.8. У квантовому ланцюзі кубіт з індексом 0 (найменш значущий кубіт) показуємо зверху (рис. 5.9). Кубіт з найвищим індексом (найбільш значущий кубіт) знаходиться внизу. При записі кубітів послідовно, починаємо з найбільш значущого кубіта ліворуч і закінчуємо найменш значущим кубітом праворуч.

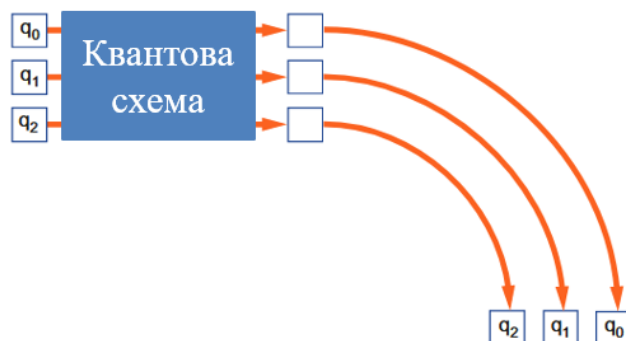


Рис. 5.9 – Порядок кубітів, записаних у вигляді послідовності.

Звернемо увагу, що у всіх цих випадках b вимірюється як 0. Інші кубіти можуть містити або 0, або 1, але b завжди дорівнює 0. Цей результат вказує, що схема/алгоритм телепортує кубіт від Аліси до Боба. Принаймні, коли значення кубіта Аліси дорівнює 0, результуюче значення кубіта Боба також дорівнює 0.

5.2 Формальний опис квантової телепортації

Як було зазначено у попередньому підрозділі, використовуючи заплутані стани, реалізовується важлива властивість, яка називається *квантовою телепортацією*. Метою квантової телепортації є переміщення квантового стану між двома точками (кубітами), що знаходяться в різних місцях, навіть далеко одна від одної. У 2010 році Джин та ін. [57] змогли відправити стан на 16 км з точністю 89%.

Припустимо, що дві людини, Аліса та Боб, які зустрічаються один раз, обмінюються номерами телефонів, а також створюють стан Белла з двома кубітами. Потім вони віддаляються далеко один від одного та забирають по одному кубіту з собою. Тепер Аліса хоче відправити Бобу квантовий стан $|\psi\rangle$, але вона не може перевірити цей стан, оскільки він би зруйнувався і більше не був би квантовим станом, якби вона це зробила. Отже, їй неможливо описати цей стан Бобу будь-яким чином.

Однак, Аліса може виконувати операції над своєю частиною стану Белла разом зі станом $|\psi\rangle$ та вимірювати їх, щоб отримати один із чотирьох різних результатів, а саме 00, 01, 10 або 11, та повідомити результат операцій Бобу. Після отримання результату Боб може відтворити стан $|\psi\rangle$.

Розглянемо приклад виконаних обчислень, які ілюструють загальний спосіб виконання квантової телепортації, навіть якщо вхідні параметри можуть змінюватися, наприклад, замість обраного нижче може бути використаний інший стан Белла [54].

Стан Белла, який мали Аліса та Боб, дорівнює $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, де вказуємо, що Боб має другий кубіт, тобто $|00\rangle$ та $|11\rangle$. Стан $|\psi\rangle$ знаходиться у стані суперпозиції в обчислювальному базисі, тобто $\alpha|0\rangle + \beta|1\rangle$. Це означає, що стан усіх кубітів такий:

$$|\psi_0\rangle = |\psi\rangle |\beta_{00}\rangle = \frac{1}{\sqrt{2}} \left[\alpha |0\rangle (|00\rangle + |11\rangle) + \beta |1\rangle (|00\rangle + |11\rangle) \right].$$

Потім Аліса застосовує CNOT-вентиль до своїх двох кубітів, з контролем над кубітом у стані $|\psi\rangle$ і ціллю на своєму кубіті в стані Белла. Це змінить стан на:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}} \left[\alpha |0\rangle (|00\rangle + |11\rangle) + \beta |1\rangle (|10\rangle + |01\rangle) \right].$$

Після цього Аліса застосовує оператор Адамара до стану $|\psi\rangle$, щоб досягти виразу:

$$|\psi_2\rangle = \frac{1}{2} \left[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle) \right].$$

Цей стан можна переписати так:

$$\begin{aligned} |\psi_2\rangle = \frac{1}{2} \left[|00\rangle (\alpha |0\rangle + \beta |1\rangle) + |01\rangle (\alpha |1\rangle + \beta |0\rangle) \right. \\ \left. + |10\rangle (\alpha |0\rangle - \beta |1\rangle) + |11\rangle (\alpha |1\rangle - \beta |0\rangle) \right]. \end{aligned}$$

Розіб'ємо останнє рівняння на **чотири** різні члени залежно від того, який стан вимірюються кубітами Аліси, тобто проаналізуємо два кубіти поза дужками. Якщо кубіти Аліси знаходяться у стані $|00\rangle$, то кубіт Боба має вигляд $\alpha|0\rangle + \beta|1\rangle$, що є станом $|\psi\rangle$, який треба телепортувати. Однак, якщо в Алісі є інша комбінація її кубітів під час вимірювання, Бобу доведеться застосувати певний вентиль (або вентиля) до свого кубіта, щоб отримати стан, який мав бути телепортований. Вентиль, який застосовується залежно від кубітів Аліси, можна побачити на рис. 5.10.

Аліса	Боб	Вентиль
$ 00\rangle$	$\alpha 0\rangle + \beta 1\rangle$	No gate
$ 01\rangle$	$\alpha 1\rangle + \beta 0\rangle$	X-gate
$ 10\rangle$	$\alpha 0\rangle - \beta 1\rangle$	Z-gate
$ 11\rangle$	$\alpha 1\rangle - \beta 0\rangle$	X-gate then Z-gate

Рис. 5.10 – Застосування вентилів відповідно до стану кубітів Аліси.

Щоб вибрати, який вентиль застосовувати, треба аналізувати кубіти Аліси, якщо кубіти у стані $|01\rangle$, то застосовується Х-вентиль до кубітів Боба, амплітуди змінюються з $\alpha|1\rangle + \beta|0\rangle$ на $\alpha|0\rangle + \beta|1\rangle$.

Перевіримо роботу алгоритму, якщо кубіт Аліси, який треба телепортувати, має значення 0 (рис. 5.11) та 1 (рис. 5.12). У такому випадку очікуваний результат алгоритму завжди матиме значення $b=0$, або $b=1$ відповідно. Для цього створимо схеми за допомогою симулятора [58]. Якщо правильно побудувати схему, можна переконатися, що алгоритм також працює, коли кубіт знаходиться в стані 1.

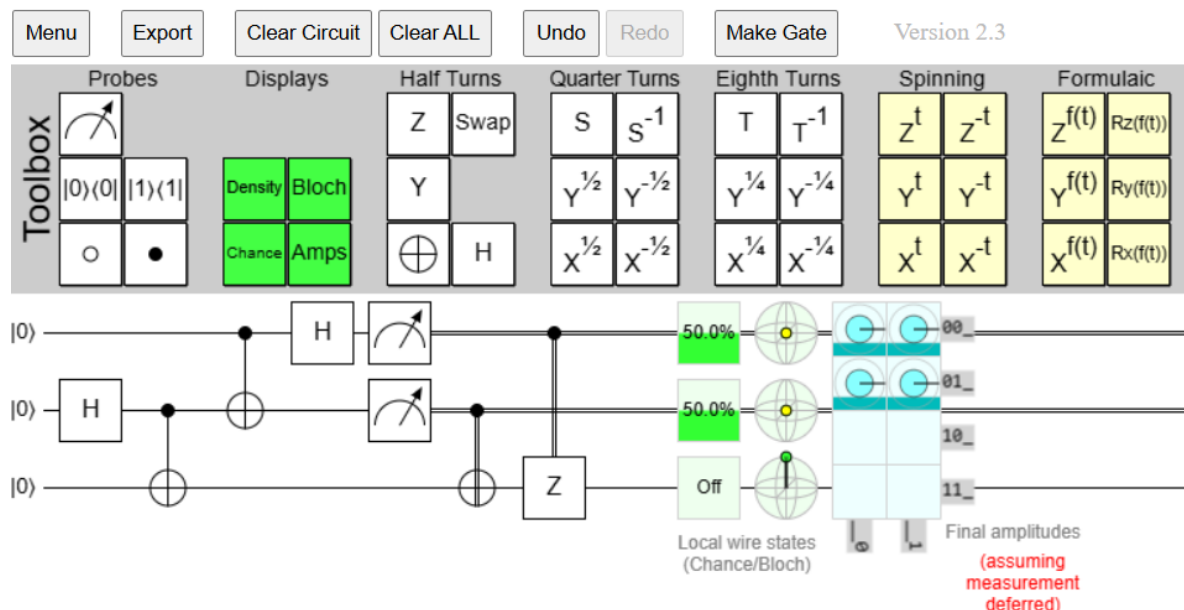


Рис. 5.11 – Схема квантової телепортації в симуляторі [58], кубіт Аліси у стані 0.

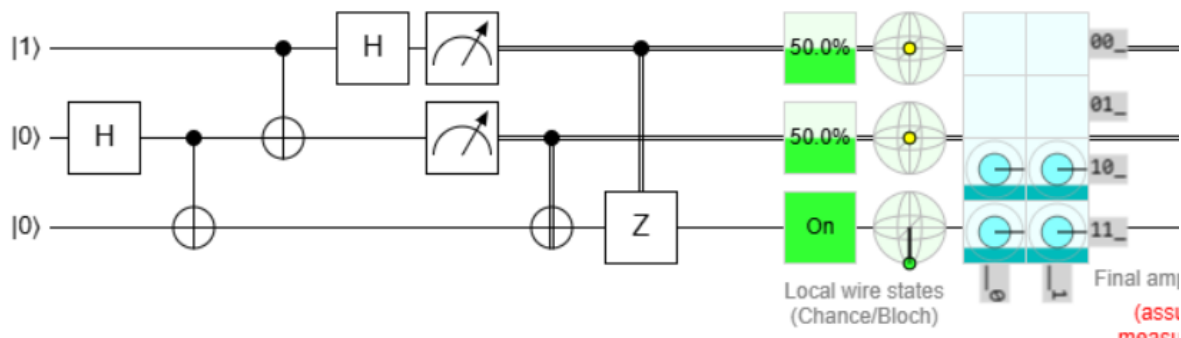


Рис. 5.12 – Схема квантової телепортації в симуляторі, кубіт Аліси у стані 1.

Але якщо кубіт знаходиться в суперпозиції? Наприклад, встановимо значення початкового кубіта $q[0]=0.4$ (рис. 5.13). Ймовірність вимірювання 0 дорівнює квадрату $q[0]$, що означає, що ймовірність вимірювання 0 становитиме 16%, а ймовірність вимірювання 1 становитиме 84%.

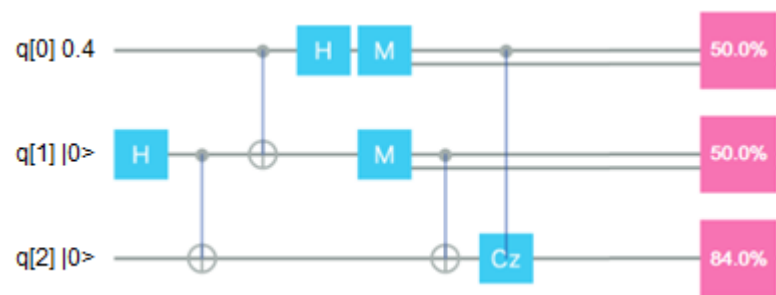


Рис. 5.13 – Схема алгоритму квантової телепортації для кубіту в стані, відмінному від 0.

На рис. 5.13 показано, що тепер існує 84% ймовірність, що кубіт Боба $q[2]$ вимірюється як 1. Рис. 5.14 показує подібний результат. Алгоритм телепортує квантову інформацію, яка спочатку містилась у кубіті, що утримується Алісою, до кубіта, що утримується Бобом. Математичний доказ можемо знайти у статті [19].

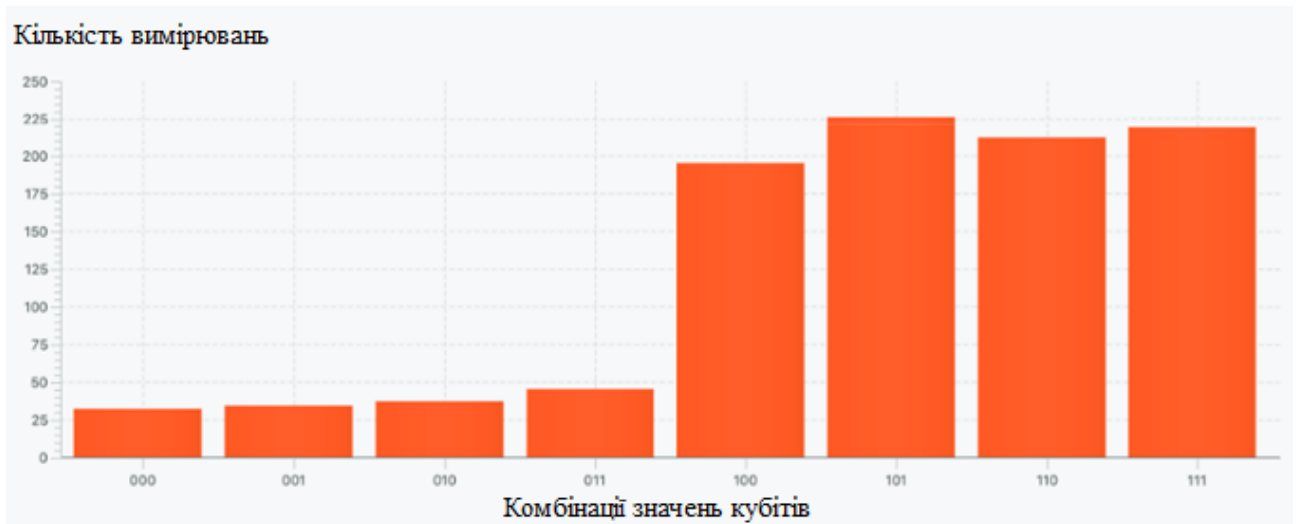


Рис. 5.14 – Кількість вимірювань для усіх комбінацій кубітів.

5.3 Загальна схема квантових обчислень та квантовий паралелізм

Загальна схема роботи квантового алгоритму представлена на рис. 5.15. Початковий стан кожного кубіта – $|0\rangle$. Потім до кожного кубіта застосовується вентиль Адамара, готуючись до квантового паралелізму. Далі до всіх кубітів застосовується унітарний оператор U . Нарешті, для кожного кубіта існує лічильник, який повертає бітовий рядок.

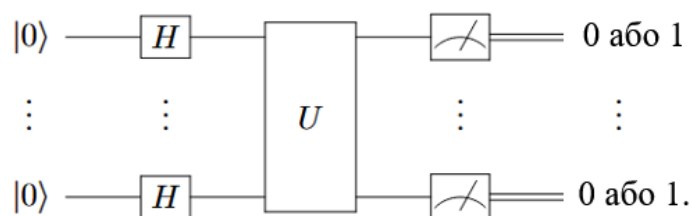


Рис. 5.15 – Загальна схема роботи квантового алгоритму.

Квантовий паралелізм – це одночасне виконання алгоритму з більш ніж одним входом до одного квантового процесора.

Для виконання того самого завдання на класичному комп'ютері знадобиться експоненціальна кількість класичних процесорів.

Щоб зрозуміти цю концепцію, зосередимося лише на операторі U , не враховуючи вентилі H . U та лічильники можна інтерпретувати як алгоритм з використанням ймовірностей у класичному сенсі. Якщо x – n -бітовий рядок, а U – оператор (матриця $2^n \times 2^n$), то $U|x\rangle$ – лінійна комбінація 2^n векторів обчислювального базису. Після вимірювань повертається n -бітовий рядок y , де

$|y\rangle$ – один з членів лінійної комбінації. Тобто, якщо вхід до U – x , то вихід після вимірювання – y . U – це «алгоритм», який виконується для одного вхідного значення x під час обчислення $U|x\rangle$.

5.4 Надщільне кодування

Надщільне кодування (англ. Super dense coding, SDC) – це протокол передачі даних, який демонструє можливості квантової системи щодо обробки інформації. Формально SDC – це метод, який дозволяє передавати два класичні біти інформації іншій стороні за допомогою одного кубіта.

На рис. 5.16 показана схема з набором таких абстракцій: модуль генератора пари Белла для створення двох заплутаних кубітів; модуль кодування інформації, який застосовує правила Аліси для кодування двох класичних бітів інформації; модуль декодера, який отримує класичні біти з кубітів, наданих парою Белла, а також модулем кодера.

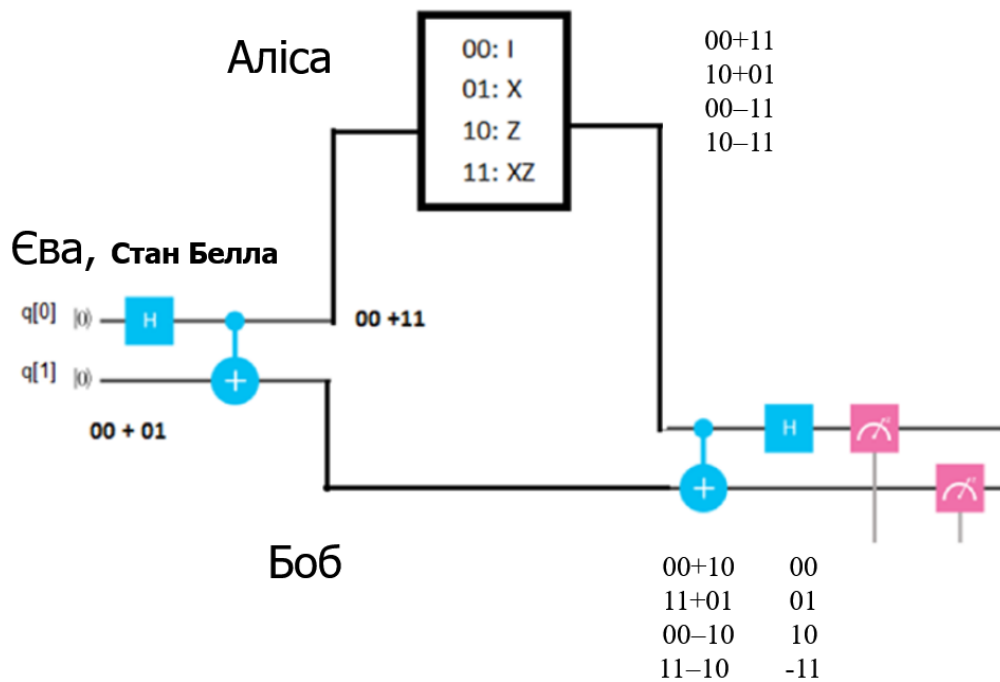


Рис. 5.16 – Схема надщільного кодування.

Процес починається з того, що третя сторона (Єва) генерує так звану пару Белла. Початковий базисний стан двох кубітів – $|0\rangle$. До першого кубіта застосовується вентиль Адамара для створення суперпозиції. Потім

застосовується вентиль $CNOT$, використовуючи перший кубіт як контрольний ($\cdot$), а другий як цільовий ($+$). Це призводить до таких станів:

1. Коли вентиль H застосовується до першого кубіта, кубіт переходить у стан суперпозиції; таким чином, стани кубітів будуть такі: $|00\rangle + |10\rangle$, де другий кубіт залишається у стані 0. Зауважте, що квадратний корінь з матриці Адамара опущено для спрощення.

2. Вентиль $CNOT$ переводить обидва кубіти у стан квантового заплутування. Зокрема, він інвертує цільовий кубіт ($+$), якщо керуючий ($\cdot$) дорівнює одиниці, інакше цільовий кубіт залишається незмінним. Таким чином, другий кубіт інвертується, якщо перший дорівнює 1, що призводить до стану $|00\rangle + |11\rangle$.

На другому кроці процесу перший кубіт надсилається Алісі, а другий – Бобу. Зауважте, що Аліса та Боб можуть перебувати на великій відстані. Мета протоколу полягає в тому, щоб Аліса надіслала Бобу два класичні біти інформації за допомогою свого кубіта. Але перш ніж вона це зробить, їй потрібно застосувати набір квантових правил (вентилів) до свого кубіта залежно від двох бітів інформації, які вона хоче надіслати (табл. 5.1).

Табл. 5.1 – Перетворення при надсиланні бітів Алісою.

Правила (Аліса має надіслати)	Результати
00: I (вентиль ідентичності)	$I(00+11) = 00 + 11$
01: X	$X(00+11) = 10 + 01$
10: Z	$Z(00+11) = 00 - 11$
11: ZX	$ZX(00+11) = 10 - 11$

Таким чином, якщо Аліса надсилає 00, вона нічого не робить зі своїм кубітом (застосовує вентиль ідентичності). Якщо вона надсилає 01, то вона застосовує вентиль X (або перевертання бітів). Для 10 вона застосовує вентиль Z . Зауважте, що вентиль Z змінює знак (фазу) кубіта, якщо кубіт дорівнює 1. Таким чином, $Z|0\rangle = |0\rangle, Z|1\rangle = -|1\rangle$. Нарешті, якщо вона надсилає 11, то вона застосовує

вентилі XZ до свого кубіта. Потім Аліса надсилає свій кубіт Бобу для останнього кроку в процесі.

Боб отримує кубіт Аліси (кубіт $q[0]$) і використовує свій кубіт, щоб здійснити зворотний процес стану Белла, створеного Євою. Він застосовує вентиль $CNOT$ до першого кубіта, потім вентиль Адамара (H) і, нарешті, виконує вимірювання в обох кубітах, щоб отримати два класичні біти, закодовані в кубіті Аліси.

Вентиль	Результати	Опис
$CNOT$	$00 + 10$ $11 + 01$ $00 - 10$ $11 - 10$	Перетворення починається зі станів Аліси з кроку 2: $00 + 11$ $10 + 01$ $00 - 11$ $10 - 11$. Вентиль $CNOT$ інвертує другий кубіт, якщо перший дорівнює 1, що призводить до станів у стовпці результати.
H	00 01 10 -11	Застосування вентилля Адамара до першого кубіта в останньому рядку призводить до таких результатів. Коли Боб виконує вимірювання в обчислювальних базисних станах, він отримує чотири можливі результати з ймовірністю 1 кожен. Ці результати відповідають тому, що Аліса мала намір надіслати на кроці 2 у стовпці 1. Однак останній результат має від'ємний знак. Тим не менш, оскільки ймовірність обчислюється як квадрат амплітуди, -1 стає 1.

На рис. 5.17 показана схема реалізації надщільного кодування. У цьому конкретному випадку закодовані біти дорівнюють 11 . Зверніть увагу, що бар'єр блокуватиме виконання, доки всі вентиля не будуть використані для обох кубітів. Запускаємо схему в симуляторі, і результатом має бути стовпчаста діаграма з ймовірністю результату 11 , дуже близькою або рівною 1 .



Рис. 5.17 – Схема реалізації надщільного кодування.

Зауважте, що під час виконання в симуляторі ймовірність завжди буде 1. Але, якщо схему запустити на реальному квантовому пристрої, через шум та вплив навколишнього середовища, кількість підрахунків має бути меншою за 1024, що призведе до ймовірності, меншої за 1. Таким чином, надщільне кодування забезпечує засоби для кодування двох класичних бітів в одному кубіті.

Зауважте, у квантових обчисленнях вважається, що неможливо зберігати більше одного класичного біта за допомогою одного кубіта, що, здається, суперечить тому, що було показано в цьому протоколі. Насправді, суперечності немає. Протокол працює, тому що кубіти Аліси та Боба заплутані через пару у стані Белла. Це дозволяє надсилати два класичні біти в заплутаному кубіті Аліси. У загальному випадку можна зберігати максимум 2 класичні біти за допомогою одного кубіту за умови, що кубіт заплутаний з іншим через пару Белла.

Але зроблено припущення, що пару Белла можна якимось чином передати між Алісою та Бобом, хоча насправді це складне завдання, яке потребує подальших досліджень.

6 ПРОЕКТУВАННЯ КВАНТОВИХ АЛГОРИТМІВ І СХЕМ

6.1 Складність алгоритмів

У галузях інженерії програмного забезпечення та комп'ютерних наук одним з найважливіших є не питання «чи можна розв'язати задачу?», а питання «наскільки ефективно можна розв'язати задачу?». *Складність алгоритмів* – це кількісна характеристика ресурсів (найчастіше часу та пам'яті), яка дозволяє оцінити, скільки ресурсів потребує алгоритм для виконання залежно від розміру (обсягу) вхідних даних.

Для математичного опису складності використовують дві основні метрики, які аналізують за допомогою асимптотичного аналізу (зокрема, нотації Ландау або *O-великого*):

1. *Часова складність* (англ. Time Complexity) – функція $T(n)$, яка визначає кількість елементарних операцій (або процесорного часу), що здійснює алгоритм для обробки вхідних даних розміром n .
2. *Просторова складність* (англ. Space Complexity) – функція $S(n)$, яка визначає обсяг оперативної пам'яті (або кількості кубітів/бітів), що додатково виділяється під час виконання алгоритму для вхідних даних розміром n .

При аналізі складності алгоритмів використовують *асимптотичний аналіз* для оцінки не точного часу у секундах, а тенденції зростання витрат при великих значеннях n . Це дозволяє абстрагуватися від швидкодії конкретного процесора. Складність завжди розглядають у вигляді трьох граничних випадків:

- *найгірший випадок* (англ. Worst-case) – верхня межа ресурсів (*O*-нотація). Основний стандарт в академічних дослідженнях;
- *середній випадок* (англ. Average-case) – математичне очікування витрат на випадкових даних (Θ -нотація);
- *найкращий випадок* (англ. Best-case) – мінімально можливі витрати (Ω -нотація).

Припустимо, що потрібно знайти конкретну книгу в бібліотеці. Якщо книги розставлені випадково, то доведеться переглядати їх одну за одною, використовуючи лінійний пошук з часовою складністю $O(n)$, де n – кількість книг. Але якщо книги відсортовані за алфавітом, є можливість використати бінарний пошук зі складністю $O(\log n)$, що значно швидше для великих бібліотек.

Розглянемо питання складності алгоритмів на прикладі пошуку простих чисел, добуток яких є заданим числом. Якщо задане число, яке складається з однієї або двох цифр, то для нього знайти множники не є складним завданням. Але при збільшенні цифр у числі завдання пошуку простих чисел-множників значно ускладнюється.

Позначимо через $T(n)$ час (кількість кроків), необхідний для розв'язання задачі пошуку двох чисел, добуток яких дорівнює числу завдовжки n знаків. Складність визначає, як росте $T(n)$ зі збільшенням n . Чи можна знайти позитивні числа k і p , за яких виконуватиметься умова $T(n) \leq kn^p$ для всіх значень n ? Якщо можна, то завдання можна вирішити за *поліноміальний час*. Якщо можна знайти позитивні числа $k > 1$ і $c > 1$, за яких виконуватиметься умова $T(n) > kc^n$ для всіх значень n , то завдання може бути вирішене за *експоненційний час*. *Основна властивість поліноміального та експоненційного зростання*: за наявності достатньої кількості часу величина з експоненційним зростанням буде зростати набагато швидше за величину з поліноміальним зростанням.

Задачі, які можна розв'язати за поліноміальний час, в інформатиці вважаються *розв'язуваними*, а задачі, які можна розв'язати за експоненційний час, *не розв'язуються*. Задачі, які розв'язуються за поліноміальний час, вважаються *простими*, а розв'язувані за експоненційний час – *складними*.

Для більшості практичних задач, що розв'язуються за поліноміальний час, цей час описується поліномом низького ступеня, тому якщо в даний момент немає технічної можливості розв'язати задачу з великим значенням n , така можливість цілком може з'явитися через кілька років.

У випадку із задачами з експоненційним часом розв'язання, навпроти, якщо n перевищить значення, для якого задачу можна розв'язати при поточному рівні розвитку техніки, то навіть невелике збільшення n може зробити задачу нерозв'язною в найближчому майбутньому.

Для аналізу кількості обчислень, необхідних алгоритмам, використовують терміни *час* і *простір*, необхідні для виконання алгоритму. Зазвичай алгоритми мають постійний, поліноміальний або експоненційний розмір вхідних даних.

Формально, алгоритм з n -бітним входом має один із наступних класів складності: $O(C)$ – константна складність $C > 0$; $O(n)$ – лінійна складність; $O(\log(n))$ – логарифмічна складність; $O(n \log(n))$ – лінійно-логарифмічна складність; $O(n^x): x < n$ – поліноміальна складність; $O(c^n): c > 1$ та $O(n!)$ – експоненційна складність. $O(\cdot)$ означає, що швидкість зростання часу виконання алгоритму асимптотично обмежена виразом всередині $O(\cdot)$. Якщо мова йде про обчислювально-складні задачі, зазвичай мають на увазі задачі, які не можна розв'язати швидше, ніж за експоненційний час. Існують алгоритми, час виконання яких, $O(n!)$. Наприклад, щоб знайти найкоротший маршрут, що проходить через всі міста рівно один раз (задача комівояжера) наївний підхід має складність $O(n!)$.

Розуміння різниці між поліноміальною та експоненційною складністю критично важливе для: *вибору алгоритмів*, що застосовуються для великих даних, при цьому перевага надається алгоритмам з меншою складністю; *при оптимізації* рекомендується використати наближений алгоритм з поліноміальною складністю, ніж точний з експоненційною; *при проектуванні систем*, щоб передбачити масштабованість рішення.

Задачі можна розбити на класи за *часовою складністю* їх розв'язання, наскільки швидко є можливість знайти правильну відповідь. Також часто використовується класифікація задач за обсягом необхідної пам'яті (*просторова складність*).

Клас задач P (поліноміальні) – це ті задачі, які можна розв’язати на класичному детермінованому комп’ютері за поліноміальний час, їх складність можна представити як $O(n^k)$. Відповідно, розв’язання задач цього класу не є витратним за часом, за винятком випадків, коли ступінь полінома висока, хоча такі алгоритми і не є типовими. Це задачі, які вважаються «легкими» або ефективно обчислюваними для класичних систем, наприклад, пошук найкоротшого шляху на графі, сортування масиву, перевірка парності числа.

Задачі другого класу, NP (недетерміновано поліноміальні) – це клас задач, для яких невідомо поліноміальне класичне рішення. При цьому перевірка розв’язків можлива за поліноміальний час на класичному комп’ютері. Знайти розв’язок може бути неймовірно важко, але якщо отримано правильну відповідь, підтвердити її правильність можна миттєво, наприклад, задача про рюкзак, задача розкладання числа на прості множники (факторизація), задача MaxCut. До прикладу, знаходження простих множників великого числа за допомогою відомих класичних алгоритмів – субекспоненціально складна задача (працює за час, більше, чим поліноміальний, але менше, чим експоненційний). Перевірити знайдений розв’язок дуже просто, достатньо перемножити знайдені прості числа.

Серед NP -задач є найбільш складні завдання, які виділяють в спеціальну групу під назвою NP -повні задачі (англ. NP -complete). Якщо знайти для них швидкий розв’язок, цей спосіб рішення також можна застосувати і до звичайних NP -задач. Є також задачі, які, хоча і не відносяться до класу NP , але, незважаючи на це, до них можна звести задачі з NP -класу. Разом з NP -повними задачами вони утворюють клас NP -важких задач (англ. NP -hard).

Наприклад, до NP -складних задач відноситься задача комівояжера, в якій потрібно встановити найкоротший шлях. Як рішення, так і його перевірка, в цьому випадку займає суперполіноміальний час, так що ця задача не входить до класу NP -повних. Якщо цю задачу трохи спростити, щоб розв’язком був шлях не довше заданого, то така різновидність задачі є прикладом класу NP -повних, оскільки перевірка її розв’язку не вимагає часу, більшого, ніж поліноміальний.

Якщо дати можливість класичному комп'ютеру розв'язувати задачі з привнесенням випадковості, то комп'ютер отримує правильну відповідь з високою ймовірністю (стандартно беруть поріг не менше $2/3$, хоча достатньо, щоб ймовірність була константою більше 0.5), у цьому випадку мова йде про клас задач *BPP* (англ. Bounded-error, probabilistic, polynomial). Такі задачі можна розв'язати за поліноміальний час, причому точність рішення можна скільки завгодно збільшувати, повторно запускаючи алгоритм.

Квантовий аналог таких задач – клас *BQP* (англ. Bounded error quantum polynomial time). Це задачі, які можна розв'язати на квантовому комп'ютері за поліноміальний час із ймовірністю помилки не більше $1/3$ (або будь-якої іншої константи, меншої за $1/2$), забезпечуючи точність розв'язку повторним запуском алгоритму. *BQP* окреслює межі того, на що теоретично здатні ідеальні квантові комп'ютери. Найбільш відомий приклад такої задачі – факторизація чисел, яка розв'язується на квантовому комп'ютері за допомогою **алгоритму Шора**. *BQP* клас включає в себе, в тому числі, і задачі, для яких не відоме поліноміальне класичний розв'язок, що дозволяє ставитись до квантових обчислень з оптимізмом. Зв'язок між класами складності задач показаний на рис. 6.1. У комп'ютерних науках загальноприйнятою (хоч і не доведеною повністю) є така структура включення: $P \subseteq BPP \subseteq BQP \subseteq PSPACE$.

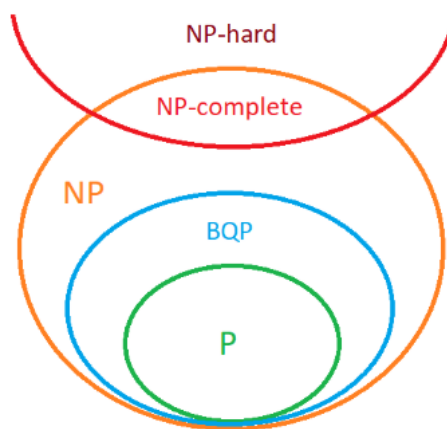


Рис. 6.1 - Зв'язок між класами складності задач

Сучасне наукове бачення зв'язку між BQP та NP можна представити так:

- $P \subseteq BQP$ означає, що задачі, які легко розв'язуються на класичному комп'ютері, можна так само ефективно розв'язати на квантовому (квантовий комп'ютер може імітувати класичні логічні вентиля);
- $BQP \not\subseteq NP$ означає, що ймовірно існують специфічні задачі, які квантовий комп'ютер швидко розв'язує (в BQP), але класичний комп'ютер не може навіть швидко *перевірити* готову відповідь;
- $NP \not\subseteq BQP$ означає, що ймовірно квантові комп'ютери не є всемогутніми. Попри поширений міф, вони не можуть розв'язувати абсолютно всі складні задачі. Наприклад, для NP -повних задач (найважчих у класі NP , як-от задача комівояжера) квантовий алгоритм Гровера забезпечує лише квадратичне прискорення $O(\sqrt{N})$, що не переводить задачу в поліноміальний час (BQP).

Квантовий алгоритм наближеної оптимізації та алгоритм варіаційного квантового розв'язувача власних значень створювалися як перехідний етап між застосуванням алгоритмів з поліноміальною складністю та алгоритмами BQP . QAOA та VQE орієнтовані на задачі класу NP (наприклад, MaxCut) або на складні задачі матеріалознавства та квантової хімії. Оскільки сучасні вентиляльні квантові системи, які належать до зашумлених квантових систем середнього масштабу, що є занадто шумними для точних поліноміальних алгоритмів (як алгоритм Шора), VQE та QAOA використовують евристичний гібридний підхід. Вони не гарантують точного розв'язку за поліноміальний час у строгому математичному сенсі, але дозволяють отримати *достатньо хороше* наближення.

6.2 Проєктування квантових алгоритмів

Проєктування квантових алгоритмів передбачає формулювання обчислювальних процедур, які використовують принципи квантової механіки для ефективнішого розв'язання задач.

Квантова парадигма пропонує унікальні принципи, такі як **суперпозиція**, **заплутаність** та квантова **інтерференція**, які можна використовувати для дослідження нових підходів до розв'язання задач. Процес проєктування вимагає нового погляду на розв'язання задач, що змінює традиційні уявлення, що визначають розробку алгоритмів для класичних комп'ютерів. Для проєктування квантових алгоритмів необхідно використовувати квантовий паралелізм через суперпозицію станів.

У квантовому комп'ютері кубіти можуть існувати в суперпозиції, що дозволяє їм одночасно представляти та обробляти кілька «можливостей». Цей паралелізм формує основу для експоненціального прискорення, як це видно в таких алгоритмах, як алгоритми Гровера та Шора.

Розробка ефективного квантового алгоритму включає такі кроки:

1. Аналіз задачі та оцінка придатності для створення квантового алгоритму.

Визначення мети алгоритму є першочерговим. Першим завданням є аналіз проблеми, щоб зрозуміти, чи вона виграє від застосування принципів квантової механіки. Задачі, що включають комбінаторний пошук або факторизацію, що характеризуються експоненційним масштабуванням у класичних умовах, часто підходять для пошуку квантових рішень.

2. Вибір квантового представлення.

Визначення подання (представлення) проблеми в рамках квантової теорії. Представлення визначає початкову підготовку стану, часто включаючи суперпозицію для паралельного дослідження кількох рішень. Відображення класичних даних у квантові стани може включати схеми кодування, такі як кодування базисного стану або амплітудне кодування.

3. Динаміка алгоритмів та вибір квантових операцій.

Розробка операцій квантового алгоритму, включаючи вибір необхідних квантових вентилів, що визначають переходи між станами. Вибір відповідних

вентилів та послідовностей для керування станами кубітів формує основну механіку. Квантові вентиля можуть бути одно- або багатокубітними, з такими операціями, як Адамар, CNOT та фазові вентиля, що формують основу квантової обробки.

4. Використання квантових підпрограм.

За потреби використовуються класичні та квантові гібридні підходи. Квантові алгоритми часто інтегрують класичні підпрограми для завдань попередньої або пост-обробки, інтерпретації даних та керування потоком. Гібридні алгоритми, особливо варіаційні квантові алгоритми (англ. Variational Quantum Algorithm, VQA), поєднують квантову обробку та класичну оптимізацію.

5. Внесення змін та налаштування за допомогою квантового моделювання.

Перевірка алгоритму за допомогою квантового моделювання. Квантову схему реалізують за допомогою вибраного фреймворку. Симулятори дозволяють удосконалити та покращити алгоритм, забезпечуючи спостереження за поведінкою кубітів та ймовірностями вимірювань без впливу реальних квантових апаратних обмежень.

6. Оцінка складності виконання.

Аналіз асимптотичної складності квантового алгоритму у порівнянні з його класичним аналогом, зосереджуючись на квантових вентилях та впливі операцій на ефективність використання часу та простору. Механізми квантового прискорення часто покращують конкретні обчислювальні завдання, що підтверджується поліноміальним або експоненційним зменшенням складності виконання.

Процес проектування квантових алгоритмів можна проілюструвати на канонічних прикладах, таких як алгоритми Гровера та Шора. Алгоритм Гровера

використовує квантову суперпозицію та посилення амплітуди для задач неструктурованого пошуку, забезпечуючи квадратичне прискорення.

Робота квантового алгоритму (або програми за допомогою симулятора) зазвичай складається з таких кроків:

1. *Ініціалізація*: підготовка кубітів у відомому квантовому стані, часто починаючи з усіх кубітів у стані $|0\rangle$.

2. *Квантові операції з вентилями*: застосування серії квантових вентилів (таких як вентиля Адамара, CNOT та вентиля фазового зсуву) для впливу на стани кубітів (англ. Qubit manipulation).

3. *Використання інтерференції та запутаності*: використання запутаності для кореляції кубітів та інтерференції для посилення правильних результатів та одночасного скасування неправильних.

4. *Вимірювання*: спостереження за квантовим станом, його згортання в класичний бітовий вихід (0 та 1).

Проектування квантових алгоритмів вимагає ретельного балансу між квантовими операціями та розумінням роботи класичних алгоритмів при створенні рішень, які є одночасно життєздатними та переважними порівняно з існуючими методологіями.

Зростаючий набір інструментів для квантового програмування та дедалі потужніше квантове обладнання сприяють розробленню квантових алгоритмів, забезпечуючи можливості для розв'язання задач, які раніше вважалися нездоланими за допомогою класичних підходів.

6.3 Визначення поняття оракула

Квантові алгоритми зазвичай призначені для оцінювання функцій. Запит складності визначає скільки разів потрібно перевірити функцію, щоб визначити її складність. Іноді таку функцію називають *чорною скринькою* або *оракулом*. Питання не ставиться у формі «оцінити результат роботи функції», питання формується у вигляді запиту до «чорної скриньки» або оракула. При цьому не

потрібно вирішувати, як написати алгоритм, що імітує функцію, тому не потрібно підраховувати кількість кроків (проектувати класичний алгоритм), які виконує функція для обробки вхідних даних. При квантовому підході потрібно підраховувати кількість запитів, що набагато спрощує виконання завдання.

Оракул буде використаний при розгляді алгоритму Дойча. Оракул використовується для опису квантової «чорної скриньки» подібно до того, як будь-яку функцію можна описати, як класичну «чорну скриньку». Внутрішньо функції мають певний обчислювальний потік, але про внутрішні механізми програми/функції нічого не відомо. Треба оцінити роботу функції для різних вхідних значень. Ті ж концепції можна застосувати до оракула.

Внутрішньо оракул складається з одного або кількох квантових вентилів, але не відомо, яких саме. Надсилення запитів до оракула (наприклад, надсилаючи вхідні дані та вимірюючи вихідні) дає можливість дізнатися більше про властивості оракула. Оскільки оракул складається з квантових вентилів, сам оракул також повинен мати можливість бути оберненим (зворотним). Отже, оракул можна вважати квантовим еквівалентом функцій «чорної скриньки», як показано на рис. 6.2. Як оракул, так і функція, виконують деякі обчислення, але при цьому невідомі внутрішні деталі цих обчислень.

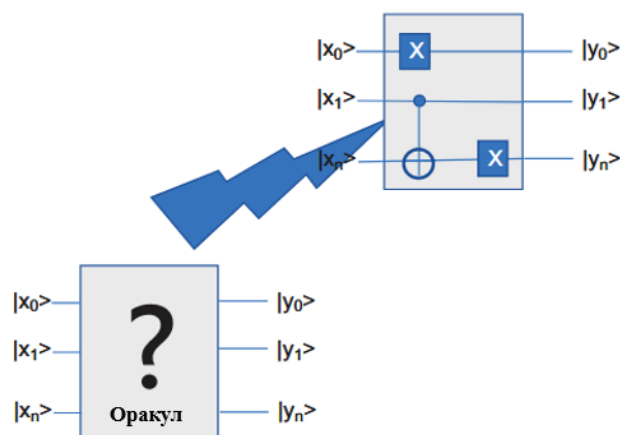


Рис. 6.2 – Оракул та його можлива реалізація.

6.4 Підходи до оцінки функцій на прикладі алгоритмів Шора та Дойча

Розглянемо приклад. Потрібно отримати відповіді на питання. Чи ділиться число на 3? Який результат ділення числа на 3? Чи є знаки після крапки?

У багатьох випадках потрібно отримати певну властивість операції, числа чи функції. Не потрібно знаходити оцінку функції (результат її виконання), але потрібно якимось чином отримати інформацію про функцію. Оцінка функції часто є найпростішим способом зробити це, але може бути ефективніше опосередковано отримати властивості функції та зробити висновки. Це представляє особливий інтерес у квантових обчисленнях. Квантовий комп'ютер з n кубітами, якому потрібно проаналізувати роботу певної функції, може виконувати лише одну оцінку функції за раз.

Застосування квантової схеми до заданого конкретного набору вхідних кубітів призведе до змінення стану цих кубітів. Їх вимірювання дає певний результат, і якщо необхідно отримати новий результат, необхідно запустити квантову схему ще раз.

Застосування вентилів Адамара до вхідних кубітів переводить їх у стан *суперпозиції*, що дозволяє одночасно оцінювати кубіти в станах 0 та 1, але в процесі роботи не можливо створити/додати нові кубіти, які міститимуть інформацію про різні випадки.

Внутрішні обчислення схеми можуть містити еквівалент багатьох класичних обчислень, але неможливо отримати їх одночасно за допомогою класичних обчислень. Обмеження – результат з n кубітів. Але цього часто достатньо для розв'язання задач. У випадку, якщо задане число є множителем 3 чи ні, одного кубіта достатньо для зберігання відповіді. Не потрібно обчислювати функцію ділення, потрібно оцінити тільки результат.

Проаналізуємо властивість функції f , що діє на n бітів, не цікавлячись окремими обчисленнями функції. Отримання властивості класичним способом вимагає $2^{n-1}+1$ обчислень функції. Використовуючи квантовий алгоритм властивість можна отримати за допомогою одного обчислення. Для «простих» функцій немає прямого варіанту їх практичного використання. Але це демонструє суттєву перевагу квантових обчислень і пояснює, чому квантові обчислення часто асоціюються з експоненційною складністю.

Легко побачити, що чим більше вхідних бітів має функція, тим важче розв'язати задачу класичним способом. Число n дійсно знаходиться в показнику ступеня, і експоненціальні функції швидко призводять до величезних значень. Якщо квантовий алгоритм може розв'язати ту саму задачу лише за одне обчислення (або, загалом, за меншу кількість), це величезна перевага для квантового комп'ютера.

Схему міркувань можна представити так. Спочатку визначаються властивості функцій та класичні способи їх отримання. Далі перетворюємо функції на квантові блоки, які називаються оракулами, при цьому повинні виконуватись певні вимоги. Як тільки створено квантовий оракул, який представляє класичну функцію, цей оракул можна буде використовувати у квантовій схемі. Оцінка цього квантового ланцюжка один раз призводить до отримання властивості функції, яку треба знайти.

У більшості типових випадків, коли задіяні функції, може бути необхідним знайти результат функції. Наприклад, розглянемо функцію $y(x) = x^2$. Якщо потрібно знайти значення цієї функції, наприклад, для $x = 4$ та $x = 7$, для цього необхідно обчислити цю функцію. Однак, у деяких випадках обчислення функцій не є важливими, на відміну від отримання (або обчислення) **характеристик** (властивостей) функцій. У цих випадках квантові алгоритми можуть бути корисними.

Приклад, який можна зустріти при розгляді алгоритму Шора, – це *періодичність* функції. Не потрібно окремо обчислювати функцію, необхідно знайти періодичність. Періодична функція – це функція, де однаковий шаблон значень повертається з фіксованою періодичністю (табл. 6.1).

Табл. 6.1 – Таблично-задана періодична функція $y=f(x)$.

x	0	1	2	3	4	5	6
y	7	9	5	7	9	5	7

З цієї таблиці можна побачити, що функція має періодичність 3. Для кожного значення x результат функції такий самий, як і функція, застосована до $x+3$. Це приклад, коли властивість функції може бути важливішою/потрібнішою, ніж обчислення самої функції. Зазвичай функції позначаються як $f(x)$, де f – це функція, яка визначається вхідною змінною x . Обчислення функції для певного вхідного значення також називається результатом y , $y = f(x)$.

Розглянемо набір функцій, на вхід яких подається лише один вхідний біт, з подальшим розширенням їх до функції з n вхідними бітами. У всіх випадках результат функції дорівнює або 0, або 1. Ці функції мають особливу властивість: вони є або збалансованими функціями, або функціями-константами (константними). Функція називається константною, коли результат не залежить від вхідного значення. У нашому випадку це означає, що результат дорівнює або 0 для всіх вхідних значень, або 1 для всіх вхідних значень. Функція називається збалансованою, якщо результат дорівнює 0 у 50% випадків та 1 в інших випадках.

Алгоритм Дойча має справу з функцією f , яка приймає один біт (булеве значення) як вхідне значення і також генерує один біт. Функція працює лише з 0 та 1, а її результатом є або 0, або 1. Поєднання двох вхідних параметрів та двох вихідних параметрів призводить до чотирьох можливих випадків для цієї функції, які позначимо як f_1, f_2, f_3 та f_4 :

$$f_1 : f(0) = 0, f(1) = 0;$$

$$f_2 : f(0) = 0, f(1) = 1;$$

$$f_3 : f(0) = 1, f(1) = 0;$$

$$f_4 : f(0) = 1, f(1) = 1.$$

З цих визначень видно, що f_1 та f_4 є функціями-константами, f_2 та f_3 – збалансованими функціями. Таблиця істинності показана в табл. 6.2.

Табл. 6.2 – Таблиця істинності заданої функції.

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	1	1
1	0	1	0	1

Цю таблицю істинності можна представити у формальному вигляді:

$$\begin{aligned} f_1(x) &= False; \\ f_2(x) &= x; \\ f_3(x) &= \bar{x}; \\ f_4(x) &= \bar{x} \vee x, True. \end{aligned}$$

У багатьох класичних алгоритмах важливо знати значення на виході функції для певних її аргументів. З іншого боку, у багатьох квантових алгоритмах корисно знати *властивості функції*, що розглядається. Це частина «іншого мислення», яке потрібно при розгляді та проєктуванні квантових алгоритмів. Завдяки суперпозиції квантовий комп'ютер може оцінювати багато можливостей одночасно. Але оскільки отримання результату вимагає *вимірювання*, суперпозиція зникає, і результатом буде одне значення. Отже, додана цінність полягає в *оцінці функції*, а не в *обчисленні її результату* при її виконанні.

В алгоритмі Дойча функція надається, але не відомо, яка саме це функція. Відомо, що це f_1 , f_2 , f_3 та f_4 . Виникає постановка задачі, чи є ця функція константною, чи збалансованою? Завдання полягає не в тому, щоб визначити результат роботи функції, або чи є надана функція саме f_1 , f_2 , f_3 , f_4 . Завдання – визначити **властивість** функції. Скільки обчислень функції потрібно зробити, щоб відповісти на це питання зі 100% впевненістю?

Якщо зробити лише одне обчислення, обчислимо лише $f(0)$ або $f(1)$, достатньо інформації для висновку не буде отримано. Припустимо, що вимірюється $f(0)$, і результат дорівнює 1. З попередньої таблиці істинності видно, що в цьому випадку функція дорівнює або f_3 (яка є збалансованою), або f_4 (яка є константою). *Отже, для отримання рішення недостатньо інформації.* Якщо результат вимірювання $f(0)$ дорівнює 0, таблиця показує, що функція дорівнює або f_1 (яка є константою), або f_4 (яка є збалансованою). Це доводить, що одного вимірювання $f(0)$ недостатньо, щоб зробити висновок про те, чи є функція константою чи збалансованою. Вона може бути будь-якою. Виявляється, потрібні **два класичні обчислення функції**, перш ніж отримаємо відповідь, чи є надана функція константою, чи збалансованою. Нескладно для перевірки

створити класичну програму, очевидно, при збільшенні кількості входів класична програма значно ускладниться. Але в кожному циклі будуть виконані два обчислення функції. Одного обчислення буде недостатньо, щоб зробити висновок про те, чи збалансована функція.

6.5 Від функції до оракула

Алгоритм Дойча доводить, що одного квантового обчислення достатньо, щоб з'ясувати, чи є надана функція константою чи збалансованою. Перед перевіркою функції, необхідно перетворити класичну функцію на квантову операцію. Не можливо просто застосувати функцію до кубіта. У розділі 4.6 було показано, що всі квантові вентиля повинні бути зворотними. Функція, яка після застосування унеможливорює отримання початкового стану на вході, не може бути використана в квантовій схемі. Тому функцію спочатку потрібно перетворити на зворотній оракул.

Створимо оракул, який потім можна використовувати в алгоритмі Дойча. Подібно до того, як класична функція передається класичному алгоритму, оракул передається квантовому алгоритму. Кожну класичну функцію можна представити певним оракулом. Оскільки задано чотири можливі функції, також є чотири можливі оракули. Загальний спосіб побудови оракула **B** на основі функції f показано на рис. 6.3. У цьому підході використовується заданий вхідний кубіт $|x\rangle$ та додатковий кубіт $|a\rangle$.

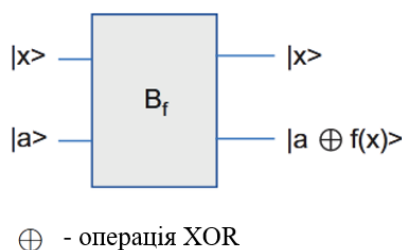


Рис. 6.3 – Побудова оракула **B** на основі функції f .

Оракул залишає кубіт $|x\rangle$ у його початковому стані, а кубіт $|a\rangle$ замінюється операцією XOR між a та $f(x)$. Розглянемо цей оракул детальніше та його вигляд для чотирьох функцій, які вже визначені раніше у табл. 6.2. Перша функція, f_1 , є

константною функцією, яка повертає 0 незалежно від вхідних даних. Отже, оскільки $f(x) = 0$ для будь-якого значення x , вихідний стан другого кубіта можна спростити у такий спосіб:

$$a \oplus f(x) = a \oplus 0 = a.$$

Отриманий оракул представлений на рис. 6.4. У цьому випадку оракул є просто одиничною матрицею. Як $|x\rangle$, так і $|a\rangle$ не змінюються між входом і виходом. Приховану логіку всередині оракула можна, таким чином, представити схемою з використанням одиничної матриці I .

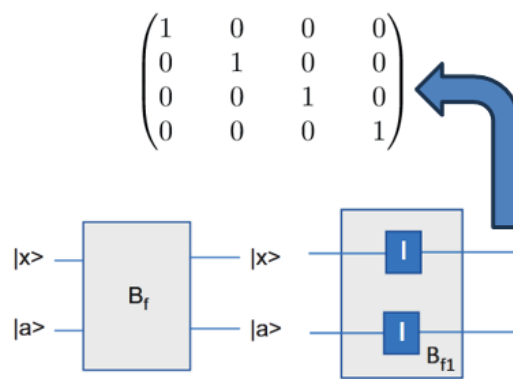


Рис. 6.4 – Оракул В для представлення функції f_1 .

Розглянемо другу класичну функцію, f_2 . Ця функція визначається так: $f(0)=0$, $f(1)=1$. Це також можна записати як $f(x)=x$. Використовуючи це в загальному описі оракула, схема спрощується, як показано на рис. 6.5. Саме такий стан був би отриманий, якби оракул мав вентиль CNOT. Отже, можлива схема для оракула, що відповідає функції f_2 , показана на рис. 6.5.

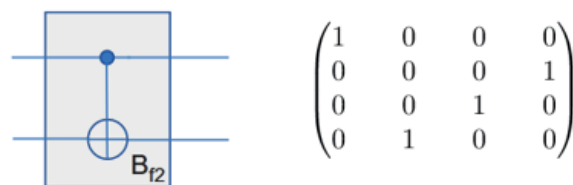


Рис. 6.5 – Оракул В для представлення функції f_2 .

Четверта функція, f_4 , є константною функцією, яка завжди повертає значення 1, незалежно від вхідних даних. Як наслідок, вихід другого кубіта після застосування оракула можна записати так:

$$a \oplus f(x) = a \oplus 1 = \bar{a}.$$

Горизонтальна смуга над змінною вказує на те, що це інвертована змінна, яка в цьому випадку відповідає вентилю Паулі-Х, що застосовується до $|a\rangle$. Отже, цей оракул схематично представлений на рис. 6.6.

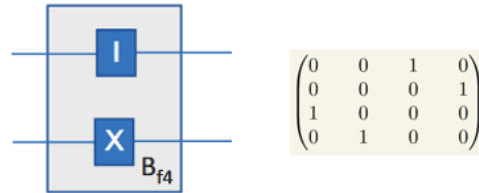


Рис. 6.6 – Оракул В для представлення функції f_4 .

6.6 Алгоритм Дойча

Класичний підхід.

Обчислимо функцію, передавши їй 0 або 1. Припустимо, що її аргумент – 0, тоді можливі два результати – отримаємо 0 або 1. Отримавши 0, можемо стверджувати лише, що $f(0)=0$. Це може бути функція f_0 або f_1 (табл. 6.2). Оскільки одна з них є константною, а інша збалансована, треба обчислити функцію ще раз, щоб зробити вибір між ними. Отже, міркуючи з класичної точки зору, щоб відповісти на питання, потрібно передати функції обидва можливі значення, 0 і 1. Тобто у класичному підході до обчислень необхідно обчислити функцію двічі. Як буде показано далі, квантовий підхід надає перевагу.

Квантовий підхід.

Чотири виходи завжди дають елементи стандартного базису, тобто матриця, що представляє вентиль (оракул), ортогональна, і модель є вентилем. Навіть при тому, що вводимо два біти інформації та отримуємо два біти на виході, інформація, яку дають ці вентиля для класичних бітів, $|0\rangle$ і $|1\rangle$, така сама, як для функцій, що отримують 0 і 1. Опис роботи алгоритму для невідомого оракула у загальному вигляді представлений на рис. 6.7.

Якщо на вхід подати $|0\rangle \otimes |0\rangle$ результатом буде $|0\rangle \otimes |f_i(0)\rangle$
 Якщо на вхід подати $|0\rangle \otimes |1\rangle$ результатом буде $|0\rangle \otimes |f_i(0) \oplus 1\rangle$
 Якщо на вхід подати $|1\rangle \otimes |0\rangle$ результатом буде $|1\rangle \otimes |f_i(1)\rangle$
 Якщо на вхід подати $|1\rangle \otimes |1\rangle$ результатом буде $|1\rangle \otimes |f_i(1) \oplus 1\rangle$

Рис. 6.7 – Опис роботи алгоритму для невідомого оракула.

Верхній кубіт на рис. 6.8 – це вхід алгоритму, тому його вихід не дає нової інформації. Вибираючи $|0\rangle$ або $|1\rangle$ для другого входу отримуємо другий вихід, що дає той же результат, що повернула функція для верхнього вхідного кету, або протилежний. Знаючи один, можемо визначити другий. Питання в квантових обчисленнях, що відповідає питанню в класичних обчисленнях, формулюється так: *вибираючи випадковий вентиль з цих чотирьох, скільки разів потрібно використовувати його, щоб визначити, чи є його базова функція константною чи збалансованою?*

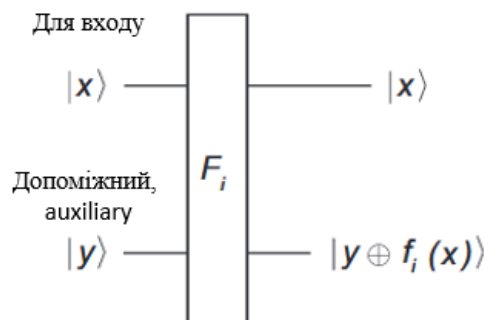


Рис. 6.8 – Загальна схема алгоритму Дойча.

Як вже було зазначено, дія операторів має бути зворотною. В алгоритмах використовуються зворотні операції, наприклад, застосування Адамара, а потім застосування цього вентиля ще раз (рис. 6.9).

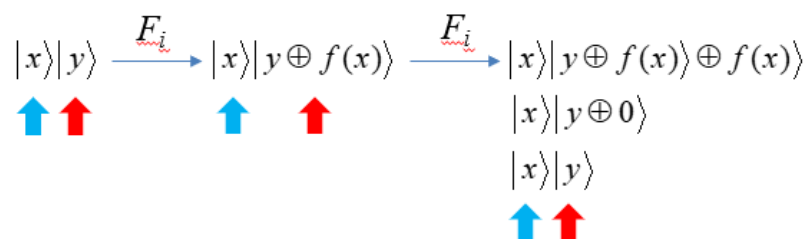


Рис. 6.9 – Ілюстрація повторної дії того ж самого оператора F до двох кубітів. Кольором показані два різні кубіти.

Якщо застосувати операцію «виключне АБО» до чогось самого з собою, наприклад, $f(x) \oplus f(x)$, то буде отримано нуль. І якщо застосувати до y «виключне АБО» з нулем, то буде отримано y . Отже, схема повернеться до стану $|x\rangle|y\rangle$. Однак, коли справа доходить до написання коду або створення схеми, функція може бути представлена будь-яким вентилям. Наприклад, F_i може бути представлена за допомогою CNOT, або контрольованого Z , або X . У формальному вигляді можна довести коректність на прикладі будь-якої функції.

Все, що має значення, це те, що необхідний тільки один запуск схеми, щоб визначити результат. Отже, квантовий алгоритм працює з «чорним ящиком» або оракулом. Це означає, що існує якась функція з невідомим обсягом та внутрішньою реалізацією, на її вхід подаються дані, і вона повертає результат на виході. Можливо задавати деякі питання до властивості цього оракула, наприклад, чи існує певна властивість, чи ні? Не відомо, що відбувається всередині «чорного ящика», але це відбувається.

Отже, почнемо з «неправильного» підходу. Припустимо, що потрібно розмістити вентиль Адамара лише для кубіта, який потрібно буде вимірювати. З'ясуємо, як буде працювати відповідна схема, представлена на рис. 6.10. Перевіримо, чи можна отримати за один запуск схеми або постійну функцію, або збалансовану функцію? Цей підхід не є правильним, але це допоможе зрозуміти, чому це неправильний підхід, і який підхід буде правильним. На основі рис. 6.8 представимо перший крок, дію оракула F_i .



Рис. 6.10 – Дія оракула F_i на стан системи після застосування вентиля Адамара.

Дія оракула є тензорним добутком двох матричних представлень кубітів $|x\rangle$ та $|y\rangle$, відповідно $|+\rangle$ та $|0\rangle$. Представимо дію оракула у формальному вигляді із усіма формальними перетвореннями, як на рис. 6.11.

$$\begin{aligned}
 F_i(|+\rangle \otimes |0\rangle) &= F_i\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle\right) = \frac{1}{\sqrt{2}} F_i|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} F_i|1\rangle \otimes |0\rangle = \\
 &\text{Після застосування } F_i \\
 &= \frac{1}{\sqrt{2}} |0\rangle f(0) \oplus |0\rangle + \frac{1}{\sqrt{2}} |1\rangle f(1) \oplus |0\rangle = \frac{1}{\sqrt{2}} |0\rangle f(0) + \frac{1}{\sqrt{2}} |1\rangle f(1)
 \end{aligned}$$

з ймовірністю 50% $|0\rangle f(0)$ $|1\rangle f(1)$ з ймовірністю 50%

Рис. 6.11 – Формальне представлення дії оракула F_i на стан системи.

Треба перевірити властивість функції лише за один цикл роботи (запуск). Для цього застосуємо інші вентиля до другого допоміжного кубіта. Додаємо вимірювання в кінці схеми. На другому кубіті застосовуємо вентиль X (NOT), а потім вентиль Адамара. У кінці застосуємо вентиль Адамара, а потім виміряємо x , як показано на рис. 6.12. Схема покаже, чи F_i постійна функція (константа), чи збалансована функція.

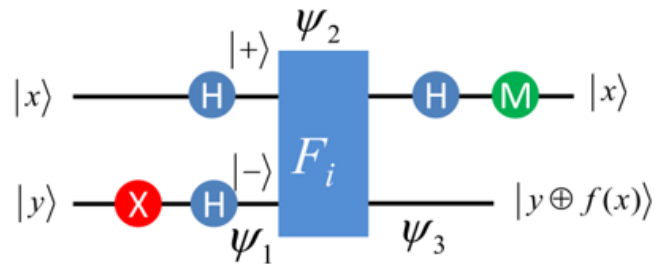


Рис. 6.12 – Модифікована схема для реалізації перевірки тільки за один крок.
Змінні ψ_1, ψ_2, ψ_3 використані для представлення кроків.

Отже, в цьому прикладі кубіт $|y\rangle$ переведений у стан суперпозиції, але з іншою фазою $|-\rangle$. Якщо застосувати суперпозицію (або вентиль Адамара) до стану $|1\rangle$, результатом буде стан $|-\rangle$. Змінні ψ_1, ψ_2, ψ_3 використані для представлення кроків. Формальне представлення цих кроків показано на рис. 6.13.

$$\begin{aligned}
 & |+\rangle \otimes |-\rangle \\
 \psi_1 : & \frac{1}{2}(|0\rangle + |1\rangle) \otimes \frac{1}{2}(|0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \\
 & = \frac{1}{2}(|0\rangle|0 \oplus f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) = \\
 \psi_2 : & = \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) \\
 & \quad \text{Результат невідомий} \quad \text{Результат невідомий}
 \end{aligned}$$

F_i
↓

Рис. 6.13 – Формальне покрокове представлення дії оракула F_i на стан модифікованої системи.

Припустимо, що заплановано зробити дві способи. Покажемо це за допомогою формальних виразів. Припустимо, якщо $f(0) = f(1)$, функція – константа. Отже, якщо вона дорівнює $f(1)$, то можливо спростити вираз і записати $f(0)$ замість $f(1)$, тобто виразити $f(1)$ через $f(0)$, як на рис. 6.14.

$$\begin{aligned}
 \psi_2 &= \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) \\
 \text{якщо } f(0) = f(1) \quad \psi_2 &= \frac{1}{2}(|0\rangle + |1\rangle)|f(0)\rangle + (-|0\rangle - |1\rangle)|1 \oplus f(0)\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|f(0)\rangle - |1 \oplus f(0)\rangle) = \\
 &= \frac{1}{\sqrt{2}}|+\rangle(|f(0)\rangle - |1 \oplus f(0)\rangle) \\
 & \quad \text{Значення другого кубіту (y)} \quad \text{Стан після застосування Адамара: } \psi_3 \quad H|+\rangle = |0\rangle
 \end{aligned}$$

Винесли «-» за дужки

Рис. 6.14 – Формальне представлення стану схеми Дойча на кроках 2 та 3 для $f(0) = f(1)$.

Якщо застосувати суперпозицію або вентиль Адамара до першого кубіту та виконати вимірювання? Другий кубіт $|y\rangle$ не будемо вимірювати, і значення другої частини в останній формулі у дужках (рис. 6.14) не має значення. Отже, якщо $f(1)=f(0)$, то буде отримано $|0\rangle$, тому що застосовано вентиль Адамара до стану $|+\rangle$, і буде виміряно 0. Це можна перевірити за допомогою конструктора схем Quirk [58].

Тепер припустимо (рис. 6.15), що $f(0) \neq f(1)$, функція – збалансована, тоді можна записати $f(0)=f(1)\text{XOR}1$. При цьому використовуємо два кубіти, один як допоміжний, але вимірювати необхідно лише перший (верхній) кубіт $|x\rangle$, тому стан другого кубіта в кінці роботи схеми фактично не потрібен.

$$\begin{aligned}
\psi_2 &= \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle) \\
\text{якщо } f(0) \neq f(1) \quad \psi_2 &= \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|f(1)\rangle + |1\rangle|f(1)\rangle - |1\rangle|f(0)\rangle) = \\
&= \frac{1}{2}(|0\rangle \otimes (|f(0)\rangle - |f(1)\rangle) - |1\rangle \otimes (|f(0)\rangle - |f(1)\rangle)) = \frac{1}{2}(|0\rangle - |1\rangle) \otimes (|f(0)\rangle - |f(1)\rangle) = \\
\psi_3 : &= \frac{1}{\sqrt{2}}|-\rangle \otimes \underbrace{(|f(0)\rangle - |f(1)\rangle)}_{\text{Стан другого кубіта}} \quad \text{Стан після застосування Адамара: } H|-\rangle = |1\rangle
\end{aligned}$$

Рис. 6.15 – Формальне представлення стану схеми Дойча на кроках 2 та 3 для $f(0) \neq f(1)$.

Отже, якщо застосувати вентиль Адамара до стану $|-\rangle$, завжди буде отримано $|1\rangle$. Це означає, що якщо функція F_i збалансована, то буде отримано $|1\rangle$, а якщо функція F_i константа, то буде отримано $|0\rangle$. Отже, залежно від вимірювання, за допомогою цієї схеми є можливість фактично знати поведінку функції, чи це збалансована функція, чи константна функція. Це буде зроблено за одну спробу, буде виміряний лише один кубіт. На рис. 6.15 не відомий результат обчислення $|f(0)\rangle - |f(1)\rangle$, позначений червоним надписом «Стан другого кубіта», але це не має значення, тому що немає необхідності його вимірювати, оскільки це допоміжний кубіт. Тому не треба застосовувати більше вентилю Адамара і для другого (нижнього) кубіта.

Схема на рис. 6.16 (зверху) починається з вентилю H Адамара, та з вентиля X до другого кубіту. Отже, схема в стані $|+\rangle$ і стані $|-\rangle$ на другому кубіті. Також встановимо вентиль I ідентичності (англ. Identity), який нічого не робить (це одинична матриця), тому що потрібен буфер у цьому місці, щоб бар'єри надали схемі кращий вигляд.

Додаємо вентиль CNOT для моделювання функції F_i . Використовуємо його для того, щоб побачити результат, що це збалансована функція. **Відомо, що CNOT є збалансованою функцією.** Також можна використати будь-який інший вентиль.

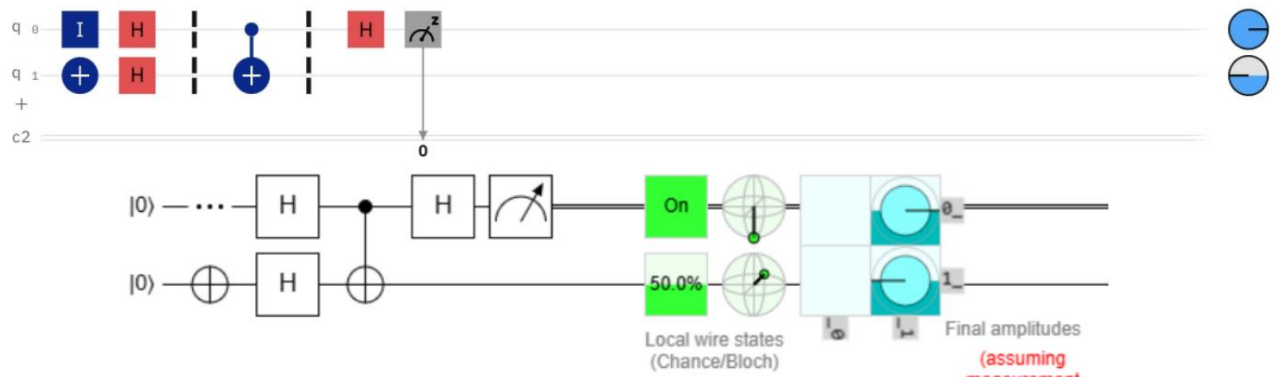


Рис. 6.16 – Схема алгоритму Дойча в Qiskit (верхня частина) та за допомогою конструктора схем Quirk [58] (нижня частина).

Далі застосуємо вентиль Адамара для першого кубіта і встановимо вентиль вимірювання, щоб виміряти перший кубіт, з другим кубітом нічого не виконуємо.

Для моделювання константного оракула можемо застосувати або вентиль X , або I до останнього кубіта, який використовується для досягнення результату 0 або 1 оракула. Якщо використовуємо X , оракул завжди повертатиме 1, а якщо використати I , він завжди повертатиме 0.

6.7 Алгоритм Дойча-Йожи

Алгоритм Дойча показує, що конкретну задачу, яка вимагає двох обчислень у класичному підході, можна розв'язати за один крок/обчислення за допомогою квантового алгоритму. Цей результат є дуже перспективним. Алгоритм Дойча можна модифікувати і отримати алгоритм Дойча-Йожи, в якому вхідна функція оперує не з одним булевим значенням, а з n булевими значеннями. У цьому випадку функція (або оракул) оброблює n вхідних бітів і представлена як $f(x_0, x_1, \dots, x_{n-1})$. Ця функція теж є або константою, що означає, що вона завжди повертає 0, або завжди повертає 1, або збалансованою (у половині випадків вона повертає 0, а в іншій половині випадків – 1). Алгоритм Дойча є окремим випадком алгоритму Дойча-Йожи, де $n = 1$, і існує лише два можливих вхідних сценарії. Якщо $n=2$, існує чотири можливі сценарії. Якщо $n=3$, існує вісім сценаріїв (рис. 6.17).

$(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)$

Рис. 6.17 – Результат роботи алгоритму Дойча-Йожи для $n=3$.

Алгоритм Дойча-Йожи формально може бути представлений за допомогою змінних та операцій над ними (рис. 6.18). Показано $n-1$ вхідних кубітів $|x_{n-1}\rangle$ та допоміжний кубіт $|y\rangle$. Похилі риски з індексом n зверху показують, що використовується n паралельних входів/виходів.

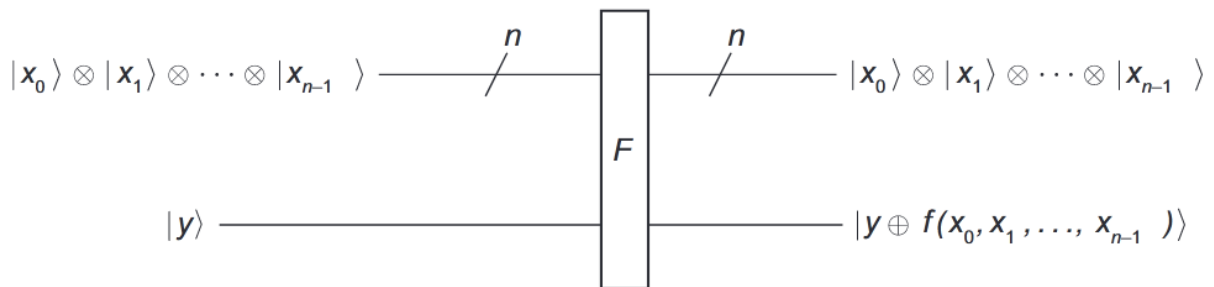


Рис. 6.18 – Формальне представлення алгоритму Дойча-Йожи [59].

Схема реалізації алгоритму Дойча-Йожи з конкретними значеннями кубітів та вентилями показана на рис. 6.19.

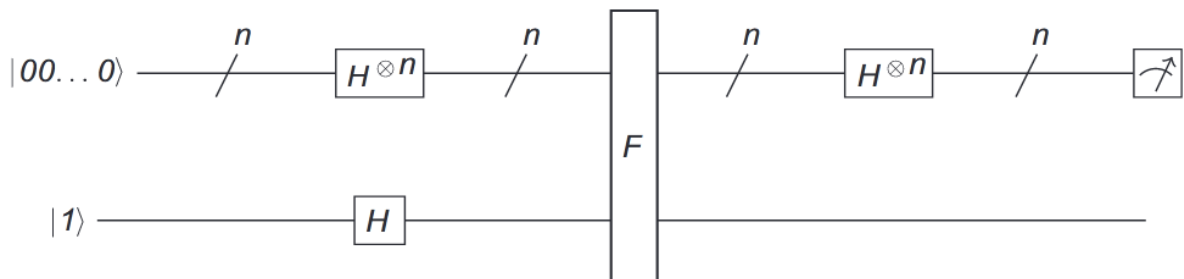


Рис. 6.19 – Схема алгоритму Дойча-Йожи [59].

Якщо $n=2$, існує чотири можливі сценарії. Загалом, існує 2^n сценаріїв, коли є n вхідних бітів. Скільки класичних обчислень потрібно виконати, щоб довести на 100%, що функція є або константою, або збалансованою? Припустимо, що необхідно обчислити половину можливих сценаріїв ($2^n/2$, що дорівнює 2^{n-1}). Якщо принаймні один з результатів дорівнює 0, і принаймні один з результатів дорівнює 1, отримуємо висновок, що функція не є константою, тому вона має бути збалансованою.

Але який висновок можна зробити, якщо всі обчислення призводять до значення 1? У такому випадку, з високою ймовірністю можна стверджувати, що

функція є константою. Але все одно потрібне ще одне обчислення, оскільки існує ймовірність того, що всі інші обчислення призведуть до 0.

Щоб бути на 100% впевненим, що функція є константою, функція з n бітами на вході потребує $2^{n-1} + 1$ обчислень, перш ніж зробити висновок, що функція збалансована або константа.

Однак, використовуючи квантову схему, подібну до тієї, що в алгоритмі Дойча, потрібне лише одне обчислення. Висновок – квантові алгоритми підходять для задач експоненціальної складності, якщо використовується класичний підхід.

Розглянемо алгоритм Дойча-Йожи для $n=2$.

Крок 1. Усі верхні n входів дорівнюють $|0\rangle$ (для $n=2$ маємо $|00\rangle$). Застосуємо до цієї групи кубітів вентиль Адамара, як показано на рис. 6.20.

$$H^{\otimes 2}(|00\rangle) = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

Рис. 6.20 – Приклад застосування двох вентилів Адамара до групи з двох вхідних кубітів.

У результаті отримуємо суперпозицію всіх можливих станів. Усі базисні кети мають однакові амплітуди ймовірності ($1/2$). Ці обчислення поширюються на будь-які значення n . Після проходження n кубітів через $H^{\otimes n}$ вентилів вони опиняються в суперпозиції всіх можливих станів, кожен з яких має ту саму

амплітуду ймовірності $\left(\frac{1}{\sqrt{2}}\right)^n$. Нижній вхід має значення $\left(\frac{1}{\sqrt{2}}\right)|0\rangle - \left(\frac{1}{\sqrt{2}}\right)|1\rangle$.

Після проходження цього кубіту через вентиль Адамара він отримує стан $|0\rangle$. На цьому кроці три вхідні кубіти отримують такий стан:

$$\begin{aligned} \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) &= \frac{1}{2\sqrt{2}}|00\rangle \otimes (|0\rangle - |1\rangle) + \\ &+ \frac{1}{2\sqrt{2}}|01\rangle \otimes (|0\rangle - |1\rangle) + \frac{1}{2\sqrt{2}}|10\rangle \otimes (|0\rangle - |1\rangle) + \frac{1}{2\sqrt{2}}|11\rangle \otimes (|0\rangle - |1\rangle). \end{aligned}$$

Крок 2. Передача кубітів через оракул F .

Після проходження через вентиль F (оракул) кубіти перейдуть у такий стан:

$$\begin{aligned} & \frac{1}{2\sqrt{2}}|00\rangle \otimes (|f(0,0)\rangle - |f(0,0) \oplus 1\rangle) + \\ & + \frac{1}{2\sqrt{2}}|01\rangle \otimes (|f(0,1)\rangle - |f(0,1) \oplus 1\rangle) + \\ & + \frac{1}{2\sqrt{2}}|10\rangle \otimes (|f(1,0)\rangle - |f(1,0) \oplus 1\rangle) + \\ & + \frac{1}{2\sqrt{2}}|11\rangle \otimes (|f(1,1)\rangle - |f(1,1) \oplus 1\rangle). \end{aligned}$$

Далі, якщо a може мати лише значення 0 або 1, тоді виконується рівність $|a\rangle - |a \oplus 1\rangle = (-1^a)(|0\rangle - |1\rangle)$:

$$\begin{aligned} & (-1)^{f(0,0)} \frac{1}{2}|00\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) + \\ & + (-1)^{f(0,1)} \frac{1}{2}|01\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) + \\ & + (-1)^{f(1,0)} \frac{1}{2}|10\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) + \\ & + (-1)^{f(1,1)} \frac{1}{2}|11\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned}$$

Тепер розглянемо два верхні кубіти. Вони перебувають у стані:

$$\frac{1}{2} \left((-1)^{f(0,0)}|00\rangle + (-1)^{f(0,1)}|01\rangle + (-1)^{f(1,0)}|10\rangle + (-1)^{f(1,1)}|11\rangle \right).$$

На цьому етапі отримуємо стан, що є суперпозицією всіх базових кетів.

Крок 3. Застосування вентилів Адамара до верхніх кубітів.

Стандартний спосіб полягає у перетворенні стану на вектор-стовпчик з подальшим множенням на відповідний кронекерівський добуток матриці Адамара:

$$\frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} (-1)^{f(0,0)} \\ (-1)^{f(0,1)} \\ (-1)^{f(1,0)} \\ (-1)^{f(1,1)} \end{bmatrix}.$$

Однак не потрібно обчислювати всі елементи вектора-стовпчика результату. Достатньо обчислити тільки верхній елемент. Його можна отримати множенням «бра», що відповідає верхньому рядку матриці на «кет», який заданий вектором-стовпчиком:

$$\frac{1}{4} \left((-1)^{f(0,0)} + (-1)^{f(0,1)} + (-1)^{f(1,0)} + (-1)^{f(1,1)} \right).$$

Це і є амплітуда ймовірності кету $|00\rangle$. Обчислимо цю амплітуду для можливих функцій. Якщо f константна і завжди повертає 0, амплітуда ймовірності дорівнює 1. Якщо f константна і завжди повертає 1, амплітуда ймовірності дорівнює -1. Для збалансованих функцій амплітуда ймовірності дорівнює 0.

Крок 4. Вимірювання верхніх кубітів.

Вимірявши верхні кубіти, отримаємо одне із значень: 00, 01, 10 або 11. Постає питання: чи буде отримано 00? Якщо функція константна, отримаємо значення 00 з ймовірністю 1. Якщо функція збалансована, буде отримано 00 з ймовірністю 0. Тобто, отримавши в результаті вимірювання значення 00, можна стверджувати, що функція є константою. У разі іншого результату, відмінного від 00, можна стверджувати, що функція є збалансованою.

Цей доказ є вірним для будь-якого n . Безпосередньо перед вимірюванням кубітів амплітуда ймовірності для $|0\dots 0\rangle$ дорівнює:

$$\frac{1}{2^n} \left((-1)^{f(0,0,\dots,0)} + (-1)^{f(0,0,\dots,1)} + \dots + (-1)^{f(1,1,\dots,1)} \right).$$

Як і для $n = 2$, в результаті отримаємо значення ± 1 , якщо f константна, і 0, якщо f – збалансована. Якщо всі вимірювання дадуть 0, то функція є константою. Якщо хоча б одне вимірювання дасть 1, функція є збалансованою. Отже, розв'язати задачу Дойча-Йожи для будь-якого n можна з використанням наведеної на рис. 6.19 схеми. *Оракулу буде поставлене лише одне питання.* Схему треба запустити тільки один раз, у термінах квантового апаратного забезпечення це називається «постріл» (англ. Shot), але у посібнику всюди

використаний термін – запуск. У класичному випадку для вирішення того ж завдання в гіршому випадку потрібно було б поставити $2^{n-1}+1$ питання, тобто прискорення є дуже суттєвим.

Ймовірність вимірювання 0 у першому кубіті визначається рівнянням:

$$\left(\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right)^2.$$

Подібно до алгоритму Дойча, коли $f(x)$ є константною функцією, це рівняння показує, що ймовірність вимірювання 0 для першого кубіта дорівнює 100%. Коли $f(x)$ є збалансованою функцією, при вимірюванні першого кубіту завжди буде отримано значення 1.

Хоча прямого практичного використання цього алгоритму немає, квантовий алгоритм Дойча-Йожи може виконувати завдання набагато швидше, ніж відповідний класичний алгоритм. Справжнє прискорення можна отримати лише за умови використання справжнього квантового комп'ютера.

Розглянуті алгоритми показують, що для розв'язання конкретної задачі потрібна одна обчислювальна операція (запуск схеми), тоді як для класичного підходу потрібна експоненціальна кількість обчислень.

Для квантових обчислень важливо:

- розробити квантовий алгоритм, який швидший за відповідні класичні алгоритми;
- знайти варіант використання алгоритму на практиці.

На рис. 6.21 представлена схема алгоритму Дойча-Йожи та результат роботи схеми [60] для постійної функції, яка представлена одиничною матрицею – оператором I . Функція є константою, отже в результаті вимірювання завжди буде отримано значення 00.

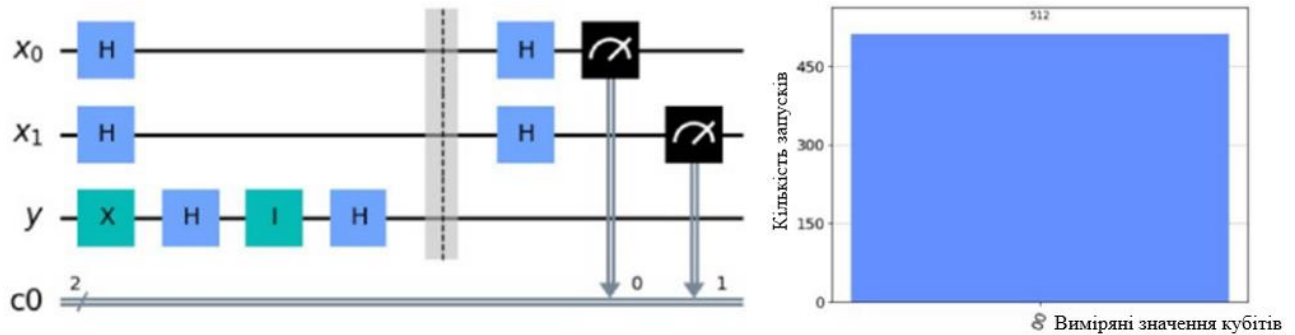


Рис. 6.21 – Схема алгоритму Дойча-Йожи та результат роботи схеми [60] для постійної функції.

На рис. 6.22 представлена схема алгоритму Дойча-Йожи та результат роботи схеми [60] для збалансованої функції, яка представлена оператором XOR . Функція є збалансованою, отже в результаті вимірювання завжди буде отримано значення 11.

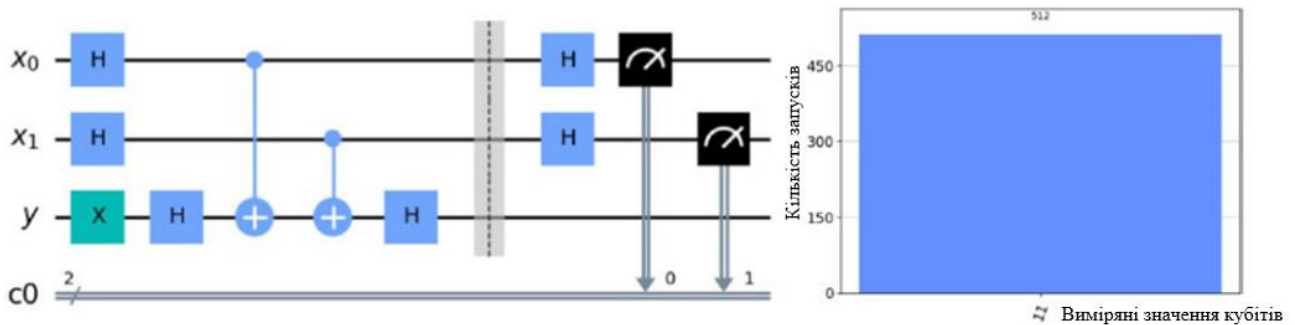


Рис. 6.22 – Схема алгоритму Дойча-Йожи та результат роботи схеми [60] для збалансованої функції.

На рис. 6.23 представлена схема алгоритму Дойча-Йожи та результат роботи схеми з трьома кубітами [60] для збалансованої функції, яка представлена оператором XOR . Функція є збалансованою, отже в результаті вимірювання завжди буде отримано значення 111.

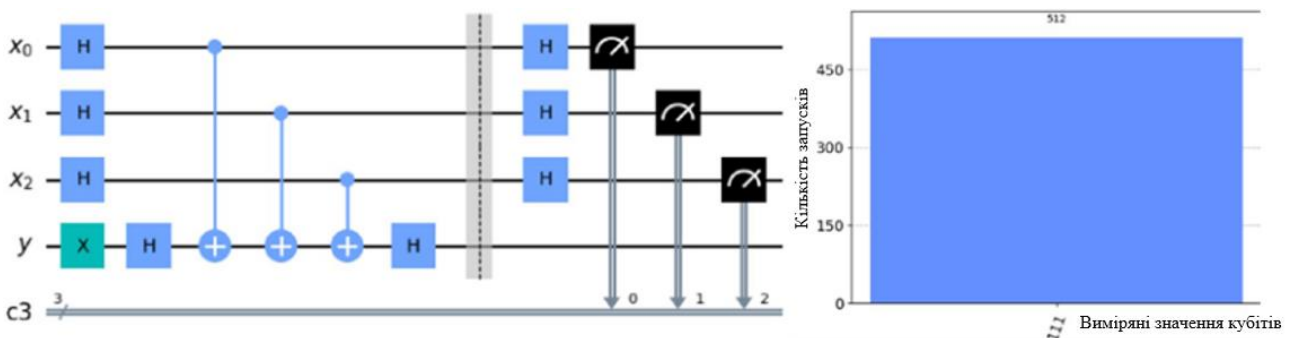


Рис. 6.23 – Схема алгоритму Дойча-Йожи та результат роботи схеми [60] для збалансованої функції з трьома кубітами.

Наведений на рис. 6.24 код Python реалізує схему алгоритму Дойча-Йожи з двома кубітами за допомогою бібліотеки Qiskit. У програмі передбачена перевірка як функції-константи (оператор I), так і збалансованої функції (оператор XOR).

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit import Aer, execute
from qiskit.tools.visualization import plot_histogram

# Створення квантових/класичних регістрів та квантової схеми
qin = QuantumRegister(2, 'x')
qout = QuantumRegister(1, 'y')
c = ClassicalRegister(2)
qc = QuantumCircuit(qin,qout,c)

# Визначення збалансованої функції XOR
def balanced2(circ, input, output) :
    circ.cx(input[0],output)
    circ.cx(input[1],output)
# Визначення константної функції, яка нічого не змінює
def constant2(circ, input, output) :
    circ.i(output)

# Побудова схеми Дойча-Йожи
# Спочатку застосовано фазовий зсув для заперечення, де  $f(x) == 1$ 
qc.h(qin)
qc.x(qout)
qc.h(qout)

# balanced2() для збалансованого або constant2() для константи
constant2(qc,qin,qout)
#balanced2(qc,qin,qout)
qc.h(qout)

# бар'єр для покращення вигляду схеми
qc.barrier()

# Потім алгоритм Уолша-Адамара на вхідних бітах
qc.h(qin)

# Для постійної функції вихід  $|00\rangle$  з ймовірністю 1
# Для збалансованої функції вихід  $|11\rangle$  з ймовірністю 1
qc.measure(qin,c)

# дисплей працює лише в Quantum Lab
display(qc.draw())

# Моделювання та показ результатів
backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=512)
result = job.result()
plot_histogram(result.get_counts())
```

Рис. 6.24 – Реалізація схеми алгоритму Дойча-Йожи за допомогою бібліотеки Qiskit.

6.8 Гібридні квантові алгоритми, квантовий алгоритм наближеної оптимізації

Квантовий алгоритм наближеної оптимізації – це гібридний квантово-класичний алгоритм для пошуку наближених розв’язків задач комбінаторної оптимізації. Він використовує як обчислювальну потужність квантового процесора для підготовки станів, так і гнучкість класичного комп’ютера для оптимізації параметрів.

Розглянемо загальну схему розв’язування задач бінарної комбінаторної оптимізації за допомогою квантового алгоритму наближеної оптимізації (QAOA), представленого в [61]. Алгоритм QAOA належить до класу гібридних квантових алгоритмів, які є допоміжним інструментом для поточної ери NISQ. У цих умовах використання квантових процесорів (NISQ) алгоритм QAOA також є новим підходом для порівняльного аналізу квантових пристроїв і є головним кандидатом для демонстрації практичного квантового прискорення на пристрої NISQ у найближчій перспективі. Для перевірки отриманих з використанням QAOA результатів, зазвичай їх порівнюють з точними результатами, отриманими від класичних вирішувачів (англ. Solver) QUBO.

Задачі комбінаторної оптимізації повсюдно зустрічаються в багатьох галузях науки та прикладних областях. Застосування можна знайти, наприклад, у логістиці, плануванні, плануванні та оптимізації фінансового портфеля. Задачі комбінаторної оптимізації – це задачі, що включають велику кількість рішень «так/ні», причому кожен набір рішень дає відповідне значення цільової функції, таке як значення витрат або прибутку. Через комбінаторний вибух простору рішень з великою кількістю змінних, пошук прийнятних або наближених до оптимальних рішень є надзвичайно складним завданням.

При використанні QAOA формальне представлення задачі зазвичай виконують за допомогою моделі квадратичної безобмеженої бінарної оптимізації (англ. Quadratic Unconstrained Binary Optimization, QUBO), яка дозволяє представити широкий спектр *NP*-складних задач комбінаторної оптимізації. Відомі приклади включають задачі розташування складу, задачі

бюджетування капіталу, задачі розподілу завдань, задачі максимального скорочення, задачі пакування та ін.

Покрокова інструкція застосування квантового алгоритму QAOA зі створенням відповідного параметризованого анзацу квантової схеми (англ. Quantum circuit ansatz) наведена на основі Braket SDK за допомогою простих модульних будівельних блоків у [62]. Крім квантового апаратного забезпечення і Braket SDK використовуються готові оптимізатори `scipy` з відкритим кодом для класичної чисельної оптимізації. При застосуванні Braket SDK є можливість використовувати класичні та квантові симулятори для виконання схеми, щоб підтвердити правильність схеми, а також використовувати фактичне квантове апаратне забезпечення, змінивши визначення об'єкта пристрою (за умови, що набір вентилів, який використовується в анзаці, підтримується пристроєм, як у випадку з IonQ). Для Rigetti схему потрібно модифікувати застосуванням додаткових вентилів.

Квантові комп'ютери мають потенціал перевершити навіть найпотужніші класичні комп'ютери в низці обчислювальних задач: оптимізації, хімії, матеріалознавстві та криптографії. Однак класичні квантові алгоритми (Саймона, Шора або Гровера) потребують велику кількість квантових вентилів, які наразі недоступні за допомогою сучасних квантових технологій. Зокрема, зазвичай вважається, що ці алгоритми працюють лише за умови відмовостійкості, яку забезпечує квантова корекція помилок. В сучасну еру шумних проміжних пристроїв (NISQ) квантові комп'ютери не мають достатньо великої кількості фізичних кубітів для реалізації протоколів корекції помилок, що робить багато класичних квантових алгоритмів непридатним для сучасних квантових процесорів.

У цих умовах фокус розробок нових алгоритмів значно змістився до класу гібридних квантових алгоритмів, які не потребують квантової корекції помилок. У цих гібридних квантових алгоритмах звичайні квантові комп'ютери з високим рівнем шумів та швидкою втратою квантових станів використовуються лише як

допоміжні засоби в межах більшого класичного циклу оптимізації, як показано на рис. 6.25. Тут небажані ефекти шуму пригнічуються шляхом навмисного обмеження квантових схем на квантовому процесорному блоці (англ. Quantum processing unit, QPU) короткими імпульсами обчислення, а необхідність тривалого часу когерентності (як це потрібно для стандартного набору квантових алгоритмів) замінюється класичними накладними витратами через повторення вимірювань та класичні обчислення.

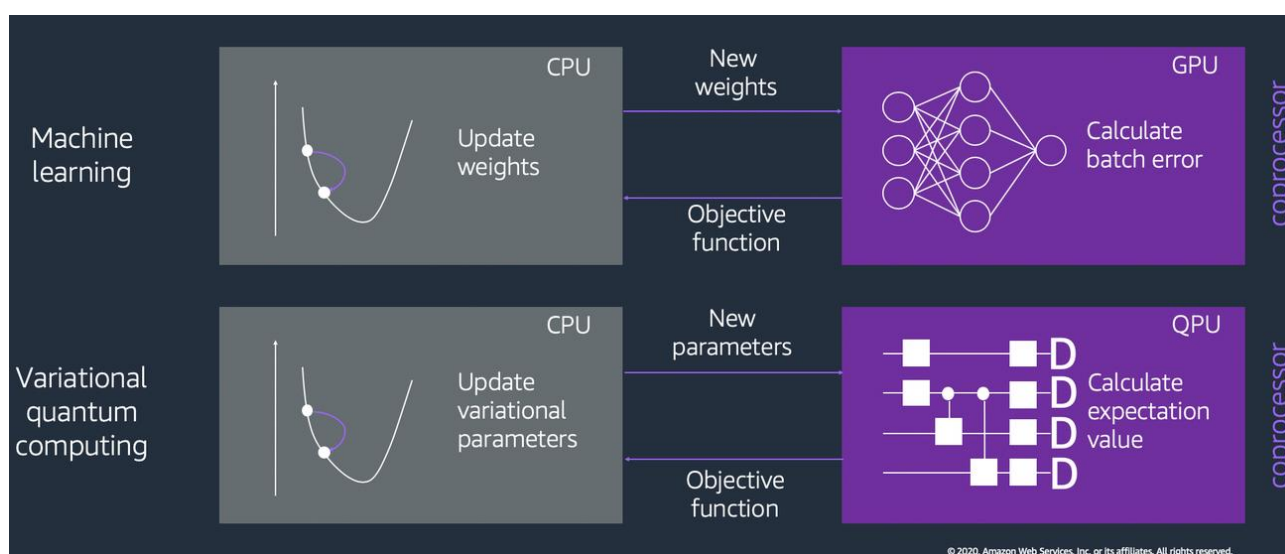


Рис. 6.25 – Схема оптимізації з використанням класичних та квантових обчислень [62].

Квантовий наближений алгоритм оптимізації відноситься до варіаційних квантових алгоритмів [61], які вважаються перспективними претендентами для демонстрації квантової переваги з використанням існуючих звичайних QPU у таких галузях, як квантова хімія [21], моделювання конденсованих середовищ [63] та задачі дискретної оптимізації [64].

Принцип роботи варіаційних квантових обчислень нагадує навчання глибоких нейронних мереж: під час навчання нейронної мережі потрібно мінімізувати задану цільову функцію, яка зазвичай представляє собою помилку при роботі з на навчальним набором. Щоб мінімізувати цю помилку, на початку зазвичай вибирають початковий стан – ваги зав’язків нейронної мережі. Співпроцесор, у випадку класичних обчислень – це графічний процесор, приймає ці ваги, які визначають точну операцію для виконання, і обчислює вихід нейронної мережі. Цей вихід потім використовується для обчислення значення

заданої цільової функції, яка, у свою чергу, використовується процесором для обґрунтованого припущення щодо оновлення ваг, і цикл продовжується.

Варіаційні квантові алгоритми працюють точно так само, використовуючи параметризовані квантові схеми замість параметризованих нейронних мереж та замінюючи графічний процесор на QPU. Так само цикл починається з вибору початкових параметрів, які визначають квантову схему, QPU виконує цю схему, виконує вимірювання для обчислення цільової функції, передає це значення (разом з поточними значеннями параметрів) назад до класичного процесора, який оновлює параметри на основі цієї інформації.

Координація цього робочого процесу для квантових комп'ютерів набагато складніша, ніж у попередньому випадку. Квантові комп'ютери розташовані в спеціалізованих лабораторних приміщеннях, зазвичай є однопоточними та мають спеціальні вимоги до затримки. Наприклад, сервіс Amazon Braket надає засоби підтримки та виконання робочих процесів для квантових комп'ютерів.

Загальна схема роботи гібридного алгоритму QAOA складається з шести кроків.

Перший крок присвячений формалізації задачі та побудові гамільтоніану. При цьому формулюється класична цільова функція $C(x)$, яка відображається на квантовий гамільтоніан задачі H_C . Власні стани H_C при цьому відповідають можливим розв'язкам задачі, а мінімальне власне значення H_C відповідає оптимальному розв'язку.

На **другому кроці** виконується ініціалізація квантового стану. Кубіти переводяться в початковий стан суперпозиції за допомогою вентилів H , який представлений рівнянням $|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$. Цей стан є основним станом змішуючого гамільтоніану $H_M = \sum_{i=1}^n X_i$.

На **третьому кроці** виконується параметризована еволюція у спосіб запуску квантової схеми. Стан еволюціонує шляхом почергового застосування двох

унітарних операторів протягом p шарів (параметрів). Загальна кількість класичних параметрів дорівнює $2p$, параметрами є вектори $\vec{\gamma}$ та $\vec{\beta}$.

Для кожного шару i від 1 до p застосовується фазовий та змішуючий оператори у такій послідовності:

1. Застосовується фазовий оператор $U(H_C, \gamma_i) = e^{-i\gamma_i H_C}$, який кодує вартість кожного варіанту у фазу стану.

2. Застосовується змішуючий оператор $U(H_M, \beta_i) = e^{-i\beta_i H_M}$, який забезпечує квантову інтерференцію між різними варіантами.

$$\text{Кінцевий стан визначається рівнянням } |\psi(\vec{\gamma}, \vec{\beta})\rangle = \prod_{i=1}^p e^{-i\beta_i H_M} e^{-i\gamma_i H_C} |\psi_0\rangle.$$

На **четвертому кроці** виконується вимірювання кінцевого стану $|\psi(\vec{\gamma}, \vec{\beta})\rangle$ в обчислювальному базисі Z . Вимірювання повторюється багатократно (серія «шотів») для отримання статистики.

Класична оптимізація реалізує цикл зворотного зв'язку на **п'ятому кроці**.

На основі результатів вимірювань обчислюється середнє очікуване значення енергії за формулою $E(\vec{\gamma}, \vec{\beta}) = \langle \psi(\vec{\gamma}, \vec{\beta}) | H_C | \psi(\vec{\gamma}, \vec{\beta}) \rangle$. Результат передається класичному оптимізатору (наприклад, COBYLA, Nelder-Mead або градієнтний спуск). Оптимізатор коригує параметри $\vec{\gamma}$ та $\vec{\beta}$ для мінімізації енергії E . Нові параметри передаються на Крок 3. Цикл повторюється до досягнення збіжності.

На **шостому кроці** виконується завершальне вимірювання. Після знаходження оптимальних параметрів $\vec{\gamma}^*$ та $\vec{\beta}^*$ вони передаються для запуску квантової схеми. Найбільш ймовірний виміряний бітовий рядок (або група рядків) є наближеним оптимальним розв'язком класичної задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7), 467-488.
2. Paul B 1980 The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines *J. Stat. Phys.* 22 563–91.
3. Paul B 1982 Quantum mechanical models of Turing machines that dissipate no energy *Phys. Rev. Lett.* 48 1581–5.
4. Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400(1818), 97-117.
5. Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124-134.
6. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212-219.
7. Bennett, C. H., Bernstein, E., Brassard, G., & Vazirani, U. (1997). Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5), 1510-1523.
8. Deutsch, D. (1989). Quantum computational networks. *Proceedings of the Royal Society of London A*, 425(1868), 73-90.
9. Deutsch, D., & Jozsa, R. (1992). Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439(1907), 553-558.
10. Bernstein, E., & Vazirani, U. (1997). Quantum complexity theory. *SIAM Journal on Computing*, 26(5), 1411-1473.
<https://doi.org/10.1137/S0097539796300921>

11. Simon, D. R. (1997). On the power of quantum computation. *SIAM Journal on Computing*, 26(5), 1474-1483.
12. Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2, 79.
13. Peruzzo, A., et al. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5, 4213.
14. Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv:1411.4028*.
15. Biamonte, J., et al. (2017). Quantum machine learning. *Nature*, 549(7671), 195-202.
16. Gisin, N., Ribordy, G., Tittel, W., & Zbinden, H. (2002). Quantum cryptography. *Reviews of Modern Physics*, 74(1), 145-195.
17. Bennett, C. H., & Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 175-179.
18. Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical Review Letters*, 67(6), 661-663.
19. Bennett, C. H., et al. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70(13), 1895-1899.
20. Bennett, C. H., & Wiesner, S. J. (1992). Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20), 2881-2884.
21. Cao, Y., et al. (2019). Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19), 10856-10915.
22. Orús, R., Múgel, S., & Lizaso, E. (2019). Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4, 100028.

23. Ajagekar, A., Humble, T., & You, F. (2020). Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems. *Computers & Chemical Engineering*, 132, 106630.
24. Schuld, M., & Petruccione, F. (2018). *Supervised Learning with Quantum Computers*. Springer. (Vol. 17, p. 3).
25. LaRose, R., & Coyle, B. (2020). Robust data encodings for quantum classifiers. *Physical Review A*, 102(3), 032420.
26. Aleksandrowicz, G., et al. (2019). Qiskit: An open-source framework for quantum computing. Zenodo.
27. Amazon Braket Accelerate quantum computing research, [Online] Available from: <https://aws.amazon.com/braket/> [Accessed Aug 20, 2026].
28. Svore, K., et al. (2018). Q#: Enabling scalable quantum computing and development with a high-level DSL. *Proceedings of the Real World Domain Specific Languages Workshop*, 1-10.
29. Bergholm, V., et al. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968.
30. Python framework for creating, editing, and running Noisy Intermediate-Scale Quantum (NISQ) circuits. [Online] Available from: <https://github.com/quantumlib/Cirq> [Accessed Aug 20, 2026].
31. Johnson M W et al 2011 Quantum annealing with manufactured spins *Nature* 473, 194–8.
32. Arute, F., et al. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779), 505-510.
33. Shor, P. W. (1995). Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4), R2493-R2496.
34. Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2), 303-332.

35. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 120-126.
36. Majorana 1 Explained: The Path to a Million Qubits, [Online] Available from: <https://www.youtube.com/watch?v=wSHmygPQukQ> [Accessed Aug 20, 2026].
37. Garhwal, S., Ghorani, M., Ahmad, A.: Quantum programming language: a systematic review of research topic and top cited languages. *Arch. Comput. Methods Eng.* 28(2), 289–310 (Mar. 2021). ISSN: 1134-3060, 1886-1784. <https://doi.org/10.1007/s11831-019-09372-6>
38. Ying, M., Zhou, L., Li, Y.: Reasoning about Parallel Quantum Programs (Aug. 2019). <https://doi.org/10.48550/arXiv.1810.11334>. <http://arxiv.org/abs/1810.11334>
39. Ömer, B.: Procedural quantum programming. *AIP Confer. Proc.* 627(1), 276–285 (Sept. 2002). ISSN: 0094-243X. <https://doi.org/10.1063/1.1503695>. eprint: https://pubs.aip.org/aip/acp/article-pdf/627/1/276/11571870/276_1_online.pdf
40. Bichsel, B., et al. (June 2020) Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics. In: *Proc. 41st ACM SIGPLAN*, pp. 286–300. ACM. ISBN: 978-1-4503-7613-6. <https://doi.org/10.1145/3385412.3386007> (visited on 24/04/2023)
41. Selinger, P.: Towards a quantum programming language. *Math. Struct. Comput. Sci.* 14(4), 527–586 (Aug. 2004). ISSN: 0960-1295, 1469-8072. <https://doi.org/10.1017/S0960129504004256>
42. Green, A.S. et al.: Quipper: a scalable quantum programming language. *SIGPLAN Not.* 48(6), 333–342 (2013). ISSN: 0362-1340. <https://doi.org/10.1145/2499370.2462177>
43. Wecker, D., Svore, K.M.: LIQUi|>: A Software Design Architecture and Domain-Specific Language for Quantum Computing (Feb. 2014). CoRR abs/1402.4467. <https://doi.org/10.48550/arXiv.1402.4467>

44. Maurer, W. (2024). Superoperators for Quantum Software Engineering. B I. Exman, R. Pérez-Castillo, M. Piattini, & M. Felderer (Ред.), Quantum Software (с. 45–68). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-64136-7_3.
45. Kong K.C. (2024) Introduction to Quantum Algorithms, Lecture notes.
46. Qiu, J., Liu, Y., Hu, L., Wu, Y., Niu, J., Zhang, L., ... & Yu, D. (2025). Deterministic quantum state and gate teleportation between distant superconducting chips. *Science Bulletin*, 70(3), 351-358.
47. NIST Team Breaks Distance Record for Quantum Teleportation, [Online] Available from: <https://www.nist.gov/news-events/news/2015/09/nist-team-breaks-distance-record-quantum-teleportation> [Accessed Aug 20, 2026].
48. Scientists Capture W State, Unlocking Quantum Teleportation, [Online] Available from: <https://scitechdaily.com/scientists-capture-w-state-unlocking-quantum-teleportation/> [Accessed Aug 20, 2026].
49. Record-Breaking Quantum Transmission Via Micius, [Online] Available from: <https://postquantum.com/industry-news/micius-quantum-communications/> [Accessed Aug 20, 2026].
50. Ruane, J., Kiesow, E., Galatsanos, J., Dukatz, C., Blomquist, E., & Shukla, P. (2025). *Quantum Index Report 2025* (No. arXiv:2506.04259). arXiv. <https://doi.org/10.48550/arXiv.2506.04259>
51. Vos, J. (2022). *Quantum Computing in Action*. Simon and Schuster.
52. Big O notation, [Online] Available from: https://web.mit.edu/16.070/www/lecture/big_o.pdf [Accessed Aug 20, 2026].
53. Johnson, R. (2024) *Quantum Algorithms in Action: A Practical Guide to Implementation with Qiskit*. HiTeX Press. 250.
54. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

55. Strubell, E. (2011). An introduction to quantum algorithms. *COS498 Chawathe Spring, 13*, 19.
56. Toffoli, T. (1980, July). Reversible computing. In *International colloquium on automata, languages, and programming* (pp. 632-644). Berlin, Heidelberg: Springer Berlin Heidelberg.
57. X. M. Jin, J. G. Ren, B. Yang, Z. H. Yi, F. Zhou, X. F. Xu, S. K. Wang, D. Yang, Y. F. Hu, S. Jiang, T. Yang, H. Yin, K. Chen, C. Z. Peng, and J. W. Pan, “Experimental free-space quantum teleportation”, *Nature Photonics*, vol. 4, no. 6, pp. 376–381, 2010, issn: 17494885. doi: 10.1038/nphoton.2010.87. [Online]. Available: <http://dx.doi.org/10.1038/nphoton.2010.87>.
58. Welcome to Quirk. A drag-and-drop quantum circuit simulator. [Online] Available from: <https://algassert.com/quirk> [Accessed Aug 20, 2026].
59. Bernhardt, C. (2019). *Quantum computing for everyone*. Mit Press.
60. V. Silva, *Quantum Computing by Practice: Python Programming in the Cloud with Qiskit and IBM-Q*. Berkeley, CA: Apress, 2024. doi: [10.1007/978-1-4842-9991-3](https://doi.org/10.1007/978-1-4842-9991-3)
61. Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.
62. Quantum approximate optimization algorithm (QAOA), [Online] Available from: https://github.com/amazon-braket/amazon-braket-examples/blob/main/examples/hybrid_quantum_algorithms/QAOA/QAOA_braket.ipynb [Accessed Aug 20, 2026].
63. Smith, A., Kim, M. S., Pollmann, F., & Knolle, J. (2019). Simulating quantum many-body dynamics on a current digital quantum computer. *npj Quantum Information*, 5(1), 106.
64. Zhou, L., Wang, S. T., Choi, S., Pichler, H., & Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2), 021067.