НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК 510.52

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Побудова ефективних алгоритмів арифметичних операцій в квантовій моделі обчислень»**

Виконав:
студент IV курсу, групи ФІ-03
Швець Катерина Михайлівна _____

Керівник:
к.ф., м.н., ст. викладач
Фесенко Андрій В'ячеславович _____

Рецензент:
заст. дир. ФТІ з наукової роботи, к.ф-м.н.
Терещенко Іван Миколайович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2024 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Швець Катерина Михайлівна

1. Тема роботи: «*Побудова ефективних алгоритмів арифметичних операцій в квантовій моделі обчислень*», науковий керівник дисертації: к.ф., м.н., ст. викладач Фесенко Андрій В'ячеславович,

затверджені наказом по університету №__ від «__» _____ 2024 р.

2. Термін подання студентом роботи: «__» _____ 2024 р.

3. Об'єкт дослідження: квантова модель обчислень

4. Предмет дослідження: алгоритми арифметичних операцій в квантовій моделі обчислень

5. Перелік завдань: побудова оцінок складності базових арифметичних операцій; побудова квантового алгоритму редукції за спеціальними модулями $2^n \pm 1$ та множення поліномів з коефіцієнтами $0; 1; -1$, побудова їх оцінок складності.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Презентація доповіді

7. Орієнтовний перелік публікацій: планується доповідь на всеукраїнській конференції

8. Дата видачі завдання: 10 вересня 2023 р.

Календарний план

№ з/п	Назва етапів виконання бакалаврської дисертації	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2023 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2023 р.	Виконано
3	Огляд алгоритмів арифметичних операцій та вивчення особливостей методів побудови оцінок складності в квантовій моделі обчислень	Листопад 2023 р.	Виконано
4	Побудова оцінок складності квантових алгоритмів арифметичних операцій додавання та множення	Грудень 2023 р.	Виконано
5	Огляд алгоритмів редукції за спеціальним модулем та множення поліномів	Січень 2024 р.	Виконано
6	Побудова квантового алгоритму редукції за спеціальним модулем	Лютий-березень 2024 р.	Виконано
7	Побудова квантового алгоритму множення поліномів	Березень-квітень 2024 р.	Виконано
8	Оформлення результатів дослідження	Травень 2024 р.	Виконано

Студент _____ Катерина ШВЕЦЬ

Керівник _____ Андрій ФЕСЕНКО

РЕФЕРАТ

Кваліфікаційна робота містить: 46 стор., 0 рисунки, 3 таблиць, 5 джерел.

У даному дослідженні вперше були побудовані квантові алгоритми спеціальних арифметичних операцій: редукція за спеціальним модулем вигляду $2^n \pm 1$ та множення поліномів з коефіцієнтами $-1; 0; 1$. Побудовані оцінки складності для аналізу їх ефективності та розглянуті області застосувань цих алгоритмів в квантовій моделі обчислень. Були проаналізовані базові алгоритми арифметичних операцій додавання та множення.

Метою дослідження є розробка, аналіз та оцінка ефективності нових квантових алгоритмів для виконання специфічних арифметичних операцій.

Об'єктом дослідження є квантова модель обчислень.

Предметом дослідження є алгоритми арифметичних операцій в квантовій моделі обчислень.

КВАНТОВІ АЛГОРИТМИ, КВАНТОВА РЕДУКЦІЯ, КВАНТОВЕ МНОЖЕННЯ ПОЛІНОМІВ, КВАНТОВІ АРИФМЕТИЧНІ ОПЕРАЦІЇ

ABSTRACT

The thesis contains: 46 pages, 0 figures, 3 tables, 5 sources.

In this research, for the first time, quantum algorithms were built for special arithmetic operations: reduction by a special modulus of the form $2^n \pm 1$ and multiplication of polynomials with coefficients $-1, 0, 1$. The complexity estimates for analyzing their efficiency are constructed and the areas of application of these algorithms are considered. the areas of application of these algorithms in the quantum model of computing. We have the basic algorithms of arithmetic operations of addition and and multiplication operations.

The purpose of the thesis is to develop, analyze, and evaluate the effectiveness of of new quantum algorithms for performing specific arithmetic operations.

The object of the thesis is the quantum model of computing.

The subject of the thesis is algorithms for arithmetic operations in the in the quantum model of computing.

QUANTUM ALGORITHMS, QUANTUM REDUCTION, QUANTUM POLYNOMIAL MULTIPLICATION, QUANTUM MULTIPLICATION OF POLYNOMIALS, QUANTUM ARITHMETIC OPERATIONS, QUANTUM ARITHMETIC OPERATIONS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\otimes$ – Тензорне множення

H – Вентиль Адамара

QFT – квантове перетворення Фур'є

IQFT – зворотне квантове перетворення Фур'є

RCA – Ripple-Carry Adder

ВСТУП

Актуальність дослідження. Актуальність даного дослідження полягає в розумінні наскільки ефективні арифметичні операції в квантовому обчисленні та полягає в виявленні шляхів оптимізації операцій виходячи з оцінок складності, що має велике значення для майбутнього розвитку квантових технологій. Також арифметичні операції є одними з базовими й вони лежать в основі багатьох потужних алгоритмів. Через це їх оптимізація має велике значення.

Також в даному досліджені побудовано нові ефективні квантові алгоритми специфічних арифметичних операцій таких як редукція за спеціальним модулем вигляду $2^n \pm 1$ та множення поліномів з коефіцієнтами $0, 1, -1$. Саме зараз досліджується ефективне застосування криптографічних алгоритмів в квантовій моделі обчислень і для цього дуже важливо побудувати ефективні вирішення для цього. Наприклад, редукція використовується в алгоритмі Шора (факторизація великих чисел), що може підірвати стійкість криптосистем, які базуються на стійкості дискретного логарифмування. Також в багатьох криптографічних алгоритмах та деяких геш-функціях використовуються модулі вигляду $2^n \pm 1$ і тому є актуальним побудова алгоритму редукції цих моделей для ефективного вирішення задач в квантовій моделі обчислень. Крім того, алгоритми множення поліномів мають теж широке застосування у квантовій криптографії та створення деяких геш-функцій (SHA-3, геш-функції на основі решіток), які відіграють важливу роль у забезпеченні цілісності даних, автентифікації і цифрових підписів.

Метою дослідження є розробка, аналіз та оцінка ефективності нових квантових алгоритмів для виконання специфічних арифметичних операцій. Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає у побудові ефективних квантових алгоритмів редукції за спеціальним модулем та множенням поліномів з коефіцієнтами

$0, 1, -1$. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести аналіз існуючих методів і алгоритмів для виконання арифметичних операцій у квантовій моделі обчислень;

2) побудувати оцінки складності існуючих алгоритмів додавання та множення;

3) розробити квантовий алгоритм редукції за спеціальним модулем $2^n \pm 1$;

4) створити квантовий алгоритм множення поліномів з коефіцієнтами $0, 1, -1$;

5) провести теоретичний аналіз і обґрунтування ефективності запропонованих алгоритмів.

Об'єктом дослідження є квантова модель обчислень.

Предметом дослідження є алгоритми арифметичних операцій в квантовій моделі обчислень.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: теорії складності алгоритмів, методи комп'ютерного та статистичного моделювання, теоретичний аналіз, алгоритмічний підхід, порівняльний аналіз.

Наукова новизна отриманих результатів полягає побудові квантових алгоритмів деяких специфічних арифметичних операцій: редукція за спеціальним модулем вигляду $2^n \pm 1$ та множення поліномів з коефіцієнтами $0, 1, -1$.

Практичне значення результатів полягає у використані їх для підвищення ефективності квантових обчислень у різних галузях, таких як криптографія, моделювання фізичних процесів, оптимізація та машинне навчання. Розроблені алгоритми можуть бути впроваджені в квантові комп'ютери для забезпечення швидшого та більш точного виконання складних арифметичних операцій.

1 ТЕОРЕТИЧНІ ОСНОВИ КВАНТОВИХ ОБЧИСЛЕНЬ

Цей розділ присвячено огляду основних понять та положень квантової системи обчислень, в якій будуть побудовані оцінки складності арифметичних операцій та алгоритмів, в яких будуть застосовуватися ці операції.

1.1 Огляд квантової моделі обчислень

В наш час розвиток напрямок квантових технологій стрімко набирає обертів. Саме цей напрямок пропонує інший підхід для вирішення задач, які є недоступними для класичних комп'ютерів. Через це зростає інтерес до квантових обчислень та їх потенційного впливу на обчислювальну сферу.

Переходячи до квантової моделі обчислень важливо розібратися з її особливостями, які ґрунтовно відрізняються від звичної нам класичної моделі. Її значна перевага в тому, що вона дозволяє частинкам знаходитися в декількох станах одночасно в будь-який час, за допомогою чого операції виконуються швидше та потребують менше ресурсів. Це і є головна відмінність від класичної моделі, де використовуються звичайні біти, які можуть перебувати лише в двох станах 0 або 1.

Квантовий комп'ютер

З появи звичайного комп'ютера прогрес не зупинився. З кожним днем зростають обсяги інформації, з якою потрібно працювати. Тому людство не зупиняється і продовжує роботу над знаходженням нових вирішень для прискорення обробки інформації та роботи алгоритмів. Саме квантовий комп'ютер постає перед нами чудовим вирішенням цих проблем. Це можливо через одну з властивостей квантового комп'ютера –

паралельність. Послідовність процесів потребує час, але його можна скоротити якщо ці процеси роботи паралельно.

Вперше ідею побудови квантових алгоритмів у 80-х роках висунув Ю.І.Манін "Вычислимое и невычислимое". Це був початок для досліджень та спроби створити його. Зараз багато компаній займаються їх розробкою і вже мають деякі результати. Наприклад, компанія IBM розробила хмарне середовище, яке дозволяє виконувати квантові обчислення через інтернет. Компанія GOOGLE вже має свій квантовий комп'ютер, який вже може виконувати деякі квантові алгоритми. Зараз ця область дуже активно розвивається і є над чим працювати.

Основна відмінність квантового комп'ютера від класичного саме в представлення інформації. Якщо вся інформація в класичній моделі представляються та обробляється через біти, то у квантовому комп'ютері через кубіти.

Основні поняття квантових обчислень

Якщо розглядати на прикладі кулі, то за допомогою бітів ми можемо перебувати лише на полюсах (0 або 1), а ось квантові біти дозволяють знаходиться в будь-якій точці кулі.

Означення 1.1. *Квантовий біт (кубіт, квабіт)* [**label1**] – одиниця інформації, що може перебувати в двох (відносно) стійких станах $|0\rangle$ і $|1\rangle$, а також у суперпозиційному стані:

$$|w\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \qquad |\alpha|^2 + |\beta|^2 = 1.$$

Якщо звичайні біти дискретно перебувають у станах 0 або 1, то кубіти знаходяться одночасно з деякою ймовірністю в кожному стані [**label2**]. Кожна ймовірність вказує на можливість знайти частинку в цьому стані (провівши спостереження), в той час як вона може перебувати в двох станах одночасно. Наприклад, для n кубітів частинка

може перебувати у 2^n станах. Можливість перебувати у декількох станах, яку дає квантовий комп'ютер, називається *суперпозицією*.

Означення 1.2. *Квантова заплутаність* – це явище, коли дві або більше квантові частинки стають взаємопов'язаними таким чином, що стан однієї частинки автоматично визначає стан іншої, незалежно від відстані між ними.

Ця властивість є одним з ключових елементів квантової механіки. При цій властивості стан однієї частинки може визначити стан іншої, незалежно від відстані між ними. Це суперечить класичній фізиці, де стани частинок залежать тільки, якщо вони локально взаємодіють.

Розглянемо стан Белла, який демонструє явище заплутаності. Це наступні стани двох кубітів, які описуються як:

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$
$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$
$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$
$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

Для створення цих станів є квантова схема з вентилями Адамара та CNOT.

Також заплутаність застосовується в квантовій криптографії для безпечного передання інформації. Наприклад, протокол розподілу ключів BB84 використовує заплутані частинки для передачі криптографічних ключів.

На кожен кубіт виділяються два класичних біта. У квантових системах обчислення для опису квантових станів використовується бра-кет нотація Дірака. Можемо визначити математичне представлення базисних станів кубіта:

$$|0\rangle = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \text{ – нульовий стан}$$

$$|1\rangle = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} \text{ – одиничний стан}$$

Означення 1.3. *Тензорне множення матриць та* $(A \otimes B)$ [**label4**] – один із способів множення матриць, при якому кожний елемент матриці множиться на кожний елемент матриці :

Нехай $A = ||\alpha_{ij}||$ порядку n та $||\beta_{ij}||$ порядку m, тоді за означенням:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \cdots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{n,1}B & \cdots & a_{n,n}B \end{bmatrix}.$$

На виході отримаємо матриці порядку mn.

Коли квантова система має два кубіта, то математично позначають так [**label3**]: $|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

Наглядно бачимо, що кожному кубіту відповідає 2 біта. Тобто розмір вектора, який описує стан квантової моделі дорівнює 2^n (n — кількість бітів).

Також для роботи в квантовій моделі обчислень нам знадобляться визначення квантової схеми та вентилів.

Для роботи з квантовими комп'ютером потрібно вміти будувати квантову схему. Наразі це є найпопулярнішим підходом.

Означення 1.4. *Квантова схема* – це модель для квантових обчислень, у якому обчислення є послідовністю квантових вентилів, які є оборотними перетвореннями на n-кубітному реєстрі.

Процес побудови квантової схеми починається з визначення числа кубітів і ініціалізації їх у початковому стані, зазвичай $|0\rangle$. Далі обираються квантові гейти, які будуть застосовані до кубітів у певному порядку для реалізації потрібної квантової операції. Схема візуалізується у вигляді квантової діаграми, де кожен кубіт зображується горизонтальною лінією, а гейти розташовуються вздовж цих ліній. Наприклад, гейт Адамара, застосований до першого кубіта, записується як H-гейт на відповідній лінії.

Означення 1.5. *Квантовий вентиль* – визначені операції над кубітами (аналог логічних операцій в класичній моделі обчислень).

Вони поділяються на прості (оперують над одним кубітом) та багатокубітні операції.

Означення 1.6. [**label5**] *Розмір квантової схеми* – мінімальна кількість елементарних операцій над фіксованим набором простих вентилів, потрібна для побудови цієї схеми.

Означення 1.7. *Квантовий регістр* – сукупність кубітів, що використовується зберігання та обробки інформації.

Означення 1.8. *Глибина квантової схеми* – кількість вентилів, які застосовуються до будь-якого з кубітів в квантовій схемі.

Глибина квантової схеми є одним з показників складності. При оптимізації ця глибина може зменшуватися, що буде значно впливати на час виконання.

Квантові вентилі

Саме квантові вентилі є одними з основних складових квантових обчислень. За їх допомогою ми можемо змінювати кубіти та робити з ними якісь дії для отримання певного результату. Тобто вони маніпулюють станами кубітів, забезпечуючи виконання квантових обчислень.

Типи квантових вентилів:

– Однокубітні

– Багатокубітні

– Ротаційні

Однокубтні вентилі використовуються для роботи над одним кубітом для маніпулювання окремими кубітами (наприклад, зміна стану). Багатокубітні використовуються вже для операцій над декількома кубітами, також двокубітні вентилі використовуються для досягнення заплутаності. Ротаційні в свою чергу забезпечують контроль над станами кубітів. Всі ці вентиль гарантують роботу квантової схеми, які в свою чергу реалізують квантові алгоритми.

Також однією з основних властивостей вентилів є унітарність. Принцип унітарності забезпечує збереження квантової інформації. Кожен унітарний оператор має обернений, так що застосування унітарного оператору до квантового стану та потім використання його оберненого приведе до початкового квантового стану. Завдяки унітарності, квантові алгоритми можуть точно контролювати квантові стани і забезпечувати можливість повернення до попередніх станів, що є важливим для задачі квантового виправлення помилок. Унітарність також дозволяє використовувати квантові ефекти, такі як інтерференція і заплутаність, для досягнення обчислювальних переваг, які недосяжні в класичних моделях.

Приклади унітарних вентилів: заперечення, H та CNOT. Їх ми

розглянемо далі.

Наведемо визначення основних квантових вентилів: прості (тотожну операцію (I), заперечення (NOT), ґейт Адамара (H)) та багатокубітні (CNOT, Тоффолі).

1. Тотожна операція (I)

При дії тотожного оператора кубіт не змінюється. Тобто, $I|0\rangle = |0\rangle$. Тотожний оператор у квантових обчисленнях відповідає тотожній матриці у класичній лінійній алгебрі. Приклад:

Для одного кубіта тотожний оператор можна представити як матрицю 2x2, яка має вигляд:

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

. Для прикладу оберемо один з базисних станів $|0\rangle = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$ Тоді:

$$I|0\rangle = \begin{bmatrix} 1 & 0 \\ \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} = \begin{bmatrix} 1*1+0*0 \\ \\ 0*1+1*0 \end{bmatrix} = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} = |0\rangle.$$

На виході отримали наш нульовий стан.

2. Заперечення (NOT)

Також відомий як квантовий вентиль Паулі-X. Цей вентиль відноситься до однокубітних та діє на кубіт так, що він на виході змінює свій стан. $NOT|0\rangle = |1\rangle$.

$$NOT = \begin{bmatrix} 0 & 1 \\ \\ 1 & 0 \end{bmatrix}$$

Для прикладу оберемо один з базисних станів $|0\rangle = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$ Тоді:

$$NOT|0\rangle = \begin{bmatrix} 0 & 1 \\ \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} = \begin{bmatrix} 0*1+1*0 \\ \\ 1*1+0*0 \end{bmatrix} = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} = |1\rangle$$

На виході отримали інший стан кубіту.

3. Ґейт Адамара (H)

Це один з найзастосовніших вентилів і є важливим для багатьох алгоритмів. Він потрібен для переведення чітко визначеного кубіта у стан суперпозиції з рівними ймовірностями. Матриця вентиля Адамара (H):

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ \\ 1 & -1 \end{bmatrix}.$$

Приклад 1.1. Для прикладу оберемо один з базисних станів $|0\rangle = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$ Тоді:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1*1+1*0 \\ \\ 1*1-1*0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Це основні вентилі для роботи з кубітом. Але дуже часто потрібно працювати з декількома кубітами. Розглянемо два основних багатокубітні вентеля.

4. Ротаційний

4. CNOT

Контрольоване заперечення відіграє велику роль у квантових

алгоритмах факторизації та пошуку, таких як алгоритм Гровера. Цей вентиль призначений для роботи з двома кубітами: один кубіт виступає як контрольний (англ. control qubit), а інший як цільовий (англ. target qubit). Цей вентиль змінює стан цільового кубіта тоді й тільки тоді, коли контрольний дорівнює одиниці.

Приклад 1.2.

$$|00\rangle \to |00\rangle$$
$$|10\rangle \to |11\rangle$$

В першому випадку контрольний кубіт є нулем, тому цільовий залишаємо без змін. У другому контрольний кубіт є одиницею, що впливає на цільовий та змінює його стан з 0 на 1.

5. Тоффолі

Це є розширенням вентиля CNOT. Основна відмінність в тому, що він працює з трьома кубітами, використовуючи два контрольних та один цільовий. Цей вентиль змінює стан цільового кубіта тоді й тільки тоді, коли контрольні кубіти одночасно дорівнюють одиниці.

Приклад 1.3.

$$|000\rangle \to |000\rangle$$
$$|100\rangle \to |100\rangle$$
$$|110\rangle \to |111\rangle$$

Також вентиль Тоффолі є необхідною складовою для реалізації класичних логічних вентилів у квантовому комп'ютері, що робить його універсальним для створення будь-яких квантових обчислювальних функцій.

З цього випливає, що виходячи за базові стани ми можемо переходити до n-кубітних систем. Наприклад, двокубітна система буде складатися з таких станів: $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ або $\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$.

[**label6**] Наприклад, стан $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ не можна зобразити через стани окремих кубітів. Це називається *сплутаністю станів*. Це явище виникає при роботі з декількома кубітами, що при взаємодії вони не можуть бути описані незалежно один від одного.

Тепер ознайомившись з основними поняттями квантової системи обчислень, можна переходити до опису квантових алгоритмів арифметичних операцій.

1.2 Квантові алгоритми

Переходячи від класичних схем, квантові алгоритми відкривають перед нами нові можливості через фундаментально новий підхід. Їх найголовніша перевага у можливості робити обчислення паралельно за рахунок суперпозиції та сплутаності. Розглянемо основні квантові алгоритми для арифметичних операцій.

Квантове додавання

Квантове додавання – одна з фундаментальних алгоритмів обчислень. Воно використовується у багатьох алгоритмах, таких як алгоритм Шора та Гровера. Алгоритм додавання на квантовому комп'ютері реалізується за допомогою двох вентилів CNOT та Тоффолі. Квантовий вентиль Тоффолі використовується для створення біту переноса (якщо обидва перші стани (контрольні) дорівнюють одиниці, то цільовий стан буде теж одиниця). CNOT є аналогом операції XOR, що дозволяє додавати біти. На виході отримуємо підсумковий біт для кожного розряду. Цей метод демонструє квантовий паралелізм, дозволяючи обчислювати результати для суперпозицій вхідних результатів одночасно.

Існує багато алгоритмів, які реалізують квантове додавання.

Найпопулярніші це алгоритм повного суматора (англ. Full Adder), алгоритм за методом Ripple-Carry та з використанням квантового перетворення Фур'є. Кожен з цих алгоритмів має свої особливості та застосовуються в різних напрямках залежно від завдання.

Квантове множення

Ця операція ґрунтується на операції суми. Нехай треба перемножити A та B. Тоді ми застосуємо операцію додавання A з самим собою B разів.[3] Два числа, та , зберігаються у відповідних регістрах, які кодуються n кубітами ($|x\rangle, |y\rangle$). Далі будемо використовувати два додаткових регістра "Control"(складається з одного кубіта та необхідний для зупинки операції) та "Accumulator"(повинен складатися з такої кількості кубітів, потрібної для зберігання результату). Також є контрольований порт D, який застосовується, коли регістр Control знаходиться в нульовому стані, для зменшення числа $|y\rangle$ на 1. На початку процедури регістри акумулятора та контролю ініціалізується нульовими станами. Перевіряємо стан $|y\rangle$ чи дорівнює він нульовому. Якщо ні, то запускаємо роботу алгоритму: зберігаємо $|y\rangle$ в акумуляторі у формі Фур'є. Далі використовуємо порт D для зменшення $|y\rangle$ на 1. І знову повторюємо цей цикл доки стан $|y\rangle$ не буде дорівнювати нулю.

Цей алгоритм представляє собою дуже складну квантову схему, яка включає повторне додавання і використання квантових Фур'є перетворень для досягнення множення.

Квантове перетворення Фур'є

Саме воно використовується у багатьох фундаментальних алгоритмах. Тому розглянемо його детальніше далі.

Спочатку сформуємо означення дискретного перетворення Фур'є.

Означення 1.9. *Дискретне перетворення Фур'є функції* $f(x)$ [**label1**] – перетворення, де – ціле число з проміжку $x \in \mathbb{Z}_n = \{0 : 2^n - 1\}$:

$$\hat{f}(x) = \frac{1}{\sqrt{2^n}} \sum_{y \in \mathbb{Z}_n} e^{\frac{i2\pi xy}{2^n}} f(y)$$

Загалом квантове перетворення Фур'є переводить n-кубітний квантовий регістр у суперпозицію, амплітуди якої дорівнюють коефіцієнтам перетворення Фур'є. Його реалізація використовує вентиль Адамара та оператор зміни фази, який змінює взаємні фази між кубітами. Отже, математично воно визначається так:

$$|x\rangle \xrightarrow{\text{QFT}} \frac{1}{2^{\frac{n}{2}}} \sum_{y} e^{\frac{i2\pi xy}{2^n}} |y\rangle$$

.

Це є ключовою складовою у багатьох квантових алгоритмах, включаючи арифметичні операції, алгоритм Шора для факторизації чисел та алгоритм Гровера для пошуку в базі даних.

Квантове ділення

Квантове ділення є складнішою арифметичною операцією порівняно з квантовим додаванням або множенням. У квантових обчисленнях ділення зазвичай реалізується через послідовність логічних операцій та використання більш складних квантових алгоритмів, що включають обчислення обернених чисел і виконання арифметичних операцій у багаторазовому вигляді. Основними компонентами, що використовуються для реалізації квантового ділення, є квантові вентилі та квантові регістри. Цей алгоритм також використовується у алгоритмі Шора.

Основні кроки для реалізації операції ділення:

1) Підготовка квантових регістрів.

2) Знаходження оберненого до дільника.

3) Множення цього оберненого на ділене.

4) Зчитування результату.

Це є найскладнішою арифметичною операцію через технічні обмеження квантового комп'ютера.

1.3 Методологія оцінки складності

Важливою частиною розробки алгоритму є визначення його ефективності. Скільки ресурсів він займає та чи є він кращим за його аналогом? Щоб відповісти на ці запитання, важливо оцінити його складність. Саме ці оцінки показують як алгоритми використовують ресурси.

Означення 1.10. *Ефективність* – швидкість знаходження відповіді та витрата невеликого обсягу ресурсів.

Складність алгоритмів у класичній моделі

Для класичного комп'ютера складність описується такими характеристиками як часова (T) та просторова (S) складність. Ці обидві функції на вхід приймають розмір вхідних даних.

Означення 1.11. *Часова складність* – оцінка, яка характеризує час, необхідний для виконання алгоритму на певній машині. Цей час, як правило, визначається кількістю операцій, які треба виконати для реалізації алгоритму.

Означення 1.12. *Просторова складність* – оцінка, яка характеризує об'єм пам'яті, необхідний для виконання алгоритму.

Тим самим важливо розібратися з алгоритмами, складність яких ми обчислюємо. Вони, в свою чергу, поділяються на *детерміновані* (на вхід подаються лише аргументи функцій) та *імовірнісні* (крім аргументів

функцій на вхід подаються ще деякі випадкові послідовності, які впливають на вихід). Також алгоритми поділяються на класи складності [**label1**]:

– *Поліноміальний, P* – клас складності, в якому алгоритми обчислюються за поліноміальний час.

– *Недетермінований поліноміальний, NP* – коли алгоритми визначають за поліноміальний час чи є випадково обраний розв'язок вірним.

– *BPP* – це клас, до якого належать імовірнісні алгоритми, які за поліноміальний час дають правильну відповідь з деякою імовірністю.

– *Просторові, PSPACE* – клас, який не обмежується часом та який потребує поліноміального простору (пам'яті і числа логічних операцій).

– *Експоненційні, EXPTIME* – клас, в якому алгоритми потребують експоненційного часу. Для великих вхідних даних вважається невирішуваною задачею для класичного комп'ютера.

Для оцінки часової складності використовують нотацію Ландау (O-нотація), яка описує зростання часу виконання алгоритму зі збільшенням розміру вхідних даних. Найпоширеніші функції: константа (1), логарифмічне ($logn$), лінійне $O(n)$, експоненціальне $O(2^n)$, факторіальне $O(n!)$, де $n \to \infty$.

Складність алгоритмів у квантовій моделі

Переходячи до квантової системи обчислень, ми знаємо, що деякі алгоритми виконуються значно ефективніше, швидше. Для відокремлення цих алгоритмів в теорії складності виділили новий клас BQP.

Означення 1.13. *BQP* – клас складності, який описує алгоритми, які можуть бути вирішені за поліноміальний час за допомогою квантових обчислень з обмеженою ймовірністю помилки.

Для визначення ефективності цих алгоритмів використовуються оцінки часової та просторової складності з класичної системи обчислень та інші оцінки, які визначають кількість витрачених ресурсів. Наприклад, кількість витрачених вентилів в алгоритмі.

Означення 1.14. *Вентильна складність* – оцінки мінімальної кількості квантових вентилів, необхідних для виконання певного квантового алгоритму.

Саме вентильна складність складається з оцінок кількості вентилів, застосованих до схеми та глибина цієї схеми. Для побудови оцінок складності треба буде знайти базис для цих алгоритмів.

Означення 1.15. *Базис квантових вентилів* – набір квантових вентилів, за допомогою яких ми можемо описати схему.

Найбільш застосовні є базиси, що складаються з вентилів групи Кліффорда та T-вентиля [**label4**]. Разом вони утворюють функціонально повний базис.

Означення 1.16. *Група Кліффорда* – група, яка містить вентилі Адамара(H), CNOT та S.

Група Кліффорда є замкненою щодо операції композиції та інверсії. Але є і обмеження цієї групи, вона не реалізує всі унітарні перетворення, для цього потрібно водити додатковий T-вентиль.

Означення 1.17. *T-вентиль ($\frac{\pi}{8}$ вентиль)* – вентиль, матричне представлення якого:

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

Коли цей вентиль застосовується до кубіта, він змінює фазу стану $|1\rangle$, залишаючи стан $|0\rangle$ без змін. T-вентиль дозволяє досягти вищої точності в квантових обчисленнях і є важливим для створення складних квантових алгоритмів.

Далі важливо розуміти за якими параметрами ми можемо оцінювати алгоритми в квантовій моделі обчислень:

– Кількість кубітів.

– Кількість вентилів.

– Глибина квантової схеми.

– Час виконання при реалізації алгоритму на квантовому комп'ютері.

Ці параметри можна визначити через алгоритм та його схеми.

Висновки до розділу 1

В цьому розділі були розглянуті основні теоретичні відомості про квантову систему обчислень, квантові алгоритми в ній та методологію теорії складності. Були відокремлені важливі визначення та поняття, з якими буде проведена робота в практичній частині цієї кваліфікаційної роботи. Основна ідея полягає у побудові оцінок цих алгоритмів та розглянути шляхи їх оптимізації.

2 АНАЛІЗ КВАНТОВИХ АЛГОРИТМІВ ДОДАВАННЯ ТА МНОЖЕННЯ

У цьому розділі представлена побудова оцінок складності арифметичних операцій в квантовій моделі обчислень та порівняння їх з класичною. Для цього розглянемо математичні моделі цих операцій, їхню складність, можливі обмеження та оптимальні підходи для їх реалізації.

Для побудови оцінок складності будуть використані наступні методи:

- Аналіз кількості квантових вентилів;
- Аналіз кількості кубітів;
- Глибина квантової схеми.

Всі ці методи дадуть повне уявлення про оцінку складності алгоритму в квантовій моделі обчислень.

2.1 Квантове додавання

Квантове додавання — одна з основних операцій у квантовому обчисленні. Ця операція полягає в додаванні кубітів, яке математично можна представити так:

$$|a\rangle + |b\rangle = |a + b\rangle \,,$$

або при оптимізації, щоб зменшити кількість кубітів, можна не створювати окремий квантовий регістр для вихідного значення, а записувати його замість одного з операндів:

$$|a\rangle + |b\rangle = |a, a + b\rangle \,,$$

Хоча операція додавання є доволі простою, вона має багато підходів до реалізацій, які застосовуються до різних задач та мають свої переваги

та недоліки. Один з перших алгоритмів – "Квантове додавання з переносом за методом Ripple-Carry"(англ. "A new quantum ripple-carry addition circuit")[1]. Це є аналогом класичного додавання в стовпчик з переносом.

Для реалізації цієї операції використовуються квантові вентилі (CNOT, Тоффолі) та регістри. В цій схемі біт переносу передається від одного розряду до наступного.

Algorithm 2.1 Квантове додавання з переносом за методом Ripple-Carry

Підготовка квантових реєстрів для чисел A та B:
for $\forall a_i$,$b_i \in A, B$ **do**
    $a_i = |a_i\rangle, b_i = |b_i\rangle$
end for
Ініціалізація бітів переносу:
Створюємо квантовий регістр для бітів переносу, довжина якого n+1 бітів $c = c_n...c_0$.
Виконання квантового додавання з переносом:
for $i \in 0,..,n-1$ **do**
    Додавання без переносу:
    Використовуючи Тоффолі-гейт: $a_i + b_i$
    Передача переносу:
    c_i - керуючий кубіт, $_{i+1}$ - контрольний кубіт. За допомогою вентиля CNOT передаємо біт переносу поточного розряду до наступного

    Результат цього додавання та біт переносу записуються відповідно до наступного розряду.
end for
Отримання результату:
Останній розряд є остаточним результатом додавання.
Зчитуємо квантові стани останніх розрядів, що представляють A+B.

Цей алгоритм є послідовним, що є аналогом з класичної схеми обчислень. Очікується, що цей алгоритм буде ефективнішим за рахунок часу та простору. Так як тут не використовується паралелізм, то за кількістю ресурсів він може бути рівним за складністю класичному аналогу або навіть більшим.

Розглянемо інший алгоритм квантового додавання "Квантове додавання з використанням адитивного квантового паралелізму який використовує паралелізм і за рахунок цього може бути значно ефективніший.

Algorithm 2.2 Квантове додавання з використанням адитивного квантового паралелізму

Підготовка квантових реєстрів для чисел A та B:
for $\forall a_i$,$b_i \in A, B$ **do**
    $a_i = |a_i\rangle, b_i = |b_i\rangle$
end for
Ініціалізація бітів переносу:
Створюємо квантовий регістр для бітів переносу, довжина якого n+1 бітів $c = c_n...c_0$.
Виконання квантового додавання з використанням адитивного квантового паралелізму:
for $i \in 0,..,n-1$ **do**
    Створення суперпозиції комбінацій бітів переносу:
    Створюємо суперпозицію всіх можливих комбінацій бітів переносу до позиції i.
    Передача переносу:
    Застосовуємо вентиль CNOT, який передає біт переносу з i-го розряду до $(i+1)$-го розряду.
end for
Отримання результату:
Зчитуємо квантові стани останніх розрядів, що представляють $A + B$.

Також існує алгоритм Драпера, представлений у 2000 році [2]. Він є дуже популярним та рунтується на квантовому перетворенні Фур'є і зворотному перетворенні Фур'є, яке потрібно для переведення числа у фазовий простір, в якому реалізація додавання стає простішою.

Algorithm 2.3 Квантове додавання Драпер

Підготовка квантових реєстрів для чисел A та B:

for $\forall a_i, b_i \in A, B$ **do**

    $a_i = |a_i\rangle, b_i = |b_i\rangle$

end for

Застосування QFT

$|a\rangle \rightarrow \text{QFT}(|a\rangle)$. Тобто переводимо число a у фазовий простір за допомогою квантового перетворення Фур'є

Додавання числа b:

for $|b_i\rangle \in 0, \ldots, n-1$ **do**

    застосовуємо фазові зміщення $e^{\frac{2\pi ibk}{2^n}} |k\rangle$

end for

Зворотній QFT (IQFT):

$QFT^{-1}\left(\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i(a+b)k}{2^n}} |k\rangle\right) = |a+b\rangle$

Цей алгоритм використовує заплутаність та суперпозицію для досягнення паралелізму, що призводить до значного прискорення роботи алгоритму.

Квантове додавання з переносом за методом Ripple-Carry.

Щоб зрозуміти наскільки ефективний алгоритм, треба знати скільки ресурсів для його реалізації потрібно. Так як алгоритм складається з логічних операцій (квантових ґейтів), то обчислимо скільки їх потрібно в залежності від вхідних даних.

Обчислимо кількість ґейтів на кожному кроці:

• Підготовка квантових реєстрів для чисел A та B: Кожен біт у кожному реєстрі потребує одного гейту для підготовки, отже, всього $2n$ ґейтів.

• Ініціалізація бітів переносу: Створення регістру бітів переносу потребує $n + 1$ ґейтів.

• Виконання квантового додавання: У кожному кроці i від 0 до $n-1$, ми використовуємо $2n - 1$ вентиль Тоффолі для знаходження переносу та $4n - 3$ вентиль CNOT, який обчислює суму (аналог операції XOR) з i-го розряду до $(i + 1)$-го розряду. Отже, $2n$ ґейтів.

Загальна кількість квантових ґейтів: $2n + 2n - 1 + 4n - 3 = 8n - 4 = O(n)$.

Цей метод за цими оцінками не є значно ефективнішим за класичний, так як в ньому не використовується основна перевага квантових обчислень - паралелізм.

Квантове додавання з використанням адитивного квантового паралелізму.

Кожен біт переносу передається паралельно до наступного розряду. Оскільки ми використовуємо суперпозицію для обчислення різних комбінацій бітів переносу одночасно, кількість квантових ґейтів буде значно меншою порівняно з послідовним методом Ripple-Carry.

Для аналізу кількості ґейтів у цьому алгоритмі, ми можемо розглянути кожен крок:

• Підготовка квантових реєстрів для чисел A та B: Кожен біт у кожному реєстрі потребує одного гейту для підготовки, отже, всього $2n$ ґейтів.

• Ініціалізація бітів переносу: Створення регістру бітів переносу потребує $n + 1$ ґейтів.

• Виконання квантового додавання з використанням адитивного квантового паралелізму: У кожному кроці i від 0 до $n - 1$, ми використовуємо один вентиль CNOT, який передає біт переносу з i-го розряду до $(i + 1)$-го розряду. Отже, це буде n вентилів CNOT.

• Отримання результату: Зчитуємо квантові стани останніх розрядів, що представляють A+B.

Загальна кількість квантових ґейтів у цьому алгоритмі буде $2n + (n + 1) + n = 4n + O(1)$ ґейтів.

Порівнюючи з послідовним методом Ripple-Carry, де було використано $2n - 1$ вентилів, ми бачимо, що "Квантове додавання з використанням адитивного квантового паралелізму"може потребувати більшої кількості гейтів. Однак враховуючи ефективність паралельного обчислення, цей метод може бути швидшим, особливо для великих чисел,

а також може мати інші переваги з точки зору обчислювальної складності.

Алгоритм Драпера

• В QFT використовується вентиль Адамара (H) та фазові зсуви. QFT ми застосовуємо тільки для числа a, яка складається з n кубітів. Тож Адамар застосовується до кожного з цих кубітів і це потребує n вентилів.

• Також в QFT є контрольовані фазові зсуви, тобто треба застосувати фазовий зсув для кожної пари кубітів. Всього таких зсувів: $1 + 2 + ... + (n-1) = \frac{n(n-1)}{2}$. Тобто кількість вентилів потрібних для QFT $n + \frac{n(n-1)}{2} = \frac{n^2-n+2n}{2} = \frac{n^2+n}{2} \approx O(n^2)$.

• Фазові зсуви для додавання числа b: для кожного кубіта числа b потрібно зробити фазові зсуви, а для кожного кубіта числа a потрібно n контрольованих фазових зсувів. Всього $O(n^2)$.

• IQFT має оцінку, як і QFT, $O(n^2)$.

Загальна кількість квантових вентилів у цьому алгоритмі буде $O(n^2) + O(n^2) + O(n^2) = 3O(n^2) \approx O(n^2)$ вентилів.Ця оцінка є значно кращою ніж у попередніх алгоритмів. Це досягається з забезпечення паралельності обчислень в цьому алгоритмі. Щоб мати уявлення про повну оцінку складності цих алгоритмів, розглянемо ще кількість кубітів.

Іншим параметром для побудови є саме аналіз кількості кубітів потрібних для реалізації певного алгоритму. Кубіти це як пам'ять, ми запитуємо порожні комірки для наших кубітів і дуже важливо щоб ця кількість була мінімальною.

Квантове додавання з переносом за методом Ripple-Carry

• На першому кроці алгоритму підготовлюємо 2 квантових регістра для чисел A та B довжиною n. Тобто $2n$ кубіта.

• На другому кроці ініціалізується квантовий регістр для бітів переносу, довжиною $n + 1$.

• Підготовка квантового регістру для бітів результату, довжиною n. Загальна кількість квантових кубітів: $2n+n+1+n = 4n+1$. Це є лінійною оцінкою, що є дуже непоганою. Але при великих вхідних даних кількість

кубітів буде стрімко зростати.

Квантове додавання з використанням адитивного квантового паралелізму

• На першому кроці алгоритму підготовлюємо 2 квантових регістра для чисел A та B довжиною n. Тобто $2n$ кубіта.

• На другому кроці ініціалізується квантовий регістр для бітів переносу, довжиною $n + 1$.

Загальна кількість квантових кубітів: $2n + n + 1 = 3n + 1$. Цей алгоритм вже більш ефективний за рахунок використання паралелізму.

Алгоритм Драпера

• На першому кроці алгоритму підготовлюємо 2 квантових регістра для чисел A та B довжиною n. Тобто $2n$ кубіта.

• Підготовка квантового регістру для бітів фазового зсува, довжиною n.

Загальна кількість квантових кубітів: $2n + n = 3n$. Ця оцінка є значно кращою за попередній алгоритм, адже вимагає менше кубітів. Хоч ця різниця невелика, але при великих вхідних даних вона швидко зростає.

Глибина квантової схеми визначається як максимальна кількість квантових вентилів, які повинні виконуватися послідовно для завершення алгоритму. Розглянемо алгоритми:

Квантове додавання з переносом за методом Ripple-Carry

Ми використовуємо вентилі в цьому алгоритмі в двух етапах: обчислення переносу та суми. Порахуємо кількість вентилів для одного біту на кожному етапі:

• Обчислення переносу: це послідовна операція, так як біт переносу c_i залежить від попередніх a_i, b_i, ci і обчислюється за допомогою вентиля Тоффолі. Для $\forall c_i \in c = c_n ... c_0$ використовується 3 Тоффолі послідовно та так як в нас n бітів переносу, то всього глибина досягає $3n$.

• Обчислення суми: це паралельна операція, після обчислення всіх бітів переносу. Поточний біт суми s_i залежить від a_i, b_i, c_i. Для цього обчислення потрібно 3 CNOT вентиля. Так як обчислення може

відбуватися паралельно, то глибина цього етапу 1.

Загальна глибина схеми: $3n + 1 \approx O(n)$.

Квантове додавання з використанням адитивного квантового паралелізму

• Створення суперпозиції комбінацій бітів переносу: застосування H до кожного біту і так як це відбувається паралельно, то глибина 1.

• Передача переносу: застосування CNOT до біту і це може знову ж таки виконуватися для кожного біту одночасно, тому глибина також 1.

Загальна глибина схеми: $O(1) + O(1) = O(1)$. Тобто складність цього алгоритму не залежить від розміру вхідних даних, що каже про ефективність алгоритму.

Алгоритм Драпера

В цьому алгоритмі вентилі потребуються на етапах: QFT, фазові зсуви для числа b та IQFT.

• QFT та IQFT мають однакову глибину: Для одного кубіту використовує n H та $\frac{n(n-1)}{2}$ фазових зсувів, що загалом дає глибину $n + \frac{n(n-1)}{2} = \frac{n^2+n}{2}$.

• Фазові зсуви для додавання числа b: для кожного біта числа b виконується n послідовних зсувів, тому глибина на цьому кроці: n.

Загальна глибина схеми: $\frac{n^2+n}{2} + \frac{n^2+n}{2} + n = n^2 + 2n \approx O(n^2)$.

Підсумуємо те, що було досліджено в попередніх підпунктах. Ми маємо 3 різних алгоритмів квантового додавання, для яких побудова оцінки складності спираючись на три параметри. Тепер за цими оцінками буде визначений кращий за ефективністю алгоритм. Всі оцінки виведено коротко у таблицю:

Таблиця 2.1 – Оцінки складності квантового додавання

	Ripple-Carry	**Адитивного квантовий паралелізм**	**Драпер**
Кіл.-ть вентилів	$O(n)$	$4n + O(1)$	$O(n^2)$
Кіл.-ть кубітів	$4n + 1$	$3n + 1$	$3n$
Глибина схеми	$O(n)$	$O(1)$	$O(n^2)$

2.2 Квантове множення

Друга арифметична операція, яка буде розглянута в цій роботі. Математично можна представити так:

$$|a\rangle * |b\rangle = |a + b\rangle$$

Відомо, що числа A та B n-бітові, тому при підготовці даних треба буде створити два n-бітових реєстри та один $2n$-бітовий реєстр для запису результату. Кількість кубітів вже досягає $4n$, тому можемо оптимізувати це:

$$|a\rangle * |b\rangle = |a, a + b\rangle$$

Тепер не треба створювати додатковий реєстр для результату, а можемо записувати результат в квантовий реєстр числа , який ми створимо довжини $2n$. При такому представленні кількість кубітів буде зменшена до $3n$.

Існує багато реалізацій та оптимізацій цього алгоритму, але більш цікавим є алгоритм квантового множення з використанням QFT та IQFT[3]. Він є потужним інструментом для ефективного виконання операцій множення в квантовій моделі обчислень. Далі розглядається даний алгоритм та його аналіз для побудови оцінок складності. Кількість кубітів, вентилів та глибина квантової схеми визначається аналогічним шляхом до додавання.

Algorithm 2.4 Квантове множення

Підготовка квантових реєстрів для чисел A та B:
for $\forall a_i$,$b_i \in A, B$ **do**
    $a_i = |a_i\rangle, b_i = |b_i\rangle$
end for
Ініціалізація квантового реєстру для результату
$rez = c_{2n-1}$,c_{2n-2},...,c_1,c_0
Застосування QFT
$|a\rangle \rightarrow$ QFT($|a\rangle$) та $|b\rangle \rightarrow$ QFT($|b\rangle$)
Квантове множення в фазовому просторі:
for $|c_i\rangle \in 0,..,n-1$, $|a_i\rangle \in 0,..,n-1$ та $|b_i\rangle \in 0,..,n-1$ **do**
    $c_i = a_i * b_i$
end for
Зворотній QFT (IQFT):
$QFT^{-1}(c) = |a * b\rangle$

Загалом в цьому алгоритмі кубіти використовуються для підготовки квантових реєстрів для чисел A та B та ініціалізація квантового реєстру результату.

• Квантові реєстри чисел A та B: $n + n = 2n$ кубітів.

• Ініціалізація квантового реєстру для бітів результату: $2n$ кубітів.

Загальна кількість квантових кубітів: $2n + 2n = 4n$. Але при оптимізації, якщо ми не будемо створювати реєстр для бітів результатів, а одразу запишемо його до реєстру B, для якого створимо $2n$ комірок. В цьому випадку, загальна кількість квантових кубітів буде дорівнювати $3n$. Ці дві оцінки є лінійними та дуже близькими одна до одної, але при великих вхідних даних різниця між ними буде значно зростати. Тому саме оптимізація на такому етапі є дуже важливою.

В даному алгоритмі використовуються QFT, IQFT та вентиль множення у фазовому просторі. Проаналізуємо їх кількість:

• Складність QFT та IQFT визначена раніше в алгоритмі додавання. Кожна з цих операцій має складність приблизно $O(n^2)$.

• Кожен біт числа A множиться на кожен біт числа B. Складність

приблизно $O(n^2)$.

Загальна кількість вентилів: $O(n^2) + O(n^2) = 2O(n^2) \approx O(n^2)$.

Для кожного кубіту послідовно виконуються такі дії: QFT, множення, IQFT. Всі ці 3 дії можуть виконуватися паралельно.

• Складність QFT та IQFT визначена раніше в алгоритмі додавання. Кожна з цих операцій має складність приблизно $O(\frac{n^2+n}{2})$.

• Кожен біт числа A множиться на кожен біт числа B. Складність приблизно $O(1)$.

Загальна глибина квантової схеми: $O(\frac{n^2+n}{2} + 1) \approx O(n^2)$.

Проаналізувавши параметри складності алгоритму, видно, що є лінійна оцінка та дві квадратичні. Це каже про те, що алгоритм достатньо ефективний та може працювати з великими вхідними даними.

Таблиця 2.2 – Оцінки складності квантового множення

	Квантове множення
Кіл.-ть вентилів	$O(n^2)$
Кіл.-ть кубітів	$3n$
Глибина схеми	$O(n^2)$

Висновки до розділу 2

3 ПОБУДОВА ЕФЕКТИВНИХ КВАНТОВИХ АЛГОРИТМІВ ДЕЯКИХ СПЕЦІАЛЬНИХ АРИФМЕТИЧНИХ ОПЕРАЦІЙ

В даному розділі дослідження

3.1 Редукція за спеціальним модулем вигляду $2^n \pm 1$

В класичній модульній арифметиці редукція використовується для знаходження остачі від ділення великого числа на модуль N. Редукція за спеціальними модулями фокусується на оптимізації модульної арифметики, коли модуль має певну структуру, часто таку, що дозволяє ефективніші обчислення. Цей метод особливо корисний у криптографічних алгоритмах та хеш-функціях, де такі редукції часто потрібні.

Основна ідея редукції за спеціальними модулями полягає у використанні специфічної форми N для спрощення операцій, пов'язаних із редукцією числа t за модулем N. Загальний підхід включає використання модулів виду $2^n \pm c$, де n - ціле число, а c – маленька константа. Ці форми дозволяють використовувати побітові операції, які менш затратні за обчислювальними ресурсами. В цій роботі буде розглянутий випадок, при якому вид модуля $2^n \pm 1$, де n - ціле число.

В основу цього методу використовується Lookup-Table Reduction, який реалізується через додавання та віднімання. Цей метод дуже ефективний для деяких спеціальних модулів.

За модулем m та x, представлений бітовою послідовністю довжини p [5]: $x = \sum_{i=0}^{p-1} x_i 2^i$, маємо:

$$x \bmod m = \left[\sum_{i=0}^{p-1} (x_i 2^i \bmod m) \right] \bmod m \tag{3.1}$$

Нехай $p \mid j$ (якщо ні, то доповнимо x до потрібної кількості), тоді ми можемо розбити x на j блоків по k біт кожен:

$$x_{j-1} = x_{kj-1}x_{kj-2}...x_{k(j-1)}...x_1 = x_{2j-1}x_{2j-2}...x_kx_0 = x_{k-1}x_{k-2}...x_{k(0)}$$

Тоді,

$$\begin{aligned} x \mod m = \big(&x_{j-1}2^{k(j-1)} \bmod m + x_{j-2}2^{k(j-2)} \bmod m + \dots \\ &+ x_1 2^k \bmod m + x_0 2^0 \bmod m\big) \bmod m \end{aligned} \tag{3.2}$$

Тобто потрібно лише обчислити $x_i 2^{ik} \bmod m (i = 0, 1, ..., j-1)$ та додати їх за модулем m.

Квантова редукція за модулем $2^n - 1$

За цим модулем:

$$2^n \bmod (2^n - 1) = [(2^n - 1) + 1] \bmod (2^n - 1) = 1 \tag{3.3}$$

З цього маємо:

$$2^{in} \bmod (2^n - 1) = [2^n \bmod (2^n - 1)]^i \bmod (2^n - 1) = 1^i \bmod (2^n - 1) = 1$$

Тому,

$$(x_i 2^{jn}) \bmod (2^n - 1) = x_i \bmod (2^n - 1)$$

Тоді 3.2 набуде вигляду:

$$\begin{aligned} x \bmod (2^n - 1) = \Big[& x_{j-1} 2^{n(j-1)} \bmod (2^n - 1) + x_{j-2} 2^{n(j-2)} \bmod (2^n - 1) + \dots \\ & + x_1 2^n \bmod (2^n - 1) + x_0 \bmod (2^n - 1)] \bmod (2^n - 1) \\ = & [x_{j-1} \bmod (2^n - 1) + x_{j-2} \bmod (2^n - 1) + \dots \\ & + x_1 \bmod (2^n - 1) + x_0 \bmod (2^n - 1)] \bmod (2^n - 1) \\ = & (x_{j-1} + x_{j-2} + \dots + x_0) \bmod (2^n - 1) \end{aligned} \tag{3.4}$$

Використовуючи виведені формули, можна створити вантовий алгоритм для обчислення зниження модуля $2^n - 1$ з використанням властивостей цього модуля.

Algorithm 3.1 Квантовий алгоритм редукції за модулем $2^n - 1$

1: **Вхідні дані:**

x - квантовий регістр з p кубітами, де $p = kj$ (припускаємо, що $p \mid j$).

2: **Розбиття на блоки:**

3: Розбиваємо квантовий регістр x на блоки по k кубітів: x_{j-1},x_{j-2},...,x_0, де $j = p/k$

4: **Виділення блоків:**

5: Використовуємо квантові вентилі для створення окремих квантових регістрів для кожного блоку x_i.

6: **Додавання блоків:**

7: Виконуємо додавання всіх блоків у допоміжний регістр довжиною n кубітів, оскільки кожен блок буде представлений як $x_i \bmod (2^n - 1)$

Детальний опис кроків алгоритму із зазначенням потрібних ресурсів

1) Ініціалізація квантових регістрів:

– Ініціалізуємо квантовий регістр для вхідного значення x довжиною p кубітів;

– регістри для j блоків: $\forall x_i$ створюємо квантовий регістр довжиною k кубітів.

2) Розбиття на блоки:

– Для перенесення блоків з квантового регістру x до x_i використовуємо вентиль CNOT. На першому кроці ми ініціалізуємо x_i як нульові регістри та контрольний кубіт буде з регістру x, а цільовий x_i.

3) Додавання:

– Для додавання блоків використовується квантове додавання за модулем.

Твердження 3.1. *У квантовому алгоритмі редукції за модулем* $2^n - 1$ *оцінка кількості кубітів є експоненційною:* $O(2^n)$*, де* p *– кількість кубітів вхідного числа,* j *– кількість підпослідовностей* x*.*

Доведення. Для оцінки кількості кубітів потрібно розглянуто скільки їх виділяється на кожному кроці для роботи алгоритму.

– Ініціалізація вхідного квантового регістру: p кубітів;

– Ініціалізація квантових регістрів для j блоків по k кубітів кожен: $k \cdot j$ кубітів.

– Операція додавання при використанні RCA потребує $k + 1$ додаткових кубітів.

Загалом: $p + k \cdot j + k + 1 = p + p + k + 1 = 2p + k + 1$. □

Твердження 3.2. *У квантовому алгоритмі редукції за модулем* $2^n - 1$ *оцінка кількості вентилів є лінійною:* p *вентилів CNOT та* $20k - 10$ *Тоффолі, де* p *– кількість кубітів вхідного числа,* k *– кількість кубіт в кожному блоці* j*.*

Доведення. Для оцінки кількості вентилів потрібно розглянуто скільки їх виділяється на кожному кроці для роботи алгоритму.

– Для створення окремих блоків використовуються CNOT вентилі. Для переносу одного блоку довжини k потрібно рівно стільки же вентилів. Враховуючи, що в нас j блоків, то кількість потрібних CNOT: $k \cdot j = p$;

– Додавання за модулем 2^n-1 на основі алгоритму за методом Ripple-Carry має оцінку кількості вентилів [4]: $20k - 10$ Тоффолі.

Загалом: p вентилів CNOT та $20k - 10$ Тоффолі. □

Таблиця 3.1 – Оцінки складності квантової редукції за модулем $2^n - 1$

	Квантове множення
Кіл.-ть вентилів	$p + 20k - 10$
Кіл.-ть кубітів	$2p + k + 1$

Квантова редукція за модулем $2^n + 1$

Розглянемо наступний спеціальний модуль вигляду 2^n+1. З [5] маємо:

$$x \bmod (2^n + 1) = [(2^n + 1) - 1]] \bmod (2^n + 1) = -1 \tag{3.5}$$

$$2^{in} \bmod (2^n + 1) = [2^n \bmod (2^n + 1)]^i \bmod (2^n + 1) =$$

$$= (-1)^i \bmod (2^n + 1) = \begin{cases} 1, & \text{якщо } i \text{ - парне} \\ -1, & \text{інакше} \end{cases}$$

З цього випливає два випадки обчислення редукції: при парному j та непарному. Розглянемо їх нижче:

1. Якщо $j \neq 0 \pmod 2$

$$x \bmod (2^n + 1) = \Big[x_{j-1}2^{n(j-1)} \bmod (2^n + 1) - x_{j-2}2^{n(j-2)} \bmod (2^n + 1) + \ldots -$$
$$-x_1 2^n \bmod (2^n + 1) + x_0 \bmod (2^n + 1)] \bmod (2^n - 1) =$$
$$= [x_{j-1} \bmod (2^n + 1) - x_{j-2} \bmod (2^n + 1) + \ldots -$$
$$-x_1 \bmod (2^n + 1) + x_0 \bmod (2^n + 1)] \bmod (2^n - 1) =$$
$$= (x_{j-1} - x_{j-2} + \cdots - x_1 + x_0) \bmod (2^n - 1)$$

2. Якщо $j = 0 \pmod 2$

$$\begin{aligned} x \bmod (2^n+1) = \Big[& -x_{j-1}2^{n(j-1)} \bmod (2^n+1) + x_{j-2}2^{n(j-2)} \bmod (2^n+1) - \ldots + \\ & + x_1 2^n \bmod (2^n+1) - x_0 \bmod (2^n+1)\Big] \bmod (2^n-1) = \\ = & \left[-x_{j-1} \bmod (2^n+1) + x_{j-2} \bmod (2^n+1) - \ldots + \right. \\ & \left. + x_1 \bmod (2^n+1) - x_0 \bmod (2^n+1)\right] \bmod (2^n-1) = \\ = & (-x_{j-1} + x_{j-2} - \cdots + x_1 - x_0) \bmod (2^n-1) \end{aligned}$$

Тобто редукція за модулем $2^n + 1$ зводиться до операцій додавання та віднімання, що є значно швидкісними за множення та ділення, які використовуються у редукції за довільним модулем. Класичний алгоритм редукції за цим модулем полягає у розбитті числа у вигляді бітової послідовності на рівні підпослідовності (блоки). Далі в залежності від довжини блоку, виконуємо додавання/віднімання за модулем $2^n + 1$.

Побудуємо квантовий алгоритм редукції за спеціальним модулем $2^n + 1$.

Algorithm 3.2 Квантовий алгоритм редукції за модулем $2^n + 1$

Вхідні дані:

x - квантовий регістр з p кубітами, де p=kn (припускаємо, що $n \mid p$).

Ініціалізація квантових регістрів для j блоків:

Створюємо j нульових квантових регістрів довжини k

Розбиття на блоки:

Розбиваємо квантовий регістр x на блоки по k кубітів: x_{j-1},x_{j-2},...,x_0, де $j = p/k$

Перевірка чи $j = 0 \pmod 2$:

if $j = 0 \pmod 2$ **then**

    Виконуємо дії для парного j:

    Виконуємо віднімання-додавання за модулем $2^n - 1$ по черзі

else

    Виконуємо дії для непарного j:

    Виконуємо додавання-віднімання за модулем $2^n - 1$ по черзі

end if

Зауваження. Для визначення чи є число парним,

використовується властивість з класичної системи обчислень. Якщо останній кубіт закінчується на 0, то число парне. Якщо 1 - непарне. Для цієї перевірки доцільно буде використовувати CNOT вентиль. Створюємо додатковий кубіт c, де буде збережено результат перевірки. Тобто останній кубіт буде цільовий кубітом, а додатковий c - контрольним.

Для визначення ефективності цього алгоритму проаналізуємо його та побудуємо оцінки складності за кількістю кубітів та вентилів. Адже ці оцінки суттєво впливать на час виконання та кількість ресурсів для реалізації.

Нижче сформуємо твердження щодо оцінок складності даного алгоритму.

Твердження 3.3. *Кількість кубітів в квантовому алгоритмі редукції за модулем* $2^n + 1$ *дорівнює* $2p + k + 1$. *Отже кількість кубітів лінійно залежить від розміру вхідних даних.*

Доведення.

– Ініціалізація вхідного квантового регістру: p кубітів;

– Ініціалізація квантових регістрів для j блоків по k кубітів кожен: $k \cdot j$ кубітів;

– 1 кубіт для зберігання результату перевірки числа на парність;

– Операція додавання/віднімання при використанні RCA потребує $k + 1$ додаткових кубітів.

Загалом: $p + k \cdot j + 1 + k + 1 = p + p + k + 1 = 2p + k + 1$. □

Твердження 3.4. *Кількість вентилів в квантовому алгоритмі редукції за модулем* $2^n + 1$ *дорівнює* $20k^2 - 10k$. *Отже кількість вентилів квадратично залежить від кількості блоків.*

Доведення. Розглянемо кожний пункт алгоритму окремо:

– Розбиття на блоки. Потребує j CNOT вентилів;

– 1 CNOT для перевірки числа на парність;

– Для додавання/віднімання використовується RCA. Для одного блоку це потребує $20k - 10$ Тоффолі вентилів. Ця операція

використовується послідовно, тому для k блоків потрібно $20k^2 - 10k$ Тоффолі вентилів.

Загалом: $20k^2 - 10$ Тоффолі вентилів. □

Таблиця 3.2 – Оцінки складності квантової редукції за модулем $2^n + 1$

	квантова редукція за модулем $2^n + 1$
Кіл.-ть вентилів	$20k^2 - 10k$
Кіл.-ть кубітів	$2p + k + 1$

3.2 Квантовий алгоритм множення поліномів з коефіцієнтами $-1, 0, 1$

Множення поліномів використовується у багатьох областях криптографії, наприклад у еліптичних кривих чи у хеш-функціях для досягнення стійкості до колізії. Використання коефіцієнтів такого вигляду спрощує процес обчислення, дозволяючи виконувати алгоритми значно швидше. Тому є дуже важливим використання цих властивостей і у квантовій криптографії [**Hoof2019**]. Для цього побудуємо квантовий алгоритм множення поліномів з коефіцієнтами $-1, 0, 1$.

Нехай $A(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ та $B(x) = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0$ – поліноми з коефіцієнтами $-1, 0, 1$. Спробуємо використовувати принципи суперпозиції та паралелізму для досягнення більш ефективної роботи алгоритму.

Коефіцієнти поліномів можемо представити у вигляді бітової послідовності $|00\rangle, |10\rangle, |11\rangle$.

Algorithm 3.3 Квантовий алгоритм множення поліномів з коефіцієнтами $-1, 0, 1$

1: **Вхідні дані:**

$A(x)$ та $B(x)$ степеню n та m відповідно.

2: **Ініціалізація квантових регістрів для $A(x)$ та $B(x)$:**

3: Створюємо три регістри: $A(x) = |a_n\rangle, \ldots |a_1\rangle, |a_0\rangle$; $B(x) = |b_m\rangle, \ldots |b_1\rangle, |b_0\rangle$ та регістр результату $R = |r_{n+m}\rangle, \ldots, |r_1\rangle, |r_0\rangle$.

4: **Створення суперпозиції**

5: Кожен кубіт переводимо у стан суперпозиції за допомогою H.

6: **Множення поліномів:**

7: Перемножуємо кубіти коефіцієнтів за допомогою Тоффолі вентиля(a_i, b_j-цільові, c_j -контрольний). Для додавання зсувів використовуємо CNOT.

Даний алгоритм базується на принципах квантових обчислень, що дозволяють досягати значно швидших результатів у порівнянні з класичними методами. Поліноми представляються у вигляді бінарних рядків за допомогою кубітів. Спочатку всі кубіти переводяться в стан суперпозиції за допомогою вентилів Адамара, що дозволяє одночасно розглядати всі можливі стани кубітів. Далі, використовуючи вентиль Toffoli, ми здійснюємо множення кубітів між собою, а отримані результати додаються з урахуванням степеня x.

Квантове обчислення завершується вимірюванням стану кубітів, що дозволяє отримати кінцевий результат у класичній формі. Цей підхід значно прискорює процес множення поліномів, адже квантові комп'ютери здатні обробляти велику кількість операцій паралельно завдяки суперпозиції і заплутаності.

Висновки до розділу 3

У цьому розділі, було досліджено перспективи використання квантових обчислень в алгоритмах деяких арифметичних операцій. Наша робота демонструє потенційні можливості квантових алгоритмів у вирішенні задач, які зазвичай вимагають значних обчислювальних

потужностей у класичній моделі обчислень.

Перш за все, розроблено квантовий алгоритм множення поліномів з коефіцієнтами 0, 1, та -1. Цей алгоритм дозволяє здійснювати операції множення поліномів на квантовому комп'ютері швидше, ніж це можна зробити за допомогою класичних алгоритмів.

Друге, було розглянуто квантовий алгоритм редукції за модулем $2^n \pm 1$, який є важливим етапом у багатьох криптографічних протоколах та алгоритмах шифрування. Дослідження показали, що квантові обчислення можуть здійснювати цю операцію ефективніше за класичні методи, що відкриває перспективи для застосування квантових алгоритмів у криптографії.

ВИСНОВКИ

У ході даної роботи був проведений детальний аналіз існуючих методів і алгоритмів для виконання арифметичних операцій у квантовій моделі обчислень. Цей аналіз дозволяє глибше зрозуміти переваги та обмеження класичних методів у порівнянні з потенціалом квантових алгоритмів.

Оцінки складності існуючих алгоритмів додавання та множення також відіграють важливу роль у цій роботі. Ці оцінки допомогли виявити, які саме операції можуть бути оптимізовані за допомогою квантових обчислень, і які переваги це може принести у практичних застосуваннях.

Зокрема, розроблений квантовий алгоритм редукції за спеціальним модулем $2^n \pm 1$ та алгоритм множення поліномів з коефіцієнтами 0, 1 та -1 демонструють потенціал квантових обчислень для вирішення задач, які раніше вважалися складними або обмеженими для класичних методів. Так як ці алгоритми активно використовуються у різних областях математика, наприклад криптографії, то розробка та аналіз цих алгоритмів є важливим кроком в переході до квантової моделі обчислень. Побудовані оцінки складності показують ефективність цих алгоритмів, так як вони лінійно або квадратично залежать від вхідних даних, що є хорошим показником.

Таким чином, дана робота підтверджує перспективи використання квантових алгоритмів у різних областях алгебри та обчислювальної науки. Для подальших досліджень рекомендується зосередитися на вдосконаленні квантових алгоритмів з метою їх практичного застосування, а також на розробці нових методів оцінки ефективності цих алгоритмів.

ПЕРЕЛІК ПОСИЛАНЬ

[1] Steven A. Cuccaro та ін. *A new quantum ripple-carry addition circuit*. 2004. DOI: 10.48550/ARXIV.QUANT-PH/0410184. URL: https://arxiv.org/abs/quant-ph/0410184.

[2] Thomas G. Draper. *Addition on a Quantum Computer*. 2000. DOI: 10.48550/ARXIV.QUANT-PH/0008033. URL: https://arxiv.org/abs/quant-ph/0008033.

[3] G. Florio та D. Picca. *Quantum implementation of elementary arithmetic operations*. 2004. DOI: 10.48550/ARXIV.QUANT-PH/0403048. URL: https://arxiv.org/abs/quant-ph/0403048.

[4] Aeyoung Kim та ін. «Quantum Modular Adder over GF(2n 1) without Saving the Final Carry». В: *Applied Sciences* 11.7 (бер. 2021), с. 2949. ISSN: 2076-3417. DOI: 10.3390/app11072949. URL: http://dx.doi.org/10.3390/app11072949.

[5] Amos R. Omondi. *Cryptography Arithmetic: Algorithms and Hardware Architectures*. Springer International Publishing, 2020. ISBN: 9783030341428. DOI: 10.1007/978-3-030-34142-8. URL: http://dx.doi.org/10.1007/978-3-030-34142-8.