УДК 004.43 DOI https://doi.org/10.35546/kntu2078-4481.2025.2.2.40

Д. О. САПОЖНИК
аспірант кафедри інженерії програмного забезпечення
Державний університет «Житомирська політехніка»
ORCID: 0000-0003-1357-2422

ПРОЦЕС ПЕРЕТВОРЕННЯ ВИСОКОРІВНЕВИХ КВАНТОВИХ ВЕНТИЛІВ У БАЗОВІ ДЛЯ ВИКОНАННЯ НА РЕАЛЬНОМУ КВАНТОВОМУ КОМП'ЮТЕРІ

У статті здійснено системне дослідження процесу транспіляції квантових алгоритмів, що є обов'язковим етапом підготовки до виконання програм на реальних квантових пристроях. Розкрито сутність поняття базових вентилів (basis gates) як елементарного набору унітарних операцій, які підтримуються конкретним апаратним забезпеченням. Обґрунтовано, чому високорівневі квантові вентилі (такі як Hadamard, Toffoli, Fredkin) не можуть бути реалізовані безпосередньо на більшості сучасних квантових процесорів через апаратні обмеження та особливості архітектури, включно з топологією зв'язків між кубітами. У якості прикладу розглянуто платформу IBM Quantum, де базовий набір для надпровідникових процесорів типу Eagle включає вентилі RZ, SX, X та двокубітну операцію ECR (Echoed Cross Resonance). Проведено поетапний аналіз процесу транспіляції у середовищі Qiskit, що включає уніфікацію схеми, розклад високорівневих вентилів у базові, перетворення логічних кубітів на фізичні, врахування карти з'єднань, вставлення SWAP-вентилів, а також оптимізацію схеми з метою зменшення її глибини та кількості операцій. Пояснено, як транспілятор автоматично адаптує логічну квантову схему до обмежень реального пристрою за допомогою вбудованих бібліотек розкладу вентилів та оптимізаційних проходів (зокрема, алгоритму SABRE). Наведено приклад транспіляції схеми з вентилями H, CX та T, де показано перехід до повністю сумісної з апаратним рівнем реалізації. Визначено переваги використання різних рівнів оптимізації (0–3) транспілятора Qiskit, розглянуто вплив оптимізації на точність результатів та стійкість до шуму. Особливу увагу приділено питанням ефективного програмування з урахуванням апаратних обмежень: зменшенню кількості багатокубітних вентилів, раціональному використанню допоміжних кубітів, обмеженню використання бар'єрів та доцільності ручного коригування відповідностей кубітів у схемі.

Ключові слова: *квантове програмування, квантові технології, транспіляція, квантові вентилі, базові вентилі, Qiskit, IBM Quantum, квантова компіляція, оптимізація квантових схем.*

D. O. SAPOZHNYK
Postgraduate Student at the Department of Software Engineering
Zhytomyr Polytechnic State University
ORCID: 0000-0003-1357-2422

THE PROCESS OF TRANSFORMING HIGH-LEVEL QUANTUM GATES INTO BASIS GATES FOR EXECUTION ON A REAL QUANTUM COMPUTER

The article provides a systematic study of the process of transpilation of quantum algorithms, which is a mandatory stage of preparation for the execution of programs on real quantum devices. The essence of the concept of basic gates is disclosed as an elementary set of unitary operations that are supported by specific hardware. It is substantiated why high-level quantum gates (such as Hadamard, Toffoli, Fredkin) cannot be implemented directly on most modern quantum processors due to hardware limitations and architectural features, including the topology of connections between qubits. As an example, the IBM Quantum platform is considered, where the basic set for superconducting processors of the Eagle type includes RZ, SX, X gates and a two-qubit ECR (Echoed Cross Resonance) operation. A step-by-step analysis of the process of transpilation in the Qiskit environment has been carried out, which includes unification of the circuit, decomposition of high-level valves into basic ones, transformation of logical qubits into physical ones, consideration of the connection map, insertion of SWAP-valves, as well as optimization of the circuit to reduce its depth and the number of operations. It explains how the transpiler automatically adapts the logic quantum circuit to the limitations of the real device using built-in libraries of valve schedules and optimization passes (in particular, the SABRE algorithm). An example of transpilation of the circuit with valves H, CX and T is given, which shows the transition to a fully hardware-compatible implementation. The advantages of using different levels of optimization (0–3) of the Qiskit transpiler are determined, the influence of optimization on the accuracy of results and noise resistance is considered. Particular attention is paid to the issues of effective programming, considering hardware limitations: reducing the number of multi-qubit valves, rational use of auxiliary qubits, limiting the use of barriers and the feasibility of manually adjusting the correspondences of qubits in the scheme.

Key words: *quantum programming, quantum technologies, transpilation, quantum gates, basis gates, Qiskit, IBM Quantum, quantum compilation, quantum circuit optimization.*

Постановка проблеми

Базові вентилі апаратного забезпечення – це набір квантових логічних операцій, які фізично реалізовані на конкретному квантовому процесорі [1]. Інакше кажучи, ці вентилі являють собою базові квантові операції, реалізація яких забезпечується безпосередньо на фізичному (апаратному) рівні квантових обчислювальних систем. Кожен виробник квантових комп'ютерів має свій набір базових вентилів залежно від технології: наприклад, в іонних пастках можуть бути базовими глобальні Mølmer–Sørensen вентилі [2], а в надпровідникових кубітах IBM – інший набір. IBM Quantum для різних поколінь процесорів визначає власні базові вентилі, для процесорів сімейства Eagle (надпровідниковий чип IBM на 127 кубітів) базовими є однокубітні вентилі *RZ* (поворот навколо осі *Z*), *SX* (квадратний корінь із *X*), *X* (Паулі-X), а також двокубітний вентиль *ECR* (*echoed cross-resonance*) [3]. У процесорах сімейства Falcon базовим двокубітним вентилем є *CX* (контрольований *X*) [4]. Всі складніші операції на реальному квантовому пристрої повинні бути розкладені на послідовність цих базових вентилів перед виконанням.

За допомогою високорівневих вентилів у квантових алгоритмах проводяться дії над абстрактними логічними операціями, незалежними від апаратури, наприклад: вентиль Адамара (Hadamard), Toffoli (вентиль з двома контрольними кубітами), *Fredkin* (контрольований обмін), фази *T* та *S* та інші. Реальні квантові комп'ютери фізично не здатні виконувати більшість таких вентилів напряму, оскільки апаратно обмежені до свого базового набору [5]. Причини цього пов'язані з особливостями фізичної реалізації кубітів та їх взаємодій. Наприклад, у надпровідникових квантових процесорах керування здійснюється за допомогою мікрохвильових імпульсів, що дозволяє реалізувати лише певні типи одно- та двокубітних унітарних операцій (обертання навколо визначених осей і спеціально відкаліброваних двокубітні взаємодії). Інші типи вентилів мають бути синтезовані на основі зазначених доступних базових операцій.

Крім того, існують архітектурні обмеження, пов'язані з тим, що безпосередня взаємодія можлива не між усіма парами кубітів на квантовому чипі. Внаслідок цього складні багатокубітні операції (наприклад, трикубітні вентилі *Toffoli* або *Fredkin*), а також двокубітні операції типу *CNOT* між кубітами, що фізично знаходяться на значній відстані, необхідно розкладати на серію елементарних вентилів, сумісних із топологією конкретного квантового пристрою. Наприклад, вентиль *Toffoli* не є фундаментальним для більшості сучасних квантових процесорів, і для його реалізації зазвичай використовують комбінацію кількох двокубітних *CNOT*-операцій і однокубітних унітарних поворотів. Аналогічним чином, вентиль *Fredkin* [6] (контрольований *SWAP*) також вимагає розкладання на базові двокубітні операції. Відповідно, компіляція квантових програм до набору фізично реалізованих елементарних вентилів є обов'язковим етапом для виконання алгоритму на конкретному апаратному забезпеченні, оскільки без такого перетворення високорівневі описи алгоритмів не можуть бути безпосередньо втілені у фізичних квантових процесорах. Саме тому процес приведення логічних квантових вентилів до базових важливий для розуміння та роботи з квантовими комп'ютерами.

Аналіз останніх досліджень і публікацій

Процес перетворення схеми із довільних вентилів у еквівалентну схему з базових вентилів називається **транспіляцією** [5]. Транспілятор приймає на вхід квантовий ланцюг (сукупність вентилів над кубітами) і проходить кілька етапів (рисунок 1), кожен з яких адаптує ланцюг до вимог цільового квантового пристрою.

Крок 1 – уніфікація опису. Всі нестандартні та складні операції у схемі уніфікуються, якщо користувач визначив вентиль не заявлений у наборі високорівневих квантових гейтів транспілятор замінить його еквівалентною підмножиною стандартних вентилів (наприклад, у Qiskit – це універсальний набір *U3*, *CX* тощо) [5]. Так схема приводиться до базового набору «віртуальної машини» – проміжного універсального представлення.

Крок 2 – розклад на базові вентилі. Кожен високорівневий вентиль зі списку, що не підтримується пристроєм, замінюється послідовністю доступних вентилів [1]. Наприклад, вентиль Адамара *H* буде розкладено на комбінацію поворотів *RZ* та *SX* (оскільки ні *H*, ні *Y* не є базовими на чипах IBM Eagle), а вентиль T (фазовий зсув на *pi*/4) – еквівалентний просто *RZ*(*pi*/4) і може напряму виконуватися через віртуальний *Z*-поворот [3]. Багатокубітні вентилі розкладаються рекурсивно: наприклад, Toffoli спершу розкладається на елементи з CNOT та одно кубітних *T*, *H* та *T-dagger* (цей стандартний розклад займає 8 елементарних вентилів, з них 6 – це *CNOT*). Кожен двокубітний *CNOT* своєю чергою може бути реалізований через базову двокубітну взаємодію цільового пристрою (для IBM Eagle це ECR). Таким чином, після цього кроку всі вентилі зі схеми належать до дозволеного набору цільового обладнання [7].

Крок 3 – Врахування зв'язностей. Транспілятор перевіряє, які кубіти у пристрої залучені та чи потрібна перестановка кубітів для врахування геометрії з'єднань. Якщо у схемі є двокубітний вентиль між кубітами, які фізично не з'єднані, алгоритм вставляє додаткові вентилі *SWAP* (розмін кубітів місцями) для маршрутизації взаємодії через сусідні зв'язки мережі кубітів [3, 5]. Транспілятор Qiskit використовує для цього спеціальні алгоритми, зокрема *heuristic Sabre*, які намагаються мінімізувати кількість *SWAP*, адже кожен *SWAP* додає три *CNOT* і збільшує глибину схеми [8]. На цьому етапі також відбувається вибір розкладки кубітів – тобто призначення логічних кубітів програми до конкретних фізичних кубітів процесора для оптимальної реалізації.

Крок 4 – Оптимізація схем. Отримавши еквівалентну схему з допустимих вентилів, транспілятор застосовує серію оптимізаційних проходів. На цьому етапі аналізуються послідовності вентилів з метою спростити або

Рис. 1. Схема транспіляції

скоротити їх. Наприклад, послідовність із *X* за яким слідує ще один *X* на тому ж кубіті еквівалентна тотожності та обидва вентилі можуть бути видалені. Так само *T* і *T-dagger* (обернені) поруч дають нульовий зсув фази й скорочуються. Транспілятор знаходить такі пари інверсних операцій і взаємопоглинаючі послідовності та видаляє їх [5]. Окрім вилучення, суміжні однокубітні повороти об'єднуються в один еквівалентний поворот, щоб не витрачати зайві базові вентилі [5]. Аналізуючи перестановку вентилів, можна іноді змінити їх порядок, щоб більше сусідніх операцій скоротились. Такі логічні оптимізації зменшують загальну кількість вентилів і глибину квантового ланцюга. На глибину також впливають вставлені *SWAP*: транспілятор намагається за можливості їх теж скоротити або переставити ближче до кінця, якщо ті не впливають на результати вимірювань.

Крок 5 – Формування вихідної програми. Після всіх перетворень отриманий квантовий ланцюг складається виключно з вентилів, які підтримує цільовий квантовий процесор, і враховує його топологію з'єднань [7]. Цей ланцюг можна виконати на реальному пристрої (або еквівалентно на емуляторі з відповідною шумовою моделлю). Зазвичай транспілятор повертає програму у вигляді OpenQASM-коду або об'єкта QuantumCircuit, готового до запуску.

Формулювання мети дослідження

Метою даного дослідження є формалізація та систематизація процесу перетворення високорівневих квантових вентилів у базові вентилі для забезпечення сумісності квантових алгоритмів із фізичними обмеженнями сучасних квантових обчислювальних платформ.

Викладення основного матеріалу дослідження

Розглянемо простий квантовий ланцюг із двох кубітів, на рисунку 2 можна побачити, що на кубіт *q0* застосовується вентиль Адамара *H* (високорівнева однокубітна операція, яка не є базовою для IBM Eagle), між *q0* (контроль) і *q1* (ціль) застосовано вентиль *CNOT*, а на кубіт *q0* в кінці ставиться вентиль *T* (фазове зміщення на *pi*/4).

```
1   from qiskit import QuantumCircuit, transpile
2
3   # Створення квантового ланцюга з 2 кубітів
4   qc = QuantumCircuit(2)
5   qc.h(0)        # вентиль Адамара на кубіті 0
6   qc.cx(0, 1)    # CNOT з контролем 0 і ціллю 1
7   qc.t(0)        # T-вентиль на кубіті 0
8   print("Оригінальний ланцюг:")
9   print(qc)
```

Рис. 2. Двокубітний квантовий ланцюг з використанням Qiskit

Функція print, у свою чергу, покаже схему з високорівневими вентилями, яку можна побачити на рисунку 3.

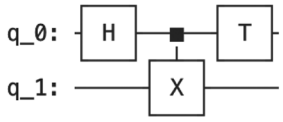

Рис. 3. Двокубітна квантова схема

Використавши функцію *transpile* з бібліотеки Qiskit можна виконати траспіляцію (рисунок 4) під конкретний квантовий комп'ютер, вказавши при цьому базові гейти ECR, RZ, SX та X [9].

```
11  transpiled_qc = transpile(
12      qc, basis_gates=['rz', 'sx', 'x', 'ecr'])
13  print("Транспільований ланцюг:")
14  print(transpiled_qc)
```

Рис. 4. Виклик функції транспіляції

Після транспіляції була отримана еквівалентна схема, яка складається лише з базисних вентилів (рисунок 5) для квантового чіпа IBM Eagle r3[9].

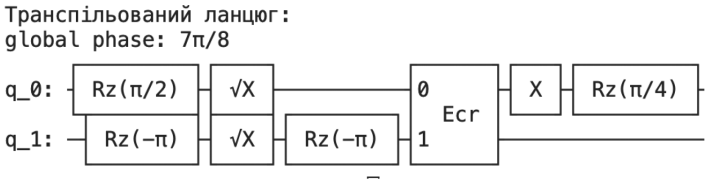

Рис. 5. Транспільований квантовий ланцюг

У транспільованій квантовій схемі вентиль *H* на кубіті *q*0 замінився на послідовність з повороту *RZ*(π/2) та *SX* (це одна з реалізацій *H* у базисі *RZ*, *SX*). Вентиль *T*, у свою чергу, став просто *RZ*(π/4) на кубіті *q*0, оскільки фазовий зсув це поворот навколо осі *Z*. Квантовий гейт *CNOT* між *q*0 і *q*1 у послідовностях дій: спочатку на контрольному кубіті *q*0 виконується фазовий поворот *RZ*(−π/2); одночасно на цільовому кубіті *q*1 виконуються послідовно три операції: *RZ*(−π), потім SX (квадратний корінь із *X*), і ще раз *RZ*(−π); застосовується базовий двокубітний вентиль ECR, після якого слідує вентиль X на контрольному кубіті *q*0, що завершує реалізацію вентиля *CNOT* [3].

В результаті отримано повністю сумісну з апаратними можливостями IBM Eagle схему транспіляції CNOT. На практиці Qiskit може генерувати варіації цієї схеми, які відрізняються порядком або фазовими зсувами, але склад використовуваних вентилів завжди залишається в межах базового набору [7].

Процес транспіляції квантових схем, описаний вище, реалізується за допомогою спеціалізованих компіляторів. В екосистемі IBM Quantum цю функцію виконує модульний компонент Qiskit Transpiler, що включає стандартні правила декомпозиції вентилів та набір алгоритмічних оптимізаційних проходів [10].

За замовчуванням транспілятором автоматично вибирається послідовність цих проходів, від початкового розміщення кубітів до скорочення квантової схеми, залежно від вибраного рівня оптимізації. Серед ключових компонентів транспілятора виділяються такі: бібліотека вентилів, облік топології пристрою та рівні оптимізації (рисунок 6).

Бібліотека вентилів транспілятора. Високорівневі вентилі розкладаються на базові за допомогою як готових схем (наприклад, Toffoli через CNOT і TTT-вентилі), так і за допомогою чисельних методів синтезу унітарних матриць. Для складних багатокубітних блоків на високих рівнях оптимізації застосовується алгоритм Картана (KAK), що мінімізує кількість двокубітних операцій [10].

Облік топології пристрою. Під час транспіляції може бути задана карта з'єднань (coupling_map) або обрано цільовий backend, з якого зчитується необхідна топологія та доступні базові вентилі. Для ефективного розміщення та маршрутизації використовується низка алгоритмів, серед яких за замовчуванням на високих рівнях оптимізації застосовується алгоритм *SABRE*. Останній стохастично визначає оптимальне розташування кубітів та мінімізує кількість вставлених *SWAP*-вентилів [10].

Рівні оптимізації. Транспілятором Qiskit підтримується параметр optimization_level (від 0 до 3), що визначає рівень зусиль, які докладаються до оптимізації. На нульовому рівні виконується лише базовий розклад вентилів без оптимізації. Перший та другий рівні здійснюють поступову оптимізацію схеми, включаючи скорочення послідовностей та застосування комутаційних правил. Третій рівень забезпечує найбільш агресивні оптимізації,

Рис. 6. Ключові компоненти транспілятора

включаючи повторну синтезацію підланцюгів та переміщення вимірювань для мінімізації SWAP-вентилів, зменшуючи глибину та кількість вентилів у схемі, хоча й вимагає більшого часу на компіляцію [10].

Висновки

Транспіляція високорівневих квантових схем у послідовності базових вентилів розглядається як обов'язковий етап підготовки квантових програм до виконання на реальних пристроях. Для досягнення максимальної ефективності обчислень має бути враховано набір базових вентилів, підтримуваних обраним квантовим процесором, а також його топологічні обмеження. З цією метою надається доступ до документації бекендів, у якій вказуються відповідні архітектурні характеристики. На основі цієї інформації може бути оцінено, наскільки ресурсомістким буде транспільований варіант вихідної схеми, особливо у випадках, коли в алгоритмі присутні вентилі, не реалізовані апаратно (наприклад, *Toffoli* або багатоконтрольовані вентилі).

Алгоритми мають бути спроєктовані з урахуванням потенційного розкладу, що дозволяє зменшити кількість багатокубітних операцій, які є найбільш чутливими до шумів. Доцільним вважається використання еквівалентних, але менш складних за реалізацією логічних конструкцій, а також залучення допоміжних кубітів для зменшення складності багатоконтрольованих вентилів. При цьому має бути забезпечена попередня оцінка впливу таких перетворень на загальну глибину та точність схеми.

Хоча сучасними транспіляторами, такими як Qiskit або Cirq, забезпечується високий рівень автоматизації, рекомендується здійснювати аналіз отриманої схеми. Зокрема, надмірна кількість *SWAP*-вентилів або значна глибина ланцюга можуть свідчити про необхідність перегляду розміщення кубітів або самої логіки алгоритму. В окремих випадках оптимізація може бути підвищена шляхом ручної перестановки логічних кубітів або модифікації вихідної схеми.

Застосування бар'єрів має бути обмеженим до тих випадків, коли це виправдано логікою обчислень, оскільки надмірна їх кількість може перешкоджати оптимізації, яку виконує транспілятор. Для досягнення оптимального співвідношення між швидкістю транспіляції та якістю отриманої схеми рекомендується тестування кількох рівнів оптимізації. У багатьох випадках третій рівень забезпечує значне зменшення кількості вентилів та глибини схеми, що позитивно впливає на точність обчислень.

Розуміння механізмів, за допомогою яких високорівневі вентилі транслюються у низькорівневі базові операції, дозволяє здійснювати більш обґрунтоване програмування квантових алгоритмів. Попри постійний розвиток квантової апаратури, базова концепція транспіляції залишається сталою: фізичні та архітектурні обмеження визначають реальний спосіб виконання алгоритмів, а ефективність квантового програмного забезпечення безпосередньо залежить від їх урахування.

Список використаної літератури

1. Rakshit, P., et al. *Hardware Trojans in Quantum Circuits, Their Impacts, and Defense*. arXiv:2402.01552, 2024. [Електронний ресурс]. Режим доступу: https://doi.org/10.48550/arXiv.2402.01552
2. A simplified Mølmer-Sørensen gate for the trapped ion quantum computer // Hiroo Azuma. [Електронний ресурс]. Режим доступу: https://doi.org/10.48550/arXiv.2112.07855
3. Ji, Y.; Polian, I. Synergistic Dynamical Decoupling and Circuit Design for Enhanced Algorithm Performance on Near-Term Quantum Devices. Entropy 2024, 26, 586. https://doi.org/10.3390/e26070586
4. Native gates and operations // IBM Quantum Documentation. [Електронний ресурс]. Режим доступу: https://docs.quantum.ibm.com/guides/native-gates
5. How Does The Qiskit Transpiler Work? [Електронний ресурс]. Режим доступу: https://medium.com/qiskit/how-does-the-qiskit-transpiler-work-6710863beaac
6. Byeongyong Park, Hansol Noh, Doyeol Ahn. Reducing T-Count in quantum string matching algorithm using relative-phase Fredkin gate. [Електронний ресурс]. Режим доступу: https://doi.org/10.48550/arXiv.2411.01283
7. Introduction to transpilation // IBM Quantum Documentation. [Електронний ресурс]. Режим доступу: https://docs.quantum.ibm.com/guides/transpile
8. Henry Zou, Matthew Treinish, Kevin Hartman, Alexander Ivrii, Jake Lishman. LightSABRE: A Lightweight and Enhanced SABRE Algorithm. [Електронний ресурс]. Режим доступу: https://doi.org/10.48550/arXiv.2409.08368
9. M. AbuGhanem. IBM Quantum Computers: Evolution, Performance, and Future Directions. [Електронний ресурс]. Режим доступу: https://doi.org/10.48550/arXiv.2410.00916
10. Transpiler stages // IBM Quantum Documentation. [Електронний ресурс]. Режим доступу: https://docs.quantum.ibm.com/guides/transpiler-stages

References

1. Rakshit, P., Naik, M., Nema, R., Sahu, A., Tripathi, S., Patel, R., Jha, R., Patnaik, S., & Rout, P. (2024). *Hardware Trojans in quantum circuits, their impacts, and defense*. arXiv preprint arXiv:2402.01552. https://doi.org/10.48550/arXiv.2402.01552
2. Azuma, H. (2021). *A simplified Mølmer–Sørensen gate for the trapped ion quantum computer*. arXiv preprint arXiv:2112.07855. https://doi.org/10.48550/arXiv.2112.07855
3. Ji, Y., & Polian, I. (2024). Synergistic dynamical decoupling and circuit design for enhanced algorithm performance on near-term quantum devices. *Entropy*, 26(7), 586. https://doi.org/10.3390/e26070586
4. IBM Quantum. (2023). *Native gates and operations*. IBM Quantum Documentation. https://docs.quantum.ibm.com/guides/native-gates
5. Qiskit. (2021). *How does the Qiskit transpiler work?* Qiskit Blog on Medium. https://medium.com/qiskit/how-does-the-qiskit-transpiler-work-6710863beaac
6. Park, B., Noh, H., & Ahn, D. (2024). *Reducing T-count in quantum string matching algorithm using relative-phase Fredkin gate*. arXiv preprint arXiv:2411.01283. https://doi.org/10.48550/arXiv.2411.01283
7. IBM Quantum. (2023). *Introduction to transpilation*. IBM Quantum Documentation. https://docs.quantum.ibm.com/guides/transpile
8. Zou, H., Treinish, M., Hartman, K., Ivrii, A., & Lishman, J. (2024). *LightSABRE: A lightweight and enhanced SABRE algorithm*. arXiv preprint arXiv:2409.08368. https://doi.org/10.48550/arXiv.2409.08368
9. AbuGhanem, M. (2024). *IBM Quantum Computers: Evolution, performance, and future directions*. arXiv preprint arXiv:2410.00916. https://doi.org/10.48550/arXiv.2410.00916
10. IBM Quantum. (2023). *Transpiler stages*. IBM Quantum Documentation. https://docs.quantum.ibm.com/guides/transpiler-stages