A Comparative Study of Quantum Circuit Mapping Techniques for NISQ Devices

You-Seok Lee, Jongheon Lee and Yousung Kang
Cyber Security Research Division
Electronics and Telecommunications Research Institute(ETRI)
Daejeon, South of Korea
yslee75@etri.re.kr

***Abstract*—Noisy Intermediate-Scale Quantum (NISQ) computers impose severe architectural constraints that prevent direct execution of logical quantum circuits. As a result, quantum circuit mapping—transforming ideal circuits into hardware-compatible forms—has become a critical task in quantum software toolchains. This paper presents a comparative study of key mapping techniques, focusing on circuit depth, SWAP gate overhead, and fidelity awareness. Through a case analysis on a 5-qubit Quantum Fourier Transform (QFT) circuit, we demonstrate the performance trade-offs of several prominent mapping strategies, including naive greedy routing, Qiskit's SABRE, and CQC's t|ket>. Our findings reveal that mapping technique selection has a profound impact on circuit quality and feasibility on real NISQ hardware.**

Keywords—NISQ, quantum circuit, quantum circuit mapping

I. INTRODUCTION

Quantum computing hardware has rapidly advanced to the NISQ era, where quantum processors comprise tens to hundreds of qubits but still suffer from noise, short coherence times, and restricted qubit connectivity [1], [2]. Under these limitations, quantum circuits designed at a logical level must be carefully adapted to the physical layout of a target device. This process—known as quantum circuit mapping—is essential to make circuits executable while minimizing performance degradation [1].

A major challenge in mapping arises when a two-qubit gate involves qubits that are not adjacent on the hardware topology. In such cases, SWAP gates must be inserted to bring qubits together, often leading to deeper and noisier circuits [1-3]. Excessive SWAP usage increases error rates and reduces the overall fidelity of computation [2], [3].

Various mapping strategies have been proposed to address this issue [2-4]. Some employ simple greedy heuristics, while others use look-ahead search or error-aware optimization. In this study, we compare representative techniques from both ends of the spectrum to evaluate their effectiveness in minimizing routing overhead and preserving circuit integrity.

II. QUANTUM CIRCUIT MAPPING TECHNIQUES

Mapping a logical quantum circuit to NISQ hardware typically involves three main steps [4]:

1. Initial Qubit Placement: Assign logical qubits to physical ones.

2. Routing: Insert SWAP gates where needed to satisfy coupling constraints.

3. Optimization: Reduce gate count or depth through commutation and rewriting.

The quality of mapping significantly affects the circuit's executable fidelity and resource requirements. We now describe three representative mapping strategies:

A. Naive Greedy Mapping

This method processes gates sequentially and inserts SWAP gates whenever two non-adjacent qubits are involved in a two-qubit gate. It does not consider future gate dependencies or topology optimization, often leading to inefficient circuits [1].

B. SABRE (SWAP-Based Bidirectional Heuristic)

SABRE, implemented in IBM's Qiskit, evaluates multiple candidate SWAPs at each step using a cost function that estimates future gate costs. It performs a forward-backward pass to find near-optimal placements with moderate runtime [4], [6].

C. t|ket> Mapping

Cambridge Quantum Computing's t|ket> employs a look-ahead search with error model integration. It selects SWAP sequences that minimize expected errors and circuit depth, yielding high-quality results at the expense of increased compile time [3], [5].

III. CASE STUDY: MAPPING THE QFT CIRCUIT

To evaluate mapping effectiveness, we consider a small but representative benchmark—the 5-qubit Quantum Fourier Transform (QFT) circuit. This circuit includes multiple two-qubit gates that span non-adjacent qubits, making it ideal for testing routing efficiency.

We assume a linear 5-qubit topology:

$$Q_0 - Q_1 - Q_2 - Q_3 - Q_4$$

The original QFT circuit has no SWAP gates and minimal depth. After mapping, however, the transformed circuits exhibit varying degrees of overhead depending on the strategy used.

TABLE I. COMPARISON OF MAPPING TECHNIQUES

<table>
<tr><th>Mapping Technique</th><th>Circuit Depth</th><th>SWAP</th><th>SIZE</th></tr>
<tr><td>Original Circuit</td><td>9</td><td>0</td><td>15</td></tr>
<tr><td>Naive Greedy</td><td>65</td><td>9</td><td>74</td></tr>
<tr><td>SABRE</td><td>63</td><td>10</td><td>68</td></tr>
<tr><td>t|ket></td><td>18</td><td>7</td><td>22</td></tr>
</table>

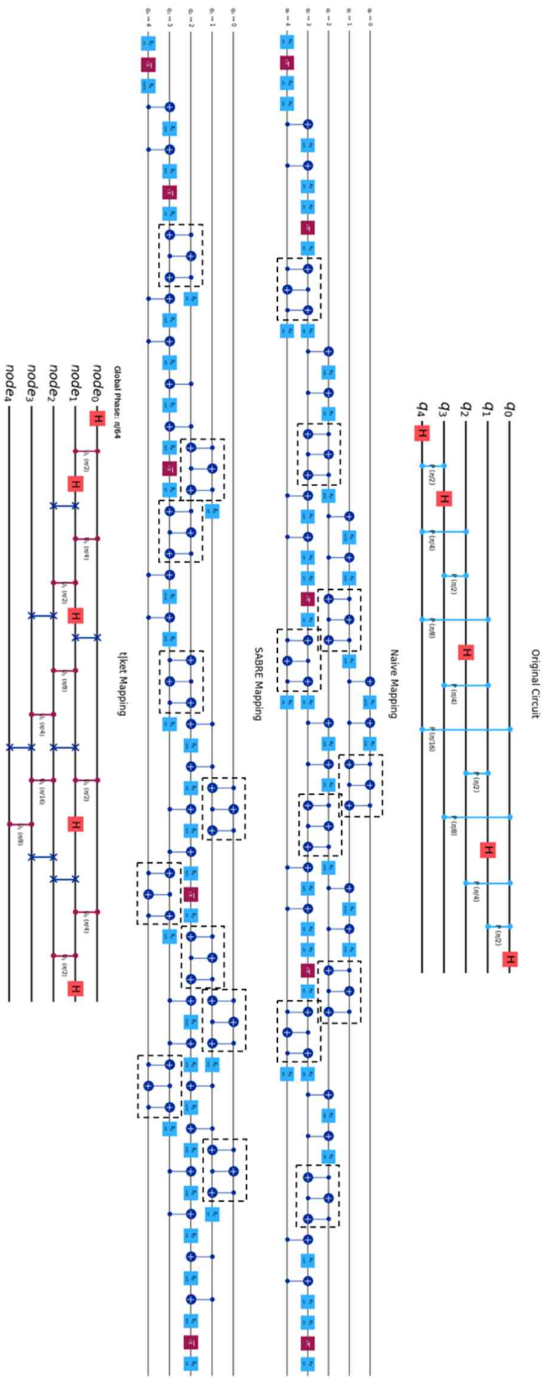

Fig. 1. Visual comparison of transpiled circuits using different mapping strategies: Original, Naive, SABRE, and t|ket>.

The results of the transpilation are summarized in Table I, which compares circuit depth, number of SWAP gates and total gate count across mapping strategies. Figure 1 shows the visual layout of the circuits resulting from each mapping method, providing an intuitive comparison of structural differences. Each SWAP gate is implemented as a sequence of three CNOT gates(see dashed box). Although SWAPs are not explicitly visible in Figure 1, they are counted analytically in Table I.

The following observations can be made from the results:

- **Naive mapping** introduces the highest overhead due to the lack of foresight in gate scheduling and placement.
- **SABRE** reduces SWAP count and circuit depth through heuristic gate reordering, but its advantage is less significant for structurally simple circuits.
- **t|ket>** produces the most compact and shallow circuit by modeling both logical structure and hardware noise.

Even for a small circuit, the differences are substantial—demonstrating up to 71% reduction in circuit depth when using advanced techniques.

IV. Conclusions

This study highlights the critical role of circuit mapping in enabling efficient quantum computation on NISQ devices. Through a case-based comparison, we showed that intelligent mapping strategies can substantially reduce circuit depth and SWAP overhead, directly improving reliability on real quantum hardware.

Among the techniques evaluated, t|ket> demonstrated the best overall performance due to its integration of look-ahead optimization and noise-awareness. While SABRE is considered effective for many real-world scenarios, our results indicate that for structurally simple circuits, its advantages over naive mapping are minimal. Naive mapping, while conceptually simple, resulted in inefficient circuits and is unsuitable for near-term quantum systems.

As quantum hardware continues to evolve, we plan to extend this comparative analysis to more complex circuits, such as those implementing cryptographic algorithms, and to conduct further studies on mapping strategies for large-scale applications.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MIST) (⟨Q|Crypton⟩, No.RS-2019-II190033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity)

References

[1] A. Zulehner, A. Paler, and R. Wille, "An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 38, no. 7, pp. 1226–1236, July 2019.

[2] G. Li, Y. Ding, and Y. Xie, "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 1001–1014.

[3] M. Cowtan et al., "On the Qubit Routing Problem," in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, 2019.

[4] A. J. Kreuzer, M. Soeken, and G. De Micheli, "SABRE: A Scalable Heuristic for Routing Quantum Circuits," *Quantum Science and Technology*, vol. 6, no. 3, 2021.

[5] Cambridge Quantum Computing, "t|ket> User Manual," 2022. [Online]. Available: https://cqcl.github.io/tket/.

[6] IBM Qiskit Documentation, "Qiskit Transpiler and SABRE Routing," [Online]. Available: https://qiskit.org/documentation/apidoc/transpiler.html.