A framework for dynamic quantum circuit execution: balancing effectiveness and efficiency

Fangzheng Chen, Hao Fu, Mingzheng Zhu, Chi Zhang, Wei Xie, Xiang-Yang Li, Fellow, IEEE

***Abstract*—Quantum computing has exhibited remarkable advancements in recent years. On superconducting quantum chips, physical qubits are interconnected with limited coupling topologies. Since two-qubit gates can only be operated between adjacent qubits, quantum circuits must undergo transformation to satisfy these connectivity constraints—a process known as qubit mapping and routing. Dynamic quantum circuits represent a critical paradigm in quantum computing, which features mid-circuit measurements and conditional control flows (controlled sub-circuits) based on measurement outcomes. A distinctive challenge arises when controlled sub-circuits are determined only after measurement results become available (online acquisition). Thus, mapping and routing dynamic quantum circuits presents complex challenges, including managing the conditional execution of controlled sub-circuits, accommodating their online/offline acquisitions, and resolving mapping misalignments between circuit segments. In this work, we propose a comprehensive framework for transforming and executing dynamic quantum circuits. In this framework, we present a mapping and routing algorithm `Sword` that effectively reduces the quantum processing time of quantum circuits, which applies SWAP(s) With Occupancy and gate Regional Density. For online scenarios, we introduce a fast variant `Sword-fast` that significantly improves transformation efficiency with negligible performance degradation. Experimental evaluations demonstrate that our framework achieves an average 37.1% reduction in end-to-end wall-clock time compared to baseline approaches for offline scenarios. In online scenarios, our framework achieves 59.2% average reduction of end-to-end wall-clock time, demonstrating 2.6× average speedup in transformation time and 30.0% average reduction in quantum processing time. By replacing the qubit mapping and routing algorithm in our framework with a fast variant, `Sword-fast`, we achieve a 9.6× average speedup in transformation time compared to the standard version.**

***Index Terms*—Quantum computing, Mapping and routing, Dynamic quantum circuit.**

I. Introduction

Quantum computing has experienced rapid development since its initial proposal. Multiple quantum algorithms are proposed for key tasks such as quantum simulation [1], large integer factoring [2] and database searching [3]. To implement quantum algorithms, it must be transformed into quantum circuits for execution on specific devices. A quantum circuit is a computational routine that consists of quantum operations and real-time classical computation. Currently, most quantum circuits are relatively simple, typically comprising qubit initialization, quantum gate operations (including single-qubit and multi-qubit gates), and final measurements. Although these universal circuits have been widely applied in practice, they are insufficient for dynamic applications such as quantum teleportation [4], long-range entanglement generation [5], quantum error correction [6], and the measurement-based model of quantum computing [7], which require measuring specific qubits in the middle of quantum circuit and controlling the subsequent execution (we call this *controlled sub-circuits*, the whole circuits as *dynamic circuits*). Recent studies have utilized dynamic circuits to achieve better performance compared to conventional quantum circuits. For example, dynamic circuits have been applied to enhance the Quantum Fourier Transform (QFT) [8] and Quantum Phase Estimation (QPE) [9], reduce the depth of conventional circuits while improving fidelity [10], link two quantum chips to function as a single unit [11], and prepare high-fidelity magic states [12]. Dynamic quantum circuits with complex controlled sub-circuits transcend simple feedback models [13], offering significant potential for quantum computing.

For the implementation of dynamic quantum circuits, it is necessary to transform them into a form compatible with hardware, as is the case for conventional serialized quantum circuits. This process is known as qubit mapping and routing. For example, on superconducting quantum hardware, circuits must satisfy the connectivity constraints, which means that a two-qubit gate can only be applied between qubits that are directly coupled. Conventional mapping approaches often focus on the performance of circuit execution: reducing the number of gates or minimizing quantum processing time of circuit on quantum hardware, as well as improving circuit fidelity [14], [15], [16], [17], [18], [19].

These conventional approaches assume a static circuit structure, where the entire transformation is performed once before execution. However, in dynamic quantum circuits, both the execution of a controlled sub-circuit and the number of times it is executed are uncertain. Conventional mapping approaches fall short in supporting DQCs for two major reasons. Firstly, they lack support for dynamic control flow: by assuming a predetermined sequence of gates, they cannot maintain consistent mappings when conditional branches converge. Secondly, they lack scenario differentiation: when controlled sub-circuits vary or arrive in real time based on measurement results [13], [20], mapping processes must be extremely efficient due to

Fangzheng Chen, Hao Fu, Mingzheng Zhu, and Wei Xie are with the LINKE lab, University of Science and Technology of China (USTC), Hefei 230027, China (e-mail: {fangzhengchen, hflash, zmzming}@mail.ustc.edu.cn, xxieww@ustc.edu.cn). Chi Zhang is with School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China and Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China (e-mail:zhangchi@hfut.edu.cn). Xiang-Yang Li is with the LINKE lab and Hefei National Laboratory, University of Science and Technology of China (USTC), Hefei 230088, China (e-mail: xiangyangli@ustc.edu.cn). Wei Xie is corresponding author. Fangzheng Chen and Hao Fu are co-first authors.

limited coherence times. We distinguish between online (requiring real-time performance) and offline scenarios (without time constraints). Offline scenarios prioritize reducing qubit interaction latency on the quantum hardware (effectiveness), while online scenarios demand low-latency compilation during runtime (efficiency). Neglecting this distinction leads to suboptimal result.

However, addressing these challenges for DQCs on near-term quantum hardware introduces a critical timing conflict. The coherence times of current superconducting qubits are typically in the range of tens to hundreds of microseconds. In contrast, the classical compilation (transformation) time required for non-trivial sub-circuits in an online scenario can consume more than one millisecond. This reality underscores the necessity of developing an execution framework for dynamic quantum circuits targeting offline and online scenarios. First, for offline scenarios, where all circuit branches are known beforehand, transformation time does not interfere with quantum execution. In this context, the primary goal remains the optimization of the final circuit's effectiveness (i.e., minimizing quantum processing time). Second, for online scenarios, by developing a systematic framework and highly efficient heuristic algorithms, it is necessary to build an essential compilation infrastructure that will become increasingly practical as quantum hardware evolves—either through significantly extended coherence times [21], [22], [23] or the development of low-latency classical control systems [24], [25], [26].

Therefore, dynamic quantum circuits require more than a conventional mapping technique. It is necessary to address challenges such as branch execution arising from execution-flow uncertainties and the different optimization priorities of compilers across scenarios. It necessitates a system-level solution that integrates circuit execution with mapping.

With these distinct requirements and current hardware limitations, this work presents a framework designed to manage DQC transformation and execution across both offline and online scenarios. In this framework, we propose `Sword`, which achieves SWAP(s) With Occupancy and gate Regional Density, as a method for efficiently optimizing the quantum processing time of dynamic quantum circuits in large-scale quantum devices with connectivity constraints. We first analyze the execution characteristics of dynamic quantum circuits and propose an efficient transformation framework tailored to their intermediate measurements in online or offline scenarios. Based on this framework, we reduce quantum processing time by incorporating occupancy of hardware qubits and gate density of quantum circuit during processing. Next, we analyze existing heuristic mapping methods and observe that their consistent preference in SWAP selection of shortest path between qubits. With this observation, we provide `Sword-fast`, which prunes the search space thereby significantly reducing transformation time. Finally, for mapping alignment, we propose `acc-Miltzow`, which modifies the original 4-approximation token-swapping algorithm [27], by modifying the selection process from choosing multiple SWAP operations per iteration to selecting just one.

Our evaluation results demonstrate significant performance improvements across various scenarios. For offline scenarios, our method achieves substantial reductions in quantum processing time, up to 46.2%, with an average reduction of 37.1%. For online scenarios, our method maintains considerable performance with a maximum end-to-end wall-clock time reduction of 73.7% and an average reduction of 59.2%. Individually, our approach achieves an average quantum processing time reduction of 30.0%, while simultaneously achieving a 2.6× average speedup in transformation time. The accelerated variant `Sword-fast` of our algorithm exhibits considerable computational efficiency, achieving a 9.6× average speedup in transformation time. Furthermore, our optimized mapping alignment algorithm `acc-Miltzow` effectively addresses mapping alignment challenges, reducing both the transformation time required for alignment resolution by 56.4% and the average depth of the resulting circuits by 11.0%, respectively.

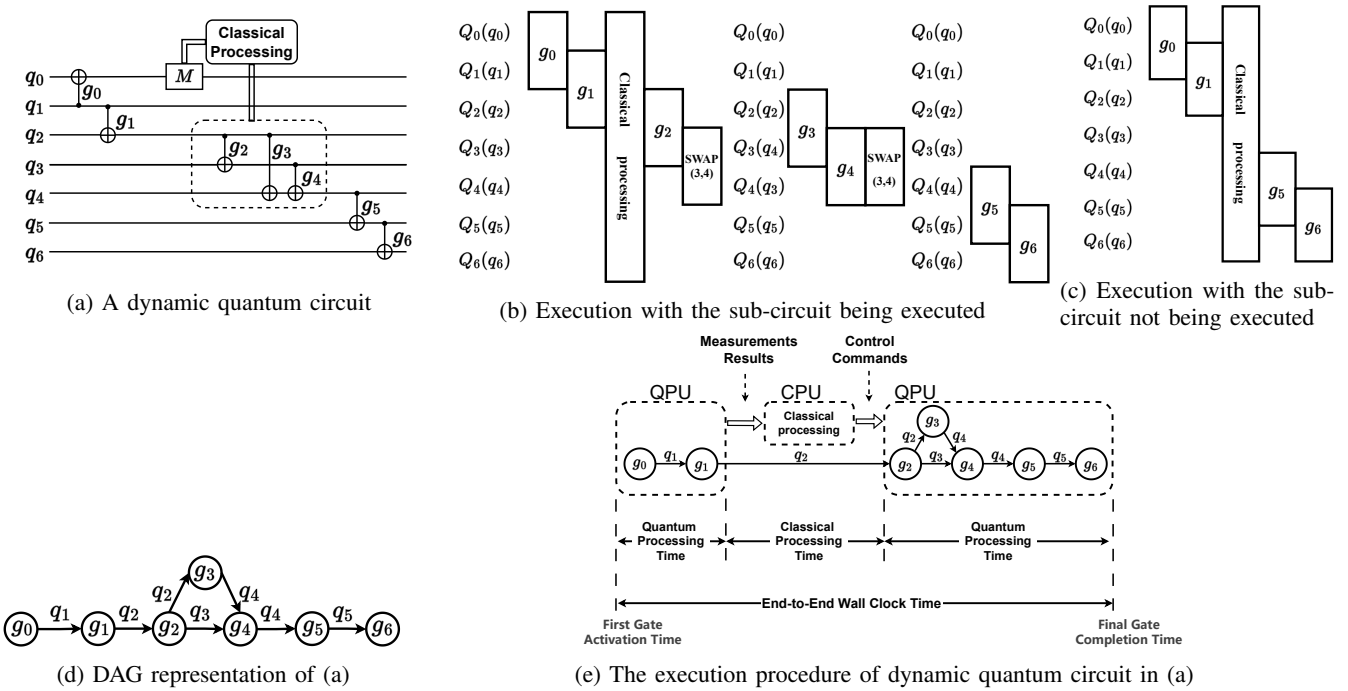
The rest of this paper is organized as follows. Section II first briefly introduces quantum computing, then formally describes qubit mapping and routing problem for dynamic circuit. Section III specifically describes the algorithmic process of our method and the optimization strategies. Section IV provides an evaluation of our method. We conclude this work with future directions in Section V.

II. Preliminaries

A. Quantum computing

Dynamic Quantum circuit. In contrast to serialized quantum circuit, dynamic quantum circuit incorporates mid-circuit measurements—either measuring a subset or all qubits—and operations based on outcomes of these measurements. These operations primarily include loop and conditional blocks, which we refer to as the sub-circuits of quantum control flows (hereinafter referred to as *controlled sub-circuits* in this paper, conditional and loop sub-circuits for conditional and loop blocks, respectively), while the remaining execution blocks are designated as the only one main circuit. The execution procedure of a dynamic quantum circuit is determined by the specific algorithm or scenario. For instance, the adaptive dynamic QAOA circuit measures all qubits and determines the next execution circuit based on the measurement results [13]; similarly, quantum loop circuits [28] decide whether to execute the next loop or continue to the main circuit based on the measurement outcomes. Generally, the algorithm and application scenarios dictate the method of acquiring controlled sub-circuits, whether they are obtained offline (the controlled sub-circuits are fully known before execution) or online (the corresponding controlled sub-circuits can only be obtained after measurement results are obtained). At present, dynamic quantum circuits are simplistic in practice; however, as quantum applications evolve and scenarios become increasingly complex—such as in distributed quantum computing—dynamic quantum circuits that incorporate complex controlled sub-circuits are expected to become more prevalent.

Superconducting quantum chip. Superconducting qubits can perform gate operations within nanoseconds. Superconducting quantum chips are typically two-dimensional and often exhibit regular configurations. In addressing the quantum mapping and



(a) A dynamic quantum circuit

(b) Execution with the sub-circuit being executed

(c) Execution with the sub-circuit not being executed

(d) DAG representation of (a)

(e) The execution procedure of dynamic quantum circuit in (a)

Fig. 1: Dynamic quantum circuit, its DAG representation and its execution procedure, whether the operations of the dashed box in (a) will be executed is determined by the measurement results of qubit q_0. (b) and (c) represent the execution flows of the dynamic quantum circuit in (a) on a one-dimensional chain-like chip, respectively, where the sub-circuit is executed (as shown in (b)) and not executed (as shown in (c)). Q_i denotes physical qubits, and q_i denotes logical qubits. In (d) and (e), the circles means two-qubit gates in (a), the solid arrows refer to the quantum data dependencies between gates, and the hollow arrow means the transmission of classical data.

routing problem, these architectures are generally represented as undirected graphs [29], [30], [15]. Along with the variety of execution time for gates operated on different qubit(s), our work accounts for gate execution times.

Qubit mapping and routing for dynamic quantum circuit. When designing quantum algorithms, it is commonly assumed that gate operations can occur between any two qubits. However, hardware connectivity limitations necessitate the transformation of the original quantum circuit into one that conforms to the actual device's constraints. The primary goal is to facilitate circuit execution, starting with mapping the qubits in the circuit to the physical qubits on the chip (*mapping*). This is followed by inserting SWAP gates to enable the execution of two-qubit gates on non-directly connected qubits (*routing*). Collectively, this is known as qubit mapping and routing (the corresponding solver for this circuit transformation is called *mapper* throughout this paper).

In dynamic quantum circuits, due to the uncertainty in the execution of controlled sub-circuits, the execution and mapping of the quantum circuit are highly coupled. When mapping a quantum circuit, the execution flow of the circuit needs to be considered. Considering the circuit shown in Fig. 1a, we execute it on a one-dimensional chain hardware and must consider both cases: if the sub-circuit is skipped (Fig.1c), only gates g_0, g_1, g_5, and g_6 are applied; if executed (Fig. 1b), additional SWAPs are needed, e.g., between Q_3 and Q_4 after g_2 and before g_5. Although g_5 is outside the sub-circuit, previous execution alters the mapping and correspondingly changes the subsequent routing. To maintain consistency, an extra SWAP before g_5 ensures *mapping alignment*, making later routing independent of sub-circuit execution.

B. Problem formulation

This work, with particular emphasis on dynamic quantum circuits, addresses the heterogeneity among qubits in real quantum devices by proposing an optimization approach that jointly considers quantum processing time and transformation time of qubit mapping and routing to optimize the end-to-end wall-clock time of execution of circuits.

We consider the dynamic quantum circuit C, which consists of n qubits (denoted as Q_c) and exclusively employs CNOT gates as two-qubit gates. This circuit is modeled as a Directed Acyclic Graph (DAG), where vertices represent quantum gates and edges depict their dependencies, as illustrated in Fig. 1d. The depth of the circuit is calculated by the length of the critical path of the DAG. For a quantum chip with m qubits (where $m \geq n$), represented as $G = (Q_d, E, W)$, Q_d denotes the qubits and E represents the couplings. Considering the variability of CNOT gates, we use the weight function $W : E \to \mathbb{R}$ to indicate execution times for each CNOT gate.

For real-world implementation of dynamic quantum circuits, the execution procedure can also be represented as a DAG, while each node in this DAG denotes a set of operations (classical or quantum) and consumes a certain amount of time. We define the end-to-end wall-clock time T_{E2E} as the

maximum total time consumed by nodes along the critical path of this DAG. We have a mapping function $\pi = f : Q_c \to Q_d$, mapping the input quantum circuit's qubits Q_c to the quantum chip's qubits Q_d. The execution procedure of input dynamic quantum circuit C is denoted as C_H, which includes a initial mapping and the transformed circuits of main circuit and controlled sub-circuits. As shown in Fig. 1, Fig. 1e indicates the execution scheme of circuit Fig. 1a whose DAG is shown in Fig. 1d.

We introduce the following notation to represent the time parameters involved in the problem:

- RT (runtime of quantum compilers): the transformation latency incurred on classical processors.
- T_{QPU} (quantum processing time): the physical execution time of all gates and measurements on the quantum processing unit (QPU).
- T_{E2E} (end-to-end wall-clock time): the total wall-clock duration from the first gate trigger to the final measurement. For offline scenarios, $T_{E2E} = T_{QPU}$; for online scenarios, $T_{E2E} = T_{QPU} + RT_{sub}$, where RT_{sub} refers to the transformation time of sub-circuits.

Based on the system model defined above, we then define the qubit mapping and routing problem:

Problem 1. *Min-end-to-end wall-clock time qubit mapping and routing for dynamic quantum circuit.*
Input: A dynamic quantum circuit C and quantum chip with exact topology depicted by G.
Output: An execution procedure C_H which satisfies chip connectivity constraints and T_{E2E} is minimized.
Problem Hardness: Min-depth Qubit Routing with initial mapping, as a formulation for mapping and routing a serialized quantum circuit is a special case of this problem. While Min-depth Qubit Routing with initial mapping is shown to be NP-hard [16], this problem is NP-hard.

C. Related work

Significant prior work exists in qubit mapping and routing [15], [16], [17], [18], [30], [31], [32], [33], [34], alternatively referred to as qubit layout, qubit allocation, or quantum circuit transformation in the literature. These studies propose diverse solutions that employ either exact or heuristic approaches while optimizing for various objectives, including gate count reduction, circuit depth minimization, and fidelity enhancement.

Molavi et al. [30] formulate the qubit mapping and routing problem as a MAXSAT instance, demonstrating its NP-completeness. Wille et al. [31] transform the problem into a symbolic optimization problem, employing solvers to minimize gate counts. Their experimental results quantitatively characterize the performance gap between heuristic approaches and theoretical lower bounds. Wagner et al. [35] decomposes the mapping problem into two problems: allocation subproblem and token swapping, improving the efficiency of solving. Zulehner et al. [32] adopt gate count minimization as the optimization objective, implementing a layered quantum circuit approach with A^* search applied per layer. Siraichi et al. [18] recast the problem as subgraph isomorphism coupled with token swapping. Sinha et al. [33] enhance Monte Carlo tree search using graph neural networks to optimize circuit depth, while Pozzi et al. [34] employ reinforcement learning tailored to specific chip architectures.

Zhang et al. [15] establish an execution-time-optimized theoretical model for qubit mapping and routing, proposing a constrained search framework. Li et al. [17] develop a heuristic method that efficiently identifies appropriate SWAP operations, demonstrating scalable performance even for large-scale quantum circuits. Fu et al. [16] discover two patterns that degrade conventional greedy methods. Liu et al. [36] proposes a method for dividing large circuits into smaller circuits and designs a mapping algorithm for large-scale quantum circuits. Similarly, Cheng et al. [37] addresses large-scale quantum circuits by employing limitedly exhaustive search and shortest-path estimation approach.

In addition, Li et al. [38] propose a mapping scheme that incorporates single-qubit gates into the mapping problem. Padda et al. [39] reduce the depth and gate count by introducing entanglement during the mapping process. Meanwhile, Huang et al. [40] consider not only SWAP gates but also the insertion of BRIDGE gates. Saravanan et al. [41] accounts for hardware noise during mapping, improving the fidelity of the circuit. Ji et al. [42] specifically focus on optimizing mapping for variational quantum algorithms.

While these conventional mapping strategies are effective for static circuits, they cannot directly address execution uncertainty in DQCs. A naive unfolding of all possible conditional branches results in exponential growth in circuit number and compilation overhead, making such approaches impractical for real-world DQCs.

To support dynamic quantum circuits, Cross et al. [43] propose the OpenQASM 3. Qiskit [44] provides a basic framework that transforms controlled sub-circuits independently before integrating them via mapping alignment for dynamic quantum circuit. However, their approaches do not differentiate between online and offline scenarios, consequently lacking specialized optimization strategies.

A critical challenge is the mapping alignment problem—the process of converting logical-to-physical mapping of qubits into target mapping through SWAP gate insertion. This problem can be formulated as a token swapping problem: given an undirected graph representing hardware topology (vertices as physical qubits and tokens as logical qubits), the objective is to route all logical qubits to their target physical locations via adjacent token swaps (SWAP gates). Previous work [27], [45], [46] prove the complexity of this problem, with Miltzow et al. [27] developing a 4-approximation algorithm.

III. System design

In this section, we present a framework for transforming and executing dynamic quantum circuits.

Design insights: Based on our analysis of the execution scheme for dynamic quantum circuits, we propose a comprehensive framework three challenges: effectiveness (quantum processing time), efficiency (transformation time) and mapping alignment. This framework balances the effectiveness

(quantum processing time) and efficiency (transformation time of circuits with classical hardware) with several optimization strategies. Firstly, we introduce `Sword`, which achieves SWAP optimization with occupancy and gate regional density, an approach to solving the qubit mapping and routing problem. Secondly, to improve efficiency to satisfy the time constraints for transformation time, we develop `Sword-fast`, applying a search space pruning strategy based on mapping and routing results analysis. Moreover, we provide a mapping alignment algorithm `acc-Miltzow` to better accommodate the requirements of dynamic sub-circuits, achieving simultaneous reductions in both transformation time and quantum processing time. The overall framework is shown in Fig. 2.

A. Overall Framework

Dynamic quantum circuits introduce mid-circuit measurements and the controlled sub-circuits based on these measurements. While the quantum processing time of circuits must also be considered, these distinctive characteristics introduce novel challenges to qubit mapping and routing, including mapping misalignment between circuit segments (main circuit or controlled sub-circuits) and efficiency requirements for circuit processing. Consequently, there is a critical need for methodologies that effectively and efficiently address these challenges, adapting to the execution paradigm of dynamic quantum circuits.

A straightforward strategy for processing a dynamic quantum circuit involves enumerating all possible execution paths for each conditional branch (e.g., whether the conditional sub-circuits will be executed). However, this strategy requires the exponential resource growth with the number of controlled sub-circuits increasing. An alternative approach involves just-in-time (JIT) processing, triggered by mid-circuit measurement outcomes to determine how the controlled sub-circuits be executed. JIT processing approach introduces latency in quantum hardware execution while waiting for the results of the circuit transformation, thereby consuming valuable hardware time, i.e., the limited lifetime of qubits.

As shown in Fig. 2, we propose a comprehensive execution framework for dynamic quantum circuits in offline and online scenarios that comprises three essential components: circuit segmenting, main circuit processing, and controlled sub-circuit tailoring across various scenarios. The detailed explanation is listed as follows:

- Circuit segmenting: partitioning the dynamic quantum circuit into a main circuit and sub-circuits;
- Main circuit transformation: processing the main circuit to generate both the transformed main circuit and initial mappings of qubits for sub-circuits;
- Sub-circuit processing: sequentially transforming each sub-circuit (performing real-time transformation during execution in online scenarios) while performing mapping alignment to generate transformed sub-circuit.

We show a simple example to illustrate the procedure of our framework in Fig. 3. The example considers two scenarios: offline (sub-circuits are known prior to execution), and online (sub-circuits are determined only during execution based on measurement outcomes). The unitary U_2 constitutes a conditional sub-circuit whose execution depends on the measurement outcome of qubit q_0.

Additionally, the term π_i in Fig. 3a denotes the mapping from qubits in circuit to physical qubits on hardware after circuit segments are executed. When the condition is satisfied (sub-circuit will be executed), after the controlled sub-circuits U_2 executed, the mapping from qubits in circuit to physical qubits will be changed from π_0 to π_1. This resulting mapping π_1 of U_2 is then used as the initial mapping for the subsequent part U_3 of main circuit. Conversely, when the condition is not satisfied (sub-circuit will not be executed), the U_3 block inherits the original mapping π_0 directly. Thus, mappers must consider the mapping misalignment introduced by the conditional execution of the sub-circuits.

Controlled sub-circuits known (offline): As shown in Fig. 3b, our framework segments the original dynamic quantum circuit into two types of components: the two parts U_1 and U_3 of main circuit, and the controlled sub-circuits U_2. Each component undergoes independent transformation. The processing of the controlled sub-circuit U_2 is through a two-stage procedure. First, U_2 is transformed with initial mapping π_0, after which mapping alignment is applied to align the final mapping of qubits to the initial mapping. The end-to-end wall-clock time is given by $T_{E2E} = T_{QPU}$, representing the sum of the execution times of U_1,U_2,U_3 on the hardware.

Controlled sub-circuits unknown (online): As shown in Fig. 3c, our approach behaves similar to the offline version, with the key distinction lying in sub-circuits processing, due to the online arrival of controlled sub-circuit U_2. The approach begins with segmenting the original quantum circuit into main circuit and controlled sub-circuits. The main circuit is processed offline, while the processing of U_2 is dynamically determined during execution. The end-to-end wall-clock time is given by $T_{E2E} = T_{QPU} + RT_{sub}$, which additionally includes the transformation time of U_2.

Building upon this framework, we introduce three optimization approaches specifically designed to address the mapping and routing challenges inherent in dynamic quantum circuits, which are listed as follows:

- **Effectiveness enhancement in mapping and routing**: The effectiveness of qubit mapping and routing, remains a paramount consideration in quantum computing, as demonstrated by extensive prior research focusing on optimization metrics including gate count reduction [17], [32] and quantum processing time minimization [15], [16]. Within our framework, we target the minimization of end-to-end wall-clock time as our primary optimization objective. We propose a novel heuristic function for SWAP gate selection, which demonstrates significant reduction in quantum processing time.
- **Efficiency promotion in mapping and routing**: Optimization of mapping and routing efficiency is particularly critical in scenarios where sub-circuits are dynamically determined by measurement outcomes during execution. Consequently, the processing time for mapping and routing on classical hardware significantly impacts the overall execution performance of dynamic quantum circuits.

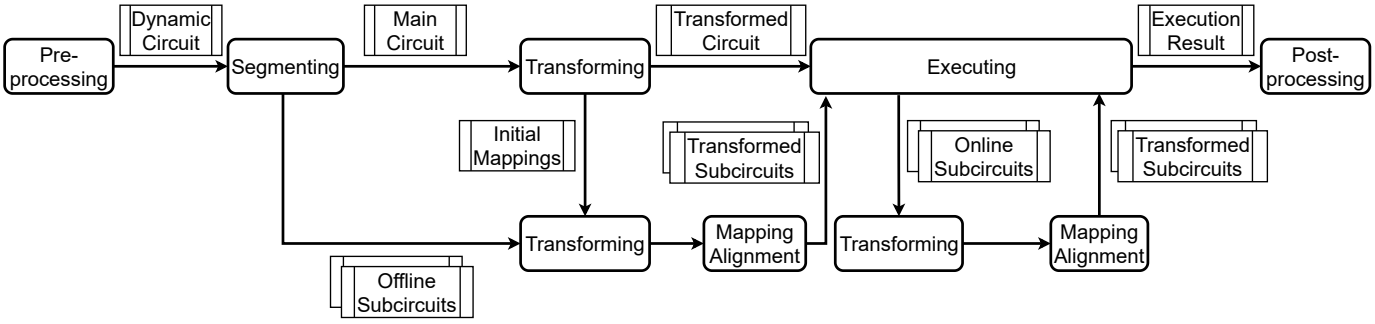
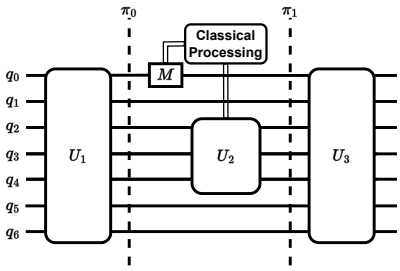
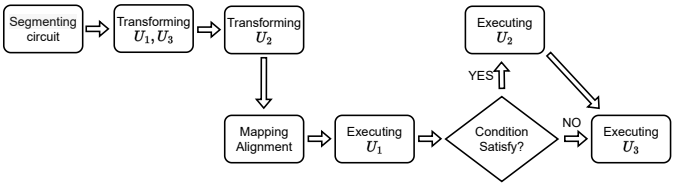


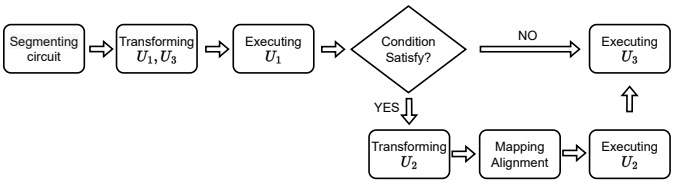
Fig. 2: Transformation and execution framework for dynamic quantum circuit.



(a) Simplified diagram of circuit in Fig. 1a



(b) Work flow in offline scenarios (U_2 known)



(c) Work flow in online scenarios (U_2 unknown)

Fig. 3: An example to illustrate our framework for two scenarios. For offline scenarios, the end-to-end wall-clock time only consists of quantum processing times for each segment of the input dynamic quantum circuit. For online scenarios, the end-to-end wall-clock time must take the necessary transformation time of controlled sub-circuits (including mapping alignment) into account.

Through systematic observation and analysis of serialized circuit routing outcomes, we develop an efficient pruning methodology. This approach prioritizes operations based on a strategic sorting algorithm that evaluates both quantum gates within the circuit and candidate SWAP operations, resulting in substantial enhancement of circuit mapping and routing efficiency across diverse quantum architectures.

- **Mapping alignment optimization**: It represents a critical challenge in the execution of dynamic quantum circuits, where mappers must address the misalignment introduced by the conditional execution of controlled sub-circuits. This challenge can be formulated as a token swapping problem, which has been previously analyzed with respect to computational complexity [27], [45], [46]. Prior research [27] has proposed a 4-approximation algorithm for minimizing the number of inserted SWAPs. However, this approximation approach fails to adequately address the primary objective of optimizing quantum processing time. To overcome this limitation, we introduce an enhanced token swapping algorithm `acc-Miltzow` with an efficient SWAP selection strategy specifically designed for optimizing the quantum processing time. Our evaluations demonstrate that this modified approach achieves significant improvements in both quantum processing time effectiveness and computational efficiency compared to the original 4-approximation algorithm.

B. Main algorithm

We propose three optimization schemes for mapping and routing dynamic quantum circuits in offline and online scenarios. We utilize a qubit mapping and routing algorithm with a refined heuristic routing function to efficiently tackle this task. First, when designing the heuristic routing function, we integrate hardware information and characteristics of circuits to reduce the quantum processing time (enhancing the circuit execution parallelism), thus reducing end-to-end wall-clock time. Second, considering time sensitivity in online scenarios, we focus on balancing effectiveness and efficiency by pruning during routing. Finally, we propose a modified mapping alignment algorithm `acc-Miltzow` , which improves the 4-approximation token swapping algorithm [27] to suit dynamic quantum circuits better.

1) Mapping and routing: Our mapping and routing algorithm, `Sword`, which applies SWAP(s) with occupancy and gate regional density, employs a two-step process: preprocessing, and the mapping and routing procedure. First, the input quantum circuit C is converted into a directed acyclic graph (DAG), denoted as G_d, and the shortest path between any two qubits on the chip is precomputed as D_{ij}. We restrict candidate SWAP operations to those associated with qubits involved in gates within the front layer F, which consists of two-qubit gates whose predecessors have already

Algorithm 1: `Sword`

```
1  Input: Circuit C, chip G, initial mapping π_Init;
2  Output: Resulting circuit C_out;
3  Transform C to DAG G_d, calculate distance matrix D_ij;
4  Initialize front layer F and extended layer E, set C_out ← ∅;
5  Execute all executable gates in F and update F, E and add the executed gates to C_out;
6  while F is not empty do
7      Candidate SWAP set S ← GetCandidateSWAPs();
8      Density ρ ← |F|/n;
9      if ρ > r then
10         minScore ← +∞;
11         while True do
12             s_opt ← arg min_{s∈S} score(s);
13             if score(s_opt) < minScore then
14                 Append s_opt and all single-qubit gates not affecting s_opt to C_out;
15                 minScore ← score(s_opt);
16                 update S;
17             else
18                 break;
19     else
20         s_opt ← arg min_{s∈S} score(s);
21         Append s_opt and all single-qubit gates not affecting s_opt to C_out;
22     Append all executable gates to C_out and update F and E;
23 Return C_out
```

been processed. From these candidates, we select the most suitable SWAP(s). Additionally, we dynamically adjust the SWAP selection strategy based on the circuit's regional gate density ρ. We define this density as the ratio of the number of two-qubit gates in the front layer F to the total number of qubits n (i.e., $\rho = |F|/n$). This value is then compared against a predefined threshold, r, to switch between a parallel or a sequential SWAP selection strategy. A higher density suggests that a parallel approach may yield better performance. Given an input quantum circuit C, an initial mapping π_{Init}, and the quantum chip's topology graph G, our algorithm produces a transformed circuit that adheres to the connectivity constraints while minimizing the quantum processing time. The pseudo-code is presented in Algorithm 1, with further details provided below:

Pre-process (line 3∼5): Firstly, we convert the quantum circuit C into a directed acyclic graph (DAG). Then, the shortest path matrix D_{ij} is then computed using the Floyd-Warshall algorithm [47]. Next, the front layer F is initialized. Simultaneously, the extended layer E is initialized, which comprises two-qubit gates in the subsequent layer(s) of F.

Get candidate SWAPs (line 7): If F is not empty, identify candidate SWAP set S. For each qubit involved in gates of F, it can be swapped with any of its adjacent qubits on the quantum chip G. The set S of all possible SWAP operations between these qubits and their neighbors forms the candidate SWAP set. If the front layer F is empty, terminate the SWAP selection process and return the routing result C_{out}.

Select SWAP(s): Get the current circuit density ρ **(line 8)**, if:

(1) $\rho > r$: employ the SWAP combinations selection strategy. Initialize the score to a sufficiently large value **(line 10)**. Iterate over the candidate SWAP set S and select the optimal SWAP that maximizes the score reduction if the SWAP is applied **(line 12)**. Append the SWAP to C_{out} **(line 14)** and remove from the candidate SWAP set all elements that share qubits with the selected SWAP **(line 16)**. Repeat the process until no further score-reducing SWAP exists.

(2) $\rho \leq r$ **(line 19∼21)**: employing the one by one SWAP selection strategy. Select the SWAP operation from the candidate set that yields the lowest score when applied, then append it to C_{out}.

Update (line 22): Remove all executable gates from the DAG, and update front layer F and extended layer(s) E accordingly. Append these executable gates to C_{out}, then return to getting candidate SWAPs **(line 7)**.

In the routing process, it is necessary to evaluate the impact of candidate SWAPs. Therefore, we build upon common features from existing heuristic functions to assess whether the total remaining distance decreases after applying the SWAPs. The general formula expression for this evaluation is as follows:

$$\bar{d} = \frac{1}{|F|} * \sum_{(q_i,q_j)\in F} D_{ij} + \omega_1 * \frac{1}{|E|} * \sum_{(q_i,q_j)\in E} D_{ij}, \quad (1)$$

where F represents the set of nodes with a zero in-degree in G_d, E denotes the set of nodes from the preceding layer(s) of the DAG, excluding those in F. The pair (q_i, q_j) represents a node in DAG G_d, indicating that a two-qubit gate must be executed between the physical qubits q_i and q_j. D_{ij} means the shortest path matrix of qubits on hardware. The parameter ω_1 serves as a weight factor to balance the influence of the front layer F against entended layers E.

`Sword` extracts information from the physical qubits' occupancy. The occupancy will change during the routing process. When selecting SWAPs, our algorithm integrates these information into a coefficient that multiplies the score from Eq. (2), serving as our heuristic function. Furthermore, our algorithm considers the characteristics of the circuit, particularly the density of two-qubit gates. We explore the SWAP selection strategies of one by one and SWAP combinations, analyze the impact of density, and propose an adaptive strategy selection method to further reduce the overall circuit depth and quantum processing time.

Heuristic function: To construct an appropriate heuristic function to estimate the impact of SWAP s, we provide a more detailed characterization of the *occupancy*, allowing our approach to better utilize this state information of qubits on the chip. The expression of our heuristic function is:

$$score(s) = \bar{d} * (1 + \omega_2 * \frac{\max\limits_{q_i\in s} \mathcal{O}(q_i) - \min\limits_{q_i\in G} \mathcal{O}(q_i)}{\max\limits_{q_i\in G} \mathcal{O}(q_i) - \min\limits_{q_i\in G} \mathcal{O}(q_i)}), \quad (2)$$

where $\mathcal{O}(q_i)$ represents the operation time that a qubit q_i is occupied. The parameters ω_2 is weight factor that controls the penalty for selecting SWAPs involving highly occupied qubits.

This formulation implies that SWAP operations involving highly occupied qubits will not be selected, despite potentially offering significant reductions of average remaining distance ($\bar{d}$). Instead, our method tends to select SWAPs acting on lower occupied qubits. Consequently, our algorithm prefer to choose a path with less execution time.

And the updated $\mathcal{O}'$ after any gate execution is:

$$\mathcal{O}'(q_j) = \max_{q_i \in g} \mathcal{O}(q_i) + \mathcal{T}(g),\ \forall q_j \in g. \tag{3}$$

while $\mathcal{T}(g)$ denotes the time required to execute a gate g.

Occupancy of qubits on the chip. Considering the qubits' occupancyduring the routing process is crucial. Previous work [17] introduce a similar concept as decay; however, their formula adds a fixed value for each executed SWAP and frequently resets these values to 0. Although this method behaves effectively in practice to some extent, it fails to accurately reflect and utilize the qubits' occupancy effectively. As the routing progresses, the number of gates, including SWAPs, executed on different qubits vary, leading to fluctuations in occupancy. As illustrated in the Fig. 4, if a CNOT operation is required between Q_1 and Q_3, the start time of the SWAP between Q_2 and Q_3, i.e., $\mathcal{O}(Q_2)$, is earlier than the one between Q_1 and Q_2 ($\mathcal{O}(Q_1)$). Thus, the target CNOT operation will be earlier executed if we choose the SWAP between Q_2 and Q_3.

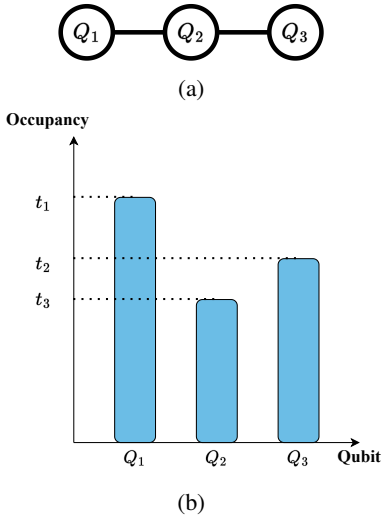


Fig. 4: Example of qubits' occupancy: (a) the topological architecture of a quantum device, and (b) the corresponding qubits' occupancy during quantum circuit execution.

Regional Gate density impact: It is crucial to consider both the characteristics of the quantum chip and the regional gate density of the circuit. Circuits with the same depth may exhibit considerable variance in gate count, especially two-qubit gates (we use gate density ρ to describe this variance). Gate density can vary not only across different circuits but also within one circuit across multiple layers. Some regions of the circuit are dense, while others are relatively sparse. Our method adaptively chooses between the SWAP combinations and one by one SWAP selection strategies based on the current gate density during the routing process. Consequently, when calculating the circuit's density, we consider only the front layer's information, i.e., two-qubit gates in F. The density is expressed as $\rho = |F|/n$, where $|F|$ represents the number of two-qubit gates in the front layer and n denotes the total number of qubits in the circuit. If ρ exceeds a defined threshold r, the SWAP combinations selection strategy is applied; otherwise, the one by one strategy is employed. r is empirically set to 0.25 in our evaluations.

2) Transformation time reduction: The greedy heuristic approach operates by iteratively selecting the suitable SWAP operation(s) from a candidate set during each algorithm iteration. A key determinant to promote algorithmic efficiency is the reduction in the number of candidate SWAP operations evaluated per iteration. Certain heuristic methods achieve this by constraining candidate SWAPs to those gates in front layer F. Our theoretical analysis and experimental results reveal that current candidate SWAP selection strategies still possess substantial optimization potential.

From the standpoint of heuristic function design, applicable to both our method and SABRE [17], the fundamental principle remains unchanged: a SWAP operation that decreases the distance between two qubits participating in a two-qubit gate is preferred. For any specific two-qubit gate, the shortest path between the corresponding physical qubits in the hardware topology naturally constitutes an optimal sequence of such preferred SWAP operations. This inherent property leads to a clear preference in practical qubit mapping and routing: SWAP operations that reduce inter-qubit distances are significantly more likely to be selected than those that increase them.

TABLE I: Proportion of two types of SWAP.

Benchmark name	SABRE		Sword	
	In-path	Out-of-path	In-path	Out-of-path
adder_n118	63.8%	1.8%	59.7%	4.2%
cat_n130	86.5%	1.3%	82.7%	1.9%
ghz_n127	86.5%	1.3%	82.3%	2.0%
ising_n98	92.0%	1.9%	87.2%	7.1%
knn_129	73.0%	2.0%	48.3%	3.8%
multiplier_n75	50.5%	1.7%	51.4%	1.3%
qft_n63	63.4%	1.5%	61.0%	0.7%
qugan_n111	50.6%	3.2%	50.9%	3.2%
swap_test_n115	70.2%	0.9%	55.6%	4.6%
wstate_n118	66.4%	1.2%	81.7%	1.3%

Our experimental results validate this finding through a comparative evaluation of SABRE and our proposed method. We categorize a SWAP operation as "In-path" if it lies on the shortest path between the two qubits of its corresponding two-qubit gate; otherwise, it is classified as "Out-of-path". Using benchmark circuits from QASMBench [48], a widely used quantum circuit benchmarks, we statistically analyze the distribution of In-path versus Out-of-path SWAP selections. As shown in Table I, both methods demonstrate minimal selection

Algorithm 2: `acc-Miltzow`

```
1  Input: Chip G, origin mapping π_i, target mapping π_t;
2  Output: SWAP sequence S;
3  S ← ∅;
4  while True do
5      swap ← get_swap(G, π_i, π_t);
6      if swap is not None then
7          S.push(swap);
8      else
9          happychain ← get_chain(G, π_i, π_t);
10         if happychain is not None then
11             S.push(happychain);
12         else
13             break;
14     Apply swap or happychain in π_i;
15 Return SWAP sequence S.
```

of Out-of-path SWAPs while maintaining a strong preference for In-path SWAPs. Notably, the cumulative percentage of In-path and Out-of-path SWAPs does not reach 100% because some SWAPs may lie on the shortest path for one gate but not for the other.

Therefore, we propose a transformation time optimization method: For each two-qubit gate in the front layer F, we take one qubit of its two qubits as an example (we call the other a partner qubit). We sort the potential SWAP operations associated with this qubit in ascending order by the distance to the partner qubit after the SWAP applied. Only a part of the SWAPs are considered lately, after applied which the distances between the qubit and its partner qubit are less. While this approach resembles the routing heuristics above, it offers computational efficiency by requiring only a single lookup in the precomputed hardware distance matrix. This evaluation involves only the partner qubit rather than the entire front layer F and extended layers E.

The core assumption of this method is that qubit routing paths generally follow the shortest paths, which holds true in most but not all cases. To mitigate this limitation, we incorporate suboptimal paths in practical implementations. In our evaluation, we apply this approach by ranking potential SWAP operations and selecting the top 50% part as candidate operations.

This offers an alternative perspective on the qubit mapping and routing problem: once routing paths are determined, the critical task is to prioritize two-qubit gates in the front layer F. Thus, we introduce the gate priority based method to further reducing transformation time.

The conventional approach to generating candidate SWAP operations involves evaluating SWAP operations linked to all two-qubit gates in the front layer F. We recognize that not all such gates are equally critical. Consequently, we prioritize these two-qubit gates and initially focus on SWAP operations associated with high-priority gates.

When computing gate priorities, we restrict our consideration to the first few circuit layers. This design choice serves two key purposes: (1) it enhances computational efficiency by limiting the analysis scope; (2) it acknowledges the diminishing influence differential of front-layer gates on circuit portions after extended layer(s). Therefore, we modify our approach by counting only successor gates within the extended layer E. The priority of each gate in front layer F increases with its number of successors. After ranking gates using this metric, we selectively process a subset of front-layer and corresponding SWAP operations. The number of selected gates are determined by the gate selection ratio p, which is empirically set in our experiment.

3) Mapping Alignment optimization: As an application of token swapping problem, the mapping alignment problem introduced by dynamically execution of controlled sub-circuits could be handled with an 4-approximation algorithm [27]. The 4-approximation algorithm employs two atomic operations. A swap is classified as an "unhappy swap" if, prior to execution, one token is already in its correct position, and after the swap, the other token's distance to its target position decreases. A sequence of swaps $(v_1, v_2), (v_2, v_3)...(v_l, v_{l+1})$ constitutes a "happy swap chain" if, upon full execution, the token initially at v_1 moves to v_{l+1}, while the remaining tokens shift from v_j to v_{j-1}, with all swapped tokens reducing their distance to their target positions. The algorithm iteratively searches for either happy swap chains or unhappy swaps to execute. If neither is found, all tokens are correctly positioned, and the algorithm terminates. However, this method lacks inherent parallelism optimization, while the computational procedure for identifying happy swap chains is not efficient enough.

We propose `acc-Miltzow`, adapting the original token swapping algorithm to improve both parallelism and efficiency by restricting selection to one SWAP per step. `acc-Miltzow` operates as follows: in each iteration, it first attempts to search either a happy swap or an unhappy swap. If neither is available, it then searches for a happy swap chain. The algorithm terminates when no further operations are applicable. Since a single happy swap inherently qualifies as a happy swap chain [27], this adaptation ensures the algorithm's completeness while maintaining solution validity. The pseudo-code is shown in Algorithm 2. A detailed explanation is listed as follows:

- Firstly, we initialize the return result Swap sequence $\mathcal{S}$ as empty (**line 3**);
- Then, we iteratively attempt to find either a happy swap or an unhappy swap (**line 5∼7**);
- If no suitable swap is found, we then attempt to seach a happy swap chain (**line 9∼11**). If no such chain exists, the loop terminates (**line 13**) and return the results of SWAP sequence $\mathcal{S}$.
- At the end of each iteration, the identified swap or swap chain is applied to update the current qubit mapping π_i (**line 14**).

IV. Evaluation

We evaluate the performance of our proposed method using benchmark dynamic quantum circuits generated by

TABLE II: Main results.

Benchmark name[1]	Baseline				Ours offline[2]				Ours online[2]					
	N[3]	T_{QPU}[4](ms)	RT_{sub}[4](ms)	RT_{full}[4](ms)	N	T_{QPU} (ms)	RT_{full} (ms)	$\Delta_{T_{QPU}}$	N	T_{QPU} (ms)	RT_{sub} (ms)	$\Delta_{T_{QPU}}$	$speedup$	$\Delta_{T_{E2E}}$
QC10-S05-D10-K2	72783	0.66	81.88	155.10	79386	0.36	594.20	46.2%	78750	0.36	21.38	45.6%	3.8	73.7%
QC10-S05-D10-K3	79335	0.76	144.75	218.92	88338	0.44	698.45	42.1%	89943	0.48	40.72	37.3%	3.6	71.7%
QC10-S05-D10-K4	111237	1.06	207.89	294.54	125304	0.64	1320.18	39.7%	125694	0.66	69.91	38.0%	3.0	66.2%
QC10-S05-D20-K2	119373	0.93	112.23	217.80	151350	0.60	1163.26	35.2%	144966	0.62	38.96	32.7%	2.9	65.0%
QC10-S05-D20-K3	158520	1.23	210.70	320.11	199767	0.83	1517.91	32.6%	194742	0.99	82.79	19.4%	2.5	60.5%
QC10-S05-D20-K4	162435	1.43	228.40	364.58	199368	0.86	1910.08	39.9%	204288	0.98	92.05	31.9%	2.5	59.5%
QC10-S10-D20-K2	109026	0.87	103.40	219.51	139608	0.57	1094.75	33.8%	145980	0.63	31.63	27.6%	3.3	69.1%
QC10-S10-D20-K3	154329	1.30	181.83	327.62	184218	0.76	1405.64	41.3%	178938	0.88	76.62	32.1%	2.4	57.7%
QC10-S10-D20-K4	154518	1.29	223.97	376.52	183762	0.79	1491.72	39.0%	189825	0.90	73.21	30.4%	3.1	67.1%
QC15-S05-D10-K2	108000	0.96	111.55	199.40	125661	0.54	983.28	43.5%	123039	0.60	44.99	37.0%	2.5	59.5%
QC15-S05-D10-K3	124353	1.15	164.36	275.62	142062	0.69	1508.12	40.2%	148314	0.76	63.96	34.0%	2.6	60.9%
QC15-S05-D10-K4	142503	1.34	195.98	324.08	153342	0.79	1794.83	40.9%	162492	0.88	72.55	34.4%	2.7	62.8%
QC15-S05-D20-K2	178752	1.38	154.35	317.79	232929	0.93	2019.99	32.4%	238602	1.11	73.76	19.6%	2.1	51.9%
QC15-S05-D20-K3	175083	1.39	170.56	339.58	220488	0.89	1612.90	35.9%	214707	0.92	57.57	33.8%	3.0	66.0%
QC15-S05-D20-K4	225753	1.82	279.19	479.15	272892	1.16	2237.67	36.2%	277698	1.37	130.00	25.1%	2.1	53.3%
QC15-S10-D20-K2	168051	1.29	153.62	339.72	215349	0.89	1779.10	31.1%	216528	0.93	52.74	27.6%	2.9	65.3%
QC15-S10-D20-K3	200184	1.64	207.09	382.37	249924	1.02	1910.72	37.6%	251631	1.18	86.35	27.9%	2.4	58.1%
QC15-S10-D20-K4	221985	1.85	279.28	512.17	266052	1.15	2353.24	38.0%	276846	1.28	106.26	30.6%	2.6	61.7%
QC20-S05-D10-K2	123717	1.08	107.63	211.04	150597	0.69	1002.79	36.0%	149376	0.75	39.99	31.2%	2.7	62.5%
QC20-S05-D10-K3	157221	1.42	184.76	298.43	188598	0.90	1587.28	36.7%	191631	1.00	74.53	29.3%	2.5	59.4%
QC20-S05-D10-K4	170568	1.54	246.29	426.00	198858	0.98	1670.53	36.4%	202977	1.11	97.39	27.9%	2.5	60.3%
QC20-S05-D20-K2	205458	1.53	134.28	337.02	265107	1.05	1893.95	31.1%	264150	1.16	54.70	23.8%	2.5	58.9%
QC20-S05-D20-K3	285981	2.20	273.04	492.35	356037	1.45	2532.21	33.9%	360675	1.70	130.68	22.7%	2.1	51.9%
QC20-S05-D20-K4	285825	2.19	325.49	538.78	333816	1.36	2443.65	37.7%	347514	1.66	132.37	24.4%	2.5	59.1%
QC20-S10-D20-K2	216669	1.67	148.36	324.60	266790	1.06	2117.60	36.4%	268872	1.19	67.58	28.9%	2.2	54.2%
QC20-S10-D20-K3	237645	1.85	198.70	419.55	295884	1.24	2266.44	32.7%	302220	1.35	93.53	26.8%	2.1	52.7%
QC20-S10-D20-K4	253491	2.09	124.25	343.77	325281	1.34	2521.18	35.6%	321285	1.46	112.99	29.9%	1.1	9.4%

Baseline and `Sword` are implemented in Rust and C++, respectively.
[1] The circuit labeled **QC10-S05-D10-K2** indicates a depth of 1000, a QUEKO gate density vector of (0.05, 0.10) [14], and a subcircuit count of 2;
[2] Our results are presented as quantum processing time reduction ratio ($\Delta_{T_{QPU}}$) ,transformation time speedup ($speedup$) and end-to-end wall-clock time reduction ratio ($\Delta_{T_{E2E}}$);
[3] N denotes the number of extra gates;
[4] T_{QPU}, RT_{sub} and RT_{full} denotes the quantum processing time, control sub-circuits transformation time and full circuit transformation time, respectively;

QUEKO [14], a generation tool that can set circuit characteristics (such as depth and gate density), and hardware specifications from the IBM torino chip [49]. Comprehensive results are presented in Sections IV-B and IV-C. Our key findings include:

- In offline scenarios, our approach demonstrates significant quantum processing time T_{QPU} improvements, achieving reductions of up to 46.2% with an average of 37.1%, where end-to-end wall-clock time T_{E2E} consists solely of the T_{QPU} of circuits.
- In online scenarios, our method outperforms baselines in both effectiveness and efficiency, reducing T_{E2E} by an average of 59.2% (maximum 73.7%). Specifically, our algorithm achieves a 2.6× average speedup in transformation time while simultaneously reducing T_{QPU} by 30.0%. In these settings, T_{E2E} comprises both T_{QPU} and transformation time of controlled sub-circuits, highlighting our algorithm's adaptability to dynamic environments.
- The accelerated variant `Sword-fast` of our algorithm exhibits substantial improvements in transformation efficiency, achieving a 9.6× average speedup in transformation time compared to the standard implementation.
- As an improved mapping alignment algorithm, our approach `acc-Miltzow` demonstrates significant enhancements in both solution quality (11.0% reduction in average circuit depth) and computational efficiency (56.4% reduction in computation time).

A. Methodology

Hardware Model: All experiments utilize the IBM torino quantum chip, with data collected on April 5, 2025, which only includes execution times for two-qubit gates. We set the execution time of single-qubit gate to half the average execution time of the corresponding two-qubit gates according to a previous setting [50].

Dataset: Part of our benchmark circuits is from QASMBench [48], a widely used benchmark suite. We also generate quantum circuits by QUEKO [14], which allows us to set circuit parameters such as depth and gate density. The resulting benchmark is both controllable and diverse: it covers dynamic circuits with different depths, gate densities, and branching structures, avoiding bias from relying on a single type of circuit. This provides a more comprehensive validation of our framework's generality.

The generation procedure is as follows: For each circuit, we specify the number of qubits (n), the main circuit depth (D), and the maximum controlled sub-circuits depth ($d, d \ll D$). A total of k sub-circuits are randomly inserted into the main circuit. Both the main circuit and controlled sub-circuits are generated using QUEKO [14]. When introducing control flow, we randomly select classical registers corresponding to measured qubits and attach controlled sub-circuits. Additionally, we assume that measured qubits remain available for subsequent operations. The configurations of dynamic quantum circuit are set as follows:

- Qubit count (n): 133 (matching IBM torino's architecture);
- Main circuit depths (D): 1000, 1500, and 2000;
- Sub-circuit depths (d): randomly sampled from 5 to D/5;
- Sub-circuit counts (k): 2, 3, and 4;
- Gate density vectors (distinct from our algorithm's gate density): (0.05, 0.2), (0.05, 0.1), and (0.1, 0.2) as defined in QUEKO [14].

Evaluation Platform: Our experiments are performed on 2 Intel Xeon Platinum 8360Y @ 2.40GHz, with 1T DDR4 memory. The operating system is ubuntu 18.04.

Algorithm Configuration: We configure the parameters as follows: the weight factor of extended layer(s) in heuristic function $\omega_1 = 0.1$, the weight factor of occupancy $\omega_2 = 1$ ($\omega_2 = 0.1$ when transforming controlled sub-circuits in online scenarios), gate density threshold $r = 0.25$, and set the number

of layers of the extended layer(s) E to 1. In `Sword-fast`, after sorting gates in the front layer F, we select the first 20% gate into consideration ($p = 0.2$) and the top 50% rankings candidate SWAPs. In online scenarios, we process main circuit and controlled sub-circuits with `Sword` and `Sword-fast`, respectively.

Baselines: We use the SABRE algorithm [17] (as implemented in IBM's Qiskit [44]), which achieves a favorable balance between effectiveness and efficiency, with 4-approximation token swapping algorithm [27] as baseline. For the comparisons of quantum processing time and transformation time in Section IV-C, we also compare our methods with TOQM [15]. For both baseline and our method, the initial mapping is set as trivial mapping.

Metrics: For offline scenarios, end-to-end wall-clock time only consists of quantum processing time. We use $\Delta_{T_{E2E}}$ to denote the performance improvements, as $\Delta_{T_{E2E}} = \Delta_{T_{QPU}}$ in this setting. We employ the quantum processing time reduction ratio ($\Delta_{T_{QPU}}$) and transformation time speedup for controlled sub-circuits ($speedup$) to indicate the improvements in effectiveness and efficiency of our method. The specific formulas for calculating these ratios are as follows:

$$\Delta_{T_{QPU}} = 1 - T_{QPU}(Ours)/T_{QPU}(Baseline), \quad (4)$$

$$speedup = RT_{sub}(Baseline)/RT_{sub}(Ours), \quad (5)$$

a higher $\Delta_{T_{QPU}}$ and $speedup$ means a better performance. For online scenarios, end-to-end wall-clock time consists of quantum processing time and transformation time of controlled sub-circuits RT_{sub}. We use $\Delta_{T_{E2E}}$ to denote the performance improvements. For online scenario:

$$\Delta_{T_{E2E}} = 1 - \frac{RT_{sub}(Ours) + T_{QPU}(Ours)}{RT_{sub}(Baseline) + T_{QPU}(Baseline)}, \quad (6)$$

in this setting, a higher $\Delta_{T_{E2E}}$ means a better performance. In our experiments, we assume all sub-circuits execute exactly once (including loop sub-circuits). For the comparison of mapping alignment algorithms, we consider depth as the quality of solutions, which directly cor- responds to execution time given uniform execution time of SWAP operations.

B. Experimental Results

Offline scenarios: Our approach demonstrates significant improvement in end-to-end wall-clock time compared to baseline approaches, as shown in Table II. Specifically, it achieves quantum processing time reductions of up to 46.2% with an average reduction of 37.1%, where end-to-end wall-clock time consists solely of quantum processing time. These results validate our design objectives: by explicitly incorporating both qubits' occupancy and gate density characteristics into the optimization process, our algorithm effectively prioritizes routing paths that minimize execution time. Additionally, our parallelism-promoting mapping alignment strategy contributes substantially to these performance gains.

Online scenarios: Our method achieves quantum processing time reductions of up to 45.6% with an average reduction of 30.0%, while simultaneously accelerating controlled sub-circuits transformation by up to 3.8× with an average speedup of 2.6×. Our optimization strategies yield an average end-to-end wall-clock time reduction of 59.2%, reaching up to 73.7% in maximum cases. These results demonstrate significant reductions in end-to-end wall-clock time for online scenarios, maintaining the quantum processing time advantages observed in offline scenarios while adding substantial improvements in transformation efficiency.

The results indicate that our method provides clear advantages in both execution time and compilation time, with the cost of introducing additional gates. This arises because our mapping strategy prioritizes shorter execution paths and faster runtime over minimizing gate count alone. The trade-off ultimately reduces the circuit's effective execution time, which is one of the key performance metrics in practical scenarios.

C. Sensitivity Analysis

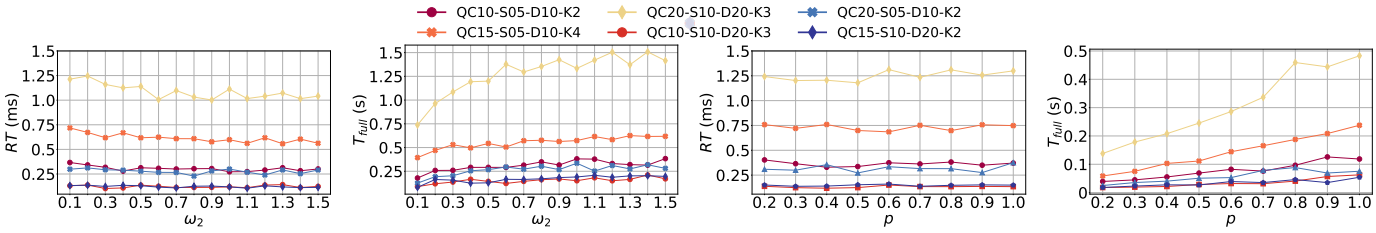
Effectiveness Improvement: The comparative results, presented in Table III, demonstrate that `Sword` achieves a maximum quantum processing time reduction of 45.1% and an average improvement of 33.7% over the baseline. Accelerated version `Sword-fast` also performs well, with averagely 20.2% reduction in quantum processing time.

TABLE III: Comparison of quantum processing time for mapping and routing algorithms.

Benchmark name	Baseline	TOQM	`Sword`		`Sword-fast`	
	T_{QPU} (ms)	T_{QPU} (ms)	T_{QPU} (ms)	$\Delta_{T_{QPU}}$	T_{QPU} (ms)	$\Delta_{T_{QPU}}$
adder_n118	0.19	0.11	0.13	31.7%	0.13	29.2%
dnn_n33	0.03	0.03	0.03	6.9%	0.03	1.5%
multiplier_n75	1.44	1.20	1.24	13.6%	1.41	1.8%
qugan_n111	0.15	0.13	0.10	30.9%	0.15	-2.6%
wstate_n118	0.07	0.03	0.04	34.8%	0.04	36.1%
QC10-S05-D10-K2	0.48	N/A	0.27	43.2%	0.40	16.1%
QC10-S05-D10-K3	0.53	N/A	0.31	41.7%	0.42	21.5%
QC10-S05-D20-K4	1.09	N/A	0.65	40.3%	0.81	25.7%
QC10-S10-D20-K3	1.07	N/A	0.64	39.9%	0.75	30.2%
QC15-S05-D10-K3	0.91	N/A	0.50	45.1%	0.69	23.4%
QC15-S10-D20-K3	1.36	N/A	0.88	35.4%	0.99	27.4%
QC20-S05-D10-K4	1.15	N/A	0.73	37.1%	0.96	16.7%
QC20-S05-D20-K4	1.81	N/A	1.15	36.6%	1.39	23.2%
QC20-S10-D20-K3	1.63	N/A	1.11	31.8%	1.24	23.8%
QC20-S10-D20-K4	1.73	N/A	1.09	37.0%	1.24	28.8%

N/A indicates that the mapper failed to produce a result within one hour.

We additionally investigate the impact of the weight factor ω_2, as presented in Fig. 5a and Fig. 5b. The experimental results reveal a trend that quantum processing time decreases with increasing weight. This behavior likely stems from the gradual suppression of the shortest-path heuristic's dominance by elevated weights, while simultaneously amplifying the effect of physical qubits' occupancy. Such weighting shifts may compromise the primary factor governing heuristic effectiveness. Furthermore, increased weights correlate with prolonged transformation time. As noted above, the reduced emphasis on the shortest-path heuristic promotes the selection of circuitous routes, thereby requiring more SWAP operations. Since our method employs an iterative SWAP selection process, higher weights necessarily increase the transformation time due to the increased iterations.



(a) Impact of ω_2 on T_{QPU} (b) Impact of ω_2 on RT_{full} (c) Impact of p on T_{QPU} (d) Impact of p on RT_{full}

Fig. 5: Impact of the weight factor of occupancy ω_2 and gate selection ratio p.

Efficiency Improvement: As shown in Table IV, compared to Baseline, `Sword-fast` achieves significant improvements in transformation time. We employ the full transformation time reduction ratio ($\Delta_{RT_{full}}$) to indicate the improvement. The specific formula is:

$$\Delta_{RT_{full}} = 1 - RT_{full}(\texttt{Sword-fast})/RT_{full}(Baseline). \tag{7}$$

`Sword-fast` demonstrates an average reduction of 14.1%, with a maximum of 40.5%. Compared to `Sword`, `Sword-fast` achieves an average speedup of 9.6×. For some benchmark circuits, `Sword-fast` exhibits longer transformation times than the baseline. This occurs because although our method reduces the number of candidate SWAP to evaluate, it introduces additional computational overhead, including occupancy of qubits calculations and sorting operations. The overall transformation time improvement reported in Table II incorporates optimizations from token swapping `acc-Miltzow` as well.

TABLE IV: Transformation time comparison for mapping and routing algorithms.

Benchmark name	Baseline	TOQM	`Sword`	`Sword-fast`	
	RT_{full} (ms)	RT_{full} (ms)	RT_{full} (ms)	RT_{full} (ms)	$\Delta_{RT_{full}}$
adder_n118	1.87	5792.04	43.52	2.37	-27.0%
dnn_n33	0.21	4236.09	0.82	0.22	-2.8%
multiplier_n75	8.58	411595.00	55.30	6.06	29.4%
qugan_n111	1.88	23712.80	9.76	1.01	46.3%
wstate_n118	0.42	821.05	0.93	0.30	29.1%
QC10-S05-D10-K2	47.77	N/A	388.28	39.48	17.4%
QC10-S05-D10-K3	59.05	N/A	319.20	36.44	38.3%
QC10-S05-D20-K4	95.13	N/A	855.92	89.91	5.5%
QC10-S10-D20-K3	103.33	N/A	810.32	90.39	12.5%
QC15-S05-D10-K3	89.57	N/A	635.74	53.33	40.5%
QC15-S10-D20-K3	111.57	N/A	1145.79	115.78	-3.8%
QC20-S05-D10-K4	94.45	N/A	804.34	74.73	20.9%
QC20-S05-D20-K4	210.22	N/A	1618.72	165.87	21.1%
QC20-S10-D20-K3	132.08	N/A	1360.28	141.00	-6.8%
QC20-S10-D20-K4	135.08	N/A	1635.29	147.00	-8.8%

N/A indicates that the mapper failed to produce a result within one hour.

We also consider TOQM [15], a mapping algorithm that directly minimizes circuit depth. While TOQM is effective for small and medium scale circuits, it often needs significantly more compilation time and does not scale well to larger (over one hundred qubits and thousands of depth) quantum circuits. Consequently, SABRE remains the primary baseline for online execution due to its balance between compilation speed and solution quality.

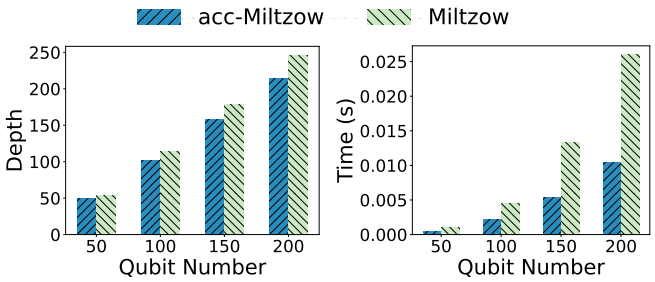
We further examine the effect of varying selection ratios for gates in the sorted front layer F (Figs. 5c and 5d). The results reveal a characteristic trade-off between selection ratio and performance metrics. Specifically, higher selection ratios yield improved quantum processing time but at the cost of increased transformation time. This behavior stems from the expanded candidate SWAP set associated with larger selection ratios. Although this expansion may enhance the quality of selected SWAP operations, it concurrently elevates the computational overhead during each iteration's SWAP evaluation phase. Furthermore, as shown in Figs. 5c and 5d, our results suggests that only a subset of gates are performance-critical during circuit transformation, as optimization efforts beyond these key gates produce diminishing returns. These findings empirically validate the effectiveness of our approach.

Token Swapping: We conduct a comprehensive evaluation of the token swapping algorithm across multiple system scales. Our test configurations employ qubit counts of 50, 100, 150, and 200, with each case generating a corresponding random sparse graph (maintaining vertex degrees not over 4 and uniform edge weights). For each graph configuration, we randomly generate 500 pairs of initial and target mappings, that compare the original token swapping algorithm (denotes as `Miltzow`) [27] with our optimized approach `acc-Miltzow`.

As shown in Fig. 6, our enhanced method achieves simultaneous reductions in both solution depth (which directly corresponds to execution time given uniform edge weights) and computational time. Notably, these performance improvements exhibit strong scaling characteristics, becoming increasingly substantial with larger qubit counts. Our method achieves an average reduction of 11.0% in depth, with a maximum reduction of 13.2%. It also reduced solving time by up to 59.8%, with an average reduction of 56.4%.

V. Conclusion and Discussion

This work introduces a transforming and executing framework designed for dynamic quantum circuits. To optimize both quantum processing time and transformation time, we propose three key strategies. Firstly, we design a mapping and routing algorithm `Sword`, which achieves SWAP with occupancy and gate regional density. Secondly, with analysis of existing mappers, we provide `Sword-fast`, an accelerated version of `Sword`, which significantly scales down the search space, thereby reducing transformation time. Finally, we propose a modified mapping alignment algorithm, optimizing the



(a) Result depth (b) Solving time

Fig. 6: Impact of qubit number on solution depth and solving time for token swapping.

transformation time and resulting circuit depth, based on a 4-approximation algorithm [27]. Numerical tests show that our method significantly improves effectiveness and efficiency for executing and transforming dynamic quantum circuits. For offline scenarios, we reduce end-to-end wall-clock time by an average of 37.1%, with a maximum reduction of 46.2%. In online scenarios, our approach demonstrates a reduction of end-to-end wall-clock time with an average of 59.2%. Specifically, our method achieves an average acceleration of $2.6\times$ in transformation time, while reducing quantum processing time by 30.8% on average. Improved mapping alignment `acc-Miltzow` also indicates a considerable reduction in solving time and the resulting circuit depth.

Acknowledgments

This work was partially supported by the Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0302901), and the National Natural Science Foundation of China (Grant No. 62102388).

References

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6-7, pp. 467–488, 1982.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[4] D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger, "Experimental quantum teleportation," *Nature*, vol. 390, no. 6660, pp. 575–579, 1997.

[5] E. Bäumer, V. Tripathi, D. S. Wang, P. Rall, E. H. Chen, S. Majumder, A. Seif, and Z. K. Minev, "Efficient long-range entanglement using dynamic circuits," *PRX Quantum*, vol. 5, no. 3, p. 030339, 2024.

[6] B. M. Terhal, "Quantum error correction for quantum memories," *Reviews of Modern Physics*, vol. 87, no. 2, pp. 307–346, 2015.

[7] R. Jozsa, "An introduction to measurement based quantum computation," *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment*, vol. 199, pp. 137–158, 2006.

[8] E. Bäumer, V. Tripathi, A. Seif, D. Lidar, and D. S. Wang, "Quantum fourier transform using dynamic circuits," *Phys. Rev. Lett.*, vol. 133, p. 150602, 10 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.133.150602

[9] A. D. Córcoles, M. Takita, K. Inoue, S. Lekuch, Z. K. Minev, J. M. Chow, and J. M. Gambetta, "Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits," *Phys. Rev. Lett.*, vol. 127, p. 100501, 8 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.100501

[10] C. Cao and J. Eisert, "Measurement-driven quantum advantages in shallow circuits," *arXiv preprint arXiv:2505.04705*, 2025.

[11] A. Carrera Vazquez, C. Tornow, D. Riste, S. Woerner, M. Takita, and D. J. Egger, "Combining quantum processors with real-time classical communication," *Nature*, vol. 636, no. 8041, pp. 75–79, 2024.

[12] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder *et al.*, "Encoding a magic state with beyond break-even fidelity," *Nature*, vol. 625, no. 7994, pp. 259–263, 2024.

[13] N. Yanakiev, N. Mertig, C. K. Long, and D. R. Arvidsson-Shukur, "Dynamic adaptive quantum approximate optimization algorithm for shallow, noise-resilient circuits," *Physical Review A*, vol. 109, no. 3, p. 032420, 2024.

[14] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Transactions on Computers*, vol. 70, no. 9, pp. 1363–1373, 2020.

[15] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen, and E. Z. Zhang, "Time-optimal qubit mapping," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 360–374.

[16] H. Fu, M. Zhu, J. Wu, W. Xie, Z. Su, and X.-Y. Li, "Effective and efficient qubit mapper," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[17] 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*, 2019, pp. 1001–1014.

[18] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation as a combination of subgraph isomorphism and token swapping," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–29, 2019.

[19] A. Zulehner, S. Gasser, and R. Wille, "Exact global reordering for nearest neighbor quantum circuits using a*," in *International conference on reversible computation*. Springer, 2017, pp. 185–201.

[20] R. S. Gupta, E. Van Den Berg, M. Takita, D. Riste, K. Temme, and A. Kandala, "Probabilistic error cancellation for dynamic quantum circuits," *Physical Review A*, vol. 109, no. 6, p. 062617, 2024.

[21] M. Tuokkola, Y. Sunada, H. Kivijärvi, J. Albanese, L. Grönberg, J.-P. Kaikkonen, V. Vesterinen, J. Govenius, and M. Möttönen, "Methods to achieve near-millisecond energy relaxation and dephasing times for a superconducting transmon qubit," *Nature Communications*, vol. 16, no. 1, p. 5421, 2025.

[22] M. Bal, A. A. Murthy, S. Zhu, F. Crisa, X. You, Z. Huang, T. Roy, J. Lee, D. v. Zanten, R. Pilipenko *et al.*, "Systematic improvements in transmon qubit coherence enabled by niobium surface encapsulation," *npj Quantum Information*, vol. 10, no. 1, p. 43, 2024.

[23] S. Ganjam, Y. Wang, Y. Lu, A. Banerjee, C. U. Lei, L. Krayzman, K. Kisslinger, C. Zhou, R. Li, Y. Jia *et al.*, "Surpassing millisecond coherence in on chip superconducting quantum memories by optimizing materials and circuit design," *Nature Communications*, vol. 15, no. 1, p. 3687, 2024.

[24] Y. Salathé, P. Kurpiers, T. Karg, C. Lang, C. K. Andersen, A. Akin, S. Krinner, C. Eichler, and A. Wallraff, "Low-latency digital signal processing for feedback and feedforward in quantum computing and communication," *Phys. Rev. Appl.*, vol. 9, p. 034011, 5 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.9.034011

[25] L. Caune, L. Skoric, N. S. Blunt, A. Ruban, J. McDaniel, J. A. Valery, A. D. Patterson, A. V. Gramolin, J. Majaniemi, K. M. Barnes *et al.*, "Demonstrating real-time and low-latency quantum error correction with superconducting qubits," *arXiv preprint arXiv:2410.05202*, 2024.

[26] N. Fruitwala, G. Huang, Y. Xu, A. Rajagopala, A. Hashim, R. K. Naik, K. Nowrouzi, D. I. Santiago, and I. Siddiqi, "Distributed architecture for fpga-based superconducting qubit control," *arXiv preprint arXiv:2404.15260*, 2024.

[27] T. Miltzow, L. Narins, Y. Okamoto, G. Rote, A. Thomas, and T. Uno, "Approximation and hardness for token swapping," *arXiv preprint arXiv:1602.05150*, 2016.

[28] W. Fang, M. Ying, and X. Wu, "Differentiable quantum programming with unbounded loops," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 1, pp. 1–63, 2023.

[29] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1001–1016.

[30] A. Molavi, A. Xu, M. Diges, L. Pick, S. Tannu, and A. Albarghouthi, "Qubit mapping and routing via maxsat," in *2022 55th IEEE/ACM*

International Symposium on Microarchitecture (MICRO). IEEE, 2022, pp. 1078–1091.
[31] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.
[32] A. Zulehner, A. Paler, and R. Wille, "Efficient mapping of quantum circuits to the ibm qx architectures. in 2018 design, automation test in europe conference exhibition (date)," 2018.
[33] A. Sinha, U. Azad, and H. Singh, "Qubit routing using graph neural network aided monte carlo tree search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 9935–9943.
[34] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," *ACM Transactions on Quantum Computing*, vol. 3, no. 2, pp. 1–25, 2022.
[35] F. Wagner, A. Bärmann, F. Liers, and M. Weissenbäck, "Improving quantum computation by optimized qubit routing," *Journal of Optimization Theory and Applications*, vol. 197, no. 3, pp. 1161–1194, 2023.
[36] J. Liu, E. Younis, M. Weiden, P. Hovland, J. Kubiatowicz, and C. Iancu, "Tackling the qubit mapping problem with permutation-aware synthesis," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 745–756.
[37] C.-Y. Cheng, C.-Y. Yang, Y.-H. Kuo, R.-C. Wang, H.-C. Cheng, and C.-Y. R. Huang, "Robust qubit mapping algorithm via double-source optimal routing on large quantum circuits," *ACM Transactions on Quantum Computing*, vol. 5, no. 3, Sep. 2024. [Online]. Available: https://doi.org/10.1145/3680291
[38] S. Li, K. D. Nguyen, Z. Clare, and Y. Feng, "Single-qubit gates matter for optimising quantum circuit depth in qubit mapping," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.
[39] G. Padda, E. Tham, A. Brodutch, and D. Touchette, "Improving qubit routing by using entanglement mediated remote gates," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2024, pp. 1770–1776.
[40] C.-Y. Huang and W.-K. Mak, "Efficient qubit routing using a dynamically extract-and-route framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 10, pp. 2978–2989, 2024.
[41] V. Saravanan and S. M. Saeed, "Noise adaptive quantum circuit mapping using reinforcement learning and graph neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 5, pp. 1374–1386, 2024.
[42] Y. Ji, X. Chen, I. Polian, and Y. Ban, "Algorithm-oriented qubit mapping for variational quantum algorithms," *Phys. Rev. Appl.*, vol. 23, p. 034022, 5 2025. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.23.034022
[43] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta *et al.*, "Openqasm 3: A broader and deeper quantum assembly language," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, pp. 1–50, 2022.
[44] ibm. (2025) Ibm quantum computing. Accessed: 2025-04-13. [Online]. Available: https://www.ibm.com/quantum/qiskit
[45] K. Yamanaka, E. D. Demaine, T. Ito, J. Kawahara, M. Kiyomi, Y. Okamoto, T. Saitoh, A. Suzuki, K. Uchizawa, and T. Uno, "Swapping labeled tokens on graphs," *Theoretical Computer Science*, vol. 586, pp. 81–94, 2015.
[46] J. Kawahara, T. Saitoh, and R. Yoshinaka, "The time complexity of the token swapping problem and its parallel variants," in *WALCOM: Algorithms and Computation: 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29–31, 2017, Proceedings 11*. Springer, 2017, pp. 448–459.
[47] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, pp. 345–345, 1962.
[48] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, pp. 1–26, 2023.
[49] ibm. (2025) Ibm quantum platform. Accessed: 2025-04-5. [Online]. Available: https://quantum.ibm.com/
[50] H. Deng, Y. Zhang, and Q. Li, "Codar: A contextual duration-aware qubit mapping for various nisq devices," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

Fangzheng Chen received his B.S. degree in Mechatronic Engineering from the Suzhou University, Suzhou, China, in 2022. He is currently a master candidate at the School of Computer Science and Technology, University of Science and Technology of China, Hefei. His research interests include quantum algorithms and quantum information theory.

Hao Fu received his Bachelor's degree in Software Engineering from Fuzhou University, China, in 2017. He is currently pursuing his Ph.D. at the University of Science and Technology of China. His research interests include quantum architecture, quantum circuit optimization, and quantum-classical hybrid systems.

Mingzheng Zhu received her B.S. degree in Information and Software Engineering from the University of Electronic Science and Technology of China, Sichuan, China, in 2019. She is currently pursuing a Ph.D. in Computer Science and Technology at the University of Science and Technology of China, Hefei. Her research interests focus on quantum error correction and quantum architecture.

Chi Zhang is an associate professor in Hefei University of Technology. He received his B.S. degree in Computer Science and Technology from the University of Science and Technology of China (USTC) with the honor of The Talent Program in Computer and Information Science and Technology in 2017 and got his Ph.D. degree in Computer Science and Technology from USTC in 2023. He is currently an associate professor at Hefei University of Technology. His main research interests are data center networking, cloud computing, and algorithms.

Wei Xie is an Associate Professor. He obtained his Bachelor's degree from Nanjing University in 2011, Master's degree from Tsinghua University, in 2013 and Ph.D. from the University of Technology Sydney in 2020. During his Ph.D. studies, he was awarded the UTS President's Scholarship (UTSP) and the International Research Scholarship. In July 2020, he joined the School of Computer Science and Technology at the University of Science and Technology of China. His research interests primarily focus on quantum computing and quantum information. He has published several papers in renowned journals and conferences such as IEEE TIT, PRA, Conference on QIP, QIC, IEEE ISIT, and AQIS.

Xiang-Yang Li (Fellow, IEEE) is a full professor and executive dean at the School of Computer Science and Technology, USTC, Hefei, China. He was a full professor at Illinois Institute of Technology, Chicago, USA. He is an ACM/IEEE Fellow and an ACM Distinguished Scientist. Dr. Li received an MS (2000) and a PhD (2001) degree at the Department of Computer Science from the University of Illinois at Urbana-Champaign, a Bachelor's degree at the Department of Computer Science, and a Bachelor's degree at the Department of Business Management from Tsinghua University, both in 1995. His research spans Artificial Intelligent Internet of Things, mobile computing, data sharing and trading, and privacy. He published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications".