

# Quantum Circuit Compilation for Superconducting Bus-Resonator Architectures

Patrick Hopf<sup>\*†</sup>, Lukas Burgholzer<sup>\*†</sup> Robert Wille<sup>\*†‡</sup>,

<sup>\*</sup>Munich Quantum Software Company (MQSC), Garching near Munich, Germany

<sup>†</sup>Technical University of Munich, Munich, Germany

<sup>‡</sup>Software Competence Center Hagenberg GmbH, Hagenberg, Austria

{patrick, lukas, robert}@munichquantum.software  
www.munichquantum.software

**Abstract**—Superconducting quantum computers are fundamentally limited by restricted qubit connectivity. Bus-resonator architectures alleviate this constraint by enabling effective all-to-all interactions. This advantage, however, comes at the cost of significant operational overhead. Realizing the full potential of such hardware thus requires sophisticated compilation techniques that minimize this overhead. In this work, we present the first formalization of the underlying compilation problem for bus-resonator architectures amenable to so-called SAT-CP solvers. This formalization yields *optimal* solutions for small quantum circuits. For larger instances, we propose a linear-time heuristic. Experimental evaluations confirm that the formalization makes it possible to find optimal solutions even in vast search spaces and that the heuristic provides near-optimal compilation while scaling efficiently to circuits of practical size. Together, these contributions establish both a rigorous baseline and a practical path toward low-overhead compilation for superconducting bus-resonator devices.

**Index Terms**—quantum computing, compiler, superconducting, bus-resonator, quantum circuit compilation.

## I. INTRODUCTION

In today’s quantum hardware landscape, superconducting qubits [1] represent one of the leading platforms, alongside a variety of other technologies, each with its own distinct advantages and trade-offs. A central challenge for superconducting architectures is their typically limited qubit connectivity, which contrasts with technologies such as trapped ions or neutral atoms that offer all-to-all connectivity [2]–[5]. Recent hardware developments have addressed this challenge through the use of a *bus resonator* [6]–[9], as exemplified by IQM’s star architecture [10], [11], which enables effective all-to-all connectivity among the qubits attached to the resonator.

While the bus-resonator design alleviates connectivity constraints, its practical use introduces a non-negligible overhead through so-called MOVE operations for loading and unloading the resonator. In fact, MOVE durations are comparable to those of native two-qubit entangling gates. Given the relatively short coherence times of superconducting qubits, any such load-unload sequence can become a dominant contributor to error accumulation. Consequently, harnessing the full potential of the architecture requires a software-side compilation strategy that minimizes the MOVE overhead while remaining computationally efficient.

To this end, this work makes three contributions. First, we provide the first formalization of the underlying compilation problem for bus-resonator architectures, rendering it amenable to exact methods (solvable via so-called SAT-CP techniques) and thereby yielding *optimal* solutions. Second, we introduce a scalable heuristic that operates in linear time, effectively

reducing the number of required MOVE operations in practice. Third, through an experimental evaluation on a diverse benchmark suite, we show that the exact method quantifies the headroom available beyond straightforward compilation methods, while the heuristic captures much of this headroom and scales to circuit sizes of practical relevance.

Ultimately, the proposed formalization and optimal solution establish a rigorous baseline for the future development of compilation strategies tailored to superconducting bus-resonator architectures, while the proposed heuristic is a first step toward leveraging this technology for executing circuits of practical scale.

## II. BACKGROUND

To keep this paper self-contained, this section briefly reviews the basic aspects of quantum circuit compilation, with a focus on superconducting qubits and the recently introduced bus-resonator architectures.

### A. Quantum Circuit Compilation

To run a general quantum program—typically described as a quantum circuit, i.e., a sequence of (possibly parameterized) gates acting on one or more qubits—its constituent operations must be translated into a form that can be executed on the targeted hardware. This process, known as *quantum circuit compilation*, is realized through a sequence of transformation passes that are iteratively applied to the circuit.

In the context of superconducting architectures, two fundamental steps are required: *circuit synthesis* [12]–[17], which ensures that the program contains only operations supported by the hardware, and *circuit mapping* [18]–[26], which ensures that the circuit satisfies the hardware’s connectivity constraints (i.e., contains only gates that can be directly executed on physical qubits).

In architectures with limited nearest-neighbor connectivity, mapping typically involves SWAP-based routing, where arbitrary qubit interactions are made possible by inserting consecutive SWAP operations that exchange the logical states of two adjacent qubits.

Since every quantum operation is affected by noise, the fidelity of a computation decreases with each additional gate. Compilation therefore aims to optimize figures of merit such as the circuit depth, the total number of gates, or related metrics, which serve as cost functions for evaluating circuit configurations without execution [25], [27], [28]. In particular, minimizing fidelity-reducing operations such as SWAPs helps limit error accumulation and improves overall computational fidelity.

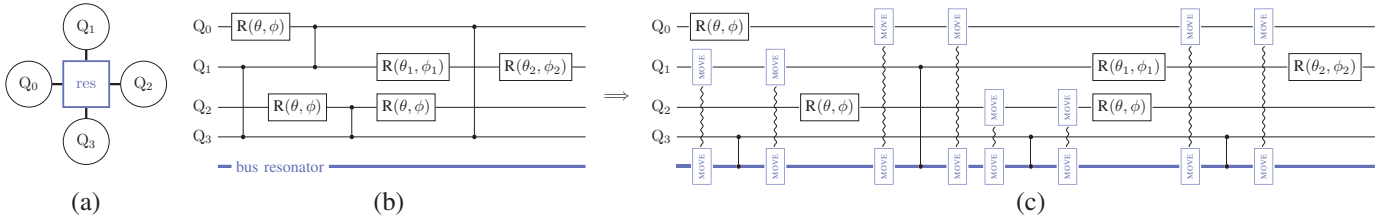


Fig. 1: (a) *Bus-resonator* architecture providing CZ gates between resonator and connected qubits. (b) Circuit requiring effective all-to-all connectivity among four qubits. (c) Compiled circuit with MOVE operations using the bus to mediate CZ gates.

### B. Bus-Resonator Architecture

In an effort to address the costly connectivity constraints that necessitate SWAP-based compilation methods, a class of superconducting architectures employing a *central* or *bus resonator* has recently gained popularity [6]–[9]. In 2024, IQM introduced its star architecture with frequency-tunable qubits linked to a bus resonator via tunable couplers [10] and subsequently demonstrated error-detection capabilities on a six-qubit device [11]. In analogy to a data bus transferring qubit excitations into and out of the central resonator, a resonator of this kind (illustrated in Fig. 1a) can mediate native two-qubit entangling gates between any pair of connected qubits, thereby providing effective all-to-all connectivity among them.

To perform such a mediated gate, one of the participating qubits must first be *moved* into the resonator—an operation that can be modeled as a MOVE operation. After the desired two-qubit gates have been executed, the corresponding qubit must be moved back, thereby enabling subsequent operations on distinct qubit pairs as well as single-qubit gates, which can only be performed *outside* the resonator.

In contrast to a conventional SWAP operation, which exchanges the quantum states of two qubits, a MOVE operation transfers a single quantum state between its original qubit and the connected bus resonator. Conceptually, this mechanism resembles the shuttling of qubits into interaction zones in all-to-all connectivity platforms, such as trapped ions or neutral atoms.

**Example 1.** Consider the circuit depicted in Fig. 1b, which involves a sequence of CZ gates between all qubits and, hence, requires effective all-to-all connectivity. A straightforward compilation strategy, making use of the bus-resonator architecture reviewed above, is to move the lower-indexed qubit into the resonator for each two-qubit gate. Once another CZ gate requires use of the resonator or a single-qubit gate must be applied to the resonator qubit, it is moved back to its original position. This procedure yields the compiled circuit shown in Fig. 1c.

The example illustrates how the bus resonator can be leveraged to enable all-to-all connectivity among its surrounding qubits. Such an architecture eliminates the need for costly SWAP operations, which are typically realized using three consecutive entangling gates together with additional single-qubit rotations and can, therefore, significantly reduce computational fidelity. Yet, it introduces an overhead in the form of MOVE operations, which themselves are non-negligible.

### III. CONSIDERED PROBLEM

With the advent of the bus-resonator architecture, all-to-all connectivity among qubits can be achieved without relying on costly SWAP operations. However, this approach requires

the use of MOVE operations, which—although typically less costly than SWAP operations—still introduce a non-negligible overhead. The architecture described in [10] implements a MOVE operation with a duration of 89.33 ns, which is nearly identical to the 90.67 ns required for a direct CZ gate. In the worst case, a single mediated CZ (load–CZ–unload) may last up to three times longer than a direct CZ gate.

Consequently, the new hardware capabilities can only be fully exploited if the overhead of required MOVE operations is kept small. This necessitates the development of new compilation methods, such as those proposed in this work.

To this end, we consider the native gate set demonstrated in IQM’s bus-resonator architecture [10], consisting of single-qubit rotations  $R(\theta, \phi)$  and the CZ gate, which together form a universal set and can thus be assumed without loss of generality. Based on that, the following properties can be leveraged during compilation:

- P1** Consecutive rotation gates acting on the same qubit can be merged into a single  $2 \times 2$  unitary, which in turn can be expressed as a single rotation gate with updated effective angles. This enables compression of quantum circuits through a straightforward single sweep that combines consecutive rotations on each qubit.
- P2** Consecutive CZ gates commute provided that no single-qubit gate is applied in between. This allows reordering of CZ gates within a *patch*, defined as a maximal sequence of consecutive gates of the same type with no intervening gates of a different type acting on any of the involved qubits.

The following example illustrates how these properties can be utilized to keep the number of MOVE operations small.

**Example 2.** Consider again the circuit from Example 1 shown in Fig. 2a. The circuit is first compressed (using property P1), resulting in an effective rotation  $R(\theta_{\text{eff}}, \phi_{\text{eff}})$  on qubit  $Q_1$  (see Fig. 2b). Next, the maximal patch of commuting CZ gates is identified (see Fig. 2b), within which the execution order can be freely chosen (using property P2). This flexibility is then exploited by swapping the orange and purple gates, thereby grouping together the maximum number of gates that act on the same qubit. Consequently, inserting the MOVE operations in accordance with this gate order (taking advantage of the shared resonator qubit) reduces the number of MOVE operations by half, from 8 (cf. Fig. 1c) to 4 (cf. Fig. 2c).

Property P1 can be straightforwardly realized as a (linear-time) preprocessing step. In contrast, exploiting property P2 in practice is significantly more challenging. Finding the optimal order of CZ gates (without violating the circuit’s partial order) that minimizes the number of MOVE operations is a non-trivial combinatorial optimization problem with a super-exponential search space.

This work addresses the challenge with the following contributions:

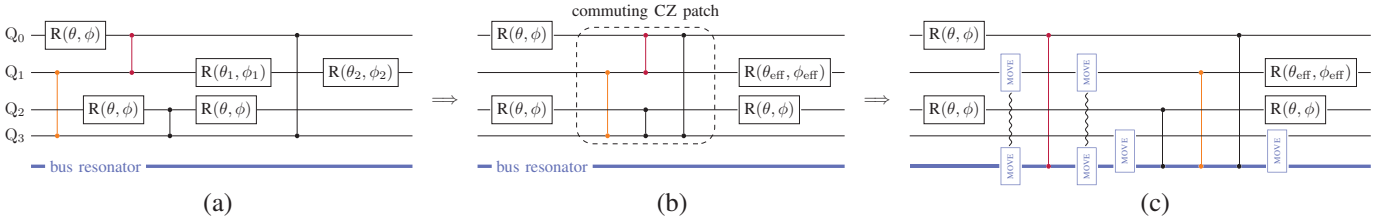


Fig. 2: (a) Circuit requiring effective all-to-all connectivity among four qubits. (b) Circuit after compressing consecutive rotation gates and identifying a maximal patch of commuting CZ gates. (c) Compiled circuit with MOVE operations taking advantage of the reordered CZ gates.

- A comprehensive symbolic formalization of the underlying compilation problem for bus-resonator architectures, which allows the problem to be solved in an exact (i.e., optimal) fashion;
- a scalable heuristic method that allows large circuits to be tackled in a practical runtime; and
- an experimental evaluation and benchmarking of the proposed methods.

#### IV. OPTIMAL SOLUTION

To address the problem of finding an ordering of CZ gates that minimizes the number of MOVE operations, we introduce a symbolic formulation that enables an exact optimization strategy, as shown for similar compilation problems, e.g., in [3], [29]–[31]. Such exact optimization is useful for small to medium instances, where it reveals the structure of the search space and quantifies the potential for improvement. Its main purpose is to provide a precise baseline against which heuristic compilation methods—designed to scale to large circuits—can be evaluated.

##### A. Symbolic Formulation

To capture all possible gate orderings for a given quantum circuit on the bus-resonator architecture in a symbolic form, we introduce two main variables. Let  $N$  denote the number of gate operations and  $M$  the number of qubits in the architecture. Furthermore, assume that each gate is executed in a distinct timestep  $t \in \{0, \dots, N-1\}$ , with parallelism recoverable through methods such as as-soon-as-possible scheduling. Then, the following decision variables are defined:

- $\pi : \{U_0, \dots, U_{N-1}\} \rightarrow \{0, \dots, N-1\}$ , a permutation variable assigning each of the  $N$  unitary gates to a unique execution slot. One can think of  $\pi$  as a temporal ordering of the gates, where  $\pi(U_i) = t$  means gate  $U_i$  is executed at timestep  $t$ .
- $s = (s_0, s_1, \dots, s_{N-1})$ , a sequence of state variables representing the content of the resonator, with domain  $s_t \in \{-1, 0, \dots, M-1\}$ . Here,  $s_t = -1$  denotes that the resonator is empty at timestep  $t$ , while  $s_t = i$  denotes that the resonator holds qubit  $Q_i$ . This variable is required in addition to  $\pi$ , since the temporal ordering alone does not uniquely determine the state of the full system.

**Example 3.** Consider the circuit and exemplary variable assignment shown in Fig. 3. This assignment corresponds to  $\pi(R(\theta_1, \phi_1)) = 0$ ,  $\pi(\text{CZ}) = 1$ , and  $\pi(R(\theta_2, \phi_2)) = 2$ , i.e., the two rotation gates and the CZ gate are executed at timesteps  $t = 0$ , as well as  $t = 2$  and  $t = 1$ , respectively. In addition, the assignment of the  $s_t$  variable indicates that the resonator is empty at timesteps  $t = 0$  as well as  $t = 2$ , and holds qubit  $Q_0$  at timestep  $t = 1$ . Together, this fully captures one possible compiled quantum circuit. Note that choosing  $\pi(\text{CZ}) = 2$

would yield a non-equivalent quantum circuit, since the CZ gate does not commute with the surrounding rotation gates.

Such a formulation enables a symbolic description of all possible circuit realizations. In fact, there are  $N!$  permutations  $\pi$ , and each state variable  $s_t$  can take  $M+1$  values, resulting in a total of  $N! \cdot (M+1)^N$  possible assignments. The task is therefore to determine an assignment that is valid and, among all valid candidates, yields the minimal overhead in terms of MOVE operations.

##### B. Constraints

In order to allow only *valid* assignments, i.e., assignments that correspond to circuit configurations that are equivalent to the original circuit and correctly capture the resonator state, the following constraints must be imposed, defining the feasible region:

- **Gate ordering:**  $\pi$  must be a valid topological sort of any circuit DAG consistent with allowed CZ gate commutations.
- **CZ gate constraint:** if operation  $\pi^{-1}(t)$  is a CZ gate acting on qubits  $q_1, q_2$ , then  $s_t \in \{q_1, q_2\}$ .
- **Single-qubit gate constraint:** if operation  $\pi^{-1}(t)$  is a single-qubit gate acting on qubit  $q$ , then  $s_t \neq q$ .

These constraints significantly reduce the size of the feasible search space. The number of valid permutations  $\pi$  is limited by the circuit's partial order, which typically yields far fewer total orders than  $N!$ . For a given permutation  $\pi$ , the domain of  $s_t$  is much more restricted for an individual timestep  $t$ : it excludes one value in the case of single-qubit gates, and is limited to just two values when  $\pi^{-1}(t)$  is a CZ gate. The structured constraints allow effective search space pruning via constraint propagation in so-called SAT-CP solvers such as [32], [33].

However, among the many valid assignments, only a small subset is desirable—namely those that minimize the number of MOVE operations. This criterion can be enforced through an objective function to be minimized.

##### C. Objective Function

To model the cost of resonator usage—and thereby enabling minimization of the number of MOVE operations—two auxiliary variables are defined at each timestep  $t$ . The first variable

$$\text{unload}_t = \begin{cases} 1 & \text{if } s_t \neq -1 \wedge s_t \neq s_{t+1} \\ 0 & \text{otherwise,} \end{cases} \quad \text{for } t < N-1,$$

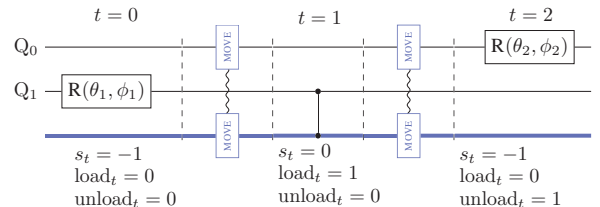


Fig. 3: Exemplary variable assignment.

captures the situation where a previously held qubit must be removed from the resonator. Such removal is required either to return the qubit to its original place before a single-qubit gate can be applied or to free the resonator in preparation for loading a different qubit for another CZ gate. This motivates the second variable

$$\text{load}_t = \begin{cases} 1 & \text{if } s_t \neq -1 \wedge s_{t-1} \neq s_t \quad \text{for } t > 0, \\ 0 & \text{otherwise,} \end{cases}$$

which accounts for loading a new qubit into the resonator. The boundary conditions of loading and unloading the resonator at the beginning and end of the computation are handled explicitly:  $\text{load}_0 = 1$  if  $s_0 \neq -1$  (and 0 otherwise), and  $\text{unload}_{N-1} = 1$  if  $s_{N-1} \neq -1$  (and 0 otherwise).

The total number of physical MOVE operations for a given permutation  $\pi$  with resonator states  $s$  is then given by

$$\text{moves}_{\pi,s} = \sum_{t=0}^{N-1} (\text{unload}_t + \text{load}_t),$$

which is the expression to be minimized in the objective function.

**Example 4.** *It can easily be seen that  $\text{moves}_{\pi,s} = 2$  correctly reflects the number of necessary MOVE operations for the variable assignment illustrated in Fig. 3.*

The objective is linear in the number of operations and well-suited for SAT-CP solvers. Such a constrained combinatorial optimization problem can be modeled using the following components:

- Integer variables for gate positions  $\pi(i)$ , with an *all-different constraint* and topological order constraints.
- Integer variables  $s_t \in \{-1, 0, \dots, M-1\}$  with dynamically restricted domains.
- Boolean reified variables for load/unload events and their linear combination in the objective.

Although the structured nature of quantum circuits and the specific constraints imposed by the resonator architecture can be leveraged, the problem remains computationally challenging, particularly for larger instances with many admissible permutations.

## V. HEURISTIC SOLUTION

To address the computational overhead of determining an optimal CZ permutation, this section introduces a scalable heuristic compilation strategy for minimizing MOVE operations on bus-resonator architectures. Prior to this, we outline the vendor-provided solution.

### A. Vendor-Provided Solution

Accompanying IQM’s novel bus-resonator star architecture, the vendor provides software that ensures the practical operability of its devices. In this implementation, MOVE insertion is carried out using a heuristic strategy that processes the ordered sequence of gates in a single pass.

For each gate, a *badness score* is calculated for all of its possible *resolutions*, i.e., the alternative physical realizations of the gate that involve the resonator. Each CZ gate admits two such resolutions. The badness score is then computed with respect to the current resolution state as follows:

- +0 if one of the two qubits is already in the resonator,
- +1 if a qubit needs to be moved into the resonator, and
- +1 if a qubit needs to be moved out of the resonator.

Subsequently, for each resolution, the compiler looks ahead and identifies the gates that act next on the target qubit(s). It then calculates the *follower badness* (again considering all possible resolutions) using the same rules. Finally, the resolution with the lowest overall badness score—considering both the current gate and its immediate followers—is selected.

**Example 5.** *Consider again the circuit in Fig. 2a, where the orange CZ<sub>1</sub> precedes the purple CZ<sub>2</sub>. Evaluating the badness for the CZ<sub>1</sub> gate yields two resolution options, both with a badness of +1 (moving either Q<sub>1</sub> or Q<sub>3</sub> into the resonator and keeping it there for the following CZs). Then the badness evaluation for the CZ<sub>2</sub> gate yields:*

- Move Q<sub>0</sub> into the resonator (+1) and keep it for the following CZ gate (+0), for a total badness of +1.
- Move Q<sub>1</sub> into the resonator (+1) and move it out again for the following rotation gate (+1), for a total badness of +2.

In terms of complexity, this approach is linear in the number of gates, as it requires only a single pass over the circuit. For each gate, at most two possible resolutions are considered, and for each of these, up to two follower resolutions are evaluated. The only overhead arises from the implemented lookahead procedure used to locate the next gate acting on the involved qubits, which is itself linear in the number of gates.

Even though the approach finds locally good resolutions, it fails to account for the global structure of the circuit. This results in a considerable number of MOVE operations, ultimately negating the computational effort spent evaluating local badness. To address this, a less locally constrained strategy is needed.

### B. Proposed Approach

To achieve a more globally informed yet computationally efficient compilation for bus-resonator architectures, a greedy two-pass heuristic is proposed.

In the first pass, Algorithm 1 identifies patches of commuting CZ gates and interleaves them with pure rotation gate patches. This step runs in linear time and maintains a set (per patch) of buckets—one per qubit—tracking all gates acting on that qubit.

The second pass processes each CZ patch individually using Algorithm 2, which greedily selects an execution order for the commuting CZ gates. The ordering accounts for both the preceding resonator assignment and the qubits of the next CZ patch, aiming to maximize consecutive gates acting on the same qubits.

The overall complexity of the heuristic is linear in the number of gates, as both passes iterate exactly once over every gate in the circuit.

**Example 6.** *Application of Algorithm 1 turns the circuit from Fig. 2a into a patch of four commuting CZ gates shown in Fig. 2b. With no previously assigned  $q_{res}$  and no upcoming  $q_{next}$ , Algorithm 2 begins by assigning the last qubit to the bucket with the most gates—that is, all CZ gates involving Q<sub>3</sub>. This leaves only the blue CZ gate involving Q<sub>0</sub> for the first position, resulting in the MOVE assignment shown in Fig. 2c.*

The example demonstrates how the proposed heuristic identifies commuting CZ gate patches and determines a beneficial ordering, which reduces the number of MOVE operations. Its practical performance is assessed in the next section.

---

**Algorithm 1** Circuit patching

---

**Input:** Ordered list of gates  $\mathbf{G} = [g_0, g_1, \dots, g_{N-1}]$ , with each  $g_i$  either a CZ or R gate.

**Output:** Ordered list of patches  $\mathbf{P} = [p_1, p_2, \dots, p_{K < N}]$ , each containing a list of buckets. Each  $p_i$  corresponds to a pure CZ or R patch and holds gates grouped into buckets, sorted by the qubits they act on within the patch.

- 1:  $P \leftarrow [[q_1 : \emptyset, \dots, q_M : \emptyset], [\dots], [\dots]] \triangleright M$  qubits
- 2:  $last \leftarrow [0, 0, \dots, 0]$ ,  $next \leftarrow [0, 0, \dots, 0] \triangleright M$  entries
- 3: **for each**  $g_i$  in  $\mathbf{G}$  **do**
- 4:    $q_0, q_1 \leftarrow \text{GetQubits}(g_i)$
- 5:   **if**  $g_i$  is CZ **then**
- 6:      $idx = \max(last[q_0], last[q_1]) + 1$
- 7:      $next[q_0] \leftarrow \max(idx + 1, next[q_0])$
- 8:      $next[q_1] \leftarrow \max(idx + 1, next[q_1])$
- 9:      $P[idx][q_0] \leftarrow P[idx][q_0] \cup \{g_i\}$
- 10:     $P[idx][q_1] \leftarrow P[idx][q_1] \cup \{g_i\}$
- 11:   **else**  $\triangleright g_i$  is R, i.e.,  $q_0 = q_1$
- 12:      $idx = q_{idx}[q_0]$
- 13:     **if**  $idx \neq last[q_0]$  **then**
- 14:        $last[q_0] \leftarrow idx$
- 15:     **end if**
- 16:      $P[idx][q_0] \leftarrow P[idx][q_0] \cup \{g_i\}$
- 17:   **end if**
- 18: **end for**

---

## VI. EVALUATION

With a symbolic formalization amenable to exact optimization, it is now possible to determine *optimal* solutions and, with that, rigorously evaluate compilation strategies for bus-resonator architectures. Additionally, the proposed heuristic promises near-optimal results in scalable (linear-complexity) runtime. In this section, we evaluate the performance of the proposed solutions and demonstrate that the expectations are met in practice.

## A. Evaluation Setup

The evaluation uses a large set of circuits from MQT Bench<sup>1</sup> [34] (v2.0), part of the Munich Quantum Toolkit [35], which provides 29 algorithms scaled between 3 and 24 qubits. The lower end of this range (3 qubits) corresponds to the smallest instances where MOVE-considering compilation can have an effect, while the upper end (24 qubits) reflects benchmarks that can be executed on the largest commercially available<sup>2</sup> bus-resonator architecture to date.

To isolate the compilation impact on MOVE counts, all circuits were first precompiled into the architecture's native R and CZ gate set and compressed according to property P1 (cf. Section IV) using Qiskit's `transpile` function. Circuits exceeding 1,000 gates after preprocessing were discarded, as current hardware does not yet provide meaningful results for such sizes.

The optimal solution was computed using the Google OR-Tools (v9.14) SAT-CP solver [32] with a time limit of three hours per circuit on an AMD processor with 16 cores and 128 GB RAM. The heuristic was evaluated on a standard laptop. All results and an open-source implementation are publicly available at Zenodo<sup>3</sup>.

<sup>1</sup><https://github.com/munich-quantum-toolkit/bench><sup>2</sup><https://resonance.meetiqm.com><sup>3</sup><https://doi.org/10.5281/zenodo.18223346>

---

**Algorithm 2** Reorder patch of commuting CZ gates

---

**Input:** A CZ patch  $\mathbf{p} = [q_1 : \{g_i, \dots\}, \dots, q_M : \{g_j, \dots\}]$ , i.e., a list of buckets with  $i, j \in [0, N - 1]$ . The next CZ patch qubits  $q_{next}$ . Previous resonator qubit  $q_{res} \in Q = \{q_1, q_2, \dots, q_M\} \supseteq q_{next}$ .

**Output:** Ordered list of gates **ordered**.

- 1: **if**  $q_{res} \in \mathbf{p}$  **then**
- 2:    $first \leftarrow q_{res}$
- 3:    $seen \leftarrow p[first]$
- 4:    $last \leftarrow \arg \max_{q \in Q} |p[q] \setminus seen|$
- 5:    $seen \leftarrow seen \cup p[last]$
- 6: **else**
- 7:    $last \leftarrow \arg \max_{q \in Q} |p[q]|$
- 8:    $seen \leftarrow p[last]$
- 9:    $first \leftarrow \arg \max_{q \in Q} |p[q] \setminus seen|$
- 10:    $seen \leftarrow seen \cup \{p[first]\}$
- 11: **end if**
- 12:  $ordered \leftarrow \langle \rangle$   $\triangleright$  empty list
- 13:  $remaining \leftarrow Q \setminus \{first, last\}$
- 14: **for each**  $g$  in  $p[first]$  **do**
- 15:   Append  $g$  to  $ordered$
- 16: **end for**
- 17: **while**  $remaining \neq \emptyset$  **do**
- 18:    $q^* \leftarrow \arg \max_{q \in remaining} |p[q] \setminus seen|$
- 19:   **for each**  $g$  in  $p[q^*]$  **do**
- 20:     Append  $g$  to  $ordered$
- 21:      $seen \leftarrow seen \cup \{g\}$
- 22:   **end for**
- 23:    $remaining \leftarrow remaining \setminus \{q^*\}$
- 24: **end while**
- 25: **for each**  $g$  in  $p[last]$  **do**
- 26:   Append  $g$  to  $ordered$
- 27: **end for**

---

## B. Evaluation of the Optimal Solution

Leveraging the symbolic formulation, a first series of experiments obtained *optimal* solutions for 168 circuit instances within the given time limit. This even includes instances composed of up to 243 gates, requiring the SAT-CP solver to handle cases with a search space of up to approximately  $10^{815}$  possible variable assignments. The top plot in Fig. 4 shows an orange line, which indicates the obtained optimal solution, i.e., the minimum attainable number of MOVE operations aggregated over all circuits (grouped by algorithm).

The results confirm that, despite the enormous complexity, *optimal* solutions for numerous non-trivial instances can be obtained within acceptable runtimes (at most three hours).

## C. Evaluation of the Proposed Heuristic

In a second series of experiments, the proposed heuristic was applied to the full dataset of 441 circuits. Thanks to its linear complexity, the entire compilation could be completed within minutes on a standard laptop, rendering runtime concerns negligible.

The results are shown in Fig. 4, with the top plot covering instances with a proven minimal result and the bottom plot covering all benchmarked circuits. The gray and blue bars indicate the total aggregated number of MOVE operations obtained by the vendor-provided and proposed heuristic solutions, respectively.

The results show that a small subset of circuits (13%) exhibits a slightly increased number of MOVE operations under

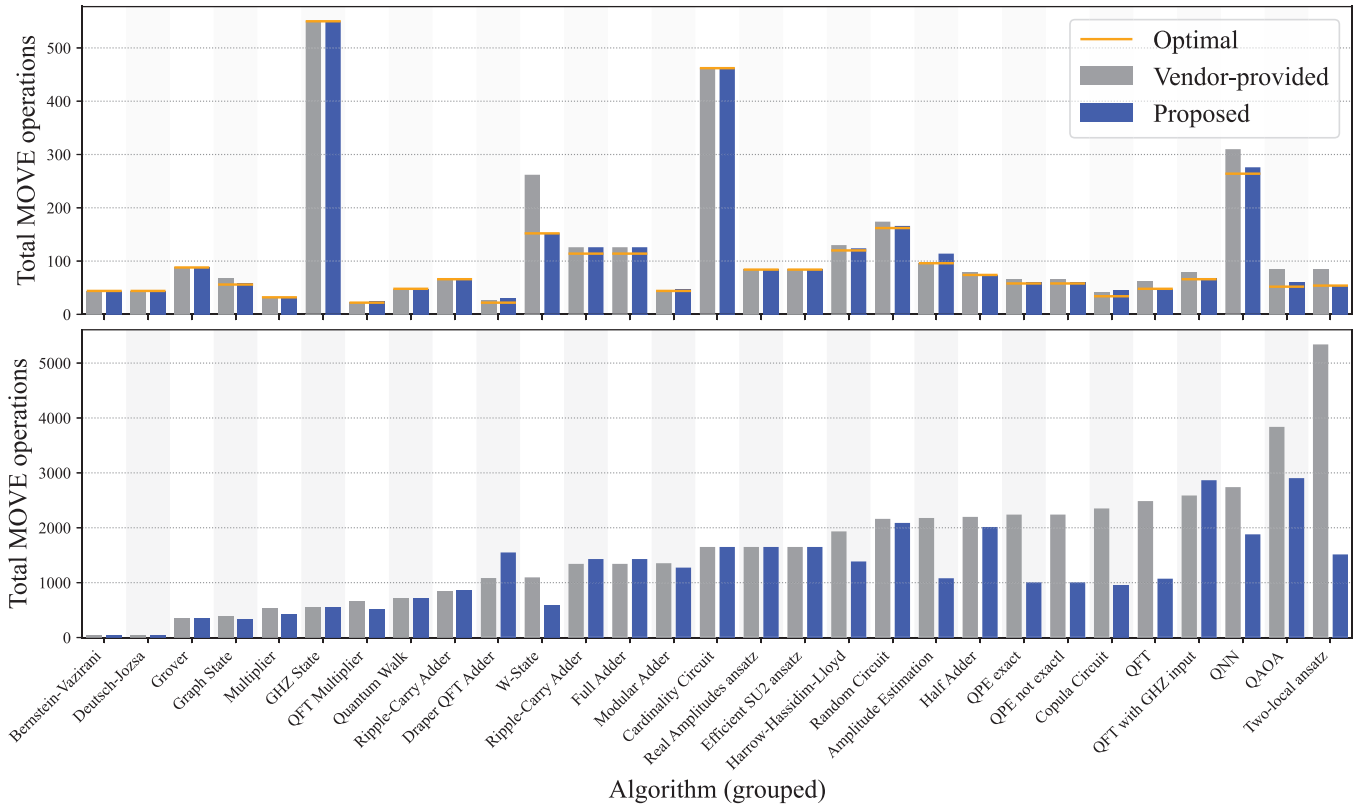


Fig. 4: Results obtained for 168 small circuits amenable to optimal compilation (top), and for the full benchmark set of 441 circuits compiled with the heuristics (bottom). The circuits, comprising between 3 and 24 qubits, are grouped by algorithm, and the total number of MOVE operations is aggregated for each group.

the proposed heuristic. This can be attributed to structural properties of a few circuits particularly advantageous for the vendor-based approach (after all, these are heuristics, and edge cases are expected where one method outperforms the other). In contrast, more than 46% of the circuits benefit from the proposed heuristic, achieving an average reduction of 14.5% in MOVE operations. The advantage is particularly pronounced for larger instances, with reductions of up to 78%.

Moreover, the value of the proposed heuristic becomes evident when compared to the optimal results. In fact, a heuristic is particularly valuable if its performance remains close to the optimum while remaining computationally efficient. This is the case with the proposed heuristic. For results with known optima (top plot in Fig. 4), the proposed approach matches the optimum in over 85% of cases. In the remaining cases, the deviation from the optimal MOVE count amounts on average to only 2.8%. Hence, the proposed heuristic consistently produces either the optimal or a near-optimal solution for the evaluated cases.

Overall, the evaluation confirms that the formalization generates optimal results for many instances, establishing a rigorous baseline for future compilation strategies on superconducting bus-resonator architectures, while the heuristic enables near-optimal compilation and scales efficiently to practical circuits.

## VII. CONCLUSION

This work provides the first symbolic formalization of the compilation task for superconducting bus-resonator architectures solvable via SAT-CP. This leads to optimal solutions for small quantum circuit instances. To complement this exact

approach, we introduced a linear-time heuristic that reduces the operational overhead required to exploit the architecture at scale.

Across 441 compiled circuits, the heuristic achieved improvements in more than 46% of the instances over the vendor-provided approach. On a subset of 168 smaller circuits, exact optimization revealed the available headroom, of which the heuristic captured 85% exactly, while the remaining cases deviated by only 2.8% from the optimal.

The results confirm that the formalization renders optimal solutions tractable even for large search spaces, while the heuristic enables near-optimal compilation for circuits of practical size. Together, they provide both a rigorous baseline and a scalable software solution that makes it possible to harness the full potential of superconducting bus-resonator architectures. Future work will extend this direction toward multi-resonator layouts and aim at integrating the proposed solutions in software stacks such as the Munich Quantum Software Stack [36].

## VIII. ACKNOWLEDGEMENTS

The authors thank the IQM Quantum Computers team for fruitful discussions and their support. The authors acknowledge funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program grant agreement No. 101001318 and No. 101114305, the Munich Quantum Valley, supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and the BMFTR under FullStaQD and grant number 13N17298 (SYNQ), and the Deutsche Forschungsgemeinschaft under grant number 563436708.

## REFERENCES

- [1] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, “Superconducting qubits: Current state of play,” *Annual Review of Condensed Matter Physics*, vol. 11, pp. 369–395, Mar. 2020.
- [2] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, “Trapped-ion quantum computing: Progress and challenges,” *Applied Physics Reviews*, vol. 6, May 2019.
- [3] D. Schoenberger, S. Hillmich, M. Brandl, and R. Wille, “Shuttling for scalable trapped-ion quantum computers,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 6, pp. 2144–2155, 2025.
- [4] M. Saffman, “Quantum computing with atomic qubits and Rydberg interactions: Progress and challenges,” *Journal of Physics B: Atomic, Molecular and Optical Physics*, vol. 49, p. 202001, Oct. 2016.
- [5] L. Schmid, D. F. Locher, M. Rispler, S. Blatt, J. Zeiher, M. Müller, and R. Wille, “Computational capabilities and compiler development for neutral atom quantum processors—connecting tool developers and hardware experts,” *Quantum Science and Technology*, vol. 9, p. 033001, Apr. 2024.
- [6] J. Majer, J. M. Chow, J. M. Gambetta, J. Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, *et al.*, “Coupling superconducting qubits via a cavity bus,” *Nature*, vol. 449, pp. 443–447, Sept. 2007.
- [7] M. A. Sillanpää, J. I. Park, and R. W. Simmonds, “Coherent quantum state storage and transfer between two phase qubits via a resonant cavity,” *Nature*, vol. 449, pp. 438–442, Sept. 2007.
- [8] C. Song, K. Xu, W. Liu, C.-P. Yang, S.-B. Zheng, H. Deng, *et al.*, “10-qubit entanglement and parallel logic operations with a superconducting circuit,” *Physical Review Letters*, vol. 119, p. 180511, Nov. 2017.
- [9] C. Song, K. Xu, H. Li, Y.-R. Zhang, X. Zhang, W. Liu, Q. Guo, *et al.*, “Generation of multicomponent atomic Schrödinger cat states of up to 20 qubits,” *Science*, vol. 365, no. 6453, pp. 574–577, 2019.
- [10] M. Renger, J. Verjauw, N. Wurz, A. Hosseinkhani, C. Ockeloen-Korppi, W. Liu, A. Rath, *et al.*, “A superconducting qubit-resonator quantum processor with effective all-to-all connectivity,” *arXiv preprint*, 2025.
- [11] F. Vigneau, S. Majumder, A. Rath, P. Parrado-Rodríguez, F. R. F. Pereira, H.-S. Ku, F. Šimkovic, S. Pogorzalek, T. Jones, N. Wurz, M. Renger, J. Verjauw, P. Yang, W. Kindel, F. Deppe, and J. Heinsoo, “Quantum error detection in qubit-resonator star architecture,” *PRX Quantum*, vol. 6, p. 040369, Dec 2025.
- [12] A. Meijer-van de Griend and R. Duncan, “Architecture-aware synthesis of phase polynomials for NISQ devices,” *Electronic Proceedings in Theoretical Computer Science*, vol. 394, pp. 116–140, Nov. 2023.
- [13] S. Schneider, L. Burgholzer, and R. Wille, “A SAT encoding for optimal Clifford circuit synthesis,” in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 190–195, ACM, 2023.
- [14] B. Giles and P. Selinger, “Exact synthesis of multiqubit Clifford+T circuits,” *Physical Review A*, vol. 87, no. 3, p. 032332, 2013.
- [15] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, “A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013.
- [16] T. Peham, N. Brandl, R. Kueng, R. Wille, and L. Burgholzer, “Depth-optimal synthesis of Clifford circuits with SAT solvers,” in *International Conference on Quantum Computing and Engineering*, pp. 802–813, IEEE, 2023.
- [17] E. Younis and C. Iancu, “Quantum circuit optimization and transpilation via parameterized circuit instantiation,” in *International Conference on Quantum Computing and Engineering*, pp. 465–475, IEEE, 2022.
- [18] W.-H. Lin, J. Kimko, B. Tan, N. Bjørner, and J. Cong, “Scalable optimal layout synthesis for NISQ quantum processors,” in *Design Automation Conference*, pp. 1–6, 2023.
- [19] A. Matsuo and S. Yamashita, “An efficient method for quantum circuit placement problem on a 2-D grid,” in *International Conference of Reversible Computation*, pp. 162–168, 2019.
- [20] B. Tan and J. Cong, “Optimal qubit mapping with simultaneous gate absorption,” in *International Conference on Computer-Aided Design*, pp. 1–8, 2021.
- [21] G. Li, Y. Ding, and Y. Xie, “Tackling the qubit mapping problem for NISQ-era quantum devices,” in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.
- [22] T. Peham, L. Burgholzer, and R. Wille, “On optimal subarchitectures for quantum circuit mapping,” *ACM Transactions on Quantum Computing*, 2023.
- [23] J. Liu, E. Younis, M. Weiden, P. Hovland, J. Kubiawicz, and C. Iancu, “Tackling the qubit mapping problem with permutation-aware synthesis,” in *International Conference on Quantum Computing and Engineering*, vol. 1, pp. 745–756, IEEE, 2023.
- [24] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, “On the qubit routing problem,” in *Conference on the Theory of Quantum Computation, Communication and Cryptography*, pp. 5:1–5:32, 2019.
- [25] R. Wille and L. Burgholzer, “MQT QMAP: Efficient quantum circuit mapping,” in *International Symposium on Physical Design*, 2023.
- [26] 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*, vol. 38, pp. 1226–1236, July 2019.
- [27] G. Li, Y. Ding, and Y. Xie, “Tackling the qubit mapping problem for NISQ-era quantum devices,” in *International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1001–1014, 2019.
- [28] P. Hopf, N. Quetschlich, L. Schulz, and R. Wille, “Improving figures of merit for quantum circuit compilation,” in *Design, Automation and Test in Europe Conference*, pp. 1–7, IEEE, 2025.
- [29] T. Peham, L. Schmid, L. Berent, M. Müller, and R. Wille, “Automated synthesis of fault-tolerant state preparation circuits for quantum error-correction codes,” *PRX Quantum*, vol. 6, p. 020330, May 2025.
- [30] R. Wille, L. Burgholzer, and A. Zulehner, “Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations,” in *Proceedings of the 56th Annual Design Automation Conference*, DAC ’19, pp. 142–147, 2019.
- [31] L. Berent, L. Burgholzer, and R. Wille, “Towards a SAT encoding for quantum circuits: A journey from classical circuits to Clifford circuits and beyond,” in *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, LIPIcs*, pp. 18–32, Aug. 2022.
- [32] L. Perron, F. Didier, and S. Gay, “The CP-SAT-LP solver,” in *International Conference on Principles and Practice of Constraint Programming*, vol. 280, pp. 3:1–3:2, 2023.
- [33] L. De Moura and N. Bjørner, “Z3: an efficient SMT solver,” in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, 2008.
- [34] N. Quetschlich, L. Burgholzer, and R. Wille, “MQT Bench: Benchmarking software and design automation tools for quantum computing,” *Quantum*, vol. 7, p. 1062, July 2023.
- [35] R. Wille, L. Berent, T. Forster, J. Kunasaikaran, K. Mato, T. Peham, N. Quetschlich, D. Rovara, A. Sander, L. Schmid, D. Schoenberger, Y. Stade, and L. Burgholzer, “The MQT handbook: A summary of design automation tools and software for quantum computing,” in *2024 IEEE International Conference on Quantum Software (QSW)*, pp. 1–8, IEEE, July 2024.
- [36] L. Burgholzer, J. Echavarría, P. Hopf, Y. Stade, D. Rovara, L. Schmid, E. Kaya, B. Mete, M. N. Farooqi, M. Chung, M. De Pascale, L. Schulz, M. Schulz, and R. Wille, “The Munich Quantum Software Stack: Connecting end users, integrating diverse quantum technologies, accelerating HPC,” in *Proceedings of the Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region, SCA/HPCAsia ’26*, pp. 55–67, 2026.