Exact Physical Design of Quantum Circuits for Ion-Trap-based Quantum Architectures

Oliver Keszocze*, Naser Mohammadzadeh†, and Robert Wille‡§
*Department of Computer Science, Friedrich-Alexander-Universität (FAU), Erlangen, Germany
†Department of Computer Engineering, Shahed University, Tehran, Iran
‡Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
§Software Competence Center Hagenberg GmbH (SCCH), Austria
oliver.keszoecze@fau.de, mohammadzadeh@shahed.ac.ir, robert.wille@jku.at

Abstract—**Quantum computers exploit quantum effects in a controlled manner in order to efficiently solve problems that are very hard to address on classical computers. The ion-trap-based technology is a particularly advanced concept of realizing quantum computers with advantages with respect to physical realization and fault-tolerance. Accordingly, several physical design methods aiming at realizing quantum circuits to corresponding architectures have been proposed. However, all these methods are heuristic and cannot guarantee minimality. In this work, we propose a solution which can generate *exact* physical designs, i.e., solutions which require a minimal number of time steps. To this end, satisfiability solvers are utilized. Experimental evaluations confirm that, despite the underlying computational complexity of the problem, this allows to generate minimal physical designs for several quantum circuits for the first time.**

I. Introduction

Allowing for quantum effects in a controlled manner might be exploited as a feature. Richard Feynman originally proposed using a well-controlled quantum system to efficiently solve problems that are very hard to address on classical computers and named the device a *quantum computer* [1]. They operate on the entangled superposition states, where the power of quantum algorithms comes from. Factorization, unsorted database search, and the simulation of quantum-mechanical systems are some classic hard problems that benefit from quantum algorithms [2].

Several candidate technologies have been proposed for the realization of a quantum computer to date [3]. Of these technologies, trapped ions are currently one of the most advanced. It not only adequately fulfills the DiVincenzo criterion but also possesses several prominent properties demonstrated experimentally which are conducive to architectures for the fault tolerant quantum computation [4]. Kielpinski et al. [5] proposed a multiplexed trap architecture which was modular and more scalable. Current scalable ion-traps have precision electrode structures that can be fabricated by using standard semiconductor processing techniques [4]. Because of these promising attributes, the ion-trap technology is considered in this paper.

However, to use this technology, corresponding quantum circuits need to be properly realized onto the architecture – requiring a design flow. In an abstract fashion, this quantum circuit design flow can be divided into two main processes: synthesis and physical design [6]. The synthesis process takes a description and generates an optimized technology-independent netlist. The purpose of physical design is to embed an abstract circuit description, such as a netlist, into a detailed geometric layout. Physical design consists of scheduling, placement, and routing processes. The scheduling process determines the execution order of gates. The placement one figures out where blocks are placed. The routing process routes data between blocks [7].

Some heuristics for these processes have been proposed for ion-trap-based architectures, see, e.g. [8–13] for scheduling, [14–17] for the placement and routing of operations, or [18–22] for optimization techniques. They try to minimize metrics such as the number of required time steps [6]. However, all of them are heuristic, i.e., none of them can guarantee minimality. This is a severe problem, as it leaves uncertainties on the best possible realizations, spoils comparisons between heuristics, and makes the design of larger functionality relying on (preferably minimal) building blocks rather inefficient.

In this paper, we are closing this gap by proposing an *exact* approach for the physical design of quantum circuits for ion-trap-based architectures. The proposed approach takes an initial quantum circuit and a corresponding architecture as inputs and determines the optimal initial locations for qubits, the optimal locations for gates, the optimal routing, and the optimal order for the execution of gates so that all operations can be executed within a minimal number of time steps.

In order to guarantee minimality, we consider all possible solutions in a symbolic fashion. Afterwards, we apply satisfiability solvers together with an objective function (here, minimizing the number of time steps) to eventually determine an explicit instance representing the minimum. Although the underlying solving process is computationally rather expensive, today's satisifiability solvers are capable of tackling this instance at least for selected relevant quantum functions. In fact, experimental evaluations confirm that, using the proposed approach, minimal physical designs for several quantum circuits can be derived for the first time.

The remainder of this article is organized as follows: the background material including a review of quantum circuits and an abstraction of the ion-trap technology is provided in Section II. Section III includes the main motivation of our work. We propose the exact approach for physical design in Section IV. Section V provides the experimental results. Finally, Section VI concludes the paper.

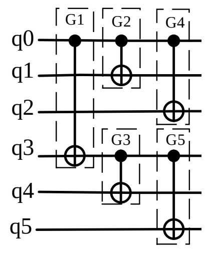


Fig. 1: A sample circuit including six qubits and five two-gubit gates.

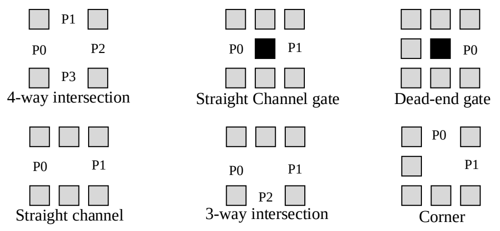


Fig. 2: Basic macroblocks for ion-trap layouts. Black boxes are gate locations, gray boxes are abstract electrodes, and wide white channels are valid paths for qubit movement [14]. Each macroblock has a specific number of ports (shown as P0-P3) along with a set of electrodes used for ion movement and trapping.

II. Background

In this section, we present background material on the quantum circuit model and the abstraction of the considered target technology (namely ion-trap-based quantum architectures).

A. Quantum Circuits

There are several models of quantum computing, including the quantum circuit model, the quantum Turing machine, the adiabatic quantum computer, the one-way quantum computer, topological quantum computing, and various quantum cellular automata [23]. The quantum circuit model is the most popular and developed model for quantum computation. The quantum circuit model uses a notation similar to representations of classical circuits, where qubits are represented as signals and operations are represented as gates. In fact, sequences of one- and two-qubit gates represent the fundamental logic for transforming a quantum state over time. However, there are several unique features for quantum computing including the inability to copy or clone arbitrary quantum states. Furthermore, the number of inputs into the circuit must be equivalent to the number of outputs and all quantum gates are unitary and can be described by unitary matrices [2].

More precisely, an n-qubit circuit is represented with n horizontal lines, with time flowing from left to right. A quantum computation typically requires the application of several quantum gates, sequentially or in parallel, to various subsets of qubits [2]. Note that, in the following, the precise functionality of those gates is not important, but the information on which qubits the respective operation has to be conducted. For more details on the functionality of quantum gates, we refer to [2].

Example 1. *Fig. 1 provides an example of a quantum circuit composed of six qubits and five two-qubit gates. As can be seen, the first gate works on qubits $q1$ and $q3$. The second and the third gate (working on qubits $q0$ and $q1$ as well as $q3$ and $q4$, respectively) can be executed in parallel.*

B. Ion-Trap-based Quantum Architectures

In this work, we consider ion-trap-based quantum architectures. In this technology, an ion represents a physical qubit which can be transported through rectangular channels, lined with electrodes, called wires. Within these channels, each qubit could be trapped or physically moved between gate locations by applying pulse sequences to the discrete electrodes. By this, qubits are moved through an architecture towards certain gate locations where the respective operation on those qubits is eventually executed.

In ion-trap technology, some details, such as which type of ion is used, specific electrode sizing and geometry, as well as exact voltage levels necessary for trapping and movement, may vary. Therefore, it is necessary to use a structure for certain design tasks that is independent of these details. Whitney et al. [14] defined a library of macroblocks and used them as the basic building blocks of layouts. By using these macroblocks, some low-level details, such as ion types, size of electrodes, and precise voltage levels needed for trapping and moving ions, are omitted. All of these details are condensed within the macroblocks. The basic blocks of an ion-trap layout are depicted in Fig. 2. In this figure, gate locations are indicated by black squares. Each macroblock has some ports to allow qubits to move between the macroblocks. Various orientations of each macroblock could be used in a layout. In this article and also many other studies in the field of physical design of quantum circuits [21, 22, 24], these macroblocks are used to construct quantum circuit layouts.

Overall, this leads to a model of ion-trap-based quantum architectures composed of [14]:

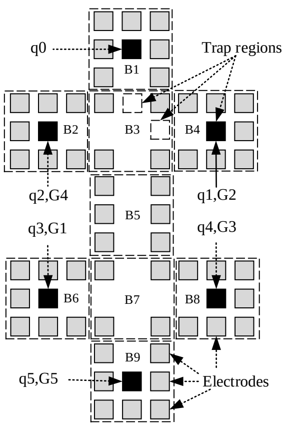
- Qubits, i.e., physical ions representing a quantum state which may be held in gate locations or moved between gate locations through a channel of control electrodes.
- Wires, i.e., channel elements with electrodes that allow to move trapped ions via pulses applied to those electrodes.
- Gate Locations, i.e., certain positions in the architecture which allow to execute quantum operations by applying laser pulses to the ions occupying this location (note that multiple gates may use the same gate location).
- Macroblocks, i.e., a certain combination of gate locations and wires which work as building blocks. Each macroblock has one or more "ports" through which qubits may enter and exit and which connect to an adjacent macroblock.

Example 2. *Fig. 3a shows a sample layout generated from macroblocks for the circuit depicted in Fig. 1. Trap regions are white spaces between electrodes that ion-qubits can be trapped in and, hence, moved along. For example, if qubits q0 and q3 are initially in macroblocks B1 and B6, respectively, to execute gate G1 in macroblock B6 qubit q0 should move through macroblocks B3, B5, B7, B6 in order to reach macroblock B6.*

In order to evaluate the quality of a realization, usually the execution time (i.e., the number of time steps) is considered as main cost metric. To calculate the number of time steps that one circuit needs to be run on a layout, physical delays for the gates and for the two types of move operations are used whose values are provided in Table I [25]. One microsecond corresponds to one time step.

TABLE I: Latency values for various physical operations in the ion-trap technology [25]

Operation	Latency [μs]
Single Gate	1
Double Gate	10
Move	1
Turn	10



(a) Qubit initial placement and gate placement. The list assigned to each gate location determines the initial location of each qubit and the gates that are to be performed in that gate location.

Time 0: Move q0: Blocks B3,B5,B7,B6
Time 14: Gate laser (G1): Blocks B6
Time 24: Move q0: Blocks B7,B5,B3,B4
Time 36: Move q3: Blocks B7,B8
Time 38: Gate laser (G3): Block B8
Time 48: Gate laser (G2): Block B4
Move q3: Block B7,B9
Time 58: Move q0: Block B3,B2
Time 60: Gate laser (G4): Block B2
Gate laser (G5): Block B9
Time 70: Finished

(b) Scheduling of the circuit on the layout.

Fig. 3: A sample layout generated from macroblocks for the circuit depicted in Fig. 1 and a sample mapping of the circuit onto the layout.

III. Motivation

This section provides the motivation of our work. To this end, we first review the physical design of quantum circuits for ion-trap-based quantum architectures. Afterwards, we discuss the currently available state-of-the-art implementations of its main building blocks and their shortcomings. Based on that, the remainder of this paper introduces an alternative method which tries to address these shortcomings.

A. Physical Design for Ion-Trap Quantum Architectures

In this work, we consider the physical design for ion-trap quantum architectures. The physical design process consists of scheduling, placement, and routing processes. The scheduling process determines the instruction execution sequence as well as the order of qubit movements across channels; the placement process assigns initial locations to qubits as well as locations of gates; and the routing process determines the movement paths of qubits.

Example 3. *Fig. 3 shows a sample mapping of the circuit depicted in Fig. 1 onto the layout. The macroblocks are labeled by the macroblock number. The list assigned to each gate location determines the initial location of each qubit and the gates that are to be performed in that gate location. Fig. 3b shows the schedule for the running sample circuit on the layout. Times show the start time for an action. This circuit needs 70 time steps to be run on the layout. The blocks that a qubit traverses are written in order in the front of "Blocks."*

B. Related Work and its Shortcomings

In this section, a review on related work in the field of quantum physical design for ion-trap architectures is presented to highlight the position of the proposed approach in the current literature. Balensiefer et al. [8] developed some tools to appraise layouts. Whitney et al. [14] proposed a greedy algorithm that is appropriate for small circuits and a dataflow-based algorithm for larger circuits aiming at placement and routing. Metodi et al. [10] presented a physical operations scheduler (QPOS) on a given physical layout. Dousti et al. [11] developed an iterative heuristic approach for placement based on forward and backward computations in a quantum instruction dependency graph (QIDG). Goudarzi et al. [17] proposed a mapper which considers the effect of the routing time on quantum instruction scheduling and placement. Goudarzi et al. [17] used a net-weighting timing-driven placement solution based on a modified version of the force-directed placement tool. Yazdani et al. [16] presented a scheduler using ILP and a layout generator using a graph-drawing algorithm [26]. In [20], a layout optimization technique, namely GLC, was proposed to improve the number of time steps. Bahreini et al. [27] proposed a mathematical model for scheduling and placement. They solved the MILP model for scheduling by the GAMS toolset [28] and the placement by a combination of genetic algorithm and tabu search methods. Ahsan et al. [13] proposed a physical design flow for the MUSIQC architecture [29]. They used a greedy ASAP scheduling algorithm and a partitioning-based placement approach. Mohammadzadeh et al. [30] proposed the SAQIP architecture and a design flow for it. Mohammadzadeh et al. [18] introduced the physical synthesis concept and proposed four techniques [18, 19, 21, 22] for it.

As can be seen, a substantial amount of work has been developed in the past years aiming to provide efficient solutions mapping a quantum circuit to ion-trap architectures. However, all these work rely on heuristics, i.e., cannot guarantee minimality with respect to the number of time steps. This is a severe shortcoming, since

- it remains unknown for almost all quantum functionality what the best possible realization would be,
- it makes it substantially harder to evaluate the quality of heuristics (e.g., proposing a new heuristic improving previous heuristics by 10% is marginal, if the actual minimum is still factors away, but impressive if this yields the minimum), and
- it makes the design of larger functionality relying on (preferably minimal) building blocks rather inefficient.

Motivated by this, we are proposing an *exact* physical design approach, i.e., a solution with the minimal number of time steps, in the following.

IV. Proposed Solution

In this section, we discuss the proposed solution in detail, which precisely considers the ion-trap architectures introduced in Section III-A and addresses the issues discussed in Section III-B. The generated solution is guaranteed to implement the desired circuit in as few time steps as possible. The proposed design flow is to model the design problem as a series of SMT (*Satisfiability Modulo Theories*, see, e.g., [31]) problem

Algorithm 1: Overall Exact Design Algorithm

```
input : Quantum circuit $\mathcal{C}$, physical layout $\mathcal{L}$
input : Initial number of time steps to use $start\_t$
input : maximal number of allowed time steps $T$
output: Exact design solution or no solution
1 for $t \leftarrow start\_t \dots T$ do
      ▷ Create SMT instance
2     $instance \leftarrow DESIGN(\mathcal{C}, \mathcal{L}, t)$
      ▷ Try to satisfy the instance
3     $res \leftarrow$ solve($instance$)
4     if $res == SAT$ then
5         return extract_solution($res$)
6 return no solution
```

instances whose satisfying solution (if it exists) corresponds to the solution of the design problem. In the literature, important research results have been achieved with such schemes (see, e.g., [32–35]), but none of them investigated the potential for the physical design of quantum circuits for ion-trap-based quantum architectures.

A. Overall Approach

Given a circuit $\mathcal{C}$, a layout $\mathcal{L}$, and a number of time steps t, the idea is to formulate an SMT instance $DESIGN(\mathcal{C}, \mathcal{L}, t)$ that is satisfiable if and only if the circuit is realizable on the layout within t time steps. This process is then iterated with increasing values for t until a solution is found (or a pre-specified maximal number of time steps is reached). This process is illustrated in Algorithm 1. Note that a safe lower bound for $stating\_t$ is the sum of the gate operation times d_g (see Table I) along the longest path in the dataflow graph $\mathcal{D}(\mathcal{C})$ of a given circuit $\mathcal{C}$. Starting with this value will reduce the overall computation time. Since, by this, all possible values for the number of time steps are tested, the method is guaranteed to determine the solution with the smallest number of time steps.

B. Modeling the Design Space

Let $\mathcal{Q}$ and a $\mathcal{G}$ denote the set of all (the IDs of the) qubits and gates used in the circuit $\mathcal{C}$, respectively.

To model the design space, we make use of the following SMT variables (internally stored as bit-vectors):

$$\begin{array}{ll} q_i^{(t)} \in \mathbb{N} & \text{for } i \in \mathcal{Q},\ t \in \{start\_t, \dots, T\} \\ op_g \in \mathbb{N} & \text{for } g \in \mathcal{G}\,. \end{array}$$

The value v of $q_i^{(t)}$ indicates that the qubit with ID i at time step t is at the trap position with ID v. The value w of op_g in turn indicates that the gate g has become operational in time step w.

Example 4. *Consider the circuit depicted in Figure 1 with $\mathcal{Q} = \{0, 1, 2, 3, 4, 5\}$ and $\mathcal{G} = \{1, 2, 3, 4, 5\}$ and the layout depicted in Figure 3 with $\mathcal{L} = \{1, \dots, 9\}$. The variables and their corresponding assignments describing the movement of qubit q0 to the macro block B6 are given as follows.*

$$q_0^{(0)} = 1 \quad q_0^{(1)} = 3 \quad q_0^{(2)} = 5 \quad q_0^{(3-13)} = 7 \quad q_0^{(14)} = 6$$

The operation of G1 starting in time step 14 is indicated by the assignment of $op_1 = 14$.

C. Constraining the Design Space

So far, no constraints regarding the assignments of the variables have been made. This would allow to create nonsensical assignments not representing a physically realizable solution. A random assignment of the variables, for example, could have qubits "jumping" between unconnected blocks in the layout. Therefore, the design space has to be constraint further.

Note that in the following, in order to make the description more concise and easier to follow, some implementation details are omitted. We will, for example, not show explicit checks ensuring that no negative time steps are used.

1) Qubit placement: Initially, qubits can only be placed on macro blocks with a gate position. Let $\mathcal{M}_g$ denote the set of macro blocks with a gate location. For the layout shown in Fig. 3a, for example, the set is given by $\mathcal{M}_g = \{1, 2, 4, 6, 8, 9\}$. The corresponding constraint is then given by

$$\bigwedge_{i \in \mathcal{Q}} q_i^{(0)} \in \mathcal{M}_g. \tag{1}$$

The next constraint is to make sure that the qubits do stay within the bounds of the layout. This is enforced by

$$\bigwedge_{i \in \mathcal{Q}} q_i^{(t)} \leq \#\,\mathcal{L}, \tag{2}$$

i.e., the qubits' positions are valid macro blocks.

Every macro block, except blocks with a gate position, can only have one qubit in it at a time. This is enforced by

$$\bigwedge_{t=start\_t+1}^{T} \ \bigwedge_{m \in \mathcal{M}_g} \left(\sum_{i \in \mathcal{Q}} q_i^{(t)} \leq 2 \right) \tag{3}$$

$$\bigwedge_{t=start\_t+1}^{T} \ \bigwedge_{m \in \mathcal{L} \setminus \mathcal{M}_g} \left(\sum_{i \in \mathcal{Q}} q_i^{(t)} \leq 1 \right). \tag{4}$$

2) Movement constraints: For each macro block $m \in \mathcal{L}$, we denote all the neighboring macro blocks that are reachable by a qubit with $N(m)$. The set $N(m)$ always contains m itself. To ensure that only valid qubit movements take place, we enforce that, if a qubit is present in a given macro block at time step t, it had to be within the neighborhood of this particular macro block in the previous time step, i.e.,

$$\bigwedge_{t=start\_t+1}^{T} \ \bigwedge_{\substack{i \in \mathcal{Q} \\ m \in \mathcal{L}}} \left(q_i^{(t)} = m \implies q_i^{(t-1)} \in N(m) \right). \tag{5}$$

To ensure that the turning time is handled correctly, the following constraints ensure that the qubit stays long enough in the macro block that was used for turning. For a given macro block m by $N^{\times}(m)$, we denote the set of all macro blocks that can be reached by turning once. Note that this set can be empty.

Example 5. *Consider the layout as shown in Figure 3. We have, for example, the sets $N^{\times}(B1) = \{B2, B4\}$ and $N^{\times}(B7) = \emptyset$.*

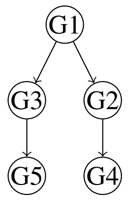


Fig. 4: Dataflow graph $\mathcal{D}(\mathcal{C})$ for the circuit $\mathcal{C}$ depicted in Fig. 1.

This allows to construct the following turning time constraint. The idea is that a qubit might not reach a block m from $N^{\times}(m)$ within d_t time steps ($d_t = 10$, see Table I).

$$\bigwedge_{\substack{i\in\mathcal{Q}\\ m\in\mathcal{L}}} \bigwedge_{t=start\_t+1} \left(q_i^{(t)} = m \implies \bigwedge_{t'=t-d_t}^{t-1} q_i^{(t')} \notin N^{\times}(m)\right). \tag{6}$$

This does already correctly model the movement of the individual qubits but does not prevent qubits from switching places. This is not allowed since there is no "space" in an individual channel for two qubits to move past each other. A qubit q0 must not move from macro block B3 in time step t to B5 in time step $t+1$ if another qubit q1 moves from macro block B5 to block B3. This is captured in the following constraint:

$$\bigwedge_{t=start\_t+1}^{t} \bigwedge_{\substack{i\in\mathcal{Q}\\ j\in\mathcal{Q}\setminus\{i\}}} \bigwedge_{\substack{m\in\mathcal{L}\\ n\in N(m)}} \neg\left(q_i^{(t)} = m \wedge q_j^{(t)} = n \wedge q_i^{(t-1)} = n \wedge q_j^{(t-1)} = m\right) \tag{7}$$

3) Gate Operation Constraints: In order to ensure that the operations are performed in the correct order, we construct the dataflow graph $\mathcal{D}(\mathcal{C})$ of the circuit $\mathcal{C}$. Each node g corresponds to a gate in $\mathcal{C}$ while the edges $(g^*, g^\dagger)$ connect gates that consecutively operate on shared quits.

Example 6. *Consider the circuit depicted in Fig. 1. The corresponding dataflow graph $\mathcal{D}(\mathcal{C})$ is shown in Fig. 4.*

Using the operation latency d_g for gate g (see Table I), the constraints for the correct ordering is then given by

$$\bigwedge_{(g^*,g^\dagger)\in\mathcal{D}(\mathcal{C})} op_{g^\dagger} \geq (op_{g^*} + d_{g^*}) \tag{8}$$

These constraints automatically enforce that all operations are started at all. The following constraint makes sure that the operations are started early enough to finish their operation within the specified time limit of T:

$$\bigwedge_{g\in\mathcal{C}} op_g \leq T - (d_g - 1) \tag{9}$$

The last constraint in the formulation is making sure that, for each gate in operation, the corresponding qubits are on the same gate location during the full time of the gate being operational. This is captured in the following constraint:

$$\bigwedge_{g\in\mathcal{C}} \bigwedge_{t=start\_t}^{T} \Big((op_g = t) \implies$$
$$\bigwedge_{i\in\mathcal{Q}(g)} \Big(\underbrace{q_i^{(t)} \in \mathcal{M}_g}_{a)} \wedge \underbrace{\bigwedge_{t'=t+1}^{t+1+d_g} q_i^{(t)} = q_i^{(t')}}_{b)} \wedge \underbrace{\bigwedge_{j\in\mathcal{Q}(g)} q_i^{(t)} = q_j^{(t)}}_{c)}\Big) \tag{10}$$

Here, $\mathcal{Q}(g)$ denotes the set of qubits being operated on by gate $g \in \mathcal{C}$. The individual parts ensure that a) the qubits are on a gate location, b) stay on their position during the operation of the gate, and c) are on the same macroblock.

Combining all these constraints faithfully models the design problem. Passing the resulting instance to corresponding SAT solvers (such as [36, 37]) allows to determine the solution with the minimal number of time steps.

V. Experimental Evaluation

In order to evaluate the proposed method, we implemented a C++ program which takes a quantum circuit to be synthesized as well as the considered layout and generates a corresponding SMT instance as described in Section IV. Afterwards, we passed the resulting instance to the SMT solver Z3 [36]. The respectively obtained (minimal) results are finally compared to the (heuristic) results reported in [27] which, to date, represents one of the state-of-the-art (heuristic) solutions for physical design of quantum circuits for ion-trap-based architectures. All experiments have been conducted on a Ubuntu 18.04.4 LTS machine with an Intel(R) Xeon(R) CPU E5-2630 v3 CPU running at 2.40GHz and 8GB of main memory with a timeout of 60 minutes.

The results of these comparisons are shown in Table II. The first columns provide the name of the considered benchmarks (taken from [27]) as well as their corresponding number of qubits and number of gates. Afterwards, the latency (i.e., the number of required time steps) of the solutions obtained by the heuristic method proposed in [27] and obtained by the exact method proposed in this work are given. Finally, the runtime of the proposed approach (in minutes) is given. Since the runtime of the heuristic method from [27] is negligible (i.e., always less than a minute), it is not explicitly reported in Table II.

From these results we can see that doing *exact* physical design of quantum circuits for ion-trap-based architectures is computationally rather expensive and, hence, only applicable to smaller quantum circuits. But, for the first time, this allows to generate *minimal* results for the benchmarks listed in Table II. Moreover, this additionally shows that there is still substantial room for improvements in today's heuristics. In fact, the number of required time steps often is still more than twice as large as the actual minimum. Finally, the results for benchmarks such as *1bitadde-rd32* can now be used as minimal building block for larger functionality. By this, all the shortcomings of the state-of-the-art as summarized in the end of Section III-B are addressed.

TABLE II: EXPERIMENTAL RESULTS

Benchmark	#qubits	#gates	T [27]	T proposed	Runtime [m]
1bitadder-rd32	4	16	303	**126**	9.40
input4-2-2	4	7	108	**75**	3.36
mod5-D4	3	14	173	**103**	36.84
input7-1-3	7	18	157	**84**	18.25
input5-0-3	5	11	132	**59**	4.32
input5-2-2	5	8	81	**57**	2.45
input5-1-3	5	12	152	**70**	3.42
input6-0-2	6	6	14	**13**	0.42
input6-2-2	6	10	76	**58**	4.62
ham3-D1	5	12	217	**101**	1.87
ham3-D2	3	13	242	**110**	2.73
input7-0-3	7	20	247	**118**	54.03

VI. Conclusions

In this work, we proposed an exact method for the physical design of quantum circuits on ion-trap-based architectures. With the proposed method, we were able to generate minimal results for the first time, to evaluate that heuristics available thus far still have substantial room for improvement, and to generate very compact building blocks to be used for realizing larger functionality. We are confident that future work will build up upon those results in the development of more sophisticated and improved design methods for ion-trap-based architectures.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

[1] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys*, vol. 21, no. 6/7, 1982.

[2] M. A. Nielsen and I. L. Chuang, "Quantum information and quantum computation," *Cambridge: Cambridge University Press*, vol. 2, no. 8, p. 23, 2000.

[3] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. L. O'Brien, "Quantum computers," *Nature*, vol. 464, no. 7285, pp. 45–53, 2010.

[4] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, 2019.

[5] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, vol. 417, no. 6890, pp. 709–711, 2002.

[6] N. Mohammadzadeh, "Physical design of quantum circuits in ion trap technology–a survey," *Microelectronics journal*, vol. 55, pp. 116–133, 2016.

[7] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of algorithms for physical design automation*. CRC press, 2008.

[8] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures," in *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE, 2005, pp. 186–196.

[9] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," in *38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05)*. IEEE, 2005, pp. 12–pp.

[10] ——, "Scheduling physical operations in a quantum information processor," in *Quantum Information and Computation IV*, vol. 6244. International Society for Optics and Photonics, 2006, p. 62440T.

[11] M. J. Dousti and M. Pedram, "Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 840–843.

[12] M. J. Dousti, A. Shafaei, and M. Pedram, "Squash: a scalable quantum mapper considering ancilla sharing," in *Proceedings of the 24th edition of the great lakes symposium on VLSI*, 2014, pp. 117–122.

[13] M. Ahsan, R. V. Meter, and J. Kim, "Designing a million-qubit quantum computer using a resource performance simulator," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 4, pp. 1–25, 2015.

[14] M. G. Whitney, *Practical fault tolerance for quantum circuits*. University of California, Berkeley, 2009.

[15] M. C. Moghadam, N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "A hierarchical layout generation method for quantum circuits," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*. IEEE, 2013, pp. 51–57.

[16] M. Yazdani, M. S. Zamani, and M. Sedighi, "A quantum physical design flow using ilp and graph drawing," *Quantum information processing*, vol. 12, no. 10, pp. 3239–3264, 2013.

[17] H. Goudarzi, M. J. Dousti, A. Shafaei, and M. Pedram, "Design of a universal logic block for fault-tolerant realization of any logic operation in trapped-ion quantum circuits," *Quantum information processing*, vol. 13, no. 5, pp. 1267–1299, 2014.

[18] N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "Quantum physical synthesis: improving physical design by netlist modifications," *Microelectronics Journal*, vol. 41, no. 4, pp. 219–230, 2010.

[19] N. Mohammadzadeh, M. S. Zamani, and M. Sedighi, "Auxiliary qubit selection: a physical synthesis technique for quantum circuits," *Quantum Information Processing*, vol. 10, no. 2, pp. 139–154, 2011.

[20] N. Mohammadzadeh, M. Sedighi, and M. S. Zamani, "Gate location changing: an optimization technique for quantum circuits," *International Journal of Quantum Information*, vol. 10, no. 03, p. 1250037, 2012.

[21] N. Mohammadzadeh, M. S. Zamani, and M. Sedighi, "Quantum circuit physical design methodology with emphasis on physical synthesis," *Quantum information processing*, vol. 13, no. 2, pp. 445–465, 2014.

[22] Z. Mirkhani and N. Mohammadzadeh, "Physical synthesis of quantum circuits using templates," *Quantum Information Processing*, vol. 15, no. 10, pp. 4117–4135, 2016.

[23] S. P. Jordan, "Quantum computation beyond the circuit model," *arXiv preprint arXiv:0809.2307*, 2008.

[24] G. Wang and O. Khainovski, "A fault-tolerant, ion-trap-based architecture for the quantum simulation algorithm," *Measurement*, vol. 10, no. 6, pp. 10–4.

[25] N. Isailovic, "An investigation into the realities of a quantum datapath," Ph.D. dissertation, UC Berkeley, 2010.

[26] G. D. Toolkit, "An object-oriented library for handling and drawing graphs."

[27] T. Bahreini and N. Mohammadzadeh, "An minlp model for scheduling and placement of quantum circuits with a heuristic solution approach," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 3, pp. 1–20, 2015.

[28] G. D. Corporation, "General Algebraic Modeling System (GAMS) Release 24.2.1," Washington, DC, USA, 2013. [Online]. Available: http://www.gams.com/

[29] C. Monroe, R. Raussendorf, A. Ruthven, K. Brown, P. Maunz, L.-M. Duan, and J. Kim, "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects," *Physical Review A*, vol. 89, no. 2, p. 022317, 2014.

[30] S. Sargaran and N. Mohammadzadeh, "Saqip: A scalable architecture for quantum information processors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 16, no. 2, pp. 1–21, 2019.

[31] L. De Moura and N. Bjørner, "Satisfiability modulo theories: Introduction and applications," vol. 54, no. 9, pp. 69–77. [Online]. Available: https://doi.org/10.1145/1995376.1995394

[32] R. Wille, M. Soeken, N. Przigoda, and R. Drechsler, "Exact synthesis of toffoli gate circuits with negative control lines," in *International Symposium on Multiple-Valued Logic*, 2012, pp. 69–74.

[33] O. Keszöcze, R. Wille, K. Chakrabarty, and R. Drechsler, "A general and exact routing methodology for digital microfluidic biochips," in *International Conference on Computer-Aided Design*, pp. 874–881.

[34] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler, "An exact method for design exploration of quantum-dot cellular automata," in *Design, Automation & Test in Europe Conference*, 2018, pp. 503–508.

[35] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Design Automation Conference*, 2019.

[36] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, C. R. Ramakrishnan and J. Rehof, Eds. Springer, pp. 337–340. [Online]. Available: ttps://github.com/Z3Prover/z3

[37] R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler, "SWORD: A SAT like prover using word level information," in *International Conference on Very Large Scale Integration of System-on-Chip*, 2007, pp. 88–93.