SAQIP: A Scalable Architecture for Quantum Information Processors

SAHAR SARGARAN and NASER MOHAMMADZADEH, Quantum Architectures and Computation Group (QACG), Department of Computer Engineering, Shahed University, Tehran, Iran

Proposing an architecture that efficiently compensates for the inefficiencies of physical hardware with extra resources is one of the key issues in quantum computer design. Although the demonstration of quantum systems has been limited to some dozen qubits, scaling the current small-sized lab quantum systems to large-scale quantum systems that are capable of solving meaningful practical problems can be the main goal of much research. Focusing on this issue, in this article a scalable architecture for quantum information processors, called SAQIP, is proposed. Moreover, a flow is presented to map and schedule a quantum circuit on this architecture. Experimental results show that the proposed architecture and design flow decrease the average latency and the average area of quantum circuits by about 81% and 11%, respectively, for the attempted benchmarks.

CCS Concepts: • **Computer systems organization** → **Quantum computing**;

Additional Key Words and Phrases: Quantum architecture, quantum circuits, scheduling, mapping

ACM Reference format:

Sahar Sargaran and Naser Mohammadzadeh. 2019. SAQIP: A Scalable Architecture for Quantum Information Processors. *ACM Trans. Archit. Code Optim.* 16, 2, Article 12 (April 2019), 21 pages.

<https://doi.org/10.1145/3311879>

1 INTRODUCTION

Quantum computing is a rapidly evolving research area. Its huge computational power [1–4] compared to classical computing is the main motivation of researchers. The architecture of a quantum computer plays a key role in its performance. The main task of a quantum architecture is to determine the organization of processing units and how they interconnect and communicate to manipulate data [5, 6]. Much of the quantum research until now has been concentrated on the two extremes: quantum algorithms and complexity theory at the top, and quantum physics at the bottom. Quantum information processing has moved up to the point where architecture-level solutions can be beneficial to close this gap [7]. In spite of their importance, they have not received much attention, leaving much area for high-impact research. Although several candidate

Authors' addresses: S. Sargaran and N. Mohammadzadeh, Department of Computer Engineering, Shahed University, Persian-Gulf Highway, Tehran, Iran; emails: s.sargaran@shahed.ac.ir, mohammadzadeh@shahed.ac.ir.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

1544-3566/2019/04-ART12

<https://doi.org/10.1145/3311879>

technologies have been presented for the realization of a quantum computer to date [8, 9], developing a scalable computer architecture for each one is still a challenging problem. The fundamental criteria [10] that a quantum-computing technology must have, and the challenge of designing quantum systems that are able to perform quantum error correction have motivated some research groups to describe extra categories to address the architectural requirements of a large-scale quantum machine [7, 11].

Ion trap technology is a very promising candidate for building a quantum computer [1–3]. It not only sufficiently satisfies the DiVincenzo’s requirements but also has several important properties demonstrated experimentally. It provides robust, high-fidelity state preparation [4] and readout [5], high-fidelity universal gate operations [6], and long qubit coherence times [4, 7–9]. Kielpinski et al. [10] in a high-impact paper presented an architecture for an ion-trap quantum processor, called QCCD,¹ which uses separate memory and interaction regions. The scalability of this architecture faces a wide range of technological challenges [11]. For example, it has been noticed that the individual addressing of ions becomes prohibitively difficult due to the geometry constraints of the planar traps containing in excess of 100 electrodes. Hence, processing a very large number of ions in these chips seems practically difficult. This problem can be managed if reliable means of connecting these restricted size chips can be found. Monroe et al. [12, 13] presented a modular distributed quantum-computer architecture, called MUSIQ, in which Elementary Logic Units (ELU), each composed of a string of 50–100 trapped ions, are connected through an optical switch. The hybrid structure of MUSIQ benefits from the strengths and counters the weaknesses of both ionic and photonic technologies. This new architecture also has some challenges. For example, Monz et al. [14] showed that the coherence of an N -qubit state decays by a factor N^2 faster than the coherence of a single qubit. However, the authors of Reference [12] showed that with loading of up to ~ 10 qubits an acceptable fidelity is reachable. However, decreasing the number of qubits in each ELU means that we have a lot of ELU that is challenging because of the complexity of the resultant optical switch. Ahsan et al. [15, 16] used the MUSIQ hardware but modified each ELU to have the QLA architecture [17] instead of being a string of ions. This architecture also faces some technological difficulties. First, since each ELU in this architecture is an array of gate locations including at most two qubits, the resulted layout has a large number of electrodes, therefore, a large area. Second, laser beams should be distributed to a large number of gate locations over the entire layout. Hence, building such a large-area chip seems practically hard. Focusing on these issues, our architecture, SAQIP, mitigates these scalability concerns by using technology capabilities and creating customized ELUs. Our architecture is consisted of a large number of full-custom blocks connected by a reconfigurable optical switch network. The overview of the SAQIP architecture is shown in Figure 1. The reconfigurable optical switch network can also be organized in a tree-like structure such that the height of the tree scales only logarithmically with the number of blocks. The blocks are designed in the full-custom style based on the multiplexed ion trap structure [6]. In other words, in our architecture, the structure of each processing unit is fully customized for a partition. Each block is consisted of some interaction zones and each zone confines a few ions. Quantum data is transferred between zones of a block by ballistically shuttling ions and between blocks by photons via the optical switch network.

The procedure of mapping and scheduling a given quantum circuit on this architecture is a significant job, but existing quantum computer-aided design flows are not applicable to this work because of its different structure. Therefore, we propose a design flow to map and schedule a quantum circuit on this architecture. Figure 2 shows our computer-aided design (CAD) flow that has four main parts: *partitioning*, *placement*, *channel routing*, and *scheduling*. The partitioning step

¹Quantum Charged-Coupled Device.

Fig. 1. Overview of the SAQIP architecture.

Fig. 2. Design flow.

divides qubits among blocks such that the number of connections between blocks is minimized. The placement and routing steps generate a physical layout for each block. Finally, the scheduling step determines the execution order of gates on the generated layout. We automate the tasks involved in the flow. Although at first glance, some parts of this flow is like a classical CAD flow, the different nature of quantum domain necessitates new algorithms for tasks. Full details of the flow including the algorithms are mentioned in Section 3.

The rest of this article is organized as follows: An overview of the prior work is presented in Section 2, followed by the details of the proposed architecture in Section 3. Section 3 also includes the proposed design flow to map and schedule a quantum circuit on this architecture. To illustrate

the proposed architecture and design flow, an example is presented in Section 4. Section 5 shows the experimental results, and Section 6 concludes the article.

2 RELATED WORK

The quantum architecture determines the structure of processing and memory units and how they interconnect and communicate to manipulate data [18].

The nearest-neighbor architecture that has attractive properties from a hardware design point of view has been investigated in some research works [19–24]. The recent achievements in application of the topological quantum error-correcting codes and their good properties such as needing only nearest-neighbor operations have inspired some researchers to focus on architectures that are capable of generating multidimensional cluster states [25–32].

QuMA implements the quantum microinstruction set QuMIS [33]. QuMIS does not offer feedback control, and is tightly bound to the hardware implementation. Moreover, as the number of qubits grows, QuMA cannot fetch and execute instructions fast enough to apply all operations on qubits on time. Therefore, Fu et al. [34] proposed eQASM² that mitigates the quantum operation issue rate problem by efficient timing specification, single-operation-multiple-qubit execution, and a very-long-instruction-word architecture. Some methods [35, 36] have been proposed to map a quantum circuit onto the IBM QX Architectures [37]. All of these studies are based on superconducting technology. Linke et al. [2] compared the ion-trap system proposed by Monroe et al. [13] and the IBM Quantum Experience superconducting system experimentally. The results showed that the ion-trap system outperforms the superconducting system on all results [2].

Since our architecture is based on the ion-trap technology, in the remainder of this section, we review the studies whose underlying technology is the trapped ion one. We partition the studies into two parts. In all works mentioned in the first part, qubits are transported ballistically within regions and teleported between regions. In the second part, we describe most recent and related works based on photonic interconnects. At the end, we explain the main differences between the most recent works and ours.

The movement of data in a quantum computer has been studied in some research works. Kielpinski et al. [10] presented an architecture for a scalable ion-trap quantum processor, called QCCD¹, which uses separate memory and interaction regions. Some homogeneous systems have been developed around this basic idea. Mariantoni et al. [38] proposed the quantum von Neumann architecture including dedicated hardware for memories, zeroing registers, and a quantum bus. Metodi et al. [17] used local gates and quantum teleportation to transport quantum data across their ion-trap QLA architecture. Automated layout generation and optimization of quantum circuits for ion-trap architectures were explored in References [39, 40]. Isailovic et al. [41, 42] studied interconnection networks and communication channel bandwidth for such architectures with emphasis on the ancilla preparation for teleportation-based communication [43]. Svore et al.'s tool [44, 45] maps a technology-independent QASM netlist³ to a technology-dependent fault-tolerant netlist during its design flow. The authors automatically generate an H-Tree-based layout built from a single tile and schedule a circuit on it using the ASAP⁴ method.

Four main microarchitectures have been proposed for quantum computing in ion trap technology. They are QLA [17, 46, 47], CQLA⁵ [48], Qalypso [40], and Requip⁶ [49], and can be thought as a

²Executable quantum instruction set architecture.

³A netlist consists of a list of connections in a circuit and a list of the gates they are connected to.

⁴As Soon As Possible.

⁵Compressed quantum logic array.

⁶Reconfigurable quantum processor.

range from inflexible to flexible architecture. Arrangement of compute regions, ancilla generation regions, memory regions for idle qubits, and teleportation network resources are different in these microarchitecture. In the QLA microarchitecture, all units are identical like an FPGA. Each unit can perform a two-qubit gate. Each such compute region also includes ancilla generation resources, enough room for two encoded qubits, and a teleportation router for communication. CQLA enhanced QLA by adding a new type of data region to it. Therefore, compute regions are identical to those in QLA and memory regions store eight qubits. Qalypso [41] is an enhanced version of CQLA with more flexibility in assigning of ancilla generation resources. It utilizes optimized and pipelined ancilla generators to create ancilla for use in processing units.

Dousti et al. proposed Requip [49], a multi-core reconfigurable quantum processor architecture. The quantum reconfigurable compute region (QRCR) proposed in this architecture distributes the ancilla qubits in the processing unit based on the issued instructions. This feature prevents from wasting the resources because of overestimating the ancilla qubits needed.

Heckey et al. [50] proposed an ion trap multi-SIMD architecture composed of k SIMD operating regions each with a width or qubit capacity of d to execute many quantum applications written in Scaffold programming language. Its architecture model is so similar to QLA and has the same disadvantages. This work and the next studies done on it [51–54] have mainly focused on compiling and scheduling a quantum program on this architecture.

Maslov [55] presented an approach to compile quantum algorithms into physical-level operations. Goudarzi et al. [56] formulated the quantum instruction placement problem and used a net-weighting timing-driven placement solution based on a modified version of the force-directed placement tool, SimPL [57] to solve it.

Lekitsch et al. [1] proposed a trapped ion-based quantum computer module based on long-wavelength radiation quantum operations. They use the microwave-based multi-qubit operations instead of the laser-based ones, and laser beams are only applied for state preparation and detection, photoionization, and sympathetic cooling. In this architecture, modules are aligned next to each other in a two-dimensional array, and ions are ballistically transported from one module to the adjacent module. This architecture encounters some technical issues such as the creation of strong magnetic field gradients and the requirement of calibration operations and well-controlled voltages, which are required to perform quantum operations. The other challenging part of this scheme is to strictly align all modules to each other to avoid large barriers or interruptions of the overlapping electric fields from occurring. Communication between distant ions is another challenging issue in this architecture. Since it features nearest-neighbor ion-ion interactions, they can only take place between nearest-neighbor modules. Therefore, if two distant qubits (ions) in two non-adjacent modules want to communicate with each other, then one of them should be ballistically transported the entire path. Ballistic transport of an ion-qubit decreases the fidelity of its state. Therefore, the distance that an ion qubit may be transported ballistically before quantum error correction must be applied, is limited. There is a general agreement that large quantum systems need different forms of communication for longer distances [58, 59].

Coupling of far qubits and communication between distant sites is a challenging issue in the designing of a quantum processor architecture. Cirac et al. [60] suggested using photonic interconnects for communication between distant atoms. Using the photons in the cavity to act as a quantum channel to mediate interactions between non-local qubits has been explored in other works [61–65]. These photonic channels have been utilized to propose a modular, scalable distributed quantum architecture [66].

Distributed quantum computing has recently drawn attention of some groups of researchers. Devitt et al. integrated the topological quantum model with the photonic module to build a

Fig. 3. (a) Original MUSIQC Architecture [13]. (b) Modified MUSIQC architecture with QLA-style ELUs [16].

photonic topological quantum network over different areas [67]. Van Meter et al. investigated the performance of arithmetic algorithms on a quantum multicomputer system [68]. One of the main foundations of their paper is that large-scale quantum systems need heterogeneous interconnections. Van Meter et al. extended this idea with a design in that electron-spin quantum dots were coupled through nano-photonic waveguides [69]. The different nodes of this quantum multicomputer communicate via a multi-level interconnect.

Although the QCCD architecture [10] proposed by Kielpinski et al. seems to be good for small ion-trap quantum processors with about 100-1000 qubits, scaling it to a million-qubit processor is difficult because of the challenges with interconnects, diffraction of optical beams, and the complexity of qubit control [12]. To scale beyond the QCCD architecture, Monroe et al. proposed a modular distributed quantum-computer architecture, called MUSIQC, in which Elementary Logic Units (ELU), each composed of a string of 50–100 trapped ions, are connected through an optical switch (Figure 3(a)) [12, 13]. Ahsan et al. used the MUSIQC architecture but modified each ELU to have the QLA architecture [17] instead of being a string of ions used in the original MUSIQC proposal (Figure 3(b)) [15, 16]. However, the area and delay of this architecture framework can be improved by customizing each ELU based on a sub-circuit. In other words, Ahsan et al.'s proposal is not optimized in terms of area and delay, because all units have an identical structure like an FPGA. Focusing on this issue, this article modifies Ahsan's proposal to have an architecture with *full-custom blocks* and *photonic interconnects* (Figure 1). Moreover, this article proposes a flow to map a quantum circuit on this architecture. Since building blocks of this architecture are fully customized, quantum circuits mapped onto this architecture potentially have less delay and less area than those mapped onto Ahsan et al.'s architecture.

3 THE SAQIP ARCHITECTURE AND THE MAPPING PROCEDURE

In the original MUSIQC architecture proposed by Monroe et al. [13], the basic logic unit is composed of a string of 50–100 trapped atomic ions. Ahsan et al. [15, 16] modified the original proposal and changed the ELU architecture from the string of ions to the QLA architecture. This architecture makes some assumptions that can be relaxed based on the recent advancements in ion-trap technology [12, 70–75] to reach a better architecture. The current state of ion-trap technology provides the requirements to present a suitable quantum architecture. In the following four paragraphs, the new technology features used by our architecture are mentioned.

A circuit mapped onto the QLA occupies more area and takes longer than it mapped onto a full-custom fabric. In other words, if each ELU is customized based on the circuit to be mapped on it, then the latency and area may be dramatically improved.

The second assumption of the QLA is that the maximum number of qubits placed in each gate location is two. This assumption increases communication in each ELU and has high cost in time and fidelity of a circuit [76]. Moreover, it results in a large number of gate locations in each ELU. In ion-trap technology, switching the laser from one trap to the next over the entire layout at a random order has a significant overhead. Therefore, the architecture must minimize the gate locations in the layout where lasers are applied. The disadvantages caused by the second assumption may imply that if the number of qubits in each gate location increases, then the error and the latency are improved. However, trapping a very large number of qubits in a single gate location is impractical, because lengthening the ion chain progressively increases the difficulty of identifying the vibration modes [77] and therefore, decreases the gate fidelities. Moreover, the heating rate and hence, the dephasing rate increases.

Experiments show that placing up to ~ 10 qubits in each gate location gives a good fidelity for gates [12]. Putting more than two qubits in each location has another significant effect. The gates that have the same control qubits and are ready to be scheduled can be performed at the same time. This capability may reduce the circuit latency dramatically. Following this line of reasoning, the number of qubits in each gate location can increase up to 10.

The third assumption of QLA restricts quantum operations to only one- and two-qubit gates. The gates with more than two qubits such as a Toffoli gate have been already demonstrated with high fidelity in the ion-trap technology [70, 73, 74, 78, 79]. Using gates with more than two-qubits may decrease the latency and error probability.

The aforementioned matters motivated us to propose a *Scalable Architecture for Quantum Information Processors (SAQIP)* that uses the full capabilities of the technology. This architecture is composed of full-custom blocks interconnected by photonic links.

ALGORITHM 1: Mapping

[Input]: A logic circuit netlist

[Output]: Circuit mapped onto the SAQIP architecture

- Generate dataflow graph $G_D = (G, E_D), g \in G$
 - Generate interaction hypergraph of qubits $G_{CI} = (Q, E_{CI}), q \in Q$
 - Remove edges (hyperedges) with outlier values $G_I = (Q, E_I), q \in Q$
 - Partition G_I into k sub-graphs $G_1, G_2, \dots, G_k$
 - **For each** partition G_i
 - Find a linear arrangement for qubits S (**ALGORITHM 2**)
 - Partition S into s strings $S_1, S_2, \dots, S_s$ (**ALGORITHM 3**)
 - Place $S_1, S_2, \dots, S_s$ on a 2D layout (**ALGORITHM 4**)
 - Insert routing channels
 - Schedule the dataflow graph
-

Figure 4 shows a big picture of our mapping flow. Its pseudo code is given in Algorithm 1. Its input and output are a fault-tolerant logic circuit netlist and the mapping of the circuit on the architecture, respectively. We use the same quantum error correcting (QEC) code (Steane code [80]) as Ahsan's proposal to make the netlist fault-tolerant. In the rest of this section, we explain the technical details of the procedure, justification of the usage of Steane code, and time complexity of the flow.

Fig. 4. The big picture of our mapping procedure.

3.1 Dataflow Graph and Interaction Hypergraph Generation

In the beginning, the dataflow graph of the circuit G_D (line 1, Algorithm 1) and the interaction hypergraph⁷ of the qubits G_{CI} are generated (line 2-Algorithm 1). In the dataflow graph, each node represents a gate and each edge represents a qubit dependency. In the interaction hypergraph of qubits, each node denotes a qubit and each edge (hyperedge) denotes the gate that is performed between the qubits connected by the edge (hyperedge). Each edge (hyperedge) in the interaction graph is labeled with the level number of the gate in the netlist. Since qubits interact in different parts of the circuit, the numbers on the edges (hyperedges) between two (three) qubits may be distant from others. Therefore, it seems that if the edges (hyperedges) with outlier values are removed from the graph, the next clustering step could be more efficient. In this article, the IBM SPSS analytics toolkit [81] is used to detect and remove edges (hyperedge) with outlier values (line 3, Algorithm 1).

3.2 The K-way Partitioning

The modified interaction graph G_I is partitioned into k subgraphs where k is the number of ELUs (line 4, Algorithm 1). The multilevel k -way hypergraph partitioning algorithm [82] is used to partition the hypergraph into subgraphs where each subgraph is to be mapped onto an ELU. This

⁷A hypergraph consists of nodes and hyperedges, with each hyperedge being a subset of two or more nodes.

Fig. 5. The various steps of the multilevel k-way partitioning algorithm.

algorithm generates high-quality parts while enforcing tight balancing constraints. It reduces the size of the hypergraph by collapsing vertices and edges (coarsening phase), finds a k -way partition of the smaller graph, and then it constructs a k -way partition for the original graph by projecting and refining the partition to successively finer graphs (uncoarsening phase). The various steps of the multilevel k-way partitioning algorithm are depicted in Figure 5.

3.3 Qubit Arrangement Problem and Solution

After the hypergraph is partitioned into the subgraphs, a full-custom physical layout is generated for each subgraph (ELU) (lines 5–9, Algorithm 1). Since each region should have up to 10 qubits, qubits of each ELU are segmented into s strings. Before partitioning qubits into strings, a linear arrangement is found for all qubits assigned to an ELU to keep a global view in the partitioning step (line 6, Algorithm 1). The algorithm used for finding linear arrangement for qubits of an ELU is mentioned in Algorithm 2.

The qubit arrangement problem takes the interaction graph $G_i(Q_i, E_i)$ to put each qubit q_k in place x_k on a straight line such that the metric $\sum_{i,j} w_{ij} |x_i - x_j|$ is minimized. The places are labeled $1, 2, 3, \dots, n$. The w_{ij} is the number of edges between vertex q_i and q_j in G_i . This problem is known as an optimal linear arrangement problem [83, 84]. The time complexity of the qubit arrangement problem is NP-hard. In other words, there is no known polynomial-time algorithm, which finds the optimal order. Therefore, the approximate polynomial time algorithm proposed in References [83, 84] is used to obtain an approximately optimal arrangement.

The algorithm used for solving the qubit arrangement problem is provided in Algorithm 2. It generates the adjacency matrix $A(G_i)$ representation of the interaction graph $G_i(Q_i, E_i)$ (line I). Then, the corresponding Laplacian matrix $L(G_i)$ of $A(G_i)$ is calculated, which is positive semi-definitive (line II). When $L(G_i)$ is diagonalized, its spectrum has non-negative eigenvalues (line III). In the next step, the second eigenvector V_2 corresponding to the second smallest eigenvalue λ_2 is computed (line IV). The components of V_2 is indexed using qubit identifier q_i for $i = 1, 2, 3, \dots, n$, where n is the number of qubits assigned to the ELU (line V). In other words, $V_2[1]$ is assigned to qubit 1, $V_2[2]$ is assigned to qubit 2 and so on. At the end, the position of qubit q_i is the index of $V_2[i]$ in sorted V_2 (V'_2) (lines VI and VII).

ALGORITHM 2: Qubit linear arrangement**[Input]:** Interaction graph $G_i(Q_i, E_i)$ **[Output]:** An approximately optimal linear arrangement S

- I. Generate the adjacency matrix $A(G_i)$ of the interaction graph;
- II. Calculate Laplacian matrix $L(G_i)$ of $A(G_i)$;
- III. Diagonalize $L(G_i)$ Let $\lambda_1(G_i) \geq \lambda_2(G_i) \geq \lambda_3(G_i) \geq \dots \geq \lambda_e(G_i)$ are sorted eigenvalues;
- IV. Find eigenvector V_2 corresponding to the second smallest eigenvalue $\lambda_2(G_i)$;
- V. Index V_2 by qubit number q_i for $i = 1, 2, 3, \dots, n$;
- VI. Sort V_2 to obtain V'_2 For each q_i , find new index j by comparing $V_2[i] = V'_2[j]$;
- VII. Put qubit q_i at location j on the line;

ALGORITHM 3: Partitioning of a sorted string**[Input]:** A sorted string $S(Q_i)$ with $|Q_i|$ nodes where each node $q \in Q_i$ and an interaction graph $G_i(Q_i, E_i)$ **[Output]:** s disjoint strings $S_1(Q_1), S_2(Q_2), \dots, S_s(Q_s)$ where $\sum_{j=1}^s |Q_j| = |Q_i|$ Partition(S)

1. Cut string s from the minimum-cut point into two strings S_{p1} & S_{p2} .
2. If the number of qubits in $S_{p1} > 10$ then
3. Partition (S_{p1});
4. If the number of qubits in $S_{p2} > 10$ then
5. Partition (S_{p2});

3.4 Partitioning of the Sorted String of Qubits

When the sorted long string is obtained, it is segmented into strings with 3–10 qubits by Algorithm 3 (line 7, Algorithm 1). A hierarchical minimum-cut approach⁸ is followed to segment the string obtained from the previous step into s strings. Algorithm 3 recursively partitions the long-sorted string S into shorter strings of 3–10 qubits. In each function call, the function cuts the input string from the minimum-cut point into two shorter strings. If one of the generated strings is longer than 10 qubits, then the function is recursively called with it. If the interaction graph is highly connected, then the partitioning may perform not very well. However, it should be noted that removing the edges with outlier values from the original interaction hypergraph in the previous step improves the result of this step.

3.5 Placement and Routing

As the partitioning step divided the string into shorter strings, a partition-aware placement is performed (line 8, Algorithm 1). The algorithm used for placement and routing (lines 8 and 9 of Algorithm 1) is mentioned in Algorithm 4. A position should be determined for each of strings generated in the previous step. The cluster growth placement idea [85] from the classical CAD literature is used to place strings in this article. It is a constructive placement algorithm using a bottom-up approach. The algorithm selects unplaced strings sequentially and places them in the layout. In this algorithm, the string that is most highly connected to the already placed ones is selected to be placed. Then, an exhaustive search is carried out to find the best possible location for the string. The outline of the cluster growth algorithm is shown in Algorithm 4.

At this point, the positions of strings are determined, and channels are added to the layout for communication between strings. Ions (qubits) can be ballistically transported along these channels. Although the approach proposed in this article tries to minimize the communication between

⁸In graph theory, a minimum cut of a graph is the cut of smallest edge weights.

ALGORITHM 4: Placement of Strings**[Input]:** Set of Strings $S = \{S_1, S_2, \dots, S_s\}$ and the interaction graph $G_i(Q_i, E_i)$ **[Output]:** Placement of strings on a two dimensional layout

1. Let S be the set of strings to be placed;
2. Select the string S_j from S with the most connectivity;
3. Place S_j in the layout;
4. $S = S - S_j$;
5. While ($S \neq \emptyset$) do
6. Select the string S_j that is highly connected to the already placed strings;
7. Place S_j in the layout;
8. $S = S - S_j$;

strings, when the number of strings increases the number of routes for ion transportation increases and may make the routing a challenge to design. In this article, a modified version of a maze routing algorithm is used [86] to choose a path for ion transportation.

3.6 Schedule the Netlist

When the layout of all ELUs are generated, the timing information is extracted and the dataflow graph of the circuit is updated with the timing information. Then, the dataflow graph including the timing information is scheduled (Line 10, Algorithm 1). The scheduler implements a greedy scheduling scheme. It maintains a list of operations that have all their dependencies fulfilled and therefore, are ready to be executed. Among the ready instructions, the instruction with the highest priority will be run and is more likely to gain access to the resources it needs. These contested resources include both gates and channels/intersections. Once all the possible instructions are scheduled, time advances until one or more resources are freed and more instructions can be scheduled. This scheduling process continues until the full instruction sequence is executed.

3.7 The Choice of Quantum Error Correcting Code

We have two reasons to choose the Steane code. The first reason is to be fair in comparing our architecture with the Ahsan's work, because they use the Steane code as the quantum error correcting code. The second reason is the good fault-tolerance properties of this code. These properties can be summarized as follows:

- a. **Well-known encoding and error correction procedures:** Encoding and error correction on the encoded qubit block are well-defined processes and can be easily confirmed [87].
- b. **Transversal implementation of Clifford gates:** The Clifford gates can be executed in a bitwise manner, which ensures that the error in one qubit doesn't propagate to multiple qubits in the encoded qubit block.
- c. **Distillation-free demonstration of fault-tolerant non-Clifford gates:** The implementation method for the non-Clifford gates such as Toffoli and T gate is slightly complicated but can be broken down into two steps: magic-state preparation and data injection into the magic state.
- d. **Scheduling on hardware with different types of constraints:** The fault-tolerant procedures of the Steane code can be efficiently scheduled on a hardware with different qubit connectivity constraints specified by the quantum device physical environment. In contrast to the widely famous topological codes [88], which are tailored to the hardware that supports 2D nearest-neighbor interactions, the Steane code can be mapped onto a range of

hardware architectures such as a linear chain of qubits, 2D nearest neighbor hardware [89] and the one which supports flexible physical shuttling mechanism for the qubits for the implementation non-distance fault tolerant quantum computation [12, 13]. Historically, the Steane code has been considered as a premier choice for protecting trapped ion qubits, which are considered as one of the strongest candidates for the realistically achievable large-scale quantum system [12].

Even though the topological codes generally achieve a higher noise threshold to provide greater noise protection against noise as compared to the Steane code, when it comes to the overall resource overhead, reliability and the time to execute large-size quantum algorithms on the trapped ion computer, the Steane code edges ahead of the topological code [90].

- e. **Supporting the variety of noise models:** The Steane code is one of the codes that has been studied in the context of noise models other than depolarization channel. It is also known that the Steane code can be used to correct errors, which are correlated in time [91]. In a quantum hardware where qubits suffer from errors specified by multiple types of noise processes, the Steane code acts as a powerful tool of protection against noise.

It is worthy to note that other concatenated codes can be supported with slight modification in the basic structure of our toolset.

3.8 Time Complexity Analysis

The time complexity of our mapping flow can be calculated as follows. Since the input netlist should be parsed to generate the dataflow graph and the interaction graph, the time complexities of Steps 1 and 2 are $O(n_g)$ where n_g is the number of gates. All edges of the interaction graph should be examined in Step 3; therefore, this step needs $O(n_g)$ examinations. The k -way multilevel partitioning algorithm (Step 4) computes a k -way partitioning of the interaction graph of qubits in $O(n_e) = O(n_g)$ time [50] where n_e is the number of edges of the interaction graph. The time complexity of Algorithm 2 (Line 6) is $O((\frac{n_q}{k})^2)$, where n_q and k are the total number of qubits and the number of ELUs, respectively [51]. Algorithm 3 partitions a sorted string of qubits in $O(\log(\frac{n_q}{k}))$ time. The time complexity of Algorithm 4 is $O(s \log s) = O((\frac{n_q}{k}) \log(\frac{n_q}{k}))$, where s is the number of strings, because first the strings should be sorted based on the connectivity metric and then strings are selected one by one. Step 9 can be performed in $O(s^2) = O((\frac{n_q}{k})^2)$ time. Finally, the time complexity of list-based scheduling algorithm is $O(n_g \log(n_g))$ [92]. Therefore, the overall complexity of the flow can be calculated as

$$3 \times O(n_g) + O(n_g \log(n_g)) + k \times \left(O\left(\log\left(\frac{n_q}{k}\right)\right) + 2 \times O\left(\left(\frac{n_q}{k}\right)^2\right) + O\left(\left(\frac{n_q}{k}\right) \log\left(\frac{n_q}{k}\right)\right) \right) \xrightarrow{k \text{ is constant}} O(n_g \log(n_g) + n_q^2).$$

The time complexity scales log-linearly with the number of gates, but quadratically with the number of qubits in an application. The scheduling step contributes $n_g \log(n_g)$ and the routing and finding qubit linear arrangement steps contribute n_q^2 to the time complexity.

4 AN EXAMPLE

In this section, an example is given to illustrate the SAQIP architecture and the mapping flow. Figure 6(a) shows the QASM instruction sequence operating on qubits $q_0, q_1, q_2, \dots, q_{17}$ for an 8-qubit quantum ripple carry adder [93]. This netlist has 52 gates and 18 qubits. The last qubit of each gate is the target qubit. Figure 6(b) shows the interaction hypergraph of the qubits. The

Fig. 6. (a) Circuit netlist, (b) the interaction hypergraph of qubits, (c) two disjoint subsets generated after the outlier removal step and the first level partitioning.

output of the k -way partitioning step is presented in Figure 6(c). In this example, the qubits have been partitioned into two parts.

To save the paper space, only the qubits assigned to ELU1 are placed and routed. Figure 7 shows the steps taken to place and route the qubits of ELU 1. According to the interaction hypergraph, some gates are performed between qubit q_3 from ELU 2 and qubits q_{12} and q_4 from ELU1. Therefore, q_3 should be transferred from ELU2 to ELU1 when needed. Thus, in addition to the qubits assigned to ELU1, a dummy qubit, called Dummy_ q_3 , is also considered in placement and routing. Figure 7(a) is the Laplacian matrix calculated from the adjacency matrix. The eigenvalues are mentioned in Figure 7(b). The eigenvector V_2 corresponding to the second smallest eigenvalue ($\lambda_2 = 1.00013098006427$) is computed as shown in Figure 7(c). The qubits are sorted and partitioned based on this vector (Figure 7(d)). The vertical dotted lines shown in this figure separate the qubits into three strings. Then, the strings are placed and routed as presented in Figure 7(e) by Algorithm 4. The dataflow graph annotated with delays and the critical path is extracted. Each edge is labeled by the time a qubit requires to move from one gate location to the next gate location plus the delay of the gate located at the head of that edge. Latency of a circuit is the total time that it takes for a circuit to be executed on a particular layout. After a circuit is scheduled on a layout, its total time (Latency) can be calculated. We use the same propositions as SQripT [16, 94] to calculate the execution time. The latency of a circuit can be written as sum of delays on the critical path. Its first component is the sum of delays due to the transportation of qubits of gates located on the critical path, through ballistic shuttling channels inside ELUs. Its second component is the sum of delays due to the logical EPR pair generations for communication, which are on the critical path. Its third component is the sum of delays due to swap operations that are on the critical path and

ELU 1: q4, q12, q5, q13, q6, q14, q7, q15, q17, Dummy_q3

$$\begin{bmatrix} 15 & -4 & -4 & -3 & 0 & 0 & 0 & 0 & -4 \\ -4 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & -3 \\ -4 & 0 & 15 & -4 & -4 & -3 & 0 & 0 & 0 \\ -3 & 0 & -4 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & 0 & 13 & -4 & -2 & -2 & -1 \\ 0 & 0 & -3 & 0 & -4 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 5 & -2 & -1 \\ 0 & 0 & 0 & 0 & -2 & 0 & -2 & 5 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 3 \\ -4 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 7 \end{bmatrix}$$

(a)

$\mathbf{E} = [2.61339634538752\text{E-}15, \mathbf{1.00013098006427}, 3.29524562380479, 4.41484240246885, 6.07056622854316, 7, 9.99999999999999, 14.2304689724185, 16.9063312418839, 21.0824145508165]$

(b)

$\mathbf{V}_2 = [0.304415979778352, 0.405905694889137, 0.0929833812704848, 0.214201586759099, -0.152031097703483, -0.0548635721727811, -0.380157414039859, -0.380157414039859, -0.456202839630227, 0.405905694889137]$

(c)

Fig. 7. (a) Laplacian matrix, (b) eigenvalues, (c) eigenvector V_2 , (d) sorted qubits, and (e) final layout.

Table 1. The Latency Values for Various Physical Operations [16]

Physical operation	Latency (μs)
Single-qubit	1
Two-qubit (CNOT)	10
Three-qubit (Toffoli)	100
Measurement	100
EPR pair Generation	5000

are required in each gate location to access to a qubit. Finally, its fourth component is the sum of delays of gates located on the critical path. The latency of this circuit calculated from the scheduled dataflow graph and based on the values mentioned in Table 1 is $67,327\mu\text{s}$.

5 THE SAQIP PERFORMANCE

To quantify the performance of the SAQIP architecture, a CAD tool, called SAQIPSim, has been developed that (1) places and routes a quantum circuit onto the SAQIP architecture, (2) schedules the sequence of quantum operations, and (3) reports performance metrics such as total execution

Table 2. List of the Benchmarks, with Quantum Gate Count and Number of Qubits Processed in the Circuit

#	Circuit name	Qubit count	Gate count
1	QCLA_64	249	487
2	QRCA_64	130	444
3	QCLA_128	504	996
4	QRCA_128	258	892
5	QCLA_256	1015	2,017
6	QRCA_256	514	1,788
7	QCLA_512	2,038	4,062
8	QRCA_512	3,580	1,026
9	QCLA_1024	8,155	4,085
10	QRCA_1024	7,164	2,050
11	QCLA_2048	16,344	8,180
12	QRCA_2048	14,332	4,098
13	QFT_256	256	32,896
14	QFT_128	128	8,256
15	RC 1	500	6,050
16	RC 2	2,000	15,400

time. The device parameters mentioned in Table 1 are used in the SAQIPSim tool. Although the assumed values for the parameters are optimistic, those can be achieved in the near future through rapid technology advancement [95, 96].

5.1 Benchmark Circuits

To evaluate the proposed architecture and compare it with the best architecture in the literature [16], quantum adders with different size, two QFT⁹ circuits, and two randomly generated quantum circuit are used. All results of this section are obtained on a 2.6GHz Intel Core2 Duo with 4 gigabytes of memory.

The first step of Shor’s quantum factorization algorithm is called quantum modular exponentiation [97, 98], which can be constructed from quantum adder circuits [99]. In this article, two candidate adders QRCA¹⁰ and QCLA¹¹ that use two different addition strategies are chosen to evaluate SAQIPSim. QRCA has a linear-depth structure. An n -bit QRCA uses about $2n$ qubits to perform $2n$ Toffoli and $5n$ CNOT gates [93]. However, QCLA has a logarithmic-depth structure using $4n$ qubits to perform up to n concurrently executable gates [100]. This circuit is composed of about $5n-3\log_2 n$ CNOT and Toffoli gates for n -bit addition. The benchmarks are shown in Table 2. The number of qubits and the number of gates are mentioned in the table.

5.2 Simulation Results

Table 3 shows the latency of the benchmark circuits achieved by the SAQIPSim compared with the best tool in the literature, SQripT toolbox¹² [16]. The column “SAQIPSim (SQripT Config.)” under “Latency” shows the latency of circuits obtained by our toolbox with the configuration mentioned as the *optimal architecture configuration* in the SQripT toolbox. The column “SAQIPSim

⁹Quantum Fourier Transform.

¹⁰Quantum Ripple Carry Adder.

¹¹Quantum Carry Look-ahead Adder.

¹²The source code of the toolbox is available on <https://users.cs.duke.edu/~ahsan/SQripT/ToolBox/>.

Table 3. The latency of the Benchmark Circuits achieved by the SAQIPSim Toolbox Compared with the SQripT toolbox [16]

#	Circuit name	Latency (s)			Improvement (%)	
		SAQIPSim (SQripT Config.)	SAQIPSim (Optimal Config.)	SQripT [16]	SQripT Config.	Optimal Config.
1	QCLA_64	0.12	0.12	1.53	92.16	92.16
2	QRCA_64	0.22	0.34	2.32	90.52	85.34
3	QCLA_128	0.14	0.13	1.78	99.88	92.70
4	QRCA_128	1.37	0.68	4.62	70.35	85.28
5	QCLA_256	0.24	0.14	1.93	87.56	92.75
6	QRCA_256	2.91	1.4	9.31	68.74	84.96
7	QCLA_512	0.26	0.2	2.28	88.60	91.23
8	QRCA_512	6.15	2.73	18.48	66.72	85.23
9	QCLA_1024	0.24	0.17	2.66	90.98	93.61
10	QRCA_1024	12.39	5.78	36.84	66.37	84.31
11	QCLA_2048	0.31	0.22	81.81	99.62	99.73
12	QRCA_2048	24.71	22.17	260.86	90.53	91.50
13	QFT_256	204.6	198.2	370.15	44.73	46.45
14	QFT_128	98.52	95.2	152.75	35.50	37.68
15	RC 1	15.21	13.15	40.66	62.59	67.66
16	RC 2	75.7	70.5	206.65	63.37	65.88
Average					75.65	81.03

(Optimal Config.)” under “Latency” contains the best latency of circuits achieved by our toolbox. The column “SQripT” shows the latency of circuits obtained by the best prior toolbox SQripT. The “SQripT Config” and “Optimal Config.” under “Improvement” include the improvement resulted from our architecture over SQripT with the best SQripT configuration and our best configuration, respectively.

The results show that the SAQIP architecture and our mapping flow can dramatically decrease the circuit latency over the best previous architecture and toolbox. It seems that the QCLA circuits benefit greatly from the SAQIP architecture. The experiments show that using the k-way method for partitioning qubits into ELU and the eigenvector approach for segmenting qubits of each ELU into strings results in a significant decrease in the latency of executing of a circuit.

It is worthy to note that the structure of a circuit has a great impact on the effectiveness of a mapping flow. The QFT applications have more regular structures than the other benchmark circuits. These circuits can be partitioned into recognizable parts. In part p , qubit q_p is the target qubit of gates and the control qubits of those are q_{p+1} , q_{p+2} , ..., and q_n , respectively. This regular structure causes that Ahsan’s approach behaves for the QFT applications more efficient than for the other benchmark circuits. Therefore, the improvement percentages for these circuits on average are less than those for the other benchmark circuits.

Quantum data is transferred inside ELUs by ballistically shuttling ions and between ELUs by photons via the optical switch network. Our mapping algorithm focuses on this fact that minimizing the communication between ELUs and inside ELUs have great effects on the circuit latency. Since communication between ELUs has a significant delay in comparison to the delay of other physical operations, the main objective of the algorithm that partitions qubits into ELUs (multilevel k-way hypergraph partitioning algorithm [82]) is minimizing the interaction between

Table 4. The Runtime of our Mapping Flow in Comparison to Ahsan's Flow (in s)

#	Circuit name	SQripT [16]	SAQIPSim	Improvement (%)
1	QCLA_64	1.69	1.43	15.38
2	QRCA_64	1.43	0.56	60.84
3	QCLA_128	2.82	2.23	20.92
4	QRCA_128	3.25	1.06	67.38
5	QCLA_256	5.58	4.21	24.55
6	QRCA_256	9.98	2.11	78.86
7	QCLA_512	13.59	9.88	27.30
8	QRCA_512	49.38	4.21	91.47
9	QCLA_1024	35.42	22.68	35.97
10	QRCA_1024	339.97	8.43	97.52
11	QCLA_2048	109.63	68.52	37.50
12	QRCA_2048	2616.92	8.55	99.67
13	QFT_256	75.46	64.36	14.71
14	QFT_128	14.6	13.65	6.51
15	RC 1	24.75	18.6	24.85
16	RC 2	47.95	30.21	37.00
Average				46.28

ELUs. However, since qubits moves ballistically to communicate with other qubits inside ELUs, minimizing the distance that each qubit traverses, has a direct effect on the circuit latency. Therefore, the proposed partitioning algorithm (Algorithm 3) and placement one (Algorithm 4) (that partitions and places the qubits assigned to an ELU) keep most interacting qubits close together. These algorithms play great roles in our improvement.

Table 4 shows the runtime of the mapping program for the benchmark circuits in detail. The main reason of this great improvement is that Ahsan's toolset solves the linear arrangement problem to partition all qubits into ELUs, while we use the k-way partitioning technique for partitioning. Our approach has smaller runtime than Ahsan's approach. It is worthy to note that in each group of benchmarks (QCLA and QRCA), the improvement increases when the number of qubits increases. The results show that our mapping flow is good not only at the latency but also at the runtime.

Table 5 reports the area consumption for benchmark circuits obtained by our toolset and Ahsan's one. The number of macroblocks consumed for each benchmark circuit is reported in this table. Unlike the Ahsan's architecture that uses the similar QLA-like ELUs, ours maps sub-circuits onto full-custom fabrics. Therefore, circuits mapped by our flow theoretically consume less area than those mapped by the Ahsan's flow. As results shows, our approach decreases area (number of macroblocks) by about 11.62% (on average) for the attempted benchmark circuits.

6 CONCLUSIONS

In this article, an architecture called SAQIP was presented based on the MUSIQC hardware to build large quantum computers using new technology features that are available today. It benefits from the capabilities recently provided by advancements in ion-trap technology to generate layouts with better latency and area. It is composed of full-custom blocks connected by the photonic network. It gives better area and delay for the quantum circuits in comparison to Ahsan's SQripT architecture. We use area and latency, because, to a first approximation, lower area and lower latency are likely

Table 5. The Area Consumption of Benchmark Circuits Achieved by the SAQIPSim Toolbox Compared with the SQripT Toolbox [16] (in Macroblocks)

#	Circuit name	SQripT [16]	SAQIPSim	Improvement (%)
1	QCLA_64	192	185	3.65
2	QRCA_64	96	88	8.33
3	QCLA_128	368	340	7.61
4	QRCA_128	320	309	3.44
5	QCLA_256	1,600	1,470	8.13
6	QRCA_256	1,328	1,250	5.87
7	QCLA_512	6,464	5,855	9.42
8	QRCA_512	5,248	4,870	7.20
9	QCLA_1024	26,032	20,751	20.29
10	QRCA_1024	20,976	18,050	13.95
11	QCLA_2048	42,240	32,024	24.19
12	QRCA_2048	37,248	30,448	18.26
13	QFT_256	21,664	15,630	27.85
14	QFT_128	1,904	1,650	13.34
15	RC 1	2,480	2,260	8.87
16	RC 2	19,664	18,564	5.59
Average				11.62

to decrease decoherence and error probability. Moreover, a design flow was proposed to map a quantum circuit onto the proposed architecture. The experimental results show that the proposed architecture and flow can improve the latency and the area by up to 99% and 27%, respectively, for the attempted circuits.

ACKNOWLEDGMENTS

We acknowledge support from the Iran National Science Foundation under Grant No. 94812110.

REFERENCES

- [1] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich et al. 2017. Blueprint for a microwave trapped ion quantum computer. *Sci. Adv.* 3 (2017), e1601540.
- [2] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman et al. 2017. Experimental comparison of two quantum computing architectures. In *Proceedings of the National Academy of Sciences*, Vol. 114, 3305–3310.
- [3] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe. 2016. Demonstration of a small programmable quantum computer with atomic qubits. *Nature* 536 (2016) 63.
- [4] T. Harty, D. Allcock, C. J. Ballance, L. Guidoni, H. Janacek, N. Linke et al. 2014. High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit. *Phys. Rev. Lett.* 113 (2014) 220501.
- [5] A. Myerson, D. Szwer, S. Webster, D. Allcock, M. Curtis, G. Imreh et al. 2008. High-fidelity readout of trapped-ion qubits. *Phys. Rev. Lett.* 100 (2008), 200502.
- [6] N. Akerman, N. Navon, S. Kotler, Y. Glickman, and R. Ozeri. 2015. Universal gate-set for trapped-ion qubits using a narrow linewidth diode laser. *New J. Phys.* 17 (2015) 113060.
- [7] N. Timoney, I. Baumgart, M. Johanning, A. Varón, M. B. Plenio, A. Retzker et al. 2011. Quantum gates and memory using microwave-dressed states. *Nature* 476 (2011), 185.
- [8] Y. Wang, M. Um, J. Zhang, S. An, M. Lyu, J.-N. Zhang et al. 2017. Single-qubit quantum memory exceeding ten-minute coherence time. *Nature Photon.* 11 (2017) 646.
- [9] C. Langer, R. Ozeri, J. D. Jost, J. Chiaverini, B. DeMarco, A. Ben-Kish et al. 2005. Long-lived qubit memory using atomic ions. *Phys. Rev. Lett.* 95 (2005), 060502.
- [10] D. Kielpinski, C. Monroe, and D. J. Wineland. 2002. Architecture for a large-scale ion-trap quantum computer. *Nature* 417 (2002), 709–711.

- [11] N. D. Guise, S. D. Fallek, K. E. Stevens, K. Brown, C. Volin, A. W. Harter et al. 2015. Ball-grid array architecture for microfabricated ion traps. *J. Appl. Phys.* 117 (2015), 174901.
- [12] C. Monroe and J. Kim. 2013. Scaling the ion trap quantum processor. *Science* 339 (2013), 1164–1169.
- [13] C. Monroe, R. Raussendorf, A. Ruthven, K. Brown, P. Maunz, L.-M. Duan et al. 2014. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Phys. Rev. A* 89 (2014), 022317.
- [14] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish et al. 2011. 14-qubit entanglement: Creation and coherence. *Phys. Rev. Lett.* 106 (2011), 130506.
- [15] M. Ahsan and J. Kim. 2015. Optimization of quantum computer architecture using a resource-performance simulator. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. 1108–1113.
- [16] M. Ahsan, R. V. Meter, and J. Kim. 2016. Designing a million-qubit quantum computer using a resource performance simulator. *ACM J. Emerg. Technol. Comput. Syst.* 12 (2016) 39.
- [17] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang. 2005. A quantum logic array microarchitecture: Scalable quantum data movement and computation. In *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05)*. 12.
- [18] R. v. Meter and M. Oskin. 2006. Architectural implications of quantum computing technologies. *ACM J. Emerg. Technol. Comput. Syst.* 2 (2006), 31–63.
- [19] J. Levy. 2001. Quantum-information processing with ferroelectrically coupled quantum dots. *Phys. Rev. A* 64 (2001), 052306.
- [20] A. G. Fowler, S. J. Devitt, and L. C. Hollenberg. 2004. Implementation of Shor's algorithm on a linear nearest neighbour qubit array. *Arxiv Preprint Quant-ph/0402196* (2004).
- [21] P. Aliferis and A. W. Cross. 2007. Subsystem fault tolerance with the bacon-shor code. *Phys. Rev. Lett.* 98 (2007), 220502.
- [22] J. E. Levy, A. Ganti, C. A. Phillips, B. R. Hamlet, A. J. Landahl, T. M. Gurrieri et al. 2009. Brief announcement: The impact of classical electronics constraints on a solid-state logical qubit memory. In *Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures*. 166–168.
- [23] J. E. Levy, M. S. Carroll, A. Ganti, C. A. Phillips, A. J. Landahl, T. M. Gurrieri et al. 2011. Implications of electronics constraints for solid-state quantum error correction and quantum circuit failure probability. *New J. Phys.* 13 (2011), 083021.
- [24] A. Farghadan and N. Mohammadzadeh. 2017. Quantum circuit physical design flow for 2D nearest-neighbor architectures. *Int. J. Circuit Theory Appl.* 45, 7 (2017), 989–1000.
- [25] Y. S. Weinstein, C. S. Hellberg, and J. Levy. 2005. Quantum-dot cluster-state computing with encoded qubits. *Phys. Rev. A* 72 (2005), 020304.
- [26] R. Stock and D. F. James. 2009. Scalable, high-speed measurement-based quantum computer using trapped ions. *Phys. Rev. Lett.* 102 (2009), 170501.
- [27] S. J. Devitt, A. G. Fowler, T. Tilma, W. J. Munro, and K. Nemoto. 2010. Classical processing requirements for a topological quantum computing system. *Int. J. Quant. Info.* 8 (2010), 121–147.
- [28] S. J. Devitt, A. M. Stephens, W. J. Munro, and K. Nemoto. 2011. Integration of highly probabilistic sources into optical quantum architectures: Perpetual quantum computation. *New J. Phys.* 13 (2011), 095001.
- [29] X.-C. Yao, T.-X. Wang, H.-Z. Chen, W.-B. Gao, A. G. Fowler, R. Raussendorf et al. 2012. Experimental demonstration of topological error correction. *Nature*, 482 (2012), 489–494.
- [30] K. Nemoto, M. Trupke, S. J. Devitt, A. M. Stephens, B. Scharfenberger, K. Buczak et al. 2014. Photonic architecture for scalable quantum information processing in diamond. *Phys. Rev. X* 4 (2014), 031022.
- [31] A. Paler, S. J. Devitt, K. Nemoto, and I. Polian. 2014. Mapping of topological quantum circuits to physical hardware. *Sci. Rep.* 4 (2014).
- [32] J. Sau and M. Barkeshli. 2016. Physical architecture for a universal topological quantum computer based on a network of majorana nanowires. In *Proceedings of the APS Meeting*.
- [33] X. Fu, M. Rol, C. Bultink, J. Van Someren, N. Khammassi, I. Ashraf et al. 2017. An experimental microarchitecture for a superconducting quantum processor. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. 813–825.
- [34] X. Fu, L. Rieseboos, M. Rol, J. van Straten, J. van Someren, N. Khammassi et al. 2018. eQASM: An executable quantum instruction set architecture. *Arxiv Preprint Arxiv:1808.02449* (2018).
- [35] A. Zulehner, A. Paler, and R. Wille. 2018. An efficient methodology for mapping quantum circuits to the IBM QX architectures. [Online]. <https://arxiv.org/abs/1712.04722v3>.
- [36] M. Y. Siraichi, V. F. d. Santos, S. Collange, and F. M. Q. Pereira. 2018. Qubit allocation. In *Proceedings of the International Symposium on Code Generation and Optimization*. 113–125.
- [37] IBM. 2019. *IBM QX Backend Information*. Retrieved from <https://www.research.ibm.com/ibm-q/>.
- [38] M. Mariani, H. Wang, T. Yamamoto, M. Neeley, R. C. Bialczak, Y. Chen et al. 2011. Implementing the quantum von neumann architecture with superconducting circuits. *Science* 334 (2011), 61–65.

- [39] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiawicz. 2007. Automated generation of layout and control for quantum circuits. In *Proceedings of the 4th International Conference on Computing Frontiers*. 83–94.
- [40] M. G. Whitney, N. Isailovic, Y. Patel, and J. Kubiawicz. 2009. A fault tolerant, area efficient architecture for shor's factoring algorithm. *ACM SIGARCH Comput. Architect. News* 37 (2009), 383–394.
- [41] N. Isailovic, M. Whitney, Y. Patel, and J. Kubiawicz. 2008. Running a quantum circuit at the speed of data. In *ACM SIGARCH Computer Architecture News*. 177–188.
- [42] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiawicz. 2006. Interconnection networks for scalable quantum computers. In *ACM SIGARCH Computer Architecture News*. 366–377.
- [43] D. Gottesman and I. L. Chuang. 1999. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature* 402 (1999), 390–393.
- [44] K. Svore, A. Cross, A. Aho, I. Chuang, and I. Markov. 2004. Toward a software architecture for quantum computing design tools. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages (QPL'04)*. 145–162.
- [45] K. M. Svore, A. V. Aho, A. W. Cross, I. Chuang, and I. L. Markov. 2006. A layered software architecture for quantum computing design tools. *IEEE Comput.* 39 (2006), 74–83.
- [46] T. S. Metodi, D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang. 2005. A general purpose architectural layout for arbitrary quantum computations. In *Proceedings of the Conference on Defense and Security*. 91–102.
- [47] T. Metodi, D. Thaker, A. Cross, F. T. Chong, and I. L. Chuang. 2005. QLA: Quantum logic array microarchitecture, a brief overview. *Recreation Pool Lodge* 2 (2005).
- [48] D. D. Thaker, T. S. Metodi, A. W. Cross, I. L. Chuang, and F. T. Chong. 2006. Quantum memory hierarchies: Efficient designs to match available parallelism in quantum computing. In *ACM SIGARCH Computer Architecture News*. 378–390.
- [49] M. J. Dousti, A. Shafaei, and M. Pedram. 2014. Squash: A scalable quantum mapper considering ancilla sharing. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*. 117–122.
- [50] J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes, D. Kudrow, K. R. Brown et al. 2015. Compiler management of communication and parallelism for quantum computation. In *ACM SIGARCH Computer Architecture News*. 445–456.
- [51] R. Risque and A. Jog. 2016. Characterization of quantum workloads on SIMD architectures. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC'16)*. 1–9.
- [52] F. T. Chong, D. Franklin, and M. Martonosi. 2017. Programming languages and compiler design for realistic quantum hardware. *Nature* 549 (2017), 180.
- [53] M. Spišiak and J. Kollár. 2017. Quantum programming: A review. In *Proceedings of the IEEE 14th International Scientific Conference on Informatics*. 353–358.
- [54] M. Soeken, T. Haener, and M. Roetteler. 2018. Programming quantum computers using design automation. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'18)*. 137–146.
- [55] D. Maslov. 2017. Basic circuit compilation techniques for an ion-trap quantum machine. *New J. Phys.* 19 (2017), 023035.
- [56] H. Goudarzi, M. J. Dousti, A. Shafaei, and M. Pedram. 2014. Design of a universal logic block for fault-tolerant realization of any logic operation in trapped-ion quantum circuits. *Quant. Info. Process.* 13 (2014), 1267–1299.
- [57] M.-C. Kim, D.-J. Lee, and I. L. Markov. 2012. SimPL: An effective placement algorithm. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 31 (2012), 50–60.
- [58] K. R. Brown, J. Kim, and C. Monroe. 2016. Co-designing a scalable quantum computer with trapped atomic ions. *Npj Quant. Info.* 2 (2016), 16034.
- [59] N. Mohammadzadeh. 2016. Physical design of quantum circuits in ion trap technology—a survey. *Microelectron. J.* 55 (2016), 116–133.
- [60] J. Cirac, P. Zoller, H. Kimble, and H. Mabuchi. 1997. Quantum state transfer and entanglement distribution among distant nodes in a quantum network. *Phys. Rev. Lett.* 78 (1997), 3221.
- [61] S. Van Enk, H. Kimble, J. Cirac, and P. Zoller. 1999. Quantum communication with dark photons. *Phys. Rev. A* 59 (1999), 2659.
- [62] A. M. Steane and D. Lucas. 2000. Quantum computing with trapped ions, atoms and light. *Arxiv Preprint Quant-ph/0004053* (2000).
- [63] L.-M. Duan, M. Lukin, J. I. Cirac, and P. Zoller. 2001. Long-distance quantum communication with atomic ensembles and linear optics. *Nature* 414 (2001), 413–418.
- [64] R. Van Meter, K. Nemoto, and W. Munro. 2007. Communication links for distributed quantum computation. *IEEE Trans. Comput.* 56 (2007), 1643–1653.
- [65] L.-M. Duan and C. Monroe. 2010. Colloquium: Quantum networks with trapped ions. *Rev. Mod. Phys.* 82 (2010), 1209.
- [66] J. Kim and C. Kim. 2009. Integrated optical approach to trapped ion quantum computation. *Quant. Info. Comput.* 9 (2009), 181–202.
- [67] S. J. Devitt, W. J. Munro, and K. Nemoto. 2011. High performance quantum computing. *Progr. Info.* 8 (2011), 49–55.

- [68] R. Van Meter, K. Nemoto, W. Munro, and K. M. Itoh. 2006. Distributed arithmetic on a quantum multicomputer. In *ACM SIGARCH Computer Architecture News*. 354–365.
- [69] R. Van Meter, T. D. Ladd, A. G. Fowler, and Y. Yamamoto. 2010. Distributed quantum computation architecture using semiconductor nanophotonics. *Int. J. Quant. Info.* 8 (2010), 295–323.
- [70] T. Monz, K. Kim, W. Hänsel, M. Riebe, A. Villar, P. Schindler et al. 2009. Realization of the quantum toffoli gate with trapped ions. *Phys. Rev. Lett.* 102 (2009), 040501.
- [71] I. Buluta and F. Nori. 2009. Quantum simulators. *Science* 326 (2009), 108–111.
- [72] C. Figgatt, D. Maslov, K. Landsman, N. Linke, S. Debnath, and C. Monroe. 2017. Complete 3-qubit grover search on a programmable quantum computer. *Nature Commun.* 8 (2017), 1918.
- [73] T. Choi, S. Debnath, T. Manning, C. Figgatt, Z.-X. Gong, L.-M. Duan et al. 2014. Optimal quantum control of multi-mode couplings between trapped ion qubits for scalable entanglement. *Phys. Rev. Lett.* 112 (2014), 190502.
- [74] L. Postler, Á. Rivas, P. Schindler, A. Erhard, R. Stricker, D. Nigg et al. 2018. Experimental quantification of spatial correlations in quantum dynamics. *Arxiv Preprint Arxiv:1806.08088* (2018).
- [75] P. Kaufmann, T. F. Gloger, D. Kaufmann, M. Johanning, and C. Wunderlich. 2018. High-fidelity preservation of quantum information during trapped-ion transport. *Phys. Rev. Lett.* 120 (2018), 010501.
- [76] D. Hucul, M. Yeo, W. Hensinger, J. Rabchuk, S. Olmschenk, and C. Monroe. 2008. On the transport of atomic ions in linear and multidimensional ion trap arrays. *Quant. Info. Comput.* 8 (2008), 501–578.
- [77] M. Rowe, J. A. Ben-Kish, B. Demarco, D. Leibfried, V. Meyer, J. Beall et al. 2002. Transport of quantum states and separation of ions in a dual RF ion trap. *Quant. Info. Comput.* 2 (2002), 257–271.
- [78] H. Häffner, C. F. Roos, and R. Blatt. 2008. Quantum computing with trapped ions. *Phys. Rep.* 469 (2008), 155–203.
- [79] Y. Cao, G. Wang, H. Liu, and C. Sun. 2018. Implementation of a toffoli gate using an array of coupled cavities in a single step. *Sci. Rep.* 8 (2018), 5813.
- [80] A. M. Steane. 1996. Error correcting codes in quantum theory. *Phys. Rev. Lett.* 77 (1996), 793.
- [81] IBM. 2017. *IBM SPSS Analytics Toolkit*.
- [82] G. Karypis and V. Kumar. 2000. Multilevel k-way hypergraph partitioning. *VLSI Design* 11 (2000), 285–300.
- [83] S. B. Horton. 1997. The optimal linear arrangement problem: Algorithms and approximation. Doctoral Dissertation. School of Industrial and Systems Engineering, Georgia Institute of Technology.
- [84] M. Juvan and B. Mohar. 1992. Optimal linear labelings and eigenvalues of graphs. *Discrete Appl. Math.* 36 (1992), 153–168.
- [85] Q. Liu and M. Marek-Sadowska. 2005. Pre-layout physical connectivity prediction with application in clustering-based placement. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'05)*. 31–37.
- [86] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. Wong. 2005. Redundant-via enhanced maze routing for yield improvement. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. 1148–1151.
- [87] M. A. Nielsen and I. L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.
- [88] A. Y. Kitaev. 2003. Fault-tolerant quantum computation by anyons. *Ann. Phys.* 303 (2003), 2–30.
- [89] K. M. Svore, D. P. DiVincenzo, and B. M. Terhal. 2006. Noise threshold for a fault-tolerant two-dimensional lattice architecture. *Arxiv Preprint Quant-ph/0604090* (2006).
- [90] M. Suchara, J. Kubiatowicz, A. Faruque, F. T. Chong, C.-Y. Lai, and G. Paz. 2013. QuRE: The quantum resource estimator toolbox. In *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD'13)*. 419–426.
- [91] F. Lu and D. C. Marinescu. 2007. Quantum error correction of time-correlated errors. *Quant. Info. Process.* 6 (2007), 273–293.
- [92] A. M. Sllame and V. Drabek. 2002. An efficient list-based scheduling algorithm for high-level synthesis. In *Proceedings of the Euromicro Symposium on Digital System Design*. 316–323.
- [93] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. 2004. A new quantum ripple-carry addition circuit. *Arxiv Preprint Quant-ph/0410184* (2004).
- [94] M. Ahsan. 2016. SQripT Toolset. Retrieved from <https://users.cs.duke.edu/~ahsan/>.
- [95] D. Moehring. 2017. IONQ. Retrieved from <https://ionq.co/>.
- [96] D. Wineland. NIST Ion Storage Group. Retrieved from <https://www.nist.gov/pml/time-and-frequency-division/ion-storage>.
- [97] R. Van Meter and C. Horsman. 2013. A blueprint for building a quantum computer. *Commun. ACM* 56 (2013) 84–93.
- [98] V. Vedral, A. Barenco, and A. Ekert. 1996. Quantum networks for elementary arithmetic operations. *Phys. Rev. A* 54 (1996), 147.
- [99] P. W. Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* 41 (1999), 303–332.
- [100] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore. 2004. A logarithmic-depth quantum carry-lookahead adder. *Arxiv Preprint Quant-ph/0406142* (2004).

Received July 2018; revised January 2019; accepted February 2019