Microelectronics Journal 55 (2016) 116–133

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Physical design of quantum circuits in ion trap technology – A survey

Naser Mohammadzadeh

Department of Computer Engineering, Shahed University, Tehran, Iran

ARTICLE INFO

Article history:
Received 28 December 2015
Received in revised form
22 June 2016
Accepted 1 July 2016
Available online 21 July 2016

Keywords:
Quantum circuit design
Physical design
Placement
Routing
Scheduling
Ion trap technology

ABSTRACT

Quantum circuits have indicated incredible potential for having the capacity to take care of specific issues, which are unmanageable on classical machines. Research in quantum circuit design distinguishes between logic synthesis and physical design. In this survey, we review the approaches, exact and heuristic, proposed for physical design automation of quantum circuits in ion-trap technology that is one of the most advanced quantum technologies. We finish up the review by sketching out major open problems for quantum circuit design.

1. Introduction

As feature sizes in CMOS[1] technology shrink into the 10s of nanometer range, “quantum effects” should be managed [1]. On the other hand, quantum effects such as superposition and entanglement are amplified and utilized in a quantum computer. Quantum circuits will have the capacity to solve some important mathematics and physics problems with fascinating asymptotic improvements [2–5]. A quantum computer needs a large number of qubits and quantum gates to tackle a complex job such as factoring large numbers. To manage the complexity of the big systems, researchers divide the design flow into two main processes: logic synthesis and physical design. The logic synthesis process takes a design description as input and creates a technology-dependent gate-level netlist as output. On the other hand, the physical design process takes the output of the synthesis process and generates a specific layout constructed from the building blocks of the target technology.

The finding of effective quantum algorithms in the mid-1990s [6] and remarkable progress in quantum technologies motivate research on physical design. Extensive research has concentrated on finding physical systems that can provide a large number of qubits while satisfying the scalability criteria [6]. Ion traps have been the physical system of choice [7] to demonstrate the most advanced quantum logic operations [8–13]. A preliminary architecture has been proposed for assembling a large number of ions into a multiplexed trap on a chip [14]. Much research has been done on the physical design of quantum circuits in ion trap technology. This review talks about methodologies, architectures, optimization techniques, open issues, and future directions related to the physical design of quantum circuits in ion trap technology.

Some papers presented physical principles of ion traps such as specific electrode sizing and geometry, and exact voltage levels necessary for trapping and movement, and technology choices for building a large-scale ion trap quantum information processor (QIP) such as which ion species are used [14–20]. The main concentration of those papers is on the physical requirements of such a system. Such papers about the physics of the ion trap technology are not the purpose of this paper. This paper mainly focuses on the methods proposed for automation of the quantum circuit physical design in ion trap technology. The papers discussed in this survey use an abstract model of ion trap technology. They encapsulate all physical details within some building blocks.

A few works were performed on the physical design in other technologies. In research performed by Maslov et al. [21], an efficient heuristic algorithm was proposed for quantum circuit placement in the NMR[2] technology. In the other work, Shafaei et al. [22] formulated the qubit placement on 2D optical lattice by mixed integer programming. Lin et al. [23] proposed a physical design-

E-mail address: mohammadzadeh@shahed.ac.ir

[1] Complementary metal oxide semiconductor.

[2] Nuclear magnetic resonance.

http://dx.doi.org/10.1016/j.mejo.2016.07.001
0026-2692/© 2016 Elsevier Ltd. All rights reserved.

aware quantum circuit synthesis methodology in the optical lattice technology, called PAQCS[3], that includes two algorithms for qubit placement and channel routing.

The rest of this paper is organized as follows: basic concepts are presented in Section 2, followed by an introduction to the ion trap technology in Section 3. Section 4 includes the design flow for quantum circuits. Moreover, works on the physical design are described in that section. Software tools developed for the physical design are presented in Section 5. Finally, Section 6 concludes the paper, and gives the future directions and open problems in quantum circuit design.

2. Background

In this section, quantum bit, quantum logic gates, and quantum circuits are introduced. Some basic concepts that can be helpful in understanding the rest of the paper, are also mentioned.

2.1. Quantum bit (Qubit)

The bit is the key concept of classical computation and information. Quantum information processing is based upon an analogous concept, the quantum bit, or qubit for short [24]. In fact, a qubit is a unit vector in a two-dimensional Hilbert space. The state of a qubit, $|\psi\rangle \in H^2$, can be represented using the canonical base $|0\rangle$ and $|1\rangle$ as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad |\alpha|^2 + |\beta|^2 = 1,$$

Therein, the numbers α and β are complex numbers. These linear combinations of states are often called superposition. The special states $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ are known as computational basis states, and form an orthonormal basis for the vector space. What the vectors $|0\rangle$and $|1\rangle$ physically mean depends on the physical demonstration used for quantum-information processing. For example, the vectors may represent spin states of an electron, $|0\rangle = |\uparrow\rangle$and $|1\rangle = |\downarrow\rangle$. Electrons are replaced by nuclei with spin $\frac{1}{2}$ in NMR quantum technology [25].

2.1.1. Physical qubit

In ion trap technology, a physical qubit is represented by a single positively charged ion.

2.1.2. Logical qubit

A logical qubit is a bit of data used in the computation. It may be physically encoded in some number of physical qubits.

2.1.3. Ancilla qubit

Ancillae are helper qubits used in quantum computation. They include qubits that are created, used, and recycled as a part of a computation. The main usage of these qubits is in quantum error detection/correction algorithms.

2.1.4. Quantum register

An ordered set of a finite number of qubits is called a quantum register.

2.2. Quantum gate

An n-qubit quantum gate is an operator which performs a $2^n \times 2^n$ unitary operation $\mathbf{G}$ on n qubits in a particular period of time. A matrix $\mathbf{G}$ is unitary if $\mathbf{G}\mathbf{G}^{\dagger} = \mathbf{I}$ where $\mathbf{G}^{\dagger}$is the conjugate transpose of $\mathbf{G}$ and $\mathbf{I}$ is the identity matrix. The number of input qubits of a quantum gate should be equal to the number of its output qubits. The inverse of a quantum gate g with a unitary matrix $\mathbf{G_g}$ shown by g^{-1} implements the unitary matrix $\mathbf{G_g^{-1}}$. Unitary operators are often represented by particular schematic symbols, which are useful in the quantum circuit design. Basic quantum gates are shown in Fig. 1. A group of multi-qubit logic gates is the controlled-U gates. These gates have some control qubits and a target qubit. If all control qubits are set to 1, the unitary matrix $\mathbf{U}$ is applied to the target qubit. The set of reversible gates whose matrix elements are just 0s and 1s is part of the set of quantum gates [26].

Quantum wires connect gates together and move qubits forward in time or space. A quantum bit with unknown state cannot be cloned; therefore, quantum wires cannot fanout. Matrix multiplication and tensor product are used to model composition of gates in series and in parallel respectively.

2.2.1. Macro gate

A macro gate has more than three inputs. For example, a C^4NOT that has four control lines and one target line is known as a macro gate [27].

2.2.2. Auxiliary qubit

An auxiliary qubit is a qubit that is not in the set of primary inputs of a macro gate but used to decompose the macro gate into primitive gates. The main feature of auxiliary qubits is that their values before and after a macro gate are equal [6].

2.3. Quantum circuit

The quantum circuit can be used as a computational model similar to a modern digital circuit to represent a quantum algorithm. A quantum circuit comprises of qubits, quantum gates, quantum wires, and qubit measurements. The quantum circuit for a 4-bit Quantum Fourier Transform (QFT) is shown in Fig. 2, which includes the Hadamard and controlled phase shift gates. In quantum circuit model, time goes from left to right and each line shows the evolution of each qubit through time. If each gate in a quantum circuit is inverted and the order of gates is reversed, the inverse of the circuit will be constructed. All quantum operations in a quantum circuit are reversible, except for measurements that are often deferred to the end of computation.

2.3.1. Universal set of gates

A universal set of quantum gates is a set of gates that any unitary operation may be implemented to arbitrary precision by a circuit, including only those gates [6]. For example, Hadamard, phase, CNOT, and $\frac{\pi}{8}$ gates make a universal set.

2.3.2. Stabilizer circuits and states

Stabilizer circuits are a valued subclass of quantum circuits, which can be simulated on classical computers efficiently in polynomial-time and by keeping track of the Pauli operators that stabilize the quantum state. Such stabilizer operators are kept up during simulation and uniquely show stabilizer states up to an unobservable global phase factor. Therefore, this technique offers an exponential improvement over the computational resources needed to simulate stabilizer circuits using vector-based representations. Most research [6,29–31] has been done on stabilizer circuits because of their broad applications in quantum error correcting codes and quantum fault-tolerant architectures.

2.3.3. Quantum threshold theorem

One of the most important theorems in the quantum computation is quantum threshold one [32]. It states that an arbitrarily

[3] Physical design-aware fault-tolerant quantum circuit synthesis.

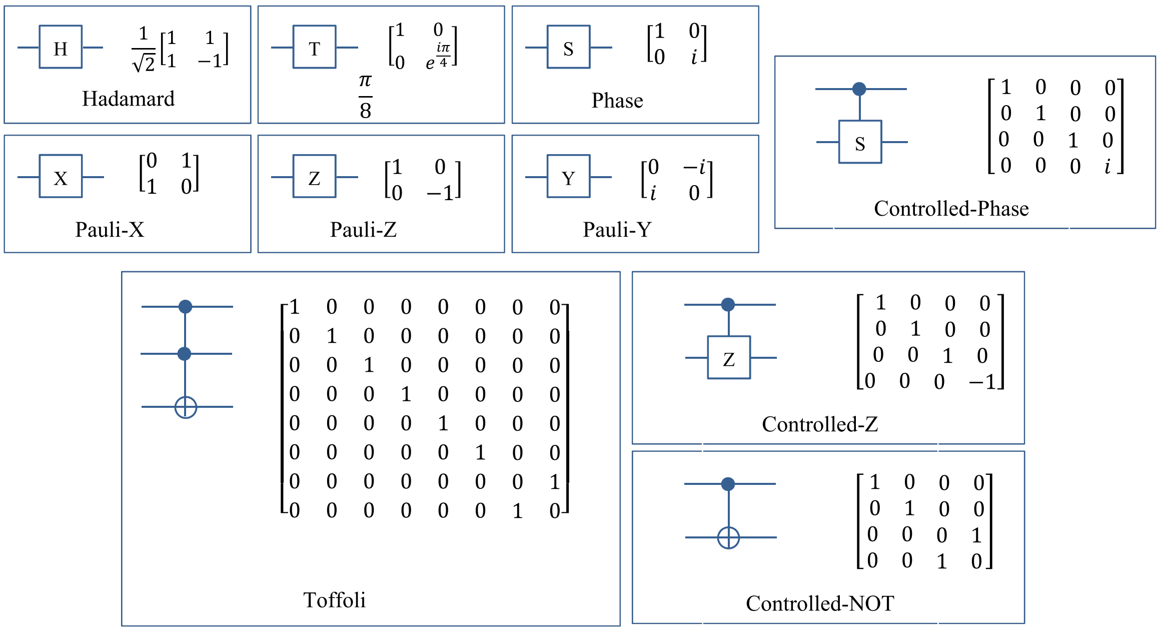

H
$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$
Hadamard
T
$\begin{bmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{bmatrix}$
$\frac{\pi}{8}$
S
$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$
Phase
X
$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
Pauli-X
Z
$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$
Pauli-Z
Y
$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$
Pauli-Y
S
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&i\end{bmatrix}$
Controlled-Phase
$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$
Toffoli
Z
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
Controlled-Z
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$
Controlled-NOT

Fig. 1. Important quantum gates [6].

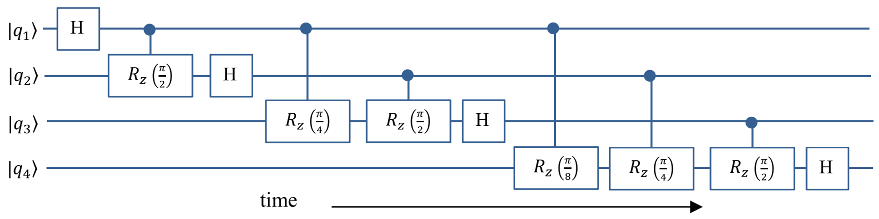

$|q_1\rangle$
$|q_2\rangle$
$|q_3\rangle$
$|q_4\rangle$
H
$R_z\left(\frac{\pi}{2}\right)$
H
$R_z\left(\frac{\pi}{4}\right)$
$R_z\left(\frac{\pi}{2}\right)$
H
$R_z\left(\frac{\pi}{8}\right)$
$R_z\left(\frac{\pi}{4}\right)$
$R_z\left(\frac{\pi}{2}\right)$
H
time

Fig. 2. Quantum circuit for the 4-qubit Quantum Fourier Transform [28].

reliable quantum circuit can be built using just imperfect gates, if they have failure probability less than a certain critical threshold.

2.4. *Quantum computing technologies*

DiVincenzo, in an important paper [33], gave the following five requirements that any physical system must fulfill to be a practical quantum computer:

I. Physically scalable with well-characterized qubits.
II. The capability to initialize the state of the qubits to a simple known state, such as $|00\ldots0\rangle$.
III. Long coherence times, much longer than the gate operation time.
IV. A universal set of quantum gates.
V. A qubit-specific measurement capability.
Moreover, quantum system should can send and store quantum information to build a quantum network. This "networkability" needs the following two more requirements to be fulfilled:
VI. The ability to convert stationary qubits to flying qubits and vice versa.
VII. The ability to faithfully transmit flying qubits between determined locations.

Several candidate technologies have been proposed for realization of a quantum computer to date [34]. The aforementioned criteria are used to evaluate quantum computing technologies. The leading candidate technologies are listed as follows:

I. Ion trap
II. Neutral atoms in optical lattice
III. Liquid-state/Solid-state NMR/ENDOR
IV. Cavity QED[4] with atoms
V. Linear optics
VI. Quantum dots (spin-based, charge-based)
VII. Josephon junctions (charge, flux, and current-biased qubits)
VIII. Electron on liquid helium surface

3. Ion trap technology

Of the above technologies that have been proposed for constructing a quantum computer, trapped ions are currently one of the most advanced [35]. Four out of the five core criteria specified in the previous section were realized with ion trap technology in the laboratory long before than the idea of quantum information processing was used by experimentalists: initialization [36] and read-out of the internal electronic states of trapped ions [37–39], extremely long coherence times [40] and laser cooled ion crystals with many ions [41–44] serving as a qubit register. Then Cirac and Zoller discovered that quantum information processing can be done by coupling the ions via a collective motional degree of freedom. Thus, a path to demonstrate a two-qubit operation was introduced. Moreover, they showed that as the number of qubits increases, the required resources to control trapped ions do not increase exponentially [7,45]. Not long after this proposal,

[4] Cavity quantum electrodynamics.

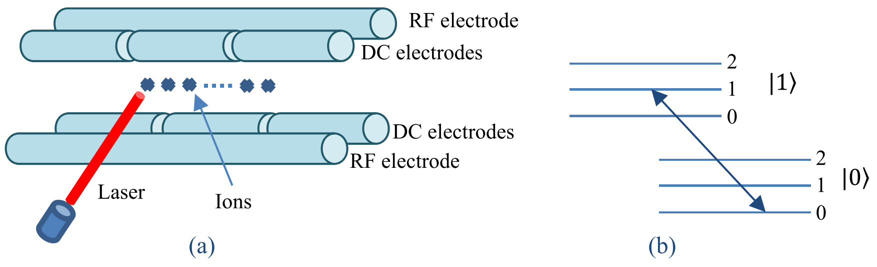


Fig. 3. (a) *N* ions in a linear trap consisted of four parallel rods. An RF voltage is applied to the continuous rods shown, while the segmented rods are held at a DC potential. This gives an oscillating field which is zero along a line between the rods. To provide axial confinement, a positive voltage is applied to the outer segments of the DC rods, while the inner segments are held at ground or a negative potential. (b) The combined motional and internal states of a trapped ion. In the figure, internal states are labeled |0⟩ and |1⟩, and motional states are labeled as 0, 1, 2. Combined motional and spin transitions can be driven. The first blue sideband is illustrated [71]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Wineland et al. in the NIST[5] [8] demonstrated its key idea a conditioned phase shift. They also implemented a few other two-qubit gates [10,46,47], entangled up to four ions [46], introduced a so-called decoherence-free subspace [48] and simulated a nonlinear beam-splitter [49].

Gulde et al. implemented the Deutsch-Jozsa algorithm with a single Ca+-ion [50], then Schmidt-Kaler et al. [9] implemented for the first time a set of universal gates on a two-ion string. Furthermore, Roos et al. [51] demonstrated the creation of various entangled states and the partial read-out of an entangled quantum register.

More breakthroughs in ion-trap technology were works on quantum teleportation [11,12], an error correction protocol [52], entanglement of six and eight ions [53,54], and entanglement purification [55]. Ion-photon entanglement has been used to entangle ions in separate traps [13,56–58].

3.1. Principles of ion-trap quantum computers

Wineland et al. [59] and Sasura et al. [60] have a good detailed survey of the fundamental issues in ion-trap quantum computing. Moreover, the advances in the control and manipulation of single ions are reviewed by Leibfried et al. [61]. Blatt and Wineland [62] surveyed the creation and uses of entangled ions. An ion-trap quantum processor satisfies DiVincenzo criteria as follows [63].

(1) **Physically scalable with well-characterized qubits:** Long-lived internal levels of the ions serve as the qubits. Strings of ions in a trap make the qubit register. This approach has some scalable issues; Therefore, the distribution of the ions among multiple traps was suggested [14,59].
(2) **The capability to initialize the state of the qubits:** Optical pumping is easily used to initialize the state of the qubits to a well-known state. The fidelity of this operation is about 0.99.
(3) **A coherence time much longer than the operational time:** Coherence times of more than 10s have been obtained with Raman-transitions between magnetic-field insensitive transitions [64]. The Oxford and Innsbruck groups [65,66] performed similar experiments with ^{43}Ca+. Moreover, a coherence time of more than 10 minutes is achieved using a microwave drive instead of a Raman laser system by Bollinger et al. [40].
(4) **A universal set of quantum gates:** One-qubit gates are realized by addressing individual ions with laser Rabi pulses. One-qubit gates can be represented as rotations of the state vector of a qubit on the Bloch sphere. The axis of rotation can be chosen by changing the phase of the exciting laser field or the phase difference of the two Raman beams. A one-qubit phase-gate can be demonstrated directly by an off-resonant laser via an AC-Stark shift [67].
Two-qubit gates are implemented by the collective vibrational motion of the trapped ions. In the original proposal, the quantum information of one ion is swapped to the common motional degree of freedom of the ion string [7]. Then an operation conditioned on the motional state can be performed on a second ion before the quantum information is swapped back from the motion to the first ion. The CNOT gate and controlled-NOT gate with multiple control qubits have been already implemented [9,68,69].
(5) **A qubit-specific measurement:** The qubit state can be measured by *fluorescence* detection. One of the qubit levels is energized to a higher lying auxiliary short-lived level via a closed transition while the other qubit level is not manipulated. As a consequence, ions in the state that is coupled to the transition will fluoresce, while the ions in the other qubit state appear dark.
(6) **The ability to convert stationary qubits to flying qubits and vice versa:** Storing ions in high-finesse cavities can sufficiently increase the coupling between the *ion* and photons in the *cavity* mode and *thus* allows *mapping* the *ions' internal state onto* a photonic state [70].
(7) **The ability to faithfully transmit flying qubits between determined locations:** Photons carrying quantum information can be transmitted through fibers and at the target location converted to the target stationary qubit via another high-finesse cavity.

3.2. Ion-trap basic architecture

3.2.1. Original proposal

As stated before, trapped ions as a processor for quantum information were introduced by Cirac and Zoller in 1995 [7]. The architecture is a single string of ions stored in a linear quadrupole ion trap (Fig. 3). One qubit is implemented by two electronic levels of an ion, well protected against environment disturbance. The necessary interactions for gate operations can be precisely performed by concentrated laser beams addressing the qubits individually. One-qubit gates and a controlled two-qubit gate are needed to have a universal set of gates to be able to demonstrate various quantum algorithms. One-qubit gates are operated by laser pulses exciting resonant transitions between the internal levels of the qubit in question. Two-qubit gates use one normal mode of collective vibration of the trapped ions as a data bus between qubits. This data bus can be exploited to couple the possibly non-adjacent qubits in the gate [7].

Although this proposal inspired a novel branch in trapped-ion research, it soon became clear that the scaling of the original architecture to hundred bits is very difficult. On the other hand, to implement the probably unavoidable error correction, ancillae

[5] National institute of standards and technology

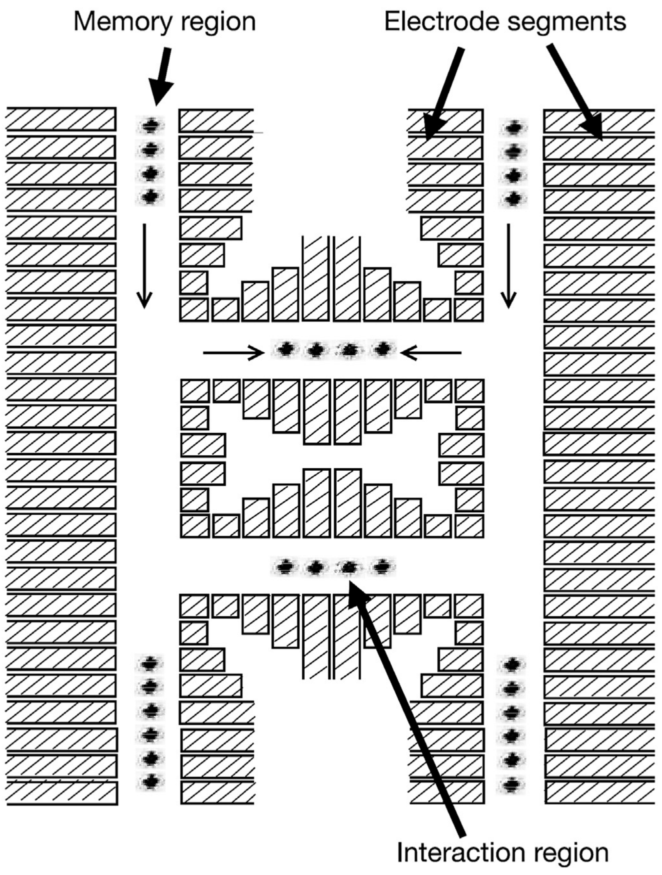


Fig. 4. Multiplexed trap architecture. Ions are stored in the memory region and moved to the interaction region for logic operations. Thin arrows show transport and confinement along the local trap axis. [14].

qubits should be also considered that may increase the number of required qubits to go for towards 10^5 ions [6]. Confining a linear string of thousands of ions in a single trap would result in high potentials on the endcap electrodes to prevent their Coulomb repulsion. The distance between adjacent ions should keep above the diffraction limit to address ions individually. This requirement leads to rather low axial potentials. Therefore, as the number of ions increases motional frequencies decrease. Low motional frequency and therefore, a slow data bus restricts gate speeds for the computation. At last, from a practical point of view, $3\times N$ normal modes for N ions in addition to their sum and difference frequencies result in an increasingly crowded excitation spectrum where individual components are difficult to recognize, and off-resonant coupling to spectator transitions is hard to prevent.

3.2.2. *Multiplexed trap proposal*

In 1998, a multiplexed trap architecture [14,59,72] was proposed at NIST that may lighten the aforementioned issues and is modular, thus scaling to higher qubit numbers appears to be achievable. This architecture is made of an array of many independently controllable sub-traps (Fig. 4).

Memory regions store qubits that do not participate in a given step of the computation. To perform a gate on specific qubits, they are split from the memory regions and moved into one of the computing regions. As the motion is just used to couple ions during the gate operation in the computing region, moving ions do not result in decoherence in the computational Hilbert space covered by the qubits. The movement may raise the motional energy of the ion qubits. However, ions can be sympathetically recooled close to the ground state by another refrigerator ion

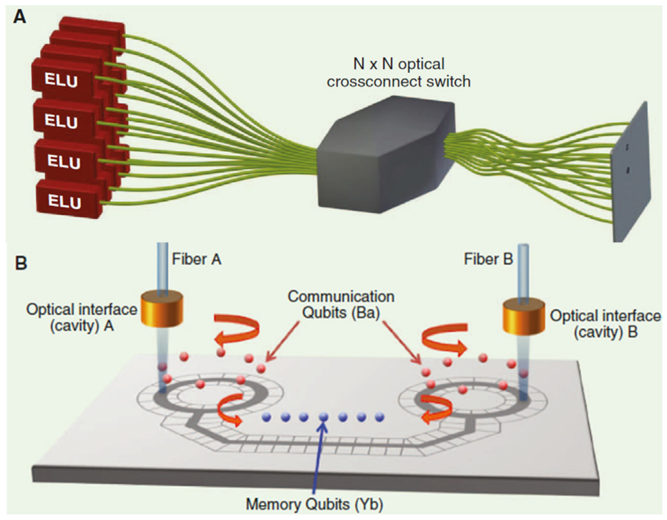

A
ELU
ELU
ELU
ELU
N x N optical crossconnect switch
B
Fiber A
Fiber B
Optical interface (cavity) A
Communication Qubits (Ba)
Optical interface (cavity) B
Memory Qubits (Yb)

Fig. 5. MUSIQC architecture (A) Several ELUs are connected through a photonic network by using an optical cross-connect switch, inline fiber beamsplitters, and a photon-counting imager. (B) Trapped ion quantum repeater node made up of communication qubit ions (such as Ba+) and memory qubit ions (such as Yb+), with two optical interfaces per node. Multiple communication qubits are used per optical interface to inject photons into the optical channel, while the results for successful entanglement generation at the detectors are reported back to this node. Only qubits corresponding to successful events will be transported to the memory qubit region for use in the quantum repeater protocol [76].

before the next gate is applied [73,74]. This cooling process conquers the motional heating; Therefore, the time available for processing would be restricted only by the decoherence time of the internal qubit states. Since the lifespan of the hyperfine ground states is so long, the memory decoherence is caused mainly because of phase errors induced by external factors. The order of decoherence time in a precisely controlled environment may be many days. Experimentalists have demonstrated long-term memories that lower limits of their decoherence times are several minutes [40]. Another benefit of the multiplexed trap architecture is the capability to measure a qubit without disturbing adjacent ion qubits through the spread photons. These measurements can be done in districts that are adequately spatially separated from the remaining qubits.

All steps mentioned above can be performed in a highly parallel form, an important requirement for efficient error correction. Scaling to numerous ions is practically challenging, however, appears to be conceivable without fundamental limitations.

The multiplexed trap architecture is made of the following basic building blocks:

- **Trap arrays:** Arrays contain several independent traps and crossings and/or "T"-junctions and have to be built in a precise and repeatable way. The methods have to be scalable to large arrays that can confine and manage thousands of ions.
- **Ion Movement:** Ion movement in a trap array must be dependable and repeatable. It should not add too much extra energy to ions. The typical delays of these movements should not considerably restrict the speed of the quantum algorithm.
- **Separation and recombination of ions:** To perform quantum logic gates, specific ions must be moved from the memory regions dependably and combined with another ion-qubit and the refrigerator-ion in the computing unit. The delays of these operations should also not substantially restrict the speed of the quantum algorithm.
- **Sympathetic recooling:** Extra kinetic energy of the ions, gained by their movement in the array, by external heating mechanisms, or by the recoil suffered in ancilla-measurement steps, can

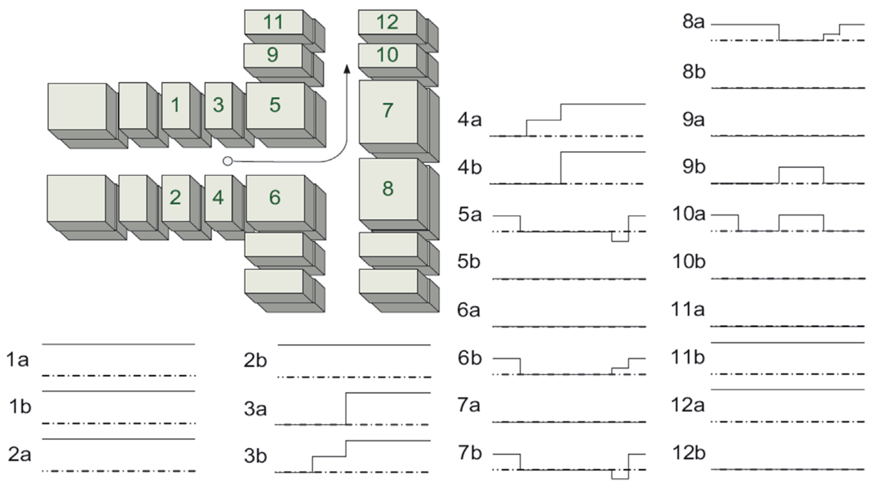


Fig. 6. Electrode structure and pulse sequences needed to transport a physical ion-qubit from between electrodes 3 and 4 to between 9 and 10. The dashed line in each waveform is ground; *a* and *b* refer to top and bottom electrodes at each location [78].

be removed by sympathetic laser cooling with a second ion species. Recooling must leave the ions adequately near the motional ground state and should not disturb the qubit state.

- **Robust one-qubit and two-qubit gates:** Gate fidelities for bigger algorithms should be on the order of 0.9999. All single-qubit and two-qubit gate demonstration techniques should reach this fidelity with technically realistic advances and be compatible with the other features of the architecture, for example, the presence of a refrigerator ion during gate operation.

3.2.3. *MUSIQC architecture with atomic memory and photonic interconnects*

The MUSIQC[6] architecture [75] includes a series of identical ion traps with each holding a linear chain of equally-spaced ions in a single anharmonic potential well. Local entangling gates within each ion chain are implemented by coupling their internal qubit states through shared motional modes. These individual ion chains, or 'Elementary Logic Units' (ELU), are then coupled to each other using photonic links for remote entanglement. Fig. 5 shows the MUSIQC architecture.

3.3. *Transport of ion-qubits*

To perform a two-qubit quantum gate both ion-qubits must be physically neighbor to each other and should move from one place to another. Movement can be divided into two classes: short-distance and long-distance. In short-distance communication, ion-qubits move ballistically while in long-distance communication, ion transport happens via an interconnection network based on teleportation or via a photonic network. In the rest of this section, these three communication approaches are discussed in more details.

3.3.1. *Ballistic transport*

An ion trap comprises of an arrangement of electrodes which an ion is trapped in the space between them. By putting some ion traps in sequence and applying specific pulse sequences to the electrodes, the ion-qubit can ballistically be moved along the channel, thus behaving like such a simple wire [77]. Fig. 6 demonstrates a perspective of some ion traps [78]. It also shows control pulses needed to move an ion across the traps. In this figure, the gray rectangles are electrodes. The white area between electrodes is the transport channel. The ballistic transport is the most fundamental transport operation in an ion-trap computer. As illustrated in Fig. 6, although this operation seems to be simple, it is rather complicated. Some MEMS[7] fabrication methods have been proposed that could be scaled to build many integrated qubits [79].

3.3.2. *Teleportation*

Ballistic transport of an ion-qubit decreases the fidelity of its state, called decoherence. 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 will need different forms of communication for longer distances. Teleportation is one solution.

Fig. 7 shows an abstract representation of teleportation [80]. The goal is transmitting the state of qubit *Q* from the source to the distant destination without physically transporting the data qubit. At the first step, an EPR[8] pair of qubits ($E1$ and $E2$) is generated and sent to two endpoints. At the second step, some local operations are performed on qubits $E1$ and Q that lead to two classical bits. Next, the two classical bits are transmitted to the target location. Finally, the local operations on qubit $E2$ are performed based on the values of two classical bits to transform the state of qubit $E2$ into the original state of qubit Q. There are only two non-local operations in teleportation: the transport of an EPR pair to source and target locations, and the transmission of digital bits from source to target. In fact, a quantum channel generates the EPR qubits and delivers them to the source and target. This EPR pair must have high fidelity to make reliable communication. Since EPR pair can be distributed in advance, teleportation time can approach the latency of classical communication. It should be noted that the channel setup time increases with distance as well as fidelity.

3.3.3. *Photonic network*

Two ion-qubits from a pair of ELUs (or EELUs) shown in Fig. 5 can be entangled by each emitting photon that interferes with each other. One or more atomic qubits within each of the ELU registers are coupled to photonic quantum channels, and through a reconfigurable optical cross-connect switch, fiber beamsplitters, and position sensitive imager (right), qubits between different registers can be entangled and states of ion-qubits are transmitted via photons between ELUs.

[6] Modular universal scalable ion-trap quantum computer

[7] Micro-electromechanical systems.

[8] Einstein-Podolsky-Rosen.

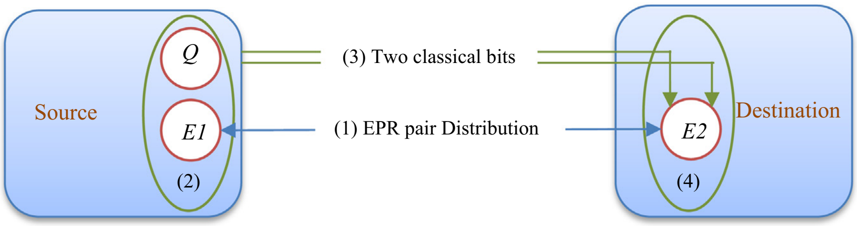


Fig. 7. Steps for teleporting qubit Q from source to destination (1) a high-fidelity EPR pair generation and distribution ($E1/E2$), (2) local operations at the source, (3) transmission of classical bits, and (4) correction operations to transform the state of $E2$ into the state of Q.

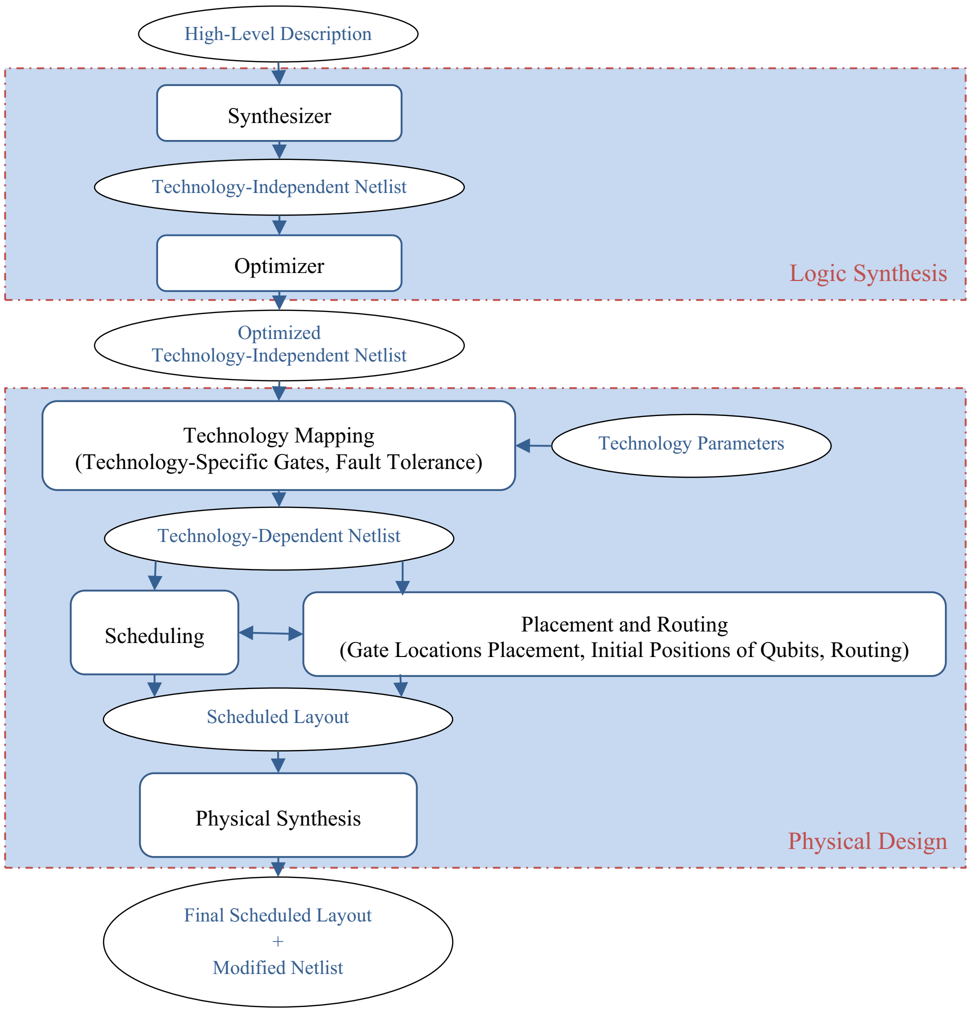

High-Level Description
Synthesizer
Technology-Independent Netlist
Optimizer
Logic Synthesis
Optimized Technology-Independent Netlist
Technology Mapping (Technology-Specific Gates, Fault Tolerance)
Technology Parameters
Technology-Dependent Netlist
Scheduling
Placement and Routing (Gate Locations Placement, Initial Positions of Qubits, Routing)
Scheduled Layout
Physical Synthesis
Physical Design
Final Scheduled Layout + Modified Netlist

Fig. 8. Quantum circuit design flow.

4. Quantum circuit design

Logic synthesis and physical design are two main processes of the design flow. The synthesis process converts a description into an optimized gate-level and technology-independent netlist. The physical design process maps the resulted netlist onto the quantum circuit fabric.

Fig. 8 represents an overview of the quantum circuit design flow. Rectangles and ovals represent tools and intermediate formats respectively. The flow starts with a high-level description of the quantum algorithm. Some formats have been presented to describe quantum circuits such as schematic entry and mathematical formulae. Different quantum formats were overviewed and discussed in survey papers [81–83] and research publications [84–86]. At present, the quantum assembly language (QASM) is used by most CAD[9] tools. Balensiefer et al. [87,88] proposed this language to describe quantum circuits for the first time.

The logic synthesizer takes the high-level description and creates a technology-independent gate-level netlist. Then the optimizer applies some optimization algorithm to the resultant netlist with the aim of improving one or more circuit metrics. As the abstraction level decreases from top to down, the design is provided with more and more low-level details. However, higher abstraction level information such as encoded qubit groupings and distinguishing between ancillae and data qubits should be also saved to allow low-level CAD programs to make more clever choices in scheduling, placing and routing of qubits.

As the optimized technology-independent netlist is generated

[9] Computer-aided design

by the synthesis process, the physical design process begins. The physical design process is a technology-dependent process. Therefore, the technology should be known before the process is started. As stated and discussed in the previous sections, ion trap technology is selected in this paper. Technology parameters specify the complete set of physical characteristics of the technology such as layout building blocks as well as design rules for connecting them.

The primary task of the technology mapping program is to convert a technology-independent netlist into target technology blocks to generate the technology-dependent netlist. The complexity of technology mapping phase depends on the complexity of the technology blocks. It may translate the gates to technology-specific gates and make the circuit fault tolerant by encoding the qubits and automatically adding the ancilla and sub-circuits necessary for error detection and correction. Moreover, if the quantum circuit is to be implemented on the architecture supporting teleportation-based interconnections [89], they should be inserted into the netlist in this phase using the higher level qubit grouping information. At this point, some technology-specific optimizations may be also applied to the circuit.

Once the technology mapping process finished, the placement and routing process and scheduling process begin. The scheduling process determines the execution order of gates. The placement and routing process lays out the netlist. It places gate locations and adds routing channels between them for communication. Moreover, it determines the positions of the qubits at the beginning of the execution. The placement and routing process and the scheduling process are interdependent and the results of each one can improve the results of the other. Therefore, these processes are often iterated in a loop to refine and improve the result. After scheduled layout is generated, some optimization approaches can be applied to the resulted layout.

At this point, that the scheduling and layout information is ready, physical synthesis techniques, introduced first in [90], can be applied to the design. These methods modify the gate-level netlist with considering the layout and scheduling information to optimize the objectives or meet the design constraints.

4.1. Logic synthesis

The logic synthesis in the quantum CAD flow is a process by which an abstract form of the desired circuit behavior is turned into a netlist of logic gates and connections between them. Several studies are focused on the synthesis of quantum circuits. Interested readers are referred to the research papers [91–95].

4.2. Physical design

The main focus of this paper is on physical design of quantum circuits in ion trap technology. This section is organized as follows. In Section 4.2.1, the metrics proposed in quantum physical design research papers are mentioned. Section 4.2.2 contains the quantum datapath organizations presented in the research for ion trap technology. The main part of physical design flow starts from Section 4.2.3 with technology mapping. Section 4.2.4 includes the studies focused on placement and routing process. The scheduling algorithms are described in Section 4.2.5. Section 4.2.6 includes physical synthesis concept and techniques proposed for it. The works that have considered the fault-tolerant design are presented in Section 4.2.7.

4.2.1. Design metrics

Some metrics are needed to evaluate the quality of algorithms proposed for automation of physical design. The proposed metrics in quantum physical design literature and their descriptions are mentioned here.

4.2.1.1. Area. The quantum layout size directly affects ease and cost of fabrication. Moreover, smaller layouts consume less power and have less routing delay because of shorter channels.

4.2.1.2. Latency. Applying more gates in less time is often the most important factor in the quantum circuit design and optimizations. The latency of a quantum circuit is the total time that it takes for a circuit to be run on a given layout.

4.2.1.3. Reliability. Quantum circuits are definitely more error-prone than static CMOS circuits; this makes reliability a major measure in the quantum circuit design.

4.2.1.4. Success probability. The success probability of a quantum circuit is the probability that the output will be error-free after evaluation. It is estimated with hybrid simulation. Finally, the overall success probability of the circuit can be calculated as follows:

$$p_{\text{success}} = \frac{\text{number of successes}}{\text{number of failures} + \text{number of successes}}$$

4.2.1.5. Area-Delay-to-Correct-Result (ADCR). Whitney et al. [96] proposed a hybrid measure that is the probabilistic version of the Area-Delay product, called Area-Delay-to-Correct-Result (ADCR). This measure is calculated as follows:

$$ADCR = \text{Area} \times E(\text{Latency}_{Total})$$
$$= \text{Area} \times \sum_{n=1}^{\infty} n \times \text{Latency}_{Single} \times P_{Success}(1 - P_{Success})^{n-1} =$$
$$ADCR = \text{Area} \times \frac{\text{Latency}_{Single}}{P_{Success}}$$

The expected time to get a correct answer is:

$$E(\text{Delay}) = \text{Delay}_{Singlerun} \times E(\text{runs to correct result})$$
$$= \text{Delay}_{Singlerun} \times \sum_{n=1}^{\infty} \frac{n}{P_{Success}(1 - P_{Success})^{n-1}} = Delay_{Singlerun} \times \frac{1}{P_{Success}}$$

4.2.2. Microarchitecture design

Fig. 9 represents four main microarchitectures proposed for quantum computing in ion trap technology. They are QLA[10] [97–99], CQLA[11] [100], Qalypso [96], and Requp[12] [101], and can be thought as a 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 [89,97]. 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. Data in memory and processing units is affected by different error types: idle errors in memory unit versus interaction errors in processing unit. To consider this issue, different error-correcting codes are used in memory units and compute units in the CQLA. In addition, [96] introduced LQLA and CQLA+, which

[10] Quantum logic array.
[11] Compressed quantum logic array.
[12] Reconfigurable quantum processor.

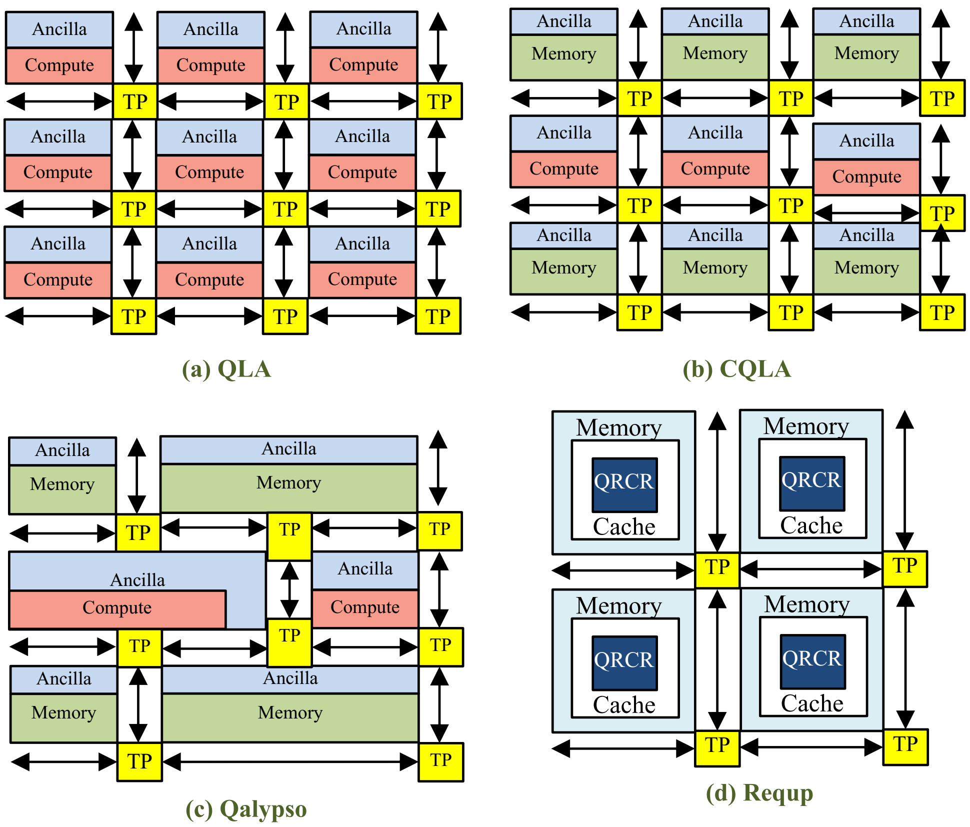


Fig. 9. Quantum microarchitectures: (a) Quantum Logic Array (QLA): An FPGA-like array of two-qubit operations (compute regions), where each region has dedicated ancilla resources. (b) Compressed QLA (CQLA): QLA compute regions surrounded by memory regions. (c) Qalypso: Variable sized compute and memory regions with shared ancilla resources for each tile; teleportation network can have variable bandwidth links. (d) Reconfigurable quantum processor architecture (Requp): reconfigurable compute region with different number of ancilla qubits.

Table 1
Quantum microarchitectures and their specifications.

Microarchitecture	Description
QLA	• Quantum Logic Array [97] • Compute regions only • no specialization
CQLA	• Compressed QLA [100] • compute and memory regions specialization • original ancilla generator
Qalypso	• Variable sized compute and memory regions [103] • Variable resources in ancilla generators and teleport network • “Pipelined” ancilla factory optimized from design in [110]
iQalypso	• improved Qalypso [104] • Qalypso with a new mapper
Requp	• Reconfigurable quantum processor architecture [101] • reconfigurable compute regions • quantum operations with different number of ancilla qubits

are variants of QLA and CQLA with improved ancilla generators from [102] and [103] respectively. Qalypso [103] 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 using in processing units. Ancilla generators and compute regions can be sized based on quantum circuit requirements. Wang and Khainovski proposed iQalypso [104] that improves the mapping phase of quantum design flow for Qalypso by reducing the number of expensive long-distance communication. Dousti et al. proposed Requp [101], 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. The motivation behind this architecture is that the compute region with pre-allocated resources and a fixed number of ancilla qubits for all of instructions wastes the resources because of overestimating the ancilla qubits needed. In other words, the ancilla qubits are shared among the operations which are being executed, with considering ancilla qubit requirements of the operations. In all four microarchitectures, a teleport router is placed adjacent to each processing and memory unit. Qubits are transported ballistically within regions and teleported between regions. Table 1 summarizes above organizations and their specifications.

One method to alleviate the communication constraints is to consider distributed quantum architecture [75,105–109]. It has been shown that entanglement between remote atomic qubits (such as trapped ions) can be achieved through photon interactions.

4.2.3. *Technology mapping*

In most cases, the gate-level netlist resulted from the synthesis process is not directly implementable on the target technology. To physically realize a quantum circuit, all operations in the netlist should be translated into operations that are available in the given technology. In some research, technology mapping is applied before post-synthesis optimization and in some others, it is done as the first step of the physical design process. The goal of technology mapping step in this paper is the process done during the physical design. Most of the techniques proposed for technology mapping in the synthesis process use a specific library such as the NCV (NOT, CNOT, C, and V+) library [111] and map a netlist into gates available in that library. In the most cases, some gates of these libraries are not directly realizable in any technology. Therefore, a more technology mapping step is needed at the beginning of the physical design process. Technology mapping techniques done during the synthesis process have been reviewed and discussed by Sasanian in [112].

Although some of studies done on physical design included the technology mapping in their flows, the input of the majority of them is a technology-dependent netlist. To the best of our knowledge, only two groups explicitly considered the technology mapping in the physical design. Balensiefer et al. [87,88] transforms a quantum algorithm described in a QCL[13] [113] language into a technology-dependent netlist by a two-step process. In the first step, a source compiler compiles the input description into a technology-dependent netlist described in QASM[14] language. Then, an error correction compiler converts the resulted netlist into an equivalent, fault tolerant version. The 7-qubit Steane code [114] and the recursive construction process [115] are used in these papers to build a fault-tolerant netlist. Svore et al.'s tool [116,117] maps a technology-independent QASM netlist to a technology-dependent fault-tolerant netlist during the third phase of its data flow.

4.2.4. Placement and routing

The gates in the resultant netlist from the technology mapping process are mapped into physical macroblocks. The goals of this placement stage are to assign a physical element in the target technology to each gate element and to place gates that are directly connected in the netlist very near to each other. After this, wires are routed between the physical gates. Since both gates and wires occupy physical space on the layout, the placement and routing processes are iterative and converge on a design where everything fits onto the physical substrate.

In ion trap technology, some details such as which type of ion is used, specific electrode sizing and geometry and exact voltage levels necessary for trapping and movement are varied and ambiguous. Therefore, it seems to be necessary to use a structure that is independent of these details. Whitney et al. [118] 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 removed. All of these details are condensed within the macroblocks. Basic block abstraction and the basic macroblocks of an ion trap layout suggested by Whitney et al. are depicted in Fig. 10. 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. Many studies [27,90,96,103,118–126] in the field of physical design of quantum circuits used these macroblocks to construct quantum circuit layouts.

The works considering both placement and routing can be divided into the following categories:

- **Manual approach.** Kreger-Stickles et al. [102] investigated several manually generated layouts for the same code in an effort to determine the best one in terms of latency, area, and some measurement of errors. This approach is not scalable for large quantum circuits.

[13] Quantum computation language.

[14] Quantum assembly language.

- **Greedy approach.** Whitney et al. [118] proposed a simple greedy approach. At the beginning of the algorithm, the number of gate locations and the number of qubits are equal and there is no channel between the gate locations. The algorithm uses scheduling information to displace gate locations and insert or reshape channels. The scheduling of instructions, moving and connecting of gate locations are executed in a loop until the qubits can communicate adequately to realize the given circuit. The main concentration of this work is the delay minimization of the critical path, not minimizing intersections of channels. Moreover, despite the fact that critical gates are mapped and placed close to each other, the routing algorithm has a tendency to push these gate locations away as more channels cross the center of the circuit. Therefore, the greedy algorithm rapidly begins to be non-optimal as circuit size increases but is appropriate for small size, low-congestion quantum circuits. Fig. 11 illustrates the steps followed by the algorithm to generate a layout from a given netlist.

- **Grid-based approach.** In all grid-based placement approaches, first, a basic cell is designed and then tiled into a larger physical layout to build the final one. The authors of [97,98,100,127] manually design a single basic tile and then use that to generate an appropriately sized layout for any given circuit. In [116,117,128,129], the authors automatically generated an H-Tree-based layout built from a single tile. Similarly, [87,88] used a tile pattern like that proposed in [116,117] and developed a tool to explore the performance of a quantum circuit when the number of channels and gate locations within the tile is varied. They used a modified version of the PathFinder algorithm [130] to find a set of efficient paths from source to destination in the layout. The main modification made in the PathFinder algorithm is that they find 5–10 paths on the first movement between a source-destination pair. This computation is just performed one time, and following movements simply use the best path from the stored set. Whitney et al. [118] combined above techniques to develop a toolset that automatically generates a grid-based layout for a given circuit. In this method, the best tile pattern used to generate the grid-based layout is dependent on the input quantum circuit. Grid-based physical layouts have the high amount of channel congestion because of limited bandwidth of channels. Moreover, some of gate locations and channels in many of grids are not used by the scheduler that this wastes the physical space. For example, Fig. 12 shows how a 2×2 sized cell can be tiled to create the layout used in [127].

- **Other heuristic approaches.** Whitney et al. [118] proposed a dataflow-graph-based algorithm to placement and routing process. In this approach, the initial places of qubits are implicitly determined. The authors followed a gate-array-style design that places gate locations in columns according to the circuit dataflow graph. The algorithm leaves some space between each pair of columns for routing channels. To save wasted space due to uneven column sizes, a folding operation is applied. After sorting and local routing, the move with the largest delay on the critical path is computed and the nodes located on the two sides of this move are merged into one node group. The layout is then updated to find the next pair to merge. After a few repetitions, the latency increases with each further merge due to congestion at some heavily merged groups. At this point, the process is stopped. The layouts resulted from dataflow-graph-based heuristic have better delay and pipelinability than its predecessors at the cost of more area. Since this approach has been used by many other research groups, an example layout generated by this approach is illustrated in

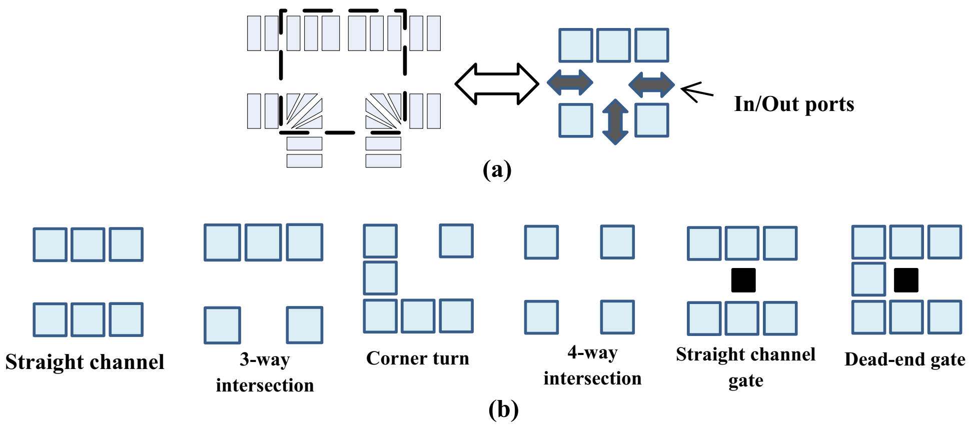


Fig. 10. (a) Basic block abstraction [77]. (b) A library of macroblocks [118].

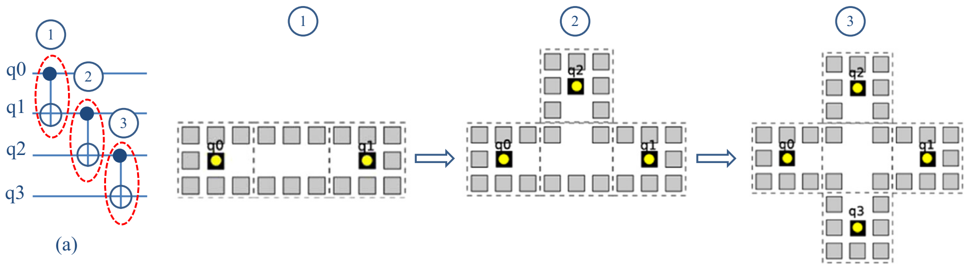


Fig. 11. The greedy approach in generating a layout from a sample netlist [118].

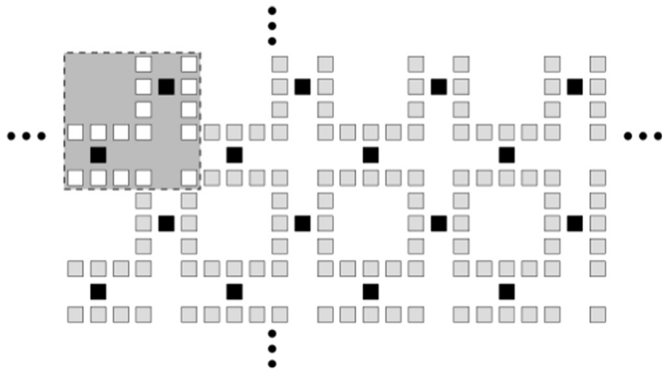


Fig. 12. QPOS grid structure constructed by tiling the highlighted 2×2 macroblock cell.

Fig. 13. Dousti et al. [131] presented a mapper, called QSPR[15], that during an iterative process traverses a quantum instruction dependency graph forward and backward to place qubits and then select the best one. Yazdani et al. [126] proposed a layout generation approach in which a scheduled instruction graph is laid out by first applying the Giotto graph drawing algorithm [132] to the graph and then substituting the nodes and interconnections by the macroblockes [118]. A hierarchical layout generation algorithm was proposed in [122]. This algorithm uses a min-cut placement-aware approach [133] for partitioning and the terminal propagation concept [133] for passing information between levels. When the number of nodes in each partition is less than eight, the partitioning is stopped and the nodes are placed with a greedy algorithm. Goudarzi et al. [134] 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 [135] to solve it. Dousti et al. [101] modeled the placement problem as a 0-1 quadratic program (0-1 QP) and solved it by Gurobi optimization tool [136]. In [137,138], a partitioning-based placement approach was followed. Qubits are distributed among elementary logic units by a simple greedy algorithm that prioritizes the assignment of qubits to the ELUs in the ascending order of their total edge weights (total number of two-qubit gates involving a qubit). In each iteration, a qubit is selected from the priority list along with the set of its interacting qubits. An ELU which can accommodate most of these qubits is selected and assigned to the set. Then, the placer arranges the qubits within an ELU by maximizing locality among frequently interacting qubits. It finds an approximate solution to the Optimal arrangement Problem, by using a graph-theory-based algorithm proposed in [139].

[15] Quantum scheduler, placer, and router.

4.2.4.1. Quantum interconnection network. Some research is concentrated only on interconnection topologies. Isailovic et al. [89] explored the entanglement swapping strategy [140] (Fig. 14) for interconnection networks. In this scheme, EPR qubits are created in the middle EPR generator and then are transported via teleportation until they reach the final teleporter nodes. After they reached to the final teleporters, they are ballistically moved to corrector nodes and then purifier nodes [141]. The authors assumed that communication structure is a 2-D mesh of teleporter nodes. A sample structure of a 5×3mesh grid is shown in Fig. 15. They customized dimension-ordered mesh routing idea [142] to deliver EPR pairs. In the other work, Metodi et al. [143] investigated the teleportation-based interconnect network in the QLA architecture discussed various tradeoffs in the communication network. Unlike Isailovic et al.'s approach [89] that in that EPR pairs were generated directly between source and destination without purifying at any intermediate scope, in this paper a purely linear strategy was followed.

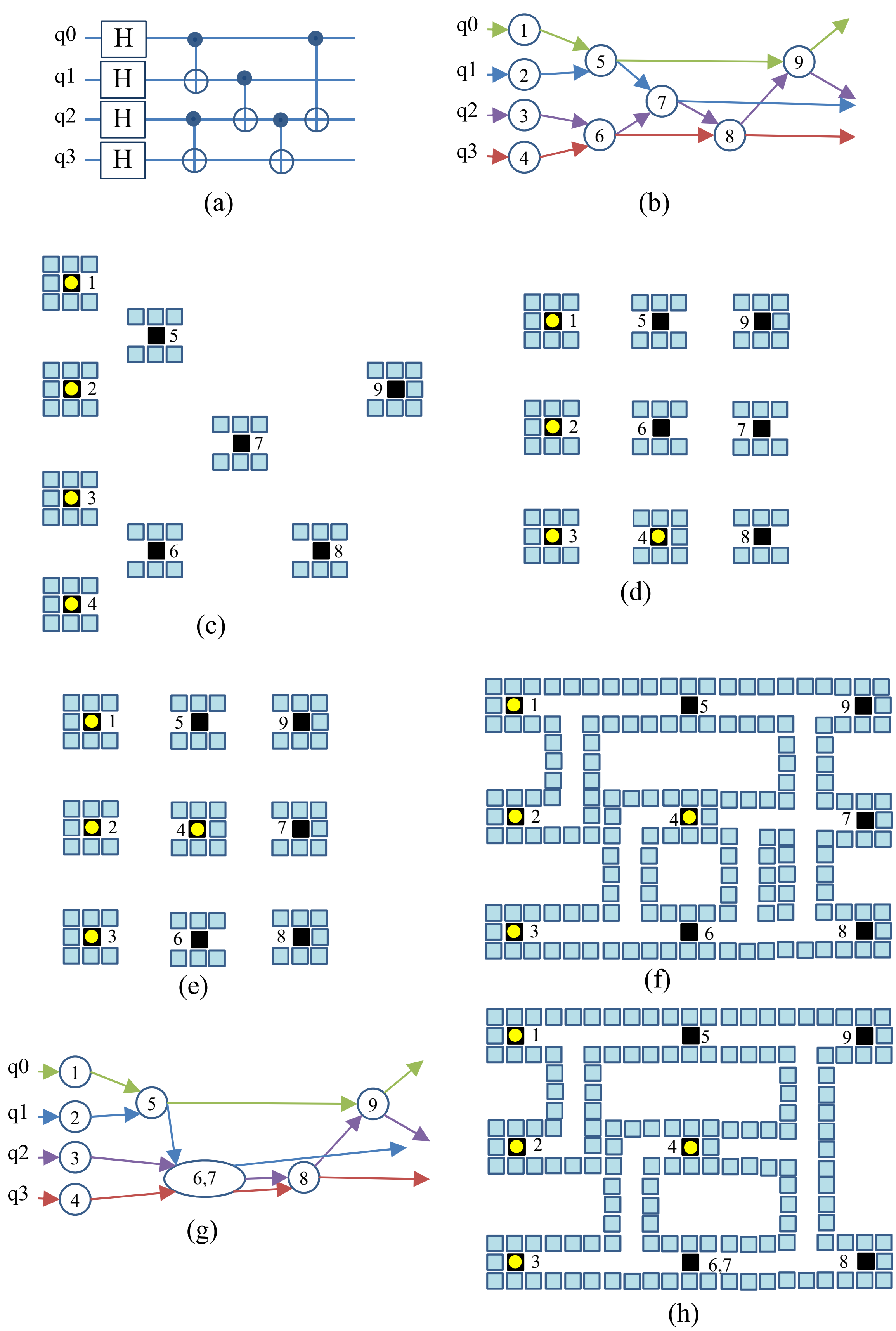


Fig. 13. The steps of layout generation by the dataflow-based approach. (a) A quantum circuit. (b) The dataflow graph equivalent to the quantum circuit in (a). (c) Gate locations are placed in dataflow order. (d) Gate locations are folded (e) Gate locations are sorted. (f) Channels are routed to reflect dataflow edges. g,h) Critical node pair is merged and channels are rerouted.

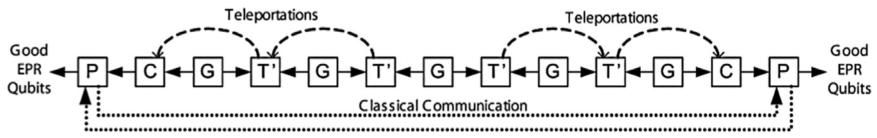


Fig. 14. Chained Teleportation Distribution Methodology. Generator (G), local Teleporter (T′), Corrector (C), and Purifier (P) nodes [89].

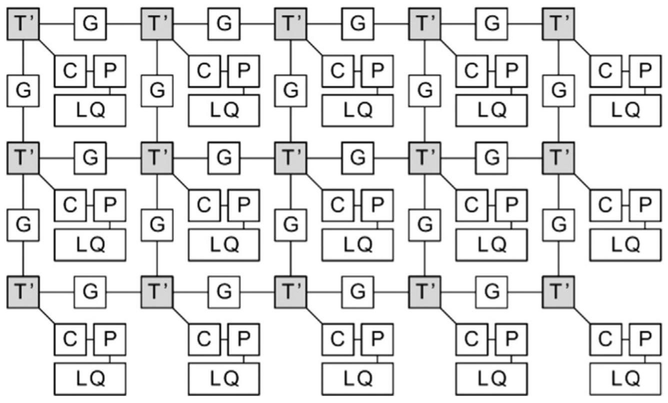


Fig. 15. Sample Layout of a 5×3 mesh grid containing Logical Qubits (LQ) and generator (G), T′ (local teleporter), corrector (C), and purifier (P) nodes (not to scale) [89].

4.2.4.2. Post-layout optimization. The results obtained by heuristic placement and routing methods are often sub-optimal but can be improved by local optimization. Mohammadzadeh et al. [121] presented Gate Location Changing (GLC) technique that benefits from physical layout and scheduling information to minimize the critical path delay by changing the locations assigned to gates on the critical path. A mixed integer linear programming model for assigning of gates to gate locations in a layout generated by Whitney et al.'s dataflow graph algorithm [118] was introduced in [124].

4.2.5. Scheduling

As stated before, the scheduler takes three inputs (the netlist to be scheduled, a description of the layout to schedule the net list on, and a description of the technology parameters and constraints) and generates a schedule of operations that should be performed on the physical layout. One of the main challenges of quantum circuit design is scheduling of a quantum circuit onto a physical layout while considering the cost of routing, the classical resources, and the maximum exploitable parallelism.

Cross et al. [129] used a manually-optimized scheduler to schedule operations on H-tree-based layout. The scheduler proposed in [87,88] employs a variant of list-scheduling method [144]. First, the netlist is parsed and a graph representation is generated. Then, a latest-possible greedy approach is applied to the graph. At any given time, the scheduler keeps a list of operations that can be scheduled and tries to allocate physical resources for the required number of time steps. For operations, this implies just waiting onto the trap for that time. For movement, it implies selecting the path with no conflicts from the precomputed path set for the time needed. Operations that cannot be scheduled because of resource conflicts are simply postponed, and another scheduling attempt is made at the next suitable time step. Metodi et al. [97–99] used a heuristic greedy scheduler. This scheduler takes all available channel bandwidth whenever it can. If it cannot find the required paths, it returns backward and retries with a different set of start and end points. To perform a two-qubit operation between qubits *A* and *B*, first *A* is teleported to *B*'s location, then the gate is applied, and *A* is teleported back. An optimization on this scheduling approach is that qubit *A* only is moved back if it is necessary. As a result, the qubits may be displaced from one location to another during the execution of the quantum circuit. Although this makes the scheduling process more complicated, it decreases the amount of communication. In this scheduler, the channel bandwidth is assumed to be two. Svore et al. [116,117] expanded QASM instructions to include movements and developed a scheduler that uses implicitly specified paths to generate minimal distances.

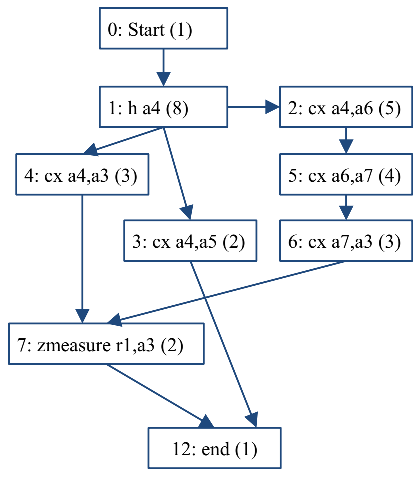
QPOS[16] [127] schedules a quantum circuit on a fixed grid-based fabric. It is an iterative scheduler that customized traditional classical scheduling heuristics [145–147] through precise computation at the both, the circuit level and the layout level. At the circuit level, gate priorities are computed based on the number of gates that depend on each gate. After the source qubits and the destination qubits have been unambiguously determined, the priorities are used to select the desired paths. If the gates have the same priority, least path interference and shortest path are used to prioritize the paths. This scheduler attempts to maximize the parallelism of the movement operations. This scheduler unlike the prior works [87,88] takes into account moving both source and destination qubits in two-qubit gates to improve the final scheduling results. Since this scheduling algorithm made a great improvement on earlier works and some next scheduling approaches were a modified version of it, an example and the pseudo code of the algorithm are depicted in Fig. 16. Whitney et al. [118,119] modified the QPOS [127] to utilize gate times rather than the size of the dependent subtree in computing of the critical path. Their scheduling technique is a greedy one. It maintains the set of operations whose all predecessors have already executed, and thus are ready to be executed. The scheduler tries to schedule the highest priority ready operation (with considering the critical path) first, and thus it has more chance to get access to the required resources, including gate locations and channels/intersections. When all possible operations have been scheduled, time progressed until one or more resources is freed and more operations may be scheduled. The scheduling and stalling cycle proceeds until all operations are executed or deadlock happens. The deadlock is detected and the highest priority unscheduled operation at that time is reported. This scheduler uses both gate delays and movement delay in the prioritization of the operations because with this modification each qubit's critical path can be approximated better. The scheduler used in [131] utilizes a linear summation of the size of the dependent subtree and the length of the longest path delay from that operation to the end node of the instruction dependency graph to prioritize the unscheduled ready operations. The authors of [126] addressed the scheduling problem using Integer Linear Programming (ILP) considering the classical resources (qubits) and operation dependencies. Moghadam et al. [122] used a customized version of the ASAP[17] algorithm that is applied on a timed-DFG, including both time and data dependency information. Goudarzi et al. [134] decomposed the scheduling problem into two steps to solve it; In the first step, the list scheduling algorithm [144] is utilized to totally determine the order of the operations with the same operation parent; In the second step, an improved version of the force-directed scheduling

[16] Quantum physical operation scheduler.

[17] As soon as possible.

```
1. h a4;
2. cx a4,a6;
3. cx a4,a5;
4. cx a4,a3;
5. cx a6,a7;
6. cx a7,a3;
7. zmeasure r1,a3
```

(a)



(b)

```
While (1)
{
    1.  Get all available operations at current cycle;
    2.  If (leaf_node in available gates): break;
    3.  Execute any available classical gates;
    4.  Get the available parallelism;
    5.  For (each available single-qubit gate) {
    6.      Execute it and update the available parallelism;
    7.  }
    8.  If (parallelism=0): continue;
    9.  Else {
    10.     Disambiguate source-destination paths;
    11.     Choose N paths for execution such that
    12.         N= # of distinct sources;
    13.     Schedule chosen paths;
    14.     Eliminate redundant stall cycles.
    15.  }
    16.  Update execution cycle;
}
```

(c)

Fig. 16. (a) A sample netlist. (b) Direct output DAG of QPOS given the sequence in (a). The dependence number of each instruction is marked in parenthesis next to the instructions. (c) QPOS scheduling routine [127].

algorithm [148] is applied to the modified quantum instruction dependency graph to minimize a cost function which is a function of the number of scheduling levels and the number of concurrent instructions per scheduling level. Dousti et al. [101] used a modified version of the list scheduling method to solve the scheduling problem. The authors of [125] proposed a mixed integer nonlinear programming (MINLP) model, called MSA_QSPP, for scheduling and placement, and developed a metaheuristic solution method for it. Ahsan et al. [137,138] used a greedy ASAP algorithm that dispatches gates for execution as soon as resources become available.

Table 2
Main algorithms proposed for automation of physical design.

Approach	Main Focus on	Method		Metric
Kreger-Stickles et al. [102]	Layout Generation	Manual		Latency, Area, Error
Whitney et al. [118]	Layout Generation	Greedy		Latency
Metodi et al. [97,98,100]	Design flow	Layout	Grid-based	Latency
		Scheduling	Greedy	
Metodi et al. [127]	Scheduling	iterative scheduler with customized traditional classical scheduling heuristics		Latency
Svore et. al. [116,117]	Design flow	Layout	Grid-based and H-tree based	Latency
		Scheduling	ASAP	
Cross [128]	Design flow	Layout	Grid-based and H-tree based	Latency
Metodiev [129]		Scheduling	ASAP	Fault Tolerance
Balensiefer et al. [87,88]	Design flow	Layout	Grid-based and H-tree based	Latency
		Routing	Pathfinder	Fault Tolerance
		Scheduling	list-scheduling	
Whitney et al. [118]	Layout Generation	Grid-based		Latency
Whitney et al. [118]	Design Flow	Layout	Dataflow-graph based	Latency-Area tradeoff
		Scheduling	QPOS	
QSPR [131]	Design Flow	Layout	Dataflow-graph based	Latency
		Scheduling	prioritized one	
Yazdani et al. [126]	Design flow	Layout	Giotto graph drawing algorithm	Latency
		Scheduling	ILP	
Moghadam et al. [122]	Layout Generation	Layout	Hierarchical min-cut based Placement	Latency
		Scheduling	ASAP	
QUFD [134]	Design Flow	Layout	force-directed placement	Fault Tolerance
		Scheduling	List-scheduling + Force-directed	Latency
Squash [101]	Design Flow	Layout	ILP	Latency
		Scheduling	list-Scheduling	
Isailovic et al. [89]	Interconnection Network	Entanglement swapping		Latency
Metodi et al. [143]	Interconnection Network	Entanglement swapping and purify linearly		Latency
GLC [121]	Post-layout optimization	Gate location changing		Latency
MSA_QSPP [125]	Scheduling	MINLP		Latency
GE [90,120]	Physical Synthesis	Gate Exchanging		Latency
AQS [27]	Physical Synthesis	Auxiliary qubit selection		Latency
AQI [123]	Physical Synthesis	Auxiliary qubit insertion		Latency
Mohammadzadeh et al. [123]	Physical synthesis flow	AQS, AQI, and GE		Latency
Ahsan et al. [137,138]	Design Flow	Layout	Polynomial Time Graph-Theory algorithm solving Optimal Linear Arrangement Problem	Latency
		Scheduling	Greedy ASAP	

4.2.6. *Physical Synthesis*

Physical synthesis manipulates the netlist or layout locally using the scheduling and physical layout information to optimize the objectives or meet the design requirements. This concept, the interaction between synthesis and physical design processes, was presented in the 1990s for the classical CMOS design flow. In [90,120], Mohammadzadeh et al. first brought this concept to the quantum circuit flow and proposed a technique, called gate exchanging, for it. This heuristic attempts to properly order two swappable gates based on the layout and scheduling information. In their next papers [27,123], the authors presented two techniques, called auxiliary qubit selection (AQS) and auxiliary qubit insertion (AQI), for the physical synthesis of quantum circuits. The AQS technique [27] uses the layout and scheduling information to properly choose auxiliary qubits for decomposing of macrogates. The AQI method [123] inserts some auxiliary qubits for better decomposing of the macrogates that are on the critical path to reach a lower latency while preserving the overall circuit functionality. Mohammadzadeh et al. [123] proposed a methodological framework for physical synthesis that includes all above-mentioned techniques.

Main algorithms proposed for automation of physical design are mentioned and compared in Table 2.

4.2.7. *Fault tolerance consideration*

Errors are many orders of magnitude more probable in quantum logic than in classical CMOS one. This fact forces an extra requirement on a quantum circuit to be strongly fault-tolerant. Unlike classical bits, quantum bits do not suffer only from a bit-flip error. Therefore, traditional error-correcting codes (QECC) cannot be used directly and should be modified. Because of the more fragility of quantum states (e.g. superposition), quantum bits undergo not only amplitude and phase-flip errors, but almost any single-qubit unitary operator error is probable. Moreover, one of the fundamental principles of quantum mechanics known as the no-cloning theorem [6] challenges quantum computation. This theorem states that it is impossible to create an identical copy of an arbitrary unknown quantum state. This means that quantum error correction cannot utilize the classical techniques, but that it must introduce a new set of error-correcting codes. The overall principle of quantum error correction is fundamentally the same to classical one where a majority vote is used to detect the type of error happened. It differs from classical error correction in using the properties of entanglement to encode physical qubits into logical qubits. In the quantum error correction, parity is measured using the two-qubit CNOT gate instead of the classical XOR gate. Entanglement is used to prevent direct measurement and collapse of the state of the system and to provide fault tolerance from one step to the next. For a more detailed discussion of error correcting codes, the interested reader is referred to the literature [6,149,150].

Some works considered fault tolerance in the physical design. The Monte-Carlo simulation method was used in references [87,88,129] to simulate a fault-tolerant ion-trap system and evaluate the critical threshold. The underlying organization of QLA [97–99] is designed for fault tolerance that uses the [1,3,7] Steane code at the level two recursion. This microarchitecture is suitable for quantum error correction because the organization of its basic blocks is compatible with the error-correction algorithm used. Cross [128] used the general set counting to calculate threshold bounds, the Monte-Carlo simulation to estimate depolarizing noise thresholds, CHP[18] to functionally verify quantum stabilizer circuits [31]. CQLA [100] uses a level two encoding for the memory that is slow and reliable, a level one encoding for cache that is faster and less reliable, and a level one encoding for the compute region that is fastest and same reliability as cache.

4.2.7.1. Ancilla factories. Steane originally introduced the idea of an ancilla factory in [151]. The actual correction of data takes clean encoded zero ancilla. The main idea for ancilla factory is that specialized modules produce this ancilla. Whitney et al. [119] adapted Svore ancilla factory for the [1,3,7] code [152]. This factory was designed to minimize movement and idle errors, so it is not particularly area efficient. Kreger-Stickles et al. [102] proposed 4-bit Linear Offset factory for generating [1,3,7] ancilla. This design is the best one in terms of fault point count as well as latency. Moreover, its area is very small. In [103,119], Isailovic et al. presented a pipelined ancilla factory for the [1,3,7] that attempted to maximize ancilla throughput and can ready multiple ancilla in parallel.

4.2.7.2. Optimizing QEC. The traditional brute-force approach for QEC[19] corrects errors after every gate to guarantee that other qubits are not infected. This strategy has two disadvantages. First, it consumes considerable resources. Under this strategy, more than 90% of physical operations will be consumed for error correction purpose instead of useful computation [96]. Second, although this strategy attempts to overcome errors, it adds too many operations and movement operations, which are likely to cause new errors. Considering these drawbacks, Whitney et al. [96] proposed a QEC synthesis flow to perform selective error correction, first introduced in [153]. They customized the retiming idea [154] from the classical CAD literature to model and solve this problem. The authors of [104] suggested the selective XZ error correction strategy which improves Whitney et al.'s proposal [96] by differentiating between bit-flip errors and phase-flip errors that causes a better estimation of error propagation.

5. Software tools

The QUALE[20] [87] is a tool chain to study quantum architectures. The flow provided with this tool takes a description of a quantum algorithm in QCL [113], compiles it, adds fault tolerance support to the generated netlist, schedules the netlist, and simulates the circuit on a simple tile-based layout to analyze the speed and reliability of the algorithm. The open-source software package LEQA[21] [155,156] was developed to estimate the latency of a quantum circuit on a tile-based layout by computing the neighborhood population counts of qubits. The toolkit QSPR developed by Dousti and Pedram [131,157] is an open-source mapping tool that schedules, places, and routes a quantum netlist on the ion-trap layout. The toolset QUFD[22] [134] produces the optimal Universal Logic Block (ULB) that can perform any logical fault-tolerant (FT) quantum gate with the minimum latency.

Dousti et al. [101] proposed a scalable quantum mapper considering ancilla sharing, called Squash. It at first partitions a quantum circuit into some cores. Next, a quantum operation dependency graph (QODG) is constructed for each core and divided into k sub-graphs by METIS tool [158]. These sub-graphs are then scheduled and mapped to the Requp architecture with k cores. Finally, resultant mappings are combined to create the entire mapping of the given circuit.

Ahsan et al. [137,138] developed an open-source toolbox for the

[18] CNOT-Hadamard-Phase

[19] Quantum error correction

[20] Quantum architecture layout evaluator.

[21] Latency estimation for a quantum algorithm.

[22] Quantum ULB factory designer.

MUSIQC architecture that includes two main components: Tile Designer and Performance Analyzer (TDPA) and Architecture Designer and Performance Analyzer (ADPA). The main tasks of both components are mapping, scheduling, and error analysis.

6. Conclusion and future directions

Quantum logic circuit design has been studied for more than 30 years [159] because of its potential to perform certain classes of problems more efficiently than traditional classical computer. Physical design of quantum circuits is typically partitioned into (I) placement and routing, (II) scheduling, (III) physical synthesis, and (IV) optimization that attempts to improve circuit metrics. The ion-trap technology has been selected as the target technology in the most papers because it is the most advanced one for realizing quantum circuits to date. Despite significant research in quantum logic synthesis, physical design of quantum circuits in ion trap technology has recently attracted attentions. Focusing on this issue, in this paper, I surveyed the methods proposed for physical design automation of quantum circuits in ion trap technology.

New concepts and techniques may be proposed for each process in the physical design flow. Further considerations for future research in the quantum computation field are summarized below.

- **New quantum algorithms:** Some quantum algorithms have already been found such as Shor's algorithm and its extensions, Grover's algorithm, graph reachability via quantum walks, etc. However, some other problems exist that can be potentially benefited from quantum computation power, but any quantum algorithm has not been proposed for them. Some candidate problems are group-theoretic problems, lattice problems, and simulating physical systems.
- **Quantum circuit synthesis:** Despite the significant research on the synthesis of reversible circuits, a few studies have explored the synthesis of quantum circuits [91]. Moreover, the input of current synthesis techniques is not high-level. In other words, generating a quantum netlist from a high-level description is currently a challenge.
- **Quantum error correction:** Quantum fault tolerance techniques offer a solution to the problem of protecting quantum systems against noise induced by interactions with the environment or caused by imperfect control of the quantum system. No-cloning theorem, continuous nature of quantum errors, and destruction of quantum information due to measurement are challenges in quantum error correction. Therefore, proposing novel fault-tolerance techniques are still appreciated.
- **Technology abstraction:** Successful implementation of a quantum computer is a joint cooperation that needs different fields and expertize. All of this expertize cannot be possessed by a single group, so each group should concentrate on one aspect with abstracting further details. This is very similar to the current approach in the classical CMOS design that abstract models are used by higher level designers and low level implementation and associated physical challenges are left to fabrication engineers. Focusing on this issue, extracting a standard abstract model of ion trap technology can be one of the main future tasks.
- **Quantum physical design automation:** The works done on the physical design in ion trap technology are reviewed and explored in this paper. This study shows that the physical design automation of quantum circuits is not a mature area like that for traditional classic circuits. On the other hand, the proposed algorithms in physical design each one has focus on one metric. Therefore, proposing new methods that simultaneously optimize multiple metrics such as area, latency, and fault tolerance can be one of the future research directions in this field.
- **CAD tools:** In spite of the fact that many factors challenge the development of a practical quantum computer, it is believed that the design space for a future quantum computer should be explored now because it can be useful in organizing the variety of proposed technologies, fault tolerance methods, and other realization choices. Quantum computation has progressed to the point where system-level solutions can be useful for closing the gap between emerging quantum technologies and real-world computing requirements. Design tools for physical design of quantum circuits in ion trap technology have been developed and reported by a number of groups, but in most cases, they are restricted to specific algorithms or do not use or model all aspects of the technology. Developing of some powerful interoperating CAD tools may be necessary to scale quantum circuit design beyond its current limitations.

References

[1] T.P. Spiller, W.J. Munro, S.D. Barrett, P. Kok, An introduction to quantum information processing: applications and realizations, Contemp. Phys. 46 (2005) 407–436.
[2] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: 1994 Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134.
[3] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219.
[4] C. Zalka, Simulating quantum systems on a quantum computer, Proc. R. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. 454 (1998) 313–322.
[5] D. Aharonov, V. Jones, Z. Landau, A polynomial quantum algorithm for approximating the Jones polynomial, Algorithmica 55 (2009) 395–421.
[6] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, New York, 2010.
[7] J.I. Cirac, P. Zoller, Quantum computations with cold trapped ions, Phys. Rev. Lett. 74 (1995) 4091.
[8] C. Monroe, D. Meekhof, B. King, W. Itano, D. Wineland, Demonstration of a fundamental quantum logic gate, Phys. Rev. Lett. 75 (1995) 4714.
[9] F. Schmidt-Kaler, H. Häffner, M. Riebe, S. Gulde, G.P. Lancaster, T. Deuschle, et al., Realization of the Cirac–Zoller controlled-NOT quantum gate, Nature 422 (2003) 408–411.
[10] D. Leibfried, B. DeMarco, V. Meyer, D. Lucas, M. Barrett, J. Britton, et al., Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate, Nature 422 (2003) 412–415.
[11] M. Riebe, H. Häffner, C. Roos, W. Hänsel, J. Benhelm, G. Lancaster, et al., Deterministic quantum teleportation with atoms, Nature 429 (2004) 734–737.
[12] M. Barrett, J. Chiaverini, T. Schaetz, J. Britton, W. Itano, J. Jost, et al., Deterministic quantum teleportation of atomic qubits, Nature 429 (2004) 737–739.
[13] B. Blinov, D. Moehring, L.-M. Duan, C. Monroe, Observation of entanglement between a single trapped atom and a single photon, Nature 428 (2004) 153–157.
[14] D. Kielpinski, C. Monroe, D.J. Wineland, Architecture for a large-scale ion-trap quantum computer, Nature 417 (2002) 709–711.
[15] T. Schaetz, D. Leibfried, J. Chiaverini, M. Barrett, J. Britton, B. DeMarco, et al., Towards a scalable quantum computer/simulator based on trapped ions, Appl. Phys. B 79 (2004) 979–986.
[16] J. Kim, S. Pau, Z. Ma, H. McLellan, J. Gates, A. Kornblit, et al., System design for large-scale ion trap quantum information processor, Quantum Inf. Comput. 5 (2005) 515–537.
[17] A.M. Steane, How to build a 300 bit, 1 Giga-operation quantum computer, Quantum Inf. Comput. 7 (2007) 171–183.
[18] D. Stick, J. Sterk, C. Monroe, The Trap Technique, IEEE Spectr. 44 (2007) 36–43.
[19] C. Monroe, J. Kim, Scaling the ion trap quantum processor, Science 339 (2013) 1164–1169.
[20] P. Schindler, D. Nigg, T. Monz, J.T. Barreiro, E. Martinez, S.X. Wang, et al., A quantum information processor with trapped ions, New J. Phys. 15 (2013) 123012.
[21] D. Maslov, S.M. Falconer, M. Mosca, Quantum circuit placement, IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 27 (2008) 752–763.
[22] A. Shafaei, M. Saeedi, M. Pedram, Qubit placement to minimize communication overhead in 2D quantum architectures, in: Proceedings of the 2014 19th Asia and South PacificDesign Automation Conference (ASP-DAC), 2014, pp. 495–500.
[23] C.-C. Lin, S. Sur-Kolay, N.K. Jha, PAQCS: Physical Design-Aware Fault-Tolerant Quantum Circuit Synthesis.

[24] B. Schumacher, Quantum coding, Phys. Rev. A 51 (1995) 2738.
[25] J.A. Jones, NMR quantum computation, Prog. Nucl. Magn. Reson. Spectrosc. 38 (2001) 325–360.
[26] V.V. Shende, A.K. Prasad, I.L. Markov, J.P. Hayes, Synthesis of reversible logic circuits, IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 22 (2003) 710–722.
[27] N. Mohammadzadeh, M.S. Zamani, M. Sedighi, Auxiliary qubit selection: a physical synthesis technique for quantum circuits, Quantum Inf. Process. 10 (2011) 139–154.
[28] T.S. Metodi, A.I. Faruque, F.T. Chong, Quantum computing for computer architects, Synth. Lect. Comput. Arch. 6 (2011) 1–203.
[29] S. Aaronson, B. Toth, Simulation and Synthesis of Stabilizer Quantum Circuits, 2003.
[30] G.F. Viamontes, I.L. Markov, J.P. Hayes, Quantum Circuit Simulation, Springer, New York, 2009.
[31] S. Aaronson, D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A 70 (2004) 052328.
[32] D. Aharonov, M. Ben-Or, Fault-tolerant quantum computation with constant error, in: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, 1997, pp. 176–188.
[33] D.P. DiVincenzo, The physical implementation of quantum computation, arXiv Preprint quant-ph/0002077, 2000.
[34] T.D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, J.L. O'Brien, Quantum computers, Nature 464 (2010) 45–53.
[35] C.H. Bennett, G. Brassard, A Quantum Information Science and Technology Roadmap, Part 2 (2004) 12.
[36] D. Wineland, J. Bergquist, W.M. Itano, R. Drullinger, Double-resonance and optical-pumping experiments on electromagnetically confined, laser-cooled ions, Opt. Lett. 5 (1980) 245–247.
[37] W. Nagourney, J. Sandberg, H. Dehmelt, Shelved optical electron amplifier: observation of quantum jumps, Phys. Rev. Lett. 56 (1986) 2797.
[38] T. Sauter, W. Neuhauser, R. Blatt, P. Toschek, Observation of quantum jumps, Phys. Rev. Lett. 57 (1986) 1696.
[39] J. Bergquist, R.G. Hulet, W.M. Itano, D. Wineland, Observation of quantum jumps in a single atom, Phys. Rev. Lett. 57 (1986) 1699.
[40] J. Bollinger, D. Heizen, W. Itano, S. Gilbert, D. Wineland, A 303-MHz frequency standard based on trapped Be/sup+/ions, IEEE Trans. Instrum. Meas. 40 (1991) 126–128.
[41] F. Diedrich, E. Peik, J. Chen, W. Quint, H. Walther, Observation of a phase transition of stored laser-cooled ions, Phys. Rev. Lett. 59 (1987) 2931.
[42] D.J. Wineland, J. Bergquist, W.M. Itano, J. Bollinger, C. Manney, Atomic-ion Coulomb clusters in an ion trap, Phys. Rev. Lett. 59 (1987) 2935.
[43] M. Raizen, J. Gilligan, J. Bergquist, W. Itano, D. Wineland, Linear trap for high-accuracy spectroscopy of stored ions, J. Mod. Opt. 39 (1992) 233–242.
[44] M. Raizen, J. Gilligan, J. Bergquist, W. Itano, D. Wineland, Ionic crystals in a linear Paul trap, Phys. Rev. A 45 (1992) 6493.
[45] J.I. Cirac, P. Zoller, A scalable quantum computer with ions in an array of microtraps, Nature 404 (2000) 579–581.
[46] C. Sackett, D. Kielpinski, B. King, C. Langer, V. Meyer, C. Myatt, et al., Experimental entanglement of four particles, Nature 404 (2000) 256–259.
[47] B. DeMarco, A. Ben-Kish, D. Leibfried, V. Meyer, M. Rowe, B. Jelenković, et al., Experimental demonstration of a controlled-NOT wave-packet gate, Phys. Rev. Lett. 89 (2002) 267901.
[48] D. Kielpinski, V. Meyer, M. Rowe, C. Sackett, W. Itano, C. Monroe, et al., A decoherence-free quantum memory using trapped ions, Science 291 (2001) 1013–1015.
[49] D. Leibfried, B. DeMarco, V. Meyer, M. Rowe, A. Ben-Kish, J. Britton, et al., Trapped-ion quantum simulator: experimental application to nonlinear interferometers, Phys. Rev. Lett. 89 (2002) 247901.
[50] S. Gulde, H. Häffner, M. Riebe, G. Lancaster, C. Becher, J. Eschner, et al., Quantum information processing with trapped Ca+ ions, Philos. Trans. R. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. 361 (2003) 1363–1374.
[51] C.F. Roos, M. Riebe, H. Häffner, W. Hänsel, J. Benhelm, G.P. Lancaster, et al., Control and measurement of three-qubit entangled states, Science 304 (2004) 1478–1480.
[52] J. Chiaverini, D. Leibfried, T. Schaetz, M. Barrett, R. Blakestad, J. Britton, et al., Realization of quantum error correction, Nature 432 (2004) 602–605.
[53] D. Leibfried, E. Knill, S. Seidelin, J. Britton, R.B. Blakestad, J. Chiaverini, et al., Creation of a six-atom 'Schrödinger cat'state, Nature 438 (2005) 639–642.
[54] H. Häffner, W. Hänsel, C. Roos, J. Benhelm, Scalable multiparticle entanglement of trapped ions, Nature 438 (2005) 643–646.
[55] R. Reichle, D. Leibfried, E. Knill, J. Britton, R. Blakestad, J. Jost, et al., Experimental purification of two-atom entanglement, Nature 443 (2006) 838–841.
[56] P. Maunz, D. Moehring, S. Olmschenk, K. Younge, D. Matsukevich, C. Monroe, Quantum interference of photon pairs from two remote trapped atomic ions, Nat. Phys. 3 (2007) 538–541.
[57] D. Moehring, P. Maunz, S. Olmschenk, K. Younge, D. Matsukevich, L.-M. Duan, et al., Entanglement of single-atom quantum bits at a distance, Nature 449 (2007) 68–71.
[58] D. Matsukevich, P. Maunz, D. Moehring, S. Olmschenk, C. Monroe, Bell inequality violation with two remote atomic qubits, Phys. Rev. Lett. 100 (2008) 150404.
[59] D. J. Wineland, C. Monroe, W. Itano, D. Leibfried, B. King, D. Meekhof, Experimental issues in coherent quantum-state manipulation of trapped atomic ions, arXiv preprint quant-ph/9710025, 1997.
[60] M. Šašura, V. Bužek, Cold trapped ions as quantum information processors, J. Mod. Opt. 49 (2002) 1593–1647.
[61] D. Leibfried, R. Blatt, C. Monroe, D. Wineland, Quantum dynamics of single trapped ions, Rev. Mod. Phys. 75 (2003) 281.
[62] R. Blatt, D. Wineland, Entangled states of trapped atomic ions, Nature 453 (2008) 1008–1015.
[63] D.P. DiVincenzo, Dogma and heresy in quantum computing, Quantum Inf. Comput. 1 (2001) 1–6.
[64] C. Langer, R. Ozeri, J.D. Jost, J. Chiaverini, B. DeMarco, A. Ben-Kish, et al., Long-lived qubit memory using atomic ions, Phys. Rev. Lett. 95 (2005) 060502.
[65] D. Lucas, B. Keitch, J. Home, G. Imreh, M. McDonnell, D. Stacey, et al., A long-lived memory qubit on a low-decoherence quantum bus, arXiv preprint arXiv:0710.4421, 2007.
[66] J. Benhelm, G. Kirchmair, C. Roos, R. Blatt, Experimental quantum-information processing with C 43 a+ ions, Phys. Rev. A 77 (2008) 062306.
[67] H. Häffner, C.F. Roos, R. Blatt, Quantum computing with trapped ions, Phys. Rep. 469 (2008) 155–203.
[68] T. Monz, K. Kim, W. Hänsel, M. Riebe, A. Villar, P. Schindler, et al., Realization of the quantum Toffoli gate with trapped ions, Phys. Rev. Lett. 102 (2009) 040501.
[69] P. Zou, J. Xu, W. Song, S.-L. Zhu, Implementation of local and high-fidelity quantum conditional phase gates in a scalable two-dimensional ion trap, Phys. Lett. A 374 (2010) 1425–1430.
[70] J. Cirac, P. Zoller, H. Kimble, H. Mabuchi, Quantum state transfer and entanglement distribution among distant nodes in a quantum network, Phys. Rev. Lett. 78 (1997) 3221.
[71] M. Rowe, A. Ben-Kish, B. Demarco, D. Leibfried, V. Meyer, J. Beall, et al., Transport of quantum states and separation of ions in a dual RF ion trap, arXiv preprint quant-ph/0205094, 2002.
[72] J. Chiaverini, R.B. Blakestad, J. Britton, J.D. Jost, C. Langer, D. Leibfried, et al., Surface-electrode architecture for ion-trap quantum information processing, arXiv preprint quant-ph/0501147, 2005.
[73] M.D. Barrett, B. DeMarco, T. Schaetz, V. Meyer, D. Leibfried, J. Britton, et al., Sympathetic cooling of 9 Be+ and 24 Mg+ for quantum logic, Phys. Rev. A 68 (2003) 042302.
[74] B. Blinov, L. Deslauriers, P. Lee, M. Madsen, R. Miller, C. Monroe, Sympathetic cooling of trapped Cd+ isotopes, Phys. Rev. A 65 (2002) 040304.
[75] C. Monroe, R. Raussendorf, A. Ruthven, K. Brown, P. Maunz, L.-M. Duan, et al., Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects, Phys. Rev. A 89 (2014) 022317.
[76] J. Stajic, The future of quantum information processing, Science 339 (2013) 1163.
[77] D. Hucul, M. Yeo, S. Olmschenk, C. Monroe, W.K. Hensinger, J. Rabchuk, On the transport of atomic ions in linear and multidimensional ion trap arrays, Quantum Inf. Comput. 8 (2008) 501–578.
[78] W. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, et al., T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation, Appl. Phys. Lett. 88 (2006) 034101.
[79] M. Madsen, W. Hensinger, D. Stick, J. Rabchuk, C. Monroe, Planar ion trap geometry for microfabrication, Appl. Phys. B 78 (2004) 639–651.
[80] C.H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W.K. Wootters, Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels, Phys. Rev. Lett. 70 (1993) 1895.
[81] D.A. Sofge, "A survey of quantum programming languages: History, methods, and tools," in: Proceedings of the 2008 Second International Conference on Quantum, Nano and Micro Technologies, 2008, pp. 66–71.
[82] S.J. Gay, Quantum programming languages: Survey and bibliography, Math. Struct. Comput. Sci. 16 (2006) 581–600.
[83] P. Selinger, A brief survey of quantum programming languages, in: Functional and Logic Programming, ed: Springer, 2004, pp. 1–6.
[84] A.S. Green, P.L. Lumsdaine, N.J. Ross, P. Selinger, B. Valiron, An introduction to quantum programming in Quipper, in: Reversible Computation, ed: Springer, 2013, pp. 110–124.
[85] M. Ying, N. Yu, Y. Feng, Defining quantum control flow, arXiv preprint arXiv:1209.4379, 2012.
[86] J.-F. Xu, F.-M. Song, S.-J. Qian, J.-A. Dai, Y.-J. Zhang, Quantum programming language NDQJava, J. Softw. 19 (2008) 1–8.
[87] S. Balensiefer, L. Kreger-stickles, and M. Oskin, QUALE: Quantum architecture layout evaluator, in: Proceedings of SPIE, the International Society for Optical Engineering, 2005, pp. 103–114.
[88] S. Balensiefer, L. Kregor-Stickles, M. Oskin, An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures, in: ACM SIGARCH Computer Architecture News, 2005, pp. 186–196.
[89] N. Isailovic, Y. Patel, M. Whitney, J. Kubiatowicz, Interconnection networks for scalable quantum computers, ACM SIGARCH Comput. Arch. News 34 (2006) 366–377.
[90] N. Mohammadzadeh, M. S. Zamani, M. Sedighi, Improving latency of quantum circuits by gate exchanging, in: Proceedings of the 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, 2009. DSD'09, 2009, pp. 67–73.
[91] M. Saeedi, I.L. Markov, Synthesis and optimization of reversible circuits—a survey, ACM Comput. Surv. (CSUR) 45 (2013) 21.
[92] V.V. Shende, S.S. Bullock, I.L. Markov, Synthesis of quantum-logic circuits, IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 25 (2006) 1000–1010.
[93] C.-C. Lin, Reversible and Quantum Circuit Synthesis, Princeton University, Princeton, 2014.

[94] C.-C. Lin, A. Chakrabarti, N.K. Jha, FTQLS: Fault-Tolerant Quantum Logic Synthesis.
[95] S.B. Mandal, A. Chakrabarti, S. Sur-Kolay, A synthesis method for quaternary quantum logic circuits, in Progress in VLSI Design and Test, ed: Springer, 2012, pp. 270–280.
[96] M.G. Whitney, N. Isailovic, Y. Patel, J. Kubiatowicz, A fault tolerant, area efficient architecture for Shor's factoring algorithm, ACM SIGARCH Comput. Arch. News 37 (2009) 383–394.
[97] T.S. Metodi, D.D. Thaker, A. W. Cross, F. T. Chong, I.L. Chuang, A quantum logic array microarchitecture: Scalable quantum data movement and computation, in: Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture, 2005. MICRO-38, 2005, p. 12 pp.
[98] T. S. Metodi, D. Thaker, A. W. Cross, F. T. Chong, I.L. Chuang, A general purpose architectural layout for arbitrary quantum computations, in Defense and Security, 2005, pp. 91–102.
[99] T. Metodi, D. Thaker, A. Cross, F.T. Chong, I.L. Chuang, QLA: Quantum Logic Array Microarchitecture, a Brief Overview, Recreation Pool Lodge Davis, California, USA October 8th, 2005, p. 2.
[100] D.D. Thaker, T.S. Metodi, A.W. Cross, I.L. Chuang, F.T. Chong, Quantum memory hierarchies: Efficient designs to match available parallelism in quantum computing, ACM SIGARCH Comput. Arch. News (2006) 378–390.
[101] M.J. Dousti, A. Shafaei, 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.
[102] L. Kreger-Stickles, M. Oskin, Microcoded architectures for ion-tap quantum computers, ACM SIGARCH Comput. Arch. News 36 (2008) 165–176.
[103] N. Isailovic, M. Whitney, Y. Patel, J. Kubiatowicz, Running a quantum circuit at the speed of data, in ACM SIGARCH Computer Architecture News, 2008, pp. 177–188.
[104] G. Wang, O. Khainovski, A Fault-tolerant, Ion-trap-based Architecture for the Quantum Simulation Algorithm, Measurement, vol. 10, pp. 10-4.
[105] L. K. Grover, Quantum telecomputation, arXiv preprint quant-ph/9704012, 1997.
[106] J. Cirac, A. Ekert, S. Huelga, C. Macchiavello, Distributed quantum computation over noisy channels, Phys. Rev. A 59 (1999) 4249.
[107] Y.L. Lim, S.D. Barrett, A. Beige, P. Kok, L.C. Kwek, Repeat-until-success quantum computing using stationary and flying qubits, Phys. Rev. A 73 (2006) 012304.
[108] D.K. Oi, S.J. Devitt, L.C. Hollenberg, Scalable error correction in distributed ion trap computers, Phys. Rev. A 74 (2006) 052313.
[109] D.D. Thaker, T.S. Metodi, F.T. Chong, A realizable distributed ion-trap quantum computer, in High Performance Computing-HiPC 2006, ed: Springer, 2006, pp. 111–122.
[110] A.M. Steane, Overhead and noise threshold of fault-tolerant quantum error correction, Phys. Rev. A 68 (2003) 042322.
[111] D. Maslov, G.W. Dueck, D.M. Miller, C. Negrevergne, Quantum circuit simplification and level compaction, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 27 (2008) 436–444.
[112] Z. Sasanian, Technology mapping and optimization for reversible and quantum circuits, University of Victoria, Victoria, 2012.
[113] B. Ömer, Quantum Programming in QCL: na, 2000.
[114] A.M. Steane, Error correcting codes in quantum theory, Phys. Rev. Lett. 77 (1996) 793.
[115] D. Aharonov, Noisy quantum computation, Hebrew University of Jerusalem, Jerusalem, 1999.
[116] K. Svore, A. Cross, A. Aho, I. Chuang, I. Markov, Toward a software architecture for quantum computing design tools, in: Proceedings of the 2nd International Workshop on Quantum Programming Languages (QPL), 2004, pp. 145–162.
[117] K.M. Svore, A.V. Aho, A.W. Cross, I. Chuang, I.L. Markov, A layered software architecture for quantum computing design tools, IEEE Comput. 39 (2006) 74–83.
[118] M. Whitney, N. Isailovic, Y. Patel, J. Kubiatowicz, Automated generation of layout and control for quantum circuits, in: Proceedings of the 4th International Conference on Computing Frontiers, 2007, pp. 83–94.
[119] M.G. Whitney, Practical Fault Tolerance for Quantum Circuits, University of California, Berkeley, 2009.
[120] N. Mohammadzadeh, M. Sedighi, M. Saheb Zamani, Quantum physical synthesis: improving physical design by netlist modifications, Microelectron. J. 41 (2010) 219–230.
[121] N. Mohammadzadeh, M. Sedighi, M.S. Zamani, Gate location changing: an optimization technique for quantum circuits, Int. J. Quantum Inf. 10 (2012).
[122] M.C. Moghadam, N. Mohammadzadeh, M. Sedighi, M.S. Zamani, A hierarchical layout generation method for quantum circuits," in: Proceedings of the 2013 17th CSI International Symposium on Computer Architecture and Digital Systems (CADS), 2013, pp. 51-57.
[123] N. Mohammadzadeh, M.S. Zamani, M. Sedighi, Quantum circuit physical design methodology with emphasis on physical synthesis, Quantum Inf. Process. 13 (2014) 445–465.
[124] N. Mohammadzadeh, T. Bahreini, H. Badri, Optimal ILP-based approach for gate location assignment and scheduling in quantum circuits, Model. Simul. Eng. 2014 (2014) 7.
[125] T. Bahreini, N. Mohammadzadeh, An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach, ACM J. Emerg. Technol. Comput. Syst. (JETC) 12 (2015) 29.
[126] M. Yazdani, M.S. Zamani, M. Sedighi, A quantum physical design flow using ILP and graph drawing, Quantum Inf. Process. 12 (2013) 3239–3264.
[127] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "Scheduling physical operations in a quantum information processor," in *Defense and Security Symposium*, 2006, pp. 62440T-62440T-12.
[128] A.W. Cross, "Synthesis and evaluation of fault-tolerant quantum computer architectures,", Massachusetts Institute of Technology, 2005.
[129] T. Metodiev, A. Cross, D. Thaker, K. Brown, D. Copsey, F.T. Chong, et al., Preliminary results on simulating a scalable fault tolerant ion-trap system for quantum computation, in: Proceedings of the 3rd Workshop on Non-silicon Computing, 2004.
[130] L. McMurchie, C. Ebeling, PathFinder: a negotiation-based performance-driven router for FPGAs, in: Proceedings of the 1995 ACM Third International Symposium on Field-programmable Gate Arrays, 1995, pp. 111–117.
[131] M.J. Dousti and M. Pedram, Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric, in: Proceedings of the Conference on Design, Automation and Test in Europe, 2012, pp. 840–843.
[132] R. Tamassia, G. Di Battista, C. Batini, Automatic graph drawing and readability of diagrams, IEEE Trans. Syst. Man Cybern. 18 (1988) 61–79.
[133] C.J. Alpert, D.P. Mehta, S.S. Sapatnekar, Handbook of Algorithms for Physical Design Automation, CRC Press, 2008.
[134] H. Goudarzi, M. Dousti, A. Shafaei, M. Pedram, Design of a universal logic block for fault-tolerant realization of any logic operation in trapped-ion quantum circuits, Quantum Inf. Process. 13 (2014) 1267–1299.
[135] M.-C. Kim, D.-J. Lee, I.L. Markov, SimPL: An effective placement algorithm, IEEE Trans. Computer-Aided Des. Integr. Circuits Syst. 31 (2012) 50–60.
[136] R. Bixby, Z. Gu, E. Rothberg, Gurobi optimizater, ed, 2010.
[137] M. Ahsan, R. Van Meter, J. Kim, Designing a Million-Qubit Quantum Computer Using Resource Performance Simulator.
[138] M. Ahsan, J. Kim, Optimization of quantum computer architecture using a resource-performance simulator, in: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, 2015, pp. 1108–1113.
[139] M. Juvan, B. Mohar, Optimal linear labelings and eigenvalues of graphs, Discret. Appl. Math. 36 (1992) 153–168.
[140] M. Zukowski, A. Zeilinger, M. Horne, A. Ekert, "Event-ready-detectors" bell experiment via entanglement swapping, Phys. Rev. Lett. 71 (1993) 4287–4290.
[141] C.H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J.A. Smolin, W. K. Wootters, Purification of noisy entanglement and faithful teleportation via noisy channels, Phys. Rev. Lett. 76 (1996) 722.
[142] V.S. Adve, M.K. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, IEEE Trans. Parallel Distrib. Syst. 5 (1994) 225–246.
[143] T.S. Metodi, D.D. Thaker, A.W. Cross, I.L. Chuang, F.T. Chong, High-level interconnect model for the quantum logic array architecture, ACM J. Emerg. Technol. Comput. Syst. (JETC) 4 (2008) 1.
[144] T. Yang, A. Gerasoulis, List scheduling with and without communication delays, Parallel Comput. 19 (1993) 1321–1344.
[145] C. Chekuri, R. Johnson, R. Motwani, B. Natarajan, B.R. Rau, M. Schlansker, Profile-driven instruction level parallel scheduling with application to super blocks, in: Proceedings of the 29th Annual ACM/IEEE International Symposium on Microarchitecture, 1996, pp. 58–67.
[146] B.L. Deitrich, W.-M. Hwu, Speculative hedge: Regulating compile-time speculation against profile variations, in: Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture, 1996, MICRO-29, 1996, pp. 70–79.
[147] A.E. Eichenberger, W.M. Meleis, Balance scheduling: Weighting branch tradeoffs in superblocks, in: Proceedings of the 32nd Annual International Symposium on Microarchitecture, 1999, MICRO-32, 1999, pp. 272–283.
[148] S. Davidson, D. Landskov, B.D. Shriver, P.W. Mallett, Some experiments in local microcode compaction for horizontal machines, IEEE Trans. Comput. 100 (1981) 460–477.
[149] A.W. Cross, D.P. DiVincenzo, B.M. Terhal, A comparative code study for quantum fault-tolerance, arXiv preprint arXiv:0711.1556, 2007.
[150] D.A. Lidar, T.A. Brun, Quantum Error Correction, Cambridge University Press, Cambridge, United Kingdom, 2013.
[151] A. Steane, Space, time, parallelism and noise requirements for reliable quantum computing, arXiv preprint quant-ph/9708021, 1997.
[152] K.M. Svore, D.P. DiVincenzo, B.M. Terhal, Noise threshold for a fault-tolerant two-dimensional lattice architecture, arXiv preprint quant-ph/0604090, 2006.
[153] M. Oskin, F.T. Chong, I.L. Chuang, A practical architecture for reliable quantum computers, Computer 35 (2002) 79–87.
[154] C. E. Leiserson, F.M. Rose, J.B. Saxe, Optimizing synchronous circuitry by retiming (preliminary version), in: Proceedings of the Third Caltech Conference on Very Large Scale Integration, 1983, pp. 87–116.
[155] D. MJ, LEQA, 2012.
[156] M.J. Dousti, M. Pedram, LEQA: latency estimation for a quantum algorithm mapped to a quantum circuit fabric, in: Proceedings of the 50th Annual Design Automation Conference, 2013, p. 42.
[157] D. M. P. M, 2012.
[158] G. Karypis, V. Kumar, Multilevel algorithms for multi-constraint graph partitioning, in: Proceedings of the 1998 ACM/IEEE conference on Supercomputing, 1998, pp. 1–13.
[159] R.P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. 21 (1982) 467–488.