Layout Synthesis for Topological Quantum Circuits With 1-D and 2-D Architectures

Yibo Lin, *Student Member, IEEE*, Bei Yu, *Member, IEEE*, Meng Li, *Student Member, IEEE*, and David Z. Pan, *Fellow, IEEE*

***Abstract*—Quantum computing has raised great interests for its potential to achieve an asymptotic speedup on specific problems. Current quantum devices suffer from noise which needs robust and scalable error-correcting schemes. Topological quantum error correction (TQEC) is among the most promising error-correcting techniques with exponential suppression of error with linear increase of space-time complexity. In this paper, we present the first work to explore space-time optimization between 1-D and 2-D architectures for TQEC circuits. We prove the NP-hardness of the qubit routing problem in the layout synthesis and propose an efficient algorithm to optimize space-time volumes for both 1-D and 2-D qubit architectures with promising experimental results.**

***Index Terms*—Layout synthesis, quantum computing, topological quantum error correction (TQEC).**

I. Introduction

QUANTUM computing is able to achieve asymptotic speedup on specific classes of problems, including data search [1] and cryptosystems [2]. Currently, quantum devices are not large enough to solve difficult problems in real world, where scalability is one of the critical issues. IBM has released its general quantum computer based on superconducting qubits via cloud, where users are granted with an access to a five-qubit quantum processor [3]. It is reported that the processor suffers from significant noise in the output results [4], indicating the urgent needs of fault-tolerant circuit design for scalability.

A *topological cluster state* is a kind of scheme for quantum computing with error correction using specific underlying structures tiled in a 3-D lattice [5]. Quantum error-correcting codes based on topological cluster states are capable of executing scalable quantum computation, with the probability of failure below 1% (threshold), which is considered as the state of the art in current technology [6]. Although there exist some other codes enabling the threshold as high as 3%, they suffer from high qubit overhead and long-range interactions between qubits, leading to impracticality of implementation [7]. The topological quantum error correction (TQEC) scheme is based on the Raussendorf code [8], which is a kind of error-corrected quantum circuits that operate on information encoded into topological cluster states. It enables exponential suppression of error with linear increase of space-time volume using only interactions between neighboring qubits. Here, space volume means the amount of resources used for quantum computing, such as number of qubits, and time volume denotes the required number of operation steps. The logical abstraction of TQEC utilizes topological cluster states, where a lattice of physical qubits are entangled into a large graph state for storage and operations of logical qubits.

To implement a quantum algorithm with TQEC circuits, it is necessary to go through several steps in the design flow, such as circuit implementation, decomposition, synthesis, mapping and technology mapping [7]. Circuit implementation maps a quantum algorithm to a quantum circuit by decomposing the algorithm into a specific set of quantum gates, proper initializations, and measurements. This is analogous to the logic synthesis in classical computing, where logic operations are replaced with primitive logic gates like inverters and AND gates with some input bits initialized to zeros or ones and the output results are measured. Then the circuits can be synthesized to generate the geometric description for technology related mapping in later steps like the physical design stage in digital circuits. Despite the similarity to classical circuits, due to different computing schemes and design architectures, various existing approaches for classical computing fail to work in the quantum case.

Previous works include circuit decomposition with different objectives, e.g., minimizing total number of qubits, the depth of quantum circuits, and different constraints from hardwares [9]–[16]. Paler *et al.* [17] proposed a compact representation of geometric description and the first automatic synthesis of geometric description from a circuit netlist, which we refer to as layout synthesis. Then they further propose a tool that incorporates decomposition of quantum circuits and synthesis of geometric description for 1-D arrangement of qubits, where all the qubits are placed along a 1-D line, aligning next to each other [6], as shown in Fig. 1(a). However, their approaches focus on generating feasible geometric descriptions without considering the optimization of space-time

Manuscript received May 5, 2017; revised July 27, 2017; accepted September 20, 2017. Date of publication October 6, 2017; date of current version July 17, 2018. This work was supported in part by the Chinese University of Hong Kong Direct Grant for Research, and in part by the University Graduate Continuing Fellowship from the University of Texas at Austin. This paper was recommended by Associate Editor L. Behjat. *(Corresponding author: Yibo Lin.)*

Y. Lin, M. Li, and D. Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78741 USA (e-mail: yibolin@utexas.edu).

B. Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2017.2760511

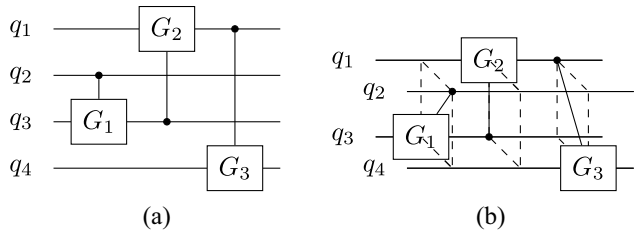


Fig. 1. Examples of (a) 1-D and (b) 2-D implementations of quantum circuits, where qubits are arranged in 1-D line or 2-D space [25]. Squares labeled with "G" represent quantum gates.

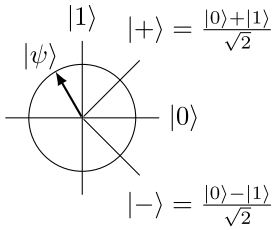


Fig. 2. Quantum state and different bases ($|0\rangle$ and $|1\rangle$ for Z-basis and $|+\rangle$ and $|-\rangle$ for X-basis).

volumes, which may result in inefficiency in completing all operations of the circuits, such as large latency. Yamashita [18] solved 1-D qubit and gate ordering problem by searching for maximum cliques in a graph model. Fowler and Devitt [19] identified some rules for topological conversion to simplify the geometric description with manual efforts. The geometric description of TQEC circuits can be easily mapped to physical hardware in polynomial time [20].

Although there are plenty of previous works on logic-level optimization of quantum operations, qubit placement and routing for linear nearest neighbor architectures [21]–[30], they assume the availability of SWAP gates for long-range interactions between qubits which is different from TQEC-related physical geometry. SWAP gates are used to bring qubits originally far away from each other into physically adjacent locations such that other functional gates involving both qubits can be applied. For example, in Fig. 1(a), we can insert an SWAP gate to swap q_2 and q_3 after gate G_1 such that the gate G_2 involving q_1 and q_3 can be implemented in linear nearest neighboring architectures. These works have proposed approaches to minimize SWAP gates in quantum circuits for not only 1-D qubit arrangement, but also 2-D structure, where qubits are placed in a 2-D grids, as shown in Fig. 1(b).

In this paper, we focus on the layout synthesis of TQEC circuits for general multiple-layer (1-D and 2-D) arrangement of qubits with space-time volume optimization, where qubits can be placed either in a 2-D space or a 1-D line from input configurations. Our major contributions are summarized as follows.

1) We propose the first systematic study on automatic layout synthesis of TQEC circuits with 1-D and 2-D architectures.
2) We prove the NP-hardness of the qubit routing problem.
3) We design an effective way to generate routing solutions for a single net utilizing the unique structure of the multiple-layer architecture and further propose an efficient qubit routing algorithm for the entire circuit.
4) We demonstrate the effectiveness of our algorithm in space-time volume optimization in the experimental results.

The rest of the this paper is organized as follows. Section II introduces basic concepts in TQEC and the problem formulation. Section III describes the algorithms for qubit routing to minimize space-time volumes in geometries. Then the algorithms are validated by experimental results in Sections IV and V concludes this paper.

II. Preliminaries

In this section, we will briefly introduce basic concepts in quantum computing and components in TQEC circuits followed by the problem formulation.

A. Qubits, Initialization, Measurement, and Gates

In quantum computing, information is passed by quantum bits (qubits) which can represent 0, 1, or superpositions of both [30]. The quantum state to hold the information is a unit vector usually represented with bra-ket notation shown in Fig. 2

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where $|0\rangle$ and $|1\rangle$ are orthonormal basis vectors. The probability of $|0\rangle$ is $|\alpha|^2$, the probability of $|1\rangle$ is $|\beta|^2$ and $|\alpha|^2 + |\beta|^2 = 1$. The qubit state can be represented in different bases. Fig. 2 shows two kinds of bases. Z-basis uses $|0\rangle$ and $|1\rangle$ as the basis vectors, and X-basis uses $|+\rangle$ and $|-\rangle$ as the basis vectors. The above quantum state can be written in X-basis as follows:

$$|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle. \tag{2}$$

To perform computation on qubits, initialization, and measurement on the state of qubits are necessary like that in classical computing for classical bits. Initialization and measurement have to be performed with specific basis, e.g., Z-basis or X-basis. Initialization usually adopts the same mechanism as measurement because quantum states collapse to the measuring basis when measurement is performed [31]. For example, measurement on Z-basis can initialize the qubit to $|0\rangle$ state and measurement on X-basis can initialize the qubit to $|+\rangle$.

The computation is then realized by quantum operations which perform transformations of quantum states of qubits, e.g., rotation of vector $|\psi\rangle$ in Fig. 2, where quantum gates are necessary to implement quantum operations for computation. It might be difficult to implement any quantum operation as a single quantum gate, while it is possible to come up with a finite set of primitive quantum gates that can realize any quantum operation by using them as building blocks, which is usually referred to as a universal set of gates, like the logic gates in classical digital circuits. The TQEC circuits use the universal set of gates $\{\text{CNOT}, V, P, T\}$ as the primitive gates to implement complicated operations [6], where the primitive gates V, P, and T are used for single qubit rotation and CNOT gate involves operations for multiple qubits.

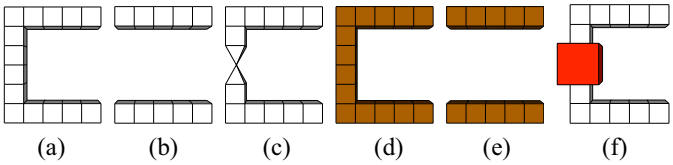


Fig. 3. Geometric components for measurement and initialization qubits. White cuboids denote primal defects and brown cubuids denote dual defects. (a) and (e) Z-basis measurement and $|0\rangle$ initialization. (b) and (d) X-basis measurement and $|+\rangle$ initialization. (c) State injection for $|A\rangle$ or $|Y\rangle$ initialization. (f) Generalized pin representation for primal defects, where the red cube generalizes the operation of initialization or measurement [6].

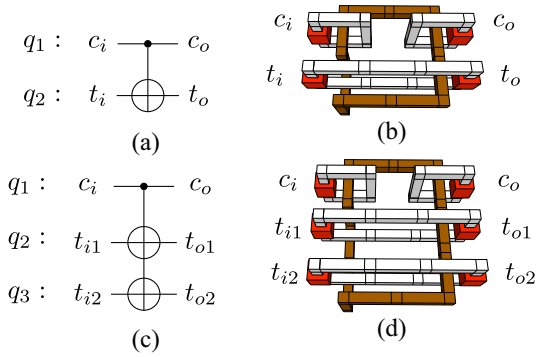


Fig. 4. (a) CNOT gate with q_1 as control qubit and q_2 as target qubit. (b) Primal-primal implementation of CNOT gate, where dual defects (brown) braid around primal defects (white) and red cubes represent inputs and outputs [8]. (c) Multiple-target CNOT gate with q_1 as control qubit and q_2 and q_3 as target qubits. (d) Geometric description of the multiple-target CNOT gate.

CNOT gate usually contains one control qubit and one target qubit and its functionality can be briefly explained as the control qubit determines whether or not to apply NOT operation on the target qubit, though the mathematics behind are more complicated. If we view the quantum state of a qubit as a vector, then the V, P, T gates rotate the vector in the space with various angles. These rotation gates can be implemented by teleportation-based schemes with CNOT gates and ancilla qubits initialized to $|A\rangle$ and $|Y\rangle$ [6]

$$|A\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle\right) \tag{3a}$$

$$|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \tag{3b}$$

where $|A\rangle$ and $|Y\rangle$ are orthonormal basis vectors like $|0\rangle$ and $|1\rangle$ in Z-basis. Therefore, TQEC circuits consist of CNOT gates and proper initialization and measurement of qubits.

B. Primal and Dual Defects

The quantum information in TQEC is encoded into topological cluster states which have a lattice structure, where the vertices are physical qubits. The removal of specific vertices from the lattice abstracts the state, where the results of removal are defined as *defects* [6]. The defects are generally represented by cuboids to describe the removal of vertices inside. TQEC circuits introduce a parallel pair of defects to represent a logical qubit. The propagation of defects behaves as the propagation of quantum states.

Fig. 3 shows the components for initialization and measurement in TQEC circuits [6]. We define the white cuboids as *primal defects* and brown cuboids as *dual defects*. Detailed description and explanation for the representation of primal and dual defects can be found in the previous work [6], [8], [31]. The major difference between primal and dual defects lies in the basis (Z-basis and X-basis) used for initialization and measurement. We only need to know the functionality of the defects and how TQEC circuits are composed with defects for layout synthesis. As aforementioned, a measurement on Z-basis initializes the qubit to $|0\rangle$ state and a measurement on X-basis initializes the qubit to $|+\rangle$. If the ends of two primal defects are joined like Fig. 3(a), Z-basis measurement is performed, while joining the ends of two dual defects means X-basis measurement as shown in Fig. 3(d). If the ends of two parallel defects are left open like Fig. 3(b) and (e), X-basis measurement is performed for primal defects and Z-basis measurement is performed for dual qubits. State injection, where two pyramid-shaped defects join one lattice vertex is adopted for the initialization of $|A\rangle$ and $|Y\rangle$ for ancilla shown as in Fig. 3(c). Since we focus on the generation of geometric description, the qubit initialization and measurement of inputs and outputs are abstracted to pins [red cube in Fig. 3(f)], which serve as placeholders for proper initialization and measurement on specific bases.

Fig. 4(b) implements a primal-primal CNOT gate [6], [8], meaning that qubits q_1 and q_2 are encoded to primal defects at both input and output. The ancilla dual defects braid around primal defects to form a single-target CNOT gate. Here, we use *braiding* to describe a path going through the face of a circle like a knot. For example, there are three braidings between primal and dual defects in Fig. 4(b). Be aware that Z-basis measurement is performed to the primal defects for c_i and Z-basis initialization is performed to the primal defects for c_o. This implementation is adopted to synthesize CNOT gates in TQEC circuits. The CNOT gate can also support more than one target, i.e., multiple-target CNOT gate with one control and multiple targets in Fig. 4(d).

C. Problem Formulation

With different configurations of qubit positions and geometry, the space-time volumes of TQEC circuits vary. Fig. 5 gives the geometries of the same TQEC circuit with both 1-D and 2-D qubit arrangements. Please recall the 1-D and 2-D architectures in Fig. 1. In Fig. 5(a), one CNOT gate connects qubits q_1 and q_3 and the other connects qubits q_2 and q_4. The two CNOT gates have the same *logic level* because their control and target signals are independent, which means two logic operations can be performed simultaneously. The width (w) and height (h)-axes in Fig. 5(b) denote the space dimensions, while the depth (d) axis represents the temporal/time dimension. Fig. 5(b) implements two CNOT gates using depth of 2 and Fig. 5(c) implements them with depth of 1 by stacking gates in parallel along h-axis. With similar space volumes of Fig. 5(b) and (c), two arrangements end up with different time volumes, as one can share depth steps between gates of

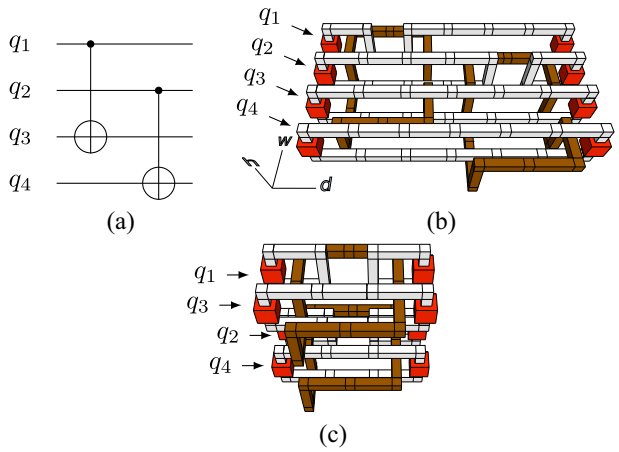


Fig. 5. Example of (a) circuit consisting of two CNOT gates with (b) 1-D (single-layer) qubit arrangement with depth of 2 and (c) 2-D (two-layer) qubit arrangement with depth of 1. Axis d denotes the depth axis. Axis w and h are axes for space volumes.

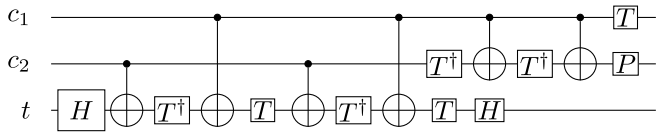


Fig. 6. Example of Toffoli gate implemented by a sequence of CNOT, T, $T^\dagger$, P, and H gates [32]. Its TQEC implementation, where T, $T^\dagger$, P, and H gates are implemented using teleportations can been seen in [6, Fig. 20].

the same logic levels. The space dimensions are usually constrained by the settings of quantum devices. In this paper, we assume the height for qubit arrangement is given as the number of layers such that qubits can be arranged according to the space dimensions and geometric descriptions can be generated according to the circuit netlist.

Problem 1 (Layout Synthesis for TQEC Circuits): Given a TQEC circuit netlist and configuration of space dimensions, e.g., width and height, and qubit arrangement, we generate geometric descriptions in qubit routing with minimum depth (time volume).

Width can be derived from given height and number of qubits with compact arrangement. Since all single-qubit rotation gates and Toffoli gates in TQEC circuits can be decomposed to CNOT gates and ancillas in the preprocessing stage [6], we assume there are only CNOT gates in the circuits when describing the algorithms for simplicity. Fig. 6 shows the decomposition of Toffoli gate, whose geometry can be found in Fig. 20 of [6]. The decomposition stage has also considered to leave placeholders of distillation circuits for initialization for ancillas, we do not consider the synthesis of distillation boxes in this paper.

III. Layout Synthesis Algorithms

In this section, we explain the framework to generate geometric descriptions, which consists of two phases, i.e., qubit placement and routing. Qubits are placed to grids according to space dimensions, while the geometric descriptions are generated according to the qubit arrangement and circuit netlist.

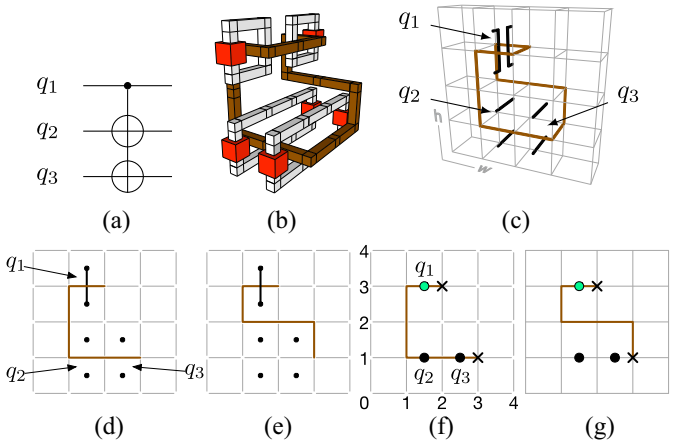


Fig. 7. (a) Example of a CNOT gate with two targets and (b) its implementation with one depth step and qubits placed in 2-D space and (c) corresponding 3-D stick diagram and (d) its front view and (e) its back view. Corresponding stick diagram: (f) front face and (g) back face.

A. Stick Diagram Representations

Before introducing details on qubit placement and routing, we introduce simplified stick diagram representations for routing of a net from the geometric descriptions shown in Figs. 3 and 5. The stick diagram representation has equivalent 3-D and 2-D versions. Since in our qubit architecture logical qubits propagate in pairs of primal defects, we only need to determine the routing of dual defects and the braiding with qubits for construction of CNOT gates. In the first step, a 3-D routing grid system is introduced with two grids in depth axis, shown as Fig. 7(c), where grids are available for dual defects. The primal defects for a qubit appear in the centers of vertically neighboring squares. Both primal and dual defects are simplified from cuboids to lines. For multiple nets at different depth steps, we can cascade multiple grid systems together along the depth axis, as shown in Fig. 9(b).

A 3-D stick diagram of single depth step can be divided into two faces: 1) front face and 2) back face. Fig. 7(d) and (e) give a front view and back view of Fig. 7(c). A 2-D stick diagram is derived from further simplification of each pair of primal qubits in the front and back view to a circle, shown in Fig. 7(f) and (g). Each circle is labeled with its corresponding qubit in the stick diagram for the front face. We mark the qubits of control signals of nets to light green and target signals are marked black. The routing segments are also divided into routing in front face and routing in back face, where the cross marks denote the segments connecting two faces. For routing of a net, we draw both front and back faces with only qubits in the net and selected routing segments. In other words, qubits not in the net are usually not explicitly drawn for brevity. In qubit placement, we only draw the front face of a single grid system to show the positions of all qubits.

B. Qubit Placement

Previous work on TQEC assumes 1-D arrangement of qubits [6], [17]–[20] shown as Fig. 8(a). We try to enable 2-D arrangement of qubits with a multiple-layer architecture, shown as Fig. 8(b) with an example of two layers. We allow

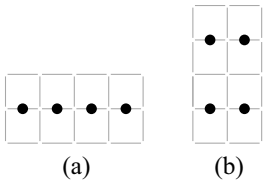


Fig. 8. (a) Example of single-layer qubit arrangement like that in Fig. 5(b) (view from left). (b) Two-layer qubit arrangement like that in Fig. 5(c).

one horizontal track for routing between neighboring layers, like the horizontal gridlines at height 0, 2, and 4 in Fig. 7(f); as a result, qubits have to be placed in odd-height grids (index starts from zero). The architecture is flexible and we can always insert more routing tracks between neighboring layers for more routing resources.

To tackle Problem 1, both qubit placement and routing are important issues. The quality of placement may impact the minimum total depth achieved from routing. In other words, qubit placement can be optimized to reduce total depth after routing. Various previous work for qubit placement focuses on minimizing the SWAP gates in linear nearest neighbor architectures [21]–[27]. Insertion of each SWAP gate is assumed to have a constant overhead to the space-time volumes, so optimizing the SWAP gates helps reduce the resource consumption and total depth. In TQEC, the long-range interaction between two qubits can be achieved by directly routing a CNOT gate properly to the corresponding qubits instead of inserting SWAP gates. However, previous approaches for placement in linear nearest neighbor architectures may be adapted to minimize the objective here, which is left for future work. In this paper, we take the positions of qubits in the grid system from input.

C. Qubit Routing

In this section, we explain two different routing strategies for qubits, i.e., a mixed integer linear programming (MILP) scheme and an approximation approach by generating candidate routing solutions. Since nets of different logic levels are independent in terms of routing resources, the routing process can be conducted at each individual logic level. For convenience, the nets in qubit routing always belong to one logic level if not specially mentioned in this section.

Considering the structure of CNOT gate in Fig. 7, the dual defects actually form a circle that covers all qubits in the net. We denote *front path* as a path in the front face, *back path* as a path in the back face. We also use the word *cover* to indicate a path visits a qubit and occupies the gridline, where the qubit locates. Following [6] one CNOT gate is implemented in one depth step. To simplify the problem, we assume the routing circle of a net consists of a front path, a back path, and two segments connecting the front and back faces. To reduce the solution space, we add an additional constraint for braiding at target qubits that the target qubits are only covered by the front path, as shown in Fig. 7(f).

Problem 2 (Qubit Routing): Given a net set N, where each net has a single control qubit and one or more target qubits, together with a 3-D rectilinear grid system G, where qubits are only located in the center of odd-height horizontal gridlines, generate dual defects for all the nets on G (in other words, route all the nets) with minimum depth step, while subjecting to following constraints.

1) Segments of dual defects run on grids.
2) Dual defects form a circle for each net after visiting each grid at most once, which can be split into a front path and a back path.
3) The circle has to visit the control qubit in both faces; e.g., in Fig. 7(f) and (g), q_1 is covered in both faces.
4) The circle has to cover the grids of the target qubits in the front face, while in the back face, they must not be covered, like q_2 and q_3 in Fig. 7(f) and (g).
5) Circles of different nets must be vertex-disjoint.

Theorem 1: Qubit routing in Problem 2 is NP-hard.

The proof is shown in the Appendix.

1) Route Multiple Nets by MILP: A trivial solution to route nets can be constructed by routing one net at one depth step, with $|N|$ depth steps in total, which is the maximum depth. However, we can merge some nets into one depth step if their routing solutions do not have any conflict (resource sharing). The example in Fig. 5(c) routes two nets in one depth step. Hence, the objective is to minimize the depth required to route all nets, i.e., minimize the latency.

Some notations are explained in Table I. For a graph G, we use $G - v$ to represent the subgraph after G excludes vertex v, i.e., $(V(G)\backslash\{v\}, \{uw|uw \in E(G)$ and $u, w \in V(G)\backslash\{v\}\})$. We use $G - H$ to represent the subgraph after G excludes its subgraph H, i.e., $(V(G)\backslash V(H), \{uw|uw \in E(G)$ and $u, w \in V(G)\backslash V(H)\})$.

We split the formulation into different parts for easier explanation. The objective is to minimize the depth steps D required

$$\min \quad D. \tag{4}$$

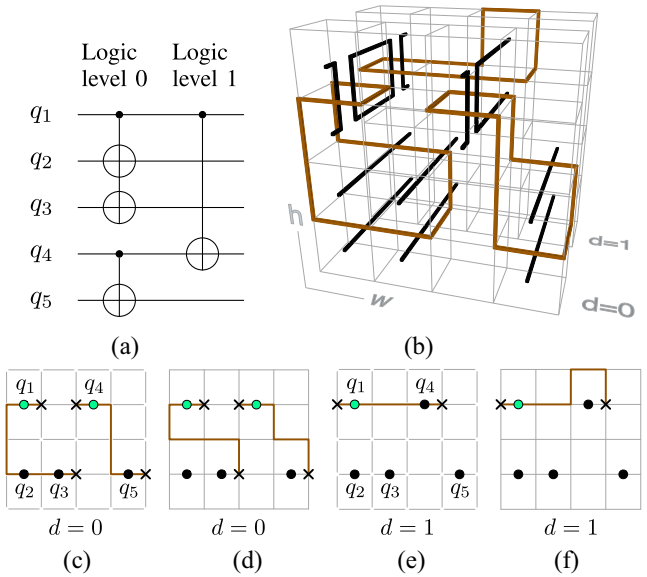


Fig. 9. (a) Example of a CNOT gate with two logic levels and (b) 3-D stick diagram of its corresponding routing solution in two depth steps. Stick diagram at depth step 0: (c) front face and (d) back face. Stick diagram at depth step 1: (e) front face and (f) back face.

TABLE I
NOTATIONS USED IN LAYOUT SYNTHESIS

$G = (V, E)$	3D grid graph (Fig. 9(b)) for one logic level where V is the set of vertices and E is the set of edges.
N	The set of nets to be routed in one logic level.
S	The set of control qubits of nets in N.
T	The set of target qubits of nets in N.
Q	The set of qubit lines.
e_f, e_b	An edge in front face (e_f) and an edge in back face (e_b) with the same coordinates in space dimensions of a grid system.
$E(v)$	Set of edges connected with vertex v. If v is a qubit, it means the set of edges containing the qubit.

Now, we explain the constraints. For a net n, $|N|$ binary variables $b_{n,d}$, are introduced to represent in which depth step it is routed. For brevity in the following discussion, the range of depth d always satisfy $1 \leq d \leq |N|$ if not specially mentioned. The number of depth steps required is equal to the maximum depth step selected among all nets, so we can compute the final depth D with (5a). The constraint in (5b) ensures only one depth step is selected

$$\sum_{d=1}^{|N|} d \cdot b_{n,d} \leq D, \forall n \in N \tag{5a}$$

$$\sum_{d=1}^{|N|} b_{n,d} = 1, \forall n \in N. \tag{5b}$$

We introduce the concept of *degree* of vertex v as the number of selected gridlines connecting to vertex v in the grid system. The binary variable $b_v^{n,d}$ is introduced to represent the degree of vertex v in the grid system at depth step d for net n. The binary variable $b_e^{n,d}$ denotes whether edge e at depth step d is selected by net n in its routing solution. As the routing for a net is a circle, the degree of any vertex on the path is 2, while the degree of any other vertex is 0

$$\sum_{e \in E(v)} b_e^{n,d} = 2b_v^{n,d}, \forall v \in V, n \in N. \tag{6}$$

Although (6) ensures the selected gridlines always form cycles, it fails to guarantee that only a single cycle is formed for one net, since the constraint will also be satisfied for multiple disjoint cycles. Thus, the challenge comes from the exclusion of solutions with multiple disjoint cycles formed for one net. Considering a solution contains multiple disjoint cycles for one net, if we remove any vertex from the solution, the rest selected gridlines still contain at least one cycle. Therefore, if we are able to ensure that after removing an arbitrary vertex, the rest selected gridlines do not form any cycle, but a tree structure instead, it is possible to guarantee a single cycle for one net.

We employ the difference of the maximum average degree between a cycle and a tree to forbid multiple disjoint cycles. The *average degree* of a graph G_n is defined as $ad(G_n) = [2|E(G_n)|/|VG_n|]$, and the *maximum average degree* is defined as

$$\text{mad}(G) = \max_{H \subseteq G_n} \text{ad}(H) \tag{7}$$

where H is any subgraph of G_n. Intuitively, maximum average degree denotes the densest part of the graph and it is no smaller than the average degree of any subgraph of G_n. It is observed that the maximum average degree of a tree is exactly its average degree $[2(|V(G_n)| - 1)/|V(G_n)|]$, while any cycle results in the average degree of a graph no smaller than 2 [33]. If we remove an arbitrary vertex v^* that has to be covered by the circle, then the remained path must not form any cycle.

The maximum average degree is computed by forcing each selected edge to send a flow of 2 to its vertices and each vertex only receives non-negative flow [33]. Specifically in this problem, supposing a graph $G_n \subseteq G$ with edge set $E(G_n)$ and vertex set $V(G_n)$ is selected as the routing solution for net n, its $|E(G_n)|$ edges will send flow of $2|E(G_n)|$ to $|V(G_n)|$ vertices. Note that any vertex $v \in V(G - G_n)$ will not receive any flow since all the edges connect to v send zero flow. Then at least one of the $|V(G_n)|$ vertices receive a flow no smaller than $[2|E(G_n)|/|V(G_n)|]$, which implies that the maximum average degree can be obtained by minimizing the maximum flow received by any vertex. If G_n contains cycles, even the minimum value of the maximum flow received by a vertex is no smaller than 2. We can ensure acyclic nature of G_n by constraining the amount of flow received by any vertex is smaller than 2. In linear programming, it is hard to constrain an equation to be "smaller than ($<$)" a value, because only "smaller than or equal to ($\leq$)" is allowed. The aforementioned maximum average degree of a tree is smaller than or equal to $[2(|V(G_n)| - 1)/|V(G_n)|] \leq 2 - [2/|V(G)|]$ as $|V(G_n)| \leq |V(G)|$. Thus, we can constrain the flow of each vertex to be smaller than or equal to the upper bound as shown in (8b), with two continuous non-negative variables $x_{e,u}^{n,d}$ and $x_{e,v}^{n,d}$ introduced for edge e at depth d of net n, denoting the flows sent to vertices u and v, respectively. Equation (8a) ensures each selected edge sends a flow of 2. Equation (8c) restricts each vertex only receives non-negative flow from edges

$$x_{e,u}^{n,d} + x_{e,v}^{n,d} = 2b_e^{n,d}, \forall e \in E(G - v^*), n \in N \tag{8a}$$

$$\sum_{e \in E(v)} x_{e,v}^{n,d} \leq 2 - \frac{2}{|V(G)|}, \forall v \in V(G - v^*), n \in N \tag{8b}$$

$$x_{e,u}^{n,d}, x_{e,v}^{n,d} \geq 0, \forall e \in E(G - v^*), n \in N. \tag{8c}$$

Any vertex of the edges containing the control qubit of the net can serve as v^* since the path has to visit it.

Free of conflict between any solution of different nets is ensured by (9a). Equation (9b) makes sure the path goes through the control qubit of the net and (9c) guarantees that path braids with target qubits. Qubits not in the net are avoided by

$$\sum_{n \in N} b_v^{n,d} = 1, \forall v \in V \tag{9a}$$

$$b_{e_f}^{n,d} = b_{e_b}^{n,d} = b_{n,d}, \forall e_f, e_b \in E(S(n)), n \in N \tag{9b}$$

$$b_{e_f}^{n,d} = b_{n,d}, b_{e_b}^{n,d} = 0, \forall e_f, e_b \in E(T(n)), n \in N \tag{9c}$$

$$b_e^{n,d} = 0, \forall e \in (E(Q) \backslash E(S(n)) \backslash E(T(n))), n \in N. \tag{9d}$$

To sum up, (4)–(6), (8), and (9) compose the full MILP formulation, where $b_e^{n,d}$, $b_v^{n,d}$, and $b_{n,d}$ are binary variables while

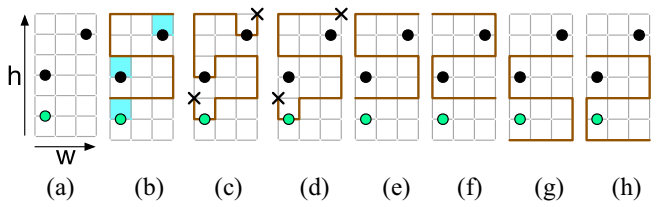


Fig. 10. (a) Example of net, where control qubit is marked by green dots and target qubits are marked by black dots and (b) a guideline for generating routing solution and (c) generated front face and (d) back face. Four possible guidelines for a net are shown in (e)–(h).

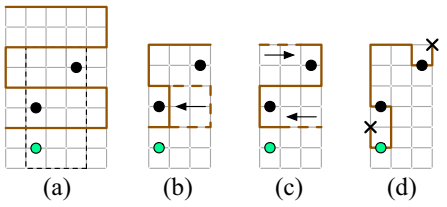


Fig. 11. Example of optimization for guidelines: (a) extract minimum bounding box that covers all qubits in the net (vertically extend by one grid for different candidates), (b) shift vertical segments to remove redundant horizontal segments, and (c) shorten dangling ending segments. (d) Example of generated routing in front face from the guideline in (b).

$x_{e,u}^{n,d}$ and $x_{e,v}^{n,d}$ are continuous. We can see that the MILP formulation is very expensive because its number of binary variables is related to the number of nets as well as the total edges and vertices in the grid system. Supposing that the total numbers of edges and vertices in the system are linear to total amount of qubits, the number of binary variables is in the order of $|N|^2|Q|$ due to the existence of $b_e^{n,d}$ and $b_v^{n,d}$, which is not affordable for large circuits.

2) Candidate Routing Solution Generation for Single Net: Although qubit routing for multiple nets is difficult, it is possible to generate feasible routing solutions with specific patterns for any net due to the unique structure of multiple-layer architecture. Since the qubits only appear at odd-height grids, a zig-zag line can be generated by covering all the even height grids such that all qubits in the net is one grid away from the line, as shown in Fig. 10(a) and (b). The zig-zag line is the guideline for generating a feasible solution.

The process of generating the routing in front face is illustrated in Fig. 10(b) and (c), which can be summarized as follows.

1) Given a guideline, mark all the grids covered by the guideline to 1 (selected) and others to 0 (not selected).
2) Detect the positions of qubits in the net and mark the squares above qubits, shown as blue squares in Fig. 10(b).
3) For each square, flip the selection of its four grids and it yields the routing for front face, shown as Fig. 10(c).

The routing in the back face can be generated in a similar way by only flipping the square of the control qubit in the net. Two ending points of the guideline, which are shown as cross marks in Fig. 10(c) and (d), connect the front and back face to form the closed path.

According to different directions, there are four different guidelines for a net, shown as Fig. 10(e)–(h), which can generate four different routing solutions with the procedure (slight variation) mentioned above. However, the guidelines may not be compact enough to generate compact routing solutions. The example in Fig. 10(c) occupies a 3×5 grid box with many redundant segments. The compactness of a routing solution is determined by that of the provided guideline. Therefore, it is necessary to compress the guideline for generating compact routing solutions. Here, we propose several ways to optimize a guideline.

1) Generating the guidelines within the minimum grid box that covers all the qubits results in more compact solutions, shown as the dashed box in Fig. 11(a).
2) Each vertical segment can be shifted by checking the positions of qubits below and above it to remove unnecessary horizontal segments in the middle, shown as Fig. 11(b).
3) In the top and bottom of each guideline, it is possible to shorten dangling segments by checking the positions of qubits below or above it, shown as Fig. 11(c).

These optimization techniques only need to locally check the positions of qubits. Fig. 11(d) shows an example for the routing in front face generated from the optimized guideline, which is more compact that in Fig. 10(c).

3) Candidate Routing Solution Assignment: With four candidate solutions for each net from Section III-C2, we need to select one solution for each net such that all the nets are routed in minimum total depth.

Problem 3 (Routing Solution Assignment): Given a net set N and four candidate routing solutions for each net in a logic level, assign one candidate solution and depth step to each net such that no routing solutions share the same resources (no conflict) and the total depth in this logic level is minimized.

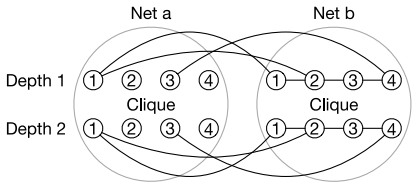


Fig. 12. Example of conflict graph for candidate routing solution assignment. The vertices within each gray circle form a clique and the connections are not drawn for brevity.

Fig. 12 gives an example of assignment problem of two nets, which needs at most two depth steps. Since four candidate solutions are generated for each net in Section III-C2, each net corresponds to $4 \times 2 = 8$ numbered vertices in the graph, where number within each vertex denotes the candidate solution. All the vertices for a net is encompassed by a gray circle and its subgraph forms a clique because only one of them can be selected. The first row of vertices denote the net is routed in depth 1, while the second row of vertices denote the net is routed in depth 2. An edge is inserted between two vertices when the corresponding routing solution cannot be both selected due to conflicts. Vertices of different depth between different nets are not connected since they do not cause any conflict. We need to select one vertex for each

TABLE II
COMPARISON OF DIFFERENT CONFIGURATIONS AND ALGORITHMS

Design				1D Architecture - Single layer				2D Architecture - 4 layers			
				MILP		CRSA		MILP		CRSA	
Name	$\|Q\|$	$\|N\|$	MT	D	T(s)	D	T(s)	D	T(s)	D	T(s)
b1	10	10	5	10	0.66	10	0.00	9	0.37	10	0.00
b2	10	97	2	NA	NA	80	0.01	NA	NA	79	0.02
b3	100	235	2	NA	NA	88	0.26	NA	NA	84	0.18
b4	100	233	5	NA	NA	119	0.22	NA	NA	131	0.17
b5	100	97	10	NA	NA	66	0.09	NA	NA	79	0.07
b6	100	1051	2	NA	NA	403	1.27	NA	NA	345	0.88
b7	100	1022	5	NA	NA	557	0.97	NA	NA	651	0.75
b8	100	909	10	NA	NA	646	0.90	NA	NA	786	0.81
b9	1000	2954	2	NA	NA	1079	186.24	NA	NA	799	88.40
b10	1000	1989	5	NA	NA	889	53.16	NA	NA	907	22.82
b11	1000	1070	10	NA	NA	535	16.19	NA	NA	740	9.21
b12	1000	10066	2	NA	NA	3656	555.38	NA	NA	2754	268.14
b13	1000	9894	5	NA	NA	4403	251.03	NA	NA	4564	117.33
b14	1000	10411	10	NA	NA	5189	180.11	NA	NA	7184	103.94
avg.		2860	-	-	-	1266	88.99	-	-	1365	43.77
ratio		2.26	-	-	-	1.00	1.00	-	-	1.08	0.49

net without any conflict while minimize the maximum depth of selected vertices.

With the graph model in Fig. 12, Problem 3 can be formulated into a variation of maximum weighted independent set (MWIS) problem, where each vertex is weighted by the negative value of its depth. Any independent set is able to derive a legal routing solution for nets. Unlike classic MWIS problem that maximizes the total weight, the objective here is to find an independent set to maximize the minimum weight of vertices in the set.

While MWIS problem is well-known NP-hard, we solve it with a fast heuristic approach that tries to assign routing solution to the net with maximum degree in the conflict graph first in each iteration. The details are shown in Algorithm 1. We first construct a net conflict graph in line 3, where vertices correspond to nets and two nets are connected if any of their candidate solutions have conflicts. At the beginning of each iteration, we search for the net with maximum degree that has not been processed yet. All candidate solutions of the net are traversed to compute costs by checking the conflicts with candidate solutions of other nets in lines 6–14. Any candidate solution that has conflict with processed nets is assigned to infinity cost. Otherwise, the cost function in line 12 ensures that candidate solution with smaller depth and fewer conflicts with unprocessed nets have higher priority to be selected. Then the solution with minimum cost is assigned to the net in line 15. In the worst case, the algorithm will route each net at each individual depth step resulting in an overall depth of $|N|$.

Algorithm 1 Candidate Qubit Routing Solution Assignment

Require: A set of nets N and candidate routing solutions.
Ensure: Assign one candidate solution to each net with minimized maximum depth.
1: Define M as a large number;
2: Define c as a candidate solution;
3: Construct net conflict graph;
4: **while** there are unprocessed nets **do**
5: Find unprocessed net n with maximum degree;
6: **for** each candidate solution c of net n **do**
7: **if** c has conflict with any processed net **then**
8: $c.cost \leftarrow \infty$;
9: **else**
10: $n_c \leftarrow$ number of conflicts with candidate
11: solutions of other unprocessed nets;
12: $c.cost \leftarrow c.depth \cdot M + n_c$;
13: **end if**
14: **end for**
15: Assign candidate solution with minimum cost to net n;
16: **end while**

IV. EXPERIMENTAL RESULTS

Our algorithm was implemented in C++ and tested on an eight-core 3.40 GHz Linux server with 32 GB RAM. We experiment on two sets of benchmarks, RevLib [34] and synthetic benchmarks. The Toffoli gates in RevLib benchmarks are decomposed to single-target CNOT gates and ancillas by [6]. In Tables II and III, the number of qubits is denoted by "$|Q|$," the number of nets (CNOT gates) is denoted by "$|N|$," and the maximum number of pins (including control and targets) for multiple-target CNOT gates is shown as "MT." The RevLib benchmarks contain circuits with $|Q|$ from 131 to 3753 and $|N|$ from 168 to 4938 and only single-target CNOT gates are used. Qubits are placed according to the order of input netlists. The synthetic benchmarks are generated with various number of qubits $|Q|$ from 10 to 1000, number of nets $|N|$ from 10 to 10 411, and multiple-target CNOT gates with MT from 2 to 10, shown in the first three columns Table II. Given number of qubits, number of CNOT gates, and maximum number of pins for multiple-target CNOT gates as input, we randomly select the pins for a CNOT gate every time until all the qubits are covered by at least one CNOT gate. Gurobi [35] is used as the ILP solver.

The framework supports arbitrary configuration of layers for the TQEC system. Tables II and III show the results of final depth ("D") and runtime ("T") for a single layer and

TABLE III
COMPARISON OF DIFFERENT CONFIGURATIONS AND ALGORITHMS ON REVLIB BENCHMARKS [34]

Design				1D Architecture - Single layer				2D Architecture - 4 layers			
				MILP		CRSA		MILP		CRSA	
Name	$\|Q\|$	$\|N\|$	MT	D	T(s)	D	T(s)	D	T(s)	D	T(s)
4gt4-v0_73	257	341	2	NA	NA	222	0.52	NA	NA	222	0.47
4gt10-v1_81	131	168	2	NA	NA	115	0.15	NA	NA	115	0.12
rd84_142	897	1162	2	NA	NA	475	5.75	NA	NA	475	5.00
add16_174	1394	1792	2	NA	NA	572	19.01	NA	NA	577	16.43
cycle17_3_112	1911	2478	2	NA	NA	1620	33.02	NA	NA	1620	27.83
hwb5_53	1307	1729	2	NA	NA	1132	13.85	NA	NA	1132	12.09
sym6_145	1519	1980	2	NA	NA	1137	17.93	NA	NA	1137	15.65
ham15_107	3753	4938	2	NA	NA	3005	138.30	NA	NA	3005	103.76
avg.		1824	-	-	-	1035	28.56	-	-	1035	22.67
ratio		1.76	-	-	-	1.00	1.00	-	-	1.00	0.79

four layers. The MILP-based qubit routing in Section III-C1 is shown as "MILP" and the algorithm based on candidate routing solution assignment in Section III-C3 is denoted as "CRSA." We set the maximum runtime of MILP to 10 000 s and "NA" indicates that the program fails to finish within given time.

In Table II, MILP only returns solutions for some small benchmarks with only ten qubits, while the runtime is not scalable enough to solve all benchmarks. Among those benchmarks, CRSA gives a similar depth in much more reasonable time. Without depth optimization, each net uses one depth step and thus the overall depth is the same as the number of nets $|N|$, which we use as the baseline for comparison of overall depth. From the column $|N|$ and column D under CRSA, the baseline ends up to be 2.26 times of the overall depth for 1-D architecture and 2.09 times of that for 2-D architecture with four layers. Table III shows the results for RevLib benchmarks. CRSA achieves 1.76 times smaller depth for both 1-D and 2-D architectures with efficient generation of layout. It can also be observed that small benchmarks with fewer than 100 qubits generally have limited benefits from depth optimization, while for the rest, more than 50% reduction of overall depth from the number of nets $|N|$ for synthetic benchmarks and 30% reduction for RevLib benchmarks are possible. As aforementioned, the depth measures the time volume of TQEC circuits and smaller depth contributes to smaller latency to the system. The results demonstrate that the proposed approach is able to achieve significantly smaller latency without introducing additional resources like space volumes.

We also observe that the 2-D CRSA has on average smaller runtime than 1-D CRSA. The major difference in runtime comes from the conflict graph construction of Algorithm 1. The reason lies in the grid-based check for conflicts of any pair of candidate routing solutions, where early exit is possible once a conflict is detected. For example, in benchmark b14, the conflict graph for 2-D architecture ends up with more entries than 1-D in the experiment, which means early exit happens more often in 2-D than that in 1-D. In addition, 2-D arrangement uses fewer grids than 1-D with similar overall area of grids, e.g., 30% fewer grid vertices and 20% fewer gridlines in this benchmark, which is also a reason for smaller runtime.

V. CONCLUSION

We propose a framework on multiple-layer layout synthesis of TQEC circuits by enabling 1-D and 2-D arrangement of qubits. We prove the NP-hardness of the qubit routing problem and propose an efficient algorithm to optimize space-time volumes. For TQEC circuits further exploration of qubit placement, automatic exploration of best number of layers for various designs, and automatic geometric simplification are included in the future work. It is also valuable to explore the opportunity of layout optimization for reliability issues of TQEC circuits, such as error chain and qubit defects. With the rapid advancement in quantum computing, there are lots of emerging layout challenges. Error correction scheme like lattice surgery [36] removes the need of braiding by splitting and merging planar code surfaces; architecture like Multi-SIMD [37] scheme creates a circuit with distributed regions connected by teleportation networks, while each region only consists of small number of qubits. These new schemes raise different constraints to the layout, remaining to be explored. The layout optimization aims at creating efficient and reliable implementation of quantum circuits for high performance computing.

APPENDIX
PROOF OF THEOREM 1

To find the minimum depth step in Problem 2, we need to answer the decision problem: given an integer $d(1 \leq d \leq |N|)$, can the nets in Problem 2 be routed with total depth d. While the answer to $d = |N|$ is always true, it is nontrivial to solve for the special case of $d = 1$, i.e., in one depth step. It turns out the decision problem for single depth step is already difficult even for 2-pin nets only. We define the decision problem of qubit routing in single depth step for 2-pin nets as follows.

Problem 4 (Single-Depth Qubit Routing Decision): Given a net set N with 2-pin nets only (one control qubit as source and one target qubit as sink for each net) and one depth step of 3-D grid system G, where qubits are only located in the centers of odd-height horizontal gridlines, can the nets be routed such that:

1) routing segments run along gridlines only;
2) routing for each net forms a circle that consists of a front path and a back path;

3) the front path covers both source and sink;
4) the back path covers only the source, not the sink;
5) the routing circles of different nets must be vertex-disjoint.

Lemma 1: If the single-depth qubit routing decision problem (Problem 4) is NP-complete, then the qubit routing problem (Problem 2) is NP-hard.

Proof: Suppose that Problem 2 is polynomially solvable. For any instance of Problem 4, we can construct an instance of Problem 2. If the minimum depth to Problem 2 is larger than 1, the answer to the decision problem is no; otherwise, it is yes. It follows that Problem 4 can be solved in polynomial time, which is a contradiction to the assumption. Therefore, Problem 2 is at least as hard as NP-complete. Considering that Problem 2 is not in the set of NP (not a decision problem), it is NP-hard. ■

The NP-completeness of similar routing problem has been proved for 2-pin nets by reduction from 3-satisfiability (3-SAT) problem [38]. Our problem on single-depth qubit routing decision has following major differences from previous problem.

1) The pins are located in the centers of odd-height horizontal gridlines rather than arbitrary vertices on grids.
2) To cover a pin, a path has to occupy the full gridline which contains the pin.
3) The routing of a net has to be a circle rather than a path.

Due to the differences, it is difficult to apply previous conclusion to our problem, while we can still follow the procedure of the previous proof by reduction from 3-SAT.

Previous work first proves the NP-completeness of a variation of routing problem with obstacles and then derive the conclusion for the original routing problem [38]. We follow this procedure as well. Let an obstacle be a rectangular domain on grids, as shown in Fig. 19(a). For technical reasons, we first prove the NP-completeness of an obstacle routing, which is a variation of Problem 4 with obstacles that cannot be used for routing. Then we derive that the original obstacle-free routing in Problem 4 is also NP-complete.

Problem 5 (Single-Depth Qubit Routing Decision With Obstacles): Given a net set N with 2-pin nets only (one source and one sink) and M obstacles on a 3-D grid system G with one depth step, where qubits are only located in the centers of odd-height horizontal gridlines, can the nets be routed such that:

1) routing segments run along gridlines only;
2) routing for each net forms a circle that consists of a front path and a back path;
3) the front path covers both source and sink;
4) the back path covers only the source, not the sink;
5) the routing circles of different nets must be vertex-disjoint.

Lemma 2: The single-depth qubit routing decision with obstacles problem (Problem 5) is NP-complete.

Proof: The proof is conducted by polynomial transformation from a 3-SAT instance to an obstacle routing instance. Given a 3-SAT instance I, let $\{x_1, x_2, \ldots, x_n\}$ denote n variables, $\{x_1, \bar{x_1}, \ldots, x_n, \bar{x_n}\}$ denote $2n$ literals, respectively. Let set $C = \{c_1, c_2, \ldots, c_m\}$ denote m clauses with 3 literals per clause. We can reduce I to an obstacle routing instance I' with $\mathcal{O}(nm)$ nets on a rectangular grid of area $\mathcal{O}(nm)$.

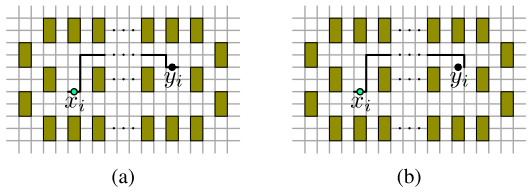


Fig. 13. Gadget to represent truth or false assignment of a variable. Example of (a) front path and (b) back path. The green rectangles denote obstacles. The source pin is marked light green and sink pin is marked black. The front path and back path are connected through their ending points.

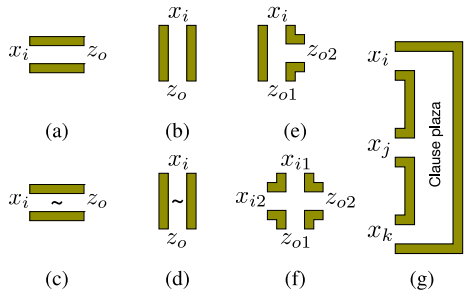


Fig. 14. Some symbols of gadgets. (a) Horizontal and (b) vertical pipe ($z_o = x_i$). (c) Horizontal and (d) vertical inverter ($z_o = \bar{z_i}$). (e) Junction ($z_{o1} = z_{o2} = x_i$). (f) Crossover ($z_{o1} = x_{i1}, z_{o2} = x_{i2}$). (g) Clause plaza, where there is feasible routing solutions iff the clause has truth assignment.

To construct instance I', it is necessary to have a gadget to represent truth and false assignment of a variable, shown as Fig. 13. The green rectangles denote obstacles and x_i and y_i denote the source and sink in a net. There are two available horizontal channels for routing, bottom and top. If the front path of the routing circle from x_i to y_i goes through top channel like the solid line in Fig. 13(a), we identify this case as false assignment, i.e., the corresponding variable x_i in 3-SAT is assigned to 0; if the front path of the routing circle goes through the bottom channel, it is regarded as truth assignment. This gadget can be implemented in vertical direction by utilizing vertical channels as well.

Note that although the back path of the routing circle also has the option to go through the top or bottom channel, shown as Fig. 13(b), it does not influence the Boolean assignment. In other words, only the front path determines the Boolean assignment. In the construction, we only show the front paths to represent routing solutions for brevity. We will explain later on how to derive the back paths from the front paths.

We define several gadgets to help construct the routing instance I'. A *pipe* propagates the assignment, whose symbols are shown as Fig. 14(a) and (b), where the output z_o is equal to the input x_i. An *inverter* flips the assignment, whose symbols are shown as Fig. 14(c) and (d), where the output z_o is equal to $\bar{x_i}$. A *junction* takes one input and copies to its two output ports, whose symbol is shown as Fig. 14(e), where both outputs z_{o1} and z_{o2} are equal to x_i. A *crossover* copies its left input assignment to right and top input assignment to bottom, whose symbol is shown as Fig. 14(f), where $z_{o1} = x_{i1}$ and $z_{o2} = x_{i2}$. A *clause plaza* takes three literal assignments as input and exists a feasible routing solution if only if any of the literals has truth assignment; i.e., the clause has truth assignment. In addition, due to the alternate grid height for pins

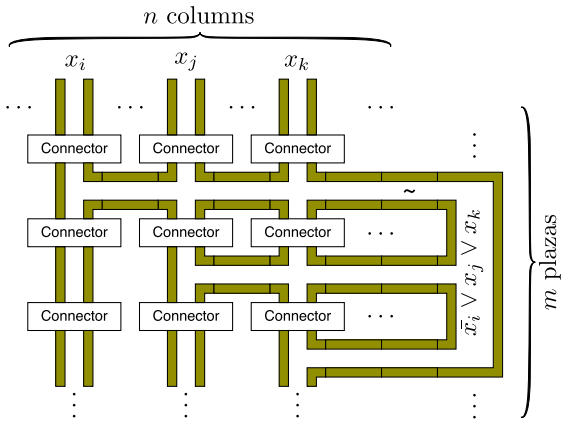


Fig. 15. Outline of the routing problem composed with gadgets.

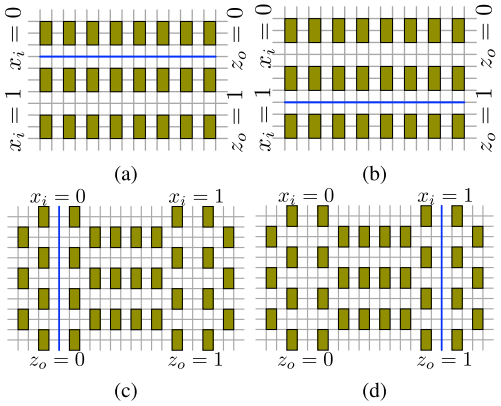


Fig. 16. Pipes with two cases of input: horizontal pipe (a) $x_i = 0$, (b) $x_i = 1$ and vertical pipe (c) $x_i = 0$, (d) $x_i = 1$.

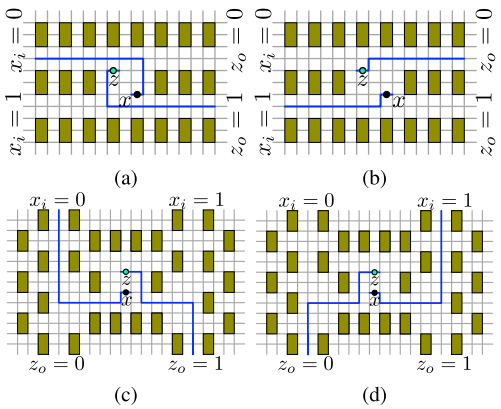


Fig. 17. Inverters with two cases of input: horizontal inverter (a) $x_i = 0$, (b) $x_i = 1$ and vertical inverter (c) $x_i = 0$, (d) $x_i = 1$. Sources are in light green and sinks are in black, same for other figures in this section.

(sources and sinks) in the problem, *connectors* are introduced to connect various gadgets vertically so that the assignments to variables can propagate from top to bottom.

Fig. 15 gives an outline of the routing instance I' consisting of gadgets, where n columns are introduced for n variables and m clause plazas are introduced for m clauses. We show how to construct one clause with gadgets. The assignments to variables propagate along columns and rows with pipes, junctions, and crossovers. We can insert an inverter to create

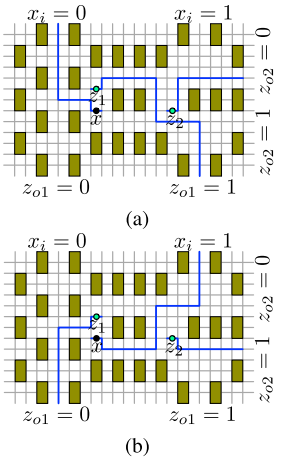


Fig. 18. Junction with two cases of input: (a) $x_i = 0$ and (b) $x_i = 1$. Additional inverter is required for output z_{o1} at the bottom.

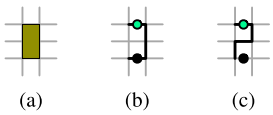


Fig. 19. (a) Representation of an obstacle. (b) Corresponding net with front path and (c) back path. The front path and back path are connected through their ending points.

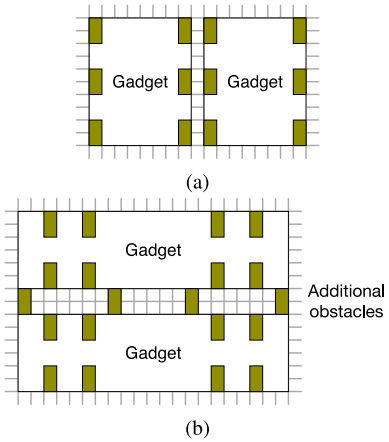


Fig. 20. (a) Horizontal and (b) vertical connection of gadgets. four additional obstacles are required for vertical connection.

a literal for inversion of a variable (e.g., $\bar{x}_i$) shown as the inverter next to the clause plaza in the figure. For each clause, we need a clause plaza to take three corresponding literals as input, shown as the right side of the figure, where the clause $\bar{x}_i \vee x_j \vee x_k$ is mapped. If only if any of $\bar{x}_i, x_j, x_k$ is equal to 1, the plaza has a feasible routing solution; otherwise, it cannot be routed.

We will explain the implementation of gadgets later. There are n columns and 3 m rows (one clause requires three rows) for this construction. Thus, the amounts of gadgets are in an order of $\mathcal{O}(nm)$. As each gadget will be implemented with constant number of nets and grids, the full instance I' requires $\mathcal{O}(nm)$ nets and $\mathcal{O}(nm)$ grids, which is polynomial in problem size.

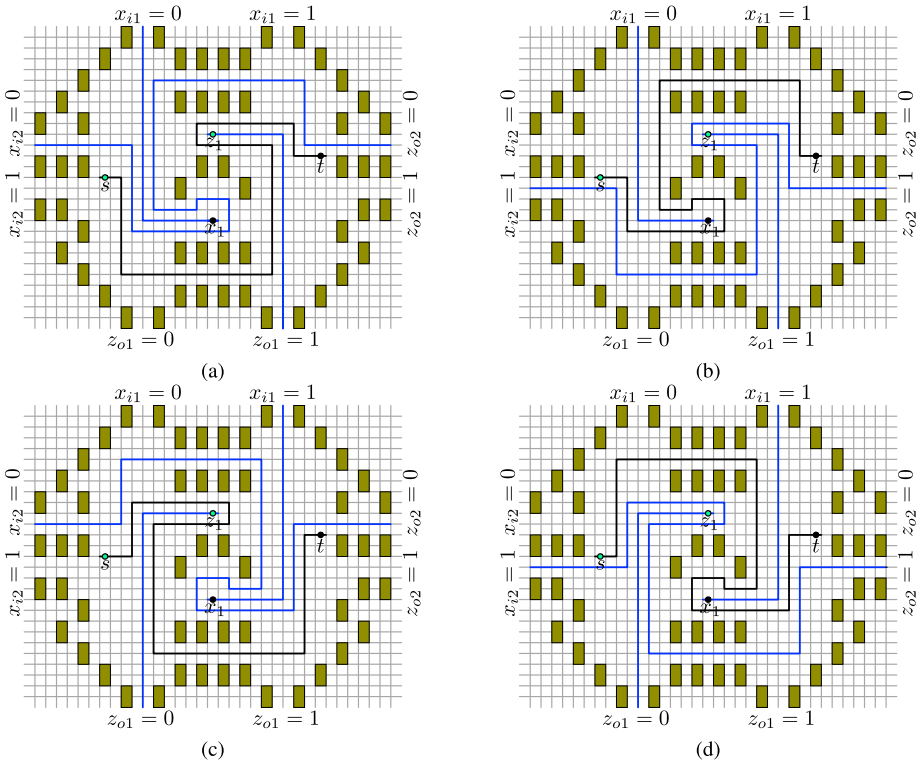


Fig. 21. Crossover with four cases of input: $(x_{i1}, x_{i2}) =$ (a) 00, (b) 01, (c) 10, and (d) 11. Additional inverter is required for output z_{o1} at the bottom.

Given the outline of the construction, we now explain the implementation of each gadget in details. The implementation of gadgets only involve the obstacles shown in Fig. 19(a) which will help transform back to obstacle-free routing later.

Pipe: Fig. 16 shows the implementation of horizontal and vertical pipes. For each implementation, we enumerate all the combinations of input patterns to verify its correctness. Since a pipe only propagates the assignment, the routing path always exits from the same channel as that of the input. It needs to mention that the vertical pipes are designed with special ports at top and bottom for alignment with other gadgets vertically, which will be discussed together with connectors.

Inverter: Fig. 17 shows the implementation of horizontal and vertical inverters. We take Fig. 17(a) as an example, where the input $x_i = 0$. Two additional pins x and z are introduced. The input x_i must connect to sink x and source z must connect to output z_o. When $x_i = 0$, due to the existence of obstacles, the routing path enters the gadget from the top channel and block the way after connecting to pin x for pin z to exit from the top channel which represents $z_o = 0$. As a consequence, the routing for pin z can only go through the bottom channel to z_o which represents $z_o = 1$. The entire gadget behaves like an inverter as it switches the routing channel. Other cases and vertical inverters can be verified in a similar way.

Junction: Fig. 18 shows the implementation of a junction, where three pins are introduced with three nets. Pin x has to connect to input x_i, pin z_1 has to connect to output z_{o1} and pin z_2 has to connect to output z_{o2}. Again we take the case of $x_i = 0$ shown in Fig. 18(a) as an example. The routing between input x_i and pin x separates pin z_1 from the channel of $z_{o1} = 0$, so the routing path of pin z_1 has to exits from the bottom right port, indicating $z_{o1} = 1$. At the same time, the routing path of z_2 is forced to go through the top right port as $z_{o2} = 0$. The implementation in the figure will flip x_i at z_{o1}, so an inverter is required at the bottom to ensure the functionality of $z_{o1} = z_{o2} = x_i$. The inverter is not drawn for brevity.

Crossover: Fig. 21 shows the implementation of a crossover, where four pins are introduced with four nets. Pin x_1 must connect to input x_{i1}, pin s must connect to pin t, pin z_1 must connect to output z_{o1}, and input x_{i2} must connect to output z_{o2}. We mark the routing path between pin s and t with different color because it does not associate with any input and output. We enumerate all the four possible input patterns to verify the functionality of the gadget. Take the case $(x_{i1}, x_{i2}) = 00$ as an example in Fig. 21(a). The routing paths of input x_{i2} to output z_{o2} and pin s to t have to take the only two horizontal gridlines below pin x_1; otherwise, if they go anywhere above pin x_1, it is not possible to finish the connection between input x_{i1} and x_1. Input x_{i1} has to access pin x_1 from the left of the obstacle in the middle; otherwise, routing between pin z_1 and output z_{o1} cannot finish. After careful analysis, we are able to derive that the routing paths have to exit from $z_{o1} = 1$ and $z_{o2} = 0$. Other cases can be analyzed in the same way. One additional inverter is required for z_{o1} at the bottom of the gadget.

Clause Plaza: Fig. 22 show the implementation of a clause plaza, where seven pins are introduced with five nets. A clause plaza only has three inputs without any output. These nets include input x_i to pin y_i, input x_j to pin y_j, input x_k to pin y_k, pin s to t, and pin u to v. In this gadget, we need to ensure

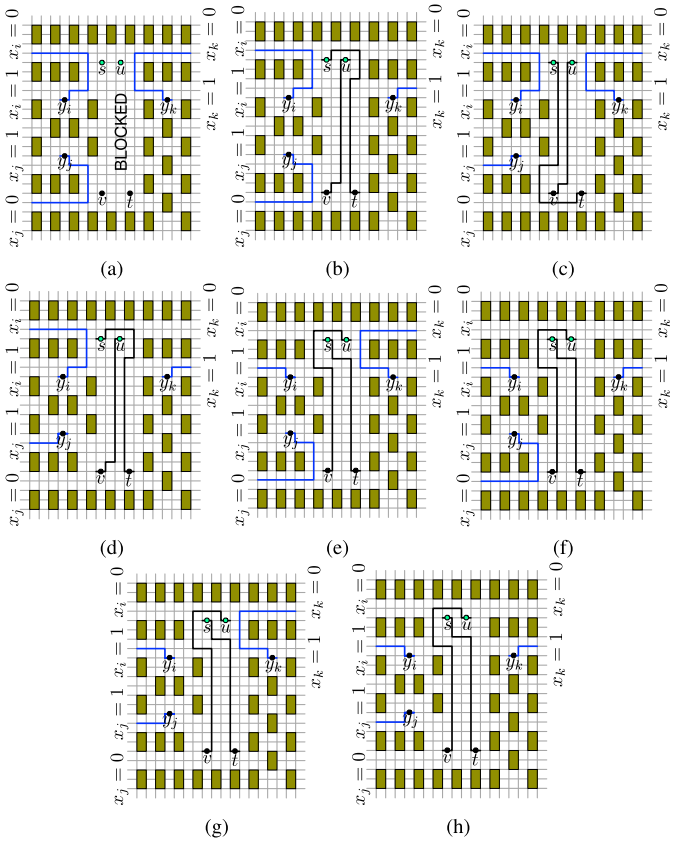


Fig. 22. Clause plaza with eight cases of input: $(x_i, x_j, x_k) =$ (a) 000, (b) 001, (c) 010, (d) 011, (e) 100, (f) 101, (g) 110, and (h) 111. Additional inverter is required for input x_j at the bottom left. Additional inverter and pipes are needed to lead x_k from left side of the gadget to right. Sources are in light green and sinks are in black.

infeasibility when $x_i = x_j = x_k = 0$, shown as Fig. 22(a). In this case, due to limited number of available vertical channels in the middle, it is not possible for pin s, t, u, v to finish connection without using neighboring vertical gridlines which are taken for routing of other nets. As a result, no feasible routing solution can be found, which is used to indicate the false assignment of the clause. For any other case with at least one literal assigned to truth, it is not difficult to find a feasible routing solution. Although the input ports for x_k appear on the right of the clause plaza, we can redirect them to left with pipes and inverters.

Connector: Connectors are introduced to connect gadgets vertically. For horizontal connection of gadgets, we need to align the ports of two gadgets with one grid gap as shown in Fig. 20(a), while for vertical connection, vertical ports need to align and a connector is inserted between two gadgets, shown as four additional obstacles in the middle of Fig. 20(b).

Although the back paths are not shown in the figures, we can derive them from front paths for any gadget. In the figures for gadgets, we design the routing of front paths in a way that the back paths can be found as follows. The back path can follow the routing of the front path except that at the sink of each net, where the back path needs to avoid the sink and connect to the ending point of the front path. Take Fig. 13 as an example. The only difference between the front path and back path lies in the segments near the sink y_i. The back paths of gadgets can be derived in the same way such that the entire routing solution for each net in Figs. 16–18, 21, and 22 can be filled.

With the construction in polynomial time and amount of resources, we conclude that the obstacle routing instance I' is a consistent image of the 3-SAT instance I. The clauses C are simultaneously satisfied if only if a feasible routing solution for all clause plazas exists. Thus, 3-SAT polynomially transforms to obstacle routing, which finishes the proof for obstacle routing from the NP-completeness of 3-SAT. ■

With the proof of obstacle routing in Problem 5, we still need to prove the obstacle-free routing in Problem 4. We will show that an obstacle routing instance can be transformed to an obstacle-free routing instance in polynomial time and vice versa.

Lemma 3: The single-depth qubit routing decision with obstacles problem (Problem 5) polynomially transforms to the single-depth qubit routing decision problem (Problem 4).

Proof: Given an instance I of obstacle routing, we construct an equivalent instance I' of the routing in Problem 4. While all the nets and pins remain the same, the obstacles are replaced with local nets, shown in Fig. 19. Note that in the transformation from 3-SAT to obstacle routing, we only use obstacles in Fig. 19(a). Due to the implementation of obstacles, they cannot be placed in arbitrary positions, but we have already considered that in the transformation from 3-SAT to obstacle routing. By replacing these obstacles with local nets, we can construct an instance I' of Problem 4 in polynomial time.

Any solution to I can translate to the solution of I' in polynomial time by replacing the obstacles with local pairs of pins shown in Fig. 19. Conversely, consider any solution to I'. We assume direct connection of the locally adjacent pairs of pins. If they do not connect in this way, we can adapt the routing to this way, because it results in the minimum regions that these pins block out and leave the remaining area free for other nets. It means that these adjacent pairs of pins behave like obstacles in instance I. Then the solution to I' translates back into a solution of I in polynomial time. ■

Lemma 4: The single-depth qubit routing decision problem (Problem 4) is NP-complete.

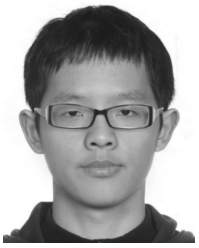
Proof: Proof followed by combining Lemmas 2 and 3. ■

With Lemmas 1 and 4, we conclude the NP-hardness for qubit routing in Theorem 1.

REFERENCES

[1] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. STOC*, Philadelphia, PA, USA, 1996, pp. 212–219.
[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. FOCS*, Santa Fe, NM, USA, 1994, pp. 124–134.
[3] *The IBM Quantum Experience*, Int. Bus. Mach. Corporat., Armonk, NY, USA, 2016. [Online]. Available: http://researchweb.watson.ibm.com/quantum/
[4] S. J. Devitt, "Performing quantum computing experiments in the cloud," *Phys. Rev. A*, vol. 94, no. 3, 2016, Art. no. 032329.
[5] A. G. Fowler and K. Goyal, "Topological cluster state quantum computing," *Quantum Inf. Comput.*, vol. 9, nos. 9–10, pp. 721–738, 2009.
[6] A. Paler, S. J. Devitt, and A. G. Fowler, "Synthesis of arbitrary quantum circuits to topological assembly," *Sci. Rep.*, vol. 6, Aug. 2016, Art. no. 30600.

[7] I. Polian and A. G. Fowler, "Design automation challenges for scalable quantum architectures," in *Proc. DAC*, San Francisco, CA, USA, 2015, pp. 1–6.
[8] R. Raussendorf, J. Harrington, and K. Goyal, "Topological fault-tolerance in cluster state quantum computation," *New J. Phys.*, vol. 9, no. 6, p. 199, 2007.
[9] S. Beauregard, "Circuit for Shor's algorithm using 2n+3 qubits," *Quantum Inf. Comput.*, vol. 3, no. 2, pp. 175–185, 2003.
[10] D. Maslov, "Linear depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures," *Phys. Rev. A*, vol. 76, no. 5, 2007, Art. no. 052310.
[11] A. Broadbent and E. Kashefi, "Parallelizing quantum circuits," *Theor. Comput. Sci.*, vol. 410, no. 26, pp. 2489–2510, 2009.
[12] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818–830, Jun. 2013.
[13] N. Abdessaied, R. Wille, M. Soeken, and R. Drechsler, "Reducing the depth of quantum circuits using additional circuit lines," in *Proc. Int. Conf. Reversible Comput.*, Kolkata, India, 2013, pp. 221–233.
[14] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Inf. Comput.*, vol. 6, no. 4, pp. 351–369, 2006.
[15] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 355–377, 2011.
[16] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum nearest-neighbor algorithms for machine learning," *Quantum Inf. Comput.*, vol. 15, pp. 0318–0358, Mar. 2015.
[17] A. Paler, S. Devitt, K. Nemoto, and I. Polian, "Synthesis of topological quantum circuits," in *Proc. NANOARCH*, Amsterdam, The Netherlands, 2012, pp. 181–187.
[18] S. Yamashita, "An optimization problem for topological quantum computation," in *Proc. ATS*, Niigata, Japan, 2012, pp. 61–66.
[19] A. G. Fowler and S. J. Devitt, "A bridge to lower overhead quantum computation," *arXiv preprint arXiv:1209.0510*, 2012.
[20] A. Paler, S. J. Devitt, K. Nemoto, and I. Polian, "Mapping of topological quantum circuits to physical hardware," *Sci. Rep.*, vol. 4, Apr. 2014, Art. no. 4657.
[21] A. Chakrabarti, S. Sur-Kolay, and A. Chaudhury, "Linear nearest neighbor synthesis of reversible circuits by graph partitioning," *arXiv preprint arXiv:1112.0564*, 2011.
[22] A. Shafaei, M. Saeedi, and M. Pedram, "Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures," in *Proc. DAC*, Austin, TX, USA, 2013, pp. 1–6.
[23] R. Wille, A. Lye, and R. Drechsler, "Optimal SWAP gate insertion for nearest neighbor quantum circuits," in *Proc. ASPDAC*, Singapore, 2014, pp. 489–494.
[24] A. Shafaei, M. Saeedi, and M. Pedram, "Qubit placement to minimize communication overhead in 2-D quantum architectures," in *Proc. ASPDAC*, Singapore, 2014, pp. 495–500.
[25] A. Lye, R. Wille, and R. Drechsler, "Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits," in *Proc. ASPDAC*, Tokyo, Japan, 2015, pp. 178–183.
[26] R. Wille *et al.*, "Look-ahead schemes for nearest neighbor optimization of 1-D and 2D quantum circuits," in *Proc. ASPDAC*, 2016, pp. 292–297.
[27] M. Pedram and A. Shafaei, "Layout optimization for quantum circuits with linear nearest neighbor architectures," *IEEE Circuits Syst. Mag.*, vol. 16, no. 2, pp. 62–74, May 2016.
[28] C.-C. Lin, S. Sur-Kolay, and N. K. Jha, "PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1221–1234, Jul. 2015.
[29] N. Mohammadzadeh, "Physical design of quantum circuits in ion trap technology—A survey," *Microelectron. J.*, vol. 55, pp. 116–133, Sep. 2016.
[30] R. Wille, B. Li, U. Schlichtmann, and R. Drechsler, "From biochips to quantum circuits: Computer-aided design for emerging technologies," in *Proc. ICCAD*, Austin, TX, USA, 2016, Art. no. 132.
[31] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, no. 3, 2012, Art. no. 032324.
[32] M. A. Nielsen, I. Chuang, and L. K. Grover, "Quantum computation and quantum information," *Amer. J. Phys.*, vol. 70, no. 5, pp. 558–559, 2002.
[33] N. Cohen, "Several graph problems and their linear program formulations," INRIA, Rocquencourt, France, Tech. Rep., Jul. 2010.
[34] *RevLib*. Accessed: Apr. 19, 2017. [Online]. Available: http://www.revlib.org
[35] *Gurobi Optimizer Reference Manual*, Gurobi Optim. Inc., Houston, TX, USA, 2014. [Online]. Available: http://www.gurobi.com
[36] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, "Surface code quantum computing by lattice surgery," *New J. Phys.*, vol. 14, no. 12, 2012, Art. no. 123011.
[37] J. Heckey *et al.*, "Compiler management of communication and parallelism for quantum computation," in *Proc. ACM SIGARCH Comput. Archit. News*, vol. 43. Istanbul, Turkey, 2015, pp. 445–456.
[38] M. R. Kramer and J. van Leeuwen, "Wire-routing is NP-complete," Dept. Comput. Sci., Univ. at Utrecht, Utrecht, The Netherlands, Tech. Rep. RUU-CS-82-4, 1982.



Yibo Lin (S'17) received the B.S. degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

Mr. Lin was a recipient of the Franco Cerrina Memorial Best Student Paper Award at the SPIE Advanced Lithography Conference 2016, and the University Graduate Continuing Fellowship in 2017.

Bei Yu (S'11–M'14) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

Dr. Yu was a recipient of four best paper awards at International Symposium on Physical Design 2017, the SPIE Advanced Lithography Conference 2016, the International Conference on Computer Aided Design (ICCAD) 2013, the Asia and South Pacific Design Automation Conference 2012, and plus three additional best paper Award nominations at Design Automation Conference/International Conference on Computer Aided Design/Asia and South Pacific Design Automation Conference, and three ICCAD contest awards in 2012, 2013, and 2015. He has served in the Editorial Boards of *Integration*, the *Very Large Scale Integration Journal*, and *IET Cyber-Physical Systems: Theory & Applications*.

Meng Li (S'15) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Texas (UT) at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

His current research interests include hardware-oriented security, reliability, power grid simulation acceleration, and deep learning.

Mr. Li was a recipient of the Best Paper Award in IEEE International Symposium on Hardware Oriented Security and Trust 2017 and Graduate Fellowship from UT Austin in 2013.

David Z. Pan (S'97–M'00–SM'06–F'14) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, CA, USA.

He is currently the Engineering Foundation Professor with the University of Texas at Austin, Austin, TX, USA. He has published over 280 refereed technical papers, and holds eight U.S. patents. He has graduated over 20 Ph.D. students who are currently holding key academic and industry positions. His current research interests include cross-layer nanometer IC design for manufacturability, reliability, security, physical design, analog design automation, and CAD for emerging technologies.

Prof. Pan was a recipient of the number of awards for his research contributions, including the SRC 2013 Technical Excellence Award, the DAC Top 10 Author in Fifth Decade, the ASP-DAC Frequently Cited Author Award, and 14 best paper awards. He has served as a Senior Associate Editor for *ACM Transactions on Design Automation of Electronic Systems*, an associate editor for a number of other journals. He has served in the executive and program committees of many major conferences, including ASPDAC 2017 Program Chair and ICCAD 2018 Program Chair. He is a fellow of SPIE.