Quantum Information and Computation, Vol. 20, No. 3&4 (2020) 0194–0212

A METHOD OF MAPPING AND NEAREST NEIGHBOR OPTIMIZATION FOR 2-D QUANTUM CIRCUITS

YUXIN ZHANG ZHIJIN GUAN[a] LANGYONG JI QIN FANG LUAN YIZHEN WANG
School of Information Science and Technology, Nantong University
Nantong, Jiangsu 226001, China

Received August 3, 2019
Revised January 19, 2020

In some practical quantum physical architectures, the qubits need to be distributed on 2-dimensional (2-D) grid structure to implement quantum computation. In order to map an 1-dimensional (1-D) quantum circuit into a 2-D grid structure and satisfy the nearest neighbor constraint of qubit interaction in the grid structure, a mapping method from 1-D quantum circuit to 2-D grid structure is proposed in this paper. This method firstly determines the order of placing qubits, and then presents the layout strategy of qubits in 2-D grid. We also proposed an algorithm for establishing interaction paths between non-adjacent qubits in 2-D grid structure, which can satisfy the physical constraints of the interaction of quantum bits in the grid in the process of mapping an 1-D quantum circuit to a 2-D grid structure. For some benchmark circuits, after using the method of this paper to place qubits, it is possible to make every 2-qubit gate in the circuit have a nearest neighbor, so that there is no need to use SWAP gate to establish channel routing. Compared with the latest available methods, the average optimization rate is 82.38%.

Keywords: Quantum circuit; 2-D grid; Nearest-neighboring; Quantum interaction cost; Interaction path
Communicated by: S Braunstein & M Mosca

1 Introduction

Quantum computers can implement parallel processing by using the characteristics of superposition of qubit states. Quantum computing's ability to solve some problems much better will be unmatched by traditional computers, such as integer factorization algorithm proposed by P. W. Shor [1] and fast quantum algorithm proposed by L. K. Grover [2] for database search. Compared to classical research, the work of Shor and Grover has inspired an in-depth study of quantum computing. Since then, quantum computing has been a very exciting and rapidly growing field of research. With the advancement of quantum technology, physical quantum computing devices have already emerged and been developing rapidly. In the past two years, IBM [3] and Rigetti [4] have provided their QC services to the general public over the internet. A new era is coming, and we can run quantum algorithms physically on a real quantum device rather than virtually on a quantum simulator. However, the current quantum devices are far from ideal. They have limited qubit resources and are susceptible to noise, so they are referred to as Noisy Intermediate-Scale Quantum (NISQ) devices. Further-

[a]Email: guan.zj@ntu.edu.cn

more, there exists a significant limitation to use such NISQ devices, i.e., the limited coupling between qubits.

In the logic design of quantum circuits, quantum gates are considered to be able to operate on qubits at any position, regardless of the physical position of qubits in the actual quantum system. At present, in most quantum circuit design, the distribution of qubits in 1-dimensional (1-D) is considered. In fact, the physical qubits of some quantum technologies appear as 2-dimensional (2-D) structures that can be abstracted into 2-D grids [5]. Limited by the current physical implementation technology of quantum computing [6]-[9], some quantum structures only support the interaction between adjacent qubits, so the problem of nearest neighbor of qubits needs to be solved. In general, qubit neighbors can be achieved by adding the SWAP gate. In the 1-D quantum circuit structure, a quantum bit only interacts with a qubit. In order to make the quantum circuit nearest neighbors, more SWAP gates will be used. In the 2-D grid structure [5], one qubit can interact with four qubits, which can reduce the number of SWAP gates in the process of neighbor. Each additional SWAP gate in the circuit means that the quantum cost will be increased accordingly. Therefore, how to reduce the number of SWAP gates has become an important topic in the field of quantum circuit research.

In this paper, according to the characteristics of quantum circuit of 2-qubit, the definition of qubit activity is proposed. According to the activity of qubits, a method for mapping an 1-D quantum circuit to a 2-D grid structure is presented. In order to reduce the number of SWAP gates inserted in the neighbor process of quantum circuits that method makes the qubits with high activity get priority placement. In the process of neighbor the mapped 2-D quantum circuit, because 2-qubit gate data set can measure the quality of interaction path, so determines the size of the binary gate data set prospectively. For non-neighbor quantum gate in quantum circuit, the SWAP gates are added to establish interactive path to realize the neighbor of 2-D quantum circuit.

2 Background

2.1 Quantum gate

In the computational model of quantum circuits, the quantum gate is a basic unit of quantum circuits that operates qubits. An 1-qubit gates perform their operations on their respective qubit lines, a 2-qubit quantum gate has two qubits, one of which is the control qubit (ctl_b) and the other is the target qubit (tar_b). The representative 2-qubit gate is the CNOT gate, whose the ctl_b is represented by "$\bullet$", and the tar_b by "$\oplus$". The state of tar_b is inverted when the state of the ctl_b is 1, and the state of the tar_b is not when the state of the ctl_b is 0. The quantum gate library is composed of a set of quantum gates with specific functions. The common quantum gate libraries include Multiple-control Toffoli (MCT) gates library, NCV (NOT, CNOT, V, V^+) gates library, and Clifford+T gate library etc. Toffoli gate is a universal reversible logic gate with three quantum bits, which is complete for traditional operations. Toffoli gate can implement all the operations of Boolean function. Clifford+T gate library contains $n \times (n + 6)$ quantum gates, where n is the possible number of qubits. The quantum gates contained in the NCV gate library are shown in Table.1.

Table 1. The NCV Gate Library

Type	Symbol	Diagram
NOT	N	
CNOT	C	
Controlled V	V	
Controlled V $^{-1}$	V^{+}	

2.2 Quantum logic circuit

The quantum circuit is a quantum computing model composed of quantum gate sequence. In general, a quantum computing process can be represented as a unitary transformation of a whole system. Therefore, the unitary transformation of any system can be expressed as a combination of 2-qubit CNOT gates and single-qubit rotation gates, and the combination process is quantum circuit. There are n horizontal lines in a quantum logic circuit, and each horizontal line represents a qubit, called a qubit line.

The 1-D quantum logic circuit contains only the time information of the circuit, but not the spatial information. Quantum circuits containing spatial information can be represented by placing qubits on a 2-D grid. Some quantum systems require the interaction of two quantum bits to be near neighbors. If two qubits of a 2-qubit gate are not adjacent, then the channel routing needs to be created for their communication. This process can be realized by adding a SWAP gate. For 1-D quantum circuit, more SWAP gates need to be added in order to establish channel routing for communication. Some quantum system structures, such as superconductor [24, 25] and quantum dot [26, 27], require 2-D nearest neighbor structures. Therefore, 2-D quantum circuit can be represented by 2-D grid [5], each grid cell represents a qubit, and the 1-D quantum circuit can be mapped to the 2-D grid.

Definition 2.1: It is called a regular 2-D grid structure when the number of grid cells of each row or column are the same in a 2-D grid structure. The regular 2-D grid structure is studied in this paper, so it is directly referred to as 2-D grid structure.

It is easy to know that in the 2-D grid structure $G(m, n), (m, n > 1)$, the grid cells with four edges have three neighbors (except the grid cells with four corners), while the non-edge grid cells have four neighbors. And each grid cell can correspond to a unique qubit. In a 2-D grid, the maximum number of neighbors of each qubit changes from 2 to 4, as shown in Fig.1.

2.3 Qubit distance

In the 1-D quantum circuit structure shown in Fig.1(a), each qubit (q0, q1... q8) in linear order from top to bottom, q0 and q8 have only one adjacent qubit, and the rest have at most two adjacent qubits. However, in the 2-D grid structure of this circuit, q4 has 4

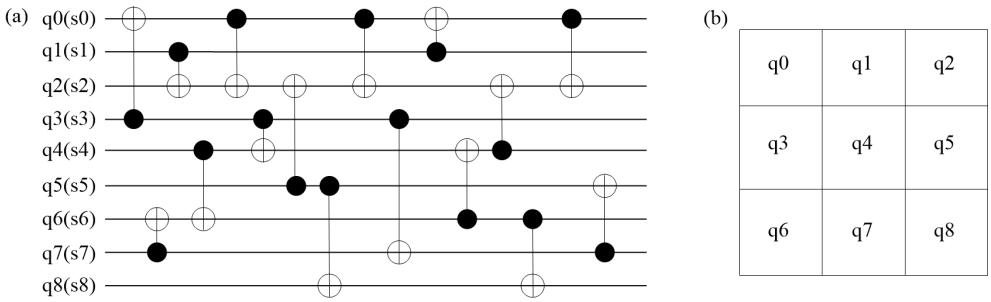

Fig. 1. (a) An Example of Quantum Logic Circuit (b) Placement in 2-D grids

"neighbors" and the remaining qubits also have 3 or 2 "neighbors" respectively, as shown in Fig.1(b). Compared with 1-D quantum circuit, each qubit in 2-D grid structure can increase the number of adjacent qubits, that is to say, the number of adjacent qubits that can interact with these qubits also increases. In a 2-D grid structure, the distance between qubits can be represented by the Manhattan distance.

3 Quantum circuit mapping

3.1 Qubit activity

Suppose the set of 2-qubit gates of an 1-D quantum circuit is G=$\{g_1,...,g_m\}$, whose corresponding set of qubits is Q=$\{q_1,...,q_n\}$. In order to realize the mapping from 1-D quantum circuit to 2-D grid structure, this paper uses the method of reducing the quantum cost of qubit near-neighbor operation (such as the number of SWAP gates) in advance to find the priority of qubit placement in 2-D grid (that is, the priority of qubit placement), and generate the queue of qubit placement order.

Definition 3.1: In a quantum circuit, the each qubit participates times in the 2-qubit gate activity is the qubit activity.

The value of qubit activity indicates the frequency of qubit interaction in quantum circuit. For example, in Fig. 2(a), the activity of q0-q8 is 5, 2, 6, 3, 4, 3, 4, 3, 2, respectively where the activity of q1 was lower than that of q2 and the difference of activity was 4.

When mapping an 1-D quantum circuit to a 2-D grid, a priority should be given to placing qubits with high activity in the grid with more idle neighboring locations, so as to make the closely related qubits gather together as much as possible, so as to achieve a pre-reduction of the number of SWAP gates in the process of near neighbor.

Definition 3.2: With the *ctl_b* and *tar_b* of the 2-qubit gate as the vertex, the *tar_b* as the head and the *ctl_b* as the tail, the 2-qubit quantum circuit can form a directed graph, which is called the quantum circuit mapping graph. As can be seen from definition 3.2, the in-degrees of a vertex in the quantum circuit mapping graph are the number of the target bit of 2-qubit gates that the qubit represented by the vertex, and the out-degree is the number of the control

bit.

Definition 3.3: In a quantum circuit map, the sum of the out-degrees and in-degrees of the vertex is called the activity of the qubit. Therefore, converting the quantum circuit mapping graph to an undirected graph is called a qubit activity graph. The adjacency matrix of the activity graph is called the activity matrix. The activity vector can be obtained from the activity matrix, and the priority queue vector of qubit layout can be obtained.

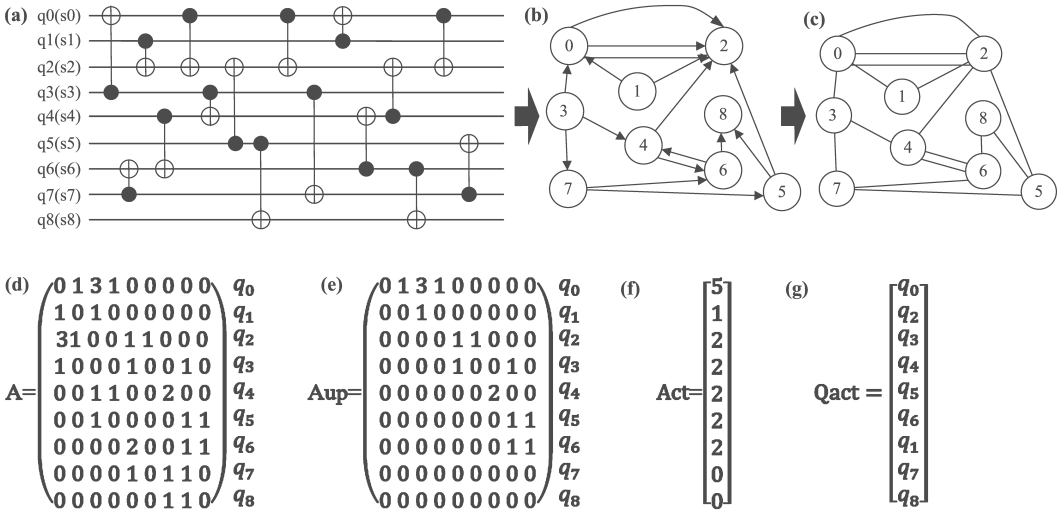

Fig. 2. (a) 1-dimensional quantum circuit, (b) mapping graph, (c) qubit activity graph, (d) activity matrix, (e) activity upper triangle matrix, (f) activity vector, (g) priority queue of qubit

The Fig.2 shows the mapping graph, qubit activity graph, activity matrix, activity upper triangle matrix, activity vector and priority queue vector, of quantum circuit figure 2.1 (a). The activity vector of the qubit in the figure is $Act = \{act_i\}$, where $act_i = aup_{i0}+aup_{i1}+...+aup_{i8}$, $0 \leq i \leq 8$. As can be seen from the definition 3.2, all the vertices represent qubits, and the indegree of a vertex represents the number of times the qubit participates as a *tar_b* in the 2-qubit quantum gate, and the outdegree is as a *ctl_b* in the activity digraph of a qubit, and the sum of the indegree and outdegree is the activity of the qubit. The activity digraph, activity matrix, activity upper triangle matrix, activity vector and priority queue of qubit in Fig.1(a) is showed in Fig.2.

3.2 Qubit placement order search

It can be seen from the above analysis that when the qubits are placed in the 2-D grid, the qubits placed first have certain advantages over the qubits placed later in the position selection, which is bound to affect the selection of the placement position of the next qubit. Therefore, It is important to find the order in which the qubits are placed. A method to find the order of placing qubits when an 1-D quantum circuit is mapped to a 2-D grid structure is presented below.

According to definition 3.3, the sum of the number of control bits and target bits on the

qubit line is the activity of qubit on the qubit line, which is denoted as QB_act. The higher the value of QB_act is, the more frequently the qubits participate in the interaction. In other words, the more activity QB_act qubits are, the more qubits will interact with them.

Algorithm 1: Qubit placement order search

Input: Interaction activity matrix CArr
Output: Priority queue qu
1 **Note** n as a quantum number;
2 **Initialize** the 2-D arrays *mat[][n]*;
3 **Initialize** the 1-D array *QBact[n]*;
4 **Initialize** the priority queue stack qu;
5 **if** $i < j$ **then**
6 *mat[i][j]=CArr[i][j]*;
7 **else**
8 *mat[i][j]=0*;
9 **for** $i = 0; j < n; i++$ **do**
10 **for** $i = 0; j < n; i++$ **do**
11 *QBact[i]+= mat[i][j]*;
12 **Compare** all elements of *QBact* and sort by size;
13 *qu.push()* $\leftarrow$ ***Stack*** *in order from smallest to largest*;
14 **Out** the priority queue;

An algorithm for finding priority queues is presented below, so that the most active qubits are placed in priority as much as possible. This order is placed for the reasonable arrangement of qubits laid a solid foundation, make active degree of qubits are placed in the process of configuration with more free neighbor position of the grid, and close the qubits together as much as possible, so as to achieve the purpose of reducing the number of SWAP gates inserts in the process of near-neighbor in advance.

For a given 1-D quantum circuit, it is easy to traverse each of its 2-qubit gates from left to right, and record each qubit participating in the 2-qubit gate, and then count the number of times each qubit participates in the interaction of qubit gates in the circuit, that is, the activity of the qubit QB_act.

The $\_mat$ in the algorithm is used to record the relevant information about each gate in the Ql, and Qact is used to record the activity vector. Lines 1-5 initialize $\_mat[][n], Qact[n], qu$, where n is the total number of qubits contained in the quantum circuit. Lines 6-10 are the pretreatment of the $CArr$, the propose is to avoid the redundancy of consideration, only the upper triangle region of the matrix is considered. Line 8 assigns the upper triangle region of the $CArr$ to the upper triangle region of the new $\_mat[][n]$. The purpose of line 10 is to set the lower triangle area of $\_mat[][n]$ to 0. Lines 11-13 operate on $\_mat[][n]$, superimpose each line element of $\_mat$ and assign it to the $Qact$, whose subscript corresponds to the row subscript of $\_mat$ one by one. The elements are sorted and put into the qu, according to the elements in $Qact$, which means the priority queue is stored in the stack qu.

3.3 2-D mapping of quantum circuits

3.3.1 Interaction graph

According to Ref. [5], interaction graph describes the relationship between objects and the information transfer between objects. In the research of quantum circuit, interaction graph is used to represent the interaction between different qubits. Among them, the vertex represents the qubit, the edge between the two vertices represents the existence of the 2-qubit gates (that is, there is an interaction relationship), and the weight on the edge represents the number of the 2-qubit gates. The interaction graph generated by the circuit in Fig.1(a) is shown in Fig.3.

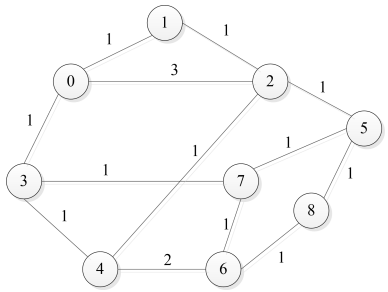

Fig. 3. An interaction graph of qubits

3.3.2 Cost function

In order to lay out qubits in a 2-D grid, the concept of cost function [9] is introduced and defined as follows:

$$f = \begin{cases} f_1 & act(v) = 0 \\ f_1 + f_2 & act(v) \neq 0 \end{cases} \tag{1}$$

Where $f_1 = \sum(dist(d, loc(va) - 1 \times ew))$, and $f_2 = min(0, deg(v) - frd(d)) \times act(v)/deg(v)$.

The $act(v)$ in the f represents the activity of qubit v which is not placed in the grid. The interaction between the qubit and the unplaced qubit is cleared, and $act(v)$ is refreshed accordingly while a qubit is placed in a 2-D grid. When $act(v) = 0$, it shows that there is no unplaced qubit associated with v. The dist represents the Manhattan distance between two qubits. The loc represents the position coordinates of qubits in a 2-D grid structure. The ew represents the weights of the edges between two qubits in the interaction graph. The $deg(v)$ represents the number of qubits related to the v in the unplaced qubit. The $frd(d)$ represents the number of grid cells whose Manhattan distance from the 2-D grid to the cell d is equal to 1 and where no qubits are placed.

The cost function consists of two parts: f_1 and f_2, where f_1 represents the cost of the relationship between the current v and the placed qubit va. In the case that v is not arranged at the va nearest neighbor position, the SWAP gates need to be inserted to realize the nearest neighbor of the 2-D quantum circuit, which increases the communication overhead between qubits, and the quantum cost of the circuit will increase correspondingly. Since there may be more than one va interacting with v, the relational cost of all vas needs to be calculated and summed. The $dist$ gives the distance between the position d, being considered currently, of the qubit and the position of the va.

The f_2 represents the average interaction cost between the v to be placed currently and the unplaced qubit. If $act(v)$=0, there are no unplaced qubits associated with the qubit to be placed, namely f_2=0, which does not need to be considered. If $act(v) \neq 0$, and $frd(d)$ is less than $deg(v)$. SWAP gates need to be added which increases the cost.

3.3.3 Qubit layout

In the process of placing qubits into a 2-D grid structure, if a qubit is placed and the placement position of the next qubit q needs to be determined, then according to the cost function f_1, the physical cost (Manhattan distance) between qubit q and the placed qubits related to it, and the number of two qubit gates between two qubits are considered comprehensively.

When determining the position of the qubit q to be placed, the closer the qubit q is to be placed and has an interaction with it, the smaller the cost f1 will be. When q and the qubits already placed of interacting with q are near neighbor, the cost f_1 is 0. The method of placing qubits as close as possible to the qubits that have been placed in the grid is called the nearest neighbor optimization principle when placing qubits to the grid cell. According to the nearest neighbor optimization principle, the traversal range of grid selection can be optimized.

Fig.4 shows an example to illustrate. $V[k](k = 0, 1, 2)$ represents the qubits that have been placed, $A[i](i = 0, 1, 2, ..., 6)$ represents a grid cell adjacent to $V[k], B[j](j = 0, 1, 2, ..., 14)$ represents a grid cell non-adjacent to $V[k]$. Assuming that the number of 2-qubit gates between various qubits have been determined, then only the Manhattan distance between the qubit to be placed and the qubit already placed is considered. The distance between arbitrary grid cell $A[i]$ and the placed grid cell $V[k]$ is

$$Dva[i] = \sum_{k=0}^{k<=2} |A[i] - V[k]|(i = 0, 1, 2, , 6) \tag{2}$$

And the distance between arbitrary grid $B[j]$ and the placed grid $V[k]$ is

$$Dvb[j] = \sum_{k=0}^{k<=2} |B[j] - V[k]|(j = 0, 1, 2, , 14) \tag{3}$$

By calculation, for any $Dva[i]$ and $Dvb[j]$, $Dva[i] < Dvb[j]$ is satisfied. Hence when calculating the cost, only the grid in $A[i]$ region needs to be considered, and the grid in $B[j]$ region be no longer considered.

The layout algorithm of qubits is given, according to the cost function and the optimized traversal range.

Algorithm 2 is a qubit layout algorithm whose input is a 2-qubit circuit Ql (Quantum_logic) and a priority queue qu (the search method for qu is given in section 3.2), and output is the information of the qubit layout in Ql on the 2-D grid. Lines 1-6 generate G_i and set the size of Gd when mapping the logic circuit to the 2-D grid, while initializing the tables $deg, act, frd, nbr.deg(v)$ records the number of edges associated with the vertices in the interaction graph. The $act(v)$ records the sum of weights on the edges related to v. The frd records the number of grid cell adjacent to a grid cell without qubits. The nbr is a pointer array, which records the information of vertices related to the v placed on the grid. The size of Gd depends on the number of qubits q, the height $H = \sqrt{q}$, and the width W=$\lceil q/H \rceil$.

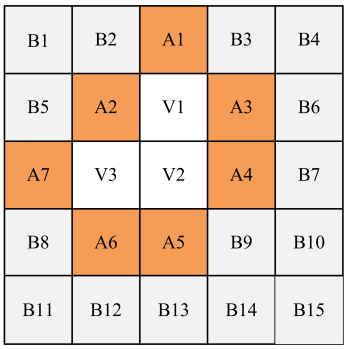

Fig. 4. An example of traversal range optimization

Algorithm 2: Qubit layout

Input: Two qubit circuit Ql (Quantum logic) and its qubit priority queue *qu*
Output: The arrangement of qubits in Ql on a 2-D grid

```
The interaction graph G_i = (V, E), V is the vertex set, E is the edge set;
Initialize table deg (v);
Initialize table act (v);
Setthe size of grid Gd when mapping logic circuits to 2-D (W × H);
Initialize table frd(d);
Initialize table nbr[v] to record nearest-neighboring information;
Select the first qubit v_0 in Qu to get the information of v_0 in the Gi;
Place v_0 in Gd and update tables frd and nbr;
if qu is not empty then
    v ← qu.pop() ;
    if v is not placed on Gd then
        Determine the range of areas on the Gd that need to be traversed to
          optimally place v;
        Put v on dv,dv ← min f(v,d);
        Update frd(d), nbr, deg(v), act(v);
Until all qubits are placed
```

The qubit layout is performed on lines 7-16, and the qubits are traversed according to the priority queue until all qubits are placed into the 2-D grid. The lines 8-9 place the first qubit in the priority queue, which is the most active, as the geometric center of the grid, then update frd and nbr. The lines 10-12 sequentially take out the qubit v from qu and judge whether it has been placed. The line 13 makes a preliminary judgment on the placement area where the qubit will be placed. To reduce unnecessary traversal and reduce the complexity of the algorithm, the grid cell far away from the grid cell where the qubit has been placed will not be considered according to the principle of nearest-neighbor optimization. In lines 14-15, the possible placement area needs to be traversed if v is not placed in Gd. Finally, the qubit is placed at dv where the f is minimized. The data in the relevant tables $frd(v), nbr, deg(v)$, and $act(v)$ need to be updated after the current placement of the qubit. The line 16 repeat the algorithm for lines 10-15 until all qubits have been placed into grid Gd.

3.3.4 Illustration

In the Fig.5 shows the interaction graph generated by the circuit in Fig.3(a), which contains nine qubits v0, v1, v2, v3, v4, v5, v6, v7, v8 and 16 2-qubit gates (the sum of weights on all edges).

In Fig.5(a), deg, act, the frd and nbr are initialized, and the priority queue to determine the placement of qubits is qu=2,0,1,3,4,5,6,7,8. In Fig.5(b), the first qubit $v2$ in qu is firstly selected. Since v0, v1, v4, v5 all interact with it, their corresponding $deg(v)$ and $act(v)$ are updated after they are placed in the grid. For example, in Fig.5(b), the $deg(0)$ and $act(0)$ corresponding to $v0$ are updated to 2 and 2 respectively from the original 3 and 5, the Manhattan Distance from the place of $v2$ in frd is 1, and the frd value corresponding to the grid cell without qubits is reduced by 1, the $nbr(0)(va, ew)$ corresponding to $v0$ is updated to (2,3). Similarly, the other qubits v1,v4,v5 associated with $v2$ are updated accordingly. After the layout of algorithm 2, the qubits have been placed on the 2-D grid. However, the algorithm 2 is to map the qubits in the circuit globally without considering the neighboring problem of qubits of each quantum gate in the circuit. Therefore, it is necessary to near-neighbor operation on the mapped 2-D quantum circuit, that is, to dynamically generate channel routing of qubits in the grid.

4 Neighboring for 2-D quantum circuit

4.1 Interaction cost

Definition 4.1: The number of SWAP gates needed to be added in order to realize the nearest neighbor of all 2-qubit quantum gates in the quantum circuit is called the qubit nearest neighbor cost .

This definition applies to 1-D quantum circuits and 2-D grid structures.

The location of the qubit is closely related to the nearest neighbor cost of the quantum circuit [10] when mapping the qubits of 1-D quantum circuit to the 2-D grid. The activity information of qubits and the placement of qubits in 2-D grid directly determine the nearest neighbor cost of quantum circuits. The qubits with high activity need to be given priority when placing the qubits to make the ctl_b and tar_b of the 2-qubit gate as close as possible (measured by Manhattan distance) and to minimize the qubit interaction cost. In this way,

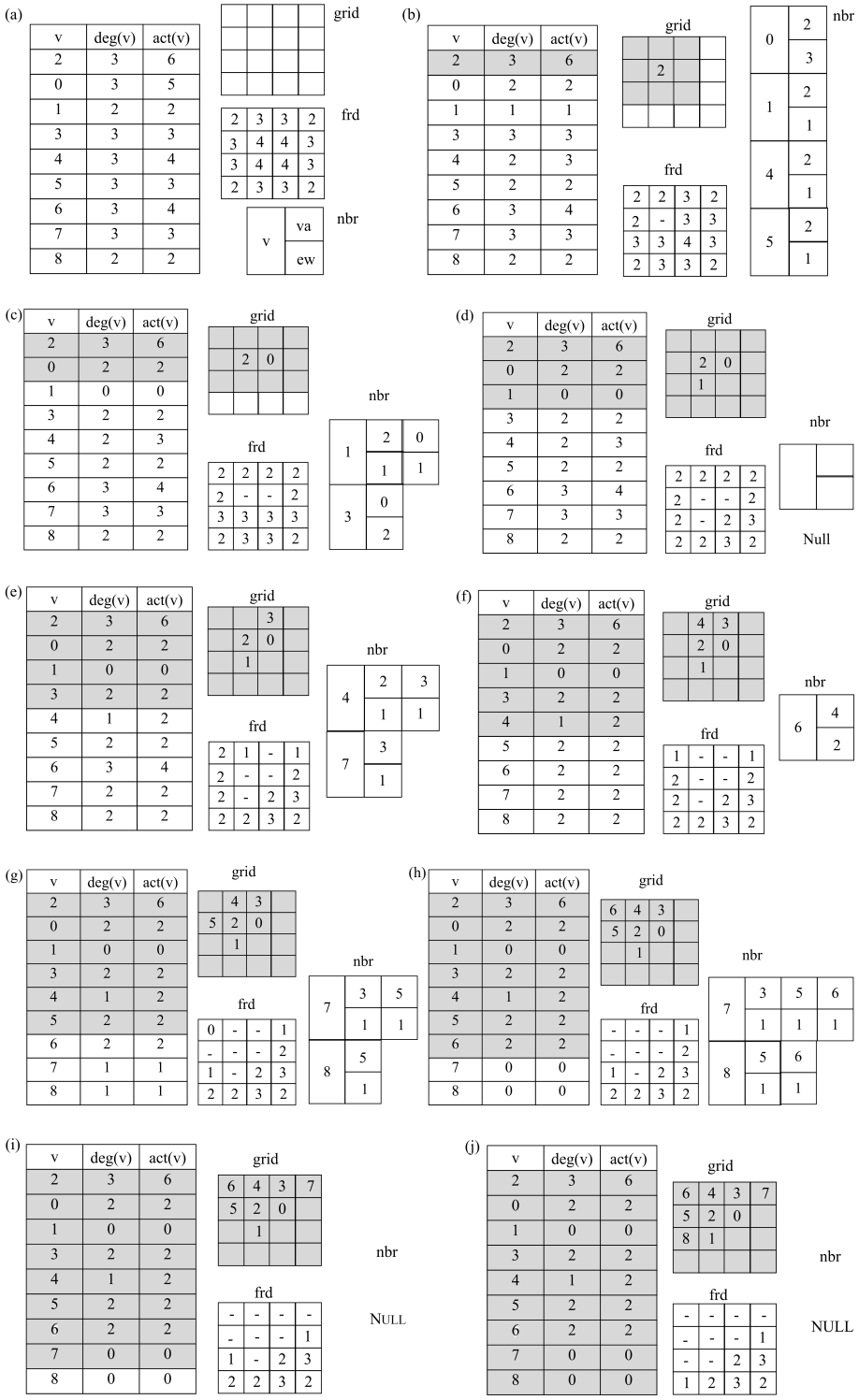

Fig. 5. Qubit layout example. (a) Initialize tables (b) v2 (c) v0 (d) v1 (e) v3 (f) v4 (g) v5 (h) v6 (i) v7 (j) v8

the nearest neighbor cost of quantum circuit is reduced. The height H of the placed 2-D grid is $\lceil\sqrt{q}\rceil$, width W is $\lfloor q/H\rfloor$, where the number of qubits in an 1-D quantum circuit is q [10]. Furthermore, the most active qubit in the set of qubits is placed at a position where the coordinates are (x, y)=($\lceil H/2\rceil$, $\lceil W/2\rceil$).

Theorem 1: Let the 2-qubit gate be G(q_i, q_j), i and j be the number of qubit line of the qubit gate in the circuit, then

(a) In an 1-D quantum circuit, the number of SWAP gates that need to be added for the nearest-neighbor of quantum gate depends on the positions i and j of the qubit line where the quantum gate is located, and the number of SWAP doors added is

$$s_1 = |i - j| - 1 \tag{4}$$

(b) The number of SWAP gates that need to be added for the nearest-neighbor of quantum gates in the 2-D structure depends on the position of the qubit line of the quantum gate in the 1-D quantum circuit and the width of the 2-D grid, and the number of SWAP gates added is

$$s_2 = |n - m + w(\lfloor\frac{m}{w}\rfloor - \lfloor\frac{n}{w}\rfloor)| + \lfloor\frac{m}{w}\rfloor - \lfloor\frac{n}{w}\rfloor - 1 \tag{5}$$

Proof: Let alt be the vertical distance between q_i and q_j, and hor be the horizontal distance between q_i and q_j. In order to describe the number of SWAP gates required for qubit nearest-neighbor in 1-D quantum circuit and 2-D grid structure, only the sequential placement of qubits is considered here. That is, the qubits are placed from left to right and from top to bottom in the 2-D grid according to the order of subscripts from small to large in 1-D circuit.

(a) For 1-D quantum circuit, the distance between the control bit and the target bit of the 2-qubit gate is $d = |i - j|$. Obviously, the nearest-neighbor of the quantum gate can be achieved by adding $d-1$ SWAP gates, that is, the number of added swap gates is $s_1 = |i-j|-1$.

(b) For 2-D grid structure, the horizontal and vertical distances (Manhattan distance, the same below.) of the two qubits in the quantum gate are respectively the number of SWAP gates to be added. Let w be the width of 2-D grid. Considering that the control bit of the quantum gate is independent of the order of the target bit in this paper, let $m = Maxi, j$, $n = mini, j$, then the vertical distance from q_i to q_j in the 2-D grid is

$$alt = \lfloor\frac{m}{w}\rfloor - \lfloor\frac{n}{w}\rfloor, \tag{6}$$

The position of q_m mapped to q_n is (from left to right)

$$t = m - w(alt + \lfloor\frac{n}{w}\rfloor) = m - w(\lfloor\frac{m}{w}\rfloor), \tag{7}$$

The number of the line q_m maps to q_n is

$$k = w(alt + \lfloor\frac{n}{w}\rfloor) + t = m + w(\lfloor\frac{n}{w}\rfloor - \lfloor\frac{m}{w}\rfloor) = m - w \times alt, \tag{8}$$

Thus, the transverse distance is

$$hor = |n - k|, \tag{9}$$

The number of SWAP gates required in the 2-D structure is

$$s_2 = alt + hor - 1 = |n - m + w \times (\lfloor\frac{m}{w}\rfloor - \lfloor\frac{n}{w}\rfloor)| + \lfloor\frac{m}{w}\rfloor - \lfloor\frac{n}{w}\rfloor - 1 \tag{10}$$

It is proved by s_1 and s_2 theorem.

4.2 Channel routing optimization

In order to find the channel routing of nearest neighbor qubit interaction in 2-D grid, the size of the two-qubit gate data set (i.e. the optimization window) that measures the quality of the routing should be selected first, and then the minimum number of SWAP gate inserts in the optimization window should be found, that is, the nearest neighbor cost of the optimal quantum circuit. The size of optimization window is determined before the nearest neighbor of quantum circuit, which can effectively reduce the complexity of the nearest neighbor algorithm to find the optimal qubit channel routing.

Algorithm 3: Optimization-window size selection

Input: The initial value of *rw*
Output: Optimal rw and minimum *Sp[rw]*

```
1  Initialize rw=i, i=1;
2  Initialize count=0;
3  t1[1] ← Sp[rw],t2[1] ← rw,i++, i ≤ 10;
4  for each result=sgn(Sp[i]-Sp[rw]), sgn is a sign function, rw=i do
5      if result changes -1 (or 0) to 1 then
6          count++, t1[count] ← Sp[rw],t2[count] ← rw;
7      if count equals 2 then
8          break;
9  if count equals 1 or 0 then
10     t1[2]] ← Sp[rw],t2[count] ← rw;
11 Select index k of minimum;
12 k ← Select t2[1], t2[2] ← min t1[1], t1[2];
```

A heuristic algorithm to optimize window size selection (*Select_rw*) was presented in this paper. The main purpose of $Select_rw$ is to try to determine the size of the optimization window rw.

When the window size is rw, the number of SWAP gates to be added is $Sp[rw]$ in algorithm 3. Algorithm 3 tries to find the minimum value of $Sp[rw]$, that is, to find the value of the optimal window rw for the minimum cost of qubit interaction (the qubit nearest neighbor cost).

The input of algorithm is an initial value of the optimization rw, which is set to 1 (line 2), a counter i is also set to record the possible window size, where $i < 10$. (In a large number of experimental statistics, when the optimization window is less than 10, the nearest neighbor cost will be smaller twice, when the optimization window is greater than 10, it will not be better.) Some counters are initialized in lines 3-4. The window size is selected in lines 5-11, line 5 initializes counter array t_1 and t_2, the number of SWAP gates to be added are stored in t_1 when window size is 1, and window size 1 is stored in t_2. In line 6, *sgn* is a sign function for judging the sign of $Sp[i] - Sp[rw]$. Set $result = 1$ when the sign of $Sp[i] - Sp[rw]$ is positive; set $result$ = -1 when the sign of $Sp[i] - Sp[rw]$ is negative; set $result = 0$ when $Sp[i]$ is equal to $Sp[rw]$. Line 7-8, count++ whenever the value of $result$ changes from - 1 to 1, and the current rw is assigned to $t2[count]$, $Sp[rw]$ to $t1[count]$. The count is judged as whether it is equal to 2 in line 9, and loop ends when *count* is equal to 2. The 10th set of data is default

as the end point, and relevant information is assigned to $t_1[2]$, $t_2[2]$ when count is 1 or 0 in line 10-11. In the lines 12-14, the smallest number of SWAP gates is selected in the array t_1, namely $Sp[rw]$, and the corresponding window size rw is found in t_2.

The algorithm is used to find the optimal window size, and only suitable for finding the optimal window size not exceeding 10 when establishing the channel routing in the process of 2-D quantum circuit nearest neighbor. The algorithm only needs to find the value of two cost (the number of added SWAP gates) at the trough value (i.e. the minimum value point) to measure the quality of interaction paths, and compare the two cost values. The smaller cost value is considered to be the number of SWAP gates added corresponding to the optimal window size. In Fig.6(a), the trough values of the two costs of ham7 are 25 and 30, respectively. The trough value of cost are minimum, the optimization window size is 4 correspondingly. Similarly, the optimization window size in hwb4 is 5 in Fig.6(b).

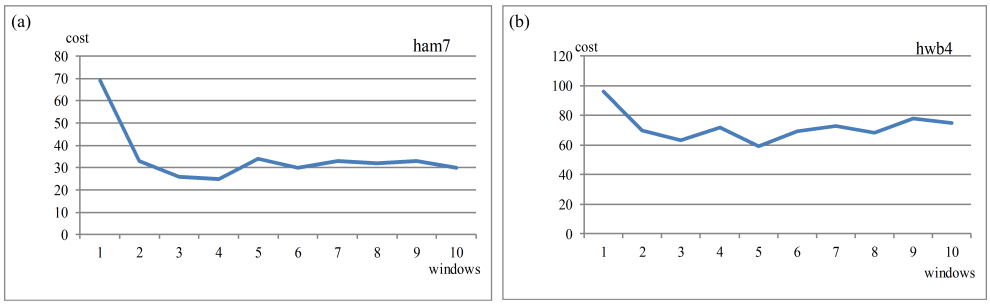

Fig. 6. (a) The ham7 windows result (b) The hwb4 windows result

4.3 2-D quantum circuit nearest neighbor algorithm

The input of the 2-D quantum circuit near neighbor algorithm is a non-neighbor quantum circuit arranged on a $W \times H$ grid, and the output is a nearest neighbor 2-D circuit. The cost calculation in this algorithm is based on the sum of Manhattan distances between each ctl_b and tar_b of all 2-qubit gatesin the optimization window rw.

The pseudo-code of 2-D quantum circuit nearest-neighboring (i.e., seeking the channel routing of non-neighboring qubits) is given in algorithm 4. In line 2, the optimization-window size selection algorithm is used. The stack ga is initialized to trace the insertion of the SWAP gate chain (line 3), which is used to restore the qubits to their original state at the end of the calculation. In lines 4-13, an channel routing is constructed by each non-neighboring 2-qubit gate of the circuit. In line 5, the optimization window rw slides a gate. In line 6, the array va that consists of all possible intersection points of the ctl_b and tar_b of the gate is generated, which is the grid position where the ctl_b and tar_b meet for adjacent interaction. In lines 7-13, $pcv(ptv)$ represents the set of all routes from ctl_b (tar_b) to $va[i]$. The best channel routing $pctv[k]$ in $pctv$ is found in each iteration $va[i]$. $pctv[k]$ is the path connecting $pcv[k]$ and $ptv[k]$ which has the minimum interaction cost. The optimal intersection point $va[k]$ is determined simultaneously. The SWAP gate is inserted based on $pctv[k]$, then the ctl_b and tar_b of the quantum gate are in the nearest neighbor state. The history of SWAP gate chain inserts is traced in lines 14-21. The dependency table 4 dt is established to track

the dependency of the current SWAP gate and the previous SWAP gate. The operation is redundant and can be cancelled when the two SWAP gates continuously operate on the same two qubits.

Algorithm 4: Quantum circuit near-neighboring

Input: A quantum circuit arranged on a $W \times H$ grid
Output: Nearest-neighboring 2-D grid circuit

```
Window size selection algorithm;
Initialize stack ga for tracking the SWAP chain;
for each non-adjacent two-qubit gate do
    Optimization window rw slides a gate (ctl_b, tar_b);
All possible encounters between ctl_b and ctl_b are stored in va;
for each va[i] do
    for all paths, from ctl_b to va[i] do
        pcv[i] ← paths with minimum cost;
    for all paths, from tar_b to va[i] do
        ptv[i] ← paths with minimum cost;
    pctv[i] = pcv[i] + ptv[i];
Compare all pctv to find the smallest pctv[k] → optimal convergence point va[k];
Set a dependency table dt to record the current and previous SWAP gates;
if the qubit of the current gate equals the qubit of the previous gate then
    Redundant exchange, cancelled;
else
    Put into stack ga
while ga not empty do
    Insert SWAP gate ga.back();
    ga.pop_back();
```

An 1-D quantum circuit (including q qubit lines and r 2-qubit gates) is mapped to a $n \times n$ grid, where $n = q$. The maximum distance between two qubits in the grid is $2n$-2 (worst case) when two qubits are diagonally arranged in the grid. There is no circuit which only contains qubit-pair with the maximum distance (the *ctl_b* and the *tar_b* of a quantum gate are called qubit-pair) when it is arranged for general case. The whole circuit needs to be considered synthetically to get the average distance between the two qubits, the farther the distance is, the more encounters between the two qubits, and the more computational effort is needed to find the optimal channel routing. The distance of all possible qubit-pair depends on the Manhattan distance d of the gate in the 2-qubit grid, and the number of pairs of quantum bits under different conditions is obtained.

4.4 Complexity analysis

The number of qubit-pairs in the same row or column of the grid is given by $c0$, the number of qubit-pairs is given by $c1$ with distance greater than or equal to 2 but less than n. The number of qubit-pairs is given by $c2$ when the distance is greater than or equal to n but less

than $2n$-2.

$$P_c = \begin{cases} 2n(n-d) & P_c0 \text{ :on the same row or column} \\ 2\sum_{i=1}^{d-1}(n-d-i)(n-i) & P_c1 : 2 \le d \le n \\ 2\sum_{i=0}^{2n-d}(i)(2n-d-i) & P_c2 : n \le d \le 2n-2 \end{cases} \tag{11}$$

In the (4), The first case is where the qubit-pairs are on the same row or column in the grid; the qubit distance of the qubit-pairs are greater than or equal to 2 and less than n in the second case; they are greater than or equal to n and less than $2n-2$ in the third case. The average distance of qubit-pairs is

$$D_{avg} = \frac{\sum_{d=1}^{n-1}(d \times Pc0) + \sum_{d=2}^{n-1}(d \times Pc1) + \sum_{d=n}^{2n-2}(d \times Pc2)}{\sum_{d=1}^{n-1} Pc0 + \sum_{d=2}^{n-1} Pc1 + \sum_{d=n}^{2n-2} Pc2} = \frac{2n}{3} \tag{12}$$

A rectangular region for the selection of the interaction paths is given by the arrangement position of the qubit-pairs gives. The number of encounters varies with the shape of the interaction path region. For example, when a qubit-pair with Manhattan distance d is in the same row or column of the grid, there are only d encounters. On the other hand, the maximum encounter is m^2 (worst case), where $m = \lceil d/2 \rceil + 1$, when the shape of the selected region is square.

5 The experimental results and analysis

Our algorithm is implemented in C++. To test the practicality of our algorithm, we use a normal personal computer with Intel(R) 64 CPU and 8 GB memory as the runtime environment.

Among the existing heuristic-based solutions to the qubit mapping problem, the algorithm (SABRE) released several months ago in [31] is the latest one so far, and outperforms other algorithms in terms of gate count. Therefore, we will choose SABRE as a baseline to evaluate gate count of our algorithm. Since Ref.[31] is compared with Ref.[29], relevant results of Ref.[29] are also listed in the Table 2.

In Table 2, small: small quantum arithmetic. sim: quantum simulation. qft: quantum fourier transform. large: large quantum arithmetic. n: number of logical qubits in the original circuit. Gori: original number of gates. B-gadd: number of additional gates in Ref.[29]. S-gop: number of additional gates after reversal traverse in Ref.[31]. Our: number of additional gates in this paper. Out: out of memory. N: Non comparable circuit. opti-rate: optimization rate.

For some benchmark circuits, after using the method in this paper to place qubits (Algorithm 2), every 2-qubit gate in the circuit can be adjacent, so that it is not necessary to use SWAP gate to establish channel routing. The experimental results show that the number of gates mapped by the method in this paper is significantly reduced, and compared with the SABRE algorithm in Ref.[31], the average optimization rate is 82.38%, and compared with the BKA algorithm in Ref.[29], the average optimization rate is 85.15%.

Since the Ref. [31] adopts the method of step by step optimization, That is to say, the qubit is placed at the same time of optimization, the first five small-scale circuits in Table 2 can be optimized. However, the method of this paper is: first, the overall layout is based on the activity of the qubits in the circuit, and then the qubits are placed. It has more

advantages for large-scale circuit, but the optimization effect for small-scale circuit is not as good as the method of Ref.[31].

Table 2. Comparisons between reference [31] and experimental results

Type	Original circuit		B-gadd		S-gop	Our	opti-rate	
	Name	n	Gori				BKA	SABR
small	4mod5-v1_22	5	21	15	0	4	73.33%	N
small	4mod5-v1-22	5	21	15	0	4	73.33%	N
small	alu-v0-27	5	36	33	0	8	75.76%	N
small	decod24-v2-43	4	52	52	0	10	80.77%	N
small	4gt13-92	5	66	42	0	13	69.05%	N
sim	ising-model-16	16	786	out	0	43	N	N
qft	qft-10	10	200	66	54	29	56.06%	46.30%
qft	qft-13	13	403	177	93	48	72.88%	48.39%
qft	qft-16	16	512	267	186	101	62.17%	45.70%
qft	qft-20	20	970	out	372	107	N	71.24%
large	rd84-142	15	343	138	105	94	31.88%	10.48%
large	adr4-197	13	3439	1722	1614	69	95.99%	95.72%
large	radd-250	13	3213	1434	1275	67	95.33%	94.75%
large	z4-268	11	3073	1383	1365	9	99.34%	99.34%
large	sym6-145	14	3888	1806	1650	15	99.17%	99.09%
large	misex1-241	15	4813	2097	1521	65	96.90%	95.73%
large	rd73-252	10	5321	2160	2133	29	98.66%	98.64%
large	cycle10-2-110	12	6050	2802	2622	38	98.64%	98.55%
large	square-root-7	15	7630	3132	2598	56	98.21%	97.84%
large	sqn-258	10	10223	4737	4344	25	99.47%	99.42%
large	rd84-253	12	13658	6483	6147	11	99.83%	99.82%
large	co14-215	15	17936	9183	8982	35	99.62%	99.61%
large	9symml-195	11	34881	17496	17268	15	99.91%	99.91%
Average							85.15%	82.38%

6 Conclusion

In this paper, an effective method of mapping 1-D quantum circuits to 2-D grid structure quantum circuits and the nearest neighbor method are proposed. Another optimization algorithm is proposed to establish interactive paths between nonadjacent qubits in 2-D grid structure and the minimization window is quickly found. The cost of qubit interaction path of circuit is reduced in algorithms of this paper compared with the existing research results, and for many benchmark circuits, the cost is 0. In the future work, how to expand the scale of quantum circuits and find the optimization interaction path is necessary to further study.

Acknowledgements

This work was supported by Chinese National Natural Science Foundation (61402244), Natural Science Foundation of Jiangsu Province (BK20151274), Science and Technology Project Foundation of Nantong(GY12017024).

References

1. P. Shor (1994), *Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum compute*, In 35th Annual Symp. on the Foundations of Computer Science, Vol.1, pp. 124.

2. L. Grover (1996), *A fast quantum mechanical algorithm for database search*, In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing., Vol.1, pp. 212-219.
3. IBM Q Experience, *Accessed on: Jun. 20, 2019*, [Online] Available: https://www.research.ibm.com/ibm-q/technology/experience/
4. Rigetti Quantum Cloud Services, *Accessed on: Jun. 20, 2019*, [Online] Available: https://www.rigetti.com/,
5. A. Shafaei (2014), *Qubit placement to minimize communication overhead in 2-D quantum architectures*, Conf., Jan. A, 54, pp. 495500.
6. M. Saffman, T. Walker and K. Molmer (2010), *Quantum information with rydberg atoms*, Phys. Rev. Mod, 82, pp. 23132363.
7. A. Blais et al. (2014), *Quantum-information processing with circuit quantum electrodynamics*, Phys. Rev. A, 75, pp. 032329.
8. Y. Y. Tan, X. Y Cheng, Z. J Guan, Y. Liu (2018), *Multi-strategy based quantum cost reduction of linear nearest-neighbor quantum circuit*, Quantum Inf Process Vol.17, pp. 61.
9. C. C. Lin, S. Sur-Kolay and N. K. Jha (2015), *PAQCS: physical design-aware fault-tolerant quantum circuit synthesis*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems., Vol.23, pp. 1221-1234.
10. C. A. Perez-Delgado, M. Mosca, P. Cappellaro, D. D. Cory (2006), *Single spin measurement using cellular automata techniques*, Phy. Rev. Vol.97/100501.
11. J. M. Taylor et al.(2007), *Relaxation, dephasing, and quantum control of electron spins in double quantum dots*, Phys. Rev. B, Vol76, pp. 045-075.
12. P. A. Barenco, C. H. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter (1995), *Elementary gates for quantum computation*, Phys., Vol.52, pp. 34573467.
13. D.M. Miller and R. Wille (2011), *Elementary quantum gate realizations for multiplecontrol Toffolli gates*, International Symposium on Multiple-Valued Logic, IEEE, New York, pp. 217-222.
14. D. Groe, R. Wille, G.W. Dueck and R. Drechsler (2009), *Exact synthesis of elementary quantum gate circuits*, Multiple-Valued Log. Soft Comput., Vol.15(4), pp. 283300.
15. R. Wille, D. Groe, L. Teuber, G. Dueck and R. Drechsler (2008), *RevLib: An online resource for reversible functions and reversible circuits*, In Proc. Int. Symp. Multi-Valued Logic., pp. 220225.
16. Z.Y. Zhang, Z.J. Guan, H. Zhang, H.Y. Ma and W.P. Ding, (2018), *A method of synthesis and optimization for linear nearest neighbor quantum circuits by parallel processing*, Quantum Information and Computation(QIC)., Vol.18(13&14), pp. 1095-1114.
17. X.Y. Cheng, Z.J. Guan and W.P. Ding, (2018), *Mapping from multiple-control Toffoli circuits to linear nearest neighbor quantum circuits*, Quantum Inf Process., Vol.17, pp.169.
18. B. Giles and P. Selinger (2013), *Exact synthesis of multiqubit CliffordCT circuits*, Phys. Rev. A, 87(3), pp. 032332.
19. V. Kliuchnikov, D. Maslov and M. Mosca (2013), *Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates*, Quantum Inf. Comput., Vol.13, pp. 607-630.
20. R. P. Feynman (1985), *Quantum mechanical computers*, Optics News/0105090.
21. A. G. Fowler, A. M. Stephens and P. Groszkowski (2009), *High threshold quantum computation on the surface code*, Phys. Rev. A, 80(5), pp. 052312.
22. F. Azim and M. Naser (2017), *Quantum circuit physical design flow for 2-D nearest-neighbor architectures*, International Journal of Circuit Theory & Applications., Vol.45, pp. 989-1000.
23. C. Neha, B. Chandan and R. Hafizur (2017), *Improving the Design of Nearest Neighbor Quantum Circuits in 2-D Space*, VLSI Design and Test., pp. 421-426.
24. R. Barends et al., (2014), *Superconducting quantum circuits at the surface code threshold for fault tolerance*, Nature., Vol.508(7497), pp. 500-503.
25. R. Versluis et al. (2016), *Scalable quantum circuit and control for a superconducting surface code*, arXiv/1612.08208.
26. C. D. Hill et al. (2015), *A surface code quantum computer in silicon*, Science advances, l(9)/e1500707.

27. R. Li et al. (2017), *A crossbar network for silicon quantum dot qubits*, arXiv/1711.03807.
28. M. Y. Siraichi et al. (2018), *Qubit allocation*, in Proc. IS-CGO, Vsendorf, Austria, pp. 113-125.
29. A. Zulehner, A. Paler, and R. Wille. (2018), *An efficient methodology for mapping quantum circuits to the IBM QX architecture*, IEEE T. Comput. Aid. D., 38(7), pp. 1226-1236.
30. A. Zulehner, A. Paler, and R. Wille. (2018), *Efficient mapping of quantum circuits to the ibm qx architectures*, In Design, Automation & Test in Europe Conference & Exhibition, pp. 1135-1138.
31. G. Li, Y. Ding, and Y. Xie. (2019), *Tackling the qubit mapping problem for nisq-era quantum devices*, in Proc. ASPLOS, New York, NY, USA, pp. 1001-1014.
32. IBM. QISKit, Open Source Quantum Information Science Kit, *Accessed on: 2018*, [Online], Available: https: //qiskit.org/