Layout Optimization for Quantum Circuits with Linear Nearest Neighbor Architectures

Massoud Pedram and Alireza Shafaei

Special Issue Dedicated to the Memory of Prof. Ernest S. Kuh

Abstract

This paper is concerned with the physical design of quantum logic circuits. More precisely, it addresses the problem of minimizing the number of required qubit reorderings (achieved by inserting explicit SWAP gates) when mapping a quantum circuit into a linear nearest neighbor quantum architecture. First, an interaction graph that captures the interaction distances among various qubits in the quantum circuit is constructed. The interaction graph is utilized to partition the quantum circuit into a set of subcircuits such that the number of required qubit reoderings within each subcircuit is provably no more than a given threshold. Next, a Minimum Linear Arrangement problem for each subcircuit is formulated and solved to achieve the minimum number of internal qubit reorderings and determine the subcircuit input and output qubit orderings. Finally, a bubble sort algorithm is repeatedly employed to minimize the number of qubit reorderings that are required

Digital Object Identifier 10.1109/MCAS.2016.2549950
Date of publication: 24 May 2016

between the consecutive subcircuits. Experiments done on various quantum Fourier transform circuits as well as various reversible logic circuits demonstrate the effectiveness of the proposed approach.

I. Introduction

It has been shown that quantum computing can offer significantly higher performance on certain problems compared to classical computing. For example, the Shor's number factoring algorithm can factor a given value $M > 0$ in $O((\log M)^3)$ time which is exponentially faster than the best-known classical factoring algorithm, i.e., the general number field sieve, with $O(e^{(\log M)^{1/3}(\log \log M)^{2/3}})$ time complexity. As a result, substantial research effort has been made to discover new quantum algorithms with improved performance (see [1] for a comprehensive list of quantum algorithms with superpolynomial or polynomial speedups over their classical counterparts), and to investigate different quantum physical technologies to construct a scalable quantum computer (see [2]–[6] for recent break-through progresses reported by D-Wave, Google, Microsoft, IBM, etc.).

A well-known technique to implement a quantum algorithm on a quantum computer is to run a quantum physics experiment under the control of a classical computer [7]. In this scenario, the experimental apparatus, i.e., the *quantum core*, consists of physical qubits such as ions or photons where the quantum-mechanical properties of the qubits are used to perform the required computation. A *real-time classical computer* directs the experiment by issuing instructions and reading out the quantum states. The instruction sequence is generated from a high-level *quantum program* by an off-line compilation process. The final result is available after output post-processing and result verification, both of which are also done off-line.

Fig. 1 demonstrates the high-level view of a quantum computing system. In this figure, an abstract representation has been used for the quantum core where measurement and interaction between qubits occur in particular locations (called *wells*) in the quantum core. Additionally, qubits are stored in designated wells during their idle time where each well may contain zero, one or several qubits depending on the underlying quantum technology. Furthermore, some quantum technologies include channels between wells, which in turn provide means for moving qubits form one well to another. Indeed, different quantum technologies may use different concepts for the quantum core, which is described by the *physical machine description* (PMD).[1] Four PMDs with different physical topologies, qubit representations, and qubit routings are shown in Figure 2.

A realistic, non-ideal quantum computer is subject to noise and faces numerous limitations and constraints.

[1]PMD is a description for the underlying quantum physical machine technology, which includes the qubit decoherence super-operator, control response functions, inter-qubit connectivity, geometric constraints (e.g., nearest neighbor interactions), qubit movement constraints, crosstalk, and detector parameters.

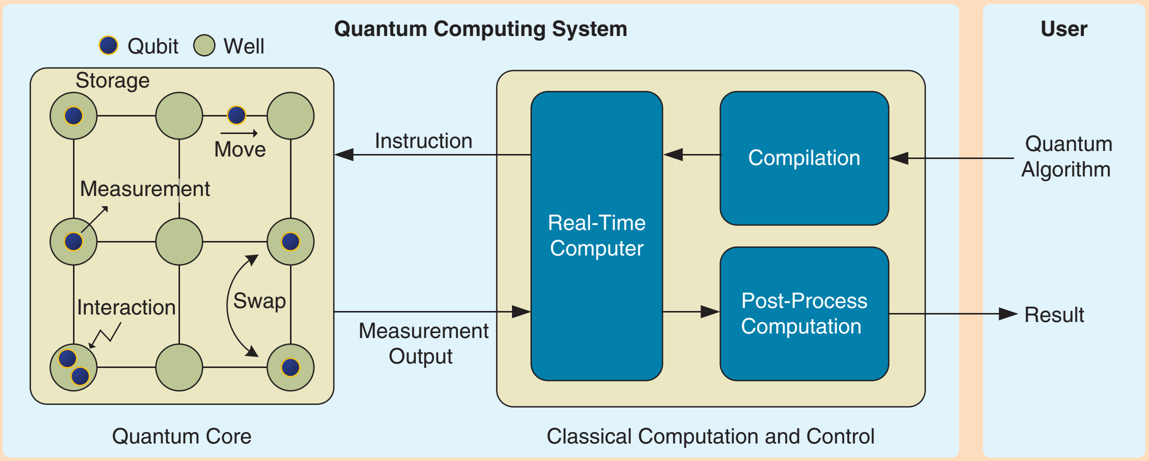


Figure 1. A high-level view of a quantum computing system. The quantum algorithm which is written using a quantum programming language such as [8] is compiled to a sequence of PMD-specific instructions. These instructions are issued step by step on the qubits in the quantum core. Quantum technologies differ in terms of the realization of the quantum core (cf. Figure 2 for four examples). Furthermore, for more information about the quantum compilation process please refer to [9].

M. Pedram and A. Shafaei are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA (e-mails: pedram@usc.edu; shafaeib@usc.edu).

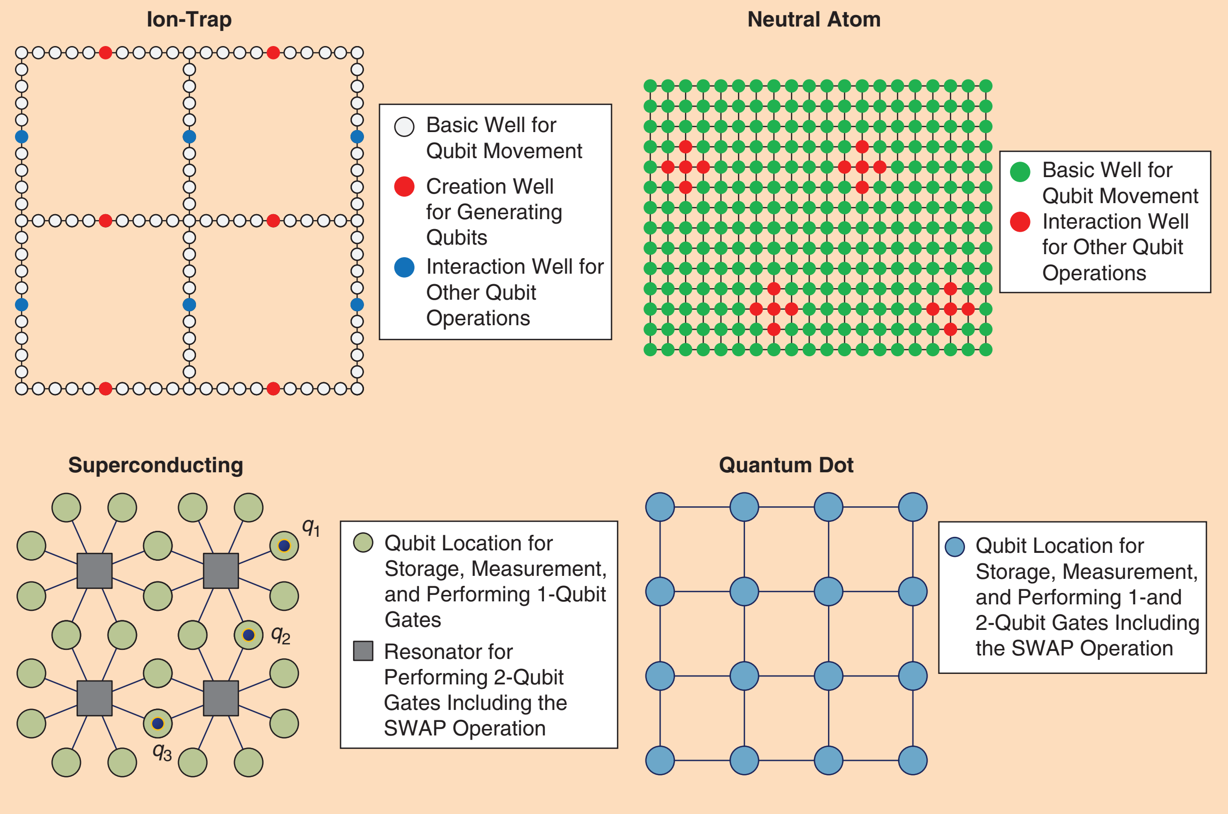

Ion-Trap
Basic Well for Qubit Movement
Creation Well for Generating Qubits
Interaction Well for Other Qubit Operations
Neutral Atom
Basic Well for Qubit Movement
Interaction Well for Other Qubit Operations
Superconducting
q_1
q_2
q_3
Qubit Location for Storage, Measurement, and Performing 1-Qubit Gates
Resonator for Performing 2-Qubit Gates Including the SWAP Operation
Quantum Dot
Qubit Location for Storage, Measurement, and Performing 1-and 2-Qubit Gates Including the SWAP Operation

Figure 2. Ion-trap (top-left), neutral atom (top-right), superconducting (bottom-left), and quantum dot (bottom-right) PMDs [29]. Interaction in afore-said technologies is limited to adjacent qubits. Ion-trap and neutral atom PMDs include an explicit qubit MOVE instruction. However, superconducting and quantum dot PMDs lack such an instruction; instead qubit routing is achieved via SWAP gates. In superconducting technology, all qubits connected to the same resonator are adjacent (e.g., q_1 and q_2, or q_2 and q_3). Accordingly, if a two-qubit gate involves q_1 and q_3, we may first swap q_1 and q_2 and then perform the gate at the bottom-right resonator.

Environmental disturbances and errors in the control systems are two common examples of noise, which if ignored, can rapidly result in computational error. The error rate limits the computation length. In addition, quantum technologies are subject to additional constraints on e.g., degree of parallelism, type of connectivity, and connection bandwidth. Such constraints further constrain the realization of quantum algorithms as quantum circuits.

Current technologies for quantum computing often need gates that involve geometrically *adjacent* qubits. The architecture of a quantum computing system can be described by a simple connected graph $G = (V,E)$ where vertices V represent qubits and edges E represent adjacent qubit pairs where gates can be applied on [10]. Accordingly, a complete graph expresses the absence of any *constraints*. Quantum algorithms usually consider no interaction constraint between qubits. However, physical implementation may impose additional geometrical constraints. Therefore, the developed quantum algorithms or quantum circuits should be modified to consider the effect of various technological limitations.

Quantum computation technologies arrange qubits of a physical layout in a one (1D), two (2D), or three (3D) dimensional architecture.[2] The *linear nearest neighbor* (LNN) architecture corresponds to a graph where edges exist only between neighboring vertices in a line. *Two-dimensional square lattices* (2DSL) corresponds to a graph on a Manhattan grid with four neighboring qubits. The *three-dimensional square lattices* (3DSL) model is a set of stacked 2D lattices with six neighboring qubits. Generally, 3DSL is less restrictive. However, it can suffer from the difficulty of controlling 3D qubits. Several quantum computing systems of trapped ions [11] have been designed based on the interactions in a line. 2DSL proposals include

[2]Quantum technologies proposed mainly for quantum communication, such as photon-based model, is not considered here.

It has been shown that quantum computing can offer significantly higher performance on certain problems compared to classical computing.

arrays of trapped ions [11] and Josephson junctions [12]. The architecture in [13] is based on the 3DSL model.

Exploring an efficient realization of a given quantum algorithm or quantum circuit for a restricted architecture has been pursued by many researchers in the recent past. Physical implementation of the quantum Fourier transformation (QFT) [14], [15], Shor's factorization algorithm [16]–[18], quantum addition [19], quantum error correction [20], and general reversible circuits [21] for the LNN/2DSL architectures have been explored in the past. Worst-case synthesis cost of a general/Boolean unitary matrix under the nearest neighbor restriction has been discussed in [22]–[25]. In [26], [27] heuristic methods for converting an arbitrary quantum circuit to its equivalent circuit on the LNN architectures have been presented.

In this work, we model the problem of improving *locality*, i.e., reducing *interaction distance*, of a given quantum circuit by graph theory. Precisely, we use the *minimum linear arrangement* (MINLA) problem in graph theory to find a *localized quantum computation kernel* as a subcircuit with upper bounded SWAP gate count in architectures where qubits are arranged in a line and any reordering of qubits occurs only by utilizing SWAP gates. We also discuss and solve the problem of how one can connect these subcircuits to minimize the same cost function. A preliminary version of this paper appeared in [28].

The rest of this paper is organized as follows. In Section II, basic concepts relate to quantum computing and quantum physical fabric descriptions and constraints are provided. Prior work is discussed in Section III. Section IV describes the proposed approaches for locality improvement of quantum circuits. Experimental results are given in Section V, and finally Section VI concludes the paper. We also discuss how the proposed MINLA-based technique can be generalized to 2D quantum architectures.

II. Basic Concepts

In the following subsections, we briefly discuss related concepts in quantum circuits and quantum architectures.

A. Quantum Gates and Circuits

A quantum bit, *qubit*, can be considered as a mathematical object which represents a quantum state with two basic states $|0\rangle$ and $|1\rangle$. In addition to the selected basis, a qubit can get any linear combination of its basic states. A quantum system which contains n qubits is often called a *quantum register* of size n. An n-qubit *quantum gate* performs a specific $2^n \times 2^n$ unitary operation on selected n qubits. The unitary matrix implemented by several gates acting on different qubits independently can be calculated by the tensor product of their matrices. Two or more quantum gates can be cascaded to construct a *quantum circuit.*

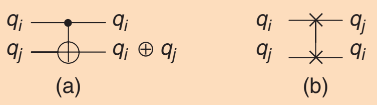


Figure 3. CNOT (left) and SWAP (right) gates. Both gates operate on two qubits.

For a cluster of k gates $g_1, g_2, \ldots, g_k$ cascaded in a quantum circuit C in sequence, the matrix of C can be calculated as $M_k M_{k-1} \cdots M_1$ where M_i is the matrix of the i-th gate $(1 \le i \le k)$. Given any unitary U over m qubits $|x_1 x_2 \cdots x_m\rangle$, a controlled-$U$ gate with k control qubits $|y_1 y_2 \cdots y_k\rangle$ may be defined as an $(m+k)$-qubit gate that applies U on $|x_1 x_2 \cdots x_m\rangle$ iff $|y_1 y_2 \cdots y_k\rangle = |11 \cdots 1\rangle$. For example, CNOT is the controlled-NOT with a single control, and Toffoli is a NOT gate with two controls. A multiple-control Toffoli gate C^{k}NOT is a NOT gate with k controls. In circuit diagrams, circuit symbol • is used for conditioning the output(s) of a controlled gate on the corresponding qubit being cluster to value one. A SWAP gate maps $|ab\rangle$ into $|ba\rangle$. We use symbol $\times$ on qubits of a SWAP gate in circuit diagrams. Figure 3 shows circuit diagrams of CNOT and SWAP gates.

B. Physical Layout

In a particular realistic physical layout, e.g., in an ion-trap quantum architecture [30], each qubit has a specific *physical* location at each time step. To apply a two-qubit gate that uses *mobile* qubits, both qubits should be available at one location.[3] This is done by routing qubits from one physical location to another during the computation. Based on how the qubit routing is handled, quantum technologies are classified as: (i) MOVE-based PMDs which include communication channels and an explicit "MOVE" operation for qubit routing,

[3]Quantum technologies with constantly moving phenomena (e.g., photons) use "flying" qubits. In this case, gates are fixed and qubits are affected upon flowing through the gates. In quantum technologies with physical locations for qubits, gates are applied in fixed locations and "mobile" qubits may travel between locations. More detail is in [31].

Environmental disturbances and errors in control systems are two common examples of noise, which if ignored, can result in computational errors in a quantum computer.

and (ii) SWAP-based PMDs which do not support a MOVE instruction, and the qubit routing is achieved by issuing a sequence of "SWAP" operations (i.e., exchanging the physical location of two adjacent qubits at one time step).

If all gates use adjacent qubits, physical implementation can be done with no further effort; otherwise, additional MOVE or SWAP operations are needed. In the following, we use the term "localized gates" to refer to quantum gates operating on adjacent qubits and "nonlocalized gates" to refer to quantum gates operating on non-adjacent qubits (i.e., requiring crossing of the corresponding input qubits before it can operate on them). A given quantum circuit may contain both localized and nonlocalized gates. However, by inserting a number of SWAP gates at the right places in the circuit, it is possible to produce a quantum circuit where all gates are localized. We call such a circuit a "fully-localized circuit". Note that our goal is to transform an arbitrary quantum circuit into one that is fully-localized and contains the minimum number of SWAP gates.

For a given quantum circuit, an initial qubit ordering $1, 2, \dots, n$ is usually assumed independent of the circuit. This qubit ordering reflects locations of qubits in the physical layout, i.e., qubit $\#i$ is assigned to the i-th physical location ($1 \le i \le n$). However, for a given quantum circuit this arrangement may not be the best in terms of the resulting localized gates and hence the circuit latency. In particular, after arranging qubits, gates should be applied to the corresponding qubits. This may require movement of qubits from one physical location to another. Latency is defined as the total number of time steps required to perform all gate operations in a quantum circuit. This time includes the time required to move qubits between gate operations, as well as the time required to apply gate operations on the corresponding qubits.

To improve the total latency, or equivalently increase the number of local two-qubit gates, one can *globally* reorder qubits to change their initial physical locations as $l_1, l_2, \dots, l_n$ for $1 \le l_i \le n$. Global reordering consumes no additional gates. To illustrate this concept, consider a sample circuit in Figure 4(a) which is the X error syndrome using Shor-EC method for $[[9,1,3]]$ code [32]. The resulting circuit after applying global reordering is shown in Figure 4(c). As can be seen, the original circuit has six nonlocalized gates, which is then reduced to four nonlocalized gates in the reordered circuit. Since localized gates use adjacent qubits, there is no need to move (or swap) the involved qubits before applying the gate. Hence, the communication overhead is reduced, which in turn decreases the circuit latency.

The problem is that even after global reordering, many nonlocalized gates can exist in the circuit. In this case, one needs to add additional MOVE or SWAP operations to *move* or *permute* qubits, respectively, such that the qubits that are involved in the originally nonlocalized gates will become adjacent. This is called *local qubit reordering*. Note that after a local qubit reordering, qubit locations will be changed and dealing with the remaining gates in the circuit may require additional re-orderings. This will be done by using additional SWAP gates.

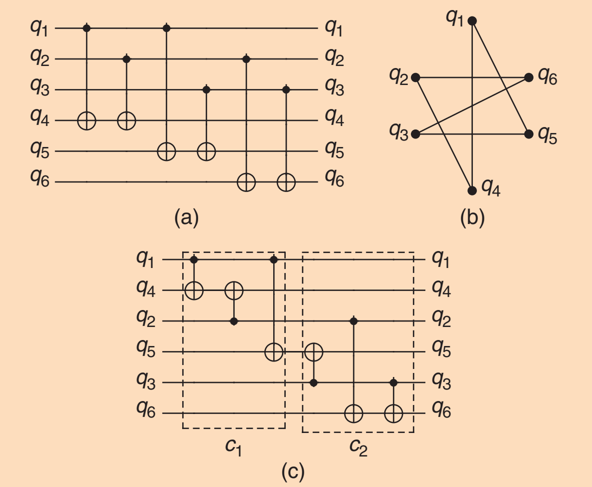


Figure 4. (a) A sample circuit, and (b) its interaction graph. (c) Circuit in (a) after applying global reordering.

C. Quantum Physical Design

Quantum physical design is a process that maps a given quantum netlist to a specific PMD. We assume that the netlist has been decomposed into one- and two-qubit gates. The quantum physical design is then comprised of the following three steps:

- **Scheduling:** Quantum gates are scheduled in order to find all gates that can be run in parallel. In other words, given a *quantum instruction dependency graph* [33], the scheduling problem is

Quantum circuit technologies are subject to additional constraints on e.g., degree of parallelism, type of connectivity, and connection bandwidth. Such constraints further constrain the realization of quantum algorithms as quantum circuits.

to reduce the resource pressure (e.g., the number of concurrent gates) while minimizing the total latency of the circuit.

- **Placement:** Given a total ordering of the instructions on the circuit as the output of the instruction scheduling step, the placement problem is to assign qubits and quantum gates to the corresponding wells such that the communication time and/or the total latency of the computation are minimized.
- **Routing:** Given the position of each qubit and the instruction schedule, the qubit routing problem is to minimize the total routing time. In SWAP-based PMDs, this problem is equivalent to minimizing the number of SWAP gates needed for qubit routing.

In this paper, we assume that the netlist has been scheduled, and our focus will be on placement and routing problems. Hence, the input to our problem is a scheduled netlist composed of one- and two-qubit gates. The target quantum fabric is a SWAP-based technology with LNN architecture. Our approach for reducing the communication cost (i.e., the number of added SWAP gates) is through proper qubit placement (ordering) such that frequently communicating qubits are placed as close as possible to each other during different phases of computation. For this purpose, various instances of MINLA, as discussed later, are solved on the netlist. Different steps of the physical design of a SWAP-based PMD for a sample circuit are illustrated in Figure 5. For the physical design of a MOVE-based PMD please refer to [33].

III. Prior Work

For SWAP-based quantum technologies, a straightforward method to overcome the interaction constraints is to insert local SWAP gates in front of a nonlocalized gate to permute lines (qubits) and move the involved lines toward each other. This should be followed by adding SWAP gates after the computation to recover the initial qubit ordering. For specific quantum circuits, one can explore more efficient implementations [14]–[17], [19], [20]. Additionally, one may try to use localized gates "during" the synthesis process [21] instead of trying to reduce the SWAP gate count during a post-synthesis optimization approach. Although this idea seems intriguing, considering locality besides other important metrics during the synthesis of quantum logic circuit can greatly complicate the overall synthesis process. In addition, several researchers investigating the overall impact of the interaction constraints on their synthesized circuits/constructions have reported that the total cost may increase by a *constant* factor (e.g., 10 in [23] and < 2 in [22], [24]).

To work with arbitrary circuits, the authors in [26] presented post-synthesis methods to reduce the number of SWAP gates. Along with three templates, the authors suggested two reordering strategies, global and local, where in the global approach, a qubit with the highest interaction factor is placed at the middle line repeatedly until no further improvement can be achieved. In the local approach, the algorithm inserts SWAP gates only before a nonlocalized gate, and the new ordering is used for the remaining gates.

In [27], the authors showed that the bubble sort algorithm generates the minimum number of SWAPs required to construct an arbitrary permutation of qubits for each gate. They also showed that in an n-qubit circuit, for two qubits of the i-th gate positioned at locations q_1^i and q_2^i, only qubits placed between q_1^i and q_2^i should be considered instead of working with all qubits (i.e., one must consider $|q_1^i - q_2^i|!$ permutations instead of $n!$ permutations). Note that finding the best local orderings for all two-qubit gates needs considering all $|q_1^i - q_2^i|!$ permutations for all gates at the same time (i.e., $|q_1^1 - q_2^1|! \times |q_1^2 - q_2^2|! \times \cdots$). To

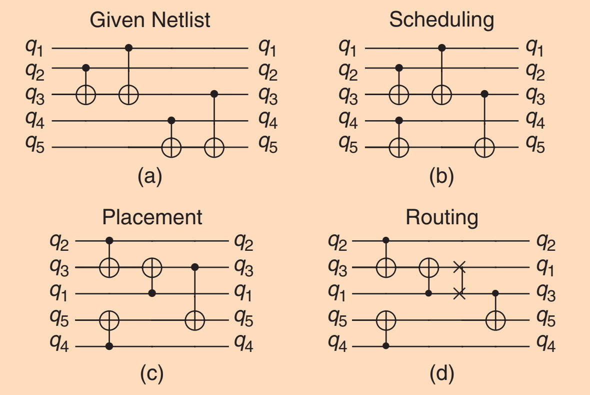


Figure 5. Quantum physical design for a SWAP-based technology: (a) initial circuit, (b) scheduled circuit using the *as soon as possible* (ASAP) gate scheduling, (c) changing the qubit placement (ordering) to reduce the communication cost, (d) inserting SWAP gates for nonlocalized gates. In this example, only the last gate in (d) needs a SWAP gate. The circuit in (d) is fully-localized.

Design of a realistic quantum computer is a highly constrained task that requires careful consideration of many, sometime conflicting, issues.

avoid this huge exponential search, the authors worked with at most k consecutive gates.

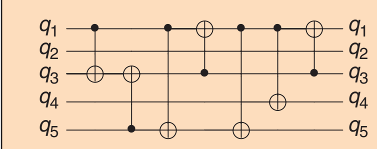
The authors of [34] considered circuits that perform "specified" operations spanning n wires with focus on the circuit depth. They showed that rotation of n wires with localized gates can be done in depth $n+5$, reversing n wires with localized gates is possible with depth $2n+2$, swapping across n wires by localized gates can be done in depth $n+7$ for even n and in depth $n+8$ for odd n requirng $6n-9$ SWAP gates.

IV. The Proposed Method

The MINLA (minimum linear arrangement, also known as optimal linear ordering) problem is defined for a weighted undirected graph $G=(V,E)$. The goal is to generate a linear placement of vertices of V, i.e., arrange the vertices on distinct integer locations on a line by a one-to-one function $f: V \to [1 \cdots |V|]$ to minimize $\sum_{\{u,v\}\in E} w_{u,v}|f(u)-f(v)|$, where $w_{u,v}$ is the weight of the edge between nodes u and v. Without lost of generality, we will assume that the input graph is connected and without self-loops. This problem can be considered as a *label assignment* of the given graph G. The MINLA problem is NP-hard in general [35, Problem GT42]. However, polynomial time algorithms to compute optimal solutions for some particular graphs such as trees, rectangular and square meshes, as well as hypercubes are known [36]. In addition, some approximation algorithms for the MINLA problem have been proposed in the past [36].

MINLA is also connected with graph drawing as described next. A bipartite drawing is a graph representation where the nodes of a bipartite graph are placed on distinct locations on two parallel lines and the edges are drawn with straight line segments connecting the points representing the end vertices of each edge. The *bipartite crossing number* of a bipartite graph is the minimum number of edge crossings over all bipartite drawings of the graph. Shahrokhi et al. [37] have shown that for a large class of bipartite graphs, reducing the bipartite crossing number is equivalent to reducing the total edge length, that is, the bipartite crossing number problem is equivalent to the MINLA problem.

In this paper, the degree of a vertex v in graph G is represented as $\deg(v)$. The maximum degree of a graph G, denoted by $\Delta(G)$, is the maximum degree of any of its vertices, i.e., $\Delta(G)=\max_{v\in V}\{\deg(v)\}$.



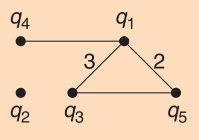


Figure 6. (a) A sample circuit, and (b) its weighted interaction graph.

A. Label Assignment for Qubit Reordering

Consider a given circuit C with n qubits $q_1, q_2, \ldots, q_n$ and m two-qubit gates $g_1, g_2, \ldots, g_m$.[4] Working on C, we construct a weighted graph G, called the *interaction graph*, with n vertices $v_1, v_2, \ldots, v_n$ corresponding to qubits in $\mathbb{C}$. Additionally, we add an edge with weight $w_{i,j}$ between pair of vertices v_i and v_j exactly when there are $w_{i,j}$ two-qubit gates operating on qubits q_i and q_j in C. If $w_{i,j}=0$, we omit the edge. Similarly, for $w_{i,j}=1$, we omit writing the weight value. For instance, Figure 4(b) and Figure 6(b) illustrate the interaction graphs for the circuits in Figure 4(a) and Figure 6(a), respectively.

For a gate g_i with qubits in line numbers x and y (and $x \neq y$), the input qubit *interaction distance* (iDist_i) is defined as $|x-y-1|$. Clearly, gate g_i with $\text{iDist}_i = 0$ is a localized gate (i.e., it requires no qubit reorderings at its inputs). The total interaction distance of a circuit, denoted by iDist, is the summation of interaction distances of all its gates, i.e., $\text{iDist} = \sum_{i=1}^{m} < \text{iDist}_i$. Accordingly, minimizing the total interaction distance in C is equivalent to solving the MINLA problem for G. Figure 4(c) shows the circuit in Figure 4(a) after applying the MINLA algorithm where the initial iDist = 12 in (a) is improved to iDist = 4 in (b). While applying the MINLA problem on the whole circuit minimizes the total interaction distance, some gates may remain nonlocal, as can be seen in Figure 4(c). Next, we show how the MINLA problem can be used to make all gates local.

[4]We ignore single-qubit gates for locality improvement since they have no effect on the circuit locality. Another reason is that single-qubit gates can be absorbed into surrounding two-qubit gates. The resulting circuit is called a "skeleton" circuit in [15]. Throughout the paper, we simply suppose that the given circuit is a skeleton circuit.

Current technologies for quantum computing often need gates that involve geometrically adjacent qubits. This means that distant qubits have to be brought together busing either SWAP or MOVE gates.

B. Proposed Algorithm

Our proposed algorithm for the SWAP minimization of LNN quantum architectures is shown in Algorithm 1. Similar to [38], we decompose the circuit into subcircuits (clusters), solve each subcircuit individually, and connect two consecutive subcircuits via a network of SWAP gates. However, our algorithm has a more global view of the impact of each qubit placement on future gates. Furthermore, we adopt an intelligent clustering approach vs. a greedy breakdown strategy as is done in [38]. Details of our algorithm are explained next.

Decomposing the circuit into clusters

To construct the circuit clusters, starting from the primary inputs of the quantum circuit, we initialize the first cluster with the first two consecutive two-qubit gates in the circuit, and then apply the MINLA problem. If a relabeling with at most s SWAP gates can be found to make all gates in the cluster free of input edge crossings, then a third gate is added to the cluster and the check is repeated. The process is stopped when adding a new gate to the cluster causes a violation of the s bound. The first cluster is then produced by backtracking to the last correct cluster solution. The whole procedure is repeated for the second cluster, etc. until all gates of the circuit have been processed. Later in this section, we will show how the initial collection of clusters produced by the aforesaid procedure can be locally optimized in order to use the lowest total SWAP count.

Algorithm 1:
SWAP minimization of LNN quantum architectures

Input: A scheduled quantum circuit C composed of one- and two-qubit gates.
Output: A fully-localized quantum circuit C' which is functionally equivalent to C.

```
1) Decompose $C$ into $k$ clusters, $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k$, where
   each cluster contains consecutive two-qubit gates;
2) Solve MINLA problem for each cluster $\mathcal{S}_j$ to
   obtain the initial qubit ordering $O_j^i$ for $1 \le j \le k$;
3) Insert intra-cluster SWAP gates for each cluster
   $\mathcal{S}_j$ using the local qubit reordering which results
   in the final qubit ordering $O_j^f$ for $1 \le j \le k$;
4) Insert inter-cluster SWAP gates by simulating the
   bubble sort algorithm to transfer $O_j^f$ to $O_{j+1}^i$ for
   $1 \le j < k$;
```

SWAP gates inside a cluster

Consider an ordered set of r two-qubit gates $A = \{g_1, g_2, \ldots, g_r\}$ successively operating on pairs of qubits in an n-qubit SWAP-based LNN architecture. Assume that the gate g_i works on qubits q_1^i and q_2^i (and $q_1^i \neq q_2^i$). For an interaction graph G, we may encounter the following situations.

- $\Delta(G) = 0$. This is a trivial case with no gates.
- $\Delta(G) = 1$. In this case, all gates use distinct pairs of qubits. Accordingly, one can find a qubit (re) ordering when for each gate g_i, qubits q_1^i and q_2^i are adjacent. To achieve this, group q_1^i and q_2^i as a new qubit for all gates in A, resulting in a total of $n - r$ qubit groups — each group can include either one qubit (if the qubit is not used by any gates in A) or two qubits (if exactly one gate in A uses these two qubits). There are $2^r \times (n-r)!$ qubit orderings to make all gates in A local. No SWAP gates are required in this case.
- $\Delta(G) = 2$ and there is no (undirected) cycle in G. For this case, there is a "staircase" construction where all gates are local. Assume there are k_0 vertices with $\deg(v) = 0$, k_1 vertices with $\deg(v) = 1$, and k_2 vertices with $\deg(v) = 2$ — equivalently k_0 qubits with no interaction, etc. For $k_0 > 0$ or $k_1 > 2$ (k_1 is even) the interaction graph is unconnected and the number of connected components is $k_0 + k_1/2$. To construct a fully-localized circuit, place all qubits (equivalently vertices in G) with $\deg(v) = 0$ close to each other, and remove such vertices from G. For a vertex with $\deg(v) = 1$, place the related qubit at the next available physical location, and remove the vertex and its edge from G. Continue the same approach for the "new" vertex (created after removing the previous vertex) with $\deg(v) = 1$. Apply this approach for all connected components in G. After all, group qubits which belong to one connected component. This leads to $k_0 + k_1/2$ groups in total. Exchanging all physical locations of one group with the ones for another group, in $(k_0 + k_1/2)!$ total ways, has no effect on total interaction distance. Again, no SWAP gates are required in this case.

We model the problem of improving locality, i.e., reducing interaction distance between inputs of a quantum logic gate, in a given quantum circuit.

- In any other non-trivial cases with $\Delta(G) \geq 3$, or $\Delta(G) = 2$ and with cycle(s) in G, at least one SWAP gate is required. Assume that we divide all two-qubit gates of C into k clusters with only localized gates, each cluster with at most s SWAP gates. For $s = 0$, all clusters belong to either case 1 or case 2. Otherwise, one needs to add SWAPs, called **intra-cluster SWAP gates**. If it is not possible to make the current r gates local with s SWAP gates, decrement r to $r-1$ and recheck. If not, re-decrement r and proceed as before. It can be verified that at least for $r = 2$, we can find a fully-localized subcircuit with no SWAP gate, i.e., $s = 0$. A larger s limit leads to a larger r.

Theorem 4.1 (SWAP-free clusters)
In a SWAP-based LNN architecture, ordered cluster $\mathcal{S}$ composed of r two-qubit gates is SWAP-free (i.e., no intra-cluster SWAP gates are needed to make its two-qubit gates local) if and only if the corresponding interaction graph G has no cycles and $\Delta(G) \leq 2$.

Proof
If cluster $\mathcal{S}$ is SWAP-free, then its interaction graph G is comprised of a single path or multiple unconnected paths, with possibly some isolated vertices (cf. Figure 7). Accordingly, the degree of each vertex in G is less than or equal to 2, which proves $\Delta(G) \leq 2$. Further, unconnected path graphs cannot create a cycle, and hence, G has no cycles. On the other hand, for an interaction graph G with no cycles and $\Delta(G) \leq 2$, a SWAP-free circuit can be constructed as was described earlier in this section.

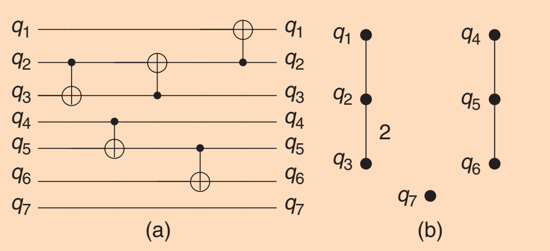


Figure 7. (a) A sample SWAP-free circuit, and (b) its interaction graph composed of two path graphs and an isolated vertex. The interaction graph G of a SWAP-free (sub-)circuit has no cycles and $\Delta(G) \leq 2$.

When $s = 0$, the circuit should be decomposed to SWAP-free clusters. In this case, we begin from the gate at position $i = 0$ and include gates at positions $1, \ldots, j$ until the added gate at position j leads to an interaction graph G with $\Delta(G) > 2$, or creates a cycle in the graph. This will form a SWAP-free cluster based on Theorem 4.1. Then, we close the current cluster without the j-th gate, and restart the same approach from the gate at position j to create a new SWAP-free cluster. This procedure is repeated until the circuit is fully partitioned into a set of non-overlapping clusters.

Intra-cluster SWAP gate count minimization
To find the best possible ordering for each cluster, we use the MINLA problem. This is done by constructing the interaction graph G for gates in each cluster and solving the MINLA problem for each cluster. The exact solution of the MINLA problem for some particular graphs can be found in a polynomial time [36]. Hence, one may be able to find optimal MINLA solutions for many of the clusters. For other graphs, approximate solutions may be used.

The MINLA solution for each cluster produces a linear ordering of the input qubits to the cluster, which is called the *initial qubit ordering* and is denoted by O_j^i for cluster j. Intra-cluster SWAP gates are then inserted using local qubit reordering [26], which identifies the number and position of internal SWAP gates, and finally results in a linear ordering of the output qubits, called the *final qubit ordering* and denoted by O_j^f for cluster j.

SWAP gates between consecutive cluster pairs
In our proposed approach, the MINLA solution and intra-cluster SWAP gates for all defined clusters are found independently of one another. Therefore, assuming that k clusters have been generated, O_j^f may have to be altered in order for it to become the same as the O_{j+1}^i for $1 \leq j < k$. Accordingly, SWAP gates are needed to transform the final qubit ordering of cluster j to the initial qubit ordering of cluster $j+1$ for $1 \leq j < k$. These SWAP gates are called **inter-cluster SWAP gates**. If O_j^f and O_{j+1}^i are the same, no inter-cluster SWAP gates are needed. Working with an unbounded s also leads to no inter-cluster SWAP gates.

Note that methods in [26], [27] are special cases of the method discussed here in the sense that these prior work references assume each cluster contains

The minimum linear arrangement problem in graph theory is a highly effective optimization technique in quantum circuit architectures where qubits are totally ordered as line segments and any reordering of qubits occurs only by utilizing SWAP gates.

one two-qubit gate, resulting in exactly m clusters for a circuit with m two-qubit gates. In other words, in their works, only inter-cluster SWAP gates are used to construct localized gates.

Inter-cluster SWAP gate count minimization

After constructing the circuit clusters and determining the optimal input and output qubit orderings for each cluster as described above, we make use of [27, Theorem 1] to find the minimum number of inter-cluster SWAP gates. More specifically, this method receives two different qubit orderings, namely O_j^f and O_{j+1}^i, and returns the minimum number of SWAP gates needed to transfer O_j^f to O_{j+1}^i by simulating the bubble sort algorithm.

C. Techniques for Further Reduce the Total SWAP Gate Count

Improving the cluster formations

To minimize the total number of SWAP gates in a quantum circuit, one should judiciously set the value of s, which denotes the maximum number of allowed intra-cluster SWAP gates. To expand the search space, we allow each cluster to have a different s value, and hence, we use s_i to denote the maximum number of intra-cluster SWAPs in cluster i. The value of s_i affects number of gates included in cluster i as well as the initial and final qubit orderings of that cluster. In addition, it affects the composition of any subsequently formed cluster(s). Figure 8 illustrates this effect with an example. Therefore, an optimization problem must be set up and solved in order to determine s_i values.

Consider an n-qubit circuit $\mathbb{C}$ with R two-qubit gates decomposed into k clusters, where each cluster contains r_i consecutive gates. Clearly, $\Sigma_{i=1}^{k} r_i = R$. The initial circuit decomposition into k clusters (and hence the initial r_i values) are obtained by assuming a fixed s value for all clusters, as was explained earlier. To improve upon this initial decomposition, we start a local neighborhood search procedure whereby the move set is defined as (i) migrating the first gate of cluster i to cluster $i-1$ and (ii) migrating the last gate of cluster $i-1$ to cluster i. The gain of a move is the difference between the initial intra- and inter-cluster SWAP gate count and the final intra- and inter-cluster SWAP gate count of the two involved clusters. We generate a total of m moves for each pair of consecutive clusters. For each such cluster pair, from the m moves, we accept only a subset of 0 to $m^* \leq m$ where the summation of the corresponding move gains is maximum. In the end, each cluster may end up having a different s value.

Lowering the number of required qubit re-orderings at cluster boundaries

The solution to the MINLA problem may not be unique, which means that for each cluster different qubit orderings resulting in the same minimum number of intra-cluster SWAP gates may exist. In this case, the solution that leads to the minimum number of inter-cluster SWAP gates must be selected. More precisely, assuming that the minimum number of inter-cluster SWAP gates between qubit orderings O_j^f and O_{j+1}^i, obtained by simulating the bubble sort algorithm, is $icswap_{j,j+1}^*$, qubit ordering of cluster j directly affects the value of $icswap_{j,j+1}^*$ and indirectly impacts values of $icswap_{i,i+1}^*$ $(i \geq j+1)$. We thus consider a moving window of size $W-1$ clusters that lie

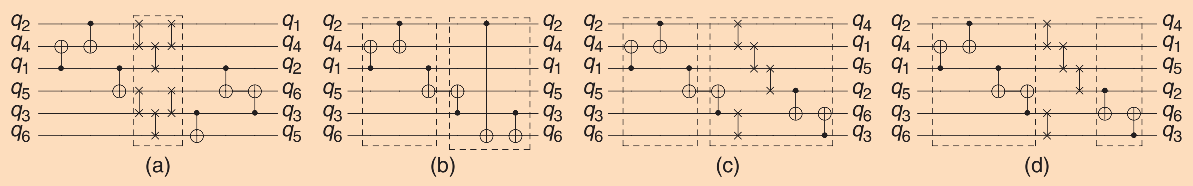


Figure 8. (a) Circuit in Figure 4(c) with only localized gates. In this circuit, our algorithm was applied for subcircuits c_1 and c_2 in Figure 4(c), and six inter-cluster SWAP gates are added and $s_1 = s_2 = 0$. (b) The value of s_i in each cluster affects the total number of SWAP gates. Circuit in (b) has the same functionality in (a) but with $s_1 = s_2 = 4$ (redrawn in (c) with SWAP gates). The second subcircuit in (b) uses intra-cluster SWAP gates as shown in (c). No inter-cluster SWAP gate is used in (c). Note that in (c) one may consider two clusters with four and two gates, shown in (d). In this case, $s_1 = s_2 = 0$, and four inter-cluster SWAP gates are applied. Comparing circuits in (d) and (a) further reveals the effect of selecting appropriate subcircuits on total cost.

ahead of cluster j, i.e., clusters $j+1,\ldots,j+W-1$. Assume that an arbitrary cluster i has M_i different MINLA solutions, where each solution includes an initial and its corresponding final qubit orderings. By enumerating all $\prod_{i=j}^{j+W-1} M_i$ cases, a qubit ordering can be selected for each cluster such that $\sum_{i=j}^{j+W-2} icswap^{*}_{i,i+1}$ is minimized. This will give us the qubit orderings for cluster j. The same process is repeated to find qubit orderings of other clusters. Notice that in order to manage the computational complexity of the above procedure, we only store $\min(p,M_i)$ MINLA solutions for each cluster i, where p is a small integer value.

Table 1.
The synthesis results (# of SWAPs) for benchmarks in [39] as well as for quantum Fourier transform (QFT) circuits after applying the method in [26] and ours. Runtime results (all in second) for [26] vary from ≈ 0 for small circuits to 1300 for large circuits. On average, the results in [26] are improved by 28%. S, m/M/A/T, and imp. represent # of clusters, minimum/maximum/average/total numbers of SWAP gates among pairs of consecutive clusters, and improvement (%), respectively. Benchmarks marked with * are proven to have the minimum number of SWAP gates based on Table 3 of [40].

			Ours			
Circuit	n	Ref. [26]	S	(m,M,A,T)	Time	imp.
3_17_13	3	6	5	(1,1,0.8,4)	0.007	33
4_49_17	4	20	10	(1,2,1.2,12)	0.006	40
4gt10-v1_81	5	30	14	(1,3,1.4,20)	0.013	33
4gt11_84 (*)	5	3	2	(2,2,1,1)	0.01	67
4gt12-v1_89	5	35	23	(1,3,1.5,35)	0.021	0
4gt13-v1_93	5	11	6	(1,2,1,6)	0.11	45
4gt4-v0_80	5	34	16	(1,5,2.2,34)	0.035	0
4gt5_75	5	17	9	(1,2,1.3,12)	0.015	29
4mod5-v1_23	5	16	8	(1,2,1.2,9)	0.015	44
4mod7-v0_95	5	28	15	(1,2,1.4,21)	0.012	25
aj-e11_165	4	39	23	(1,4,1.5,36)	0.018	8
alu-v4_36	5	23	10	(1,5,1.8,18)	0.017	22
decod24-v3_46 (*)	4	4	3	(1,2,1,3)	0.01	25
ham7_104	7	84	32	(1,7,2.1,68)	0.082	19
hwb4_52	4	14	8	(1,2,1.2,10)	0.005	29
hwb5_55	5	79	42	(1,5,1.5,63)	0.037	20
hwb6_58	6	136	54	(1,6,2.1,118)	0.049	13
hwb7_62	7	3660	961	(1,8,2.2,2128)	11.99	42
hwb8_118	8	24541	6205	(1,9,2.3,14361)	389.1	41
hwb9_123	9	36837	7344	(1,14,2.9,21166)	1200	43
mod5adder_128	6	85	31	(1,4,1.6,51)	0.064	40
mod8-10_177	5	77	43	(1,3,1.8,72)	0.02	6
rd32-v0_67 (*)	4	2	3	(1,1,0.6,2)	0.006	0
rd53_135	7	76	29	(1,7,2.3,66)	0.097	13
rd73_140	10	62	22	(1,8,2.5,56)	12.472	10
sym9_148	10	5480	1736	(1,8,1.9,3415)	833.33	38
sys6-v0_144	10	62	21	(1,7,2.8,59)	9.814	5
urf1_149	9	60235	19952	(1,11,2.2,44072)	896.1	27
urf2_152	8	25502	8652	(1,9,2.0,17670)	61.14	31
urf5_158	9	52440	17705	(1,9,2.2,39309)	1191.7	25
QFT5 (*)	5	12	3	(3,3,2,6)	0.008	50
QFT6	6	22	4	(2,7,3,12)	0.053	45
QFT7	7	39	5	(3,10,5.2,26)	0.253	33
QFT8	8	60	5	(5,13,6.6,33)	1.77	45
QFT9	9	87	6	(4,16,9,54)	22.332	38
QFT10	10	123	7	(2,18,10,70)	4.261	43

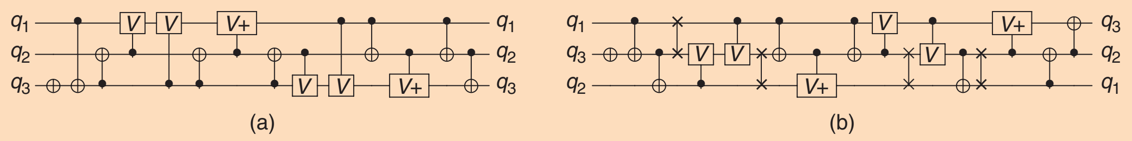


Figure 9. The result of applying the proposed method on the 3_17_13 benchmark. (a) Original circuit, (b) fully-localized circuit. Circuit in (b) is not the optimal solution, since authors of [40] prove that the minimum number of SWAP gates for this benchmark is two.

V. Experimental Results

We have implemented a simple version of the proposed optimization method in C++ – all experiments were done on an Intel Core i7-3770 machine with 16GB memory. In particular, the program initially constructs an interaction graph for the given quantum circuit. Next, it determines the number of clusters and cluster borders in such a way that each cluster is fully-localized and SWAP-free. This step is followed by running the MINLA algorithm on each cluster to find the qubit ordering that achieves SWAP-free realization of each subcircuit. Finally, the algorithm inserts SWAP gates between ever pair of adjacent clusters so as to transform the output qubit ordering of predecessor subcircuit to the input qubit order of the successor one.

To evaluate the proposed interaction distance optimization method, we have compared our results with those obtained by applying the method in [26] for reversible benchmarks in [39] as well as for the quantum Fourier transform circuit. For all cases, the number of SWAP gates added by each method to map the circuit to a LNN quantum architecture was compared. In [26], quantum costs before and after the optimization were reported where SWAP gate count was considered as a unit-cost gate (see [26, Table 3]). Accordingly, the number of SWAP gates is the difference of quantum cost before and after the optimization. Table 1 summarizes the results.

For each circuit in Table 1, in addition to the number of SWAP gates, we have reported the number of clusters and the minimum, maximum and average numbers of SWAP gates in all clusters. Note that we limited the algorithm to use $s = 0$ in each cluster (SWAP-free clusters). Accordingly, no intra-cluster SWAP gate is used and all SWAPs are the result of applying inter-cluster SWAP gate insertion method. We also limited the algorithm to use a window size of $W = 2$ to reduce the runtime. Increasing W improves results at the cost of increased runtime. As can be seen in Table 1, the average number of SWAP gates required to transform a local ordering from one cluster to another cluster is small, and our algorithm leads to a considerable reduction in the number of SWAP gates — 28%, on average and up to 60% compared to prior work. Figure 9 and Figure 10 illustrate the results of applying the proposed method on two benchmarks. In these circuits, all SWAP gates are inserted between consecutive clusters.

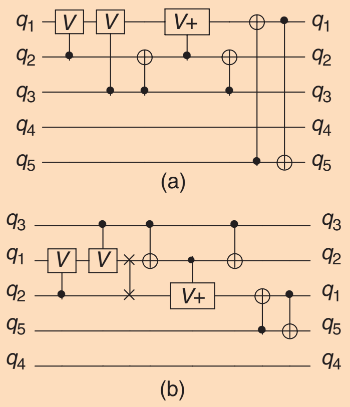


Figure 10. The result of applying the proposed method on the 4gt11_84 benchmark. (a) Original circuit, (b) fully-localized circuit (based on [40], this is an optimal solution).

VI. Conclusion

In this paper, the interaction distance constraint in SWAP-based quantum architectures with 1D interactions was optimized. We modeled all interactions among qubits of a given circuit by introducing an interaction graph, and found local qubit reorderings to minimize the number of required SWAP gates. The proposed approach decomposes a given circuit into several subcircuits (also called clusters) and uses the MINLA instances within each subcircuit to find qubit locations for qubits involved in each subcircuit. Next, (intra-cluster) SWAP gates are inserted inside each subcircuit to localize the remaining nonlocalized gates. Finally, additional (inter-cluster) SWAP gates are inserted between consecutive subcircuits to transform one qubit ordering to another to keep circuit functionally unchanged. The proposed approach applies a lookahead to determine subcircuit borders and to minimize the total number of SWAP gates.

In addition to considering circuit depth and improving runtime to handle larger circuits, a possible future research is to consider the interaction distance constraint in 2D quantum architectures. This path has been followed for specific circuits in the past e.g., [18]. However, the case of general circuits needs attention.

For architectures that do not support MOVE and use one physical location per qubit, we can extend MINLA to the 2-dimensional grid arrangement problem [41]. New methods are required to insert intra-cluster SWAPs and to determine initial qubit locations.

For quantum architectures that support MOVE and may hold several qubits in one physical location, our approach should be appropriately modified. In this case, one can construct a graph for physical locations (vs. qubits) and use one node for all qubits occupying the same location. This can be followed by a 2-dimensional grid arrangement problem to determine a qubit ordering. In addition to these challenges, the algorithm should handle the maximum size (i.e., qubit capacity) of intermediate and final physical locations when a qubit is passing from one physical location to another. The method presented in [38] is a related approach.

Acknowledgment

It is our honor to dedicate this paper to the memory of Professor Ernest Kuh. Our work was made possible (because of the training that Massoud Pedram received in physical design algorithms and automation at UC Berkeley with Ernest Kuh as his PhD advisor) and inspired by the seminal work of Ernest Kuh in this area.

The authors would like to thank Dr. Mehdi Saeedi for helpful discussions and contributions to an early version of this work. This research was supported by a grant from the Quantum Computer Science Program of the Intelligence Advanced Research Projects Activity (contract number D11PC20165).

References

[1] Stephen Jordan. Quantum Algorithm Zoo. [Online]. Available: http://math.nist.gov/quantum/zoo/

[2] D-Wave Systems Inc. The D-Wave 2X^{TM} System. [Online]. Available: http://www.dwavesys.com/d-wave-two-system

[3] V. S. Denchev *et al.*, "What is the computational value of finite range tunneling?" *arXiv:1512.02206,* 2015.

[4] Rambus Inc. Rambus Explores Future Memory Systems. [Online]. Available: http://www.rambus.com/rambus-explores-future-memory-systems/

[5] IEEE Spectrum. Australians Invent Architecture for a Full-Scale Silicon Quantum Computer. [Online]. Available: http://spectrum.ieee.org/tech-talk/computing/hardware/silicon-quantum-computers-look-to-scale-up

[6] A. Crcoles, *et al.* "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature Communications*, vol. 6:6979, Apr. 2015.

[7] T. D. Ladd, et al. "Quantum computers," *Nature*, vol. 464, no. 7285, pp. 45–53, 2010.

[8] A. J. Abhari, et al., "Scaffold: Quantum programming language," Department of Computer Science, Princeton University, Tech. Rep. TR-934-12, June 2012.

[9] M. J. Dousti, A. Shafaei, and M. Pedram, "Squash 2: a hierarchical scalable quantum mapper considering ancilla sharing," *Quant. Inf. Comput*, vol. 16, no. 3&4, pp. 0332–0356, 2016.

[10] D. Cheung, D. Maslov, and S. Severini., "Translation techniques between quantum circuit architectures*," Workshop on Quant. Inf. Proc.*, Dec. 2007.

[11] H. Häffner, et al. "Scalable multiparticle entanglement of trapped ions," *Nature*, vol. 438, pp. 643–646, Dec. 2005.

[12] B. Douçot, L. B. Ioffe, and J. Vidal, "Discrete non-Abelian gauge theories in Josephson-junction arrays and quantum computation," *Phys. Rev. B*, vol. 69, no. 21, p. 214501, June 2004.

[13] C. A. Pérez-Delgado, et al. "Single spin measurement using cellular automata techniques," *Phys. Rev. Lett.*, vol. 97, no. 10, p. 100501, Sep. 2006.

[14] Y. Takahashi, N. Kunihiro, and K. Ohta, "The quantum Fourier transform on a linear nearest neighbor architecture," *Quant. Inf. Comput*, vol. 7, pp. 383–391, 2007.

[15] D. Maslov, "Linear depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in finite neighbor quantum architectures," *Phys. Rev. A*, vol. 76, 2007.

[16] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbour qubit array," *Quant. Inf. Comput*, vol. 4, pp. 237–245, 2004.

[17] S. A. Kutin, "Shor's algorithm on a nearest-neighbor machine," *Asian Conf. on Quant. Inf. Sci.*, 2007.

[18] P. Pham and K. M. Svore, "A 2D nearest-neighbor quantum architecture for factoring," *arXiv:1207.6655*, 2012.

[19] B.-S. Choi and R. Van Meter, "On the effect of quantum interaction distance on quantum addition circuits," *J. Emerg. Technol. Comput. Sys*, vol. 7, no. 3, pp. 11:1–11:17, Aug. 2011.

[20] A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, "Quantum error correction on linear nearest neighbor qubit arrays," *Phys. Rev. A*, vol. 69, pp. 042 314.1–042 314.4, 2004.

[21] M. Arabzadeh, et al. "Depth-optimized reversible circuit synthesis," *Quant. Inf. Proc.*, 2012.

[22] M. Möttönen and J. J. Vartiainen, "Decompositions of general quantum gates," *Ch. 7 in Trends in Quantum Computing Research, NOVA Publishers, New York, 2006.* [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0504100

[23] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," *IEEE Trans. CAD*, vol. 25, no. 6, pp. 1000–1010, June 2006.

[24] M. Saeedi, et al. Block-based quantum-logic synthesis," *Quant. Inf. Comput.*, vol. 11, no. 3-4, pp. 0262–0277, 2011.

[25] M. Saeedi, Reversible circuit synthesis using a cycle-based approach," *J. Emerg. Technol. Comput. Sys*, vol. 6, no. 4, pp. 13:1–13:26, 2010.

[26] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quant. Inf. Proc.*, vol. 10, no. 3, pp. 355–377, 2011.

[27] Y. Hirata, et al. "An efficient conversion of quantum circuits to a linear nearest neighbor architecture," *Quant. Inf. Comput.*, vol. 11, no. 1–2, pp. 0142–0166, 2011.

[28] A. Shafaei, M. Saeedi, and M. Pedram, "Optimization of Quantum Circuits for Interaction Distance in Linear Nearest Neighbor Architectures," *Design Automation Conference (DAC),* June 2013.

[29] Georgia Tech Research Institutes Quantum Information Systems Group. Quantum Machine Parameterizer (QMP). [Online]. Available: http://quantum.gatech.edu/qmpManual/index.html

[30] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computers," *Nature*, vol. 417, pp. 709–711, June 2002.

[31] R. Van Meter and M. Oskin, "Architectural implications of quantum computing technologies," *J. Emerg. Technol. Comput. Sys.*, vol. 2, no. 1, pp. 31–63, 2006.

[32] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[33] H. Goudarzi, et al. "Design of a universal logic block for fault-tolerant realization of any logic operation in trapped-ion quantum circuits," *Quantum Information Processing*, pp. 1267–1299, Jan. 2014.

[34] S. Kutin, D. Moulton, and L. Smithline, "Computation at a distance," *Chicago J. of Theor. Comput. Sci.*, 2007.

[35] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*. W. H. Freeman, 1979.

[36] J. Petit, "Experiments on the minimum linear arrangement problem," *J. Exp. Algorithmics*, vol. 8, Dec. 2003.

[37] F. Shahrokhi, et al. "On bipartite drawings and the linear arrangement problem," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1773–1789, 2001.

[38] D. Maslov, S. M. Falconer, and M. Mosca, "Quantum circuit placement," *IEEE Trans. CAD*, vol. 27, no. 4, pp. 752–763, Apr. 2008.

[39] R. Wille *et al.*, "RevLib: An online resource for reversible functions and reversible circuits," *Int'l Symp. on Multiple-Valued Logic,* pp. 220–225, May 2008.

[40] R. Wille, A. Lye, *and* R. Drechsler, "Optimal swap gate insertion for nearest neighbor quantum circuits," in *Asia and South Pacific Design Automation Conference (ASP-DAC),* pp. 489–494, Jan. 2014.

[41] M. Oswald, G. Reinelt, and S. Wiesberg, "Exact solution of the 2-dimensional grid arrangement problem," *Discrete Optimization*, vol. 9, no. 3, pp. 189–199, 2012.