A New Heuristic for *N*-Dimensional Nearest Neighbor Realization of a Quantum Circuit

Abhoy Kole, *Member, IEEE*, Kamalika Datta, *Member, IEEE*, and Indranil Sengupta, *Senior Member, IEEE*

***Abstract*—One of the main challenges in quantum computing is to ensure error-free operation of the basic quantum gates. There are various implementation technologies of quantum gates for which the distance between interacting qubits must be kept within a limit for reliable operation. This leads to the so-called requirement of neighborhood arrangements of the interacting qubits, often referred to as nearest neighbor (NN) constraint. This is typically achieved by inserting SWAP gates in the quantum circuits, where a SWAP gate between two qubits exchanges their states. Minimizing the number of SWAP gates to provide NN compliance is an important problem to solve. A number of approaches have been proposed in this regard, based on local and global ordering techniques. In this paper, a generalized approach for combined local and global ordering of qubits have been proposed that is based on an improved heuristic for cost estimation and is also scalable. The approach can be extended to *N*-dimensional arrangement of qubits, for any arbitrary values of *N*. Practical constraints, however, restrict the maximum value of *N* to 3. Extensive experiments on benchmark functions have been carried out to evaluate the performance in terms of SWAP gate requirements. 3-D organization of qubits shows average reductions of 6.7% and 37.4%, respectively, in the number of SWAP gates over 2-D and 1-D organizations. Also compared to the best 2-D and 1-D results reported in the literature, on the average 8.7% and 8.4% reductions, respectively, are observed.**

***Index Terms*—Genetic algorithm, nearest neighbor (NN), quantum circuit, qubit, SWAP gate.**

I. Introduction

QUANTUM computing has been projected as an alternative to classical computing, that possesses a number of advantages [1]. There are several applications like search and optimization, public-key cryptosystem, solution of a large system of linear equations, etc. where quantum computing can prove to be immensely beneficial. Unlike bits that store information in classical computing systems, quantum computing uses the concept of qubits. The state of a qubit can be expressed as a linear combination of a set of basis states. Also pairs of qubits can interact in complex ways, based on the quantum mechanical properties of superposition and entanglement.

There has been interesting progress in the development of small-scale quantum computing systems over the last decade. One of the fundamental requirements for such development is fault tolerance in computation. This is because quantum states are very fragile and susceptible to environmental interference and decoherence depending on the quantum technology used. Quantum error correcting codes have been proposed by some researchers, which can only address errors like classical bit flip and more importantly phase flip (due to distortion in the superposition of qubits). Experimental observations suggest that interaction between qubits close to each other can reduce computational errors [2]. The limitations of most promising quantum technologies like ion trap [3] and superconducting [4] enable consideration of nearest neighbor (NN) constraint during synthesis. It is imperative that there is the necessity to develop scalable and efficient design and optimization tools [5] that can be used for automated synthesis of functions using reversible and quantum circuit.

A given quantum circuit can be made NN-compliant by inserting SWAP[1] gates wherever necessary. As qubits for a given circuit are fixed at their initial position, it is necessary to exchange states of qubits in order to make qubits involved in certain gate operation adjacent by introducing one or more SWAP gates in between. Each of these SWAP operations add more time steps in the final realization of the circuit. Reduction of the number of SWAP gates is, therefore, important. The design of an optimal NN-compliant quantum circuit by inserting minimal number of SWAP gates is an NP-complete problem [6]. Researchers have tried to address this problem by proposing several heuristic approaches that do not guarantee optimality. The 1-D or linear NN architecture is the simplest where the qubits are ordered linearly along a single dimension. Clearly, every qubit can have at most two adjacent neighbors in this organization. Several approaches have been proposed to address this problem. Exhaustive [7] and exact [8], [9] search approaches are well-suited for smaller quantum circuits. Several heuristic search approaches are reported in [10]–[13] that are scalable and often provide better solutions.

Manuscript received August 31, 2016; revised January 6, 2017; accepted March 21, 2017. Date of publication April 12, 2017; date of current version December 20, 2017. This work was supported by the Department of Science and Technology, Government of India, for the Project "Efficient Multidimensional Qubit Placement in Quantum Circuits," under Grant YSS/2015/001461. This paper was recommended by Associate Editor A. Raychowdhury. *(Corresponding author: Abhoy Kole.)*

A. Kole is with the Department of Computer Application, B. P. Poddar Institute of Management and Technology, Kolkata 700052, India (e-mail: abhoy.kole@gmail.com).

K. Datta is with the Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong 793003, India (e-mail: kdatta@nitm.ac.in).

I. Sengupta is with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India (e-mail: isg@iitkgp.ac.in).

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

Digital Object Identifier 10.1109/TCAD.2017.2693284

[1] A SWAP gate between qubits q_i and q_j exchanges the states of q_i and q_j.

The 2-D architecture, which is compatible with quantum technologies like superconducting [4] and quantum dots [14], allows more number of adjacent neighbors per qubit (namely, at most 4). This may reduce the number of SWAP gates required for NN-compliance. Recently several works have been reported for efficient NN-compliant realization of quantum circuits in the 2-D architecture [15]–[19]. The problem of synthesizing a 2-D NN-compliant quantum circuit is typically approached as a two-step process. First, an initial mapping of qubits in a 2-D grid is identified, which can lead to reduced number of SWAP gates (*global ordering*). Next, SWAP gates are actually inserted in the netlist in order to make the circuit NN-compliant (*local ordering*).

Shafaei *et al.* [15] addressed the problem of 2-D NN-mapping by transforming it into a grid embedding problem. They used mixed integer programming and observed reduction in number of SWAP gates in 2-D NN-mapping compared to corresponding 1-D mapping. Exact approach [16] is not feasible for large circuits and requires excessive computation time. Heuristic approaches [17]–[20] provide improved results and are also scalable. Alfailakawi *et al.* [17] observed improved result by considering all possible moves for inserting SWAP gates before every gate.

Consideration for NN constraint has also been made at the quantum Boolean domain [21]–[23]. In order to reduce synthesis overhead, qubit reordering approaches at the higher level of abstraction are presented in [24] and [25]. To minimize the cost of realization, a gate reordering approach is proposed in [26] that is nonscalable.

In general, both exhaustive and exact approaches are highly expensive and nonscalable whereas the previously introduced heuristic approaches mainly suffer from producing nonoptimal results due to either the metric used in the computation process or some weakness in the strategy itself. Strategies like gate reordering and obtaining optimal minimal subcircuit (e.g., [13] and [15]) seem infeasible as circuit size increases in terms of number of qubits and gates.

In this paper, a new heuristic in association with a general *N*-dimensional NN-mapping approach has been proposed, and evaluated on a number of benchmark functions. The experimental results reveal the following.

1) In 1-D organization, the proposed approach results in an overall improvement of 9% on an average and 54% in the best case over previously reported best results in [7], [11], and [13].
2) As compared to previously reported 2-D NN-mapping results in [15] and [18], the proposed approach with random path selection results in an improvement of 21% and 6%, respectively, on the average and 50% in the best case for less than 16 qubits.
3) Using proposed 2-D NN-mapping for benchmarks with larger number of qubits, an improvement of 41% in the best case and 31% on the average is obtained over [18].
4) Considering all paths in proposed 2-D NN-mapping, an improvement of 20% in the best case and 3% on the average is found over [17].
5) Using proposed approach for 3-D NN-mapping, an improvement of 40% and 4% on an average, and 71% and 41% in the best case is obtained over the respective 1-D and 2-D mapping results, respectively.

The remaining part of this paper is organized as follows. Section II discusses quantum circuits along with cost functions to measure circuit efficiency. Proposal of a new heuristic, and methods adopting the heuristic for global ordering and local ordering are presented in Section III. Results obtained by running the proposed method on benchmarks consisting of NOT, controlled-NOT, and controlled-V/V+ gates (NCV) are reported in Section IV. Finally, in Section V we present concluding remarks and scopes for future work.

II. Background

A. Quantum Circuit

In quantum computation the basic unit of computing is the qubit. Unlike classical computing where a bit can exist in one of two states 0 or 1, a qubit can exist in multiple states that can be expressed as a linear combination (or superposition) of computational basis states $|0\rangle$ and $|1\rangle$, and is characterized by the state vector

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are the amplitudes (complex coefficients), and $\alpha^2 + \beta^2 = 1$. In general, for an n-qubit quantum system the state $|\psi\rangle$ is specified by time-dependent 2^n amplitudes that provide the phase information.

Computation is carried out through the physical implementation of quantum circuits realizing specific quantum algorithms. A quantum circuit consists of a number of qubits and a sequence of quantum gates operating on those qubits. Circuit functionality is realized by evaluating each gate sequentially and updating the states of the interacting qubits accordingly. Reversibility is an inherent characteristic of quantum circuits, and quantum gates operating on qubits are capable of building and manipulating multiqubit nonclassical entangled states adhering to the laws of quantum mechanics. Quantum gates are often applied in implementing a quantum algorithm from a universal gate library. NCV [27] and Clifford+T [1] are the most commonly used libraries at the logical level. In this paper, we restrict our discussion to quantum gates from the NCV library only.

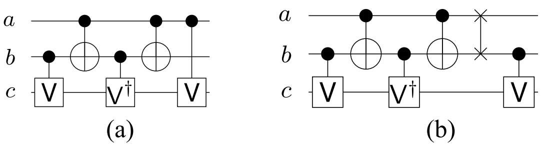
The operation performed by quantum gates on a set of n qubits can be realized by a $2^n \times 2^n$ unitary transformation matrix. The elementary gates, NOT, controlled-NOT (CNOT), controlled-V (CV), and controlled-V† (CV†), from NCV library perform the following unitary phase transformations on a target qubit.

1) The Pauli-X (NOT) operation is defined by the unitary matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

2) The V (square root of NOT) operation is defined by the matrix

$$\mathrm{V} = \frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}.$$



(a) (b)

Fig. 1. Illustration of SWAP gate insertion to achieve spatial locality of qubits. (a) Quantum circuit. (b) NN-compliant.

3) The $\mathrm{V}^\dagger$ (inverse of V) operation is defined by the matrix

$$\mathrm{V}^\dagger = \frac{1-i}{2}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}.$$

The controlled NCV gates perform their desired transformation when their control qubits are in state $|1\rangle$.

Several quantum technologies restrict quantum gates to operate only on physically adjacent qubits. Quantum circuits must satisfy this adjacency constraint to reduce the errors introduced due to environmental imperfections and other sources of quantum noise. In order to achieve spatial locality of qubits for a quantum gate operating on nonadjacent qubits, SWAP gates are inserted before the gate. A SWAP gate that is equivalent to a Fredkin gate [28] with no control qubits interchanges the states of two target qubits as given by the following transformation matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In a quantum circuit, the number of qubits (arranged in 1-D, 2-D, or 3-D configuration) gives a measure of the hardware overhead, while the number of gates gives a measure of delay (i.e., number of time steps required for evaluation) and also the total energy dissipation. Introduction of SWAP gates to make a quantum circuit NN-compliant does not increase the hardware overhead but increases the delay and energy dissipation.

Fig. 1(a) shows a quantum circuit with gates from the NCV library operating on three qubits *a*, *b*, and *c*. The circuit does not satisfy the qubit locality constraint in the last CV gate. An equivalent NN-compliant circuit is obtained by inserting a SWAP gate before the CV gate, as shown in Fig. 1(b), where the SWAP gate is shown as a vertical line between two crosses.

B. Cost Metrics

To estimate the cost of implementing a quantum circuit, various cost metrics have been proposed in the literature. Some of the important cost metrics are briefly explained as follows.

1) Quantum cost (qc) is a widely used cost metric that provides a count of the number of fundamental quantum gates. This metric assumes similar implementation efforts of various elementary gates at the physical level. However, the cost of physical mapping in technologies like ion trap are not same for all gate types, e.g., the construction of Mølmer-Sørensen gate [29], which is equivalent to the CNOT gate, requires application of a pair of single qubit microwave ($\pi/2$) pulse gate and a phase gate operating on both qubits.

   A more appropriate metric considers different costs for the different gate types. This paper concerns with the minimization of the number of SWAP gates (called SWAP count) in an NN-compliant implementation of quantum circuits.

2) The depth of a quantum circuit is another important cost metric that conveys the computational latency and energy requirement of the circuit rather than its size. A SWAP operation can be realized using a cascade of three CNOT gates [1] and physical implementation of a CNOT gate involves distinct two-particle and one-particle operations [30]. The reduction in SWAP gates for NN-compliance also reduces the circuit depth that in turn increases the speed of computation and reduces energy dissipation.

3) The number of additional qubits, i.e., ancilla and garbage qubits (required for higher dimensional qubit organizations) reflects the space requirement, and also the implementation cost at the physical level. Large number of ancilla and garbage qubits increases the overhead of achieving spatial locality of qubits.

4) The dimension of the circuit directly relates to the computational efficiency. Higher dimensional organizations provide greater flexibility in choosing neighboring qubits and thus reduces the number of SWAP gates required. The objective is to minimize the SWAP cost and at the same time reduce the number of dimensions to the extent possible.

In this paper, qc (independent of gate type) is also used for NCV benchmarks on which our experiments are carried out, and to compare our results with previously reported results.

III. Proposed Nearest Neighbor Realization Approach

The proposed qubit mapping approach on an *N*-dimensional grid is discussed in the following sections. We first review the NN cost (NNC) metric that is commonly used by researchers [10] for obtaining a good initial arrangement of the qubits. It may be noted that existing works have addressed qubit arrangements in 1-D and 2-D configurations only. In contrast, the proposed approach can be used for any number of dimensions. We next present a heuristic based on experimental observations that helps us in selecting an initial qubit configuration (IQC), which gives better results as compared to the conventional approaches. Following this approaches for global and local ordering of the qubits are presented, with the objective of minimizing the number of SWAP gates required to make a quantum circuit NN-compliant. The global reordering problem is addressed by both an exhaustive search technique and formulating it for optimization using genetic algorithm. Finally, a local reordering strategy by considering all possible path between interacting qubits is derived. All the presented reordering strategies are based on the same heuristic, also introduced subsequently.

A. Metric for Initial Qubit Ordering

Existing works on realizing NN-compliant quantum circuits basically analyze the frequencies of the interacting qubits in the gate netlist. Based on this analysis, an initial arrangement of the qubits is decided upon. An adjacency graph data structure is typically used where the vertices indicate qubits, and the edges indicate that the corresponding qubits interact in one or more gates. Every edge is also assigned a weight that indicates the number of such gates between the corresponding qubits.

The analysis is based on the NNC estimation [10] that is defined as

$$\text{NNC} = 2 \sum_{q_i, q_j \in Q} W_{q_i q_j} * D_{q_i q_j} \tag{1}$$

where $Q = \{q_1, \ldots, q_n\}$ represents the set of qubits, $W_{q_i q_j}$ denotes the weight of the edge (q_i, q_j), and and $D_{q_i q_j}$ denotes the distance between q_i and q_j for the given arrangement of qubits. For a mapping of qubits arranged on a regular grid in one or more dimensions, $D_{q_i q_j} = md_{q_i q_j} - 1$, where $md_{q_i q_j}$ denotes the Manhattan distance between q_i and q_j. It can be calculated as follows.

1) For linear (1-D) arrangement of qubits

$$D_{q_i q_j} = |x_i - x_j| - 1 \tag{2}$$

where x_i and x_j denote the coordinates of q_i and q_j, respectively.

2) For 2-D mapping of qubits

$$D_{q_i q_j} = |x_i - x_j| + |y_i - y_j| - 1 \tag{3}$$

where (x_i, y_i) and (x_j, y_j) denote the coordinates of q_i and q_j, respectively, in the 2-D configuration.

3) For 3-D mapping of qubits

$$D_{q_i q_j} = |x_i - x_j| + |y_i - y_j| + |z_i - z_j| - 1 \tag{4}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) denote the coordinates of q_i and q_j, respectively, in the 3-D configuration.

The multiplier "2" in (1) takes into account the cost of SWAP gates required to bring interacting qubits in adjacent positions, and also to restore them back in their original positions after the gate operation. Example 1 illustrates the NNC estimation of a quantum circuit for a 3-D arrangement of qubits.

Example 1: Fig. 2(a) shows the NCV gate realization of the benchmark 4gt11_84, while Fig. 2(b) shows the corresponding adjacency graph representation. Typical qubit mappings on 1-D, 2-D, and 3-D networks of sizes 5, 3 × 2, and 2 × 2 × 2, respectively, are shown in Fig. 2(c)–(e). As the netlist consists of five qubits $\{a, b, c, d, e\}$, 1 (g_1) and 3 $(g_1, g_2,$ and $g_3)$ additional garbage qubits are used to map them in the 2-D and 3-D configurations [see Fig. 2(d) and (e)], respectively. According to (1), the NNC for the 1-D qubit arrangement of Fig. 2(c) is $2 * [(W_{ab} * D_{ab}) + (W_{ac} * D_{ac}) + (W_{ae} * D_{ae}) + (W_{bc} * D_{bc})] = 2 * [(2 * 0) + (1 * 2) + (2 * 1) + (2 * 1)] = 12$. Similar calculations show that the NNC values for 2-D and 3-D qubit arrangements are 4 and 2, respectively.

The NNC estimation using (1) gives the required number of SWAP gates to make a given quantum circuit NN-compliant for a given initial ordering of the qubits. However, the estimate as provided by (1) may not always guide us in obtaining the best initial ordering of qubits, as illustrated in Example 2.

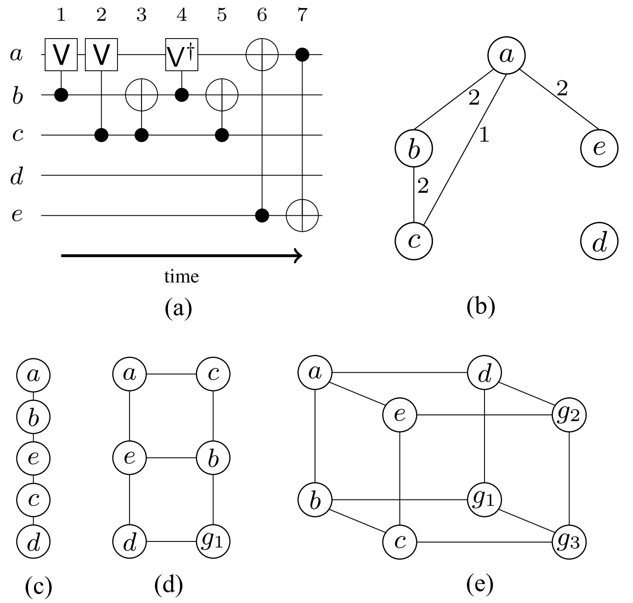


Fig. 2. (a) NCV realization of the benchmark 4gt11_84. (b) Corresponding adjacency graph representation. Qubits mapped in (c) 1-D, (d) 2-D, and (e) 3-D.

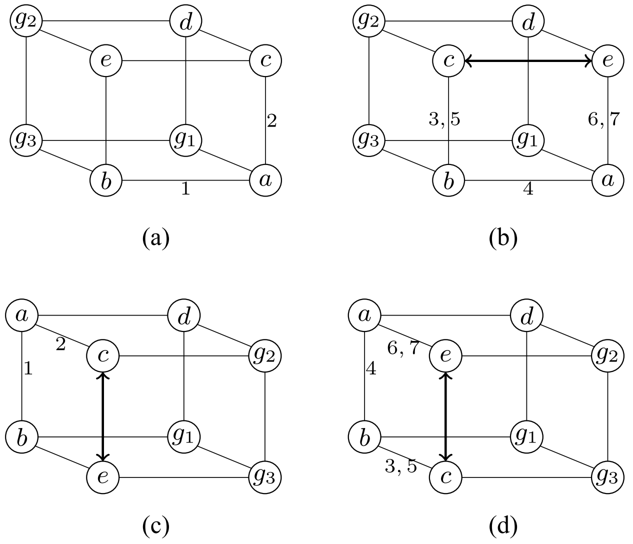


Fig. 3. Illustrative optimal SWAP gate insertion for various 3-D arrangement of qubits of the benchmark 4gt11_84. (a) Initial mapping. (b) Best local ordering scheme before time step 3. (c) Ordering before time step 2 for network shown in Fig. 2(e). (d) Ordering in time step 3 for network shown in Fig. 2(e).

Example 2: Fig. 3 shows various qubit mappings on a 3-D network for the benchmark 4gt11_84, where the labels on the edges indicate gate numbers with respect to Fig. 2(a). Fig. 3(a) shows an initial mapping for which the NNC value calculated using (1) is 8. This is much larger than that for the network shown in Fig. 2(e), which gives an NNC value of 2. The best local ordering scheme for the network requires only one SWAP gate before time step 3 for NN-compliance [see Fig. 3(b)] compared to the network shown in Fig. 2(e) that requires two

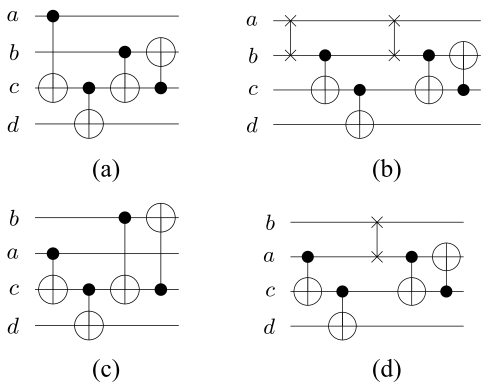


Fig. 4. Global ordering based on priority instead of simple NNC leads to reduced SWAP gate insertion. (a) NNC= 2. (b) SWAP= 2. (c) NNC= 4. (d) SWAP= 1.

SWAP gates [one before time step 2, as shown in Fig. 3(c), and the next one time step 3, as shown in Fig. 3(d)].

This example demonstrates that a qubit ordering with higher value of NNC can give a solution with lower number of SWAP gates, and vice versa.

B. Priority-Based Qubit Ordering

In this section, we present a heuristic that gives a better correlation between the NNC estimate and the actual number of SWAP gates required to make a circuit NN-compliant. The proposed heuristic assigns priority values to the elementary gates in a given quantum gate netlist.

According to the heuristic, a gate appearing earlier in the netlist should have higher priority to be operated on adjacent qubits than all the gates appearing later in the netlist. This can be expressed mathematically as

$$P_i > \sum_{j=1}^{n_i} P_{i+j} \tag{5}$$

where P_i denotes the priority of the ith gate g_i, and n_i denotes the number of gates that follow g_i.

If we incorporate this priority value in the computation of global qubit ordering, we obtain an initial qubit ordering that requires less SWAP gates during local ordering, as illustrated in Example 3.

Example 3: Fig. 4(a) shows a quantum circuit with minimal NNC estimation of 2, which requires 2 SWAP gates for NN-compliance [see Fig. 4(b)]. Incorporating the priority value during global ordering leads to a worse NNC estimation of 4 [see Fig. 4(c)]. However, the resulting qubit arrangement requires only one SWAP gate for NN-compliance [see Fig. 4(d)].

For some nonzero positive integer K, we choose

$$P_i = \frac{1}{T_i^K}, \text{ for } K \geq 1 \tag{6}$$

where T_i denotes the time step at which gate g_i is evaluated. The conjecture *loosely satisfies*[2] (5), i.e., for the proposed priority heuristic given by (5) and (6) does not hold true for all values of K and n. Table I shows for different values of K and corresponding minimal values of n (denoted as n') when (5) fails.

TABLE I
FOR VARIOUS K AND $n' = \text{MIN}(n)$ THAT FAILS TO SATISFY (5)

K	n'	i	P_i	$\sum_{j=i+1}^{n} P_j$	$1 < n < n'$
2	6	3	0.111	0.131	$1 < n < 6$
3	8	4	0.0156	0.0174	$1 < n < 8$
4	9	7	0.00077	0.00080	$1 < n < 9$

For example, if we select $K = 2$ and $i = 3$, (5) fails for $n' = 6$ as

$$P_3 = 0.111 \not> \sum_{j=4}^{6} P_j = 0.131.$$

For $K = 2$, valid (K, $1 < n < n'$) pairs are (2, 2), (2, 3), (2, 4), and (2, 5) that can also be verified from the last column of Table I.

Thus, for a given quantum circuit with n_g gates, by weighing each elementary gate g_i with its assigned priority $(1/T_i^K)$, the edge weight of the corresponding adjacency graph can be defined as

$$W'_{q_iq_j} = \sum_{1 \leq m \leq n_g} \frac{1}{T_m^K} \tag{7}$$

where T_m, $1 \leq m \leq n_g$, denotes the time step at which the mth elementary gate associated with the qubits q_i and q_j is applied. The metric no longer gives an estimation of NNC, but rather gives an IQC value for one or more dimensions. Dropping the constant term 2 from (1) and replacing $W_{q_iq_j}$ by $W'_{q_iq_j}$, the IQC for a given quantum circuit can be computed as

$$\text{IQC} = \sum_{q_i, q_j \in Q} W'_{q_iq_j} * D_{q_iq_j} \tag{8}$$

where $Q = \{q_1, \ldots, q_n\}$ represents the set of qubits, and $D_{q_iq_j}$ denotes the distance between q_i and q_j for a given arrangement of qubits. $W'_{q_iq_j}$ is computed using (7) and $D_{q_iq_j}$ is obtained using (2), (3), or (4) depending on the dimension of the network. The effectiveness of this new metric is illustrated in Example 4.

Example 4: Fig. 5(a) shows the adjacency graph of the benchmark 4gt11_84, where the weights of the edges are computed using (7) with $K = 2$. For example, $W_{bc} = (1/T_3^2) + (1/T_5^2) = (1/3^2) + (1/5^2) = 0.1511$. Using (8), the IQC for the 3-D qubit ordering shown in Fig. 5(b) which is symmetric[3] to the network shown in Fig. 2(d), is computed as $1 * 0.25 = 0.25$. Similarly, for the qubit ordering shown in Fig. 3(a) the IQC value is 0.1993. The IQC values state that the 3-D network shown in Fig. 3(a) is better than the network shown in Fig. 5(b).

For a given quantum circuit, having a corresponding adjacency graph [with edge weight computed using (7)] and a general N-dimensional grid with all qubits mapped properly,

[2] Satisfies only for certain (K, n) value pairs.

[3] Mapping of connected components of the adjacency graph in both networks of similar dimension are identical.

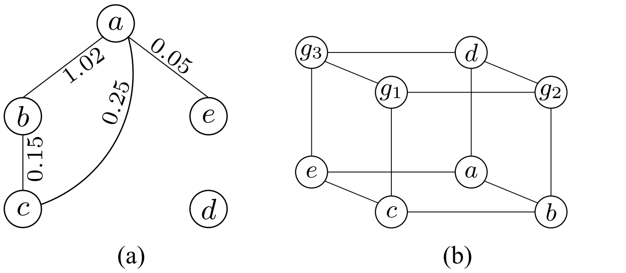


Fig. 5. IQC computation for mapping the qubits of the benchmark 4gt11_84 in a 3-D network. (a) Adjacency graph. (b) 3-D mapping of qubits.

Algorithm 1: Computation of IQC for a Given Qubit Mapping in an *N*-D Grid

```
input  : A mapped N-D grid, $G(m_N \times m_{N-1} \times \cdots \times m_1)$
         An adjacency graph of the circuit, $AG$
output : The cost of mapping, $cost$
$cost \leftarrow 0$;
for $i_N \leftarrow 0$ to $m_N - 1$ do
    $\vdots$
    for $i_1 \leftarrow 0$ to $m_1 - 1$ do
        for $i_N^{'} \leftarrow i_N$ to $m_N - 1$ do
            $\vdots$
            for $i_1^{'} \leftarrow (i_1,$ if $i_N^{'} = i_N \wedge \cdots \wedge i_2^{'} = i_2$ else $0)$ to
            $m_1 - 1$ do
                if $G_{i_N,\ldots,i_2,i_1} != GR \wedge G_{i_N^{'},\ldots,i_2^{'},i_1^{'}} != GR$ then
                    $md^{'} \leftarrow md(\{i_1, i_2, \ldots, i_N\}, \{i_1^{'}, i_2^{'},$
                    $\ldots, i_N^{'}\})$;
                    $cost \leftarrow cost + (md^{'} - 1) *$
                    $AG.edge(G_{i_N,\ldots,i_2,i_1}, G_{i_N^{'},\ldots,i_2^{'},i_1^{'}})$;
                end
            end
        end
    end
end
return $cost$;
```

i. The expression (V_1, **if** *condition* **else** V_2) returns V_1 if the *condition* satisfies, otherwise returns V_2
ii. *GR* denotes a general garbage qubit
iii. The *md* function returns the Manhattan distance between 2 points
iv. The *edge* function of the adjacency graph *AG* returns the weight of the edge presented by 2 qubits

the IQC can be obtained using the procedure outlined in Algorithm 1. The algorithm takes an *N*-D grid and an adjacency graph as inputs and computes the IQC value for mapping of qubits in the grid specified.

C. Global Ordering

The initial mapping of qubits in an *N*-D grid affects the SWAP gate requirements as we have seen in the previous section. Obtaining the optimal qubit mapping in such a grid is an NP-complete problem [6] and hence a clear candidate for optimization.

In order to validate the accuracy of the IQC metric, an exhaustive search procedure for obtaining an optimal mapping of qubits in an *N*-D grid is presented as Algorithm 2.

Algorithm 2: Exhaustive Search for Optimal Qubit Mapping in an *N*-D Grid

```
input  : A mapped N-D grid, $G(m_N \times m_{N-1} \times \cdots \times m_1)$
         An empty N-D grid, $G^{'}(m_N \times m_{N-1} \times \cdots \times m_1)$
         An adjacency graph of the circuit, $AG$
         Cost, $mcost \leftarrow iqc(G, AG)$
         Initial position, $\{x_1 \leftarrow 0, x_2 \leftarrow 0, \ldots, x_N \leftarrow 0\}$
output : An optimal N-D grid, $G^{'}$
if $x_1 = m_1 \wedge x_2 = m_2 \wedge \cdots \wedge x_N = m_N$ then
    $cost \leftarrow iqc(G, AG)$;
    if $cost < mcost$ then
        $mcost \leftarrow cost$;
        $G^{'} \leftarrow G$;
    end
    return $mcost$;
else
    for $i_n \leftarrow x_N$ to $m_N - 1$ do
        $\vdots$
        for $i_2 \leftarrow (x_2,$ if $i_3 = x_3 \wedge \cdots \wedge i_N = x_N$ else $0)$ to
        $m_2 - 1$ do
            for $i_1 \leftarrow (x_1,$ if $i_2 = x_2 \wedge \cdots \wedge i_N = x_N$ else $0)$ to
            $m_1 - 1$ do
                $exchange(G_{x_N,\ldots,x_2,x_1}, G_{i_N,\ldots,i_2,i_1})$;
                $x_1^{'} \leftarrow (x_1 + 1)\ mod\ m_1$;
                $x_2^{'} \leftarrow (x_2 + 1$ if $x_1^{'} = 0$ else $x_2)\ mod\ m_2$;
                $\vdots$
                $x_N^{'} \leftarrow (x_N + 1$ if $x_{N-1}^{'} = 0$ else $x_N)\ mod\ m_N$;
                $mcost \leftarrow ex\_search(G, G^{'}, AG, mcost,$
                $\{x_1^{'}, x_2^{'}, \ldots x_N^{'}\})$;
                $exchange(G_{x_N,\ldots,x_2,x_1}, G_{i_N,\ldots,i_2,i_1})$;
            end
        end
    end
end
return $mcost$;
```

i. The *iqc* function represents Algo. 1
ii. The *exchange* function interchanges the position of 2 qubits
iii. The *ex_search* function represents Algo. 2 itself

The algorithm takes five parameters, an *N*-D grid with randomly mapped qubits, an empty *N*-D grid to store optimal mapping result, an adjacency graph of the circuit, minimal mapping cost, and the position of a qubit in the grid to start the search, respectively, and produces an *N*-D grid with the qubits optimally mapped as output. The search begins by iterating through all the positions of a given grid (where qubits are initially placed randomly) starting from the position (p) passed as parameter, exchanging each qubit with the qubit at position p before recursively calling itself for next qubit position $p + 1$, and finally restoring the exchanged qubits in their original grid positions. The computation of IQC value is carried out in case p indicates the end of the grid and compared with the previously stored minimum IQC value. In case a better IQC is observed for the current qubit mapping, the minimal grid and IQC values are replaced with current grid and current IQC values.

The algorithm is clearly computationally expensive and infeasible for quantum circuits with large number of qubits.

Algorithm 3: Genetic Algorithm-Based Search for (Near) Optimal Qubit Mapping in an *N*-D Grid

```
input  : An adjacency graph of the circuit, $AG$
         Size of an N-D grid, $\{m_N \times m_{N-1} \times \cdots \times m_1\}$
         Size of population, $POP_{size}$
         Maximum number of generation, $MAX_{gen}$
output : A (near) optimal N-D grid, $G$
$GEN_c[\,] \leftarrow generate\_initial\_population(POP_{size})$;
for $i \leftarrow 1$ to $MAX_{gen}$ do
    $fitness[\,] \leftarrow iqc(G, AG)$ for each $G \in GEN_c[\,]$;
    $sort(GEN_c[\,], fitness[\,])$;
    $fitness\_\%[\,] \leftarrow percent\_fitness(fitness[\,])$;
    $GEN_n[\,] \leftarrow next\_generation(GEN_c[\,], fitness\_\%[\,])$;
    $GEN_c[\,] \leftarrow mutation\_\&\_copy(GEN_n[\,])$;
end
return $GEN_{c,0}$;
```

i. The *generate_initial_population* function generates *N*-D grids of *n* number
ii. The *sort* function rearranges a collection of *N*-D grids based on their fitness value passed both
iii. The $GEN_{c,0}$ denotes the first *N*-D grid from the $GEN_c[\,]$ vector

For such circuits, optimization of initial mapping of qubits is carried out using *genetic algorithm*. Initially for an *N*-D mapping a population of specified size is generated by placing the qubits randomly over the *N*-D grid positions. Then the members of the population are sorted based on their IQC values representing the fitness of each member. A relative fitness of each member is also computed based on the fitness value of entire population. The members of next generation is produced by directly copying a constant number of best fit members from the current generation. The remaining members of next generation are produced either by: 1) performing a crossover operation between a pair of randomly selected members based on their relative fitness (to produce a new pair for the next generation) or 2) generating mutates of randomly selected members. Both operations are carried out with their predefined probabilities and the process of producing population for the next generation (after replacing current generation with the newly generated next generation) continues for a specified number of generations. Finally, the first member of the currently sorted population based on the fitness values is selected as the best solution. The outline of this approach is presented in Algorithm 3.

D. Local Ordering

The insertion of each SWAP gate increases the implementation cost of a given quantum circuit. At the same time it also increases the depth (as each inserted SWAP gate increases the number of operations carried out on a qubit that is involved in the SWAP operation) of the circuit and thus results in longer computation time. Thus, reduction in the number of SWAP gates in an NN-compliant quantum circuit is of concern to reduce implementation cost, execution time and also energy dissipation. The priority heuristic in (5) does not satisfy for all possible (K, n) value pairs as discussed earlier, and selection of this pair also has an impact in minimizing SWAP gate requirements.

Algorithm 4: Local Qubit Ordering by SWAP Insertion in *N*-D Grid

```
input  : A mapped N-D grid, $G(m_N \times m_{N-1} \times \cdots \times m_1)$
         A quantum circuit, $C(g_1, g_2, \ldots, g_n)$
         The priority pair, $(K, n)$
output : An N-D grid mapped NN-compliant circuit, $C$
$\widehat{G} \leftarrow G$;// Global qubit order
for each $g_i \in C$ do
    if $g_i.size() = 2$ then // 2 qubit gate
        $c_{pos} \leftarrow get\_qubit\_pos(G, g_i.control)$;
        $t_{pos} \leftarrow get\_qubit\_pos(G, g_i.target)$;
        if $md(c_{pos}, t_{pos}) = 1$ continue;
        $paths[\,] \leftarrow get\_all\_paths(c_{pos}, t_{pos})$;
        $path \leftarrow rand(paths)$;
        for $i \leftarrow 0$ to $path.length - 2$ do
            $G^{'} \leftarrow G$;
            for $j \leftarrow 0$ to $i \wedge i\ \ != j$ do
                $exchange(G^{'}_{path_j},\ G^{'}_{path_{j+1}})$;
            end
            for $j \leftarrow path.length - 1$ to $i + 2$ do
                $exchange(G^{'}_{path_j},\ G^{'}_{path_{j-1}})$;
            end
            $AG \leftarrow create\_graph(C, g_{i+1}, K, n)$;
            if $cost = 0$ then
                $cost \leftarrow iqc(G^{'}, AG)$;
                $index \leftarrow i$;
            else if $cost > iqc(G^{'}, AG)$ then
                $cost \leftarrow iqc(G^{'}, AG)$;
                $index \leftarrow i$;
            end
        end
        for $j \leftarrow 0$ to $index$ do
            $exchange(G_{path_j},\ G_{path_{j+1}})$;
            $swaps \leftarrow gate\_swap(G_{path_j},\ G_{path_{j+1}})$;
        end
        for $j \leftarrow path.length - 1$ to $index + 2$ do
            $exchange(G_{path_j},\ G_{path_{j-1}})$;
            $swaps \leftarrow gate\_swap(G_{path_j},\ G_{path_{j-1}})$;
        end
        $c_{pos} \leftarrow get\_qubit\_position(G, g_i.control)$;
        $t_{pos} \leftarrow get\_qubit\_position(G, g_i.target)$;
        $g_i.update\_lines(\widehat{G}_{c_{pos}}, \widehat{G}_{t_{pos}})$;
        $C.insert\_before(g_i, swaps)$;
    end
end
return $G$;
```

i. The *get_qubit_pos* function returns the position of a qubit
ii. The *get_all_paths* function returns all the paths between 2 points
iii. The *rand* function returns a randomly selected path from a collection of paths
iv. The *create_graph* function returns an adjacency graph generated from a circuit and the initial gate
v. The *gate_swap* function generates a SWAP gate using 2 points
vi. The *update_lines* function updates the control and target qubits
vii. The *insert_before* function inserts a collection of SWAP gates before an elementary gate

A method for locally ordering the qubits exploiting the proposed IQC metric is presented in Algorithm 4. The algorithm takes a quantum circuit and an *N*-D grid with all qubits from the circuit properly placed. By traversing through the circuit, for each two-qubit gate it computes the Manhattan distance between the corresponding qubit pair. If the distance

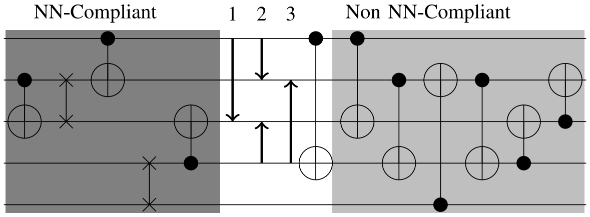


Fig. 6. Illustrative SWAP gate insertion scheme.

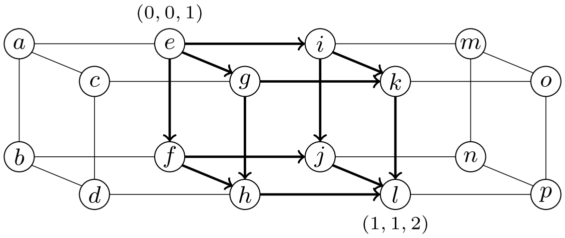


Fig. 7. Illustrative computation of paths in a 3-D grid.

is greater than 1, all the paths between the positions of the two qubits in the grid are identified. Then for a randomly chosen path, SWAP gates are inserted in all possible ways by keeping track of the way that leads to minimal IQC value for the subcircuit consisting of all the remaining gates. There are m ways SWAP gates can be inserted before a quantum gate (where m denoted the Manhattan distance between the qubits) for a given path, as shown in Fig. 6. The number of paths increases rapidly as the circuit dimension increases, and for an N-D ($m_1 \times m_2 \times \cdots \times m_N$) quantum circuit the number of such paths between two points $(i_1, i_2, \ldots, i_N)$ and $(j_1, j_2, \ldots, j_N)$ is

$$\frac{(|i_1 - j_1| + |i_2 - j_2| + \cdots + |i_N - j_N|)!}{|i_1 - j_1|!|i_2 - j_2|! \ldots |i_N - j_N|!}.$$

This is illustrated in Example 5.

Example 5: Fig. 7 shows a 3-D grid. The number of paths between the two nodes a and l at positions $(0, 0, 1)$ and $(1, 1, 2)$ is $3! = 6$, viz. $e \rightarrow f \rightarrow h \rightarrow l$, $e \rightarrow f \rightarrow j \rightarrow l$, $e \rightarrow g \rightarrow h \rightarrow l$, $e \rightarrow g \rightarrow k \rightarrow l$, $e \rightarrow i \rightarrow j \rightarrow l$, and $e \rightarrow i \rightarrow k \rightarrow l$.

As the number of paths grows, the search space for SWAP insertion becomes large (i.e., $m_{\text{path}} \times n_{\text{swap}}$ where m_{path} denotes the number of paths and n_{swap} denotes the number of ways SWAP gate can be inserted in a path). To reduce the search space, the proposed approach selects a path randomly from all possible paths for SWAP gate insertion.

Finally, SWAP gates are inserted before the current quantum gate operating on nonadjacent qubit pair, and the process continues until all the gates from the circuit are traversed.

IV. Experimental Results

The proposed method has been implemented in C++ and run on a core i3 machine with 2.4-GHz clock and 4-GB RAM running Ubuntu 12.04. Experiments were carried out on standard benchmark circuits, and the results compared with previously reported results. The value of K in (7) has been set to 2 for computing of IQC values using (8). Increasing the value of K beyond 2 does not make any difference in SWAP gate

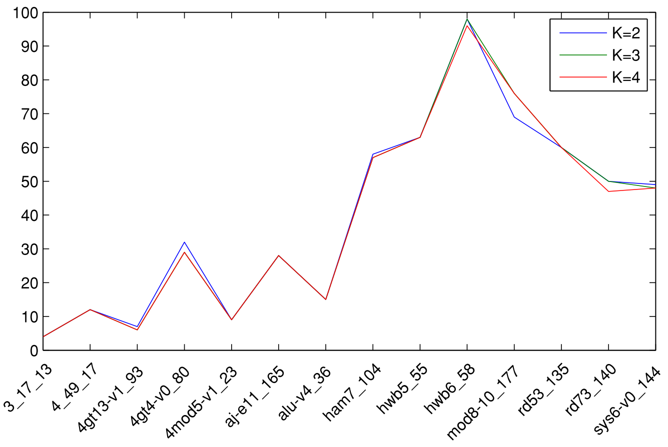


Fig. 8. SWAP gate requirements for NN compliance for various values of K.

requirements, as for $K = 2, 3,$ and 4 (5) is maximally satisfied for $n = 5, 7,$ and 8, respectively. Fig. 8 shows the SWAP gates required to realize various NN-compliant benchmarks for $K = 2, 3,$ and 4 (see Table I). All the experimental results are presented using (K, n) pair of $(2, 5)$.

All the experiments are carried out in the following two steps.

1) Initial qubits are ordered using exhaustive search method if the corresponding grid size is less than or equal to 8; otherwise, genetic algorithm is used (in both cases less than 1-min runtime is observed).
2) Actual SWAP insertion is carried out using the proposed local ordering (the runtime increases as the number of gates in the circuit grows).

To verify the correctness, functional equivalence of the resultant NN-compliant quantum circuit is compared against the original quantum circuit using quantum multiple-valued decision diagram-based method [31].

For genetic algorithm a population size of 30 is used for 500 generation by copying four best members across generations, and performing crossover and mutation among members with probabilities 0.7 and 0.1, respectively.

A. Improvements in 3-D NN-Mapping

All the experimental evaluations have been carried out on the same NCV benchmarks that are used in [15] and presented in Table II. The first three columns gives the benchmark name, number of qubits (n) and number of NCV gates or qc, in order. In the next two columns, the 3-D grid size and required number of SWAP gates for the corresponding NN-mapping using proposed method are presented. Following this, the grid size and SWAP count for the proposed algorithm over 2-D grid are presented. The 2-D grid size and the number of SWAP gates reported in [15] and [18] are presented in the next four columns, followed by comparison with our observed results in the next two columns. The next four columns provide the number of SWAP gates required by running the proposed method for 1-D architecture, the best 1-D result found in the literature, their corresponding references, and our improvements over the previous result,

TABLE II
IMPROVEMENTS IN SWAP GATES OF 3-D QUBIT PLACEMENT OVER 2-D OR 1-D PLACEMENT REPORTED IN [15] AND [18] OR [7] AND [11]–[13]

					2D								1D				$3D$ Impr.	
Benchmarks			Prop. 3D		Prop.		[18]		[15]		Impr.(%)		SWAP		Prev.	Impr.	(%) Over	
Name	n	qc [15]	Grid	SWAP	Grid	SWAP	Grid	SWAP	Grid	SWAP	[18]	[15]	Prop.	Prev.	Ref.	(%)	2D	1D
hwb4_52	4	23	2×2×2	7	2×2	7	2×2	9	2×2	9	22.2	22.2	9	9		0.0	0.0	22.2
aj-e11_165	5	60	2×2×2	15	2×3	24	3×2	22	2×3	24	-9.1	0.0	28	29		3.4	37.5	46.4
4gt12-v1_89	6	53	2×2×2	22	3×2	22	2×4	18	3×2	19	-22.2	-15.8	29	32		9.4	0.0	24.1
4gt4-v0_80	6	44	2×2×2	16	2×3	18	4×4	15	2×3	17	-20.0	-5.9	30.0	33		9.1	11.1	46.7
hwb6_58	6	146	2×2×2	65	2×3	62	2×3	76	2×3	79	18.4	21.5	92	108		14.8	-4.8	29.3
mod8-10_177	6	109	2×2×2	42	3×3	41	4×3	43	3×3	45	4.7	8.9	67	71		5.6	-2.4	37.3
ham7_104	7	87	2×2×2	33	3×3	34	2×7	45	3×3	48	24.4	29.2	56	67	[11]	16.4	2.9	41.1
rd53_135	7	78	2×2×2	34	5×2	40	2×7	40	5×2	39	0.0	-2.6	60	64		6.3	15.0	43.3
hwb7_62	8	2663	2×2×2	1185	3×3	1292	2×6	1500	3×3	1688	13.9	23.5	2101	2157		2.6	8.3	43.6
urf2_152	8	25150	2×2×2	11477	2×4	12532	4×3	16755	2×4	16822	25.2	25.5	16966	17034		0.4	8.4	32.4
urf1_149	9	57770	2×3×2	29343	3×3	29084	3×3	37722	3×3	38555	22.9	24.6	42909	43433		1.2	-0.9	31.6
urf5_158	9	51380	2×3×2	26006	3×3	25645	3×3	34416	3×3	34406	25.5	25.5	37404	38224		2.1	-1.4	30.5
hwb9_123	10	20421	2×3×2	10106	4×3	10749	6×3	11233	4×3	15022	4.3	28.4	19885	20642		3.7	6.0	49.2
rd73_140	10	76	2×3×2	30	4×3	31	3×6	43	4×3	37	27.9	16.2	50	44		-13.6	3.2	40.0
3_17_13	3	14	2×2×2	4	2×2	4	2×2	3	2×2	6	-33.3	33.3	4	4		0.0	0.0	0.0
4_49_17	4	32	2×2×2	9	2×2	9	-	-	2×2	13	-	30.8	12	12		0.0	0.0	25.0
decod24-v3_46	4	9	2×2×2	2	3×2	2	-	-	3×2	3	-	33.3	3	3		0.0	0.0	33.3
rd32-v0_67	4	8	2×2×2	3	2×3	2	-	-	2×3	2	-	0.0	2	2		0.0	-50.0	-50.0
4gt10-v1_81	5	36	2×2×2	12	3×2	13	3×2	15	3×2	16	13.3	18.8	18	20		10.0	7.7	33.3
4gt11_84	5	7	2×2×2	1	2×3	2	2×3	2	2×3	2	0.0	0.0	1	1		0.0	50.0	0.0
4gt13-v1_93	5	17	2×2×2	3	3×3	3	-	-	3×3	2	-	-50	7	6		-16.7	0.0	57.1
4gt5_75	5	22	2×2×2	8	3×3	7	2×5	10	3×3	8	30.0	12.5	9	12	[13]	25.0	-14.3	11.1
4mod5-v1_23	5	24	2×2×2	8	2×3	10	3×2	7	2×3	11	-42.9	9.1	9	9		0.0	20.0	11.1
4mod7-v0_95	5	40	2×2×2	16	3×3	14	2×5	14	3×3	13	0.0	-7.7	19	21		9.5	-14.3	15.8
alu-v4_36	5	32	2×2×2	8	2×3	9	2×5	11	2×3	10	18.2	10.0	15	18		16.7	11.1	46.7
mod5adder_128	6	87	2×2×2	37	3×2	35	2×3	36	3×2	41	2.8	14.6	45	51		11.8	-5.7	17.8
hwb5_55	5	109	2×2×2	43	3×2	44	2×7	49	3×2	45	10.2	2.2	59	63		6.3	2.3	27.1
hwb8_118	9	16610	2×3×2	7482	3×3	8091	4×3	7877	3×3	11027	-2.7	26.6	13796	14361		3.9	7.5	45.8
sym9_148	10	4452	2×3×2	1847	4×4	2065	4×4	2789	4×3	2363	26.0	12.6	4160	3415		-21.8	10.6	55.6
sys6-v0_144	10	62	2×3×2	28	4×4	32	-	-	4×4	31	-	-3.2	49	59		16.9	12.5	42.9
QFT5	5	10	2×2×2	3	3×2	3	4×2	5	3×2	5	40.0	40.0	7	6		-16.7	0.0	57.1
QFT6	6	15	2×2×2	6	2×3	8	3×2	7	2×3	6	-14.3	-33.3	11	12		8.3	25.0	45.5
QFT7	7	21	2×2×2	10	2×4	13	6×2	14	2×4	18	7.1	27.8	28	26		-7.7	23.1	64.3
QFT8	8	28	2×2×2	9	4×2	16	4×2	23	4×2	18	30.4	11.1	32	33		3.0	43.8	71.9
QFT9	9	36	2×3×2	24	3×3	25	5×2	36	3×3	34	30.6	26.5	52	54		3.7	4.0	53.8
QFT10	10	45	2×3×2	33	5×3	43	4×3	51	5×3	53	15.7	18.9	65	70		7.1	23.3	49.2
cnt3-5_180	16	125	3×3×2	57	3×6	67	4×4	84	3×6	69	20.2	2.9	119	127		6.3	14.9	52.1
cycle10_2_110	12	1212	2×3×2	557	3×4	598	4×4	588	3×4	839	-1.7	28.7	1208	2304		47.6	6.9	53.9
ham15_108	15	458	3×3×2	266	5×3	249	3×5	280	5×3	328	11.1	24.1	490	715		31.5	-6.8	45.7
plus63mod4096_163	13	29020	3×3×2	14708	5×3	16072	4×4	13316	5×3	22118	-20.7	27.3	28825	61556	[12]	53.2	8.5	49.0
plus127mod8192_162	14	65456	3×3×2	34597	5×4	37626	-	-	5×4	53598	-	29.8	69731	151794		54.1	8.1	50.4
plus63mod8192_164	14	37102	3×3×2	19503	5×3	21206	3×5	18987	5×3	29835	-11.7	28.9	39350	82492		52.3	8.0	50.4
rd84_142	15	112	3×3×2	49	5×3	64	4×5	62	5×3	54	-3.2	-18.5	122	148		17.6	23.4	59.8
urf3_155	10	132340	2×3×2	68346	4×3	70985	4×4	93558	4×3	94017	24.1	24.5	102694	154672		33.6	3.7	33.4
urf6_160	15	53700	3×3×2	31162	5×3	33152	4×4	42910	5×3	43909	22.7	24.5	54119	88900		39.1	6.0	42.4
Shor3	10	2076	2×3×2	1218	4×3	1210	4×4	1770	4×3	1710	32.5	29.2	2009	1816		-10.6	-0.7	39.4
Shor4	12	5002	2×3×2	3082	3×6	3255	-	-	3×6	4264	-	23.7	4715	5080	[7]	7.2	5.3	34.6
Shor5	14	10265	3×3×2	7034	5×4	7608	-	-	5×4	8456	-	10.0	11457	10760		-6.5	7.5	38.6
Shor6	16	18885	3×3×2	13964	4×6	14801	6×3	19980	4×6	20386	33.0	27.4	28037	20778		-34.9	5.7	50.2

respectively. In the final two columns, the improvement in SWAP count for 3-D mapping over 2-D and 1-D mappings are presented.

For 1-D architecture, the proposed method results in 54.1% improvement in best case and 8.4% on the average over previously reported best results. Similarly, for 2-D architecture the proposed method shows 40% improvement in best case over both [15] and [18], and 8.7% improvement on the average over [18] and 14.2% over [15]. Finally, using 3-D NN-mapping we get best (average) improvements as compared to 2-D and 1-D mapping as 50% (6.7%) and 71.9% (37.4%), respectively. During local ordering, the maximum runtime of 17 h is observed for the largest benchmark *urf*3_155.

B. Scalability of Proposed Method

In order to evaluate the scalability of the proposed approach, experiments are also carried out on larger benchmark circuits. The benchmarks are initially decomposed and NCV mapped using comparatively improved method reported in [32]. The 2-D NN-mapping results obtained using the proposed method are compared with the results reported in [18], and the results of comparison are reported in Table III. The 2-D grid size is kept similar to the one selected in [18].

Benchmark details, viz., name, number of qubits (n), and qc are provided in the first three columns of Table III. In the next three columns the 2-D grid size, number of SWAP gates required using the proposed approach and using the approach in [18] are, respectively, shown. The final column presents

TABLE III
IMPROVEMENTS IN SWAP GATES OF 2-D QUBIT PLACEMENT FOR LARGER BENCHMARKS

Benchmarks				SWAP		Impr.
Name	n	qc [32]	Grid	Prop.	[18]	(%)
add6_196	19	3288	6×4	2159	3081	29.9
add8_172	25	68	4×7	62	84	26.2
add16_174	49	148	6×10	187	231	19.0
add32_183	97	292	11×9	475	681	30.2
alu1_198	20	189	5×4	107	154	30.5
alu2_199	16	3129	6×3	1926	2939	34.5
alu3_200	18	1707	6×3	1076	1460	26.3
apla_203	22	2051	4×6	1291	1797	28.2
bw_291	87	719	11×8	992	1227	19.2
clip_206	14	2847	4×5	1618	2386	32.2
cm85a_209	14	397	3×5	236	358	34.1
cm150a_210	22	614	6×4	351	478	26.6
cm151a_211	28	662	4×7	456	724	37.0
cm163a_213	29	575	8×4	344	567	39.3
cu_219	25	752	6×5	503	686	26.7
dc2_222	15	1218	6×4	757	990	23.5
decod_217	21	845	6×4	485	719	32.5
dk17_224	21	1056	5×5	691	1013	31.8
example2_231	16	3129	6×4	1920	2713	29.2
f51m_233	22	20476	4×6	14136	21486	34.2
ham15_107	15	1101	4×4	595	914	34.9
in0_235	26	11297	5×6	7517	12355	39.2
in2_236	29	11911	5×7	7794	12490	37.6
inc_237	16	1312	4×5	725	1095	33.8
mlp4_245	16	2338	4×5	1361	2065	34.1
mux_246	22	533	4×6	383	415	7.7
plus127mod8192_162	14	34651	6×4	21125	33299	36.6
root_255	13	1539	5×3	865	1391	37.8
ryy6_256	17	2541	4×5	1636	2774	41.0
sqr6_259	18	720	6×5	394	590	33.2
squar5_261	13	318	4×6	172	269	36.1

TABLE IV
IMPROVEMENTS IN SWAP GATES OF 2-D QUBIT PLACEMENT CONSIDERING ALL PATHS

Benchmarks				SWAP		Impr.
Name	n	qc [15]	Grid	Prop.	[17]	(%)
3_17_13	3	14	2×2	4	4	0.0
4gt10-v1_81	5	36	3×2	12	11	-9.1
aj-e11_165	5	60	2×3	19	15	-26.7
cycle10_2_110	12	1212	3×4	457	467	2.1
ham15_108	15	458	3×5	201	199	-1.0
ham7_104	7	87	4×2	34	26	-30.8
hwb5_55	5	109	3×2	33	38	13.2
hwb6_58	5	146	3×2	52	63	17.5
hwb7_62	7	2663	3×3	934	1162	19.6
hwb8_118	8	16610	3×3	5838	6787	14.0
hwb9_123	9	20421	4×3	7914	9230	14.3
mod5adder_128	6	87	3×2	30	28	-7.1
plus63mod4096_163	12	29020	5×3	11901	13031	8.7
plus63mod8192_164	13	37102	5×3	15234	17346	12.2
QFT7	7	21	4×2	10	10	0.0
QFT8	8	28	4×2	15	16	6.3
QFT9	9	36	3×3	20	19	-5.3
QFT10	10	45	5×2	32	31	-3.2
rd53_135	7	78	4×2	29	28	-3.6
Shor3	10	2076	4×3	999	1042	4.1
Shor4	12	5002	3×4	2717	2925	7.1
Shor5	14	10265	3×5	5895	6246	5.6
Shor6	16	18885	4×4	11583	12888	10.1
urf2_152	8	25150	4×2	11388	12982	12.3

improvement in terms of the required number of SWAP gates for 2-D NN-mapping using proposed approach compared to the one reported in [18]. A maximum of 41% and on the average 31% improvement in number of SWAP gates is observed over [18]. The runtime of local ordering never exceeds 3 h for the NCV benchmarks used in 2-D NN-mapping.

C. Optimality of Proposed Method

Alfailakawi *et al.* [17] achieved best results so far by reordering qubits in all possible ways for 2-D NN-mapping. To compare with their improved results the proposed version of 2-D local ordering scheme is customized to make the best path selection among all possible paths (instead of randomly chosen path) leading to minimal IQC value for the remaining subcircuit. Due to higher runtime, the method is executed over some smaller size similar NCV benchmarks [15] that are also used in [17] and the corresponding results are reported in Table IV.

In the first three columns benchmark name, number of qubits (n), and qc are presented in order. The 2-D grid size that are used for both current approach and approach introduced in [17]) are given in the next column. The number of SWAP gates needed by running proposed approach and the corresponding result reported in [17] are provided in the next two columns. The final column gives the improvement in SWAP gate requirement for 2-D NN-mapping using proposed method over that reported in [17]. Compared to [17] an average 3% and up to 20% improvement in required number of SWAP gates for 2-D NN-mapping is obtained. Using the modified approach maximum runtime of 9 h is observed for the benchmark *plus*63*mod*8192_164.

V. CONCLUSION

Mapping qubits in an N-D grid for NN-compliance is an NP-complete problem. In this paper, the problem is addressed by introducing a new heuristic for both global qubit mapping (either using exhaustive search for quantum circuits with relatively smaller number of qubits or using genetic algorithm for larger number of qubits) and local qubit mapping by inserting SWAP gates. The proposed algorithms can be used for any arbitrary N-D NN-mapping ($N = 1, 2, \dots$). Compared to previously reported best results in 1-D and 2-D NN-mapping, on the average 9% and 3% improvements in required number of SWAP gates is observed. In 3-D architecture there are more number of paths compared to 2-D architecture and not all paths result in minimal number of SWAP gates. Also for 3-D architecture, we have only considered the grid size that requires minimal garbage qubits and achieved 4% and 40% improvements, respectively, on the average over 2-D and 1-D architectures.

As heuristic approaches does not guarantee optimal solutions, there is always scope for producing better results by improving the search strategy.

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, Oct. 2000.

[2] H. Häffner *et al.*, "Scalable multiparticle entanglement of trapped ions," *Nature*, vol. 438, no. 7068, pp. 643–646, Dec. 2005.

[3] C. Monroe and J. Kim, "Scaling the ion trap quantum processor," *Science*, vol. 339, no. 6124, pp. 1164–1169, Mar. 2013.

[4] A. D. Corcoles *et al.*, "Demonstration of a quantum error detecting code using a square lattice of four superconducting qubits," *Nat. Commun.*, vol. 6, Apr. 2015, Art. no. 6979.

[5] K. Datta, I. Sengupta, and H. Rahaman, "A post-synthesis optimization technique for reversible circuits exploiting negative control lines," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 1208–1214, Apr. 2015.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1979.

[7] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient conversion of quantum circuits to a linear nearest neighbor architecture," *Quantum Inf. Comput.*, vol. 11, no. 1, pp. 142–166, Jan. 2011.

[8] R. Wille, A. Lye, and R. Drechsler, "Exact reordering of circuit lines for nearest neighbor quantum architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1818–1831, Dec. 2014.

[9] R. Wille, A. Lye, and R. Drechsler, "Optimal SWAP gate insertion for nearest neighbor quantum circuits," in *Proc. ASP Design Autom. Conf.*, Singapore, 2014, pp. 489–494.

[10] M. Alfailakawi, L. Alterkawi, I. Ahmad, and S. Hamdan, "Line ordering of reversible circuits for linear nearest neighbor realization," *Quantum Inf. Process.*, vol. 12, no. 10, pp. 3319–3339, Oct. 2013.

[11] A. Kole, K. Datta, and I. Sengupta, "A heuristic for linear nearest neighbor realization of quantum circuits by SWAP gate insertion using N-gate lookahead," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 6, no. 1, pp. 62–72, Mar. 2016.

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

[13] A. Shafaei, M. Saeedi, and M. Pedram, "Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures," in *Proc. Design Autom. Conf.*, Austin, TX, USA, May 2013, pp. 1–6.

[14] R. Hanson, L. P. Kouwenhoven, J. R. Petta, S. Tarucha, and L. M. K. Vandersypen, "Spins in few-electron quantum dots," *Rev. Modern Phys.*, vol. 79, no. 4, pp. 1217–1265, Oct. 2007.

[15] A. Shafaei, M. Saeedi, and M. Pedram, "Qubit placement to minimize communication overhead in 2D quantum architectures," in *Proc. ASP Design Autom. Conf.*, Singapore, Jan. 2014, pp. 495–500.

[16] A. Lye, R. Wille, and R. Drechsler, "Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits," in *Proc. ASP Design Autom. Conf.*, Jan. 2015, pp. 178–183.

[17] M. G. Alfailakawi, I. Ahmad, and S. Hamdan, "Harmony-search algorithm for 2D nearest neighbor quantum circuits realization," *Expert Syst. Appl.*, vol. 61, pp. 16–27, Nov. 2016.

[18] R. R. Shrivastwa, K. Datta, and I. Sengupta, "Fast qubit placement in 2D architecture using nearest neighbor realization," in *Proc. Int. Symp. Nanoelectron. Inf. Syst.*, Indore, India, Dec. 2015, pp. 95–100.

[19] R. Wille *et al.*, "Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits," in *Proc. ASP Design Autom. Conf.*, Jan. 2016, pp. 292–297.

[20] C.-C. Lin, S. Sur-Kolay, and N. K. Jha, "PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1221–1234, Jul. 2015.

[21] A. Chakrabarti and S. Sur-Kolay, "Nearest neighbour based synthesis of quantum Boolean circuits," *Eng. Lett.*, vol. 15, no. 2, pp. 356–361, 2007.

[22] M. Khan, "Cost reduction in nearest neighbour based synthesis of quantum Boolean circuits," *Eng. Lett.*, vol. 16, no. 1, pp. 1–5, 2008.

[23] M. Perkowski, M. Lukac, D. Shah, and M. Kameyama, "Synthesis of quantum circuits in linear nearest neighbor model using positive davio lattices," *Electron. Energy*, vol. 24, no. 1, pp. 71–87, 2012.

[24] R. Wille, A. Lye, and R. Drechsler, "Considering nearest neighbor constraints of quantum circuits at the reversible circuit level," *Quantum Inf. Process.*, vol. 13, no. 2, pp. 185–199, Feb. 2014.

[25] A. Kole, K. Datta, I. Sengupta, and R. Wille, "Towards a cost metric for nearest neighbor constraints in reversible circuits," in *Proc. Int. Conf. Reversible Comput.*, Grenoble, France, Jul. 2015, pp. 273–278.

[26] A. Matsuo and S. Yamashita, "Changing the gate order for optimal LNN conversion," in *Proc. Int. Conf. Reversible Comput. (RC)*, Ghent, Belgium, 2012, pp. 89–101.

[27] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.

[28] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theor. Phys.*, vol. 21, no. 3, pp. 219–253, Apr. 1982.

[29] T. R. Tan *et al.*, "Multi-element logic gates for trapped-ion qubits," *Nature*, vol. 528, no. 7582, pp. 380–383, Dec. 2015.

[30] J. H. Plantenberg, P. C. D. Groot, C. J. P. M. Harmans, and J. E. Mooij, "Demonstration of controlled-not quantum gates on a pair of superconducting quantum bits," *Nature*, vol. 447, no. 7146, pp. 836–839, 2007.

[31] R. Wille, D. Große, D. Miller, and R. Drechsler, "Equivalence checking of reversible circuits," in *Proc. Int. Symp. Multiple Valued Logic*, Naha, Japan, May 2009, pp. 324–330.

[32] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," in *Proc. Int. Symp. Multiple Valued Logic*, Tuusula, Finland, May 2011, pp. 288–293.

Abhoy Kole (M'16) received the M.Tech. degree in information and communication technology from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2015.

He is currently an Assistant Professor with the B. P. Poddar Institute of Management and Technology, Kolkata, India. He has published nine papers in peer reviewed journals and conferences. His current research interests include reversible and quantum computing, synthesis and optimization of quantum circuits, nearest neighbor realizations, and ternary reversible logic.

Kamalika Datta (M'10) received the M.C.A. degree from the Biju Pattanaik University of Technology, Bhubaneswar, India, in 2006, the M.S. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2010, and the Ph.D. degree from the Indian Institute of Engineering Science and Technology Shibpur, Howrah, India, in 2014.

She joined the National Institute of Technology Meghalaya, Shillong, India, as an Assistant Professor with the Department of Computer Science and Engineering, in 2014. She has published over 48 papers in peer reviewed journals and conferences. Her current research interests include synthesis and optimization of reversible and quantum circuits, memristor-based logic design, and information security.

Indranil Sengupta (SM'13) received the B.Tech., M.Tech., and Ph.D. degrees in computer science and engineering from the University of Calcutta, Kolkata, India, in 1983, 1985, and 1990, respectively.

He joined the Indian Institute of Technology Kharagpur, Kharagpur, India, as a Faculty Member with the Department of Computer Science and Engineering, in 1988, where he is currently a Full Professor, and the former Head of the Department of Computer Science and Engineering, and the School of Information Technology. He has over 28 years of teaching and research experience, guided 20 Ph.D. students, and published over 180 papers in peer reviewed journals and conferences. His current research interests include reversible and quantum computing, very large-scale integration (VLSI) design and test, and network security.

Dr. Sengupta has served as the Program Chair/General Chair in several conferences in the areas of VLSI design and information security.