Nearest Neighbor Mapping of Quantum Circuits to Two-Dimensional Hexagonal Qubit Architecture

Kamalika Datta*, Abhoy Kole§, Indranil Sengupta‡§, Rolf Drechsler*†
*German Research Centre for Artificial Intelligence (DFKI), Bremen, Germany
†Institute of Computer Science, University of Bremen, Germany
‡Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India
§Department of Computer Science and Engineering, JIS University, India
E-mail: kamalika.datta@dfki.de, abhoy.kole@jisuniversity.ac.in, indranil.sengupta@jisuniversity.ac.in, drechsler@uni-bremen.de

***Abstract*—Motivated by physical realization of hexagonal architectures, in this paper we have introduced the hexagonal neighborhood structure for a set of qubits in a two-dimensional (2-D) plane along with a simple measure for estimating neighborhood distance. As compared to the traditional layout of qubits on a 2-D Cartesian plane, the hexagonal arrangement offers greater flexibility in mapping the logical qubits from a quantum circuit into an array of physical qubits. In most of the prior works, the neighborhood cost is estimated as number of *Swap* operations required to ensure that all 2-qubit quantum gates operate on physically adjacent qubits. However, in recent times, *CNOT Templates* have been used to execute 2-qubit gates where the interacting qubits are not adjacent to each other. In this paper we exploit the benefits of *CNOT Templates* in mapping quantum circuits to hexagonal 2-D arrays. We propose an evolutionary algorithm to obtain a good placement of qubits in the hexagonal structure. We also show the benefits of this approach over a simple greedy qubit placement method. We have carried out experiments on a set of benchmark suites to evaluate the efficiency of the proposed approach. The results show an average improvement of 42.9% over a very recent state-of-the-art method.**

***Index Terms*—Quantum circuits, nearest neighbor architecture, hexagonal structure, quantum cost**

I. Introduction

We have witnessed steady progress in the field of quantum computing over the last 30 years, more so with the demonstration of physical quantum computing machines in recent years. Essentially it is based on the principles of quantum mechanics [1], supporting inherent parallelism and the potential to solve many computationally difficult problems [2]. In particular, problems on searching, optimization, new drug analysis, public-key cryptosystem, number theory, etc. can benefit from quantum computing. During the last two decades extensive mathematical foundations have been built for designing quantum algorithms. But the most important challenge has been to built large-scale reliable quantum computers that can run scalable applications. In recent years lots of efforts have been put in for building quantum computers, and various industry giants like IBM, Microsoft, Intel, and Google have come up with medium-scale demonstrable quantum computers.

In quantum computing, quantum bits (or *qubits*) are the fundamental unit of computation that have the unique properties of superposition and entanglement. Various operations can be performed on one or more qubits (called *quantum gate operations*) that change their states. In terms of implementation, the qubits can be considered as the basic unit of hardware while the quantum gates as the logical operations that work on them in the axis of time. Various physical architectures have been proposed [3], [4], which are primarily classified in terms of how the qubits are physically laid out and the constraints that limit their interactions. One major challenge is the effect of noise in quantum gate operations, which restricts operations to be among neighboring qubits only [5]. This is referred to as *Nearest-Neighbor (NN) Constraint*.

Prior works have explored various qubit architectures like one dimensional (1-D) [6]–[9], two dimensional (2-D) [10]–[12], and also three dimensional (3-D) [13], which assume that the qubits are arranged regularly in terms of Cartesian co-ordinate system. The cost metric for evaluation has been typically the number of qubit Swap operations (or Swap gates) required. In a recent work, qubit interactions in hexagonal grid on a 2-D plane have been demonstrated [14]. Motivated by the results, we propose a flexible arrangement of qubits in the hexagonal 2-D plane, and present experimental results on its suitability for nearest neighbor gate operations. A modified 2-D Cartesian coordinate system has been used to enable efficient identification of neighbors and the distance between any pair of qubits. In this work we use CNOT templates [15] instead of Swap gates for making the qubits nearest neighbor. We propose a Genetic Algorithm (GA) based mapping technique for mapping the qubits in the hexagonal grid. The cost of CNOT templates for NN-compliant operations is used as the fitness function. To show the efficiency of the GA-based engine we also implemented a greedy algorithm for placing qubits in a hexagonal grid. The results show that GA based qubit placement is far more efficient than the greedy approach. The proposed hexagonal architecture can also be used for mapping qudits in a multi-valued quantum system.

The rest of the paper is organized as follows. Section II presents the fundamentals of quantum computing and nearest neighbor quantum gate operations. Section III and IV discuss the proposed hexagonal architecture, and the algorithm for mapping quantum circuits for minimizing the nearest neigh-

borhood cost. Here we discuss both the GA based mapping method and the greedy mapping approach. We also show the various operations that are performed during the GA based mapping. Section V presents and analyzes the experimental results. Section VI presents how this present work can be extended to multi-valued logic. Finally, section VII provides the concluding remarks and some pointers for future works.

II. Literature Survey

In this section we briefly discuss about quantum computation, nearest-neighbor quantum gate operations, nearest-neighbor controlled-NOT (CNOT) gate templates, and prior works on nearest neighbor architectures.

A. Quantum Computation

The fundamental entity in quantum computation is the quantum bit (or qubit). A qubit can exist in a number of states that can be represented as linear combination or superposition of computational basis states, like $|0\rangle$ and $|1\rangle$. This is characterized by the state vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex coefficients or amplitudes, and $|\alpha|^2 + |\beta|^2 = 1$. A quantum circuit consists of a cascade of quantum gates as shown in Fig. 1, which shows three qubits q_0, q_1 and q_2 on which various quantum gate operations are carried out in the axis of time.

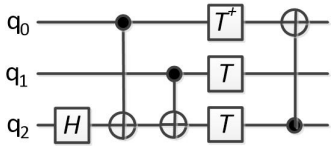


Fig. 1. An example quantum circuit

We perform circuit operations by evaluating the gates from left to right that change the states of the interacting qubits. The gate operations are inherently reversible, and the qubits can be in states of superposition or entangled. A physical quantum computer typically supports 1-qubit and 2-qubit gate operations. Some of the logical quantum gate libraries that are commonly used are NCV [16] and Clifforf+T [17]. In this paper, we are mainly concerned with 2-qubit gate operations irrespective of the gate library being used, which is equally applicable for multi-valued quantum systems as well.

B. Nearest-Neighbor Compliant Quantum Circuits

With respect to physical realization, experiments reveal that when two qubits interact, they must be in close proximity for limiting the error during computation [18]. This is an important issue in promising quantum technologies like ion trap [19] and superconducting [20], also referred to as *Nearest-Neighbor (NN) Constraint*.

Fig. 2(a) shows a 4-qubit quantum circuit with five gates. The vertical lines between solid dots in the figure represent arbitrary 2-qubit gates. The second and third gates are NN-compliant, while the remaining three are not. The states of

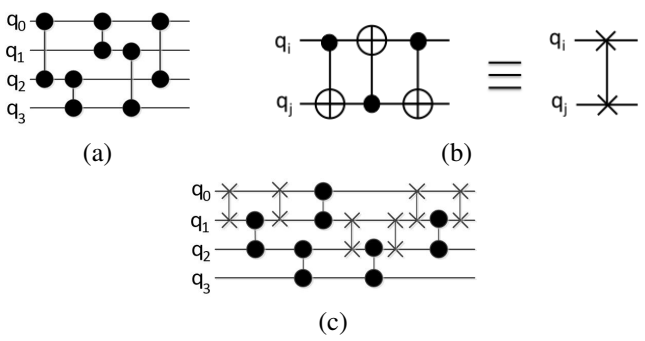


Fig. 2. (a) An example quantum circuit, (b) Swap gate realization, (c) Nearest-neighbor realization using Swap gates.

two qubits can be exchanged using a *Swap Gate* – Fig. 2(b) shows a realization of Swap gate using three back-to-back CNOT gates. A Swap gate is typically represented by two 'X' on the qubit lines with a solid vertical bar between them. If we want to execute the circuit of Fig. 2(a) in a reliable way, we need to insert Swap gates as shown in Fig. 2(c) to make the circuit NN-compliant.

In order to estimate the cost of making a quantum circuit NN-compliant, most prior works have used the number of Swap gates required as the cost metric. As an alternative, some of the works (e.g. [15]) have used *CNOT Templates*. As shown in Fig. 3(b), we need 4 Swap gates (i.e. NNC cost of 12) to make the gate in Fig. 3(a) NN-compliant. But using CNOT template we need only 8 CNOT gates [15] with NNC cost of 8 (see Fig. 3(c)).

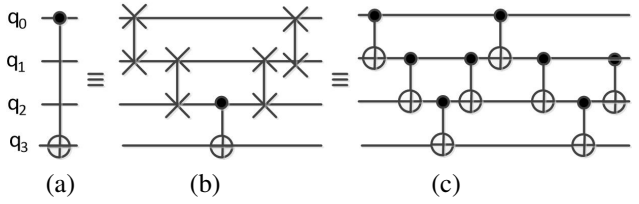


Fig. 3. (a) Gate with NNC = 2, (b) NN realization using Swap gate, (c) NN realization using CNOT gate templates [15].

C. Related Works

Several works exist in the literature for making a quantum circuit NN-compliant, based on both linear (1-D) and multi-dimensional (2-D and 3-D) architectures [6]–[8], [10]–[13], [21], [22]. The dimension determines the maximum number of neighbors a physical qubit can have. Fig. 4(a) shows a 1-D architecture for a 5-qubit circuit. The qubits at the two ends (q_2 and q_4) have one neighbor each, while the others have two neighbors. Fig. 4(b) shows a conventional 2-D architecture where 5 qubits are mapped. The qubit in the middle (q_0) has four neighbors, qubits at the corners have two neighbors, and all other qubits have three neighbors. It is clear that higher dimensions can lead to better qubit mapping with less cost for NN-compliance.

Many approaches for mapping quantum circuits to 1-D architectures have been proposed. Chakrabarti et al. [6] used

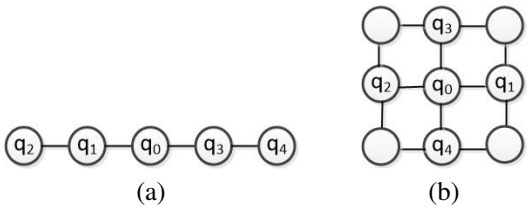


Fig. 4. (a) 1-D NN architecture, (b) 2-D NN architecture.

a graph-based method for reordering the qubits, and demonstrated the benefits that can be achieved. Hirata et al. [7] proposed a similar approach for mapping arbitrary quantum circuits to 1-D architecture; however, the method incurs high time complexity. In [23], Shafaei et al. have used an approach where they first partition the circuit into sub-circuits and then apply qubit ordering. In [10], Wille et al. used a SAT-based approach to optimally map circuits to linear qubit architecture, which can be used for small-sized circuits only.

Researchers have also explored higher dimensional 2-D architectures [11]–[13], [22]. Shafaei et al. [11] used mixed integer programming approach to map qubits to 2-D architectures. In [12], Shrivastwa et al. have used an innovative approach to speedup the process. In [24], Wille et al. have used a lookahead method for inserting Swap gates in both 1-D and 2-D architectures. Marbaniang et al. [22] proposed a lookahead mechanism during mapping, and have also considered the frequency of occurrence of the gates in a circuit. Another important concept in the context of NN-compliant circuits is *Global Ordering* and *Local Ordering* of qubits, as mentioned in [10] and used in many other subsequent works.

In [15], the authors proposed an approach using quantum gate templates rather than Swap gates to execute a CNOT gate with non-local control and target qubits. Essentially, a template consists of a set of NN-compliant CNOT gates. A CNOT gate with nearest neighbor distance of k (i.e., k intermediate qubits in the shortest path between the control and the target) can be executed by a template consisting of no more than $4k$ NN-compliant CNOT gates. This is clearly cheaper as compared to Swap gate insertion. This concept has been used in many recent works for mapping quantum circuits to various qubit architectures [25] [26] [27].

In the present work we propose a qubit mapping technique in hexagonal 2-D architecture and use CNOT template cost as the metric. We explain the hexagonal neighborhood structure in the next section.

III. The Hexagonal Neighborhood Structure

We consider a hexagonal architecture for arranging the qubits that can be conveniently laid out on a 2-D plane. This is in contrast to previous works, where qubits are assumed to exist in a regular arrangement with respect to 1-D, 2-D or 3-D Cartesian coordinate system. From the point of view of fabrication, 1-D and 2-D arrangements are most suitable. The proposed architecture offers the advantage of extended qubit neighborhood (maximum 6).

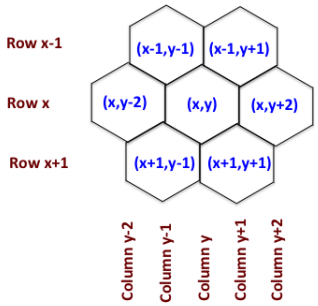


Fig. 5. The six neighbors of a central cell (x, y) in hexagonal array

The qubits are assumed to be laid out on a 2-D plane as shown in Fig. 5, conceptually as hexagonal tiles. Each qubit can have six neighbors corresponding to the six faces of the hexagon. The figure shows the proposed co-ordinate numbering convention used for a central cell with co-ordinates $(x,\ y)$. As shown, the cells are placed in rows that are numbered sequentially ($x-1$, x, $x+1$, etc.). The columns are numbered as shown, where the diagonal neighbors are placed in columns $y-1$ or $y+1$, while the horizontal neighbors in columns $y-2$ or $y+2$. Following this convention, it is easy to check whether two qubits are neighbors, and also to determine the shortest distance between any pair of qubits.

Proposition 1: Consider two qubits q_1 and q_2 with co-ordinates $(x_1,\ y_1)$ and $(x_2,\ y_2)$ respectively. The two qubits are said to be hexagonal neighbors if $|x_1-x_2|+|y_1-y_2|=2$, provided $y_1 \neq y_2$.

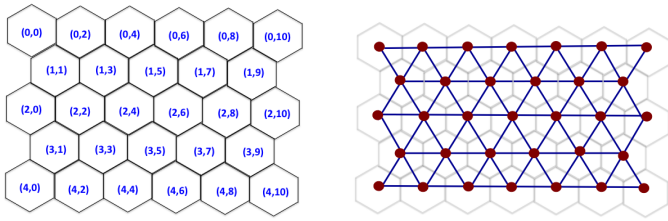
Proof: With respect to Fig. 5, the six neighbors of the central cell $(x,\ y)$ have co-ordinates $(x-1,\ y-1)$, $(x-1,\ y+1)$, $(x,\ y+2)$, $(x+1,\ y+1)$, $(x+1,\ y-1)$, and $(x,\ y-2)$. For all the neighbors, the proposition $|x_1-x_2|+|y_1-y_2|=2$ is found to be true, except the case when $y_1=y_2$ and $x_1=x_2\pm 2$. Also for any other cells beyond the hexagonal neighborhood, the value of the expression will be 4 or more. Hence the proof.

A. Hexagonal Neighborhood Coordinate Convention

Fig. 6(a) shows how we can arrange the qubits in a 2-D plane using the coordinate convention mentioned in Fig. 5. We also show the co-ordinates of the qubits in the mapped 2-D Cartesian plane. It may be verified that *Proposition 1* holds for all neighborhood relations in the 2-D arrangement of the cells. As a matter of convention, the top-left cell is assigned the co-ordinates (0, 0), simply to ensure that the x and y co-ordinates of all the other cells are non-negative.

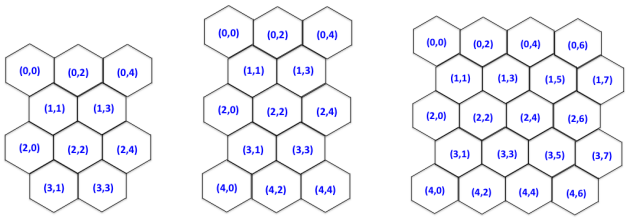
Fig. 6(b) shows the actual qubit positions in the hexagonal array, and the way they are interconnected. The interconnection depicts a unique planar triangulated structure, as a union of equilateral triangles. It can be observed that for any qubit, the Euclidean distance to each of its six neighbors are all equal.

Algorithm 1 presents the procedure to compute the neighborhood distance between two qubits at co-ordinates $(x_1,\ y_1)$ and $(x_2,\ y_2)$ on the hexagonal array. The detailed explanation is not provided here.



(a) (b)

Fig. 6. (a) Hexagonal cells in a 2-D array, (b) Qubit interactions: Red circles indicate qubits, and blue lines their interactions



(a) 4×5 (b) 5×5 (c) 5×8

Fig. 7. Cell arrays of given dimensions.

Algorithm 1 Distance between qubits (x_1, y_1) and (x_2, y_2)

Input: Qubit co-ordinates (x_1, y_1) and (x_2, y_2)
Output: Distance d on hexagonal grid
$d \leftarrow 0$;
while (true) **do**
  if $(x_1 = x_2)$ **then**
    return $d + |y_1 - y_2|/2$;
  end if
  if $(y_1 = y_2)$ **then**
    return $d + |x_1 - x_2|$;
  end if
  if $(|x_1 - x_2| \neq 1)$ **then**
    if $(x_1 < x_2)$ and $(y_1 < y_2)$ **then**
      $x_1 = x_1 + 1$; $y_1 = y_1 + 1$; $d = d + 1$;
    end if
    if $(x_1 < x_2)$ and $(y_1 > y_2)$ **then**
      $x_1 = x_1 + 1$; $y_1 = y_1 - 1$; $d = d + 1$;
    end if
    if $(x_1 > x_2)$ and $(y_1 < y_2)$ **then**
      $x_1 = x_1 - 1$; $y_1 = y_1 + 1$; $d = d + 1$;
    end if
    if $(x_1 > x_2)$ and $(y_1 > y_2)$ **then**
      $x_1 = x_1 - 1$; $y_1 = y_1 - 1$; $d = d + 1$;
    end if
  end if
  if $|x_1 - x_2| = 1$ **then**
    return $d + (|x_1 - x_2| + |(y_1 - y_2)|)/2$;
  end if
end while

Example 1: Consider two cells A (4,2) and B (1,9) as shown in Fig. 6(a). Here, the distance $D(A, B) = 4$. We have to traverse through four intermediate cells to move from A to B (e.g., right, right, diagonal-up, diagonal-up; or diagonal-up, diagonal-up, diagonal-up, right, etc.). Multiple such paths may exist between a pair of qubits.

In order to specify the size of the hexagonal array of qubit cells, we follow a simple convention. A cell array of dimension $m \times n$ shall include all the cells with co-ordinates (x, y), such that $x < m$ and $y < n$. Some example cell-array dimensions are illustrated in Fig. 7.

B. Nearest-Neighbor Mapping of Qubits

For a quantum circuit with n logical qubits $\{q_1, q_2, \ldots, q_n\}$, the first step is to map them to a physical qubit architecture. As per the NN-constraint, all 2-qubit gates must operate on physically adjacent qubits. If a gate operates on the qubit pair (q_i, q_j), where q_i and q_j are not physically adjacent, we insert CNOT templates for NN-compliance.

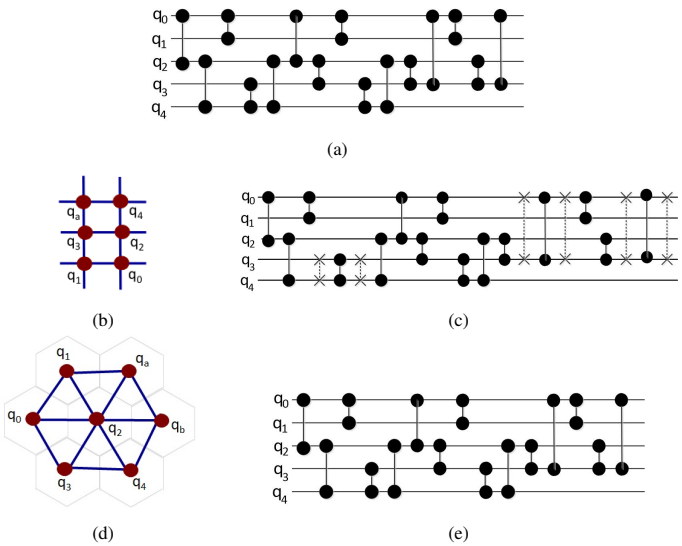


Fig. 8. (a) A quantum circuit with 5 logical qubits and 15 2-qubit gates, (b) Mapping of qubits on 2-D array, (c) NN gate evaluation on 2-D array with 6 swap operations, (d) Mapping of qubits on hexagonal array, (e) NN gate evaluation with zero NN cost

Example 2: The mapping of gate operations for 2-D and hexagonal architectures are illustrated in Fig. 8. We consider a quantum circuit with 15 2-qubit quantum gate operations, each of which is represented by a vertical bar between two solid dots (indicating the interacting qubits). Assuming the qubits are mapped to a 2-D physical qubit architecture as shown in Fig. 8(b), for NN-compliance we need to insert CNOT gate templates or Swap gates (on neighboring pairs of qubits) as shown in Fig. 8(c). However, when the qubits are mapped to a 2-D hexagonal architecture, all gate operations are on adjacent qubits with no additional overheads (see Fig. 8(d) and 8(e)).

IV. Proposed NN-Mapping for Hexagonal Architecture

In this section we discuss the mapping of qubits of a given quantum circuit to the hexagonal architecture. For generating the initial qubit placement, many earlier methods used the interaction graph data structure to capture the number of operations between every pair of qubits. This is referred to as *Global Ordering*. To reduce the number of Swap gates for NN-compliance, another step called *Local Ordering* is also used. This latter step often requires looking ahead in the gate netlist and as such is a relatively time consuming process. It

has been demonstrated in prior works that combining both global ordering and local ordering gives better solutions.

In the proposed work, we have totally done away with the local ordering step, and have created a global ordering of the qubits only. We observe later in the experimental results that incorporating only global ordering for hexagonal architecture provides substantial improvement over existing approaches. We formulate the qubit placement problem on 2-D hexagonal grid as a search problem, and use an evolutionary algorithm (viz. Genetic Algorithm or GA) to find a good ordering of the qubits [28]. GA is considered as a meta-heuristic approach influenced by natural selection process. In the proposed GA implementation, we define the solution representation, the fitness function, and the crossover/mutation operations in a suitable way. The method is found to be simple and effective in terms of computation overhead.

Algorithm 2 Qubit mapping on hexagonal grid

```
Input: Quantum gate netlist Q_G
Output: Qubit map and NNC for best solution obtained
Q_N ← Read Q_G from file;
P ← Random initial population;                    ▷ Required for GA
iter ← 0;
while iter ≠MAXITER do
    Compute_Fitness (P, Q_N);
    P ← Perform_Crossover (P);                    ▷ With probability p_cross
    P ← Perform_Mutation (P);                     ▷ With probability p_mut
    Save best solution;
    iter ← iter +1;
end while
NNC ← NN-cost of best solution;
Print qubit map and NNC of best solution;
```

The basic steps of the GA-based qubit mapping are presented in *Algorithm 2*. We start with an initial random mapping of qubits on a given hexagonal grid, and then run GA to progressively improve the fitness values across generations. Every generation comprises of a set of solutions, called the population. The solution with the lowest cost (i.e. fitness function value) is retained as the final solution. Initially, we read a quantum circuit description from a file and populate in-memory data structures. The various aspects of the GA implementation are explained below with examples.

a) *Solution Representation:* The qubit placement solution on a 2-D hexagonal grid is represented as an 1-D integer array in row-major order, with proper co-ordinate mapping. The entries in the array indicate the qubit number (if assigned), or -1 if empty. Considering Fig. 9(a), the solution representation will be $(q_3, q_0, -1, q_1, -1, q_2, q_4)$, which is illustrated in Fig. 9(b).

b) *Initial Population:* A population consists of N number of solutions, where each solution represents some arrangement of the qubits. The solutions in the initial population are generated randomly, which is refined across iterations. Across the iterations we move towards better solutions.

c) *Fitness Function:* For estimating the fitness or NN-cost (NNC) of a solution, we observe that for implementing a CNOT operation on a pair of qubits that are distance k apart, $4k$ CNOT gates are required in the CNOT template for NN-compliance [15].
For a quantum gate netlist with gates $\{g_1, g_2, \ldots, g_{ng}\}$, where k_i is the inter-qubit distance of gate g_i, the NNC can be directly computed as

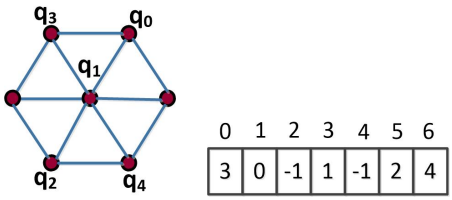


Fig. 9. (a) Given qubit mapping, (b) Mapping to 1-D array.

$$NNC = \sum_{i=1}^{ng} (4k_i) \quad (1)$$

The time complexity for computing NNC is linear in terms of the number of gates, i.e. $O(n_g)$. This simple fitness functions is found to be effective in terms of qubit placement in the hexagonal 2D grid.

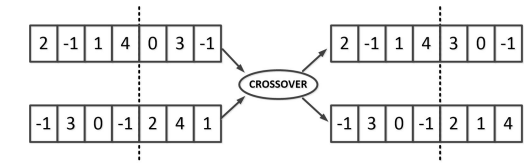
d) *Crossover and Mutation Functions:* Crossover and mutation are two important steps in the GA formulation. To move from current generation Gen_i to the next generation Gen_{i+1}, we perform crossover and mutation operations with probabilities p_{cross} and p_{mut} respectively, as per the following procedures.

i) *Retain best solutions:* We copy the best m solutions from Gen_i to Gen_{i+1}. This way, the best solutions are retained across generations.

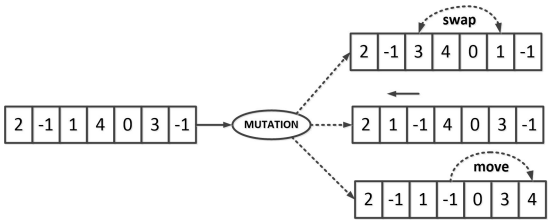
ii) *Crossover operation:* We choose a pair of solutions $S_{x,i}, S_{y,i} \in Gen_i$ based on their fitness values (using roulette wheel method [28]), and perform crossover with a probability p_{cross}. The process of crossover is illustrated in Fig. 10(a). We choose the crossover point randomly, and copy the first parts of the solutions $S_{x,i}$ and $S_{y,i}$ to the first parts of the new solutions $S_{x,i+1}$ and $S_{y,i+1}$. The second parts of the solutions $S_{x,i+1}$ and $S_{y,i+1}$ are generated by filling up the missing qubits in the same order as they appear in $S_{y,i}$ and $S_{x,i}$ respectively. Additional entries, if any, are filled with -1. This ensures that a qubit appears only once in a solution.

iii) *Mutation operation:* We carry out mutation on the solutions as they are copied from Gen_i to Gen_{i+1} with a probability p_{mut}. We randomly select one of several methods, like swap positions of two randomly selected qubits, move a randomly selected qubit to a vacant adjacent position, move a qubit to a randomly selected vacant position, etc. This is illustrated in Fig. 10(b).

The parameters of the GA have been selected through experimentation. We use a population size $N = 30$, and the



(a) Crossover example



(b) Mutation example

Fig. 10. Illustration of crossover and mutation operations.

number of generations is 200. The crossover and mutation probabilities are $p_{cross} = 0.70$ and $p_{mut} = 0.20$ respectively.

To compare the efficacy of the proposed GA-based qubit ordering, the results are compared with a greedy qubit ordering approach as depicted in *Algorithm 3*. In this greedy approach, the qubits are placed on a given hexagonal grid in descending order of qubit interactions. The choice of placement is made to ensure that the NNC cost is minimized at every step of the iteration. The results of the greedy algorithm are discussed in the next section.

Algorithm 3 Greedy approach for qubit mapping

Input: Quantum gate netlist Q_G
Output: Qubit map and NNC for best solution obtained
$Q_{int} \leftarrow$ get qubit-interactions from netlist Q_G;
$Q \leftarrow$ order qubits from Q_{int} in descending order of interactions;
Place first element of Q in central grid position;
for (all remaining $q \in Q$) **do**
  Place q in the grid position that minimizes NNC;
end for
Print qubit map and NNC of final solution;

V. Experimental Results

The proposed approach for qubit mapping on hexagonal neighborhood architecture has been implemented in C and run on an Intel i7 processor with 8 GB of memory. Experiments are carried out on various quantum circuit benchmarks, which consist of 1-qubit and 2-qubit gates only. The circuits are synthesized using technology mapping with NCV gate library. The program is general and can be used for other quantum gate libraries also; however, for comparing with previous works, we have used the NCV library. The program takes a quantum gate netlist in `.real` format [29], determines the required dimension of the hexagonal array, and provides as output the best solution found. As mentioned in the previous section, the GA initially generates a random solution, which passes through various iterations across generations where the solutions are progressively improved through crossover and mutation operations.

Table I summarizes the results of experimentation on the benchmark circuits. The first three columns of the table shows the names of the benchmark circuits, number of qubits (N_q) and number of quantum gates (N_g) respectively. The next column shows the cost (NNC) of making the circuit NN-compliant according to [30] – obtained by multiplying the number of Swap gates required by three.[1] The next column shows the NNC value for the Greedy algorithm on the hexagonal array. The last four columns depict the results obtained using the proposed GA-based approach, consisting of the hexagonal array dimension (Dim.), the NN-cost using CNOT gate templates (NNC)[2], the percentage improvements (Imp.) in NN-cost over [30], and also the run time in seconds. The run times for most of the benchmarks have been observed to be < 1 second, with the highest run time of 9.14 seconds observed for the benchmark `hwb7_62`.

It is observed from the table that with the exception of six benchmarks, the proposed method shows significant improvements over the best known method for 2-D qubit architecture [30]. In the earlier reported works (including [30]), various optimization heuristics have been used incorporating both global ordering and local ordering techniques. In contrast, the proposed method uses global ordering only, and as such no Swap gate insertions have been carried out. This establishes the advantage of the hexagonal qubit architecture in terms of making a circuit NN-compliant. Hence by just incorporating global ordering we are able to achieve better results compared to the state-of-the-art mapping methods on 2-D Cartesian architectures. Also, the results of the greedy approach show lower performance for all the benchmarks as compared to the proposed GA-based method. We can also observe that the greedy approach is even worse as compared to mapping for 2-D architecture [30] in most of the cases. Out of 38 benchmarks considered for experimental evaluation, the greedy hexagonal approach generates worse results for 24 benchmarks. This also gives us a clear indication that the proposed GA-based mapping is effective. In the present work we have used CNOT templates, but as a future work, various heuristics combining CNOT templates with Swap gate insertion shall be explored. This is expected to provide further reductions in cost.

Experiments have also been run on larger benchmark circuits with 20 qubits or more, and the results are summarized in Table II. For each of the benchmarks (with parameters N_q and N_g), the table shows the hexagonal array dimensions, the NN-costs using CNOT gate templates, and the run times. The results show that even for circuits with higher number of qubits, the run times are reasonably low with the highest value of 12.69 sec reported for the benchmark `f51m_233`.

VI. Extension to Multi-valued Quantum Systems

The proposed gate mapping approach on hexagonal quantum computing architecture can be easily extended to multi-valued quantum systems where the cells represent *qudits*. For instance, in ternary quantum computing [31], gate operations

[1]Every Swap gate requires three CNOT gates for implementation.

[2]A CNOT gate with control and target at distance k apart, will require $4k$ CNOT gates for NN-compliance.

TABLE I
COMPARATIVE RESULTS ON BENCHMARK CIRCUITS

Benchmark Circuits			NNC		Proposed GA-based Approach			
Name	N_q	N_g	[30]	Greedy	Dim.	NNC	% Imp. over [30]	Time (sec)
3_17_13	3	14	12	0	2 × 3	0	100.0	0.08
4_49_17	4	32	27	16	2 × 4	12	55.6	0.04
hwb4_52	4	23	21	8	2 × 4	8	61.9	0.03
rd32-v0_67	4	8	6	4	2 × 4	0	100.0	0.00
4gt11_84	5	7	3	0	3 × 3	0	100.0	0.01
4gt13-v1_93	5	17	6	8	3 × 4	0	100.0	0.01
4mod7-v0_95	5	40	27	48	4 × 4	12	55.6	0.03
4gt5_75	5	22	21	12	3 × 4	8	61.9	0.02
aj-e11_165	5	60	54	56	3 × 4	4	92.6	0.03
4gt10-v1_81	5	36	45	20	3 × 4	12	73.3	0.06
4mod5-v1_23	5	24	24	20	3 × 5	8	66.7	0.02
alu-v4_36	5	32	24	24	3 × 4	8	66.7	0.02
hwb5_55	5	109	132	112	3 × 4	40	69.7	0.06
QFT5_95	5	10	12	12	3 × 5	12	0.0	0.03
4gt4-v0_80	6	44	45	40	3 × 4	4	91.1	0.03
4gt12-v1_89	6	53	45	32	3 × 4	8	82.2	0.03
hwb6_58	6	146	168	192	3 × 4	140	16.7	0.08
mod5adder_128	6	87	84	64	3 × 4	52	30.1	0.05
QFT6	6	15	84	24	3 × 4	24	71.4	0.01
mod8-10_177	6	109	111	116	3 × 4	72	35.1	0.06
hwb7_62	7	2663	3798	4356	4 × 5	3996	-5.2	1.45
QFT7	7	21	42	44	4 × 5	36	14.3	0.02
rd53_135	7	78	117	84	4 × 5	56	52.1	0.05
ham7_104	7	87	87	96	4 × 5	56	35.6	0.17
hwb8_118	8	16610	20526	31360	6 × 6	25768	-25.5	9.15
QFT8	8	28	57	68	5 × 5	64	-12.3	0.02
QFT9	9	36	84	104	5 × 6	96	-14.3	0.03
QFT10	10	45	129	140	4 × 6	132	-2.3	0.05
sym9_148	10	4452	6592	6624	4 × 5	5048	23.4	2.28
sys-v0_144	10	62	96	84	5 × 6	32	66.7	0.05
Shor3	10	2076	4035	3664	5 × 7	2560	36.6	1.37
rd73_140	10	76	78	128	5 × 5	56	28.2	0.07
cycle10_2_110	12	1212	1728	2996	5 × 6	2008	-16.2	1.05
Shor4	12	5002	10368	12492	6 × 8	7500	27.7	3.68
Shor5	14	10265	22269	26824	5 × 6	17904	19.6	5.28
ham15_108	15	458	672	876	6 × 6	628	6.5	0.35
rd84_142	15	112	174	316	6 × 6	128	26.4	0.12
cnt3-5_180	16	125	192	436	6 × 6	116	39.6	0.10

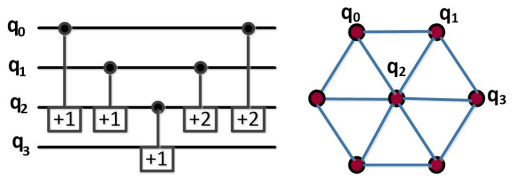
N_q: Number of qubits, N_g: Number of gates, NNC: neareat-neighbor cost,
Dim.: dimension of hexagonal array, Imp.: % improvement

TABLE II
RESULTS ON LARGE BENCHMARK CIRCUITS

Benchmark Circuits			Proposed GA-based Approach		
Name	N_q	N_g	Dim.	NNC	Time (sec)
add16_174	49	148	10 × 10	808	0.20
add32_183	97	292	14 × 14	3440	0.69
add32_185	97	192	14 × 14	2380	0.57
add64_184	193	576	20 × 20	13048	2.21
add64_186	193	384	20 × 20	7888	1.52
alu1_198	20	189	7 × 7	488	0.20
apla_203	22	2051	7 × 7	4648	1.53
arb8_323	24	522	7 × 7	1764	0.38
bw_291	87	719	14 × 14	7304	1.53
c2_181	35	328	9 × 9	1396	0.36
cm151a_211	28	639	7 × 8	2076	0.58
cu_219	25	752	7 × 7	1320	0.46
decod_217	21	845	6 × 7	1160	0.46
f51m_233	22	19724	7 × 7	65968	12.69
ham15_298	45	196	9 × 10	1092	0.22
hwb5_300	28	217	7 × 8	912	0.18
hwb7_302	73	709	12 × 13	7936	1.24
hwb9_304	170	1753	19 × 19	36492	5.58
in0_235	28	11297	7 × 8	31800	6.76
mod5adder_306	32	227	8 × 8	1040	0.21
rd73_312	25	169	7 × 7	604	0.14
rd84_313	34	225	8 × 9	1056	0.26

are defined on three basis states of qudits (called *qutrits*), $|0\rangle$, $|1\rangle$ and $|2\rangle$. There are various 1-qutrit and 2-qutrit gate operations that are defined [32]. For implementing these operations in hexagonal architecture will also demand nearest-neighbor constraints for efficient computations. Although several works have been reported on the synthesis of ternary quantum circuits [31], [33]–[35], very little efforts has been made for mapping the qutrits to 1-D and 2-D architectures. However, in the context of the present work, the qutrits of a ternary quantum circuit can be efficiently mapped to the hexagonal architecture, where each cell represents a qutrit.

Example 3: Fig. 11(a) shows an example ternary quantum circuit with four qutrits q_0, q_1, q_2 and q_3, with five 2-qutrit gate operations. If the qutrits are assumed to be linearly arranged in 1-D, the first and fifth gates shall operate on non-neighbor pair of qutrits. However, if the qutrits are mapped to a hexagonal grid as shown in Fig. 11(b), all the gate operations become NN-compliant. It may be noted that in general some of the 2-qutrit gate operations may not be NN-compliant on the hexagonal grid, and one or more qutrit Swap operations may be required in such cases.



(a) (b)

Fig. 11. Qutrit mapping in a ternary quantum circuit: (a) An example ternary quantum circuit, (b) NN-mapping of qutrits to hexagonal grid.

VII. Conclusion

We present a 2-D placement algorithm for mapping qubits in hexagonal NN architecture using GA. The formalism of the coordinate system makes the task of estimating NN distance computationally efficient. The fact that every qubit has six neighbors makes the architecture beneficial compared to prior 2-D works based on Cartesian coordinates. The GA-based approach for finding a suitable global ordering of qubits is found to be fast and efficient as compared to greedy approach. The greedy approach is found to show worse results compared to 2-D works based on Cartesian coordinates. We also observe that the use of CNOT templates is beneficial as compared to Swap gates for making the circuit NN-complaint. We have carried out experiments on various benchmarks. The results show an average improvement of 42.9% over a very recent work based on 2-D Cartesian coordinates. We have not used local ordering in this approach, and hence one of the future works can be to use local ordering using Swap gates in addition to CNOT templates to further improve the NNC cost. The proposed hexagonal architectures can also be used for multi-valued quantum systems.

References

[1] R. P. Feynman, "Simulating physics with computers," *Int'l Journal of Theoretical Physics*, vol. 21, pp. 467–488, 1982.

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

[3] "IBM QX backend information." https://github.com/Qiskit/ibmq-device-information. [Accessed: 2019-03-20].

[4] "IBM Q." https://www.research.ibm.com/ibm-q. [Accessed: 2019-03-20].

[5] A. Paler, A. Zulehner, and R. Wille, "NISQ circuit compilers: search space structure and heuristics," *CoRR*, vol. abs/1806.07241, 2018.

[6] A. Chakrabarti and S. Sur-Kolay, "Nearest neighbour based synthesis of quantum boolean circuits," *Eng. Lett.*, vol. 15, pp. 356–361, 2007.

[7] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture," in *Int'l Conf. on Quantum, Nano and Micro Technologies*, (Washington, DC, USA), pp. 26–33, 2009.

[8] M. Perkowski, M. Lukac, D. Shah, and M. Kameyama, "Synthesis of quantum circuits in linear nearest neighbor model using Positive Davio Lattices," *Elec. Energ.*, vol. 24, no. 1, pp. 71–87, 2012.

[9] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quant. Info. Proc.*, vol. 10, pp. 355–377, Jun 2011.

[10] R. Wille, A. Lye, and R. Drechsler, "Exact reordering of circuit lines for nearest neighbor quantum architectures," *IEEE Trans. on CAD*, vol. 33, pp. 1818–1831, Dec 2014.

[11] A. Shafaei, M. Saeedi, and M. Pedram, "Qubit placement to minimize communication overhead in 2D quantum architectures," in *Asia and South Pacific Design Automation Conf.*, pp. 495–500, Jan 2014.

[12] R. R. Shrivastwa, K. Datta, and I. Sengupta, "Fast qubit placement in 2D architecture using nearest neighbor realization," in *Int'l Symp. on Nanoelectronic and Information Systems*, pp. 95–100, Dec 2015.

[13] A. Kole, K. Datta, and I. Sengupta, "A new heuristic for N-dimensional nearest neighbor realization of a quantum circuit," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 182–192, January 2018.

[14] H. Tang *et al.*, "Experimental quantum fast hitting on hexagonal graphs," *Nature Photonics*, vol. 12, no. 12, p. 754–758, 2018.

[15] M. Rahman and G. W. Dueck, "Synthesis of linear nearest neighbor quantum circuits," in *10th Int'l Workshop on Boolean Problems*, pp. 1–9, 2012.

[16] A. Barenco, C. H. Bennet, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995.

[17] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 818–830, June 2013.

[18] H. Häffner, W. Hänsel, C. Roos, J. Benhelm, M. Chwalla, T. Körber, U. Rapol, M. Riebe, P. Schmidt, and C. Becher, "Scalable multiparticle entanglement of trapped ions," *Nature*, vol. 438, pp. 643–646, Dec 2005.

[19] C. Monroe and J. Kim, "Scaling the ion trap quantum processor," *Science*, vol. 339, pp. 1164–1169, Mar 2013.

[20] A. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nature communications*, vol. 6, Apr 2015.

[21] 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. Topics Circuits Syst.*, vol. 6, pp. 62–72, Feb 2016.

[22] L. Marbaniang, A. Kole, K. Datta, and I. Sengupta, "Design of efficient quantum circuits using nearest neighbor constraint in 2D architecture," in *9th Intl. Conf. on Reversible Computation*, pp. 248–253, July 2017.

[23] A. Shafaei, M. Saeedi, and M. Pedram, "Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures," in *Design Automation Conf.*, pp. 41:1–41:6, May 2013.

[24] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler, "Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits," in *Asia and South Pacific Design Automation Conf.*, pp. 292–297, Jan 2016.

[25] A. A. A. de Almeida, G. W. Dueck, and A. C. R. da Silva, "CNOT gate mappings to Clifford+T circuits in IBM architectures," in *Int'l Symp. on Multiple-Valued Logic*, pp. 7–12, 2019.

[26] X. Zhou, S. Li, and Y. Feng, "Quantum circuit transformation based on simulated annealing and heuristic search," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4683–4694, 2020.

[27] P. Niemann, C. Bandyopadhyay, and R. Drechsler, "Combining SWAPs and remote Toffoli gates in the mapping to IBM QX architectures," in *Design Automation and Test in Europe*, pp. 1–6, 2021.

[28] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addision-Wesley Professional, 1989.

[29] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Proc. Intl. Symposium on Multiple-Valued Logic*, (Texas, USA), pp. 220–225, May 2008.

[30] A. Bhattacharjee, C. Bandyopadhyay, P. Niemann, B. Mondal, R. Drechsler, and H. Rahaman, "An improved heuristic technique for nearest neighbor realization of quantum circuits in 2D architecture," *Integration, the VLSI Journal*, vol. 76, no. 1, pp. 40–54, 2021.

[31] P. M. N. Rani, A. Kole, K. Datta, and A. Chakrabarty, "Realization of ternary reversible circuits using improved gate library," *Procedia Computer Science*, vol. 93, pp. 153–160, 2016.

[32] C. Moraga, "Ternary toffoli-type reversible gates: Control alternatives and quantum models," in *Int'l Symp. on Multiple-Valued Logic*, pp. 101–106, May 2021.

[33] A. Kole, P. M. N. Rani, K. Datta, I. Sengupta, and R. Drechsler, "Exact synthesis of ternary reversible functions using ternary Toffoli gates," in *Int'l Symp. on Multiple-Valued Logic*, pp. 179–184, May 2017.

[34] P. M. N. Rani and K. Datta, "Improved ternary reversible logic synthesis using group theoretic approach," *Journal of Circuits, Systems and Computers*, vol. 29, no. 12, pp. 2050192:1–2050192:24, 2020.

[35] D. M. Miller and G. Dueck, "Descending order transformation-based synthesis of mvl reversible circuits," in *Int'l Symp. on Multiple-Valued Logic*, pp. 107–112, May 2021.