Improved Look-ahead Approaches for Nearest Neighbor Synthesis of 1D Quantum Circuits

Anirban Bhattacharjee[1], Chandan Bandyopadhyay[1], Robert Wille[2], Rolf Drechsler[3], Hafizur Rahaman[1]
[1]Indian Institute of Engineering Science and Technology Shibpur, India-711103
[2]Institute for Integrated Circuits, Johannes Kepler University Linz, A-4040 Linz, Austria
[3]Institute of Computer Science, University of Bremen & Cyber-Physical Systems, DFKI GmbH, 28358 Bremen, Germany
Email: anirbanbhattacharjee330@gmail.com, chandanb@it.iiests.ac.in, robert.wille@jku.at, drechsle@uni-bremen.de, rahaman_h@it.iiests.ac.in

***Abstract*—In the present era of computation, quantum computing may offer a new direction as it allows to solve certain problems significantly faster than classical solutions. But it also has been found that there are several constrains in performing a successful realization of quantum circuits. One such constraint is the nearest neighbor (NN) criterion which states that qubits which interact with each other have to be adjacent. Motivated by this objective, in this work we propose a linear qubit placement technique that effectively rearranges the qubits and transforms quantum circuits to improved NN-based designs by inserting SWAPs. Furthermore, for placing these SWAPs in appropriate positions, we implemented a look-ahead strategy that considers the effect of the rest of the gates and computes a corresponding impact value which guides the insertion of the SWAP gates. To this extent, we consider three different strategies to evaluate the corresponding "look-ahead effects" and their influence on existing gates. At the end of this work, we have evaluated the developed methodology over a wide range of benchmarks and compared the results with existing related works. In this comparison, we have seen that the proposed technique outperforms the related works and provides substantial reductions in SWAP overhead.**

Keywords— Quantum Circuit, quantum gate, Nearest Neighbour(NN), SWAP gate.

I. Introduction

Quantum computing has emerged as an advanced computing platform, which, unlike classical circuits that follow the principle of classical mechanics, is driven by the principles of quantum mechanics. This enables such circuits to solve certain problems significantly faster. For example, quantum circuits have shown its capacity in solving some of the most complex problems like prime factorization in RSA cryptosystems [1], discrete logarithm problems [1], or database search [2] much faster than classical solutions.

More importantly, the concept of quantum circuits is not limited to the theoretical domain only, but in the recent years, technologies like ion-trap [3], quantum dots [4], nuclear magnetic resonance [5], or superconducting qubits [6] have successfully shown their ability to realize such circuits. However, here several technological constraints arise which have to be considered. One such restriction comes in form of the Nearest Neighbour (NN) [7] criteria which demands the qubits in a circuit to be adjacent.

As many quantum circuit implementing technologies demand NN complaint designs, it becomes necessary to develop algorithms that can transform quantum circuits to NN based designs. In general, the simplest way of performing such transformations is by embedding SWAP gates in between non-adjacent qubits [7]. However, here it is of the essence to keep the number of inserted SWAP gates small in order to reduce the overhead and, there with, the cost of the design.

More importantly, it is seen that in most of the NN related works this parameter has emerged as main metric to evaluate the efficiency of the design scheme. Since a couple of years, several investigations are reported on the efficient design strategies for NN circuits. More precisely, the problem has been addressed in the work [7], which rearranges the original qubit positions using various strategies. Further to improve the design structure, a circuit transformation technique has been introduced in [8], where the authors have converted the design problem into an intermediate task assignment problem and then administered a meta-heuristic based search technique to obtain the equivalent NN-based design. In [9], the authors have demonstrated a synthesis method based on mapping of the entire circuit on a lattice structure. For making more efficient realization, a graph-partitioning approach has been considered in [10], where optimized NN based designs have been formed by reducing SWAP gate count significantly. Despite of applying a global reordering strategy for gate rearrangements [20, 21], an exact design solution based on a local scheme is introduced in [11] and this approach has showed promising results for small benchmark circuits. In [12], the authors introduced a local reordering strategy that employs a look-ahead scheme to obtain an improved design structure. Again, for large scale synthesis of NN circuits, a heuristic based look-ahead scheme has been presented in [13].

Albeit, all the works that we have stated so far are on linear representation of NN circuits, nowadays investigations have also been conducted on 2D [18, 19], 3D [22] and even multidimensional architectures [14]. However, in this work, we focus on 1D architectures. For this, we introduce a heuristic look-ahead design methodology consisting of three distinct cost estimation models to obtain a linear NN based representations. Although the notion of look-ahead scheme has already been discussed in the earlier works, the way it has been implemented there differs from the one used here which

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00054

IEEE computer society

have shown significant improvements over the reported works.

In the next section (Section II), we have discussed on the fundamentals of Quantum circuit and nearest neighbor representation. Problem formulation and motivation of the work is summarized in Section III. The proposed methodology with examples is stated in Section IV. Section V contains the experimental results with the reported works. Finally, the concluding remarks appear in Section VI.

II. Background

Quantum gates perform operations by modifying the states of qubits rather than its value. A qubit can exist in the basis states of $|0\rangle$ and $|1\rangle$, which are same as that of 0 and 1 in classical computing. Besides these basis states, it can also occur in states which can be expressed as a linear superposition of the basis states represented by the state vector $|\Psi\rangle$ as:

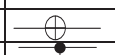
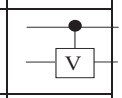
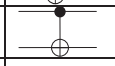
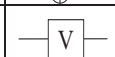
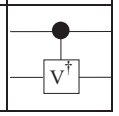
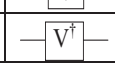
$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad (1)$$

where the notations α and β are the complex numbers that specify the probability amplitudes of the basis states and hold the condition $\alpha^2 + \beta^2 = 1$. However, this state vector cannot be detected directly as measurement causes the vector to degenerate into one of the basis states of $|0\rangle$ and $|1\rangle$ with probabilities α^2 and β^2 respectively.

The states of the qubit can be manipulated by executing quantum gates whose operations are described through unitary matrices. To this extent, quantum operations on an n-qubit system can be realized by multiplying various $2^n \times 2^n$ unitary matrices.

Schematic representation of some of the most commonly used quantum gates (both 1-qubit and 2-qubits) have been summarized in chart 1.

Chart 1: Well known quantum gates and their schematic representations

Gates	Notation	Gates	Notation
NOT		Controlled -V	
CNOT			
V		Controlled - $V^\dagger$	
$V^\dagger$			

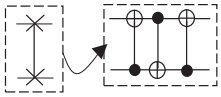


Fig. 1: Design of SWAP gate

In general, the transformation operations of 2-qubit quantum gates viz. CNOT, Controlled-V, Controlled-V^+ can be described by the following matrix representation

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix}$$

where the variables a, b, c, d denotes the generic values which varies with different 2-qubit quantum gate operations. To compact the representation further, such generic values can be combined in a separate matrix form as shown below

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

The notation *(U)* defined in the above matrix represents the operations performed by two qubit quantum gates and henceforth we used this notation in the rest of the work.

Definition1. *A circuit designed with a cascade of quantum gates over a set of control and target lines is known as quantum circuit.*

The cost of a quantum circuit is basically determined by four parameters – quantum cost [15], T-count**,** T-depth [16] and NN cost. As in our technique, we have considered the fourth cost metric only, here we are discussing on that metric and its related terms.

Definition 2: *The nearest neighbor cost of a two-qubit gate g(c,t) with control input on line c and target input on line t can be determined by computing the difference between the lines c and t.*

Mathematically, this difference value can be expressed as follows:

$$NNC_g = |c - t| - 1 \qquad (2)$$

For 1-qubit gate like NOT, the NN cost is 0.The overall NN cost of a circuit (NNC_C) is considered as the cumulative sum of NN cost contributed by each gate (g) in the circuit. This relation can be expressed as follows:

$$NNC_C = \sum_g NNC_g \qquad (3)$$

If in a circuit $\sum_g NNC_g$ becomes 0, then the circuit can be declared as an NN compliant circuit. But if the circuit does not satisfy the NN criteria then a special kind of gate namely SWAP gate (the structure of the gate is given in Fig. 1) is inserted in-between non-adjacent qubits to make them adjacent.

For clear understanding, let us consider a circuit as given in Fig. 2(a). The given design has a NN cost of 6 as qubits in all the four gates are non-adjacent. So, to make them adjacent, 12 SWAP gates have been embedded in specific positions and the equivalent NN compliant representation is finally obtained in Fig. 2(b).

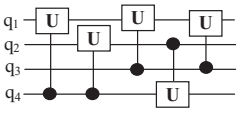


Fig. 2(a): Input circuit with NN Cost 6

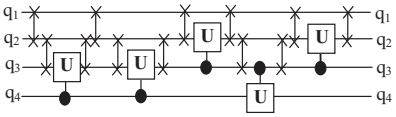


Fig. 2(b): NN equivalent design corresponding to Fig. 2(a)

III. Motivation and Problem Formulation

Basically, nearest neighbor designs can be obtained by following either of the two reordering strategies – global [7,8,10,11,20,21] and local [6,7,11,12,13,22]. As local reordering strategy implements less SWAP gate count, we considered this design approach in our work. Despite its advantages, there occur some issues regarding appropriate placement of SWAP gates which have been discussed in [12] and thereby addressed by incorporating look-ahead policy.

In this work, we have modified the implementation process of the look-ahead policy from the earlier works [12-13] and thereby introduced an improved version of it. Here, in this paper three different look-ahead strategies have been considered that estimates the effect of SWAP gate insertion before non-adjacent gates by formulating the look-ahead estimation models generically based on the number of gates to be evaluated. However, the previous look-ahead works have

considered direct and straight-forward approaches where the cost estimation has been made by evaluating a specified number of gates for nearest neighbor synthesis of quantum circuits.

As permutation of qubits has an impact over SWAP gate optimization, so to determine the best possible option for moving the non-adjacent qubits of any gate for NN based design has been discussed next.

IV. Proposed Method

To determine the best possible option for moving the interacting qubits of any non-adjacent gate, here we have presented a heuristic design based on look-ahead strategy to estimate the impact for any option over the remaining gates of the circuit. In the design process, we evaluate all the possible options of any non-adjacent gate by estimating the impact in terms of SWAP gate count on the remaining gates. Based on this estimated impact, the best possible option is chosen. For this purpose, we compute the following cost metric:

$$T.C: F.C + V.C \quad (4)$$

where *T.C, F.C, V.C* denotes total cost, fixed cost and variable cost. However, fixed cost can further be expressed as follows.

$$F.C: G_R * C_R \quad (5)$$

where G_R and C_R represent the remaining gates and implementation cost of the respective gates (here, we have considered each quantum gate to be of unit cost).The variable cost *(V.C)* used in (4) is expressed below.

$$V.C: \sum c_i * n_i/G_R = c_1*n_1/G_R + c_2*n_2/G_R + \ldots + c_m*n_m/G_R \quad (6)$$

where the notations c_i, n_i, indicates the nearest neighbor costs and number of G_R gates that possesses the corresponding cost c_i.

The above variable cost expression indicates the computation of nearest neighbor cost of the following gates that can be expressed further in terms of contribution factor, *r* as follows.

$$V.C: c_1 * r_1 + c_2 * r_2 + \ldots + c_m * r_m = \sum_{i=1}^{m} (c_i * r_i) \quad (7)$$

where r_i is represented as $r_i = n_i / G_R$

The symbol G_R (remaining gates) used in all the previous mathematical expressions can be obtained for any gate g_i of a given circuit as follows.

$$G_R = (N - g_i) \bmod N \quad (8)$$

where the notation *N* denotes the total gate count of a given circuit.

As the fixed cost *(F.C)* does not have any influence over nearest neighbor optimization, so we haven't considered it in our design policy. Therefore, only variable cost estimation determines how to insert SWAP gates in appropriate positions. Consequently, the mathematical expression shown in (4) can be rewritten as follows:

$$T.C \approx V.C \quad (9)$$

Henceforth, variable cost has been considered equivalent to that of total cost. In our design methodology, we have implemented three different schemes concerning on how we should observe (look-ahead) the following gates to determine the suitable option for SWAP gate implementation is described next.

A. Collective Look-ahead Policy: The basic purpose of this scheme is to observe the target/control qubit positions of all the gates following any non-adjacent gate viz.g_i in a collective manner. As per this strategy, we choose that qubit order that results in a least possible variable cost over the remaining gates of a given circuit. In other words, we scan each gate *g(c,t)* of a given circuit *C* to determine whether its control *(c)*/target *(t)* qubits are placed adjacent or not. If its inputs are not placed adjacent (*NNC (g(c,t)) > 0)*, then we evaluate all possible rearrangements of the qubits by estimating the impact over the remaining gates, which can transform the gate *g(c,t)* into an NN-compliant one otherwise we proceed to next gate. Repeat this procedure until the last gate has been processed.

To elaborate it in more detail, we have described the entire algorithm formally as follows:

1) Consider the initial ordering of qubits as Π of a given circuit $\boldsymbol{C}$.

2) For each gate $g_i(c,t) \in \boldsymbol{C}$, if ($NNC(g_i(c,t)) > 0$) then

a) Determine the all possible permutations (p_i) of qubits that can make the corresponding gate adjacent ($NNC(g_i(c,t)) = 0$) and record them on a set $\boldsymbol{P}$ (denoted as permutation set).

b) For each $p_i \in \boldsymbol{P}$

i) Compute variable cost *(V.C)* for all G_R gates of $\boldsymbol{C}$.

c) Identify p_{min} that leads to a smallest overall cost (variable cost), $\boldsymbol{V.C_{min}}$ obtained in the previous step for a given circuit.

d) Insert SWAP gates before $g_i(c,t)$ such that the permutation p_{min} has been established.

3) Return NN-compliant circuit, $\boldsymbol{C_{NNC}}$ for given circuit $\boldsymbol{C}$.

The permutation p_{min} obtained by following this scheme may not be unique i.e. many such permutations (p_{min}) can be found that leads to identical smallest cost (variable) value. In such scenario, we resolve the conflict by randomly choosing any one of those permutations.

Example 1: *For the circuit shown in Fig. 2(a) with initial qubit ordering $\Pi = \{q_1, q_2, q_3, q_4\}$ where the first gate $g_1(4, 1)$has control/target inputs at lines 4 and 1 respectively. It does not hold the nearest neighbor criteria as NNC $(g_1) = 2 > 0$. Hence, a permutation that can make g_1 NN-complaint needs to be determined. There are three possible options in which SWAP gates can be added to make it adjacent viz. $p_1 = \{q_2, q_3, q_1, q_4\}$, $p_2 = \{q_1, q_4, q_2, q_3\}$ and $p_3 = \{q_2, q_1, q_4, q_3\}$ by evaluating the impact on the remaining gates ($G_R = (6 - 1) \bmod 6 = 5$) of the circuit as depicted in Fig. 3(a), 3(b), 3(c). Following permutation p_1 leads to a variable cost of 1.2 (V.C = 2 * 3/5+0 * 2/5 = 1.2) whereas p_2, p_3 results in a variable cost of 0.8 (V.C = 2 * 2/5+0 * 3/5 = 0.8) and 1 (V.C = 1 * 5/5 = 1) respectively. As p_2 is having a less impact compared to other permutations thereby it has been chosen for implementation. Literally, SWAP gates are added accordingly so as to establish the permutation order p_2 and the process repeats till the end of the circuit. The corresponding NN-compliant circuit has been represented in Fig. 4.*

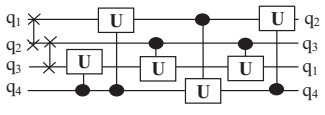


Fig. 3(a): Reordered circuit of Fig. 2(a) with p_1 permutation (*V.C = 1.2*)

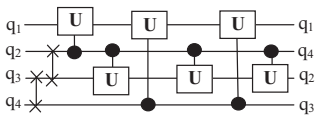


Fig. 3(b): Reordered circuit of Fig. 2(a) with p_2 permutation (*V.C = 0.8*)

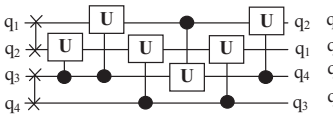


Fig. 3(c): Reordered circuit of Fig. 2(a) with p_3 permutation (*V.C = 1*)

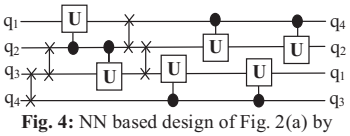


Fig. 4: NN based design of Fig. 2(a) by following collective look-ahead policy (using p_2)

B. *Bounded Windowing Policy:* It is considered as an alternative look-ahead policy that estimates the impact on a limited number of gates rather than over all the following gates collectively. To have a restricted view over the following gates, the concept of window size has been applied under this scheme. More specifically, instead of considering all the following gates only a certain number of gates as dictated by the window size are considered to choose a suitable permuted order of qubits. The number of gates needed to observe is determined by employing the following heuristic window size estimation as follows:

$$G_{size} = \lceil \sqrt{N} \rceil \quad (10)$$

The above expression determines the circuit window size (G_{size}) based on the total number of gates (N) of any given circuit. In order to arrange the qubits in a specific order, variable cost over G_{size} gates is estimated, which is obtained by replacing G_R with G_{size} in equation (6) as expressed below.

$$V.C: \sum c_i * n_i / G_{size} = c_1 * n_1 / G_{size} + c_2 * n_2 / G_{size} + \ldots + c_m * n_m / G_{size} \quad (11)$$

Similar to equation 7, the above expression can be represented further by replacing n_i / G_{size} with contribution factor r_i.

The prime motivation behind using this and the following look-ahead scheme is to prevent the impacts of gates appearing much late in the circuit to affect the decision on how to move the non-adjacent qubits of any gate. However, such an approach may turn out to be more beneficial for large benchmark circuits.

Furthermore, a formal specification of the corresponding approach has been provided for better understanding as follows:

1) Consider the initial ordering of qubits as Π of a given circuit $\boldsymbol{C}$.

2) For each gate $g_i(c,t) \in \boldsymbol{C}$, if ($NNC(g_i(c,t)) > 0$) then

a) Determine all possible permutations (p_i) of qubits that can make the corresponding gate adjacent ($NNC(g_i(c,t)) = 0$) and store them on a set $\boldsymbol{P}$ (denoted as permutation set).

b) For each $p_i \in \boldsymbol{P}$

i) Estimate $\boldsymbol{G_{size}}$ of circuit $\boldsymbol{C}$ with $\boldsymbol{N}$ gates.

ii) Compute variable cost $(V.C)$ for only $\boldsymbol{G_{size}}$ gates of $\boldsymbol{C}$.

c) Identify p_{min} that leads to a smallest cost (variable cost) value, $\boldsymbol{V.C_{min}}$ obtained in the previous step for a given circuit.

d) Insert SWAP gates before $g_i(c,t)$ in such a way that the permutation p_{min} is established.

3) Return NN-compliant circuit, $\boldsymbol{C_{NNC}}$ for given circuit $\boldsymbol{C}$.

Example 2: *Consider again the circuit shown in Fig. 2(a) and the look-ahead scheme discussed in Section B has been applied to assess all the possible options for moving the qubits of gate g_1 (4, 1)into adjacent positions. Instead of evaluating all the remaining gates (G_R), only for a certain number of gates the variable cost (V.C) has been estimated as* $\boldsymbol{G_{size}} = \lceil \sqrt{\mathbf{N}} \rceil = \lceil \sqrt{\mathbf{6}} \rceil = \boldsymbol{3}$*(using equation 10) of the corresponding circuit. Similar to example 1, there are three options (p_1, p_2, p_3) in which the first gate can be made NN-complaint but this time the costs (using eq. 11) has been computed by considering only the next 3 gates to choose the one for SWAP gate implementation as depicted in Fig. 5(a), Fig. 5(b) and Fig. 5(c) (where dashed rectangle represents the circuit window). As the option p_2 holds least variable cost thereby it has been chosen for implementation and the process continues for the rest of the gates which ultimately leads to an NN based design as shown in Fig. 6.*

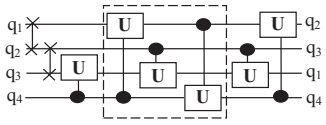


Fig. 5(a): Reordered circuit of Fig. 2(a) with p_1 permutation ($V.C = 1.33$)

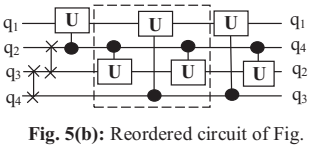


Fig. 5(b): Reordered circuit of Fig. 2(a) with p_2 permutation ($V.C = 0.66$)

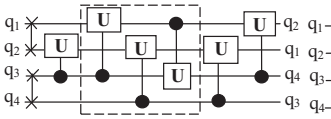


Fig. 5(c): Reordered circuit of Fig. 2(a) with p_3 permutation ($V.C = 1$)

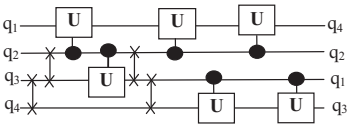


Fig. 6: NN based design of Fig. 1(a) by following bounded windowing policy (using p_2)

C. *Dynamic Windowing Policy:* Unlike, the previous windowing scheme (introduced in Section B), here a different windowing concept has been considered in which the window size differs for each non-adjacent gate g ($NNC\ (g) > 0$) of the given circuit. Basically, nearest neighbor cost of a current processing gate (say g_c) is considered as the essential driving factor that determines the dynamic nature of the window size. In other words, the change (increase/decrease) in size of the circuit window directly varies with the change (increase/decrease) in nearest neighbor cost of gate g_c, which in turn alters with the exactly previous non-adjacent gate g_p but while adapting this strategy, we have to set the initial window size of the circuit to previously computed window value i.e. G_{size} (used in Section B). Mathematically, this windowing policy can be expressed as follows:

$$W \propto C \quad (12)$$
$$W = K.\ C \quad (13)$$
$$W/C = K \quad (14)$$
$$W_c/C_c = K \quad (15)$$
$$W_p/C_p = K \quad (16)$$

Replacing the constant K from Eq. (16) into Eq. (15) derives the following mathematical relation.

$$W_c/C_c = W_p/C_p \quad (17)$$
$$W_c = \lceil Wp/Cp * Cc \rceil \quad (18)$$

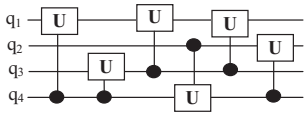


Fig. 7(a): original circuit with NNC=6

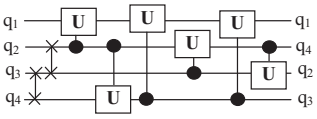


Fig. 7(b): Reordered circuit with option p_1($V.C = 1$)

The expression written in Eq. 18 derives the window size of the current gate g_c. The notations W_c, W_p, C_c, C_p represents the window sizes and nearest neighbor costs of the current and previous non-adjacent gates viz. g_c, g_p respectively.

In this algorithm, we have considered nearest neighbor cost as the pivotal element to determine the window size dynamically. The change (increase/decrease) in nearest neighbor cost proportionally affects the number of possible qubit permutations needed to obtain a NN-compliant gate. Such a relationship between cost and permutation in turn influence the search space for determining the best possible option in which SWAP gates can be inserted. Consequently, we have adjusted the window size according to the change in nearest

neighbor costs of gates of the given circuit. Suppose, if the nearest neighbor cost of a gate g_c increases/decreases with respect to the previous one g_p then we would define the window size of g_c by adjusting the window value of g_p proportionally with the change in cost value of g_c otherwise the previous window value of g_p is set for gate g_c.

To apprehend the idea more clearly, we have explained it with the help of an illustrative example as discussed next.

Example 3: *Consider the circuit shown in Fig. 7(a) having overall nearest neighbor cost of 7. The first gate g_1 has cost of 2, so it needs to be made NN-complaint by evaluating all the possible permutation order. There are three options viz. p_1, p_2, p_3 available to make g_1 adjacent by inserting SWAP gates as represented in Fig. 7(b) - 7(d). To evaluate these options, the variable costs have been estimated over the initial circuit window set as G_{size} (same as the one used in bounded policy). Hence, the costs has been computed over the next 3 gates (G_{size}= $\lceil\sqrt{N}\rceil$ = $\lceil\sqrt{6}\rceil$ = 3) and option p_3 has been implemented due to least cost. By examining the resulting circuit obtained after establishing p_3 shown in Fig. 7(d), find that gate g_3 has a cost of 1. To make this gate NN-compliant, there are two possible permutations viz. p_1 = {q_2, q_4, q_1, q_3}, p_2 = {q_2, q_1, q_3, q_4} that are evaluated by observing the number of gates obtained using equation 18 which is 2 ($W_3 = W_1/C_1 * C_3 = \left\lceil\frac{3}{2} * 1\right\rceil$=2). Unlike g_1, now the impact over the next two gates of g_3 i.e g_4 and g_5 has been considered. In this manner the procedure continues till the last gate has been processed and finally the resultant NN-compliant circuit is obtained in Fig. 8.*

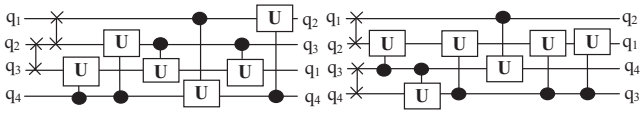


Fig. 7(c): Reordered circuit with option p_2(*V.C=1*)

Fig. 7(d): Reordered circuit with option p_3 (*V.C=0.66*)

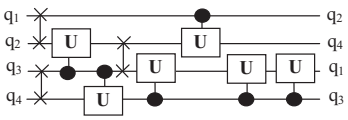


Fig. 8: NN-compliant circuit of Fig. 8(a) after using dynamic window scheme

IV. Experimental Results

In this section, the experimental results from our design methodology have been summarized. The scheme has been implemented in C and executed on a machine with Intel core i5 processor with 3.30 GHZ clock and 4GB RAM.

The performance evaluation of our methodology has been carried out over various benchmark suites [17] and all the computed results have summarized in Table1 and Table2.

From the experimental evaluations, it can be observed that our design methodology provides significant improvements over SWAP gate count against the related works results. Overall, an average improvement of about 19.54% (35% in the best case) over [13], 24.27% (64.86% in the best case) over [7] and 5.54%, 25% (12.5% and 42.10% in the best case) over [12] has been registered (from Table 1). On the other hand, our design approach also turns out to be feasible for large sized benchmarks as summarized in Table 2. An exception to be mentioned that in Table2, the results from *bounded* and *dynamic* technique are same and it might be due to the fact that the benchmarks considered there are our defined ones which follow a symmetrical pattern while the employed heuristics and applied strategies are quite different in all the three schemes.

VI. Conclusion

In this work, a look-ahead based design methodology has been developed to transform quantum circuits to their equivalent nearest neighbor structures. Transforming the quantum circuits to NN structures was not only our sole objective, but how to minimize the SWAP usages in the circuits also was under consideration. In order to explore the possible ways to reduce this parameter further, we have shown three different look-ahead policies *(Collective, Bounded* and *Dynamic)* and all the design techniques have been successfully tested over different benchmark circuits. While comparing with state-of-art NN techniques, this scheme has shown significant improvement in SWAP overhead. Future work will now focus on investigating the proposed policies for optimizing quantum circuits for further physical realizations such as IBM QX Architectures [23].

VI. References

[1] P.W Shor,"Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,"*SIAM J. Comput.* 26 (5), 1484-1509 (1997).

[2] L.K Grover, "A fast quantum mechanical algorithm for database search,"*In. Symposium on the Theory of Computing*, pp. 212-219 (1996).

[3] D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a largescale ion-trap quantum computer,*Nature*, 417(6890):709–711, 2002.

[4] J. Taylor, J. Petta, A. Johnson, A. Yacoby, C. Marcus, and M. Lukin. Relaxation, dephasing, and quantum control of electron spins in double quantum dots. *Physical Review B*, 76(3):035315, 2007.

[5] B. Criger, G. Passante, D. Park, and R. Laflamme. "Recent advances in nuclear magnetic resonance quantum information processing". *Philosophical Transactions of the Royal Society of London A: Mathematical,Physical and Engineering Sciences*, 370(1976):4620–4635, 2012.

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

[7] M. Saeedi, R. Wille, R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quant. Inf. Proc*., 10(3):355–377, 2011.

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

[9] M. Perkowski, M. Lukac, D. Shah, and M. Kameyama. Synthesis of quantum circuits in linear nearest neighbor model using positive Davio lattices. 2011.

[10] A. Chakrabarti, S. Sur-Kolay, and A. Chaudhury. Linear nearest neighbour synthesis of reversible circuits by graph partitioning. *arXiv preprint arXiv:1112.0564*, 2011.

[11] R. Wille, A. Lye, and R. Drechsler, "Exact reordering of circuit lines for nearest neighbor quantum architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1818–1831, 2014.

[12] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler, "Look-ahead schemesfor nearest neighbor optimization of 1D and 2D quantum circuits," in *Proc. ASP Design Autom. Conf.*, Jan 2016, pp. 292–297.

[13] 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, no. 1, pp. 62–72, Feb 2016.

[14] R. Wille, A. Lye and R. Drechsler, "Optimal SWAP gate insertion for nearest neighbor quantum circuits," in *Proc. ASP Design Automation Conf.* Suntec, Singapore: IEEE, 2014, pp. 489-494.

[15] M. H. A. Khan, "Cost reduction in nearest neighbour based synthesis of quantum boolean circuits," *Engineering Letters*, pp. 1–5, (2008).

[16] M. Amy, D. Maslov, M. Mosca, "Polynomial-time T-depth Optimization of Clifford+T circuits via Matroid partitioning," *6th International Conference on Reversible Computation*, Japan (*2014*).

[17] Revlib: An online resource for reversible functions and reversible circuits. URL: http://www.revlib.org/.

[18] A. Bhattacharjee, C. Bandyopadhyay, R. Wille, R. Drechsler, H. Rahaman, "A Novel Approach for Nearest Neighbor Realization of 2D Quantum Circuits," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2018*

[19] A. Bhattacharjee, C. Bandyopadhyay, H. Rahaman, "A Heuristic Qubit Placement Strategy for Nearest Neighbor Realization in 2D Architecture" *IEEE 22th International Symposium on VLSI Design and Test-2018*, India.

[20] R. Wille, N. Quetschlich, Y. Inoue, N. Yasuda, and S. Minato. Using πDDs for Nearest Neighbor Optimization of Quantum Circuits. In Conference on Reversible Computation, pages 181-196, 2016.

[21] A. Zulehner, S. Gasser, and R. Wille. Exact Global Reordering for Nearest Neighbor Quantum Circuits Using A*. In Conference on Reversible Computation, 185-201, 2017.

[22] A. Kole, K. Datta, I. Sengupta, "A New Heuristic for N-Dimensional Nearest Neighbour Realization of a Quantum Circuit". *In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12 (2017), doi 10.1109/TCAD.2017.2693284.

[23] A. Zulehner, A. Paler, and R. Wille. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), 2018.

Table 2: Results from large size benchmarks

Benchmarks	No. of qubits	Gate Count	Method 1 (collective) SWAPs	Method 2 (bounded) SWAPs	Method 3 (dynamic) SWAPs
add17_218	17	42	24	24	24
add18_218	18	42	27	23	23
add19_218	19	48	32	28	28
add20_218	20	50	33	29	29
add21_218	21	50	32	28	28
rdom_22	22	55	19	19	19
add23_218	23	58	38	34	34
rdom_24	24	47	44	44	44
add8_172	25	64	42	38	38
rdom_27	27	53	26	26	26
add29_218	29	74	48	44	44
rdom_30	30	59	29	30	30
rdom_33	33	65	32	33	33
add16_174	49	128	82	78	78

Table 1: Comparison with state-of-the-art NN techniques

Benchmark names	No. of qubits	Gate count	Method 1 (collective) SWAPs	Method 2 (bounded) SWAPs	Method 3 (dynamic) SWAPs	Best result	Prev. work [13]	Prev. work [7]	Prev. work [12]	Prev. work [12]	% improvement over			
											[13]	[7]	[12]	[12]
4gt4-v0_80	6	44	32	34	35	32	36	36	-	-	11.11	11.11	-	-
4gt10-v1_81	5	36	25	22	22	22	32	29	24	38	31.25	24.13	8.33	42.10
4mod5-v1_23	5	24	15	15	15	15	15	18	-	-	0.0	16.66	-	-
4gt11_84	5	7	4	4	4	4	5	5	-	-	20	20	-	-
rd32-v0_67	4	8	4	4	4	4	-	4	-	-	-	0.0	-	-
3_17_13	3	14	6	6	6	6	6	5	6	6	0.0	-20	0.0	0.0
4gt13-v1_93	5	17	10	9	9	9	10	10	-	-	10	10	-	-
4_49_17	4	32	15	15	15	15	19	16	-	-	21.05	6.25	-	-
hwb4_52	4	23	13	9	9	9	10	14	-	-	10	35.71	-	-
4gt5_75	5	22	19	16	13	13	20	20	-	-	35	35	-	-
alu-v4_36	5	32	21	16	19	16	22	20	-	-	27.27	20	-	-
aj-e11_165	5	60	33	33	33	33	35	43	33	42	5.71	23.25	0.0	21.42
4gt12-v1_89	6	53	33	33	33	33	37	35	-	-	10.81	5.71	-	-
4mod7-v0_95	5	40	24	26	22	22	-	30	-	-	-	26.66	-	-
mod5adder_128	6	87	65	53	55	53	65	79	46	85	18.46	32.91	15.21	37.64
rd53_135	7	78	75	68	66	66	82	96	66	85	19.51	31.25	0.0	22.35
ham7_104	7	87	72	70	63	63	83	86	72	84	24.09	26.74	12.5	25
mod8-10_177	6	109	77	71	77	71	77	77	-	-	7.79	7.79	-	-
hwb5_55	5	109	66	68	64	64	75	86	66	101	14.66	25.58	3.03	36.63
hwb6_58	6	146	111	105	98	98	127	140	111	146	22.83	30	11.71	32.87
rd73_140	10	76	64	44	55	44	63	61	-	-	30.15	27.86	-	-
QFT7	7	21	24	18	28	18	-	18	18	23	-	0.0	0.0	21.73
QFT8	8	28	31	31	34	31	-	41	31	33	-	24.39	0.0	6.06
QFT9	9	36	49	43	50	43	-	66	49	53	-	34.84	12.24	18.86
QFT10	10	45	64	60	67	60	-	96	64	67	-	37.5	6.25	10.44
0410184_169	14	90	39	39	39	39	-	111	-	-	-	64.86	-	-
rd84_142	15	112	112	76	95	76	112	148	-	-	32.14	48.64	-	-