Quantum Inf Process (2011) 10:355–377
DOI 10.1007/s11128-010-0201-2

Synthesis of quantum circuits for linear nearest neighbor architectures

Mehdi Saeedi · Robert Wille · Rolf Drechsler

Received: 11 June 2010 / Accepted: 30 September 2010 / Published online: 19 October 2010

Abstract While a couple of impressive quantum technologies have been proposed, they have several intrinsic limitations which must be considered by circuit designers to produce realizable circuits. Limited interaction distance between gate qubits is one of the most common limitations. In this paper, we suggest extensions of the existing synthesis flow aimed to realize circuits for quantum architectures with linear nearest neighbor interaction. To this end, a template matching optimization, an exact synthesis approach, and two reordering strategies are introduced. The proposed methods are combined as an integrated synthesis flow. Experiments show that by using the suggested flow, quantum cost can be improved by more than 50% on average.

Keywords Quantum circuits · Logic synthesis · Nearest neighbor architectures · Template matching

A preliminary version of this paper has been presented in [1].

M. Saeedi (✉)
Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
e-mail: msaeedi@aut.ac.ir

R. Wille · R. Drechsler
Institute of Computer Science, University of Bremen, Bremen, Germany

R. Wille
e-mail: rwille@informatik.uni-bremen.de

R. Drechsler
e-mail: drechsle@informatik.uni-bremen.de

1 Introduction

Since the invention of the integrated circuit in 1958, the number of transistors in such circuits has doubled approximately every two years (also known as Moore's Law). Currently, semiconductor technology has advanced the world towards more powerful systems by decreasing the transistor size. However, further miniaturization is beginning to appear insoluble due to the density of power dissipation and the impossibility to realize patterning features approaching the atomic scale.

The difficult barriers to the ongoing improvements in semiconductor technology have intensified the attraction of alternative computing paradigms such as quantum computing. It has been shown that quantum computing could improve the rate of advance in processing power at least for several applications [2]. In principle, there are several problems that cannot be executed on a classical Turing machine as efficiently as on a quantum computer. Quantum computers would provide exponential speedups on several problems including factoring of numbers and simulating the quantum-mechanical behavior of physical systems [3]. However, several obstacles exist in the way of physically implementing scalable quantum computers.

While several impressive physical realizations have been proposed for quantum computers (see [4] for a classification scheme of different quantum computing technologies), all of these technologies have serious intrinsic limitations [5]. Among the different technological constraints, limited interaction distance between gate qubits is one of the most common ones. Although arbitrary-distance interaction between qubits is possible in quantum computer technologies with moving qubits (for example in a photon-based system [6]), restrictions exist in other quantum technologies. In fact, many physical quantum computer proposals only permit interactions between adjacent (nearest neighbor) qubits [7]. For example, trapped ions (e.g., [8]), liquid nuclear magnetic resonance (NMR) (e.g., [9]), and the original Kane model [10] have been designed based on the interactions between linear nearest neighbor (LNN) qubits. The LNN architecture is often considered as an appropriate approximation to a scalable quantum architecture. If one can show that a circuit can efficiently be realized using an LNN architecture, it can be run in many other architectures as well [11].

The efficient realization of a given quantum algorithm for the LNN architectures is an active research area. In the recent years, the effect of restricted interactions on several specific quantum algorithms has been studied. For example, the physical implementation of the quantum Fourier transformation (QFT) [12], Shor's factorization algorithm [7,13], quantum addition [14], and quantum error correction [15] for the LNN architectures have been explored in the past. Besides that, researchers also considered the effects of LNN architectures on the synthesis of general quantum/reversible circuits. In [16] and [17], the worst-case synthesis cost of a general unitary matrix under the nearest neighbor restriction has been discussed. It has been shown that restricting CNOT gates to nearest neighbor interactions increases CNOT count of [17] by at most a factor of 9. The authors of [18] showed that translating an arbitrary circuit to the LNN architectures requires a linear increase in the quantum cost with respect to the number of qubits. In [19–21], heuristic methods for converting an arbitrary circuit to its equivalent on the LNN architectures have been proposed. However, their performance is limited as discussed later.

In this paper, we suggest extensions of the existing synthesis flow aimed to realize circuits for LNN architectures. We show that with a naive treatment of the LNN restriction, quantum circuits require up to one order of magnitude higher quantum cost in the LNN architectures. In contrast, if this restriction is explicitly considered by the proposed synthesis flow, this increase can be reduced by more than 50% on average (83% in the best case). To this end, the following approaches are proposed:

- An improved template-matching post-synthesis optimization method that reduces the circuit cost for LNN architectures,
- an exact synthesis method for small functions realizing circuits with nearest neighbor interaction, and
- reordering strategies, which modify the initial qubit locations in order to reduce the distance between non-neighbored qubits.

The remainder of this paper is organized as follows. In Sect. 2, basic concepts are introduced. Next, we briefly review the naive synthesis flow for LNN architectures in Sect. 3. Followed by this, Sect. 4 describes the proposed synthesis and optimization approaches with explicit consideration of the LNN limitation in detail. How to combine the respective approaches as an integrated flow is sketched in Sect. 5. Finally, experimental results are given in Sect. 6 and conclusions are drawn in Sect. 7, respectively.

2 Background

2.1 Reversible logic

A function $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ over variables $X = \{x_1, \ldots, x_n\}$ is reversible if it maps each input assignment to a unique output assignment. Such function must have the same number of input and output variables. In this paper, n is particularly used to refer to the number of inputs/outputs. A circuit realizing a reversible function is a cascade of reversible gates. Common reversible gates include:

- A *multiple control Toffoli gate* t_m has the form $t_m(C, t)$, where $C = \{x_{i_1}, \ldots, x_{i_m}\} \subset X$ is the set of *control lines* and $t = \{x_j\}$ with $C \cap t = \emptyset$ is the *target line*. The value of the target line is inverted iff all control lines are assigned to 1. For $m = 0$ and $m = 1$, the gates are called *NOT* gate and *CNOT* gate, respectively. For $m = 2$, the gate is called C^2NOT gate or *Toffoli* gate.
- A *multiple control Fredkin gate* f_m has two target lines and m control lines. The gate interchanges the values of the target lines iff the conjunction of all m control lines evaluates to 1. For $m = 0$, the gate is called *SWAP* gate.
- A *Peres gate* P has one control line x_i as well as two target lines x_{j_1} and x_{j_2}. It represents a $t_2(\{x_i, x_{j_1}\}, x_{j_2})$ and a $t_1(\{x_i\}, x_{j_1})$ in a cascade.

Reversible logic has applications in various fields including quantum computation.

2.2 Quantum logic

A quantum bit, qubit in short, can be realized by a physical system such as a photon. Each qubit has two basic states $|0\rangle$ as well as $|1\rangle$ and can get any linear combination of its basic states (called superposition, as shown in (1) where α and β are complex numbers).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

An n-qubit quantum gate is a device which performs a specific $2^n \times 2^n$ unitary operation on selected n qubits in a specific period of time. A matrix U is unitary if $UU^\dagger = I$ where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix. Previously, various quantum gates with different functionalities have been introduced. For examples, Hadamard (H), Controlled-V, and Controlled-V$^+$ gates are defined by the following unitary matrices:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

$$V = \frac{1+i}{2} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix} \tag{3}$$

$$V^+ = \frac{1-i}{2} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} \tag{4}$$

2.3 Synthesis cost

Each Toffoli, Fredkin, and Peres gate can be *decomposed* into a quantum circuit composed of a sequence of *elementary quantum gates* [2]. Each elementary gate performs a single physical operation in a certain quantum computing technology. The number of elementary gates required to realize a given reversible gate is called *quantum cost*. It has been shown that NOT, CNOT, Controlled-V, and Controlled-V$^+$ gates can efficiently be realized in quantum computer technologies [22]. These gates are usually considered as elementary gates for reversible Boolean functions [23]. Thus, we stay with this definition in the following sections. However, in other technologies not only this restricted set, but all one-qubit gates and all two-qubit gates, respectively, are considered as elementary gates [2]. This is separately considered in the experimental evaluation of the proposed approach in Sect. 6.

Figure 1a shows a Toffoli gate and a Fredkin gate in a cascade. The resulting (decomposed) quantum circuit is depicted in Fig. 1b. Here, the control lines are denoted by $\bullet$ while the target lines are denoted by $\oplus$, $\times$, a V box, or a V$^+$ box, respectively. As can be seen, a t_2 gate is decomposed into 5 elemantary gates, while a f_m gate is decomposed into 7 elemenary gates, respectively. For larger gates, the respective decomposition depends on the number $n - m$ of unused circuit lines: For $n \geq 5$ and $m \in \{3, 4, \cdots \lfloor n/2 \rceil\}$, a t_m gate can be decomposed into a linear-size circuit which contains $12m - 22$ elementary gates. In addition, for $n \geq 7$, a t_{n-2} gate can be decomposed into $24n - 88$ elementary gates with no auxiliary bits [24]. Finally, a t_{n-1} gate

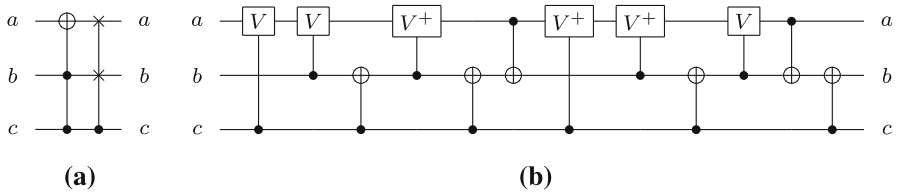


Fig. 1 A reversible circuit and its decomposed circuit

can be decomposed into $2^n - 3$ elementary gates if no unused circuit line is available [23]. The cost of a f_m gate ($1 \leq m \leq n-2$) is the cost of a t_{m+1} gate plus two [2]. Obviously, always the most efficient decomposition is applied.

3 The naive synthesis flow for the LNN architectures

Reversible circuits can be synthesized using multiple control Toffoli gates first that are afterwards mapped to elementary quantum gates. On the other hand, elementary gates can be directly applied during the synthesis process. While for the latter case, only small circuits have been determined so far (e.g., see [25,26]), approaches for Toffoli network synthesis can handle larger functions and circuits (e.g., see [27–33]). However, both approaches often lead to sub-optimal circuits with respect to the LNN architectures since the number of elementary gates (i.e., quantum cost) are improved without an explicit consideration of the LNN restriction. The same problem exists for quantum circuit synthesis algorithms [16,17].

In order to measure the cost of the LNN restriction, a cost metric is defined. Consider a 2-qubit quantum gate g where its control and target are placed at the c^{th} line and at the t^{th} line ($0 \leq c, t < n$), respectively. The *NNC (nearest neighbor cost)* of g is defined as $|c - t - 1|$ (i.e., distance between control and target lines). The NNC of a circuit is defined as the sum of the NNCs of its gates. Optimal NNC for a circuit is 0 where all quantum gates are either 1-qubit or 2-qubit gates performed on adjacent qubits.

Since synthesis algorithms may use several non-elementary gates during synthesis, all non-elementary gates should be decomposed into a set of elementary unit-cost gates for physical implementation. Decomposition methods proposed in [17,23,24] are extensively used for this purpose. On the other hand, after applying one of the available synthesis and/or decomposition algorithms, non-optimal circuits with respect to NNC may result. For example, Fig. 2a shows the standard decomposition of a Toffoli gate which leads to an NNC value of 1. To make this circuit applicable for the LNN architectures, SWAP gates must be applied for each non-adjacent quantum gate. More precisely, SWAP gates are added in front of each gate g with non-adjacent control and target lines to "move" the control (target) line of g towards the target (control) line until they become adjacent. Afterwards, SWAP gates are added to restore the original ordering of circuit lines. Similar methods have been applied by previous synthesis methods considering the LNN restriction [16,17,19–21,34].

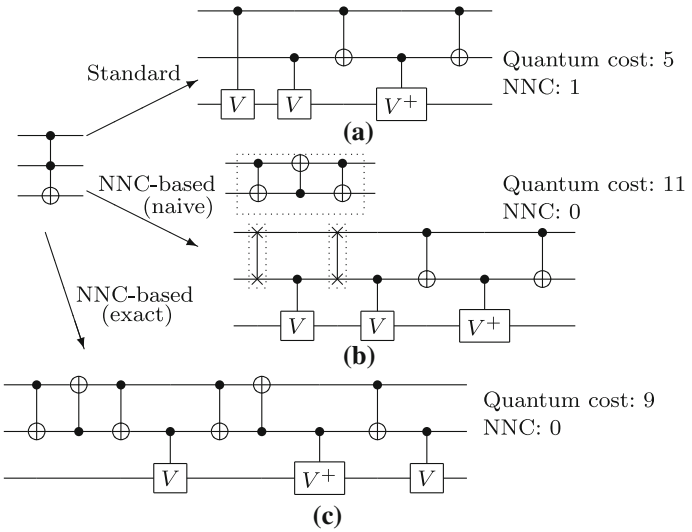


Fig. 2 Different decompositions of a Toffoli gate

Example 1 Consider the standard decomposition of a Toffoli gate as depicted in Fig. 2a. As can be seen, the first gate is non-adjacent. Thus, to achieve NNC-optimality, SWAP gates in front and after the first gate are inserted (see Fig. 2b). Since each SWAP gate requires 3 elementary quantum gates[1], this increases the total quantum cost to 11, but leads to an NNC value of 0.

By inserting SWAP gates consecutively for each non-adjacent gate, a quantum circuit with NNC of 0 (and thus applicable to LNN architectures) can be determined in linear time. This method is denoted by *naive NNC-based decomposition* in the rest of this paper. However, as can easily be seen, synthesizing quantum circuits for LNN architectures using this method (or similar approaches like [16,17,19–21]) often leads to a significant increase in the quantum cost. In contrast, often smaller realizations (with NNC of 0) are possible. As an example, consider Fig. 2c that shows an NNC-optimal decomposition with quantum cost of 9 (instead of 11). In the next sections, a synthesis flow is described that explicitly takes NNC into account. Hence, better quantum circuit realizations for the LNN architectures can be found as shown in the experimental results section.

4 Explicit consideration of NNC

In this section, we propose new synthesis and optimization approaches that explicitly take NNC into account. More precisely, a template-matching post-optimization

[1] As mentioned above, in certain quantum technologies all two-qubit gates are considered as elementary gates. Thus, in this case the SWAP gate is seen as an elementary gate increasing the costs by 1, instead of 3. While this special case is not considered in the following, it is separately evaluated in the experimental evaluation in Sect. 6.

algorithm is introduced to simplify the circuits resulted from the existing synthesis flow. Furthermore, an exact synthesis approach is proposed that determines NNC-optimal circuits with minimal quantum cost. The resulting circuits can later be exploited to optimize large circuits. Finally, two heuristic approaches are introduced that modify the initial qubit locations in order to remove unnecessary SWAP gates and therewith to reduce the cost.

4.1 NNC-based template matching

The idea of exploiting templates has originally been proposed in [35] and extended in [36] for LNN architectures. In this section, further templates for LNN architectures are proposed that outperform the previous ones as shown below.

Two neighboring gates can be interchanged if the target line of the first gate is not equal to the control lines of the second gate and vice versa (*moving rule*). In addition, two neighboring SWAP gates with the same target lines can be removed (*deletion rule*). The general idea of template matching is to replace a cascade of reversible gates by a different cascade with the same functionality and afterwards applying the moving and deletion rules to optimize the circuit. By considering this approach, templates with one, two, and three SWAP gates are proposed in Fig. 3a–c, respectively. The U_i boxes thereby represent any one-qubit or two-qubit gate. A U_i^R box represents the same gate as a U_i box, but probably with interchanged control and target lines.

As an example, consider the circuit shown in Fig. 4a with quantum cost of 16. By applying a template introduced in Fig. 3b, the circuit shown in Fig. 4b results. Now, a 1-SWAP template (Fig. 3a) can be applied leading to the circuit depicted in Fig. 4c. Finally, by applying the deletion rule, gates can be removed and, the final quantum cost is improved by about 37%. The final circuit is shown in Fig. 4d.

The authors of [36] introduced a set of nearest neighbor templates for Toffoli and CNOT combinations. It can be verified that the introduced templates in Fig. 6b,c of [36] can be found by applying the deletion rule. Moreover, consider the circuit shown

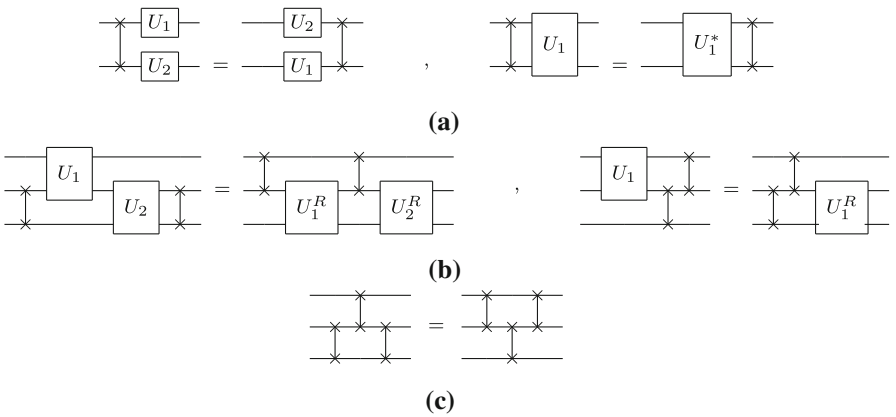


Fig. 3 Proposed templates

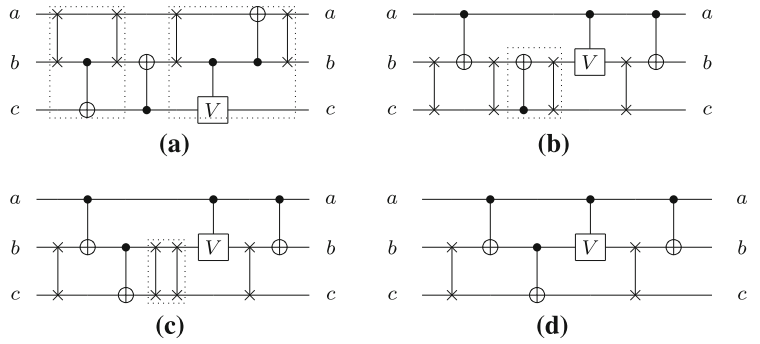


Fig. 4 Application of the proposed NNC-based templates

in Fig. 5a which includes a Toffoli-CNOT combination. Figure 5b illustrates a template as proposed in Fig. 6a of [36]. This circuit still has to be decomposed to elementary gates leading to a circuit with quantum cost of 30 as shown in Fig. 5c. On the other hand, consider the circuit shown in Fig. 5d obtained by applying the naive method on the circuit of Fig. 5a. The equivalent circuit after applying the templates introduced in Fig. 3b is given in Fig. 5e. Applying the deletion rule finally leads to a circuit with quantum cost of 24 as shown in Fig. 5f. Thus, applying the templates proposed in this paper in conjunction with the deletion rule improves the result of [36] by 20%.

Besides that, the efficiency of the proposed templates is illustrated by the following practical relevant example.

Example 2 Consider the circuit shown in Fig. 6a which is the approximate quantum Fourier transform circuit (AQFT) [37] with 36 SWAP gates obtained by the method of [12] for 8 qubits and an approximation parameter of 5. Note that R_k is the rotation by $2\pi/2^k$ and H is the Hadamard gate. Figure 6b is an equivalent circuit with 24 SWAP gates constructed by a method recently introduced in [21]. On the other hand, applying the proposed templates on the result of [12] leads to the circuit with 20 SWAP gates illustrated in Fig. 6c.

4.2 Exploiting exact synthesis

A few exact synthesis methods for quantum circuits have recently been introduced. They generate quantum circuits with minimal quantum cost (for examples see [25,26]). However, no approach to determine optimal circuits for LNN architectures has been proposed so far. In this section, an exact synthesis algorithm is proposed to construct quantum circuits with *both*, minimal quantum cost and minimal NNC.

The developed approach is similar to the one introduced in [26]. Here, the synthesis problem is expressed as a sequence of *Boolean satisfiability* (SAT) instances. For a given function f, it is checked if a circuit with c gates realizing f exists. Thereby, c is initially assigned to 1 and increased in each iteration if no realization is found.

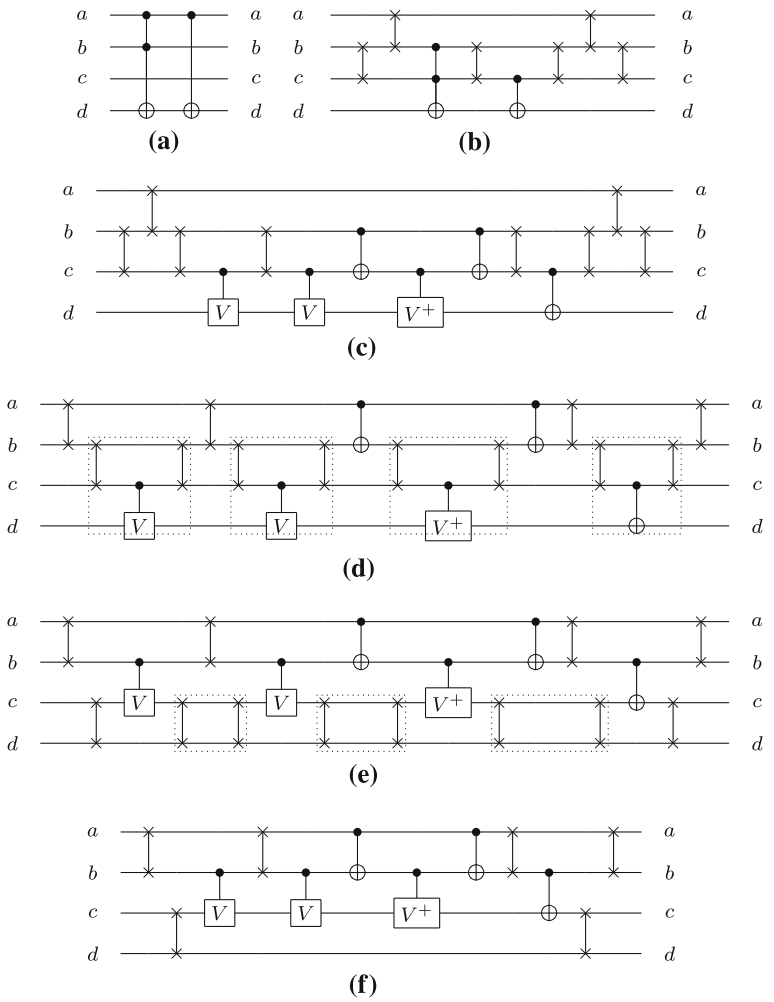


Fig. 5 An existing nearest neighbor template for a Toffoli-CNOT combination and our proposed template

More formally, for a given c and a reversible function $f : \mathbb{B}^n \to \mathbb{B}^n$, the following SAT instance is created:

$$\Phi \wedge \bigwedge_{i=0}^{2^n-1} ([\overrightarrow{inp}_i]_2 = i \wedge [\overrightarrow{out}_i]_2 = f(i)),$$

where

- $\overrightarrow{inp}_i$ is a Boolean vector representing the inputs of the network to be synthesized for truth table line i,
- $\overrightarrow{out}_i$ is a Boolean vector representing the outputs of the network to be synthesized for truth table line i, and

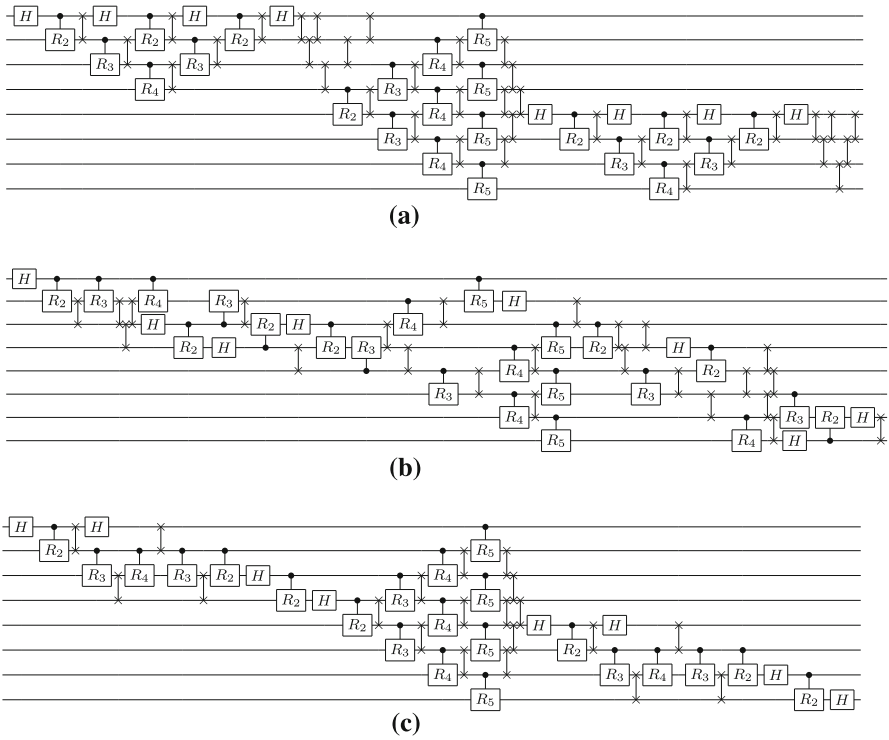


Fig. 6 Circuits realizing the Approximate Quantum Fourier Transform (AQFT)

– Φ is a set of constraints representing the synthesis problem for a given gate library.

The difference in comparison to [26] is that the constraints in Φ do not represent the whole set of elementary quantum gates and a restricted gate library with only adjacent gates is applied.

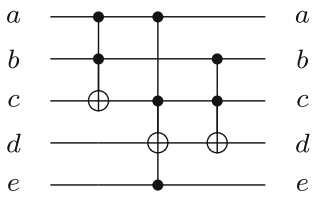
Although solving the generated SAT instances using a modern SAT solver can produce optimized circuits, the applicability of the exact method is limited to functions with a small number of qubits and gates due to the exponential search space. Actually, the proposed exact method is sufficient to construct minimal realizations with respect to both quantum cost and NNC for a set of Toffoli and Peres gate configurations as shown in Table 1. However, these optimal circuits can be exploited to improve the naive NNC-based decomposition method. More precisely, once an exact NNC-optimal quantum circuit for a function is available (denoted by *macro* in the following), the decomposition from the naive approach is replaced by the optimal circuit. The following example illustrates the idea.

Example 3 Reconsider the decomposition of a Toffoli gate as depicted in Fig. 2. By applying the proposed exact synthesis approach, an NNC-optimal quantum circuit as shown in Fig. 2c results. In comparison to the naive method (see Fig. 2b), this reduces the quantum cost from 11 to 9 while still ensuring NNC optimality.

Table 1 List of available macros

n	Macro	Cost		Impr. (%)
		Naive	Exact	
3	P({a,b},c), P({c,b},a)	12	8	33
3	P({a,c},b), P({c,a},b)	24	12	50
4	P({a,b},d), P({d,c},a)	30	11	63
3	t2({a,b},c), t2({c,b},a)	11	9	18
4	t2({a,b},d), t2({d,c},a)	29	12	59
3	t2({a,c},b)	17	13	24
4	t2({d,b},a), t2({a,c},d)	29	13	55

Fig. 7 Circuit of example 3



After finding the optimal decomposition of a given gate, it can be used as a macro to simplify other circuits. For example, consider the circuit shown in Fig. 7. Here, for the second gate the naive method is applied and SWAPs are added, while for the remaining ones the obtained macro is used. This enables a quantum cost reduction from 96 to 92.

Moreover, Fig. 8b,c show the NNC-optimal circuit of the Peres gate obtained by the naive and by the exact approach, respectively. As illustrated, applying the naive approach leads to quantum cost of 28 while the optimal circuit has only quantum cost of 11.

In total, we generated 13 macros as listed in Table 1 together with the respective costs in comparison to the costs obtained by using the naive method. As can be seen, exploiting these macros reduces the cost for each gate by up to 63%. The effect of these macros on the decomposition of larger circuits is considered in the experimental results section in detail.

4.3 Reordering circuit lines

Applying the approaches introduced so far leads to an increase in the quantum cost for each non-adjacent gate. In contrast, by modifying the ordering of the circuit lines, some of the additional costs can be saved. As an example, consider the circuit in Fig. 9a with quantum cost 3 and an NNC value of 6. By reordering the lines as shown in Fig. 9b, the NNC value can be reduced to 1 without increasing the total quantum cost. It is worth noting that manipulating the line order has been previously done to reduce the quantum cost (e.g., in [27,38]). To determine which lines should be reordered, two

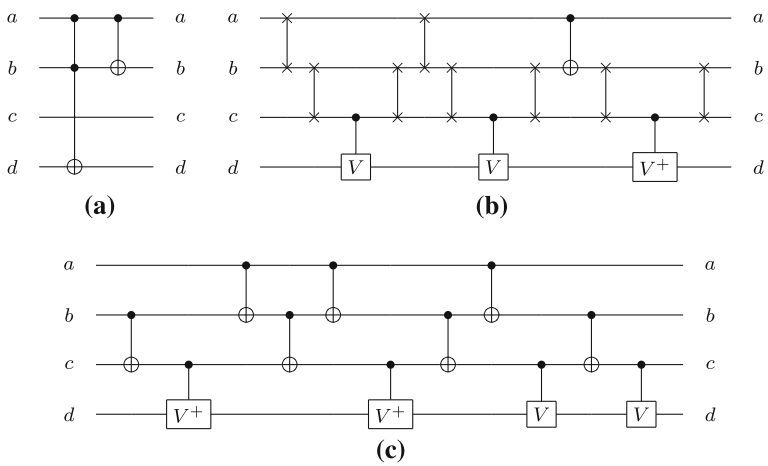


Fig. 8 NNC-based synthesis of a Peres gate

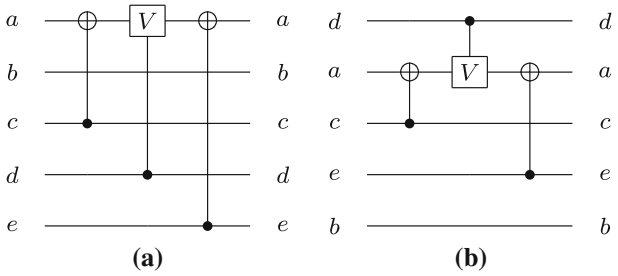


Fig. 9 Reordering circuit lines

heuristic methods are proposed in the following. The former one changes the ordering of the primary inputs and outputs according to a global view while the latter one applies a local view to assign the line ordering.

4.3.1 Global reordering

After applying the standard decomposition algorithms [23,24], a cascade of 1-and 2-qubit gates is generated. Now, an ordering of the circuit lines which reduces the total NNC value is desired. To do that, the "contribution" of each line to the total NNC value is calculated. More precisely, for each gate g with control line i and target line j, the NNC value is calculated. This value is added to variables imp_i and imp_j which are used to save the impacts of the circuit lines i and j on the total NNC value, respectively. Next, the line with the highest NNC impact is chosen for reordering and placed at the middle line (i.e., swapped with the middle line). If the selected line is the middle line itself, a line with the next highest impact is selected. This procedure is repeated until no better NNC value is achieved. Finally, SWAP operations as described in the previous sections are added for each non-adjacent gate. The following example illustrates the idea.

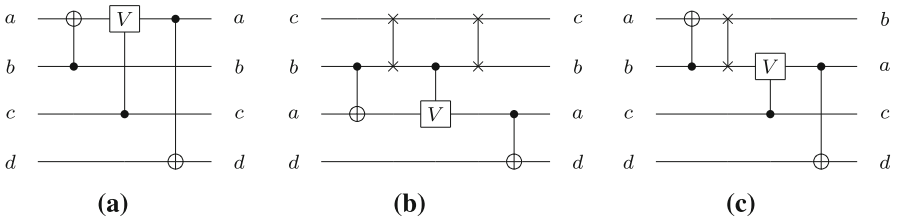


Fig. 10 Global and local reordering

Example 4 Consider the circuit depicted in Fig. 10a. After calculating the NNC contributions, we have $imp_a = 1.5$, $imp_b = 0$, $imp_c = 0.5$, and $imp_d = 1$, respectively. Thus, lines a (highest impact) and c (middle line) are swapped. Since further swapping does not improve the NNC value, reordering terminates and SWAP gates are added for the remaining non-adjacent gates. The resulting circuit is depicted in Fig. 10b and has quantum cost of 9 in comparison to 21 that results if the naive method is applied.

4.3.2 Local reordering

In order to save SWAP gates, line ordering can also be applied according to a local schema as follows. The circuit is traversed from the inputs to the outputs. As soon as there is a gate g with an NNC value greater than 0, a SWAP operation is added in front of g to enable an adjacent gate. However, in contrast to the naive NNC-based decomposition, no SWAP operation is added after g. Instead, the resulting ordering is used for the rest of the circuit (i.e., propagated through the remaining circuit). This process is repeated until all gates are traversed.

Example 5 Reconsider the circuit depicted in Fig. 10a. The first gate is not modified since it has an NNC of 0. For the second gate, a SWAP operation is applied to make it adjacent. Afterwards, the new line ordering is propagated to all remaining gates resulting in the circuit shown in Fig. 10c. This procedure is repeated until the whole circuit has been traversed. Finally, a circuit with quantum cost of 9 (in contrast to 21) is produced.

5 A synthesis flow for LNN architectures

Having the proposed approaches from the previous section available, they can be combined to an extended synthesis flow that explicitly takes the LNN limitation into account. Figure 11 illustrates this flow. As shown in this figure, first an off-the-shelf synthesis approach is applied to create an initial circuit realization. Afterwards, if macro replacement is enabled, the proposed macro replacement method from Sect. 4.2 is applied to simplify the circuit (lines 2–3). Then, one of the available standard decomposition methods is applied to decompose all non-elementary gates into a set of elementary unit-cost gates (line 4). The resulting quantum circuit can be optimized by the reordering methods proposed in Sect. 4.3 (lines 5–8). Finally, SWAP gates for

```
      input: a given reversible or quantum specification
      output: a synthesized circuit for LNN architectures

 1.   synthesize the given specification using an appropriate synthesis method
 2.   if macro replacement is enabled
 3.      apply the available macros
 4.   decompose each gate into a set of elementary gates
 5.   if global reordering is enabled
 6.      reorder initial qubit locations based on the global reordering method
 7.   if local reordering is enabled
 8.      reorder initial qubit locations based on the local reordering method
 9.   insert a set of SWAP gates for each non-adjacent gate
10.   if template matching is enabled
11.      apply the available templates
```

Fig. 11 The extended synthesis flow

the remaining non-adjacent gates have to be added (line 9) and template matching as introduced in Sect. 4.1 can additionally be applied (lines 10–11). Note that each method is applied on the result of the previous method in the proposed synthesis flow. It can be verified that for the naive method, only lines 1, 4, and 9 are executed.

6 Experimental results

In this section, experimental results are presented. We evaluated the methods introduced in Sect. 4 and compared them to the naive approach, which has been used by other synthesis methods [16,17,19–21,34] so far. All approaches have been implemented in C++ and applied to the benchmark collection available at RevLib [39] including a wide variety of circuits that already have been used by other researchers to evaluate previous reversible synthesis approaches. The experiments have been carried out on an Intel Pentium IV 2.2 GHz computer with 2GB memory.

The results are shown in Tables 2 and 3, respectively. The former table shows the results obtained by applying the established decomposition, where a SWAP gate is composed of three elementary gates. Additionally, Table 3 shows the results obtained by assuming the SWAP gate itself to be an elementary gate (as done by certain quantum technologies [2]). The first column gives thereby the names of the circuits followed by unique identifiers as used in RevLib. Then, the number of circuit lines (n), the gate count (gc), the quantum cost (qc), and the NNC value of the original reversible circuits are shown. The following columns denote the quantum cost of the NNC-optimal circuits obtained by the naive method (N) as well as by the proposed synthesis flow where a combination of macro replacement (M), global reordering (G), local reordering (L), and template matching (T) methods has been applied. For example, *MT* denotes the results obtained by the proposed flow with macro replacement and template matching methods enabled (i.e., reordering methods disabled). Besides that, the results of the best configurations are given in column *Best config*. The percentages of the best quantum cost reduction obtained by the extended synthesis flow in comparison to the widely used naive method are reported in Column *Best Impr.*. Column *Time* denotes the overall run-time needed to generate the results for *all* possible configurations (with and without any possible options). Finally, the last column shows the remaining

Table 2 Experimental results (considering a SWAP gate to be composed out of 3 elementary gates)

Circuit	Original circuit				Decomposed circuit											Time (s)	Ohead
						N	T	M	MT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
0410184_169	14	46	90	68	14	234	212	197	189	234	423	423	189	MT	19	0	2.10
3_17_13	3	6	14	8	3	32	32	28	26	32	32	32	26	MT	18	0	1.86
4_49_17	4	12	32	64	4	158	104	120	102	128	98	98	92	GT	41	0	2.88
4gt10-v1_81	5	6	34	164	5	282	120	282	120	258	150	147	120	T	57	0	3.53
4gt11_84	5	3	7	26	5	49	31	47	29	25	22	16	14	MGL	71	0	2.00
4gt12-v1_89	5	5	42	320	6	525	195	525	195	321	171	168	141	GT	73	0	3.36
4gt13-v1_93	5	4	16	104	5	173	83	173	83	77	56	53	47	GT	72	0	2.94
4gt4-v0_80	5	5	34	218	6	366	168	364	166	168	138	141	132	GT	63	0	3.88
4gt5_75	5	5	21	76	5	142	94	138	96	118	82	79	70	GT	50	0	3.33
4mod5-v1_23	5	8	24	90	5	174	84	155	101	114	78	78	72	GLT	58	0	3.00
4mod7-v0_95	5	6	38	144	5	256	140	256	140	352	127	121	121	GL	52	0	3.18
add16_174	49	64	192	220	49	762	446	473	473	762	1104	1104	428	MGL	43	0	2.23
add32_183	97	128	384	444	97	1530	894	953	953	1530	3744	3744	860	MGL	43	2	2.24
add64_184	193	256	768	892	193	3066	1790	1913	1913	3066	13632	13632	1724	MGL	43	14	2.24
add8_172	25	32	96	108	25	378	222	233	233	378	360	360	212	MGL	43	0	2.21
aj-e11_165	4	13	45	144	5	280	166	260	164	280	181	181	160	GT	42	0	3.56
alu-v4_36	5	7	31	136	5	242	146	238	148	218	113	104	98	GT	59	0	3.16
cnt3-5_180	16	20	120	1634	16	2621	613	2591	677	1457	731	728	511	GT	80	0	4.26
cycle10_2_110	12	19	1126	13472	12	21420	13700	21420	13700	21420	8046	8046	7874	LT	63	4	6.99
decod24-v3_46	4	9	9	36	4	63	27	63	27	39	21	24	21	L	66	0	2.33
ham15_108	15	70	453	9978	15	15494	10610	15390	10582	14030	2627	2588	2588	GL	83	2	5.71
ham7_104	7	23	83	624	7	1035	681	1027	695	657	342	333	327	GT	68	0	3.94

Table 2 continued

Circuit	Original circuit				Decomposed circuit										Time (s)	Ohead
					N	T	M	MT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
hwb4_52	4	11	23	40	4	107	77	83	63	107	65	65	63	MT	41	0	2.74
hwb5_55	5	24	104	470	5	823	407	817	415	595	337	340	335	LT	59	0	3.22
hwb6_58	6	42	142	710	6	1304	692	1160	672	1268	614	545	542	GLT	58	0	3.82
hwb7_62	7	331	2325	16890	8	27967	15547	27869	15533	25939	13390	12955	12853	LT	54	4	5.53
hwb8_118	8	633	14260	115030	9	187272	96906	186880	96834	182196	87495	87498	87495	L	53	39	6.14
hwb9_123	9	1959	18124	189426	10	304659	168147	304540	168160	302481	124068	124041	124041	GL	59	74	6.84
mod5adder_128	6	15	83	600	6	1011	435	978	432	675	330	333	330	L	67	0	3.98
mod8-10_177	5	14	88	582	6	975	407	969	409	621	372	363	317	GT	67	0	3.60
plus127mod8192_162	13	910	57400	661596	14	1057946	675624	1057804	675610	1057946	503516	503516	496698	LT	53	376	8.65
plus63mod4096_163	12	429	25492	254864	13	407926	256792	407784	256778	407926	210400	210400	210100	LT	48	113	8.24
plus63mod8192_164	13	492	32578	397864	14	633994	409384	633852	409358	633994	279016	279016	271030	LT	57	187	8.32
rd32-v0_67	4	2	10	10	4	38	26	19	17	20	32	20	17	MT	55	0	1.70
rd53_135	7	16	77	466	7	822	450	750	456	702	330	303	303	GL	63	0	3.94
rd73_140	10	20	76	450	10	790	400	739	401	646	304	295	286	LT	63	0	3.76
rd84_142	15	28	112	910	15	1516	626	1465	639	1696	556	586	556	L	63	0	4.96
sym9_148	10	210	4368	48736	10	77556	46110	77556	46110	67428	20643	25023	20640	LT	73	11	4.73
sys6-v0_144	10	15	67	358	10	638	326	587	329	842	263	308	263	L	58	0	3.93
urf1_149	9	11554	57770	462708	9	794582	353732	735170	329762	659150	238475	238490	238475	L	69	261	4.13
urf2_152	8	5030	25150	171284	8	297178	133606	276882	126348	297178	101683	101683	101656	LT	65	56	4.04

Table 2 continued

Circuit	Original circuit				Decomposed circuit											Time (s)	Ohead
						N	T	M	MT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
urf3_155	10	26468	132340	1282724	10	2121808	897358	2038584	874848	1933372	596368	596371	596356	LT	71	1298	4.51
urf5_158	9	10276	51380	442748	9	740084	333674	706412	321496	667484	208709	208706	208700	LT	71	231	4.06
urf6_160	15	10740	53700	951276	15	1487904	589662	1478080	586572	1334916	320412	320409	320400	LT	78	596	5.97

N, Naive method (i.e., synthesis, decomposition, SWAP insertion); T, with template matching; M, with macros replacement; G, with global reordering; L, with local reordering
Column *Time* denotes the overall run-time needed to generate the results for *all* possible configurations
Almost all results have been obtained in negligible run-time (i.e., in less than one CPU second). Only if template matching was enabled more run-time was needed

Table 3 Experimental results (considering a SWAP gate as an elementary gate)

Circuit	Original circuit				Decomposed circuit											Time (s)	Overhead
						N	NT	NM	NMT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
0410184_169	14	46	90	68	14	138	124	157	149	138	201	201	124	NT	10	0.10	1.38
3_17_13	3	6	14	8	3	20	20	24	22	20	20	20	20	N	0	0.01	1.43
4_49_17	4	12	32	64	4	74	56	76	70	64	54	54	52	GT	29	0.01	1.62
4gt10-v1_81	5	6	34	164	5	118	64	118	64	110	74	73	64	NT	45	0.02	1.88
4gt11_84	5	3	7	26	5	21	15	23	17	13	12	10	10	GL	52	0.00	1.43
4gt12-v1_89	5	5	42	320	6	205	95	205	95	137	87	86	77	GT	62	0.03	1.83
4gt13-v1_93	5	4	16	104	5	69	39	69	39	37	30	29	27	GT	60	0.00	1.69
4gt4-v0_80	5	5	34	218	6	146	80	148	82	80	70	71	68	GT	53	0.02	2.00
4gt5_75	5	5	21	76	5	62	46	66	52	54	42	41	38	GT	38	0.02	1.81
4mod5-v1_23	5	8	24	90	5	74	44	75	57	54	42	42	40	GLT	45	0.02	1.67
4mod7-v0_95	5	6	38	144	5	112	72	112	72	144	69	67	66	LT	41	0.02	1.74
add16_174	49	64	192	220	49	382	254	413	413	382	496	496	254	NT	33	0.54	1.32
add32_183	97	128	384	444	97	766	510	829	829	766	1504	1504	510	NT	33	2.32	1.33
add64_184	193	256	768	892	193	1534	1022	1661	1661	1534	5056	5056	1022	NT	33	14.07	1.33
add8_172	25	32	96	108	25	190	126	205	205	190	184	184	126	NT	33	0.11	1.31
aj-e11_165	4	13	45	144	5	124	86	128	96	124	91	91	84	GT	32	0.02	1.87
alu-v4_36	5	7	31	136	5	102	70	106	76	94	59	56	54	GT	47	0.03	1.74
cnt3-5_180	16	20	120	1634	16	957	281	987	349	569	327	326	247	GT	74	0.49	2.06
cycle10_2_110	12	19	1126	13472	12	7948	5372	7948	5372	7948	3490	3490	3430	LT	56	3.61	3.05
decod24-v3_46	4	9	9	36	4	27	15	27	15	19	13	14	13	L	51	0.00	1.44
ham15_108	15	70	453	9978	15	5470	3842	5458	3854	4982	1181	1168	1168	GL	78	2.22	2.58
ham7_104	7	23	83	624	7	403	285	411	299	277	172	169	167	GT	58	0.09	2.01

Table 3 continued

Circuit	Original circuit				Decomposed circuit											Time (s)	Overhead
						N	NT	NM	NMT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
hwb4_52	4	11	23	40	4	51	41	59	51	51	37	37	37	L	27	0.02	1.61
hwb5_55	5	24	104	470	5	347	207	353	219	271	185	186	183	LT	47	0.10	1.76
hwb6_58	6	42	142	710	6	532	328	512	348	520	302	279	278	GLT	47	0.16	1.96
hwb7_62	7	331	2325	16890	8	11023	6883	11033	6921	10347	6164	6019	5985	LT	45	3.90	2.57
hwb8_118	8	633	14260	115030	9	72060	41938	72032	42014	70368	38801	38802	38801	L	46	39.05	2.72
hwb9_123	9	1959	18124	189426	10	115167	69663	115180	69720	114441	54970	54961	54961	GL	52	73.77	3.03
mod5adder_128	6	15	83	600	6	395	203	394	212	283	168	169	168	L	57	0.08	2.02
mod8-10_177	5	14	88	582	6	387	195	393	205	269	186	183	165	GT	57	0.09	1.88
plus127mod8192_162	13	910	57400	661596	14	396286	268836	396272	268866	396286	211476	211476	209194	LT	47	269.17	3.64
plus63mod4096_163	12	429	25492	254864	13	152998	102612	152984	102642	152998	87156	87156	87048	LT	43	60.56	3.41
plus63mod8192_164	13	492	32578	397864	14	236066	161188	236052	161214	236066	117740	117740	115070	LT	51	132.34	3.53
rd32-v0_67	4	2	10	10	4	18	14	19	17	12	16	12	12	G	33	0.02	1.20
rd53_135	7	16	77	466	7	326	202	314	216	286	162	153	153	GL	53	0.11	1.99
rd73_140	10	20	76	450	10	314	176	315	197	266	152	149	138	LT	56	0.08	1.82
rd84_142	15	28	112	910	15	580	274	581	299	640	260	270	260	L	55	0.16	2.32
sym9_148	10	210	4368	48736	10	28820	18338	28820	18338	25444	9849	11309	9848	LT	65	8.88	2.25
sys6-v0_144	10	15	67	358	10	254	150	255	169	322	129	144	129	L	49	0.03	1.93
urf1_149	9	11554	57770	462708	9	303374	156424	300962	165226	258230	118005	118010	118005	L	61	193.37	2.04
urf2_152	8	5030	25150	171284	8	115826	61302	115666	65100	115826	50661	50661	50652	LT	56	35.66	2.01

Table 3 continued

Circuit	Original circuit				Decomposed circuit											Time (s)	Overhead
						N	NT	NM	NMT	G	L	GL	Best		Best		
	n	gc	qc	NNC	*n*	qc	qc	qc	qc	qc	qc	qc	qc	Method	Impr.%		
urf3_155	10	26468	132340	1282724	10	795496	387346	799448	410372	732684	287016	287017	287012	LT	63	831.05	2.17
urf5_158	9	10276	51380	442748	9	280948	145478	280052	151332	256748	103823	103822	103820	LT	63	99.52	2.02
urf6_160	15	10740	53700	951276	15	531768	232354	531664	234208	480772	142604	142603	142600	LT	73	302.61	2.66

N, Naive method (i.e., synthesis, decomposition, SWAP insertion); T, with template matching; M, with macros replacement; G, with global reordering; L, with local reordering
Column *Time* denotes the overall run-time needed to generate the results for *all* possible configurations
Almost all results have been obtained in negligible run-time (i.e., in less than one CPU second). Only if template matching was enabled more run-time was needed

overhead in terms of quantum cost needed to achieve NNC-optimality in comparison to the original circuit (*Ohead*).

As can be seen, decomposing reversible circuits to have NNC-optimal quantum circuits for LNN architectures is costly. Using the widely used naive method, the quantum cost increases significantly. This result has been obtained in recent synthesis papersas well [17,18,34]. However, using the proposed methods, this can be improved. Even if reordering may worsen the results in some few cases, in total this leads to an improvement. The results have been obtained in negligible run-time (i.e., in less than one CPU second). Only if template matching was enabled more run-time was needed.

Overall, reductions of more than 50% on average—in the best case of 83%—have been observed considering the established decomposition (see Table 2). Similar results are obtained applying the extended definition of elementary gates (see Table 3). As a result, NNC-optimal circuits can be synthesized with a moderate increase of quantum cost.

7 Conclusions

Quantum technologies are in preliminarily state and several limitations should be resolved to have a scalable quantum technology. Limited interaction distance between gate qubits is one of the most common limitations of the current technologies. In this paper, we illustrated how the synthesis flow can be modified to produce efficient circuits for quantum technologies with limited interactions. The proposed flow includes a set of NNC-based decomposition methods equipped by an NNC-based template matching algorithm. The experiments show that with a naive treatment of the LNN restriction, quantum circuits require up to one order of magnitude higher quantum cost in the LNN architectures. In contrast, using the suggested methods, this increase can be reduced by more than 50% on average (83% in the best case).

Acknowledgments Parts of this research work has been supported by the German Research Foundation (DFG) (DR 287/20-1).

References

1. Wille, R., Saeedi, M., Drechsler, R.: Synthesis of reversible functions beyond gate count and quantum cost. In: International Workshop on Logic Synthesis, pp. 43–49 (2009)
2. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
3. Mosca, M.: Quantum algorithms. Springer Encyclopedia of Complexity and Systems Science (to appear) (2008)
4. Meter, R.V., Oskin, M.: Architectural implications of quantum computing technologies. J. Emerg. Technol. Comput. Syst. **2**(1), 31–63 (2006)
5. Ross, M., Oskin, M.: Quantum computing. Commun. ACM **51**(7), 12–13 (2008)
6. Knill, E., Laflamme, R., Milburn, G.J.: A scheme for efficient quantum computation with linear optics. Nature **409**, 46–52 (2001)
7. Fowler, A.G., Devitt, S.J., Hollenberg, L.C.L.: Implementation of shor's algorithm on a linear nearest neighbour qubit array. Quantum Information and Computation **4**, 237–245 (2004)

8. Häffner, H., Hänsel, W., Roos, C.F., Benhelm, J., Chek al kar, D., Chwalla, M., Körber, T., Rapol, U.D., Riebe, M., Schmidt, P.O., Becher, C., Gühne, O., Dür, W., Blatt, R.: Scalable multiparticle entanglement of trapped ions. Nature **438**, 643–646 (2005)
9. Laforest, M., Simon, D., Boileau, J.-C., Baugh, J., Ditty, M., Laflamme, R.: Using error correction to determine the noise model. Phys. Rev. A **75**, 133–137 (2007)
10. Kane, B.: A silicon-based nuclear spin quantum computer. Nature **393**, 133–137 (1998)
11. Maslov, D.: Linear depth stabilizer and quantum fourier transformation circuits with no auxiliary qubits in finite neighbor quantum architectures. Phys. Rev. A **76** (2007)
12. Takahashi, Y., Kunihiro, N., Ohta, K.: The quantum fourier transform on a linear nearest neighbor architecture. Quantum Information and Computation **7**, 383–391 (2007)
13. Kutin, S.A.: Shor's algorithm on a nearest-neighbor machine. Asian Conference on Quantum Information Science (2007)
14. Choi, B.-S., Van Meter, R.: Effects of Interaction Distance on Quantum Addition Circuits. ArXiv e-prints (September 2008)
15. Fowler, A.G., Hill, C.D., Hollenberg, L.C.L.: Quantum error correction on linear nearest neighbor qubit arrays. Phys. Rev. A **69**, 042314.1–042314.4 (2004)
16. Möttönen, M., Vartiainen, J.J.: Decompositions of General Quantum Gates. Chapter 7 in Trends in Quantum Computing Research. NOVA Publishers, New York (2006)
17. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum-logic circuits. IEEE Trans. on CAD **25**(6), 1000–1010 (2006)
18. Cheung, D., Maslov, D., Severini, S.: Translation techniques between quantum circuit architectures. Workshop on Quantum Information Processing (December 2007)
19. Chakrabarti, A., Sur-Kolay, S.: Nearest neighbour based synthesis of quantum boolean circuits. Eng. Lett. **15**, 356–361 (2007)
20. Khan, M.H.A.: Cost reduction in nearest neighbour based synthesis of quantum boolean circuits. Eng. Lett. **16**, 1–5 (2008)
21. Hirata, Y., Nakanishi, M., Yamashita, S., Nakashima, Y.: An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture. In: International Conference on Quantum, Nano and Micro Technologies, pp. 26–33 (2009)
22. Lee, S., Lee, S.J., Kim, T., Lee, J.S., Biamonte, J., Perkowski, M.: The cost of quantum gate primitives. J. Multiple Value Logic Soft Comput. **12**(5–6) (2006)
23. Barenco, A., Bennett, C., Cleve, R., DiVincenzo, D., Margolus, N., Shor, P., Sleator, T., Smolin, J., Weinfurter, H.: Elementary gates for quantum computation. APS Phys. Rev. A **52**, 3457–3467 (1995)
24. Maslov, D., Dueck, G.W., Miller, D.M., Negrevergne, C.: Quantum circuit simplification and level compaction. IEEE Trans. on CAD **27**(3), 436–444 (2008)
25. Hung, W.N.N., Song, X., Yang, G., Yang, J., Perkowski, M.: Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans. on CAD **25**(9), 1652–1663 (2006)
26. Große, D., Wille, R., Dueck, G.W., Drechsler, R.: Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. International Symposium on Multiple Valued Logic, pp. 214–219 (2008)
27. Maslov, D., Dueck, G.W., Michael Miller, D.: Toffoli network synthesis with templates. IEEE Trans. on CAD **24**(6), 807–817 (2005)
28. Maslov, D., Dueck, G.W., Miller, D.M.: Techniques for the synthesis of reversible toffoli networks. ACM Trans. Des. Autom. Electron. Syst. **12**(4), 42 (2007)
29. Saeedi, M., Sedighi, M., Saheb Zamani, M.: A novel synthesis algorithm for reversible circuits. IEEE/ACM International Conference on Computer-aided design, pp. 65–68 (2007)
30. Gupta, P., Agrawal, A., Jha, N.K.: An algorithm for synthesis of reversible logic circuits. IEEE Trans. on CAD **25**(11), 2317–2330 (2006)
31. Große, D., Wille, R., Dueck, G.W., Drechsler, R.: Exact multiple control toffoli network synthesis with SAT techniques. IEEE Trans. on CAD **28**(5), 703–715 (2009)
32. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: DAC '09: Proceedings of the 46th annual Design Automation Conference, pp. 270–275 (2009)
33. Saeedi, M., Sedighi, M., Zamani, M. Saheb: A library-based synthesis methodology for reversible logic. Elsevier Microelectron. J. **41**(4), 185–194 (2010)
34. Saeedi, M., Zamani M., Saheb, Sedighi, M., Sasanian, Z.: Synthesis of reversible circuit using cycle-based approach. ACM J. Emerg. Technol. Comput. Syst. http://arxiv.org/abs/1004.4320 (2010)

35. Miller, D. Michael, Maslov, Dmitri, Dueck, Gerhard W.: A transformation based algorithm for reversible logic synthesis. In: DAC '03: Proceedings of the 40th annual Design Automation Conference, pp. 318–323. New York, NY: ACM (2003)
36. Chakrabarti, A., Sur-Kolay, S.: Rules for synthesizing quantum boolean circuits using minimized nearest-neighbour templates. In: International Conference on Advanced Computing and Communications, pp. 183–189 (2007)
37. Barenco, A., Ekert, A., Suominen, K.-A., Törmä, P.: Approximate quantum fourier transform and decoherence. Phys. Rev. A **54**(1), 139–146 (1996)
38. Wille, R., Große, D., Dueck, G.W., Drechsler, R.: Reversible logic synthesis with output permutation. In: VLSID '09: Proceedings of the 2009 22nd International Conference on VLSI Design, pp. 189–194, Washington, DC, USA: IEEE Computer Society (2009)
39. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: Revlib: An online resource for reversible functions and reversible circuits. International Symposium on Multiple Valued Logic, pp. 220–225 (May 2008)