Integrated Synthesis of Linear Nearest Neighbor Ancilla-Free MCT Circuits

Md Mazder Rahman[1], Gerhard W. Dueck[1], Anupam Chattopadhyay[2] and Robert Wille[3]
[1] Faculty of Computer Science, University of New Brunswick, Canada
[2] School of Computer Engineering, Nanyang Technological University, Singapore
[3] Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
Email:{Md._Mazder.Rahman, gdueck}@unb.ca anupam@ntu.edu.sg robert.wille@jku.at

***Abstract*—The rapid advances of quantum technologies are opening up new challenges, of which, protecting quantum states from errors is a major one. Among quantum error correction schemes, the surface code is emerging as a natural choice with high-fidelity quantum gates reported for experimental platforms. Surface codes also necessitate the quantum gates to be formed with strict nearest neighbour coupling. State-of-the-art reversible logic synthesis techniques for quantum circuit implementation do not ensure the logic gates to be formed in a nearest neighbor fashion, and this is handled as a post-processing optimization by the insertion of swap gates. In this paper, we propose, for the first time, the inclusion of nearest neighbourhood criteria in a widely used ancilla-free reversible logic synthesis method. Experimental results show that this method easily outperforms the earlier two-step techniques in terms of gate count without any runtime overhead.**

I. Introduction

Quantum error correction [1] is considered to be the key challenge towards building a large-scale quantum computer. Among various choices of codes, surface codes [2] are shown to be most favourable in terms of fault tolerance. Implementations of logical qubits with surface codes have been reported in different technologies [3], including silicon [4]. A fundamental requirement of the surface code realization is that the quantum gates must be formed with a nearest neighbour coupling. The resulting circuits are also termed Linear Nearest Neighbor (LNN) circuits. Though any quantum circuit can be converted to an LNN circuit by introducing additional swap gates, the size of the resulting circuit can pose a challenge towards practical implementation. As a result, efficient LNN circuit construction has been studied for important quantum benchmarks, such as, quantum error correction [5] and factorization [6].

Realization of reversible functions traditionally starts with Toffoli gates [7], [8]. Hence the process of realizing LNN quantum circuit goes through a number of steps such synthesis [9], transformations [10], [11], [12], optimizations [13], [14], decompositions [15] and so on. However, it is often not clear what the impact of a particular step has on the final result. For example, fewer Toffoli gates during an early stage may result in higher LNN cost in the final circuit. Naturally, for large benchmarks, the design approach for LNN circuit construction is not feasible and efficient design automation flows are needed.

The aim of this work is to investigate how a synthesis heuristic—transformation based-synthesis (commonly referred to as MMD) [9]—can be modified towards realizing improved LNN MCT (Multiple-Controlled Toffoli) circuits[1]. By inspection, we find that the initial choice of gates in MMD can be improved while targeting LNN MCT circuits. To do this, we develop a heuristic to synthesize LNN MCT circuits for given function specifications. The new synthesis approach outperforms over the two-step process that includes the basic MMD and LNN MCT transformation. For the best cases, experiments show that the algorithm results in 84% and 76% reduction of the gate count of circuits that are obtained from LNN transformation by using SWAP gates and without using SWAP gates.

The remainder of the paper is structured as follows: Section II briefly describes the fundamentals of reversible logic circuits. In Section III we overview an synthesis method—transformation-based algorithm—with some observations towards developing algorithm for LNN MCT circuits. Section IV illustrates a heuristic for synthesizing LNN MCT circuits for given function specifications. The significance of the proposed approach is shown with experiments in Section V. The paper concludes with some observations and directions for future research in Section VI.

[1]Note that, traditionally, optimization for LNN architectures have been conducted on the quantum circuit level only. However, recent trends such as reported in [16], [17] also motivate a consideration at the reversible circuit level.

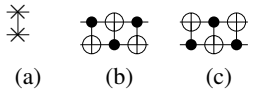


Fig. 1: SWAP gate representations.

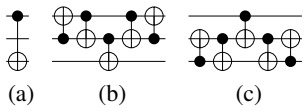


Fig. 2: LNN transformation of CNOT

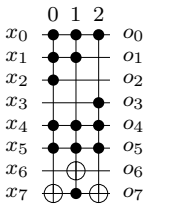


Fig. 3: An MCT circuit.

II. Background

A Boolean logic function $f : \mathbb{B}^n \to \mathbb{B}^n$ over a set of variables $X = \{x_1, \ldots, x_n\}$ is reversible if it is bijective. Classical reversible functions [18] are the special case of multiple-output Boolean functions. A MCT gate $g(C, t)$ is a reversible gate consisting of a set of control lines $C \subset \{x \mid x \in X\}$ that are literals of X and a target line $t \in X$ such that $\{t\} \cap C = \emptyset$. The value at the target line is inverted if all controls evaluate to true. All remaining values are passed through the gate unchanged. Gates for which $|C| = 0$ and $|C| = 1$ are known as NOT and CNOT, respectively. A SWAP gate consists of a sequence of 3 CNOT gates acting on the same 2 lines; and target and control of each subsequent gate act on alternating lines as shown in Figure 1.

Reversible circuits are realized by cascading reversible gates. An MCT circuit is realized by cascading MCT gates. The size of a circuit G, denoted by $|G|$, is the number of gates in G. The size of an MCT circuit is also known as the gate count of the circuit. Quantum cost — a performance measurement metric of a circuit — is defined as the number of quantum gates required to implement a reversible circuit. For a given function, a circuit of size n is minimal if the function can not be realized by other circuit with fewer gates.

A circuit is an LNN circuit if all controls and target of each gate in the circuit are acting on adjacent lines. A line that is between a control and the target or between two control lines is called an "internal line". In the following we briefly review LNN transformation [12].

A. LNN Transformation

A straightforward method of LNN Transformation is to make all controls and the target of each gate in a circuit adjacent by moving control(s)/target towards target/control(s) of a gate. Usually, moving control(s)/target towards target/control(s) of a gate is done by inserting SWAP gates. However, an optimizing approach shown in [12] can be used to reduce the size of a circuit.

The CNOT gate with one internal line as shown in Figure 2(a) can be transformed into LNN circuits as shown in Figure 2(b) (control moves towards target) and 2(c) (target moves towards control). In this approach, one move of a single control/target of an MCT gate requires 4 CNOT gates. However, the internal lines of an MCT gate with large number of controls in a standard MCT circuit (as shown in Figure 3) may not be adjacent. Therefore, it is necessary to optimize the direction of moves to minimize the number of CNOT gates.

III. Review of Synthesis Method

In this section we first discuss the transformation-based algorithm [9] with examples and then present how this method can be modified by optimizing the choice of gates towards synthesizing LNN MCT circuits.

Example 1: Consider the benchmark function 3_17 [19] shown in the first two columns in Table I, MMD (bidirectional = false) [9] results in the transformation of the output permutation in Table I and the circuit in Figure 4. It can be seen that in transforming $\langle 111 \rangle$ to $\langle 001 \rangle$ (from step i to iii in Table I), MMD uses gates at positions 7 and 6 in the circuit in Figure 4. However, if we consider that gates with low quantum costs should be chosen in each transformation, then we can construct the transformation shown in Table II with the corresponding circuit shown in Figure 5 in which the gates at position 7 and 6 are used in transforming $\langle 111 \rangle$ to $\langle 001 \rangle$. It is clear that the quantum cost of the gate at position 7 in the circuit in Figure 5 is lower than that of the gate at position 7 in the circuit in Figure 4.

For the function in Table III, MMD results in the circuit shown in Figure 6(a) which is not an LNN circuit.

TABLE I: Transformation from output side to input side.

Input	Output	Transformation								
		i	ii	iii	iv	v	vi	vii	$viii$	ix
cba	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx
000	111	000	000	000	000	000	000	000	000	000
001	000	111	011	001	001	001	001	001	001	001
010	001	110	110	110	010	010	010	010	010	010
011	011	100	100	100	100	110	111	011	011	011
100	100	011	111	101	101	111	110	110	100	100
101	010	101	101	111	011	011	011	111	101	101
110	110	001	001	011	111	101	101	101	111	110
111	101	010	010	010	110	100	100	100	110	111

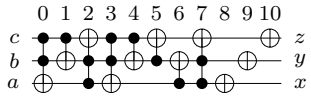


Fig. 4: MCT circuit for the benchmark 3_17 obtained from MMD with $bidirectional = false$.

TABLE II: Transformation from output side to input for the circuit in Figure 5.

Input	Output	Transformation								
		i	ii	iii	iv	v	vi	vii	$viii$	ix
cba	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx	zyx
000	111	000	000	000	000	000	000	000	000	000
001	000	111	011	001	001	001	001	001	001	001
010	001	110	010	010	010	010	010	010	010	010
011	011	100	100	100	110	111	011	011	011	011
100	100	011	111	101	111	110	110	100	100	100
101	010	101	101	111	101	101	101	111	101	101
110	110	001	001	011	011	011	111	101	111	110
111	101	010	110	110	100	100	100	110	110	111

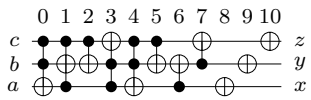


Fig. 5: MCT circuit for the benchmark 3_17.

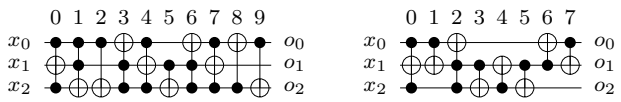
If LNN transformation [12] is done on individual gates at 2, 8, and 9 in the circuit in Figure 6(a), then the resulting LNN MCT circuit requires 12 (3×4) more CNOT gates. However, the function in Table III can be realized with a smaller LNN MCT circuit as shown in Figure 6(b).

IV. Heuristic for LNN MCT Circuits

In this section we present the basic idea for LNN MCT synthesis followed by the algorithm shown in Figure 7. For a given reversible function f in the form of a truth table, the algorithm proceeds exactly the same way as the basic MMD: For each row i $(0 \leq i < 2^n - 1)$ in truth table, if $f(i) \neq i$ then the basic MMD finds MCT

TABLE III: Truth table of a 3×3 function.

$Input$	$Output$
$x_0x_1x_2$	$o_0o_1o_2$
000	000
001	100
010	001
011	011
100	111
101	010
110	110
111	101



(a) (b)

Fig. 6: (a) MCT circuit for the function in Table III obtained from MMD and (b) MCT circuit for the function in Table III

gates to map $f(i) = i$ such that all previous rows are not affected. However, to choose lower-order MCT gates in each step, we partition the output vector as follows: Consider the input and output vectors for the i^{th} row in a truth table as shown in Figure 8, and lets say all previous rows from 0 to $i - 1$ have already been mapped. Notice that $f(i) \neq i$. **Clear Bits** in the output vector must be set to 0. The position of the **Pivot Bit** is $log_2(i)$ for which the most significant bit of input vector is 1. If the **Pivot Bit** in the output vector is not set to 1, it must be set to 1. The intuition of partitioning the output vector is that if necessary, only the right most bit in **Clear Bits** will be used as an additional control of an MCT gate in changing the pivot and **Transform Bits**. Except the right most bit in **Clear Bits**, all bits in **Clear Bits** can be set to 0 by using only CNOT gates in the form of an LNN structure.

To illustrate the algorithm, we consider the i^{th} row for the input and output vectors of f as shown in Figure 8. Given that $f(i-1) = i-1$ is done, and $i - 1^{th}$ output vector v_{i-1} is 0000010001. The algorithms are written using C++ that can be easily followed. The main algorithm shown in Figure 7 employs three major functions that are described below.

The function **clearAllBeforeBeforePivotBit**($\&G$, p, $\&v$) in Figure 9 clears all other bits in **Clear Bits** and sets 1 to the right most bit in **Clear Bits**. This results in a subcircuit $G_1 = S_1$ as indicated in Figure 12 and vector $v_i = [0001010101]$. To set **Pivot Bit** position to 1, a CNOT

```
(1)  Circuit LNNSynthesis(f)
(2)     // f is a given function specification
(3)     // G be the LNN MCT circuit of f
(4)     // G_1 be a temporary LNN circuit
(5)     // v_f be a Boolean vector containing f_{in[i]} ⊕ f_{out[i]}
(6)     // p be an integer—the position of pivot bit in a vector
(7)     for each vector v_i ∈ f
(8)        if computeBitFlip(i, f, v_f)
(9)           if !i
(10)             for k = 0 to |v_i| − 1
(11)                if v_i[k]
(12)                   G_1.appendNOT(k);
(13)             apply(G_1, &f);
(14)             G.appendCircuit(G_1);
(15)          else
(16)             p=findPivotBit(i, v_i);
(17)             if p > 0
(18)                clearAllBeforeBeforePivotBit(G_1, p, v_i);
(19)             // set PivotBit
(20)             if !v_i[p]
(21)                G_1.appendCNOT(p − 1, p);
(22)                v_i[p] = (!v_i[p]);
(23)             doTransformation(G_1, p, v_i, v_{i−1}, v_f);
(24)             if p > 0
(25)                clearLeftPivotBit(G_1, p, v_i, v_{i−1});
(26)             apply(G_1, &f);
(27)             G.appendCircuit(G_1);
(28)             G_1.clearCircuit();
(29)     reverseCircuit(&G);
(30)     return G;
```

Fig. 7: Algorithm for synthesis of LNN MCT circuits.

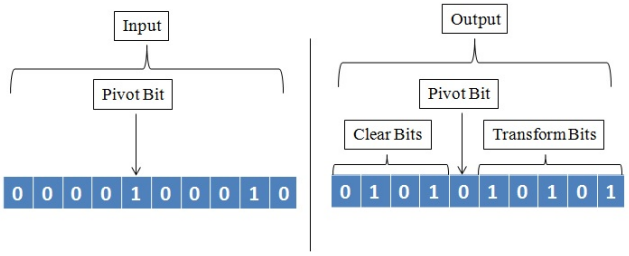


Fig. 8: Partition of an output vector.

```
(1)  clearAllBeforeBeforePivotBit(&G, p, &v)
(2)     /*
(3)     * if there any from v_i[0] to v_i[p − 2] is true,
(4)     * then v_i[p − 1] will be true and bits from v_i[0]
(5)     * to v_i[p − 2] will be false
(6)     */
(7)     // G is an MCT circuit
(8)     // p is an integer
(9)     // v is a vector
(10)    for i = 0 to p − 2
(11)       for j = i + 1 to p − 1
(12)          if v[i] && v[j]
(13)             G.appendCNOT(j, i);
(14)             v[i] = (!v[i]);
(15)          else if v[i] && !v[j]
(16)             G.appendCNOT(i, j);
(17)             v[j] = (!v[j]);
(18)             G.appendCNOT(j, i);
(19)             v[i] = (!v[i]);
(20)          break;
```

Fig. 9: Algorithm to clear all bits before before PivotBit.

```
(1)  doTransformation(&G, p, &v_c, &v_p, v_f)
(2)     // G is an MCT circuit
(3)     // p is an integer
(4)     // v_c is a vector
(5)     // v_p is a vector
(6)     // c be a vector of integer
(7)     // flag = false
(8)     /* determine whether a clear is to be used as a control */
(9)     if p − 1 ≥ 0 && v_c[p − 1] && v_c[p] && v_p[p]
(10)       flag = true;
(11)    for j = v_c.size() − 1 to p + 1
(12)       if v_f[j]
(13)          if (flag)
(14)             c.push_back(p − 1)
(15)          for k = p to v_c.size() − 1
(16)             if k! = j && v_c[k]
(17)                c.push_back(k);
(18)                if v_c[p] && !v_p[p]
(19)                   break;
(20)          G.appendToffoli(c, j);
(21)          v_c[j] =!v_c[j]
(22)          G.doLNNGateAt(G.size() − 1);
(23)          c.clear();
```

Fig. 10: Algorithm for transformation.

gate shown as a sub-circuit S_2 in Figure 12 is appended in G_1 which results in $G_1 = S_1S_2$ and vector $v_i = [0001110101]$. The function **doTransformation**(&G, p, &v_c, &v_p, v_f) in Figure 10 first determines whether a bit from **Clear Bits** will be used as a control of an MCT gate. For each bit in **Transform Bits** to be flipped, an MCT gate is chosen. A bit to be flipped in output vector is searched from right to left. If all controls and target of a resulting gate are not adjacent then the function **doLNNGateAt**() transforms it in LNN structure by moving either a control or the target of the gate. The rules for moving a control or a target of an MCT gate without inserting SWAP gates have been discussed in Section II. The function **doTransformation**() results in the sub-circuit $G_1 = S_1S_2S_3$ (the sub-circuit S_2 is shown Figure 12) and vector $v_i = [0001100010]$. **clearLeftPivotBit**(&G, p, &v_{curr}, &v_{prev}) in Figure 11 sets 0 to the right most bit in **Clear Bits** and results in the sub-circuit $G_1 = S_1S_2S_3S_4$ shown in Figure 12 and vector $v_i = [0000100010]$.

In each row i in truth table, **apply**(G_1, &f) updates the output vectors of the function f by applying inputs in resulting sub-circuit G_1. For each $f(i) \neq i$, the resulting sub-circuit G_1 is appended into a circuit G. The obtained circuit G realizes f^{-1}, hence, the reverse circuit of G realizes f.

V. Experimental Results

We implemented synthesis heuristic using C/C++. For all 3 variable Boolean functions, we obtain LNN

```
(1)  clearLeftPivotBit(&G, p, &v_curr, &v_prev )
(2)     // G is an MCT circuit
(3)     // p is an integer
(4)     // v_curr is a vector
(5)     // v_prev is a vector
(6)     // c be a vector of integer;
(7)     if (p − 1) ≥ 0 && v_curr[p − 1]
(8)        for i = p to v_curr.size() − 1
(9)           if v_curr[i]
(10)             c.push_back(i)
(11)          if !v_prev[i]
(12)             break;
(13)       G.appendToffoli(c, p − 1);
(14)       G.doLNNGateAt(G.size() − 1);
(15)       v_curr[p − 1] = (!v_curr[p − 1]);
```

Fig. 11: Algorithm to clear left bit of PivotBit.

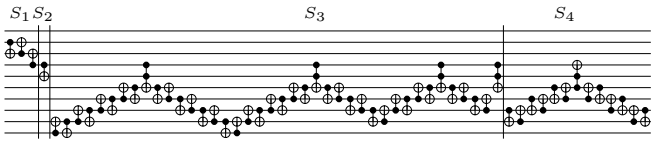


Fig. 12: A reversible circuit G_1.

MCT circuits and results are shown in Table IV. We have taken completely specified functions with different number of input lines from RevLib [19]. The first two columns in Table V show the name and the number of circuit lines. Column MMD in Table V shows the gate count of circuits obtained from MMD. Results of LNN transformations by inserting SWAP gates and using control/target moving rules (discussed in Section II-A) are shown in columns 4 and 5 in Table V respectively. Column 6 shows the results obtained from the proposed LNN synthesis heuristic. It can be observed that the gate count of circuits is high if LNN transformations are done with SWAP gates. The results of LNN transformations are compared with the new results. $Comparison_1$ and $Comparison_2$ represent the number of gate reductions and the percentage of reductions are achieved in new synthesis algorithm with respect to the LNN transformations using SWAP gates in column SWAP and without using SWAP gates in column W/O SWAP. In both the cases, a significant reduction (only few increases) can be observed.

This new synthesis approach has the property, that until the transformation is half-way done (MSB is set to 1), the number of controls of a gate that is chosen to clear a bit in **Clear Bits** is less than or equal to the number of controls of a gate chosen in basic MMD. Therefore, this approach may be beneficial in decomposing [15] the resulting circuits into quantum circuits. It can be noted that few results are negative which means that the gate count of the circuits obtained from the proposed heuristic is higher than that from other approaches. We investigate the small circuit $miller\_5$ and find that choosing gates with lower controls at the beginning has the consequence that more changes occur in the remaining rows of the function to be transformed and more gates are required in the transformation. It is well known that there is no asymptotically optimal algorithm to synthesize reversible functions. The proposed algorithms can further be improved by investigating the complexity of the functions. However, this new heuristic results in circuits with low gate counts for at least 50% of 33 benchmarks and in some cases the results are significantly better.

VI. Conclusion

We presented a new approach for synthesizing LNN circuits for given reversible functions. The experimental results show that this approach is promising and outperforms many benchmarks over the two-step LNN transformation method. Due to the ancilla-free synthesis with low gate count, transformation-based reversible logic synthesis is widely used as the prime benchmark among reversible logic synthesis tools. However, in this synthesis method, for the resulting circuits, the nearest neighbor property is not guaranteed, which is achieved via a post-processing phase. On the other hand, for practical realization of quantum circuits, gates are required to be formed in a nearest neighbor pattern to ensure fault tolerance. To enable that, we proposed an integrated synthesis flow that includes nearest neighborhood as a constraint within the transformation-based synthesis flow. Optimization of obtained circuits with LNN templates and a cost-benefit analysis in mapping circuits into LNN quantum circuits will be further research.

References

[1] A. W. Cross, D. P. Divincenzo, and B. M. Terhal, "A comparative code study for quantum fault tolerance," *Quantum Info. Comput.*, vol. 9, no. 7, pp. 541–572, Jul. 2009.

[2] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, Sep. 2012.

[3] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. OMalley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland and John M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, 2014.

TABLE IV: Results of LNN circuits for all 3 variable Boolean functions.

Size	#Function	#Function (New)
0	1	1
1	10	7
2	51	29
3	177	82
4	467	181
5	990	334
6	1742	374
7	2601	334
8	3358	337
9	3833	753
10	3996	1652
11	3976	2654
12	3925	2482
13	3855	1674
14	3620	1350
15	3076	3236
16	2256	6304
17	1376	6028
18	674	1508
19	254	1302
20	69	2566
21	12	4314
22	1	2804
Total	40320	40320
Avg. MCT gates	11.24	15.88

TABLE V: Results of benchmarks.

		MMD	LNN Transformation		Proposed Synthesis	$Comparison_1$		$Comparison_2$	
$Benchmarks$	L	G_{std}	SWAP	W/O SWAP	LNN	Rd_1	(%)	Rd_2	(%)
3_17_6	3	11	11	11	11	0	0.00	0	0.00
ex-1_82	3	4	4	4	4	0	0.00	0	0.00
fredkin_3	3	3	3	3	3	0	0.00	0	0.00
ham3_28	3	8	14	12	12	2	14.29	0	0.00
miller_5	3	6	6	6	16	-10	-	-10	-
peres_4	3	2	2	2	2	0	0.00	0	0.00
toffoli_1	3	1	1	1	1	0	0.00	0	0.00
4_49_7	4	32	68	56	58	10	14.71	-2	-
aj-e11_81	4	22	64	50	68	-4	-	-18	-
hwb4_12	4	24	102	76	40	62	60.78	36	47.37
toffoli_double_2	4	2	14	10	10	4	28.57	0	0.00
hwb5_13	5	75	351	259	164	187	53.28	95	36.68
mod5d1_16	5	8	74	52	52	22	29.73	0	0.00
mod5d2_17	5	11	41	31	30	11	26.83	1	3.23
mod5mils_18	5	9	63	45	45	18	28.57	0	0.00
graycode6_11	6	5	5	5	5	0	0.00	0	0.00
hwb6_14	6	153	1119	797	569	550	49.15	228	28.61
mod5adder_66	6	38	386	270	231	155	40.16	39	14.44
ham7_29	7	388	4330	3016	1942	2388	55.15	1074	35.61
hwb7_15	7	399	4365	3043	1826	2539	58.17	1217	39.99
hwb8_64	8	905	12953	8937	5963	6990	53.96	2974	33.28
urf2_73	8	777	10731	7413	5862	4869	45.37	1551	20.92
hwb9_65	9	2105	39431	26989	16885	22546	57.18	10104	37.44
urf1_72	9	1827	33237	22767	16321	16916	50.90	6446	28.31
urf5_76	9	730	13330	9130	10412	2918	21.89	-1282	-
urf3_75	10	3436	85264	57988	40393	44871	52.63	17595	30.34
urf4_89	11	10535	313163	212287	120805	192358	61.42	91482	43.09
cycle10_2_61	12	19	559	379	4408	-3849	-	-4029	-
plus63mod4096_79	12	342	7302	4982	2034	5268	72.14	2948	59.17
plus63mod8192_80	13	405	8805	6005	2391	6414	72.84	3614	60.18
0410184_85	14	68	3980	2676	632	3348	84.12	2044	76.38
ham15_30	15	234479	14084867	9468071	4423186	9661681	68.60	5044885	53.28
urf6_77	15	122324	6428786	4326632	4349978	2078808	32.34	-23346	-

L: No. of lines; G_{std}: No. of gates in standard MCT circuits obtained from MMD; SWAP: No. of gates in LNN circuits when SWAP gates are used in transformation; W/O SWAP: No. of gates in LNN circuits when SWAP gates are not used in transformation; LNN: No. of gates in LNN circuits obtained from proposed synthesis method and Rd.: No. of gate reduction; $(\%)$: Percentage of gate reduction

[4] C. D. Hill, E. Peretz, S. J. Hile, M. G. House, M. Fuechsle, S. Rogge, M. Y. Simmons, and L. C. L. Hollenberg, "A surface code quantum computer in silicon," *Science Advances*, vol. 1, no. 9, 2015.

[5] A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, "Quantum-error correction on linear-nearest-neighbor qubit arrays," *Phys. Rev. A*, vol. 69, p. 042314, Apr 2004.

[6] P. Pham and K. M. Svore, "A 2d nearest-neighbor quantum architecture for factoring in polylogarithmic depth," *Quantum Info. Comput.*, vol. 13, no. 11-12, pp. 937–962, Nov. 2013.

[7] T. Toffoli, "Reversible computing," *Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci*, 1980.

[8] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, pp. 219–253, 1982.

[9] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Design Automation Conference*, June 2003.

[10] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Information Processing*, vol. 10, no. 3, pp. 73–89, 2011.

[11] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," in *Proceedings of the International Symposium on Multiple-Valued Logic*, pp. 288–293, 2011.

[12] M. M. Rahman and G. W. Dueck, "Synthesis of linear nearest neighbor quantum circuits," in *10th International Workshop on Boolean Problems*, pp. 277–284, 2012.

[13] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 3, pp. 436–444, 2008.

[14] M. M. Rahman, G. W. Dueck, and J. D. Horton, "An algorithm for quantum template matching," *J. Emerg. Technol. Comput. Syst.*, vol. 11, no. 3, pp. 31:1–31:20, Dec. 2014.

[15] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *The American Physical Society*, vol. 52, pp. 3457–3467, 1995.

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

[17] A. Kole, K. Datta, I. Sengupta, and R. Wille, "Towards a cost metric for nearest neighbor constraints in reversible circuits," in *Conference on Reversible Computation*, 2015.

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[19] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, pp. 220–225, 2008 RevLib is available at http://www.revlib.org.