Post Synthesis Optimization of SWAP Fredkin Based Reversible Circuits

Md Asif Nashiry
Dept. of Mathematics and Computer Science
University of Lethbridge
Lethbridge, AB, Canada
Email: asif.nashiry [AT] uleth.ca

Jacqueline E. Rice
Dept. of Mathematics and Computer Science
University of Lethbridge
Lethbridge, AB, Canada
Email: j.rice [AT] uleth.ca

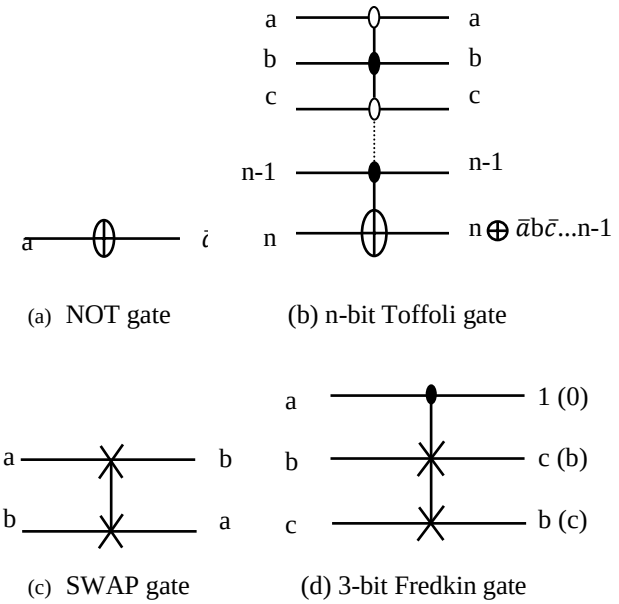
***Abstract*— Most existing synthesis approaches in reversible logic result in circuits that may not be optimal in terms of cost metrics such as the gate count, the number of garbage lines or the quantum cost. Hence post synthesis optimization approaches are used to generate simplified circuits. This paper proposes ten templates for optimizing SWAP and Fredkin gates based reversible circuits. We have also proposed the moving rule which can be used in SWAP-Fredkin based circuits. We have applied these templates with moving rule in SWAP and Fredkin gates-based circuits, and achieved (on average) a 16% reduction in quantum cost.**

Keywords-reversible logic; synthesis; post synthesis optimization; template matching; moving rule; logic gates

I. INTRODUCTION AND MOTIVATION

In recent years, reversible computing has established itself as a promising research area and emerging technology. This is motivated by a widely supported prediction that conventional computer hardware technologies will reach their limits in the near future. Limitations of traditional computing, such as heat dissipation, can become an obstacle for the further development of current technology [1]. Reversible computing [2] offers a solution to this potential deadlock of further development in traditional computing. The concept of synthesis is very important in designing reversible logic circuits. Synthesis refers to the transformation of a logic function into a corresponding logic circuit. There can be more than one reversible circuit for implementing a single function. The relationship between the inputs and the outputs of a logic function determines the number of the logic gates, type of logic gates used, and the order in which the logic gates appear in the circuit. If a logic function is already reversible, the synthesis process can take place immediately. However, if a logic function is not reversible, the first step in most synthesis algorithms is to transform the irreversible function into a reversible one. One or more garbage outputs and/or constant inputs are added to an irreversible function in order to transform the irreversible logic function into a reversible logic function [3].

The minimum number of garbage outputs which are required in order to transform an irreversible function into a reversible function is $\log_2 K$, where K is the maximum number of a repeated pattern in the output of an irreversible function [3]. A reversible circuit generated by a synthesis method may not be optimal from the perspective of the number of garbage lines, quantum cost and/or gate count. A circuit design that offers fewer garbage lines and/or lower GC and QC is desirable.



(a) NOT gate (b) n-bit Toffoli gate

(c) SWAP gate (d) 3-bit Fredkin gate

Figure 1: Commonly used reversible logic gates

After synthesis takes place, several strategies can be used in order to simplify reversible circuits, including template matching optimization [4-7] and rule based optimization [8, 9]. In this paper, we have considered template matching as post synthesis optimization technique in order to simpify reversible circuits.

The rest of the paper is organized as follows: Section II provides the fundamentals of reversible computing; Section III describes the basis of template matching; our proposed templates are introduced in Section IV; Section V describes our proposed moving rule, this section also introduces our proposed template matching algorithm followed by the experimental results; Section VI draws the conclusion of this paper and provides future research direction.

II. BACKGROUND

A. Reversible Logic

A reversible logic function has the form $f: B^n \rightarrow B^n$, where n is a non-negative integer and the domain $B = \{0,1\}$, with the key feature being that the function is bijective. More specifically, the number of inputs and the number of outputs of a reversible function are exactly the same. In particular, there is always a distinct output state for each of the possible input states [1, 2].

B. Revesible Logic Gates

Let $X \coloneqq \{x_1, x_2, \ldots\ldots, x_n\}$ be the set of Boolean variables. Then a reversible gate has the form $g(C,T)$, where $C = \{x_{i1}, \ldots\ldots, x_{ik}\} \in X$ is the set of control lines and $T = \{x_{j1}, \ldots\ldots, x_{jl}\} \in X$ with $C \cap T = \emptyset$ is the set of target lines [10].

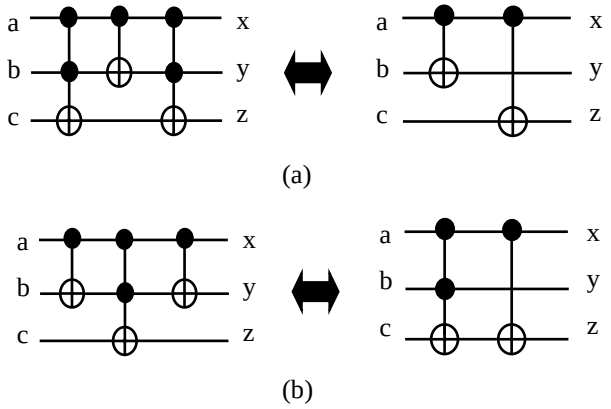


Figure 2: Two templates presented in [11]

Two commonly used reversible gates are Toffoli gates and Fredkin gates [10]. A Toffoli gate with no controls is a NOT gate i.e. $g(0, x_{j1})$. Similarly, a Toffoli gate $g(x_{i1}, x_{j1})$ can be thought of as a controlled NOT (or CNOT) gate, and $g(\{x_{i1}, \ldots\ldots, x_{in}\}, x_{j1})$ is a n-bit Toffoli gate. A Fredkin gate with no controls is a SWAP gate $g(x_{j1}, x_{j2})$, which interchanges the two target input bits at output. A n-bit positive control Fredkin gate $g(\{x_{i1}, \ldots\ldots, x_{in}\}, x_{j1}, x_{j2})$ interchanges the two target bits at output when all the control inputs are equal to 1. A reversible gate may also have negative control. In this case the gate becomes active when negative control has a value of 0. Fig. 1 shows several commonly used reversible logic gates.

C. CostMetrics

Two important metrics used to compare reversible circuit implementations are gate count and quantum cost. The gate count (GC) is the number of gates in a circuit and the quantum cost (QC) is the number of basic quantum gates required to implement macro-level reversible gates such as the Toffoli and Fredkin gates [12, 13]. For example, the QC of a CNOT gate is 1, and QC of a SWAP gate is 3. The QC of a (3×3) Toffoli and a (3×3) Fredkin gate is 5 [14].

III. TEMPLATE MATCHING

A template consists of two patterns of gates which are equivalent to each other. Template matching is a process to find a pattern of gates that can be replaced by another equivalent pattern of gates in order to simplify a circuit design. Miller et al. introduce templates for 2 and 3 input reversible circuits as well as a template matching algorithm [11]. This algorithm searches for a pattern of gates in a reversible circuit and replaces the pattern by another simpler pattern of gates. An extension of this algorithm is presented in [38]. Fig. 2 shows two templates presented in [11]. The output functions of both circuits in Fig. 2(a) are evaluated as $x = a$, $y = a \oplus b$ and $z = a \oplus c$. Thus these two circuits perform the same reversible function. The left hand circuit has a GC of 3 and QC of 11. However, both the GC and QC of the right hand circuit are 2. Therefore, the right hand circuit design is more efficient in terms of GC and QC. Maslov et al. introduced some templates based on Toffoli and Fredkin gates in [15]. Templates based on both positive and negative controls are presented by Datta et al. [16] and Rahman et al. [7]. Iwama et al. also present rules which can be used to simplify reversible circuits [9]. Other rule based post synthesis optimization works include [8, 17].

The deletion rule for NCT-based circuits can also be used to optimize SF circuits. However, not all NCT-based rules are useful for optimizing SF based circuits. For example, the moving rule proposed for NCT gates is a useful approach for simplifying reversible circuits. The moving rule states that two adjacent gates $g_1(c_1, t_1)$ and $g_2(c_2, t_2)$ can be interchanged if the target of one gate is not a control of another gate, i.e. $c_1 \cap t_2 = \emptyset$ and $c_2 \cap t_1 = \emptyset$.

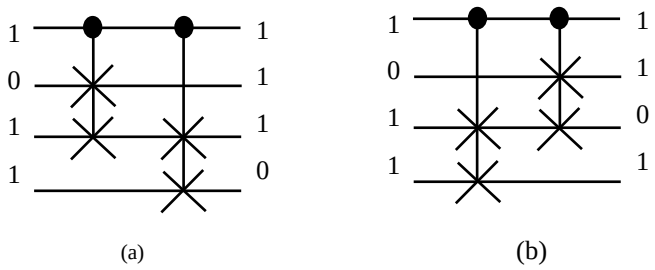


Figure 3: A SF circuit where the moving rule does not work

The moving rule is particularly useful in order to find a template in a circuit. However, this moving rule works only on NCT based reversible circuits. Fig. 3 shows an example of a SF based circuit where the moving rule cannot be applied. According to the moving rule, two gates can be interchanged if controls of one gate are not the target of other gate. Fig. 3(a) shows a circuit that consists of two 3-bit Fredkin gates. Since the control of one gate is not a target of another gate, the two gates are interchanged as shown in Fig. 3(b). However, these two circuits are not equivalent. In this paper, we have modified the moving rule for applications in SF based reversible circuits.

IV. PROPOSED TEMPLATES

Most existing reversible circuit optimization techniques focus on NCT gates. This section presents templates for SF gate based reversible circuits. We consider both template matching and rule based simplification for circuit optimization. The basic difference between rule based simplification and template matching is that templates must match specific patterns of gates, while rules can be applied to a broad group of gates. For better understanding we refer to both templates and rules as templates in this dissertation. Note that G(C;T) represents a gate G from the SF gates family. C and T represent the sets of the control points and the targets of G, respectively. We are describing the operations of only two templates in detail due to the page limitation.

Template 1: Two adjacent gates $G_1(C_1;T_1)$ and $G_2(C_2;T_2)$ can be removed from a circuit if $C_1 = C_2$ and $T_1 = T_2$. That is, if targets of a SWAP gate are on the same line as that of an adjacent SWAP gate, the two SWAP gates can be removed from the circuit. In case of a Fredkin gate, when the controls and targets of two adjacent gates are the same in polarity and operate on the same line, the two Fredkin gates have no effect on circuit operation.

(i) $SWAP(t_1,t_2)SWAP(t_1t_2) \equiv I$
(ii) $FRED(c;t_1,t_2)FEED(c;t_1t_2) \equiv I$
(iii) $FRED(\bar{c};t_1,t_2)FEED(\bar{c};t_1t_2) \equiv I$

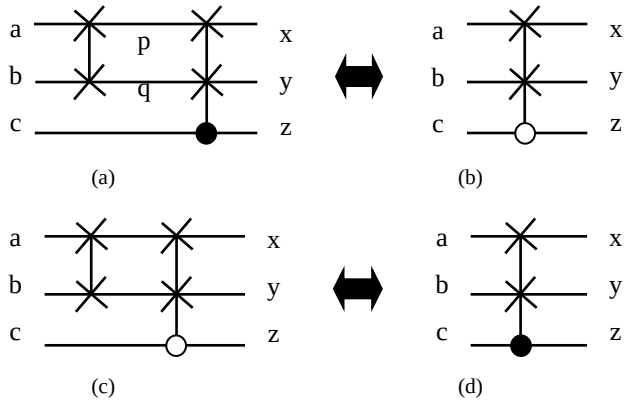


Figure 4: Template 2

Template 2: The next template can be applied when a cascade of a SWAP gate and a 3-bit positive control Fredkin gate appear in such a way that the targets of the SWAP and the Fredkin gates are on the same lines of a circuit. This sequence of two gates can be replaced by a 3-bit negative control Fredkin gate. The control and the targets of the negative control Fredkin gate appear on the corresponding lines where the control and the targets of the positive control Fredkin gate appear. That is, for two adjacent gates $G_1(T_1)$ and $G_2(C;T_2)$ if $T_1 = T_2$, the two gates can be replaced by $G_1(\bar{C};T_1)$.

(i) $SWAP(t_1,t_2)FRED(c;t_1,t_2) \equiv FRED(\bar{c};t_1,t_2)$

(ii) $SWAP(t_1,t_2)FRED(\bar{c};t_1,t_2) \equiv FRED(c;t_1,t_2)$

Suppose p and q are the two outputs of the SWAP gate in Fig. 4(a). Here $p = b$ and $q = a$. The output of the Fredkin gate will be $x = \bar{c}p \oplus cq = \bar{c}b \oplus ca, y = cp \oplus \bar{c}q = cb \oplus \bar{c}a, z = c$. The outputs of the negative control Fredkin gate in this figure are $x = \bar{c}b \oplus ca, y = cb \oplus \bar{c}a, z = c$. Thus, the two circuits in this figure are equivalent to each other. Template 2 reduces both GC and QC by 1.

Template 3: Two adjacent gates $G_1(C_1;T_1)$ and $G_2(\bar{C}_2;T_2)$ by $G(T_1)$ if $C_1 = \bar{C}_2$ and $T_1 = T_2$. Template 3 reduces QC by 70% and GC by 1.

$$FRED(c;t_1,t_2)FRED(\bar{c};t_1,t_2) \equiv SWAP(t_1,t_2)$$

Template 4: This template is applicable when a 3-bit Fredkin gate, $FRED(C_1;T_1)$ and a 4-bit Fredkin gate, $FRED(C_2;T_2)$ appear in such a way that $C_1 \cap C_2 = C_1$ and $T_1 = T_2$. Template 4 reduces QC from 18 to 13. The GC is also reduced to 1.

$$FRED(c_1;t_1,t_2)FRED(c_1,c_2;t_1,t_2) \equiv FRED(c_1,\bar{c}_2;t_1,t_2)$$
$$FRED(\bar{c}_1;t_1,t_2)FRED(\bar{c}_1,\bar{c}_2;t_1,t_2) \equiv FRED(\bar{c}_1,c_2;t_1,t_2)$$

Template 5: The next template is an example of a circuit optimization using template matching when a pattern of gates is replaced by another pattern of gates. The QC is reduced by 3 after applying this template to a circuit.

$$SWAP(t_1,t_2)FRED(\bar{t}_2;t_1,c)FRED(c;t_1,t_2) \equiv FRED(\bar{c};t_1,t_2)FRED(\bar{t}_2;t_1,c)$$

Template 6: Template 6 can be applied to simplify two n-bit Fredkin gates when $n \geq 4$. Two adjacent n-bit Fredkin gates $G_1(C_1 \cup c_i;T_1)$ and $G_2(C_2 \cup \bar{c}_i;T_2)$ can be replaced by a $n-1$-bit Fredkin Gate $G_3(C_3;T_3)$, where $C_1 = C_2 = C_3$, and $T_1 = T_2 = T_3$.

$$FRED(c_1,c_2;t_1,t_2)FRED(\bar{c}_1,c_2;t_1,t_2) \equiv FRED(c_2;t_1,t_2)$$
$$FRED(\bar{c}_1,\bar{c}_2;t_1,t_2)FRED(c_1,\bar{c}_2;t_1,t_2) \equiv FRED(\bar{c}_2;t_1,t_2)$$
$$FRED(c_1,c_2,c_3;t_1,t_2)FRED(c_1,c_2,\bar{c}_3;t_1,t_2) \equiv FRED(c_1,c_2;t_1,t_2)$$
$$FRED(\bar{c}_1,\bar{c}_2,c_3;t_1,t_2)FRED(\bar{c}_1,c_2,\bar{c}_3;t_1,t_2) \equiv FRED(\bar{c}_1,\bar{c}_3;t_1,t_2)$$

Template 7: Template 7 can be applied when two adjacent 4-bit Fredkin gates appear in such a way that the controls and the targets of both gates are on the same line. However, the polarities of control points on same line are different. The GC remains the same after applying this template, however, the QC reduces from 26 to 10.

$$FRED(c_1,\bar{c}_2;t_1,t_2)FRED(\bar{c}_1,c_2;t_1,t_2) \equiv FRED(c_1;t_1,t_2)FRED(c_2;t_1,t_2)$$

Template 8: The GC remains the same after applying this template, however, the QC is reduced by 2.

$$FRED(\bar{c}_1; t_1, t_2)FRED(c_1, c_2; t_1, t_2) \equiv SWAP(t_1, t_2)FRED(c_1, \bar{c}_2; t_1, t_2)$$
$$FRED(c_1; t_1, t_2)FRED(\bar{c}_1, \bar{c}_2; t_1, t_2) \equiv SWAP(t_1, t_2)FRED(\bar{c}_1, c_2; t_1, t_2)$$

Template 9: Template 9 is applicable to 5-bit Fredkin gates. Unlike other templates, Template 9 increases GC by 1. However, QC is reduced from 58 to 31.

$$FRED(c_1, c_2, c_3; t_1, t_2)FRED(\bar{c}_1, \bar{c}_2, c_3; t_1, t_2) \equiv FRED(c_3; t_1, t_2)FRED(c_1, c_3; t_1, t_2)FRED(c_2, c_3; t_1, t_2)$$
$$FRED(\bar{c}_1, \bar{c}_2, \bar{c}_3; t_1, t_2)FRED(c_1, c_2, \bar{c}_3; t_1, t_2) \equiv FRED(\bar{c}_3; t_1, t_2)FRED(\bar{c}_1, \bar{c}_3; t_1, t_2)FRED(\bar{c}_2, \bar{c}_3; t_1, t_2)$$

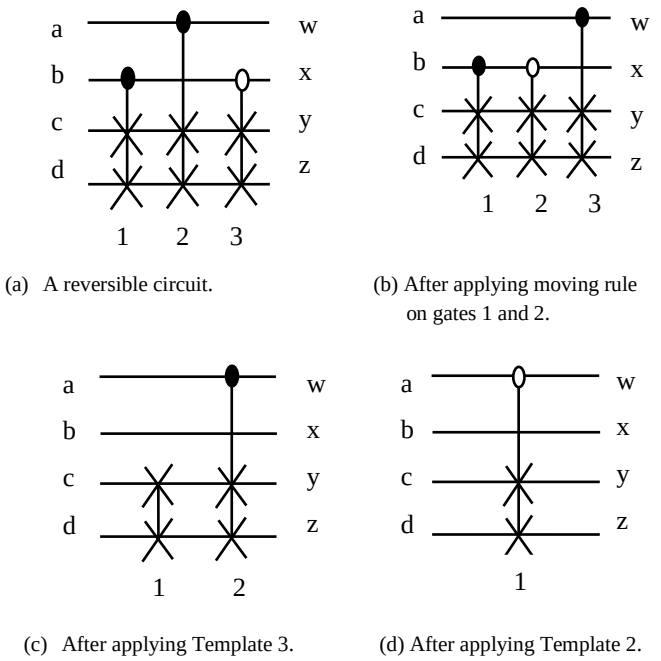
Template 10: Template 10 works for two 5-bit Fredkin gates. The GC for this template remains the same, however QC is reduced from 58 to 26.

$$FRED(c_1, \bar{c}_2, c_3; t_1, t_2)FRED(\bar{c}_1, c_2, c_3; t_1, t_2) \equiv FRED(c_2, c_3; t_1, t_2)FRED(c_1, c_3; t_1, t_2)$$

V. Reversible Circuits Optimization

This section introduces a proposed modified moving rule which can be used for optimizing SF gate based circuits. Two adjacent SF gates $G_1(C_1, T_1)$ and $G_2(C_2, T_2)$ can be interchanged if $C_1 \cap T_2 = \emptyset$ and $C_2 \cap T_1 = \emptyset$, and either $T_1 \cap T_2 = \emptyset$ or $T_1 = T_2$. In other words, two adjacent gates from the SF gate family can be interchanged if two conditions hold: (i) no control of one gate is a target of another gate, and (ii) targets of both gates are on the same line, or targets of both gates are on different lines. We have used an algorithm that incorporates the moving rule in order to apply the proposed templates. This algorithm is based on the algorithm presented in [7]. The moving rule increases the chances to match more templates, which can optimize a circuit even further. For example, consider the (4×4) reversible circuit shown in Fig. 5(a). The GC and QC of this circuit are 3 and 15 respectively. The gates of this circuit do not match any of the proposed templates. However, it can be observed that for the gates labeled 1 and 2, no control point of any gate is on the target lines of the other gate. In addition, the targets of both gates are on the same line. So according to the moving rule, it is possible to interchange the position of these two gates. After applying the moving rule, the circuit becomes as shown in Fig. 5(b). Now gates 2 and 3 match Template 3. Gates 2 and 3 can be replaced by a SWAP gate, as shown in Fig. 5(c). The two gates in Fig. 5(c) match Template 2. The resulting circuit after applying Template 2 is presented in Fig. 5(d) with a GC of 1 and QC of 5.

The template matching algorithm maintains two lists of gates: an input list and an output list. The input list includes all the gates which appear in the original circuit. The output list stores the gates after the process of simplification is finished. The algorithm reads the input list and the list of all templates, and applies templates when a match is found. When a sequence of gates is replaced by a template, the new sequence of gates is stored in the output list. The algorithm processes the next sequence of gates from the input list. At each step, the algorithm decides whether a sequence of gates is to be replaced by a template or not. If no match is found for a sequence of gates, the algorithm applies the moving rule to increase the possibility of finding a match. A sequence of gates that does not match any template is also stored in the output list. Algorithm 1 shows the major steps involved in the



(a) A reversible circuit.

(b) After applying moving rule on gates 1 and 2.

(c) After applying Template 3.

(d) After applying Template 2.

Figure 5: An example to show the role of moving rule and templates in optimization

template matching algorithm. The algorithm executes the CHECK TEMPLATE (input gate list) procedure in order to find a match. The algorithm terminates its execution when no more templates can be applied to the input gate list.

Algorithm 1: Template matching algorithm.
1: Input : input gate list
2: Output : output gate list
3: procedure CHECK TEMPLATE(input gate list)
4: Count the number of gates in the input gate list
5: Repeat while the number of gates in the input gate list > 2
6: if a match is found then
7: Apply Template (input gate list)
8: else
9: Apply Moving Rule (input gate list)
10: end if
11: end procedure
12:

13: *procedure APPLY TEMPLATE(input gate list)*
14: if two adjacent gates g_i and g_{i+1} match a template then
15: Append the template gates to the output gate list
16: Remove g_i and g_{i+1} from the input gate list
17: end if
18: if three adjacent gates g_i, g_{i+1} and g_{i+2} match Template 5 then
19: Append the template gates to the output gate list
20: Remove g_i, g_{i+1} and g_{i+2} from the input gate list
21: end if
22: Return
23: end procedure
24:
25: *procedure APPLY MOVING RULE(input gate list)*
26: if two gates g_i and g_{i+1} can be interchanged then
27: Append g_{i+1} to the output gate list
28: Remove g_{i+1} from the input gate list
29: else
30: Append g_i to the output gate list
31: Remove g_i from the input gate list
32: end if
33: Return
34: end procedure

We have tested the algorithm on the benchmark circuits available on RevLib [18]. In the benchmarks there are only six circuits based on the SF gate family. The result of this experiment is presented in Table 1. The column labled 'Lines' in this table indicates the number of input bits. The GC and QC under the 'original circuit' column indicates the GC and QC of the circuits before applying the templates. The GC and QC under the 'optimized circuit' column represent GC and QC of the circuits after applying the templates. The template matching reduces GC and QC for two of the six benchmark circuits. The best results are achieved for hwb4 circuit, which sees 18% GC and 9% QC reduction. The number of circuits considered for this experiment is not large enough, since the number of benchmark circuits based on the SF gate family are very few. In addition, one circuit consists of only one gate which cannot be further optimized. In order to evaluate the efficiencies of the proposed templates from a broader perspective, we randomly generated 500 SF based circuits. The number of lines of these circuits varied from 3 to 7, similar to the benchmark circuits in RevLib. Based on the number of gates, these circuits are of three different sizes: 10, 50 and 100. Our proposed approach for circuit optimization has been applied to these randomly generated circuits, and a portion of the result is presented in Table 2. The highest percentage of reduction of both GC and QC is 91%. The percentage of reduction of GC on average is 17%. The average reduction of QC is 16%.

VI. CONCLUSION

Reversible circuits generated with the transformation based synthesis [11] may not be optimal. Template matching and rule based optimization techniques are two common approaches to simplify reversible circuits generated by transformation based synthesis. In this paper we have presented 10 templates based on template matching and rule based optimization. We have tested the proposed templates to simplify reversible circuits consisting of only SF gates. We have considered templates for both positive and negative control Fredkin gates. We have identified the fact that some rules proposed for NCT gates can not be applied for SF gates, e.g. the moving rule. We have modified the moving rule in order to apply this rule for optimizing SF based reversible circuits. We have also proposed an algorithm by following the principle proposed in [7] in order to apply the proposed templates and the moving rule. Since few SF gate based circuits are available as benchmark circuits in RevLib, we have randomly generated 500 SF based reversible circuits. The results of experiments suggest that our proposed templates can contribute to optimizing SF gate based reversible

Table 1: Results after applying the proposed algorithm on benchmark circuits.

Circuits		Original circuits		Optimized circuits		% of reduction	
Functions	Lines	GC	QC	GC	QC	GC	QC
fredkin	3	1	5	1	5	0	0
hwb4	4	11	65	9	59	18.18	9.23
hwb5	5	24	214	24	214	0	0
decode24	6	3	15	3	15	0	0
hwb6	6	65	1115	64	1112	1.54	0.27
hwb7	7	116	3998	166	3998	0	0

Table 2: Results after applying the proposed algorithm on randomly generated circuits.

Circuits		Original circuits		Optimized circuits		% of reduction	
Functions	Lines	GC	QC	GC	QC	GC	QC
random173	5	100	695	94	673	6	3.17
random297	4	10	42	8	36	20	14.29
random172	6	55	878	55	878	0	0
random298	6	55	715	54	710	1.82	0.7
random171	7	10	203	10	203	0	0
random299	6	100	1385	97	1376	3	0.65
random178	7	55	1093	55	1093	0	0
random8	5	100	624	95	609	5	2.4
random163	3	55	165	31	93	43.64	43.64
random286	5	55	375	50	360	9.09	4
random166	3	55	165	15	45	72.73	72.73
random283	6	55	887	54	882	1.82	0.56
random164	4	100	410	77	329	23	19.76
random422	3	100	300	16	48	84	84
random302	3	100	300	30	90	70	70
random423	4	10	40	10	40	0	0
random303	3	10	30	2	6	80	80
random424	7	55	1253	52	1238	5.45	1.2
random500	6	100	1234	100	1234	0	0
random4	7	55	1097	51	1085	7.27	1.09
random193	5	55	392	53	384	3.64	2.04

circuits. The highest reduction in QC is 9% after applying the proposed templates on benchmark circuits. In case of randomly generated circuits we have achieved 16% reduction in QC on average.

Identifying more templates based on SF gates or on a combination of NCT and SF gates is an area for future research. Our experimental results show that the QC is reduced up to 9% after applying the proposed templates on benchmark circuits. The QC can be reduced further using an efficient template matching algorithm. The algorithm which we have used to apply the templates uses an exhaustive search approach to match templates. In [7], Rahman et al. proposed a template matching algorithm that assigns ranks to the templates based on the amount of QC reduction offered by the templates. Thus their proposed algorithm applies templates that offer the best possible reduction in QC at a particular instant. This indicates that developing an efficient SF gate based template matching algorithm can also be an area of future study.

REFERENCES

[1] Michael P Frank. "Approaching the physical limits of computing". In Proceedings of 35th International Symposium on Multiple-Valued Logic, IEEE, pp. 168–185, 2005.

[2] Michael P Frank. "Introduction to reversible computing: motivation, progress, and challenges". In Proceedings of the 2nd Conference on Computing Frontiers, ACM, pp. 385–390, 2005.

[3] Dmitri Maslov and Gerhard W Dueck. "Reversible cascades with minimal garbage". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23(11), pp. 1497–1509, 2004.

[4] Nabila Abdessaied, Mathias Soeken, RobertWille, and Rolf Drechsler. "Exact template matching using Boolean satisfiability". IEEE 43rd International Symposium on In Multiple-Valued Logic (ISMVL), pp 328–333, 2013.

[5] Kamalika Datta, Indranil Sengupta, and Hafizur Rahaman. "A post-synthesis optimization technique for reversible circuits exploiting negative control lines". IEEE Transactions on Computers, vol. 64(4), pp. 1208–1214, 2015.

[6] Dmitri Maslov, Gerhard W Dueck, and D Michael Miller. "Simplification of Toffoli networks via templates". In Proceedings of 16th Symposium on Integrated Circuits and Systems Design. pp. 53–58, 2003.

[7] Md Zamilur Rahman and Jacqueline E Rice. "Templates for positive and negative control Toffoli networks". In International Conference on Reversible Computation, pp. 125–136, 2014.

[8] Mona Arabzadeh, Mehdi Saeedi, and Morteza Saheb Zamani. "Rule-based optimization of reversible circuits". In Proceedings of the Asia and South Pacific Design Automation Conference, pp. 849–854, 2010.

[9] Kazuo Iwama, Yahiko Kambayashi, and Shigeru Yamashita. "Transformation rules for designing cnot-based quantum circuits". In Proceedings of the 39th annual Design Automation Conference, pp. 419–424, 2002.

[10] T. Toffoli. "Reversible computing", MIT, Tech. Rep. MIT/LCS/TM No. 151, 1980.

[11] D Michael Miller, Dmitri Maslov, and Gerhard W Dueck. "A transformation based algorithm for reversible logic synthesis". In Proceedings of the 40th annual Design Automation Conference, pp. 318–323, 2003.

[12] 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 computations," Physical Review A, vol. 52, no. 5, pp. 3457–3467, 1995.

[13] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum fredkin gate," Physical Review A, vol. 53, no. 4, pp. 2855–2856, 1996.

[14] Md Asif Nashiry, Mozammel H. A. Khan and Jacueline E. Rice. "Controlled and uncontrolled SWAP gates in reversible logic synthesis", International Conference on Reversible Computation. pp. 141-147, 2017.

[15] Dmitri Maslov, Gerhard W Dueck, and D Michael Miller. Fredkin/Toffoli templates for reversible logic synthesis. In Proceedings of international conference on Computer-aided design, pp. 256, 2003.

[16] Kamalika Datta, Gaurav Rathi, Robert Wille, Indranil Sengupta, Hafizur Rahaman, and Rolf Drechsler. "Exploiting negative control lines in the optimization of reversible circuits". In International Conference on Reversible Computation, pp. 209–220, 2013.

[17] Xueyun Cheng, Zhijin Guan,WeiWang, and Lingling Zhu. "A simplification algorithm for reversible logic network of positive/negative control gates". In Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on, pp. 2442–2446. 2012.

[18] Mathias Soeken, Stefan Frehse, Robert Wille, and Rolf Drechsler. Revkit: "A toolkit for reversible circuit design". Multiple-Valued Logic and Soft Computing, vol. 18(1). pp. 55–65, 2012.