SWAP-Fredkin Based Synthesis in Reversible Logic

Md Asif Nashiry, Jacqueline E. Rice

Abstract— 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. This paper proposes a transformation based synthesis approach for realizing conservative reversible functions using SWAP and Fredkin gates. The proposed SWAP and Fredkin gates approach is compared with NOT, CNOT and Toffoli gates approach. Experimental results show that synthesizing conservative reversible functions using SWAP and Fredkin gates is more efficient than comparable approaches using NOT, CNOT and Toffoli gates.

Index Terms—Revesible logic, logic gates, tranformation based synthesis, conservative functions, .

1 Introduction and Motivation

THE amount of information processed by digital devices continues to increases over time. In order to process this increasing volume of information, the number of components fabricated on integrated circuits of a digital device is also increasing over time. Over the past few decades, the size of the components has been reduced in order to increase the density of these components on integrated circuits. However, this pattern cannot continue forever because the current technology is approaching the physical limits of computing [1]. Limitations of traditional computing, such as heat dissipation, can become an obstacle for the further development of current technology [1, 2]. 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. According to some synthesis approaches, if a logic function is irreversible, the first step of a logic synthesis is to transform the function into its reversible equivalent. One or more garbage lines and/or constant inputs are included in the original irreversible function in order to make the function reversible. The final step is to transform the reversible function into a logic circuit consisting of one or more reversible gates which are connected in cascade. The resulting circuit is a reversible circuit. There can be more than one reversible circuit for implementing a single function. Two important factors play a significant role in transforming irreversible functions into reversible circuits: (1) the number of garbage lines and/or constant inputs, which are included in order to transform the irreversible function to a reversible one; and (2) the use of different reversible logic gates for realizing the reversible function by a reversible circuit. During the process of transforming an irreversible function into the corresponding reversible function, it is necessary to observe the output of the irreversible function. If the output of an irreversible function has a maximum number of k identical patterns, the minimum number of garbage lines required to make the function reversible is $\log_2 k$ [3]. A number of logic synthesis techniques in reversible logic have been proposed as described in [4]. In this work we focus on transformation based logic synthesis.

The organization of this paper is as follows: Section 2 presents the fundamentals of reversible logic; Section 3 describes the basis of the transformation based synthesis; Section 4 introduces our proposed approach; Section 5 and Section 6 present the experimental result; Section 7 concludes the paper and provides future directions.

- *Md Asif Nashiry, Dept.of Mathematics and Computer Science, University of Lethbridge, Alberta, Canada. E-mail:asif.nashiry@uleth.ca*
- *Jacqueline E. Rice, Dept.of Mathematics and Computer Science, University of Lethbridge, Alberta, Canada. E-mail:j.rice@uleth.ca*

2 Background

2.1 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 [2]. When the number of 1s in the input and the output are equal, such a function is called a conservative function.

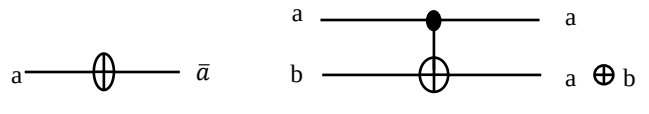
2.2 Reversible Logic Gates

Let $X \coloneqq \{x_1, x_2, \dots\dots, x_n\}$ be the set of Boolean variables. Then a reversible gate has the form $g(C,T)$, where $C = \{x_{i1}, \dots\dots, x_{ik}\} \in X$ is the set of control lines and T= $\{x_{j1}, \dots\dots, x_{jl}\} \in X$ with $C \cap T = \emptyset$ is the set of target lines [8 pacrim]. Two commonly used reversible gates are Toffoli gates and Fredkin gates. 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 $g(\{x_{i1}, \dots\dots, x_{in}\}, x_{j1})$ is a n-bit Toffoli gate. A Fredkin gate

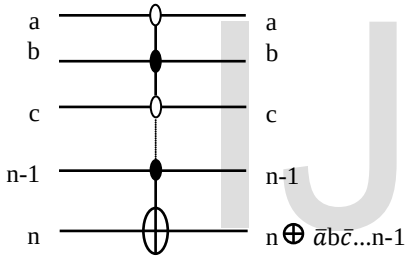
with no controls is a SWAP gate $g(x_{j1}, x_{j2})$, which interchanges the two target input bits at output. A n-bit positive control Fredkin gate $g(\{x_{i1}, \ldots\ldots, x_{in}\}, x_{j1}, x_{j2})$ interchanges the two target bits at output when all the control inputs are equal to 1. A reversible gate may also have negative control. In this case the gate becomes active when negative control has a value of 0.

2.3 Cost Metrics

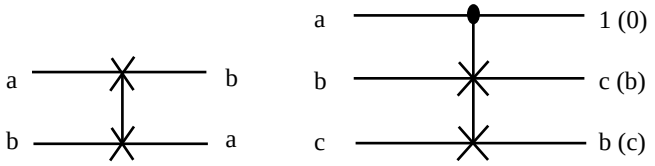
Two important metrics used to compare reversible circuit implementations are gate count and quantum cost. The gate count (GC) is the number of gates in a circuit and the quantum cost (QC) is the number of basic quantum gates required to implement macro-level reversible gates such as the Toffoli and Fredkin gates [5, 6]. For example, the QC of 1-CNOT gate is 1, and the QC of a SWAP gate is 3. The QC of a (3 × 3) Toffoli and a (3 × 3) Fredkin gates is 5 [7].



(a) NOT gate (b) CNOT gate



(c) n-bit Toffoli gate



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

3 The Transformation Based Synthesis Approach

A transformation based approach [8] takes as its input a truth table of a reversible function, and applies reversible logic operations to transform the function into an identity function. The gates which perform these logic operations during the tranformation constitute the circuit that implements the input reversible function. The gates appear in the circuit in the same order in which the logical operations are performed during transformation. Before the synthesis takes place if a function is not reversible, the first step is to transform the irreversible function into a reversible function. Works such as proposed by Maslov et al. [3] and Miller et al. [9] describe techniques for this. One of the major advantages of a transformation-based synthesis approach is that the process of generating circuits based on this approach does not create any garbage output or constant input lines. Thus, in terms of the number of inputs and outputs lines, the size of the circuit generated by a transformation based synthesis is minimal. The transformation based synthesis algorithm was proposed by Miller et al. [8]. The authors demonstrated two variations: a basic algorithm and a bidirectional algorithm, both based on the NCT gate library. In the basic algorithm, the reversible logic operations are applied to the output of the function's truth table. The following is the basis of transformation based logic synthesis approach.

Step 0: If $f(0) = 0$, no transformation is required; go to step 1. If $f(0) \neq 0$, apply a (1×1) Toffoli gate (NOT gate) in order to achieve $f(0) = 0$.

Step 1: For $1 \leq i < 2^m - 1$: If $f(i) = i$, no transformation is required and proceed to next i. If $f(i) \neq i$, apply the smallest $(k \times k)$ Toffoli gate, $k = 2$ to n in order to make $f(i) = i$.

Table 1: Truth table of a (3 × 3) reversible function.

	Input			Output			
	a_i	b_i	c_i	a_o	b_o	c_o	
(0)	0	0	0	0	0	0	(0)
(1)	0	0	1	1	0	0	(4)
(2)	0	1	0	0	0	1	(1)
(3)	0	1	1	0	1	1	(3)
(4)	1	0	0	0	1	0	(2)
(5)	1	0	1	1	0	1	(5)
(6)	1	1	0	1	1	0	(6)
(7)	1	1	1	1	1	1	(7)

The choice of gate during each step of transformation is crucial in order to maintain convergence. The gate chosen in each step of transformation must not change the order of bits of the previous steps. Consider the (3×3) reversible function $f = \sum(0,4,1,3,2,5,6,7)$ in Table 1. The circuit which is generated by following the basic transformation algorithm is presented in Figure 2.

4 SF Based Synthesis Approach

Before describing our proposed approach, it is important to observe a significant property of the function shown in Table 1. The truth table of the function shows that for each row, the number of 1s in the input is equal to the number of 1s in the output. Thus, the function is a conservative function. Our hypothesis is that a circuit realization for a conservative reversible function will be more efficient if we use SF gates instead of NCT gates. Thus we propose a SF gate based transformation approach. The underlying idea of SF-based transformation synthesis is the same as the approach described previously in this chapter. The difference is that

instead of using the logic gates from the NCT gate family, we use only SWAP and Fredkin gates to realize the transformations. While NCT gates manipulate bits by inverting the bits, SF gates interchange the bits when the control points of these gates satisfy the necessary condition. Since a conservative function has an equal number of 1s in the input and the output, our hypothesis is that the interchange of bits rather than the inversion of bits during the process of transformation will generate efficient circuits. The transformation based approach involves the mapping between two values consisting of the same number of 1s. The output of a SF gate consists of the same number of 1s as the input. Thus, the SF gates will be more suitable and require fewer logical operations than NCT gates for mapping one value into another of a conservative function.

As in the NCT version, the proposed approach examines one row of the truth table at each step. The objective is to make $f(i) = i$, for $i = 0$ to $2^n - 1$, where i is the i-th row of a reversible function f , and n is the number of inputs/outputs (bits) of the function. The following is the basis of SF gate base transformation approach.

Step 0: Since the function is a conservative function, the first row of the truth table of the function will be $f(0) = 0$. Thus, no transformation is required and go to step 1.

Step 1: For $i = 1$: If $f(1) = 1$, no transformation is required. If $f(1) \neq 1$, apply a SWAP gate in order to make $f(1) = 1$.

Step 2: Repeat for $i = 2$ to $2^n - 1$: If $f(i) = i$, no transformation is required. If $f(1) \neq 1$, apply a SWAP gate or the smallest $(k \times k)$ Fredkin gate in order to make $f(i) = i$, where $k = 3$ to n. One or more gates may be required in order to achieve $f(i) = i$.

Table 2: Transformation stages of the function in Table 1 using SF based synthesis

Output			Step 0			Step 1			Step 2			Step 3			Step 4		
			(i)			(ii)			(iii)			(iv)			(v)		
a	b	C	a^o	b^o	c^o	a^1	b^1	c^1	a^2	b^2	c^2	a^3	b^3	c^3	a^4	b^4	c^4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1
0	0	1	0	0	1	1	0	0	0	1	0	0	1	0	0	1	0
0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	0	1	1
0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0	1	0	1	1	1	1	0	1	0	1
1	1	0	1	1	0	0	1	1	1	0	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
			---			S(a,c)			S(a,b)			F(b;a,c)			F(a;b,c)		

We use the same function from Table 1 to demonstrate the SF-based transformation synthesis. As before, the proposed approach begins with the output of the function. Table 2 shows the transformation stages. In this table S(a;b) represents a SWAP gate with two targets, 'a' and 'b'. F(a;b,c) represents a Fredkin gate with a control point on line, 'a', and two targets on lines 'b' and 'c'. The resulting circuit realization of the function from Table 1 is displayed in Figure 3.

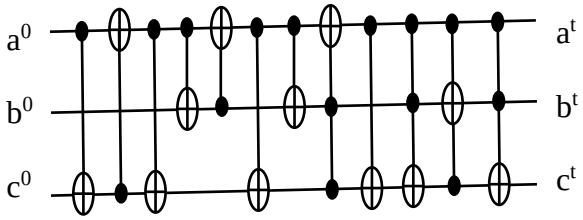


Figure 2: Circuit based on transformation based synthesis

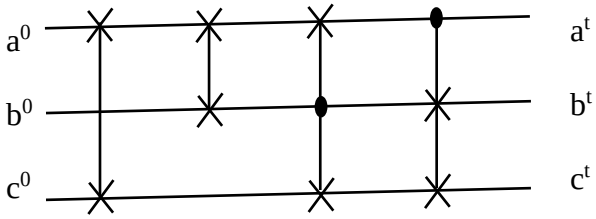


Figure 3: Circuit based on SF based synthesis

5 Comparison of Transformation Based Approaches

It is important to observe that the function in Table 1 is a conservative function and Figures 2 and 3 show two circuit designs for this function. In Figure 3, we have a gate count of 4 as compared to a gate count of 12 for the circuit in Figure 2. The quantum cost of the implementation in Figure 3 is (2x3)+(2x5) = 16, where the quantum cost for the circuit realization in Figure 2 is 28. The percentages of decrease in gate count and quantum cost are 67% and 43% respectively, which is a very significant improvement.

In order to compare the SF based transformation approach with NCT based transformation from a wider perspective, we have generated all possible (3×3) conservative reversible functions. We have realized all 36-(3×3) conservative functions using both algorithms. The highest percentage of reduction in gate count is 67% for more than half of the (3×3) conservative reversible functions. The ability of changing two bits at a time gives SF gates an advantage over the NCT gate family for realizing conservative reversible circuits.

SF based synthesis also performs better than NCT based synthesis when comparing quantum cost. Among the 36 functions, we have achieved lower QC for almost 70% of the functions. For the remaining functions, the QC is the same for both approaches. There is not a single instance where the NCT based synthesis performs better than our proposed approach. The highest percentage of decrease in quantum cost is 70% and the average percentage of reduction of quantum cost is 29%.

As mentioned above, the proposed transformation algorithm using the SF gate family follows the greedy approach. We have designed our algorithm in this way in order to offer a fair comparison, since the basic transfor-

mation based synthesis algorithm which is proposed in [8] also follows the greedy approach. At every step of transformation, the algorithm selects a gate which costs less in terms of quantum cost. For example, if we observe column (ii) of Table 2, we need to transform 100 into 010. There are two choices for this mapping. We could use either a SWAP gate $S(a, b)$ or a negative controlled Fredkin gate, $\bar{F}(c; a, b)$. The proposed SF gate based transformation selects a SWAP gate, $S(a, b)$ because a SWAP gate has lower quantum cost than a Fredkin gate. However, if we use a $\bar{F}(c; a, b)$ at this stage, we get a circuit which is presented in Figure 4. The use of F $\bar{F}(c; a, b)$ gate reduces the quantum cost from 16 to 13 as we compared with the circuit in Figure 3. Moreover, one less gate is needed in this circuit realization.The circuit in Figure 5 is even more simplified design for the reversible function from Table 1. Figure 5 shows that the gate count is 2 and the quantum cost is 10. Now if we compare the gate count and quantum cost of Figure 5 with that of the NCT gate based basic transformation synthesis (Figure 2), the gate count has been reduced from 12 to 2, a 6 times reduction. The quantum cost has been reduced from 28 to 10, which is an improvement of almost a factor of 3.

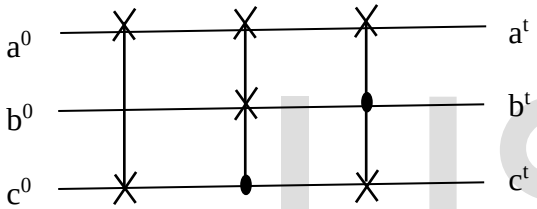


Figure 4: Another circuit of the function in Table 1

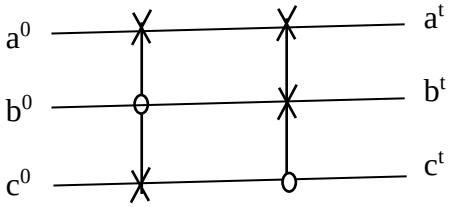


Figure 5: More efficient circuit for function in Table 1

As mentioned above, the proposed transformation algorithm using the SF gate family follows the greedy approach. We have designed our algorithm in this way in order to offer a fair comparison, since the basic transformation based synthesis algorithm which is proposed in [8] also follows the greedy approach. At every step of transformation, the algorithm selects a gate which costs less in terms of quantum cost. For example, if we observe column (ii) of Table 2, we need to transform 100 into 010. There are two choices for this mapping. We could use either a SWAP gate $S(a, b)$ or a negative controlled Fredkin gate, $\bar{F}(c; a, b)$. The proposed SF gate based transformation selects a SWAP gate, $S(a, b)$ because a SWAP gate has lower quantum cost than a Fredkin gate. However, if we use a $\bar{F}(c; a, b)$ at this stage, we get a circuit which is presented in Figure 4. The use of F $\bar{F}(c; a, b)$ gate reduces the quantum cost from 16 to 13 as we compared with the circuit in Figure 3. Moreover, one less gate is needed in this circuit realization.The circuit in Figure 5 is even more simplified design for the reversible function from Table 1. Figure 5 shows that the gate count is 2 and the quantum cost is 10. Now if we compare the gate count and quantum cost of Figure 5 with that of the NCT gate based basic transformation synthesis (Figure 2), the gate count has been reduced from 12 to 2, a 6 times reduction. The quantum cost has been reduced from 28 to 10, which is an improvement of almost a factor of 3.

We have also generated all possible 414720 conservative (4×4) reversible function. However unlike the case of (4×4) functions, there are some circuit realizations where the gate count and quantum cost increase when using SF gate based transformation synthesis. Among all the (4×4) conservative reversible functions, the quantum cost increases for 27213 (6.5%) functions and the gate count increases for 2 functions. The highest percentage of reduction in gate count by using our proposed synthesis algorithm is 87% and the reduction in gate count, on average, is 61%. We achieve the highest percentage of reduction of quantum cost is 87%. The average percentage of decrease of quantum cost over all 414720 functions is 35%.

6 Comparison of SF Based Approach With Exact Synthesis Approach

We have also compared the results from our proposed SF-based approach with the results from applying an exact synthesis approach [10] available in RevKit [11]. The comparison is shown in Table 3. The exact approach results in a circuit implementation with minimal gate count using the NCT gate library. There is no known exact approach that uses the SF gate library. There is not a single instance where the exact synthesis generates circuits with lower GC than the SF based transformation approach. The highest percentage of reduction in GC is 67%. However, the average percentage of reduction of GC using the SF based synthesis is 54%. The negative values in the table indicate the increment in QC using SF based synthesis over exact synthesis. The QC increases in the case of 10 functions out of all 36 conservative functions. The highest percentage of reduction in QC using our proposed synthesis approach is 67% and the percentage of reduction in QC on average is 8%. The reduction in GC can be explained by the fact that SF gates require fewer operations to implement swaps, which are the main operations carried out in conservative functions. However in general, SF gates have higher QC than their NCT equivalents, so there is less saving in QC. For example, a SWAP gate $S(b, c)$ can be used in order to transform (a, b, c) = 010 into 001. The GC and the QC for a single SWAP gate are 1 and 3 respectively. However, two NCT gates, $T(b; c)$ and $T(c; b)$, will be required in order to transform (a, b, c) = 010 into 001. In this case GC and QC are 2. Thus the GC is reduced by using the SF gate family, while the QC (for this example) is not.

Table 3: Performance comparison of the minimal circuits

generated using exact synthesis with the circuits generated using SF gate based synthesis.

	Exact synthesis		SF based synthesis		Reduction		Percentage of decrease	
No	GC	QC	GC	QC	GC	QC	GC	QC
1	6	14	3	13	3	1	50.00	7.14
2	6	10	2	8	4	2	66.67	20.00
3	6	14	3	13	3	1	50.00	7.14
4	6	10	2	8	4	2	66.67	20.00
5	3	3	1	3	2	0	66.67	0.00
6	4	8	2	8	2	0	50.00	0.00
7	6	10	3	11	3	-1	50.00	-10.00
8	7	15	4	16	3	-1	42.86	-6.67
9	6	10	3	11	3	-1	50.00	-10.00
10	6	6	2	6	4	0	66.67	0.00
11	6	10	3	11	3	-1	50.00	-10.00
12	7	15	4	16	3	-1	42.86	-6.67
13	6	14	3	13	3	1	50.00	7.14
14	6	10	2	8	4	2	66.67	20.00
15	3	3	1	3	2	0	66.67	0.00
16	6	10	2	8	4	2	66.67	20.00
17	6	14	3	13	3	1	50.00	7.14
18	4	8	2	8	2	0	50.00	0.00
19	6	10	3	11	3	-1	50.00	-10.00
20	6	6	2	6	4	0	66.67	0.00
21	6	10	3	11	3	-1	50.00	-10.00
22	7	15	4	16	3	-1	42.86	-6.67
23	6	10	3	11	3	-1	50.00	-10.00
24	7	15	4	16	3	-1	42.86	-6.67
25	3	3	1	3	2	0	66.67	0.00
26	6	10	2	8	4	2	66.67	20.00
27	6	14	3	13	3	1	50.00	7.14
28	6	10	2	8	4	2	66.67	20.00
29	6	14	3	13	3	1	50.00	7.14
30	4	8	2	8	2	0	50.00	0.00
31	3	15	1	5	2	10	66.67	66.67
32	4	20	2	10	2	10	50.00	50.00
33	3	7	1	5	2	2	66.67	28.57
34	4	20	2	10	2	10	50.00	50.00
35	3	7	1	5	2	2	66.67	28.57
36	0	0	0	0	0	0	0	0

7 Conclustion and Future Works

Transformation based synthesis offers function realization without including any additional garbage lines to circuits. In this paper we have presented a transformation based synthesis approach based on SF gates to realize conservative reversible functions. We have generated all possible 3-bit and 4-bit reversible functions and realized these functions with both our proposed approach and the approach proposed in [8]. The approach presented in [8] is based on the NCT gate families. Our experimental results suggest that realization of conservative functions with SF gates is more efficient than NCT gates in terms of GC and QC. We have also compared the circuits generated using exact synthesis with SF based synthesis for implementing 3-bit conservative functions. Experimental results show that SF based synthesis generates significantly more efficient circuits than exact synthesis when comparing gate count, although slightly less so when comparing quantum cost. This is likely due to the high quantum costs of the SF gate family. Our proposed SF based synthesis follows the principle of the NCT transformation based synthesis presented in [8]. A NCT transformation based synthesis approach works by mapping a reversible function into an identity function. During the process of transformation the operations performed at each stage must not affect the previous stages. One or more logic gates are applied to perform the logical operations at each stage. We have shown in Section 5 that the choice of gates at each stage is very important in order to achieve a simplified circuit.

The outcome of this work indicates that the synthesis process in reversible logic could be more efficient if we know the class of a reversible function in advance. Therefore, classifying reversible functions and using the benefitts of SF-gates in circuit realization for different classes of functions will be an important area of future study. In addition, improving the gate selection process during each stage of the transformation based synthesis is an another important area of further research. Lastly, generating minimal circuits using SF based exact synthesis is an open area of further research.

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] Mehdi Saeedi and Igor L Markov. "Synthesis and optimization of reversible circuits a survey". *ACM Computing Surveys (CSUR)*, vol.45(2) PP. 21, 2013.

[5] 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.

[6] 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.

[7] 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.

[8] 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*. ACM, pp. 318–323. 2003.

[9] D Michael Miller, Robert Wille, and Gerhard W Dueck. "Synthesizing

reversible circuits for irreversible functions". *In 12th IEEE Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD'09*. pp 749–756, 2009.

[10] Daniel Große, Robert Wille, Gerhard W Dueck, and Rolf Drechsler. "Exact multiple-control toffoli network synthesis with sat techniques". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.28(5). pp. 703–715, 2009.

[11] 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.