Controlled and Uncontrolled SWAP Gates in Reversible Logic Synthesis

Md Asif Nashiry[1], Mozammel H A Khan[2], and Jacqueline E. Rice[1]

[1] Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, AB T1K 3M4, Canada
{asif.nashiry,j.rice}@uleth.ca

[2] Department of Computer Science and Engineering
East West University
Aftabnagar, Dhaka 1212, Bangladesh
mhakhan@ewubd.edu

Abstract. There is a very little work in the literature that deals with reversible logic synthesis using only SWAP and Fredkin (SF) gates. This paper presents several applications using these gates, as well as a quantum-level realization and a synthesis approach. The positive-controlled Fredkin gate is essential for synthesizing falling edge-triggered and asynchronous loadable reversible sequential circuits. Conversely, negative-controlled Fredkin gates are required if the circuit is designed to be rising edge-triggered. Our quantum realization of negative-controlled Fredkin gate requires five 2-qubit elementary quantum gates, the same as that required for realizing a positive-controlled Fredkin gate. We also propose and evaluate the performance of a synthesis approach using SF gates for realizing conservative reversible functions. Our results shows that circuit realization for conservative function using SF gates is more efficient than Toffoli gates.

Keywords: reversible logic, SWAP gate, Fredkin gate, Toffoli gates, multiple control gates, mixed polarity gates, quantum gates, logic synthesis

1 Introduction

Reversible circuits have drawn the attention of researchers for their many advantages over traditional irreversible circuits. A reversible function is a bijective function. A logic gate is a reversible gate if the output function of the gate is bijective [1]. The Two most widely used reversible logic gate families are NOT-CNOT-Toffoli (NCT) and SWAP-Fredkin (SF). A SWAP gate is a (2×2) reversible logic gate which interchanges the input bits at the output. Fredkin and Toffoli proposed a reversible controlled swap gate (also called Fredkin gate) in [2]. This gate is a positive-controlled gate i.e., it swaps the two target inputs when the control input is 1. The authors showed that it is a universal gate and any reversible circuit can be synthesized using only Fredkin gates. Smolin and DiVincenzo presented an implementation of the positive-controlled Fredkin gate using five 2-qubit elementary quantum gates in [3]. Although the positive-controlled Fredkin gate is a universal gate, little work has focused on synthesizing reversible circuits using only Fredkin gates. So far as we know, only Bruce et al. has proposed a design for a full-adder using five positive-controlled Fredkin gates ~~in~~ [4].

Positive-controlled Fredkin gates are essential for designing reversible sequantial circuits [5–7]. Positive-controlled Fredkin gates can be used to make the sequential circuit falling edge-triggered and asynchronous loadable. For example, Fig. 1 shows the reversible realization of a 2-bit falling edge-triggered up counter with asynchronous load using the design method of [7]. The feedback section of the circuit provides the feedback of the state outputs $Q1$ and $Q0$. The next state logic section generates the next states. When the clock is $C = 1$, the target inputs of the two positive-controlled Fredkin gates of the falling edge trigered section are swapped and the fed-back state values are passed to the state outputs. When C goes to 0 (falling edge), the target inputs are not swapped and the generated next states are passed to the state outputs. On the other hand, when the load control value is $L = 0$, the target inputs of the two positive-controlled Fredkin gates of the asynchronous load section are not swapped and the generated next states or fed-back states are passed to the state outputs. When L is 1, the target inputs of the positive-controlled Fredkin gates are swapped and the asynchronous load data $D1$ and $D0$ are loaded to the state outputs $Q1$ and $Q0$, respectively. If the reversible sequential circuit is required to be rising edge-triggered, then the positive-controlled Fredkin gates of the edge trigger section of the circuit must be replaced by a negative-controlled Fredkin gate. In this case, the swapping is done when the control input is 0. Similarly, if the asynchronous load is required to be done when L is 0, then the positive-controlled Fredkin gates of asynchronous load section of the circuit must be replaced by negative-controlled Fredkin gates.

A positive-controlled Fredkin gate can be realized using five 2-qubit elementary quantum gates [3]. We propose a realization of the negative-controlled Fredkin gate, that ~~which is~~ like the positive-controlled Fredkin gate, requires five 2-qubit elementary quantum gates. This offers a unique addition to the literature, as there are currently no other proposals for the design of negative-controlled Fredkin gates. We demonstrate ~~have shown~~ the use of negative-controlled Fredkin gate in the design of rising edge-triggered reversible sequential circuit with asynchronous load when the load input value is 0.

In this paper~~,~~ we ~~have~~ also propose~~d~~ a transformation based synthesis algorithm using SF gates for realizing ~~in order to realize the~~ conservative reversible functions. ~~Note that~~ A ~~a~~ conservative reversible function is one where the number of 1s on the ~~in~~ inputs is equal to the number of 1s at the ~~corresponding~~ outputs. We compared our proposed approach with the basic transformation approach ~~which is~~ proposed in [10] ~~and the comparative result shows that~~ Our proposed algorithm performs better ^ from the perspective of gate count and quantum cost.

for three and four input conservative reversible functions.

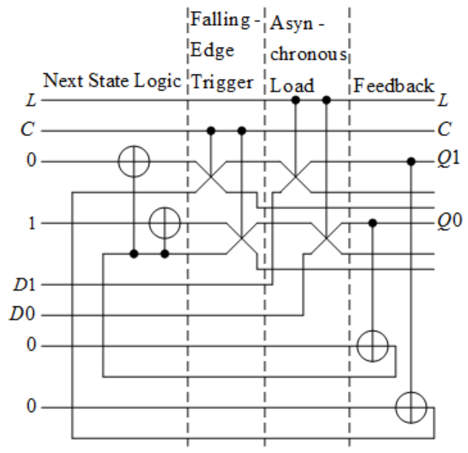

Fig. 1. Reversible realization of 2-bit falling edge-triggered up counter with asynchronous load.

The organization of the paper is as follows. In section 2, we present three 2-qubit elementary quantum gates needed to realize the negative-controlled Fredkin gate. We illustrate ~~also~~ a realization of a Toffoli gate with top negative control and bottom positive control, which is used as an intermediate gate for realizing the negative-controlled Fredkin gate in section 3. In section 4, we present the realization of the negative-controlled Fredkin gate. We present our proposed SF based synthesis approach with performance evaluation of this approach in section 6. Finally, we conclude the paper in section 7.

2 ~~Some~~ 2-Qubit Elementary Quantum Gates

In this section we discuss ~~some~~ the 2-qubit elementary quantum gates which ~~we~~ are used in our ~~incorporate in our~~ proposed realization of a negative-controlled Fredkin gate.

2.1 Feynman Gate

The 2-qubit Feynman gate, ~~also known as~~ or CNOT gate is shown in Fig. 2. The input x is the control input and passe~~d~~s unchanged to the output. The input t is the target input. When the control input is $x = 1$, the target input t is complemented at the output. When $x = 1$, the target input is passed unchanged to the output. The target output is expressed as $r = x \oplus t$.

2.2 Square-Root-of-NOT Gates

Fig. 3(a) shows the controlled-V (or CV) gate. The input x is the control input and the input y is the target input. The unitary transform shown in Equ. 1

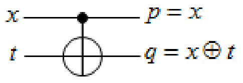

Fig. 2. Feynman gate.

$$V = \begin{bmatrix} (1+i)/2 & (1-i)/2 \\ (1-i)/2 & (1+i)/2 \end{bmatrix} \tag{1}$$

is applied on the target input y when the control input is $x = 1$. When $x = 0$ the target input y is passed unchanged to the output. The CV gate is called the *square-root-of-NOT* gate, since

$$V \times V = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

which is the unitary transform of the NOT gate. Therefore, the cascaded application of two CV gates acts as a Feynman gate. Fig. 3(b) shows the controlled-V$^+$ (or CV$^+$) gate. The unitary transform ~~which is~~ shown in Equ. 3

$$V^+ = \begin{bmatrix} (1-i)/2 & (1+i)/2 \\ (1+i)/2 & (1-i)/2 \end{bmatrix} \tag{3}$$

is applied on the target input y when the control input is $x = 1$. For $x = 0$ the target input y is passed unchanged to the output. The CV$^+$ gate is also called the *square-root-of-NOT* gate, since

$$V^+ \times V^+ = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

A ~~C~~ascade application of two CV$^+$ gates acts as a Feynman gate. The CV and the CV$^+$ gates are the inverse of each other, since $V \times V^+ = V^+ \times V = I$. Therefore, the cascaded application of a CV and a CV$^+$ gates acts as a 2-qubit identity gate which restores the control and the target input~~s~~ at the outputs.

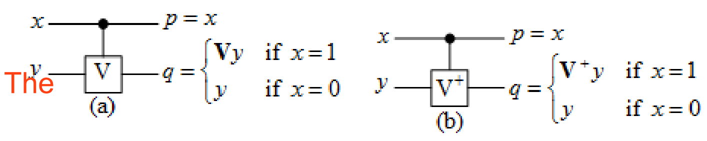

Fig. 3. 2-qubit square-root-of-NOT gates.

3 Realization of Toffoli Gate with Top Negative Control and Bottom Positive Control

In the realization of our proposed negative-controlled Fredkin gate, we use a Toffoli gate with top negative control and bottom positive control as an intermediate gate. ~~In this section we discuss the realization of a top negative control and bottom positive control (3×3) Toffoli gate using 2-qubit elementary quantum gates.~~

The symbol of a Toffoli gate with top negative control and bottom positive control is shown in Fig. 4(a). The inputs a and b are the control inputs and the input c is the target input. When the control inputs are $a = 0$ and $b = 1$, the target input c is complemented at the output, otherwise the target input c is passed unchanged to the output. The target output is expressed as $r = c \oplus a'b$. The realization of a Toffoli gate with two positive controls using five 2-qubit elementary quantum gates is presented in [8]. Realization of a Toffoli gate with top positive control and bottom negation control using a similar technique is presented in [9]. This realization also requires five 2-qubit elementary quantum gates. We follow this technique and present a realization of a Toffoli gate with top negative control and bottom positive control in Fig. 4(b). This realization also requires five 2-qubit elementary quantum gates. The behaviour of the circuit shown in Fig. 4(b) is discussed for all possible four inputs of a and b. When $a = 0$ and $b = 0$, then $x = 0$ and the two V gates and the V^+ gate will be inactive and the target input c will be passed unchanged to the target output r. When $a = 0$ and $b = 1$, then $x = 1$ and the two V gates will be active and the V^+ gate will be inactive. Thus the target input c will be complemented at the target output r, since application of two V gates acts as a NOT gate. When $a = 1$ and $b = 0$, then $x = 1$ and the first V gate will be inactive. However, the V^+ gate and the second V gate will be active. Thus the target input c will be passed unchanged to the target output r, since cascaded application of V and V^+ gates act as an identity gate. Finally, when $a = 1$ and $b = 1$, then x becomes 0. The first V gate and the V^+ gate will be active and the second V gate will be inactive. Thus the target input c will be passed unchanged to the target output r.

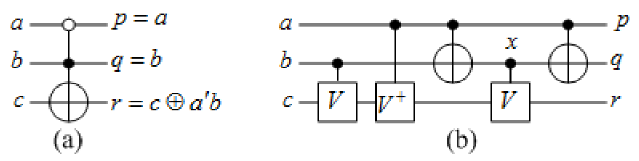

Fig. 4. (a) Toffoli gate with top negative control and bottom positive control and (b) its realization using 2-qubit elementary quantum gates.

4 Realization of Negative-Controlled Fredkin Gate

The symbol of the negative-controlled Fredkin gate is shown in Fig. 5 (a). The input a is the control input and the inputs b and c are the target inputs. When the control input $a = 0$, the two target inputs are swapped at the target outputs. For $a = 1$, the target inputs pass to the target outputs without interchanging the values. The two target outputs are expressed as $q = ab \oplus a'c$ and $r = a'b \oplus ac$. The realization of the negative-controlled Fredkin gate using one top negative control and bottom positive control Toffoli gate and two Feynman gates is shown in Fig. 5 (b). In Fig. 5 (b), $x = b \oplus c$. The two target outputs are as follows:

$$\begin{aligned} r &= c \oplus a'x \\ &= c \oplus a'(b \oplus c) \\ &= c \oplus a'b \oplus a'c \\ &= a'b \oplus c(1 \oplus a') \\ &= a'b \oplus ac \end{aligned}$$

$$\begin{aligned} q &= x \oplus r \\ &= b \oplus c \oplus a'b \oplus ac \\ &= b(1 \oplus a') \oplus c(1 \oplus a) \\ &= ab \oplus a'c \end{aligned}$$

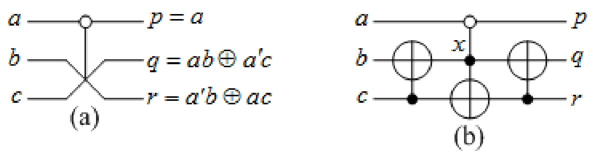

Fig. 5. (a) Negative-controlled Fredkin gate and (b) its realization with two Feynman gates and one Toffoli gate with top negative control and bottom positive control.

If the Toffoli gate shown in Fig. 5 (b) is decomposed using the realization illustrated in Fig. 4(b), then the ~~realization of a~~ negative-controlled Fredkin gate can be realized ~~done~~ as shown in Fig. 6. The two circuits of Fig. 7 are equivalent. Using this equivalency, the circuit from Fig. 6 can be rearranged as in Fig. 8 by commuting the last two Feynman gates. The two 2-qubit quantum gates in two dashed boxes act on the same lines. The operation of the first gate can be expressed using a 4×4 unitary matrix. Similarly, the operation of the second gate can also be expressed by another 4×4 unitary matrix. As these two gates are in cascade, their final operation can be described by the matrix multiplication of the two 4×4 unitary matrices, which will be another 4×4 unitary matrix representing an elementary 2-qubit quantum gate. Therefore, the two gates in the dashed boxes ~~are~~ in practice work as one 2-qubit quantum gate. Thus, the realization of the negative-controlled Fredkin gate requires five 2-qubit elementary quantum gates. A similar argument is used in [3].

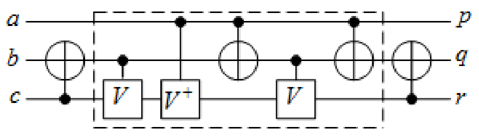

Fig. 6. Realization of negative-controlled Fredkin gate by decomposing the Toffoli gate from Fig. 5 (b) using the realization of Fig. 4(b).

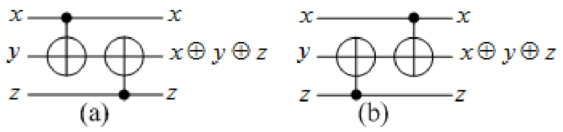

Fig. 7. Circuits of (a) and (b) are equivalent.

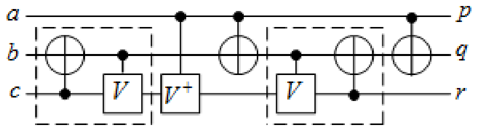

Fig. 8. Rearranged circuit of Fig. 6.

5 Negative-Controlled Fredkin Gates in Reversible Sequential Circuit Design

The reversible circuit for a 2-bit up counter as shown in Fig. 1 can be made rising edge-triggered and asynchronours loadable when the load input value is 0 using negative-controlled Fredkin gates as shown in Fig. 9. When the clock input $C = 0$, the target inputs of the two negative-controlled Fredkin gates of the rising edge trigger section are swapped and the state feedback is passed to the status outputs $Q1$ and $Q0$. When C goes from 0 to 1 (rising edge), then the two target outputs of the negative-controlled Fredkin gates are not swapped and the generated next states are passed to the state outputs $Q1$ and $Q0$. When the load control value $L = 1$, then the target inputs of the two negative-controlled Fredkin gates of the asynchronous load section are not swapped and the generated or fed-back states are passed to the state outputs $Q1$ and $Q0$. When L is set to 0, the target outputs of the negative-controlled Fredkin gates are swapped and the asynchronous load data $D1$ and $D0$ are loaded to the state outputs $Q1$ and $Q0$ respectively.

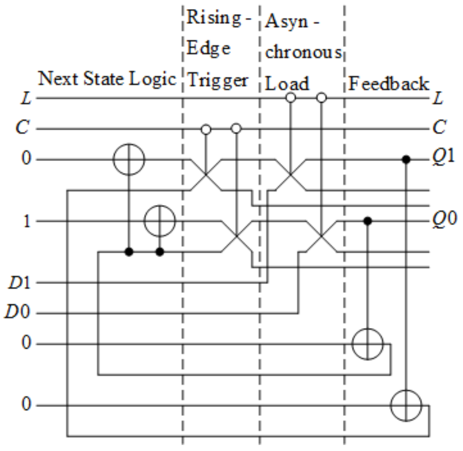

Fig. 9. Reversible realization of 2-bit rising edge-triggered up counter with asynchronous load value ~~is~~ 0.

6 ~~Proposed~~ SF Based Synthesis Approach

6.1 Logic Synthesis in Reversible Logic

Logic synthesis ~~means~~ is the process of transforming a logic function into a corresponding logic circuit. During the process of transformation, the relationships between the inputs and the outputs of a logic function determine the number, the type and the order in which the logic gates should appear in the circuit. If a logic function is already reversible, then the synthesis process can take place immediately in order to transform the function into a circuit design. 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. In most cases, a reversible circuit design with lower garbage outputs and/or constant inputs is considered a desirable design. There are a number of logic synthesis techniques in reversible logic [11] and a transformation based ~~logic synthesis is~~ approach has been one of the most popular ~~one~~ techniques. One of the major advantages of a transformation based synthesis algorithm is that a circuit realization based on this algorithm does not create any garbage outputs or ~~and~~ constant input lines. Thus in terms of input-output lines, the size of the circuit which is realized using a transformation based approach ~~synthesis~~ is minimal ~~minimum~~.

6.2 Transformation Based Synthesis ~~Algorithm~~

The input to a ~~of the~~ transformation based synthesis algorithm is a truth table of a reversible function. ~~Thus if a function is not reversible, the first step is to transform the irreversible function into a reversible function.~~ The basic working principle of the transformation based synthesis algorithm is to apply reversible operations to a reversible function in order to generate an identity function. The first such ~~transformation based synthesis~~ algorithm was proposed by Miller et al. [10]. They demonstrated two variations: a basic algorithm and a bidirectional algorithm. The authors used the NCT gate library in circuit realization. In the basic algorithm, the reversible logic operations are applied to the output of the function's truth table. We assume that we are applying the algorithm to a reversible function of n variables. The objective is to make $f(i) = i$, for $i = 0$ to $2^m - 1$. For example for $n = 3$ after applying the transformation algorithm the output function $f(3)$ should be 011. The following is the basis of transformation based logic synthesis approach.

Step 0: If $f(0) = 0$, no transformation is required; ~~and~~ go to step 2. Otherwise, if $f(0) \neq 0$, apply a ($1x1$) Toffoli gate (NOT gate) in order to achieve $f(0) = 0$. After applying a NOT gate, the bit combination 000 will be at the top row of the output truth table.

Step 1: For $1 \leqslant i < 2^m - 1$: 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 $f^i(i) = i$. One or more gates require in order to achieve $f^i(i) = i$.

The choice of ~~a~~ gate during each step of transformation is crucial in order to maintain convergence. The gate chosen in each ~~in one~~ step of transformation must not change the order of bits of the previous steps. Consider the following (3×3) reversible function $f = \sum(0, 4, 1, 3, 2, 5, 6, 7)$ in Table 1. The transformation based synthesis transforms the function to an identity function, $f^t = \sum(0, 1, 2, 3, 4, 5, 6, 7)$.

The circuit which is generated by following the basic transformation algorithm is presented in Fig. ~~figure~~ 10. The entire ~~whole~~ transformation process requires 8 CNOT gates and 4 Toffoli gates. Thus the gate count is 12. The quantum cost is $((8 \times 1) + (4 \times 5)) = 28$. The next subsection describes our proposed algorithm. We also show the circuit realization for the same function using our proposed algorithm.

6.3 ~~Our Proposed~~ SF-based Synthesis Approach

Before describing our proposed approach, it is important to observe the conservative property of the function which is shown in Table 1. The truth table of the function shows that the number of 1xs in each input bit combination ~~and the bit combination in the corresponding output is equal.~~ Thus the function is a conservative function.

is equal to the number of 1s in the outputs.

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)

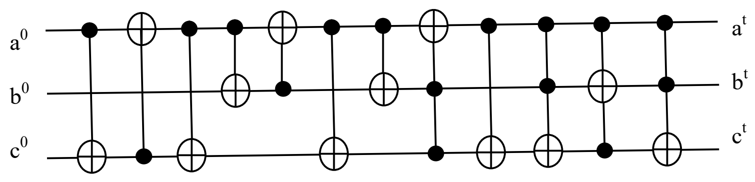

Fig. 10. The ~~equivalent~~ circuit implementing the function in Table 1, ~~of function in table 1~~, generated using transformation based synthesis

~~We hypothesize that a circuit realization for a conservative reversible function would be more efficient if we use a SF-based transformation synthesis instead of a NCT-based synthesis approach.~~ The basic idea of ~~the~~ SF-based transformation synthesis is the same as the approach ~~which~~ ~~was~~ presented in [10]. 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. The algorithm examines one row of the truth table at each step to verify whether the bit combination of the row at output is equal to the corresponding bit combination of the row at input. If $f(i) = i$, no transformation is required. However at step i, if $f(i) \neq i$ the algorithm follows a greedy approach to generate a sequence of one or more reversible logic gates that realize the required transformation. We use the same function from Table 1 to demonstrate the SF-based transformation synthesis. We also use the simple one direction transformation for this example. As before, the algorithm begins with the output of the function. Table 2 shows the transformation stages.

Step 0: The input output relation for the first row of the function (Table 1) is $f(0) = 0$, which satisfies the required input-output relation. So we do not need any transformation.

Step 1: In the second row of the table we have, $f(4) \longrightarrow 1$. We need to map 100 to 001. A single SWAP gate, S(a,c) is enough for this mapping. A SWAP gate, S(a,c) interchanges the bits of a and c at output.

Step 2: After using SWAP gate at the 1st Step, we have 100 in the third row. We need to transform the bits into 010. A SWAP gate, S(a,b) does the required transformation at this stage.

Step 3: At this stage of transformation, we need to map $f(6) \longrightarrow 3$. A SWAP gate, S(a,c) could transform 110 to 011, however this mapping would also change the bits in the previous rows. One of the basic concepts of the transformation based synthesis is that a mapping in one stage must not alter any bit in any of the previous rows. Hence, we need a (3×3) Fredkin gate for the required mapping at this stage. We use a Fredkin gate F(b;a,c), which swaps the two target bits of a and c, when the control bit of the gate b = 1.

Step 4: At this step, the bit combination in fifth row is already in proper position. We need to transform 110 into 101. A (3×3) Fredkin gate F(a;b,c) swaps the values of b and c when the value of 'a' is 1. By applying the Fredkin gate F(a;b,c) at this step the rest of the entries of the truth table are organized properly. The resulting circuit realization of the function from Table 1 is displayed in Figure 11.

Table 2. Transformation stages of the function in table 1 using SF based Transformation

			step 0			step 1			step 2			step 3			step 4		
output			(i)			(ii)			(iii)			(iv)			(v)		
a	b	c	a^0	b^0	c^0	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
			No Transformation			S(a,c)			S(a,b)			F(b;a,c)			F(a;b,c)		

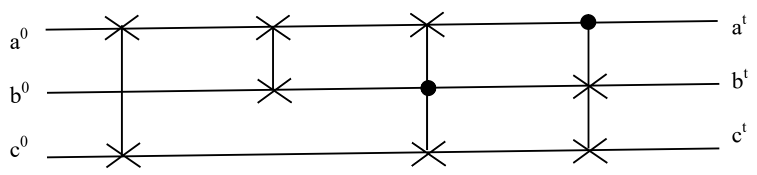

Fig. 11. SF based synthesis result for the function from Table 1

6.4 Comparison of NCT and SF-based Synthesis Approaches

~~The~~ Figures 10 and 11 show two circuits for the same function from Table 1. The former circuit was realized following the basic transformation based algorithm from [10], while the latter circuit realization is generated by the SF-based transformation algorithm. In Figure 11, we have a gate count of 4 as compared to a gate count of 12 for the circuit in Figure 10. Moreover in terms of quantum cost analysis, the SF gate based transformation synthesis has shown much better result as compared with the basic transformation based synthesis using NCT gate family. It is seen that the quantum cost of the implementation is $(2 \times 3) + (2 \times 5) = 16$, where the quantum cost for the circuit realization in Figure 10 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 ~~gate~~ based transformation approach with NCT ~~gate~~ based transformation ~~approach~~ from a wider perspective, we have generated all possible (3×3) conservative reversible functions. There are 36 conservative (3×3) reversible functions in total. We have realized all the 36 conservative (3×3) reversible functions using both basic transformation based synthesis algorithm and our proposed SF-based algorithm. Performance was evaluated in terms of gate count and quantum cost. Table 3 shows the results. The first row of the table shows that the circuit realization using SF-based synthesis requires 6 ~~less~~ fewer gates compared to the basic approach. Moreover, the quantum cost reduces from 25 to 13. The percentages of reduction in gate count and quantum cost are 67% and 48% respectively. After observing the entire table, the highest percentage of reduction in gate count is 67%. We achieve the highest percentage of reduction in gate count for more than half of the (3×3) conservative reversible functions.

As we see from Table 3, the SF-based synthesis approach performs extremely well as compared to the other approach as far as gate count is concerned. One of the reasons behind the performance improvement is that the nature of operation of a SWAP gate or a controlled SWAP gate (Fredkin gate) allows the gate to adapt well with the property of a conservative function. Since a function is conservative, we need to transform a bit combination into another bit combination with the same number of 1x's and 0x's. Most of this type of transformation can be done with fewer operations if we exchange the bits of a row instead of inverting the bits. The ability of changing two bits at a time gives SF gates an advantage ~~benefit~~ over the NCT gate family for realizing conservative reversible circuits.

From the perspective of quantum cost, the performance of SF based synthesis is also better as compared to the NCT based synthesis. 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. So there is not a single instance where the NCT based synthesis performs better than our proposed approach. By using SF based synthesis over the NCT based synthesis, the highest percentage of decrease in quantum cost is 70%. However, the average percentage of reduction of quantum cost using the SF based synthesis is 29% as compared to the NCT based synthesis. The last row of Table 3 shows 0 in all columns. Because the input-output relationship $f(i) = i$ holds for all the bit combination of the particular function i.e. the function is an identity function. So no transformation is necessary for the function.

Table 3. Performance comparison of basic transformation based synthesis and SF gate transformation based synthesis

Basic Transformation		SF gate transformation		Reduction		Percentage of Decrease	
GC	QC	GC	QC	GC	QC	GC	QC
9	25	3	13	6	12	66.67	48.00
6	10	2	8	4	2	66.67	20.00
9	25	3	13	6	12	66.67	48.00
6	10	2	8	4	2	66.67	20.00
3	3	1	3	2	0	66.67	0.00
6	18	2	8	4	10	66.67	55.56
7	11	3	11	4	0	57.14	0.00
10	26	4	16	6	10	60.00	38.46
9	21	3	11	6	10	66.67	47.62
6	6	2	6	4	0	66.67	0.00
7	11	3	11	4	0	57.14	0.00
10	26	4	16	6	10	60.00	38.46
9	25	3	13	6	12	66.67	48.00
6	10	2	8	4	2	66.67	20.00
3	3	1	3	2	0	66.67	0.00
6	18	2	8	4	10	66.67	55.56
7	23	3	13	4	10	57.14	43.48
4	8	2	8	2	0	50.00	0.00
9	21	3	11	6	10	66.67	47.62
6	6	2	6	4	0	66.67	0.00
7	11	3	11	4	0	57.14	0.00
10	26	4	16	6	10	60.00	38.46
9	13	3	11	6	2	66.67	15.38
12	28	4	16	8	12	66.67	42.86
3	3	1	3	2	0	66.67	0.00
6	18	2	8	4	10	66.67	55.56
9	25	3	13	6	12	66.67	48.00
6	10	2	8	4	2	66.67	20.00
7	23	3	13	4	10	57.14	43.48
4	8	2	8	2	0	50.00	0.00
3	7	1	5	2	2	66.67	28.57
6	22	2	10	4	12	66.67	54.55
3	7	1	5	2	2	66.67	28.57
6	22	2	10	4	12	66.67	54.55
3	15	1	5	2	10	66.67	66.67
0	0	0	0	0	0	0	0

As mentioned above, the proposed transformation algorithm using the SF gate family follows the greedy approach. We have designed our algorithm in this way, because the basic transformation based synthesis algorithm which is proposed in [10] also follows the greedy approach. Thus in order to make a fair comparison, we have chosen the greedy approach. At every step of transformation, the algorithm selects a gate which costs less in terms of quantum cost. When there is a choice between a SWAP gate and a controlled SWAP gate in order to make a transformation happen, the algorithm selects a SWAP gate, since a SWAP gate has a minimum quantum cost than a Fredkin gate. For example, if we observe the 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, $F^{'}(a,b;c)$. A SWAP gate S(a,b) exchanges the bits of a and b. A negative controlled Fredkin gate, $F^{'}(a,b;c)$ swaps the values of a and b when c is 0. Any two of the gates can sever the purpose at this stage. However, the proposed SF gate based transformation selects a SWAP gate, S(a,b) in this case. Because a SWAP gate has lower quantum cost than a Fredkin gate. However, if we use a $F^{'}(a,b;c)$ at this stage, we get a circuit which is presented in Figure 12. The use of $F^{'}(a,b;c)$ gate reduces the quantum cost from 16 to 13 as we compared with the circuit in Figure 11. Moreover, one less gate is needed in this circuit realization. The interesting fact is that the circuit in Figure 12 is not even optimum. The choice of gate is one of the crucial factors in order to design an optimum circuit. The circuit which is represented in Figure 13 is further optimized circuit representation for the reversible function in consideration. The Figure 13 shows that the gate count is 2 and the quantum cost is 10. Now if we compare the gate count and quantum cost of the Figure 13 with that of the NCT gate based basic transformation synthesis (Figure 10), the gate count has been reduced to 12 to 2, which is 6 times reduction. The quantum cost has been reduced from 28 to 10, which is an improvement of almost a factor of 3 ~~almost 3 times of improvement~~ in quantum cost of a circuit.

In a nutshell, the SF-based transformation approach performs much better than the NCT based transformation for all the (3×3) conservative reversible functions. The performance of the SF gate based transformation can be improved further if the selection of gate at each stage can be done intelligently rather than following a greedy ~~way~~. approach

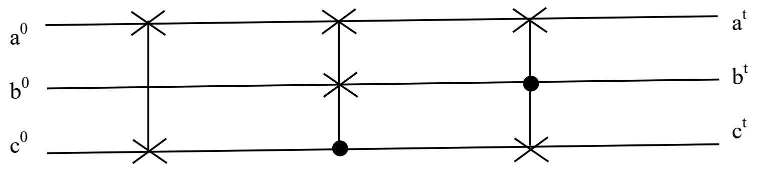

Fig. 12. Another circuit realization for the function from T table 1

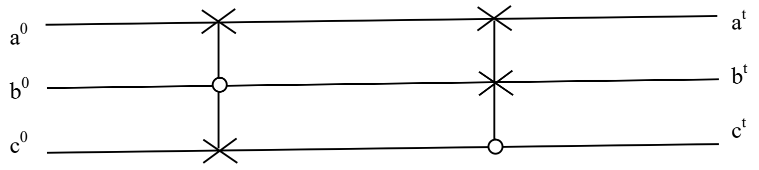

Fig. 13. More efficient circuit realization for the function from ~~truth~~ T table 1

We have also generated all the possible (4×4) conservative reversible function. The are 414720 conservative (4×4) reversible functions in total. We have investigated the circuit realization for each of the function using SF based synthesis and NCT based synthesis. However unlike the case of (3×3) functions, there are some circuit realization where the gate count and quantum cost increase by using SF gate bases transformation synthesis as compared to that of NCT gate based synthesis. Among all the 414720 (4×4) conservative reversible functions, the quantum cost increases in case of 27213 functions and the gate count increase for only 2 functions. We have found that the gate count has been increased by only 1 in those two functions. On the other hand, the highest number of reduction in gate count by using SF based transformation as compared with that of NCT based transformation is 27. The highest percentage of reduction in gate count by using our proposed synthesis algorithm is 87% and the highest reduction in gate count, on average, is 61%. By using the SF based synthesis we achieve the highest reduction in quantum cost is 104, whereas the highest percentage of reduction of quantum cost is 87%. By considering all the functions, the average percentage of decrease of quantum cost is 35%.

7 Conclusion

The contribution of this work is twofold. First, we present a unique realization of a negative-controlled Fredkin gate using five 2-qubit elementary quantum gates. This is identical to the number of gates required for the implementation of a positive-controlled Fredkin gate presented in [3]. Note that an addition of a NOT gate in the realization of a positive-controlled Fredkin gate could also be a realization of a negative-controlled Fredkin gate. However, unlike our proposal, this would increase the quantum cost of the realization. Secondly, we propose a transformation based synthesis algorithm using SF gate in order to realize conservative reversible function. A conservative function maintains parity between its inputs and outputs. This property makes a conservative function an important class of reversible fucnion in applications such as fault dection, fault testing and the desing of fault tolarent reversible circuits. We have generated all possible (3×3) and (4×4) conservative functions. We compare our approach with the NCT based synthesis which is proposed in [10]. For those functions we see that the synthesis of conservative reversible functions using SF gates is more efficient than using NCT gates based on two important performance matrices: gate count and quantum cost.

We also show that a negative-controlled Fredkin gate is very useful for designing rising edge-triggered reversible sequential circuits, while a negative-controlled Fredkin gate is also important to make a sequential circuit asynchronous loadable for load input value 0. In addition, this paper shows that the usefulness of a negative control Fredkin gate in circuit realization. The outcome of this work indicates the necessity of classifying reversible functions. The synthesis process in reversible logic would be more efficient if we know the class of a reversible function in advance. Therefore, classifying reversible functions and using the benefits of SF-gates in circuit realization for different classes of functions will be an important area of further research.

Acknowledgment The second author was involved in this work at the University of Lethbridge, Lethbridge, AB, Canada while on sabbatical from East West University, Dhaka, Bangladesh.

References

1. Shende, V. V., Prasad, A. K., Markov, I. L., Hayes, J. P.: Synthesis of reversible logic circuits. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 6, pp. 710–722, IEEE (2003)
2. Fredkin, E., Toffoli, T.: Conservative logic. Collision-based computing. pp. 47–81, Springer (2002).
3. Smolin, J. A., DiVincenzo, D. P.: Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate. Jour. Phys. Rev. A, vol. 53, no. 4, pp. 2855 (1996)
4. Bruce, J. W., Thornton, M. A., Shivakumaraiah, L., Kokate, P. S., Li, X.: Efficient adder circuits based on a conservative reversible logic gate. in Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002, pp. 83–88 (2002)
5. Thapliyal, H., Ranganathan, N., Kotiyal, S.: Design of testable reversible sequential circuits. IEEE Trans. VLSI Syst., vol. 21, no. 7, pp. 1201–1209, IEEE (2013)
6. Khan, M. H. A.: Design of reversible synchronous sequential circuits using pseudo Reed-Muller expressions, IEEE Trans. VLSI Syst., vol. 22, no. 11, pp. 2278–2286, IEEE (2014)
7. Khan, M. H. A., Rice, J. E. Improved synthesis of reversible sequential circuits, in preparation.
8. Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A., Weinfurter, H.: Elementary gates for quantum computation. jour. Phy. Rev. A, vol. 52, no. 5, pp. 3457, APS (1995)
9. Maslov, D., Miller, D. M.: Comparison of the cost metrics for reversible and quantum logic synthesis. jour. arXiv preprint quant-ph/0511008, (2005)
10. Miller, D. M., Maslov, D., Dueck, G. W.: A transformation based algorithm for reversible logic synthesis, in Proceedings on Design Automation Conference, pp. 318–323, IEEE (2003)
11. Saeedi, M., Markov, I. L.: Synthesis and optimization of reversible circuitsa survey. Jour. ACM Computing Surveys (CSUR), vol. 45, no. 2, pp. 21, ACM (2013)