J Biochem Tech (2018) Special Issue (2): 50-60
ISSN: 0974-2328

Synthesis of Reversible Functions based on the Realization of Distinct Cycles

Seyed Mansour Shahidi* and Shahram Etemadi Borujeni

Received: 22 November 2017 / Received in revised form: 18 May 2018, Accepted: 23 May 2018, Published online: 05 September 2018

Abstract

One of the issues that arises today in the field of quantum circuits is how to synthesize a reversible circuit using the reversible gates. Due to the fact that the implementation of reversible circuits has high hardware costs, much effort has been made to find the circuit with the lowest cost. In this paper, we present a method that, by relying on the cycles in the truth table, produces a circuit in which these cycles are implemented, without changing in other combinations of the truth table. Therefore, for functions with low cycle sizes, they will generate much less expensive circuits than other methods. In addition, for functions with don't care combinations or don't care outputs, in this method, we have tried to reduce the number of cycles and their size, and hence the quantum cost of generated circuits, using these don't cares. The proposed method of this article has improved the average costs between 27% and 40% in comparison with similar tasks in the case of circuits with a low cycle number and size.

Keywords: Reversible Circuits, Cycle-based Approach, Don't Care Conditions, Quantm Computing.

Introduction

Irreversible gates have various drawbacks such as energy wasting as well as security weakness due to the possibility of attack through leakage power analysis. The advantage of using reversible gates is less power consumption, because classical gates that are not reversible will waste some energy in heat as a result of the loss of data per bit (Landauer, 1961). Therefore, in this section, the problem is implementing circuits using these gates and reducing the implementation cost such as quantum cost, gate number, depth of circuit, and so on. In (Saeedi et al., 2010) a method is proposed for the synthesis of reversible circuits, which, in order to reduce quantum cost and runtime, decomposes the circuit into predetermined blocks by an algorithm and perform synthesis at the level of these blocks.

Seyed Mansour Shahidi*
Faculty of Engineering, Ayatollah Borujerdi University, Borujerd, Iran.

Shahram Etemadi Borujeni
Department of Computer Architecture Engineering, University of Isfahan, Isfahan, Iran.

***Email:** m.shahidi@abru.ac.ir

In the face of attacks that leakage power analysis can be done, reversible logic could well be used (Bhagyalakshmi & Venkatesha, 2010). The reason for this is that the reversible circuits do not waste power. This paper introduces a scheme that uses standard gates instead of the complex and costly gates used in previous work, and has designed the ALU for a cryptographic processor.

The synthesis of reversible circuits is a challenge, with many work done on it. Different algorithms try to provide a more optimal implementation for each description of a reversible circuit, using reversible gates (Metodi & Chong, 2006). In (Miller, Maslov & Dueck, 2003, Maslov, Dueck & Miller, 2003, Maslov, Dueck & Miller, 2005), a method is proposed, which design the reversible circuit for the given truth table by the Toffoli gates. In this method, the truth table is scanned from top to bottom, and for each input combination, a gate is added to change the output combination to the desired value. This method ensures that the reversible circuit for the given truth table is made. But the optimality of the circuit has not been considered.

In (Shende et al., 2003), methods for the synthesis of reversible circuits are presented. One of these methods is the implementation of cycles of size 2 called transposition. In this paper, we try to convert larger cycles into multiple transpositions, if necessary, and then implement these transpositions. The proposed method in this paper tries to convert larger cycles into multiple transpositions, if necessary, and then implement these transpositions. (Sasanian et al., 2009; Saeedi et al., 2010), using the method of the previous paper and extending it, have developed a method that adds the reversible gate to the circuit based on cycles that are recognized in the truth table of a reversible circuit. As a result, the truth table is fulfilled.

In (Zakablukov, 2016), a method is proposed that converts the reversible function into distinct groups of permutations and then performs synthesis according to each group. In (Drechsler, Finder & Wille, 2011; Fazel, Thornton & Rice, 2007), the function is transformed into ESOP form and then implemented using the available reversible quantum gates such as Toffoli gate. In these methods, for simplicity of the proposed algorithms, for qubits in the circuit, instead of quantum quantities, the classical values of zero and one are taken into account.

In (Zhu et al., 2018) Based on the cycle-based method, all cycles in the truth table of a function are considered, and each cycle is defined and implemented using transposition cycles. In this method, for the implementation of an n-input function, only n-input Toffoli gates will be used. In addition, all these methods are designed in such a way that, considering and trying to implement each cycle, the status of the truth table will generally changes, which can increases the number of necessary gate to function synthesis.

Reversible circuits

Reversible circuits are made from reversible gates. These gates have special properties that are not found in most classical gates. One of these properties is that their descriptive function is one to one. Therefore, the number of inputs and outputs of these gate is equal. Also, knowing the output value of these gates, you can find their input value. Figure 1 shows a number of known reversible gates. These gates are called basic reversible quantum gates and can be used for quantum implementation of each reversible gate (Nielsen & Chuang, 2000, Maslov et al., 2008).

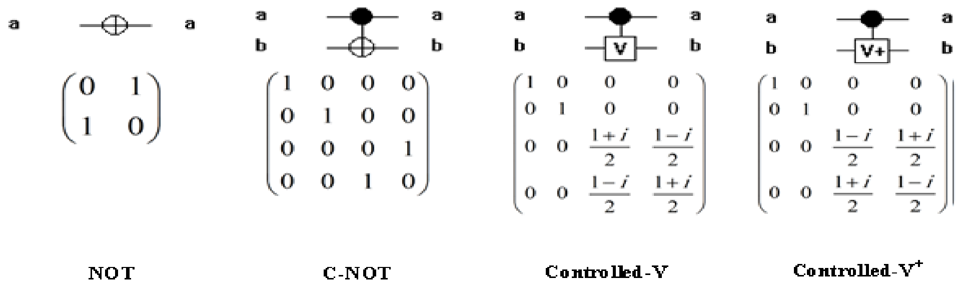

a
a
a
b
a
b
a
b
a
b
V
a
b
a
b
V+
$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1-i}{2} & \frac{1+i}{2} \\ 0 & 0 & \frac{1+i}{2} & \frac{1-i}{2} \end{pmatrix}$
NOT
C-NOT
Controlled-V
Controlled-V^+

Fig. 1: Symbol of basic quantum gates and their description matrix (Nielsen & Chuang, 2000)

In this article, the NOT and the CNOT gates are used more than other gates. So a little explanation will be given about them. A NOT gate is a single-input gate that complements its logical input value. The CNOT gate is in fact the NOT gate with a control input whose control output value is always equal to the input control value and the target output value is equal to exclusive-OR of both inputs. Therefore, if the control input value is 1, the target output value will be complement of the target input value. This is also called the Feynman gate.

Evaluation of quantum circuits is done by different criteria. One of these criteria is quantum cost. The definition of quantum cost is seen in the following.

Definition 1: The quantum cost of single input and two input gates is 1. The quantum cost of gates with an input number of more than 2 is equal to the number of basic gates required to make that gate. The quantum cost of a circuit is equal to the total quantum cost of the gates that make up it.

There are other reversible gates that can be built using basic gates. For example, the Toffoli Gate is one of them (Feynman, 1986, Szyprowski & Kerntopf, 2011). The Toffoli gate is in fact the extension of the C-NOT gate, with the number of control inputs being more than 1 (Yanofsky, Mannucci & Mannucci, 2008, Golubitsky, Falconer & Maslov, 2010). In Figure 2, the Toffoli gates with 2 and 3 control inputs are observed.

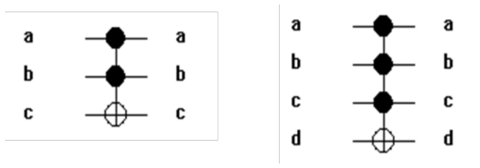


Fig. 2. Toffoli gates with 2 and 3 control inputs

In each gate, if the value of all control inputs is equal to 1, the target output value is equal to the complement of target input value and otherwise equal to the target input value. A set of all Toffoli gates with any number of inputs is a general set, and any reversible function can be implemented using gates of such a set (Miller, Maslov & Dueck, 2003).

In this paper, the Toffoli gates will also be very useful. The expansion on the Toffoli gate is that its control inputs can be positive or negative (Fazel, Thornton & Rice, 2007). The Toffoli gate, as seen in Figure 3, has both control inputs. Positive control inputs are displayed with a black circle and negative control inputs with a white circle. The target output value in these gates will only be equal to the complement of the target input value, if the value of all positive control inputs are 1, and the value of all negative control inputs are 0.

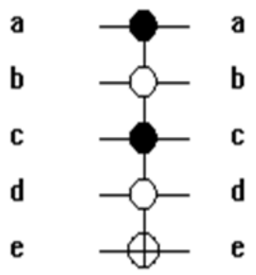


Fig. 3: Toffoli gate with 5 inputs and positive and negative control inputs (e = a b 'c d')

To implement reversible circuits, it is sometimes necessary to add inputs to the circuit, which are constant over the duration of the circuit operation. These inputs are called Ancilla. There may also be a number of outputs in the circuit so that their final values in all of our desired combinations are not significant. These outputs are called Garbage. For example, in the implementation of a reversible Full Adder circuit, because the output value of the circuit in the three different input combinations are the same, it is necessary to add two garbage outputs to make it reversible. Since the number of inputs and outputs of the reversible circuit is equal, one ancilla input will also be added to the circuit. In Section 3.3 we will look more closely into this circuit.

Cycle-based Method

Preliminaries

In this method, the reversible circuit synthesis is performed based on the cycles in the truth table. We will see the definition of the cycle.

Definition 2: If the return function f is defined as:

$$f(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_n) \qquad (1)$$

A cycle $C = (c_1, c_2, ... c_k)$ is defined as:

$$f(c_1, c_2, ..., c_k) = (c_2, c_3, ..., c_k, c_1) \qquad (2)$$

This description means that

$$f(c_1) = c_2,\ f(c_2) = c_3,\ ...,\ f(c_k) = c_1 \qquad (3)$$

Each cycle in the truth table will be implemented by a sequence of Toffoli gates, and the sequence for each cycle is completely separate from other cycles. This method is designed so that the set of gates in the circuit to realize each cycle has no effect on the other rows of the truth table. Therefore, if we have a function with a large number of inputs, that the number and the size of the cycles of its truth table are low, the number of gates intended to implement it will be low, and the circuit will be relatively small and cost-effective. This feature has made our method unique to most of the previous methods. Because in most of the existing methods, for the synthesis of reversible circuits, the gates that are considered for realization of a desired row of the truth table will cause other states of the truth table to be changed and so there is no guarantee that the truth table, whose number and size of cycles are low, must necessarily be implemented by a small circuit with a low gate number. While the method presented in this paper only applies gate in the circuit for states, in which the input value with its corresponding output value is not equal and these gates will not change any other truth table states.

Here are some definitions needed and then we will explain the method.

Definition 3: C_b is the binary representation of the integer and non-negative C.

$$C_b = C_{b(n-1)}\ C_{b(n-2)}\ ...\ C_{b1}\ C_{b0} \qquad (4)$$

C_{bi} means the bit number i in the binary display is C.

Example 1: For C = 13 we have:

$$C_b = 1101,\ C_{b0} = 1,\ C_{b1} = 0,\ C_{b2} = 1,\ C_{b3} = 1 \qquad (5)$$

Definition 4: F (v0, v1) is a Feynman gate whose control line connects to the input v0 and its target line to the input v1. This gate is visible in Figure 4.

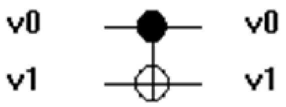


Fig. 4: The Feynman Gate F (v0, v1)

Definition 5: $T^{c}_{n+1} = T(v_0, v_1, \dots v_{n-1}, v_n)$ is a Toffoli gate with n + 1 inputs whose control lines are connected to inputs v_0 to v_{n-1} and its target line to the input v_n and We have:

$$v_i = \begin{cases} x_i \ if\ c_{bi} = 1 \\ x'_i \ if\ c_{bi} = 0 \end{cases} \ for\ i = 0\ to\ n-1 \qquad (6)$$
$$v_n = x_n$$

The Toffoli gate control lines are connected to the inputs x_0 to x_{n-1} so that if $c_{bi} = 0$, the control line is negative and if $c_{bi} = 1$, the control line will be positive. The target line will also be connected to the x_n input.

Synthesis of cycles

Suppose that the function f contains the cycle $C = (c_1, c_2, ..., c_k)$. The size of the cycle C is equal to the number of combinations in the cycle C. Cycles of size 1 mean that the inputs of the circuit with the corresponding output are the same. In these combinations, the input value must appear unchanged at the output and the gates in the circuit do not change the input value.

For example, the truth table of Figure 5 can be described as the function f(0,1,2,3,4,5,6,7) = (2,3,6,1,7,5,0,4). The cycles of this function are C = (0,2,6), (1,3), (4,7), (5).

Inputs			Outputs		
a	b	c	a	b	c
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	0	0

Fig. 5: The truth table of a reversible function with 3 Inputs / Outputs

As you can see, this function has a cycle of size 1, two cycles of size 2 and a cycle of size 3. The gates of this circuit should be arranged so that if the input takes "101" value, this combination will appear without changing in the output.

Cycles larger than 1 represent the combinations that should be converted into other combinations by the gates of the circuit. Cycles of size 2 are called Transpositions. For example, in the function corresponding to Figure 5, two transpositions are available: (1,3) , (4,7). To implement the cycle (4,7), if the binary combination of input is equivalent to 4, then the circuit gates must recognize it and convert the binary combination to 7, and vice versa. The same applies to the cycle (1.3).

In the case of cycles that are larger than 2, the subject is slightly different. For example, for implementation of the cycle (0, 2, 6), the gates are placed in such a way that the input combination 0 is converted to the output combination 2, the combination 2 to 6, and the combination 6 to 0. We will examine how each kind of these cycles will be implemented.

- *Cycles with a size of 2:*

Assume that the cycle (c_1, c_2) exists in the description of the function f. Therefore, the circuit is designed to be converted to a binary combination equal to c_2 if the binary combination of c_1 is applied to the input, and vice versa. Assuming that this cycle is the only cycle larger than 1 in function f, other possible combinations of input must not be changed by the gates in the circuit. For this purpose, the $T_{n+1}^{c_1}$ Toffoli gate is located at the beginning of the circuit to detect the c_1 combination at the input. The control lines of this gate will be connected to the circuit inputs (x_0 to x_{n-1}) and its target line will be connected to the input x_n with initial value 0. The control lines connected to the inputs whose corresponding bits in the binary representation of c_1 have a value of 1 are of positive type and others will be negative.

If the input value of the circuit is the binary representation of the number c_1, then the ancilla line x_n will be 1 by this gate, otherwise the value of the line x_n will remain unchanged at 0. In the following, we must have gates converting c_1 to c_2. For this purpose, the values of the lines where the binary representation of c_1 and c_2 is different in the bits corresponding to those lines should be changed. Therefore, Feynman gates are used where their control lines are connected to the ancilla input x_n and their target lines to the corresponding inputs for bits with a value of 1 in $c_1 \oplus c_2$. For example, consider the function f with the following definition:

f(0,1,2,3,4,5,6,7) = (0,1,2,6,4,5,3,7) (7)

The only cycle of this function whose size is greater than 1 is (3.6). To implement this cycle, a Toffoli gate $T_4^3 = T(x_0, x_1, x_2', x_3)$ is first needed to detect the value of 3 and two Feynman gates to convert it to 6. According to (8), the target line of the Feynman gates is connected to the inputs x0 and x2.

$$\begin{aligned} &c_1 = 3, c_2 = 6 \\ &(c_1 \oplus c_2)_b = 101 \end{aligned} \tag{8}$$

Therefore, the required Feynman gates are F (x3, x0) and F (x3, x2). These gates are shown in Figure 6(a). In this circuit, if the input has a value of 3, line x3 will get a value of 1, so the value of the bits x0 and x2 will be changed and the output will be converted to 6. Otherwise, none of the bits will change and the input value will appear exactly on the output.

In the following, the same trend for converting c_2 to c_1 should be repeated. In this sense, a Toffoli gate is needed to detect c_2 and several Feynman gates for converting c_2 to c_1.

In the case of the function shown in (7), the F (x3, x0) and F (x3, x2) gates are used to convert the input value 6 to value 3. The full circuit of function f is shown in Fig. 6 (b).

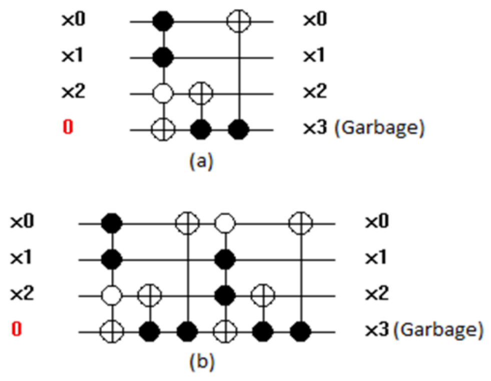


Fig. 6 - a. Reversible circuit piece to convert value from 3 to 6. b. The reversible circuit is related to function f in (7)

- *Cycles larger than 2*

If the function f has one cycle as C = (c_1, c_2, ..., c_k), to implement it, there must be a set of gates in the circuit that can identify each input combination from the values c_1 to c_k and convert it Into the next combination in the C cycle. Therefore, a Toffoli gate $T_{n+1}^{c_1}$ is needed to detect c_1 and multiple Feynman gates to convert it to c_2. Similarly, gates are required to convert c_2 to c_3 and ... and finally, gates are required to convert c_k to c_1. Also, other possible combinations of input values should not be changed by these gates. For example, consider the function f as f(0,1,2,3,4,5,6,7) = (0,1,6,2,4,5,3,7). This function has a cycle of size 3 in the form (2, 6, 3). To implement this function, three sets of gates are needed to detect the input combination 2 and convert it to 6, convert 6 to 3, and convert 3 to 2. Figure 7 shows the circuit of the above function.

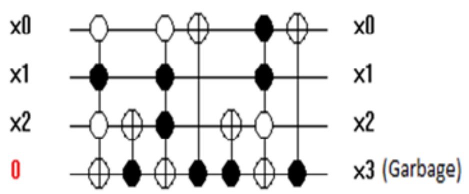


Fig. 7: The reversible circuit for the function f(0,1,2,3,4,5,6,7) = (0,1,6,2,4,5,3,7)

- *Functions containing several distinct cycles*

If the function f has more than one cycle larger than one, each of the cycles must be implemented separately. Suppose the function f has the following cycles:

$$C = (c_1, c_2, \dots, c_k), (c_{k+1}, c_{k+2}, \dots, c_p) \qquad (9)$$

You can see the circuit of the function f in Figure 8. This circuit consists of two parts. The first part is related to the first cycle of the function f and the second part of the second cycle. But we must be careful that if the input value of the circuit is one of the combinations c_1 to c_k, the value of the ancilla line x_n will be 1 at the output of the first section of circuit, which will cause the gates in the second section to change the output value. Therefore, at the end of the first part of the circuit, there should be a Toffoli gate to zero down the value of line x_n.

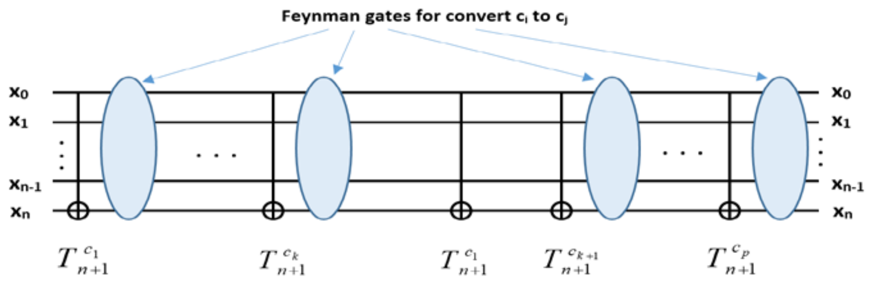


Fig. 8: The reversible circuit for the function containing the cycles $C = (c_1, c_2, \dots, c_k), (c_{k+1}, c_{k+2}, \dots, c_p)$

To synthesize each function with any number of individual cycles, there should be a Toffoli gate after the circuit for each cycle. For example, a function whose truth table is shown in Figure 5, has a circuit as Figure 9. Remember that this function has cycles C = (0,2,6), (1,3), (4,7), (5).

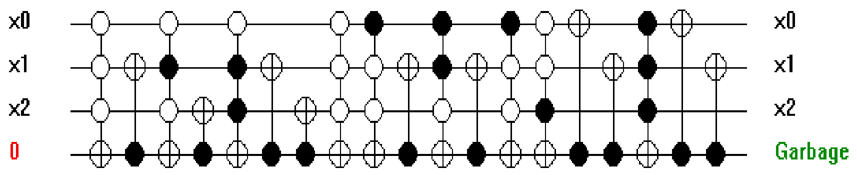


Fig. 9: The reversible circuit for the function f(0,1,2,3,4,5,6,7) = (2,3,6,1,7,5,0,4)

Improvement in proposed algorithm

In this section, we will examine some of the improvements that can be made to the proposed algorithms in some cases and provide better results.

Identifying adjacent transpositions

If a cycle of size 2 in the form (c1, c2) exists in function f so that the binary representation of c1 and c2 are different in only one bit, then to implement it, the method presented in (Zhu et al., 2018) can be used. In this method, a Toffoli gate with n inputs is used. The gate control lines will connect to the inputs corresponding to the bits with the same value in c1 and c2, and the gate target line will connect to input corresponding to the bit with different values in c1 and c2. For example, for implementation of cycle (2,6), instead of using the circuit of Figure 10.a, we can use the circuit of Figure 10.b.

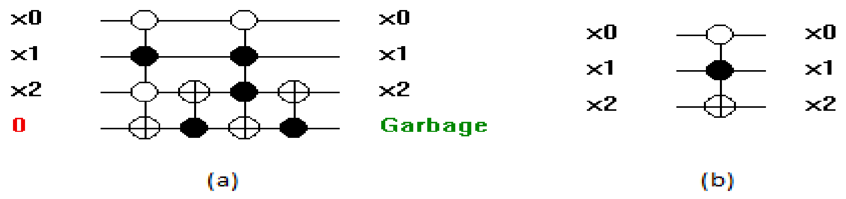


Fig. 10: Circuit related to the cycle (2,6). a. Synthesized with proposed algorithm b. Synthesized with the algorithm presented in (Zhu et al., 2018)

For another example, the circuit corresponding to the truth table of Figure 5, with this improvement, is shown in Figure 11. Note that in this function, the cycle (1,3) can be implemented by the algorithm presented in (Zhu et al., 2018). The quantum cost of circuit in figure 9 is 131. While the quantum cost of this circuit decreases to 95 after the improvement.

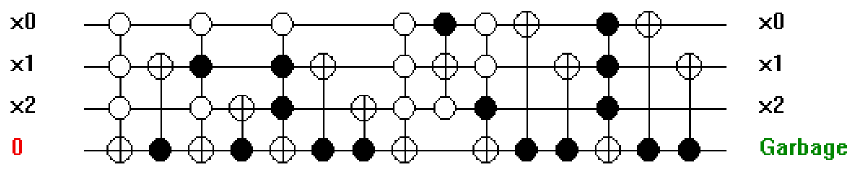


Fig. 11: Reversible circuit related to function f(0,1,2,3,4,5,6,7) = (2,3,6,1,7,5,0,4) with improvement

- *Adding an ancilla input per cycle*

In the proposed algorithm, as shown, after a circuit for each cycle, a Toffoli gate with n + 1 input must be placed to return the input value of x_n to 0 again and prevent unwanted change by the gates of the next section of the circuit. Toffoli gates with high inputs have high quantum cost. But instead of using them after the implementation of each cycle, it is possible to use a new ancilla input for the next cycle. For example, the circuit related to the function f with cycles C = (0,5), (1,3,7), (2,4) which is generated by the proposed algorithm, is shown in Fig. 12.a. This function consists of three cycles. So, as seen, between the circuits of these three cycles, there are two 4-input Toffoli gates. The quantum cost of this circuit is 133. Now adding the two ancilla inputs to this circuit can eliminate the above gates and reduce the quantum cost of the circuit. The circuit obtained in this way is shown in Figure 12.b. The quantum cost of the improved circuit is reduced to 105.

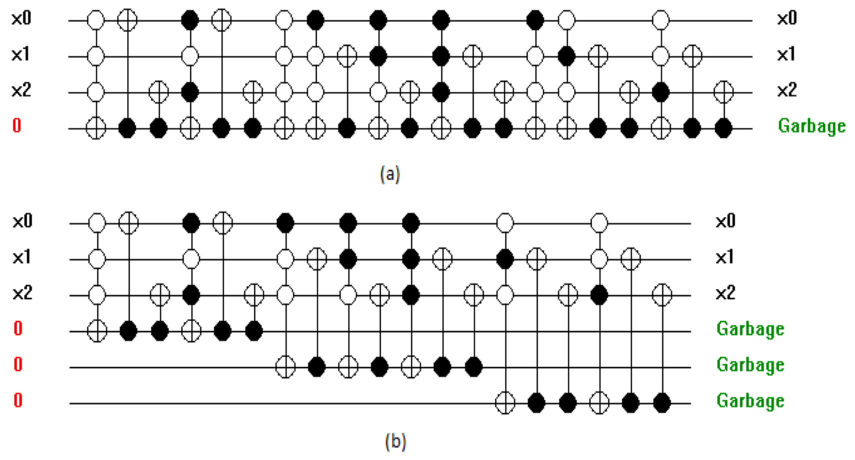


Fig. 12: The circuit related to the function f with cycles C = (0,5), (1,3,7), (2,4). a. With one ancilla input. b. With several ancilla inputs

- *Considering don't care combinations and outputs*

For some reasons, including adding ancilla inputs, some of the input combinations of the function may be considered don't care. In such cases, the values that are considered as output of care combinations can be excluded and it is possible to assign other values in some way to the output of don't care combinations in which the minimum number of cycles is created. To do this, we must try to consider the output value of any don't care combination, if possible, equal to the input value of the same combination. Thus, since cycles of size 1 are generated, this algorithm does not consider any gate to implement the function.

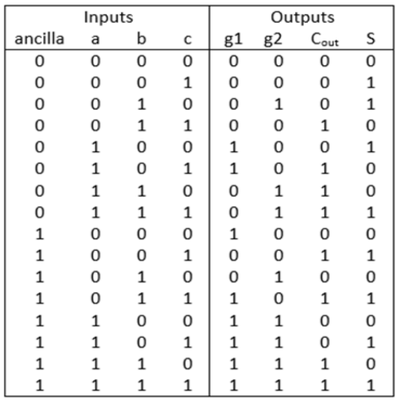
Also, if we have garbage outputs that have the don't care value in all combinations, in the initialization of these outputs, it can be operated in such a way that, if possible, the input and output values are identical in different combinations.

For example, the truth table in Figure 13 relates to a reversible Full Adder circuit with one ancilla input and two garbage outputs. As you know, due to the fact that the output of the 3 combinations in this table is the same, in order to make this circuit reversible, two garbage outputs have to be added to it, so one ancilla input will also be added to the circuit.

Inputs				Outputs			
ancilla	a	b	c	g1	g2	C_{out}	S
0	0	0	0	x	x	0	0
0	0	0	1	x	x	0	1
0	0	1	0	x	x	0	1
0	0	1	1	x	x	1	0
0	1	0	0	x	x	0	1
0	1	0	1	x	x	1	0
0	1	1	0	x	x	1	0
0	1	1	1	x	x	1	1
1	0	0	0	x	x	x	x
1	0	0	1	x	x	x	x
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

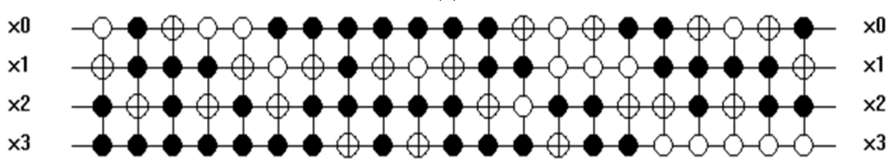
Fig. 13: The truth table of reversible Full Adder with don't care combinations and outputs

To synthesize this circuit, uncertain situations must be identified in the truth table. For example, the value of outputs in combinations 0, 1, 6 and 7 is set to a value equal to the same input values in the output of these combinations, respectively. We set the output of other don’t care combinations with the goal of minimizing the number of cycles. The generated truth table and the circuit for it by the algorithm presented in [8] and also the proposed algorithm are shown in Figure 14. In this truth table, the desired function consists of a cycle C = (2,5,10,4,9,3). As can be seen, the quantum cost of the synthesized circuit by the algorithm presented in [8] is 273. While the quantum cost of the circuit generated by the proposed algorithm is 190.

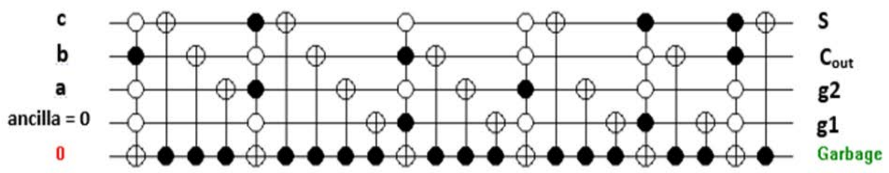


Inputs				Outputs			
ancilla	a	b	c	g1	g2	C_{out}	S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1
0	0	1	1	0	0	1	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1

(a)



(b)



(c)

Fig. 14: a. The truth table of reversible Full Adder without any don’t care. b. Synthesized circuit by the algorithm presented in (Zhu et al., 2018). c. Synthesized circuit by proposed algorith

Practical Results

In this section we will compare the results of the proposed algorithm implementation with the previous work. Note that the algorithm of this paper has a very good performance compared to other algorithms in cases where the number of cycles of the function is small compared to the number of inputs. The reason for this is that our method adds the gate to the circuit only for combinations in the truth table, which does not have the same input value as the output and these gates will not affect other combinations. But in the previous methods, the gate that is added to the circuit for one combination in the truth table can also change other combinations and, for functions with limited number of cycles, large and expensive circuits may be produced.

First, we examine a number of functions as examples and compare the results of each of the algorithms presented in (Miller, Maslov & Dueck, 2003; Zhu et al., 2018), as well as the proposed algorithm of this paper. Then we will compare the results of the implementation of these algorithms on different functions with the different number of inputs, as well as the different number and size of cycles in a table.

First, consider the following function with the four inputs below:

$$f(0,1,2,3,\ldots,15) = (0,2,1,3,\ldots,15) \qquad (10)$$

As can be seen, this function only contains a cycle C = (1,2). In fact, in only two possible input combinations, its output value varies with the input value. Figure 15(a) shows the synthesis of this function by the Transformation-based algorithm presented in (Miller, Maslov & Dueck, 2003). Figure 15.b shows the result of the algorithm presented in (Zhu et al., 2018) on this function. And also in Figure 15.c, the circuit obtained from the algorithm of this paper, along with the quantum cost of each, is seen. As it is known, the previous algorithms produced a fairly large circuit for a simple function that varies in the input and output values in only two cases. This defect has been resolved in our algorithm.

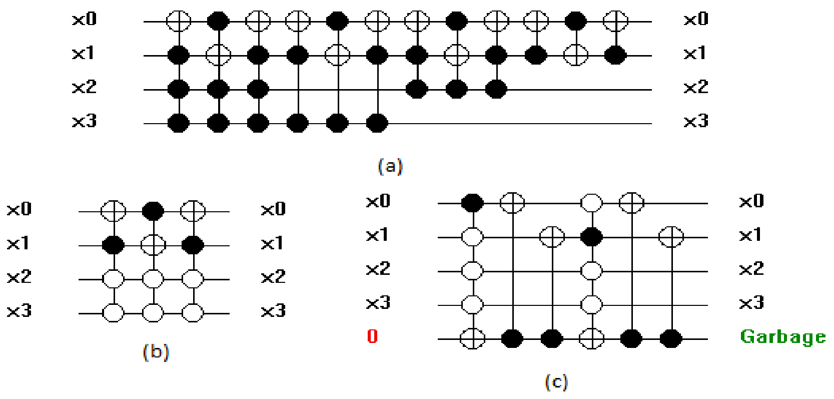


Fig. 15: Synthesis of a 4-input function with a cycle C = (1,2) a. By the method presented in (Miller, Maslov & Dueck, 2003) with quantum cost 72. b. By the method presented in (Zhu et al., 2018) with quantum cost 39. c. By proposed algorithm with quantum cost 62.

As the size and dimensions of the function become larger, the difference between the results of this algorithm and the previous works is revealed. As another example, we synthesize the 5-input function containing the cycle C = (1,2,5) by the above algorithms, the results of which are shown in Figure 16.

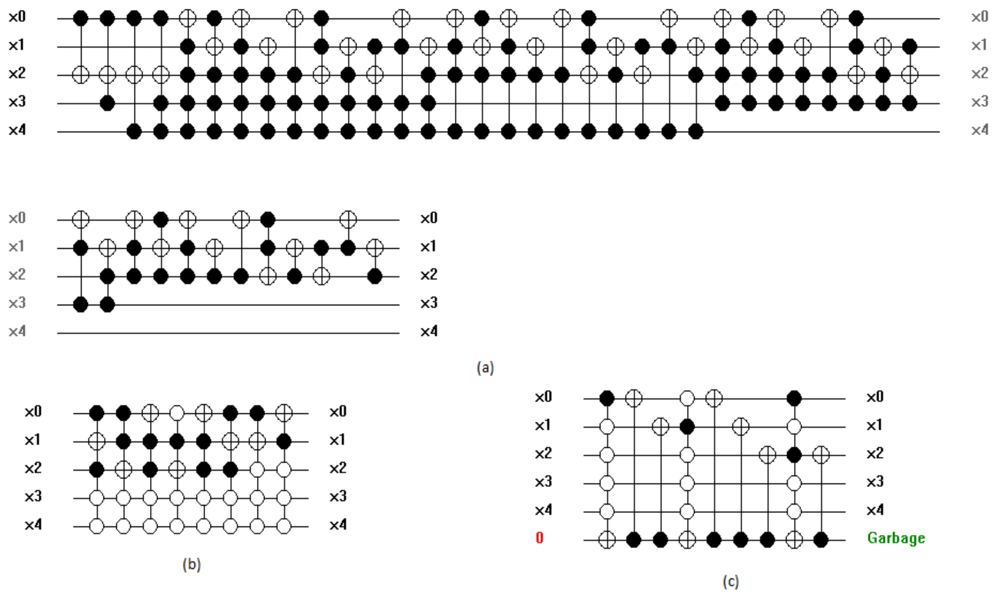


Fig. 16: Synthesis of the input function 5 with a cycle C = (1,2,5) a. By the method presented in (Miller, Maslov & Dueck, 2003) with a quantum cost of 406. b. By the method presented in (Zhu et al., 2018) with a quantum cost of 232. c. The proposed algorithm with a quantum cost of 189.

In Table 1, the generated circuits are compared in terms of quantum cost for different functions, based on the dimensions of the circuit as well as the number of cycles and the size of the cycles in the description of the function. Generated circuits are simulated and analyzed by RC viewer tool. In this table, different functions are considered with different input numbers. These functions have a different number and size of the cycles. The results of the synthesis of these functions by the algorithm presented in (Miller, Maslov & Dueck, 2003), which is one of the basic algorithms for the synthesis of any type of function, the algorithm presented in (Zakablukov, 2016), which is based on the most recent cycle-based algorithms, as well as the proposed algorithm is presented in the table. The first to third column shows the number of inputs of the requested function, the number of cycles in the description of the function, and the size of these cycles. The next three columns represent the number of ancilla inputs that have generated the by algorithm in the synthesis of the circuit. The last three columns represent the quantum cost of the circuit generated by these three algorithms. If you look at the values in the table, you will see that whatever the requested function is larger and the number and size of the cycles in the function is lower, the proposed algorithm yields a better result than the other two algorithms.

Table 1 - Results of Synthesis Algorithms on Different Functions

#	Number of Inputs	Size of Cycles	Number of Cycles	Number of Ancilla Inputs			Quantum Cost		
				(Miller, Maslov & Dueck, 2003)	(Zhu et al., 2018)	Proposed Algorithm	(Miller, Maslov & Dueck, 2003)	(Zhu et al., 2018)	Proposed Algorithm
1	2	2	1	0	0	1	3	3	14
2	3	2	1	0	0	1	18	15	30
3	3	3	1	0	0	1	25	40	45
4	3	5	2	0	0	1	21	45	50
5	4	2	1	0	0	1	72	39	62
6	4	3	1	0	0	1	112	104	93
7	4	4	1	0	0	1	78	91	124
8	4	5	1	0	0	1	74	104	155
9	4	5	2	0	0	1	90	117	106
10	5	2	1	0	0	1	252	87	126
11	5	3	1	0	0	1	406	232	189
12	5	4	1	0	0	1	250	203	252
13	5	5	1	0	0	1	267	232	315
14	5	5	2	0	0	1	324	261	218
15	5	6	2	0	0	2	230	232	324
16	5	6	2	0	0	1	320	234	283
17	6	2	1	0	0	1	828	183	254
18	6	3	1	0	0	1	1354	488	381
19	6	4	1	0	0	1	834	427	508
20	6	5	1	0	0	1	811	488	635
21	6	5	2	0	0	1	1080	549	433
22	6	6	2	0	0	2	796	2318	500
23	6	7	1	0	0	1	1206	2196	897
24	6	7	2	0	0	2	921	2135	584
25	6	7	3	0	0	2	1080	1708	456
26	6	8	1	0	0	1	1251	2501	1024
27	6	8	2	0	0	2	1339	2318	666
28	6	8	3	0	0	3	967	2135	666

Table 2- Normalized results for comparison of cost and performance criteria

#	Cost Criteria	(Miller, Maslov & Dueck, 2003)	(Zhu et al., 2018)	Proposed Algorithm
4	0/8	2/625	5/625	6/25
9	1/6	5/625	7/3125	6/625
1	2	0/75	0/75	3/5
3	2/666667	3/125	5	5/625
15	2/666667	7/1875	7/25	10/125
16	2/666667	10	7/3125	8/84375
28	2/666667	15/10938	33/35938	10/40625
25	3/047619	16/875	26/6875	7/125
8	3/2	4/625	6/5	9/6875
14	3/2	10/125	8/15625	6/8125
2	4	2/25	1/875	3/75
7	4	4/875	5/6875	7/75
27	4	20/92188	36/21875	10/40625
24	4/571429	14/39063	33/35938	9/125
6	5/333333	7	6/5	5/8125
22	5/333333	12/4375	36/21875	7/8125
13	6/4	8/34375	7/25	9/84375
21	6/4	16/875	8/578125	6/765625
5	8	4/5	2/4375	3/875
12	8	7/8125	6/34375	7/875
26	8	19/54688	39/07813	16
23	9/142857	18/84375	34/3125	14/01563
11	10/66667	12/6875	7/25	5/90625
20	12/8	12/67188	7/625	9/921875
10	16	7/875	2/71875	3/9375
19	16	13/03125	6/671875	7/9375
18	21/33333	21/15625	7/625	5/953125
17	32	12/9375	2/859375	3/96875
Average	7/3748	10/50725	12/87723	7/702009

To better analyze the results, we define the criterion for evaluating the complexity of the requested function as follows:

$$Complexity\ Criterion = \frac{2^{number\ of\ input}}{number\ of\ cycles * cycle\ size} \quad (11)$$

Also, because the quantum cost is dependent on the magnitude of the circuit, to quantify the cost of different circuits, we obtain the standardized normalized cost with the following definition:

$$Cost\ Criterion = \frac{Quantum\ cost}{2^{number\ of\ input}} \quad (12)$$

The results in Table 1 are normalized in Table 2. For a more straightforward comparison, we arrange the table based on the complexity criterion. As you can see, the larger the number of inputs in the circuit and the smaller the cycles and the size of the reversible function cycles are required, the quantum cost of the circuit generated by the proposed algorithm will be less than the previous algorithms. As explained earlier, this is because earlier algorithms usually check the truth table from top to bottom and they try to make each combination of the truth table and this will change the next lines. But the proposed method only affects the current combination, without changing other combinations.

At the end of Table 2, we get the average value of each column. As can be seen, the proposed method has improved in the cost criterion, on average, between 27% and 40% in circuits where the use of this algorithm is justified.

Conclusion

One of the challenges in the field of modern integrated circuit technologies, such as quantum, optical, etc., is the synthesis of reversible circuits. A lot of work has been done in this field. Many of the works done on the synthesis of reversible circuits has focused on the cycles in their function. These tasks generally find the cycles in the truth table and try to implement them, using existing reversible gates. In these methods, when a combination of the truth table is considered and a gate is added to the circuit for its implementation, this gate usually changes the other combinations of the table. In cases where the original truth table has small cycles, these methods cause many gates to be used to implement such a function.

The proposed method, by relying on the cycles in the truth table, produces a circuit in such a way that these cycles are implemented without any change in the other combinations of the truth table. Therefore, for functions with low-size cycles, it will generate much less expensive circuits than other methods. The cycles to which they are considered can be of the size of 2 or more than 2. In the end, we have made improvements that reduce the cost of the circuit generated or extend the application of this algorithm. In the case of cycles of size 2 that are between adjacent states, we can use less costly gates. Also, for functions whose descriptions have don’t care combinations or outputs, a method is proposed that reduces the number of cycles and their size and, consequently, reduces the cost of the generated quantum circuit.

We compare the results with the other works presented in (Miller, Maslov & Dueck, 2003; Zhu et al., 2018). In (Miller, Maslov & Dueck, 2003), the basic method for the synthesis of reversible circuits was proposed in 2003, and (Zhu et al., 2018) a method based on the cycles in the truth table is presented in 2018. The results of the comparison show that in cases where a large function with a small number and size of cycles is present, the previous methods will generate a costly circuit for it. But the proposed method works much better in these cases. Considering the criteria for evaluation and comparison, for the above functions, the proposed algorithm improved the cost of generated reversible circuit between 27% and 40%.

References

Bhagyalakshmi HR, Venkatesha MK. “An improved design of a multiplier using reversible logic gates.” International journal of engineering science and technology. 2010 Feb;2(8):3838-45.

Drechsler R, Finder A, Wille R. “Improving ESOP-based synthesis of reversible logic using evolutionary algorithms.” InEuropean Conference on the Applications of Evolutionary Computation 2011 Apr 27 (pp. 151-161). Springer, Berlin, Heidelberg.

Fazel K, Thornton MA, Rice JE. “ESOP-based Toffoli gate cascade generation.” InCommunications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on 2007 Aug 22 (pp. 206-209). IEEE.

Feynman R. P., "Quantum mechanical computers." Foundations of physics 16, no. 6 (1986): 507-531.

Golubitsky O., Falconer S. M., Maslov D., "Synthesis of the optimal 4-bit reversible circuits." In Proceedings of the 47th Design Automation Conference, pp. 653-656. ACM, 2010.

Landauer R. “Irreversibility and heat generation in the computing process.” IBM journal of research and development. 1961 Jul;5(3):183-91.

Maslov D., Dueck G. W., Miller D. M., "Fredkin/Toffoli templates for reversible logic synthesis." In Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design, p. 256. IEEE Computer Society, 2003.

Maslov D., Dueck G. W., Miller D. M., "Synthesis of Fredkin-Toffoli reversible networks." IEEE Transactions on Very Large Scale Integration (VLSI) Systems13, no. 6 (2005): 765-769.

Maslov D., Dueck G. W., Miller D. M., Negrevergne C., "Quantum circuit simplification and level compaction." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27, no. 3 (2008): 436-444.

Metodi T. S., Chong F. T., "Quantum computing for computer architects." Synthesis Lectures in Computer Architecture 1, no. 1 (2006): 1-154.

Miller DM, Maslov D, Dueck GW. “A transformation based algorithm for reversible logic synthesis.” InDesign

Automation Conference, 2003. Proceedings 2003 Jun 2 (pp. 318-323). IEEE.

Nielsen, M. A. & Chuang, I. L. “Quantum Computation and Quantum Information” (Cambridge Univ. Press, Cambridge, UK, 2000).

Saeedi M, Zamani MS, Sedighi M, Sasanian Z. “Reversible circuit synthesis using a cycle-based approach.” ACM Journal on Emerging Technologies in Computing Systems (JETC). 2010 Dec 1;6(4):13.

Sasanian Z, Saeedi M, Sedighi M, Zamani MS. “A cycle-based synthesis algorithm for reversible logic.” InProceedings of the 2009 Asia and South Pacific Design Automation Conference 2009 Jan 19 (pp. 745-750). IEEE Press.

Shende VV, Prasad AK, Markov IL, Hayes JP. “Synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.” 2003 Jun;22(6):710-22.

Szyprowski M., Kerntopf P., "Reducing quantum cost in reversible Toffoli circuits." arXiv preprint arXiv:1105.5831 (2011).

Yanofsky NS, Mannucci MA, Mannucci MA. “Quantum computing for computer scientists.” Cambridge: Cambridge University Press; 2008 Aug 11.

Zakablukov DV. “Application of permutation group theory in reversible logic synthesis.” InInternational Conference on Reversible Computation 2016 Jul 7 (pp. 223-238). Springer, Cham.

Zhu W, Li Z, Zhang G, Pan S, Zhang W. “A Reversible Logical Circuit Synthesis Algorithm Based on Decomposition of Cycle Representations of Permutations.” International Journal of Theoretical Physics. 2018 May:1-9.