www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2019, VOL. 65, NO. 1, PP. 33-38
Manuscript received December 13, 2018; revised January, 2019. DOI: 10.24425/123562

The Transforming Method Between Two Reversible Functions

Andrzej Skorupski, and Krzysztof Gracki

***Abstract*—This paper presents an original method of designing some special reversible circuits. This method is intended for the most popular gate set with three types of gates CNT (Control, NOT and Toffoli). The presented algorithm is based on two types of cascades with these reversible gates. The problem of transformation between two reversible functions is solved. This method allows to find optimal reversible circuits. The paper is organized as follows. Section 1 and 2 recalls basic concepts of reversible logic. Especially the two types of cascades of reversible function are presented. In Section 3 there is introduced a problem of analysis of the cascades. Section 4 describes the method of synthesis of the optimal cascade for transformation of the given reversible function into another one.**

***Keywords* —reversible logic, reversible circuits, reversible gate, CNT set of the gates**

I. Introduction

THE conventional Boolean function synthesis starts with a universal gate library and the target of this process is to finding logic circuits that implement this function. These circuits need to meet the criterion of minimization, e.g., the circuits need to consist of a minimal number of the gates. Usually the number of output bits is relatively small compared with the number of input bits (Fig. 1a). However, there are applications, e.g. digital signal processing, computer graphics, cryptography and so on, which require an equal number of input and output bits (Fig. 1b). These circuits could be lossless information circuits as well as reversible ones if mapping of the vector input into the vector output is mutually unambiguous.

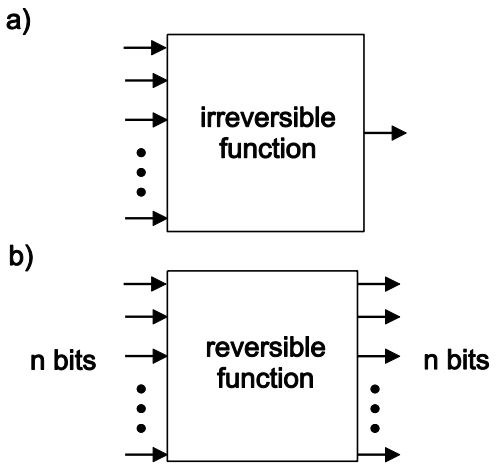

Fig. 1. a) Irreversible circuit, b) reversible circuit

The authors are from the Institute of Computer Science, Warsaw University of Technology, Poland (e-mail: ask@ii.pw.edu.pl, kgr@ii.pw.edu.pl).

Landauer showed that the loss of information implies energy loss [1]. The result of this theorem is the possibility of construction of the energy lossless circuits. The condition of this design is reversibility. The other conditions are: no fan-outs and no feed-backs [2].

A function with n inputs and n outputs is called reversible if is bijective, i.e., if each input pattern uniquely maps to an output pattern, and vice versa [21]. Each of the Boolean functions included in the reversible function has the same number of the 0-s and 1-s minterms. This is a mutual unambiguity of the function.

Applications of the reversible circuits can be found in the emerging fields of the quantum computation and low-power computation. The quantum computations require the use of the reversible gates. These gates can be built using various technologies such as the semiconductor CMOS [3,4], optical [5] and thermodynamic ones [6] and so on. In order to build the gates researchers must overcome many difficulties.

The other area of development is the logic synthesis of reversible circuits. The base of the synthesis are the types of the gates used in this procedure. There are many types of the gates as: NOT, Controlled NOT, Toffoli, Fredkin, Kerntopf gates and others [7,8,9,10]. One of the most popular set of the gates is the CNT set (Controlled NOT, NOT and Toffoli gates).

The result of the synthesis procedure is the circuit consisting of the gates cascade from a given library set (Fig. 2). These gates are serially connected creating the chain called cascade.

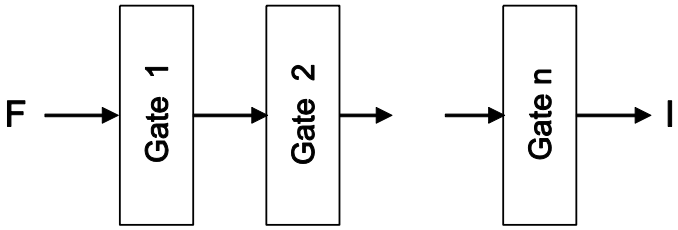

Fig. 2. The cascade of the reversible gate

In general case this cascade transforms the input vector into the output vector. This transformation can be described by true table similarly to the true table for Boolean function (Tab. I).

There are many methods of synthesis of the reversible functions. The main target of these methods is to find the circuits which transform the given function (input of this circuit) into the identical function (output of this circuit) using a minimal number of gates. This circuit is called optimal. Usually there exist many optimal circuits for the given function.

TABLE I
TRUE TABLE OF THREE VARIABLE REVERSIBLE FUNCTION

$X_2X_1X_0$	$Y_2Y_1Y_0$
000	000
001	001
010	010
011	100
100	011
101	101
110	110
111	111

Researchers try to find the computer aided methods which could find the optimal circuits for any reversible functions. For the three-variable reversible function the problem is solved by the transformation method [11,12], ESOP method [13,14] or BBD method [15]. But for the functions of more variables the algorithms are very time and memory space consuming [16,17,18,19,20,21,22].

In this paper we will assume that the solutions for any functions are known. This assumption is real because in this paper will concern the three-variable functions and many methods are efficient enough to find optimal solutions in a simple way. The presented method is scalable for the functions with bigger number of the variables. For example there is the algorithm to find optimal circuits for four-variables functions [18].

II. REVERSIBLE GATES AND FUNCTION

The classical synthesis problem of reversible functions concerns the transformation of the given function F into identical function I (Fig..2). The solution of this problem is a cascade of reversible gates containing a minimal number of gates. In this paper there will be used the CNT set of the gates containing 12 gates with three variables each. Four gates with XOR gate on line Y_0 are shown in Fig. 3.

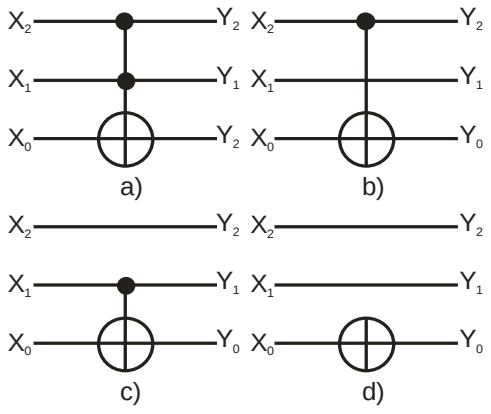

Fig.3. Four reversible gates with XOR on line Y_0: a) T0, b) C0-2, c) C0-1, d) N0

The gates from Fig. 3 implement the reversible function:

$$Y_2 = X_2$$
$$Y_1 = X_1$$
$$Y_0 = X_0 \oplus ab \text{ where:}$$

$a = X_2$ and $b = X_1$ Fig. 3a
$a = X_2$ and $b = 1$ Fig. 3b
$a = 1$ and $b = X_1$ Fig. 3c
$a = 1$ and $b = 1$ Fig. 3d

The remaining eight gates are defined in the same manner: four with XOR on line Y_2 (T2, C2-1, C2-0, N2) and four on line Y_1 (T1, T1-2, T1-0, N1).

The implementation of the given reversible function F is the cascade of reversible gates [20]. There are two types of cascades shown in Fig. 4. It should be noted that these cascades differ in terms of the gates order.

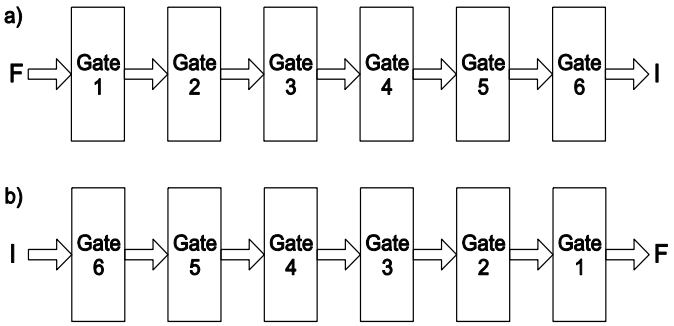

Fig. 4. Cascades with 6 reversible gates

The cascade in Fig. 4a (Type 1) transforms the function F into I (I is the identical function - left side of the true table in Table II) and the cascade in Fig. 4b (Type 2) transforms function I into the given function F.

Let be given the reversible function of three variables presented in Table II.

TABLE II
EXAMPLE OF THREE VARIABLE REVERSIBLE FUNCTION

No.	$X_2X_1X_0$	$Y_2Y_1Y_0$
0	000	000
1	001	001
2	010	010
3	011	100
4	100	011
5	101	101
6	110	110
7	111	111

The function from Table II could be presented as minterms permutation i.e. <0,1,2,4,3,5,6,7>. This is a sequence of the vectors $Y_2Y_1Y_0$ in the order relating to the vectors $X_2X_1X_0$. Only two vectors (100 and 011) are in improper places relative to the identical function. Each gate in the cascade of type 1 swapped the proper pairs of minterms. In Table III there are collected the gates names and suitable minterms of the function I swapped when the given gate is used.

For example, if on the input of the gate C1-2 there is the function F then on the output of this gate there will be the function F_r(C1-2) where the function F_r(C1-2) has swapped minterms from row 4 with 6 and 5 with 7 of the true table. The index r indicates that rows are swapped. If F = <0,1,2,4,3,5,6,7> will be the input function of the gate C1-2, the output function of this gate

F_r(C1-2) = <0,1,2,4,6,7,3,5>. This output function results from the swap of the minterms 3 (row 4) with 6 (row 6) and 5 (row 5) with 7 (row 7).

Each gate in the cascade of type 2 also swapped the proper pairs of minterms. In general there are different minterms than those for the cascade of type 1. In this case there are swapped the minterms with the values presented in the second column of Table III.

For example, if on the input of the gate C1-2 there is the function F=<0,1,2,4,3,5,6,7> then on the output of this gate there will be the function F_v(C1-2)=<0,1,2,6,3,7,4,5> where the function F_v(C1-2) has swapped minterms 4 with 6 and 5 with 7 relating to the function F. The index *v* indicates the value swapped operation.

TABLE III
SWAPPED MINTERMS FOR REVERSIBLE GATES

Gate	Swapped rows/values
T0	6,7
C0-1	2,3 & 6,7
C0-2	4,5 & 6,7
N0	0,1 & 2,3 & 4,5 & 6,7
T1	5,7
C1-0	1,3 & 5,7
C1-2	4,6 & 5,7
N1	0,2 & 1,3 & 4,6 & 5,7
T2	3,7
C2-0	1,5 & 3,7
C2-1	2,6 & 3,7
N2	0,4 & 1,5 & 2,6 & 3,7

Definition 1.

The invers cascade C' relating to the given cascade C is the cascade with opposite setting of the gates.

For example, the invers cascade to cascade G1, G2, G3, G4 is the cascade with gates order: G4, G3, G2, G1.

Definition 2.

The function F' is called the invers function to function F if the function F' is implemented by the invers cascade to the cascade implemented by the function F.

Corollary 1.

Let be the cascade Type 1 where F_{in}(G1)=F and F_{out}(GN)=I. To calculate the function F_{out}(Gi) on the output of the gate Gi when on the input of this gate is the function F_{in}(Gi) we should use the formula:

$$F_{out}(Gi) = F_r(F_{in}(Gi))$$

where $F_r(F_{in}(Gi))$ is the input function F_{in}(Gi) of the gate Gi with swapping minterms from proper rows determined by the gate Gi.

Corollary 2.

Let be the invers cascade (Type 2) where F_{in}(G1)=I. To calculate the function F_{out}(Gi) on the output of the gate Gi when on the input of this gate is the function F_{in}(Gi) we should use the formula:

$$F_{out}(Gi) = F_v(F_{in}(Gi))$$

where $F_v(F_{in}(Gi))$ is the input function F_{in}(Gi) of the gate Gi with swapping minterms from proper rows determined by the gate Gi. In this case F_{out}(GN) = F.

Corollary 3.

Let be the cascade Type 1 where F_{in}(G1)=I. To calculate the function F_{out}(Gi) on the output of the gate Gi when on the input of this gate is the function F_{in}(Gi) we should use the formula:

$$F_{out}(Gi) = F_v(F_{in}(Gi))$$

where $F_v(F_{in}(Gi))$ is the input function F_{in}(Gi) of the gate Gi with swapping minterms from proper values determined by the gate Gi. In this case F_{out}(GN)=F.

Corollary 4.

Let be the invers cascade (Type 2) where F_{in}(G1)=F and F_{out}(GN)=I. To calculate the function F_{out}(Gi) on the output of the gate Gi when on the input of this gate is the function F_{in}(Gi) we should use the formula:

$$F_{out}(Gi) = F_r(F_{in}(Gi))$$

where $F_r(F_{in}(Gi))$ is the input function F_{in}(Gi) of the gate Gi with swapping minterms from proper rows determined by the gate Gi.

Example 1.

Let be the function F=<4,1,3,6,0,5,2,7>. The optimal cascade Type 1 C=(N2,T0,C2-0) implements the function F. From Corollary 1 when the cascade Type 1 and the input function F=<4,1,3,6,0,5,2,7> will be used then:

The output of the gate N2 is the function:

$$F_r(N2)=<0,5,2,7,4,1,3,6>.$$

The output of the gate T0 is the function:

$$F_r(N2,T0)=<0,5,2,7,4,1,6,3>.$$

The output of the gate C2-0 is the function:

$$F_r(N2,T0,C2\text{-}0)=<0,1,2,3,4,5,6,7>.$$

From Corollary 2 when the invers cascade (Type 2) and the input function I=<0,1,2,3,4,5,6,7> will be used then:

The output of the gate C2-0 is the function:

$$F_r(C2\text{-}0)=<0,5,2,7,4,1,6,3>.$$

The output of the gate T0 is the function:

$$F_r(C2\text{-}0,T0)=<0,5,2,7,4,1,3,6>.$$

The output of the gate N2 is the function:

$$F_r(C2\text{-}0,T0,N2)=<4,1,3,6,0,5,2,7>.$$

From Corollary 3 when the cascade Type 1 and the input function I=<0,1,2,3,4,5,6,7> will be used then:

The output of the gate N2 is the function:

$$I(_v(N2)=<4,5,6,7,0,1,2,3>.$$

The output of the gate T0 is the function:

$$I_v(N2,T0)=<4,5,7,6,0,1,2,3>.$$

The output of the gate C2-0 is the function:

$$I_v(N2,T0,C2\text{-}0)=<4,1,3,6,0,5,2,7>.$$

From Corollary 4 when the invers cascade (Type 2) and the input function F=<4,1,3,6,0,5,2,7> will be used then:

The output of the gate C2-0 is the function:

$$F_v(C2\text{-}0)=<4,5,7,6,0,1,2,3>.$$

The output of the gate T0 is the function:

$$F_v(C2\text{-}0,T0)=<4,5,6,7,0,1,2,3>.$$

The output of the gate N2 is the function:

$$F_v(C2\text{-}0,T0,N2)=<0,1,2,3,4,5,6,7>.$$

Definition 3.

The serial connection of the two cascades C1 and C2 will be called concatenation of these cascades C1○C2 where the input of the cascade C2 is controlled by the output of the cascade C1.

Let us assume that the cascade C2 is a one-gate cascade. Using the below Lemma 1 it is possible to determine the function implemented by the two cascades C1○C2.

Lemma 1.

If the function F is implemented by the cascade C Type 1 then the cascade C○GX implements the function $F_v(GX)$.

Proof. If on the output of the last gate GX in the cascade C○GX there is the identical function I, then on the input of this gate there is the function $I_v(GX)$. Assuming that on the inputs of the all gates Gi of the cascade C there are the functions Fi. The result of the concatenation of the gate GX to the cascade C all these functions Fi will be the functions $Fi_v(GX)$. One of these gates is the first gate in the cascade and on its input there will be the function $F1_v(GX)$.

Example 2.

Let be two cascades:

First cascade C1=(N2,T0,C2-0) and the second one-gates cascade T1. The first cascade C1 implements the function F=<4,1,3,6,0,5,2,7>. The gate T1 swaps the minterms 5 and 7. The function implemented by the cascade N2,T0,C2-1,T1 will be $F_v(T1)$= <4,1,3,6,0,7,2,5>.

The generalization of the Lemma 1 leads to Lemma 2.

Lemma 2.

If the function F1 is implemented by the cascade C1 and the function F2 is implemented by the cascade C2 then the cascade C1○C2' implements the function $F=F1_v(C2')$.

Proof. This Lemma 2 is illustrated in the Fig. 5 where is showed cascade C1○C2'. If the first three gates (cascade C1=G1,G2,G3) implement the function F1 then on the output of the gate G3 there is the identical function. The function F2 is implemented by cascade C2=G7,G6,G5,G4. The next four gates in cascade C1○C2' are the invers cascade to C2. From Corollary 2 on the output of this cascade there is the function F2 (Fig. 5). If we will use four times the result of the Lemma 1 we receive sequentially:

- joining only the gate G4 to cascade C1 the function F1 change into the function $F1_v(G4)$,
- joining the gate G5 to cascade C1○G4 the function $F1_v(G4)$ change into the function $F1_v(G4,G5)$,
- joining the gate G6 to cascade C1○G4○G5 the function $F1_v(G4,G5)$ change into the function $F1_v(G4,G5,G6)$,
- joining the gate G7 to cascade C1○G4○G5○G6 the function $F1_v(G4,G5,G6)$ change into the function $F1_v(G4,G5,G6,G7)=F1_v(C2')$.

If on the output of the cascade C1○C2' is identical function I then on the input of this cascade C1○C2' is the function $F1_v(C2')$, where F1 is the function implementing by cascade C1.

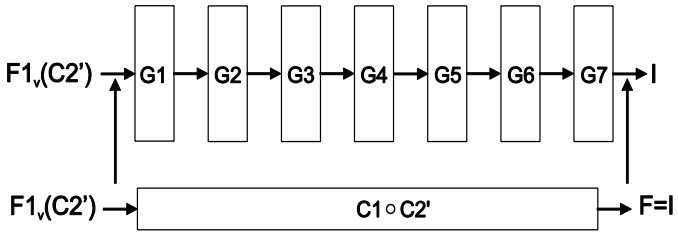

Fig. 5. The concatenation of two cascades

Example 3.

Let be two cascades:

1. C1=(N2,T0,C2-0) the optimal cascade of the function F1, where F1=<4,1,3,6,0,5,2,7>.
2. C2=(C1-2,T1,C0-1,C2-0) the optimal cascade of the function F2, where F2=<0,5,7,2,3,1,4,6>.

The cascade C1○C2' implements the function $F1_v(C2\text{-}0,C0\text{-}1,T1,C1\text{-}2)$. This function is <6,5,4,7,0,1,32> and on the output of the cascade there will be the identical function I.

III. Analysis of the Reversible Circuits

The main target of the analysis of the reversible circuits is to find of the reversible function realized by the given cascade.

Let be given the cascade of the reversible gates:

$$C = G1, G2, \ldots, GN.$$

Lemma 3.

The reversible function F implemented by the C cascade should be calculated as the $I_v(G1,G2,\ldots,GN)$ or as the $I_r(GN,\ldots,G2,G1)$.

Proof

There are the cascade C = G1, G2, …, GN. The invers cascade C'= GN, ….., G2, G1. From Corollary 2 if on the input of the invers cascade C' will be the function I then on the output of the last gate GN will be the function F when we will use the formula:

$$F_{out}(GN) = F_r(F_{in}(GN))$$

From Corollary 3 if on the input of the cascade C will be the function I then on the output of the last gate GN will be the function F when we will use the formula:

$$F_{out}(GN) = F_v(F_{in}(GN))$$

To illustrate the Lemma 3 consider the four gates cascades presented in Fig. 6.

The cascade in Fig. 6a corresponds to the Corollary 1, in Fig. 6b to the Corollary 2, in Fig. 6c to the Corollary 3 and in Fig. 6d to the Corollary 4.

The cascades in Fig. 6b and Fig. 6c transforms the identical function I into the function F according to Lemma 3.

The functions at points marked with the same numbers have the same values.

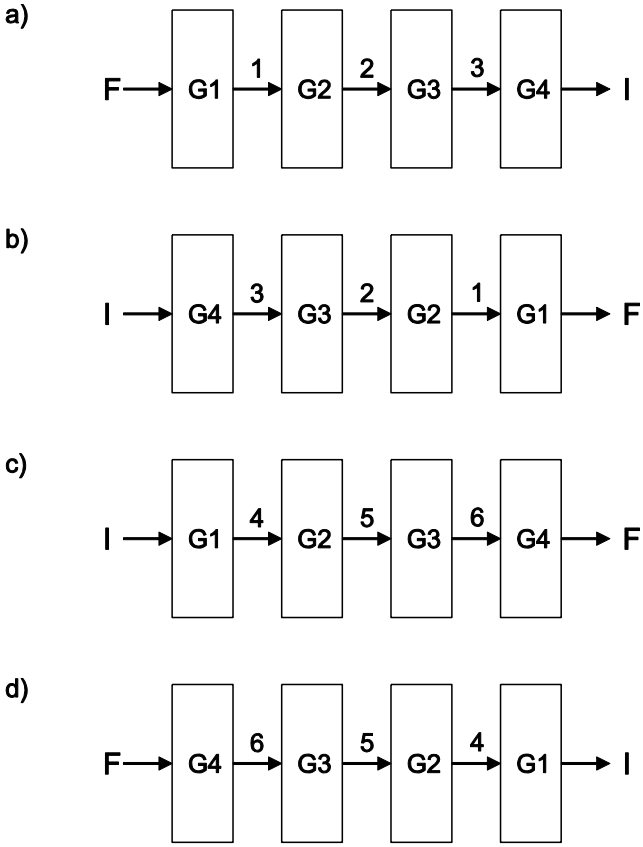

Fig. 6. The four cascades for given function F

IV. TRANSFORMATION METHOD

Let be given the two cascades implementing the function FA with the gates GA1, GA2, GA3, GA4 and the function FB with the gates GB1, GB2, GB3 (Fig. 7). The target is to find an optimal cascade which transforms the function FA into the function FB.

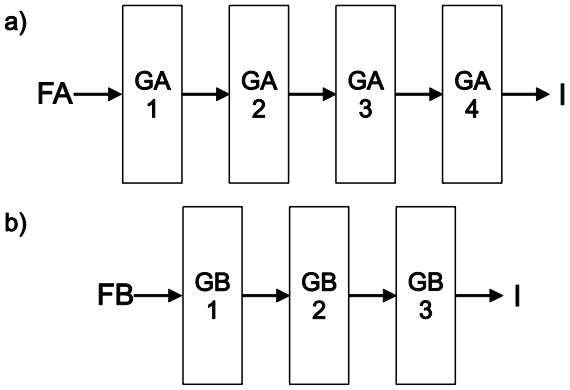

Fig. 7. Two cascades for functions FA and FB

In order to receive the cascade which transforms the function FA into the function FB we should concatenate these two cascades as it is shown in Fig 8.

The cascade from Fig. 8 contains connections between two groups of the gates in bilateral directions. If we use the invers cascade to the cascade with the gates GBi, we obtain the cascade as in Fig. 9. If on the input of the gate GA1 there will be the function FA then on the output of the gate GB3 there will be the function FB.

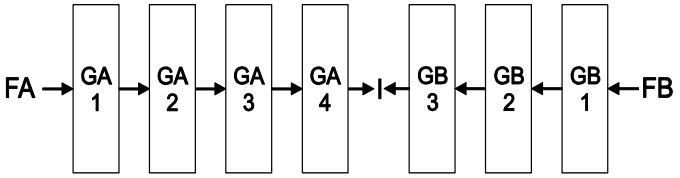

Fig. 8. The cascade transforming functions FA into FB

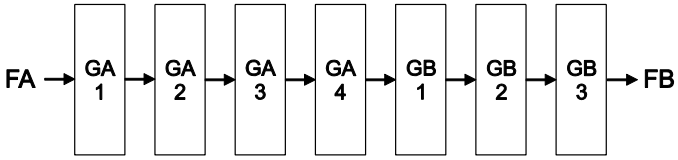

Fig. 9. The cascade from Fig. 4 with output I

In order to receive the function implemented by this cascade must be calculate the function $FA_v(GB1,GB2,GB3)$. If this function is on the input of the gate GA1 (on the cascade from Fig. 9) then on the output of the gate GB3 there will be the identical function I. This cascade is not an optimal cascade for the transformation the function FA into the function FB. But if the optimal cascade implementing the function FA_v is known then this cascade also transforms the function FA into FB.

Example 4.

Let be given two reversible functions:
FA=<0,5,7,2,3,1,4,6> and FB= <4,1,3,6,0,5,2,7>.
The one of the optimal cascades implementing function FA is the cascade C1-2,T1,C0-1,C2-0 and implementing function FB is N2,T0,C2-0.

The first step of the algorithm is the concatenation of the two cascades FA and FB'. There will be built the cascade C1-2,T1,C0-1,C2-0,C2-0,T0,N2. If on the input of this cascade there will be the function FA then on the output of this cascade there will be the function FB (Lemma 2). But this cascade is not optimal to this transformation.

The second step is to calculate the function $FA_v(C2\text{-}0,T0,N2)$. It is the function <4,5,7,6,2,1,0,3>. This function could be implemented by two optimal cascades:

C1=(C1-2,T1,N2,T0) and C2=(T1,C1-2,N2,T0)

This both cascades are the optimal cascades which transforms the function FA into the function FB. The first of these two solutions is shown in Fig. 10.

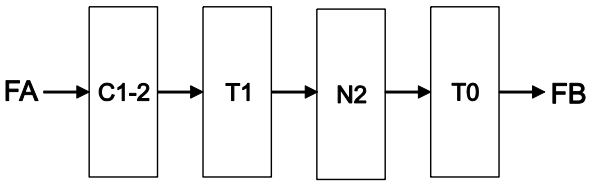

Fig. 10. The optimal cascade for example 4

Example 5.

Let be given the same two functions as in the previous example:
FA=<0,5,7,2,3,1,4,6> and FB=<4,1,3,6,0,5,2,7>.
In this example there will be found the optimal cascade transforming the function FB into the function FA. First we build the concatenation of the two cascades as in Fig. 11.

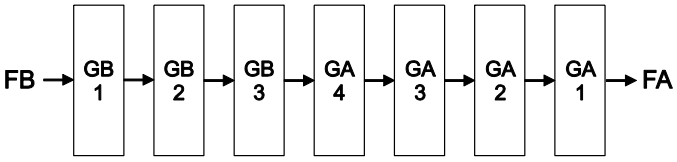

Fig. 11. The new cascade for example 4

The cascade in Fig. 10 transforms the function FB into the function FA. In order to find the optimal cascade we should calculate the function FB_v(GB3,GB2,GB1). It is the function <6,5,4,7,0,1,3,2>. This function could be implemented by two optimal cascades:

C1= (T0,N2,C1-2,T1) and C2=(T0,N2,T1,C1-2)

This both cascades are the optimal cascades which transforms the function FB into the function FA. The first of these two solutions is shown in Fig. 12.

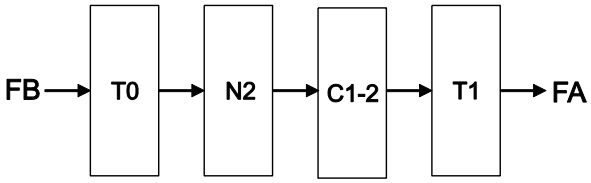

Fig. 12. The new cascade for example 5

Summarizing the presented method of searching for a solution to the problem of finding the optimal cascade transforming the reversible function FX into the other function FY the algorithm contains two steps:

1. Calculation FY_v
2. Indicating the optimal cascade for the function FY_v.

The first step was described above. During this step the designer use the data base with optimal circuits for the given function. This problem is easy to overcome for functions of the three variables. This problem is much more difficult when the number of the variables increase. The same problem must be solved during the second step.

V. CONCLUSIONS

The main aim of this paper is to present the design of optimal reversible cascade which enables implementation of the transformation between two given functions. The presented examples illustrate the algorithm for the synthesis of the reversible functions of the three variables. But this algorithm is scalable for more variables.

In this paper was presented also the method of the reversible circuits analysis. Was showed how to find the reversible function implemented by the given cascade.

REFERENCES

[1] R. Landauer, Irreversibility and heat generation in the computing process. IBM Journal of Research and Development , 5(3):183–191, July 1961.

[2] M. Nielsen, I. Chuang, Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[3] B. Desoete, A. De Vos, M. Sibinski, T. Widerski, Feynman's reversible gates implemented in silicon, 6th International Conference MIXDES, pages 496–502, 1999.

[4] M. Veldhorst, C. H. Yang, J. C. C. Hwang, W. Huang, J. P. Dehollain, J. T. Muhonen, S. Simmons, A. Laucht, F. E. Hudson, K. M. Itoh, A. Morello, A. S. Dzurak, A two-qubit logic gate in silicon, Nature, 526, 410–414, October 2015

[5] P. Picton, Opoelectronic, multivalued, conservative logic, International Journal of Optical Computing, 2:19–29, 1991.

[6] R. C. Merkle, K. E. Drexler, Helical logic, Nanotechnology, 7:325–339, 1996.

[7] E. Fredkin T. Toffoli. Conservative logic. International Journal of Theoretical Physics, 21:219–253, 1982.

[8] R. Feynman. Quantum mechanical computers. Optic News, 11:11–20, 1985.

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

[10] P. Kerntopf, Maximally efficient binary and multi-valued reversible gates, International Workshop on Post-Binary ULSI Systems, pp. 55–58, Warsaw, Poland, May 2001.

[11] K. Iwama, Y. Kambayashi, S. Yamashita, Transformation rules for designing CNOT-based quantum circuits, Design Automation Conference, New Orleans, Louisiana, USA, June 10-14 2002.

[12] D. M. Miller, D. Maslov, W. Dueck, A transformation based algorithm for reversible logic synthesis, Proceedings of the Design Automation Conference, pages 318–323, June 2003.

[13] K. Fazel, M. A. Thornton, J. E. Rice, ESOP-based Toffoli Gate Cascade Generation, Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 206 –209, 2007.

[14] M. H. A. Khan, M. A. Perkowski, Multi-output ESOP Synthesis with Cascades of New Reversible Gate Family, Proc Int. Symp. On Representations and Methodology of Future Comp. Technology, pp.43-55,2003.

[15] R. Wille, R. Drechsler, BDD-based synthesis of reversible logic for large functions, Design Automation Conf. , pp. 270–275, 2009.

[16] M. Hawash, M. Perkowski, N. Alhagi, Synthesis of Reversible Circuits with No Ancilla Bits for Large Reversible Functions, Proc. ISMVL, 2010, p. 1-7.

[17] D. Wang, S. Sun, H. Chen, Matrix-based algorithm for 4-qubit reversible logic circuits synthesis, Energy Procedia, vol. 13, pp. 365-371, 2011.

[18] O. Golubitsky, D. Maslov, A study of optimal 4-bit reversible Toffoli circuits and their synthesis, IEEE Transactions on Computers, vol. 61, no. 9, 2012,. pp. 1341-1353.

[19] A. Khlopotine, M. Perkowski, P. Kerntopf, Reversible logic synthesis by iterative compositions, International Workshop on Logic Synthesis, 2002.

[20] M. Soeken, N. Abdessaied, G. De Micheli, Enumeration of reversible functions and its application to circuit complexity, Conference on Reversible Computation, 2016, 255–270.

[21] P. Gupta, A. Agrawal, N. K. Jha. An algorithm for synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, pp. 2317-2330, 2006.

[22] P. Kerntopf. A new heuristic algorithm for reversible logic synthesis. *ACM*/IEEE *DAC*, pages 834-837, 2004.