www.czasopisma.pan.pl www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 2, PP. 281-286
Manuscript received November 14, 2019; revised April, 2020 DOI: 10.24425/ijet.2020.131875

The Algorithm for Reversible Circuits Synthesis

Andrzej Skorupski, Krzysztof Gracki

***Abstract*—In this paper the new synthesis method for reversible networks is proposed. The method is suitable to generate optimal circuits. The examples will be shown for three variables reversible functions but the method is scalable to larger number of variables. The algorithm could be easily implemented with high speed execution and without big consuming storage software. Section 1 contains general concepts about the reversible functions. In Section 2 there are presented various descriptions of reversible functions. One of them is the description using partitions. In Section 3 there are introduced the cascade of the reversible gates as the target of the synthesis algorithm. In order to achieve this target the definitions of the rest and remain functions will be helpful. Section 4 contains the proposed algorithm. There is introduced a classification of minterms distribution for a given function. To select the successive gates in the cascade the condition of the improvement the minterms distribution must be fulfilled. Section 4 describes the algorithm how to improve the minterms distributions in order to find the optimal cascade. Section 5 shows the one example of this algorithm.**

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

I. Introduction

Logic synthesis of the reversible circuits is the initial step towards synthesis of quantum circuits [1]. Any irreversible logic computation dissipates a certain amount of energy [2]. This amount of heat dissipation will be problematic in the near future since the number of transistors on an integrated circuits is growing exponentially. Reversible logic constitutes a potential solution to this problem. The reversible circuits are able not to dissipate power because there is no information loss [3].

The classic synthesis problem is transformation any description of any function into some circuit implementation. These implementations are the cascades of the reversible gates.

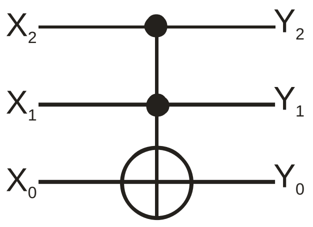


Fig. 1. Example of the reversible gates

The one possible reversible gate is shown on the Fig. 1. The gate has the same input number (on the Fig. 1 three) as output number. On the one of the three lines is XOR gate. One of the input of this gate is connected to input of the gate. The second and third input of the gate could be connect to the another input of the gate or not. These connection methods allow for four variants of the reversible gates with XOR gate on this line.

There are many other types of the reversible gates. In this paper there will be used the NCT set of gates. The problem of the synthesis is to find the cascades which transform the given reversible function F into the identical function I. Furthermore, these cascades could be optimal ea. must contain a minimal number of gates. There are many methods leading to the solution of this problem [4,5]. In this paper will be presented a new algorithm able to execute manually as well as using appropriate software.

II. Reversible functions descriptions

The reversible function can be presented using many type of descriptions. They could be: true table, minterm permutation or hex notation of each Boolean function being components of reversible functions [6]. To ensure reversibility the number of inputs is the same as the outputs number (the number of variables is the same as the number of Boolean functions).

Let be given the reversible function F with three variables. This function could be described using truth table shown in Table I.

TABLE I
TRUTH TABLE OF EXAMPLE OF REVERSIBLE FUNCTION

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

For the three variables X_2, X_1 and X_0 there are defined the three balanced (the same number of zeros and ones) Boolean functions Y_2, Y_1 and Y_0. The set of the output vectors $Y_2Y_1Y_0$ contain all eight input minterms. None of the output vectors repeats. The output vectors are one of the possible minterms permutation.

The second description of this function is the minterm permutation: <4, 0, 3, 2, 7, 6, 5, 1>. It is the sequence of the decimal form of the output vectors $Y_2Y_1Y_0$ as an answer to increasing sequence of input vectors. The permutation <0, 1, 2, 3, 4, 5, 6, 7> will be called the identical function I.

The third description of this function is hex notation of the three functions Y_2, Y_1 and Y_0. For our given function it will be 8E3C2B (8E corresponds with Y_2, 3C with Y_1 and 2B with Y_0) [7].

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

In this paper there will be introduced the fourth description of the reversible function using minterms partitions. For reversible functions with three variables the three partitions will be introduced. Each partition contains four two-elements blocks. Depending on the function the minterms occupy well-defined positions. The three partitions for the identical function I are:

Y_2:{0,4;1,5;2,6;3,7} Y_1:{0,2;1,3;4,6;5,7} Y_0:{0,1;2,3;4,5;6,7}

Left elements in each blocks contain minterms with 0 for function Y_i and right elements contain minterms with 1 for this function. These partitions designate the minterms positions in partitions for any reversible function.

The three partitions for the given function presented in Table I are:

Y_2:{4,7;0,6;3,5;2,1} Y_1:{4,3;0,2;7,5;6,1} Y_0:{4,0;3,2;7,6;5,1}

On the minterms 0 position (the minterm 0 place in partitions of the identical function) is minterm 4. On the minterms 1 position is minterm 0, on the minterms 2 position is minterm 3, on the minterms 3 position is minterm 2, on the minterms 4 position is minterm 7 and so on.

In order to indicate if the minterm *i* of the given function has the opposite bit value as the minterm on the same place in partition of the identical function the minterm *i* will be overlined.

For example, the minterm 4 takes the place of the minterm 0 in the partitions of the identical function. These two minterms differ only on the most significant position so in the corresponding partition Y_2 it will be denoted by over-lining minterm 4. For our example function from Fig. 1 this partitions of the function F will be indicated as below:

Y_2:{$\overline{4}$,7;0,6;3,5;2,$\overline{1}$} Y_1:{4,3;0,2;$\overline{7}$,$\overline{5}$;$\overline{6}$,$\overline{1}$} Y_0:{4,$\overline{0}$;$\overline{3}$,$\overline{2}$;$\overline{7}$,$\overline{6}$;$\overline{5}$,1}

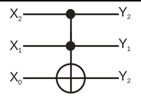
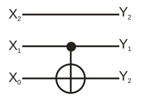
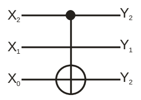
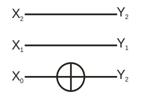
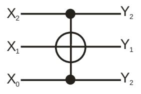
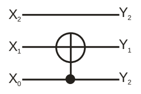
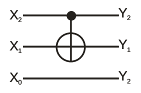
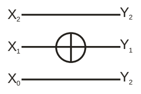
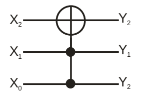
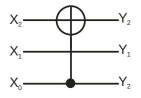
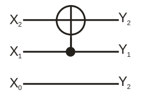
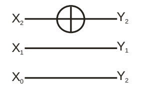
TABLE II
SWAPPED MINTERMS FOR REVERSIBLE GATES

Gate	Swapped positions
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

Each reversible gate transforms the input function into another reversible function. On the outputs of this gate there will be the function with different partitions. The target of the synthesis algorithm is to find the gate sequence which converts the partitions of the given function into partitions of the identical function. In this paper we will present the new method of the reversible functions synthesis using the NCT set of reversible gates. This NCT set for three variable functions contains 12 gates. The one of this gates swaps proper minterms in the partitions of the input function and these new partitions defined the output function of this gate. The names of the gates and appropriate swapped minterms are presented in Table II.

In Table III are showed the diagrams of all gates from the NCT set. The gate C0-1 has the XOR gate on the line X_0 and with the second input connected to X_1. The gate C0-2 has the XOR gate on the line X_0 and with the second input connected to X_2. The gate N0 has the XOR gate on the line X_0 and with the second input connected to X_1 and X_2 (logical AND of the two inputs X_1 and X_2).

TABLE III
THE NCT SET OF THE REVERSIBLE GATES

Gate	Swapped positions
T0	
C0-1	
C0-2	
N0	
T1	
C1-0	
C1-2	
N1	
T2	
C2-0	
C2-1	
N2	

The right column in Table II presents the gates operations. Each gate swaps appropriate positions indicated by the minterms positions of the identical function. For example the gate T0 swaps positions occupied by minterms 6 and 7 of the identical function. For our example function the gate T0 swaps minterms 5 and 1 because these minterms occupy the 6 and 7 positions of the identical function. These minterms 5 and 1 will be swapped in all partitions. The output function of the gate T0 will be the function F(T0) with partitions as below:

Y_2:{$\bar{4}$,7;0,6;3,$\bar{1}$;2,5} Y_1:{4,3;0,2;$\bar{7}$,$\bar{1}$;$\bar{6}$,$\bar{5}$} Y_0:{4,$\bar{0}$;$\bar{3}$,$\bar{2}$;$\bar{7}$,$\bar{6}$;$\bar{1}$,5}

The partition corresponding to Y_1 could be ordered by the gate C1-2 (both minterms in two last blocks with over-lining minterms will be swapped by this gate).

III. Reversible functions implementation

The implementations of the reversible functions are the cascades of the reversible gates presented in . Fig. 2. There are two types of the cascades.

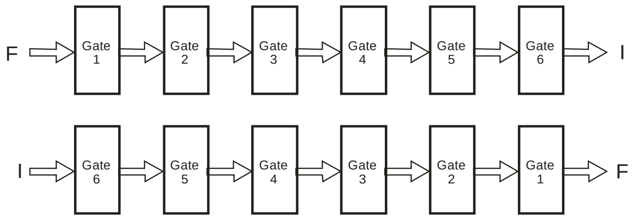


Fig. 2. Examples of the cascades with 6 reversible gates

The cascade in Fig. 2a (first type cascade) transforms function F into identical function I. The cascade in Fig. 2b (second type cascade) transforms function I into the given function F. Both the cascades contain the same gates but the order of the gates in both cascades is reversed. In the presented algorithm there will be used concepts of the rest function and the remain function.

Definition 1: The output function $F_r(G_i)$ of the gate G_i will be called the rest function of the input function F.

The gate G_i which swaps the corresponding minterms (as in Table II) gives the new function denoted $F_r(G_i)$. For the first type cascade the rest function $F_r(G_1)$ is realized by the gates G_2 to G_6. The rest function $F_r(G_6)$ is the identical function. The gate G_6 in the first type cascade swapps the appropriate positions to receive on the output the identical function. The identical function determines these positions by their own minterm values.

How to determine the function realized by the gates G_1 to G_5? It will be calculated by swapping the same minterms of the function F as the minterms swapped by the gate G_6.

Definition 2: The input function $F_v(G_i)$ of the gate G_i will be called the remain function if on the output of the gate G_i is the function F.

To determine the function $F_v(G_i)$ the minterms values corresponding to the given gate G_i. should be swapped.

Example 1: The remainder function $F_v(G_6)$ of the function from Table I if the gate G_6 is T0 is:

Y_2:{$\bar{4}$,6;0,7;3,5;2,$\bar{1}$} Y_1:{4,3;0,2;$\bar{6}$,$\bar{5}$; $\bar{7}$,$\bar{1}$} Y_0:{4,$\bar{0}$; $\bar{3}$,$\bar{2}$;6,7;$\bar{5}$,1}

In all partitions the minterms 6 and 7 are swapped (see Table II).

Definition 3: The Boolean function Y_i is ordered when corresponding partition has in all blocks minterm “0” on the left side of the block and “1” on the right side of the block.

From definition 3 the partition corresponding to Boolean function Y_i has all blocks without over-lined elements. The Boolean function Y_1 of the given reversible function F is ordered when the front gate of the first type of cascade G_1 will be the gate C1-2. The gate C1-2 swaps positions of the identical function 4 with 6 and 5 with 7. On the positions 4 and 6 in partition Y_1 there are minterms 7 and 5 and on the positions 5 and 7 there are minterms 6 and 1. They will be swapped by the gate C1-2. The rest function $F_r(G_1)$ will be:

Y_2:{$\bar{4}$,5;0,$\bar{1}$;3,7;2,6} Y_1:{4,3;0,2;5,7;1,6} Y_0:{4,$\bar{0}$; $\bar{3}$,$\bar{2}$; $\bar{5}$,1;$\bar{7}$,$\bar{6}$}

The Boolean function Y_1 is ordered.

There are 576 (4!×4!=24x24) ordered partitions for each Boolean functions Y_i. One of them is the identical function:

Y_2:{0,4;1,5;2,6;3,7} Y_1:{0,2;1,3;4,6;5,7} Y_0:{0,1;2,3;4,5;6,7}

because every function Y_i is ordered.

All Boolean functions Y_2, Y_1, Y_0 used in the reversible function are the balanced functions. Hence there exist $\binom{8}{4} = 70$ various combinations of the minterms distributions with and without over-lining. One of these combinations corresponds to the ordered partition (all blocks are without over-lines).

There is another combination where all blocks contain both over-lined elements. For ordering this partition one gate N_i is needed.

There are 14 other combinations with blocks containing both elements over-lined or without over-line: 4 combinations with one block containing both elements over-lined, 6 combinations with two blocks containing both elements over-lined and 4 combinations with three blocks containing both elements over-lined.

Lemma 1. The partitions containing one from 15 above combinations could be ordered by the string of the gates with XOR on this line. This string could be one-, two-, three- or four-goal string gates.

Proof.
Four combinations: {$x,x;\bar{x},\bar{x};x,x;\bar{x},\bar{x}$}, {$x,x;x,x;\bar{x},\bar{x};\bar{x},\bar{x}$}, {$x,x;x,x;x,x;\bar{x},\bar{x}$}, {$\bar{x},\bar{x}; \bar{x},\bar{x}; \bar{x},\bar{x};\bar{x},\bar{x}$} could be ordering by one gate. Six combinations {$x,x;x,x;\bar{x},\bar{x};x,x$}, {$x,x;\bar{x},\bar{x};x,x;x,x$}, {$x,x;\bar{x},\bar{x};\bar{x},\bar{x};x,x$}, {$\bar{x},\bar{x};x,x;\bar{x},\bar{x};x,x$}, {$\bar{x},\bar{x};\bar{x},\bar{x};x,x;x,x$}, {$\bar{x},\bar{x}; \bar{x},\bar{x}; \bar{x},\bar{x};x,x$} could be ordering by two gates. Four combinations {$\bar{x},\bar{x};x,x;x,x;\bar{x},\bar{x}$}, {$x,x;\bar{x},\bar{x};\bar{x},\bar{x};\bar{x},\bar{x}$}, {$\bar{x},\bar{x};x,x;\bar{x},\bar{x};\bar{x},\bar{x}$}, {$\bar{x},\bar{x};\bar{x},\bar{x};x,x;\bar{x},\bar{x}$} could be ordering by three gates. One combination {$\bar{x},\bar{x};x,x;x,x;x,x$} could be ordered by four gates.

In Table III there are introduced the binary partitions corresponding to all gates. These partitions indicate the positions swapped by corresponding gate. The given gate swaps the minterms on the positions indicated by the same numbers in the partition but different from zero.

TABLE III
SWAPPED MINTERMS POSITIONS FOR REVERSIBLE GATES

Gate	Swapped positions
T0	X_2:{0,0;0,0;0,1;0,1} X_1:{0,0;0,0;0,1;0,1} X_0:{0,0;0,0;0,0;1,1}
C0-1	X_2:{0,0;0,0;2,1;2,1} X_1:{0,2;0,2;0,1;0,1} X_0:{0,0;2,2;0,0;1,1}
C0-2	X_2:{0,2;0,2;0,1;0,1} X_1:{0,0;0,0;2,1;2,1} X_0:{0,0;2,2;0,0;1,1}
N0	X_2:{4,2;4,2;3,1;3,1} X_1:{4,3;4,3;2,1;2,1} X_0:{4,4;3,3;2,2;1,1}
T1	X_2:{0,0;0,1;0,0;0,1} X_1:{0,0;0,0;0,0;1,1} X_0:{0,0;0,0;0,1;0,1}
C1-0	X_2:{0,0;2,1;0,0;2,1} X_1:{0,0;2,2;0,0;1,1} X_0:{0,2;0,2;0,1;0,1}
C1-2	X_2:{0,2;0,1;0,2;0,1} X_1:{0,0;0,0;2,2;1,1} X_0:{0,0;0,0;2,1;2,1}
N1	X_2:{4,4;3,1;2,2;3,1} X_1:{4,4;3,3;2,2;1,1} X_0:{4,3;4,3;2,1;2,1}
T2	X_2:{0,0;0,0;0,0;1,1} X_1:{0,0;0,1;0,0;0,1} X_0:{0,0;0,1;0,0;0,1}
C2-0	X_2:{0,0;2,2;0,0;1,1} X_1:{0,0;2,1;0,0;2,1} X_0:{0,2;0,1;0,2;0,1}
C2-1	X_2:{0,0;0,0;2,2;1,1} X_1:{0,2;0,0;0,2;1,1} X_0:{0,0;2,1;0,0;2,1}
N2	X_2:{4,4;3,3;2,2;1,1} X_1:{4,2;1,1;4,2;3,3} X_0:{4,3;2,1;4,3;2,1}

Let consider the function F:

$Y_2:\{\bar{4},7;0,6;3,5;2,\bar{1}\}$ $Y_1:\{4,3;0,2;\bar{7},\bar{5};\bar{6},\bar{1}\}$ $Y_0:\{4,\bar{0};\bar{3},\bar{2};\bar{7},\bar{6};\bar{5},1\}$

To calculate the partitions of the rest function we use the operation indicated by symbol • facilitating this operation. For example the partitions of the rest function F_r(T0) of the given function F are:

$Y_2:\{\bar{4},7;0,6;3,5;2,\bar{1}\}\bullet\{0,0;0,0;0,1;0,1\}=\{\bar{4},7;0,6;3,\bar{1};2,5\}$

$Y_1:\{4,3;0,2;\bar{7},\bar{5};\bar{6},\bar{1}\}\bullet\{0,0;0,0;0,1;0,1\}=\{4,3;0,2;\bar{7},\bar{1};\bar{6},\bar{5}\}$

$Y_0:\{4,\bar{0};\bar{3},\bar{2};\bar{7},\bar{6};\bar{5},1\}\bullet\{0,0;0,0;0,0;1,1\}=\{4,\bar{0};\bar{3},\bar{2};\bar{7},\bar{6};\bar{1},5\}$

The gates with XOR on line Y_i swap minterms in the same block in the partition Y_i. Then these minterms in this block change the over-line situation: if the minterm was with over-line then it would be without over-line and if it was without over-line then it would be with over-line. A different situation is on the remaining lines. The minterms in different blocks are swapped between these blocks without any changes in the over-line situation.

IV. Algorithm

The proposed algorithm based on the partitions analysis to evaluate the minterms distributions. The algorithm indicates the gates set containing the best gates leading to ordering any function Y_i. This set will be called the front best gates FBG (front gate is the first gate in cascade). Is possible to appoint the closing best gates CBG (the closing gate is the last gate in the cascade).

The target of the analysis is to find the shortest way (minimal gates number) to sequentially ordering all the partitions. The algorithm starts for the given function and verifies the partitions if there exist one gate ordering any line. If it exist it will be the first member of FBG set. The next step of the algorithm is the calculations of the rest functions for every gate from NCT set. For each the rest functions must be verified if there exist the partition ordered by the string of the gates (one-, two-, three- or four-gates). If it exist will be the member of FBG set.

Lemma 2
The last gate G_i in the first type of the cascade ordering one of the lines Y_i is the gate with XOR on line Y_i.

Proof:
The gates with XOR on line Y_i swap the minterms in one block (see Table III). If on the output of the gate the minterms distribution is ordered then on the input of this gate the minterm distribution on this line contains over-lining block corresponding with this gate.

In each step of the algorithm the target of the analysis can be the series of the gates ordering the individual lines. In order to do it the minterms distribution should be transformed so as to receive the partitions only with blocks where both minterms are over-lined.

Lemma 3
The sequence of the gates with XOR on the same line Y_i can be put in the cascade in freely order.

Proof:
Each of the gates with XOR on line Y_i swaps the minterms only inside the given blocks. Swapping minterms inside the blocks in the same partition could be done in any order.

TABLE IV
THE DISTRIBUTIONS

Gates numaber	Minterm distributions
1	{0,0;0,0;0,0;1,1}
	{0,0;0,0;2,2;1,1}
	{0,0;2,2;0,0;1,1}
	{4,4;3,3;2,2;1,1}
2	{0,0;0,0;1,1;0,0}
	{0,0;1,1;0,0;0,0}
	{0,0;2,2;1,1;0,0}
	{2,2;0,0;1,1;0,0}
	{2,2;1,1;0,0;0,0}
	{1,1;2,2;3,3;0,0}
3	{2,2;0,0;0,0;1,1}
	{0,0;1,1;2,2;3,3}
	{1,1;0,0;2,2;3,3}
	{1,1;2,2;0,0;3,3}
4	{1,1;0,0;0,0;0,0}

The synthesis problem can be reduced to a problem of transforming the given minterms distribution to one of the presented by Lemma 1. To solve the problem of ordering the partitions two criterions will be introduced:

1. If one (or more) partition of the given function could be ordered by the string one-, two-, three- or four-gates the first gate of this string will be the member of FBG set.
2. If one (or more) partition of the rest function $F_r(G_i)$ could be ordered by the string one-, two-, three- or four-gates the first gate of this string will be the member of FBG set.

The minterms distribution is better if it requires less gates to order some partition.

The above algorithm was presented under assumption non empty FBG set. If his set is empty the designer could change the given function F by determination the last gate in the cascade. This is possible by using the remain function. For given function F should be calculated the remain functions $F_v(G_i)$ for all reversible gates G_i and create the CBG set (closing best gates set). If the gate G_i is a member of CBG set than for the function $F_v(G_i)$ should be finding the FBG set. This method will be illustrated in next section.

V. Example

Let consider the example function F=<4, 0, 3, 2, 7, 6, 5, 1>.
The partitions of the given function are:

Y_2:$\{\bar{4},7;0,6;3,5;2,\bar{1}\}$ Y_1:$\{4,3;0,2;\bar{7},\bar{5};\bar{6},\bar{1}\}$ Y_0:$\{4,\bar{0};\bar{3},\bar{2};\bar{7},\bar{6};\bar{5},1\}$

Criterion 1: The partition Y_1 could be ordered by the gate C1-2. This gate will be the member of the FBG.

Criterion 2. Below 12 rest functions will be calculated:

F_r(T0) =

$Y_2\{\bar{4},7;0,6;3,\bar{1};2,5\}$ $Y_1\{4,3;0,2;\bar{7},\bar{1};\bar{6},\bar{5}\}$ $Y_0\{4,\bar{0};\bar{3},\bar{2};\bar{7},\bar{6};\bar{1},5\}$

partition Y_1 could be ordered by one gate, the gate T0 will be the member of the FBG.

F_r(C0-1) =

$Y_2\{\bar{4},7;0,6;2,\bar{1};3,5\}$ $Y_1\{4,2;0,3;\bar{7},\bar{1};\bar{6},\bar{5}\}$ $Y_0\{4,\bar{0};2,3;\bar{7},\bar{6};\bar{1},5\}$

partition Y_1 could be ordered by one gate, the gate C0-1 will be the member of the FBG.

F_r(C0-2) =

$Y_2\{\bar{4},6;0,7;3,\bar{1};2,5\}$ $Y_1\{4,3;0,2;\bar{6},\bar{1};\bar{7},\bar{5}\}$ $Y_0\{4,\bar{0};\bar{3},\bar{2};6,7;\bar{1},5\}$

partition Y_1 could be ordered by one gate, the gate C0-2 will be the member of the FBG.

F_r(N0) =

$Y_2\{0,6;\ \bar{4},7;2,\bar{1};3,5\}$ $Y_1\{0,2;4,3;\bar{6},\bar{1};\bar{7},\bar{5}\}$ $Y_0\{0,\bar{4};2,3;6,7;\bar{1},5\}$

partition Y_1 could be ordered by one gate, the gate N0 will be the member of the FBG.

F_r(T1) =

$Y_2\{\bar{4},7;0,\bar{1};3,5;2,6\}$ $Y_1\{4,3;0,2;\bar{7},\bar{5};1,6\}$ $Y_0\{4,\bar{0};\bar{3},\bar{2};\bar{7},1;\bar{5},\bar{6}\}$

partition Y_1 could be ordered by two gates, the gate T1 will be the member of the FBG.

F_r(C1-2) =

$Y_2\{\bar{4},5;0,\bar{1};3,7;2,6\}$ $Y_1\{4,3;0,2;5,7;1,6\}$ $Y_0\{4,\bar{0};\bar{3},\bar{2};\bar{5},1;\bar{7},\bar{6}\}$

partition Y_1 is ordered, the gate C1-2 will be the member of the FBG.

F_r(C1-0) =

$Y_2\{\bar{4},7;2,\bar{1};3,5;0,6\}$ $Y_1\{4,3;\bar{2},\bar{0};\bar{7},\bar{5};1,6\}$ $Y_0\{4,\bar{2};\bar{3},\bar{0};\bar{7},1;\bar{5},\bar{6}\}$

partition Y_1 could be ordered, by two gates, C1-0 will be the member of the FBG.

F_r(N1) =

$Y_2\{3,5;2,\bar{1};\bar{4},7;0,6\}$ $Y_1\{\bar{3},\bar{4};\bar{2},\bar{0};5,7;1,6\}$ $Y_0\{\bar{3},\bar{2};4,\bar{0};\bar{5},1;\bar{7},\bar{6}\}$

partition Y_1 could be ordered, by two gates, N1 will be the member of the FBG.

F_r(T2) =

$Y_2\{\bar{4},7;0,6;3,5;1,\bar{2}\}$ $Y_1\{4,3;0,\bar{1};\bar{7},\bar{5};\bar{6},2\}$ $Y_0\{4,\bar{0};\bar{3},1;\bar{7},\bar{6};\bar{5},\bar{2}\}$

the gate T2 does not be the member of the FBG.

F_r(C2-0) =

$Y_2\{\bar{4},7;\bar{6},\bar{0};3,5;1,\bar{2}\}$ $Y_1\{4,3;\bar{6},\bar{1};\bar{7},\bar{5};0,2\}$ $Y_0\{4,\bar{6};\bar{3},1;\bar{7},\bar{0};\bar{5},\bar{2}\}$

partition Y_1 could be ordered, by two gates, C2-0 will be the member of the FBG.

F_r(C2-1) =

$Y_2\{\bar{4},7;0,6;\bar{5},\bar{3};1,\bar{2}\}$ $Y_1\{4,\bar{5};0,\bar{1};\bar{7},3;\bar{6},2\}$ $Y_0\{4,\bar{0};\bar{5},1;\bar{7},\bar{6};\bar{3},\bar{2}\}$

the gate C2-1 does not be the member of the FBG.

F_r(N2) =

$Y_2\{\bar{7},4;\bar{6},\bar{0};\bar{5},\bar{3};1,\bar{2}\}$ $Y_1\{\bar{7},\bar{5};\bar{6},\bar{1};4,3;0,2\}$ $Y_0\{\bar{7},\bar{6};\bar{5},1;4,\bar{0};\bar{3},\bar{2}\}$

partition Y_1 could be ordered, by two gates, N2 will be the member of the FBG.

The FBG set contains 10 gates. Only the gates T2 and C2-1 are not the members of the FBG set. The algorithm divide out into 10 branches. The sequence of branches analysis depends on the number of the gates that order the partition.
In this example there are:
1. C1-2 gate because this gate order partition.
2. The rest functions of the gates T0, C0-1, C0-2 and N0 could be ordered by one gate.
3. The rest functions of the gates T1,C1-0, N1, C2-0 and N2 could be ordered by two gates.

The first **branch no. 1** of the algorithm starts with the rest function F_r(C1-2) ea. the first gate in cascade is the gate C1-2. In this branch the algorithm is repeated ea. the FBG set for the rest function F_r(C1-2) should be appointed.
The partitions for this function are:

Y_2:$\{\bar{4},5;0,\bar{1};3,7;2,6\}$ Y_1:$\{4,3;0,2;5,7;1,6\}$ Y_0:$\{4,\bar{0};\bar{3},\bar{2};\bar{5},1;\bar{7},\bar{6}\}$

The calculation of the rest functions gives the FBG set containing only two gates: C2-0 and C0-2.

F_r(C2-0) =

$Y_2\{\bar{4},5;1,\bar{0};3,7;\bar{6},\bar{2}\}Y_1\{4,3;1,6;5,7;0,2\}$ $Y_0\{4,1;\bar{3},\bar{6};\bar{5},\bar{0};\bar{7},\bar{2}\}$

The rest function F_r(C2-0) could be ordered by three gates and the gate C2-0 is the member of the FBG set.

F_r(C0-2) =

$Y_2\{\bar{4},\bar{1};0,5;3,6;2,7\}Y_1\{4,3;0,2;1,6;5,7\}$ $Y_0\{4,\bar{0};\bar{3},\bar{2};\bar{1},5;6,7\}$

The rest function F_r(C0-2) could be ordered by four gates and the gate C0-2 is the member of the FBG set. This branch is divided out for next two subbranches.
During the next step the functions F(C1-2,C2-0) (branch 1.1) and F(C1-2,C0-2) (branch 1.2) will be analysed.

Branch 1.1

The function F(C1-2,C2-0) has partition Y_2 requiring three gates for ordering. The gates C0-2, C0-1 and T0 are the members of the FBG set.

The FBG set of the rest functions F_r(C1-2,C2-0) contains five gates: C2-1, T2, C0-2, C0-1 and T0.

Hence the result of the functions F(C1-2,C2-0,C0-2), F(C1-2,C2-0,C0-1) and F(C1-2,C2-0,T0) could be ordered by the two gates. But the algorithm found the rest function F_r(C1-2,C2-0,C0-2,N2) could be ordered by one gate T0.

The function F(C1-2,C2-0,C0-2,N2,T0) could be ordered by the gate C2-0. The function F(C1-2,C2-0,C0-2,N2,T0,C2-0) is the identical function and the six gates cascade was found. The remaining subbranches gives longer cascades.

Branch 1.2

The function F(C1-2,C0-2) has partition Y_2 requiring four gates to ordering. The gates N2, C2-0, C2-1 and T2 are the members of the FBG set. The rest function F_r(C1-2,C0-2) gives also the gate N0 as the member of FBG set because the partitions Y_2 of the function F(C1-2,C0-2,N0) could be ordered by two gates C2-0 and T2. The functions F(C1-2,C0-2,N0,C2-0,T2) and F(C1-2,C0-2,N0,T2,C2-0) have only one member of FBG set and it is the gate C0-2. The functions F(C1-2,C0-2,N0,C2-0,T2,C0-2) and F(C1-2,C0-2,N0,T2,C2-0,C0-2) are the identical functions and two cascades with six gates were found.

From Lemma 2 we can add the next two cascades:
C1-2,N0,C0-2,T2,C2-0,C0-2
C1-2,N0,C0-2,C2-0,T2,C0-2.

In this branch of the algorithm when it starts from the gate C1-2 we found five optimal cascades with six gates each. But the set of cascades with the gate C1-2 as the first gate in the cascade contains two cascades more. We can find these two solutions during the second type of cascade analysis.

The remain function

Let we try to find the remain functions for all gates although the LBG set for the given function is not empty.
The partitions of given function are:

Y_2:{$\bar{4}$,7;0,6;3,5;2,$\bar{1}$} Y_1:{4,3;0,2;$\bar{7}$,$\bar{5}$;$\bar{6}$,$\bar{1}$} Y_0:{4,$\bar{0}$;$\bar{3}$,$\bar{2}$;$\bar{7}$,$\bar{6}$;$\bar{5}$,1}

As was shown from the definition 2 the partitions of the remain functions F_v(C2-0) are:

Y_2:{$\bar{4}$,$\bar{3}$;0,6;$\bar{7}$,$\bar{1}$;2,5} Y_1:{4,7;0,2;$\bar{3}$,$\bar{1}$;$\bar{6}$,$\bar{5}$} Y_0:{4,$\bar{0}$;$\bar{7}$,$\bar{2}$;$\bar{3}$,$\bar{6}$;$\bar{1}$,5}

Two gates C2-0 and N2 can ordered the Y_2 partition.
The partitions of the remain function F_v(C0-2) are:

Y_2:{$\bar{5}$,6;0,7;3,4;2,$\bar{5}$} Y_1:{5,3;0,1;$\bar{6}$,$\bar{4}$;$\bar{7}$,$\bar{1}$} Y_0:{$\bar{5}$,$\bar{0}$;$\bar{3}$,$\bar{2}$;6,7;4,1}

Two gates C0-2 and N0 can ordered the Y_0 partition.
These two gates C2-0 and C0-2 are the members of the CBG set.

Let consider the gate C2-0 as the last gate in the cascade. Now the function F_v(C2-0)=F_n should be design. The members of the FBG set are the gates C2-0, N0 and C1-2. The algorithm divide out into 3 branches. In first branch we will take the gate C2-0 as the first gate in this cascade. This cascade is presented on Fig. 3.

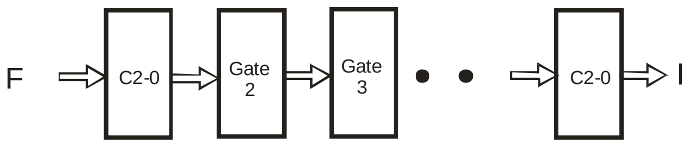


Fig. 3 The cascade with the gate C2-0 as the last gate

We consider the cascade with the gate C2-0 in the first and last positions. The FBG set for the function F_n(C2-0) should be appointed. This set contain three gates:

- the gate N2 could ordered the line Y_2,
- the gates C1-2 and C0-2 (these gates together with the gate C1-0 could ordered the line Y_2.

If we take the gates N2 or C1-2 we obtain the 7 gates cascade. Only the branch with the gates C0-2 leads to the cascades with 6 gates. Finally we obtain two optimal cascades:

C2-0,C0-2,N2,C1-0,T0,C2-0
C2-0,C0-2,C1-0, N2,T0,C2-0

In this example was shown a few branches of the algorithm and was found 7 optimal cascades. The remaining branches of the algorithm give us the rest 17 cascades. Our given function has 23 optimal cascades contain 6 gates each.

VI. Conclusion

The main aim of this paper was presentation of the design of optimal reversible cascades for the three variables reversible functions. The target of this work was developing such method of the reversible function design which allows the “manually” design. The software implementation of this algorithm gives the speed program execution and low storage consuming. This algorithm was presented for the synthesis of the reversible functions of the three variables. But this algorithm is scalable for more variables.

References

[1] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes. Reversible logic circuit synthesis. In IEEE/ACM International Conference on Computer Aided Design (ICCAD), pages 353–360, 2002. November 10–14, San Jose, CA, USA, ACM.

[2] C.H. Bennett, Logical Reversibility of Computation, IBM Journal Reaserch and Developmet, Nov. 1973

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

[4] A. De Vos, B. Raa, L. Storme, Generating the group of reversible logic gates, J. Phys. A: Math. Gen. 35 (2002), 7063–7078

[5] O. Golubitsky and 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.

[6] D. M. Miller, D. Maslov, G. W. Dueck, A transformation based algorithm for reversible logic synthesis, in: Design Automation Conf., 2003, pp. 318–323.

[7] P. Kerntopf, A new heuristic algorithm for reversible logic synthesis, Design Automation Conf., 2004, pp. 834–837.

[8] C. Bandyopadhyay, H. Rahaman, R. Drechsler, “A Cube Pairing Approach for Synthesis of ESOP-Based Reversible Circuit,” *Proceedings of the IEEE International Symposium on Multiple-Valued Logic*, pp. 109-114, May 19-21, 2014

[9] C. S. Cheng, A. K. Singh, “Heuristic Synthesis of Reversible Logic - A Comparative Study”, *Advances in Electrical and Electronic Engineering*, vol. 12, no. 3, pp. 210-225, September 2014

[10] M. Krishna, An. Chattopadhyay, “Efficient Reversible Logic Synthesis via Isomorphic Subgraph Matching”, *Proceedings of the IEEE International Symposium on Multiple-Valued Logic*, pp. 103-108, May 19-21, 2014

[11] C.-C. Lin, N. K. Jha, “RMDDS: Reed-Muller Decision Diagram Synthesis of Reversible Logic Circuits”, *ACM Journal on Emerging Technologies in Computing Systems*, vol. 10, no. 2, pp. 14:1–14:25, February 2014

[12] S. J. Roy, K. Datta, C. Bandyopadhyay, H. Rahaman, “A Transformation Based Heuristic Synthesis Approach for Reversible Circuits”, *Proceedings of the International Conference on Advances in Electrical Engineering*, pp. 1-5, January 2014.

[13] E. Schönborn, K. Datta, R. Wille, I. Sengupta, H. Rahaman, R. Drechsler: “BDD-based Synthesis for All-optical Mach-Zehnder Interferometer Circuits”, *International Conference on VLSI Design*, 2015

[14] A. Skorupski, Graphical Method of Reversible Circuits Synthesis, IJET, Vol 63, No 3, 2017.

[15] E. F. Fredkin, T. Toffoli, Conservative logic, International Journal of Theoretical Physics 21 (1982), pp. 219–253.

[16] P. Gupta, A. Agrawal, N. Jha, An algorithm for synthesis of reversible logic circuits, IEEE Trans. on CAD 25 (2006), pp. 2317–2330

[17] D. M. Miller, D. Maslov, G. W. Dueck, A transformation based algorithm for reversible logic synthesis, in: Design Automation Conf., 2003, pp. 318–323.