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 optional 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 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](image)

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>
<thead>
<tr>
<th>No.</th>
<th>(X_2X_1X_0)</th>
<th>(Y_1Y_0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>000</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>000</td>
</tr>
</tbody>
</table>

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).

© The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/), which permits use, distribution, and reproduction in any medium, provided that the Article is properly cited.
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:

\[ Y_1: \{0, 4, 1, 5 \} \quad Y_2: \{0, 2, 1, 3 \} \quad Y_3: \{0, 6, 7, 2, 3 \} \]

Left elements in each blocks contain minterms with 0 for function \( Y_1 \) 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:

\[ Y_1: \{4, 7, 0, 6, 3, 5, 2, 1\} \quad Y_2: \{4, 3, 0, 2, 7, 5, 6, 1\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\} \]

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 over-lined.

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:

\[ Y_1: \{4, 7, 0, 6, 3, 5, 2, 1\} \quad Y_2: \{4, 3, 0, 2, 7, 5, 6, 1\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\} \]

### TABLE II

<table>
<thead>
<tr>
<th>Gate</th>
<th>Swapped positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>6, 7</td>
</tr>
<tr>
<td>C0-1</td>
<td>2, 3 &amp; 6, 7</td>
</tr>
<tr>
<td>C0-2</td>
<td>4, 5 &amp; 6, 7</td>
</tr>
<tr>
<td>N0</td>
<td>0, 1 &amp; 2, 3 &amp; 4, 5 &amp; 6, 7</td>
</tr>
<tr>
<td>T1</td>
<td>5, 7</td>
</tr>
<tr>
<td>C1-0</td>
<td>1, 3 &amp; 5, 7</td>
</tr>
<tr>
<td>C1-2</td>
<td>4, 6 &amp; 5, 7</td>
</tr>
<tr>
<td>N1</td>
<td>0, 2 &amp; 1, 3 &amp; 4, 6 &amp; 5, 7</td>
</tr>
<tr>
<td>T2</td>
<td>3, 7</td>
</tr>
<tr>
<td>C2-0</td>
<td>1, 5 &amp; 3, 7</td>
</tr>
<tr>
<td>C2-1</td>
<td>2, 6 &amp; 3, 7</td>
</tr>
<tr>
<td>N2</td>
<td>0, 4 &amp; 1, 5 &amp; 2, 6 &amp; 3, 7</td>
</tr>
</tbody>
</table>

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 define 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

<table>
<thead>
<tr>
<th>Gate</th>
<th>Swapped positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C0-1</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C0-2</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>N0</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>T1</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C1-0</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C1-2</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>N1</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>T2</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C2-0</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>C2-1</td>
<td>( x \rightarrow y )</td>
</tr>
<tr>
<td>N2</td>
<td>( x \rightarrow y )</td>
</tr>
</tbody>
</table>

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 \( T_0 \) will be the function \( F(T_0) \) with partitions as below:

\[
Y_{\phi} = \{4,7,0,6,3,\bar{1},2,5\} \quad Y_{\phi} = \{4,3,0,2,7,\bar{1},6,5\} \quad Y_{\phi} = \{4,0,3,2,7,6,\bar{1},5\}
\]

The partition corresponding to \( Y_1 \) could be ordered by the gate \( C_1 \rightarrow 2 \) (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](image)

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 \( F_i(G_i) \) 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 \( F_i(G_i) \). For the first type cascade the rest function \( F_i(G_i) \) is realized by the gates \( G_i \rightarrow G_2 \). The rest function \( F_i(G_i) \) is the identical function. The gate \( G_i \) in the first type cascade swaps 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 \( G_1 \rightarrow G_2 \)? It will be calculated by swapping the same minterms of the function \( F \) as the minterms swapped by the gate \( G_0 \).

**Definition 2:** The input function \( F_i(G_i) \) 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 \( F_i(G_i) \) the minterms values corresponding to the given gate \( G_i \) should be swapped.

**Example 1:** The remainder function \( F_i(G_i) \) of the function from Table I if the gate \( G_0 \) is \( T_0 \) is:

\[
Y_{\phi} = \{4,6,0,7,3,5,2,\bar{1}\} \quad Y_{\phi} = \{4,3,0,2,6,\bar{5},,7,\bar{1}\} \quad Y_{\phi} = \{4,0,3,2,6,\bar{7},,\bar{5},1\}
\]

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

**Definition 3:** The Boolean function \( Y_1 \) 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 \( Y_1 \), 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 \( G_0 \) is the gate \( C_1 \rightarrow 2 \). The gate \( C_1 \rightarrow 2 \) 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 \( C_1 \rightarrow 2 \). The rest function \( F_i(G_i) \) will be:

\[
Y_{\phi} = \{4,5,0,\bar{1},3,7,2,6\} \quad Y_{\phi} = \{4,3,0,2,5,7,1,6\} \quad Y_{\phi} = \{4,0,\bar{3},2,\bar{5},1,7,6\}
\]

The Boolean function \( Y_1 \) is ordered.

There are 576 (4!×4!=24×24) ordered partitions for each Boolean functions \( Y_1 \). One of them is the identical function:

\[
Y_{\phi} = \{0,4,1,5,2,6,3,7\} \quad Y_{\phi} = \{0,2,1,3,4,6,5,7\} \quad Y_{\phi} = \{0,1,2,3,4,5,6,7\}
\]

because every function \( Y_1 \) is ordered.

All Boolean functions \( Y_2, Y_1, Y_0 \) used in the reversible function are the balanced functions. Hence there exist \( \frac{1}{4} = 70 \) 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 \( N_1 \) 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: \( \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\} \) could be ordering by one gate. Six combinations \( \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\} \) could be ordering by two gates. Four combinations \( \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\}, \{x, x, x, x, x, x, x\} \) could be ordering by three gates. One combination \( \{x, x, x, x, x, x, x\} \) 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.
Let consider the function F:
\[ Y_C = \{4,7,0,6,3,5,2,1\} Y_I = \{4,3,0,2,7,5,6,1\} Y_C = \{4,0,3,2,7,6,5,1\} \]
To calculate the partitions of the rest function we use the operation indicated by symbol \( \cdot \) facilitating this operation. For example the partitions of the rest function \( F(T_0) \) of the given function \( F \) are:
\[ Y_I = \{4,7,0,6,3,5,2,1\} \cdot \{0,0,0,0,1,0,1\} = \{4,7,0,0,3,1,2,5\} \]
\[ Y_I = \{4,3,0,2,7,5,6,1\} \cdot \{0,0,0,0,1,0,1\} = \{4,3,0,2,7,1,6,5\} \]
\[ Y_C = \{4,0,3,2,7,6,5,1\} \cdot \{0,0,0,0,0,1,1\} = \{4,6,3,2,7,6,1,5\} \]
The gates with XOR on line \( Y_1 \) swap minterms in the same block in the partition \( Y_1 \). 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 chances 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 algorithm 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 it 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 in 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 position could be done in any order.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Swapped positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>{0,0,0,0,1,0,1} X{0,0,0,0,0,1,1} X{0,0,0,0,1,1}</td>
</tr>
<tr>
<td>C0-1</td>
<td>{0,0,0,0,2,1,2} X{0,0,0,0,2,0,1} X{0,0,0,0,2,0,1}</td>
</tr>
<tr>
<td>C0-2</td>
<td>{0,0,0,0,2,0,1} X{0,0,0,0,2,0,1} X{0,0,0,0,2,0,1}</td>
</tr>
<tr>
<td>N0</td>
<td>{4,2,4,2,3,1,3,1} X{4,3,4,3,2,1,2,1} X{4,4,4,3,2,2,1,1}</td>
</tr>
<tr>
<td>T1</td>
<td>{0,0,0,1,0,0,1} X{0,0,0,0,0,1,1} X{0,0,0,0,0,1,1}</td>
</tr>
<tr>
<td>C1-0</td>
<td>{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1}</td>
</tr>
<tr>
<td>C1-1</td>
<td>{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1}</td>
</tr>
<tr>
<td>C1-2</td>
<td>{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1} X{0,2,0,2,1,0,2,1}</td>
</tr>
<tr>
<td>N1</td>
<td>{4,4,3,2,1,2,3,1} X{4,4,4,3,2,2,1,1} X{4,4,4,3,2,1,2,1}</td>
</tr>
<tr>
<td>T2</td>
<td>{0,0,0,0,0,0,1,1} X{0,0,0,0,1,0,0,1} X{0,0,0,0,1,0,0,1}</td>
</tr>
<tr>
<td>C2-0</td>
<td>{0,2,2,0,2,0,1} X{0,2,2,0,2,0,1} X{0,2,2,0,2,0,1}</td>
</tr>
<tr>
<td>C2-1</td>
<td>{0,2,2,0,2,0,1} X{0,2,2,0,2,0,1} X{0,2,2,0,2,0,1}</td>
</tr>
<tr>
<td>N2</td>
<td>{4,4,3,3,2,2,1,1} X{4,2,1,1;4,2,3,3;} X{4,3,2,1;4,3,2,1;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gates number</th>
<th>Minterm distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{0,0,0,0,0,1,1}</td>
</tr>
<tr>
<td></td>
<td>{0,0,0,0,2,1,1}</td>
</tr>
<tr>
<td></td>
<td>{0,0,2,0,0,1,1}</td>
</tr>
<tr>
<td></td>
<td>{4,4,3,3,2,2,1,1}</td>
</tr>
<tr>
<td>2</td>
<td>{0,0,0,1,1,0,0}</td>
</tr>
<tr>
<td></td>
<td>{0,0,1,1,0,0,0}</td>
</tr>
<tr>
<td></td>
<td>{0,0,2,2,1,1,0}</td>
</tr>
<tr>
<td></td>
<td>{2,2,0,2,1,0,0}</td>
</tr>
<tr>
<td></td>
<td>{2,2,1,1,0,0,0}</td>
</tr>
<tr>
<td></td>
<td>{1,1,2,3,3,0}</td>
</tr>
<tr>
<td>3</td>
<td>{2,2,0,0,0,0,1,1}</td>
</tr>
<tr>
<td></td>
<td>{0,0,1,2,2,3,3}</td>
</tr>
<tr>
<td></td>
<td>{1,1,0,0,2,2,3,3}</td>
</tr>
<tr>
<td></td>
<td>{1,1,2,2,0,0,3,3}</td>
</tr>
<tr>
<td>4</td>
<td>{1,1,0,0,0,0,0,0}</td>
</tr>
</tbody>
</table>

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 \( F(G_i) \) 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 \( F(G_i) \) for all reversible gates \( G_i \) and create the CBG set (closing best gates). If the gate \( G_i \) is a member of CBG set than for the function \( F(G_i) \) 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:

\[
Y_1: \{4, 7, 0, 6, 3, 5, 2, 1\} \quad Y_2: \{4, 3, 0, 2, 7, 5, 6, 1\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\}
\]

**Criterion 1:** The partition \( Y_1 \) could be ordered by the gate \( C_1 \). This gate will be the member of the FBG.

**Criterion 2:** Below 12 rest functions will be calculated:

\[
F_r(C_0-1) = Y_1: \{4, 7, 0, 6, 3, 2, 1, 5\} \quad Y_2: \{4, 3, 0, 2, 7, 6, 1, 5\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\}
\]

The partitions \( Y_1 \) could be ordered by one gate, the gate \( C_0 \) will be the member of the FBG.

\[
F_r(C_0-0) = Y_1: \{4, 7, 0, 6, 3, 2, 1, 5\} \quad Y_2: \{4, 3, 0, 2, 7, 6, 1, 5\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\}
\]

The function \( F \) could be ordered by two gates, the gate \( C_1 \) will be the member of the FBG.

\[
F_r(C_0-3) = Y_1: \{4, 7, 0, 6, 3, 2, 1, 5\} \quad Y_2: \{4, 3, 0, 2, 7, 6, 1, 5\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\}
\]

The function \( F \) could be ordered by two gates, the gate \( C_2 \) will be the member of the FBG.

\[
F_r(C_0-2) = Y_1: \{4, 7, 0, 6, 3, 2, 1, 5\} \quad Y_2: \{4, 3, 0, 2, 7, 6, 1, 5\} \quad Y_3: \{4, 0, 3, 2, 7, 6, 5, 1\}
\]

The function \( F \) could be ordered by three gates, the gate \( C_2 \) will be the member of the FBG.

The calculation of the rest functions gives the FBG set containing only two gates: \( C_2, C_0 \).

**Branch 1.1**

The function \( F(C_1-2, C_2-0) \) has partition \( Y_3 \) requiring three gates for ordering. The gates \( C_0, C_1 \) and \( T_0 \) are the members of the FBG set.

The FBG set of the rest functions \( F(C_1-2, C_2-0) \) contains five gates: \( C_1, T_2, C_2, C_0, C_1 \) and \( T_0 \).

**Branch 1.2**

The function \( F(C_1-2, C_2-0) \) has partition \( Y_2 \) requiring four gates for ordering. The gates \( C_2, C_0, C_1, T_2 \) are the members of the FBG set. The rest function \( F(C_1-2, C_0-2) \) gives also the gate \( C_2 \) as the member of FBG set because the partitions \( Y_2 \) of the function \( F(C_1-2, C_0-2, N_0) \) could be ordered by two gates \( C_2, T_2 \) and \( C_0 \). The functions \( F(C_1-2, C_0-2, N_0, T_0, C_2, T_2) \) have only one member of FBG set and it is the gate \( C_0 \). The functions \( F(C_1-2, C_0-2, N_0, T_0, C_2, T_2) \) and \( F(C_1-2, C_0-2, N_0, T_0, C_2, C_0) \) are the identical functions and two cascades with six gates were found.

From Lemma 2 we can add the next two cascades:

\[
C_2, N_0, C_0, T_2, C_2, C_0, C_0, T_2, C_2, C_0, C_1, N_0, C_0, T_2, C_2, C_0 \quad \text{and} \quad C_2, N_0, C_0, T_2, C_2, C_0, C_0, T_2, C_2, C_0, C_1, N_0, C_0, T_2, C_2, C_0
\]

In this branch of the algorithm when it starts from the gate \( C_1 \) 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 us 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:

\[ Y_1: \{4,7,0,6,3,5,2,1\} \]

\[ Y_2: \{4,3,0,2,7,5,6,1\} \]

\[ Y_3: \{4,0,3,2,7,6,5,1\} \]

As was shown from the definition 2 the partitions of the remain functions \( F_1(C2-0) \) are:

\[ Y_1: \{4,3,0,6,7,1,2,5\} \]

\[ Y_2: \{4,7,0,2,3,1,6,5\} \]

\[ Y_3: \{4,6,7,2,3,6,1,5\} \]

Two gates C2-0 and N2 can ordered the \( Y_1 \) partition.
The partitions of the remain function \( F_1(C0-2) \) are:

\[ Y_1: \{5,6,0,7,3,4,2,5\} \]

\[ Y_2: \{5,3,0,1,6,4,7,1\} \]

\[ Y_3: \{5,0,3,2,6,7,4,1\} \]

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 \( F_1(C2-0) = F_0 \) 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.

![Diagram](image)

**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 \( F_0(C2-0) \) 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**


