On Decomposing Systems of Boolean Functions via Ternary Matrix Cover Approach

Saeid Taghavi Afshord[1] and Yuri Pottosin[2]

[1]*Computer Engineering Department, Shabestar Branch, Islamic Azad University, Shabestar, Iran*
[2]*United Institutes of Engineering Cybernetics, National Academy of Sciences of Belarus, Minsk, Belarus*
taghavi@iaushab.ac.ir, pott@newman.bas-net.by

Abstract

The problem of two-block disjoint decomposition of completely specified Boolean functions is considered. Recently a good method in functional decomposition category was proposed. This method is based on using the ternary matrix cover approach. Due to investigation and analysis of this method and to search for an appropriate partition, a computer program was developed. After running the program on thousands of systems of Boolean functions, experimental results show that more than 95% of the inspected systems are decomposable. To obtain a solution of the task in decomposable systems, an efficient technique is also proposed. Using this technique, investigation of one partition was enough to determine decomposability of the inspected systems.

Keywords: *Boolean functions; functional decomposition; cover map; compact table; logic synthesis*

1. Introduction

The problem of decomposition of Boolean functions is one of the most important problems of logical design that makes it an object of great attention by many researchers in this field [4, 9, 11-17]. It is important to find a successful solution for this problem because it has a direct influence on the quality and cost of digital devices designed. Searching for an appropriate partition is NP-hard problem because it has been proved that this problem is equivalent to the well-known set covering problem [4, 9], but to be aware of decomposability of a given system of Boolean functions, finding only one solution is satisfying. So to find a solution of the task, we used the ternary matrix cover approach [1]. Using a compact table one can find rather easily the existence of a solution of the problem for a given system of Boolean functions, and if it does exist, the corresponding superposition can be easily found.

2. Main Definitions and Setting the Problem

Let a system of completely specified functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ be given by matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ that are the matrix representation of the system of disjunctive normal forms (DNFs) of the given functions [10]. Matrix $\boldsymbol{U}$ is a ternary matrix of $l \times n$ dimension where l is the number of terms in the given DNFs. The columns of $\boldsymbol{U}$ are marked with the variables $x_1, x_2, \ldots, x_n$, and the rows represent the terms of the DNFs . The matrix $\boldsymbol{V}$ is a Boolean matrix. Its dimension is $l \times m$, and its columns are marked with the variables

$y_1, y_2, \dots, y_m$. The ones in this columns point out the terms in the given DNFs. A row $\boldsymbol{u}$ in $\boldsymbol{U}$ *absorbs* a Boolean vector $\boldsymbol{a}$ if $\boldsymbol{a}$ belongs to the interval represented by $\boldsymbol{u}$.

The task considered is set as follows. Given a system of completely specified Boolean functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, the superposition $\boldsymbol{y} = \varphi(\boldsymbol{w}, \boldsymbol{z}_2)$, $\boldsymbol{w} = g(\boldsymbol{z}_1)$ must be found where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are vector variables whose components are Boolean variables in the subsets of the set $X = \{x_1, x_2, \dots, x_n\}$, Z_1 and Z_2 respectively such that $X = Z_1 \cup Z_2$ and $Z_1 \cap Z_2 = \varnothing$. At that, the number of components of the vector variable $\boldsymbol{w}$ must be less than that of $\boldsymbol{z}_1$. Such a kind of decomposition is called two-block disjoint decomposition [8-10]. The subsets Z_1 and Z_2 are called bound and free sets respectively. Only a few papers deal with the search for the partition $\{Z_1, Z_2\}$, at which this problem has a solution [2-8]. The main attention is paid to the search for subsets Z_1 and Z_2 such that the task would have a solution.

3. Introducing Cover Map and Compact Table

Any family π of different subsets (blocks) of a set L whose union is L, is called a *cover* of L. Let $L = \{1, 2, \dots, l\}$ be the set of numbers of rows of a ternary matrix $\boldsymbol{U}$. A cover π of L is called a *cover of the ternary matrix* $\boldsymbol{U}$ if for each value $\boldsymbol{x}^*$ of the vector variable $\boldsymbol{x}$ there exists a block in π containing all the numbers of those and only those rows of $\boldsymbol{U}$, which absorb $\boldsymbol{x}^*$. Block $\varnothing$ corresponds to the value $\boldsymbol{x}^*$, which is absorbed by no row of $\boldsymbol{U}$. Other subsets are not in π.

Let $t(\boldsymbol{x}^*, \boldsymbol{U})$ be the set of numbers of those rows of $\boldsymbol{U}$, which absorb $\boldsymbol{x}^*$. For every block π_j of π, we define the Boolean function $\pi_j(\boldsymbol{x})$ having assumed that $\pi_j(\boldsymbol{x}^*) = 1$ for any $\boldsymbol{x}^* \in \{0,1\}^n$ if $t(\boldsymbol{x}^*, \boldsymbol{U}) = \pi_j$, and $\pi_j(\boldsymbol{x}^*) = 0$ otherwise. Let us define an operation $\vee(\pi_i, \boldsymbol{V})$ over the rows of a binary matrix $\boldsymbol{V}$, the result of which is the vector $\boldsymbol{y}^*$ ($\boldsymbol{y}^* = \vee(\pi_i, \boldsymbol{V})$) obtained by component-wise disjunction of rows $\boldsymbol{V}$ whose numbers are in the block π_i. If $\pi_i = \varnothing$, all the components of $\boldsymbol{y}^*$ are equal to 0. It is shown that $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{y}^* = \vee(\pi_i, \boldsymbol{V})$ if $\pi_i(\boldsymbol{x}^*) = 1$ [1].

Let a pair of matrices, $\boldsymbol{U}$ and $\boldsymbol{V}$, give a system of completely specified Boolean functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, and let the matrix $\boldsymbol{U}_1$ be composed of the columns of $\boldsymbol{U}$, marked with the variables from the set $\boldsymbol{Z}_1$ and the matrix $\boldsymbol{U}_2$ from the columns marked with the variables from $\boldsymbol{Z}_2$. The covers of $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are $\pi^1 = \{\pi^1_1, \pi^1_2, \dots, \pi^1_r\}$ and $\pi^2 = \{\pi^2_1, \pi^2_2, \dots, \pi^2_s\}$. Let us construct a table M. Assign the blocks $\pi^1_1, \pi^1_2, \dots, \pi^1_r$ and the Boolean functions $\pi^1_1(\boldsymbol{z}_1), \pi^1_2(\boldsymbol{z}_1), \dots, \pi^1_r(\boldsymbol{z}_1)$ to the columns of M, and $\pi^2_1, \pi^2_2, \dots, \pi^2_s$ and $\pi^2_1(\boldsymbol{z}_2), \pi^2_2(\boldsymbol{z}_2), \dots, \pi^2_s(\boldsymbol{z}_2)$ to the rows of M. At the intersection of the i-th column, $1 \le i \le r$ and the j-th row, $1 \le j \le s$, of M, we put the value $\boldsymbol{y}^* = \vee(\pi^1_i \cap \pi^2_j, \boldsymbol{V})$. The table M is called *compact table*. It gives the system of Boolean functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ in the following way: the value of the vector Boolean function $\boldsymbol{f}(\boldsymbol{x}^*)$ is $\vee(\pi^1_i \cap \pi^2_j, \boldsymbol{V})$ at any set argument values $\boldsymbol{x}^*$, for which $\pi^1_i(\boldsymbol{z}_1) \wedge \pi^2_j(\boldsymbol{z}_2) = 1$ [1].

Having the compact table for a system of functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, it is easy to construct the desired systems $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$. The columns of the compact table are encoded with binary codes; equal columns may have the same codes. The length of the code is equal to $\lceil \log_2 r' \rceil$ where r' is the number of different columns of the table and $\lceil a \rceil$ is the least integer, which is not less than a. So, the system of functions $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$ is defined. The value of the vector variable $\boldsymbol{w}$ at any set of values of the vector variable $\boldsymbol{z}_1$ turning the function $\pi^1_i(\boldsymbol{z}_1)$ into 1 is the code of the i-th column, $1 \le i \le r$. Naturally, there is no solution to this task at the given partition $\{Z_1, Z_2\}$ of the set X of arguments if the length of the code is not less than the length of $\boldsymbol{z}_1$. Otherwise, the compact table

whose columns are assigned with the values of the variable $\boldsymbol{w}$ can be considered as a form of representation of the other desired system of functions $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$. The value of $\boldsymbol{y}$ at the value of $\boldsymbol{w}$ assigned to the i-th column, $1 \le i \le r$, and at any value of $\boldsymbol{z}_2$ turning $\pi^2_j(\boldsymbol{z}_2)$ into 1, $1 \le j \le s$, is the vector that is at the intersection of the i-th column and the j-th row.

4. Search for an Appropriate Partition

To search for an appropriate partition of the set of arguments we use ternary matrix covers and compact tables induced by them. Let a few free variables be to find that constitute the set Z_2 (then the set of bound variables would be $Z_1 = X \setminus Z_2$). To do this, we use the operation of dividing a ternary matrix cover by the cover of a column of the matrix. Let us determine the operation to divide the cover π of a ternary matrix $\boldsymbol{U}$ by the cover π^i of its i-th column as: $\pi / \pi^i = \pi^1 \times \pi^2 \times \ldots \times \pi^{i-1} \times \pi^{i+1} \times \ldots \times \pi^n$. This operation can be easily fulfilled using the *cover map*, which, as well as Karnaugh map, has the lines of symmetry related to the variables of the Boolean space represented by this map [5]. To transform the cover map of a ternary matrix $\boldsymbol{U}$ into that of the matrix obtained from $\boldsymbol{U}$ by deleting the i-th column, one should superpose pair-wise the entries that are symmetric with regard to the lines relative to x_i, and put the unions of the superposed entries into the obtained entries. The obtained cover map would represent the desired cover.

Example 1. Let a system of completely specified functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ is given by the following pair of matrices:

$$U = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & - \\ 0 & 1 & 0 & 0 & - \\ 0 & 1 & - & 0 & 1 \\ 0 & - & 0 & 0 & - \\ 0 & 0 & - & 0 & 1 \\ 1 & 1 & 0 & 1 & - \\ 1 & 1 & - & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}, \quad V = \begin{array}{c} \begin{matrix} y_1 & y_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}.$$

For the partition of the set of arguments into subsets $Z_1 = \{x_1, x_3, x_5\}$ and $Z_2 = \{x_2, x_4\}$, we have the following matrices:

$$U_1 = \begin{array}{c} \begin{matrix} x_1 & x_3 & x_5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & - \\ 0 & 0 & - \\ 0 & - & 1 \\ 0 & 0 & - \\ 0 & - & 1 \\ 1 & 0 & - \\ 1 & - & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}, \quad U_2 = \begin{array}{c} \begin{matrix} x_2 & x_4 \end{matrix} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ - & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}$$

Figure 1 shows the cover map of the ternary matrix $\boldsymbol{U}$ and the cover of $\boldsymbol{U}$ is $\pi = \{\varnothing, \{1\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{2, 4\}, \{4, 5\}, \{6, 7\}, \{2, 3, 4\}\}$. As it can be seen from Figure 2 the division of π by the cover of the column x_2 will be $\{\varnothing, \{1\}, \{6\}, \{7\}, \{2, 4\}, \{3, 5\}, \{6, 7\}, \{2, 3, 4, 5\}\}$. Having transformed this map by the described way with regard to x_4, we obtain $\{\varnothing, \{6\}, \{7\}, \{3, 5\}, \{6, 7\}, \{1, 2, 4\}, \{1, 2, 3, 4, 5\}\}$ as a result of dividing π by the covers of the columns x_2 and x_4 (see Figure 3). Similarly the result of dividing π by the covers of the columns x_1, x_3 and x_5 will be $\pi^2 = \{\{1\}, \{4, 5\}, \{6, 7\}, \{2, 3, 4\}\}$.

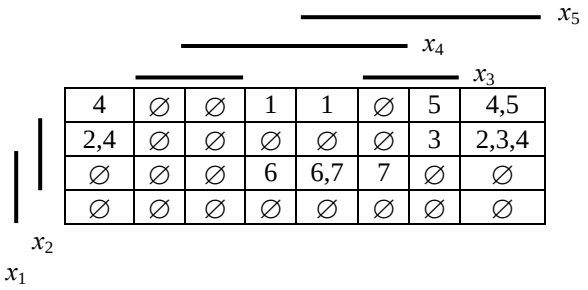


Figure 1. The Cover Map of Matrix U from Example 1

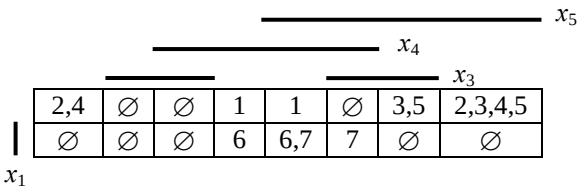


Figure 2. The Cover Map obtained by Dividing π by the Cover of the Column x_2

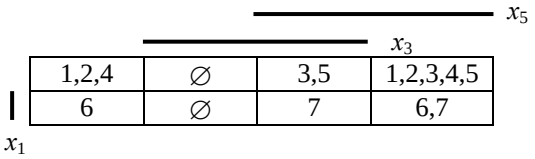


Figure 3. The Cover Map obtained by Dividing π by the Covers of the Columns x_2 and x_4

The compact table for the covers π^1 and π^2 is represented by Table 1 that has four different columns. To encode these columns, two variables are sufficient. The codes of the columns are shown at the bottom of Table 1. To construct the systems of functions $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$ that are the solution of the task, the functions connected with the blocks of the covers obtained must be constructed.

The DNFs of the functions connected with the blocks of π^1 can be obtained from the cover map in Figure 3:

$\pi^1{}_1(\boldsymbol{z}_1) = x_3\ \bar{x}_5$, $\pi^1{}_2(\boldsymbol{z}_1) = x_1\ \bar{x}_3\ \bar{x}_5$, $\pi^1{}_3(\boldsymbol{z}_1) = x_1\ x_3\ x_5$, $\pi^1{}_4(\boldsymbol{z}_1) = \bar{x}_1\ x_3\ x_5$, $\pi^1{}_5(\boldsymbol{z}_1) = x_1\ \bar{x}_3\ x_5$, $\pi^1{}_6(\boldsymbol{z}_1) = \bar{x}_1\ \bar{x}_3\ \bar{x}_5$, $\pi^1{}_7(\boldsymbol{z}_1) = \bar{x}_1\ \bar{x}_3\ x_5$.

Similarly, the DNFs of the functions connected with the blocks of π^2 are obtained: $\pi^2{}_1(\boldsymbol{z}_2) = \bar{x}_2\ x_4$, $\pi^2{}_2(\boldsymbol{z}_2) = \bar{x}_2\ \bar{x}_4$, $\pi^2{}_3(\boldsymbol{z}_2) = x_2\ x_4$, $\pi^2{}_4(\boldsymbol{z}_2) = x_2\ \bar{x}_4$.

Table 1.The Compact Table for the Partition from Example 1

	∅	6	7	3,5	6,7	1,2,4	1,2,3,4,5
1	00	00	00	00	00	10	10
4,5	00	00	00	01	00	01	01
6,7	00	01	01	00	01	00	00
2,3,4	00	00	00	10	00	11	11
	00	**01**	**01**	**10**	**01**	**11**	**11**

As a result of simple minimization we obtain the following matrices representing the desired superposition $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$, $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$:

$$\begin{matrix} w_1 & w_2 & x_2 & x_4 \end{matrix} \quad \begin{matrix} y_1 & y_2 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & - & 1 & 0 \\ 1 & - & 0 & 0 \\ 1 & 1 & - & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}; \quad \begin{matrix} x_1 & x_3 & x_5 \end{matrix} \quad \begin{matrix} w_1 & w_2 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & - \\ - & 0 & 0 \\ 1 & - & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Example 2. Let the system of completely specified Boolean functions from *Example 1* be given. Consider this variant that $Z_1 = \{x_1, x_2, x_3, x_4\}$ and $Z_2 = \{x_5\}$. For this variant with the cover map in Figure 1, we obtain the cover maps are shown in Figures 4, 5, 6 and 7, by dividing π by the cover of the columns x_1, x_1 and x_2, x_1, x_2 and x_3, and x_1, x_2, x_3 and x_4, respectively. We also obtain the cover map by dividing π by the cover of the column x_5 in the similar manner, to achieve to the cover π^1.

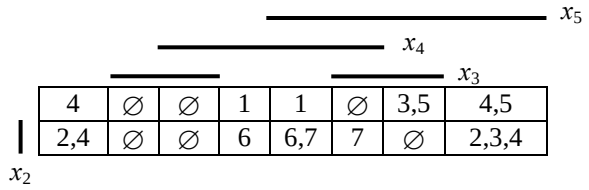


Figure 4. The Cover Map obtained by Dividing π by the Cover of the Column x_1

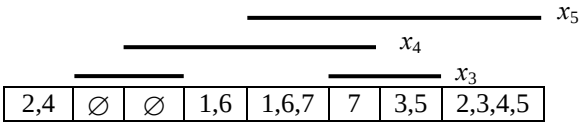


Figure 5. The Cover Map Obtained by Dividing π by the Cover of the Columns x_1 and x_2

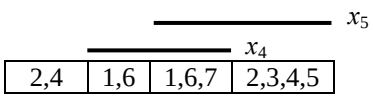


Figure 6. The Cover Map Obtained by Dividing π by the Cover of the Columns x_1, x_2 and x_3

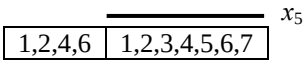


Figure 7. The Cover Map Obtained by Dividing π by the Cover of the Columns x_1, x_2, x_3 and x_4

So we have the covers $\pi^1 = \{\varnothing, \{1\}, \{3\}, \{5\}, \{7\}, \{4, 5\}, \{6, 7\}, \{2, 3, 4\}\}$ and $\pi^2 = \{\{1, 2, 4, 6\}, \{1, 2, 3, 4, 5, 6, 7\}\}$. The compact table for these covers is represented by Table 2 that have six different columns. To encode these columns with the values of $\boldsymbol{w}$, three variables are needed that is less than the length of $\boldsymbol{z}_1$. The codes of the columns are shown at the bottom of Table 2. To construct the system of functions $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$ that are the solution of the task, anybody can do the same steps as *Example 1* and we did not calculate them again.

Table 2. The Compact Table for the Partition from Example 2

	∅	1	3	5	7	4,5	6,7	2,3,4
1,2,4,6	00	10	00	00	00	01	01	11
1,2,3,4,5,6,7	00	10	10	01	01	01	01	11
	000	**001**	**010**	**011**	**011**	**100**	**100**	**101**

5. Implementation and Results

We designed and developed a computer program in C++ to find a solution for systems of Boolean functions. Our program based on using the ternary matrix cover approach and the general scheme of the implemented algorithm summarized in Figure 9. The experiments run on a Pentium 2.26GHz CPU with 3 GByte of the main memory. We generated systems of completely specified Boolean functions using a prepared library which has been explained in [18, 19]. We considered three parameters for these systems; number of rows of matrix $\boldsymbol{U}$ that indicate number of DNFs, number of columns of matrix $\boldsymbol{U}$ and number of columns of matrix $\boldsymbol{V}$. After generating matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ as SOP (Sum-of-Product), we expand matrix $\boldsymbol{U}$ to obtain corresponding matrix without don't cares. Those rows that contain don't cares, will replace with several suitable rows and consequently the number of conjunctions will be increased.

Then we begin to provide cover map; for that we used Gray code encoding system. On contrary to *Examples 1* and *2* that cover maps are two dimensional tables, due to simplicity to store in the computer memory and also for the future calculations of the compact table, we implemented it as a one dimensional array. An example of our approach with three variables is represented in Figure 8. The order of the replacement of the variables on the array is important and this can be extended for any number of variables. In the array is stored the values that was explained in the previous section and Gray codes in the array at Figure 8 are symbolically shown to represent the correctness of the approach. We save the Gray codes in the other list as well.

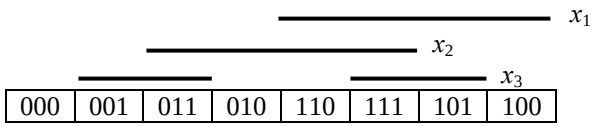


Figure 8. The Cover Map Model for Three Variables using Gray Code Encoding System

```
for Con ← ConDown to ConUp
  for Arg ← ArgDown to ArgUp
    for Fun ← FunDown to FunUp
     ➢ Generate new SOP(Con,Arg,Fun)
     (i.e. Matrices U and V)
     ➢ Expand Matrix U
     (i.e. remove don't cares in Matrix U and
     replace them with suitable ones)
     ➢ Compute Cover Map
     (Generate Gray Codes with Length 2^n and fill
     out the Cover Map Array According to the
     Algorithm Rules)
     ➢ for k ← n-1 downto 2
        » for each combination of (n k)
         Check the Current Partition
           1- Divide Cover Map Over Z1
           2- Divide Cover Map over Z2
           3- Compute Compact Table
           4- Compute the Number of Different
              Columns (r) of Compact Table
           5- Encode the Columns of the Compact
              Table
           6- if r ≤ 2^(k-1) then
                Solution Founded
                (Produce Matrices ϕ, Y, X and W)
        » if Solution Founded then
             Declare the System is decomposable
             and Stop
     ➢ if Solution not Founded then
         Declare the System is not decomposable
```

Figure 9. The Implemented Algorithm to Determine Decomposability of Systems of Boolean Functions

In fact, the method of storing information in the mentioned array is as follows. Each row in the expanded matrix of $\boldsymbol{U}$ is numbered with the integers according to the row numbers of matrix $\boldsymbol{U}$. We compare the value of each row in this matrix with Gray codes list until the equal value is found. Then we add the row number of the compared row in the corresponding element of the array. We continue this method until all rows are compared and the row numbers are added to the array elements. At the end, we sweep the array and put the empty set to the elements with no values added.

To find a solution of the task anyone should enquire into all possible partitions which are constructing Z_1 and Z_2. The relatively simple method to address the appropriate partition can be done by lexicographical enumeration. But this method for those systems of Boolean functions which have more arguments is not suitable and in practice it is time consuming. We are notice that the main subject of the current work is determining decomposability of the given system of Boolean functions.

In analyzing of all decompositions of the several systems of Boolean functions, we understood, if $|Z_1|$ is selected the maximum possible size, a solution of the task will be

found with high probability. So to specify decomposability of the given system, it is better to consider lexicographical order according to the maximum size of Z_1. We start $|Z_1| = n-1$ (n is the number of arguments) and search lexicographically for all possible partitions. If the result of the search at this stage is unsuccessful the size of Z_1 can be decreased and continued. In fact we investigate all possible partitions from the biggest size of Z_1 to the smallest one. Empirical results confirmed the validation of our idea and in all the cases; we perceived the task has a solution in the first turn. For example if $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, we observed the task has a solution with $Z_1=\{x_1, x_2,\ldots, x_{n-1}\}$ and $Z_2=\{x_n\}$ (See *Example 2* from Section 4). The proposed technique is very important, because the amount of the necessary calculations will be decreased exponentially and consequently the efficiency of computational time will be increased. With consideration of our goal which is determining decomposability of a system, finding a solution of the task will be satisfying. In Table 3 and in column NQS (Number of Quick Solutions) we declare how many systems have a solution according to this approach.

After computing cover map of the current system of Boolean functions and with regarding to the mentioned technique; we used Knuth's algorithm [20] to generate all combinations of the arguments and for each partition we check whether it is a solution of the task or not. To obtain all *k*-element subsets of an *n*-element set, this algorithm is one of the fastest ones. Each *k*-element subsets is used to construct Z_1 elements and rest of the arguments will be the elements of Z_2. If a partition as a solution is found, the program will stop and calculate four matrices; matrix Φ, matrix Y, matrix X and matrix W. These matrices are the solution of the task. In fact the current system of Boolean functions changes to two new systems with fewer arguments. Although we obtain these matrices but they have not influence in our results in this paper. This manner is repeated for all partitions and if an appropriate partition is not found, the program will declare the current system of Boolean functions is not decomposable.

Now, we report the experimental results for our approach in decomposition of Boolean functions, described in the previous sections. Due to space and time limitations, the results are shown refer only to the decompositions of systems with few arguments and the few functions as well. The results summarized in Table 3, are quite promising. They show that more than 95% of generated systems are decomposable and all of them have a solution when the size of Z_1 is the maximum.

Table 2. The Experimental Results

Con		Arg		Fun		NS	PD	ND	NQS
From	to	from	to	from	To				
6	15	4	6	2	4	90	90	81	81
10	15	5	8	2	6	120	96	116	116
10	25	6	8	4	8	240	93	224	224
10	30	5	10	3	8	756	95	725	725
10	30	6	10	4	12	945	90	851	851
20	50	8	10	4	8	465	99	461	461
15	40	5	12	2	6	1040	99	1032	1032
20	30	8	12	6	9	220	99	218	218
59	60	10	12	4	8	30	100	30	30
59	60	10	12	6	10	30	100	30	30
50	50	10	14	6	8	15	100	15	15
60	60	12	15	6	12	28	100	28	28
50	50	14	16	6	10	15	100	15	15

The first three columns in Table 3 show intervals of conjunctions (Con), arguments (Arg) and functions (Fun) respectively. And as it can be seen from Figure 9, they used in nested loops, it means that all combinations of these intervals will inform the parameters of a system of Boolean functions. The Number of Systems (NS) in each program execution was investigated, the percentage of decomposition (PD) is percent of decomposable NSs and the number of decomposable systems (ND) was represented in a separate column. As discussed before, the last column indicates the number of quick solutions (NQS) that is the solutions according to our approach in finding an appropriate partition.

6. Conclusion and Future Works

We developed a computer program to determine decomposability of systems of Boolean functions via ternary matrix cover approach. The ternary matrix cover and the representation of a system of Boolean functions in the form of compact table are simple to be realized and we implemented thousands of systems. The experimental results were interesting and show that more than 95% of the inspected systems are decomposable and all of them have a solution when the bound set has the maximum possible size. This means that investigation of only one partition among the exponentially growing partitions is enough to determine decomposability of a given system.

As a future work, optimization in encoding of compact table is proposed, because it has direct influence on the quality of the obtained solutions. It is also useful to find the best solution among the all solutions from the circuit size point of view which is useful in practical scenes.

Acknowledgements

This work was done in the logical design laboratory at the united institute of informatics problems of the NAS of Belarus. The authors like to thank this laboratory by its support in providing the benchmark source codes.

References

[1] Y. V. Pottosin and E. Shestakov, “Choice of Free Arguments in Decomposition of Boolean Functions Using the Ternary Matrix Cover Approach”, Proceeding of the 5th International Conference on Neural Networks and Artificial Intelligence (ICNNAI), Brest, Belarus, **(2010)** June, pp. 123-127.

[2] P. N. Bibilo, “Decomposition of Boolean Functions Based on Solving Logical Equations,” Byelaruskaya Navuka, Minsk, Belarus, (In Russian), **(2009)**.

[3] L. Jóźwiak and A. Chojnacki, “An Effective and Efficient Method for Functional Decomposition of Boolean Functions Based on Information Relationship Measures”, Proceeding of 3rd Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS), Bratislava, Slovakia, **(2000)** April, pp. 242-249.

[4] M. A. Perkowski and S. Grygiel, “A Survey of Literature on Functional Decomposition Version IV,” Technical report, Department of Electrical Engineering, Portland State University, Portland, USA, **(1995)**.

[5] A. D. Zakrevskij, “Decomposition of Partial Boolean Functions: Testing for Decomposability According to a Given Partition”, Informatika Journal, (In Russian), vol. 1, no. 13, **(2007)**, pp. 16-21.

[6] M. Rawski, “Heuristic Algorithm of Bound Set Selection in Functional Decomposition for Heterogeneous FPGAs,” 21st International Conference on Systems Engineering (ICSEng), Las Vegas, USA, **(2011)** August, pp. 465-466.

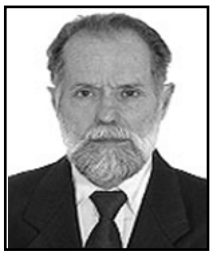
[7] V. Muthukumar, R. J. Bignall and H. Selvaraj, “An efficient variable partitioning approach for functional decomposition of circuits,” Journal of Systems Architecture, vol. 53, no. 1, **(2007)**, pp. 53-67.

[8] C. M. Files and M. A. Perkowski, “New Mutivalued functional decomposition algorithms based on MDDs”, IEEE Transactions on Computer-Aided Design of Integrated Ciruits and Systems, vol. 19, no. 9, **(2000)**, pp. 1081-1086.

[9] S. Hassoun and T. Sasao, "Logic Synthesis and Verification", The Springer International Series in Engineering and Computer Science, Kluwer Academic Publishers, **(2001)**.
[10] A. Zakrevskij, Y. V. Pottosin and L. Cheremisinova, "Optimization in Boolean Space", Tallinn: TUT Press, **(2009)**.
[11] T. Bengtsson, A. Martinelli and E. Dubrova, "A BDD-based fast heuristic algorithm for disjoint decomposition", Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC03), Kitakyushu, Japan, **(2003)** January, pp. 191-196.
[12] Y-T. Lai, K-R. Pan and M. Pedram, "OBDD-based function decomposition: Algorithms and implementation", IEEE Transactions on Computer-Aided Design of Integrated Ciruits and Systems, vol. 15, no. 8, **(1996)**, pp. 977-990.
[13] A. Martinelli, "Advances in Functional Decomposition: Theory and Applications", Doctoral Dissertation, Royal Inctitute of Technology (KTH), Stokholm, Sweden, **(2006)**.
[14] M. Rawski, "Evolutionary Algorithms in Decomposition-Based Logic Synthesis, Evolutionary Algorithms", Edited Prof. E. Kita (Ed.), InTech Publisher, **(2011)**.
[15] P. Porwik and R. S. Stankovic, "Dedicated Spectral Method of Boolean Function Decomposition", International Journal of applied mathematics and computer science (amcs), vol. 16, no. 1, **(2006)**, pp. 271-278.
[16] P. V. Athira and S. R. Ramesh, "An Approach towards Logic Synthesis by Functional Decomposition", International Journal of Engineering Research and Applications (IJERA), vol. 2, no. 3, **(2012)**, pp. 324-330.
[17] F. Yu, L. F. Wang, R. H. Tan and H. Jin, "An improved functional decomposition method based on FAST and the method of removal and operation", Proceedings of the International Conference on System Science and Engineering (ICSSE), Dalian, China, **(2012)** June, pp. 487-492.
[18] V. I. Romanov, "Tools development for logic designing", Logic Design, Institute of Engineering Cybernetics of NASB, Minsk, Belarus, (In Russian), **(2001)** November, pp. 151-170.
[19] V. I. Romanov, "Tools for programming Boolean calculations", Abstracts of the XVIII European Conference, Combinatorics for modern manufacturing, logistics and supply chains, (UIIP-NASB) Minsk, Belarus, **(2005)** May, pp. 57-58.
[20] D. E. Knuth, "The Art of Computer Programming", Vol. 4A, Combinatorial Algorithms, part 1, Addison-Wesley Professional, **(2011)**.

Authors

Saeid Taghavi Afshord received his BSc degree in applied mathematics and MSc degree in computer engineering from the Islamic Azad University, Tabriz and Qazvin branches in 2003 and 2006, Iran respectively. He joined the Islamic Azad University, Shabestar branch, Iran, as a faculty member in 2008. Currently he is a PhD student in computer engineering at the United Institute of Informatics Problems of the NAS of Belarus, from March 2011. His research areas are Energy Saving and Topology Control in Ad Hoc and Sensor Networks, and Methods for Boolean functions Decomposition.



Yuri Vasilievich Pottosin graduated from Tomsk State University (Russia), department of radio-physics and electronics, in 1960. From the beginning of 1961 until 1973, he worked at that university as a researcher. In 1970, he defended his PhD thesis. From 1973 until now, he works at the Institute of Engineering Cybernetics of National Academy of Sciences of Belarus. Now he is a leading researcher. His main scientific interest is logical design. He teaches "Discrete Mathematics" and "Theory of Designing Digital Devices and Systems" for the students of Byelorussian State University of Informatics and Radio-Electronics.