Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 4, No. 2, May 2013), Pages: 75-85
www.jacr.iausari.ac.ir

Analyzing Decompositions of a System of Boolean Functions Using the Ternary Matrix Cover Approach

Saeid Taghavi Afshord[1]✉, Yuri Pottosin[2]

(1) Computer EngineeringDepartment, Shabestar Branch, Islamic Azad University, Shabestar, Iran
(2)United Institute of Engineering Cybernetics, National Academy of Sciences of Belarus,Minsk, Belarus

taghavi@iaushab.ac.ir; pott@newman.bas-net.by

Received: 2013/02/17; Accepted: 2013/04/11

Abstract

The problem of series two-block disjoint decomposition of completely specified Boolean functions is considered. Analysis and investigation of such systems are very important in logical design context. Recently, a good method for solving this problem was suggested which has been based on the ternary matrix cover approach. Using this method a computer program was developed. This paper is focused on decomposability of a system of Boolean functions. The experiments were done on generated systems and standard benchmarks. In decomposable systems, the total number of solutions and the time elapsed to achieve them are inspected. The total number of solutions among all partitions for investigated systems, ranged between 3% and 87% in generated systems and also, 1% and 96% in standard benchmarks.

Keywords: *Boolean functions, 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. It has been shown in [9] a considerable number of papers are already published on this topic and still it is interesting for the research [4, 6, 7]. 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. Functional decomposition relies on breaking down a complex system into a network of smaller and relatively independent co-operating subsystems, in such a way that the original system's behavior is perceived. A system is decomposed into a set of smaller subsystems, such that each of them is easier to analyze, understand and synthesize. Decomposition-based synthesis methods are not limited only to logic synthesis of logic circuits. The strong motivation for developing decomposition techniques comes recently from modern research areas such as pattern recognition, knowledge discovery and machine learning in artificial intelligence [15].

The problem of decomposition of a system of Boolean functions can be considered in the following statement. A system of completely specified Boolean functions $\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x})$is given where $\boldsymbol{y}=(y_1, y_2, \ldots, y_m)$, $\boldsymbol{x}=(x_1, x_2, \ldots, x_n)$, $\boldsymbol{f}(\boldsymbol{x})=(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$. The superposition $\boldsymbol{y}=\varphi(\boldsymbol{w}, \boldsymbol{z}_2), \boldsymbol{w}=g(\boldsymbol{z}_1)$ where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are vector variables whose components are Boolean variables in the subsets Z_1 and Z_2 respectively that form a

partition of the set $X = \{x_1, x_2, \ldots, x_n\}$ of arguments. 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].

Searching for a solution of this kind is NP-hard problem because it has been proved that this problem is equivalent to the well-known set covering problem [4]. To be aware of decomposability of a given system of Boolean functions, finding only one such a pair is satisfying. But, due to analysis of the task and to be prepared to search for the best solution,it is useful to find all possible solutions. For that, it is usedthe ternary matrix cover approach[1] in this paper. Using a compact table one can find rather easily the existence of a solution of the problem for a given system of functions, and if it does exist, the corresponding superposition can be easily found.

2. Definitions and Setting the Problem

Let a system of completely specified Boolean 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 intervals of the space of the variables $x_1, x_2, \ldots, x_n$). The matrix $\boldsymbol{V}$ is a Boolean matrix. Its dimension is $l \times \tau$, and its columns are marked with the variables $y_1, y_2, \ldots, y_m$. 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 Z_1 and Z_2 of the set $X = \{x_1, x_2, \ldots, x_n\}$, 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 $\boldsymbol{z}_1$. The main attention is paid to the search for subsets Z_1 and Z_2 such that the task would have a solution. It is clear that the subset Z_1 should have at least two members while the subset Z_2 may have only one.

3. 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 $L = \{1, 2, \ldots, l\}$ 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, the Boolean function $\pi_j(\boldsymbol{x})$ is defined and it is 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 an operation $\vee(\pi_i, \boldsymbol{V})$ is defined 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 in [1] that $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{y}^* = \vee(\pi_i, \boldsymbol{V})$ if $\pi_i(\boldsymbol{x}^*) = 1$.

There is a convenient way to construct the cover of a ternary matrix $\boldsymbol{U}$ when the number of arguments is not large. This technique uses the cover map that has the structure of the Karnaugh map. In any cell of a cover map of $\boldsymbol{U}$ corresponding to a vector $\boldsymbol{x}^*$, there is the set $t(\boldsymbol{x}^*, \boldsymbol{U})$, which is a block of the cover of $\boldsymbol{U}$.

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 a table M is construct. 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, the value $\boldsymbol{y}^* = \vee(\pi^1_i \cap \pi^2_j, \boldsymbol{V})$is defined. The table M is called the *compact table*. It gives the system of Boolean functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ in the following way: the value of the Boolean vector function $\boldsymbol{f}(\boldsymbol{x}^*)$ is $\vee(\pi_{1i} \cap \pi_{2j}, \boldsymbol{V})$ at any set argument values $\boldsymbol{x}^*$, for which $\pi_{1i}(\boldsymbol{z}_1) \wedge \pi_{2j}(\boldsymbol{z}_2) = 1$.

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.

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

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

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

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

To find the length of $\boldsymbol{w}$ in the superposition $\boldsymbol{y}=\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2), \boldsymbol{w}=\boldsymbol{g}(\boldsymbol{z}_1)$ where $\boldsymbol{z}_1=(x_1, x_2, x_3)$ and $\boldsymbol{z}_2=(x_4, x_5)$, the covers of the ternary matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$are constructed: $\pi^1=\{\varnothing, \{3\}, \{5\}, \{7\}, \{6, 7\}, \{1, 4, 5\}, \{2, 3, 4\}\}$ and $\pi^2=\{\{1, 6\}, \{2, 4\}, \{1, 6, 7\}, \{2, 3, 4, 5\}\}$ (In examples 2 and 3, these covers will be obtained in details). The corresponding compact table is represented in Table 1 that has seven different columns. Clearly, this task has no solution at the given subsets Z_1 and Z_2, because to encode the columns of the compact table with the values of $\boldsymbol{w}$, three variables are needed that is not less than the length of $\boldsymbol{z}_1$.

Table 1. The compact table for the system of functions in Example 1

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

4. Search for Appropriate Partition

To search for an appropriate partition of the set of arguments the ternary matrix covers and compact tables induced by the mare used. 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, the operation of dividing a ternary matrix cover by the cover of a column of the matrix is used. Let the operation is determine 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 [1].

Example 2. Figure 1 shows the cover map of the ternary matrix $\boldsymbol{U}$ from Example 1. The cover of $\boldsymbol{U}$ isπ = {∅, {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_4 will be {∅, {3}, {5}, {6}, {7}, {1, 4}, {2, 4}, {6, 7}, {1, 4, 5}, {2, 3, 4}}. Having transformed this map by the described way with regard to x_5, the set{∅, {3}, {5}, {7}, {6, 7}, {1, 4, 5}, {2, 3, 4}} as a result of dividing π by the covers of the columns x_4 and $x_5$is obtained(see Figure 3).

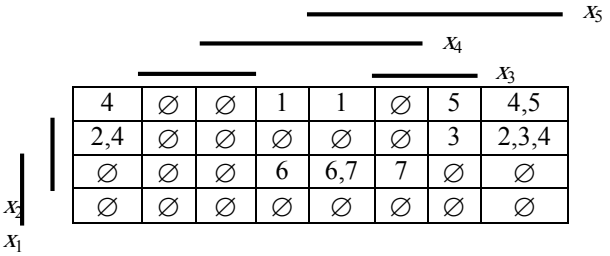

Figure 1.The cover map of the matrix U from Example 1

The method used to search for an appropriate partition consists in fulfilling the lexicographical enumeration and testing by the above way every variant of the set Z_2 if it would provide a solution of the task.

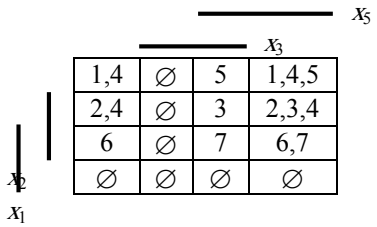

Figure 2.The cover map obtained by dividing π by the cover of the column

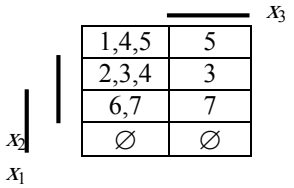

Figure 3.The cover map obtained by dividing π by the covers of the columns x4 and x5

Example 3. Let the system of completely specified Boolean functions from Example 1 be given. Consider this variant that $Z_2 = \{x_2, x_4\}$, $Z_1 = \{x_1, x_3, x_5\}$. For that, with the cover map in Figure 1, the cover map shown in Figure 4 is obtained and then Figure 5 from Figure 4 is also obtained.

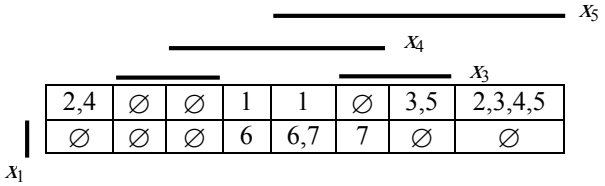

Figure 4.The cover map obtained by dividing π by the cover of the column x2

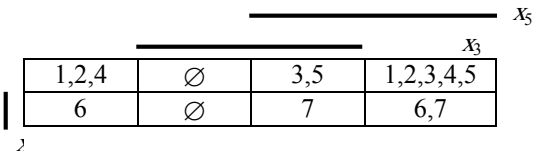

Figure 5.The cover map obtained by dividing π by the covers of the columns x2 and x4

The compact table for the covers $\pi^1 = \{\varnothing, \{6\}, \{7\}, \{3, 5\}, \{6, 7\}, \{1, 2, 4\}, \{1, 2, 3, 4, 5\}\}$ and $\pi^2 = \{\{1\}, \{4, 5\}, \{6, 7\}, \{2, 3, 4\}\}$ is represented by Table 2 that have four different columns. To encode these columns, two variables are sufficient. 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, the functions connected with the blocks of the covers obtained must be constructed.

Table 2. The compact table for the partition from Example 3

	∅	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**

The DNFs of the functions connected with the blocks of π^1 can be obtained from the cover map in Figure 5: $\pi^1_1(\boldsymbol{z}_1) = x_3 \bar{x}_5$, $\pi^1_2(\boldsymbol{z}_1) = x_1 \bar{x}_3 \bar{x}_5$, $\pi^1_3(\boldsymbol{z}_1) = x_1 x_3 x_5$, $\pi^1_4(\boldsymbol{z}_1) = \bar{x}_1 x_3 x_5$, $\pi^1_5(\boldsymbol{z}_1) = x_1 \bar{x}_3 x_5$, $\pi^1_6(\boldsymbol{z}_1) = \bar{x}_1 x_3 x_5$, $\pi^1_7(\boldsymbol{z}_1) = \bar{x}_1 x_3 x_5$. Similarly, the DNFs $\pi^2_1(\boldsymbol{z}_2) = \bar{x}_2 x_4$, $\pi^2_2(\boldsymbol{z}_2) = \bar{x}_2 \bar{x}_4$, $\pi^2_3(\boldsymbol{z}_2) = x_2 x_4$, $\pi^2_4(\boldsymbol{z}_2) = x_2 \bar{x}_4$ are obtained. As a result of simple minimization, the following matrices are obtained which are representing the desired superposition $\boldsymbol{y}= \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$, $\boldsymbol{w}= \boldsymbol{g}(\boldsymbol{z}_1)$:

$$\begin{matrix} w_1 & w_2 & x_2 & x_4 \end{matrix} \quad \begin{matrix} y_1 & y_2 \end{matrix} \qquad \begin{matrix} x_1 & x_3 & x_5 \end{matrix} \quad \begin{matrix} w_1 & w_2 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & - & 1 & 0 \\ 1 & - & 0 & 0 \\ 1 & 1 & - & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}; \qquad \begin{bmatrix} 0 & 0 & - \\ - & 0 & 0 \\ 1 & - & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

5. Implementation and Results.

The special computer program in C++ was designed and developed to find all the solutions of systems of Boolean functions. The program is based on using the ternary matrix cover approach and the general scheme of implemented algorithm is summarized in Figure 7.This algorithm is also applied to evaluation of the standard benchmarks with small modifications. The experiments run on a Pentium 2.26GHz CPU with 3 GByte of main memory.

Due to evaluation of explained approach, three types of benchmarks were utilized. At first the systems of completely specified Boolean functions were generated using a prepared library which has been explained in [11, 12].Then the standard benchmarks were used which are well-known in the literature, both industrial and mathematical benchmarks [13]. Three parameters for the all evaluated systems were considered; the number of rows of matrix $\boldsymbol{U}$ that indicate the number of conjunctions, the number of columns of matrix $\boldsymbol{U}$ or the number of arguments and the number of columns of matrix $\boldsymbol{V}$ or the number of functions. In generated systems, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ as SOP (Sum of Product) were prepared. After providing these matrices, first of all, the matrix $\boldsymbol{U}$ is expanded to obtain the corresponding matrix without don't cares. The rows which have don't cares will be replaced with several suitable rows.

Then the cover map will be provided; for that Gray code encoding system was used. On contrary to the examples in section 4 that cover map is a two dimensional table, due to simplicity to store in computer memory and also for the future calculations of compact table, it was implemented as a one dimensional array. An example of this approach with three variables is represented in Figure 6. The order of replacement of the

variables in the array is important and this can be extended for any number of variables. In the array the values explained in section 4 is stored and Gray codes in the array of Figure 6 are symbolically shown to represent the correctness of the approach, but Gray codes in the other l is t is also saved.

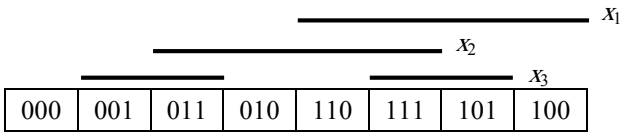

Figure 6.The cover map model for three variables using Gray code encoding system

-Algorithm for one system of Boolean functions

- *Con ← ConValue*
- *Arg ← ArgValue*
- *Fun ← FunValue*
- *Generate* **SOP***(Con,Arg,Fun)*
  (i.e. Matrices ***U*** *and* ***V****)*
- *Expand Matrix* ***U***
  (Removing don't cares in Matrix ***U*** *and replacing these rows 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)*
- *C ← 0 (To Count Number of all Solutions)*
- **for** *k ← 2* **to** *n-1*

»*Combination Generator(n,k)*

»**for** *each combination of* $\binom{n}{k}$

Check the Current Partition

1-Divide Cover Map Over Z_1
2-Divide Cover Map over Z_2
3-Compute Compact Table (CT)
4-Compute Different Columns (r) of CT
5-Encode the Columns of CT
*6-***if** $r \leq 2^{k-1}$ **then**
Solution Founded
(Calculate Matrices ***φ,w,y*** and***x****)*

»**if** *Solution Founded* **then**

Add this partition to the set of Solutions and C ← C + 1

- **if** *C=0* **then**

Declare the system is not decomposable

- **else**

Declare the system is decomposable and Print C

Figure 7.The implemented algorithm for determining decomposability of a system of Boolean functions and to find all solutions of the system on generated benchmarks

In fact, the way of storing information in the mentioned array is as follows. Each row in matrix $\boldsymbol{U}$, numbered with integers started from one. The value of each row with Gray codes list is compared until the equal value to be founded. Then the row number of compared row in corresponding element of the array is added. This manner is continued until all the rows to be compared and the row numbers to be added to the array elements. At the end, the array is swept and it is put the empty set to the elements with no value added.

To find all solutions 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. After computing cover map of the current system of Boolean functions, in each stage Knuth algorithm [14] was used to generate all combinations of the arguments and of course for each partition it is checked 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 the rest of the arguments will be the elements of the Z_2.

If a partition as a solution is found, the program will keep it and will calculate four matrices; matrix $\boldsymbol{\Phi}$, matrix $\boldsymbol{Y}$, matrix $\boldsymbol{X}$ and matrix $\boldsymbol{W}$. These matrices are a solution of the task. In fact the current system of Boolean functions converts to two new systems with less arguments; matrices $\boldsymbol{\Phi}$ and $\boldsymbol{Y}$ as $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, for the first system and also matrices $\boldsymbol{X}$ and $\boldsymbol{W}$ as $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, for the second system.

This method is repeated for all partitions and if appropriate partition is not found, the program declares the current system of Boolean functions is not decomposable; otherwise the program prints the number of solutions for the current system. Now, the experimental results for the explained approach in decomposition of Boolean functions are reported which is described in the previous sections. Due to space and time limitations, the results are shown refer only to the decomposition of systems with few arguments and few functions as well. The results are summarized in Tables 3a, 3b and 3c.

The results show that more than 95% of generated systems are decomposable and all of them have several solutions when the system is decomposable. The first three columns in Tables 3 represent the number of conjunctions (Con), the number of arguments (Arg) and the number of functions (Fun) respectively and these informs the parameters of a generated system of Boolean functions. The total number of partitions (TNP) are counted when $2 \leq |Z_1| \leq n-1$. So it implies that the total partitions will be $\sum_{k=2}^{n-1}\binom{n}{k}$ which it is equal to $2^n - (n+2)$.

The Number of Solutions (NS) is part of the results which is found after the program was executed and the percentage of the Solutions (PS) is percent of NS to NTP. The value of NS and PS are zero if the evaluated system of Boolean functions is not decomposable. The last column represents elapsed time (ET) which is the run time of the program for each system of Boolean functions during obtaining all solutions and it was calculated in seconds.

Table 3a.Experimental results on generated systems

Con	Arg	Fun	TNP	NS	PS	ET
8	5	2	25	21	84	<1
10	6	2	56	49	87	4
10	6	4	56	18	32	4
15	6	8	56	26	46	5
20	7	3	119	25	21	21
20	7	10	119	15	13	25
25	7	14	119	7	6	34
15	8	4	246	40	16	134
20	8	6	246	104	42	123
40	8	10	246	8	3	201
30	9	5	501	125	25	926
25	9	9	501	30	6	650
25	9	16	501	61	12	824
30	10	3	1012	58	6	3634
30	10	5	1012	55	5	3197
30	10	8	1012	491	49	1362
30	12	6	4082	673	16	20413

Table 3b.Experimental results on Mathematical benchmarks

Bench	Con	Arg	Fun	TNP	NS	PS	ET
rd53	31	5	3	25	15	60	1
fsm.pla	15	5	7	25	7	28	1
rd73	147	7	3	119	98	82.35	47
z4	127	7	4	119	37	31.09	37
log8mod	46	8	5	246	134	54.47	102
Radd	120	8	5	246	59	23.98	253
Root	255	8	5	246	40	16.26	285
adr4	255	8	5	246	59	23.98	536
Dist	256	8	5	246	6	2.44	345
sqr6	63	6	12	56	0	0	6
z5xp1	128	7	10	119	0	0	42
f51m	256	8	8	246	0	0	362
addm4	512	9	8	501	0	0	1268

Table 3c.Experimental results on industrial benchmarks

Bench	Con	Arg	Fun	TNP	NS	PS	ET
newapla2	7	6	7	56	34	60.71	2
m1	32	6	12	56	12	21.43	3
sqn	96	7	3	119	5	4.2	82
dc2	58	8	7	246	3	1.22	166
m2	96	8	16	246	80	32.52	178
m3	128	8	16	246	59	23.98	195
luc	27	8	27	246	43	17.48	84
max512	512	9	6	501	24	4.79	1329
sex	23	9	14	501	127	25.35	418
newtpla1	4	10	2	1012	973	96.15	1604
newtpla2	9	10	4	1012	967	95.55	3163
clpl	20	11	5	2035	1095	53.81	6394
newapla1	10	12	7	4082	3769	92.33	14917
newapla	17	12	10	4082	3649	89.39	16935
newcwp	11	4	5	10	0	0	<1
dc1	15	4	7	10	0	0	<1
prom2	287	9	21	501	0	0	337

6. Conclusion and Future Works

A computer program as an application was developed to determine decomposability of a system 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. The several systems with different parameters were developed. The experimental results are interesting and show that usually a system has many solutions when it is decomposable. In the most cases a system has more solutions when the number of its functions is few or its conjunctions contain more don't care values.

As a future work, optimization in encoding of compact table is proposed, because it has direct influence on 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.

7. Acknowledgement

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.

8. References

[1] Yu.V. Pottosin, and E. Shestakov, "Choice of Free Arguments in Decomposition of Boolean Functions Using the Ternary Matrix Cover Approach," *In the 5th Int. Conf. on Neural Networks and Artificial Intelligence* (*ICNNAI*), Brest, Belarus, Jun. 2010, pp. 123-127.

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

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

[4] A. Martinelli, "Advances in Functional Decomposition: Theory and Applications,"*Doctoral Dissertation, Royal Institute of Technology* (*KTH*), Stockholm, Sweden, 2006.

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

[6] M. Rawski, "Heuristic Algorithm of Bound Set Selection in Functional Decomposition for Heterogeneous FPGAs," *In21st Int. Conf. on Systems Engineering* (*ICSEng*), Las Vegas, USA, Aug. 2011, pp. 465-466.

[7] V. Muthukumar, R. J. Bignall, and H. Selvaraj, "An efficient variable partitioning approach for functional decomposition of circuits," *J. of Systems Architecture*, vol. 53, (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, (9), 2000, pp. 1081-1086.

[9] S. Hassoun, and T. Sasao, "Logic Synthesis and Verification," *The Springer Int. Series in Engineering and Computer Science*, Kluwer Academic Publishers, 2001.

[10] A. Zakrevskij, Yu. V. Pottosin, and L. Cheremisinova, "Optimization in Boolean Space," Tallinn, Estonia, TUT Press, 2009.

[11] V.I. Romanov, "Tools development for logic designing," *The 4th Int. Conf. on Computer-Aided Design of Discrete Devices*, Minsk, Belarus, 2001, pp. 151-170 (In Russian).

[12] V.I. Romanov, "Tools for programming Boolean calculations," *InAbstracts of ECCO XVIII Conf.; Combinatorics for modern manufacturing, logistics and supply chains*, Minsk, Belarus, May. 2005, pp. 57-58 (In Russian).

[13] *Abailabe online at*: http://www1.cs.columbia.edu/~cs4861/sis/espresso-examples/ex/

[14] D. E. Knuth, "The Art of Computer Programming," First ed., vol. 4A, Combinatorial Algorithms, part 1, Reading, Massachusetts, Addison-Wesley, 2011.

[15] M. Rawski, "Evalutionary Algorithms in Decomposition-based Logic Synthesis, Evalutionary Algorithms" *Prof. Eisuke Kita* (*Ed.*), InTech Publisher, 2011.