Journal of Mathematical Sciences 3 (2016) 74-81 **Betty Jones & Sisters Publishing**
ISSN 2372-5214 **http://www.bettyjonespub.com**

The Search of a Cut in a Graph Used in Logical Design

Yu.V. Pottosin[1], S.A. Pottosina[2]

1. United Institute of Informatics Problems, NAS of Belarus

2. Byelorussian State University of Informatics and Radio-Electronics

Abstract: Two optimization problems in logical design are considered: decomposition of Boolean functions and state assignment of a finite automaton. A common approach to those problems is suggested. This approach connected with the search a maximal cut in a graph with weighted edges. Heuristic methods based on this approach to solve the problems are suggested.

Keywords: logical design, decomposition of Boolean functions, finite automaton, state assignment, cut in a graph.

1. Introduction

According to [1], given a graph $G = (V, E)$ with vertex set V and edge set E and some disjoint subsets $V_1, V_2 \subseteq V$, the cut is a set of edges whose ends are in different sets, V_1 and V_2. The problem of finding a maximum cut in a graph with weighted edges, i.e. such a cut whose sum of edge weights is maximal, is considered. This problem arises in logical design when it is necessary to come from multi-valued variables to Boolean vectors.

We consider two problems of logical design: two-block disjoint decomposition of completely specified Boolean functions solved by the tabular method [2] and state assignment of finite automata with the goal of the minimum of area complexity [3] and low power consumption of the implementing circuit [4]. When solving the problem of decomposition, the input system of Boolean functions is given in the form of a two-dimensional table, and it necessary to encode the columns of the given table with Boolean vectors. In the case of state assignment of finite automata, any state is assigned with a Boolean vector in order to obtain a system of Boolean functions that is necessary to construct a logical circuit. In both cases, the complexity of resulting Boolean functions depends strongly on the variant of encoding. The aim of encoding is reduction of the number of different conjunctive terms in disjunctive normal forms of resulting Boolean functions.

Corresponding author: Yu.V. Pottosin, Ph.D., research fields: combinatorial problems in logical design of discrete devices. E-mail: pott@newman.bas-net.by.

2. Decomposition of Boolean Functions

The problem of decomposition of Boolean functions is one of important problems in the field of logical design. This makes it be an object of great attention from many researchers. As it is shown in the review [5], many papers had been written on this topic. Decomposition of a system of Boolean functions that describes behavior of a discrete device leads to partition the device into separate blocks. It makes easier the subsequent procedure of logical design of the device and, sometimes, makes smaller the chip implementing the given functions. We consider this problem in the following statement. Given a system $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ of completely specified Boolean functions where $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))$, $\boldsymbol{y} = (y_1, y_2, ..., y_m)$, find a superposition $\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$, $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$ where $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are vector variables whose components are Boolean variables in subsets Z_1 and Z_2, correspondingly, that form a partition of the set $X = \{x_1, x_2, \ldots, x_l\}$. At that, the number of components of vector variable $\boldsymbol{w}$ must be less than that of $\boldsymbol{z}_1$. We add to this that the complexities of the functions $\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{g}(\boldsymbol{z}_1)$ must be as low as possible.

The tabular method for decomposition [2] provides for representation of a given system of Boolean functions in the form of two-dimensional table M whose columns correspond to values of vector variable $\boldsymbol{z}_1$ and rows to values of $\boldsymbol{z}_2$. The entry at column $\boldsymbol{z}_1$ and row $\boldsymbol{z}_2$ is the corresponding value of vector $\boldsymbol{y}$. The different columns of M being assigned with different Boolean vectors, we obtain the function $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$, and its values are those vectors. The function $\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ is obtained by substitution of $\boldsymbol{z}_1$ in $\boldsymbol{f}(\boldsymbol{z}_1, \boldsymbol{z}_2) = \boldsymbol{f}(\boldsymbol{x})$ by $\boldsymbol{w} = \boldsymbol{g}(\boldsymbol{z}_1)$. The complexity of functions $\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{g}(\boldsymbol{z}_1)$ depends strongly on the variant of such an assignment.

In [6] a method for encoding the columns of table M by Boolean vectors targeting simplification of functions $\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{g}(\boldsymbol{z}_1)$ is described. The method uses a multi-step process where at any i-th step the variable w_i is introduced that is a component of vector $\boldsymbol{w}$, and its values are defined. For this, a graph $G = (V, E)$ is constructed whose vertices correspond to columns of table M. Any column of M is considered as a vector of dimension sm where s is the number of rows of M and m is the number of components of vector $\boldsymbol{y}$. The initial meaning of G is a complete graph with weighted edges. The edge weight is the Hamming distance h between the columns corresponding to the edge ends.

At the i-th step of the process, a maximal cut is found in the graph G. The cut is represented by the pair V_1, V_2 of subsets of vertex set V, and the variable w_i gets the value 0 (or 1) for the columns corresponding to the vertices in V_1 and 1 (or 0) for columns corresponding to the vertices in V_2. Then, The edges connecting V_1 and V_2 are removed, and the next $(j + 1)$-th step is executed as it is described above. The process comes to the end when G becomes an empty graph. The using of the weight function h as it is defined above makes more possibility of merging terms in disjunctive normal forms of functions $\boldsymbol{\varphi}(\boldsymbol{w}, \boldsymbol{z}_2)$ and $\boldsymbol{g}(\boldsymbol{z}_1)$.

As an example, let the system of Boolean functions $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ where
$\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$, be given by disjunctive normal forms represented by the following matrices:

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \left[\begin{array}{ccccccc} 1 & 0 & 1 & 1 & - & 1 & - \\ - & 1 & 0 & - & 1 & - & - \\ 1 & 0 & 1 & 0 & - & 0 & 0 \\ 0 & 0 & - & - & 0 & 0 & 1 \\ - & - & 0 & - & - & - & - \\ - & - & 1 & - & 0 & 0 & - \\ 1 & - & - & 0 & 0 & - & - \\ - & 0 & 0 & 1 & 1 & - & 0 \\ 1 & - & 1 & - & - & - & - \\ 1 & - & - & 0 & 1 & 0 & 0 \\ 1 & 0 & - & - & 1 & - & 0 \\ - & 1 & - & 1 & 1 & - & - \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{array} \end{array},$$

$$\begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right] \end{array}.$$

Let $\boldsymbol{z}_1 = (x_2, x_5, x_6, x_7)$ and $\boldsymbol{z}_2 = (x_1, x_3, x_4)$. Then, the table M is represented by Table 1. To encode its columns, three variables are sufficient and the function $\boldsymbol{g}(\boldsymbol{z}_1) = \boldsymbol{w}$ can be obtained where $\boldsymbol{w} = (w_1, w_2, w_3)$.

Table 1

	$x_2\, x_5\, x_6\, x_7$						
$x_1x_3x_4$	0101,101–	0111,001–	111–,11–1	100–,0000	0110,0100	1100	0001
010	0000	0000	0000	0100	0000	0000	1111
000	1110	1110	1111	1110	1110	1111	1111
011	0000	0000	1111	0100	0000	1111	1111

111	1110	1111	1111	1110	1111	1111	1110
001	1110	1110	1111	1110	1111	1111	1111
100	1110	1110	1111	1110	1111	1111	1110
101	1110	1110	1111	1110	1111	1111	1110
110	1110	1110	1110	1110	1111	1111	1110

The values of vector $\boldsymbol{w}$ may be defined arbitrary. For example, if the columns are simply numbered in binary system we obtain the following matrices representing disjunctive normal forms of functions $\boldsymbol{y} = \boldsymbol{\varphi}(w_1, w_2, w_3, x_1, x_3, x_4)$ and $\boldsymbol{w} = \boldsymbol{g}(x_2, x_5, x_6, x_7)$ (after minimization):

$$\begin{array}{cccc} x_2 & x_5 & x_6 & x_7 \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & - & 1 & 1 \\ 1 & 1 & 1 & - \\ 0 & 1 & - & 0 \\ 1 & 1 & 0 & 0 \\ - & 0 & 0 & 0 \\ 1 & - & 0 & 1 \\ 1 & 0 & 0 & - \end{array}\right. & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4, \\ 5 \\ 6 \\ 7 \\ \left.\begin{array}{c} 8 \end{array}\right. \end{array} \quad \begin{array}{ccc} w_1 & w_2 & w_3 \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right. \end{array};$$

$$\begin{array}{cccccc} x_1 & x_3 & x_4 & w_1 & w_2 & w_3 \\ \left[\begin{array}{cccccc} 1 & - & - & 1 & 0 & - \\ 1 & 1 & 1 & - & 0 & 1 \\ 0 & - & - & 1 & 1 & - \\ 0 & 0 & - & - & - & - \\ - & - & 1 & 0 & 1 & 0 \\ - & 0 & - & 1 & - & 1 \\ - & - & 1 & 1 & - & 1 \\ - & 0 & - & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & - & - \\ 1 & - & - & - & - & - \\ - & - & - & - & 1 & 1 \end{array}\right. \end{array} \quad \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ \left.\begin{array}{c} 11 \end{array}\right. \end{array},$$

$$\begin{array}{cccc} y_1 & y_2 & y_3 & y_3 \\ \left[\begin{array}{cccc} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right. \end{array}.$$

The described method for encoding the columns of Table 1 is executed on this example in three steps. Let construct the graph $G = (V, E)$ with weighted edges by Hamming distances between corresponding columns of Table 1. The first two unfolded columns of the table are

00001110000011101110111011101110

00001110000011111110111011101110

The distance between them is 1. At the beginning of Step 1, G is a complete graph. All the weights in G are given in Table 2.

Table 2

v_2	v_3	v_4	v_5	v_6	v_7	
1	9	2	5	10	10	v_1
	8	3	4	9	11	v_2
		9	6	1	7	v_3
			7	10	8	v_4
				5	13	v_5
					8	v_6

Application the “greedy” method from [7] for constructing a cut in the given graph G allows us to obtain $V_1 = \{v_1, v_2, v_4, v_5\}$ and $V_2 = \{v_3, v_6, v_7\}$ at Step 1. We introduce the variable w_1 with value 0 at vertices in V_1 and value 1 at vertices in V_2. After removing from G the edges connecting the vertices in V_1 with vertices in V_2, we obtain Table 3.

Table 3

v_2	v_3	v_4	v_5	v_6	v_7	
1		2	5			v_1
		3	4			v_2
				1	7	v_3
			7			v_4
						v_5
					8	v_6

At Step 2 we obtain $V_1 = \{v_1, v_3, v_4, v_6\}$ and $V_2 = \{v_2, v_5, v_7\}$, and at Step 3 $V_1 = \{v_1, v_2, v_6\}$ and $V_2 = \{v_3, v_4, v_5\}$. So the columns of Table 1 get the following codes:

000 010 101 001 011 100 11–

This encoding defines functions $\boldsymbol{y} = \boldsymbol{\varphi}(w_1, w_2, w_3, x_1, x_3, x_4)$ and $\boldsymbol{w} = \boldsymbol{g}(x_2, x_5, x_6, x_7)$ by the following matrices that are more compact than the previous ones:

$$\begin{array}{cccc} x_2 & x_5 & x_6 & x_7 \\ \left[\begin{array}{cccc} 0 & – & 1 & – \\ 1 & 1 & – & – \\ 0 & 0 & 0 & 1 \\ 0 & 1 & – & 0 \\ – & 0 & 0 & – \\ 1 & 1 & 1 & – \\ 1 & 1 & – & 1 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}, \quad \begin{array}{ccc} w_1 & w_2 & w_3 \\ \left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{array}\right] \end{array};$$

$$\begin{array}{cccccc} x_1 & x_3 & x_4 & w_1 & w_2 & w_3 \\ \left[\begin{array}{cccccc} 0 & – & – & 1 & 1 & 0 \\ – & 0 & 1 & – & 1 & 1 \\ 1 & – & – & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & – \\ – & – & 1 & 1 & – & – \\ – & 0 & – & 1 & 0 & – \\ 1 & – & – & – & 1 & 1 \\ – & 0 & – & – & – & – \\ 1 & – & – & – & – & – \\ – & – & 0 & 0 & – & 1 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \end{array},$$

$$\begin{array}{cccc} y_1 & y_2 & y_3 & y_3 \\ \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \end{array}.$$

3. State Assignment of Finite Automata

One of behavioral models of a discrete device is a finite automaton that consists of a set of inputs A, a set of outputs B, a set of states Q and two functions, output function $\Phi(a, q) = b$ and transition function $\Psi(a, q) = q^+$ where $a \in A$, $b \in B$, $q, q^+ \in Q$ and q^+ is the state, to which the automaton goes from q at the input a.

In the synthesis of a logical circuit, the functions Φ and Ψ are transformed into a system of Boolean functions by introducing Boolean vectors instead of abstract symbols a, b and q. Often, inputs and outputs are presented in the functional description of the synthesized circuit as binary signals. The problem is to assign Boolean vectors $(z_1, z_2, \ldots, z_k)$ to abstract symbols q of states of a given automaton according to some optimization criterion. In this case, any state of the automaton will be presented in the circuit as a set of states of binary memory elements where the state of i-th memory element is the value of intermediate variable z_i. A Boolean vector $(z_1, z_2, \ldots, z_k)$ assigned to a state of an automaton is called the code of the state.

At present time, a great attention is paid to decreasing power consumption in designing discrete devices based on CMOS technology. It is caused by the tendency to increase the working time of power supply for portable devices and, on the other hand, by the tendency to lower acuity of the problem of heat rejection in designing VLSI circuits. Therefore one of the main optimization criteria in designing discrete devices is amount of power consumption. As it is said in [8, 9] the power consumption of a circuit built on the base of CMOS technology is proportional to switching activity of its logical and memory elements. It allows solving this problem at the level of logical design. In particular, decreasing power consumption can be achieved in state assignment [4]. The states of an automaton must be encoded in such a way that during the transition between its states, as few as possible memory elements change their state.

We consider the probabilities of transitions between states, and the larger probability of transition between any pair of states, the less number of different components in the codes of that states change their values, no matter what is the direction of the transition. To calculate the probabilities of transitions between

states of an automaton, the following assumption are accepted: the automaton must be completely specified; all the states are mutually accessible, i. e. for any two states there exists an input sequence, which transfers the automaton from one of them to the other. The automaton is supposed to be working infinitely long.

The probability of transition from state q_i to state q_j caused by input a is equal to the probability of input a. If there are several inputs causing the transition from state q_i to state q_j, the conditional probability p'_{ij} of this transition is equal to the sum of those probabilities. The condition is that the automaton is at the state q_i. The absolute probability p_{ij} of transition from state q_i to state q_j during all the time the automaton working is equal to $P_i p'_{ij}$ where P_i is the probability of the automaton being at the state q_i.

To calculate the probabilities P_i, $i = 1, 2, \ldots, |Q|$, the Chapmann–Kolmogorov equations for discrete-time Markov Chains [10] can be used. Similarly to Kirchhoff's low of electrical engineering, one may say that the sum of the transition probabilities to some state is equal to the sum of the transition probabilities from this state. Based on the considerations above, the following equations can be derived:

$$\sum_{i=1}^{|Q|} P_i p'_{ij} = P_j, \qquad j = 1, 2, \ldots, |Q|,$$

$$\sum_{i=1}^{|Q|} P_i = 1.$$

The probabilities p'_{ij} must be known. So, having solved this system of equations the probabilities P_i will be obtained. As it was said above, the absolute probability p_{ij} is defined as $p_{ij} = P_i p'_{ij}$.

The values of internal variables $z_1, z_2, \ldots, z_k$ are specified in the following way.

The current situation in this process is characterized by partial codes of states $(z_1, z_2, \ldots, z_j)$, $j < k$, and a weighted graph $G = (V, E)$ whose vertices correspond to the states of the automaton. Two vertices of the graph are adjacent if the corresponded states have the same partial codes. Each edge $v_s v_t \in E$ is weighted with $w_{st} = 1 - p^*_{st}$ where p^*_{st} is the probability of the transition between the states q_s and q_t corresponding to vertices v_s and v_t, no matter in which direction, i. e. $p^*_{st} = p_{st} + p_{ts}$ where p_{st} is the probability of the transition from q_s to q_t. Evidently, to lower the switching activity of memory elements, the short Hamming distance between the codes of states q_s and q_t should be made if the probability p^*_{st} is high.

The process of state assignment of a given automaton is a sequence of steps. At the j-th step, a partition of the vertex set V of G into two subsets, V_1 and V_2, is obtained, the variable z_j is introduced and receives the value 0 (or 1) for the states corresponding to vertices in V_1 and value 1 (or 0) for the states corresponding to vertices in V_2. Then, the edges connecting vertices in V_1 with vertices in V_2 are removed, and the next step, $(j + 1)$-th one is executed. The process is over when graph G becomes empty.

The problem to partition V into V_1 and V_2 is reduced to finding the maximum cut of G, i. e. finding such a partition that the sum of weights of the edges connecting the vertices of V_1 with the vertices of V_2 would be maximal. At the last step, the graph G is bipartite, and its edges connect the vertices corresponding to the states, the probabilities of transition between which are comparatively high. So, the Hamming distance between the codes of those states will be equal to one.

The model of automaton with abstract state [3] is used here, which is described by one multi-valued variable q and many Boolean input and output variables, $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ instead of a and b, respectively. Let the transitions between states of a given automaton be shown in Table 4 where rows and columns correspond to the states of the automaton, and the entry at i-th row and j-th column is the condition of transition from q_i to q_j.

Table 4

	q_1	q_2	q_3	q_4	q_5	q_6
q_1			– – – 0			– – – 1
q_2			1 1 – 1	0 – – –	1 – – 0	1 0 – 1
q_3				– – 1 –		– – 0 –
q_4	– 0 0 1		– 1 0 1			– – 0 0,– – 1 –
q_5			1 – 0 1	0 – 0 1	– – 0 0	– – 1 –
q_6		1 1 – 1	1 1 – 0	0 – – –	1 0 – –	

The automaton has six states and four input variables. The output variables are not taken into consideration in this task. For example, the entry at the row q_2 and column q_3 shows that the automaton goes from q_2 to q_3 when $x_1 = 1$, $x_2 = 1$, $x_4 = 1$ and no matter what is x_3 equal to. An empty entry means that the corresponding transition does not exist.

Assume that the probabilities of input signals are distributed uniformly. Then the conditional probabilities p'_{ij} of transitions (the transition from q_i to q_j when the automaton is at the state q_i) are represented in Table 5.

To find out the probabilities of states of the automaton, the following equation system must be solved:

$P_4 = 8P_1$; $P_6 = 8P_2$;

$4P_1 + P_2 + P_4 + P_5 + P_6 = 8P_3$;

$4P_2 + 4P_3 + P_5 + 4P_6 = 8P_4$; $P_2 + P_5 + P_6 = 4P_5$;

$4P_1 + P_2 + 4P_3 + 6P_4 + 4P_5 = 8P_6$;

$P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$.

Having solved these equations we obtain $P_1 = 10/279$, $P_2 = 116/2511$, $P_3 = 103/837$, $P_4 = 80/279$, $P_5 = 116/837$, $P_6 = 928/2511$.

The absolute probabilities p_{ij} of transitions are shown in Table 6.

Table 5

	q_1	q_2	q_3	q_4	q_5	q_6
q_1			1/2			1/2
q_2			1/8	1/2	1/4	1/8
q_3				1/2		1/2
q_4	1/8		1/8			3/4
q_5			1/8	1/8	1/4	1/2
q_6		1/8	1/8	1/2	1/4	

Table 6

	q_1	q_2	q_3	q_4	q_5	q_6
q_1	5/279					5/279
q_2			29/5022	58/2511	29/2511	29/5022
q_3				103/1674		103/1674
q_4	10/279		10/279			60/279
q_5			29/1674	29/1674	29/837	58/837
q_6		116/2511	116/2511	464/2511	232/2511	

The edge weights $w_{st} = 1 - p^*_{st}$ of allowed accuracy of graph G where p^*_{st} is the probability of the transition in both directions between the states q_s and q_t respective to vertices v_s and v_t are shown in Table 7.

At the first step we obtain the cut defined as $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$. We introduce the variable z_1 with value 0 at vertices in V_1 and value 1 at vertices in V_2. After removing from G the edges connecting the vertices in V_1 with vertices in V_2, we obtain Table 8.

At Step 2 we obtain $V_1 = \{v_2, v_5\}$ and $V_2 = \{v_1, v_3, v_4, v_6\}$, and at Step 3 $V_1 = \{v_1, v_4\}$ and $V_2 = \{v_2, v_3, v_5, v_6\}$. The result of the process of state assignment is the following encoding of the states: $q_1 - 000$, $q_2 - 011$, $q_3 - 001$, $q_4 - 100$, $q_5 - 111$, $q_6 - 101$.

Table 7

v_2	v_3	v_4	v_5	v_6	
1	0.9821	0.9642	1	0.9821	v_1
	0.9942	0.9769	0.9885	0.9480	v_2
		0.9027	0.9827	0.8929	v_3
			0.9827	0.6002	v_4
				0.8395	v_5

Table 8

v_2	v_3	v_4	v_5	v_6	
1	0.9821				v_1
	0.9942				v_2
					v_3
			0.9827	0.6002	v_4
				0.8395	v_5

The quality of a solution of the state assignment problem can be appreciated by the value of $D = \sum p^*_{ij}(d_{ij} - 1)$ where p^*_{ij} is the probability of the transition between states q_i and q_j in both directions, d_{ij} is the Hamming distance between the codes of q_i and q_j, and summation is made over all the pairs of states. This value was introduced in [11]. Evidently, the less value of D, the better solution, and $D = 0$ if any transition between states corresponds to switching only one memory element in the circuit implementing the automaton.

For the variant of encoding obtained above we have $D = 0.196$. If we take arbitrary encoding, e. g. assign to the states q_1, q_2, q_3, q_4, q_5, q_6 the sequence of natural numbers with zero in binary system – 000, 001, 010, 011, 100, 101 – then we obtain $D = 0.7369$. It is clear that the first variant is better then the second one.

4. Search of a Cut in a Graph

Let $G = (V, E)$ be a graph with edges weighted by real numbers. A cut in G close to a minimal one can be found out as follows. Because of heuristic approach to the problems considered here, it is enough to use the "greedy" algorithm from [7] that is a sequence of steps, at each of which a vertex v is selected in V_2 and carried to V_1. The initial meanings are $V_1 = \varnothing$ and $V_2 = V$, and the vertex v is selected in the following way.

Let $d(v)$ be the sum of weights of the edges incident with v, and let $c(v)$ be the sum of weights of edges connecting v with the vertices in V_1. The transfer of the vertex v from V_2 to V_1 changes the sum of weights of edges connecting the vertices of V_1 with the vertices of V_2 by $h(v) = d(v) - 2c(v)$. At the first step, this value is equal to $d(v)$. At the subsequent steps it can be negative. At any step, the vertex v with maximum value of $h(v)$ is selected. The process comes to the end when this value is not positive for all the vertices in V_2.

To explain this process, let us take for example the graph G that is given by Table 2. According to Table 2 we have $d(v_1) = 37$, $d(v_2) = 36$, $d(v_3) = 40$, $d(v_4) = 39$, $d(v_5) = 40$, $d(v_6) = 43$ and $d(v_7) = 57$.

At the beginning of the first step of partitioning the set V, $c(v_i) = 0$ and $h(v) = d(v_i)$ for any $v_i \in V_2$. The maximum is $d(v_7) = 57$ and the result of the first step is $V_1 = \{v_7\}$ and $V_2 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.

At the second step we have $h(v_1) = 17$, $h(v_2) = 14$, $h(v_3) = 26$, $h(v_4) = 23$, $h(v_5) = 14$ and $h(v_6) = 27$. The maximum is $h(v_6) = 27$ and then $V_1 = \{v_6, v_7\}$ and $V_2 = \{v_1, v_2, v_3, v_4, v_5\}$.

At the third step we have $h(v_1) = -3$, $h(v_2) = -4$, $h(v_3) = 24$, $h(v_4) = 3$ and $h(v_5) = 4$. The maximum is $h(v_3) = 24$ and then $V_1 = \{v_3, v_6, v_7\}$ and $V_2 = \{v_1, v_2, v_4, v_5\}$. The third step is the final one, because the values of $h(v)$ for all $v \in V_2$ become negative.

5. Conclusion

The graph theory methods have successful applications in combinatorial problems arising in logical design. The approach described here confirms this. Exact solutions of the considered problems of logical design can be obtained only by exhaustive search of many variants. It cannot be always executed because of non-polynomial complexity of the problems. Heuristic methods are of practical interest. Such methods are suggested here. They are intended for an automated system of logical design. Comparison of results obtained by the proposed methods with those obtained arbitrarily shows the expediency of the approach.

References

[1] N. Christofides, Graph Theory: An Algorithmic Approach, New York, London, San Francisco: Academic Press, 1975.

[2] Yu.V. Pottosin, E.A. Shestakov, Tabular Methods for Decomposition of System of Completely Specified Boolean Functions, Minsk: Byelorusskaya nauka, 2006 (in Russian).

[3] A. Zakrevskij, Yu. Pottosin, L. Cheremisinova, Design of Logical Control Devices, Tallinn: TUT Press, 2009.

[4] L. Benini, G. de Micheli, State assignment for low power dissipation, IEEE Journal of Solid-State Circuits 30 (3) (1995) 32-40.

[5] M.A. Perkowski, S. Grygiel, A Survey of Literature on Functional Decomposition, Version IV (Technical report), Portland: Portland State University, Department of Electrical Engineering, 1995.

[6] S. Taghavi Afshord, Yu.V. Pottosin, A new suboptimal decomposition algorithm based on the tabular method, Tanaevskie Chteniya: Proceedings of the 6th International Scientific Conference (March 27-28, 2014, Minsk), Minsk: UIIP NAS of Belarus, 2014, pp. 161-165.

[7] A.D. Zakrevskij, Graph coloring in decomposition of Boolean functions, Logical Design, Minsk: ITC NAS Belarus 5 (2000) 83-97 (in Russian).

[8] S. Muroga, VLSI System Design: When and How to Design Very-Large-Scale Integrated Circuits, New York: John Viley & Sons, Inc., 1982.

[9] M. Pedram, Power minimization in IC design: Principles and applications, ACM Trans. Design Automat. Electron. Syst. 1 (1996) 3-56.

[10] E. Macii, M. Pedram, F. Somenzi. High-level power modeling, estimation and optimization, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems 17 (11) (1998) 1061-1079.

[11] D. Zakrevskij, Algorithms for low power state assignment of an automaton, Informatika 29 (1) (2001) 68-78 (in Russian).