Monika WIŚNIEWSKA, Remigiusz WIŚNIEWSKI, Marian ADAMSKI
UNIWERSYTET ZIELONOGÓRSKI

Reduction of the Microinstruction Length in the designing process of Microprogrammed Controllers

mgr inż. Monika WIŚNIEWSKA

Ukończyła studia na Wydziale Elektrycznym Uniwersytetu Zielonogórskiego, o specjalności Inżynieria Komputerowa. Obroniła pracę magisterską w 2003 r.

Od 2003 r. jest słuchaczem studiów doktoranckich, specjalność informatyka. Jej zainteresowania naukowe to analiza systemów dyskretnych z wykorzystaniem hipergrafów.

e-mail: M.Wisniewska@weit.uz.zgora.pl

prof. dr hab. inż. Marian ADAMSKI

Profesor zwyczajny, dyrektor Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych. Członek IEEE, IEE, ACM, PTI oraz PTETiS.

e-mail: M.Adamski@iie.uz.zgora.pl

Abstract

The problem of the microinstruction length reduction is a very important part of the designing process of the microprogrammed controllers. Such a problem is NP-hard, therefore many various algorithms have been developed. Almost all proposed ideas are based on the traditional graph theory and its modifications (heuristics, stochastic, etc.).
In the paper, we propose the method of microinstruction length reduction, where the hypergraph theory is applied. A hypergraph permits to store and reduce the information about the compatibility classes in comparison with traditional graphs. The microinstruction length reduction is reached thanks to the calculation of the dual hypergraph and computation of its minimum transversal (minimal vertices cover).
All steps that are required in order to perform the microinstruction length reduction of microprogrammed controllers will be shown. The proposed method will be illustrated by way of example and compared with the traditional solution, based on the graph theory.

Keywords: hypergraph, compatibility class, microprogrammed controller (control unit), microoperation, microinstruction, reduction of the microinstruction length.

Redukcja rozmiaru mikroinstrukcji w procesie projektowania sterowników mikroprogramowanych

Streszczenie

Problem redukcji rozmiaru mikroinstrukcji jest ważnym etapem w procesie projektowania sterowników mikroprogramowanych. Jest to problem NP-trudny, dlatego też powstało wiele metod poszukujących rozwiązania. Zdecydowana większość zaproponowanych algorytmów bazuje na tradycyjnej teorii grafów.
W artykule zaprezentowano nowatorską metodę redukcji rozmiaru mikroinstrukcji sterowników mikroprogramowanych, częściowo opierającą się na rozwiązaniach klasycznych. Metoda bazuje na wykorzystaniu teorii hipergrafów do wyznaczenia klas kompatybilności dla poszczególnych mikrooperacji. Mikrooperacje, które są parami kompatybilne mogą zostać zakodowane z wykorzystaniem mniejszej liczby bitów. Dzięki temu rozmiar pamięci układu mikroprogramowanego może zostać w znacznym stopniu zmniejszony. Zaproponowane rozwiązanie bazuje na wyznaczeniu hipergrafu dualnego, a następnie znalezieniu jego minimalnej transwersali (minimalnego pokrycia wierzchołkowego).

dr inż. Remigiusz WIŚNIEWSKI

Dr inż. Remigiusz Wiśniewski jest absolwentem Uniwersytetu Zielonogórskiego (2003). Ukończył studia o specjalności Inżynieria Komputerowa. W roku 2000 odbył przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych.
Od roku 2003 pracuje jako asystent na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: R.Wisniewski@iie.uz.zgora.pl

Idea metody zostanie zilustrowana przykładem. Pokazane zostaną wszystkie kroki, jakie są niezbędne do zaprojektowania zmodyfikowanego układu pamięci. Zaproponowana metoda zostanie porównana z rozwiązaniami klasycznymi, bazującymi na teorii grafów.

Słowa kluczowe: hipergraf, klasa kompatybilności, sterownik (układ) mikroprogramowany, mikrooperacja, mikroinstrukcja, redukcja rozmiaru pamięci.

1. Introduction

A control unit is one of the most important parts of any digital system [1,2,3,4,5]. It can be found in almost all devices that contain microelectronics; such as computers (central processor unit, CPU), cellular phones, cars and even remote controllers. The control unit is responsible for managing all modules of the designed system - it sends adequate microinstructions that should be executed [6].

One of the realization of the control unit is a microprogrammed controller (also named as microprogrammed control unit) where the device is decomposed into two main parts. The first is responsible for addressing microinstructions that are kept in the control memory. The role of the second part is to hold and generate adequate microinstructions. Typically, the control memory is implemented as a ROM or RAM memory.

Many controllers (especially realized as a Complex Instruction Set Computers, *CISC*) have a long microinstruction length [7]. It may cause serious problems in the prototyping process. If the design is realized as a System-On-Programmable-Chip (*SoPC*), the memory can be implemented with dedicated memory blocks of Field Programmable Gate Arrays (*FPGA*). However, if the microinstruction length exceeds the length of the dedicated memory block offered by an FPGA, the controller's memory ought to be decomposed. In case of controllers implemented as a System-On-Chip (*SoC*), the memory is treated as an independent module. It means that each additional bit in the microinstruction width increases the total cost of the memory and the whole device. Therefore, the microinstruction length reduction is a very important part of the designing process of the microprogrammed controllers in a digital system.

2. Problem formulation and current state of the art

The idea of the microinstruction length reduction is basically based on the special encoding of microoperations. Two microoperations may be encoded together, if they are not executed concurrently, at the same time. It means that they are pairwise compatible. Similarily, two microoperations are incompatible, if they are executed at the same time.

The problem of the reduction of the microinstruction length is NP-hard [8]. Almost all methods presented in previous works are based on the graph theory [9,10,11]. Here proper microoperations

are represented by vertices while their compatibility is shown by an edge (if there is an edge between two vertices, it means that two microoperations are compatible). Next, compatibility classes are formed. All microoperations that are pairwise compatible are grouped into classes (it is obvious that compatibility classes correspond to cliques in a graph). The microinstruction length minimization problem is equivalent to finding the partition of the compatibility graph into disjoint compatibility classes, where the length of a new (encoded) word is minimal. In practice it means, that the minimum covering of the graph ought to be calculated. Since the graph covering problem is NP-hard [7,8], there were many various ideas presented in previous works. Most of algorithms are based on the finding and analyzing all maximal subgraphs. Such a technique is known as *the maximal clique problem* [10,11,12].

In this paper we propose a new approach of the microinstruction length reduction. The main idea is an application of the hypergraph to represent the relations between compatibility classes. A hypergraph is generalization of the graph. Its edges can connect any number of vertices [13,14]. Therefore, a hypergraph is much more efficient than any classic undirected graph [1,13,14,15]. Its application permits to store and reduce information about compatibility classes and their relations needed for further synthesis. Moreover, the current analysis methods of hypergraphs are much faster compared with traditional solutions.

3. Main definitions

A *hypergraph* is generalization of the graph. Its edges - known as hyperedges - can connect any number of vertices [1,13,14], while classic graph can connect only two vertices. Formally, a hypergraph is a pair (V,E) where $V=\{v_1,\ldots,v_X\}$ is a set of vertices, and $E=\{e_1,\ldots,e_M\}$ is a set of edges. Hyperedges are arbitrary sets of vertices, and can therefore contain an arbitrary number of vertices. Figure 1 shows the graphical representation of the hypergraph. One of the most popular representations of hypergraph is incidence matrix where vertices are represented by columns and hyperedges by rows of the matrix.

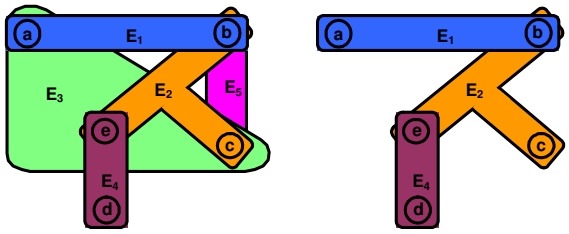

Fig. 1. Hypergraph H and its minimum covering
Rys. 1. Hipergraf H i jego minimalne pokrycie

Hypergraph H^* is called the *dual hypergraph of H*, if its vertices correspond to edges of H, and respectively its edges correspond to vertices of H. The incidence matrix of H^* is the transpose of the incidence matrix of H, and similarly $(H^*)^*=H$ [14].

A *transversal* (*hitting set*, *vertices cover*) of a hypergraph H is defined as a set of vertices $T \subset V$, so that each hyperedge of H is incident with at least one vertex in T:

$$T \cap e_i \neq \varnothing \ (i = 1, \ldots, m). \tag{1}$$

A *minimum transversal* (*transversal number*) of the hypergraph H is defined to be the minimum number of vertices in the transversal. It is represented as:

$$\tau(H) = \min | T |. \tag{2}$$

There are several methods to compute the minimum transversal of a hypergraph. It can be achieved through the reduction of the incidence matrix (also known as *exact covering*), via heuristic methods (i.e. *backtracing* algorithms) or stochastic solutions (like *greedy* method) [1,13,15,19].

An *edge cover* (or just *cover*) of a hypergraph H is defined as the C set of edges, so that each vertex is incident with at least one edge in C. A *minimum cover* of a hypergraph H is defined to be the minimum number of edges in the C set. An exemplary covering of the hypergraph H is shown in the Fig. 1.

Let's point out, that the minimum transversal problem refers to the minimum covering problem (and vice versa). The minimum cover of a hypergraph H can be solved by calculation of the minimum transversal $\tau(H^*)$ of a dual hypergraph H^*. Moreover, the minimum transversal of $\tau(H)$ can be achieved via computation of the minimum cover of the dual hypergraph H^*.

A *microoperation* y_n is an action (output) generated by a control unit, that is executed by a driven object (also called as a *datapath*). The set (collection) $Y = \{y_1, \ldots, y_N\}$ of microoperations that are executed at the same time is defined as *microinstruction* (also named as *word* or *microword*). In the microprogrammed controllers, microinstructions are organized into memories and kept in the *control memory* [16,17,18].

4. Main idea of the proposed method

The reduction of the microinstruction length can be divided into the following steps:

1. *Formation of the C_C set of compatibility classes* for all microoperations kept in the control memory. As it was already mentioned, two microoperations are compatible if they are not executed concurrently (at the same of microinstruction). At this stage, all possible compability classes are calculated and represented as the set $C_C = \{C_1,\ldots,C_K\}$.
2. *Determination of the cost (weight) of each compatibility class*. A *cost* (*weight*) of the compability class C_i is equal to the minimum number of bits that are required for its encoding. Each compatibility class is encoded with the natural binary code. The weight of the compability class C_i can be easily found as:

$$L_i = \lceil \log_2(|C_i|+1) \rceil. \tag{3}$$

   An additional bit is necessary for representation of the no-operational state (where none of microoperations in the C_i set are to be executed).
3. *Formation of the H compatibility hypergraph*. The hypergraph represents relations between compability classes and microoperations. The incidence matrix A of the hypergraph H may contain the following values [19]:

$$A_{ij} = \begin{cases} 1 \Rightarrow compability \\ 0 \Rightarrow incompability \end{cases}, \tag{4}$$

   where $i = \{1, \ldots, K\}$ represents the i-th compability class, and $j = \{1, \ldots, N\}$ means the j-th microoperation. If the filed A_{ij} of the incidence matrix contains 1, it means that j-th microoperation belongs to the i-th compability class.
4. *Transformation of the* H*, to the dual* H* *hypergraph*. At this step the dual H^* hypergraph is formed. Formally it means the transposition of the incidence matrix A to the matrix A^*.
5. *Computation of the minimum transversals* τ(H*) *of the dual* H* *hypergraph*. Let's point out that this stage may be solved with application of any algorithm of transversals recognition. However, due to the comparison with graph theory, this paper deals with the basic reduction algorithm. Such a solution finds exact vertices covers and it is based on a consecutive reduction of vertices and edges from the hypergraph. The algorithm is perfectly described in [1].

6. *Calculation of the total cost of each minimum transversal*. At this stage for each minimum transversal $\tau_s \in \tau$ the total covering cost is calculated. This value can be determined as:

$$W_s = \sum L_i \quad (i=1, \ldots, I), \tag{5}$$

where W_s is a total weight (cost) of a τ_s transversal and it is equal to a L_i sum of weights of all I compability classes that belong to the τ_s transversal. The transversal τ_{Min} with the lowest W_s weight is selected for the further analysis.

7. *Encoding of compatibility classes* that belong to the τ_{Min} transversal with the lowest W_s weight. The number of required bits is equal to W_s ($Q=\{q_1, \ldots, q_{Ws}\}$). The encoding style is not important, however in this paper we will use natural binary code.

8. *Formation of the new control memory content*. The content of the control memory is now determined as a concatenation of codes achieved in the previous step (each new microinstruction is formed as a concatenation of Q codes of encoded compability classes). The total width of new microinstruction is equal to W_s, thus the reduction of the control memory can be calculated as:

$$t = \left(1 - \frac{\sum_{i=1}^{|S|} L_i}{N} \right) \cdot 100\ \%\ , \tag{6}$$

where t means the (%) gain of the control memory volume achieved by the microinstruction length reduction; $|S|$- is the number of compability classes that belongs to the τ_{Min} transversal; L_i- is the weight of the i-th class that belongs to the transversal τ_{Min}; N- is the original size of each microinstruction (number of microoperations) before reduction.

5. Exemplary application of the proposed method

The idea of microinstruction length reduction based on the hypergraph theory will be illustrated by way of example. Let's assume the hypothetical content control memory with N=6 microoperations $Y = \{y_1, \ldots, y_6\}$, that are formed into four microinstructions $\mu = \{\mu_1, \ldots, \mu_4\}$. The total volume of the control memory is equal to $V = 4*6 = 24$ (Tab. 1).

Tab. 1. The initial content of the control memory
Tab. 1. Pierwotna zawartość pamięci sterownika

Micro-instruction	Microoperation					
	y_1	y_2	y_3	y_4	y_5	y_6
μ_1	0	1	0	0	0	1
μ_2	0	1	0	1	0	0
μ_3	1	0	0	0	1	0
μ_4	0	0	1	0	1	0

According to the algorithm presented in the previous section, at the beginning the set of compability classes should be formed. For the presented example there are $K = 4$ compability classes $C = \{C_1, ..., C_4\}$, where: $C_1 = \{y_1, y_2, y_3\}$, $C_2 = \{y_1, y_3, y_4, y_6\}$, $C_3 = \{y_2, y_5\}$, $C_4 = \{y_4, y_5, y_6\}$. Next, the weight of each compability class is calculated. The class C_1 contains $|C_1| = 3$ elements, therefore its weight will be equal to $L_1 = \lceil \log_2(|C_1|+1) \rceil = \lceil \log_2(3+1) \rceil = 2$. Similarily weights for all remaining classes are determined, and finally $L_2 = 3$; $L_3 = 2$; $L_4 = 2$.

At the third stage, the compability H_1 hypergraph is formed. Such a hypergraph will contain $|V| = 6$ vertices, that refer to microoperations ($V=\{y_1, \ldots, y_6\}$) and $|E| = 4$ edges, that correspond to proper compability classes ($E=\{C_1, \ldots, C_4\}$). The compability hypergraph and its incidence A_1 matrix are shown in the Fig. 2.

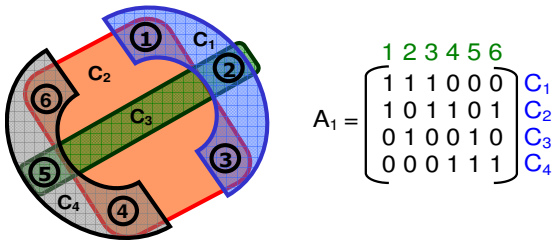

Fig. 2. The Compability H_1 hypergraph and its incidence A_1 matrix
Rys. 2. Hipergraf kompatybilności H_1 i jego macierz incydencji A_1

Next, the dual H_1* hypergraph is formed. Here the set of vertices of H_1 is transformed into the set of edges of H_1*, and respectively edges of H_1 are transformed to vertices of H_1*. The dual H_1* hypergraph and its incidence matrix A_1* are presented in the Fig. 3.

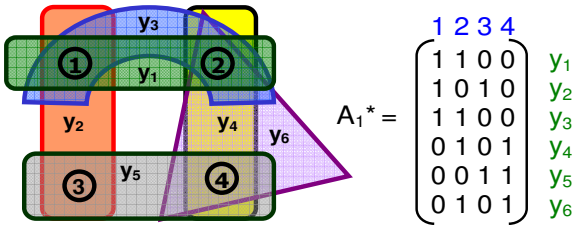

Fig. 3. The Dual H_1* hypergraph and its incidence A_1* matrix
Rys. 3. Hipergraf kompatybilności H_1* i jego macierz incydencji A_1*

There are two minimum transversals of the dual H* hypergraph. Both contain two vertices, the first one consists of $\tau_1=\{1,4\}$, while the second one includes $\tau_2=\{2,3\}$. Similarily, it means that the initial H_1 hypergraph can be covered in two ways: via hyperedges C_1 and C_4 or with C_2 and C_3. The minimum covering is shown in the Fig. 4.

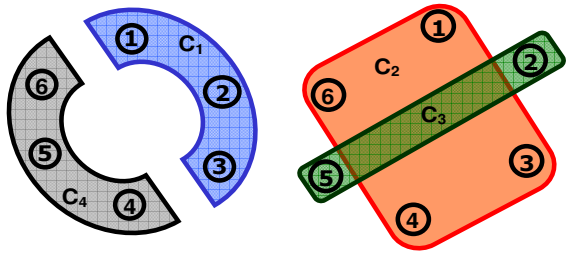

Fig. 4. Two minimum coverings of the H_1 hypergraph
Rys. 4. Dwa pokrycia minimalne hipergrafu H_1

At the next stage the total cost of each minimum transversal is calculated. For the presented example, the total weight of the τ_1 transversal is equal to $W_1 = L_1 + L_4 = 2+2 = 4$, while the total cost of the τ_2 transversal is calculated as $W_2 = L_2 + L_3 = 3+2 = 5$. Therefore, for the further analysis the τ_1 transversal is selected, because its total weight is the lowest. Next, all compability classes that belong to the τ_1 transversal are encoded. There are $W_1=4$ bits required, where $Q=\{q_1, \ldots, q_4\}$. In the presented example classes C_1 and C_4 are encoded with natural binary code (consecutive values are encoded in NB-code, the "00" code is reserved for the non-operational microinstruction). The encoding style is illustrated in the Tab. 2.

Tab. 2. The initial content of the control memory
Tab. 2. Pierwotna zawartość pamięci sterownika

Class C_1			Code		Class C_4			Code		K
y_1	y_2	y_3	Q_1	q_2	y_4	y_5	y_6	q_3	q_4	
0	1	0	0	1	0	0	1	0	1	1
0	1	0	0	1	1	0	0	1	0	2
1	0	0	1	0	0	1	0	1	1	3
0	0	1	1	1	0	1	0	1	1	4

Finally, new content of the control memory is formed. New microinstructions are determined as a concatenation of codes achieved in the previous step (each new microinstruction is formed as a concatenation of Q codes of the encoded compability classes). The new content of the control memory is shown in the Tab. 3.

Tab. 3. The content of the control memory after encoding
Tab. 3. Zawartość pamięci sterownika po zakodowaniu

Micro-instruction	Microoperation			
	q_1	q_2	q_3	q_4
μ_1	0	1	0	1
μ_2	0	1	1	0
μ_3	1	0	1	1
μ_4	1	1	1	1

The width of the new (encoded) microinstruction is equal to W_I=4. The total volume of the reduced memory is equal to V^*= 4*4 = 16. It means that the initial volume of the control memory was reduced by t = 33%.

6. Results of investigations

The proposed method of the microinstruction length reduction was compared with the traditional solution, based on the graph theory. To achieve the most comparable results between graphs and hypergraphs, in both cases exact covering algorithms presented in [1] and [12] were used. There were totally 1000 random bench-marks (contents of control memory) prepared. For each memory, the microinstruction length reduction based on the graph and hypergraph theory was performed. The initial benchmarks were divided into groups depending on the numbers of microoperations and microinstructions. The average results achieved during investigations are shown in the Tab. 4. The table includes the following values: the number of microoperations, average number of microinstructions, average time achieved during execution of the algorithm based on the hypergraph and graph theories.

Tab. 4. The results of investigations
Tab. 4. Wyniki badań

No. of micro-operations	No. of micro-instructions	Average time [s]	
		hypergraph	graph
10	**55**	**0,009**	**0,003**
20	**55**	**0,011**	**0,020**
30	**55**	**0,019**	**0,044**
40	**55**	**0,017**	**0,092**
50	**55**	**0,022**	**0,159**
60	**55**	**0,025**	**0,256**
70	**55**	**0,030**	**0,386**
80	**55**	**0,033**	**0,540**
90	**55**	**0,039**	**0,736**
100	**90**	**0,060**	**1,111**
120	**150**	**0,128**	**2,219**
140	**150**	**0,141**	**3,555**
160	**150**	**0,148**	**5,094**
180	**150**	**0,164**	**7,115**
200	**150**	**0,193**	**9,828**

The performed investigations showed that application of algorithms based on hypergraphs highly speeds-up the process of the microinstruction length reduction in comparison with traditional graphs. Only in case of small memories, where the number of microoperations does not exceed 10, the application of graphs benefited better results. However in all other cases, methods based on the hypergraph theory were much faster. It should be pointed out that increasing the number of microoperations effects in better results of hypergraphs algorithms. In case of memories that contain 200 microoperations and 150 microinstructions, hypergraphs were on average over 50 times faster than adequate graphs.

7. Summary

In the paper a new method of the microinstruction length reduction in the designing process of Microprogrammed Controllers was proposed. The algorithm is based on the representation of the compability classes with hypergraphs. The minimum covering is achieved via computation of the minimum transversal of the hypergraph. The performed investigations proved the effectiveness of the proposed method. The process of the microinstruction length reduction is much faster (even more than 50 times) in comparison with solutions based on the traditional graph theory.

8. References

[1] G. De Micheli: Synthesis and Optimization of Digital Circuits. McGrawHill, 1994.
[2] A. Clements: The principles of computer hardware. Oxford University Press, New Jersey, 2000.
[3] T. Łuba: Synteza układów cyfrowych. Praca zbiorowa pod redakcją prof. Tadeusza Łuby, WKŁ, Warszawa, 2003.
[4] M. Bolton: Digital Systems Design with Programmable Logic. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
[5] D. Burskey: Embedded Logic and Memory Find Home in FPGA. Electronic Design, No. 14, pp. 43-56 1999.
[6] D. Gajski: Principles of Digital Design, Prentice Hall, New Jersy1997.
[7] R. Puri, J. Gu: Microword Length Minimization in Microprogrammed Controller Synthesis. IEEE Trans. on Computer-Aided Design, 1993.
[8] E. L. Robertson: Microcode bit optimization is NP-complete. IEEE Trans. Comput., vol. C-28, pp. 316–319, Apr. 1979.
[9] S. Dasgupta: The Organization of Microprogram Stores. Computing Surveys, 1979.
[10] S. K. Hong, I. C. Park, C. M. Kyung: An O(n^3logn) Heuristic for Microcode Bit Optimization. In ICCAD-90, pp. 180-183, 1990.
[11] J. Lam, J. M. Delosme. Simulated Annealing: A Fast Heuristic for Partitioning VLSI Networks. In ICCAD-88, pp. 510-513, 1988.
[12] A. V. Aho, J. E. Hopcroft, J. D. Ullman: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, 1974.
[13] T. Eiter, G. Gottlob: Identifying the Minimal Transversals of a Hypergraph and Related Problems. SIAM Journal on Computing Volume 24, Issue 6 (December 1995) pp. 1278 - 1304 Year of Publication: 1995.
[14] C. Berge: Graphs and Hypergraph. North-Hols.r Mathematical Library, Amsterdam 1976.
[15] T. Eiter, G. Gottlob: Hypergraph transversal computation and related problems in logic and AI. LNCS, pp. 549—564, Springer, 2002.
[16] H.H. Hoos, T. Stutzle: Local Search Algorithms: An Empirical Evaluation. Journal of Automated, pp. 421-481, 2000.
[17] M. V. Wilkes: The best way to design an automatic calculating machine. Manchester University inaugural conference, Manchester, England, 1951.
[18] M. Molski: Modułowe i mikroprogramowalne układy cyfrowe. WKŁ, Warszawa, 1986.
[19] P. Sapiecha, Algorytmy syntezy funkcji i relacji boolowskich w aspekcie metod reprezentacji i kompresji danych. PW, Wydział Elektroniki i Technik Informacyjnych, Warszawa, 1998.

Artykuł recenzowany