Alexander BARKALOV, Larysa TITARENKO, **Sławomir CHMIELEWSKI**
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI,
ul. licealna 9, 65-417 Zielona Góra

Synthesis of Finite State Machines with use of pseudoequivalent states

Prof. dr hab. inż. Alexander BARKALOV

Prof. Alexander A. Barkalov worked in Donetsk National Technical University (DNTU) from 1976 till 1996 as a tutor . He cooperated actively with Kiev Institute of Cybernetics (IC) named after Victor Glushkov. He got his degree of doctor of technical sciences (Informatics) in 1995 from IC. From 1996 till 2003 he worked as a professor of DNTU. From 2003 he has been working as a professor on Department of Electrotechnics, Informatics and Telecommunications of University of Zielona Gora.

e-mail: A.Barkalov@iie.uz.zgora.pl

Dr hab. inż. Larysa TITARENKO

Prof. Larysa Titarenko got her degree of doctor of technical sciences (Telecommunications) in 2005 from Kharkov National University of Radioelectronics (KNURE). Till September, 2003 she worked as a professor of KNURE. From 2005 she has been working as a professor on Department of Electrotechnics, Informatics and Telecommunications of University of Zielona Gora.

e-mail: L.Titarenko@iie.uz.zgora.pl

Mgr inż. Sławomir CHMIELEWSKI

Sławomir Chmielewski (M.Sc.) graduated from the Department of IT at the University of Zielona Gora. Research interest: decrease of use macrocells in CPLD.

e-mail: S.Chmielewski@weit.uz.zgora.pl

Abstract

A new two-stage method of FSMs synthesis for PAL-based CPLD is proposed. It is based on both wide fan-in of PAL cells and existence of the classes of pseudoequivalent states of Moore FSM. The first step aims at decreasing the number of PAL cells used for implementing the input memory functions. The purpose of the second step is decrease in the number of PAL cells in the block of microoperations. An example of application of the proposed method as well as the results of experiments carried out for standard benchmarks are given.

Keywords: Moore FSM, Logic Synthesis, State assignment, CPLD.

Synteza skończonych automatów stanów z wykorzystaniem pseudorównoważnych stanów

Streszczenie

W artykule przedstawiono metody syntezy mikroprogramowalnego układu sterującego z użyciem wbudowanych bloków pamięci. Postęp w technologii półprzewodnikowej powoduje pojawienie się coraz to bardziej złożonych układów cyfrowych VLSI, takich jak złożone programowalne układy cyfrowe CPLD, gdzie funkcje logiczne są implementowane przy użyciu programowalnych bloków logicznych PAL. Obecnie jedną z istotnych kwestii w przypadku implementowania automatów FSM przy zastosowaniu układów CPLD jest zmniejszenie liczby zużycia makrokomórek PAL. Proponowane metody są ukierunkowane na zmniejszenie rozmiaru układu sterującego poprzez zastosowanie transformacji kodów klas pseudorównoważnych w pamięci. Podejście takie pozwala uzyskać uproszczoną formę funkcji przejścia części adresowej układu, dzięki któremu możliwa jest redukcja zasobów sprzętowych potrzebnych do implementacji jednostki sterującej w układach programowalnych typu CPLD bez zmniejszenia wydajności systemu cyfrowego. W artykule zamieszono wprowadzenie teoretyczne, przykład oraz wyniki badań uzyskanych podczas syntezy testowych sieci działań.

Słowa kluczowe: automat Moore'a, Synteza logiczna, CPLD.

1. Introduction

The overwhelming majority of digital systems include a control unit responsible for interplay of all blocks of other systems [1]. In many practical cases, the model of Moore finite state machine (FSM) is used for representing a control unit [2]. Nowadays, two families of programmable logic devices are used mostly for implementing logic circuits of FSMs: the field programmable gate arrays (FPGA) and the complex programmable logic devices (CPLD) [3]. As a rule, FPGAs are used for implementing rather complex FSMs, whereas CPLDs target fast FSMs [4, 5].

The majority of CPLD are based on PAL cells connected with programmable flip-flops [6, 7]. Each cell can be viewed as q s-input AND gates connected with an OR gate. The general model of the PAL–based cell is shown, for example, in [5]. The number q determines the upper limit for product terms to be implemented by a cell. This value is rather small, and $q \le 6$ in the most common cases [6,7]. Obviously, Boolean functions having more than q terms in their sum-of-products (SOP) forms need more than one cell for implementation. In the case of functions with more than q terms, the decomposition methods should be applied [8, 9, 10]. It leads to increasing the number of circuit layers and, therefore, for the total delay of the resulting circuit performance. To avoid this negative effect, it is important to decrease the numbers of terms in SOP of Boolean functions representing an FSM circuit. The next specific of PAL cells is their wide fan-in (the number of cell inputs $S > 20$) [6, 7].

The state assignment is one of the most important steps in FSM design [11]. A proper state assignment can decrease the number of PAL cells in the FSM logic circuit [8]. On the other hand, it can decrease the power consumption. The decreasing of the circuit area leads to decreasing its power consumption. So, it is very important to diminish the number of PAL-cells in the FSM logic circuits. We propose a design method based on using embedded memory blocks (EMB) for decreasing the number of PAL cells. But nowadays, only Delta39K family by Cypress includes EMBs. In this paper we propose a new approach targeting homogenous CPLD chips including only PAL macrocells. The method is based on existence of pseudoequivalent states (PS) of Moore FSM [11]. To represent a control algorithm, the language of graph-schemes of algorithms (GSA) is used [1].

2. Theoretical background

Let the control algorithm of a digital system be specified by a GSA $\Gamma = (B, E)$, where $B = \{b_0, b_E\} \cup E_1 \cup E_2$ is a set of the vertices and E is a set of edges. Here b_0 is an initial vertex, b_E is a final vertex, E_1 is a set of operational vertices, and E_2 is a set of conditional vertices. A vertex $b_q \in E_1$ contains a collection of microoperations $Y(b_q) \subseteq Y$, where $Y = \{y_1,...,y_N\}$ is a set of microoperations of a digital system's data-path [11]. A vertex $b_q \in E_2$ contains some logical condition $x_e \in X$, where $X = \{x_1,...,x_L\}$ is a set of logic conditions. The initial and final

vertices of graph-scheme of algorithm correspond to the initial state $a_1 \in A$, where $A = \{a_1,...,a_M\}$ is a set of internal states of a Moore FSM. Each operational vertex $b_q \in E_1$ corresponds to unique state $a_m \in A$. The logic circuit of the Moore FSM U_1 is represented by the following systems of Boolean functions:

$$\Phi = \Phi(T, X), \quad (1)$$

$$Y = Y(T), \quad (2)$$

where $T = \{T_1,...,T_R\}$ is a set of internal variables encoding the states $a_m \in A$, $\Phi = \{D_1,...,D_R\}$ is a set of the FSM input memory functions. The value of R is determined as

$$R = \lceil \log_2 M \rceil, \quad (3)$$

where $\lceil a \rceil$ is a minimum integer not less than a. The systems (1) and (2) are formed on the basis of a structure table with columns [1]: a_m is a current FSM state, $K(a_s)$ is a code of the state a_m, a_s is a next state (state of transition), $K(a_s)$ is a code of the state a_s, X_h is a conjunction of some elements of the set X determining the transition $< a_m, a_s >$, Φ_h is a collection of input memory functions that are equal to 1 to switch the memory from $K(a_m)$ into $K(a_s)$; $h = \overline{1, H_1(\Gamma)}$ is the row number. The column a_m contains a collection of microoperations $Y(a_m) \subseteq Y$, that are generated in the state $a_m \in A$. It is clear that $Y(b_q) = Y(a_m)$, where the vertex $b_q \in E_1$ is marked by the internal state $a_m \in A$. The structure diagram of Moore FSM U_1 is shown in Fig. 1.

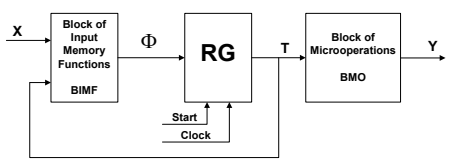

Fig. 1. Structure diagram of Moore FSM U_1
Rys. 1. Schemat blokowy automatu Moore'a FSM U_1

In the FSM U_1, a block of input memory functions (BIMF) implements the system (1), whereas a block of microoperations (BMO) generates microoperations (2). The codes $K(a_m)$ are kept into a register (RG). If a PAL cell of CPLD chip forms a function $D_r \in \Phi$, then the cell output is registered. It means that flip-flops of RG are distributed among the PAL cells of BIMF. A pulse *Start* loads the code of initial state $a_1 \in A$ into RG. A pulse *Clock* is used to change the content of RG. All blocks of U_1 are implemented using PAL cells.

Let us consider a GSA Γ_1 (Fig. 2a) marked by the states of Moore FSM using the rules [1].

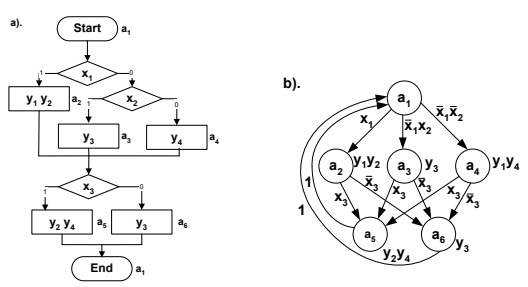

Fig. 2. Initial GSA Γ_1 (a) and corresponding state transition graph (b) of Moore FSM
Rys. 2. Sieć działań Γ_1 (a) graf (b) automatu Moore'a

The Moore FSM $U_1(\Gamma_1)$ has the following characteristics: $A = \{a_1,...,a_6\}$, $M = 6$, $X = \{x_1, x_2, x_3\}$, $L = 3$, $Y = \{y_1,...,y_4\}$, and $N = 4$. The FSM $U_1(\Gamma_1)$ can be represented by a state transition graph (STG) shown in Fig. 2b. This STG has $U_1(\Gamma_1) = 11$ edges. Each edge corresponds to one row of the structure table [1]. In turn, each row of the structure table determines up to R product terms in the system (1). So, it is important to diminish the number of rows, $H_1(\Gamma)$.

Let a symbol $U_0(\Gamma)$ stand for a Mealy FSM implementing a control algorithm represented by a GSA Γ. As a rule, the number of transitions $H_1(\Gamma)$ exceeds the number of transitions $H_0(\Gamma)$ of the equivalent Mealy FSM. It leads to increasing the number of PAL cells in the logic circuit of BIMF for Moore FSM $U_1(\Gamma_j)$ in comparison with this parameter for the equivalent Mealy FSM $U_0(\Gamma_j)$. The value $H_1(\Gamma_j)$ can be decreased due to using the pseudoequivalent states of Moore FSM [3, 11].

The states $a_m, a_s \in A$ are pseudoequivalent states if identical inputs result in identical next states for both $a_m, a_s \in A$. This is possible if the outputs of the operational vertices marked by these states are connected with the input of the same vertex of the GSA Γ [3]. Let $\Pi_A = \{B_1,...,B_I\}$ be a partition of the set A by the classes of PSs ($I \le M$).

In the case of FSM $U_1(\Gamma_1)$, the partition $\Pi_A = \{B_1, B_2, B_3\}$ can be found, where $B_1 = \{a_1\}$, $B_2 = \{a_2, a_3, a_4\}$, and $B_3 = \{a_5, a_6\}$. Let us replace the states $a_m \in B_i$ by blocks $B_i \in \Pi_A$ having $M_i = |B_i|$ inputs. It allows transforming an initial GSA Γ into a block GSA $B(\Gamma)$. The block GSA $B(\Gamma_1)$ is shown in Fig. 3a. Obviously, the initial STG of Moore FSM can be transformed, too. To transform an STG, the vertices corresponding to $a_m \in B_i$ should be replaced by a single vertex $B_i \in \Pi_A$. Let us name the resulting STG as a block transition graph (BTG) (Fig. 3b). This graph determines a reduced structure table (RST) of Moore FSM.

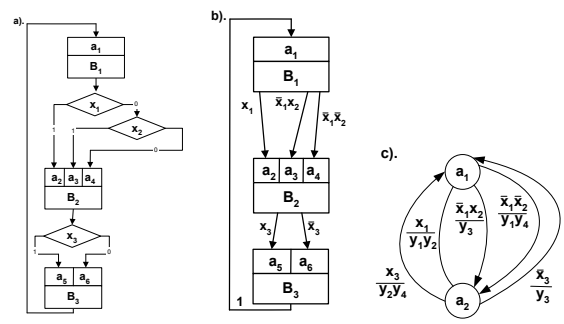

Fig. 3. Block GSA $B(\Gamma_1)$ (a), block transition graph of Moore FSM $U_1(\Gamma_1)$ (b) and state transition graph of Mealy FSM $U_0(\Gamma_1)$ (c)
Rys. 3. Blok GSA $B(\Gamma_1)$ (a) blok połączeń automatu Moore'a (b) graf automatu Mealy'ego (c)

Let us point out that there are only two states in the Mealy FSM $U_0(\Gamma_1)$ (Fig. 3c). The comparison of graphs Fig. 3b and Fig. 3c allows finding the following relations:

$$\left.\begin{aligned} I &= M_0 + 1, \\ H_2 &= H_0 + 1. \end{aligned}\right\} \quad (4)$$

In (4), M_0 is the number of states for Mealy FSM $U_0(\Gamma_1)$, whereas H_2 is the number of edges for BTG $B(\Gamma)$.

Therefore, the replacement of internal states of Moore FSM by the classes of PS leads to diminishing the number of product terms

in the system (1) up to $H_2 \approx H_0$. It can result in decreasing the number of PAL cells in the logic circuit of BIMF.

Two methods can be used for representing the classes $B_i \in \Pi_A$ [6]: optimal state assignment, transformation of the codes of states into the codes of classes of pseudoequivalent states.

In the first case, the states $a_m \in A$ are encoded in such a way that the codes of the states $a_m \in B_i$ $(i = 1,...,I)$ belong to a single generalized interval of the R-dimensional Boolean space. This leads to a Moore FSM U_2 that has the same structure as the Moore FSM U_1. The algorithm from [11] can be used for such a state assignment. In the case of $U_2(\Gamma_1)$, there are $R = 3$ and $T = \{T_1, T_2, T_3\}$. One of the possible variants of state assignment (optimal encoding) is shown in Fig. 4.

As follows from Fig. 4, the class B_1 is determined by the interval *00, the class B_2 by **1, and the class B_3 by *10. These intervals are treated as the class codes.

T_1 \ T_2T_3	00	01	11	10
0	a_1	a_2	a_3	a_5
1	*	a_4	*	a_6

Fig. 4. Optimal state codes for Moore FSM $U_2(\Gamma_1)$
Rys. 4. Kodowanie stanów automatu Moor'a $U_2(\Gamma_1)$

But such an encoding is not always possible [10]. In the second case, the classes $B_i \in \Pi_A$ are encoded by the binary codes $K(B_i)$ with $R_1 = \lceil \log_2 I \rceil$ bits. The variables $\tau_r \in \tau$ are used for such an encoding, where $|\tau| = R_1$. This approach leads to a Moore FSM U_3 (Fig. 5) including a block of code transformer (BCT).

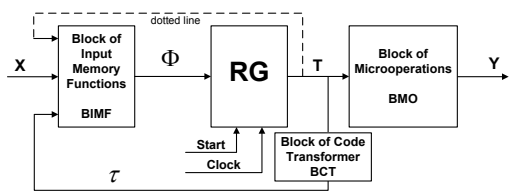

Fig. 5. Structure diagram of Moore FSM U_3 without dotted line, U_4 with dotted line
Rys. 5. Schemat blokowy automatu U_3 bez linii kreskowanej, U4 z linią kreskowaną

In the FSM U_3, the BIMF generates the functions

$$\Phi = \Phi(\tau, X) \tag{5}$$

and the block of code transformer implements the functions

$$\tau = \tau(T). \tag{6}$$

The number of transitions of Moore finite-state-machine U_3 is equal to $H_0(\Gamma)$. The drawback of FSM U_3 is the existence of BCT that consumes additional resources of a chip (in comparison with U_1).

In this paper we propose to combine the application of optimal state assignment and transformation of the codes of the states. In this case the block of code transformer can be even eliminated if some condition holds. The proposed method is based on the wide fan-in of PAL cells.

3. Main idea of proposed method

Let a partition Π_A be found for some Moore FSM. Let us encode the states $a_m \in A$ in the optimal way. Let $I(B_i)$ be the number of generalized intervals representing the class $B_i \in \Pi_A$. Let us represent the set Π_A as $\Pi_B \cup \Pi_C$, where

$$\left.\begin{aligned} I(B_i) = 1 \rightarrow B_i \in \Pi_B, \\ I(B_i) > 1 \rightarrow B_i \in \Pi_C. \end{aligned}\right\} \tag{7}$$

Obviously, only classes $B_i \in \Pi_C$ need transformation based on BCT. Let $L(D_r)$ be the number of logical conditions used as literals in the SOP of the function $D_r \in \Phi$. Let the following condition take place:

$$L(D_r) + R + R_C \le S(\overline{1,R}). \tag{8}$$

In (8), the symbol S stands for the fan-in of PAL cells, whereas R_C determines the number of bits required for encoding classes $B_i \in \Pi_C$:

$$R_C = \lceil \log_2(|\Pi_C| + 1) \rceil. \tag{9}$$

In (9), the value of $|\Pi_C|$ is incremented to take into account the relation $B_i \notin \Pi_C$. Obviously, $B_i \notin \Pi_C$ means that $B_i \in \Pi_B$.

If the condition (8) takes place, then we propose to use two sources of codes $K(B_i)$. If $B_i \in \Pi_B$, then the code $K(B_i)$ is represented by the register RG. If $B_i \in \Pi_C$, then the code $K(B_i)$ is represented by the block BCT. It leads to the Moore FSM U_4 whose structure diagram is shown in Fig. 5 (with dotted line).

In the case of U_4, the block BIMF implements functions

$$\Phi = \Phi(T, \tau, X). \tag{10}$$

The block BCT generates functions (6), but, in contrast to the FSM U_3, the set τ includes only R_c elements. As in all other cases, the block BMO implements functions (2).

If $\Pi_C = \emptyset$, then the FSM U_4 does not use BCT and it is the same as U_2. If $\Pi_B = \emptyset$, then all states $a_m \in A$ should be transformed and the FSM U_4 is the same as U_3.

The main goal of the optimal state assignment is to find the state codes leading to $\Pi_C = \emptyset$. If there are more than one variant of such an assignment, then it should be found a version leading to optimization of the block BMO. Let $Q(y_n)$ be the number of terms in the minimized SOP of a function $y_n \in Y$. To optimize the block BMO, it is necessary to find the variant of state assignment leading to satisfying the condition

$$Q(y_n) \le q \quad (n = \overline{1,N}). \tag{11}$$

In this case, any function $y_n \in Y$ is implemented using one PAL cell. It means that only N cells are used to implement the logic circuit of BMO.

If there is $\Pi_C \neq \emptyset$ after execution of the state assignment, then the resoling state codes should be changed to minimize the numbers of cells in both BMO and BCT. Let $Q(\tau_r)$ be the number of terms in the SOP of the function $\tau_r \in \tau$. If $\Pi_C \neq \emptyset$, then it is necessary to find the codes $K(a_m)$ leading to satisfying both condition (11) and the following condition

$$Q(\tau_r) \le q \quad (r = \overline{1,R_C}). \tag{12}$$

If condition (12) takes place, then only R_C PAL cells are used to implement the block BCT.

So, we propose the two-stage procedure for state assignment in the case of FSM U_4. The first stage is the optimal state assignment. The second state, the refined state assignment, is

executed in such a manner that the following condition takes place:

$$\Delta n = n(BCT) + n(BMO) - (N + R_C) \to \min . \quad (13)$$

In (13), symbols $n(BCT)$ and $n(BMO)$ stand for the numbers of PAL cells in logic circuits of blocks BCT and BMO, respectively. Let us determine the formulae for finding $n(BCT)$ and $n(BMO)$. It is shown in many papers, for example in [5], that the number of PAL cells required for implementing f is determined as

$$n(f) = \left\lceil \frac{Q(f) - q}{q - 1} \right\rceil + 1 . \quad (14)$$

To construct the RST, a system of generalized formulae of transitions [10] should be constructed. This system can be derived from the block GSA. In the case of block GSA $B(\Gamma_1)$, this system is the following one:

$$B_1 \to x_1 a_2 \vee \bar{x}_1 x_2 a_3 \vee \bar{x}_1 x_2 a_4 ;$$
$$B_2 \to x_3 a_5 \vee \bar{x}_3 a_6 ; \; B_3 \to a_1 . \quad (15)$$

Obviously, such a system can be obtained using an initial GSA Γ and the partition Π_A.

Because an RST does not contain a column with microoperations, the table of microoperations (TMO) should be constructed. The symbol $Y(a_m)$ stands for the collection of microoperations generated in the state $a_m \in A$. In the case of FSM $U_4(\Gamma_1)$ this table has $M = 6$ lines (Table 1).

Tab. 1. Table of microoperations of FSM $U_4(\Gamma_1)$
Tab. 1. Tabela mikrooperacji automatu FSM $U_4(\Gamma_1)$

a_m	$K(a_m)$	$Y(a_m)$	m	a_m	$K(a_m)$	$Y(a_m)$	m
a_1	000	-	1	a_4	101	y_4	4
a_2	001	y_1y_2	2	a_5	010	y_2y_4	5
a_3	011	y_3	3	a_6	110	y_3	6

In Table 1, the states codes from Fig. 4 are used.

A function $y_n \in Y$ is represented as

$$y_n = \bigvee_{m=1}^{M} C_{nm} A_m \quad (n = \overline{1, N}) . \quad (16)$$

In (16), the Boolean variable C_{nm} is equal to 1 only if $y_n \in Y(a_m)$, the symbol A_m stands for the conjunction of internal variables $T_r \in T$ corresponding to the code $K(a_m)$. For example, the following expression $y_2 = A_2 \vee A_5$ can be derived from Table 1. Analysis of Fig. 4 shows that $y_2 = \bar{T}_1 \bar{T}_2 T_3 \vee \bar{T}_1 T_2 \bar{T}_3$ and, therefore, $Q(y_2) = 2$. If the places for states a_2 and a_3 are interchanged, then $y_2 = \bar{T}_1 T_2 T_3 \vee \bar{T}_1 T_2 \bar{T}_3 = \bar{T}_1 T_2$ and now there is $Q(y_2) = 1$. So, the stage of refined state assignment has a sense.

To construct the system (6), a table of code transformation (TCT) should be formed. The column τ_m includes the functions $\tau_r \in \tau$ corresponding to ones in the code of class $B_i \in \Pi_A$ from the m-th row of the table.

Let us encode the classes $B_i \in \Pi_A$ for FSM $U_3(\Gamma_1)$ in the following manner: $K(B_1) = 00$, $K(B_2) = 01$, $K(B_3) = 10$. In this case, the TCT has $M = 6$ rows (Table 2).

Tab. 2. Table of code transformation of FSM $U_3(\Gamma_1)$
Tab. 2. Kodowanie przejść automatu FSM $U_3(\Gamma_1)$

a_m	$K(a_m)$	B_i	$K(B_i)$	Γ_1	m
a_1	000	B_1	00	-	1
a_2	001	B_2	01	τ_2	2
a_3	011	B_2	01	τ_2	3
a_4	101	B_2	01	τ_2	4
a_5	010	B_3	10	τ_1	5
a_6	110	B_3	10	τ_1	6

A function $\tau_r \in \tau$ is represented as

$$\tau_r = \bigvee_{m=1}^{M} C_{rm} A_m \quad (r = \overline{1, R_C}) . \quad (17)$$

In (23), the Boolean variable C_{rm} is equal to 1 only if $l_{mir} = 1$, where l_{mir} is a value of the r-th bit of the code $K(B_i)$ from the m-th row of TCT. For example, the following equation can be derived from Table 2: $\tau_1 = A_5 \vee A_6 = T_2 \bar{T}_3$

As in the previous case, the refined state assignment should be executed to minimize the equations (17). It should be done for functions $\tau_r \in \tau$, such that $Q(\tau_r) > q$.

4. Experimental results

To prove the usefulness of the proposed strategy of FSM design, a lot of practical experiments were carried out. In experiments, the standard benchmarks [13] represented in the KISS2 format were used. Basing on [5], we conducted our investigations using 36 benchmarks. We used JEDI for executing the optimal state assignment. To do it, we took the input dominant version of JEDI. The obtained results are shown in Table 3. In this table, the numbers of PAL cells with $q = 5$ are shown for logic circuits of FSMs represented by the benchmarks. The columns U_2-U_4 contain the design outcomes for our methods discussed in this paper. To obtain them, we used the input dominant version of JEDI algorithm. The last column includes results, from [5].

Let us point out that the paper [5] was chosen for comparison, because we think that these results are the best. Among all examined 36 benchmarks, the proposed method U_4 produced 21 solutions (58,3%), which required less amount of logic cells than it is required for U_1 (JEDI). Let us point out that these 58,3% include all complex examples (with $n(U_1) \geq 13$). There are no cases when $n(U_2) < n(U_2)$. So, only applications of U_2 and U_4 have sense. There are 20 solutions (55,5%) which required less amount of logic cells than it is required for FSMs based on [5]. In 4 cases (11,1%) the logic circuits required the same amount of cells for both U_4 and [5]. So, the proposed approach produced solutions which did not require greater number of logic cells for 66,6% of discussed benchmarks in comparison with [5].

Tab. 3. Some results of experiments
Tab. 3. Przykładowe wyniki eksperymentu

Bench-k	**bbara**	**bbsse**	**bbtas**	**dk15**	**mark1**	**opus**	**mc**	**s208**	**s420**	**sand**	**train11**	**train4**
U_2	*9*	*19*	*6*	*8*	*20*	*15*	*7*	*13*	*14*	*62*	*5*	*3*
U_3	*14*	*22*	*8*	*9*	*22*	*17*	*9*	*12*	*12*	*58*	*7*	*4*
U_4	*13*	*18*	*7*	*9*	*16*	*14*	*8*	*10*	*10*	*52*	*6*	*3*
[5]	*12*	*17*	*5*	*9*	*18*	*12*	*7*	*11*	*12*	*64*	*8*	*3*

5. Conclusion

In this paper the design method targeting the Moore FSM logic circuit and PAL-based CPLD chips is proposed. The proposed method is based on the wide fan-in of PAL cells, as well as on the existence of pseudoequivalent states of Moore FSM. The main idea of the proposed method is using two sources of codes of the classes of pseudoequivalent states.

Our strategy is oriented towards the area optimization. The positive back effect of the proposed method is possible decrease in the number of layers of PAL cells in the FSM logic circuits. But we did not investigate this property of the proposed method. The method includes two stages. First of all, the optimal state assignment is executed based of JEDI approach. Next, the codes are rearranged to optimize the number of cells in the circuit implementing the system of microoperations. We called the second stage as "the refined state assignment".

The paper contains a lot of experimental results based on investigations carried out for commonly used benchmarks. These results show that the proposed approach is especially advantageous for designing rather complex FSMs.

6. Literatura

[1] Anderson J., Brown S.: Technology mapping for large complex PLDs, In Proceedings of Design Automation Conference, 1998, pp. 698-703.
[2] Baranov S.: Logic Synthesis for Control Automata, Boston, Kluwer Academic Publishers, 1994, pp. 405.
[3] Barkalov A., Titarenko L.: Logic Synthesis for FSM-Based Control Units, Berlin, Springer, 2009, pp. 234.
[4] Kania D.: The logic synthesis for the PAL-based complex programmable logic devices, Gliwice, Silesian University of Technology 2004, pp. 212. (in Polish).
[5] Kania D., Czerwinski R.: Area and speed oriented synthesis of FSMs for PAL-based CPLDs, Microprocessors and Microsystems, 2012, pp. 45-61, V.36, N:1.
[6] Czerwinski R., Kania D.: Synthesis of finite state machines for CPLDs, International Journal of Applied Mathematics and Computer Science, 2009, pp. 647-659,V. 19, N:4.
[7] Kania D., Milek A.: Logic Synthesis Based on Decomposition for CPLDs, Microprocessors and Microsystems, 2010, pp. 25-38, V.34, N:1.
[8] Barkalov A., Titarenko L., Chmielewski S.: Reduction in the number of PAL macrocells in the circuit of a Moore FSM, International Journal of Applied Mathematics and Computer Science, 2007, pp. 565-575, V.17, N:4.
[9] www.altera.com.
[10] www.xilinx.com.
[11] Barkalov A.: Principles of logic optimization for Moore microprogrammed automaton, Cybernetics and Systems Analysis, 1998, pp. 54-63, V. 34, N:1.
[12] Cypress Programmable Logic: Delta 39K. Data Sheet, http://cypress.com/pld/delta39k.html.
[13] Yang S.: Logic Synthesis and Optimization Benchmarks User Guide, Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, Microelectronics Center of North Carolina, 1991.

otrzymano / received: 21.08.2013
przyjęto do druku / accepted: 01.10.2013 *artykuł recenzowany / revised paper*

INFORMACJE

Informacje dla Autorów

Redakcja przyjmuje do publikacji tylko prace oryginalne, nie publikowane wcześniej w innych czasopismach. Redakcja nie zwraca materiałów nie zamówionych oraz zastrzega sobie prawo redagowania i skracania tekstów oraz streszczeń.

Artykuły naukowe publikowane w czasopiśmie PAK są formatowane jednolicie zgodnie z ustaloną formatką zamieszczoną na stronie redakcyjnej www.pak.info.pl. Dlatego artykuły przekazywane redakcji należy przygotowywać w edytorze Microsoft Word 2003 (w formacie DOC) z zachowaniem:

- wielkości czcionek,
- odstępów między wierszami tekstu,
- odstępów przed i po rysunkach, wzorach i tabelach,
- oznaczeń we wzorach, tabelach i na rysunkach zgodnych z oznaczeniami w tekście,
- układu poszczególnych elementów na stronie.

Osobno należy przygotować w pliku w formacie DOC notki biograficzne autorów o objętości nie przekraczającej 450 znaków, zawierające podstawowe dane charakteryzujące działalność naukową, tytuły naukowe i zawodowe, miejsce pracy i zajmowane stanowiska, informacje o uprawianej dziedzinie, adres e-mail oraz aktualne zdjęcie autora o rozmiarze 3,8 x 2,7 cm zapisane w skali odcieni szarości lub dołączone w osobnym pliku (w formacie TIF).

Wszystkie materiały:

- artykuł (w formacie DOC),
- notki biograficzne autorów (w formacie DOC),
- zdjęcia i rysunki (w formacie TIF lub CDR),

prosimy przesyłać w formie plików oraz dodatkowo jako wydruki na białym papierze (lub w formacie PDF) na adres e-mail: wydawnictwo@pak.info.pl lub pocztą zwykłą, na adres: Redakcja Czasopisma Pomiary Automatyka Kontrola, Asystent Redaktora Naczelnego mgr Agnieszka Skórkowska, ul. Akademicka 10, p.21A, 44-100 Gliwice.

Wszystkie artykuły naukowe są dopuszczane do publikacji w czasopiśmie PAK po otrzymaniu pozytywnej recenzji. Autorzy materiałów nadesłanych do publikacji są odpowiedzialni za przestrzeganie prawa autorskiego. Zarówno treść pracy, jak i wykorzystane w niej ilustracje oraz tabele powinny stanowić dorobek własny Autora lub muszą być opisane zgodnie z zasadami cytowania, z powołaniem się na źródło cytatu.

Przedrukowywanie materiałów lub ich fragmentów wymaga pisemnej zgody redakcji. Redakcja ma prawo do korzystania z utworu, rozporządzania nim i udostępniania dowolną techniką, w tym też elektroniczną oraz ma prawo do rozpowszechniania go dowolnymi kanałami dystrybucyjnymi.