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

Zmniejszenie zużycia makrokomórek PAL w realizacjach układowych automatów Moore'a

Prof. dr hab. inż. Alexander BARKALOV

Prof. Alexander A. Barkalov w latach 1976 - 1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywnie z Instytutem Cybernetyki im. V. M. Glushkova w Kijowie, gdzie uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. W latach 1996 - 2003 pracował jako profesor w Instytucie Informatyki Narodowej Politechniki Donieckiej. Od 2004 pracuje jako profesor na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

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

Dr hab. inż. Larysa TITARENKO

Dr hab. Larysa Titarenko w 2004 roku obroniła rozprawę habilitacyjną i uzyskała tytuł doktora habilitowanego ze specjalnością telekomunikacja. W latach 2004 - 2005 pracowała jako profesor w Narodowym Uniwersytecie Radioelektroniki w Charkowie. Od 2007 pracuje jako profesor na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

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

Streszczenie

W artykule została przedstawiona metoda zmniejszenia wymaganych zasobów sprzętowych do implementacji automatów Moore'a w matrycowym układzie programowalnym typu PAL. Cechą automatów Moore'a jest regularny charakter mikrooperacji, które daje się implementować z użyciem wbudowanych bloków pamięci. Metoda oparta jest na zastosowaniu transformacji kodów pseudorównoważnych stanów. Zaproponowane podejście pozwala zmniejszyć liczbę wymaganego zużycia sprzętowego bez zmniejszenia wydajności systemów cyfrowych. Przedstawiony zostanie również przykład zaproponowanego rozwiązania oraz wyniki eksperymentu.

Słowa kluczowe: automat Moore'a, jednostka sterująca, układ cyfrowy, układy programowalne.

Hardware reduction for Moore FSM implemented with CPLD

Abstract

The method of decrease in the number of programmable array logic (PAL) macrocells in logic circuit of Moore finite-state-machine (FSM) is proposed. This method is based on the use of free outputs of embedded memory blocks to represent the code of the class of pseudoequivalent states. The proposed approach allows minimizing the hardware without decreasing of the digital system performance. An example of application of the proposed method is given. Control unit of any digital system can be implemented as the Moore FSM. Recent achievements in semiconductor technology have resulted in development of such sophisticated VLSI chips as field-programmable logic arrays (FPGA) and complex programmable logic devices (CPLD). Very often CPLD are used to implement complex controllers. In CPLD, logic functions are implemented using programmable array logic macrocells. One of the issues of the day is decrease in the number of PAL macrocells required for implementation of FSM logic circuit. A proper state assignment can be used to solve this problem. The peculiarities of Moore FSM are existence of pseudoequivalent states and dependence of microoperations only on FSM internal states. The peculiarity of CPLD is a wide fan-in of PAL macrocell. It permits to use different sources for representation of a current state code.

Keywords: Moore finite-state-machine, control unit, logic circuit, programmable logic device.

Mgr inż. Sławomir CHMIELEWSKI

Mgr inż. Sławomir Chmielewski absolwent Uniwersytetu Zielonogórskiego. Ukończył studia informatyczne o specjalizacji inżynieria komputerowa. Uczestnik trzeciego roku studiów doktoranckich.

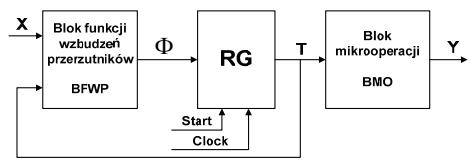
e-mail: S.Chmielewski@weit.uz.zgora.pl

1. Wprowadzenie

Jednostka sterująca (ang. Control Unit, CU) we wszystkich systemach cyfrowych może zostać zaprojektowana jako skończony automat stanów (ang. Finite State Machine, FSM) [1, 2]. Aktualnie programowalne układy cyfrowe są bardzo często stosowane do realizacji układu logicznego automatu FSM [3, 4]. Postęp w technologii półprzewodnikowej powoduje pojawienie się coraz to bardziej złożonych układów cyfrowych (ang. Very Large Scale Integration, VLSI), takich jak złożone programowalne układy cyfrowe (ang. Complex Programmable Logic Devices, CPLD), gdzie funkcje logiczne są implementowane przy użyciu programowalnych bloków logicznych (ang. Programmable Array Logic, PAL) [5, 6]. Obecnie jedną z istotnych kwestii w przypadku implementowania automatów FSM przy zastosowaniu układów CPLD jest zmniejszenie liczby zużycia makrokomórek PAL [7, 8]. Efektywne metody bazują na minimalizacji symbolicznej, algorytmach genetycznych i innych metodach heurystycznych [9]. W artykule proponowana jest metoda zmniejszania liczby makrokomórek PAL, gdzie przydzielenie stanów jest ukierunkowane na optymalizację mikrooperacji w bloku FSM. W pracy przedstawiono wyniki eksperymentu, które przeprowadzone zostały na popularnych układach testowych [10].

2. Opis projektowania automatów Moore'a

Najpopularniejszym sposobem opisu automatu Moore'a jest struktura tablicy [1], w której a_m jest początkowym stanem automatu, $a_m \in A$, gdzie $A=\{a_1,\ldots a_M\}$ jest zbiorem stanów automatu; $K(a_m)$ jest kodem stanu a_m zapisanym na $R = \lceil \log_2 M \rceil$ bitach; a_s jest następnym stanem automatu; $K(a_s)$ jest kodem stanu $a_s \in A$; X_h jest koniunkcją zmiennych ze zbioru wejściowych warunków cyfrowych $X=\{x_1,\ldots x_L\}$, wymuszając przejście $<a_m, a_s>$; Φ_h jest zbiorem wejściowych funkcji wzbudzeń pamięci, które są równe 1 w celu przełączenia pamięć automatu z kodu $K(a_m)$ na kod $K(a_s)$, gdzie $\Phi=\{D_1,\ldots D_R\}$; h jest numerem przejścia automatu (h=1,…, H). Stan a_m zawiera zbiór mikrooperacji $Y(a_m) \subseteq Y$, gdzie $Y=\{y_1,\ldots, y_N\}$. Ustalona tablica automatu Moore'a U_1 pokazana jest na rys. 1.



Rys. 1. Struktura diagramu Moore'a FSM U_1
Fig. 1. Structural diagram of Moore FSM U_1

Na tej podstawie wyprowadzany jest blok funkcji wzbudzeń przerzutników (BFWP)

$$\Phi = \Phi(T, X) \tag{1}$$

i mikrooperacji (BMO)

$$Y=Y(T), \tag{2}$$

gdzie $T=\{T_1,\ldots, T_R\}$ jest zbiorem zmiennych wewnętrznych kodujących stany $a_m \in A$. Sygnał „Start" jest zastosowany do załadowania stanu początkowego $a_1 \in A$ do rejestru (ang. ReGister, RG), który służy do zapamiętania kodu $K(a_m)$ stanu $a_m \in A$. Sygnał „Clock" przełącza stan RG.

Zmniejszanie zasobów sprzętowych w układach cyfrowych przy projektowaniu automatów FSM U_1 może być zrealizowane przy użyciu jednej z proponowanych metod [2, 5]. W przypadku optymalizacji liczby użytych stanów [2], klasa pseudorównoważnych stanów jest reprezentowana przez R-wymiarową przestrzeń boolowską.

Stan a_s, gdzie $a_s \in A$ jest pseudorównoważnym stanem, jeżeli stan ten jest kolejnym stanem automatu dla wyjść poprzedzających go stanów. W takim przypadku mamy udoskonaloną metodę wykorzystania liczby stanów [5], każda mikrooperacja $y_n \in Y$ jest reprezentowana przez R-wymiarową przestrzeń boolowską. W obu przypadkach stany mogą być kodowane za pomocą znanych metod, takich jak ESPRESSO [6]. Oczywiście jest to niemożliwe do stosowania obu metod równocześnie. Oznacza to, że zasoby sprzętowe mogą być zmniejszone albo dla bloku BFWP, albo dla bloku BMO. W artykule proponujemy metodę pozwalającą na zmniejszenie zużycia sprzętowego dla obu bloków kombinacyjnych automatu Moore'a U_1.

3. Idea proponowanej metody

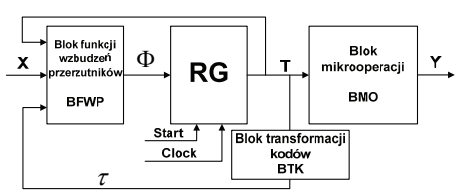
Niech $\Pi_A=\{B_1,\ldots B_I\}$ będzie zbiorem części A klasy pseudorównoważnych stanów. Niech $\Pi_A=\Pi_B \cup \Pi_C$, gdzie $B_i \in \Pi_B$, jeżeli $|B_i| \geq 2$, i w przeciwnym wypadku $B_i \in \Pi_C$. Następnie kodujemy $B_i \in \Pi_B$ binarnym kodem $K(B_i)$ używając

$$R_1 = \lceil \log_2 I_B \rceil \tag{3}$$

bitów $\tau_r \in \tau$, gdzie $I_B=|\Pi_B|$.

Specyficzną cechą układu typu PAL jest duża ilość wejść makrokomórek i ilość termów na mikrokomórkę [5, 8], dlatego też możliwe jest użycie więcej niż jednego źródła kodowania stanów dla bloku BFWP [11].

Blok BFWP jest optymalizowany dzięki istnieniu pseudorównoważnych stanów, przez co dany stan automatu przypisywany jest do kodu odpowiedniej klasy obiektu przy użyciu makrokomórek PAL. Jako obiekt rozumiany jest wewnętrzny stan automatu i zbiór mikrooperacji. To jest możliwe dla automatów Moore'a U_2 o strukturze pokazanej na rysunku 2.



Rys. 2. Struktura diagramu Moore'a U_2
Fig. 2. Structural diagram of Moore U_2

Dla skończonego automatu U_2, BFWP implementuje funkcje

$$\Phi = \Phi(T, \tau, X), \tag{4}$$

blok transformacji kodów (BTK) przekształca kody pseudorównoważnych stanów $a_m \in B_i$ w kod klasy $K(B_i)$. Blok BTK generuje funkcję

$$\tau=\tau(T). \tag{5}$$

Dzięki takiemu podejściu możliwe jest zmniejszenie liczby użytych makrokomórek w obu blokach BMO i BTK. Pozwala to również na to, aby była ona przedstawiona za pomocą sieci działań algorytmu GSA [1].

Metoda składa się z następujących etapów projektowania:

- Utworzenie tablicy przejścia automatu Moore'a (bez kodowania stanów).
- Przydzieleniu stanów do odpowiednich klas Π_A, Π_B, i Π_C.
- Kodowanie poszczególnych klas $B_i \in \Pi_B$.
- Minimalizacja stanów w funkcji (2) i (5).
- Utworzenie tabeli transformacji, gdzie stan obecny $a_m \in B_i$ jest zastąpiony przez odpowiednią klasę $B_i \in \Pi_A$, a kod stanu $K(a_m)$ jest zastąpiony przez odpowiedni kod $K(B_i)$.
- Utworzenie tabeli dla bloku BTK i funkcji (5).
- Utworzenie funkcji (2) i (4).
- Zaimplementowanie automatu Moore'a U_2 w układzie cyfrowym przy użyciu funkcji (2), (4) i (5) oraz zastosowaniu dedykowanego układ CPLD.

4. Przykład proponowanej metody

Rozważmy przykład automatu Moore'a S_1 o następującej charakterystyce: M=13, N=8, $\Pi_A=\{B_1,\ldots,B_7\}$, gdzie $B_1=\{a_1\}$, $B_2=\{a_2, a_3\}$, $B_3=\{a_4\}$, $B_4=\{a_5, a_6, a_7\}$, $B_5=\{a_8, a_9\}$, $B_6=\{a_{10}\}$ i $B_7=\{a_{11}, a_{12}, a_{13}\}$. Niech funkcja (2) będzie reprezentowana jako:

$$\begin{aligned} &y_1 = A_4 \vee A_8 \vee A_9 \vee A_{10}\,;\; y_2 = A_5 \vee A_{11}\,;\\ &y_3 = A_2 \vee A_3 \vee A_{12}\,;\; y_4 = A_6 \vee A_7 \vee A_8\,;\\ &y_5 = A_3 \vee A_7 \vee A_8 \vee A_{11}\,;\; y_6 = A_3 \vee A_7\,;\\ &y_7 = A_4 \vee A_6 \vee A_{10}\,;\; y_8 = A_3 \vee A_7 \vee A_{12}\,. \end{aligned} \tag{6}$$

W tym przypadku zmienne funkcji A_m odpowiadają koniunkcji pewnych zmiennych stanów $T_r \in T$ wyznaczonych przez kod stanu $K(a_m)$ (m=1,…, M). Niech przejście dla klasy obiektu $B_2, B_3 \in \Pi_A$ jest wyznaczone w następujący sposób:

$$\begin{aligned} &B_2 \rightarrow x_3 a_4 \vee \bar{x}_3 x_4 a_7 \vee \bar{x}_3 \bar{x}_4 a_6\,;\\ &B_3 \rightarrow a_9\,. \end{aligned} \tag{7}$$

Po czym, uzyskuje się R=4, $T=\{T_1,\ldots, T_4\}$, $\Phi=\{D_1,\ldots, D_4\}$. Używając algorytmu ESPRESSO [7] otrzymuje się optymalne kodowanie stanów dla automatu FSM S_1 (rys. 3).

T_1T_2 \ T_3T_4	00	01	11	10
00	a_1	a_2	*	a_4
01	a_5	*	a_6	a_{10}
11	a_{11}	a_3	a_7	a_8
10	a_{13}	a_{12}	*	a_9

Rys. 3. Kodowanie stanów automatu Moore'a S_1
Fig. 3. Optimal states codes of Moore S_1

Przekształcając (6) za pomocą kodowania stanów (rys. 3) otrzymuje się następujące funkcje:

$$\begin{aligned} &y_1 = T_3\bar{T}_4\,;\; y_2 = T_2\bar{T}_3\bar{T}_4\,;\; y_3 = \bar{T}_3 T_4\,;\\ &y_4 = T_3T_4 \vee T_1T_2T_3\,;\; y_5 = T_1T_2\,;\\ &y_6 = T_1T_2T_4\,;\; y_7 = \bar{T}_1T_3\,;\; y_8 = T_1T_4\,. \end{aligned} \tag{8}$$

W praktyce każda makrokomórka typu PAL ma nie mniej niż 8 wejść [3, 4]. Znaczy to, że każda funkcja z systemu (8) może być implementowana przy wykorzystaniu tylko jednej makrokomórki.

Analiza rysunku 3, która pokazuje, że kod $K(a_8)$ i $K(a_9)$ zapisać można za pomocą iloczynu uogólnionego, z tabeli Karnaugh'a. Oznacza to, że należy do klasy $B_5 \in \Pi_B$, ponieważ stany $a_8, a_9 \in B_5$. Po dokonaniu podziału wszystkich stanów, mamy następujące przypisanie do odpowiednich klas $\Pi_B=\{B_1,B_3,B_5,B_6\}$, $\Pi_C=\{B_2,B_4,B_7\}$. Stąd otrzymuje się I_C=3, R_1=2 oraz $\tau=\{\tau_1,\tau_2\}$.

Obiekty klasy $B_i \in \Pi_C$ mogą być kodowane przy użyciu algorytmu ESPRESSO [7], natomiast w przedstawionym przykładzie dokonano kodowania przypisując odpowiednio: $K(B_4)$=01, $K(B_2)$=10, $K(B_3)$=11. Kod 00 jest przypisany do rozróżnienia klasy $B_i \notin \Pi_C$.

Tabela 1 dla przekształceń automatu Moore'a przedstawia funkcje (7).

Tab. 1. Fragment tabeli z przekształceniem dla S_1
Tab. 1. Fragment of transformed table structure for S_1

B_i	$K(B_i)$	a_S	$K(a_S)$	X_h	Φ_h	h
B_2	10****	a_4	0010	x_3	D_3	1
		a_7	1111	$/x_3 x_4$	$D_1D_2D_3D_4$	2
		a_6	0111	$/x_3 /x_4$	$D_2D_3D_4$	3
B_3	00001*	a_9	1010	1	D_1D_3	4

W kolumnie $K(B_i)$ pierwsze R_1 bitów przedstawia zmienne $\tau_\gamma \in \tau$, ostatnie R bitów to $T_\gamma \in T$. Jeżeli $\tau_1=\tau_2=0$, to RG jest źródłem dla kodu $K(B_i)$. W przeciwnym przypadku kody są brane z tabeli 2. Z tabeli 1 uzyskuje się funkcje (4). Dla przykładu równanie funkcji dla przerzutnika typu „D": przedstawia się następująco: $D_1 = \tau_1 \overline{\tau}_2 \overline{x}_3 x_4 \vee \overline{\tau}_1 \overline{\tau}_2 \overline{\mathrm{T}}_1 \overline{\mathrm{T}}_2 \mathrm{T}_3$.

W tabeli 2 zmienna $\tau_\gamma \in \tau$ jest równa 1 dla kodu $K(B_i)$ dla m-tej linii w tabeli. W przedstawionym przykładzie I_M=7 linii.

Tab. 2. Fragment tabeli BTK Moore'a S_1
Tab. 2. Fragment of the BTK Moore S_1

a_m	$K(a_m)$	B_i	$K(B_i)$	τ_m	m
a_2	0001	B_2	10	τ_1	1
a_3	1101				2
a_5	0100	B_4	01	τ_2	3
a_6	0111				4
a_7	1111				5
a_{11}	1100	B_7	11	$\tau_1\tau_2$	6
a_{12}	1001				7

W tabeli 2 przedstawiona jest funkcja (5), dla przykładu uzyskuje się równanie: $\tau_1 = \overline{T}_3 T_4 \vee T_1 T_2 \overline{T}_3$.

Implementując układy logiczne dla automatów Moore'a U_2 stosuje się redukcję do implementacji funkcji (2), (4) i (5) używając standardowego pakietu [12, 13, 14].

5. Wyniki eksperymentu

Zaproponowana metoda została porównana ze standardowym automatem Moore'a, przy użyciu specyfikacji testowych [10]. Badania zostały przeprowadzone dla 30 przykładów, na układzie CPLD rodziny XC9500 [12]. W celu przeprowadzenia badań napisany został program w języku C#, który w pierwszej kolejności przekształca automat Mealy'ego reprezentowane w formacie KISS2 do automatów Moore'a w tym samym formacie. Kolejnym etapem jest utworzenie programu w języku VHDL i podaniu go syntezie do konkretnego układu. W tabeli 3 przedstawione zostało 10 przykładowych otrzymanych rezultatów, gdzie reprezentowane są one przez więcej niż 60 przejść zapisanych w formacie KISS2.

W tabeli 3 symbol Q_i określa liczbę użytych makrokomórek w automacie Moore'a dla konkretnego przykładu wyszczególnionego w kolumnie B-mark, gdzie $i \in \{1,2\}$. We wszystkich przypadkach zastosowano algorytm ESPRESSO [6], który został wykorzystany do kodowania stanów. Średni zysk dla optymalnego kodowania stanów wynosi 1,71.

Tab. 3. Wyniki eksperymentu
Tab. 3. Results of experiments

B-mark	cse	dk16	keyb	planet	pma	S208	S420	S820	S832	tbk
Q_1	21	30	22	40	24	8	8	28	36	84
Q_2	13	17	12	22	12	8	7	15	17	43
Q_1/Q_2	1, 61	1, 76	1, 83	1, 82	2	1	1, 14	1, 87	2, 12	1, 95

6. Podsumowanie

Zaprezentowana metoda pozwala na zmniejszenie zużycia zasobów sprzętowych wymaganych do realizacji automatu Moore'a. Metoda ta bazuje na transformacji kodu stanu w kod klasy pseudorównoważnych stanów automatu Moore'a, która prowadzi do zmniejszenia kosztów realizowanego układu.

Przeprowadzone badania analityczne [8], jak również eksperymentalne pokazały, że proponowana metoda wykorzystuje mniejszą liczbę makrokomórek PAL, niż ma to miejsce w klasycznych metodach projektowania automatów Moore'a.

Zmniejszenie zużycia zasobów sprzętowych wiąże się ze zmniejszeniem układów kombinacyjnych w danym systemie. Dzięki temu uzyskujemy mniejszą liczbę cykli automatu, a co za tym idzie zwiększenie wydajności.

Przyszłe badania ukierunkowane zostaną na implementacje jednostki sterującej przy użyciu technologii FPGA [12, 13, 14] z zastosowaniem możliwości proponowanej metody. W proponowanej metodzie również zostaną wykorzystane specyfikacje testowe [10].

7. Literatura

[1] Baranov S.: Logic and System Design of Digital Systems. Tallinn: TUT Press, 2008.
[2] Barkalov A., Węgrzyn M.: Design of Control Units with Programmable Logic, Zielona Gora: University of Zielona Gora Press, 2006.
[3] Barkalov A., Titarenko L.: Logic Synthesis for FSM – based Control Units. Berlin: Springer, pp. 233, 2009.
[4] Solovjev V., Klimowicz A.: Logic design of digital Systems with programmable logic devices. Moscow: Hotline – Telecom, pp. 376, 2008.
[5] Barkalov A., Barkalov A.: Design of Mealy finite-state-machines with the transformation of objects codes. International Journal of Applied Mathematics and Computer Science, nr 1, pp. 151-158, 2005.
[6] De Micheli G.: Synthesis and Optimization of Digital Circuits, N.Y.: McGraw Hill, 1994.
[7] Kania D:, Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL. Gliwice: Silesian Technical Univ., 2004.
[8] Barkalov A., Titarenko L., Chmielewski S.: Hardware reduction for Moore FSM implemented with CPLD. Electronics and Telecommunications Quarterly, nr 2, pp. 317-333, 2009.
[9] Chattopadhyay S.: Area Conscious State Assignment with Flip-Flop and Output Polarity Selection for Finite State Machine Synthesis: A Genetic Algorithm Approach. The Computer Journal, nr 4, pp. 443-450, 2005.
[10] Yang S.: Logic Synthesis and Optimization Benchmarks User Guide. Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, 1991.
[11] Devadas S., Ma H.-K., Newton R., Sangiovanni-Vincentelli A.: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations, IEEE Transactions on Computer-Aided Design, pp. 1290-1300, 1988.
[12] http://www.xilinx.com
[13] http://www.altera.com
[14] http://www.cypress.com

otrzymano / received: 13.05.2010
przyjęto do druku / accepted: 01.10.2010 *artykuł recenzowany*