Zmniejszenie zużycia makrokomórek PAL w realizacjach układowych automatów Moore'a

Aleksander Barkalov, Larysa Titarenko, Sławomir Chmielewski

Streszczenie: W artykule została przedstawiona metoda zmniejszenia wymaganych zasobów sprzętowych do implementacji skończonych automatów stanów z wyjściami typu 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 pseudo-równoważnych stanów. Zaproponowane podejście pozwala zmniejszyć liczbę wymaganego zużycia zasobów sprzętowych bez zmniejszenia wydajności układów cyfrowych. Przedstawiony jest również przykład zaproponowanego rozwiązania oraz wyniki eksperymentu.

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

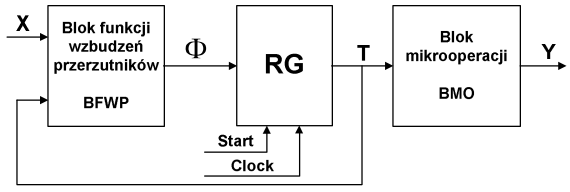
1. WPROWADZENIE

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 [14]. 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 cyfrowych (ang. Programmable Array Logic, PAL) [3,4,15,16].

Obecnie jedną z istotnych kwestii w przypadku implementowania automatów FSM przy wykorzystaniu układów CPLD jest zmniejszenie liczby zajętości makrokomórek PAL [8,12,13]. Minimalna liczba użytych stanów w dedykowanej sieci działań algorytmu (ang. Graph-Scheme of Algorithm, GSA) może rozwiązać ten problem. Koszt projektowania, a co za tym idzie, zużycie zasobów sprzętowych jest jednym z głównych celów w projektowaniu automatów FSM. Ponieważ jest to ważny czynnik, dlatego też zmniejszenie wykorzystanych zasobów sprzętowych powinno być nieustannie udoskonalane biorąc zawsze pod uwagę kompromis pomiędzy ceną, a wydajnością układu. Efektywne metody bazują na minimalizacji symbolicznej, algorytmach genetycznych i innych metodach heurystycznych [6,17]. W artykule proponowana jest metoda zmniejszania liczby makrokomórek PAL, gdzie przydzielenie stanów jest ukierunkowane na optymalizację mikrooperacji w bloku FSM. W pracy zawarto wyniki eksperymentu, które przeprowadzone zostały na popularnych układach testowych [18].

2. OPIS PROJEKTOWANIA AUTOMATÓW MOORE'A

Najpopularniejszym sposobem opisu automatu skończonego z wyjściami typu Moore'a jest struktura tablicy [1], w której a_m jest początkowym stanem automatu, $a_m \in A$, gdzie $A = \{a_1,...,a_M\}$ 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 $X = \{x_1,...,x_L\}$, wymuszając przejście $\langle a_m, a_s \rangle$; 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 $\Phi = \{D_1,...,D_R\}$; h jest numerem przejścia automatu ($h = 1$, …, H). Stan a_m zawiera zbiór mikrooperacji $Y(a_m) \subseteq Y$, gdzie $Y = \{y_1,...,y_N\}$. Tak ustalona struktura automatu Moore'a U_1 pokazana jest na rysunku 1.



Rys. 1. Schemat blokowy automatu Moore'a U_1

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

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

i blok mikrooperacji (BMO)

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

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

$K(a_m)$ stanu $a_m \in A$. Sygnał „Clock" przełącza stan rejestru RG.

Zmniejszanie zasobów sprzętowych w układach cyfrowych przy projektowaniu automatów FSM U_1 może być zrealizowana przy wykorzystaniu jednej z proponowanych metod [2,3]. W przypadku optymalizacji liczby użytych stanów [2], klasa pseudo-równoważnych stanów jest reprezentowana przez *R*-wymiarową przestrzeń boolowską.

Stan a_s, gdzie $a_s \in A$ jest pseudo-równoważnym stanem, jeżeli stan ten jest kolejnym stanem automatu dla wyjść poprzedzających go stanów. W takim przypadku mamy udoskonaloną metodę wykorzystania liczby stanów [3], każda mikrooperacja $y_n \in Y$ jest reprezentowana przez *R*-wymiarową przestrzeń boolowską. W obu przypadkach stany mogą być kodowane za pomocą dobrze znanych metod, takich jak ESPRESSO [7]. 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 zasobów sprzętowych dla obu bloków kombinacyjnych automatu Moore'a U_1.

3. IDEA PROPONOWANEJ METODY

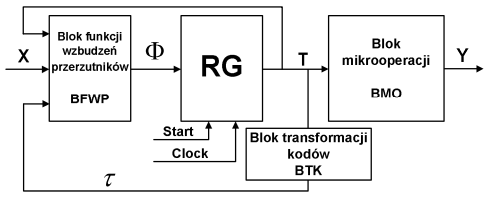
Niech $\Pi_A = \{B_1,...,B_I\}$ będzie zbiorem części A klasy pseudo-równoważnych stanów. Niech $\Pi_A = \Pi_B \cup \Pi_C$, gdzie $B_i \in \Pi_B$, jeżeli $|B_i| \geq 2$, i w przeciwnym wypadku $B_i \in \Pi_C$. Następnie kodujemy $B_i \in \Pi_B$ binarnym kodem $K(B_i)$ używając

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

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

Specyficzną cechą układu typu PAL jest duża liczba wejść makrokomórek i liczba termów na mikrokomórkę [3,4], dlatego też możliwe jest zastosowanie więcej niż jednego źródła kodowania stanów dla wejściowego bloku pamięci (BFWP) [8].

Wejściowy blok pamięci BFWP jest optymalizowany dzięki istnieniu pseudo-równoważnych stanów, przez co dany stan automatu przypisywany jest do kodu odpowiedniej klasy obiektu, przy wykorzystaniu makrokomórek PAL. Jako obiekt rozumiany jest wewnętrzny stan automatu i zbiór mikrooperacji, co jest możliwe dla automatów Moore'a U_2 o strukturze pokazanej na rysunku 2.



Rys. 2. Schemat blokowy automatu Moore'a U_2

Dla skończonego automatu U_2, blok BFWP implementuje funkcje

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

blok transformacji kodów (BTK) przekształca kody pseudo-równoważnych 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 układach cyfrowych przy użyciu funkcji (2), (4) i (5) oraz zastosowaniu dedykowanego układ CPLD.

4. PRZYKŁAD PROPONOWANEJ METODY

Rozważmy przykład automatu Moore'a FSM S_1 o następującej charakterystyce: $M = 13$, $N = 8$, $\Pi_A = \{B_1,...,B_7\}$, 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:

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

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:

$$B_2 \to x_3 a_4 \vee \bar{x}_3 x_4 a_7 \vee \bar{x}_3 \bar{x}_4 a_6;$$
$$B_3 \to a_9. \tag{7}$$

Po czym, uzyskuje się $R = 4$, $T = \{T_1,...,T_4\}$, $\Phi = \{D_1,...,D_4\}$. 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 FSM S_1

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

$$y_1 = T_3\overline{T}_4 ; y_2 = T_2\overline{T}_3\overline{T}_4 ; y_3 = \overline{T}_3T_4 ;$$
$$y_4 = T_3T_4 \vee T_1T_2T_3 ; y_5 = T_1T_2 ;$$
$$y_6 = T_1T_2T_4 ; y_7 = \overline{T}_1T_3 ; y_8 = T_1T_4 . \quad (8)$$

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.

Analizując rysunek 3, który pokazuje, że kod $K(a_8)$ i $K(a_9)$ zapisać można za pomocą iloczynu uogólnionego, z tabeli Karnaugh. 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 $\Pi_B = \{B_1, B_3, B_5, B_6\}$, $\Pi_C = \{B_2, B_4, B_7\}$. Z czego 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 zawiera następujące kolumny: B_i, $K(B_i)$, a_S, $K(a_S)$, X_h, Φ_h, h. Fragment tabeli 1 przedstawia funkcje (7).

Tabela 1.
Fragment tabeli z przekształceniem dla 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	$/x_3x_4$	$D_1D_2D_3D_4$	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 $\tau_r \in \tau$, ostatnie R bitów to $T_r \in T$. Jeżeli $\tau_1 = \tau_2 = 0$, to rejestr 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: $D_1 = \tau_1\overline{\tau}_2\overline{x}_3x_4 \vee \overline{\tau}_1\overline{\tau}_2\overline{T}_1\overline{T}_2T_3$.

Tabela 2 zawiera kolumny a_m, $K(a_m)$, B_i, $K(B_i)$, τ_m, m, gdzie τ_m zawiera zmienną $\tau_r \in \tau$, która jest równy 1 dla kodu $K(B_i)$ dla m-tej linii w tabeli. W przedstawionym przykładzie $I_M = 7$ linii.

Tabela 2.
Fragment tabeli BTK Moore'a 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: $\tau_1 = \overline{T}_3T_4 \vee T_1T_2\overline{T}_3$.

Implementując układy logiczne dla automatów Moore'a U_2 stosuje się redukcję do implementacji funkcji (2), (4) i (5) używając standardowego pakietu [9-11].

5. WYNIKI EKSPERYMENTU

Zaproponowana metoda została porównana z standardowym automatem Moore'a, przy użyciu specyfikacji testowych [18]. Badania zostały przeprowadzone dla 30 przykładów, na układzie CPLD rodziny XC9500 [11]. W celu przeprowadzeniu badań napisany został program w języku C#, który w pierwszej kolejności przekształca automat Mealy'ego FSM reprezentowane w formacie KISS2 do automatów Moore'a FSM w tym samym formacie. Kolejny etap to utworzenie programu w języku VHDL i podaniu go syntezie do konkretnego układu. W tabeli 3 przedstawione zostały 10 przykładowych otrzymanych rezultatów, gdzie reprezentowane są one przez więcej niż 60 przejść zapisanych w formacie KISS2.

Tabela 3.
Wyniki eksperymentu

B-	Q_1	Q_2	Q_1/Q_2
cse	21	13	1,61
dk16	30	17	1,76
keyb	22	12	1,83
planet	40	22	1,82
pma	24	12	2,00
S208	8	8	1,00
S420	8	7	1,14
S820	28	15	1,87
S832	36	17	2,12
tbk	84	43	1,95

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 użyty został dobrze znany algorytm ESPRESSO [7] do kodowania stanów.

Średni zysk dla optymalnego kodowania stanów wynosi 1.71.

6. PODSUMOWANIE

Zaprezentowana metoda pozwala na zmniejszenie zasobów sprzętowych wymaganych do realizacji automatu skończonego z wyjściami typu Moore'a. Metoda ta bazuje na transformacji kodu stanu w kod klasy pseudo-równoważnych stanów automatu Moore'a, która prowadzi do zmniejszenia kosztów realizowanego układu.

Przeprowadzone badania analityczne [5], jak również eksperymentalne pokazały, że proponowana metoda wykorzystuje mniejszą liczbę makrokomórek PAL, niż ma to miejsce w klasycznych metodach projektowania skończonych automatów.

Zmniejszenie wykorzystania zasobów sprzętowych bardzo często jest ukierunkowane z zmniejszeniem układów kombinacyjnych w danym systemie. Dzięki czemu uzyskujemy mniejszą liczbę cykli automatu, a co za tym idzie zwiększenie wydajności.

Przyszłe badania ukierunkowane zostaną na implementacji jednostki sterującej przy użyciu technologii FPGA [9-11] z zastosowaniem możliwości proponowanej metody. W proponowanej metodzie również zostaną wykorzystane specyfikacje testowe [18].

LITERATURA

[1] Baranov S., *Logic and System Design of Digital Systems.* Tallinn: TUT Press, 2008.

[2] Barkalov A. and Węgrzyn M., *Design of Control Units with Programmable Logic*. Zielona Góra: University of Zielona Gora Press, 2006.

[3] 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, ss. 151-158, 2005.

[4] Barkalov A., Titarenko L., *Logic Synthesis for FSM – based Control Units*. Berlin: Springer, ss. 233, 2009.

[5] Barkalov A., Titarenko L., Chmielewski S., "Hardware reduction for Moore FSM implemented with CPLD", *Electronics and Telecommunications Quarterly*, nr 2, ss. 317-333, 2009.

[6] 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, ss. 443-450, 2005.

[7] De Micheli G., *Synthesis and Optimization of Digital Circuits*, N.Y.: McGraw Hill, 1994.

[8] 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*, ss. 1290-1300, 1988.

[9] http://www.altera.com

[10] http://www.cypress.com

[11] http://www.xilinx.com

[12] Kam T., Villa T., Brayton R., Sangiovanni-Vincentelli A., *Synthesis of Finite State Machines: Functional Optimization*, Boston/London/Dordrecht: Kluwer Academic Publishers, 1998.

[13] Kania D., *Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL*, Gliwice: Silesian Technical University, 2004.

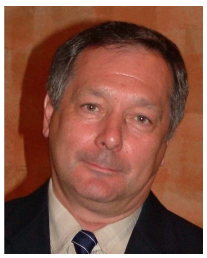
[14] Solovjev V., Klimowicz A., *Logic design of digital Systems with programmable logic devices*, Moscow: Hotline – Telecom, ss. 376, 2008.

[15] Maxfield C., *The Design Warrior's Guide to FPGAs*, Amsterdam: Elsevier, ss. 561, 2004.

[16] Navabi Z., *Embedded Core Design with FPGAs*, N.Y.: McGraw Hill, ss. 457, 2007.

[17] Xia Y. and Almaini A., "Genetic algorithm based state assignment for power and area optimization", *IEEP. – Comput. Dig. T.*, ss. 128-133, 2002.

[18] Yang S., "Logic Synthesis and Optimization Benchmarks User Guide", *Microelectronics Center of North Carolina*, Research Triangle Park, North Carolina, 1991.



prof. dr hab. inż. Aleksander Barkalov
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

tel.: 68 328 26 93
e-mail: A.Barkalov@iie.uz.zgora.pl

dr hab. inż. Larysa Titarenko, prof. UZ
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

tel.: 68 328 26 93
e-mail: L.Titarenko@iie.uz.zgora.pl

mgr inż. Sławomir Chmielewski
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Licealna 9
65-417 Zielona Góra

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