Larysa TITARENKO, Olena HEBDA, **Alexander BARKALOV**
UNIWERSYTET ZIELONOGÓRSKI,
ul. Licealna 9, 65-417 Zielona Góra

Optymalizacja układu logicznego mikroprogramowanego automatu Moore'a przy użyciu nano-PLA

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 2005 pracuje na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

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

Mgr inż. Olena HEBDA

Mgr inż. Olena Hebda – absolwentka Narodowego Aerokosmicznego Uniwersytetu „KhAI". Ukończyła w roku 2009 studia o specjalności biotechniczne i medyczne aparaty i systemy. Od 2009 roku jest doktorantką na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: O.Shapoval@weit.uz.zgora.pl

Streszczenie

W artykule została przedstawiona metoda syntezy mikroprogramowanego automatu Moore'a implementowanego w układach nano-PLA. Metoda jest ukierunkowana na redukcję zasobów sprzętowych, potrzebnych do implementacji automatu Moore'a. Jest ona oparta na optymalnym kodowaniu stanów i rozbijaniu matrycy termów na dwie części. Takie podejście pozwala zmniejszyć liczbę linii w tablice przejść automatu Moore'a do odpowiedniej liczby linii w równoważnym automacie z wyjściami typu Mealy'ego.

Słowa kluczowe: mikroprogramowany automat Moore'a, nano-PLA, stany pseudoekwiwalentne, układ logiczny.

Optimization of a logic circuit of the microprogrammed Moore machine with use of nano-PLA

Abstract

The model of the microprogrammable Moore automaton [6] is often used during the digital control systems realization [1, 4]. The development of microelectronics has led to appearance of different programmable logic devices [13, 15, 18], which are used for implementing microprogrammable automaton (MPA) logic circuits. One of the important problems of MPA synthesis is the decrease in the chip space occupied by MPA logic circuit. Solution of this problem allows decreasing the power consumption and increasing the clock rate. The methods of solution of this problem depend strongly on logic elements used for implementing the MPA logic circuit [2, 3, 13, 15]. In this paper we discuss the case when nanoelectronic programmable logic arrays (PLA) are used for implementing Moore MPA logic circuit. The approach is connected with optimal state encoding and decomposition of a matrix of terms in two sub-matrices (Fig. 2). To do it, the classes of the pseudoequivalent states are used [1, 4]. Such an approach allows reducing the number of rows of the structure table of Moore MPA up to this value of the equivalent Mealy MPA. As a result the area of the matrices generating input memory functions is optimized. The example of application of the proposed methods is given.

Keywords: microprogrammable Moore automaton, nano-PLA, pseudo-equivalent states, logic circuit.

Prof. dr hab. inż. Alexander BARKALOV

Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki w Donieckiej. Współpracował aktywnie z Instytutem Cybernetyki im. V.M. Glushkova w Kijowie, gdzie uzyskal 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

1. Wprowadzenie

Jednostka sterująca [2, 5] jest jednym z najważniejszych elementów systemu cyfrowego. Jest odpowiedzialna za zarządzanie innymi blokami projektowanego układu. Jedną z popularnych metod realizacji jednostek sterujących jest zastosowanie modelu mikroprogramowanego automatu (MPA) Moore'a [7]. Jednym z najbardziej istotnych problemów syntezy MPA jest redukcja liczby jednostek sprzętowych, potrzebnych dla implementacji układu logicznego MPA. Jego rozwiązanie polepszy takie parametry jak szybkość działania i zużycie energii [1, 5]. Ponadto może pomóc rozwijającemu się przemysłowi, np. na terenie województwa lubuskiego, ponieważ pozwoli producentom na uzyskanie przewagi konkurencyjnej na rynku w stosunku do innych produktów.

Metody rozwiązania tego problemu w istotny sposób zależą od elementów logicznych wykorzystanych do implementacji układu logicznego MPA [11, 16]. W tym artykule zostanie rozpatrzony przypadek wykorzystania nano-PLA (ang. nanoelectronic Programmable Logic Arrays) dla implementacji MPA Moore'a.

Już w latach 70-tych PLA zostały popularną bazą dla implementacji układów logicznych [8, 12, 17]. Jak zostało zaznaczone w pracy [3], udane struktury obliczeniowe wracają z powrotem w kolejnych etapach rozwoju technologicznego. Obecnie powrót PLA można zaobserwować w hybrydowych FPGA (ang. hybrid Field Programmable Gate Array) [13, 15], w CoolRunner CPLD (ang. Complex Programmable Logic Device) firmy Xilinx [18] oraz we współczesnej nanoelektronice [3, 10]. W nanoelektronice takie urządzenia nazywają się nano-PLA. Aktualnie są prowadzone intensywne badania w dziedzinach związanych z nano-PLA [3, 14]. Największym problemem w układach nanoelektronicznych jest wysokie prawdopodobieństwo wystąpienia wad technologicznych [3, 14]. Dlatego opracowanie nowych metod, które pozwolą zmniejszyć złożoność układu logicznego MPA oraz zmniejszyć liczbę zasobów sprzętowych potrzebnych dla implementacji tego układu, jest bardzo ważne.

2. Klasyczne metody projektowania MPA Moore'a

Automat Moore'a może być przedstawiony jako sieć działań (GSA – ang. Graph Sheme of Algorithm), a także opisany za pomocą tablicy przejść [1]. Tablica przejść składa się z następujących kolumn: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Φ_h, h. a_m jest początkowym stanem automatu, gdzie $a_m \in A$ jest zbiorem stanów automatu; $K(a_m)$ jest kodem stanu a_m zapisanym na $R = \lceil log_2 M \rceil$ bitach (do kodowania stanów wykorzystujemy zmienne wewnętrzne ze zbioru $T = \{T_1, \ldots, T_R\}$). a_s jest następnym

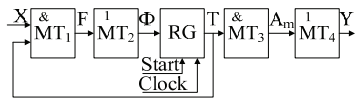
stanem automatu; $K(a_s)$ jest kodem stanu a_s. X_h jest koniunkcją zmiennych ze zbioru wejściowych warunków cyfrowych $X=\{x_1,\ldots,x_L\}$, wyznaczających przejście $\langle a_m,\ a_s\rangle$. Φ_h jest zbiorem wejściowych funkcji wzbudzeń pamięci, które są równe 1 przy przełączeniu pamięci automatu z kodu $K(a_m)$ na kod $K(a_s)$, $\Phi_h \subseteq \Phi=\{\varphi_1,..,\varphi_R\}$. h jest numerem przejścia automatu, $h=\overline{1,H_1}$. W kolumnie a_m jest wpisany zbiór zmiennych wyjściowych (ZZW) Y_q. Te wyjściowe zmienne są generowane w stanie $a_m \in A$ ($Y_q \subseteq Y=\{y_1,\ldots,y_N\}$, $q=1,...,Q$). Opisana tablica jest wykorzystywana dla tworzenia następujących układów funkcji:

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

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

Układy funkcji (1) – (2) wyznaczają układ logiczny MPA. Przy wykorzystaniu PLA-bazy układ logiczny MPA Moore'a będzie składać się z dwóch PLA: „rdzeniowego" PLA (CPLA – ang. Core PLA) i wyjściowego PLA (OPLA – ang. Output PLA). W tym modelu CPLA implementuje (1), OPLA implementuje układ (2).

Układy funkcji (1) – (2) mogą być zaimplementowane za pomocą tylko dwóch matryc. Pierwsza z nich (AND-matryca) implementuje koniunkcje termów systemów. Druga matryca (OR-matryca) implementuje funkcje. Ale takie rozwiązanie prowadzi do układu MPA z największym możliwym zużyciem zasobów sprzętowych [3]. Znacznie częściej w praktyce używano 4 matryc [2, 5, 6] (rys. 1). Oznaczymy taki model MPA Moore'a jako MPA U_1.



Rys. 1. Matrycowy układ automatu Moore'a U_1
Fig. 1. Matrix implementation of FSM U_1

Przeanalizujmy komponenty układu matrycowego, pokazanego na rys. 1. Koniunkcyjna matryca MT_1 implementuje układ termów $F=\{F_1,\ldots,F_{H_1}\}$; dysjunkcyjna matryca MT_2 – system (1); koniunkcyjna matryca MT_3 – termy A_m ($m=\overline{1,M}$), odpowiednio do stanów MPA; dysjunkcyjna matryca MT_4 – system (2). Rejestr RG przechowuje kody stanów. Sygnał „Start" wykorzystuje się dla załadowania stanu początkowego $a_1 \in A$, a sygnał „Clock" przełącza stan rejestru w zależności od funkcji Φ. Matryce MT_1 i MT_2 wyznaczają blok CPLA, a matryce MT_3 i MT_4 blok OPLA.

Główną wadą automatu Moore'a jest znaczne przekraczanie parametru H_0, który oznacza liczbę wierszy w tablice przejść równoważnego automatu Mealy'ego. Oprócz tego liczba stanów automatu Moore'a może znacznie przekraczać liczbę stanów M_0 równoważnego automatu Mealy'ego [1]. Dlatego w praktyce często występują następujące warunki:

$$H_0 < H_1;\ R_0=\lceil log_2 M_0 \rceil < R. \tag{3}$$

Złożoność każdej z matryc można określić jako powierzchnię $S(M_i)$ kryształu potrzebną dla jej realizacji $i=\overline{1,4}$. W artykułach teoretycznych tę powierzchnię wyznacza się w jednostkach umownych [2]. Dla automatu U_1 można otrzymać następujące oceny:

$$\begin{aligned} &S(M_1)=2(L+R)H_1;\ S(M_2)=H_1R;\\ &S(M_3)=2R\cdot M;\ \ S(M_4)=M\cdot N. \end{aligned} \tag{4}$$

Powierzchnia $S(U_1)$, którą zajmuje automat U_1, wyznaczana jest jako suma powierzchni (4). Biorąc pod uwagę nierówności (3), powierzchnia zajmowana przez układ logiczny automatu Moore'a zawsze będzie większa od powierzchni potrzebnej dla równoważnego automatu Mealy'ego.

Istnieje wiele metod ukierunkowanych na redukcję zasobów sprzętowych, potrzebnych do realizacji układu logicznego automatu Moore'a [11, 16], ale dotyczą one wykorzystania CPLD lub FPGA technologii i dlatego nie mogą być bezpośrednio użyte w przypadku realizacji MPA Moore'a na bazie PLA. Biorąc to pod uwagę, proponujemy nową metodę syntezy MPA Moore'a przy wykorzystaniu PLA. Jest ona rozwinięciem metody opisanej w [6].

Jedną z cech automatu Moore'a są stany pseudoekwiwalentne [5]. Stany a_i, a_j nazywamy pseudoekwiwalentnymi jeśli odpowiednie do nich operacyjne wierzchołki sieci działań są połączone z wejściem tego samego wierzchołka [1]. Są dwie podstawowe metody oparte na istnieniu klas pseudoekwiwalentnych [5]: metoda optymalnego kodowania stanów i metoda transformacji kodów stanów.

1. *Metoda optymalnego kodowania stanów*. Stan $a_m \in A$ jest kodowany w taki sposób, żeby każdą klasę $B_i \in \Pi_A$ przedstawić jednym uogólnionym interwałem R-wymiarowej przestrzeni Boole'a. To prowadzi do automatu U_2, którego schemat pokrywa się ze schematem na rys. 1, ale matryca M_1 realizuje tylko $H_0=H_2<H_1$ termów. Jednak takie kodowanie nie zawsze może być wykonane [5], co prowadzi do tego, że $H_2>H_0$.

2. *Metoda transformacji kodów stanów*. Każdej klasie $B_i \in \Pi_A$ odpowiada kod $K(B_i)$, $R_B=\lceil log_2 I \rceil$. Zwróćmy uwagę, że $I=M_0$, gdzie M_0 jest liczbą stanów w równoważnym automacie Mealy'ego. Dla kodowania wykorzystuje się zmienne ze zbioru τ, gdzie $|\tau|=R_B$. Potem formuje się układ funkcji

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

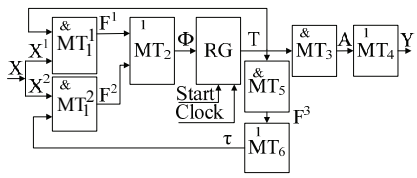
który wyznacza zasadę transformacji kodów $K(a_m)$ w kody $K(B_i)$. Tablice przejść przekształca się w taki sposób, że stany $a_m \in B_i$ zastępuje się klasami $B_i \in \Pi_A$ w kolumnie stanu początkowego. Takie podejście zmniejsza długość tablicy przejść do H_0, a ilość zmiennych sprzężenia zwrotnego zmniejsza się do $R_B<R$. Układ (5) jest realizowany za pomocą bloku transformacji kodów stanów (CTPLA) (ang. Code Transformer PLA), co prowadzi do schematu U_3. Takie podejście pozwala zmniejszyć powierzchnie matryc MT_1 i MT_2 (w porównaniu z U_1), jednak matryce MT_5 i MT_6 również angażują zasoby sprzętowe.

3. Podstawowa idea proponowanej metody

Dla zmniejszenia powierzchni każdej z matryc w układzie MPA Moore'a wystarczy zmniejszyć liczbę wejść i wyjść. Aby zmniejszyć powierzchnię układu matrycy dla bloku CPLA MPA Moore'a, należy wziąć pod uwagę istnienie stanów pseudoekwiwalentnych.

Niech $\Pi_A=\{B_1,\ldots,B_I\}$ będzie rozbiciem zbioru stanów na klasy pseudoekwiwalentne. Warto zauważyć, że każda klasa stanów pseudoekwiwalentnych odpowiada unikalnemu stanowi równoważnego automatu Mealy'ego. Zakodujmy stany $a_m \in A$ w taki sposób, żeby maksymalnie możliwa liczba klas $B_i \in \Pi_A$ była przedstawiona jednym interwałem R-wymiarowej przestrzeni

Boole'a. Odniesiemy takie klasy do podzbioru Π_{RG}, a resztę do podzbioru Π_{CT}, gdzie $|\Pi_{CT}| = I_{CT}$. Zakodujmy klasy $B_i \in \Pi_{CT}$ za pomocą binarnego kodu $K(B_i)$, gdzie $R_{CT} = \lceil log_2(I_{CT}+1) \rceil$. Takie podejście prowadzi do MPA Moore'a U_4 (rys. 2).



Rys. 2. Matrycowy układ automatu Moore'a U_4
Fig. 2. Matrix implementation of FSM U_4

W automacie U_4 blok CPLA składa się z matryc MT_1^1, MT_1^2 i MT_2, blok OPLA z matryc MT_3 i MT_4, a blok CTPLA z matryc MT_5 i MT_6. Zbiór termów F jest podzielony na podzbiory F^1 (realizowany w MT_1^1) i F^2 (realizowany w MT_1^2).

Matryca MT_1^1 wytwarza termy $F_h \in F^1$:

$$F_h = (\bigwedge_{r=1}^{R} T_r^{l_{ir}}) \cdot X_h, \tag{6}$$

gdzie $l_{ir} \in \{0,1,*\}$ jest wartością r-tego bitu kodu $K(B_i)$, $T_r^0 = \overline{T_r}$, $T_r^1 = T_r$, $T_r^* = 1$ ($r = \overline{1,R}$, $h = \overline{1,|F^1|}$).

Matryca MT_1^2 wytwarza termy $F_h \in F^2$:

$$F_h = (\bigwedge_{r=1}^{R_1} \tau^{l_{ir}}) \cdot X_h \quad (h = \overline{1,|F^2|}). \tag{7}$$

Matryce MT_3 i MT_4 są odpowiednikami matryc MT_3 i MT_4 MPA U_1 (rys. 1). Matryca MT_5 realizuje układ termów F^3, które tworzą funkcje (5). Zatem źródłem wejść do matrycy MT_1^1 jest rejestr RG, podczas gdy dla MT_1^2 jest transformator kodów.

Takie podejście pozwala zmniejszyć całkowitą liczbę termów w układach F^1 i F^2 do H_0. Zatem łączna powierzchnia matryc MT_1^1, MT_1^2 i MT_2 jest mniejsza niż powierzchnia matryc MT_1 i MT_2 w U_1. Blok CTPLA używa mniej zasobów sprzętowych niż CTPLA w MPA U_3, ponieważ jest potrzebny dla przekształcenia tylko części kodów stanów (tylko dla $a_m \in B_i$, gdzie $B_i \in \Pi_{CT}$).

Złożoność układu logicznego MPA Moore'a U_4 może być wyrażona jako:

$$S(MT_1^1) = 2(L_1 + R)H_0^1; S(MT_1^2) = 2(L_2 + R_{CT})H_0^2; S(MT_2) = H_0 R; \\ S(MT_5) = 2H(F^3)R; \quad S(MT_6) = H(F^3)R_{CT}. \tag{8}$$

$S(MT_3)$ i $S(MT_4)$ wyznaczamy jak w (4).

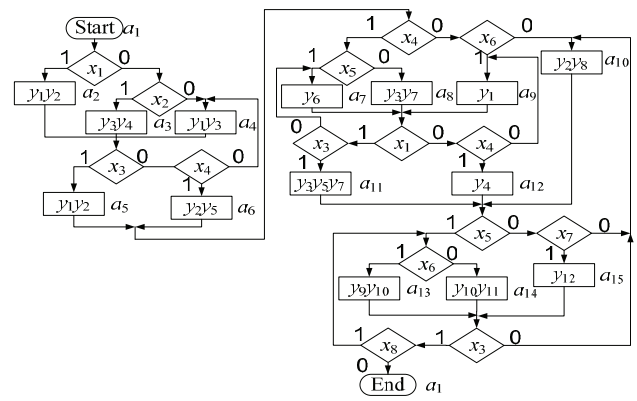
Zaproponowana metoda składa się z następujących etapów syntezy:

1. Utworzenie zbioru stanów A i rozdzielenie go na klasy stanów pseudoekwiwalentnych Π_A.
2. Optymalne kodowanie stanów $a_m \in A$ i wyznaczenie podzbiorów Π_{RG} i Π_{CT}.
3. Kodowanie klas $B_i \in \Pi_{CT}$.
4. Utworzenie tablicy przejść dla klas $B_i \in \Pi_{RG}$ i $B_i \in \Pi_{CT}$.
5. Utworzenie tablicy transformacji kodów.
6. Utworzenie układów funkcji, definiujących matryce U_4.
7. Zaimplementowanie MPA Moore'a przy użyciu nano-PLA.

4. Przykład zastosowania zaproponowanej metody

Przeanalizujmy GSA Γ_1 pokazaną na rys. 3. Z rys. 3 mogą być otrzymane następujące parametry: $A = \{a_1,\ldots,a_{15}\}$, $M = 15$, $X = \{x_1,\ldots,x_8\}$, $L = 8$, $Y = \{y_1,\ldots,y_{12}\}$, $N = 12$, $R = 4$ i $T = \{T_1,\ldots,T_4\}$. Dla GSA Γ_1 można obliczyć, że $H_1 = 53$ i $H_0 = 21$. Używając (4) można określić potrzebne zasoby sprzętowe: $S(MT_1) = 2(8+4)\cdot 53 = 1272$; $S(MT_2) = 53\cdot 4 = 212$; $S(MT_3) = 2\cdot 8\cdot 15 = 240$; $S(M_4) = 15\cdot 12 = 180$; $S(U_1) = 1904$. Zatem matryce MT_1 i MT_1 zajmują 67% całkowitej powierzchni kryształu.

W przypadku GSA Γ_1 istnieje 6 klas pseudoekwiwalentnych $\Pi_A = \{B_1,\ldots,B_6\}$, gdzie $B_1 = \{a_1\}$, $B_2 = \{a_2,a_3,a_4\}$, $B_3 = \{a_5,a_6\}$, $B_4 = \{a_7,a_8,a_9\}$, $B_5 = \{a_{10},a_{11},a_{12}\}$, $B_6 = \{a_{13},a_{14},a_{15}\}$. Zakodujmy stany $a_m \in A$ jak pokazano na rys. 4. Z siatki Karnaugha (rys. 4) można otrzymać następujące kody: $K(B_1) = 000$, $K(B_2) = **10$, $K(B_3) = 00*1$, $K(B_4) = 10**$. Klasa B_5 i klasa B_6 potrzebują dwóch interwałów czterowymiarowej przestrzeni boolowskiej. Można obliczyć, że $H_2 = 28$ i $S(MT_1) = 2(8+4)28 = 672$, $S(MT_2) = 28\cdot 4 = 112$, $S(U_2) = 1204$. W tym przypadku matryce MT_1 i MT_2 zajmują 56% całkowitej powierzchni kryształu.



Rys. 3. Sieć działań Γ_1
Fig. 3. Initial graph-scheme of algorithm Γ_1

T_1T_2 \ T_3T_4	00	01	11	10
00	a_1	a_5	a_6	a_2
01	a_{10}	a_{11}	a_{12}	a_3
11	a_{13}	a_{14}	a_{15}	a_4
10	a_7	a_8	a_9	*

Rys. 4. Kody stanów automatu U_4
Fig. 4. The states codes for FSM U_4

Zatem $\Pi_{RG} = \{B_1,\ldots,B_4\}$, $\Pi_{CT} = \{B_5,B_6\}$, $I_{CT} = 2$, $R_{CT} = 2$, $\tau = \{\tau_1,\tau_2\}$. Użyjmy kodu 00 dla identyfikacji $B_i \notin \Pi_{CT}$ i niech $K(B_5) = *1$, $K(B_6) = 1*$.

Dla utworzenia transformowanej tablicy przejść jest potrzebny układ uogólnionych funkcji przejść. W przypadku Γ_1 układ dla klas $B_i \in \Pi_{RG}$ jest następujący:

$$B_1 \to x_1a_2 \vee \bar{x}_1x_2a_3 \vee \bar{x}_1\bar{x}_2a_4;\ B_2 \to x_3a_5 \vee \bar{x}_3x_4a_6 \vee \bar{x}_3\bar{x}_4a_4;$$
$$B_3 \to x_4x_5a_7 \vee x_4\bar{x}_5a_8 \vee \bar{x}_4x_6a_9 \vee \bar{x}_4\bar{x}_6a_{10};$$
$$B_4 \to x_1x_3a_{11} \vee x_1\bar{x}_3a_7 \vee \bar{x}_1x_4a_{12} \vee \bar{x}_1\bar{x}_4a_9. \quad (9)$$

Dla klas $B_i \in \Pi_{CT}$ możemy otrzymać następujący układ:

$$B_5 \to x_5x_6a_{13} \vee x_5\bar{x}_6a_{14} \vee \bar{x}_5x_7a_{15} \vee \bar{x}_5\bar{x}_7a_{10};$$
$$B_4 \to x_3x_8x_6a_{13} \vee x_3x_8\bar{x}_6a_{14} \vee x_3\bar{x}_8a_1 \vee \bar{x}_3a_{10}. \quad (10)$$

Zatem $H_0^1 = 14$ i $H_0^2 = 8$, $S(M_2) = (14+8)4 = 88$ oraz $X^1 = \{x_1,\ldots,x_6\}$ i $X^2 = \{x_3, x_5, x_6, x_7, x_8\}$, $L_1 = 6$ i $L_2 = 5$. Więc $S(MT_1^1) = 2(6+4)14 = 280$, $S(MT_1^2) = 2(5+2)8 = 112$, natomiast $S(MT_3) + S(MT_4) = 240 + 180 = 420$ (jak w MPA U_1).

W tym przypadku układ (5) ma $H(F^3) = 4$ termów. Zatem $S(M_5) = 2 \cdot 4 \cdot 4 = 32$ i $S(M_6) = 4 \cdot 2 = 8$. Dla realizacji matryc MT_3 i MT_4 jest potrzebne utworzenie układu (2). Na przykład $y_1 = A_2 \vee A_4 \vee A_5 \vee A_9$. Ze względu na oczywistość kolejnych działań nie są one rozpatrywane poniżej.

5. Analiza proponowanej metody

Porównajmy proponowaną metodę z MPA U_2 i U_3. Użyjemy probabilistycznego podejścia zaproponowanego w [9] i opracowanego w [4]. Takie podejście ma trzy kluczowe punkty:

1. Zastosowanie klasy GSA zamiast określonego GSA. Ta klasa charakteryzuję się parametrami p_1 i p_2. Oznaczymy liczbę wierzchołków w GSA Γ przez Q, wierzchołków operacyjnych przez Q_1, a wierzchołków warunkowych przez Q_2. $p_1 = Q_1/(Q-2)$ ($p_2 \approx 1 - p_1$) jest prawdopodobieństwem tego, że okreslony wierzchołek GSA Γ będzie operacyjnym (warunkowym).
2. Zastosowanie matrycowej implementacji układu logicznego jednostki sterującej [2]. Umożliwia to wyznaczenie ilości zasobów sprzętowych jako objętośći matryc danej jednostki sterującej.
3. Zastosowanie charakterystyk względnych zamiast bezwzględnych. Będziemy analizować relacje $\eta = S(U_i)/S(U_j)$, gdzie $S(U_{i,j})$ jest liczbą zasobów sprzętowych potrzebnych do implementacji układu jednostki sterującej.

Skorzystajmy z wyników prac [2, 4], gdzie oszacowanie głównych parametrów MPA przedstawiono jako funkcje charakterystyk GSA i współczynników:

$$H_0 = 4.44 + 1.44p_1Q/p_3;\ H = 12.6 + 2.16p_1Q/p_3;$$
$$M = p_1Q + 1;\ L = (1-p_1)Q/p_4. \quad (11)$$

Q_0 jest liczbą zbiorów warunków wyjściowych, $p_3 = p_1Q/Q_0 \in \{1.1;1.2\}$, $p_4 = p_2Q/L \in \{1.1;1.2\}$.

Przeanalizujmy metodę optymalnego kodowania stanów oraz metodę transformacji kodów stanów. Zużycie zasobów sprzętowych dla MPA U_2 i U_3 jest zdefiniowane jako

$$S(U_2) = \sum_{i=1,4} S(M_i) = (p_5H_0 + (1-p_5)H_1)(3R + 2L) + M(2R + N); \quad (12)$$

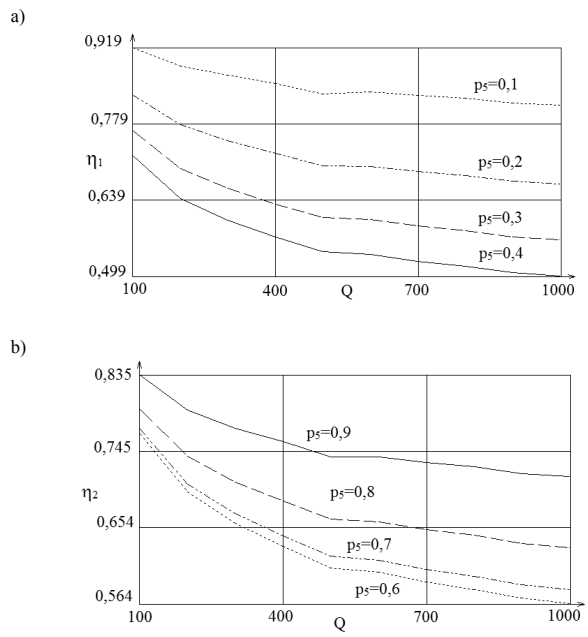
$$S(U_3) = \sum_{i=1,6} S(M_i) = H_0(2(R_B + L) + R) + M(2R + N) + H(F^3)(2R + R_B). \quad (13)$$

$p_5 \in [0,1]$ jest współczynnikiem efektywności metody optymalnego kodowania. Przy $p_5 = 1$ wszystkie klasy stanów pseudoekwiwalentnych mogą być przedstawione jednym interwalem R-wymiarowej przestrzeni boolowskiej. Używając równania (12) i (13), możemy otrzymać odpowiednio $S(U_2) = f(p_1, Q, p_3, p_4, N, p_5)$ i $S(U_3) = f(p_1, Q, p_3, p_4, N)$. Równania końcowe są zbyt duże i mało informatywne dla czytelnika, dlatego nie będziemy ich tu zamieszczać.

W podobny sposób możemy otrzymać $S(U_4) = f(p_1, Q, p_3, p_4, N, p_5, p_6)$, gdzie $H(F^3) = Mp_6$, $p_6 = \overline{0,5;0,8}$.

Analiza różnych GSA pokazała, że $Q \in \{100,\ldots,1000\}$ i $N \le 100$. Przeanalizujmy funkcje $\eta_1 = S(U_4)/S(U_2)$ i $\eta_2 = S(U_4)/S(U_3)$. Przykładowe wyniki zostały pokazane na rys. 5 ($p_1 = 0,6$; $p_3 = 1,1$; $p_4 = 1.1$; $N = 100$; $p_6 = 0,6$).

Analiza funkcji η_1 i η_2 pokazała, że zużycie zasobów sprzętowych w modelu U_4 jest niższe niż w U_2 i w U_3. Najlepsze wyniki otrzymujemy dla GSA z liczbą wierzchołków $700 \le Q \le 1000$. Oprócz tego funkcje $\eta_{1,2}$ w istotny sposób zależą od p_5. Przy wzroście p_5 efektywność zaproponowanej metody w porównaniu z U_2 zmniejsza się, ale nawet przy $p_5 = 0,9$ można otrzymać zysk około 6 – 7 % w zależności od innych parametrów GSA. Maksymalny zysk przy $p_5 = 0,9$ wyniósł 7,5%. Odwrotna sytuacja jest przy porównaniu U_4 z U_3, przy $p_5 = 0,1$ można zyskać około 7 –8 % powierzchni. Maksymalny zysk przy $p_5 = 0,1$ wyniósł 8,1%.



Rys. 5. Porównanie modelu U_4 i U_2 (a) oraz U_4 i U_3 (b)
Fig. 5. Comparison of models: a – U_4 and U_2; b – U_4 and U_3

6. Wnioski

Proponowana metoda dekompozycji matrycy MT_1 ma dwie zalety. Gwarantuje zmniejszenie liczby termów w układzie funkcji wzbudzeń pamięci MPA Moore'a do odpowiedniej wartości w równoważnym automacie Mealy'ego oraz wykorzystanie dwóch źródeł kodów stanów pozwala zmniejszyć złożoność układu bloku transformacji kodów stanów (w porównaniu do układu z jednym źródłem kodu stanów).

Ponadto dekompozycja tablicy przejść na dwie pod-tablicy umożliwia zmniejszenie złożoności bloku tworzenia termów układu funkcji wzbudzenia pamięci. Jest to związane z redukcją liczby wejściowych zmiennych określających termy dla każdej pod-tablicy w porównaniu ze znanymi metodami syntezy automatu Moore'a.

Dalszy kierunek badań jest związany z zastosowaniem proponowanej metody w przypadkach, gdy układ logiczny MPA jest realizowany przy użyciu CPLD i FPGA.

Współautorka – Olena Hebda– jest stypendystą w ramach Poddziałania 8.2.2 „Regionalne Strategie Innowacji", Działania 8.2 „Transfer wiedzy", Priorytetu VIII „Regionalne Kadry Gospodarki" Programu Operacyjnego Kapitał Ludzki współfinansowanego ze środków Europejskiego Funduszu Społecznego Unii Europejskiej i z budżetu państwa.

7. Literatura

[1] Baranov S.: Logic and system design of digital systems. Tallinn: TUT Press, 2008.
[2] Baranov S.: Logic synthesis for control automata. Norwell, MA, USA: Kluwer Academic Publishers, 1994.
[3] Baranov S., Levin I., Koren O., Karpovski M.: Designing fault tolerant FSM by nano-PLA. in Proc. of the 15th IEEE International On-Line Testing Symposium; 2009, Sembra-Lisbon, Portugal, 229-234.
[4] Barkalov A.: Synthesis of Control Units on Programmable Logic Devices. DNTU, Donetsk, 2002, in Russian.
[5] Barkalov A., Titarenko L.: Logic synthesis for FSM-based control units. Lecture notes in electrical engineering. Berlin: Springer Verlag Heidelberg, 2009, no. 53.
[6] Barkalov A., Titarenko L., Hebda O.: Optimization of Moore finite-state-machine matrix circuit. Pomiary, Automatyka, Kontrola 2011; 57(8), 939-941.
[7] DeMicheli G.: Synthesis and optimization of digital circuits. NY:McGraw-Hill, 1994.
[8] DeMicheli G., Brayton R., Sangiovanni-Vincentelli A.: Optimal state assignment for finite state machines. IEEE Trans. on CAD 1985; 4, 269-284.
[9] Novikov G.: About one approach for finite state machines research, Contr. Syst. Mach. 1974; 2, 70–75, (in Russian).
[10] Hon A.D., Wilson M.: Nanowire-based sublithographic programmable logic arrays. International Journal of Applied Mathematics and Computer Sciences 2007; 17(4), 565–575.
[11] Kania D., Czerwiski R.: Area and speed oriented synthesis of FSM for PAL-based CPLDs. Microprocessors and Microsystems 2012; 36(1), 45-61.
[12] Levin I.: Decomposition design of automata based on PLA with memory. Automatic Control and Computer Sciences 1986; 20(2), 61-68.
[13] Navabi Z.: Embedded core design with FPGA. NY: McGraw-Hill, 1997.
[14] Shrestha A., Takaota A., Tayu S., Ueno S.: On two problems of nano-PLA design. IEEE Trans. on Informatics and Systems 2011; E94(1), 25-41.
[15] Singh S., Singh R., Bhatia M.: Performance evaluation of hybrid reconfigurable computing architecture over symmetrical FPGAs. International Journal of Embedded Systems and Applications 2012; 2(3), 107-116.
[16] Skliarova I., Sklyarov V., Sudnitson A.: Design of FPGA-based circuits using hierarchical finite state machines. Tallinn:TUTPress, 2008.
[17] Sklyarov V.: Synthesis of automata with matrix VLSIs. Minsk: Nauka i Technika, 1984,(in Russian).
[18] Website of the Xilinx Corporation; 2012. http://www.xilinx.com

otrzymano / received: 16.08.2013
przyjęto do druku / accepted: 01.10.2013 *artykuł recenzowany / revised paper*

INFORMACJE

Newsletter PAK

Wydawnictwo PAK wysyła drogą e-mailową do osób zainteresowanych Newsletter PAK, w którym są zamieszczane:
- spis treści aktualnego numeru miesięcznika PAK,
- kalendarz imprez branżowych,
- ważniejsze informacje o działalności Wydawnictwa PAK.

Newsletter jest wysyłany co miesiąc do osób, które w jakikolwiek sposób współpracują z Wydawnictwem PAK (autorzy prac opublikowanych w miesięczniku PAK, recenzenci, członkowie Rady Programowej, osoby które zgłosiły chęć otrzymywania Newslettera).

Celem inicjatywy jest umocnienie w środowisku pozycji miesięcznika PAK jako ważnego i aktualnego źródła informacji naukowo-technicznej.

Do newslettera można zapisać się za pośrednictwem:
- strony internetowej: www.pak.info.pl, po dodaniu swojego adresu mailowego do subskrypcji,
- adresu mailowego: wydawnictwo@pak.info.pl, wysyłając swoje zgłoszenie.

Otrzymywanie Newslettera nie powoduje żadnych zobowiązań ze strony adresatów. W każdej chwili można zrezygnować z otrzymywania Newslettera.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK