Małgorzata KOŁOPIEŃCZYK, Alexander BARKALOV, Larysa TITARENKO
UNIWERSYTET ZIELONOGÓRSKI,
Licealna 9, 65-417 Zielona Góra

Synteza automatu Moore'a z wbudowanym blokiem pamięci w strukturach programowalnych

Dr inż. Małgorzata KOŁOPIEŃCZYK

Absolwentka Wydziału Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego (1999 r.). Obecnie adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zajmuje się następującymi zagadnieniami: projektowaniem i syntezą sterowników logicznych oraz inżynierią oprogramowania.

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

Prof. dr hab. inż. Alexander BARKALOV

Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem Instytutu 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.barkalovl@ iie.uz.zgora.pl

Streszczenie

W artykule zostanie przedstawiona metoda umożliwiająca syntezę skończonego automatu stanów typu Moore'a z wbudowanym blokiem pamięci (ang. Embedded Memory Blocks, EMB) w strukturach programowalnych typu FPGA (ang. Field Programmable Gate Array, FPGA). Zaproponowana metoda bazuje na kodowaniu pewnej wybranej części zbioru warunków logicznych przez dodatkowe zmienne. W artykule zostanie zaprezentowany przykład projektowania układu.

Słowa kluczowe: Moore FSM, FPGA, RAM, wbudowane bloki pamięci, projektowanie, układy logiczne.

EMB-based synthesis of Moore FSM

Abstract

The model of the Moore finite state machine (FSM) is very often used for representing a control unit [1]. Nowadays, two classes of programmable logic devices: complex programmable logic devices (CPLD) and field-programmable gate arrays (FPGA) are used for implementing logic circuits of FSMs [2, 3]. This paper deals with FPGA-based Moore FSMs. It is very important to use EMBs in the logic design. It leads to decreasing in both the number of interconnections and chip area occupied by an FSM logic circuit. In turn, it results in decrease in the propagation time as well as the consumed power of a circuit [9]. A lot of methods for implementing an FSM logic circuit with RAMs are known [10 – 19]. For rather complex FSMs, the method of replacement of logical conditions [20] is used. In this case, optimization efforts target hardware reduction for the multiplexer executing the replacement. In this paper we propose a method based on existence of pseudoequivalent states of the Moore FSM for solving this problem [21]. The method is based on replacement of some part of the set of logical conditions by additional variables. It results in diminishing the number of LUTs in the multiplexer used for replacement of logical conditions. To represent a control algorithm, the language of graph-schemes of algorithms [20] is used. An example of application of the proposed design method is given.

Keywords: Mealy FSM, FPGA, Embedded Memory Block, design, logic circuit.

Dr hab. 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 jako adiunkt na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: l.titarenko@iie.uz.zgora.pl

1. Wprowadzenie

Skończony automat stanów (ang. Finite State Machine, FSM) typu Moore'a bardzo często jest wykorzystywany do reprezentacji jednostki sterującej [1]. Obecnie, dwie klasy układów programowalnych są wykorzystywane do implementacji układu logicznego automatu: złożone programowalne układy cyfrowe CPLC (ang. Complex Programmable Logic Devices, CPLD) i układy FPGA [2, 3]. Wiadomym jest, że układy FPGA wykorzystywane są do implementacji raczej skomplikowanych automatów FSM [4], natomiast układy CPLD do realizacji szybkich jednostek sterujących [5].

Niniejszy artykuł dotyczył automatów Moore'a implementowanych w strukturach programowalnych typu FPGA.

Znaczna większość układów FPGA zawiera tablice LUT (ang. Look – Up Table, LUT) oraz wbudowane bloki pamięci (ang. Embedded Memory Blocks, EMB) [6, 7].

Tablica LUT może zostać skonfigurowana jako pamięć o dostępie swobodnym RAM (ang. Random Access Memory, RAM) o S wejściach adresowych i jednym wyjściu. Regułą jest że liczba wejść tablicy LUT jest raczej mała $S \leq 6$. Na wyjściu tablicy LUT umieszczono przerzutniki, które można pominąć. Zatem wyjście tablicy LUT może być kombinacyjne lub rejestrowe. Blok EMB można skonfigurować jako pamięć RAM o S_A wejściach adresowych i t_F wyjściach. Główną właściwością bloków EMB jest ich rekonfiguracja, tzn. że zarówno liczba komórek jak i liczba wyjść może zostać zmieniona w celu dopasowania do konkretnego projektu [3, 8]. Typowe konfiguracje bloków EMB są następujące 16K×1, 8K×2, 4K×4, 2K×8, 1K×16, 512×32 [6, 7]. Oznacza to, że $S_A \in \{14, 13, 12, 11,10, 9\}$ i $t_F \in \{1, 2, 3, 4, 8, 16, 32\}$. Bardzo ważną rzeczą jest zastosowanie bloków EMB w projektowaniu FSM. Prowadzi to do zmniejszenia ilości połączeń, jak również redukuje powierzchnię zajmowaną przez układ logiczny FSM, co z kolei skutkuje zmniejszeniem czasu propagacji i zmniejszeniem zużycia energii w projektowanym układzie [9]. Znanych jest wiele metod implementacji układu logicznego automatu FSM z pamięcią RAM [10-19]. W przypadku bardziej skomplikowanych automatów FSM, stosowana jest metoda kodowania warunków logicznych [20]. W takim przypadku celem optymalizacji jest zmniejszenie liczby zasobów sprzętowych w multiplekserze realizującym kodowanie. W artykule zaproponowano metodę umożliwiająca rozwiązanie tego problemu. Metoda ta bazuje na wykorzystaniu pseudo-równoważnych stanów w automacie Moore'a. Do opisu algorytmu sterowania zastosowano sieć działań (ang. Graph Scheme of Algorithm, GSA) [20].

2. Automat Moore'a

Niech dany będzie algorytm sterowania przedstawiony jako sieć działań $\Gamma=(B, E)$, składający się ze zbioru wierzchołków $B=\{b_0, b_E\} \cup B_1 \cup B_2$ oraz zbioru krawędzi E. Zbiór wierzchołków

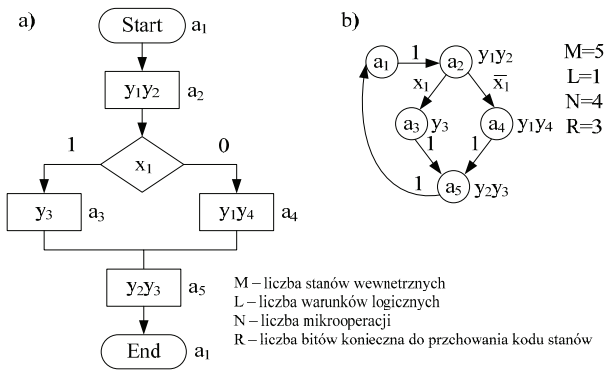
składa się z wierzchołka początkowego b_0, wierzchołka końcowego b_E, zbioru wierzchołków operacyjnych B_1 oraz zbioru wierzchołków warunkowych B_2 [20]. Wierzchołki operacyjne $b_q \in B_1$ zawierają zestaw mikrooperacji (ang. Collection of Microoperations, CMO) $Y(b_q) \subseteq Y$ gdzie gdzie $Y=\{y_1,...,y_N\}$ jest zbiorem mikrooperacji (wyjść automatu FSM). Wierzchołek warunkowy $b_t \in B_2$ zawiera jeden element $x_l \in X$, gdzie $X=\{x_1,...,x_L\}$ jest zbiorem warunków logicznych (wejść automatu FSM).

Na potrzeby syntezy skończonego automat stanów Moore'a, oznaczmy początkową sieć działań stanami automatu [20]. Oznaczmy wierzchołek początkowy b_0 oraz wierzchołek końcowy b_E przez stan początkowy a_1, natomiast każdy wierzchołek $b_q \in B_2$ oznaczmy przez unikalny stan. Pozwala to na utworzenie zbioru stanów wewnętrznych $A=\{a_1,...a_M\}$ automatu. Aby zaprojektować układ logiczny automatu FSM należy, w następnym etapie zakodować poszczególne stany [20]. Zakodujmy każdy stan $a_m \in A$ jako kod $K(a_m)$ na R bitach. Zakodujmy stany przez zmienne, których liczba jest równa:

$$R = \lceil \log_2 M \rceil \tag{1}$$

Do kodowania wykorzystano zmienne $T_\Gamma \in T$, gdzie T jest zborem zmiennych i $|T|=R$.

Rozpatrzmy oznakowaną sieć działań Γ_1 (rys. 1, a).



Rys. 1. Początkowa sieć działań Γ_1 (a) i graf przejść (b)
Fig. 1. Initial GSA Γ_1 (a) and state transition graph (b)

Oznakowana sieć działań Γ_1 określa graf przejść pokazany na rysunku 1, b. Automat Moore'a S_1 reprezentowany przez graf przejść ma następujące parametry: $M=5$, $L=1$, $N=4$, $R=3$. Zakodujmy stany $a_m \in A$ automatu S_1 w następujący sposób: $K(a_1)=000,...,K(a_5)=100$. Na podstawie grafu przejść (rys. 1, b) oraz zaproponowanego kodowania skonstruujmy strukturalną tabelę przejść (ang. structural table, ST) automatu Moore'a S_1 (tab. 1).

Tab. 1. Strukturalna tabela przejść automatu Moore'a S_1
Tab. 1. Structure table of Moore FSM S_1

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Φ_h	h
a_1	000	a_2	001	1	D_3	1
$a_2\,(y_1,y_2)$	001	a_3	010	x_1	D_2	2
		a_4	011	$/x_1$	D_2D_3	3
$a_3\,(y_3)$	010	a_5	100	1	D_1	4
$a_4\,(y_1,y_4)$	011	a_6	100	1	D_1	5
$a_5\,(y_2,y_3)$	100	a_1	000	1	–	6

Tabela ta ma $H=6$ wierszy. Każdy z nich odpowiada jednej krawędzi grafu przejść. Kolumna X_h zawiera sygnały wejściowe, które powodują przejście $\langle a_m,a_s \rangle$ (w grafie przejść zapisujemy je powyżej krawędzi). Kolumna Φ_h zawiera wejściowe funkcje wzbudzeń $D_\Gamma \in \Phi$, które są równe 1 i powodują przełączenie pamięci automatu z kodu $K(a_m)$ na kod $K(a_s)$. Zbiór wejściowych funkcji wzbudzeń Φ zawiera R elementów. Regułą jest, że do przechowywania kodu stanów wykorzystujemy rejestr z wejściem typu D [9]. Z tego względu funkcje wzbudzeń z tabeli 1 są oznaczone jako $D_1 - D_3$. Kolumna a_m zawiera bieżący stan automatu jak również kolekcję mikrooperacji $Y(a_m) \subseteq Y$ generowanych w tym stanie. Kolumna a_s zawiera stan następny automatu. Związek pomiędzy grafem przejść a strukturalną tabelą przejść jest oczywisty.

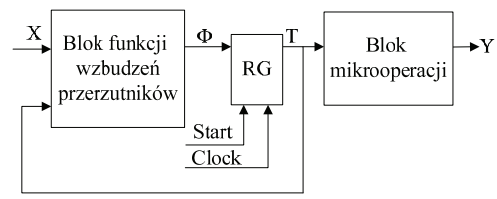
Na podstawie strukturalnej tabeli przejeść wyprowadzany jest następujący system funkcji opisujących układ logiczny automatu:

$$\Phi = \Phi(T, X), \tag{2}$$

$$Y = Y(T). \tag{3}$$

Na przykład, na podstawie tabeli 1 możemy wyprowadzić następujące funkcje: $D_3 = \overline{T_1}\overline{T_2}\overline{T_3} \vee \overline{T_1}\overline{T_2}T_3\overline{x_1}$, $y_1 = A_2 \vee A_4 = \overline{T_1}\overline{T_3}$.

Systemy (2) – (3) określają strukturę automatu Moore'a U_1 (rys. 2).



Rys. 2. Struktura automatu Moore'a U_1
Fig. 2. Structure diagram of Moore FSM U_1

W układzie U_1 blok funkcji wzbudzeń przerzutników (BFWP) implementuje funkcję (2), natomiast blok mikrooperacji (BMO) funkcję (3). Rejestr RG przechowuje kody stanu $K(a_m)$. Sygnał *„Start"* służy do załadowania stanu początkowego $a_1 \in A$ do rejestru RG. Sygnał *„Clock"* zmienia zawartość rejestru RG.

W artykule zaproponowano metodę umożliwiającą syntezę skończonego automatu stanów z wbudowanym blokiem pamięci w strukturach FPGA. Opiera się ona na trzech głównych zagadnieniach: po pierwsze, tylko część warunków logicznych będzie zakodowana; po drugie, przekształcenie kodu pseudo-równoważnych stanów w kody ich klas będzie zrealizowane; po trzecie zastosowano specjalne kodowanie klas pseudo-równoważnych stanów.

3. Idea proponowanej metody

Stany a_m, $a_s \in A$ nazywane są stanami pseudo-równoważnymi (ang. pseudoequivalent states, PES), jeżeli wyjścia wierzchołków operacyjnych oznaczonych przez te stany są połączone z tym samym wejściem (z wejściem tego samego wierzchołka). Na przykład, stany a_3 i a_4 są stanami pseudo-równoważnymi (rys. 1, a). Stany te tworzą klasy pseudo-równoważnych stanów.

Niech $\Pi_A=\{B_L,...,B_1\}$ będzie zbiorem klas pseudo-równoważnych stanów powstałym w wyniku podziału zbioru A.

W przypadku automatu Moore'a S_1, $\Pi_A=\{B_1,B_2,B_3,B_4\}$, gdzie $B_1=\{a_1\}$, $B_2=\{a_2\}$, $B_3=\{a_3,a_4\}$, $B_4=\{a_5\}$. Niech Π_A' oznacza zbiór klas $B_i \in \Pi_A$ nie zawierających krawędzi $\langle b_q,b_E \rangle \in E$, w którym wierzchołek $b_q \in B_1$ oznaczono stanem $a_m \in B_i$.

W przypadku automatu S_1, zbiór $\Pi_A'=\{B_1,B_2,B_3\}$. Niech I_0 będzie liczbą klas w zbiorze Π_A'.

Zakodujmy każdą klasę $B_i \in \Pi_A'$ kodem $K(B_i)$ używając R_B zmiennych, gdzie

$$R_B = \lceil \log_2 I_0 \rceil. \tag{4}$$

Do kodowania zastosujmy zmienne $\tau_\Gamma \in \tau$, gdzie $|\tau|=R_B$.

Przedstawmy zbiór warunków logicznych X jako sumę zbiorów $X^1 \cup X^2$ gdzie $X^1 \cap X^2 = \varnothing$, $|X^1|=L^1$ i $|X^2|=L^2$. Dokonajmy kodowania warunków logicznych $x_l \in X^2$ przez dodatkowe zmienne $P_g \in P$ gdzie $|P|=G$.

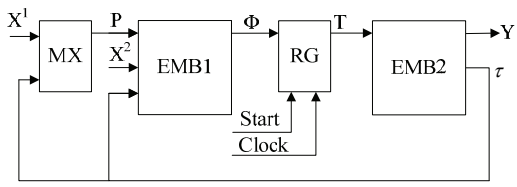
Niech będzie spełniony warunek

$$2^{G+L^1+R_B} \times R \le V_0 , \tag{5}$$

$$2^R \cdot (R_B + N) \le V_0 . \tag{6}$$

Symbol V_0 oznacza liczbę komórek bloku EMB jeżeli liczba wyjść $t_F=1$.

Dla takiego przypadku zaproponowano model automat Moore'a U_2 , którego strukturę przedstawiono na rysunku 3.



Rys. 3. Struktura automatu Moore'a U_2
Fig. 3. Structure diagram of Moore FSM U_2

W automacie U_2 multiplekser MX koduje warunki logiczne $x_l \in X^1$ oraz generuje funkcje

$$P = P(\tau, X^1). \tag{7}$$

Blok EMB1 implementuje funkcje $D_\Gamma \in \Phi$:

$$\Phi = \Phi(\tau, P, X^2). \tag{8}$$

Blok EMB2 implementuje funkcje (3) i funkcje

$$\tau = \tau(T). \tag{9}$$

Układ logiczny automatu FSM U_2 wymaga zastosowania 2 bloków pamięci RAM.

W artykule zaproponowano metodę projektowania automatu U_2. Metoda ta składa się z następujących etapów:

1. Utworzenie oznaczonej sieci działań Γ.
2. Utworzenie zbiorów A, Π_A i Π_A'.
3. Zakodowanie stanów binarnym kodem o minimalnej liczbie bitów.
4. Utworzenie podziału zbioru X i znalezienie zbiorów X^1 i X^2.
5. Utworzenie tabeli kodowania elementów podzbioru warunków logicznych.
6. Wykonanie optymalnego kodowania klas $B_i \in \Pi_A'$.
7. Utworzenie zredukowanej strukturalnej tabeli.
8. Utworzenie przekształconej zredukowanej strukturalnej tabeli.
9. Utworzenie tabeli mikrooperacji.
10. Implementacja automatu.

Metoda projektowania układu FSM $U_2(\Gamma_2)$ zostanie omówiona na przykładzie sieci działań Γ_2 przedstawionej na rysunku 4.

4. Przykład zastosowania proponowanej metody

Zastosujmy układ FPGA o następującej charakterystyce: $S=4$ (liczba wejść tablicy LUT), $V_0=512$. Niech blok EMB ma następujące konfiguracje: $2^9\times1$, $2^8\times2$, $2^7\times4$, $2^6\times8$ i $2^5\times16$. Dlatego: $S_A=\{5,...,9\}$, $t_F=\{1,2,4,8,16\}$.

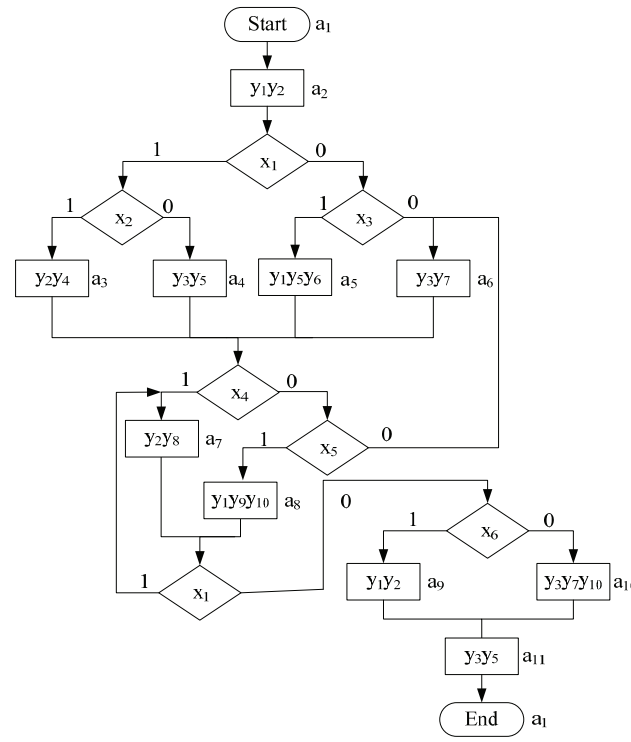
Na podstawie sieci działań Γ_2 znaleziono następujące parametry: $M=11$, $L=6$, $N=10$ i $R=4$.

Przejście ze stanu a_2 zależy od 3 warunków logicznych dlatego G=3.

Z analizy sieci działań Γ_2 wynika, że zbiór pseudorównoważnych stanów zawiera $\Pi_A=\{B_1,...,B_6\}$ gdzie $B_1=\{a_1\}$, $B_2=\{a_2\}$, $B_3=\{a_3,..,a_6\}$, $B_4=\{a_7,a_8\}$, $B_5=\{a_9,a_{10}\}$ i $B_6=\{a_{11}\}$. Klasa $B_6 \notin \Pi_A'$, dlatego $\Pi_A'=\{B_1,...,B_5\}$.

Zwróćmy uwagę, że kody stanów nie mają wpływu na liczbę tablic LUT w multiplekserze MX dla układu FSM U_2. W związku z tym, zakodujmy poszczególne stany w trywialny sposób: $K(a_1)$=0000, $K(a_2)=0001,...,K(a_{11})=1010$.

Ponieważ R=4, konfiguracja bloku EMB $2^7\times4$ powinna zostać wybrana. Oznacza to, że $S_A=7$. Zbiór Π_A' zawiera $I_0=5$ elementów.



Rys. 4. Początkowa sieć działań Γ_2
Fig. 4. Initial GSA Γ_2

Z równania (4) wynika, że $R_B=3$. Z równania (5) wynika, że $G+L^1+R_B=7$, dlatego $G+L^1=4$.

Utwórzmy następujące zbiory: $X^2=\{x_2,x_3\}$ i $X^1=\{x_1,x_4,x_5,x_6\}$. Z analizy sieci działań wynika, że jest wystarczająca liczba zmiennych $G=2$ do kodowania warunków logicznych $x_l \in X^1$. Stwórzmy tabelę kodowania warunków logicznych dla automatu $U_2(\Gamma_2)$ (tabela 2).

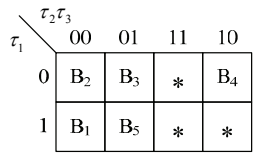
Tab. 2. Tabela kodowania warunków logicznych dla układu $U_2(\Gamma_2)$
Tab. 2. Table of replacement of logical conditions for Moore FSM $U_2(\Gamma_2)$

P	B_1	B_2	B_3	B_4	B_5
P_1	-	x_1	x_4	x_1	-
P_2	-	-	x_5	x_6	-

W wierszach tej tabeli umieszczono zmienne $P_g \in P$, natomiast w kolumnach klasy $B_i \in \Pi_A'$. Jeżeli warunek logiczny $x_l \in X^1$ jest kodowany przez zmienne $P_g \in P$ dla klasy $B_i \in \Pi_A'$, wówczas symbol x_l wpisywany jest w miejscu przecięcia wiersza P_g i kolumny B_i.

W celu zmniejszenia liczby tablic LUT w układzie multipleksera MX, umieśćmy klasy $B_i \in \Pi_A'$ z warunkowym przejściem w możliwie najmniejszej liczbie uogólnionych interwałów R_B wymiarowej Boolowskiej przestrzeni.

Nazwijmy takie kodowanie, kodowaniem optymalnym. Jedno z możliwych rozwiązań przedstawiono na rysunku 5.



Rys. 5. Optymalne kodowanie klas
Fig. 5. Optimal codes of classes

Na podstawie tabeli 2 i rysunku 5, możemy dla funkcji (7), znaleźć następujące równania:

$$P_1 = \left[B_2 \vee B_4\right]x_1 \vee B_3 x_4 = \overline{\tau}_3 x_1 \vee \tau_3 x_4 \,; \tag{10}$$

$$P_2 = B_3 x_5 \vee B_4 x_6 = \overline{\tau}_2 \tau_3 x_5 \vee \tau_2 \overline{\tau}_3 x_6 \,.$$

Każda z funkcji (10) potrzebuje do implementacji tylko jednej tablicy LUT z $S=4$. W związku z tym blok multipleksera MX zawiera tylko dwie tablice LUT i jest blokiem jednopoziomowym.

W celu utworzenia zredukowanej strukturalnej tabeli przejść sformułujmy system uogólnionych formuł przejść [9]. W systemie tym, zgodnie z tabelą 2, zastosujmy zmienne $P_g \in P$. Dla automatu Moore'a $U_2(\Gamma_2)$ utworzono następujący system formuł:

$$B_1 \to a_2 \,; \tag{11}$$

$$B_2 \to P_1 x_2 a_3 \vee P_1 \overline{x_2} a_4 \vee \overline{P_1} x_3 a_5 \vee \overline{P_1}\,\overline{x_3} a_6 \,;$$

$$B_3 \to P_1 a_7 \vee \overline{P_1} P_2 a_8 \vee \overline{P_1}\,\overline{P_2} a_6 \,;$$

$$B_4 \to P_1 a_7 \vee \overline{P_1} P_2 a_9 \vee \overline{P_1}\,\overline{P_2} a_{10} \,;$$

$$B_5 \to a_{11} \,.$$

Każda ze składowych systemu (11) odpowiada jednemu wierszowi zredukowanej strukturalnej tabeli. Tabela ta składa się z następujących kolumn: B_i, $K(B_i)$, a_s, $K(a_s)$, P_h, X_h, Φ_h, h. W przypadku automatu $U_2(\Gamma_2)$, tabela ta zawiera $H_2(\Gamma_2)=12$ wierszy. Fragment omawianej tabeli przedstawiono w tabeli 3.

Tab. 3. Fragment zredukowanej strukturalnej tabeli dla układu Moore'a $U_2(\Gamma_2)$
Tab. 3. Part of the reduced ST for Moore FSM $U_2(\Gamma_2)$

b_i	$K(B_i)$	a_s	$K(a_s)$	P_h	X_h	Φ_h	h
B_1	100	a_2	0001	1	1	D_4	1
B_2	001	a_3	0010	P_1	x_2	D_3	2
		a_4	0011	P_1	$/x_2$	D_3D_4	3
		a_5	0100	$/P_1$	x_3	D_2	4
		a_6	0101	$/P_1$	$/x_3$	D_2D_4	5

Związek pomiędzy systemem (11) i tabelą 3 jest oczywisty.

Przekształcona zredukowana strukturalna tabela ST reprezentuje blok EMB1. Tabela ta składa się z następujących kolumn: $K(B_i)$, P, X^2, Φ, v. Przejście pomiędzy klasami $B_i \in \Pi_A{}'$ jest reprezentowane przez V_G wierszy tej tabeli, gdzie

$$V_G = 2^{G+L^1} \,. \tag{12}$$

W przypadku automatu $U_2(\Gamma_2)$, $V_G=16$. Fragment przekształconej zredukowanej tabel ST zaprezentowano w tabeli 4.

Tab. 4. Fragment przekształconej zredukowanej tabeli ST dla układu $U_2(\Gamma_2)$
Tab. 4. Part of the transformed reduced ST for Moore FSM $U_2(\Gamma_2)$

$K(B_i)$	P	X^2	Φ	v
$\tau_1\tau_2\tau_3$	P_1P_2	x_2x_3	$D_1D_2D_3D_4$	
000	00	00	0101	1
000	00	01	0100	2
000	00	10	0101	3
000	00	11	0100	4
000	01	00	0101	5

Tabela mikrooperacji składa się z następujących kolumn: $K(a_m)$, $Y(a_m)$, m. W przypadku automatu $U_2(\Gamma_2)$, tabela mikrooperacji zawiera $m=10$ wierszy. Fragment tabeli mikrooperacji zaprezentowano w tabeli 5.

Tab. 5. Fragment tabeli mikrooperacji dla układu $U_2(\Gamma_2)$
Tab. 5. Part of the microoperation table for Moore FSM $U_2(\Gamma_2)$

$K(a_m)$	$Y(a_m)$	m
$T_1T_2T_3T_4$	$y_1y_2y_3y_4y_5y_6y_7y_8y_9y_{10}$	
0000	0 0 0 0 0 0 0 0 0 0	1
0001	1 0 1 0 0 0 0 0 0 0	2
0010	0 1 0 1 0 0 0 0 0 0	3
0011	0 0 1 0 1 0 0 0 0 0	4
0100	1 0 0 0 1 1 0 0 0 0	5

Implementacja układu logicznego automatu powinna zostać wykonana z wykorzystaniem przemysłowych narzędzi takich jak np. WebPack. Niestety wykracza to poza zakres tego artykułu.

5. Podsumowanie

W artykule zaproponowano metodę umożliwiającą zmniejszenie zużycia zasobów sprzętowych w automatach Moore'a, które będą implementowane z wykorzystaniem wbudowanych bloków pamięci. Zaproponowana metoda polega na kodowaniu tylko pewnej wybranej części zbioru warunków logicznych. Zastosowanie metody umożliwiło zmniejszenie zużycia tablic LUT w układzie logicznym multipleksera w porównaniu do przypadku, w którym kodowane są wszystkie warunki logiczne.

Dalsze kierunki prac autorów obejmują: opracowanie efektywnej metody podziału zbioru warunków logicznych - metoda ta powinna umożliwić zastosowanie zaproponowanego automatu Moore'a z minimalnym rozmiarem multipleksera; opracowane narzędzi umożliwiających projektowanie zaproponowanych automatów; narzędzia te powinny współpracować z istniejącym obecnie oprogramowaniem CAD np. z programem WebPack firmy Xilinx.

6. Literatura

[1] Minns P, Elliott I.: FSM-based digital design using Verilog HDL. – Chichester: John Wiley & Sons, 2008.
[2] Kania D., Czerwiński R.: Area and sped oriented synthesis of FSMs for PAL-based CPLDs//Microprocessors and Microsystems. – 2012, V.36, № 1. – pp. 45 – 61.
[3] Grout I.: Digital Systems Design with FPGAs and CPLDs. – Amsterdam: Elsevier, 2008.
[4] Kaviani A., Brown S.: Technology mapping issues for an FPGA with look up tables and PLA like blocks. – In: Proceedings of the 2000 ACM/SIGDA 8th International Symposium on Field Programmable Gate Arrays, 2000. – pp. 60– 66
[5] Kim J., Byun S., Kim H.: Development of technology mapping algorithm for largr complex PLDs. – In: Proceedings pf Design Automation Conference DAC'98, 1998 – pp. 698–703.
[6] www.altera.com
[7] www. xilinx.com
[8] Maxfeld C. FPGA World Class Design. – Amsterdam. Elsevier, 2009.

[9] Sutteer G., Todorowich E., Lopez-Buedo S., Boemo E. Lower-power FSMs in FPGA: Encoding Alternatives. – Lecture Notes in Computer Science 2451. – Berlin: Springer, 2002. – pp. 363 – 370.
[10] Borowik G.: Finite State Machines Synthesis for FPGA Structures with Embedded Memory Blocks, PhD Thesis. –Warsaw: WUT Press, 2007 -168pp.
[11] Cong J., Yan K.: Synthesis for FPGAs with embedded memory blocks. – In.: Proceedings of the 2000 ACM / SIGDA 8th International Symposwium on FPGAs. – 2000, pp. 75 – 82.
[12] Rawski M., Selvaraj H., Łuba T.: An application of functional decomposition in ROM-based FSM implementation in FPGA devices // Journal of System Architecture. – 2005, V. 51, № 6 – 7. – pp. 424 – 434.
[13] Rawski M., Tomaszewicz P., Borowski G., Łuba T.: Logic Synthesis Method of Digital Circuits Designed for Implementation with Embedded Memory Blocks on FPGAs. – In: M. Adamski, A. Barkalov, M. Węgrzyn (Editors). Design of Digital Systems and Devices. LNEE 79. – Berlin: Springer, 2011. – pp. 121 – 144.
[14] Sklyarov V.: Hierarchical Finite-State Machines and their use for digital control. // IEEE Transactions on VLSI Systems. – 1999, V.7. №2. – pp. 222 – 228.
[15] Sklyarov V.: Reconfigurable models of finite state machines and their implementation in FPGAs. – Journal of Systems Architecture V. 47, 2002 Issue 14-15, 2002 -pp. 1043 – 1064.
[16] Sklyarova I., Sklyarov V., Sudnitson A.: Design of FPGA-based circuits using Hierarchical Finite State Machines – Tallin: TUT Press, 2012.
[17] Sklyarov V.: Synthesis and Implementation of RAM-Based Finite State Machines in FPGAs –In: Proceedings of Conference on Field Programmmable Logic–Villach, 2000. –pp. 718–728.
[18] Tiwari A., Tomko K.: Saving power by mapping finite state machines into embedded memory blocks in FPGAs. – In: Proceedings of Design Automation and Test in Europe. – 2004, V. 2. – pp. 916 – 921.
[19] Garcia-Vargas I., Senhadji-Navarro R., Civit-Balcells A., Guerra-Gutierrezz P.: ROM-based finite state machine implementation in low cost FPGAs. – In: IEEE International Symposium on Industrial Electronics. – Vigo, 2007. – pp. 2342 – 2347.
[20] Baranov S.: Logic Synthesis for Control Automata. – Boston: Kluwer Academic Publishers, 1994.

otrzymano / received: 15.05.2013
przyjęto do druku / accepted: 03.07.2013 *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.