Arkadiusz BUKOWIEC, Alexander BARKALOV
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Synteza automatów stanów typu Mealy'ego z liniowym przekształceniem sieci działań i adresowaniem mikroinstrukcji

Mgr inż. Arkadiusz BUKOWIEC

Ukończył studia inżynierskie (2001) o specjalności inżynieria komputerowa na Politechnice Zielonogórskiej a następnie studia magisterskie (2004) o tej samej specjalności na Uniwersytecie Zielonogórskim. Od roku 2004 pracuje jako asystent oraz jest studentem studiów doktoranckich na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego. W roku 2006 odbył jeden semestr studiów doktoranckich na Universidade Nova de Lisboa w ramach projektu Sokrates-Erasmus.

e-mail: a.bukowiec@iie.uz.zgora.pl

Prof. dr hab. inż. Alexander BARKALOV

Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywno 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

Streszczenie

W referacie została przedstawiona metoda zmniejszenia wymaganych zasobów sprzętowych w programowalnym układzie matrycowym do implementacji skończonego automatu stanów (FSM) z wyjściami typu Mealy'ego. Zaproponowana metoda oparta jest na liniowym przekształceniu początkowej sieci działań. W rezultacie takiego przekształcenia wszystkie mikrooperacje w przekształconej sieci działań stają się kompatybilne. Umożliwia to zakodowanie każdej mikrooperacji za pomocą binarnego kodu na możliwie minimalnej liczbie bitów. W sytuacji takiej do implementacji systemu mikrooperacji potrzebny jest tylko jeden dekoder. Dodatkowo w celu zachowania tej samej liczby stanów zastosowano adresowanie mikroinstrukcji. Adres mikroinstrukcji generowany jest przez układ kombinacyjny automatu, następnie dekoder generuje mikrooperację na podstawie adresu mikroinstrukcji i kodu mikrooperacji, generowanego przez licznik mikrooperacji. Metoda ta zapewnia zmniejszenie liczby wyjść części kombinacyjnej automatu Mealy'ego w porównaniu z tą samą charakterystyką automatu Mealy'ego z kodowaniem klas kompatybilnych mikrooperacji.

Słowa kluczowe: Automat stanów, jednostka sterująca, układy programowalne.

Synthesis of Mealy FSMs with Verticalization of Flow Chart and Addressing of Microinstructions

Abstract

The method of decreasing of logic amount in programmable device implementing the logic circuit of finite state machine (FSM) is proposed. Method is based on verticalization of flow chart. As a result of verticalization all microoperations are compatible ones. It permits to encode each microoperation by code with minimal possible number of bits. In this case only one decoder is used for implementation of the microoperations system. Additionally, there is used a register for microinstruction addresses and a counter for generation of code of microoperation. This manipulation allows to secure the same number of states like for algorithm before verticalization. This method permits to minimize number of outputs of the combinational part of Mealy FSM in comparison with the same characteristic of Mealy FSM with encoding of fields of compatible microoperations.

Keywords: FSM, Control unit, FPD.

1. Wstęp

Popularnym sposobem projektowania jednostek sterujących jest zastosowanie skończonych automatów stanów z wyjściami typu Mealy'ego [3]. Układy FPGA są bardzo często stosowane do implementacji takiego automatu [8]. Główną cechą układów FPGA jest występowanie w nich dużej liczby tablic LUT, które mogą zrealizować dowolną funkcje logiczną [7, 8]. Ograniczeniem jednak jest stosunkowo mała liczba wejść (typowo 4) jaką posiadają tablice LUT, z drugiej strony funkcje logiczne realizowane przez kombinacyjny układ automatu posiadają znacznie więcej argumentów. Rozbieżność ta prowadzi do konieczności zastosowania blokowej dekompozycji funkcji boolowskich [7]. Negatywnym wynikiem takiej dekompozycji jest zwiększenie liczby poziomów w układzie logicznym realizującym automat stanów, co może prowadzić do zmniejszenia wydajności takiego systemu.

W prezentowanym referacie przedstawiona jest metoda umożliwiająca zmniejszenie liczby funkcji logicznych realizowanych przez kombinacyjną część automatu. Prowadzi to do zmniejszenia wymaganej liczby tablic LUT w porównaniu ze znanymi metodami projektowania [1, 5].

Głównym założeniem prezentowanej metody jest przeprowadzenie liniowego przekształcenia sieci działań opisującej algorytm sterowania [2] oraz zastosowaniu rejestru dla adresu mikroinstrukcji oraz licznika do generowania kolejnych kodów mikrooperacji w ramach danej mikroinstrukcji.

2. Kodowanie klas μoperacji

Jednym z popularniejszych sposobów reprezentacji algorytmu sterowania jest sieć działań [1]. Niech bloki operacyjne należące do sieci działań Γ tworzą zbiór bloków operacyjnych $O(\Gamma) = \{O_1,...,O_K\}$. Każdy blok operacyjny $O_k \in O(\Gamma)$ zawiera mikro(μ)instrukcje $Y(O_k) \subseteq Y$, gdzie $Y = \{y_1, ..., y_N\}$ jest zbiorem mikro(μ)operacji. Każdy blok decyzyjny zawiera jeden element $x_l \in X$, gdzie $X = \{x_1, ..., x_L\}$ jest zbiorem warunków logicznych. Niech taka sieć działań będzie oznaczona stanami automatu z wyjściami typu Mealy'ego [5] i niech tworzą one zbiór stanów $A = \{a_1, ..., a_M\}$. Każdy stan $a_m \in A$ można zakodować na $R = \lceil \log_2 M \rceil$ bitach za pomocą binarnego kodu $K(a_m)$. Do reprezentacji tego kodu wykorzystane zostaną zmienne $Q_r \in Q = \{Q_1, ..., Q_R\}$. Działanie układu logicznego skończonego automatu stanów z wyjściami typu Mealy'ego może zostać opisane przez tablicę przejść-wyjść [3, 7] z następującymi kolumnami: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h (tab. 1), gdzie $a_m \in A$ jest początkowym stanem automatu; $a_s \in A$ jest następnym stanem automatu; $K(a_s)$ jest kodem stanu a_s; X_h jest koniunkcją afirmacji lub negacji pewnych zmiennych ze zbioru wejściowych warunków logicznych wymuszając przejście $\langle a_m, a_s \rangle$; $Y_h \subseteq Y$ jest μinstrukcją formowaną podczas przejęcia $\langle a_m, a_s \rangle$; Φ_h jest zbiorem funkcji wzbudzeń, które są równe 1 w celu przełączenia pamięć automatu z kodu $K(a_m)$ na kod $K(a_s)$, $\Phi_h \subseteq \Phi = \{D_1, \ldots, D_R\}$; h jest numerem przejścia automatu ($h = 1, ..., H$). Na tej podstawie wyprowadzany jest system wzbudzeń przerzutników i system μoperacji:

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

$$Y = Y(Q, X). \tag{2}$$

Tab. 1. Fragment tablicy przejść-wyjść automatu
Tab. 1. A part of DST table of Mealy FSM

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_h	Φ_h	h
a_2	0 1 0	a_3	0 1 1	1	$y_1\, y_2$	$D_2\, D_3$	1
a_3	0 1 1	a_4	1 0 0	$\neg x_1$	y_3	D_1	2
		a_5	1 0 1	x_1	$y_1\, y_3\, y_4$	$D_1\, D_3$	3

Systemy te realizowane są przez kombinacyjną część automatu, natomiast aktualny stan zapamiętywany jest w zbiorze rejestrów. W sytuacji gdy do implementacji wykorzystuje się układy FPGA zbudowany jest on z przerzutników typu D [7, 8]. Układ taki posiada jednopoziomową strukturą [1], jednak ze względu na ograniczenia tablic LUT dekompozycja funkcji logicznych wymusza wielopoziomową realizację układu kombinacyjnego.

Jednym ze sposobów optymalizacji zasobów sprzętowych automatu jest zmniejszenie liczby funkcji realizowanych przez część kombinacyjną. Jedną z umożliwiających to metod jest kodowanie klas kompatybilnych μoperacji [5]. μoperacje $y_n, y_m \in Y$ są kompatybilne, jeżeli

$$\forall_k \left(y_n \in Y(O_k) \rightarrow y_m \notin Y(O_k)\right) \ (k = 1,\ldots,K). \tag{3}$$

Niech zbiór $\Pi_Y = \{Y^1, ..., Y^I\}$ będzie podziałem zbioru Y na klasy kompatybilnych μoperacji. Niech każda μoperacja $y_n \in Y^i$ zostanie zakodowana za pomocą kodu $K(y_n)$ na r_i bitach, gdzie $r_i = \lceil \log_2(|Y^i|+1) \rceil$ ($i = 1, ..., I$). Tak więc do zakodowania wszystkich μoperacji wymagane będzie

$$R_1 = \sum_{i=1}^{I} r_i \tag{4}$$

zmiennych $z_r \in Z = \{z_1,\ldots,z_{R_1}\}$. W takim przypadku automat Mealy'ego zostanie zaimplementowany w dwupoziomowej strukturze PD (rys. 1), gdzie dekoder D_i dekoduje μoperacje należące do klasy $Y^i \in \Pi_Y$. W sytuacji takiej układ P realizuje funkcje (1) i

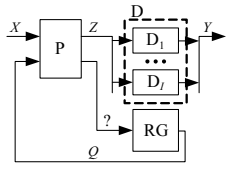
$$Z = Z(Q, X). \tag{5}$$

a dekoder D, zbudowany z I dekoderów i realizuje funkcję

$$Y = Y(Z). \tag{6}$$

Prowadzi to do zmniejszenia liczby funkcji zależnych od warunków logicznych i zmiennych wewnętrznych realizowanych przez kombinacyjną część automatu.

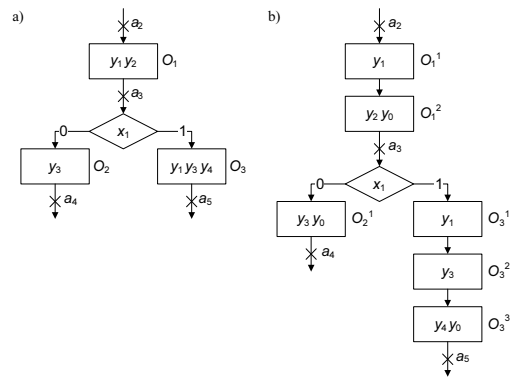
Wadą takiego rozwiązania jest wciąż relatywnie duża liczba funkcji Z oraz konieczność implementacji I dekoderów. Najlepszym rozwiązaniem problemu zmniejszenia liczby wyjść układu P bez zmniejszania wydajności jednostki sterującej jest sytuacja, gdy wszystkie μoperacje realizowane przez automat są wzajemnie do siebie kompatybilne. Przypadek taki odzwierciedlony jest przez liniową sieć działań [2, 5] gdzie każdy blok operacyjny zawiera tylko jedną μoperację. Problemem jednak jest to, że takie sieci działań bardzo rzadko występują w rzeczywistości. W celu uzyskania takiej sieci działań można zastosować liniowe przekształcenie sieci działań [2, 5]. W celu lepszego zagospodarowania zasobów sprzętowych zaproponowana zostanie struktura układu cyfrowego implementującego układ automatu Mealy'ego w oparciu o przekształconą sieć działań oraz odpowiednia metoda syntezy.



Rys. 1. Schemat blokowy automatu Mealy'ego o strukturze PD
Fig. 1. Block diagram of the Mealy PD FSM

3. Liniowe przekształcenie sieci działań

Niech każdy blok operacyjny sieci działań Γ zawiera $N_k = |Y(O_k)|$ μoperacji. Przekształćmy zatem każdy blok operacyjny $O_k \in O(\Gamma)$ w sekwencje bloków $\beta_k = \langle O_k^1,\ldots,O_k^{N_k}\rangle$, w której każdy blok O_k^j zawiera jedną unikalną μoperację $y_n = \mathrm{pr}_j Y(O_k)$. Dodatkowo w celu identyfikacji końca sekwencji w każdym ostatnim bloku $O_k^{N_k}$ dla każdej sekwencji β_k wstawmy specjalny sygnał y_0 (rys. 2). Jeżeli wejście do bloku operacyjnego O_k w początkowej sieci działań było oznaczone stanem $a_m \in A$ to wejście bloku operacyjnego O_k^1 (początek sekwencji) w wynikowej sieci działań oznaczone jest tym samym stanem [6]. Zabieg taki powoduje, że liczba stanów w sieci działań przed i po przekształceniu jest taka sama, co stanowi zysk w porównaniu z metodą, w której przekształcona sieć działań ponownie jest znakowana stanami [4].



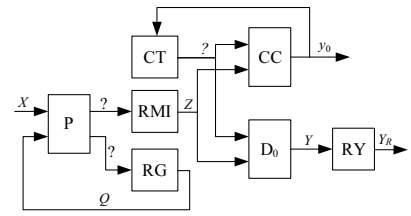
Rys. 2. Fragment sieci działań przed (a) i po (b) liniowym przekształceniu
Fig. 2. The subgraph of marked flow-chart before (a) and after (b) verticalization

4. Układ automatu dla liniowo przekształconej sieci działań z adresowaniem μinstrukcji

Zakodujmy μinstrukcje $Y_t \subseteq Y$ binarnym kodem $K(Y_t)$ na $R_2 = \lceil \log_2 T \rceil$ bitach, gdzie T jest liczbą wszystkich różnych μinstrukcji. Do reprezentacji tego kodu wykorzystane zostaną zmienne $z_r \in Z = \{z_1,\ldots,z_{R_2}\}$. Niech każda μinstrukcja $Y_t \subseteq Y$ zawiera $M_t = |Y_t|$ μoperacji i $M_0 = \max(M_1, \ldots, M_T)$. Wprowadźmy symbol y_n^t oznaczający, że $y_n \in Y_t$. Zakodujmy każdą μoperację $y_n^t = \mathrm{pr}_j Y(O_k)$ ($j = 1, \ldots, N_k$; $k = 1, \ldots, K$) binarnym kodem $K(y_n^t)$ na $R_3 = \lceil \log_2 M_0 \rceil$ bitach. Do reprezentacji tego kodu wykorzystane zostaną zmienne $\tau_r \in \tau = \{\tau_1,\ldots,\tau_{R_3}\}$. W takiej sytuacji kod $C(y_n)$ μoperacji y_n reprezentowany jest jako

$$C(y_n) = K(Y_t) * K(y_n^t). \tag{7}$$

Oczywiste jest, że każda μoperacji y_n może posiadać do T różnych kodów $C(y_n)$. W sytuacji takiej układ automatu może zostać zrealizowany w strukturze PD_{VRC} (rys. 3) gdzie rejestr RMI wykorzystany jest do zapamiętania kodu μinstrukcji $K(Y_t)$ a licznik CT wykorzystany jest do generowania kodów kolejnych μoperacji $K(y_n^t)$.



Rys. 3. Schemat blokowy automatu Mealy'ego o strukturze PD_{VRC}
Fig. 3. Block diagram of the Mealy PD_{VRC} FSM

Układ P realizuje funkcje (1) oraz system wzbudzeń rejestru RMI

$$\Psi = \Psi(Q, X), \tag{8}$$

układ CC realizuje funkcję

$$y_0 = y_0(Z, \tau), \tag{9}$$

która służy do zerowania licznika CT. µoperacje y_n formowane są przez dekoder D_0

$$Y = Y(Z, \tau). \quad (10)$$

i zapisywane w rejestrze wyjściowym RY. Rejestr RY wymagany jest w każdym układzie realizującym automat Mealy'ego w celu stabilizacji jego działania [1, 3].

Układ ten działa w następujący sposób:

- w początkowej chwili $\mathbf{t} = 1$ rejestr RG zawiera kod $K(a_m)$ a rejestr RMI kod $K(Y_t)$. Licznik CT przechowuje wartość 0, odpowiadającą kodowi $K(y_n^t)$ µoperacji z pierwszego elementu sekwencji β_k. Kody te dekodowane są w dekoderze D_0 a µoperacja zapisana jest w n-tym przerzutniku rejestru RY. Układ CC sprawdza czy został osiągnięty koniec sekwencji i w takiej sytuacji ustawia sygnał y_0;
- w chwili $\mathbf{t} = 2$ jeżeli $y_0 = 0$ to licznik generuje kod następnej µoperacji a stan automatu i zawartość rejestru RMI nie zostają zmienione. Cykl jest powtarzany do osiągnięcia końca sekwencji;
- jeżeli $y_0 = 1$ to ścieżka danych jest wykonywana. W rejestrze RG zapisywany jest kod następnego stanu $K(a_s)$, do rejestru RMI wpisywany jest kod odpowiedniej mikroinstrukcji a licznik jest zerowany.

5. Metoda syntezy automatu PD_{VRC}

W tym rozdziale zostaną przedstawione etapy metody syntezy automatu o strukturze PD_{VRC}. Wejściem do procesu syntezy jest tablica przejść-wyjść automatu [3, 5]. Metoda liniowego przekształcenia stosowana jest bezpośrednio do µinstrukcji zapisanych w tablicy a nie do sieci działań z której została ona uzyskana.

Proponowana metoda syntezy składa się z następujących kroków:

1. *Wyróżnienie i kodowanie µinstrukcji*. Krok ten polega na wyszukaniu wszystkich mikroinstrukcji Y_t występujących w tablicy przejść-wyjść i zakodowaniu ich binarnym kodem $K(Y_t)$. Dla przykładu z tab. 1 $Y_1 = \{y_1, y_2\}$, $Y_2 = \{y_3\}$, $Y_3 = \{y_1, y_3, y_4\}$ i $K(Y_1) = 00$, $K(Y_2) = 01$, $K(Y_3) = 10$.
2. *Utworzenie sekwencji µoperacji*. Jeżeli $Y(O_i) = Y(O_j)$ to $\beta_i = \beta_j$. Oznacza to, że wystarczy utworzyć tylko jedną sekwencje β_t dla każdej unikalnej µinstrukcji Y_t. µoperacje $y_n \in \beta_t$ zastępowane są odpowiadającymi im µoperacjami y_n^t. Wszystkie sekwencje utworzą zbiór $B = \{\beta_1, ..., \beta_T\}$. Dla przykładu z tab. 1 $B = \{\beta_1, \beta_2, \beta_3\}$ i $\beta_1 = \langle y_1^1, y_2^1 \rangle$, $\beta_2 = \langle y_3^2 \rangle$, $\beta_3 = \langle y_1^3, y_3^3, y_4^3 \rangle$.
3. *Kodowanie µoperacji*. Każdemu i-temu elementowi wektora β_t przypisujemy binarny kod $K(y_n^t)$ odpowiadający wartości i-1 ($i = 1, ..., N_t$). Dla przykładu z tab. 1 $K(y_1^1)=00$, $K(y_2^1)=01$, $K(y_3^2)=00$, $K(y_1^3)=00$, $K(y_3^3)=01$, $K(y_4^3)=10$.
4. *Utworzenie tablicy przejść-wyjść automatu PD_{VRC}*. Tablica ta tworzona jest poprzez zastąpienie kolumny Y_h z początkowej tablicy przejść-wyjść kolumną Ψ_h. Do kolumny tej wpisuje się zmienne ψ_r, które są równe 1 w kodzie $K(Y_t)$. Dla przykładu z tab. 1 jest to tab. 2.
5. *Utworzenie funkcji Φ i Ψ*. Funkcje te reprezentują systemy (1) i (8) i tworzone są na podstawie tablicy przejść-wyjść automatu PD_{VRC}.
6. *Utworzenie tablicy układu CC*. Jest to tablica prawdy służąca do implementacji systemu (9). W układzie cyfrowym może zostać zrealizowana z wykorzystaniem pamięci ROM.
7. *Utworzenie tablicy dekodera*. Tablica ta służy do realizacji systemu (10) i również może zostać zrealizowana z wykorzystaniem pamięci ROM.

Tab. 2. Fragment tablicy przejść-wyjść automatu PD_{VRC}
Tab. 2. A part of DST table of Mealy PD_{VRC} FSM

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Ψ_h	Φ_h	h
a_2	0 1 0	a_3	0 1 1	1	-	$D_2 D_3$	1
a_3	0 1 1	a_4	1 0 0	$\neg x_1$	ψ_2	D_1	2
		a_5	1 0 1	x_1	ψ_1	$D_1 D_3$	3

Ze względu na to iż tablica dekodera i tablica układu CC posiada te same zmienne wejściowe może ona zostać zapisana w jednej tabeli. Dla przykładu z tab. 1 jest to tab. 3.

Tab. 3. Tablicy układu CC i dekodera D_0 dla automatu PD_{VRC}
Tab. 3. Table of CC circuit and D_0 decoder of Mealy PD_{VRC} FSM

Y_t	y_n^t	$C(y_n)$ $K(Y_t)$ $K(y_n^t)$	y_0	y_1	y_2	y_3	y_4	i
Y_1	y_1^1	0 0 0 0	0	1	0	0	0	1
Y_1	y_2^1	0 0 0 1	1	0	1	0	0	2
Y_2	y_3^2	0 1 0 0	1	0	0	1	0	3
Y_3	y_1^3	1 0 0 0	0	1	0	0	0	4
Y_3	y_3^3	1 0 0 1	0	0	0	1	0	5
Y_3	Y_4^3	1 0 1 0	1	0	0	0	1	6

6. Podsumowanie

Zaproponowana metoda realizacji dwupoziomowego skończonego automatu stanów z wyjściami typu Mealy'ego z zastosowaniem liniowego przekształcenia sieci działań i adresowaniem mikroinstrukcji pozwala na zmniejszenie liczby funkcji zależnych od warunków logicznych oraz zmiennych wewnętrznych realizowanych przez kombinacyjny układ automatu. Liniowe przekształcenie sieci działań zapewnia konieczność użycia tylko jednego dekodera do implementacji systemu mikrooperacji. Przeprowadzone badania analityczne, oparte o metodę przedstawioną w [3], pokazały, że proponowana metoda pozwala na zmniejszenie wymaganych, do implementacji układu tablic LUT w strukturach FPGA. Ubocznym efektem zastosowania proponowanej metody jest zwiększenie $M_t\times$ liczby cykli potrzebnych do wygenerowania µinstrukcji Y_t.

Zaprezentowana metoda syntezy oparta o liniowe przekształcenie sieci działań operuje jednak tylko na tablicy przejść-wyjść. Powoduje to, że jest ona znacznie łatwiejsza do implementacji podczas realizacji programowego sytemu do automatycznej syntezy automatów.

W porównaniu z metodą syntezy gdzie nie jest stosowany licznik a wynikowa sieć działań jest ponownie znakowana stanami [4] zmniejszona zostaje liczba zmiennych potrzebnych do kodowania stanu. Powoduje to iż układ kombinacyjny realizuje mniejszą liczbę funkcji wzbudzeń przerzutników, w układzie użyta jest mniejsza liczba przerzutników w rejestrze stanu oraz funkcje realizowane przez układ kombinacyjny są zależne od mniejszej liczby zmiennych. Z drugiej strony, ze względu iż jedna mikrooperacja może posiadać kilka kodów, układ kombinacyjny może realizować więcej funkcji potrzebnych do kodowania mikrooperacji. Natomiast w porównaniu z metodą gdzie adresowane są mikrooperacje [6] zmniejszony zostaje cykl licznika oraz liczba funkcji realizowanych przez układ kombinacyjny. Zwiększeniu jednak może ulec liczba wejść dekodera. Dobór odpowiedniej metody zależny jest od charakterystyki konkretnego algorytmu sterowania.

Pracę wykonano w ramach projektu badawczego finansowanego ze środków Zintegrowanego Programu Operacyjnego Rozwoju Regionalnego (Działanie 2.6: Regionalne strategie innowacyjne i transfer wiedzy) z udziałem Europejskiego Funduszu Społecznego.

7. Literatura

[1] Adamski M., Barkalov A.: Architectural and Sequential Synthesis of Digital Devices, University of Zielona Góra Press, Zielona Góra, 2006.
[2] Adamski M., Barkalov A., Bukowiec A.: Synthesis of Control Units on Verticalized Flow-Chart, Proceedings of the 12th Conference MIXDES'05, Kraków, 2005, vol. 1, s. 209-213
[3] Baranov S.: Logic Synthesis for Control Automata, Kluwer, 1994.
[4] Barkalov A., Bukowiec A.: Synteza automatów skończonych z wyjściami typu Mealy'ego z zastosowaniem liniowego przekształcenia sieci działań, Materiały VIII konferencji RUC'05, Szczecin, 2005, s. 9-16
[5] Barkalov A., Węgrzyn M.: Design of Control Units with Programmable Logic, University of Zielona Góra Press, Zielona Góra, 2006.
[6] Barkalov A., Titarenko L., Bukowiec A.: Optimization of Circuit of Mealy Finite State Machine on FPGA Based on Encoding of the Fields of Compatible Microoperations and Verticalization of Flow-chart, Proceedings of the 3rd IFAC Workshop DESDes'06, Rydzyna, 2006, s. 297-300
[7] Łuba T. (ed.): Synteza układów cyfrowych, WKŁ, Warszawa 2003.
[8] Salcic Z.: VHDL and FPLDs in Digital Systems Design, Prototyping and Customization, Kluwer, 1998.

Artykuł recenzowany