Alexander A. BARKALOV, Larysa TITARENKO, Jacek BIEGANOWSKI
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Synteza jednostki sterującej z wykorzystaniem zmodyfikowanych liniowych łańcuchów bloków operacyjnych

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

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

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

Streszczenie

W artykule przedstawiono metodę optymalizacji liczby makrokomórek PAL mikroprogramowalnego układu sterującego. Proponowana metoda wykorzystuje dodatkowe mikroinstrukcje zawierające kody pseudorównoważnych liniowych łańcuchów bloków operacyjnych. Rozwiązanie wykorzystuje osadzone bloki pamięci, które często pozostają niezagospodarowane, do implementacji pamięci sterownika. W artykule przedstawiono także przykład zastosowania omawianej metody.

Słowa kluczowe: mikroinstrukcja, mikroprogramowy układ sterujący, CPLD, PAL.

Synthesis of control unit with modified operational linear chains

Abstract

The method of optimization of amount of PAL macrocells in the circuit of compositional microprogram control unit is proposed. The method is based on introducing of additional microinstructions with codes of the classes of pseudo-equivalent operational linear chains. The proposed method is based on usage of natural redundance of embedded memory blocks that are used to implement the control memory. An example of application of proposed method is given.

Keywords: microinstruction, control memory, compositional microprogram control unit, CPLD, PAL.

1. Wprowadzenie

Jednostka sterująca (ang. *Control Unit* – CU) jest jednym z najważniejszych elementów składowych większości systemów cyfrowych. Jej zadaniem jest koordynowanie współpracy pozostałych bloków systemu [2]. Obecnie jednostki sterujące implementuje się między innymi w układach programowalnych CPLD [3, 4]. Przy czym nadal aktualny pozostaje problem optymalizacji zasobów zajmowanych przez układ sterujący [1, 4], co przekłada się na zmniejszenie kosztów całego systemu cyfrowego [9]. Jednym ze sposobów optymalizacji jest dokonanie wyboru struktury CU odpowiadającej właściwościom realizowanego algorytmu sterującego [8]. Na przykład, liniowy algorytm sterujący może zostać zrealizowany za pomocą mikroprogramowalnego układu

Mgr inż. Jacek BIEGANOWSKI

Ukończył w roku 2003 studia na kierunku informatyka o specjalności inżynieria komputerowa. Od 2004 roku pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego na stanowisku asystenta. Jego zainteresowania związane są z systemami operacyjnymi, techniką cyfrową oraz algorytmami ewolucyjnymi.

e-mail: j.bieganowski@iie.uz.zgora.pl

sterującego (ang. *Compositional Microprogram Control Unit* – CMCU) [7, 8]. W przypadku optymalizacji liczby elementów PAL w układach wykorzystujących architekturę CPLD, minimalizacja polega na zredukowaniu liczby implikantów realizowanej funkcji boolowskiej przedstawionej w dysjunkcyjnej postaci normalnej (DNF). W artykule zaproponowano metodę minimalizacji liczby mikrokomórek PAL realizujących logikę układu CMCU. Zaproponowana metoda opiera się na wprowadzeniu dodatkowych makroinstrukcji do algorytmu sterującego.

2. Mikroprogramowalny układ sterujący

Niech algorytm sterowania systemu cyfrowego będzie reprezentowany przez sieć działań Γ [9] składającą się ze zbioru węzłów $B=\{b_0,b_E\}\cup E_1\cup E_2$ oraz zbioru łuków $E=\{\langle b_q,b_t\rangle \mid b_q,b_t\in B\}$. Niech węzeł początkowy będzie oznaczony jako b_0, węzeł końcowy jako b_E, zbiór węzłów operacyjnych jako E_1, natomiast zbiór węzłów warunkowych jako E_2. Węzeł operacyjny $b_1\in E_1$ zawiera zestaw mikrooperacji $Y(b_q)\subseteq Y$, gdzie $Y=\{y_1,...,y_N\}$ jest zbiorem mikrooperacji układu cyfrowego. Węzeł warunkowy $b_q\in E_2$ zawiera element ze zbioru warunków logicznych $X=\{x_1,...,x_L\}$. Sieć działań Γ jest nazywana liniową siecią działań LFCA (ang. *Linear Flow-Chart of Algorithm* – LFCA), jeżeli liczba wezłów operacyjnych M=|E$_1$| przekracza 75% wszystkich węzłów sieci [5].

Niech zbiór $C=\{\alpha_1,...,\alpha_G\}$ będzie utworzony na podstawie LFCA Γ, gdzie $\alpha_g\in C$ jest liniowym łańcuchem bloków operacyjnych OLC (ang. O*perational Linear Chain*). OLC α_g jest ciągiem węzłów operacyjncych $\langle b_{g1},...,b_{gFg}\rangle$, takich, że krawędź $\langle b_{gi},...,b_{gi+1}\rangle\in$E występuje dla każdej pary sąsiednich elementów (i=1,...,F_g–1). Każdy OLC α_g posiada tylko jedno wyjście O_g oraz skończoną liczbę wejść. Formalna definicja liniowych łańcuchów bloków operacyjnych, ich wejść oraz wyjść jest opisana np. w [5, 7, 8]. Niech każdy węzeł $b_q\in E_1$ odpowiada mikroinstrukcji MI_q o adresie $A(b_q)$, niech adres ten składa się z

$$R=\left\rceil log_2 M\right\lceil \qquad (1)$$

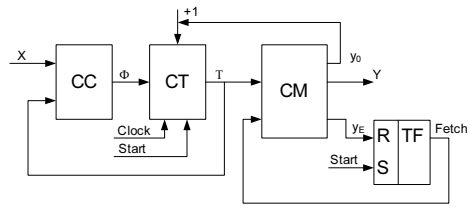
bitów.

Dla naturalnego kodowania [7, 8] mikroinstrukcji MI_q, gdzie $b_q\in E_1$, i przy spełnionym warunku

$$A\left(b_{gi+1}\right)=A\left(b_{gi}\right)+1\ \left(g=1,...,G;i=1,...,F_g\right) \qquad (2)$$

do adresowania mikroinstrukcji może zostać użyty algorytm opisany w [5]. Sieć działań LFCA Γ w tym przypadku może być zrealizowana przy użyciu układu CMCU U_1 (rys. 1) [2].

Układ CMCU U_1 działa w następujący sposób. Jeżeli *Start*=1, wtedy do licznika CT jest ładowany adres pierwszej mikroinstrukcji (MI) algorytmu sterującego. Jednocześnie przerzutnik TF jest ustawiany i *Fetch*=1, co powoduje rozpoczęcie pobierania mikroinstrukcji z pamięci CM odpowiadającej węzłowi $b_q\neq O_g$.



Rys. 1. Struktura mikroprogramowego układu sterującego U_1
Fig. 1. The structucture diagram of CMCU U_1

W wyniku odczytania mikroinstrukcji z pamięci sterującej zostaje wygenerowana mikrooperacja $Y(b_q)$ oraz zmienna y_0. Jeżeli y_0=1, wtedy wraz z nadejściem impulsu zegarowego następuje zwiększenie licznika CT i zaadresowanie następnej mikroinstrukcji MI$_j$ z pamięci CM, gdzie $\langle b_q,b_j\rangle \in$E. W przypadku, gdy $b_q = O_g$ zmienna $y_0 = 0$ i adres następnej mikroinstrukcji (adres tranzycji) jest generowany przez układ CC, który powoduje odpowiednie ustawienie przerzutników w układzie licznika CT.

$$\Phi = \Phi(T, X). \tag{3}$$

Niech $T=\{T_1,...,T_R\}$ będzie zbiorem wewnętrznych zmiennych odpowiadających poszczególnym bitom adresu mikroinstrukcji. Jeżeli z pamięci zostanie odczytana mikroinstrukcja MI_q i $\langle b_q,b_E\rangle \in E$, wtedy y_E=1, przerzutnik TF jest zerowany (*Fetch*=0) i praca układu CMCU U_1 zostaje zakończona.

Jeżeli układ CMCU U_1 jest implementowany jako SoPC (ang. *System-on-a-Programmable-Chip*) [4,6] wtedy układ CC może zostać zaimplementowany z wykorzystaniem makrokomórek PAL układu CPLD, natomiast pamięć sterująca może zostać zrealizowana z wykorzystaniem osadzonych bloków pamięci EMB (ang. *Embedded Memory Block*). Zaletą układu CMCU U_1 jest minimalna liczba wyjść układu CC w porównaniu z innymi układami CMCU [5, 7, 8]. Rozwiązanie to pozwala wiec potencjalnie na minimalizację liczby makrokomórek. Jednak układ CC i licznik CT tworzą automat typu Moore'a [5], gdzie liczba implikantów funkcji (3) może być znacznie większa niż w przypadku realizacji układu jako automat typu Mealy'ego [9]. Aby dokonać minimalizacji makrokomórek układu CC, liczba implikantów w systemie (3) dla układu CMCU U_1 powinna być jak najmniejsza. Cel ten można osiągnąć dzięki wykorzystaniu pseudorównoważnych liniowych łańcuchów bloków operacyjnych (POLC) w sieci działań Γ [8], które odpowiadają pseudorównoważnym stanom automatu Moore'a [10].

Łańcuchy OLC $\alpha_i,\alpha_j \in$C są nazywane POLC, jeżeli ich wyjścia są połączone do jednego wejścia węzła sieci działań Γ [5]. Niech $\Pi_C=\{B_1,...,B_I\}$ będzie podziałem zbioru $C' \subset C$ na klasy POLC, gdzie $\alpha_j \notin$C′ jeżeli wyjście łańcucha jest połączone z węzłem b_E. Niech każda klasa $B_i \in \Pi_C$ będzie zakodowana binarnym kodem $K(B_i)$ o długości R_1 bitów, gdzie

$$R_1 = \left]\log_2 I\right[. \tag{4}$$

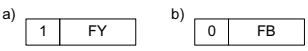
Liczba implikantów systemu (3) może być zmniejszona do wartości równoważnej liczbie implikantów automatu Mealy'ego dzięki zastosowaniu bloku transformatora adresów AT [1]. Blok ten przekształca adresy wyjść łańcuchów POLC $\alpha_i \in B$ na kody $K(B_i)$, gdzie i=1,...,I. Należy przy tym zauważyć, że takie podejście powoduje zużycie dodatkowych zasobów SoPC do implementacji układu AT i ma sens tylko wtedy gdy całkowity rozmiar bloku CC i AT jest mniejszy niż rozmiar bloku CC w układzie CMCU U_1.

3. Idea metody

Pamięć sterująca układu CMCU U_1 posiada 2^R słów, z których M zawiera mikroinstrukcje algorytmu. Jeżeli jest spełniony warunek

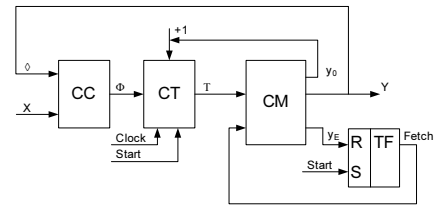
$$2^R - M \geq |C'|, \tag{5}$$

wtedy każdy łańcuch OLC $\alpha_j \in$C′ może zostać zmodyfikowany przez dodanie dodatkowego węzła O_g. W tym przypadku mikroprogram będzie zawierał dwa typy mikroinstrukcji (rys. 2).



Rys. 2. Formaty mikroinstrukcji
Fig. 2. Formats of microinstructions

Pierwszy bit, w obu formatach przedstawionych na rysunku (rys. 2) odpowiada zmiennej y_0. Pole FY (rys. 2a) zawiera informację o mikrooperacji, która ma zostać sformowana, pole FB (rys. 2b) zawiera kod $K(B_i)$. Pozostałe mikroinstrukcje posiadają format przedstawiony na rys. 2b. Węzeł O_g zawiera kod $K(B_i)$, gdzie $\alpha_g \in B_i$. Takie podejście prowadzi do układu CMCU U_2 zawierającego zmodyfikowany łańcuch OLC (rys. 3).



Rys. 3. Struktura mikroprogramowego układu sterującego U_2
Fig. 3. Structural diagram of CMCU U_2

Zasada działania układów U_1 i U_2 pozostaje taka sama, z wyjątkiem funkcji wzbudzeń układu CT, która w przypadku CMCU U_2 zależy dodatkowo od zmiennych $\tau_r \in \tau=\{\tau_1,...,\tau_{R1}\}$, wykorzystywanych do kodowania klas $B_i \in \Pi_C$.

$$\Phi = \Phi(\tau, X). \tag{6}$$

Zmienne $\tau_r \in \tau$ wchodzą w skład pola FB. Należy przy tym zauważyć, że jednostka wykonawcza [2, 7] systemu cyfrowego pozostaje w stanie jałowym, jeśli licznik CT zawiera adres nadmiarowej mikroinstrukcji.

Autorzy proponują następującą metodę projektowania układu CMCU U_2:

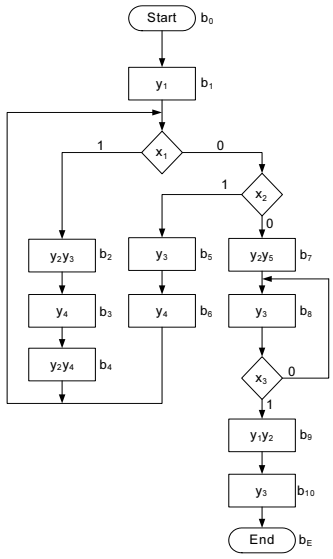
1. Przygotowanie zbioru C sieci LFCA Γ.
2. Modyfikacja łańcuchów OLC $\alpha_g \in C'$.
3. Naturalne adresowanie mikroinstrukcji.
4. Zakodowanie klas POLC $B_i \in \Pi_C$.
5. Przygotowanie zawartości pamięci sterującej.
6. Przygotowanie tabel przejść dla układu CMCU.
7. Implementacja układu CC z użyciem makrokomórek PAL, oraz implementacja bloku CM za pomocą osadzonych bloków pamięci.

4. Przykład zastosowania proponowanej metody

Niech symbol $U_i(\Gamma_j)$ oznacza układ CMCU U_i realizujący sieć działań LFCA Γ_j. Rysunek 4 przedstawia przykładową sieć działań LFCA Γ_1, dla której zastosowana zostanie omawiana metoda.

Dla sieci działań Γ_1 otrzymuje się zbiór C = $\{\alpha_{1,...,}\alpha_5\}$, gdzie $\alpha_1 = \langle b_1\rangle$, $I_1^1 = b_1$; $\alpha_2 = \langle b_2, b_3, b_4\rangle$, $I_2^1 = b_2$; $\alpha_3 = \langle b_5, b_6\rangle$, $I_3 = b_5$; $\alpha_4 = \langle b_7, b_8\rangle$, $I_4^1 = b_7$, $I_4^2 = b_8$; $\alpha_5 = \langle b_9, b_{10}\rangle$, $I_5^1 = b_9$. Przy czym $\alpha_5 \notin C'$, |G′|=4, M = 10, 2^R = 16. Dla analizowanej sieci spełniony jest warunek (5) co pozwala na zastosowanie omawianej metody.

W pierwszym kroku sieć działań zostaje uzupełniona o węzły O_g w każdym łańcuchu OLC $\alpha_g \in C'$, w rezultacie otrzymuje się zmodyfikowane łańcuchy $\alpha_1 = \langle b_1,O_1\rangle$, $\alpha_2 = \langle b_2,b_3,b_4,O_2\rangle$, $\alpha_3 = \langle b_5,b_6,O_3\rangle$, $\alpha_4 = \langle b_7,b_8,O_4\rangle$. W wyniku zastosowania naturalnego adresowania (2) przy użyciu metody z [5] otrzymuje się następujące adresy mikroinstrukcji (rys. 5).



Rys. 4. Początkowa sieć działań Γ_1
Fig. 4. Initial flow-chart of algorithm Γ_1

T_3T_4 \ T_1T_2	00	01	10	11
00	A(b1)	A(b4)	A(O3)	A(b9)
01	A(O1)	A(O2)	A(b7)	A(b10)
10	A(b2)	A(b5)	A(b8)	*
11	A(b3)	A(b6)	A(O4)	*

Rys. 5. Adresy mikroinstrukcji układu $U_2(\Gamma_1)$
Fig. 5. Addresses of microinstructions of CMCU $U_2(\Gamma_1)$

Kolejnym krokiem jest dokonanie podziału liniowej sieci działań Γ na klasy $\Pi_C = \{B_1,B_2\}$, gdzie $B_1 = \{\alpha_1,\alpha_2,\alpha_3\}$, $B_2 = \{\alpha_4\}$, w wyniku podziału $I = 2$, $R_1 = 1$, $\tau = \{\tau_1\}$. Kodowanie można zatem przyjąć następująco $K(B_1) = 0$, $K(B_2) = 1$. W rezultacie układ $U_2(\Gamma_1)$ zajmuje 14 komórek EMB. Pierwsze 4 linie pamięci sterującej przedstawia tabela 1.

Tab. 1. Fragment zawartości pamięci sterującej układu CMCU $U_2(\Gamma_1)$
Tab. 1. Fragment of content of control memory of CMCU $U_2(\Gamma_1)$

Adres	y_0	FY						Oznacznie	
$T_1T_2T_3T_4$		FB						b_q	B_i
		y_1/τ_1	y_2	y_3	y_4	y_5	y_E		
0000	1	1	0	0	0	0	0	b_1	B_1
0001	0	0	*	*	*	*	*	O_1	B_1
0010	1	0	1	1	0	0	0	b_2	B_1
0011	1	0	0	0	1	0	0	b_2	B_1

Drugi bit każdego słowa w pamięci CM jest równy y_1 (jeżeli $y_0 = 1$) lub τ_1 (jeżeli $y_0 = 0$). Stan jałowy części operacyjnej układu można uzyskać blokując impulsy zegarowe, gdy $y_0 = 0$.

Aby utworzyć tabelę przejść dla układu U_2 należy wykorzystać system funkcji przejść opisany w [9]. W omawianym przykładzie funkcje przejść przyjmują następującą postać:

$$B_1 \to x_1 b_2 \vee \overline{x}_1 x_2 b_5 \vee \overline{x}_1 \overline{x}_2 b_7;$$
$$B_2 \to x_3 b_9 \vee \overline{x}_3 b_8. \qquad (7)$$

Na podstawie (7) można utworzyć tabelę przejść (tabela 2) zawierającą następujące kolumny B_i, $K(B_i)$, b_q, $A(b_q)$, X_h, Φ_h, h.

Na podstawie tabeli 2 można stwierdzić, że $\Phi = \{D_1,\ldots,D_4\}$. Tabela przejść pozwala na utworzenie funkcji wzbudzeń przerzutników na podstawie (6). Na przykład równanie wzbudzeń dla przerzutnika jest następujące $D_3 = \overline{\tau}_1 x_1 \vee \overline{\tau}_1 \overline{x}_1 x_2 \vee \tau_1 \overline{x}_3$.

Tab. 2. Tabela przejść mikropogramowego układu sterującego $U_2(\Gamma_1)$
Tab. 2. Table of transitions of CMCU $U_2(\Gamma_1)$

B_i	$K(B_i)$	b_q	$A(b_q)$	X_h	Φ_h	h
B_1	0	b_2	0010	x_1	D_3	1
		b_5	0110	$/x_1x_2$	D_2D_3	2
		b_7	1001	$/x_1/x_2$	D_1D_4	3
B_2	1	b_9	1100	x_3	D_1D_2	4
		b_8	1010	$/x_3$	D_1D_3	5

Implementacja układu CMCU U_2 sprowadza się do realizacji funkcji (6) przy użyciu makrokomórek układu CPLD i umieszczenia zawartości pamięci sterującej w blokach EMB. Realizacja tych etapów została w artykule pominięta.

Liczba implikantów funkcji (6) w postaci DNF jest równa liczbie linii $H_2(\Gamma)$ w tabeli przejść. W omawianym przykładzie $H_2(\Gamma_1) = 5$, dla porównania tabela przejść dla układu CMCU $U_1(\Gamma_1)$ składa się z $H_1(\Gamma_1) = 11$ linii. W związku z tym istnieje możliwość redukcji rozmiaru bloku CC w układzie $U_2(\Gamma_1)$ w porównaniu z blokiem CC w układzie $U_1(\Gamma_1)$.

5. Podsumowanie

Przedstawiona metoda modyfikacji łańcuchów OLC pozwala na redukcję liczby makrokomórek PAL realizujących część kombinacyjną układu CMCU. Optymalizacja jest możliwa dzięki wykorzystaniu zazwyczaj niezagospodarowanych bloków EMB do realizacji pamięci sterującej. Proponowane rozwiązanie nie wprowadza dodatkowych bloków do struktury CMCU, a w szczególności nie jest wprowadzany blok AT.

Badania autorów wskazują, że redukcja zasobów sprzętowych jest proporcjonalna do współczynnika:

$$\eta = \frac{H_1(\Gamma)}{H_2(\Gamma)}. \qquad (8)$$

Wartość η jest zależna od charakterystyki danej sieci działań LFCA Γ i jest równa współczynnikowi stosunkowi liczby linii w tablicach przejść równoważnych automatów Moore'a i Mealy'ego użytych do implementacji sieci działań Γ.

Główną wadą proponowanej metody jest zwiększenie czasu wykonywania operacji przez system cyfrowy na skutek jałowych cykli w części operacyjnej (ścieżka danych). Metoda ta może wiec być stosowana, gdy parametry czasowe projektu spełniają założenia. Badania autorów wskazują, że zastosowanie omawianej metody pozwala na 18–22% redukcję zasobów sprzętowych w porównaniu do układu CMCU bez modyfikacji łańcuchów OLC.

6. Literatura

[1] De Micheli G. Synthesis and Optimization od Digital Circuits. McGraw Hill, NY, 1994. – 578 pp.
[2] Gajski D. Principles of Digital Design. Prentice Hall, NY, 1997. – 418 pp.
[3] V. V. Solovjev. Design of digital systems using the programmable logic integrated circuits. Hot Line-Telecom, Moscow 2001, (in Russian) – 636 pp.
[4] D. Kania. Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL. Zeszyty Naukowe Politechniki Śląskiej, Gliwice, 2004. – 240 str.
[5] A. A. Barkalov. Synthesis of control units on programmable logic devices. Donetsk National Technical University, Donetsk 2002, (in Russian) – 262 pp.
[6] C. Maxfield. The Design Warrior's Guide to FPGAs. Academic Press, Inc., Orlando, FL, USA, 2004. – 542 pp.
[7] M. Adamski and A. Barkalov. Architectural and sequential synthesis of digital devices, ISBN: 83-7481-039-4. University of Zielona Góra, Zielona Góra, 2006. – 199 pp.
[8] A. Barkalov and M. Węgrzyn. Design of control units with programmable logic, ISBN: 83-7481-042-4. University of Zielona Góra Press, Zielona Góra, 2006. – 150 pp.
[9] S. I. Baranov. Logic synthesis of Control Automata. Kluwer Academic Publishers, Boston, 1994. – 312 pp.
[10] A. A. Barkalov. Principles of optimization of logic circuit of Moore FSM, pages 65–72. Cybernetics and system analysis, 1998, (in Russian).

Artykuł recenzowany