Małgorzata KOŁOPIEŃCZYK, Larysa TITARENKO, Aleksander BARKALOV
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Projektowanie CMCU z elementarnymi łańcuchami i kodowaniem kolekcji mikrooperacji

Mgr inż. Małgorzata KOŁOPIEŃCZYK

Mgr inż. Małgorzata Kołopieńczyk jest absolwentem Uniwersytetu Zielonogórskiego. Ukończyła studia o specjalności Inżynieria Komputerowa. Od roku 1999 pracuje jako asystent na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: m.kolopienczyk@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

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 artykule zaprezentowano unikalny sposób projektowania mikroprogramowanego układu sterującego z wykorzystaniem metody współdzielenia kodów z kodowaniem kolekcji mikrooperacji. W metodzie tej wprowadzono specjalny konwerter adresów, który generuje adresy mikroinstrukcji reprezentowane poprzez pary <kod łańcucha, kod elementu łańcucha>. Do opisu algorytmu sterowania wykorzystano sieć działań z elementarnymi łańcuchami. Takie podejście umożliwia wykorzystanie wszystkich zalet metody współdzielenia kodów niezależnie od charakterystyki interpretowanej sieci działań. Zaproponowana metoda umożliwia zmniejszenie rozmiaru pamięci w stosunku do wszystkich innych znanych metod projektowania jednostek sterujących. W artykule przedstawiono także przykład zastosowania proponowanej metody.

Słowa kluczowe: mikroprogramowany układ sterujący, konwerter adresów, łańcuchy elementarne.

Design of CMCU with EOLC and Encoding of Collections of Microoperations

Abstract

The method of design of compositional microprogram control unit with codes sharing and collections of microoperations is proposed. The proposed method is based on application of special address transformer to form an address of microinstruction on the base of its representation as pair <code of operational linear chain, code of component>. The control algorithm is described using flow-chart with elementary operational linear chains. Such approach permits to use all positive features of codes sharing independently on characteristics of interpreted flow-chart of algorithm. The proposed method permits to decrease the size of control memory in comparison with all known methods of such control units design. An example of proposed method application is given.

Keywords: compositional microprogram control unit, address transformer, elementary operational linear chains.

1. Wstęp

Jednostka sterująca może być implementowana jako mikroprogramowany układ sterujący. W chwili obecnej do implementacji jednostki sterującej często wykorzystywane są programowalne układy logiczne, takie jak CPLD i FPGA [4, 5]. Problem optymalizacji ilości sprzętu w obwodzie jednostki sterującej jest ciągle aktualnym zadaniem w informatyce [5, 6]. W przypadku zastosowania układów FPGA, jednym ze sposobów rozwiązania tego problemu jest zmniejszenie ilości warunków w systemie funkcji, który opisuje obwód jednostki sterującej [7]. W przypadku mikroprogramowanego układu sterującego problem ten może być rozwiązany poprzez zastosowanie metody współdzielenia kodów [2]. Metoda ta może być stosowana przy spełnieniu pewnych warunków, których naruszenie spowoduje zwiększenie rozmiaru pamięci [2]. W artykule przedstawiono metodę projektowania mikroprogramowanego układu sterującego z elementarnymi łańcuchami (ang. *Elementary Operational Linear Chains, EOLC*) i kodowaniem kolekcji mikrooperacji umożliwiającą zmniejszenie rozmiaru pamięci projektowanego układu.

Niech dany będzie algorytm sterowania [5] przedstawiony jako sieć działań, składający się ze zbioru węzłów $B=\{b_0, b_E\}\cup B_1\cup B_2$ i zbioru połączeń $E=\{(b_i, b_j) \mid b_i, b_j \in B\}$ gdzie: b_0 jest węzłem początkowym; b_E jest węzłem końcowym; B_1 jest zbiorem węzłów operacyjnych; B_2 jest zbiorem węzłów warunkowych. Niech zbiór $C=\{\alpha_1, \ldots, \alpha_G\}$ będzie zbiorem wszystkich elementarnych łańcuchów sieci działań Γ spełniających warunek:

$$\begin{aligned} &\left|D^i \cap D^j\right| = 0 \left(i \neq j, i, j \in \{1, \ldots, G\}\right); \\ &B_1 = D^1 \cup D^2 \cup \ldots \cup D^G; \\ &G \rightarrow \min. \end{aligned} \quad , \tag{1}$$

Niech F_g będzie ilością elementów łańcucha $\alpha_g \in C$ i niech $F_{max}=max(F_1, \ldots, F_G)$. Zakodujmy łańcuch $\alpha_g \in C$ jako kod $K(\alpha_g)$ na R_1 bitach, gdzie $R_1=]log_2 F_{max}[$, i niech warunek (2) będzie spełniony dla każdej pary przyległych elementów b_{gi}, b_{gi+1} $(i=1, \ldots, F_g-1)$ łańcucha $\alpha_g \in C$, gdzie:

$$K\left(b_{gi+1}\right) = K\left(b_{gi}\right) + 1 \quad \left(g = 1, \ldots, G\right). \tag{2}$$

W takim przypadku adres $A(b_t)$ mikroinstrukcji $Y(b_t)$ od węzła $b_t \in B_1$, gdzie $b_t \in D^g$, jest reprezentowany jako:

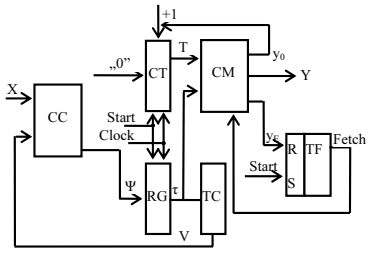
$$A(b_t) = K(\alpha_g) * K(b_t), \tag{3}$$

gdzie $*$ jest znakiem łączenia. Taka reprezentacja adresu mikroinstrukcji nazywana jest współdzieleniem kodów [2]. Reprezentacji tej odpowiada układ U_1 przedstawiony na rys. 1.

Układ kombinacyjny *CC* implementuje funkcje wzbudzeń rejestru *RG*:

$$\psi = \psi\left(V, X\right), \tag{4}$$

natomiast na wejście licznika *CT* podawany jest bezpośrednio kod *0*.



Rys. 1. Struktura mikroprogramowanego układu sterującego U_1
Fig. 1. Structural diagram of CMCU U_1

Konwerter kodu *TC* implementuje funkcje

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

Zmienna τ jest zbiorem zmiennych używanych do zakodowania elementarnych łańcuchów $\alpha_g \in C$ i $|\tau| = R_1$. Elementy $\alpha_g \in C$ są kodowane przez zmienne *T*, gdzie $|T| = R_2$. Pamięć *CM* przechowuje mikrooperację *Y*, sygnał synchronizujący y_0 oraz sygnał y_E służący do sterowania pobieraniem mikroinstrukcji z pamięci.

Taka konstrukcja *CMCU* umożliwia użycie metody kodowania łańcuchów opartej na wynikach badań opisanych w [6]. Jeżeli zachodzi warunek

$$R_1 + R_2 > R, \tag{6}$$

gdzie $R =]\log_2 M[$, $M = |B_1|$, wówczas rozmiar pamięci CM zwiększy się w porównaniu z minimalna wartością

$$V_{min} = 2^R (N + 2) \tag{7}$$

a to doprowadzi do zwiększenia ilości dedykowanych bloków pamięci w implementowanej pamięci *CM*.

W artykule zaproponowano nowy sposób konstrukcji *CMCU*, który umożliwia zastosowanie metody współdzielenia kodów przy jednoczesnym zachowaniu minimalnego rozmiaru pamięci *CM* oraz umożliwia zmniejszenie wymaganego rozmiaru pamięci w porównaniu z parametrem V_{min}.

2. Idea proponowanej metody

Niech początkowa sieć działań Γ zawiera *Q* mikroinstrukcji $Y_q \subseteq Y$. Zakodujmy każdą mikroinstrukcję Y_q kodem $K(Y_q)$ na R_4 bitach $q = (1, \ldots, Q)$. Do kodowania mikroinstrukcji wykorzystajmy zmienne $z_r \in Z$. Wprowadźmy konwerter adresów *AT* do układu o strukturze U_1, który będzie przekształcał adresy (3) w kody $K(Y_q)$ $q = (1, \ldots, Q)$. Wykorzystajmy układ *LCS* (ang. *Local Control Signal, LCS*) do generowania wewnętrznych zmiennych

$$y_0 = y_0(\tau, T), \tag{8}$$

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

Wprowadzenie układu *LCS* do układu sterującego ze współdzieleniem kodów prowadzi do utworzenia mikroprogramowanego układu sterującego o strukturze U_2 (rys. 2).

Konwerter adresów implementuje system funkcji:

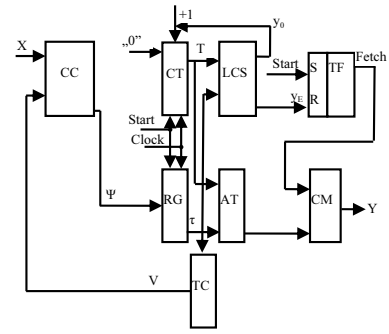
$$Z = Z(T, \tau). \tag{10}$$

Układ *LCS* steruje synchronizacją rejestru *RG* oraz licznika *CT* (używając wewnętrznej zmiennej y_0) oraz pobieraniem mikroinstrukcji (używając wewnętrznej zmiennej y_E).

Jeżeli spełniony jest warunek (6), to taka konstrukcja umożliwia zachowanie wszystkich pożądanych cech metody współdzielenia kodów, a przy spełnionym warunku

$$R_4 < R \tag{11}$$

rozmiar pamięci *CM* zmniejszy się w porównaniu z wartością V_{min}.



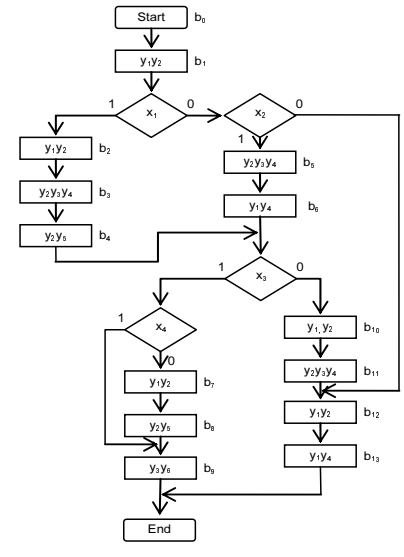
Rys. 2. Struktura mikroprogramowanego układu sterującego U_2
Fig. 2. Structural diagram of CMCU U_2

Zaproponowana metoda syntezy CMCU U_2 składa się z następujących etapów:

1. Przekształcenie początkowej sieci działań.
2. Utworzenie zbioru łańcuchów elementarnych.
3. Zakodowanie łańcuchów elementarnych.
4. Zakodowanie elementów łańcuchów.
5. Zakodowanie kolekcji mikrooperacji.
6. Wyznaczenie zawartości pamięci.
7. Zakodowanie klasy łańcuchów pseudoekwiwalentnych.
8. Utworzenie tabeli przejść.
9. Utworzenie tabeli konwertera adresów.
10. Utworzenie tabeli układu LCS.
11. Utworzenie tabeli konwertera kodu.
12. Wyznaczenie funkcji wzbudzeń.
13. Implementacja.

3. Przykład

W tym rozdziale omówimy przykład zastosowania opisanej metody projektowania *CMCU* $U_2(\Gamma_1)$, dla interpretowanej sieci działań *FCA* Γ_1 (rys. 3).



Rys. 3. Początkowa sieć działań
Fig. 3. Initial flow-chart of algorithm

Przekształcenie początkowej sieci działań. Przekształcenie to polega na dodaniu dodatkowych węzłów oraz zmiennych do początkowej sieci działań. Dla *FCA* Γ_1 przekształcenie to ogranicza się do wprowadzenia zmiennej y_E do węzłów b_9 i b_{13}. Struktura *FCA* pozostaje niezmieniona.

Utworzenie zbioru elementarnych łańcuchów. Zgodnie z metodą [3] dla *FCA* Γ_1 otrzymano następujący zbiór *EOLC*: $C=\{\alpha_1, \ldots, \alpha_7\}$, gdzie $\alpha_1=\langle b_1\rangle$, $I_1^1=O_1=b_1$, $\alpha_2=\langle b_2, b_3, b_4\rangle$, $I_2^1=b_2$, $O_2=b_4$, $\alpha_3=\langle b_5, b_6\rangle$, $I_3^1=b_5$, $O_3=b_6$, $\alpha_4=\langle b_7, b_8\rangle$, $I_4^1=b_7$, $O_4=b_8$, $\alpha_5=\langle b_9\rangle$, $I_5^1=O_5=b_{10}$, $\alpha_6=\langle b_{10}, b_{11}\rangle$, $I_6^1=b_{10}$, $O_6=b_{11}$, $\alpha_7=\langle b_{12}, b_{13}\rangle$, $I_7^1=b_{12}$, $O_7=b_{13}$. Z analizy zbioru C wynika, że $G=7$, $F_{max}=3$, $O(\Gamma_1)=\{b_1,b_4,b_6,b_8,b_9,b_{11},b_{13}\}$, $R_1=3$, $R_2=2$, $M=13$, $R=4$.

Kodowanie EOLCs. Łańcuchy *CMCU* $U_2(\Gamma_1)$ zakodowano w następujący, trywialny sposób: $K(\alpha_1)=000$, $K(\alpha_2)=001$, …, $K(\alpha_7)=110$.

Kodowanie elementów EOLC. Niech pierwszy element *EOLC* $\alpha_g \in C$ zawiera kod zero. Kody następnych elementów wyznaczane są zgodnie z (2). Dal *CMCU* $U_2(\Gamma_1)$ otrzymano: $K(b_1) = K(b_2) = K(b_5) = K(b_7) = K(b_9) = K(b_{10}) = K(b_{12}) = 00$; $K(b_3) = K(b_6) = K(b_8) = K(b_{11}) = K(b_{13})=01$; $K(b_4)=10$.

Zakodowanie kolekcji mikrooperacji. Początkowa sieć działań zawiera Q=5 kolekcji mikrooperacji $Y_1=\{y_1, y_2\}$, $Y_2=\{y_2, y_3, y_4\}$, $Y_3=\{y_2, y_5\}$, $Y_4=\{y_1, y_4\}$, $Y_5=\{y_3, y_6\}$. Zatem $R_4=3$, $Z=\{z_1, \ldots, z_4\}$, warunek (11) jest spełniony, tak, więc zastosowanie metody ma sens. Kolekcje mikrooperacji zakodowano następująco: $K(Y_1)=000$, $K(Y_2)=001$, …, $K(Y_5)=100$.

Wyznaczenie zawartości pamięci. Pamięć *CMCU* U_2 reprezentowana jest poprzez tabelę [4, 5] o następujących kolumnach: $K(Y_q)$, Y_q, b_t. Dla *CMCU* $U_2(\Gamma_1)$ tabela ta zawiera $Q=5$ wierszy (tab. 1).

Tab. 1. Zawartość pamięci
Tab. 1. Content of control memory

$K(Y_q)$	Y_q	b_t
000	y_1, y_2	$b_1, b_2, b_7, b_{10}, b_{12}$
001	y_2, y_3, y_4	b_3, b_5, b_{11}
010	y_2, y_5	b_4, b_8
011	y_1, y_4	b_6, b_{13}
100	y_3, y_6	b_9

Zakodowanie klas łańcuchów pseudoekwiwalentnych. Dla *CMCU* $U_2(\Gamma_1)$ otrzymano $\Pi_C=\{B_1, B_2, B_3, B_4, B_5\}$, gdzie $B_1=\{\alpha_1\}$, $B_2=\{\alpha_2, \alpha_3\}$, $B_3=\{\alpha_4\}$, $B_4=\{\alpha_6\}$, $B_5=\{\alpha_5, \alpha_7\}$. Klasy łańcuchów zakodowano w następujący sposób: $K(B_1)=000$, …, $K(B_5)=100$.

Utworzenie tabeli przejść CMCU. Tabela przejść jest podstawą do utworzenia funkcji (4). Jej fragment przedstawiono w tabeli 2.

Tab. 2. Fragment tabeli przejść *CMCU* U_2
Tab. 2. Fragment of the *CMCU* U_2 transitions table

B_i	$K(B_i)$	I_g^j	$A(I_g^j)$	X_h	Ψ_h	h
B_1	000	I_2^1	00100	x_1	D_3	1
		I_3^1	01000	$/x_1x_2$	D_2	2
		I_7^1	11000	$/x_1/x_2$	D_1D_2	3

Utworzenie tabeli konwertera adresów. Tabela ta jest podstawą do utworzenia funkcji (10). Jej fragment przedstawiono w tabeli 3.

Tab. 3. Fragment tabeli konwertera adresów
Tab. 3. Fragment of the table of address transformer

b_m	$A(b_m)$	Y_q	$K(Y_q)$	Z_m	m
b_1	00000	Y_1	000	–	1
b_2	00100	Y_1	000	z_3	2
b_3	00101	Y_2	001	z_2	3
b_4	00110	Y_3	011	z_2z_3	4

Utworzenie tabeli układu LCS. Tabela układu LCS jest podstawą do utworzenia funkcji (8) – (9).

Utworzenie tabeli konwertera kodu. Tabela ta jest podstawą do utworzenia funkcji (5). Zawiera następujące kolumny: α_g, $K(\alpha_g)$, B_i, $K(B_i)$, V_g, g.

Wyznaczenie systemu funkcji Ψ, Z, V. Funkcje Ψ są tworzone na podstawie tabeli przejść układu C*MCU* U_2. Funkcje Z są tworzone na podstawie tabeli konwertera adresów. Funkcje V są tworzone na podstawie tabeli konwertera kodu.

Implementacja CMCU. Implementacja układu polega na zakodowania otrzymanych funkcji w matrycach FPGA. Część logiczna układu może być zaimplementowana z wykorzystaniem elementów LUT natomiast pamięć za pomocą dedykowanych bloków pamięci. Sposoby implementacji można znaleźć w [3, 4, 7].

4. Podsumowanie

Zaproponowana metoda projektowania mikroprogramowanego układu *CMCU* U_2 umożliwia użycie metody współdzielenia kodów niezależnie od charakterystyk interpretowanej sieci działań. Zachowana jest możliwość optymalnego zakodowania pseudoekwiwalentnych łańcuchów przy jednoczesnym zmniejszeniu ilości zasobów sprzętowych w układzie kombinacyjnym *CC* w stosunku do standardowego kodowania łańcuchów. Należy zwrócić uwagę, że zastosowanie konwertera adresów prowadzi do zwiększenia czasu cyklu *CMCU* U_2 w porównaniu z *CMCU* U_1. Jednak metoda ta daje potencjalną możliwość zmniejszenia rozmiaru pamięci *CMCU* U_2 w stosunku do wszystkich znanych obecnie metod implementacji CMCU [2]. Badania autorów wykazały, że w ten sposób możliwe jest zmniejszenie wykorzystania sprzętu o 18-19% w stosunku do wszystkich innych znanych metod projektowania CMCU, jeśli spełnione są warunki (6) i (11).

5. Literatura

[1] R. Grushnitsky, A. Mursaev, E. Ugrjumov: Development of systems on chips with programmable logic. SPb: BHV, Petersburg, 626 p., 2002.

[2] H. Iwai: Future CMOS Scaling. Proceeding of the 11th International Conference “Mixed Design of Integrated Circuits and System.”, Szczecin, Poland, 2004.

[3] G. De Micheli: Synthesis and Optimization of Digital Circuits. McGraw Hill, NY, 1994.

[4] A. A. Barkalov: Synthesis of Control Units on PLDs. DonNTU, Donetsk, 262p., 2002.

[5] A. A. Barkalov.: Principles of optimization of logical circuit of Moore finite-state machine. Cybernetics and system analysis, №1,1998.

[6] T. Łuba: Synteza układów cyfrowych //Praca zbiorowa pod redakcją prof. Tadeusza Łuby, WKŁ, Warszawa, 2003.

[7] V. Solovjev: Design of digital systems using the programmable logic integrate circuits. Hot line - Telecom, Moscow, 636 p., 2001.

Artykuł recenzowany