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

Optymalizacja mikroprogramowanego układu sterującego poprzez zastosowanie współdzielenia kodów

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

Mgr inż. Małgorzata KOŁOPIEŃCZYK

Od 1999 roku zatrudniona na stanowisku asystenta w Instytucie Informatyki i Elektroniki. Specjalizuje się w projektowaniu układów sterowania dyskretnego z wykorzystaniem sieci SFC.

e-mail: m.kolopienczyk@iie.uz.zgora.pl

Streszczenie

Zaproponowana w artykule metoda współdzielenia kodów umożliwia minimalizację rozmiaru pamięci niezależnie od charakterystyki implementowanego algorytmu sterowania. Metoda oparta jest na przekształceniu adresu, reprezentowanego jako pary <kod łańcucha, kod elementu łańcucha>, w adres mikroinstrukcji. W artykule przedstawiono także przykład zastosowania proponowanej metody.

Słowa kluczowe: Mikroprogramowany układ sterujący, sieć działań, współdzielenie kodów.

Optimization of circuit of compositional microprogram control unit with code sharing

Abstract

A presented code sharing method permits to save minimal size of control memory independently on characteristics of implemented control algorithm. Method is based on transformation of address of the pair <code of operational linear chain, code of component> into address of microinstruction. An example of proposed method application is given.

Keywords: Compositional Microprogram Control Unit, flow-chart, operational linear chains, code sharing.

1. Wprowadzenie

Jednostka sterująca dowolnego systemu cyfrowego może być zrealizowana przy użyciu mikroprogramowanego układu sterującego (UMS) [2, 3]. Obecnie do implementacji obwodu jednostki sterującej często wykorzystuje się 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 CPLD lub 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 drastyczne zwiększenie rozmiaru pamięci [2]. W artykule przedstawiono metodę minimalizacji rozmiaru pamięci UMS niezależną od charakterystyki implementowanego algorytmu sterowania.

2. Wprowadzenie teoretyczne i podstawowe

Niech algorytm sterowania systemu cyfrowego będzie reprezentowany przez sieć działań Γ [2], która składa się z węzła początkowego b_0, węzła końcowego b_E, węzłów operacyjnych ze zbioru B_1 i węzłów warunkowych ze zbioru B_2. Węzły sieci działań Γ ze zboru $B=B_1\cup B_2\cup\{b_0, b_g\}$ połączone są łukami $\langle b_t, b_q\rangle$ ze zbioru E. Na podstawie [3] wprowadzamy następujące definicje:

Def. 1. Łańcuch bloków operacyjnych sieci działań Γ reprezentowany jest przez skończoną sekwencję węzłów operacyjnych $\alpha_g=\langle b_{g1}, ..., b_{gFg}\rangle$ takich, że dla każdej pary sąsiadujących bloków wektora α_g, istnieje połączenie $\langle b_{gi}, b_{gi+1}\rangle\in E$.

Def. 2. Wejściem łańcucha α_g jest węzeł $b_q\in B_1$, dla którego istnieje połączenie $\langle b_t, b_q\rangle\in E$, gdzie $b_t=b_0$ lub $b_t\in B_2$ lub $b_t\notin D^g$, gdzie $D^g\subseteq B_1$ jest zbiorem węzłów łańcucha α_g.

Def. 3. Wyjściem łańcucha α_g jest węzeł $b_q\in B_1$, dla którego istnieje połączenie $\langle b_q, b_t\rangle\in E$, gdzie $b_t=b_E$ lub $b_t\in B_2$ lub $b_t\notin D^g$.

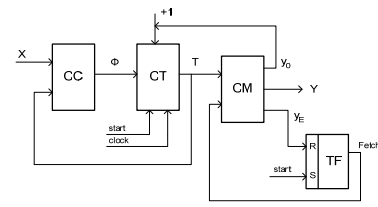
Niech zbiór $C=\{\alpha_1, \ldots, \alpha_G\}$ będzie zbiorem wszystkich łańcuchów sieci działań Γ spełniających warunek

$$\begin{aligned} &D^i\cap D^j=0(i\neq j; i,j\in\{1,...,G\});\\ &D^1\cup D^2\cup\ldots\cup D^G=B^1\\ &G\Rightarrow\min. \end{aligned} \tag{1}$$

Przy spełnionym warunku (1), każdy operacyjny węzeł początkowej sieci działań jest unikalnym elementem łańcucha $\alpha_g\in C$. Niech każdemu węzłowi $b_q\in B_1$ odpowiada unikalny binarny kod $A(b_q)$ i niech następujący warunek zostanie spełniony dla elementów dowolnego łańcucha $\alpha_g\in C$:

$$\begin{aligned} &A(b_{g_{i+1}})=A(b_{g_i})+1\\ &(i=\overline{1,F_g-1}; g=\overline{1,G}). \end{aligned} \tag{2}$$

W takim przypadku do interpretacji sieci działań Γ może zostać użyty UMS U_1 przedstawiony na rys. 1. Układ taki nazywamy UMS ze wspólną pamięcią [2].



Rys. 1. Struktura mikroprogramowanego układu sterującego U1
Fig. 1. Structural diagram of CMCU with common memory

Układ kombinacyjny implementuje system funkcji wzbudzeń przerzutników licznika CT

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

gdzie $X=\{x_1, \ldots, x_L\}$ jest zbiorem warunków logicznych z węzłów warunkowych sieci działań, $T=\{T_1, \ldots, T_R\}$ jest zbiorem wewnętrznych zmiennych, używanych do adresowania mikroinstrukcji z pamięci CM. Wówczas

$$R=]\log_2 M[, \tag{4}$$

gdzie $M=|B_1|$. System (3) formułuje adres $A(I_g^j)$ j-tego wejścia łańcucha $\alpha_g\in C$. Pamięć CM zawiera mikrooperacje $y_n\in Y$, gdzie $Y=\{y_1, \ldots, y_N\}$ jest zbiorem mikrooperacji z węzłów operacyjnych sieci działań Γ. Sygnałem sterującym dla licznika CT jest sygnał y_0. Jeżeli $y_0=1$, to wykonywana jest operacja $CT:=CT+1$; jeżeli

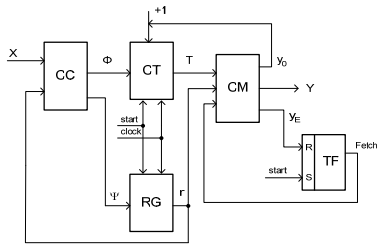
y_0=0, to zawartość licznika jest ustalana na podstawie funkcji (3). Sygnał y_E służy do implementacji trybu „stop" układu U_1.

U_1 działa w następujący sposób: gdy sygnał Start=1, to do licznika CT wczytywany jest adres pierwszej mikroinstrukcji interpretowanej sieci działań i ustawiany jest przerzutnik TF – sygnał Fetch=1 – co umożliwia pobranie mikroinstrukcji z pamięci CM. Jeżeli licznik CT zawiera adres $A(b_q)$ mikroinstrukcji $Y(b_q)$ i węzeł $b_q \neq O_g$, gdzie O_g jest wyjściem łańcucha $\alpha_g \in C$, to pamięć CM formułuje odpowiadającą $Y(b_q)$ mikrooperację oraz sygnał y_0. W taki sposób można zaimplementować przejścia pomiędzy mikroinstrukcjami, z węzłów tego samego łańcucha. Jeżeli węzeł $b_q \neq O_g$, wówczas y_0=0 i adres przejścia jest formowany zgodnie z systemem (3). Jeżeli zostanie osiągnięta mikroinstrukcja $Y(b_t)$, taka że zachodzi połączenie $\langle b_t, b_E \rangle \in E$, to generowany jest sygnał y_E, co powoduje wyzerowanie przerzutnika TF, a tym samym zakończenie pobierania mikroinstrukcji.

Zauważmy, że UMS U_1 jest skończonym automatem stanów z wyjściami typu Moore'a (ang. Moore FSM). Zawartość licznika CT jest interpretowana jako kod stanu w skończonym automacie stanów (FSM), który jest równy adresowi wyjścia łańcucha. Niech R_1 będzie liczbą zmiennych potrzebnych do zakodowania łańcucha $\alpha_g \in C$, gdzie

$$R_1 =]\log_2 G[. \tag{5}$$

Jeśli $R_1 < R$, wówczas ilość zasobów sprzętowych w obwodzie układu kombinacyjnego można zmniejszyć poprzez zastosowanie metody współdzielenia kodów [2], stosując przedstawiony na rys. 2 układ UMS U_2.



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

W układzie U_2 adres $A(b_q)$ mikroinstrukcji $Y(b_q)$ jest określany poprzez połączenie

$$A(b_q) = K(\alpha_g) * K(b_q), \tag{6}$$

gdzie $K(\alpha_g)$ jest kodem łańcucha $\alpha_g \in C$ zakodowanym na R_1, $K(b_q)$ jest kodem węzła $b_q \in D^g$(g=1, …, G) zakodowanym na

$$R_2 =]\log_2 F_{\max}[, \tag{7}$$

bitach, gdzie F_{max} to maksymalna długość łańcucha: $F_{max}=max(F_1,...,F_G)$, * jest znakiem łączenia. Zasada działania układów U_1 oraz U_2 jest identyczna, jednakże układ kombinacyjny układu U_2 generuje następujący system funkcji boolowskich

$$\Phi = \Phi(\tau, X), \tag{8}$$
$$\Psi = \Psi(\tau, X), \tag{9}$$

gdzie Ψ jest zbiorem funkcji wzbudzeń przerzutników rejestru RG utrzymującego kod $K(\alpha_g)$, reprezentowany poprzez zmienne τ, gdzie $|\tau|=R_1$. Warunek (2) przekształcamy w warunek

$$K(b_{g_{i+1}}) = K(b_{g_i}) + 1$$
$$i = \overline{1, F_g - 1}; g = \overline{1, G} \tag{10}$$

Takie podejście umożliwia zastosowanie dobrze znanej metody optymalizacji skończonego automatu stanów z wyjściami typu Moore [6], w celu zmniejszenia ilości zasobów sprzętowych w obwodzie układu kombinacyjnego. Jednakże, jeżeli równanie

$$R_1 + R_2 = R \tag{11}$$

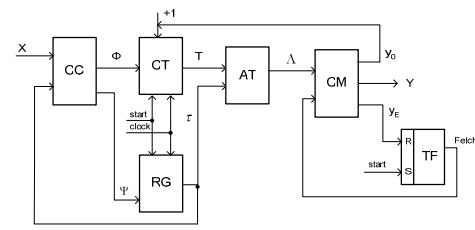
zostanie naruszone, pojemność pamięci układu U_2 drastycznie się zwiększy w porównaniu z układem U_1. W artykule zaproponowano sposób projektowania UMS, umożliwiający zastosowanie metody współdzielenia kodów bez zwiększenia rozmiaru pamięci w przypadku naruszenia warunku (11).

3. Idea proponowanej metody

Niech warunek (11) zostanie naruszony dla sieci działań Γ, co oznacza, że

$$R_1 + R_2 = R \tag{12}$$

Zastosujmy w układzie U_2 transformer adresów AT, wówczas otrzymamy, przedstawiony na rys. 3, układ U_3.



Rys. 3. Struktura mikroprogramowanego układu sterującego z współdzieleniem kodów i transformerem adresów
Fig. 3. Structural diagram of CMCU with code sharing and address transformer

Transformer adresu AT implementuje system funkcji

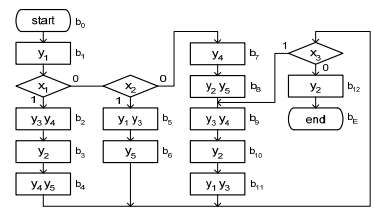
$$\Lambda = \Lambda(T, \tau) \tag{13}$$

gdzie $\Lambda=\lambda_1,..., \lambda_R$ jest zbiorem zmiennych kodujących adres mikroinstrukcji, które są utrzymywane w pamięci. Funkcje pozostałych elementów strukturalnych obydwu mikroprogramowanych układów sterujących U_3 oraz U_2 są takie same.

Zaproponowana metoda projektowania UMS U_3, składa się z następujących etapów: 1. Przekształcenie początkowej sieci działań. 2. Utworzenie zbioru łańcuchów C zgodnie z warunkiem (1). 3. Zakodowanie łańcuchów. 4. Zakodowanie węzłów operacyjnych przekształconej sieci działań. 5. Adresowanie mikroinstrukcji. 6. Wygenerowanie zawartości pamięci. 7. Utworzenie tabeli przejść. 8. Utworzenie tabeli przejść transformera adresów. 9 Utworzenie funkcji Φ, Ψ, Λ. 10. Implementacja UMS.

4. Przykład

Przykładową sieć działań Γ1 przedstawia rys. 4.



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

Przekształcenie początkowej sieci działań. Przekształcenie sieci działań ogranicza się do dodania sygnału y_E do wszystkich operacyjnych węzłów b_q, takich że $\langle b_q, b_E \rangle \in E$. Jeżeli istnieje połączenie $\langle b_t, b_g \rangle \in E$, gdzie $b_t \in B_2$, wówczas do sieci działań dodawany jest dodatkowy węzeł b_q z sygnałem y_E, a połączenie $\langle b_q, b_E \rangle$ zastąpione jest poprzez połączenia $\langle b_t, b_q \rangle$, $\langle b_q, b_E \rangle$. Jeżeli dane jest połączenie $\langle b_t, b_q \rangle \in E$, gdzie $b_q \in B_2$, wówczas do sieci działań dodawany jest dodatkowy węzeł b_t z sygnałem y_0, a początkowe połączenie $\langle b_0, b_q \rangle$ zastąpione jest poprzez połączenia $\langle b_0, b_t \rangle$, $\langle b_t, b_q \rangle$.

W przypadku sieci działań Γ_1 przekształcenie zostało zredukowane do wstawienia sygnału y_E do węzła b_{12}.

Utworzenie zbioru łańcuchów. Zastosowanie procedury opisanej w [3] prowadzi do utworzenia zbioru łańcuchów $C=\{\alpha_1, \ldots, \alpha_5\}$, gdzie $\alpha_1=\langle b_1 \rangle$, $\alpha_2=\langle b_1, b_3, b_4 \rangle$, $\alpha_3=\langle b_5, b_6 \rangle$,

$\alpha_4=\langle b_7, \ldots, b_{11}\rangle$, $\alpha_5=\langle b_{12}\rangle$. $I_1^1=O_1=b_1$, $I_2^1=b_2$, $O_2=b_4$, $I_3^1=b_5$, $O_3=b_6$, $I_4^1=b_7$, $I_4^2=b_9$, $O_4=b_{11}$, $I_5^1=O_5=b_{12}$. Z analizy zbioru C wynika, że $G=5$, $F_1=F_5=1$, $F_2=3$, $F_3=2$, $F_4=5$, $F_{max}=5$. Do zakodowania łańcucha $\alpha_g\in C$ wystarczy więc $R_1=3$ bitów, do zakodowania elementów łańcucha – $R_2=3$ bitów, a do zakodowania adresu mikroinstrukcji – $R=4$ bitów. Warunek (11) zostanie naruszony, w związku z czym zastosowanie modelu UMS U_3 jest uzasadnione.

Zakodowanie łańcuchów. Ten etap może zostać wykonany w analogiczny sposób, jak kodowanie pseudoekwiwalentnych stanów FSM z wyjściami typu Moore'a [1]. W celu optymalizacji zużycia sprzętu w obwodzie układu kombinacyjnego, kod pseudoekwiwalentnych łańcuchów powinien mieścić się w pojedynczym uogólnionym interwale R_1 – przestrzeni boolowskiej [2].

Część Π_C zbioru C, rozpatrywana jako klasa pseudoekwiwalentnych łańcuchów będzie zawierała następujące bloki: $B_1=\{\alpha_1\}$, $B_2=\{\alpha_2, \alpha_3, \alpha_4\}$, $B_3=\{\alpha_5\}$.. Zakodujmy łańcuchy $\alpha_g\in C$ w następujący sposób: $K(\alpha_1)=000$, $K(\alpha_2)=001$, $K(\alpha_3)=011$, $K(\alpha_4)=101$. Jeżeli weźmiemy również pod uwagę nieustalone stany na wejściu, wówczas klasie B_1 odpowiada kod $K(B_1)=*00$, klasie B_2 kod $K(B_2)=**1$ a klasie B_3 odpowiada kod $K(B_3)=*10$.

Zakodowanie węzłów operacyjnych sieci działań. Kodowanie węzłów powinno odbywać się tak, aby spełniony był warunek (10). Stosując metodę [2] otrzymamy następujące kody: $K(b_1)=K(b_2)=K(b_5)=K(b_7)=K(b_{12})=000$, $K(b_3)=K(b_6)=K(b_8)=001$, $K(b_4)=K(b_9)=010$, $K(b_{10})=011$, $K(b_{11})=100$.

Adresowanie mikroinstrukcji. Adres mikroinstrukcji może zawierać dowolną wartość, gdyż będzie generowany przez transformer adresów AT. Mikroinstrukcja $Y(b_q)$ i $Y(b_t)$ może mieć taki sam adres, jeżeli zawartość węzła b_q i b_t będzie równa. W rezultacie może to spowodować minimalizację zawartości pamięci.

W naszym przykładzie mikroinstrukcjom nadano następujące adresy: $A(b_1)=0000$, $A(b_2)=0001$, …, $A(b_{12})=1011$.

Wygenerowanie zawartości pamięci. Zawartość pamięci CM opisana jest tabelą, gdzie każdy jej wiersz zawiera adres mikroinstrukcji i odpowiadającą jej mikrooperację. Sygnał y_0 dodawany jest do mikroinstrukcji $Y(b_q)$ ($b_q\neq O(\Gamma)$, $O(\Gamma)$ jest zbiorem wyjść łańcucha $\alpha_g\in C$). Np. mikroinstrukcji $Y(b_1)$ odpowiada wiersz o adresie 0000 zawierający mikrooperację y_1, mikroinstrukcji $Y(b_2)$ odpowiada wiersz o adresie 0001 zawierający y_0, y_3, y_4.

Utworzenie tabeli przejść. Do przygotowania tej tabeli konieczne jest utworzenie systemu przejść:

$$B_i \to \bigvee X_i(I_g^j)\ I_g^j, (B_i \in \Pi_c') \tag{14}$$

gdzie $X_i(I_g^j)$ jest koniunkcją warunków logicznych, ustalonych na podstawie przejścia z wyjścia łańcucha $\alpha_q\in B$, do węzła operacyjnego będącego wejściem łańcucha I_g^j, $\Pi_C'\in \Pi_C$ zawiera klasy pseudoekwiwalentnych łańcuchów, takich że ich wyjścia nie są połączone z węzłem końcowym sieci działań.

W rozpatrywanej sieci działań Γ zbiór $\Pi_C' =\{B_1, B_2\}$. Formuły tranzycji (14) są następujące:

$$B_1 \to x_1 I_2^1 \vee \overline{x_1} x_2 I_3^1 \vee \overline{x_1}\,\overline{x_2} I_4^1, B_2 \to x_3 I_4^2 \vee \overline{x_3} I_5^1$$

Tabela przejść zawiera kolumny: B_i, $K(B_i)$, I_g^j, $A(I_g^j)$, X_h, Φ_h, Ψ_h, h, gdzie $A(I_g^j)$ jest adresem wejścia I_g^j; $X_h\equiv X_i(I_g^j)$; Φ_h jest zbiorem funkcji wzbudzeń przerzutników licznika CT, które gdy są równe 1, generują kod elementu łańcucha α_g odpowiadający wejściu I_g^j CT; Ψ_h jest zbiorem funkcji wzbudzeń przerzutników rejestru RG, które gdy są równe 1, generują kod łańcucha $\alpha_g\in C$ RG; $h=\overline{1,H}$ oznacza kolejny numer wiersza tabeli. W przypadku sieci działań Γ_1 tabela przejść układu U_3 ma H=5 wierszy.

Tab. 1. Tabela przejść mikroprogramowanego układu sterującego U_3
Tab. 1. Table of transitions of CMCU U3

B_i	$K(B_i)$	I_g^j	$A(I_g^j)$	X_h	Φ_h	Ψ_h	h
B_1	*00	I_2^1	001000	x_1	–	D_3	1
		I_3^1	011000	$/x_1x_2$	–	D_2D_3	2
		I_4^1	101000	$/x_1/x_2$	–	D_1D_3	3
B_2	**1	I_4^2	101010	x_3	D_5	D_1D_3	4
		I_5^2	010000	$/x_3$	–	D_2	5

Z tabeli 1 wynika, że zarówno rejestr RG jak i licznik CT ma wejścia typu D, dlatego też $\Psi=\{D_1,D_2,D_3\}$, $\Phi=\{D_4,D_5,D_6\}$.

Utworzenie tabeli przejść transformera adresu. Tabela ta reprezentuje system (13) i zawiera następujące kolumny: b_q, $K(\alpha_g)$, $K(b_q)$, $A(b_q)$, λ_h, h. Gdzie λ_h jest zbiorem bitów adresu równych 1, w h-tym wierszu tabeli ($h = \overline{1,H_1}$). Dla sieci działań Γ_1 tabela transformera adresu AT ma H_1=12 wierszy i $H_1=|B_1|$. Pierwsze cztery wiersze omawianej tabeli pokazano w tabeli 2.

Tab. 2. Tabela Fragment tabeli transformera adresów
Tab. 2. Fragment of the table of address transformer

b_q	$K(\alpha_g)$	$K(b_q)$	$A(b_q)$	λ_h	h
b_1	000	000	0000	–	1
b_2	001	000	0001	λ_4	2
b_3	001	001	0010	λ_3	3
b_4	001	010	0011	$\lambda_3\lambda_4$	4

Utworzenie systemu funkcji Φ, Ψ, λ. Funkcje Φ i Ψ są tworzone na podstawie tabeli przejść układu. Na przykład możemy utworzyć równanie $D_1 = \overline{\tau_2}\,\overline{\tau_3}\,x_1\,\overline{x_2} \vee \tau_3\,x_3$, które odpowiada trzeciemu i czwartemu wierszowi tabeli 1. Funkcja $\lambda_r \in \lambda$ jest generowana na podstawie tabeli transformera adresu AT. Na przykład z trzeciego i czwartego wiersza tabeli 2 otrzymamy następujące równanie: $\lambda_3 = \overline{\tau_1}\,\overline{\tau_2}\,\tau_3\,\overline{T_1}\,\overline{T_2}\,T_3 \vee \overline{\tau_1}\,\overline{\tau_2}\,\tau_3\,\overline{T_1}\,T_2\,\overline{T_3} = \overline{\tau_1}\,\overline{\tau_2}\,\tau_3\,\overline{T_1}$.

Realizacja układu. Obwód układu kombinacyjnego jest realizowany przy użyciu FPGA lub CPLD zgodnie z (8) – (9), obwód transformera adresów AT jest realizowany przy użyciu FPGA lub CPLD zgodnie z systemem (13), pamięć CM jest implementowana przy użyciu dedykowanych bloków pamięci, będących częścią wszystkich nowoczesnych układów FPGA i CPLD [5, 6]. Dostępne są efektywne metody rozwiązania tego problemu [5, 8], lecz zagadnienie to wykracza poza obszar tego artykułu.

5. Podsumowanie

Odpowiednie przekształcenie adresów umożliwia zastosowanie współdzielenia kodów niezależnie od właściwości interpretowanej sieci działań, takich jak pojemność bitów kodu łańcucha, kodu jego komponentów oraz minimalna pojemność bitowa adresu mikroinstrukcji. Pozwala to na wykorzystanie przedstawionej metody do optymalnego kodowania pseudoekwiwalentnych łańcuchów i zmniejszenia zużycia zasobów sprzętowych w układzie generującym kod łańcucha, oraz kody jego elementów. Zastosowanie transformera adresu umożliwia zmniejszenie rozmiaru pamięci w porównaniu z dobrze znanymi metodami projektowania UMS. Głównym minusem proponowanej metody jest zwiększenie czasu cyklu UMS ze względu na dodatkowy czas propagacji transformera adresu. Badania autorów wykazały, że proponowana metoda umożliwi zmniejszenie wykorzystania sprzętu o 12-16% w porównaniu z UMS ze wspólna pamięcią. Jest to możliwe wówczas, kiedy zostanie spełniony warunek (12).

6. Literatura

[1] Barkalov A.A: Principles of optimization of logical circuit of Moore finite-state machine. - Cybernetics and system analysis. 1998, №1.c.65 – 72.

[2] Barkalov A. A.: Synthesis of Control Units on PLDs, DonNTU, Donetsk, 2002 (in Russian). – 262 p.

[3] Barkalov A. A, Palagin A. V.: Synthesis of Microprogram Control Units, IC NAC of Ukraine, Kiev, 1997 (in Russian). – 135 p.

[4] Grushnitsky R., Mursaev A., Ugrjumov E.: Development of systems on chips with programmable logic. – SPb: BHV – Petersburg, 2002 (in Russian)- 626 p.

[5] Luba T.: Synteza układów logicznych. – Warszawa: WSIZ, 2001. – 238 s.

[6] Salcic Z.: VHDL and FPLDs in Digital Systems Design, prototyping and Customization. – Kluwer Academic Publishers, 1998. – 412 pp.

[7] Sasao T.: Switching Theory for Logic Synthesis. – Kluwer Academic Publishers, 1999. – 316 pp.

[8] Solovjev V.: Design of digital systems using the programmable logic integrate circuits. – Moscow: Hot line - Telecom, 2001 (in Russian).-636 p.

Artykuł recenzowany