II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

PROJEKTOWANIE UKŁADÓW MIKROPROGRAMOWANYCH Z WYKORZYSTANIEM WBUDOWANYCH BLOKÓW PAMIĘCI W MATRYCACH PROGRAMOWALNYCH

Remigiusz Wiśniewski

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: r.wisniewski@iie.uz.zgora.pl

STRESZCZENIE

W artykule przedstawiony został nowy sposób syntezy układów mikroprogramowanych pod kątem implementacji w matrycach FPGA. Ideą proponowanej metody jest wykorzystanie zasobów wbudowanych bloków pamięci układów FPGA. Część adresująca systemu zrealizowana jest z wykorzystaniem bloków logicznych, poprzez generatory funkcji, tzw. tablice LUT (ang. Look-Up Tables). Pamięć układu mikroprogramowanego jest zaś implementowana w dedykowanych pamięciach matryc FPGA.

1. WPROWADZENIE

Jednostka sterująca jest jednym z najważniejszych elementów układu cyfrowego [5]. Bardzo szybki rozwój w dziedzinie techniki cyfrowej spowodował pojawienie się zintegrowanych układów takich jak System-on-a-Chip (SoC) czy System-on-Programmable-Chip (SoPC), w których bloki funkcjonalne projektowanego układu implementowane są z wykorzystaniem matryc programowalnych FPGA (Field Programmable Gate Arrays) [7]. Takie podejście wymusza modyfikację klasycznych metod projektowania jednostek sterujących. Główną cechą matryc FPGA jest wykorzystanie elementów LUT (ang. *Look-Up Tables*) do realizacji funkcji logicznych. Ilość wejść elementu LUT jest ściśle ograniczona [6,7], co wiąże się z zastosowaniem dekompozycji funkcji boolowskich w projektowym układzie [8,9].

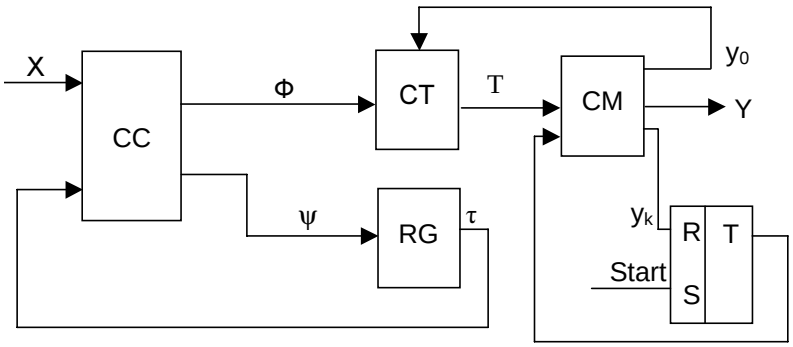
Jedną z metod zmniejszenia ilości elementów LUT jest zredukowanie ilości funkcji wykorzystanych do implementacji jednostki sterującej. Wówczas system jest realizowany jako układ dwupoziomowy, gdzie mikrooperacje mogą zostać zaimplementowane w dedykowanych blokach pamięci układu FPGA. Ponieważ wielkość dedykowanych bloków pamięci układów programowalnych jest ograniczona [7,8], rozmiar pamięci jednostki sterującej powinien być jak najmniejszy. Wymagania te w zupełności spełnia mikroprogramowany układ sterujący.

W niniejszym artykule zaproponowana zostanie nowatorska metoda zmniejszenia ilości funkcji logicznych w mikroprogramowanym układzie sterującym. Rozwiązanie opiera się na modyfikacji metody syntezy mikroprogramowanego układu sterującego o strukturze podstawowej.

W referacie pominięto podstawowe definicje związane z mikroprogramowanymi układami sterującymi, gdyż są one szczegółowo opisane w literaturze [6,8] oraz w licznych artykułach [2,4,5,10].

2. MIKROPROGRAMOWANY UKŁAD STERUJĄCY O STRUKTURZE PODSTAWOWEJ

Jednostka sterująca może zostać zrealizowana jako mikroprogramowany układ sterujący[2,8]. Podstawową cechą takiego systemu jest podział mikrokontrolera na część zarządzającą oraz część przechowującą i generującą mikrooperacje. Mikroprogramowany układ sterujący o strukturze podstawowej przedstawiony został na rys. 1.



Rys. 1. Struktura mikroprogramowanego układu sterującego

W układzie zilustrowanym poprzez rys.1. układ kombinacyjny CC oraz rejestr RG tworzą uproszczony skończony automat stanów. Automat ten jest odpowiedzialny za wyznaczanie funkcji wzbudzeń dla licznika CT. Licznik oraz pamięć układu CM stanowią blok przechowujący oraz generujący mikroinstrukcje. Struktura oraz sposoby projektowania mikroprogramowanego układu sterującego są szczegółowo opisane w literaturze [4,10].

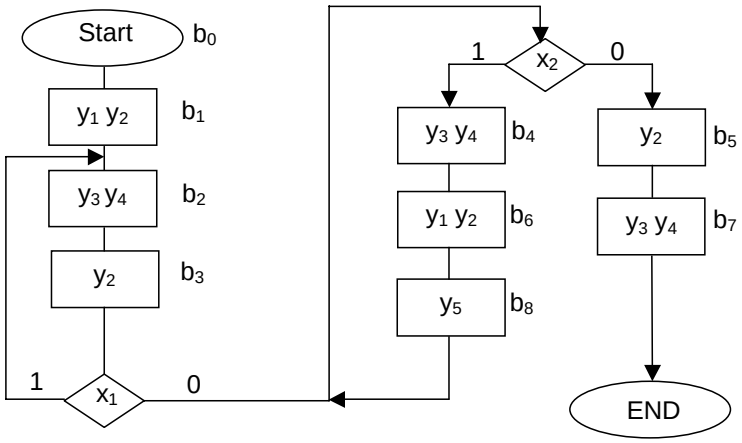
Podstawową zaletą tak zaprojektowanego systemu jest możliwość implementacji pamięci mikroprogramowanego układu sterującego w dedykowanych pamięciach struktur FPGA [8]. Pozostałe bloki realizowane są z wykorzystaniem podstawowych elementów logicznych matryc FPGA – bloków LUT (ang. *Look-Up Tables*). Takie rozwiązanie pozwala znacznie zaoszczędzić ilość wykorzystanych elementów LUT, w porównaniu do klasycznej realizacji jednostki sterującej [2]. Należy także zauważyć, że ilość wykorzystanych elementów logicznych jest ściśle związana z ilością przerzutników wykorzystanych do realizacji rejestru RG (parametr R) oraz funkcji logicznych realizowanych przez układ CC (parametr R_1). Ilość

ta jest ograniczona do wartości parametru $t_1=R+R_1$, co oznacza znaczne zmniejszenie ilości wykorzystanych elementów LUT w porównaniu do klasycznej realizacji jednostki sterującej [4].

Pomimo, iż układ U_1 wykorzystuje znacznie mniej zasobów niż jednostka sterująca oparta o klasyczne rozwiązania (skończony automat stanów), w dalszym ciągu większość bloków realizowana jest z wykorzystaniem elementów LUT. W niniejszym artykule zaprezentowana zostanie nowa idea projektowania mikroprogramowanych układów sterujących. Rozwiązanie bazuje na wprowadzeniu modyfikacji w strukturze układu U_1.

3. IDEA PROPONOWANEJ METODY

Niech dany będzie algorytm sterowania, przedstawiony za pomocą sieci działań Γ_1 (rys. 2).



Rys. 2. Przykładowa sieć działań Γ_1

W przedstawionej sieci działań można wyróżnić trzy łańcuchy bloków operacyjnych $\alpha_1=<b_1, b_2, b_3>$, $\alpha_2=<b_4, b_6, b_8>$, $\alpha_3=<b_5, b_7>$ [1]. Łańcuchy tworzą zbiór $C=\{\alpha_1, \alpha_2, \alpha_3\}$. Kolejny krok to wyznaczenie zawartości pamięci układu mikroprogramowanego. Mikroinstrukcje adresowane są z wykorzystaniem naturalnego kodu binarnego [2]. Tabela 1 przedstawia zawartość pamięci układu $U_1(\Gamma_1)$.

Tab. 1. Zawartość pamięci mikroprogramowanego układu sterującego U_1 (Γ_1)

A (b_K)	Y (b_K)	K o m e n t a r z		A (b_t)	Y (b_t)	K o m e n t a r z	
000	$y_0\ y_1\ y_2$	b_1	I_1^1	100	$y_0\ y_1\ y_2$	b_6	
001	$y_0\ y_3\ y_4$	b_2	I_1^2	101	y_5	b_8	O_2
010	y_2	b_3	O_1	110	$y_0\ y_2$	b_5	I_3^1
011	$y_0\ y_3\ y_5$	b_4	I_2^1	111	$y_3\ y_4\ y_K$	b_7	O_3

W przedstawionej tabeli I_g^j jest j-tym wejściem łańcucha $\alpha_g \in C$, O_g jest wyjściem łańcucha $\alpha_g \in C$ ($j \le F_g$, g=1, …, G). Blok b_7 jest połączony z ostatnim blokiem sieci działań Γ_1. Dlatego też, do bloku dodany został dodatkowy sygnał y_K. Z tabeli 1 widać, że dwa pierwsze bity określające adres wyjść łańcuchów $\alpha_g \in C$ mają zawsze różną wartość.

Niech ilość różnych bitów oznaczona zostanie symbolem R_2. W ogólnym przypadku

$$R_2 =]\log_2 G[. \tag{1}$$

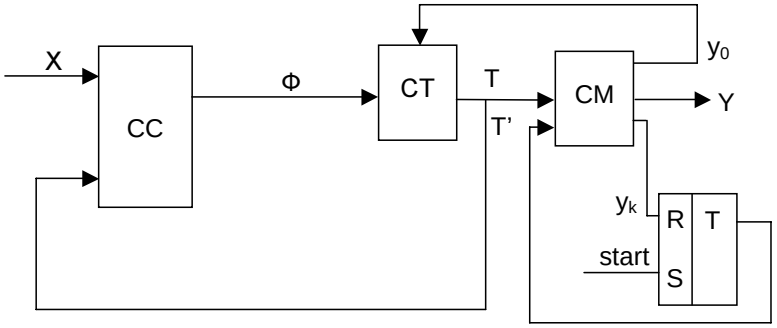
Wynika stąd, że każdy łańcuch może zostać rozpoznany po dokładnie R_2 starszych bitach adresu jego wyjścia.

Następny etap to wyznaczenie tabeli przejść mikroprogramowanego układu sterującego $U_1(\Gamma_1)$. Oczywiste jest, że pod uwagę brane tylko te łańcuchy, które nie są połączone z końcowym blokiem sieci działań. Formalnie oznacza to utworzenie nowego zbioru C' łańcuchów bloków operacyjnych, będącego podzbiorem zbioru C (C'⊆C). Łańcuch $\alpha_g \in C'$ wtedy i tylko wtedy, gdy wyjście O_g nie jest połączone z końcowym blokiem sieci działań Γ. Wówczas wartość R_2 może zostać wyznaczona ze wzoru:

$$R_2 =]\log_2 G_0[. \tag{2}$$

gdzie G_0=|C'|.

Struktura mikroprogramowanego układu sterującego (oznaczanego później jako U_2), zaprojektowanego z wykorzystaniem powyższej metody zobrazowana została na rys. 3.



Rys. 3. Struktura mikroprogramowanego układu sterującego U_2

Przejścia pomiędzy łańcuchami $\alpha_g \in C$ mikroprogramowanego układu sterującego U_2 są wyznaczane na podstawie funkcji:

$$\Phi = \Phi(T', X) . \tag{3}$$

gdzie T'⊆T, |T'|=R_2, T'={T_1, …, T_{R2}}.

4. SYNTEZA MIKROPROGRAMOWANEGO UKŁADU STERUJĄCEGO U_2

Synteza mikroprogramowanego układu sterującego z wykorzystaniem proponowanej metody może zostać podzielona na następujące etapy:

1. Utworzenie zbioru łańcuchów bloków operacyjnych, określenie adresów mikroinstrukcji oraz wyznaczenie zawartości pamięci układu mikroprogramowanego. Ten krok szczegółowo opisany został w [Barkalov, 2002]. Dla układu U_2 przedstawionego za pomocą sieci działań Γ_1 (rys. 2) otrzymamy: C'={α_1, α_2}, G_0=2, R_2=1, T'={T_1}.

2. Utworzenie tabeli przejść mikroprogramowanego układu sterującego. Jest ona niezbędna do określenia funkcji (3). Tabela przejść zawiera kolumny: O_g, $SA(O_g)$, I_q^j, $A(I_q^j)$, X_h, Φ_h, h, gdzie O_g jest wyjściem łańcucha $\alpha_g \in C$; $SA(O_g)$ to R_2 starsze bity adresu $A(O_g)$ mikroinstrukcji; $A(I_q^j)$ jest adresem j-tego wejścia łańcucha $\alpha_g \in C$; X_h to sygnał wejściowy, określający przejścia pomiędzy stanami; Φ_h jest zbiorem funkcji wzbudzeń dla licznika; h oznacza kolejny numer wiersza tabeli przejść. Tabela zawiera tylko przejścia dla łańcuchów $\alpha_g \in C'$, gdzie $C' \subseteq C$. Zbiór C' zawiera łańcuch $\alpha_g \in C$ jeśli jego wyjście nie jest połączone z końcowym blokiem sieci działań. Dla układu U_2 (Γ_1) wyznaczona zostanie tabela przejść zawierająca H=5 wierszy (Tabela 2).

Tab. 2. Tabela przejść mikroprogramowanego układu sterującego U_2 (Γ_1)

O_g	S.A.(O_g)	I_q^j	$A(I_q^j)$	X_h	Φ_h	H
O_1	0	I_1^2	001	x_1	D_3	1
		I_2^1	011	$\overline{x_1}x_2$	$D_2 D_3$	2
		I_3^1	110	$\overline{x_1}\overline{x_2}$	$D_1 D_2$	3
O_2	1	I_2^1	011	x_2	$D_2 D_3$	4
		I_3^1	110	$\overline{x_2}$	$D_1 D_2$	5

3. Wyznaczenie funkcji wzbudzeń dla licznika. Funkcja (3) jest wyznaczana na podstawie tabeli przejść układu U_2. Poszczególne zmienne wchodzące w skład funkcji wzbudzeń określane są na podstawie kodu stanu aktualnego (T={T_1, …, T_{R2}}) oraz wartości warunków (X={x_1,…,x_L}). Dla przykładu zmienna D_1 będzie miała wartość:

$$D_1 = \overline{T_1}\,\overline{x_1}\,\overline{x_2} \vee T_1\overline{x_2}. \qquad (4)$$

4. Implementacja mikroprogramowanego układu sterującego U_2. Ostatni etap to implementacja układu w matrycach FPGA. Pamięć mikroprogramowanego układu sterującego realizowana jest z wykorzystaniem dedykowanych bloków pamięci struktur programowalnych. Krok ten jest szczegółowo opisany w literaturze [5,8] i dlatego nie będzie opisywany w niniejszym artykule.

5. ZAKOŃCZENIE

Zaproponowana w artykule metoda syntezy mikroprogramowanych układów sterujących pozwala na zmniejszenie zasobów wykorzystanych elementów logicznych LUT matryc FPGA. Zysk został osiągnięty poprzez zredukowanie liczby funkcji realizujących przejścia pomiędzy łańcuchami bloków operacyjnych. Liczba ta została zmniejszona z wartości $t_1=R+R_1$ (układ U_1) do wartości $t_2=R_1$ (układ U_2). Z powyższych wyliczeń wynika następująca zależność:

$$\iota = \frac{R+R_1}{R_1} = 1+\frac{R}{R_1}.. \tag{5}$$

Na podstawie (5) przeprowadzone zostały badania, określające w jakim stopniu realizacja mikroprogramowanego układu sterującego jako U_2 jest efektywniejsza niż realizacja jako U_1. Badania przeprowadzone przez autora wykazały, że maksymalny zysk można uzyskać, gdy $R_2 \le R.$W tym przypadku ilość elementów LUT niezbędnych do realizacji układu kombinacyjnego jednostki sterującej U_2 w porównaniu do ilości elementów potrzebnych do implementacji układu kombinacyjnego jednostki sterującej U_1 może zostać zmniejszona nawet o 35-42 %.

LITERATURA

[1] S. Baranov: *Logic Synthesis for Control Automata*, Kluwer Academic Publishers, 1994, pp. 312

[2] A. A. Barkalov, A. V. Palagin: *Synthesis of microprogram control units*, Kiev: IC NAS of Ukraine, 1997

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

[4] A. A. Barkalov: *Synthesis of control units on programmable logic devices*, Donetsk: DNTU, 2002, p. 262

[5] A. A. Barkalov: *Synthesis of operational units*, Donetsk: DonNTU, 2003, pp. 306

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

[7] R. Grushnitsky, A. Mursaev, E. Ugrjumov: *Development of systems on chips with programmable logic*, SPb: BHV – Petersburg, 2002, pp. 608

[8] T. Łuba (Praca zbiorowa pod redakcją prof. Tadeusza Łuby): *Synteza układów cyfrowych*, Warszawa: WKŁ, 2003, pp. 296

[9] T. Sasao: *Switching theory for logic synthesis*, Kluwer Academic Publishers, 1999, pp. 362

[10] R. Wisniewski: *Design of compositional microprogram control units with sharing of the codes*, Gliwice: Archiwum Konferencji PTETiS , 2004 - Vol. 19, z. 4, pp. 217-220