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

SYNTEZA JEDNOSTEK STERUJĄCYCH W STRUKTURACH PROGRAMOWALNYCH

Alexander A. Barkalov

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

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

STRESZCZENIE

W artykule przedstawiono metody projektowania układów sterujących w strukturach programowalnych. Zaprezentowane zostały dwa sposoby syntezy układów sterujących. Pierwsza metoda bazuje na realizacji dwupoziomowego skończonego automatu stanów, druga to realizacja układu sterującego jako mikroprogramowany układ sterujący.

1. WPROWADZENIE

Jednostka sterująca [6] jest jednym z najważniejszych elementów systemu cyfrowego. Jest odpowiedzialna za zarządzanie innymi blokami projektowanego układu. Klasyczna metoda implementacji jednostki sterującej w postaci skończonego automatu stanów [1, 8] często pochłania zasoby układów programowalnych [7], które mogą być w efektywniejszy sposób wykorzystane przez inne bloki projektowanego systemu.

Jednym z rozwiązań tego problemu może być wykorzystanie dwupoziomowej syntezy skończonego automatu stanów (ang. *Finite-State-Machine*, *FSM*) z wyjściami typu Mealy'ego [2]. Metoda bazuje na strukturalnej dekompozycji systemu funkcji wyjść automatu. W podejściu tym mikroinstrukcjom zostają przydzielone odpowiednie kody, zaś układ kombinacyjny automatu realizuje funkcje wzbudzeń przerzutników oraz generuje kody mikroinstrukcji. Mikroinstrukcje natomiast są dekodowana na podstawie kodu w układzie drugiego poziomu.

Innym sposobem, pozwalającym znacznie zmniejszyć ilość wykorzystanych zasobów sprzętowych jest implementacja jednostki sterującej jako mikroprogramowany układ sterujący [3, 8]. Składa się on z dwóch podstawowych jednostek: bloku zarządzającego oraz układu generującego i przechowującego mikroinstrukcje. Blok zarządzający to uproszczony skończony automat stanów, odpowiedzialny za wyznaczanie adresu mikroinstrukcji. Zbudowany jest z układu kombinacyjnego, generującego funkcję wzbudzeń oraz rejestru przechowującego wartość aktualnego stanu, w jakim znalazł się układ. Drugi blok stanowi licznik oraz pamięć, w której przechowywane są mikroinstrukcje sterownika. Ideą metody

jest wyznaczenie połączenie węzłów operacyjnych początkowej sieci działań w tzw. Łańcuchy Bloków Operacyjnych (ang. *Operational Linear Chains*) [5]. Pamięć układu mikroprogramowanego jest wyznaczana na podstawie mikroinstrukcji generowanych w kolejnych blokach operacyjnych sieci działań. Na tej podstawie wyznaczana zostaje tablica przejść układu oraz równania wzbudzeń dla licznika i rejestru.

W niniejszym artykule przedstawione zostaną obie metody projektowania jednostek sterujących.

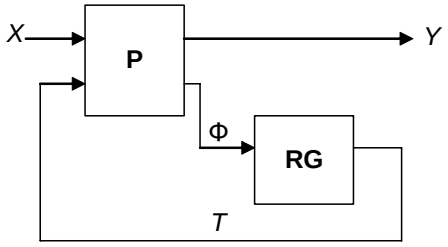
2. DWUPOZIOMOWY AUTOMAT STANÓW

Układ logiczny skończonego automatu stanu z wyjściami typu Mealy'ego jest definiowany przez system funkcji boolowskich:

$$Y = Y(T, X), \tag{1}$$

$$\Phi = \Phi(T, X). \tag{2}$$

W sytuacji takiej $X = \{x_1, \ldots, x_L\}$ jest zbiorem warunków logicznych, $Y = \{y_1, \ldots, y_N\}$ jest zbiorem mikrooperacji, $T = \{T_1, \ldots, T_R\}$ jest zbiorem zmiennych wewnętrznych, wykorzystywanych do kodowania stanu automatu $a_m \in A = \{a_1, \ldots, a_M\}$, gdzie $R =]\log_2 M[$, $\Phi = \{\phi_1, \ldots, \phi_R\}$ jest zbiorem funkcji wzbudzeń przerzutników, stanowiących pamięć automatu. Automat opisany w taki sposób może zostać zrealizowany przez jednopoziomowy układ cyfrowy automatu skończonego (rys. 1) nazywany automatem-P [2].



Rys. 1. Struktura automatu-P

W układzie takim układ P stanowi kombinacyjną część automatu i realizuje systemy (1) i (2), natomiast układ RG stanowi pamięć automatu i jest zbudowanych z przerzutników typu D [9]. Systemy (1) i (2) tworzone są na podstawie tablicy przejść i wyjść automatu [1]. Tablica zawiera następujące kolumny: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h, gdzie $a_m \in A$ jest początkowym stanem automatu; $K(a_m)$ jest binarnym kodem stanu a_m; $a_s \in A$ jest następnym stanem automatu; $K(a_s)$ jest binarnym kodem stanu a_s; X_h stanowi koniunkcję pewnych elementów ze zbioru X, które powodują przejście $\langle a_m, a_s \rangle$; $Y_h \subseteq Y$ jest zbiorem mikrooperacji formowanych podczas przejścia $\langle a_m, a_s \rangle$; $\Phi_h \subseteq \Phi$ jest zbiorem funkcji wzbudzeń

przerzutników, które są równe 1 i powodują przełączenie pamięci automatu z $K(a_m)$ na $K(a_s)$; h jest numerem przejścia ($h = 1, \ldots, H$).

System (1) jest formowany w następujący sposób:

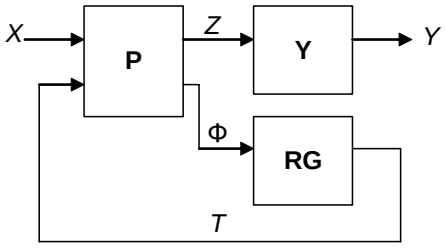
$$y_n = \bigvee_{h=1}^{H} C_{nh} A_m^h X_h \quad (n = 1, \ldots, N), \tag{3}$$

natomiast system (2) w następujący:

$$\phi_r = \bigvee_{h=1}^{H} C_{rh} A_m^h X_h \quad (r = 1, \ldots, R). \tag{4}$$

C_{nh} (C_{rh}) jest zmienną boolowską, która jest równa 1, jeżeli funkcja y_n (ϕ_r) jest zapisana w h-tym wierszu tablicy przejść-wyjść automatu, A_m^h jest koniunkcją zmiennych wewnętrznych odpowiadających kodowi $K(a_m)$ stanu początkowego $a_m \in A$ z h-tego wiersza tablicy przejść-wyjść automatu ($h = 1, \ldots, H$).

Tak zaprojektowany układ automatu cyfrowego posiada możliwie największą wydajność w przypadku implementacji z wykorzystaniem układów PLD. Jednak podstawową wadą jednopoziomowego układu automatu Mealy'ego jest stosunkowo duża liczba funkcji, zależnych od warunków logicznych i zmiennych wewnętrznych, realizowanych przez układ kombinacyjny P. Zmniejszenie liczby funkcji realizowanych przez układ P doprowadzi do redukcji wymaganych zasobów sprzętowych wymaganych do implementacji układu automatu cyfrowego. Jednym ze sposobów jest zastosowanie dwupoziomowej struktury automatu [2]. Strukturę taką można uzyskać poprzez zastosowanie maksymalnego kodowania zbiorów mikrooperacji [2]. Wyróżnijmy w tablicy przejść-wyjść automatu Q różnych zbiorów mikrooperacji $Y_q \subseteq Y$. Zakodujmy każdy taki zbiór $Y_q \subseteq Y$ binarnym kodem $K(Y_q)$. Do tego celu potrzebne będzie $R_1 =]\log_2 Q[$ zmiennych boolowskich. Zmienne te będzie reprezentował zbiór $Z = \{z_1, \ldots, z_{R_1}\}$. W takim przypadku automat Mealy'ego może zostać przedstawiony jako dwupoziomowa struktura – automat-PY (rys. 2).



Rys. 2. Struktura automatu-PY

W strukturze tej występują dwa układy kombinacyjne P i Y. Układ P realizuje system (2) formowany w sposób (4) oraz system:

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

Układ Y dekoduje mikrooperacje, realizując system:

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

W celu realizacji systemu (5) należy przekształcić początkową sieć działań poprzez usunięcie kolumny Y_h a wstawienie kolumny Z_h. Kolumna Z_h zawiera zmienne $z_p \in Z$, które są równe 1 w kodzie $K(Y_q)$, dla zbioru mikrooperacji Y_q znajdującego się w tym samy wierszu początkowej tablicy przejść-wyjść.

System (5) jest formowany na podstawie przekształconej tablicy przejść-wyjść w następujący sposób:

$$z_p = \bigvee_{h=1}^{H} C_{ph} A_m^h X_h \ \ (p = 1, \ldots, R_1), \tag{7}$$

gdzie C_{ph} jest zmienną boolowską, która jest równa 1, jeżeli funkcja z_p jest zapisana w h-tym wierszu przekształconej tablicy przejść-wyjść automatu.

W celu realizacji systemu (6) należy utworzyć tablice dekodera zbiorów mikrooperacji. Tablica ta zawiera kolumny $K(Y_q)$, Y_q i Q. Na podstawie tej tablicy można uformować system (6) w następujący sposób:

$$y_n = \bigvee_{q=1}^{Q} C_{nq} Z_q \ \ (n = 1, \ldots, N), \tag{8}$$

gdzie C_{nq} jest zmienną boolowską, która jest równa 1, jeżeli funkcja y_n jest zapisana w q-tym wierszu tablicy dekodera, Z_q jest koniunkcją zmiennych ze zbioru Z, odpowiadających kodowi $K(Y_q)$, (q = 1, …, Q).

Podejście takie zapewnia zmniejszenie liczby funkcji realizowanych przez układ kombinacyjny P z N+R (struktura P) do R_1+R (struktura PY), co prowadzi do zmniejszenia wymaganych zasobów sprzętowych do implementacji tego układu. Ponieważ układ Y posiada regularną strukturę może zostać zaimplementowany z wykorzystaniem pamięci ROM. Kod $K(Y_q)$ były by adresem, natomiast zbiory mikrooperacji były by przechowywane jako dane. Pamięć taka musiałaby przechowywać Q R_1-bitowych słów.

3. MIKROPROGRAMOWANY UKŁAD STERUJĄCY

Mikroprogramowany układ sterujący [6] jest rozwinięciem dwupoziomowej syntezy skończonego automatu stanów z wyjściami typu Mealy'ego. W tym przypadku jednostka sterująca zostaje poddana dekompozycji na dwa podstawowe bloki: układ zarządzający oraz układ pamięci, w którym przechowywane są mikroinstrukcje kontrolera.

Do zrozumienia zagadnień związanych z mikroprogramowanymi układami sterującymi, niezbędne będzie wprowadzenie podstawowych definicji.

Niech dany będzie algorytm sterowania, przedstawiony za pomocą sieci działań Γ [1], składającej się ze zbioru bloków operacyjnych $B = \{b_1, \ldots, b_k\}$, zbioru bloków decyzyjnych P oraz zbioru połączeń E. Każdy blok operacyjny $b_k \in B$ zawiera mikroinstrukcję, będącą zbiorem mikrooperacji $Y(b_k) \subseteq Y$, gdzie $Y = \{y_1, \ldots, y_N\}$ jest zbiorem wszystkich mikrooperacji. Każdy blok decyzyjny sieci działań zawiera warunek logiczny x_i, będący elementem zbioru warunków logicznych $X = \{x_1, \ldots, x_L\}$. Zbiór połączeń E określa drogę oraz kierunek przepływu informacji w sieci działań. Każda sieć działań zawiera węzeł początkowy b_0 oraz węzeł końcowy b_E.

Definicja 1. Łańcuch (bloków operacyjnych) w sieci działań Γ, to skończona sekwencja węzłów operacyjnych $\alpha_g = \langle b_{g1}, \ldots, b_{gFg} \rangle$ takich, że dla każdej pary sąsiadujących bloków wektora α_g istnieje połączenie $\langle b_{gi}, b_{bg+1} \rangle \in E$, gdzie i jest numerem bloku wektora α_g (i=1, … F_g-1).

Definicja 2. Blok $b_q \in B$ jest wejściem łańcucha α_g, jeśli istnieje takie połączenie $\langle b_t, b_q \rangle \in E$, że b_t jest blokiem początkowym lub blokiem decyzyjnym sieci działań Γ, który nie należy do łańcucha α_g.

Definicja 3. Blok $b_q \in B$ jest wyjściem łańcucha α_g, jeśli istnieje takie połączenie $\langle b_q, b_t \rangle \in E$, że b_t jest blokiem końcowym lub blokiem decyzyjnym sieci działań Γ, który nie należy do łańcucha α_g.

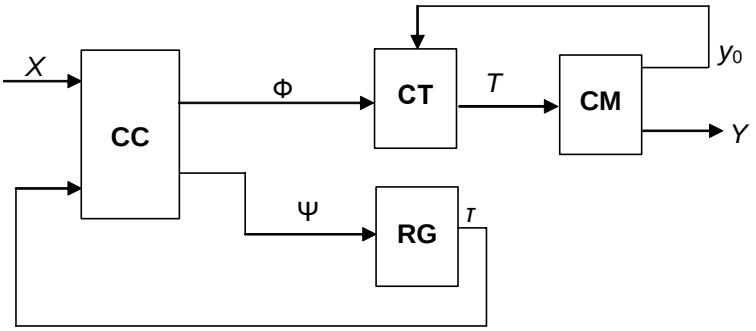
Niech D^g będzie zbiorem bloków operacyjnych wchodzących w skład łańcucha α_g. Niech zbiór $C = \{\alpha_1, \ldots, \alpha_G\}$ będzie zbiorem wszystkich łańcuchów sieci działań Γ spełniających warunek:

$$\begin{aligned} &D^g \cap D^q = \varnothing (g \neq q; g, q \in \{1, \ldots, G\}); \\ &B = D^1 \cup D^2 \cup \ldots \cup D^G; \\ &D^g \neq \varnothing (g = 1, \ldots, G). \end{aligned} \tag{9}$$

Jeśli mikroinstrukcje zostaną zakodowane zgodnie z warunkiem:

$$A(b_{gi+1}) = A(b_{gi}) + 1 (i = 1, \ldots, F_{g-1}), \tag{10}$$

gdzie $A(B_g)$ jest adresem mikroinstrukcji generowanej przez blok operacyjny $b_q \in B$, to sieć działań Γ może zostać zrealizowana za pomocą mikroprogramowanego układu sterującego U_1 przedstawionego na rys. 3.



Rys. 3. Schemat blokowy mikroprogramowanego układu sterującego

W mikroprogramowanym układzie sterującym U_1 wyróżnić można cztery podstawowe bloki funkcjonalne: układ kombinacyjny CC, rejestr RG, licznik CT oraz pamięć CM. Układ kombinacyjny oraz rejestr tworzą uproszczony skończony automat stanów S_1, którego zadaniem jest wyznaczenie adresu mikroinstrukcji. Licznik oraz pamięć wchodzą w skład układu mikroprogramowanego S_2, w którym mikroinstrukcje adresowane są zgodnie z warunkiem (10) [4]. Rejestr RG przechowuje kod $K(a_m)$ stanu $a_m \in A$, w jakim znalazł się automat S_1, gdzie $A = \{a_1, \ldots, a_M\}$ jest zbiorem wszystkich stanów automatu S_1. Ilość przerzutników niezbędnych do zbudowania rejestru można wyznaczyć ze wzoru $R =]\log_2 M[$, gdzie M oznacza ilość stanów automatu S_1. Zmienne $\tau_r \in \tau$ określają kod stanu automatu S_1. Licznik CT jest odpowiedzialny za generowanie mikroinstrukcji z pamięci CM. Zmienne $T_r \in T$ reprezentują adres mikroinstrukcji $A(b_k)$, gdzie $b_k \in B$. Niech współczynnik R_1 określa liczbę zmiennych ($|T| = R_1$), jego wartość można wyznaczyć ze wzoru $R_1 =]\log_2 K[$. Pamięć CM składa się z 2^{R1} słów (mikroinstrukcji). Każde słowo jest kodowane z wykorzystaniem N+1 bitów. Nadmiarowy bit (sygnał y_0) niezbędny jest do poprawnego określenia trybu adresowania, zgodnie z warunkiem (10).

Zasada działania układu może zostać zobrazowana następująco: w momencie aktywacji układu U_1 (aktywny blok Start w sieci działań), wartość rejestru RG ustawiona zostanie zgodnie z początkowym stanem automatu S_1, wartość licznika CT wskazywać będzie adres pierwszej mikroinstrukcji, jaka się znajduje w pamięci CM. Jeśli kolejny blok w sieci działań należy do tego samego łańcucha $\alpha_g \in C$, sygnał $y_0 = 1$. Oznacza to zwiększenie wartości licznika CT i jednoczesne zablokowanie zmiany stanu automatu S_1. Układ funkcjonuje w ten sposób tak długo, aż osiągnięte zostanie wyjście łańcucha $\alpha_g \in C$. Wówczas $y_0 = 0$ i automat S_1 zmieni stan, generując funkcje wzbudzeń dla licznika $\Psi = \{\Psi_1, \ldots, \Psi_R\}$ oraz rejestru $\Phi = \{\Phi_1, \ldots, \Phi_{R1}\}$:

$$\Phi = \Phi(\tau, X) \tag{11}$$

$$\Psi = \Psi(\tau, X) \tag{12}$$

Funkcja (11) określa kod $K(a_s)$ następnego stanu, w jakim znajdzie się układ. Funkcja (12) wyznacza adres wejścia kolejnego łańcucha $\alpha_g \in C$.

Największą zaletą mikroprogramowanych układów sterujących jest zmniejszenie liczby funkcji logicznych realizowanych przez sterownik. Jest to spowodowane wykorzystaniem bloku pamięci, w którym przechowywane są mikroinstrukcje kontrolera. W ten sposób blok CC generuje tylko niezbędne przejścia pomiędzy łańcuchami, co znacznie zmniejsza wielkość wykorzystanych zasobów układów programowalnych.

4. ZAKOŃCZENIE

Zaprezentowano dwa sposoby realizacji układu sterującego. Pierwsza metoda bazowała na wykorzystaniu dwupoziomowej syntezy skończonego automatu stanów z wyjściami typu Mealy'ego. System funkcji wyjść automatu został poddany dekompozycji. W efekcie uzyskano układ o strukturze dwupoziomowej, w której układ kombinacyjny automatu generuje kody mikroinstrukcji, które z kolei są dekodowane na podstawie kodu w układzie drugiego poziomu.

Druga z przedstawionych metod to mikroprogramowany układ sterujący. Stanowi on rozwinięcie dwupoziomowej syntezy skończonego automatu stanów z wyjściami typu Mealy'ego. W tym przypadku jednostka sterująca także została poddana dekompozycji, jednakże mikroinstrukcje przechowywane są w pamięci kontrolera. Taki sposób pozwala znacznie zmniejszyć ilość funkcji realizowanych przez układ kombinacyjny.

Badania przeprowadzone przez autora pokazują, że liczba elementów logicznych niezbędnych do implementacji układu sterowania w strukturach programowalnych, zaprojektowanego jako dwupoziomowy automat Mealy'ego albo mikroprogramowalny układ sterujący, może zostać zmniejszona nawet o 20% w stosunku do układu zaprojektowanego metodą klasyczną.

LITERATURA

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

[2] A. A. Barkalov: *Structures of the multilevel circuits of microprogram automata on PLA*, Cybernetics and System Analysis, No. 4, 1994, s. 22-29

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

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

[5] A. A. Barkalov: *Synthesis of control units on programmable logic devices*, DonNTU, Donetsk, 2002

[6] A. A. Barkalov: *Synthesis of operational units*, DonNTU, Donetsk, 2003

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

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

[9] V. V. Solovjev: *Design of Digital Systems Using the Programmable Logic Integrate Circuits*, Hot Line - Telecom, Moscow, 2001