Arkadiusz BUKOWIEC
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Synteza skończonych automatów Mealy'ego z liniowym przekształceniem sieci działań i adresowaniem mikrooperacji

Mgr inż. Arkadiusz BUKOWIEC

Ukończył studia inżynierskie (2001) o specjalności inżynieria komputerowa na Politechnice Zielonogórskiej a następnie studia magisterskie (2004) o tej samej specjalności na Uniwersytecie Zielonogórskim. Od roku 2004 pracuje jako asystent oraz jest studentem studiów doktoranckich na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego. W roku 2006 odbył jeden semestr studiów doktoranckich na Universidade Nova de Lisboa w ramach projektu Sokrates-Erasmus.

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

Streszczenie

W artykule została przedstawiona metoda zmniejszenia wymaganych zasobów sprzętowych w programowalnym układzie matrycowym do implementacji skończonego automatu stanów (FSM) z wyjściami typu Mealy'ego. Zaproponowana metoda oparta jest na liniowym przekształceniu początkowej sieci działań. W rezultacie takiego przekształcenia wszystkie mikrooperacje w przekształconej sieci działań stają się kompatybilne. Umożliwia to zakodowanie każdej mikrooperacji za pomocą binarnego kodu na możliwie minimalnej liczbie bitów. W sytuacji takiej do implementacji systemu mikrooperacji potrzebny jest tylko jeden dekoder. Dodatkowo w celu zachowania tej samej liczby stanów do generowania kolejnych adresów mikrooperacji wykorzystany zostaje licznik. Metoda ta zapewnia zmniejszenie liczby wyjść części kombinacyjnej automatu Mealy'ego w porównaniu z tą samą charakterystyką automatu Mealy'ego z kodowaniem kompatybilnych mikrooperacji. W artykule zaproponowana również została metoda syntezy z wykorzystaniem powyższych przekształceń. Metoda ta została zilustrowana przykładem.

Słowa kluczowe: automat stanów, jednostka sterująca, układy programowalne.

Synthesis of Mealy FSMs with Verticalization of Flow Chart and Addressing of Microoperations

Abstract

The method of decreasing of logic amount in programmable device implementing the logic circuit of finite state machine (FSM) is proposed. Method is based on verticalization of flow chart. As a result of verticalization all microoperations are compatible ones. It permits to encode each microoperation by code with minimal possible number of bits. In this case only one decoder is used for implementation of the microoperations system. Additionally, there is used a counter for generation of microoperations addresses. This manipulation allows to secure the same number of states like for algorithm before verticalization. This method permits to minimize number of outputs of the combinational part of Mealy FSM in comparison with the same characteristic of Mealy FSM with encoding of fields of compatible microoperations.

Keywords: FSM, Control unit, FPD.

1. Wstęp

Jednym ze sposobów projektowania jednostek sterujących układem cyfrowym lub obiektem przemysłowym jest zastosowanie skończonych automatów stanów z wyjściami typu Mealy'ego [4]. Układy programowalne są bardzo często stosowane do implementacji takiego automatu [8]. Główną cechą układów FPGA jest występowanie w nich dużej liczby tablic LUT, które mogą zrealizować dowolną funkcje logiczną n zmiennych [7, 8]. Ograniczeniem jednak jest stosunkowo mała liczba n (typowo n=4) wejść jaką posiadają tablice LUT, z drugiej strony funkcje logiczne realizowane przez kombinacyjny układ automatu posiadają znacznie więcej argumentów. Rozbieżność ta prowadzi do konieczności zastosowania blokowej dekompozycji funkcji boolowskich opisujących zachowanie automatu skończonego [7]. Negatywnym wynikiem takiej dekompozycji jest zwiększenie liczby poziomów w układzie logicznym realizującym algorytm sterowania, co może prowadzić do zmniejszenia wydajności takiego systemu.

W prezentowanym referacie przedstawiona jest metoda umożliwiająca zmniejszenie liczby funkcji logicznych realizowanych przez kombinacyjną część automatu. Prowadzi to do zmniejszenia wymaganej liczby tablic LUT w porównaniu ze znanymi metodami projektowania [1, 5].

Głównym założeniem prezentowanej metody jest przeprowadzenie liniowego przekształcenia sieci działań opisującej algorytm sterowania [2] oraz zastosowaniu licznika do generowania adresów mikrooperacji.

2. Analiza struktury z kodowaniem klas kompatybilnych mikrooperacji

Najpopularniejszym sposobem reprezentacji algorytmu sterowania jest sieć działań [1]. Sieć taka zbudowana jest z bloków operacyjnych i decyzyjnych. Niech bloki operacyjne należące do sieci działań Γ tworzą zbiór bloków operacyjnych $O(\Gamma) = \{O_1,...,O_K\}$. Każdy blok operacyjny $O_k \in O(\Gamma)$ zawiera mikroinstrukcje $Y(O_k) \subseteq Y$, gdzie $Y = \{y_1, ..., y_N\}$ jest zbiorem mikrooperacji. Każdy blok decyzyjny zawiera jeden element $x_l \in X$, gdzie $X = \{x_1, ..., x_L\}$ jest zbiorem warunków logicznych. Niech taka sieć działań będzie oznaczona stanami automatu z wyjściami typu Mealy'ego [5] i niech tworzą one zbiór stanów $A = \{a_1, ..., a_M\}$. Każdy stan $a_m \in A$ można zakodować na $R = \lceil \log_2 M \rceil$ bitach za pomocą binarnego kodu $K(a_m)$. Do reprezentacji tego kodu wykorzystane zostaną zmienne $Q_r \in Q = \{Q_1, ..., Q_R\}$. Działanie układu logicznego skończonego automatu stanów z wyjściami typu Mealy'ego może zostać opisane przez tablicę przejść-wyjść [4, 7] z następującymi kolumnami: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h (tab. 1), gdzie $a_m \in A$ jest początkowym stanem automatu; $a_s \in A$ jest następnym stanem automatu; $K(a_s)$ jest kodem stanu a_s; X_h jest koniunkcją afirmacji lub negacji pewnych zmiennych ze zbioru wejściowych warunków logicznych wymuszając przejście $\langle a_m, a_s \rangle$; $Y_h \subseteq Y$ jest mikroinstrukcją formowaną podczas przejęcia $\langle a_m, a_s \rangle$; Φ_h jest zbiorem funkcji wzbudzeń, które są równe 1 w celu przełączenia pamięć automatu z kodu $K(a_m)$ na kod $K(a_s)$, $\Phi_h \subseteq \Phi = \{D_1, \ldots, D_R\}$; h jest numerem przejścia automatu ($h = 1, ..., H$). Na tej podstawie wyprowadzany jest system wzbudzeń przerzutników i mikrooperacji:

$$\Phi = \Phi(Q, X), \tag{1}$$

$$Y = Y(Q, X). \tag{2}$$

Systemy te realizowane są przez kombinacyjną część automatu, natomiast aktualny stan zapamiętywany jest w zbiorze rejestrów. W przypadku implementacji z wykorzystaniem układów FPGA do jego budowy stosuje się przerzutniki typu D [7, 8].

Tab. 1. Fragment tablicy przejść-wyjść automatu
Tab. 1. A part of DST table of Mealy FSM

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_h	Φ_h	h
a_3	0 1 1	a_4	1 0 0	1	$y_1\, y_2\, y_3$	D_1	1
a_4	1 0 0	a_5	1 0 1	x_1	$y_1\, y_2\, y_3$	$D_1\, D_3$	2
		a_6	1 1 0	$\sim x_1$	$y_1\, y_3$	$D_1\, D_2$	3

W takiej sytuacji mamy doczynienia z jedno-poziomową strukturą automatu [1], jednak ze względu na ograniczenia tablic LUT dekompozycja funkcji logicznych doprowadzi do struktury wielopoziomowej.

Jednym ze sposobów optymalizacji wymaganych zasobów sprzętowych do realizacji automatu jest zmniejszenie liczby funkcji zależnych od warunków logicznych i zmiennych wewnętrznych. Jedną z umożliwiających to metod jest kodowanie klas kompatybilnych mikrooperacji [5]. Mikrooperacje $y_n, y_m \in Y$ są kompatybilne, jeżeli warunek

$$\forall_k \left(y_n \in Y(O_k) \rightarrow y_m \notin Y(O_k) \right) \ (k = 1, \ldots, K), \tag{3}$$

jest spełniony. Niech zbiór $\Pi_Y = \{Y^1, ..., Y^I\}$ będzie podziałem zbioru Y na klasy kompatybilnych mikrooperacji. Niech każda mikrooperacja $y_n \in Y^i$ zostanie zakodowana za pomocą kodu $K(y_n)$ na r_i bitach, gdzie $r_i = \lceil \log_2(|Y^i|+1) \rceil$ ($i = 1, ..., I$). Tak więc do zakodowania wszystkich mikrooperacji wymagane będzie

$$R_1 = \sum_{i=1}^{I} r_i \tag{4}$$

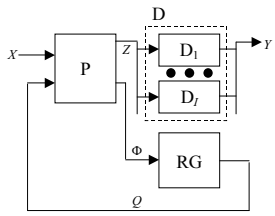
zmiennych $z_r \in Z = \{z_1, \ldots, z_{R_1}\}$. W takim przypadku automat Mealy'ego zostanie zaimplementowany w dwu-poziomowej strukturze PD (rys. 1), gdzie dekoder D_i dekoduje mikrooperacje należące do klasy $Y^i \in \Pi_Y$. W sytuacji takiej układ P realizuje funkcje (1) i

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

a dekoder D, zbudowany z I dekoderów i realizuje funkcję

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

Prowadzi to do zmniejszenia liczby funkcji zależnych od warunków logicznych i zmiennych wewnętrznych realizowanych przez kombinacyjną część automatu.

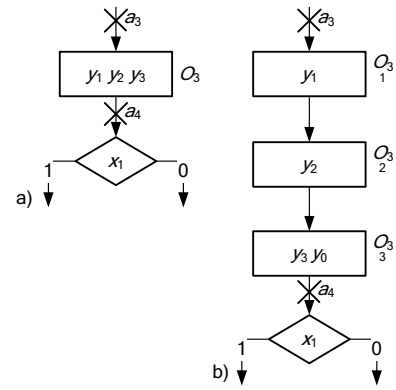


Rys. 1. Schemat blokowy automatu Mealy'ego o strukturze PD
Fig. 1. Block diagram of the Mealy PD FSM

Wadą takiego rozwiązania jest wciąż relatywnie duża liczba funkcji Z oraz konieczność implementacji I dekoderów. Najlepszym rozwiązaniem problemu zmniejszenia wyjść układu P bez zmniejszania wydajności jednostki sterującej jest sytuacja, gdy wszystkie mikrooperacje realizowane przez automat są wzajemnie do siebie kompatybilne. Przypadek taki odzwierciedlony jest przez liniową sieć działań [2, 5] gdzie każdy blok operacyjny zawiera tylko jedną mikrooperację. Głównym problem rzeczywistych sieci działań jest fakt, iż nie są one liniowe. W następnym rozdziale zaprezentowane zostanie liniowe przekształcenie sieci działań [2, 5]. Zabieg ten doprowadzi do uzyskania kompatybilności pomiędzy wszystkimi mikrooperacjami. Następnie zaproponowana zostanie struktura układu cyfrowego implementującego układ automatu Mealy'ego w oparciu o przekształconą sieć działań.

3. Liniowe przekształcenie sieci działań

Niech każdy blok operacyjny sieci działań Γ zawiera $N_k = |Y(O_k)|$ mikrooperacji. Przekształćmy zatem każdy blok operacyjny $O_k \in O(\Gamma)$ w sekwencje bloków $\beta_k = \langle O_k^1, \ldots, O_k^{N_k} \rangle$, w której każdy blok O_k^j zawiera jedną unikalną mikrooperację $y_n = \mathrm{pr}_j Y(O_k)$. Dodatkowo w celu identyfikacji końca sekwencji w każdym ostatnim bloku $O_k^{N_k}$ dla każdej sekwencji β_k wstawmy specjalny sygnał y_0 (rys. 2). Jeżeli wejście do bloku operacyjnego O_k w początkowej sieci działań było oznaczone stanem $a_m \in A$ to wejście bloku operacyjnego O_k^1 (początek sekwencji) w wynikowej sieci działań oznaczone jest tym samym stanem [3]. Zabieg taki powoduje, że liczba stanów w sieci działań przed i po przekształceniu jest taka sama, co stanowi zysk w porównaniu z metodą, w której przekształcona sieć działań ponownie jest znakowana stanami [6].



Rys. 2. Fragment sieci działań przed (a) i po (b) liniowym przekształceniu
Fig. 2. The subgraph of marked flow-chart before (a) and after (b) verticalization

4. Struktura automatu dla liniowo przekształconej sieci działań z adresowaniem mikrooperacji

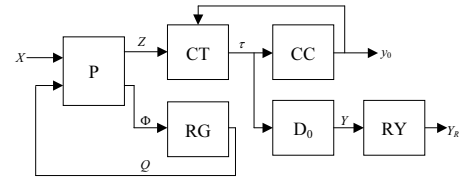
Niech mikroinstrukcja $Y_t \subseteq Y$ zawiera $M_t = |Y_t|$ mikrooperacji oznaczmy każdą taką mikrooperację $y_n \in Y$ jako y_n^t i niech one utworzą zbiór mikrooperacji $Y_0 = \{y_1^1, y_1^2, ..., y_N^T\}$ o

$$N_0 = \sum_{t=1}^{T} M_t \tag{7}$$

elementach. W takiej sytuacji mikrooperacje $y_n^t \in Y_0$ można zakodować binarnym kodem $K(y_n^t)$ na $R_2 = \lceil \log_2 N_0 \rceil$ bitach. Do reprezentacji tego kodu wykorzystane zostaną zmienne $\tau_r \in \tau = \{\tau_1, \ldots, \tau_{R_2}\}$. Podczas kodowania należy zapewnić aby mikrooperacje y_n^t i y_m^t należące do tej samej sekwencji β_k posiadały następujące po sobie kody:

$$\begin{aligned} &\text{if} \quad y_n^t = \mathrm{pr}_i \beta_k \ \text{and} \ y_m^t = \mathrm{pr}_{i+1} \beta_k \\ &\text{then} \ K\left(y_m^t\right) = K\left(y_n^t\right) + 1 \end{aligned} \tag{8}$$

W sytuacji takiej układ automatu może zostać zrealizowany w strukturze PD_{VC} (rys. 3) gdzie licznik wykorzystany jest do generowania kodów kolejnych mikrooperacji.



Rys. 3. Schemat blokowy automatu Mealy'ego o strukturze PD_{VC}
Fig. 3. Block diagram of the Mealy PD_{VC} FSM

Układ P realizuje funkcje (1) i (5), Licznik CT generuje kod $K(y_n^t)$, układ CC realizuje funkcję

$$y_0 = y_0(\tau), \tag{9}$$

która służy do ładowania licznika CT. Mikrooperacje y_n^t formowane są przez dekoder D_0

$$Y = Y(\tau). \tag{10}$$

i zapisywane w rejestrze wyjściowym RY. Należy tu wspomnieć, że rejestr wyjściowy RY wymagany jest w każdym układzie realizującym automat Mealy'ego w celu stabilizacji jego działania [1, 4].

Układ ten działa w następujący sposób:

- w początkowej chwili $\mathbf{t} = 1$ rejestr RG zawiera kod $K(a_m)$ a licznik CT kod $K(y_n^t)$ pierwszego elementu sekwencji β_k. Kod ten jest zdekodowany w dekoderze D_0 i zapisany w n-tym przerzutniku rejestru RY. Układ CC sprawdza czy został osiągnięty koniec sekwencji i w sytuacji takiej ustawia sygnał y_0;
- w chwili $\mathbf{t} = 2$ jeżeli $y_0 = 0$ to licznik generuje kod następnej mikrooperacji i stan automatu nie zostaje zmieniony. Cykl jest powtarzany do osiągnięcia końca sekwencji;
- jeżeli $y_0 = 1$ to ścieżka danych jest wykonywana. W rejestrze RG zapisywany jest kod następnego stanu $K(a_s)$ a do licznika wpisywany jest kod pierwszego elementu odpowiadającej mu sekwencji.

5. Metoda syntezy automatu PD_{VC}

W tym rozdziale zostaną omówione kroki metody syntezy automatu do struktury PD_{VC}. Punktem wejścia do procesu syntezy jest tablica przejść-wyjść automatu [4, 5]. Metoda liniowego przekształcenia stosowana jest bezpośrednio do mikroinstrukcji zapisanych w tablicy a nie do sieci działań z której została ona uzyskana.

Proponowana metoda syntezy składa się z następujących kroków:

1. Utworzenie sekwencji mikrooperacji.
   Jeżeli $Y(O_i) = Y(O_j)$ to $\beta_i = \beta_j$. Oznacza to, że wystarczy utworzyć tylko jedną sekwencje β_t dla każdej unikalnej mikroinstrukcji Y_t. Mikrooperacje $y_n \in \beta_t$ zastępowane są odpowiadającymi im mikrooperacjami $y_n^t \in Y_0$. Wszystkie sekwencje utworzą zbiór $B = \{\beta_1, ..., \beta_T\}$. Dla przykładu z tab. 1 $B = \{\beta_1, \beta_2\}$ i $\beta_1 = \langle y_1^1, y_2^1, y_3^1 \rangle$, $\beta_2 = \langle y_1^2, y_3^2 \rangle$.
2. Kodowanie mikrooperacji.
   Aby zapewnić zgodność z (8) tworzymy wektor $\beta = \beta_1 * \beta_2 * ... * \beta_T$, gdzie $*$ jest symbolem konkatenacji. Następnie i-temu elementowi wektora β przypisujemy binarny kod $K(y_n^t)$ odpowiadający wartości i-1 ($i = 1, ..., N_0$). Dla przykładu z tab. 1 $\beta = \langle y_1^1, y_2^1, y_3^1, y_1^2, y_3^2 \rangle$ i $K(y_1^1)$=000, $K(y_2^1)$=001, ..., $K(y_3^2)$=100,
3. Utworzenie tablicy przejść-wyjść automatu PD_{VC}.
   Tablica ta tworzona jest poprzez zastąpienie kolumny Y_h z początkowej tablicy przejść-wyjść kolumną Z_h. Do kolumny tej wpisuje się zmienne z_r, które są równe 1 w kodzie $K(y_n^t)$ mikrooperacji $y_n^t = \mathrm{pr}_1\beta_t$, gdzie sekwencja β_t odpowiada mikroinstrukcji Y_t z kolumny Y_h początkowej tablicy przejść-wyjść. Dla przykładu z tab. 1 jest to tab. 2.
4. Utworzenie funkcji Φ i Z.
   Funkcje te reprezentują systemy (1) i (5) i tworzone są na podstawie tablicy przejść-wyjść automatu PD_{VC}.
5. Utworzenie tablicy układu CC.
   Jest to tablica prawdy służąca do implementacji systemu (9). W układzie cyfrowym może zostać zrealizowana z wykorzystaniem pamięci ROM.
6. Utworzenie tablicy dekodera.
   Tablica ta służy do realizacji systemu (10). Należy wspomnieć, że każda mikrooperacja $y_n \in Y$ może posiadać od 1 do T kodów. Ze względu na to iż tablica dekodera i tablica układu CC posiada te same zmienne wejściowe może ona zostać zapisana w jednej tabeli. Dla przykładu z tab. 1 jest to tab. 3.

Tab. 2. Fragment tablicy przejść-wyjść automatu PD_{VC}
Tab. 2. A part of DST table of Mealy PD_{VC} FSM

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Z_h	Φ_h	h
a_3	0 1 1	a_4	1 0 0	1	-	D_1	1
a_4	1 0 0	a_5	1 0 1	x_1	-	$D_1 D_3$	2
		a_6	1 1 0	$\sim x_1$	z_1	$D_1 D_2$	3

Tab. 3. Tablicy układu CC i dekodera D_0 dla automatu PD_{VC}
Tab. 3. Table of CC circuit and D_0 decoder of Mealy PD_{VC} FSM

Y_t	y_n^t	$K(y_n^t)$	y_0	y_1	y_2	y_3	i
Y_1	y_1^1	0 0 0	0	1	0	0	1
Y_1	y_2^1	0 0 1	0	0	1	0	2
Y_1	y_3^1	0 1 0	1	0	0	1	3
Y_2	y_1^2	0 1 1	0	1	0	0	4
Y_2	y_3^2	1 0 0	1	0	0	1	5

7. Wnioski

Zaprezentowana metoda projektowania dwupoziomowego automatu skończonego z wyjściami typu Mealy'ego z liniowym przekształceniem sieci działań i zastosowaniem licznika do adresowania mikrooperacji pozwala na zmniejszenie liczby funkcji realizowanych przez kombinacyjną część automatu zależnych od warunków logicznych oraz zmiennych wewnętrznych automatu. Liniowe przekształcenie sieci działań zapewnia konieczność użycia tylko jednego dekodera do implementacji systemu mikrooperacji. Przeprowadzone badania analityczne pokazały, że proponowana metoda pozwala na zmniejszenie wymaganych, do implementacji układu w strukturach FPGA, tablic LUT.

Zaproponowana metoda syntezy oparta jest o liniowe przekształcenie sieci działań jednak operuje ona tylko na tablicy przejść-wyjść automatu. Powoduje to, że jest ona znacznie łatwiejsza w programowej implementacji podczas realizacji sytemu do automatycznej syntezy automatów. Tablica przejść-wyjść jest obecnie również bardziej popularnym formatem do reprezentacji algorytmu automatu oraz formatem wejściowym do procesu syntezy niż sieć działań.

W porównaniu z metodą syntezy gdzie nie jest stosowany licznik a wynikowa sieć działań jest ponownie znakowana stanami [6] zmniejszona zostaje liczba zmiennych potrzebnych do reprezentacji stanu. Powoduje to iż układ kombinacyjny realizuje mniejszą liczbę funkcji wzbudzeń przerzutników, w układzie użyta jest mniejsza liczba przerzutników do pamiętania stanu oraz funkcje realizowane przez układ kombinacyjny są zależne od mniejszej liczby zmiennych wewnętrznych. Z drugiej strony, ze względu iż jedna mikrooperacja może posiadać kilka kodów, układ kombinacyjny może realizować więcej funkcji potrzebnych do kodowania mikrooperacji.

Ubocznym efektem liniowego przekształcenia sieci działań jest zwiększenie liczby cykli systemu cyfrowego ze strukturą PD_{VC} potrzebnych do realizacji zadanego algorytmu sterowania.

Pracę wykonano w ramach projektu badawczego finansowanego ze środków Zintegrowanego Programu Operacyjnego Rozwoju Regionalnego (Działanie 2.6: Regionalne strategie innowacyjne i transfer wiedzy) z udziałem Europejskiego Funduszu Społecznego.

8. Literatura

[1] Adamski M., Barkalov A.: Architectural and Sequential Synthesis of Digital Devices, University of Zielona Góra Press, Zielona Góra, 2006.
[2] Adamski M., Barkalov A., Bukowiec A.: Synthesis of Control Units on Verticalized Flow-Chart, Proceedings of the 12th International Conference Mixed Design of Integrated Circuits and Systems MIXDES'05, Kraków, 2005, vol. 1, s. 209-213
[3] Adamski M., Barkalov A., Bukowiec A.: Structures of Mealy FSM Logic Circuits under Implementation of Verticalized Flow-Chart, Proceedings of IEEE East-West Design & Test Workshop EWDTW '05, Odessa (Ukraina), 2005, s. 70-74
[4] Baranov S.: Logic Synthesis for Control Automata, Kluwer Academic Publishers, 1994
[5] Barkalov A., Węgrzyn M.: Design of Control Units with Programmable Logic, University of Zielona Góra Press, Zielona Góra, 2006.
[6] Bukowiec A.: Synteza skończonych automatów stanów z zastosowaniem szeregowego przekształcenia mikroinstrukcji, Pomiary Automatyka Kontrola, 2006, nr 6 bis, s. 35-37
[7] Łuba T. (ed.): Synteza układów cyfrowych, WKŁ, Warszawa 2003.
[8] Salcic Z.: VHDL and FPLDs in Digital Systems Design, Prototyping and Customization, Kluwer Academic Publishers, 1998.

Artykuł recenzowany