Valery SALAUYOU, Irena BUŁATOWA
POLITECHNIKA BIAŁOSTOCKA,
ul. Wiejska 45A, 15-351, Białystok

Synteza hierarchicznych struktur automatów mikroprogramowalnych

Dr hab. inż. Valery SALAUYOU

Ukończył w 1978 r. studia na wydziale Matematyki Stosowanej w Białoruskim Państwowym Uniwersytecie w Mińsku. Obronił pracę doktorską w 1986 r. i uzyskał tytuł doktora habilitowanego w 2003 r. Jest adiunktem na Wydziale Informatyki Politechniki Białostockiej. Od 30 lat pracuje naukowo w dziedzinie projektowania logicznego systemów cyfrowych.

e-mail: v.salauyou@pb.edu.pl

Dr inż. Irena BUŁATOWA

Ukończyła studia na Wydziale Systemów Komputerowych Uniwersytetu Informatyki i Radioelektroniki w Mińsku na Białorusi, gdzie też obroniła pracę doktorską w 1992 r. Pracuje jako starszy wykładowca na Wydziale Informatyki Politechniki Białostockiej. Zainteresowania naukowe są związane z opracowaniem metod syntezy automatów skończonych na programowalnych układach logicznych.

e-mail: i.bulatowa@pb.edu.pl

Streszczenie

Przedstawiono metodę syntezy hierarchicznych struktur automatów mikroprogramowalnych, algorytmy sterowania których opisywane są za pomocą sieci działań. Metoda syntezy umożliwia realizację złożonych układów sterowania w postaci sieci hierarchicznie podporządkowanych automatów. Opracowany został algorytm dekompozycji sieci działań na fragmenty realizowane jako komponenty struktury hierarchicznej. Przeprowadzono badania wpływu parametrów sieci działań na możliwość oraz koszt realizacji struktury hierarchicznej.

Słowa kluczowe: automat mikroprogramowalny, sieć działań, struktura hierarchiczna, programowalne układy logiczne.

Synthesis of hierarchical structures of microprogram automata

Abstract

In this paper a method for synthesis of hierarchical structures of microprogram automata specified by the Algoritmic State Machine (ASM) charts [4] is presented. The proposed method enables the synthesis of complex control systems as a network of hierarchically subordinated automata (Fig. 1), each of which can be implemented on a separate PLD device with limited parameters. Two-level hierarchical structure can also be used to implement control algorithms with repeated fragments [6]. In this approach each repeated section is implemented in the structure only once, and is called many times during the algorithm execution. Additionally, a modified hierarchical structure that allows parallel execution of algorithm fragments is proposed (Fig. 4). The algorithm of decomposition of the ASM chart into fragments which are implemented as components of a hierarchical structure was developed. The synthesis algorithm considers limitations on the fragments size and minimizes the number of links between the different automata. The conditions the expediency of ASM decomposition into fragments to be implemented in a separate automata of the hierarchical structure are taken into consideration, too. A prerequisite for implementation of the method is decomposition of the ASM to fragments having only one input and one output, which is not always possible to fulfill. The experimental results show how the possibility of realization and the cost of implementation of the microprogram automata hierarchical structures depend on the parameters of the ASM charts.

Keywords: microprogram automaton, Algorithmic State Machine (ASM), hierarchical structure, programmable logic devices (PLD).

1. Wstęp

W przypadku projektowania bardzo złożonych układów sterowania, często ich realizacja nie jest możliwa na jednym układzie programowalnym ze względu na ograniczenia parametrów układu PLD. Synteza takich układów sterowania wymaga wykonania dekompozycji automatów z uwzględnieniem ograniczeń na zasoby układów programowalnych.

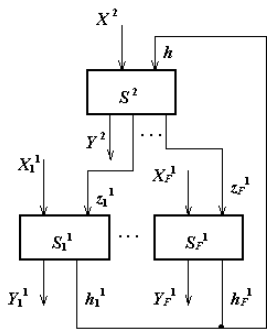
Metody dekompozycji automatów skończonych są ciągle rozwijane [1-5], oprócz zapewnienia możliwości realizacji, często są opracowywane w celu podwyższenia wydajności złożonych układów sterowania [1, 2]. W [4, 5] opisane zostały metody dekompozycji stosowane na poziomie algorytmów sterowania przedstawionych w postaci sieci działań. Umożliwiają one realizację złożonych automatów na układach PLD o ograniczonych parametrach.

W artykule zostanie przedstawiona nowa metoda syntezy automatów mikroprogramowalnych oparta na dekompozycji sieci działań na fragmenty o ograniczonych rozmiarach, z których każdy może być realizowany na oddzielnym układzie PLD. Opracowany został algorytm dekompozycji sieci działań, który uwzględnia ograniczenia na wykorzystywane zasoby PLD oraz minimalny rozmiar fragmentu, realizacja którego w postaci oddzielnego automatu ma sens.

Wyróżnione fragmenty sieci działań są realizowane w postaci automatów połączonych w strukturę hierarchiczną. Zaproponowane podejście pozwala na sekwencyjną pracę poszczególnych automatów, synchronizowaną za pomocą dodatkowych sygnałów aktywowania automatów oraz sygnałów informujących o zakończeniu ich pracy. W podejściu do syntezy hierarchicznych struktur dąży się do ograniczenia do minimum liczby powiązań pomiędzy poszczególnymi automatami.

2. Hierarchiczne struktury automatów mikroprogramowalnych

Dwupoziomowa hierarchiczna struktura automatów mikroprogramowalnych przedstawiona jest na Rys.1. Pierwszy poziom struktury tworzą automaty $S_1^1, ..., S_F^1$, którymi steruje automat drugiego poziomu S^2 . Realizacja algorytmu sterowania rozpoczyna się od uruchomienia automatu S^2. Pozostałe automaty w tym czasie znajdują się w stanie początkowym. Aby uruchomić automat niższego poziomu S_f^1, f=1..F, automat S^2 generuje sygnał z_f^1, a sam przechodzi w stan oczekiwania. Gdy automat S_f^1 zakończy pracę, generuje sygnał h_f^1, który wyprowadza automat S^2 ze stanu oczekiwania.

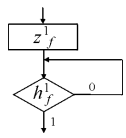


Rys. 1. Dwupoziomowa struktura hierarchiczna
Fig. 1. Two-level hierarchical structure

Realizowany jest w takim przypadku sekwencyjny algorytm sterowania, gdyż w każdym momencie czasu pracuje tylko jeden

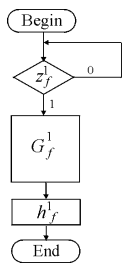
automat. Możliwe jest więc połączenie sygnałów h_1^1, ..., h_F^1 poprzez realizację sprzętowej funkcji OR i podanie na wejście automatu S^2 tylko jednego sygnału h^1, sygnalizującego zakończenie pracy automatów niższego poziomu (S_1^1, ..., S_F^1).

Niech algorytm funkcjonowania złożonego automatu mikroprogramowalnego jest opisany za pomocą sieci działań G, a automaty S_f^1, f=1..F wykonują wybrany fragment sieci działań G_f^1, który posiada jedno wejście i jedno wyjście. W celu otrzymania sieci działań G^2 realizowanej przez automat drugiego poziomu S^2, każdy fragment G_f^1, f=1..F G należy zamienić na 2 bloki: operacyjny i warunkowy oczekujący (rys. 2). Na podstawie przekształconej sieci działań zostaje stworzony automat drugiego poziomu.



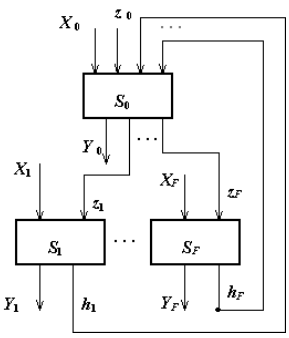
Rys. 2. Elementy sieci działań zastępujące fragment
Fig. 2. ASM elements substituting a fragment

Algorytmy działania automatów S_f^1, f=1..F należy przedstawić w postaci nowych sieci działań, gdzie na początku zostaje umieszczony blok warunkowy oczekujący, następnie wszystkie bloki fragmentu, oraz blok operacyjny generujący sygnał zakończenia pracy h_f^1. Sieć działań stworzoną z fragmentu G_f^1 obrazuje rys. 3.



Rys. 3. Sieć działań stworzona z fragmentu
Fig. 3. ASM implementing a fragment

Struktura hierarchiczna automatu może składać się z większej liczby poziomów. Jeśli sieć działań G^2 nie może być realizowana na jednym układzie PLD, to należy w niej wyróżnić fragmenty, które będą realizowane oddzielnymi automatami kolejnego poziomu. Przekształceń dokonuje się dopóki nie zostanie otrzymana sieć działań najwyższego poziomu, której realizacja będzie możliwa na jednym układzie PLD.



Rys. 4. Struktura hierarchiczna umożliwiająca równoległe wykonanie fragmentów
Fig. 4. Hierarchical structure for the parallel fragment execution

Dwupoziomowa struktura hierarchiczna może być zastosowana do realizacji sieci działań z powtarzającymi się fragmentami algorytmów [6]. W takim podejściu każdy powtarzający się fragment realizowany jest tylko jeden raz w postaci podporządkowanego automatu niższego poziomu, a wywoływany może być wielokrotnie podczas działania automatu. Automaty S_f^1, f=1..F realizują w takim układzie poszczególne egzemplarze powtarzających się fragmentów. Prowadzi to do zmniejszenia kosztów realizacji, zwłaszcza jeśli krotność powtarzania się fragmentów jest duża.

Na rys. 4 pokazano zmodyfikowaną strukturę hierarchiczną, która umożliwia równoległą pracę automatów niższego poziomu. W takiej strukturze automat nadrzędny S^0 może aktywować pracę automatów S_f, f=1..F bez konieczności oczekiwania na zakończenie pracy (h_1,...,h_f) poszczególnych automatów.

3. Algorytm syntezy struktury hierarchicznej automatów mikroprogramowalnych

Niech X^j i Y^j będą odpowiednio zbiorami zmiennych wejściowych i wyjściowych automatów j-go poziomu, j=1..J, G^j – pewna sieć działań realizowana automatem poziomu j. Algorytm syntezy struktury hierarchicznej można przedstawić w następujący sposób.

Algorytm 1:

1. Przyjmuje się j:=1, X^j:=X, Y^j:=Y, G^j:=G, gdzie X i Y –zbiory warunków logicznych i mikrooperacji.
2. Sprawdzane są warunki:

$$|Y^j| \le m, \ |X^j| \le n + m - |Y^j|, \tag{1}$$

   gdzie n, m – liczba wejść i wyjść układu PLD. Jeśli oba warunki są spełnione, następuje przejście do p.6, inaczej – przejście do p. 3.
3. W sieci działań G^j są określane fragmenty G_1^j, ..., G_f^j, które będą realizowane przez automaty poziomu j.
4. Wykonywane jest przekształcenie sieci działań G^j poprzez zamianę każdego z fragmentów G_f^j, f=1..F_j blokami: operacyjnym oraz warunkowym oczekującym. W wyniku przekształcenia powstaje nowa sieć działań G^{j+1}. Następuje również określenie zbiorów X^{j+1}, Y^{j+1}, czyli warunków logicznych i mikrooperacji G^{j+1}.
5. Przyjmuje się j:=j+1, wykonywane jest przejście do p.2.
6. Wykonywana jest realizacja automatów wszystkich poziomów na oddzielnych układach PLD.
7. Koniec algorytmu.

W celu określenia fragmentów sieci działań, należy wykonać dekompozycję sieci działań G^j tak, aby możliwa była realizacja każdego fragmentu na jednym układzie PLD. Niech U jest zbiorem bloków sieci działań G; U_t – zbiór bloków fragmentu P_t, $U_t \subseteq U$, t=1,T; $\varphi(u_i)$ i φ_t – zbiór bloków sieci działań, połączenia z których prowadzą do bloku u_i oraz do bloków zbioru U_t odpowiednio; $\psi(u_i)$ i ψ_t – zbiór bloków, do których prowadzą połączenia z bloku u_i oraz z bloków zbioru U_t; $x(u_i)$ – warunek logiczny zapisany w bloku warunkowym u_i; X – zbiór warunków zapisanych w blokach warunkowych zbioru U_t.; $y(u_i)$ i Y_t – zbiór mikrooperacji zapisanych w bloku operacyjnym u_i oraz w blokach operacyjnych zbioru U_t. Jako ξ_t jest oznaczona liczba bloków fragmentu P_t, do których prowadzą połączenia spoza fragmentu.

Fragment P_t sieci działań może być realizowany na jednym układzie PLD, jeśli spełnione są następujące warunki:

$$|Y_t| \le m\text{-}1, \ |X_t| \le n + m - |Y_t| - 2, \tag{2}$$

oprócz tego fragment P_t powinien mieć tylko jedno wejście oraz jedno wyjście:

$$\xi_t = |\psi_t| = 1 \tag{3}$$

Całkowita minimalna liczba zmiennych wejściowych i wyjściowych jest ograniczona:

$$C_{min} \le |X_t| + |Y_t|, \tag{4}$$

gdzie C_{min} jest liczbą, określającą stopień wykorzystania PLD, $C_{min} \le n + m - 2$.

Tworzenie fragmentu P_t, t=1..T, zaczyna się od wyboru bloku podstawowego. Aby otrzymać wszystkie możliwe fragmenty, jako blok podstawowy wybierany jest każdy z bloków sieci działań.

Algorytm tworzenia zbioru fragmentów sieci działań, które mogą być realizowane na jednym PLD, jest następujący:

Algorytm 2:

1. Przyjmuje się $t := 1$.
2. Blok u_i wybiera się jako podstawowy dla fragmentu P_t. Jeśli nie ma więcej bloków, następuje przejście do p. 10.
3. Blok u_i dodaje się do zbioru U_t:
   $U_t := \{u_i\}$;
   $X_t := \{x(u_i)\}$ i $Y_t := \varnothing$, jeśli blok u_i jest blokiem warunkowym;
   $Y_t := y(u_i)$ i $X_t := \varnothing$, jeśli u_i jest blokiem operacyjnym;
   $\varphi_t := \varphi(u_i) \setminus U_t$;
   $\psi_t := \psi(u_i) \setminus U_t$.
4. Sprawdzany jest warunek (2). Jeśli się nie spełnia, dalsze tworzenie fragmentu P_t nie jest możliwe, następuje przejście do p. 2 i zaczyna się tworzenie kolejnego fragmentu; w innym przypadku algorytm przechodzi do p. 5.
5. Sprawdzany jest warunek (3). Jeśli się nie spełnia, fragment ma więcej niż jedno wejście lub wyjście, ale dodanie do fragmentu kolejnych bloków może to zmienić, następuje przejście do p. 7, w innym przypadku – do p. 6.
6. Sprawdzany jest warunek (4). Jeśli warunek się nie spełnia, fragment jest zbyt prosty, kontynuuje się jego tworzenie, przechodząc do p. 7, inaczej, fragment P_t został stworzony i rozpoczyna się tworzenie kolejnego fragmentu przez dodawanie do P_t kolejnych bloków:
   $t := t+1$; $U_t := U_{t-1}$; $X_t := X_{t-1}$; $Y_t := Y_{t-1}$; $\varphi_t := \varphi_{t-1}$; $\psi_t := \psi_{t-1}$.
7. W przypadku gdy zbiór ψ_t zawiera tylko blok końcowy, wykonuje się przejście do p. 2, w innym przypadku następuje przejście do p. 8.
8. Z elementów zbioru ψ_t wybiera się blok u_j, w pierwszej kolejności wybiera się blok, który równocześnie należy do zbioru φ_t. Jeśli wybór nie jest jednoznaczny, to wybiera się blok, dla którego $|(\varphi_t \cup \varphi(u_j)) \setminus (U_t \cup \{u_j\})| + |(\psi_t \cup \psi(u_j)) \setminus (U_t \cup \{u_j\})| = \min$ (minimalnie zwiększa się liczba wejść i wyjść fragmentu).
9. Blok u_j jest dołączany do zbioru U_t, przyjmuje się:
   $U_t := U_t \cup \{u_j\}$;
   $X_t := X_t \cup \{x(u_j)\}$, jeśli u_j jest blokiem warunkowym;
   $Y_t := Y_t \cup y(u_j)$, jeśli u_j jest blokiem operacyjnym;
   $\varphi_t := (\varphi_t \cup \varphi(u_j)) \setminus U_t$;
   $\psi_t := (\psi_t \cup \psi(u_j)) \setminus U_t$;
   następuje przejście do p. 4.
10. Koniec algorytmu.

Na kolejnym etapie syntezy ze zbioru $P = \{P_1,...,P_T\}$ należy wybrać zbiór $V=\{G_1,...,G_F\}$ fragmentów, które będą realizowane automatami $S_1,...,S_F$. Przy wyborze fragmentów $G_1,...,G_F$ należy przede wszystkim minimalizować liczbę F. Kolejnym kryterium jest to, żeby zbiory X_f w miarę możliwości nie pokrywały się ze sobą oraz ze zbiorami zmiennych wejściowych innych poziomów. Te wymagania dotyczą także zbiorów Y_t. Oprócz tego liczba bloków we fragmentach $G_1,...,G_F$ powinna być maksymalna.

Do wyboru fragmentów realizowanych automatami $S_1,...,S_F$ można zastosować następujący algorytm.

Algorytm 3:

1. Przyjmuje się $V:=\varnothing$.
2. Spośród elementów zbioru P wybierany jest fragment P_t, $P_t \in P$, dla którego $|X_t|+|Y_t| = \max$ (zasoby PLD są wykorzystywane maksymalnie). Jeśli jest ich kilka, wybiera się fragment, dla którego $| X_t \cap \overline{X}_t | + | Y_t \cap \overline{Y}_t | = \min$ (minimalizacja powtórzeń zmiennych wejściowych i wyjściowych). W sytuacji gdy takich fragmentów również jest więcej niż jeden, wybiera się fragment z $|U_t| = \max$.
3. Przyjmuje się $V:=V\cup\{P_t\}$.
4. Ze zbioru P wykluczany jest fragment P_t a także fragmenty, które mają wspólne bloki z fragmentem P_t.
5. Jeśli $P = \varnothing$, następuje przejście do p. 6, w przeciwnym wypadku – przejście do p. 2.
6. Koniec algorytmu.

4. Badania eksperymentalne

Badania eksperymentalne zaimplementowanej metody syntezy struktur hierarchicznych przeprowadzono na przykładach testowych z biblioteki sieci działań [7]. Celem badań była ocena kosztów realizacji struktury hierarchicznej na układach PLD oraz określenie warunków, jakie musi spełniać sieć działań, aby była możliwa jej dekompozycja na fragmenty ograniczonego rozmiaru.

Analiza wyników badań pokazała, że duży wpływ na możliwość podziału sieci działań na fragmenty ma udział procentowy bloków warunkowych w ogólnej liczbie bloków sieci działań. W przypadku, gdy podział na fragmenty był możliwy, średni procent bloków warunkowych wynosił 47,87%, a maksymalna wartość tego parametru – 66,67%. Dla sieci działań, których dekompozycja nie była możliwa, wartość ta była dużo wyższa i średnio wynosiła 61,38%. Ważnym parametrem decydującym o możliwości podziału sieci działań na fragmenty jest współczynnik $D=DL/B$, gdzie DL – długość łańcuchów bloków warunkowych, B – ogólna liczba bloków sieci działań. Dla algorytmów, do których nie dało się zastosować metody, średnia wartość współczynnika wynosiła 0,55, natomiast dla pozostałych – 0,44.

Badania kosztów realizacji struktur hierarchicznych pokazały, że praktycznie we wszystkich przypadkach zwiększa się liczba makrokomórek użytych do realizacji automatów. Średni wzrost kosztów realizacji struktury hierarchicznej w porównaniu do struktury standardowej na układach MAX9000 wynosił 16.67%, a na układach FLEX10K firmy Altera – 18,55%.

5. Wnioski

Przedstawiono metodę syntezy hierarchicznych struktur automatów programowalnych, która pozwala na realizację złożonych układów sterowania w postaci hierarchicznej sieci automatów i oparta jest na dekompozycji sieci działań na fragmenty ograniczonego rozmiaru.

Badania pokazały, że podział sieci działań na fragmenty nie zawsze jest możliwy, w dużym stopniu zależy to od ilości bloków warunkowych sieci działań. Im jest ich więcej w stosunku do ogólnej liczby bloków, tym mniejsze jest prawdopodobieństwo, że dla sieci działań podział na fragmenty będzie możliwy. Na podstawie przykładów wykorzystanych do badań można stwierdzić, że jeśli procentowa ilość bloków warunkowych w sieci działań wynosi powyżej 67%, to dekompozycja takiej struktury na fragmenty jest praktycznie niemożliwa.

Praca została zrealizowana w ramach projektu badawczego S/WI/4/08.

6. Literatura

[1] Ashar P., Devadas S., Newton A.R.: Optimum and heuristic algorithms for an approach to finite state machine decomposition. IEEE Transactionsons on Computer-Aided Design of Integrated Circuits and Systems, 1991, V.10, No. 3, pp. 296-310.

[2] Lam K., Devadas S.: Performance-oriented decomposition of sequential circuits. IEEE International Symposium on Circuits and Systems, 1990, V. 4, pp. 2642-2645.

[3] Kuo M.T., Liu L.T., Cheng C.K.: Finie state machine partitioning for I/O-based design. VLSI Technology, Systems, and Applications, 1995, pp. 68-72.

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

[5] Baranov S.: Logic and System Design of Digital Systems. TTU Press and SiB Publishers, Tallinn, 2008.

[6] Salauyou W., Klimowicz A.: Synteza logiczna układów cyfrowych w strukturach programowalnych. Oficyna Wydawnicza Politechniki Białostockiej, 2010.

[7] Baranov S.: High level synthesis in EDA tool "Abelite". Electronics and Telecommunications Quarterly, 2009, Vol. 55, No 2, pp. 123-156.

otrzymano / received: 11.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*