Adam KLIMOWICZ
POLITECHNIKA BIAŁOSTOCKA, WYDZIAŁ INFORMATYKI
ul. Wiejska 45a, 15-351 Białystok

Minimalizacja automatów skończonych z uwzględnieniem ich kosztu realizacji w układach programowalnych o strukturze CPLD

Dr inż. Adam KLIMOWICZ

Ukończył studia w Instytucie Informatyki Politechniki Białostockiej. Obronił rozprawę doktorską w 2007 r. na Wydziale Informatyki Politechniki Białostockiej, gdzie zajmuje stanowisko adiunkta. Jego zainteresowania naukowe to synteza układów kombinacyjnych i sekwencyjnych na bazie programowalnych układów logicznych, ze szczególnym uwzględnieniem układów o strukturze PLD/CPLD.

e-mail: a.klimowicz@pb.edu.pl

Streszczenie

W pracy opisano heurystyczną metodę minimalizacji nie w pełni określonych automatów skończonych, która pozwala już na etapie minimalizacji stanów wewnętrznych uwzględniać parametry bazy technologicznej, metodę kodowania stanów oraz optymalizować koszt realizacji automatu w strukturze programowalnej. Opisano kryteria minimalizacji liczby stanów automatu ze względu na koszt ich realizacji w strukturze CPLD, gdzie głównym parametrem wpływającym na realizację jest liczba termów podłączonych do makrokomórki. Dodatkowym efektem działania metody jest minimalizacja liczby przejść automatu.

Słowa kluczowe: automaty skończone, synteza logiczna, minimalizacja, CPLD.

Minimization of finite state machines taking into account the cost of realization in CPLD devices

Abstract

In the paper a heuristic method of minimization of incompletely specified finite state machines is described. This method allows taking into account parameters of technological base, the method of state assignment and realization costs. The presented method is focused on realization of an FSM in the CPLD structure. The method is based on an operation of merging two states. In addition to reducing internal states, this method minimizes the number of FSM transitions and FSM input variables. In contrast to the previously developed methods, in each step of the algorithm there is considered not only one, but the entire set of all pairs of states for which it is permissible to merge. Then the pair of states which best matches the criteria of minimizing is selected from the set. Two FSM states can be merged if they are equivalent. FSM behavior does not change after the states are merged, if the transition conditions from these states that lead to different states are orthogonal. If there are transitions from the states that lead to the same states, the transition conditions for such transitions should be equal. Moreover, the output vectors generated in these states should not be orthogonal. It should be noted that wait states can be formed at the merging of FSM states. This paper describes the criteria for minimizing the number of states of the machine because of the cost of their implementation in the CPLD structure, where the main parameter influencing the implementation is a number of terms connected to one macrocell.

Keywords: finite state machines, logic synthesis, minimization, CPLD.

1. Wstęp

Automat skończony jest modelem matematycznym szeroko wykorzystywanym przy projektowaniu różnych systemów cyfrowych takich, jak układy sekwencyjne, kontrolery, mikroprocesorowe systemy sterowania, systemy telekomunikacyjne. Często przy projektowaniu nakładane są ograniczenia na różne parametry systemu. Jednym z takich parametrów jest koszt realizacji.

Tradycyjne podejście do syntezy automatów skończonych składa się z następujących, kolejno wykonywanych etapów: minimalizacji liczby stanów, kodowania stanów oraz syntezy części kombinacyjnej automatu. Jednak takie podejście często utrudnia zadanie optymalizacji automatu na etapie syntezy logicznej, ponieważ zarówno minimalizacja liczby stanów, jak i ich kodowanie nie uwzględnia cech szczególnych bazy technologicznej oraz wymagań stawianych przez zadania syntezy logicznej, na przykład, ograniczenia stawiane parametrom projektowanego urządzenia. Jednym z takich parametrów może być koszt realizacji.

Pierwsze próby połączenia dwóch procedur (minimalizacji i kodowania stanów) zaproponowano w [1-3]. W pracy [1] zadanie jest rozpatrywane dla automatów asynchronicznych, przy czym minimalizuje się tam długość kodu stanu. Metoda [2] pracuje jedynie dla niewielkich automatów, o liczbie stanów niewiększej niż 10. W pracy [3] przedstawiono program do minimalizacji i kodowania stanów pozwalający formować kody stanów z wartościami nieokreślonymi.

W pracach [4-9] przy kodowaniu stanów wewnętrznych rozwiązywane jest zadanie minimalizacji zajmowanej powierzchni (kosztu realizacji) oraz pobieranej mocy. W większości tych prac [5-7, 9] zastosowano algorytmy genetyczne. W pracy [4] zadanie rozwiązano dla układów dwupoziomowych realizowanych w strukturach PLA (Programmable Logic Array). W pracy [8] wykorzystano procedurę rozszczepienia stanów, przy czym zauważono, że dla rozwiązania przedstawionego zadania minimalna liczba stanów wewnętrznych automatu nie jest wymagana.

Przedstawiona analiza pokazuje, że brak jest prac, w których jeszcze na etapie minimalizacji stanów wewnętrznych jest wykonywana optymalizacja zajmowanej powierzchni oraz innych parametrów.

W niniejszej pracy przedstawiono opisano heurystyczną metodę minimalizacji nie w pełni określonych automatów skończonych, która pozwala już na etapie minimalizacji stanów wewnętrznych uwzględniać parametry bazy technologicznej, metodę kodowania stanów oraz optymalizować koszt realizacji automatu w strukturze programowalnej. Przedstawione podejście jest zorientowane na realizację automatów skończonych w układach programowalnych o strukturze dwóch matryc - CPLD.

2. Idea proponowanego podejścia

W przedstawionym podejściu, minimalizacja liczby stanów wewnętrznych odbywa się droga kolejnego sklejania dwóch stanów, podobnie jak w pracy [10]. W odróżnieniu od metody pokazanej w [10] w tym podejściu proponuje się na każdym kroku rozpatrywać nie jedna parę stanów, ale zbiór G wszystkich par stanów, które dopuszczają sklejanie. Następnie ze zbioru G wybiera się taką parę stanów, która w największym stopniu odpowiada kryteriom minimalizacji z punktu widzenia kosztu realizacji oraz dalszej minimalizacji liczby stanów wewnętrznych automatu.

Nie w pełni określony automat skończony może zawierać wektory wyjściowe, w których występują wartości nieokreślone. Ma to miejsce, kiedy wartość zmiennej wyjściowej w danej chwili nie wpływa na funkcjonowanie sterowanego obiektu.

Niech $X = \{x_1,...,x_L\}$ – zbiór zmiennych wejściowych, $Y = \{y_1,...,y_N\}$ – zbiór zmiennych wyjściowych, $A = \{a_1,...,a_M\}$ – zbiór stanów wewnętrznych automatu skończonego, $D = \{d_1,\ldots,d_R\}$ – zbiór funkcji wzbudzeń elementów pamięci, gdzie R – liczba elementów pamięci (liczba bitów kodu automatu), $R \in [\text{intlog}_2 M, M]$. Funkcjonowanie automatu będzie opisane za pomocą listy przejść, która jest tablicą złożoną z czterech kolumn: a_m, - stan bieżący, $X(a_m,a_s)$ – warunki przejścia, a_s – stan następny i $Y(a_m,a_s)$ – wektor zmiennych wyjściowych generowany podczas przejścia.

Zostaną wprowadzone następujące oznaczenia: $Z(a_i)$ – zbiór przejść ze stanu a_i; $C(a_i)$ – zbiór przejść do stanu a_i; $A(a_i)$ – zbiór stanów, do których wykonywane są przejścia ze stanu a_i, $W(a_i)$ – zbiór wektorów zmiennych wyjściowych formowanych na przejściach ze stanu a_i, $a_i \in A$.

Niech z_h – pewne przejście ze stanu a_i, $z_h \in Z(a_i)$, $a_i \in A$. Warunek przejścia jest zapisywany w postaci koniunkcji zmiennych wejściowych, można też go oznaczyć przy pomocy wektora, np. "1-10-0", gdzie '1' – oznacza, że odpowiadająca zmienna wejściowa wchodzi do koniunkcji w postaci prostej, '0' – w zanegowanej, a kreska oznacza, że wartość zmiennej nie wpływa na dane przejście. Z warunku determinizmu funkcjonowania automatu skończonego wynika, że warunki przejścia z każdego stanu automatu powinny być wzajemnie ortogonalne. Dwa warunki przejścia są ortogonalne, jeżeli chociażby na jednej pozycji mają one różne wartości znaczące (0 lub 1).

Dwa stany wewnętrzne automatu skończonego a_i i a_j mogą zostać sklejone, tzn. zastąpione jednym stanem a_{i_j}, jeśli są one równoważne. Równoważność stanów wewnętrznych automatu oznacza, że funkcjonowanie automatu nie zmienia się w rezultacie połączenia tych stanów w jeden. Funkcjonowanie automatu przy sklejeniu stanów będzie identyczne, jeżeli warunki przejść ze stanów a_i i a_j, które prowadzą do różnych stanów będą ortogonalne. Jeżeli istnieją przejścia, które ze stanów a_i i a_j wiodą do tego samego stanu, to warunki przejść dla takich stanów powinny być jednakowe. Oprócz tego wartości zmiennych wyjściowych formowanych na przejściach w jednakowe stany nie powinny być ortogonalne. Można także zauważyć, że przy sklejaniu dwóch stanów mogą pojawić się także stany oczekiwania.

Przy sklejaniu stanów wewnętrznych wektory wyjściowe z wartościami nieokreślonymi mogą łączyć się tylko, jeżeli nie są one ortogonalne. Przy czym określone wartości (0 i 1) wektorów są pozostawiane bez zmian, a wartości nieokreślone przyjmują wartość (0 lub 1) odpowiednio do wartości wektorów. Na przykład, wektory „1-0-0" i „-1010" można zastąpić wektorem „11010".

Główna strategia przedstawionego podejścia polega na znalezieniu zbioru G wszystkich par stanów, dla których są spełnione warunki sklejania. Następnie dla każdej pary stanów ze zbioru G wykonuje się próbne sklejanie stanu. Ostatecznie do sklejania wybiera się parę, która w największym stopniu spełnia kryteria optymalizacji pod względem kosztu realizacji oraz zapewnia największe możliwości sklejania innych par stanów ze zbioru G. Proces ten powtarza się dopóki można skleić chociaż jedną parę stanów automatu skończonego.

3. Warunki sklejania dwóch stanów

Warunkiem koniecznym i dostatecznym możliwości sklejania dwóch stanów a_i i a_j, $a_i, a_j \in A$, jest identyczność działania automatu przed i po sklejeniu tych stanów.

W sytuacji, gdy $A(a_i) \cap A(a_j) = \emptyset$, wszystkie przejścia ze stanów a_i i a_j wiodą do różnych stanów. W tym przypadku warunkiem możliwości sklejania stanów a_i i a_j jest wzajemna ortogonalność elementów zbiorów $Z(a_i) \cup Z(a_j)$. Warunek ten można zapisać następująco: dla dowolnego $X(a_i,a_h)$, $a_h \in A(a_i)$ i dowolnego $X(a_j,a_t)$, $a_t \in A(a_j)$ spełnione jest:

$$X(a_i,a_h) \perp X(a_j,a_t), \tag{1}$$

gdzie symbol „ $\perp$ " oznacza ortogonalność warunków.

Można zauważyć, że w tym przypadku wartości wektorów wyjściowych formowanych na przejściach ze stanów a_i i a_j, tzn. wartości zbiorów $W(a_i)$ i $W(a_j)$, nie wpływają na warunki możliwości sklejania.

W przypadku, gdy przejścia ze stanów a_i i a_j wiodą do tego samego zbioru stanów $A(a_i) = A(a_j)$, stany te mogą zostać sklejone, wtedy i tylko wtedy, gdy warunki przejść w te same stany są identyczne, a także wektory wyjściowe formowane na przejściach w jednakowe stany nie są ortogonalne. Warunek ten można zapisać w następujący sposób: dla dowolnego a_h, $a_h \in A(a_i)$ istnieje taki $X(a_j,a_h) \in Z(a_j)$, że:

$$X(a_i,a_h) = X(a_j,a_h) \text{ i } Y(a_i,a_h) \neg\perp Y(a_j,a_h), \tag{2}$$

oraz dla dowolnego a_t, $a_t \in A(a_j)$ istnieje taki $X(a_i,a_t) \in Z(a_i)$, że:

$$X(a_j,a_t) = X(a_i,a_t) \text{ i } Y(a_j,a_t) \neg\perp Y(a_i,a_t), \tag{3}$$

gdzie symbol „$\neg$" oznacza zaprzeczenie.

W sytuacji gdy $A(a_i) \neq A(a_j)$ i $A(a_i) \cap A(a_j) \neq \emptyset$, przejścia ze stanów a_i i a_j mogą prowadzić do zarówno do różnych jak i tych samych stanów. Niech $A' = A(a_i) \cap A(a_j)$; $A'(a_i) = A(a_i) \setminus A'$ i $A'(a_j) = A(a_j) \setminus A'$. W takim przypadku stany a_i i a_j mogą zostać sklejone wtedy i tylko wtedy gdy warunki (2) i (3) są spełnione dla przejść do stanów ze zbioru A', a warunek (1) jest spełniony dla przejść do stanów ze zbiorów $A'(a_i)$ i $A'(a_j)$.

Stanem oczekiwania określa się taki stan wewnętrzny automatu skończonego, z którego pewne przejście (lub kilka przejść) prowadzi do tego właśnie stanu. Wiadomym jest, że jeśli przy sklejaniu dwóch stanów jeden z nich lub oba są stanami oczekiwania, to stan po sklejeniu także będzie stanem oczekiwania. Przy sklejaniu dwóch stanów a_i i a_j pojawi się stan oczekiwania, gdy jeden ze stanów a_i lub a_j (lub oba stany) należą do zbioru $A(a_i) \cup A(a_j)$.

Jeżeli po sklejeniu stanów pojawia się stan oczekiwania, gdzie jednakowe warunki przejść ze stanów a_i i a_j wiodą do różnych stanów, wówczas takie sytuacje należy rozpatrywać oddzielnie. Innymi słowami, jeżeli $X(a_i,a_h) = X(a_j,a_t)$ i $Y(a_i,a_h) \neg\perp Y(a_j,a_t)$, ale $a_h \neq a_t$, to stany a_i i a_j mogą zostać sklejone jeżeli $a_h = a_i$ i $a_t = a_j$ lub $a_h = a_j$ i $a_t = a_i$. Proces sklejania stanów oraz powstawanie stanów oczekiwania dokładniej opisano w pracy [10].

Idea algorytmu budowy zbioru G wszystkich par stanów możliwych do sklejenia polega na kolejnym rozpatrywaniu wszystkich możliwych par stanów (a_i, $a_j \in A$, $a_i \neq a_j$). Jeżeli dla danej pary spełnione są warunki sklejania lub spełnione są warunki formowania stanu oczekiwania to para (a_i, a_j) jest dołączana do zbioru G. Można to zrobić korzystając z poniższego algorytmu.

Algorytm 1

1. Podstawia się $G := \emptyset$.
2. Kolejno rozpatruje się wszystkie pary stanów (a_i, a_j) ze zbioru A, $a_i \neq a_j$, $a_i, a_j \in A$. Jeśli rozpatrzono wszystkie pary wykonuje się przejście do punktu 11.
3. Dla stanów a_i, a_j określa się zbiory $Z(a_i)$ i $Z(a_j)$.
4. Dla zbiorów $Z(a_i)$ i $Z(a_j)$ kolejno rozpatruje się pary ($X(a_i,a_h)$, $X(a_j,a_t)$) warunków przejść ($X(a_i,a_h) \in Z(a_i)$, $X(a_j,a_t) \in Z(a_j)$). Jeśli wszystkie pary rozpatrzono, to dla stanów a_i, a_j spełnione są warunki sklejania, podstawia się $G := G \cup (a_i,a_j)$ i wykonuje przejście do punktu 2.
5. Sprawdza się warunki ortogonalności $X(a_i,a_h)$ i $X(a_j,a_t)$. Jeśli warunki przejść są ortogonalne, należy przejść do punktu 4.
6. Jeśli $X(a_i,a_h) \neq X(a_j,a_t)$, to nie są spełnione warunki sklejania, należy przejść do punktu 2.
7. Jeżeli $X(a_i,a_h) = X(a_j,a_t)$, określa się wektory wyjściowe $Y(a_i,a_h)$ i $Y(a_j,a_t)$, formowane na przejściach , które są inicjowane odpowiednio warunkami $X(a_i,a_h)$ i $X(a_j,a_t)$.
8. Jeśli wektory wyjściowe są ortogonalne $Y(a_i,a_h) \perp Y(a_j,a_t)$, wykonuje się przejście do punktu 2.
9. W tej sytuacji $X(a_i,a_h) = X(a_j,a_t)$ i $Y(a_i,a_h) \neg\perp Y(a_j,a_t)$. Jeżeli $a_h = a_t$, należy przejść do punktu 4.
10. W tej sytuacji $X(a_i,a_h) = X(a_j,a_t)$ i $Y(a_i,a_h) \neg\perp Y(a_j,a_t)$, ale $a_h \neq a_t$. Należy sprawdzić warunki powstawania stanu oczekiwania, jeśli są one spełnione – należy przejść do punktu 4. W przeciwnym wypadku dla stanów a_i i a_j nie są spełnione warunki sklejania. Należy przejść do punktu 2.
11. Koniec.

4. Ogólny algorytm minimalizacji liczby stanów automatu

Niech C_{st} – będzie parametrem oceny kosztów realizacji automatu skończonego w rezultacie sklejenia pary stanów (a_s,a_t) ze zbioru G. Wówczas, uwzględniając powyższe założenia, algorytm minimalizacji liczby stanów ma następującą postać.

Algorytm 2

1. Za pomocą algorytmu 1 określa się zbiór G par stanów, które można skleić. Jeżeli $G= Ø$ należy przejść do punktu 5
2. Dla każdej pary stanów (a_s,a_t) ze zbioru G określa się wartość parametru C_{st}.
3. Spośród wszystkich par stanów ze zbioru G wybiera się parę (a_i,a_j), dla której C_{ij} = min.
4. Wykonuje się sklejanie pary (a_i,a_j), wykonuje się przejście do punktu 1.
5. Wykonuje się minimalizację liczby przejść automatu [10].
6. Wykonuje się minimalizację liczby zmiennych wejściowych automatu [10].
7. Koniec.

5. Kryterium wyboru pary stanów

Przy określaniu kosztu realizacji automatu skończonego w układzie programowalnym nie uwzględnia się realizacji pamięci automatu, a jedynie koszt realizacji części kombinacyjnej. Rzecz w tym, że zarówno makrokomórki CPLD jak i elementy logiczne FPGA dopuszczają konfigurację z kombinacyjnym i rejestrowym wyjściem. Dlatego też, aby dołączyć do automatu przerzutnik, należy tylko odpowiednio skonfigurować wyjście makrokomórki, na której jest realizowana funkcja wzbudzenia elementu pamięci. Nie wymaga to żadnych dodatkowych zasobów układu CPLD, więc koszt realizacji układu nie zwiększa się. Należy zauważyć, że przed określeniem kosztu realizacji można dokonać minimalizacji zbioru funkcji boolowskich, odpowiadającego części kombinacyjnej automatu za pomocą metody, która będzie wykorzystana przy syntezie.

Architektura CPLD jest zbiorem bloków funkcjonalnych, z których każdy składa się z dwóch programowalnych matryc AND i OR. Wyjścia matrycy AND są połączone z wejściami matrycy OR i nazywane są termami. Zwykle liczba wejść bloków funkcjonalnych CPLD jest dostatecznie duża (22-36) i zwykle przewyższa liczbę argumentów funkcji realizowanych w kombinacyjnej części automatu skończonego. Krytycznym parametrem dla CPLD jest liczba termów podłączonych do jednej makrokomórki. Średnia wartość tego parametru wynosi zazwyczaj od 4 do 8 (najczęściej 4). Dlatego przy określaniu kosztu realizacji pewnej funkcji w_i, $w_i \in W$, uwzględnia się tylko ograniczenie na średnią liczbę termów q, podłączonych do jednej makrokomórki. Uwzględniając powyższe założenia algorytm oceny kosztu realizacji automatu skończonego jest następujący.

Algorytm 3

1. Określa się koszt realizacji $C(w_i)$ funkcji $w_i \in W$.
2. W przypadku realizacji automatu skończonego w strukturze CPLD, koszt realizacji określa się następująco:

$$C(w_i) =] Q(w_i)/q[, \tag{4}$$

gdzie $Q(w_i)$ – liczba składników (mintermów) w alternatywnej postaci normalnej funkcji w_i; $]A[$ - najmniejsza liczba całkowita większa lub równa A.

3. Określa się parametr oceny kosztu realizacji automatu skończonego:

$$C_{st} = \sum_{w_i \in W} C(w_i) \tag{5}$$

4. Koniec.

Należy zauważyć, że podczas oceny kosztów realizacji automatu skończonego można uwzględniać różne metody kodowania: binarne, unarne (one-hot) lub inne, określone za pomocą wybranej metody syntezy automatu.

6. Podsumowanie

W pracy przedstawiono nową metodę minimalizacji liczby stanów automatu skończonego. Pozwala ona już na pierwszym etapie uwzględniać parametry docelowego układu programowalnego w celu optymalizacji kosztu realizacji automatu. Metoda jest przeznaczona dla układów CPLD, więc kolejnym etapem prac będzie dostosowanie jej do potrzeb układów FPGA.

Innym możliwym kierunkiem rozwoju przedstawionego podejścia może być uwzględnianie innych kryteriów przy łączeniu stanów automatu skończonego, takich jak szybkość działania oraz pobór mocy.

Artykuł opracowano w ramach pracy statutowej S/WI/1/2013 Politechniki Białostockiej.

7. Literatura

[1] Hallbauer G., Procedures of state reduction and assignment in one step in synthesis of asynchronous sequential circuits, Proc. of Int. IFAC Symp. Discrete Syst., 1974, pp. 272–282.
[2] Lee E.B., Perkowski M., Concurrent minimization and state assignment of finite state machines, Proc. of Int. Conf. Syst., Man, Cybern., 1984, pp. 248–260.
[3] Avedillo M. J. et al., SMAS: A program for concurrent state reduction and state assignment of finite state machines, Proc. of ISCAS, 1991, pp. 1781–1784.
[4] Wang K.H. et al., State assignment for power and area minimization, Proc. of ICCD: International Conference on Computer Design, 1994, pp. 250–254.
[5] Xia Y., Almaini A.E.A., Genetic algorithm based state assignment for power and area optimization, IEE Proceedings on Computer and Digital Techniques 149 (4) (2002) 128–133.
[6] Chattopadhyay S., Yadav P., Singh R.K. Multiplexer Targeted' Finite State Machine Encoding for Area and Power Minimization, Proc. of IEEE India Annual Conference, 2004, pp. 12-16.
[7] Aiman M., Sadiq S.M., Nawaz K.F. Finite state machine state assignment for area and power minimization, Proc. of IEEE International Symposium on Circuits and Systems (ISCAS), 2006, 21-24 May 2006, IEEE Inc., pp. 5303-5306.
[8] Yuan L., Qu G., Villa T., Sangiovanni-Vincentelli A. An FSM Reengineering Approach to Sequential Circuit Synthesis by State Splitting, IEEE Trans. on CAD, V. 27, No. 6, 2008, pp.1159-1164.
[9] Chaudhury S., KrishnaTejaSistla K.T., Chattopadhyay S. Genetic algorithm-based FSM synthesis with area-power trade-offs,/ INTEGRATION, the VLSI journal 42 (2009) 376–384.
[10] Klimovich A.S., Solov'ev V.V. Minimization of Mealy finite-state machines by internal states gluing, Journal of Computer and Systems Sciences International, 2012, Vol 51. No. 2, pp. 244-255.

otrzymano / received: 21.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*