Valery SALAUYOU, Tomasz GRZEŚ
POLITECHNIKA BIAŁOSTOCKA, WYDZIAŁ INFORMATYKI

Minimalizacja poboru mocy wspólnego modelu automatów skończonych

Prof. dr hab. inż. Valery SALAUYOU

Ukończył w 1978 r. studia na Wydziale Matematyki Stosowanej w Białoruskim Państwowym Uniwersytecie w Mińsku. W 1986 r. obronił pracę doktorską, a w 2003 r. uzyskał tytuł doktora habilitowanego. Od 25 lat pracuje w dziedzinie projektowania logicznego systemów cyfrowych.

e-mail: walsol@wi.pb.edu.pl

Mgr inż. Tomasz GRZEŚ

Ukończył studia w Instytucie Informatyki Politechniki Białostockiej w 1999 r. Od 2000 r. jest asystentem w Instytucie Informatyki, a następnie na Wydziale Informatyki Politechniki Białostockiej. Jego zainteresowania naukowe to problemy minimalizacji mocy w układach cyfrowych opartych o struktury programowalne.

e-mail: grzes@wi.pb.edu.pl

Streszczenie

W artykule przedstawiono nowy algorytm kodowania stanów wewnętrznych automatu skończonego o obniżonym poborze mocy. Zastosowano w nim wspólny model automatu klas ADE co pozwoliło to na zmniejszenie ilości przerzutników przechowujących kod stanu. Badania symulacyjne przeprowadzone z wykorzystaniem standardowych układów testowych potwierdziły skuteczność kodowania z wykorzystaniem proponowanego algorytmu w porównaniu z algorytmami JEDI oraz NOVA, jak i zawartymi we wcześniejszych pracach autorów.

Słowa kluczowe: automat skończony, minimalizacja poboru mocy.

Minimisation of power dissipation of FSM common model

Abstract

In this paper there is addressed the problem of power minimisation of the finite state machine (FSM). Power reduction is of great importance in design of digital systems as it can improve the speed and extend the time between recharging the batteries in mobile systems. In the common model of the FSM of class ADE (Section 2) the set A of internal states consists of three subsets: A_A, A_D, and A_E. A_A is the set of internal states of the FSM of class A, A_D is the set of internal states of the FSM of class D (the output vector is identical to the next state code), and A_E is the set of internal states of the FSM of class E (the input vector is identical to the next state code) [12]. The common model of the FSM of class ADE requires an additional register used for storing the input and output vector values. These registers are present in modern programmable logic devices. In Section 3 there is proposed a new algorithm of the FSM state assignment that makes use of the common model. The assigned code consists of three parts: G – input vector, Z – output vector and E – state code. G and Z are stored in the input and output registers, respectively. With this algorithm it is possible to assign codes that are shorter than those assigned with use of classical methods, and thus less power is dissipated in registers storing the current state code during every transition. The experimental results (Section 4, Tables 1 and 2) show the significant reduction (of 13 to 51%) in power dissipation compared to classic (JEDI, NOVA, column-based) and recent (sequential and iterating) algorithms.

Keywords: finite state machine, low-power design.

1. Wstęp

Problem minimalizacji poboru mocy układów cyfrowych jest szczególnie istotny w dobie szybkiego rozwoju technologii mobilnych. Zredukowanie wartości prądu pobieranego przez urządzenie ze źródła zasilania, jakim jest akumulator, pozwala na wydłużenie czasu pracy pomiędzy ładowaniami. Prowadzi również do zwiększenia wydajności i szybkości pracy systemu.

Synteza układów cyfrowych o obniżonym poborze mocy (*low-power design*) należy do jednego z lepiej opracowanych problemów naukowych. Rozwój badań wiązał się z wykorzystaniem coraz bardziej zaawansowanych algorytmów do realizacji kodowania minimalizującego pobór mocy. Pierwsze prace (np. [1]) definiowały zadanie jako problem całkowitoliczbowego programowania liniowego (ILP – *Integer Linear Programming*). Jednakże ze względu na złożoność jego rozwiązanie wymaga zastosowania technik heurystycznych. W późniejszych pracach stosowane były różne podejścia, takie jak m.in. kodowanie kolumnowe [1], wyżarzanie (wygrzewanie) statystyczne [5, 6], algorytmy zachłanne (w tym m.in. algorytm iteracyjny [13]), aby wreszcie wykorzystać algorytmy genetyczne [2].

Wymienione algorytmy bazują na ogólnym modelu automatu skończonego (zarówno Mealy'ego, jak i Moore'a), przez co nie są optymalne. Pozwalają one na redukcję mocy do pewnego poziomu minimalnego, poniżej którego nie da się zejść. Ogromna popularność układów programowalnych (CPLD/FPGA) powoduje, że istnieje potrzeba wykorzystania szczególnych cech tych struktur do rozwiązania problemu syntezy układów o zredukowanym poborze mocy. Dalsze obniżenie mocy wymaga zastosowania nowego podejścia do problemu syntezy, jakim niewątpliwie jest zastosowanie wspólnego modelu automatu skończonego klas ADE.

Proponowana metoda opiera się na zaproponowanych wcześniej nowych modelach automatów skończonych [7-11], a w szczególności wykorzystaniu klas A, D i E automatów, połączonych w jednej strukturze. Charakteryzuje się ona wykorzystaniem w znacznym stopniu możliwości współczesnych struktur programowalnych, a w szczególności możliwości wykorzystania przy realizacji automatu skończonego przerzutnika umieszczonego w pętli sprzężenia zwrotnego. Pozwala to na zmniejszenie długości kodów stanów wewnętrznych.

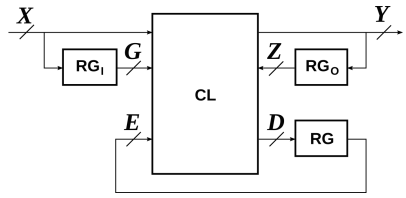
2. Wspólny model klas ADE

Niech automat skończony będzie opisany za pomocą tablicy przejść składającej się z czterech kolumn a_m, a_s, X i Y, gdzie a_m – stan obecny automatu, a_s – stan następny, X – wektor wejściowy, zaś Y – wektor wyjściowy. Jeden wiersz w tablicy przejść odpowiada jednemu przejściu automatu.

We wspólnym modelu automatu skończonego zbiór stanów wewnętrznych jest rozpatrywany jako suma podzbiorów A_A, A_D i A_E odnoszących się do stanów wewnętrznych automatów klas A, D i E [12]. Jest on automatem Mealy'ego, co oznacza, iż wartość funkcji wyjść jest uzależniona zarówno od stanu automatu, jak i wartości wektora wejściowego. Wykorzystane modele klas A, D oraz E można opisać w następujący sposób:

– automat klasy A (klasyczny automat Mealy'ego): $a_{t+1} = \varphi(z_t, a_t)$, $w_t = \psi(z_t, a_t)$;
– automat klasy D (każdy wektor wyjściowy jest taki sam, jak odpowiadający mu kod stanu następnego automatu): $a_{t+1} = \varphi(z_t, a_t)$, $w_t = a_{t+1}$;
– automat klasy E (każdy wektor wejściowy jest taki sam, jak odpowiadający mu kod stanu następnego automatu): $a_{t+1} = z_t$, $w_t = \psi(z_t, a_t)$.

gdzie: a_t – stan bieżący automatu, a_{t+1} – stan następny automatu, w_t – wektor wyjściowy automatu, z_t – wektor wejściowy automatu, φ – funkcja przejść, ψ – funkcja wyjść.



Rys. 1. Schemat modelu wspólnego automatu klas ADE
Fig. 1. Common model of finite state machine of ADE class

Struktura wspólnego modelu automatu wynikająca z powyższych założeń została przedstawiona na rys. 1. W porównaniu z klasycznym modelem automatu [3] można zauważyć dodatkowe rejestry przechowujące wartości zmiennych wejściowych (RG_I) oraz wyjściowych (RG_O). Zawartość rejestru RG_I jest określona wartościami wektorów wejściowych $X = \{x_1, ..., x_L\}$. Rejestr RG_O przyjmuje wartości wektorów wyjściowych $Y = \{y_1, ..., y_N\}$. Rejestr RG przechowuje wartości określone przez funkcję przejść ze zbioru $D = \{d_1, ..., d_R\}$. Kody stanów wewnętrznych określone są za pomocą trzech zbiorów: $G = \{g_1, ..., g_L\}$, $E = \{e_1, ..., e_R\}$, $Z = \{z_1, ..., z_N\}$. Wartości ze zbiorów G, E oraz Z są przechowywane w rejestrach odpowiednio: RG_I, RG oraz RG_O.

3. Algorytm kodowania stanów

Podstawowym problemem podczas kodowania stanów wewnętrznych automatu skończonego jest zachowanie ortogonalności kodowania, tzn. kody wszystkich stanów muszą być różne. Częściowe rozwiązanie można uzyskać wykorzystując zmienne ze zbiorów G oraz Z. Do pełnego rozwiązania konieczne może być wprowadzenie dodatkowego zbioru E. Oddzielne grupy stanów kodowane są wtedy w taki sposób, aby uzyskać jak najmniejszą moc.

Pierwszym etapem jest utworzenie macierzy W, której kolumny odpowiadają poszczególnym bitom wektorów ze zbiorów G oraz Z, natomiast wiersze odpowiadają stanom automatu skończonego. Wartości macierzy określają, czy dany bit (odpowiadający kolumnie) wchodzi w warunek przejścia w postaci prostej (1), zanegowanej (0), czy nie wchodzi w warunek przejścia (–) dla danego stanu [12].

Wiersze macierzy W nie zawsze są ortogonalne. Zapewnienie ortogonalności może wymagać wprowadzenia pewnej liczby R dodatkowych bitów rejestru RG. Wymaga to wydobycia z macierzy W grup stanów, których kody są wzajemnie ortogonalne i przeprowadzenie dodatkowego kodowania tych grup.

Do wykonania tego zadania można wykorzystać graf ortogonalności H, w którym wierzchołki i oraz j są połączone krawędzią, gdy w macierzy W wiersze i oraz j są wzajemnie ortogonalne. Graf H można podzielić na T podgrafów H_t. W każdym z podgrafów znajdują się stany, dla których wiersze macierzy W są wzajemnie ortogonalne. Stany znajdujące się w każdym z podgrafów H_t tworzą wierzchołki nowego grafu Q, który pozwoli na utworzenie kodów cząstkowych E. Graf Q składa się z T wierzchołków, gdzie T jest ilością grup wzajemnie ortogonalnych stanów. Wartości wag q_{ij} krawędzi grafu Q należy policzyć korzystając z następującej zależności:

$$q_{ij} = \sum_{\forall l: a_l \in Q_i \wedge \forall m: a_m \in Q_j} w_{ij} \qquad (1)$$

Wagi w_{ij} krawędzi grafu przejść stanów można wyliczyć na podstawie tablicy przejść automatu skończonego [4].

Graf Q można poddać kodowaniu algorytmem zorientowanym na minimalizację poboru mocy, dzięki czemu uzyskuje się zbiór kodów E, który razem ze zbiorami G oraz Z tworzy pełną tablicę kodowania stanów automatu.

Algorytm kodowania

1. Zbuduj macierz W.
2. Zbuduj graf H ortogonalności macierzy W.
3. Z grafu H usuń wierzchołki połączone z wszystkimi wierzchołkami grafu.
4. $t := 1$.
5. Znajdź największy wspólny podgraf H_t w grafie H.
6. Z grafu H usuń wszystkie wierzchołki będące wierzchołkami podgrafu H_t.
7. Jeżeli graf H jest pusty – zapamiętaj $T := t$ (liczbę pełnych podgrafów) i przejdź do punktu 9.
8. Zwiększ wartość t ($t := t + 1$) i wróć do punktu 5;
9. Zbuduj graf Q i oblicz wartości wag krawędzi q_{ij} ze wzoru (1)
10. Graf Q poddaj kodowaniu algorytmem minimalizującym pobór mocy (np. sekewncyjny z [4]).
11. Uzyskane kody umieść w macierzy, która stanowi połączenie macierzy W oraz kodów ze zbioru E. Wynikowa macierz stanowi rozwiązanie zadania kodowania stanów automatu.

4. Wyniki badań

Badania symulacyjne algorytmu kodowania stanów w oparciu o model wspólny klas ADE przeprowadzono wykorzystując standardowe testy (benchmark) [14]. Obliczenia zostały wykonane przy założeniu następujących wartości: C = 3pF, f = 5MHz, V_{DD} = 5V oraz $P(x_i = 1) = 0{,}5$. Obliczone wartości przedstawiają wartość mocy wydzielonej w rejestrach przechowujących kody stanów automatu. Pominięto moc wydzieloną na rejestrach wejściowych i wyjściowych, gdyż dla każdego algorytmu jej wartość jest identyczna.

Tab. 1. Wyniki badań symulacyjnych algorytmu kodowania z wykorzystaniem wspólnego modelu automatu klas ADE
Tab. 1. Experimental simulation results of power dissipated by a sequential circuit assigned using the common model of FSM of class ADE

Benchmark	P_N	P_J	P_K	P_S	P_I	P_Z
bbara	83,91	59,44	56,26	52,77	52,77	56,72
bbtas	144,29	112,50	83,15	83,15	83,15	83,15
beecount	160,50	108,05	108,92	89,42	89,42	75,37
cse	93,36	65,34	55,85	44,97	44,96	44,70
dk16	487,86	416,60	377,53	309,09	290,41	291,07
dk27	299,11	325,89	223,21	223,21	223,21	151,77
dk512	380,58	450,89	319,75	238,84	215,40	191,69
donfile	324,22	351,56	265,63	222,66	207,03	140,63
ex1	438,21	307,75	157,70	138,55	133,29	134,41
keyb	191,40	121,67	121,25	104,35	104,35	103,92
pma	273,73	203,53	105,55	104,76	104,28	94,93
s27	216,65	170,43	168,33	168,33	166,23	86,92
s8	53,21	42,41	33,90	33,90	33,90	36,29
shiftreg	281,25	281,25	257,81	210,94	187,50	187,50
train11	107,34	77,45	86,96	63,52	63,52	51,63
Średnia	**235,71**	**206,32**	**161,45**	**139,23**	**133,29**	**115,38**

Wyniki zebrane w tab. 1 przedstawiają wartości mocy wydzielonej w rejestrach przechowujących kody stanów automatu przy zakodowaniu za pomocą algorytmów: JEDI, NOVA, kodowania kolumnowego, sekwencyjnego, iteracyjnego (rafinacyjnego) oraz kodowania z wykorzystaniem modelu wspólnego automatu skończonego klas ADE, jak również modelu wspólnego z wykorzystaniem minimalizacji stanów wewnętrznych automatu.

Kolumna „Benchmark” zawiera nazwę układu testowego. Następne kolumny zawierają wyniki obliczeń dla czterech sposobów kodowania stanów. Dla każdego sposobu kodowania podano wartości obliczonej mocy („P_X” w mW, gdzie indeks „X” oznacza metodę kodowania: N – NOVA, J – JEDI, K – kolumnowe, S – sekwencyjne, I – iteracyjne, Z – kodowanie modelu wspólnego). Na koniec w wierszu oznaczonym „Średnia” obliczono średnią wartość mocy.

Z zestawienia przedstawionego w tab. 1 wynika, że najlepsze wyniki uzyskano po zastosowaniu algorytmu kodowania modelu wspólnego (średnia wartość mocy: 115,38), natomiast algorytm iteracyjny dał średnie wyniki gorsze o prawie 20 mW (średnia wartość mocy: 133,29).

Najlepsze wyniki algorytm kodowania modelu wspólnego dał w 11 z 15 przypadkach, algorytm iteracyjny był najlepszy w 7 przypadkach, algorytm sekwencyjny w 3, natomiast algorytm kodowania kolumnowego w 2. Porównując dwa najlepsze algorytmy (algorytm kodowania modelu wspólnego oraz algorytm iteracyjny) okazało się, że algorytm kodowania modelu wspólnego dał lepsze wyniki w 9 przypadkach, w 2 przypadkach takie same, a w 4 gorsze od algorytmu iteracyjnego.

Tab. 2. Wyniki badań porównawczych algorytmu syntezy wspólnego modelu automatu klas ADE z algorytmami: NOVA, JEDI, kolumnowym, sekwencyjnym oraz iteracyjnym

Tab. 2. Experimental comparison results of power dissipated by a sequential circuit assigned using the common model of FSM of ADE class with different algorithms: NOVA, JEDI, column-based, sequential and iterative

Benchmark	P_Z/P_N		P_Z/P_J		P_Z/P_K		P_Z/P_S		P_Z/P_I	
	war	%	war	%	war	%	war	%	war	%
Bbara	0,68	32	0,95	5	1,01	-1	1,07	-7	1,07	-7
Bbtas	0,58	42	0,74	26	1	0	1	0	1	0
beecount	0,47	53	0,7	30	0,69	31	0,84	16	0,84	16
cse	0,48	52	0,68	32	0,8	20	0,99	1	0,99	1
dk16	0,6	40	0,7	30	0,77	23	0,94	6	1	0
dk27	0,51	49	0,47	53	0,68	32	0,68	32	0,68	32
dk512	0,5	50	0,43	57	0,6	40	0,8	20	0,89	11
donfile	0,43	57	0,4	60	0,53	47	0,63	37	0,68	32
ex1	0,31	69	0,44	56	0,85	15	0,97	3	1,01	-1
keyb	0,54	46	0,85	15	0,86	14	1	0	1	0
pma	0,35	65	0,47	53	0,9	10	0,91	9	0,91	9
s27	0,4	60	0,51	49	0,52	48	0,52	48	0,52	48
s8	0,68	32	0,86	14	1,07	-7	1,07	-7	1,07	-7
shiftreg	0,67	33	0,67	33	0,73	27	0,89	11	1	0
train11	0,48	52	0,67	33	0,59	41	0,81	19	0,81	19
Średnia	**0,49**	**51**	**0,56**	**44**	**0,71**	**29**	**0,83**	**17**	**0,87**	**13**

W tab. 2 przedstawiono porównanie wyników osiągniętych za pomocą algorytmów NOVA, JEDI, kolumnowego, sekwencyjnego i iteracyjnego w stosunku do algorytmu kodowania modelu wspólnego automatu klas ADE. Poszczególne kolumny „P_Z/P_X” określają stosunek wartości mocy układu zakodowanego jednym z algorytmów: N – NOVA, J – JEDI, K – kolumnowy, S – sekwencyjny, I – iteracyjny, do mocy układu zakodowanego algorytmem kodowania modelu wspólnego. Kolumna „war” zawiera stosunek wartości, natomiast „%” – procentowy spadek wartości mocy.

Z zestawienia przedstawionego w tab. 2 wynika, że algorytm kodowania modelu wspólnego daje średnio wyniki niższe o 13% w porównaniu do algorytmu iteracyjnego (0,87 wartości mocy) i o 17% niższe niż algorytm sekwencyjny (0,83 wartości mocy). W najlepszym przypadku algorytm syntezy modelu wspólnego pozwalał na redukcję mocy o 48% zarówno w porównaniu do algorytmu sekwencyjnego jak i do algorytmu iteracyjnego. W najgorszym przypadku algorytm syntezy modelu wspólnego dał wyniki o 7% gorsze w porównaniu zarówno do algorytmu sekwencyjnego, jak i iteracyjnego oraz kolumnowego.

5. Wnioski

Zaprezentowane rezultaty potwierdzają bardzo wysoką skuteczność algorytmu kodowania z wykorzystaniem modelu wspólnego automatu klas ADE. Średnia redukcja mocy wynosiła 13% w porównaniu do algorytmu iteracyjnego oraz 17% w porównaniu do algorytmu sekwencyjnego.

Zmniejszenie mocy związane jest nie tylko ze zmniejszeniem ilości przerzutników przechowujących kod stanu. W wielu wypadkach ilość bitów na kod stanu była taka sama, jak w układach zakodowanych innymi metodami. Zmniejszenie wartości mocy było związane z tym, że do kodu stanu dołączone zostały części wektorów wejściowych i wyjściowych (G i Z). W związku z tym część kodu, która pochodziła ze zbioru E dla różnych stanów mogła być taka sama, co skutkowało brakiem wydzielania mocy przy przełączaniu między stanami.

W dalszych badaniach należy skupić się na sprawdzeniu algorytmu w fizycznych realizacjach na układach CPLD.

6. Literatura

[1] Benini L., DeMicheli G.: State Assignment for Low Power Dissipa-tion, IEEE Journal on Solid-state Circuits, Vol. 30, No. 3 (1995), pp. 259-268.

[2] Chattopadhyay, S., Low power state assignment and flipflop selection for finite state machine synthesis: a genetic algorithmic approach, IEE Proceedings Computers & Digital Techniques, 2001, pp. 147-151.

[3] Grześ T., Salauyou V.: Metody obliczania mocy w układach cyfrowych, „Pomiary, Automatyka, Kontrola” nr 7bis (2006), str. 101-102.

[4] Grześ T., Salauyou V., Algorytmy kodowania stanów wewnętrznych automatu skończonego do minimalizacji poboru mocy, Zeszyty Naukowe Politechniki Białostockiej, Informatyka – Zeszyt 3, Białystok, 2008, s. 53-66.

[5] Koegst M., Franke G., Feske K.: State Assignment for FSM Low Power Design, Proceedings of the Conference on European Design Automation, Geneva 2003, pp. 28-33.

[6] Roy K., Prasad S. C.: Circuit Activity Based Logic Synthesis for Low Power Reliable Operations, IEEE Transactions on VLSI Systems, Vol. 1, No. 4 (1993), pp. 503-513.

[7] Salauyou V., Chyzy M.: Refined CPLD macrocell architecture for the effective FSM implementation, Proc. of the 25th EUROMICRO Conference, Milan, Italy, September 8-10, 1999, Vol. 1, pp. 102-109.

[8] Salauyou V.: Synthesis of Sequential Circuits on Programmable Logic Devices Based on New Models of Finite State Machines, Proc. of the EUROMICRO Symposium on DIGITAL SYSTEMS DESIGN (DSD’2001), September 4-6, 2001, Warsaw, Poland, pp. 170-173.

[9] Salauyou V.: Projektowanie układów cyfrowych na bazie PLD, Hot-Line Telekom, Moskwa 2001, 638 s.

[10] Salauyou V., Chyzy M.: Models of the finite state machines, Proc. of the Sixth Int. Conf. on Methods and Models in Automation and Robotics (MMAR 2000), 28-31 August 2000, Miedzyzdroje, Poland, Vol 2, pp. 909-914.

[11] Salauyou V., Bułatowa I.: Synteza automatów skończonych klasy D na programowalnych układach logicznych, in Proc. of the Conf. Computer-Aided Design of Discrete Devices (CAD DD’01), Minsk, Białoruś 2001, Vol. 2, pp. 6-13.

[12] Salauyou V., Klimowicz A., Synteza wspólnych modeli automatów skończonych na PLD, Konferencja Reprogramowalne Układy Cyfrowe RUC'2002, Politechnika Szczecińska, Szczecin 2002, s. 35-42.

[13] Salauyou V., Grzes T.: FSM State Assignment Methods for Low-power Design, Proceedings of 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM’2007), IEEE Computer Society, pp. 345-348.

[14] Yang S.: Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0, Technical Report, Microelectronics Center of North Carolina, 1991, 43 p.

Artykuł recenzowany