Valery SALAUYOU, Tomasz GRZEŚ
POLITECHNIKA BIAŁOSTOCKA, WYDZIAŁ INFORMATYKI

Badania algorytmów kodowania stanów wewnętrznych automatu skończonego zorientowanych na minimalizację poboru mocy

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@chilan.com

Streszczenie

Kodowanie stanów wewnętrznych automatu skończonego jest jednym z ważniejszych procesów podczas syntezy automatu. W artykule skoncentrowano się na algorytmach minimalizujących pobór mocy. Przeprowadzono badania algorytmu kodowania kolumnowego oraz dwóch algorytmów opracowanych przez autorów: sekwencyjnego oraz iteracyjnego. Wyniki badań wykazują znaczące zmniejszenie poboru mocy układów zakodowanych z wykorzystaniem algorytmu sekwencyjnego w porównaniu z algorytmem kodowania kolumnowego (średnio o 12%), natomiast zastosowanie algorytmu iteracyjnego pozwoliło na obniżenie mocy średnio o kolejne 2% (w porównaniu do algorytmu sekwencyjnego).

Słowa kluczowe: automat skończony, kodowanie, minimalizacja poboru mocy.

Exploration of the Low Power Oriented Algorithms of the Finite State Machine's State Assignment

Abstract

Finite State Machine (FSM) state assignment is one of the most important activities during the synthesis. In this paper we focused on the low-power design oriented algorithms. We explore column-based algorithm as well as two algorithms researched by authors: sequential and iterational. Experimental results shows the significant reduction of the power dissipation after state assignment using sequential algorithm in comparison with the column-based algorithm (of about 12%). Iterational algorithm increase power reduction of about 2% (in comparison with the sequential algorithm).

Keywords: finite state machine, state assignment, low-power design.

1. Wstęp

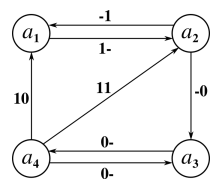
Proces kodowania stanów wewnętrznych jest jednym z ważniejszych procesów podczas syntezy automatów skończonych. Zastosowanie odpowiednich algorytmów umożliwia lepsze wykorzystanie możliwości struktury, jak również zmniejszenie wartości mocy pobieranej przez układ. Obniżenie poboru mocy jest szczególnie ważne przy konstruowaniu urządzeń mobilnych, zasilanych bateryjnie, jak również do zwiększenia wydajności i szybkości systemu.

Układy CMOS pobierają moc wyłącznie podczas przełączania stanu wyjścia bramki lub przerzutnika [5]. Każdorazowa zmiana stanu z a_i do a_j powoduje zmianę stanu wyjść takiej ilości przerzutników, na ilu pozycjach różnią się kody stanów a_i oraz a_j. W związku z tym zmiana sposobu kodowania będzie wpływała na ilość mocy pobieranej przez układ sekwencyjny. Przykładami algorytmów kodowania stanów wewnętrznych obniżających pobór mocy są: [6], gdzie zastosowano algorytm „wygrzewania" (ang. *annealing*), [4], również wykorzystujący wygrzewanie, oraz [1], gdzie zastosowano kodowanie kolumnowe.

2. Problem kodowania stanów wewnętrznych

Automaty skończone można opisywać za pomocą dyskretnych łańcuchów Markowa [2], co pozwala na obliczenie wartości prawdopodobieństw przejścia między stanami. Prawdopodobieństwa statyczne, określające prawdopodobieństwo znalezienia automatu w określonym stanie w chwili $t \to \infty$, można obliczyć korzystając z równań Chapmana-Kołmogorowa [8]. Prawdopodobieństwa zmiany stanu automatu można obliczyć korzystając ze schematu opisanego w [3].

Automat można przedstawić graficznie w postaci grafu przejść (ang. *state transition graph*). Na rys. 1 przedstawiono graf przejść przykładowego automatu skończonego. Węzłami grafu są stany automatu (a_1, a_2, a_3 oraz a_4), natomiast krawędzie odpowiadają przejściom między stanami, które następują po pojawieniu się danego wektora wejściowego. Na rys. 1 pominięto przejścia w ten sam stan, gdyż nie powodują one wydzielenia mocy w układzie.



Rys. 1. Przykładowy graf przejść automatu skończonego
Fig. 1. Example of the state transition graph of the finite state machine

Graf przejść automatu jest grafem skierowanym, jednakże zmiana stanu automatu z a_i na a_j wymaga przełączenia takiej samej ilości przerzutników, co zmiana stanu z a_j na a_i. Pozwala to na przekształcenie grafu skierowanego w nieskierowany. Wagi poszczególnych krawędzi będą miały wartość [1]:

$$w_{i,j} = P(a_i \to a_j) + P(a_j \to a_i) \tag{1}$$

gdzie: $P(a_i \to a_j)$ – prawdopodobieństwo zmiany stanu z a_i na a_j;

Implementując automat skończony w postaci układu sekwencyjnego każdemu stanowi a_i należy przypisać kod c_i. Każdy kod musi być ortogonalny z wszystkimi pozostałym kodami. Ilość bitów kodu N może być wartością z zakresu .

Problem znalezienia sposobu kodowania, który doprowadzi do minimalizacji poboru mocy można przedstawić w postaci zadania całkowitoliczbowego programowania liniowego (ang. Integer Linear Programming):

$$Min\left(\sum_{i,j=1}^{M} w_{i,j} \cdot H(c_i, c_j)\right) \tag{2}$$

przy ograniczeniach:

$$\sum_{l=1}^{N} c_i^l \otimes c_j^l \geq 1, \quad \forall c_i, c_j, \quad c_i \neq c_j \tag{3}$$

Dokładne rozwiązanie zadania opisanego równaniami (2) i (3) może być niewykonalne, szczególnie dla dużych automatów. Dlatego stosuje się rozwiązania heurystyczne [1].

3. Algorytmy kodowania stanów wewnętrznych

Algorytm kodowania kolumnowego bazuje na pojęciu klasy nierozróżnialności kodów stanów wewnętrznych [1]. Kody stanów mające identyczne kody częściowe należą do tej samej klasy nierozróżnialności (4):

$$c_i, c_j \in C_{kl} \Leftrightarrow c_i^m = c_j^m, \forall 1 \leq m \leq l \tag{4}$$

Kodowanie zakończy się sukcesem tylko wtedy, gdy ilość kodów w każdej klasie jest ograniczona zależnością (5):

$$\left|C_{kl}\right| \leq 2^{N-l} \tag{5}$$

Wykorzystując klasy nierozróżnialności można zakodować stany wewnętrzne „bit po bicie", bez konieczności wykonywania operacji na całym kodzie. W trakcie przypisywania wartości bitów do kolejnych kodów stanów automatu należy przestrzegać ograniczenia (5).

W algorytmie sekwencyjnym kodowanie uzależnione jest od przypisanych uprzednio kodów stanów wewnętrznych automatu. Kody przypisywane są kolejno, przy czym do przypisania wybierany jest stan, dla którego wartość funkcji γ, określająca sumę mocy wydzielonej przez automat przy przejściu ze stanu a_i do dowolnego stanu, jest największa [7]. Wartość funkcji γ określa wzór (6):

$$\gamma(c_i) = \sum_{j=1}^{M} w_{i,j} \cdot H(c_i, c_j) \tag{6}$$

Algorytm iteracyjny pozwala na „poprawienie" wyników kodowania przeprowadzonego za pomocą innego algorytmu. Jego działanie polega na zamianie kodów stanów w taki sposób, aby zmniejszyć wartość mocy.

Poniżej przedstawiono pseudo-kod algorytmu iteracyjnego kodowania stanów wewnętrznych automatu skończonego.

```
bez_zmiany := 0;

REPEAT

  zmiana := FALSE;
  FOR i := 1 TO M
    stan_a := ai;
    FOR kod := 0 TO 2^N - 1
      stan_b := Znajdz_Stan(kod);
      moc_przed := Oblicz_Moc();
      IF stan_b
        Zamien_Kody(stan_a, stan_b);
        IF moc_przed > Oblicz_Moc()
          Zamien_Kody(stan_a, stan_b);
        ELSE
          zmiana := TRUE;
        ENDIF
      ELSE
        stary_kod :=  stan_a.kod;
        stan_a.kod := kod;
        IF moc_przed > Oblicz_Moc()
          stan_a.kod := stary_kod;
        ELSE
          zmiana := TRUE;
        ENDIF
      ENDIF
    NEXT
  NEXT

  IF zmiana
    bez_zmiany := 0;
  ELSE
    bez_zmiany := bez_zmiany + 1;
  ENDIF

UNTIL bez zmiany > epsilon;
```

Zastosowane funkcje: `Znajdz_Stan` – wyszukuje stan, któremu przypisano podany kod; `Oblicz_Moc` – zwraca wartość mocy wydzielonej przez układ przy bieżącym kodowaniu; `Zamien_Kody` – zamienia kody stanów.

Wartość `epsilon` jest ustalana przed wykonaniem algorytmu i określa maksymalną ilość przebiegów bez zmiany kodowania.

4. Wyniki badań eksperymentalnych

Badania przeprowadzono wykorzystując standardowe testy (benchmark) [9]. Obliczenia zostały wykonane przy założeniu następujących wartości: C = 3pF, f = 5MHz, V_{DD} = 5V oraz $P(x_i = 1) = 0{,}5$.

Wyniki zebrane w tab. 1 przedstawiają wyniki badań porównawczych algorytmów kodowania kolumnowego, sekwencyjnego oraz iteracyjnego. Dodatkowo obliczono wartość mocy dla kodowania binarnego (gdzie kod odpowiadał numerowi stanu) oraz „one-hot" (gdzie pozycja jedynki w kodzie odpowiadała numerowi stanu). Kolumna „Benchmark" zawiera nazwę układu testowego, następne kolumny zawierają wyniki obliczeń dla pięciu sposobów kodowania stanów. Dla każdego sposobu kodowania podano wartości obliczonej mocy („P_X" w mW, gdzie indeks „X" oznacza metodę kodowania: B – binarne, O – one-hot, K – kolumnowe, S – sekwencyjne, I – iteracyjne). Na koniec w wierszu oznaczonym „Średnia" obliczono średnią wartość mocy oraz stosunku mocy do wartości maksymalnej.

Tab. 1. Porównanie wyników algorytmów: binarnego, one-hot, kolumnowego, sekwencyjnego oraz iteracyjnego
Tab. 1. Comparison of the results for algorithms: binary, one-hot, column-based, sequential and iterational

Benchmark	P_B	P_O	P_K	P_S	P_I
bbara	62,14	83,48	56,26	52,77	52,77
bbtas	134,51	166,3	83,15	83,15	83,15
beecount	113,28	169,56	108,92	89,42	89,42
dk14	239,67	308,59	207,28	223,65	207,28
dk16	401,14	360,8	377,53	309,09	290,41
dk27	290,18	375	223,21	223,21	223,21
dk512	298,55	375	319,75	238,84	215,4
donfile	324,22	281,25	265,63	222,66	207,03
ex1	259,04	238,56	157,7	138,55	133,29
ex5	231,89	246,79	176,4	159,29	159,29
modulo12	171,88	187,5	93,75	93,75	93,75
opus	150,2	243,89	133,38	133,38	133,32
pma	252,5	199,3	105,55	104,76	104,28
s1	329,3	274,19	250,74	200,98	198,65
s1a	329,3	274,19	250,74	200,98	198,65
s27	192,75	255,25	168,33	168,33	166,23
s8	62,27	65,65	33,9	33,9	33,9
tbk	263,16	213,09	221,31	193	191,71
train11	101,9	124,32	86,96	63,52	63,52
Średnia	221,47	233,83	174,76	154,38	149,75

Z zestawienia przedstawionego w tab. 1 wynika, że najlepsze wyniki uzyskano po zastosowaniu algorytmu iteracyjnego (średnia wartość mocy: 149,75), natomiast algorytm sekwencyjny dał średnie wyniki gorsze o niecałe 5 mW (średnia wartość mocy: 154,38), a algorytm kodowania kolumnowego o ponad 25 mW (średnia wartość mocy: 174,76). Najlepsze wyniki uzyskano przy zastosowaniu kodowania iteracyjnego (dla wszystkich testowanych układów), natomiast algorytm sekwencyjny dał najlepsze wyniki dla 18 z 19 testów. W 7 z 19 przypadków najlepsze wyniki osiągnięto przy zastosowaniu kodowania kolumnowego. Najgorsze wyniki dało zastosowanie kodowania „one-hot".

W tab. 2 przedstawiono porównanie wyników osiągniętych za pomocą algorytmów kodowania binarnego, one-hot i kolumnowego w stosunku do kodowania algorytmem sekwencyjnym. Kolumna „P_B/P_S" zawiera stosunek wartości mocy układu zakodowanego algorytmem binarnym do mocy

układu zakodowanego algorytmem sekwencyjnym. W kolumnie „P_O/P_S" umieszczono stosunek wartości mocy układu zakodowanego algorytmem one-hot do mocy układu zakodowanego algorytmem sekwencyjnym. Natomiast kolumna „P_K/P_S" zawiera stosunek wartości mocy układu zakodowanego algorytmem kodowania kolumnowego do mocy układu zakodowanego algorytmem sekwencyjnym.

Tab. 2. Porównanie algorytmów: binarnego, one-hot oraz kolumnowego z algorytmem sekwencyjnym
Tab. 2. Comparison of the algorithms: binary, one-hot and column-based with sequential

Benchmark	P_B/P_S	P_O/P_S	P_K/P_S
bbara	1,18	1,58	1,07
bbtas	1,62	2	1
beecount	1,27	1,9	1,22
dk14	1,07	1,38	0,93
dk16	1,3	1,17	1,22
dk27	1,3	1,68	1
dk512	1,25	1,57	1,34
donfile	1,46	1,26	1,19
ex1	1,87	1,72	1,14
ex5	1,46	1,55	1,11
modulo12	1,83	2	1
opus	1,13	1,83	1
pma	2,41	1,9	1,01
s1	1,64	1,36	1,25
s1a	1,64	1,36	1,25
s27	1,15	1,52	1
s8	1,84	1,94	1
tbk	1,36	1,1	1,15
train11	1,6	1,96	1,37
Średnia	1,49	1,62	1,12

Z zestawienia przedstawionego w tab. 2 wynika, że algorytm sekwencyjny jest średnio 1,49 razy lepszy niż algorytm binarny (2,41 razy w najlepszym przypadku), 1,62 razy lepszy od algorytmu one-hot (1,96 razy w najlepszym przypadku) oraz 1,12 razy lepszy od algorytmu kodowania kolumnowego (1,37 razy w najlepszym przypadku).

W tab. 3 przedstawiono porównanie wyników osiągniętych za pomocą algorytmów kodowania binarnego, one-hot, kolumnowego oraz sekwencyjnego w stosunku do najlepszego wyniku uzyskanego za pomocą algorytmu iteracyjnego.

Tab. 3. Porównanie algorytmów: binarnego, one-hot, kolumnowego oraz sekwencyjnego z iteracyjnym
Tab. 3. Comparison of the algorithms: binary, one-hot, column-based and sequential with iterational

Benchmark	P_B/P_I	P_O/P_I	P_K/P_I	P_S/P_I
bbara	1,18	1,58	1,07	1
bbtas	1,62	2	1	1
beecount	1,27	1,9	1,22	1
dk14	1,16	1,49	1	1,08
dk16	1,38	1,24	1,3	1,06
dk27	1,3	1,68	1	1
dk512	1,39	1,74	1,48	1,11
donfile	1,57	1,36	1,28	1,08
ex1	1,94	1,79	1,18	1,04
ex5	1,46	1,55	1,11	1
modulo12	1,83	2	1	1
opus	1,13	1,83	1	1
pma	2,42	1,91	1,01	1
s1	1,66	1,38	1,26	1,01
s1a	1,66	1,38	1,26	1,01
s27	1,16	1,54	1,01	1,01
s8	1,84	1,94	1	1
tbk	1,37	1,11	1,15	1,01
train11	1,6	1,96	1,37	1
Średnia	1,52	1,65	1,14	1,02

Kolumna „P_B/P_I" zawiera stosunek wartości mocy układu zakodowanego algorytmem binarnym do mocy układu zakodowanego algorytmem iteracyjnym. W kolumnie „P_O/P_I" umieszczono stosunek wartości mocy układu zakodowanego algorytmem one-hot do mocy układu zakodowanego algorytmem iteracyjnym. Kolumna „P_K/P_I" zawiera stosunek wartości mocy układu zakodowanego algorytmem kodowania kolumnowego do mocy układu zakodowanego algorytmem iteracyjnym. Natomiast kolumna „P_S/P_I" zawiera stosunek wartości mocy układu zakodowanego algorytmem sekwencyjnym do mocy układu zakodowanego algorytmem iteracyjnym.

Z zestawienia przedstawionego w tab. 3 wynika, że algorytm iteracyjny jest średnio 1,52 razy lepszy niż algorytm binarny (2,42 razy w najlepszym przypadku), 1,65 razy lepszy od algorytmu one-hot (2 razy w najlepszym przypadku), 1,14 razy lepszy od algorytmu kodowania kolumnowego (1,48 razy w najlepszym przypadku) oraz 1,02 razy lepszy od algorytmu sekwencyjnego (1,11 razy w najlepszym przypadku).

5. Podsumowanie i wnioski

Przeprowadzone badania pokazały znaczne różnice w ilości mocy pobieranej przez układ sekwencyjny, dla którego kodowanie przeprowadzono algorytmami: kodowania kolumnowego i sekwencyjnym. Wykazano również duży wpływ algorytmu iteracyjnego na wyniki uzyskane za pomocą analizowanych algorytmów.

Zastosowanie algorytmu sekwencyjnego dało średnio 1,12 razy mniejsze wartości mocy w porównaniu do algorytmu kodowania kolumnowego. W najlepszym przypadku stosunek wartości był znacznie większy: 1,37. Natomiast zastosowanie algorytmu iteracyjnego dało średnio 1,14 razy mniejsze wartości mocy w porównaniu do algorytmu kodowania kolumnowego, a 1,02 razy mniejsze w stosunku do algorytmu sekwencyjnego. W najlepszym przypadku stosunek wartości wynosił odpowiednio: 1,48 oraz 1,11.

Najniższą moc średnią osiągnął algorytm iteracyjny (149,75 mW), który może stanowić alternatywę dla pozostałych algorytmów. Dodatkowe zmniejszenie mocy może zostać osiągnięte m.in. poprzez zwiększenie ilości bitów kodu.

6. Literatura

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

[2] Freitas A. T., Oliveira A. L.: Implicit Resolution of the Chapman-Kolmogorov Equations for Sequential Circuits: An Application in Power Estimation, Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE) 2003, pp. 10764-10769.

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

[4] 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.

[5] Pedram M.: Power simulation and estimation in VLSI circuits, "The VLSI Handbook", Edited by W-K. Chen, The CRC Press and the IEEE Press, 1999.

[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., 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.

[8] Tsui C.-Y., Monteiro J., Pedram M., Devadas S., Despain A. M., Lin B.: Power Estimation Methods for Sequential Logic Circuits, IEEE Transactions on VLSI Systems, Vol. 3, No. 3 (1995), pp. 404-416.

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

Artykuł recenzowany