Monika WIŚNIEWSKA
UNIWERSYTET ZIELONOGÓRSKI

Redukcja rozmiaru mikroinstrukcji w projektowaniu sterowników mikroprogramowanych

Mgr inż. Monika WIŚNIEWSKA

Ukończyła studia na Wydziale Elektrycznym Uniwersytetu Zielonogórskiego, o specjalności Inżynieria Komputerowa. Obroniła pracę magisterską w 2003 r. Od 2004 r. jest słuchaczem studiów doktoranckich, specjalność informatyka. Jej zainteresowania naukowe to analiza systemów dyskretnych z wykorzystaniem hipergrafów.

e-mail: M.Wisniewska@weit.uz.zgora.pl

Streszczenie

Problem redukcji rozmiaru mikroinstrukcji jest ważnym etapem w procesie projektowania sterowników mikroprogramowanych. Jest to problem NP-trudny, dlatego też powstało wiele metod poszukujących rozwiązania. Zdecydowana większość zaproponowanych algorytmów bazuje na tradycyjnej teorii grafów. W artykule przedstawiono nową metodę redukcji rozmiaru mikro-instrukcji pamięci sterowników mikroprogramowanych. Algorytm bazuje na reprezentacji klas kompatybilności mikrooperacji z zastosowaniem teorii hipergrafów, a redukcja rozmiaru mikroinstrukcji obliczana jest na podstawie najmniejszej transwersali hipergrafu.

Słowa kluczowe: hipergraf, klasa kompatybilności, sterownik (układ) mikroprogramowany, mikrooperacja, mikroinstrukcja, redukcja rozmiaru pamięci.

Microinstruction length reduction in design of microprogrammed controllers

Abstract

A microprogrammed controller (also called microprogrammed control unit) consisting of two main parts is one of realizations of the control unit. The first part is responsible for addressing microinstructions that are kept in the control memory. The role of the second part is to hold and generate adequate microinstructions. Typically, the control memory is implemented as a ROM or RAM memory. Many controllers have a long microinstruction width. It may cause serious problems in the prototyping process. If the design is realized as a System-On-Programmable-Chip (SoPC), the memory can be implemented with dedicated memory blocks of Field Programmable Gate Arrays (FPGAs). However, if the microinstruction length exceeds the length of the dedicated memory block offered by an FPGA, the controller's memory ought to be decomposed. In case of controllers implemented as a System-On-Chip (SoC), the memory is treated as an independent module. It means that each additional bit in the microinstruction width increases the total cost of the memory and the whole device. Therefore, the microinstruction length reduction is a very important part of the designing process of the microprogrammed controllers in a digital system. In the paper, we propose the method of microinstruction length reduction, where the hypergraph theory is applied. The microinstruction length reduction is reached thanks to the calculation of the dual hypergraph and computation of its minimum transversal (minimal vertices cover).

Keywords: hypergraph, compatibility class, microprogrammed controller (control unit), microoperation, microinstruction, reduction of the microinstruction length.

1. Wprowadzenie

Jednostka sterująca jest ważną częścią projektowanego systemu cyfrowego [1, 2, 3, 4, 5]. Standardowa metoda implementacji jednostki sterującej w postaci skończonego automatu stanów często pochłania zasoby układów programowalnych, które mogą być w efektywniejszy sposób wykorzystane przez inne bloki projektowanego systemu. Coraz częściej spotykanym rozwiązaniem jest sterownik mikroprogramowany [6] (zwany także mikroprogramowanym układem sterującym), w którym zastosowano dekompozycję jednostki sterującej na część zarządzającą (adresującą) oraz pamięć, w której przechowywane są mikroinstrukcje kontrolera.

Problemem może być implementacja układu mikroprogramowanego ze względu na ograniczenia rozmiaru słowa pamięci dostępnych w układach programowalnych [10]. Dlatego optymalizacja rozmiaru mikroinstrukcji, generowanych przez układ mikroprogramowany, jest bardzo ważnym etapem projektowania systemów cyfrowych [1].

W niniejszym artykule zaproponowana zostanie nowatorska metoda zmniejszenia rozmiaru mikroinstrukcji przechowywanych w pamięci mikroprogramowanego układu sterującego, oparta na strukturalnej syntezie blokowej automatu cyfrowego. Metoda bazująca na formalnej dekompozycji funkcji logicznych zarysowana została w pracach [12] oraz [6]. Pokazany sposób optymalizacji rozmiaru pamięci jednostki sterującej zostanie zobrazowany przykładem.

2. Sformułowanie problemu badawczego oraz obecny stan wiedzy na świecie

Idea redukcji rozmiaru mikroinstrukcji w praktyce bazuje na specjalnym kodowaniu mikrooperacji. Dwie mikrooperacje mogą zostać zakodowane wspólnie (w obrębie tej samej klasy), jeśli nie są wykonywane równocześnie, w tym samym czasie. Oznacza to, że mikrooperacje są parami kompatybilne. Analogicznie, dwie mikrooperacje są niekompatybilne, jeśli są wykonywane w tym samym czasie.

Redukcja rozmiaru mikroinstrukcji jest klasyfikowana do problemów NP-trudnych [8]. Zdecydowana większość opracowanych dotychczas algorytmów bazuje na teorii klasycznych grafów nieskierowanych [9, 10, 11]. W tym przypadku mikrooperacje są reprezentowane poprzez wierzchołki, a kompatybilność pomiędzy nimi poprzez krawędzie (jeśli dwie mikrooperacje są kompatybilne, istnieje krawędź pomiędzy nimi). Następnie na tej podstawie tworzone są klasy kompatybilności. Wszystkie mikrooperacje, które są parami kompatybilne, grupowane są w klasy (klasa kompatybilności jest reprezentowana poprzez klikę w grafie). W efekcie problem redukcji słowa mikroinstrukcji jest równoważny znalezieniu podziału grafu na niezależne klasy kompatybilności tak, aby rozmiar otrzymanego (zakodowanego) słowa był jak najmniejszy. W praktyce oznacza to wyznaczenie najmniejszego pokrycia wierzchołkowego grafu. Ponieważ problem obliczenia pokrycia wierzchołkowego grafu jest klasyfikowany jako NP-trudny [7, 8], powstało wiele różnych algorytmów, z których większość bazowała na znalezieniu wszystkich maksymalnych podgrafów [10, 11, 12].

W artykule zaproponowana zostanie nowa metoda redukcji rozmiaru mikroinstrukcji. Idea proponowanego rozwiązania bazuje na wykorzystaniu hipergrafu do reprezentacji relacji pomiędzy klasami kompatybilności.

3. Podstawowe definicje

Hipergraf jest rozszerzeniem pojęcia grafu. Jego krawędzie mogą być incydentne do dowolnej liczby wierzchołków [1, 13, 14, 15], w odróżnieniu od grafu, w którym krawędzie mogą być incydentne maksymalnie do 2 wierzchołków. Formalnie, hipergraf definiuje para (V,E), gdzie $V=\{v_1,...,v_X\}$ jest zbior em wierzchołków, natomiast $E=\{e_1,...,e_M\}$ jest zbiorem krawędzi. Hipergraf może być reprezentowany przez macierz incydencji, w której wiersze odpowiadają krawędziom, a kolumny wierzchołkom hipergrafu. Jeśli element macierzy jest równy 1 to j-ta krawędź

jest incydentna do *i*-tego wierzchołka. W przeciwnym przypadku element ten jest równy 0.

Hipergraf H^* jest nazywany *hipergrafem dualnym* do hipergrafu H, jeśli jego wierzchołki odpowiadają krawędziom *H* i analogicznie, jego krawędzie odpowiadają wierzchołkom *H*. Macierz incydencji H^* jest transponowaną macierzą incydencji hipergrafu *H*, i analogicznie $(H^*)^*=H$ [14].

Transwersala (*pokrycie wierzchołkowe*) hipergrafu *H* to zbiór wierzchołków $T \subset V$ taki, że każda krawędź *H* jest incydentna z przynajmniej jednym wierzchołkiem należącym do zbioru *T*:

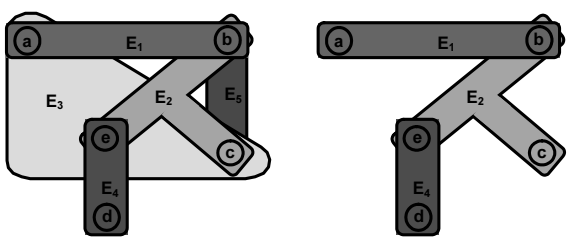
$$T \cap e_i \neq \varnothing \ (i = 1, \ldots, m). \tag{1}$$

Najmniejszą transwersalą hipergrafu *H* jest liczba wierzchomków, które wchodzą w skład transwersali o najmniejszej liczbie wierzchołków:

$$\tau(H) = \min | T |. \tag{2}$$

Istnieje wiele metod wyznaczania najmniejszej transwersali hipergrafu. Rozwiązanie można uzyskać poprzez redukcję macierzy incydencji (metoda bazująca na *algorytmie dokładnym*, ang. *exact covering*), poprzez metody heurystyczne (np. *algorytm z nawrotami*, ang. *back-tracking*) czy też stochastycznymi (np. *algorytm zachłanny*, ang. *greedy*) [1, 13, 15, 19].

Pokrycie krawędziowe hipergrafu (lub po prostu *pokrycie hipergrafu*) definiuje zbiór krawędzi *C*, taki że każdy wierzchołek hipergrafu jest incydenty z co najmniej jedną krawędzią ze zbioru *C*. *Najmniejsze pokrycie* hipergrafu *H* to liczba krawędzi pokrycia *C* zawierającego najmniejszą liczbę hiperkrawędzi. Przykładowe pokrycie hipergrafu ilustruje rys. 1.



Rys. 1. Hipergraf *H* i jego minimalne pokrycie
Fig. 1. Hypergraph *H* and its minimum covering

Należy w tym miejscu zaznaczyć, że zagadnienie wyznaczania najmniejszej transwersali ściśle wiąże się z problemem obliczenia najmniejszego pokrycia hipergrafu. Najmniejsze pokrycie hipergrafu *H* można uzyskać poprzez wyznaczenie najmniejszej trasnwersali $\tau(H^*)$ hipergrafu dualnego H^*. Analogicznie, najmniejsza transwersala $\tau(H)$ może zostać wyznaczona poprzez obliczenie najmniejszego pokrycia hipergrafu dualnego H^*.

Mikrooperacja y_n jest wyjściem generowanym przez jednostkę sterującą. Mikrooperacja jest następnie wykonywana przez sterowany obiekt (zwany także *ścieżką danych*, ang. *data-path*). Zbiór (kolekcja) $Y = \{y_1, \ldots, y_N\}$ mikrooperacji, które są wykonywane w tym samym czasie zdefiniowany jest jako *mikroinstrukcja*. W przypadku sterowników mikroprogramowanych, mikroinstrukcje tworzą *pamięć układu sterującego* [16, 17, 18].

4. Idea proponowanej metody

Minimalizacja rozmiaru mikroinstrukcji (a co za tym idzie całej pamięci mikroprogramowanego układu sterującego) może zostać podzielona na następujące etapy:

1. *Utworzenie zbioru C_c klas kompatybilności (zgodności) mikrooperacji* dla pamięci mikroprogramowanego układu sterującego. W tym punkcie określone zostaną klasy kompatybilności mikroinstrukcji wchodzących w skład pamięci mikroprogramowanego układu sterującego. Mikrooperacje są parami kompatybilne jeśli nie występują w tych samych mikroinstrukcjach. Zbiór klas kompatybilności jest reprezentowany jako $C_C = \{C_1,\ldots,C_K\}$.

2. *Określenie wagi (kosztu) dla każdej klasy kompatybilności.* Waga (koszt) klasy to minimalna liczba bitów, jakie są niezbędne do reprezentacji wchodzących w jej skład mikrooperacji. Waga klasy może zostać w bardzo łatwy sposób wyznaczona ze wzoru:

$$L_i = \lceil \log_2(|C_i|+1) \rceil. \tag{3}$$

Nadmiarowy bit jest niezbędny do określenia stanu, w którym nie jest wykonywana żadna mikroinstrukcja (gdzie C_i oznacza i-tą klasę kompatybilności, L_i - wartość wagi klasy C_i).

3. *Utworzenie macierzy incydencji hipergrafu* dla wyznaczonych klas kompatybilności. Kolejny krok to utworzenie macierzy incydencji hipergrafu H [19]:

$$A_{ij} = \begin{cases} 1 \Rightarrow \textit{kompatybilność} \\ 0 \Rightarrow \textit{niekompatybilność} \end{cases}, \tag{4}$$

gdzie $i = \{1, \ldots, K\}$ oznacza *i*-tą klasę kompatybiloności, $j = \{1, \ldots, N\}$ oznacza the *j*-tą microoperację. Jeżeli pole A_{ij} macierzy incydencji zawiera wartość 1, oznacza to, że *j*-ta mikrooperacja należy do *i*-tej klasy kompatybilności.

4. *Utworzenie hipergrafu H^* dualnego do hipergrafu H.* W tym kroku tworzony jest hipergraf dualny, w praktyce oznacza to transpozycję macierzy incydencji *A* do macierzy transponowanej A^*.

5. *Wyznaczenie najmniejszej transwersali $\tau(H^*)$ hipergrafu dualnego H^*.* Należy tu zaznaczyć, że etap ten może zostać zrealizowany z wykorzystaniem dowolnej metody umożliwiającej obliczenie pokrycia wierzchołkowego hipergrafu. Jednakże ze względu na porównanie z teorią grafów, w niniejszym referacie przedstawiony zostanie podstawowy algorytm redukcji hipergrafu. Rozwiązanie to bazuje na znalezieniu dokładnej transwersali poprzez cykliczną redukcję wierzchołków oraz krawędzi hipergrafu. Szczegółowy opis algorytmu można znaleźć w [1].

6. *Obliczenie całkowitego kosztu dla wszystkich najmniejszych transwersal*. W tym etapie dla każdej najmniejszej transwersali $\tau_s \in \tau$ obliczany jest całkowity koszt (waga). Wartość ta może zostać wyznaczona ze wzoru:

$$W_S = \sum L_i \ \ (i=1, \ldots, I), \tag{5}$$

gdzie W_s oznacza całkowitą wagę (całkowity koszt) transwersali τ_s I jest równy sumie wag L_i wszystkich klas kompatybilności, które należą do danej transwersali. Do dalszych obliczeń wybierana jest transwersala τ_{Min} o najmniejszym całkowitym koszcie W_s.

7. *Kodowanie klas kompatybilności* realizujących minimalną transwersalę hipergrafu. Kolejny krok to zakodowanie klas kompatybilności. Liczba zmiennych *Q* niezbędnych do realizacji tego zadania wynika bezpośrednio z wag przypisanych tym klasom. Dobór kodu jest dowolny, aczkolwiek przypisanie pierwszego kodu (składającego się wyłącznie z 0) jest zalecany dla tych wierszy, w których nie jest generowana żadna mikrooperacja.

8. *Wyznaczenie pamięci mikroprogramowanego układu sterującego z zakodowanymi klasami kompatybilności.* Zawartość pamięci określana jest poprzez zestawienie wszystkich

zmiennych otrzymanych w poprzednich etapach. Wynika stąd, że rozmiar słowa pamięci (mikroinstrukcji) po optymalizacji jest równy sumie wag wszystkich klas wykorzystanych do minimalnego pokrycia:

$$t = \left(1 - \frac{\sum_{i=1}^{|S|} L_i}{N} \right) \cdot 100\ \%\ , \qquad (6)$$

gdzie t oznacza zysk (w %) zajętości pomięci w stosunku przed i po optymalizacji; |S|- liczba klas kompatybilności realizujących minimalne pokrycie; L_i- waga *i*-tej klasy wchodzącej w skład minimalnego pokrycia; *N*- pierwotny rozmiar mikrooperacji.

5. Wyniki badań

Proponowana metoda redukcji rozmiaru mikroinstrukcji została porównana z rozwiązaniami tradycyjnymi, bazującymi na teorii grafów. W celu uzyskania jak najbardziej miarodajnych wyników pomiędzy grafami a hipergrafami, w obu przypadkach zastosowano dokładne algorytmy pokrycia, szczegółowo opisane w [1] oraz [12]. Do badań wykorzystano ponad 1000 losowych testów (tzw. *benach-marks*), które opisywały zawartość pamięci sterownika mikroprogramowanego. Dla każdego modułu pamięci przeprowadzono redukcję rozmiaru mikroinstrukcji w oparciu o teorię grafów oraz równoważną jej redukcję w z zastosowaniem hipergrafów. Pliki testowe zostały podzielone na grupy, w zależności od liczby mikroinstrukcji oraz mikrooperacji. Uśrednione wyniki badań przedstawia tab. 1.

Tab. 1. Wyniki badań
Tab. 1. The results of investigations

Liczba mikrooperacji	Liczba mikroinstrukcji	Średni czas [s]	
		hipergraf	graf
10	55	0,009	0,003
20	55	0,011	0,020
30	55	0,019	0,044
40	55	0,017	0,092
50	55	0,022	0,159
60	55	0,025	0,256
70	55	0,030	0,386
80	55	0,033	0,540
90	55	0,039	0,736
100	90	0,060	1,111
120	150	0,128	2,219
140	150	0,141	3,555
160	150	0,148	5,094
180	150	0,164	7,115
200	150	0,193	9,828

Tabela zawiera następujące wartości: liczba mikrooperacji, średnia liczba mikroinstrukcji, średni czas uzyskany podczas wykonywania algorytmu (odpowiednio dla hipergrafów oraz grafów). Wyniki przeprowadzonych badań pokazują, że zastosowanie algorytmów wykorzystujących hipergrafy znacznie przyspiesza proces redukcji rozmiaru mikroinstrukcji, w porównaniu do rozwiązań opierających się o teorię grafów. Jedynie w przypadku bardzo małych pamięci, gdzie liczba mikrooperacji nie przekracza 10, zastosowanie grafów zaowocowało lepszymi rezultatami. Dla wszystkich pozostałych testów algorytmy opracowane w oparciu o hipergrafy były znacznie szybsze. Należy tu podkreślić, że przewaga hipergrafów nad grafami rośnie wraz ze wzrostem liczby mikrooperacji. W przypadku pamięci zawierających 200 mikrooperacji i 150 mikroinstrukcji, algorytm bazujący na teori hipergrafów wykonywany był ponad 50 razy szybciej niż odpowiadający mu program realizowany z wykorzystaniem metod opierających się o zastosowanie grafów.

6. Podsumowanie

W artykule przedstawiono nową metodę redukcji rozmiaru mikroinstrukcji pamięci sterowników mikroprogramowanych. Algorytm bazuje na reprezentacji klas kompatybilności mikrooperacji poprzez zastosowanie teorii hipergrafów. Najmniejsze pokrycie wierzchołkowe obliczane jest poprzez wyznaczenie najmniejszej transwersali hipergrafu. Wyniki przeprowadzonych badań potwierdzają skuteczność opracowanej metody. Proces redukcji rozmiaru mikroinstrukcji jest znacznie szybszy (nawet o ponad 50 razy) w porównaniu do rozwiązań opierających się o wykorzystanie tradycyjnych grafów niekierowanych.

7. Literatura

[1] G. De Micheli: Synthesis and Optimization of Digital Circuits. McGrawHill, 1994.
[2] A. Clements: The principles of computer hardware. Oxford University Press, New Jersey, 2000.
[3] T. Łuba: Synteza układów cyfrowych. Praca zbiorowa pod redakcją prof. Tadeusza Łuby, WKŁ, Warszawa, 2003.
[4] M. Bolton: Digital Systems Design with Programmable Logic. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
[5] D. Burskey: Embedded Logic and Memory Find Home in FPGA. Electronic Design, No. 14, pp. 43-56 1999.
[6] D. Gajski: Principles of Digital Design, Prentice Hall, New Jersy1997.
[7] R. Puri, J. Gu: Microword Length Minimization in Microprogrammed Controller Synthesis. IEEE Trans. on Computer-Aided Design, 1993.
[8] E. L. Robertson: Microcode bit optimization is NP-complete. IEEE Trans. Comput., vol. C-28, pp. 316–319, Apr. 1979.
[9] S. Dasgupta: The Organization of Microprogram Stores. Computing Surveys, 1979.
[10] S. K. Hong, I. C. Park, C. M. Kyung: An O(n3logn) Heuristic for Microcode Bit Optimization. In ICCAD-90, pp. 180-183, 1990.
[11] J. Lam, J. M. Delosme, Simulated Annealing: A Fast Heuristic for Partitioning VLSI Networks. In ICCAD-88, pp. 510-513, 1988.
[12] A. V. Aho, J. E. Hopcroft, J. D. Ullman: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, 1974.
[13] T. Eiter, G. Gottlob: Identifying the Minimal Transversals of a Hypergraph and Related Problems. SIAM Journal on Computing Volume 24, Issue 6 (December 1995) pp. 1278 – 1304, Year of Publication: 1995.
[14] C. Berge: Graphs and Hypergraph. North-Hols.r Mathematical Library, Amsterdam 1976.
[15] T. Eiter, G. Gottlob: Hypergraph transversal computation and related problems in logic and AI. LNCS, pp. 549—564, Springer, 2002.
[16] H.H. Hoos, T. Stutzle: Local Search Algorithms: An Empirical Evaluation. Journal of Automated, pp. 421-481, 2000.
[17] M. V. Wilkes: The best way to design an automatic calculating machine. Manchester University inaugural conference, Manchester, England, 1951.
[18] M. Molski: Modułowe i mikroprogramowalne układy cyfrowe. WKŁ, Warszawa, 1986.
[19] P. Sapiecha: Algorytmy syntezy funkcji i relacji boolowskich w aspekcie metod reprezentacji i kompresji danych. PW, Wydział Elektroniki i Technik Informacyjnych, Warszawa, 1998.

Artykuł recenzowany