Monika WIŚNIEWSKA, Marian ADAMSKI, **Remigiusz WIŚNIEWSKI**
UNIWERSYTET ZIELONOGÓRSKI,
ul. Licealna 9, 65-417 Zielona Góra

Selekcja klas kompatybilności z zastosowaniem teorii hipergrafów

Mgr inż. Monika WIŚNIEWSKA

Ukończyła studia na Wydziale Elektrycznym Uniwersytetu Zielonogórskiego, o specjalności Inżynieria Komputerowa. Obroniła pracę magisterską w 2003 r. W roku 2001 odbyła przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Od 2004 r. jest słuchaczem studiów doktoranckich, specjalność informatyka. Jej zainteresowania naukowe to dekompozycja systemów dyskretnych z wykorzystaniem hipergrafów.

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

Prof. dr hab. inż. Marian ADAMSKI

Profesor zwyczajny, dyrektor Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych. Członek IEEE, IEE, ACM, PTI oraz PTETiS.

e-mail: M.Adamski@iie.uz.zgora.pl

Streszczenie

Redukcja pojemności pamięci jest istotnym etapem w procesie projektowania systemów dyskretnych. Często pojemność prototypowanej pamięci przekracza rozmiar docelowego bloku pamięci (np. w układach programowalnych FPGA). Najczęściej stosowanym rozwiązaniem jest redukcja rozmiaru mikroinstrukcji w projektowanym systemie. Algorytm bazuje na wyznaczeniu, a następnie selekcji klas kompatybilności poszczególnych mikrooperacji. W artykule zaprezentowane zostaną 2 autorskie algorytmy selekcji klas kompatybilności. Metody opierają się o wykorzystanie teorii hipergrafów (zastosowanie pokrycia wierzchołkowego). Proponowane rozwiązania zostaną gruntownie przeanalizowane oraz porównane z metodą tradycyjną, bazują na przekształceniach macierzowych.

Słowa kluczowe: hipergraf, selekcja klas kompatybilności, transwersale (pokrycie wierzchołkowe) hipergrafu, mikrooperacja, redukcja pojemności pamięci.

Application of the hypergraph theory to selection of compatibility classes

Abstract

The problem of memory size reduction is a very important part of design process of discrete systems. The prototyped memory size very often exceeds the size of memory blocks offered by programmable devices (in example FPGAs). One of the most popular solutions to this problem is memory size reduction. The reduction process is based on selection of the compatibility classes of microoperations. Three methods of selection of compatibility classes are presented in the paper. The first one is a well-known method of selection, to which the fast reduction algorithm is applied. The algorithm bases on the matrix operations, which can also be represented as reduction of the hypergraph incidence matrix. In each step some vertices and edges are reduced. The reduced matrix holds the final result. The two other solutions introduced in the paper are based on the idea of computation of the minimal transversal (vertices covering) of hypergraphs. Proper microoperations are represented by the hypergraph vertices, while compatibility classes are described by the hyperedges. Therefore, any method of hypergraph transversal calculation can be applied to achieve the selection. In the paper the authors propose and analyse the effectiveness of backtracking and greedy algorithms. The proposed solutions are compared with the traditional method, which is based on transformation of the hypergraph incidence matrix. The obtained results of experiments are analysed and discussed in detail.

Dr inż. Remigiusz WIŚNIEWSKI

Absolwent Uniwersytetu Zielonogórskiego, pracę doktorską obronił w 2008 roku. W latach 2000-2001 dwukrotnie odbył przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Aktualnie pracuje jako adiunkt w Uniwersytecie Zielonogórskim. Zainteresowania badawcze obejmują zagadnienia z zakresu kryptologii (zarówno sprzętowej, jak i programowej), oraz metodologii projektowania i implementacji systemów cyfrowych z wykorzystaniem struktur programowalnych FPGA.

e-mail: R.Wisniewski@iie.uz.zgora.pl

Keywords: hypergraph, selection of compatibility classes, transversals (hitting set), microoperation, memory size reduction.

1. Wprowadzenie

Pamięć zewnętrzna jest jednym z najistotniejszych elementów współczesnych systemów dyskretnych [1]. W przypadku układów realizowanych jako system-on-a-chip (SoC) pojemność modułu pamięci ściśle wiąże się z kosztami produkcji całego układu. Z kolei w kompleksowych rozwiązaniach typu system-on-a-programmable-chip (SoPC) wykorzystywane są dedykowane bloki pamięci matryc FPGA (Field Programmable Gate Arrays), a pojemność pamięci jest ściśle ograniczona [2, 3]. Dlatego też projektant często staje przed problemem związanym ze zmniejszeniem pojemności pamięci w celu obniżenia kosztów całego systemu (w przypadku rozwiązań SoC) lub dopasowania prototypowanej pamięci do rozmiarów dedykowanych bloków pamięci oferowanych przez układy FPGA. Operacja ta może zostać wykonana na wiele sposobów, jednym z nich jest dekompozycja [3], czyli podział na mniejsze bloki. Innym powszechnie stosowanym rozwiązaniem jest redukcja rozmiaru słowa pamięci poprzez odpowiednie zakodowanie mikroinstrukcji. Dzięki temu całkowita pojemność pamięci może zostać zredukowana [4, 5, 6, 7]. Zawartość pamięci musi zostać w późniejszej fazie zdekodowana, jednakże proces ten może zostać w bardzo łatwy sposób zrealizowany przez bloki logiczne docelowego układu programowalnego.

2. Obecny stan wiedzy na świecie

Metoda redukcji pojemności pamięci bazuje na selekcji klas kompatybilności poszczególnych mikrooperacji. Jest to problem NP-trudny [4, 6, 8]. Na świecie pojawiło się bardzo wiele algorytmów usprawniających proces selekcji klas kompatybilności.

Na świecie pojawiło się bardzo wiele algorytmów usprawniających proces selekcji klas kompatybilności. Pojawiły się różnego rodzaju rozwiązania: bazujące na algorytmach dokładnych, stochastycznych, czy też heurystycznych [1, 4, 6]. Zdecydowana większość metod ma „wspólny korzeń", jakim jest zastosowanie teorii grafów do reprezentacji klas kompatybilności. Klasyczna metoda selekcji klas kompatybilności (opisana w [1]) bazuje na przekształceniach macierzowych, w których graf jest reprezentowany poprzez pseudo-macierz incydencji. Wierzchołki grafu określane są poprzez kolumny pseudo-macierzy, natomiast powiązania pomiędzy wierzchołkami grafu definiują wiersze, przy czym dopuszczalne jest powiązanie więcej niż dwóch wierzchołków w jednym wierszu macierzy. Jak pokazano w [1] taki sposób reprezentacji grafu w praktyce jest jedną z możliwości opisu hipergrafu (macierz incydencji hipergrafu). Warto w tym miejscu dodać, że pomimo usystematyzowania i ujednolicenia nazewnictw struktur grafowych oraz hipergrafowych [1, 9, 10], do dziś w literaturze można spotkać mylne określenia związane z tymi aparatami matematycznymi (wielu inżynierów ciągle stosuje

pojęcia „pseudo-macierz", „pseudo-graf", „rozszerzony graf" nazywając systemy grafowe w praktyce będące hipergrafami).

W niniejszym artykule zaprezentowane zostaną dwie autorskie metody selekcji klas kompatybilności. Wszystkie algorytmy są rozwinięciem metody bazowej wykorzystującej opis relacji klas kompatybilności z wykorzystaniem hipergrafów. W takim podejściu właściwa selekcja klas kompatybilności sprowadza się do wyznaczenia najmniejszej transwersali w rozpatrywanym hipergrafie.

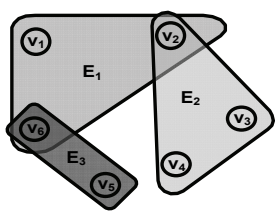
W artykule skoncentrowano się na zaprezentowaniu oraz porównaniu efektywności konkretnych metod selekcji klas kompatybilności. Dlatego też dokładnie omówiono poszczególne algorytmy wyznaczania najmniejszej transwersali (w tym także metodę bazową). Szczegółowy schemat redukcji pojemności pamięci można znaleźć w innych pozycjach autorów [4, 5, 6, 11].

3. Podstawowe definicje

Hipergraf jest rozszerzeniem pojęcia grafu. Jego krawędzie, zwane hiperkrawedziami [9], mogą być incydentne do dowolnej liczby wierzchołków. Formalnie, hipergraf H definiuje dwójka:

$$H = (V,E), \tag{1}$$

gdzie: $V=\{v_1,\ldots,v_n\}$, jest dowolnym, niepustym zbiorem wierzchołków; $E=\{E_1,\ldots,E_m\}$, jest zbiorem krawędzi hipergrafu, czyli podzbiorem zbioru $P(V)$ wszystkich możliwych niepustych zbiorów, których elementy należą do V. Na rys. 1 przedstawiono przykładowy hipergraf H_1.



Rys. 1. Hipergraf H_1
Fig. 1. Hypergraph H_1

Jedną z najpopularniejszych metod reprezentacji hipergrafu jest macierz incydencji A, w której wiersze odpowiadają krawędziom, a kolumny wierzchołkom hipergrafu. Macierz incydencji A_1 dla hipergrafu H_1 pokazano na rys. 2.

$$A_1 = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \end{matrix}$$

Rys. 2. Macierz incydencji hipergrafu H_1
Fig. 2. The incidence matrix of hypergraph H_1

Transwersalą (pokryciem wierzchołkowym) hipergrafu H jest zbiór $T \subset V$ zawierający wierzchołki incydentne do każdej krawędzi hipergrafu.

Najmniejszą transwersalą $\tau(H)$ hipergrafu H jest transwersala zawierająca najmniejszą liczbę elementów spośród wszystkich transwersali hipergrafu H [9, 10, 11]:

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

Warto w tym miejscu wspomnieć, że hipergraf może mieć więcej niż jedno najmniejsze pokrycie wierzchołkowe. Przykładowo w rozpatrywanym przykładzie najmniejszą transwersalę może tworzyć zbiór $\tau_1(H_1)=\{v_2,v_5\}$, lecz także $\tau_2(H_1)=\{v_3,v_6\}$. W takim przypadku algorytm selekcji restrykcyjnie określa tzw. wagi klas kompatybilności, na podstawie których do dalszej analizy brane jest jedno rozwiązanie (szczegółowy opis algorytmu można znaleźć w [9, 11]).

Istnieje wiele metod wyznaczania najmniejszej transwersali hipergrafu. Rozwiązanie można uzyskać poprzez redukcję macierzy incydencji, metodami dokładnymi, heurystycznymi, stochastycznymi, czy też symbolicznymi (np. wnioskowanie naturalne z wykorzystaniem rachunku Gentzena) [1, 9, 10, 11, 12]. W artykule skoncentrowano się na trzech algorytmach, które szerzej omówiono w kolejnym rozdziale.

4. Metody wyznaczania najmniejszej transwersali w hipergrafie

W rozdziale omówiono trzy algorytmy wyznaczania najmniejszych transwersal hipergrafu, które są podstawą dla późniejszej selekcji klas kompatybilności. Pierwsza z metod to rozwiązanie klasyczne, na podstawie którego opracowano dwa pozostałe algorytmy (metoda zachłanna oraz metoda z nawrotami).

4.1. Szybki algorytm redukcji

Metoda klasyczna (szybki algorytm redukcji) opiera się na wykorzystaniu zależności i powiązań, jakie przechowuje hipergraf. Pierwszym krokiem, jaki należy wykonać jest przedstawienie hipergrafu w postaci macierzy incydencji. Rzędy macierzy incydencji odpowiadają krawędziom hipergrafu, które należy pokryć wierzchołkami reprezentowanymi poprzez kolumny macierzy.

Zgodnie ze schematem przedstawionym w [1], szybki algorytm redukcji operuje na następujących własnościach macierzy incydencji hipergrafu:

- Istotna kolumna (istotny wierzchołek) - jeśli dla danego wierzchołka (kolumny) istnieje wiersz z pojedynczą jedynką, oznacza to, że tylko ten jeden wierzchołek należy do danej hiperkrawędzi i musi być częścią każdego pokrycia.
- Redukcja zdominowanych kolumn - kolumna dominuje inną kolumnę, gdy elementy poprzedniej kolumny są większe lub równe elementom następnej. Analogicznie, wierzchołek dominuje inny wierzchołek, jeżeli jest incydentny do najmniej wszystkich krawędzi incydentnych do innego wierzchołka. Zdominowane wierzchołki zostają usunięte z hipergrafu.
- Redukcja dominujących wierszy - wiersz dominuje inny wiersz, gdy elementy poprzedniego wiersza są większe lub równe odpowiednim elementom następnego. Analogicznie, krawędź dominuje inną krawędź, jeżeli jest incydentna do co najmniej wszystkich wierzchołków incydentnych do innej krawędzi. Dominujące wiersze (krawędzie) można zaniedbać, ponieważ każde pokrycie zbioru zdominowanego jest pokryciem pełnego zbioru.

Proces wyznaczenia najmniejszej transwersali bazuje na wykorzystaniu powyższych reguł, przy czym etap wyznaczania i usuwania zdominowanych kolumn oraz dominujących wierszy jest wykonywany cyklicznie, aż do uzyskania najmniejszego pokrycia wierzchołkowego. Ostateczny wynik uzyskuje się ze zredukowanej macierzy. Wierzchołki, które są incydentne do jakiejkolwiek krawędzi zredukowanego hipergrafu będą wchodzić w skład poszukiwanej najmniejszej transwersali.

$$A_2 = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \end{matrix}$$

Rys. 3. Macierz incydencji hipergrafu H_2
Fig. 3. The incidence matrix of hypergraph H_2

Przykładowo, dla hipergrafu H_2, który został przedstawiony na rys. 3, w pierwszym etapie redukcji określona zostanie istotna kolumna, reprezentująca wierzchołek v_6. Tylko ten wierzchołek jest incydentny z krawędzią E_4. Następnie, zgodnie z przedstawionym algorytmem, z macierzy usuwane są zdominowane kolumny.

Na tym etapie redukowana jest kolumna 1 (zdominowana przez kolumnę 2) oraz kolumna 5 (zdominowana przez kolumnę 4). W kolejnym kroku redukowane są dominujące wiersze. Dla rozpatrywanego przykładu wiersz 2 i 3 są identyczne, wobec czego jeden z nich może zostać usunięty. Należy tu zaznaczyć, że nie ma żadnego znaczenia, który wiersz zostanie zredukowany (inaczej sytuacja wygląda w przypadku redukowania identycznych kolumn - wówczas wiadomo na pewno, że hipergraf ma więcej niż jedną najmniejszą transwersalę).

Przedstawiony schemat jest powtarzany cyklicznie, aż do całkowitej redukcji hipergrafu. Wynikiem szybkiego algorytmu redukcji będzie zbiór wierzchołków: $\tau_1=\{v_4,v_6\}$.

4.2. Algorytm z nawrotami

W przypadku algorytmu z nawrotami, ze zbioru wierzchołków hipergrafu wybierany i redukowany jest wierzchołek oznaczony najmniejszą wartością (zgodnie z porządkiem leksykograficznym). Jeśli pozostały zbiór wierzchołków w dalszym ciągu pozostaje pokryciem hipergrafu, a uzyskany rezultat jest lepszy od dotychczas znalezionego, bieżąca transwersala staje się najlepszą. W kolejnych etapach rekurencyjnie przeprowadzane jest poszukiwanie najlepszego rozwiązania dla wszystkich wierzchołków.

W przypadku hipergrafu H_2, zbiór wierzchołków składa się z sześciu elementów: $V=\{v_1,\ldots,v_6\}$. Zgodnie z przedstawionym algorytmem, jako pierwszy do redukcji wybrany zostaje wierzchołek v_1. Po redukcji, hipergraf składa się z pięciu wierzchołków: $V^1=\{v_2,\ldots,v_6\}$. Ponieważ pozostałe elementy w dalszym ciągu stanowią pokrycie, zbiór ten staje się najlepszym aktualnie znalezionym rozwiązaniem. Następnie usuwany jest wierzchołek v_2, wobec czego zbiór wierzchołków zredukowany zostaje do $V^2=\{v_3,\ldots,v_6\}$, który także stanowi pokrycie i jest najmniejszą uzyskaną transwersalą. Podobnie jest w kolejnym etapie. Redukcja wierzchołka v_3 ogranicza zbiór pozostałych wierzchołków do trzech ($V^3=\{v_4,\ldots,v_6\}$), a uzyskane rozwiązanie w dalszym ciągu stanowi pokrycie hipergrafu. Z kolei usunięcie wierzchołka v_4 powoduje ograniczenie zbioru wierzchołków do dwóch: $V^4=\{v_5,v_6\}$, jednakże pozostałe wierzchołki nie stanowią pokrycia hipergrafu. Dlatego też algorytm rekurencyjnie powraca do rozwiązania V^3, a ze zbioru usuwany jest kolejny wierzchołek - v_5. Pozostały zbiór wierzchołków $V^5=\{v_4, v_6\}$ jest jednocześnie najmniejszą dotychczas znalezioną transwersalą.

Przedstawiona metodologia jest powtarzana dla wszystkich wierzchołków hipergrafu, a kolejne etapy są wykonywane analogiczne do wyżej przedstawionych. Ostatecznie najlepszą transwersalą okazuje się zbiór V^5. Oznacza to, że znaleziony wynik jest identyczny z rezultatem uzyskanym w przypadku poszukiwania pokrycia z wykorzystaniem szybkiego algorytmu redukcji.

4.3. Algorytm zachłanny

Przedstawione metody wyznaczania transwersali - szybki algorytm redukcji oraz algorytm z nawrotami zawsze dadzą najlepsze rozwiązanie. Wadą takich metod jest jednakże złożoność obliczeniowa, szczególnie w przypadku algorytmu z nawrotami (złożoność wykładnicza), co oznacza że dany problem może nie zostać rozwiązany, lub rozwiązanie nie zostanie osiągnięte w satysfakcjonującym czasie.

Metoda zachłanna poszukuje rozwiązania na podstawie decyzji lokalnie optymalnej. Bardzo dużą zaletą algorytmu jest jego szybkość, natomiast największą wadą - brak możliwości określenia, czy znalezione rozwiązanie jest rzeczywiście najlepsze.

Algorytm zachłanny poszukuje rozwiązania na podstawie decyzji lokalnie optymalnej. Oznacza to, że w każdym kroku wybierana jest możliwość aktualnie najlepsza i nie ma znaczenia, czy podjęta decyzja jest w danym momencie optymalna globalnie (stąd też nazwa "zachłanny"). Niewątpliwą zaletą algorytmu jest jego złożoność obliczeniowa (quasi-liniowa). Niestety metoda nie zawsze znajduje najlepsze rozwiązanie, jakim jest najmniejsze pokrycie wierzchołkowe hipergrafu. Ponadto problemem jest możliwość utknięcia algorytmu w minimum lokalnym, które jest bardzo trudne do wykrycia [8].

Zasada działania algorytmu zachłannego jest następująca:

1. Ustal wartość zbioru pokrycia $T=0$. Na początku poszukiwany zbiór pokrycia jest pusty, wierzchołki będą wyznaczane i dodawane w kolejnych cyklach.
2. Wybierz istotny wierzchołek, a jeśli takiego nie ma - wybierz wierzchołek o największym stopniu. Na tym etapie algorytm dokonuje lokalnej selekcji optymalnego rozwiązania. Wybrany zostaje wierzchołek istotny (który musi być częścią szukanego pokrycia). Jeśli nie istnieje taki wierzchołek, do dalszej analizy wybierany jest wierzchołek incydentny do największej liczby krawędzi. Jeśli istnieje więcej niż jedno rozwiązanie (np. istnieją dwa wierzchołki istotne lub dwa wierzchołki mają taki sam stopień), wówczas algorytm dokonuje wyboru losowego.
3. Dodaj wybrany wierzchołek do zbioru pokrycia T, a następnie usuń ten wierzchołek z hipergrafu oraz wszystkie incydentne z nim krawędzie.
4. Sprawdź, czy T jest transwersalą, jeśli nie, powtórz algorytm od kroku nr 2. W tym kroku sprawdzany jest zbiór T. Jeśli należące do niego wierzchołki tworzą szukane pokrycie wierzchołkowe, algorytm kończy swe działanie. W innym przypadku metoda przechodzi do kolejnego cyklu, w którym szukane jest następne optimum lokalne (czyli kolejny wierzchołek istotny lub o największym stopniu).

W przypadku hipergrafu H_2, algorytm rozpoczyna działanie od ustalenia wartości zbioru T=0. Następnie poszukiwane jest optimum lokalne poprzez selekcję wierzchołka istotnego. W przypadku H_2 jest to wierzchołek v_6, ponieważ jako jedyny jest incydentny z kolumną E_4. Dlatego też zostaje on dodany do zbioru T i jednocześnie usunięty z hipergrafu wraz z krawędziami do niego incydentnymi (czyli E_2, E_3 oraz E_4). W ten sposób hipergraf został zredukowany do pięciu wierzchołków oraz dwóch hiperkrawędzi. W kolejnym kroku sprawdzany jest warunek, czy zbiór T stanowi transwersalę hipergrafu H_2. Ponieważ zależność ta nie jest spełniona, cykl zostaje powtórzony. Tym razem nie istnieje wierzchołek istotny, wobec czego wybierany jest wierzchołek o największym stopniu. Spośród pozostałych wierzchołków, istnieje jeden wierzchołek v_4 incydentny do dwóch hiperkrawędzi, natomiast pozostałe wierzchołki są stopnia pierwszego. Oznacza to, że v_4 jest dodawany do T, a następnie zostaje usunięty z hipergrafu. Ponieważ zbiór T realizuje pokrycie wierzchołkowe, algorytm kończy działanie. Uzyskane rozwiązanie $T=\{v_4,v_6\}$ jest jednocześnie pokryciem najmniejszym, a identyczny wynik uzyskano w przypadku poszukiwania transwersali metodami dokładnymi.

Należy w tym miejscu wyraźnie zaznaczyć, że rozwiązanie zostało znalezione już w drugim cyklu poszukiwań, co wyraźnie pokazuje przewagę metody zachłannej nad przedstawionymi wcześniej algorytmami, jeśli za kryterium przyjąć złożoność obliczeniową i czas poszukiwań. Niestety, jak już wcześniej wspomniano metoda nie zawsze zwraca najlepsze rozwiązanie.

5. Badania i eksperymenty

W rozdziale szczegółowo omówiono metodologię eksperymentów, przeprowadzonych w celu określenia skuteczności opracowanych metod (w tym także biblioteka danych testowych oraz wykorzystane programy umożliwiające wyznaczenie transwersali hipergrafu). Uzyskane wyniki badań zostały gruntownie przeanalizowane.

5.1. Biblioteka modułów testowych

Wszystkie algorytmy zaprezentowane w niniejszym artykule zostały zweryfikowane eksperymentalnie. W tym celu wykorzystano ponad 100 pamięci testowych. Zdecydowana większość badanych testów (około 70) stanowiły rzeczywiste układy pamięci sterowników logicznych, zaczerpnięte z bibliotek udostępnionych przez prof. Samary Baranova (Holon Institute of Technology) oraz bazy testowej Uniwersytetu Zielonogórskiego (pod kierunkiem

prof. Mariana Adamskiego). Pozostałe moduły to pamięci hipotetyczne (wygenerowane losowo).

5.2. Metodologia przeprowadzonych badań

Każda z pamięci testowych została poddana pełnej ścieżce związanej z redukcją rozmiaru mikroinstrukcji, przy czym głównym celem przeprowadzonych eksperymentów był etap związany z selekcją klas kompatybilności. Zbadano oraz porównano trzy różne metody wyznaczania transwersali.

Za kryterium porównawcze przyjęto czas wykonania oraz rozmiar znalezionej transwersali (z uwzględnieniem kosztów poszczególnych wag, określonych w procesie redukcji pojemności pamięci – szczegóły można znaleźć w publikacjach [11, 12]).

5.3. Wyniki badań

Tabela 1 przedstawia uśrednione wyniki przeprowadzonych badań. W kolumnach zaprezentowano wyniki uzyskane podczas wyznaczania transwersali hipergrafu z wykorzystaniem poszczególnych algorytmów. Wiersze obrazują skuteczność danego algorytmu, przedstawioną jako procentową redukcję pojemności pamięci oraz uśredniony czas wykonania danego algorytmu.

Tab. 1. Uśrednione wyniki badań
Tab. 1. Average results of experiments

	Szybki algorytm redukcji	**Algorytm z nawrotami**	**Algorytm zachłanny**
Redukcja pojemności pamięci [%]	82%	69%	**66%**
Średni czas wykonania [s]	0,31273	**0,026248**	40, 806[1]

[1] - ze względu na złożoność wykładniczą algorytmu, część testów przerwano po 1 godzinie

W celu lepszego zobrazowania i porównania algorytmów, w tabeli 2 przedstawiono wyniki badań w odniesieniu do szybkiego algorytmu redukcji. Jak wcześniej wspomniano, metoda ta została wybrana jako reprezentant rozwiązań tradycyjnych, opartych na klasycznych obliczeniach macierzowych [5, 6].

Tab. 2. Uśrednione wyniki badań w porównaniu do szybkiego algorytmu redukcji
Tab. 2. Average results of experiments compared to the traditional method

	Algorytm z nawrotami	**Algorytm zachłanny**
Redukcja pojemności w stosunku do szybkiego algorytmu redukcji	84%	**81%**
Redukcja czasu w stosunku do szybkiego algorytmu redukcji	**8%**	13048%[1]

[1] - ze względu na złożoność wykładniczą algorytmu, część testów przerwano po 1 godzinie.

5.4. Analiza wyników badań

Zgodnie z oczekiwaniami najskuteczniejszy okazał się algorytm z nawrotami (średnio o 29% lepszy niż rozwiązania tradycyjne, oparte na szybkim algorytmie redukcji). Zasadniczą wadą tej metody jest jej wykładnicza złożoność, co w praktyce uniemożliwia jej zastosowanie dla relatywnie dużych pamięci (metoda sprawdza się dla pamięci do rozmiaru około 400 bitów, czyli np. 20 mikroinstrukcji x 20 mikrooperacji). Zaskakująco dobre efekty uzyskano w przypadku algorytmu zachłannego. Średnio metoda ta była aż o 16% lepsza od rozwiązania tradycyjnego, niewiele ustępując algorytmowi z nawrotami. W połączeniu ze zdecydowanie najkrótszym czasem wykonania, należy stwierdzić, że właśnie to rozwiązanie okazało się najskuteczniejsze w przypadku większych modułów pamięci. Uzyskane wyniki wstępnie zweryfikowano z wykorzystaniem rzeczywistego systemu dyskretnego. W tym celu zaimplementowano i porównano funkcjonalność losowo wybranych 10 układów testowych, zawierających różne wersje pamięci (pamięć nieminimalizowana, oraz zredukowana metodami: tradycyjną, dokładną oraz zachłanną). Do realizacji zastosowano układ FPGA *Spartan3E* firmy *Xilinx*. Wyniki eksperymentów wykazały poprawność założeń teoretycznych, wszystkie zaimplementowane systemy ze zredukowanymi modułami pamięci realizowały poprawnie zadane funkcje, generując identyczne wyniki z układami pierwotnymi (bez minimalizacji pamięci).

6. Podsumowanie

W artykule zaproponowano dwa autorskie algorytmy selekcji klas kompatybilności z zastosowaniem teorii hipergrafów. Obie metody stanowią rozwinięcie rozwiązania klasycznego, w którym pokrycie wierzchołkowe znajdowane jest na podstawie przekształceń macierzowych.

Szczegółowe wyniki badań pokazały, że zdecydowanie najbardziej efektywne jest zastosowanie algorytmu zachłannego. Metoda ta pozwala uzyskać lepsze rezultaty od metody tradycyjnej (średnio pamięć jest mniejsza o 16%) w znacznie krótszym czasie (średnio 12 razy szybciej). Warto także zaznaczyć fakt, że algorytm zachłanny w rozpatrywanym przypadku redukcji pojemności pamięci niewiele ustępuje metodzie dokładnej pod względem skuteczności (rozmiar pamięci uzyskanej po redukcji).

Współautorka – Monika Wiśniewska - jest stypendystką w ramach Poddziałania 8.2.2 "Regionalne Strategie Innowacji", Działania 8.2 "Transfer wiedzy", Priorytetu VIII "Regionalne Kadry Gospodarki" Programu Operacyjnego Kapitał Ludzki współfinansowanego ze środków Europejskiego Funduszu Społecznego Unii Europejskiej i z budżetu państwa.

7. Literatura

[1] De Micheli G.: Synthesis and Optimization of Digital Circuits. McGrawHill, 1994.
[2] Maxfield C.: The Design Warrior's Guide to FPGAs. Orlando, FL, USA: Academic Press, Inc., 2004.
[3] Łuba T.: Synteza układów cyfrowych. Praca zbiorowa pod redakcją prof. Tadeusza Łuby, Warszawa: WKŁ, 2003.
[4] Puri R., Gu J.: Microword Length Minimization in Microprogrammed Controller Synthesis", IEEE Trans. on Computer-Aided Design, 1993.
[5] Robertson E. L.: Microcode bit optimization is NP-complete, IEEE Trans. Comput., vol. C-28, pp. 316–319, Apr. 1979.
[6] Dasgupta S.: The Organization of Microprogram Stores. Computing Surveys, 1979.
[7] Hong S. K., Park I. C., Kyung C. M.: An O(n3logn) Heuristic for Microcode Bit Optimization, In ICCAD-90, pp. 180-183, 1990.
[8] Aho A. V., Hopcroft J. E., Ullman J. D.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, 1974.
[9] Berge C.: Graphs and Hypergraph. Amsterdam: North-Hols.r Mathematical Library, 1976.
[10] Eiter T., Gottlob G.: Hypergraph transversal computation and related problems in logic and AI, LNCS, pp. 549—564, Springer, 2002.
[11] Wiśniewska M.: Redukcja rozmiaru mikroinstrukcji w projektowaniu sterowników mikroprogramowanych, PAK, Nr 8, s. 575-577, 2009.
[12] Tkacz J., Adamski M.: Projektowanie sekwencyjnych układów cyfrowych z wykorzystaniem logiki sekwentów Gentzena", Przegląd Elektrotechniczny, R. 85, nr 7, pp. 196—199, 2009.

otrzymano / received: 13.03.2011
przyjęto do druku / accepted: 04.05.2011 *artykuł recenzowany*