Adam OPARA, Dariusz KANIA
POLITECHNIKA ŚLĄSKA INSTYTUT INFORMATYKI, INSTYTUT ELEKTRONIKI
ul. Akademicka 16, 44-100 Gliwice

Wykorzystanie dwupoziomowej optymalizacji do poprawy wyników syntezy z wykorzystaniem BDD

Dr inż. Adam OPARA

Ukończył studia na Wydziale Automatyki Elektroniki i Informatyki na Politechnice Śląskiej, obronił pracę magisterską w 2002 r., następnie rozprawę doktorską w 2009r. Jest pracownikiem Instytutu Informatyki. Jego zainteresowania naukowe to programowalne układy cyfrowe i układy mikroprocesorowe.

e-mail: Adam.Opara@polsl.pl

Dr hab. inż. Dariusz KANIA

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 1995, habilitacyjną w 2004r. Jest profesorem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe koncentrują się wokół programowalnych układów cyfrowych, sterowników przemysłowych, systemów mikroprocesorowych oraz zagadnień związanych z wykorzystaniem stabilografii w zagadnieniach rehabilitacji klinicznej.

e-mail: dkania@polsl.pl

Streszczenie

W artykule przedstawiona jest koncepcja syntezy ukierunkowanej na zrównoważoną optymalizację powierzchni i prędkości działania układu. Pierwszym etapem syntezy jest dekompozycja wierszowa wykorzystująca BDD, ukierunkowana na struktury PAL. Celem dekompozycji jest minimalizacja liczby bloków logicznych struktury programowalnej. Drugi etap syntezy jest ukierunkowany na optymalizację szybkości działania układu. Istotą dwupoziomowej optymalizacji jest odpowiednie wykorzystanie trójstanowych buforów wyjściowych. Uzyskane rezultaty eksperymentów dowodzą szczególnej efektywności proponowanych rozwiązań dla struktur CPLD zbudowanych z bloków typu PAL o niewielkiej liczbie iloczynów.

Słowa kluczowe: binarne diagramy decyzyjne (BDD), synteza logiczna, dekompozycja, CPLD.

Enhancing logic synthesis based on two-stage BDD decomposition by using two-level optimization

Abstract

This paper presents a concept of the original method of two-stage BDD-based decomposition combined with two-level PAL-oriented optimization. The aim of the proposed approach is oriented on the balanced (speed/area) optimization. The first step of the method is original PAL-oriented decomposition. The presented non-standard decomposition provides minimization of the implemented circuit area and reduction of necessary logic blocks in the programmable structure. This decomposition consists in sequential search for an input partition providing feasibility of implementation of the free block in one PAL-based logic block, containing a predefined number of product terms. In the presented algorithms the Reduced Ordered Binary Diagrams were used as an efficient representation of logic functions. The partitioning of the variables in a partition matrix is equivalent to the cut in the ROBDD diagram representing the logic function. To efficiently approximate the number of product terms in a sum of product form, the concept of path counting was developed. The second step of the proposed logic synthesis is oriented to the speed optimization. The original two-level optimization is based on utilizing tri-state buffers. The results of experiments prove that the presented approach is especially effective for CPLD structures which consist of PAL-based logic blocks containing a low number of product terms.

Keywords: decomposition, technology mapping, logic optimization, binary decision diagrams (BDD), CPLD.

1. Wstęp

Kluczowy wpływ na efektywność procesu syntezy ma sposób reprezentacji funkcji logicznych. Tablice prawdy wymagają dla funkcji o n zmiennych użycia 2^n komórek pamięci. Ta wykładnicza zależność sprawia, że ich znaczenie praktyczne jest bardzo niewielkie. Bardziej zwięzłą reprezentacją funkcji jest postać sumy iloczynów. Reprezentacja taka pozwala zapobiec skutkom wykładniczej zależności opisującej zajętość pamięci, jednak wymaga użycia czasochłonnych algorytmów redukcji. Znacznie efektywniej można opisywać funkcje wykorzystując binarne diagramy decyzyjne (BDD). Diagramy te zostały wprowadzone przez Akersa [1] a spopularyzowane przez Bryanta i Brace'a [2, 3].

Specyfika implementacji układów cyfrowych w strukturach programowalnych, sprowadzająca się do "zagospodarowania" gotowych zasobów logicznych. W przypadku struktur CPLD elementami, które należy najlepiej wykorzystać są bloki logiczne typu PAL zawierające ściśle określoną liczbę iloczynów dołączonych do bramki OR [7, 8, 13]. Bardzo często bloki logiczne typu PAL wyposażone są w trójstanowe bufory wyjściowe, umożliwiające dwukierunkowe wykorzystanie wyprowadzeń.

Celem artykułu jest przedstawienie oryginalnej metody syntezy przeznaczonej dla układów CPLD typu PAL. Metoda ta wykorzystuje elementy dekompozycji funkcji opisanych za pomocą BDD oraz dwupoziomową optymalizację podukładów uzyskiwanych w wyniku dekompozycji.

2. Podstawy teoretyczne

Binarny diagram decyzyjny jest skierowanym, acyklicznym grafem (drzewem) [4]. Węzły grafu skojarzone są z argumentami funkcji. Z każdego węzła wychodzą dwie krawędzie skojarzone z wartościami 0 i 1. Do każdego węzła grafu (oprócz korzenia) dochodzi, co najmniej jedna krawędź z węzłów znajdujących się na wyższym poziomie. Binarny diagram decyzyjny zawiera dwa węzły terminalne (liście), które skojarzone są z wartościami funkcji 0 lub 1. Analiza dróg występujących w binarnym diagramie decyzyjnym pozwala na określenie wartości poszczególnych zmiennych, dla których funkcja przyjmuje wartość 1 lub 0. W praktyce wykorzystuje się jedynie uporządkowane (Ordered BDD) i zredukowane uporządkowane diagramy decyzyjne (Reduced Ordered BDD). W uporządkowanych diagramach w każdej ścieżce zmienne występują w tej samej kolejności. Na danym poziomie diagramu występują węzły związane z tą samą zmienną. Diagramy zredukowane (ROBDD), zawierające przy danym uporządkowaniu zmiennych minimalną liczbę węzłów, uzyskuje się poprzez sklejanie odpowiednich węzłów i redukcję identycznych podgrafów [2, 4, 5].

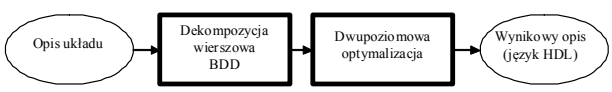
Głównym problemem syntezy, oprócz sposobu reprezentacji, jest sposób dekompozycji funkcji pozwalający na dopasowanie projektowanego układu do struktury układu programowalnego. Istota dekompozycji sprowadza się do podziału układu na bloki o zadanej liczbie wejść. Funkcja n-zmiennych $f(X)$, $X = x_1,x_2,...,x_n$, może być przedstawiona jako $h(g_1(X_b),..., g_m(X_b), X_f)$, gdzie X_b to zbiór związany (*bound set*), a X_f to zbiór wolny (*free set*) wtedy, gdy liczba wzorców kolumn matrycy podziału jest mniejsza lub równa 2^m. Podział zmiennych polega na szukaniu zbiorów X_b oraz X_f. Jeśli część wspólna zbiorów X_b, X_f jest pusta mówimy, że dekompozycja jest rozłączna.

Podział zmiennych jest równoważny dokonaniu cięcia w binarnym diagramie decyzyjnym. Węzły powyżej linii cięcia tworzą zbiór związany, a poniżej - zbiór wolny. Węzły poniżej cięcia, które są wskazywane przez przecięte krawędzie nazywane są węzłami odciętymi. Liczba węzłów odciętych odpowiada liczbie wzorców kolumn w dekompozycji Ashenhursta-Curtisa lub liczbie klas równoważności w dekompozycji Roth-Karpa [6]. W ogólnym przypadku dla n węzłów odciętych potrzeba do ich rozróżnienia $\lceil \log_2(n) \rceil$ bitów. Powstaje wtedy kilka funkcji g_i, i=1,…, $\lceil \log_2(n) \rceil$. Liczba węzłów odciętych zależy od wyboru zmiennych, które tworzą zbiór wolny i związany, czyli od kolejności zmiennych w grafie i od poziomu, na którym nastąpi cięcie. Szukając dekompozycji pozwalającej na realizację układu za pomocą k-wejściowych bloków LUT, poziom cięcia przeważnie ustala się na k licząc od korzenia diagramu. W przypadku układów CPLD typu PAL najbardziej skuteczna są dwa modele dekompozycji przedstawione w pracy [13], tzw. dekompozycja kolumnowa i dekompozycja wierszowa. Efektywniejszą pod względem właściwości dynamicznych uzyskiwanych rozwiązań jest dekompozycja wierszowa. Okazuje się jednak, że możliwe jest poprawienie parametrów dynamicznych uzyskiwanych rozwiązań poprzez dwupoziomową optymalizację ukierunkowaną na wykorzystanie powszechnie występujących w strukturach CPLD trójstanowych buforów wyjściowych.

3. Zaproponowana strategia syntezy dedykowana dla układów CPLD typu PAL

Zaproponowana strategia syntezy składa sie z dwóch głównych etapów:
- dekompozycji wierszowej wykorzystującej BDD,
- dwupoziomowej optymalizacji ukierunkowanej na wykorzystanie trójstanowych buforów wyjściowych.

Etapy te poglądowo przedstawia rysunek 1.



Rys. 1. Idea syntezy dedykowanej dla układów zbudowanych z bloków typu PAL
Fig. 1. The concept of logic synthesis for PAL-based devices

4. Dekompozycja wierszowa wykorzystująca BDD

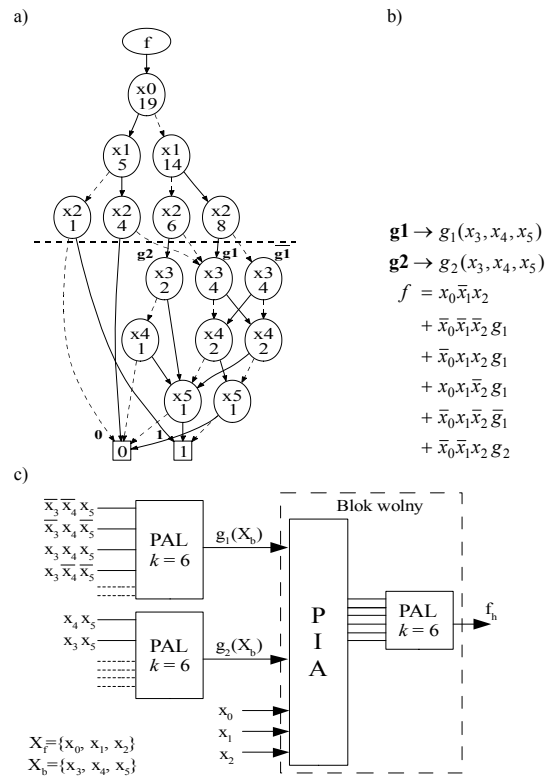
Typowo w procesie dekompozycji wierszowej do reprezentacji funkcji logicznych wykorzystuje się tablice dekompozycji [13]. Można jednak zaproponować algorytm wykorzystujący zredukowane binarne diagramy decyzyjne ROBDD (Reduced Ordered Binary Decision Diagrams), jako bardziej efektywną reprezentację funkcji [12]. Poza tym zastosowanie atrybutu negacji może dodatkowo poprawić wynik dekompozycji.

4.1. Szacowanie liczby iloczynów na podstawie liczby ścieżek

Kluczowym zagadnieniem syntezy układów cyfrowych realizowanych w strukturach CPLD jest określanie liczby iloczynów zminimalizowanej funkcji przedstawionej w postaci sumy iloczynów. W klasycznym podejściu używa się do tego celu np. algorytmu Espresso. W przypadku, gdy do reprezentacji funkcji logicznej wykorzystywane są diagramy BDD, można zastosować inne podejście. Każda ścieżka w diagramie prowadząca od korzenia do liścia 1 odpowiada jednemu iloczynowi. Zmieniając porządek zmiennych w diagramie można minimalizować liczbę ścieżek. Zaletą algorytmu obliczania liczby wymaganych iloczynów na podstawie ścieżek w diagramie jest jego mała złożoność obliczeniowa. Szczegółowo algorytm obliczania liczby iloczynów na postawie liczby ścieżek został omówiony w [12].

4.2. Dekompozycja dedykowana dla układów CPLD z zastosowaniem BDD

Istotą dekompozycji wierszowej jest poszukiwanie takiego podziału zmiennych, który zapewni realizację bloku wolnego w jednym bloku PAL o określonej liczbie iloczynów. Jednocześnie zaproponowany podział ma pozwolić na realizację układu przy jak najmniejszej liczbie wyjść bloku związanego [13]. Dokonanie podziału zbioru zmiennych na zbiór związany i wolny może nastąpić poprzez poprowadzenie linii cięcia w diagramie ROBDD. Zmienne skojarzone z węzłami powyżej linii cięcia należą do zbioru wolnego X_f, a poniżej – związanego X_b (odwrotnie niż w dekompozycji wykorzystującej ROBDD wzorowanej na dekompozycji Ashenhursta-Curtisa).



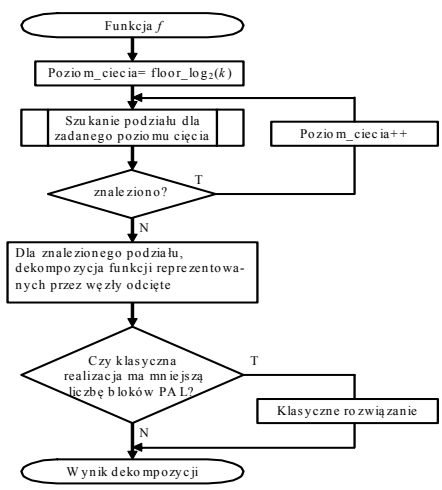
Rys. 2. Diagram funkcji z zaznaczonymi liczbami ścieżek do węzła „1” oraz realizacja układu w wyniku dekompozycji
Fig. 2. A diagram with number of paths of the function under consideration and hardware realisation

Na rysunku 2a przedstawiono diagram funkcji z zaznaczoną liczbą ścieżek do liścia „1”. Węzły wskazywane przez krawędzie przecięte linia cięcia nazywane są węzłami odciętymi. Każdy węzeł odcięty odpowiada wzorcowi wierszowemu [13]. Poprowadzenie linii cięcia poniżej trzeciego poziomu węzłów odpowiada podziałowi na zbiór wolny $X_f = \{x_0, x_1, x_2\}$ oraz związany $X_b = \{x_3, x_4, x_5\}$. Krotność wierszy $\mu(X_f|X_b)$ jest liczbą grup utworzonych przez wzorce wierszowe lub ich dopełnienia (z pominięciem wierszy pełnych i pustych). Dla podanej funkcji występują dwa wzorce wierszowe: g_1 i g_2 (krotności wierszy $\mu(X_f|X_b)$=2). Węzłom opisanym g_1 i $\overline{g_1}$ (rys. 2a) odpowiada jeden węzeł w diagramie z atrybutami negacji przypisanymi do krawędzi. Węzły odcięte w diagramie z atrybutami negacji odpowiadają grupom wzorców wierszowych, a co za tym idzie liczba węzłów odciętych (z pominięciem terminalnych) jest równa krotności wierszy. Wynika z tego, że krotność wierszy może być szybko określona na podstawie diagramu z atrybutem negacji. Węzły odcięte są równocześnie korzeniami diagramów funkcji realizowanych w bloku związanym (g_1, g_2) i w razie konieczności funkcje te poddawane są dekompozycji. Ścieżki prowadzące od korzenia diagramu funkcji f do węzłów odciętych odpowiadają iloczynom funkcji realizowanej w bloku związanym (rys. 2b). Dla podanego przy-

kładu wykorzystano po jednym blok typu PAL o 6 iloczynach do implementacji: bloku wolnego, funkcji g_1 oraz funkcji g_2 (rys. 2c).

Istotą algorytmu dekompozycji jest poszukiwanie odpowiedniego podziału zmiennych. Różne podziały otrzymywane są poprzez zmiany porządku zmiennych w diagramie przy ustalonym poziomie prowadzenia linii cięcia w diagramie.

Schemat blokowy algorytmu dekompozycji przedstawia rys. 3. Algorytm szukania podziału składa się z kilku etapów. W każdym etapie liczba zmiennych wolnych jest stała. Odpowiada to ustalonemu poziomowi cięcia diagramu. Poszukiwany jest podział, który zapewnia realizację bloku wolnego w jednym bloku typu PAL przy najmniejszej liczbie wyjść bloku związanego. Poszukiwania odpowiedniego podziału rozpoczyna się od poziomu cięcia równego $\lfloor \log_2(k) \rfloor$, gdzie k jest liczbą iloczynów w bloku typu PAL. Dla takiego poziomu cięcia zawsze istnieje podział (możliwe jest przeprowadzenie dekompozycji). Jeśli w bieżącym etapie znaleziono rozwiązanie, zwiększana jest liczność zbioru zmiennych wolnych o jeden, czyli poziom cięcia diagramu jest inkrementowany. Dla ostatniego znalezionego rozwiązania zapamiętywany jest zbiór węzłów odciętych i funkcje reprezentowane przez nie dalej rekurencyjnie poddawane są dekompozycji. Ostatnim krokiem algorytmu jest porównanie z realizacją klasyczną [13]. W przypadku uzyskania na danym etapie mniejszej liczby bloków PAL wybierana jest klasyczna realizacja.



Rys. 3. Uproszczony schemat dekompozycji dedykowanej dla układów CPLD typu PAL, wykorzystującej BDD
Fig. 3. A simplified schema of PAL decomposition with BDD application

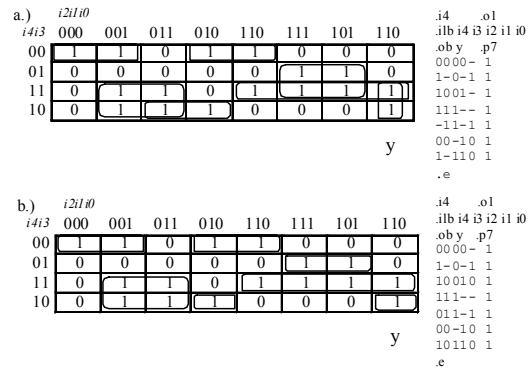
Podczas dekompozycji minimalna liczba iloczynów dla funkcji w postaci sumy iloczynów określana jest na podstawie liczby ścieżek. Obliczana jest liczba ścieżek prowadzących do liścia „0" oraz „1", po czym wybierana ta realizacja, która wykorzystuje mniejszą liczbę iloczynów (realizacja z warunków działania lub realizacja z warunków niedziałania z negacją wyrażenia).

Przedstawiony algorytm prezentuje jedynie główną ideę dekompozycji z dopasowaniem do liczby iloczynów w bloku PAL. W praktycznej implementacji algorytm został wzbogacony o szereg elementów poprawiających jego efektywność, takich jak np. sprawdzanie sensowności poszukiwania dekompozycji we wczesnych etapach algorytmu i eliminację przypadków, w których dekompozycja prowadzi do gorszych rozwiązań [12]. Na przykład, jeśli zminimalizowana funkcja opisana jest przez mniej niż $2k$ implikantów, wtedy dekompozycja nie zredukuje liczby bloków typu PAL, ponieważ jeden blok potrzebny jest do realizacji bloku wolnego i co najmniej jeden do realizacji bloku związanego.

Jako alternatywa dla algorytmu z liczeniem ścieżek przygotowano algorytm wykorzystujący algorytm espresso do oznaczania liczby iloczynów zminimalizowanej funkcji w poszczególnych etapach dekompozycji. Mimo iż algorytm z liczeniem ścieżek przyniósł dobre efekty zastosowanie (Espresso) prowadzi do dalszej poprawy wyników dekompozycji [12].

5. Dwupoziomowa optymalizacja

Kolejnym etapem strategii syntezy jest dwupoziomowa optymalizacja ukierunkowana na wykorzystanie trójstanowych buforów wyjściowych. Zadaniem tego etapu jest poprawienie własności dynamicznych układu. Punktem startowym optymalizacji jest zbiór wielowyjściowych implikantów $f: B^n \rightarrow \{0,1,-\}^m$ [14]. Optymalizacja rozpoczyna się od dwupoziomowej minimalizacji z rozszczepianiem implikantów, po którym następuje etap podziału implikantów na podzbiory o liczności mniejszej niż liczba iloczynów zawartych w bloku logicznym typu PAL. Ideę klasycznej minimalizacji i minimalizacji z rozszczepianiem implikantów obrazuje rys. 4.



Rys. 4. Idea minimalizacji z rozszczepianiem implikantów: a) wynik klasycznej dwupoziomowej minimalizacji, b) wynik dwupoziomowej minimalizacji z rozszczepianiem
Fig. 4. The essence of two-level splitting minimization: a.) results of the classical two-level minimization, b.) results of the two-level splitting minimization

Proces minimalizacji z rozszczepianiem implikantów rozpoczyna się od tradycyjnej dwupoziomowej minimalizacji wykonywanej za pomocą algorytmu espresso. Następnie, wykorzystując ideę rozszczepiania (rys. 4b), modyfikuje się wszystkie częściowo pokrywające się implikanty, w taki sposób by otrzymać zbiór implikantów, opisanych przez wyrażenia z możliwie jak największą liczbą liter.

Po minimalizacji z rozszczepianiem, kolejnym krokiem zaproponowanej strategii optymalizacji jest podział zbioru implikantów na odpowiednie podzbiory. Zbiór implikantów dzielony jest na podzbiory o liczności nie większej niż liczba iloczynów zawartych w bloku typu PAL (k). Wstępne wykonanie rozszczepienia stwarza lepsze warunki podziału, sprowadzające się do poszukiwania tzw. zmiennych podziału implikantów [7,13].

Rozważając funkcję z rys. 4, po wykonaniu minimalizacji z rozszczepianiem otrzymujemy wynik przedstawiony w skrajnej lewej kolumnie tablicy 1. Zakładając realizację układu w oparciu o 3-iloczynowe bloki typu PAL, należy dokonać podziału zbioru implikantów na dwa podzbiory **Y**1 i **Y**2, takie że liczność zbioru **Y**1 jest mniejsza lub równa liczbie iloczynów (k=3) zawartych w boku typu PAL (card(**Y**1)≤k), oraz (card(**Y**2)=min). Teoretyczne podstawy dokonywania podziału zbioru implikantów można znaleźć w [13].

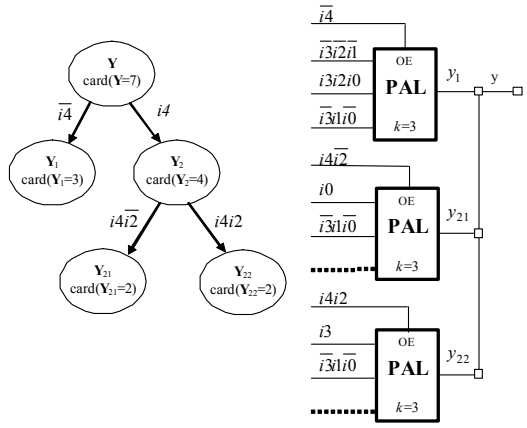
W rozważanym przykładzie, wykorzystując zmienną *i4*, można dokonać podziału na podzbiory **Y**1 i **Y**2, dla których card(**Y**1)=3 oraz card(**Y**2)=4. Uzyskiwane w wyniku podziału podzbiory implikantów, zaprezentowane są w kolejnych kolumnach tablicy 1. Pierwszy z nich określa funkcję y_1, która jest wybrana dla *i4* = 0 (tab. 1, kolumna 2), drugi podzbiór określa funkcję y_2 aktywną dla *i4* = 1 (tab. 1, kolumna 3).

W kolejnym kroku, wykorzystując zmienną *i2,* dokonywany jest podział implikantów funkcji y_2. Powstałe dwa kolejne podzbiory implikantów **Y**21 (funkcja y_{21}) i **Y**22 (funkcja y_{22}), dla których card(**Y**21) = card(**Y**22) = 2 < k, zaprezentowane są w czwartej i piątej kolumnie tablicy 1.

Schemat podziału funkcji y oraz realizacja układu z wykorzystaniem bloków typu PAL o 3 iloczynach została przedstawiona na rys. 5.

Tab. 1. Etapy podziału zbioru implikantów
Tab. 1. Partitioning of *y.pla* file

y.pla	y_1 active, if i_4=0	y_2 active, if i_4=1	y_{21} active, if i_4i_2=10	y_{22} active, if i_4i_2=11
.i 5 .o 1 .ilb i4,i3,i2,i1,i0 .ob y .p 7 0000- 1 1-0-1 1 10010 1 111-- 1 011-1 1 00-10 1 10110 1 .e	. .i 5 .o 1 . ilb i4,i3,i2,i1,i0 .ob y1 .p 3 0000- 1 011-1 1 00-10 1 .e OE 0----	.i 5 .o 1 . ilb i4,i3,i2,i1,i0 .ob y2 .p 4 1-0-1 1 10010 1 111-- 1 10110 1 .e OE 1----	.i 5 .o 1 . ilb i4,i3,i2,i1,i0 .ob y21 .p 2 1-0-1 1 10010 1 .e OE 1-0--	.i 5 .o 1 . ilb i4,i3,i2,i1,i0 .ob y22 .p 2 111-- 1 10110 1 .e OE 1-1--



Rys. 5. Wynik podziału i wpasowywania ukierunkowanego na struktury CPLD typu PAL
Fig. 5. The results of PAL-oriented partitioning and PAL mapping

6. Wyniki eksperymentów

Przydatność zaproponowanej strategii syntezy potwierdzona została szeregiem praktycznych eksperymentów wykonanych na standardowych układach testowych (benchmark). Dokonanie szerokiej gamy eksperymentów możliwe było dzięki implementacji algorytmów w prototypowych narzędziach [12, 13]. Wyniki doświadczeń zamieszczono w tablicy 2.

Tab. 2. Otrzymane rezultaty syntezy
Tab. 2. Synthesis results

Metody	Klasyczna		dekBDD+E		dekBDD+E+TS_opt	
Układ	l.bloków	l.warstw	l.bloków	l.warstw	l.bloków	l.warstw
5xp1.pla	22	2	19	2	18	2
9sym.pla	18	3	17	3	21	2
clip.pla	38	3	38	3	52	1
f51m.pla	22	2	17	3	16	2
rd53.pla	8	2	6	2	7	1
rd73.pla	36	3	16	3	16	2
rd84.pla	67	4	21	4	24	3
sao2.pla	14	2	14	2	19	1
xor5.pla	4	2	2	2	2	2
z5xp1.pla	22	2	19	2	18	2
z9sym.pla	18	3	17	3	21	2
cordic.pla	73	4	42	7	51	6
duke2.pla	60	2	60	2	71	1
e64.pla	65	1	65	1	65	1
inc.pla	16	2	16	2	15	1
t481.pla	90	4	31	7	35	6
squar5.pla	9	2	9	2	9	1
table3.pla	134	3	134	3	137	1
suma	716	46	543	53	597	37
	100%	100%	24%	15%	17%	-20%

W poszczególnych kolumnach zamieszczono uzyskane liczby bloków i warstw dla metod: klasycznej [12], dekompozycji z wykorzystaniem BDD bez dwupoziomowej optymalizacji (dekBDD+E) oraz metody z wykorzystaniem dwupoziomowej optymalizacji ukierunkowanej na użycie trójstanowych buforów wyjściowych (dekBDD+E+TS_opt).

Jak pokazują wyniki, zastosowanie dekompozycji przynosi redukcję liczby bloków, która jednak nie pociąga za sobą zmniejszenia liczby warstw. Zgodnie z oczekiwaniami dwupoziomowa optymalizacja dedykowana dla układów CPLD typu PAL jest szczególnie atrakcyjna, jeśli chodzi o poprawienie dynamicznych właściwości układu. W dziesięciu przypadkach z osiemnastu dzięki optymalizacji uzyskano zmniejszenie liczby warstw w stosunku do metody klasycznej i w 14 w stosunku do metody dekompozycji dekBDD+E. W stosunku do metody klasycznej sumarycznie dekompozycja przyniosła zmniejszenie liczby bloków o 24% i zwiększenie liczby warstw o 15%, natomiast połączona strategia – zmniejszenie liczby bloków o 17% oraz zmniejszenie liczby warstw o 20%. Dzięki optymalizacji, zamiast zwiększenia otrzymano zmniejszenie liczby warstw przy dodatkowym zmniejszeniu liczby bloków.

7. Podsumowanie

Zaprezentowana w artykule oryginalna metody syntezy może być z powodzeniem stosowana w praktyce inżynierskiej. Opis uzyskiwanych rozwiązań w języku opisu sprzętu (VHDL, Verilog) stwarza możliwość wykonania końcowego etapu odwzorowania technologicznego w dowolnym komercyjnym narzędziu syntezy. Przydatność proponowanych elementarnych rozwiązań potwierdzono szeregiem eksperymentów przestawionych między innymi w pracach [7, 8, 9, 12, 13].

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

8. Literatura

[1] Akers S. B.: Functional Testing with Binary Decision Diagrams, Eighth Annual Conf. on Fault-Tolerant Computing, 1978, pp. 75–82.
[2] Bryant R. E.: Graph-Based Algorithms for Boolean Function Manipulation, IEEE Trans. on Computer, 1986, vol. 35, no. 8, pp. 677–691.
[3] Brace K., Rudell R., Bryant R.: Efficient Implementation of a BDD Package, Proc. Design Automation Conference, 1990, p. 40–45.
[4] De Micheli G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, 1994.
[5] Minato S.: Binary Decision Diagrams and Applications for VLSI CAD, Kluwer Academic Publishers, Nov. 1996.
[6] Roth J.P., Karp R.M.: Minimization Over Boolean Graphs, IBM J. Res. Dev., 1962, pp. 227-238.
[7] Kania D.: Two-level logic synthesis on PALs, Electronics Letters, 1999, Vol.35, No. 11, pp. 879-880.
[8] Kania D., Kulisz J.: Logic synthesis for PAL-based CPLD-s based on two-stage decomposition, The Journal of Systems and Software 80, 2007, pp. 1129-1141.
[9] Opara A., Kania D.: Synteza wielowyjściowych układów logicznych prowadząca do wykorzystania wspólnych bloków logicznych, Pomiary Automatyka Kontrola, Szczecin 2007, ss. 39-42.
[10] Lai M.T., Pan K.R., Pedram M.: OBDD-Based Function Decomposition: Algorithms and Implementation, IEEE Transactions on CAD of Int. Circuits and Systems, Vol. 15, No. 8, 1996, pp. 977-990.
[11] Minato S.: Binary Decision Diagrams and Applications for VLSI CAD, Kluwer Academic Publishers, 1996.
[12] Opara A.: Dekompozycyjne metody syntezy układów kombinacyjnych wykorzystujące binarne diagramy decyzyjne, rozprawa doktorska, Politechnika Śląska, 2009.
[13] Kania D.: Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL, Gliwice, 2004.
[14] Brzozowski J.A., Łuba T.: Decomposition of Boolean Functions Specified by Cubes, Journal of Multi-Valued Logic & Soft Computing, vol. 9, pp. 377-417, Old City Publishing Inc., Philadelphia 2003.

otrzymano / received: 15.03.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*