Marcin KUBICA [1], Dariusz KANIA [2]
[1] AKADEMIA TECHNICZNO-HUMANISTYCZNA, KATEDRA ELEKTROTECHNIKI I AUTOMATYKI, ul. Willowa 2, 43-309 Bielsko-Biała
[2] POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI, ul. Akademicka 16, 44-100 Gliwice

Synteza logiczna zespołu funkcji ukierunkowana na minimalizację liczby wykorzystywanych bloków logicznych PAL w oparciu o zmodyfikowany graf wyjść

Mgr inż. Marcin KUBICA

Ukończył studia na Wydziale Budowy Maszyn i Informatyki Akademii Techniczno-Humanistycznej oraz na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Jest asystentem w Katedrze Elektrotechniki i Automatyki Akademii Techniczno-Humanistycznej w Bielsku-Białej. Jego zainteresowania naukowe koncentrują się wokół układów cyfrowych, systemów mikroprocesorowych. oraz technik informatyczno-pomiarowych.

e-mail: mkinz@wp.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 przedstawiono metodę implementacji zespołu funkcji prowadzącą do ograniczenia liczby wykorzystywanych bloków PAL. Istota metody tkwi w dopasowaniu opisu zespołu funkcji do charakterystycznej cechy każdego układu CPLD, jaką jest liczba iloczynów pojedynczego bloku PAL. Metoda wykorzystuje graf wyjść w zmodyfikowanej postaci, zawierający informacje na temat stopnia wykorzystania iloczynów w strukturze PAL. Wyniki eksperymentów wskazują, że wykorzystanie zmodyfikowanego grafu wyjść w procesie syntezy prowadzi do efektywniejszego wykorzystania zasobów struktury CPLD, w stosunku do metod implementacji opartych na klasycznym grafie wyjść.

Słowa kluczowe: synteza logiczna, graf wyjść, układ CPLD.

The Logic Synthesis of the Multi-Output Boolean Function Directed to PAL Logic Block Number Minimization Based on a Modified Graph's Nodes

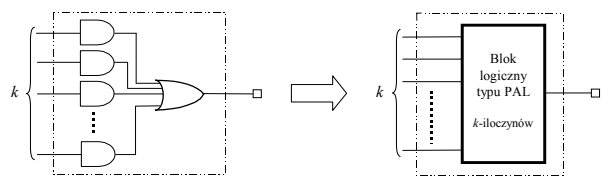
Abstract

The article is concerned with the implementation method of the multi-output Boolean function that leads to the limitation of the number of the PAL (Programmable Array Logic) logic blocks used. The essence of this technique is to match the description of a multi-output function to the distinctive feature of an each CPLD (Complex Programmable Logic Device) structure which is the number of terms of a single PAL block. This distinctive feature of a PAL block is best illustrated in the form of a picture (see Fig. 1) in which the number of terms is marked as *k*. Apart from that, the main purpose of the method is to apply a modified graph of outputs to present the degree to which terms were used in a given PAL block. In this article, the authors also present the operations of pasting and splitting in a modified graph of outputs thanks to which the degree of the terms used can be significantly improved. The process is presented in the form of three pictures (see Fig. 5, Fig. 6, Fig. 7). The experimental results show that the usage of a modified graph of outputs in the synthesis process enables to use the CPLD structure in a much more effective way (see Tab. 1) than in the case of the implementation method which is based on a classical graph of outputs. In the penultimate chapter proper conclusions were drawn on the experiment basis. The article ends with a bibliography list which presents all the works used by the authors while writing.

Keywords: logic synthesis, graph's nodes, CPLD structure.

1. Wprowadzenie

Nieefektywność narzędzi wspomagających projektowanie układów cyfrowych realizowanych w strukturach programowalnych wynika przede wszystkim z niedopasowania uniwersalnych algorytmów syntezy do specyfiki używanych układów. W przypadku układów CPLD, które w zdecydowanej większości wykorzystują bloki logiczne typu PAL (rys. 1), w początkowym etapie syntezy zwykle wykonuje się dwupoziomową minimalizację, a następnie realizuje każdą funkcję oddzielnie, wykorzystując dostępne w strukturze zasoby sprzętowe. Ponieważ możliwe jest dołączenie wybranego iloczynu tylko do jednej sumy, stąd minimalizacja jest wykonywana dla każdego wyjścia oddzielnie. Tego typu podejście, implementowane powszechnie w znanych autorom narzędziach wspomagających projektowanie, prowadzi do nieefektywnego wykorzystania struktury programowalnej.



Rys. 1. Struktura i schemat blokowy bloku logicznego typu PAL
Fig. 1. Structure and a block scheme of a PAL logic block

W literaturze, oprócz metody klasycznej, w której dominującą rolę odgrywa dwupoziomowa minimalizacja, znane są również inne zaawansowane metody syntezy [1, 2, 7, 8] często wykorzystujące różnorodne strategie dekompozycji. Niestety uzyskana tymi metodami redukcja liczby bloków w większości przypadków okupiona jest ogromnym wzrostem złożoności algorytmów syntezy. Znana jest również metoda realizacji zespołu funkcji wykorzystująca tzw. grafy wyjść [3, 4, 5, 6]. Stanowi ona alternatywę dla metody klasycznej i nie prowadzi do istotnego wydłużenia czasu syntezy.

Celem artykułu jest przedstawienie sposobu zwiększenia efektywności syntezy zespołu funkcji implementowanych w strukturach CPLD. Istota zaproponowanych pomysłów sprowadza się modyfikacji grafu wyjść, prowadzącej do lepszego dopasowania wyniku minimalizacji do architektury wykorzystywanego układu programowalnego, scharakteryzowanego liczbą iloczynów tworzących blok logiczny typu PAL.

2. Klasyczna metoda realizacji zespołu funkcji

Klasyczna metoda realizacji zespołu funkcji $f:B^n \rightarrow B^m$, gdzie $B=\{0,1\}$, w strukturach typu PAL związana jest z realizacją zminimalizowanych funkcji $f_i:B^n \rightarrow B$ ($i=1,2,...,m$) za pomocą bloków logicznych typu PAL zawierających k-iloczynów. Niech wyróżnik Δ_{fi} będzie liczbą dziesiętną równą liczbie implikantów, tworzących zbiór implikantów, dla których funkcja $f_i:B^n \rightarrow B$ przyjmuje wartość 1. Niech δ_{fi} oznacza liczbę bloków potrzebnych do realizacji i-tej funkcji. Jeżeli $\Delta_{fi} > k$, gdzie k jest liczbą iloczynów zawartych w bloku logicznym typu PAL, to realizacja implikantów wymaga użycia $\delta_{fi} = \lceil (\Delta_{fi} - k)/(k - 1) \rceil + 1$ bloków logicznych typu PAL zawierających k-iloczynów. Do realizacji m-funkcji,

gdy każda funkcja $f_i:B^n \rightarrow B$ (i=1,2,...,m) minimalizowana jest oddzielnie, konieczne jest użycie δ^1_f bloków:

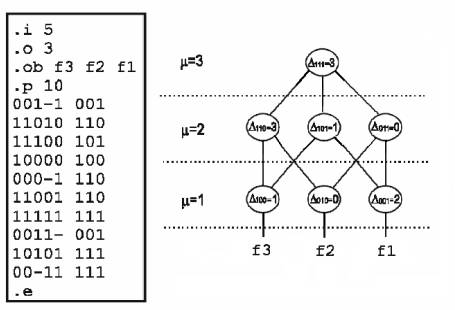
$$\delta^1_f = \sum_{i=1}^{m} \left(\left\lceil \frac{\Delta f_i - k}{k-1} \right\rceil + 1 \right) \tag{1}$$

Istota klasycznej realizacji zespołu funkcji sprowadza się do utworzenia osobnych, wzajemnie niezależnych struktur składających się z k-iloczynowych bloków typu PAL. Każda ze struktur realizuje pojedynczą funkcję, która w przypadku liczby implikantów większej od liczby iloczynów zawartych w bloku logicznym typu PAL jest strukturą wielopoziomową. Liczba warstw logicznych uzyskiwanej struktury można wyznaczyć z zależności

$$\xi_f = \max(\xi_{f1}, \xi_{f2}, ..., \xi_{fm}) \quad \text{gdzie } \xi_{fi} = \lceil \lg_k \Delta_{fi} \rceil. \tag{2}$$

3. Realizacja zespołu funkcji w oparciu o graf wyjść

Alternatywną metodą realizacji zespołu funkcji jest metoda oparta na analizę grafu wyjść, dokładnie przedstawiona w pracach [4, 5, 6]. Graf wyjść dla przykładowego zespołu przedstawiono na rys. 2.

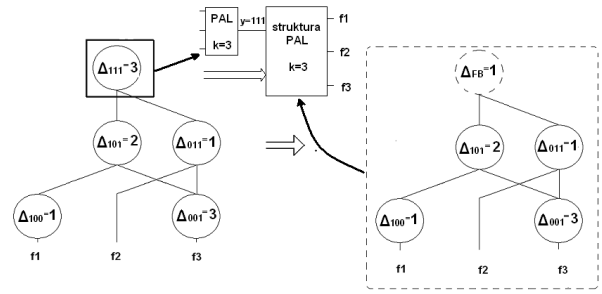


Rys. 2. Graf wyjść
Fig. 2. Graph of outputs

Wierzchołki grafu wyjść są skojarzone z odpowiednimi kombinacjami wektora wyjściowego y. Wierzchołki te są uszeregowane w zależności od liczby jedynek w wektorze wyjściowym. Liczbę jedynek w wektorze wyjściowym, odpowiadającym danemu wierzchołkowi, nazwano rzędem wierzchołka i oznaczono jako μ. W grafie połączono krawędziami wierzchołki, dla których odległość kodowa wektorów wyjściowych wynosi 1. Z każdym wierzchołkiem grafu wyjść skojarzony jest wyróżnik Δ_y. Wyróżnik ten określa dla ilu implikantów w opisie zespołu funkcji, występuje ten sam wektor wyjściowy y. Graf, w którym usunięto wierzchołki z wyróżnikami równymi 0, nazywamy zredukowanym grafem wyjść [4, 6].

Istota realizacji zespołu funkcji w oparciu o graf wyjść sprowadza się do realizacji niektórych grup implikantów w odrębnych blokach PAL, a następnie dołączeniu wyodrębnionych bloków do struktur realizujących poszczególne funkcje wchodzące w skład zespołu. Implementacja wyodrębnionych grup implikantów w osobnych blokach PAL prowadzi do ograniczenia liczby wykorzystywanych bloków PAL. Podstawy teoretyczne implementacji zespołów funkcji w strukturach CPLD typu PAL można znaleźć w [2]. Jeden krok implementacji wyodrębnionych implikantów skojarzonych z wierzchołkiem 3-go rzędu, w bloku logicznym PAL oraz sposób modyfikacji grafu prezentuje rys. 3.

Wyniki badań zawarte w pracach [2, 3, 5] wyraźnie wskazują, że realizacja zespołu funkcji w oparciu o graf wyjść prowadzi często do znacznego ograniczenia liczby użytych bloków PAL oraz może prowadzić do zmniejszenia liczby warstw logicznych. Przeprowadzone liczne eksperymenty pokazały, że możliwe jest jednak ulepszenie prezentowanej w pracach [2, 3, 5] metody, poprzez odpowiednie modyfikowanie grafu wyjść.



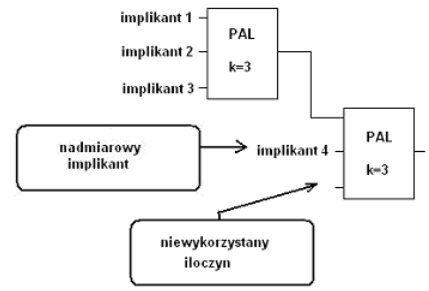
Rys. 3. Idea realizacji zespołu funkcji w oparciu o graf wyjść
Fig. 3. The idea of the multi-output Boolean function implementation based on a graph of outputs

4. Modyfikacja grafu wyjść ukierunkowana na minimalizację liczby bloków PAL

Istnieje możliwość polepszenia efektywności metody realizacji zespołu funkcji opartej o graf wyjść, dzięki dopasowaniu opisu zespołu funkcji do charakterystycznej dla każdego układu CPLD liczby iloczynów k [10]. W odróżnieniu do tradycyjnej realizacji zespołu funkcji w oparciu o graf wyjść, gdzie należało przeanalizować czy dany wierzchołek opłaca się realizować w odrębnej strukturze, należy tutaj wstępnie założyć istnienie odrębnych struktur PAL dla każdego wierzchołka grafu. W strukturach tych realizowane są implikanty skojarzone z danym wierzchołkiem oraz sprzężenia od struktur innych wierzchołków wyższego rzędu (odpowiadające krawędziom grafu). Liczbę tych sprzężeń oznaczono jako N_y. Liczba iloczynów α^i_k struktury składającej się z i bloków PAL o k iloczynach wyrażona jest zależnością (3).

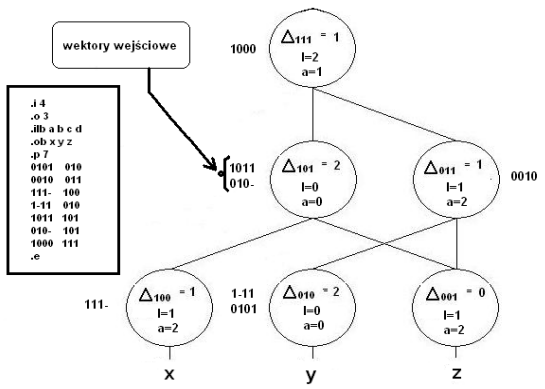
$$\alpha^i_k = \sum_{i=1} \left[k + (i-1)(k-1) \right] \tag{3}$$

Konieczne jest, aby liczba iloczynów struktury α^i_k była co najmniej równa sumie realizowanych implikantów w danym wierzchołku Δ_y oraz liczbie sprzężeń zwrotnych od wierzchołków wyższych rzędów N_y. Często zdarza się, że struktura zbudowana z niezbędnej liczby bloków PAL określonej jako i, posiada niewykorzystane iloczyny. Liczba niewykorzystanych iloczynów struktury skojarzonej z danym wierzchołkiem została oznaczona jako l_y. Do problemu niewykorzystanych iloczynów można podejść w nieco inny sposób: ile należało by usunąć implikantów, aby struktura, która składa się z i bloków zmniejszyła się o jeden blok? Liczbę tę określono jako nadmiarowość implikantów a_y. Problem niewykorzystanych iloczynów przedstawia rysunek 4. Z rysunku 4 widać, że usunięcie jednego implikantu spowoduje zmniejszenie struktury o jeden blok PAL (a_y = 1). Wdać również, że dla struktury zbudowanej z minimalnej liczby bloków (i=2), liczba niewykorzystanych iloczynów wynosi jeden (l_y =1).



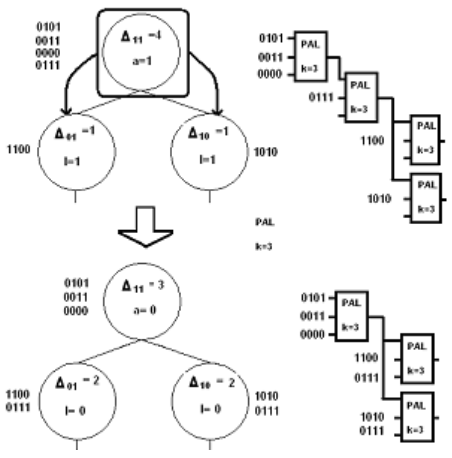
Rys. 4. Problem niewykorzystanych iloczynów w strukturach PAL
Fig. 4. The problem of unused terms in the PAL structures

Idea realizacji zespołu funkcji oparta na zmodyfikowanym grafie wyjść polega na zagospodarowaniu niewykorzystanych iloczynów w strukturze skojarzonej z danym wierzchołkiem, poprzez realizację w niej implikantów ze struktur o nadmiarowej licznie implikantów [10]. Konieczne jest zatem taka modyfikacja tradycyjnego grafu wyjść, aby graf dla każdego wierzchołka zawierał informacje na temat stopnia wykorzystania iloczynów w strukturze z nim skojarzonej. W ten sposób powstaje zmodyfikowany graf wyjść, w którym zamieszczone są informacje na temat liczby niewykorzystanych iloczynów l_y oraz liczby nadmiarowych implikantów a_y w wierzchołkach. Konieczne staje się umieszczenie informacji na grafie, jakie implikanty są realizowane w strukturze związanej z danym wierzchołkiem. Przykład zmodyfikowanego grafu wyjść ukazuje rysunek 5.



Rys. 5. Zmodyfikowany graf wyjść (k=3)
Fig. 5. A modified graph of outputs (k=3)

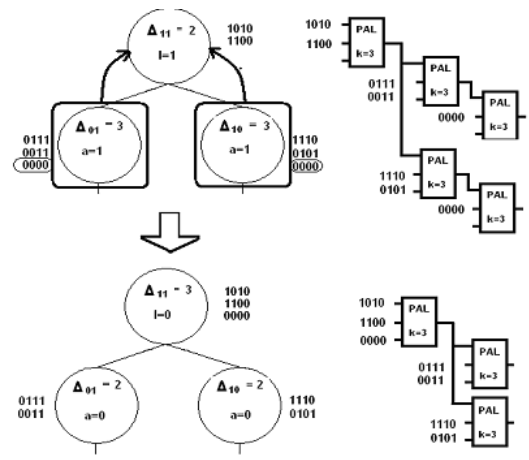
Konieczne jest utworzenie pewnych narzędzi, które umożliwiłyby „przesunięcie" nadmiarowych implikantów do struktur z niewykorzystanymi iloczynami. Opracowano dwie metody przemieszczania implikantów w zmodyfikowanym grafie wyjść [10]. Pierwsza z nich to rozszczepianie wektora wyjściowego. Rozszczepianie pozwala na przemieszczanie implikantów z wierzchołków wyższych rzędów do wierzchołków niższych rzędów, które są w sposób pośredni lub bezpośredni dołączone do analizowanego wierzchołka. Metoda ta polega na podziale wektora wyjściowego y analizowanego wierzchołka, w taki sposób, aby uzyskać kilka wektorów wyjściowych takich, że odpowiadające im wierzchołki mają takie struktury PAL w których posiadają wolne iloczyny. Dzięki temu procesowi realizacja nadmiarowego implikantu odbywa się w strukturach posiadających pierwotnie wolne iloczyny, co prowadzi do ograniczenia liczby używanych bloków PAL. Idee rozszczepiania wektora wyjściowego przedstawia rysunek 6.



Rys. 6. Idea rozszczepiania wektora wyjściowego (k=3)
Fig. 6. The idea of splitting of output vector (k=3)

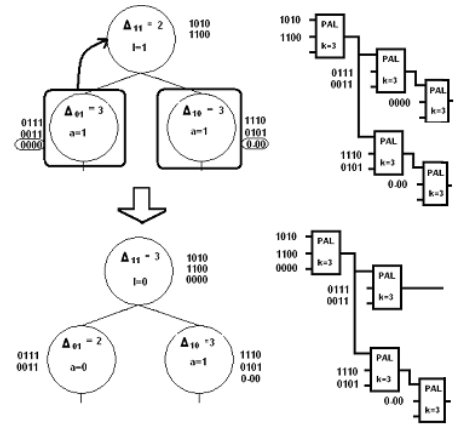
Widać na nim, że dzięki procesowi rozszczepiania udało się zmniejszyć liczbę bloków PAL z 4 do 3. Drugą metodą pozwalającą na przemieszczenie nadmiarowych implikantów w zmodyfikowanym grafie wyjść jest sklejanie wektorów wyjściowych. Dzięki sklejaniu wektorów wyjściowych możliwe jest przesunięcie nadmiarowych implikantów w kierunku wierzchołków wyższych rzędów. Metoda ta sprowadza się do sklejania wektorów wyjściowych (czyli przeprowadzenia sumowania bitowego) skojarzonych z wierzchołkami, których struktury posiadają nadmiarowość implikantów. Proces ten jest efektywny, gdy w wyniku sklejenia powstanie taki wektor, że struktura realizująca ten wektor posiada wolne iloczyny. Warunkiem koniecznym przeprowadzenia procesu sklejania jest to, aby wektorom wyjściowym, które będą poddane procesowi sklejania odpowiadały te same wektory wejściowe, dla każdego z analizowanych wierzchołków (wektory wejściowe umieszczone są na zmodyfikowanym grafie wyjść).

Ideę procesu sklejania przedstawia rysunek 7. Widać, że proces sklejania wektorów wyjściowych doprowadził do ograniczenia liczby bloków z 5 do 3.



Rys. 7. Idea sklejania wektorów wyjściowych (k=3)
Fig. 7. The idea of pasting of output vectors (k=3)

Proces sklejania można również wykonać w przypadku, gdy poszukiwany wektor wejściowy jest zawarty w innym wektorze w skutek występowania stanu "-". Sklejanie jest wtedy mniej efektywne, gdyż nie można usunąć implikantu (w którego części wejściowej na kluczowym bicie jest stan „-") ze struktury PAL skojarzonej z wierzchołkiem, któremu odpowiada jeden z wektorów składowych docelowego wektora y. Proces sklejania w takim wypadku również może być opłacalny, co przedstawia rysunek 8.



Rys. 8. Idea sklejania wektorów wyjściowych dla wektora wejściowego zawartego w innym wektorze (k=3)
Fig. 8. The idea of pasting of output vectors for an input vector included in another vector (k=3)

Analizę zmodyfikowanego grafu wyjść należy zacząć od wierzchołka najwyższego rzędu a zakończyć na wierzchołkach rzędu drugiego (wierzchołków pierwszego rzędu nie da się rozszczepić, niemożliwe jest również uzyskanie wierzchołka pierwszego rzędu w procesie sklejania). Analiza pojedynczego wierzchołka powinna rozpoczynać się od określenia czy dany wierzchołek w strukturze PAL z nim skojarzonej, posiada wolne iloczyny. Jeżeli nie, to należy przejść do kolejnego wierzchołka. Jeżeli tak, to należy określić czy da się dla niego przeprowadzić efektywny proces sklejania lub rozszczepienia wektora wyjściowego. Należy zadecydować, który z tych procesów da lepsze rezultaty, a następnie go wykonać i przejść do kolejnego wierzchołka.

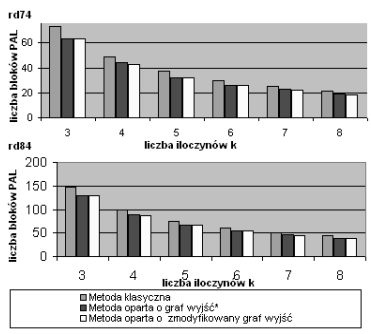
5. Wyniki eksperymentów

W celu oszacowania efektywności opracowanych metod napisano program w języku C++, który wykonywał procesy sklejania lub rozszczepiania wektorów wyjściowych przy interakcji z użytkownikiem. Użytkownik decydował, który wierzchołek chce analizować i jakie procesy chce na nim wykonywać. Pomimo wielu wad takiej formy funkcjonowania aplikacji udało się w niektórych wypadkach poprawić efektywność wykorzystania zasobów, co przedstawia tabela 1 oraz rysunek 9.

Tab. 1. Wyniki eksperymentów, LB – liczba bloków PAL, LW – liczba warstw logicznych
Tab. 1. Experimental results, LB – the number of PAL blocks, LW – the number of logic layers

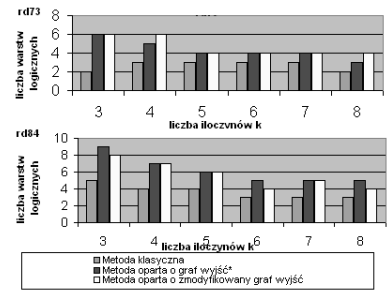
Metoda realizacji	Liczba iloczynów	Układy testowe [LB/LW]					
		rd53	rd73	rd84	clip	sao2	misex1
Metoda klasyczna (MK)	k = 3	17/3	73/2	147/5	80/4	37/3	21/3
	k = 4	12/2	49/3	99/4	54/3	25/3	15/2
	k = 5	9/2	37/3	74/4	41/3	20/2	13/2
	k = 6	7/2	30/3	61/3	34/3	16/2	10/2
	k = 7	7/2	25/3	51/3	29/2	14/2	9/2
	k = 8	6/2	21/2	44/3	24/2	11/2	9/2
Metoda oparta o graf wyjść* (GW)	k = 3	15/4	63/6	129/9	66/6	29/5	15/4
	k = 4	11/2	44/5	88/7	46/5	20/4	12/4
	k = 5	8/2	32/4	67/6	37/5	16/4	11/4
	k = 6	7/2	26/4	54/5	32/5	13/4	10/4
	k = 7	6/2	23/4	46/5	27/4	11/3	10/4
	k = 8	6/2	19/3	39/5	25/4	10/3	10/4
Metoda oparta o zmodyfikowany graf wyjść (ZGW)	k = 3	15/4	63/6	129/8	65/6	29/5	15/4
	k = 4	11/2	43/6	87/7	45/5	20/4	12/3
	k = 5	8/2	32/4	66/6	36/4	16/4	11/3
	k = 6	6/3	26/4	54/4	30/4	13/4	8/3
	k = 7	6/2	22/4	45/5	27/3	11/3	8/2
	k = 8	6/2	18/4	39/4	23/3	9/3	8/2

* bez rozbijania wierzchołków



Rys. 9. Wykresy przedstawiające liczbę użytych bloków logicznych do zaimplementowania układów rd73 i rd84 dla różnych metod syntezy
Fig. 9. Diagrams presenting the number of the logic blocks used to implementation rd73 and rd84 benchmarks for various synthesis methods

Oczywiście uzyskane efekty są zależne od tego, jakie operacje i na jakich wierzchołkach użytkownik chce wykonywać oraz w jakiej kolejności. Konieczne jest opracowanie spójnego algorytmu. Widać również zmianę w liczbie warstw logicznych układów po zastosowaniu opisanych metod, co przedstawia rysunek 10. Ważnym wnioskiem płynącym z eksperymentów jest fakt, że proces sklejania jest efektywniejszy od procesu rozszczepiania wektora wyjściowego. Należy pamiętać, że przewaga procesu sklejania odnosi się do takich implikantów, w których jest niewielka liczba stanów „-". Jeżeli zespół funkcji w wektorach wejściowych posiada sporą liczbę stanów „-" to opisane metody nie są zbyt efektywne. Sytuację może poprawić wprowadzenie minimalizacji zespołu funkcji logicznych w ramach każdego wierzchołka.



Rys. 10. Wykresy przedstawiające liczbę warstw logicznych dla układów rd73 i rd84 dla różnych metod syntezy
Fig. 10. Diagrams presenting the number of the logic layers for rd73 and rd84 benchmarks for various synthesis methods

6. Podsumowanie

Zaprezentowano w artykule sposób poprawienia efektywności wykorzystania zasobów logicznych w układach CPLD. Sposób ten sprowadza się do skorygowania opisu zespołu funkcji tak, aby dopasować go do charakterystycznego dla każdego układu CPLD parametru, jakim jest liczba iloczynów (k) pojedynczego bloku PAL. Dopasowanie to uzyskuje się po przez procesy sklejania i rozszczepiania wektora wyjściowego, które oparte są o zmodyfikowany graf wyjść. Wyniki eksperymentów wskazują, iż procesy te doprowadzają do zamierzonego celu, jakim jest zmniejszenie liczby wykorzystanych boków PAL. Widać również, że proces ten wpływa na liczbę warstw logicznych. Wydaje się bardzo prawdopodobne wykorzystanie procesów sklejania oraz rozszczepiania przy próbie ograniczenia liczby warstw logicznych. Oczywiście istnieje konieczność prowadzenia dalszych prac nad stworzeniem algorytmu opisującego cały proces. Z całą pewnością można liczyć na uzyskanie jeszcze lepszych rezultatów.

7. Literatura

[1] Ciesielski M. J., Yang S., PLADE: A two-stage PLA decomposition, IEEE Trans. on Computer-Aided Design, 11(8), 943-954, August 1992.
[2] Kania D.: Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL. Zeszyty Naukowe Politechniki Śląskiej Nr 1619, Wydawnictwo Politechniki Śląskiej, Gliwice 2004.
[3] Kania D.: Decomposition-based synthesis and its application in PAL-oriented technology mapping. Proc. of 26-th Euromicro Conference, IEEE Comp. Soc. Press, Maastricht, 2000, pp. 138-145.
[4] Kania D.: Efektywna metoda realizacji zespołu funkcji w strukturach PAL. Kwart. Elektroniki i Telekom. 1999, 45, z. 3-4, ss. 433-444.
[5] Kania D: Realizacja układów kombinacyjnych w strukturach MACH. Kwartalnik Elektroniki i Telekomunikacji, 2001, 47, z. 1, ss. 65-74.
[6] Kania D: A technology mapping algorithm for PAL-based devices using multi-output function graphs. Proc. of 26-th Euromicro Conference, IEEE Comp. Soc. Press, Maastricht, 2000, pp. 146-153.
[7] De Micheli G.: Synthesis and optimization of digital circuits, McGraw-Hill International Editors, USA 1994.
[8] Saucier G., Sicard P., Bouchet L.: Multi-level synthesis on PAL's. Proc. European Design Automation Conference, Glasgow, March 1990, pp.542-546.
[9] www.cbl.ncsu/edu Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University.
[10] Kubica M. : Synteza logiczna przeznaczona dla matrycowych struktur typu PAL. Praca mgr., Politechnika Śląska, prom. D. Kania.

otrzymano / received: 10.12.2010
przyjęto do druku / accepted: 06.06.2011 *artykuł recenzowany*