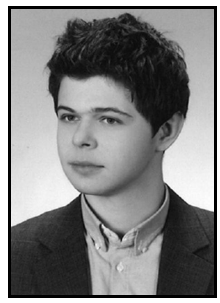
Łukasz ŁAWROCKI [1], **Robert CZERWIŃSKI** [2]
[1] MENTOR GRAPHICS
[2] POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI

Metoda syntezy logicznej ukierunkowana na wykorzystanie elementu XOR

Mgr inż. Łukasz ŁAWROCKI

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej, kierunek Elektronika i Telekomunikacja. Jest pracownikiem firmy Mentor Graphics oraz studentem Wydziału Automatyki, Elektroniki i Informatyki na kierunku Informatyka. Jego zainteresowania naukowe koncentrują się wokół problemów związanych z syntezą i dekompozycją funkcji logicznych oraz zastosowaniem języków opisu sprzętu, w szczególności SystemC.



e-mail: lukasz_lawrocki@mentor.com

Dr inż. Robert CZERWIŃSKI

Studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej ukończył w 2001 roku. Na tymże wydziale obronił pracę doktorską w 2006 roku. Obecnie jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe skupiają się wokół projektowania układów cyfrowych z wykorzystaniem układów logiki programowalnej oraz systemów mikroprocesorowych.

e-mail: robert.czerwinski@polsl.pl

Streszczenie

W artykule przedstawiono nową metodę syntezy logicznej przeznaczonej dla matrycowych struktur programowalnych CPLD. Opisywana metoda wykorzystuje elementy znane z rozłącznej dekompozycji Curtisa, jednocześnie pozwalając ukierunkować syntezę logiczną na efektywne wykorzystanie elementu XOR. Wstępne wyniki eksperymentów potwierdzają skuteczność opracowanej metody syntezy logicznej.

Słowa kluczowe: układy programowalne, CPLD, PAL, XOR, synteza logiczna, dekompozycja.

The XOR oriented logic synthesis

Abstract

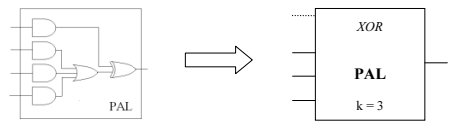
This paper presents XOR-based logic synthesis approach for CPLD devices. A novel decomposition-based logic synthesis is introduced in the paper. The method is based on the Curtis functional decomposition and is developed paying special attention to utilizing XOR gates. As opposed to the Curtis functional decomposition, the number of complements of column patterns in described method is known, and it isn't greater than four. This feature allows carrying out the process of decomposition using only *n*-1 column patterns, with *n* occurring in the logical function. Each pattern appears in a logical function, so it is linked to a number of vectors. The process of decomposition should be carried out in such a way, that pattern excluded from the analysis was related to the greatest possible number of vectors. This implies to obtain the best result of decomposition of logic functions. The way of encoding column patterns is also presented in the paper. The described method was compared with the method in the Quartus II. Primary experimental results, carried out using thirteen benchmarks, prove an effectiveness of the method. Ten percentage improvement in performance compared to bests Quartus II methods was achieved. However, the method has few weaknesses and should be treated as a work in progress.

Keywords: CPLD, PAL, XOR, logic synthesis, decomposition.

1. Wstęp

Matrycowe struktury CPLD (Complex Programmable Logic Devices) stanowią, obok układów FPGA (Field Programmable Gate Array), podstawowe układy logiki programowalnej. Krótkie czasy propagacji sygnałów przez strukturę, stałe i przewidywalne opóźnienia, stały się głównymi przyczynami ich popularności. Architektura wykorzystywana w strukturach CPLD, jest charakterystyczna dla układów PAL (Programmable Array Logic). Podstawową komórkę stanowi blok logiczny typu PAL, zawierający określoną liczbę iloczynów (zazwyczaj od 3 do 8) dołączonych do bramki sumy logicznej (rys. 1). Bloki logiczne układów CPLD wyposaża się standardowo w: konfigurowalne przerzutniki, wyjściowe bufory trójstanowe, bramki XOR.

Jednym z podstawowych problemów syntezy logicznej dedykowanej dla struktur CPLD jest optymalne wykorzystanie iloczynów dostępnych w blokach logicznych. Dzięki możliwości wykorzystania bramek XOR, występujących w blokach logicznych większości układów CPLD, realnym stało się opracowanie metod dekompozycji ukierunkowanych na efektywne wykorzystanie struktury bloku typu PAL.



Rys. 1. Struktura logiczna bloku PAL zawierająca bramkę XOR
Fig. 1. Structure of PAL-based logic block consisting the XOR gate

2. Rozłączna dekompozycja Curtisa dla struktur CPLD typu PAL

Twórcą klasycznej teorii dekompozycji był Ashenhurst, który opublikował swoje prace w drugiej połowie lat 50. Badania te były następnie rozwijane przez Curtisa. W efekcie czego, opracowany został klasyczny model dekompozycji Ashenhursta–Curtisa, określany zwyczajowo dekompozycją rozłączną Curtisa [2].

Rozłączna dekompozycja Curtisa nie uwzględnia możliwości zagospodarowania elementów XOR. Jednak jej charakterystyczna cecha w postaci procesu kodowania wejść oraz wyjść pozwala w łatwy sposób zmodyfikować metodę tak, aby umożliwiała ona wykorzystanie bramek XOR. Jest to główny powód wykorzystania właśnie tej metody dekompozycji, jako podstawy opisanych w dalszej części artykułu sposobów dekompozycji funkcji logicznych.

Twierdzenie o dekompozycji funkcjonalnej

Twierdzenie o dekompozycji funkcjonalnej stanowi rozwinięcie twierdzenia o dekompozycji prostej:

$$v(X_2 \mid X_1) \le 2^p \Leftrightarrow f(X_2, X_1) = F[g_1(X_1), g_2(X_1), ..., g_p(X_1), X_2]$$

Opierając się na takim modelu dekompozycji, możliwy jest podział bloku opisanego uporządkowaną parą liczb (liczba wejść, liczba wyjść) wynoszącą (n,1), na dwa bloki o następujących parametrach: (n_z, p) oraz (n – n_z + p, 1) [2].

3. Ukierunkowanie rozłącznej dekompozycji Curtisa na wykorzystanie elementu XOR

Przedstawiony model rozłącznej dekompozycji Curtisa można w bardzo prosty sposób ukierunkować na wykorzystanie elementu XOR. Umożliwia to efektywniejszą realizację funkcji w strukturach typu PAL [3, 4]. Dodatkowo element XOR stanowi część toru sygnału, więc włączenie go do procesu dekompozycji nie zwiększa czasu propagacji sygnału przez wykorzystywaną strukturę programowalną. Sposób wykorzystania bramek XOR w rozłącznej dekompozycji Curtisa zostanie przedstawiony na przykładzie.

ab \ cde	000	001	011	010	110	111	101	100
00	0	1	0	1	0	1	1	1
01	1	0	1	0	1	0	0	0
11	0	1	0	1	0	1	1	1
10	1	1	1	0	1	1	0	0
	A	B	A	C	A	B	C	C

y = f(a, b, c, d, e)

Rys. 2. Siatka Karnaugha przykładowej funkcji *y=f(a,b,c,d,e)*
Fig. 2. Karnaugh map of the example function *y=f(a,b,c,d,e)*

Przykład 1: Niech będzie dana funkcja f : $B^5 \rightarrow B$ przedstawiona za pomocą siatki Karnaugha. Funkcja zostanie zrealizowana za pomocą bloków logicznych typu PAL zawierających 3 iloczyny.

Dla analizowanego przykładu krotność kolumnowa (złożoność kolumnowa matrycy podziałów) wynosi $v(X_2 \mid X_1) = 3$. Oznacza to, że w siatce Karnaugha występują 3 różniące się między sobą wzorce kolumn, które zostały oznaczone literami A, B, C. Liczba możliwych kombinacji uzyskanych na wyjściach bloku związanego odpowiada maksymalnej liczbie wzorców, które mogą zostać w sposób jednoznaczny zakodowane. Tak więc liczba wyjść bloku związanego opisywana przez zmienną *p* powinna wynosić 2 –zgodnie z twierdzeniem Curtisa: $v(X_2 \mid X_1) \leq 2^p$.

Analizując siatkę Karnaugha z rys. 2 można jednakże zauważyć, iż wzorce A i C stanowią swoją negację (rozumianą w sensie kombinacji 0 i 1). Poprzez zanegowanie wzorca A możliwe jest uzyskanie wzorca C (0101 -> 1010). Sytuacja taka określana jest mianem **wzorców dopełniających** i występuje wtedy i tylko wtedy, gdy w zbiorze par komórek należących do dwóch różnych kolumn nie występują pary (1,1) i (0,0).

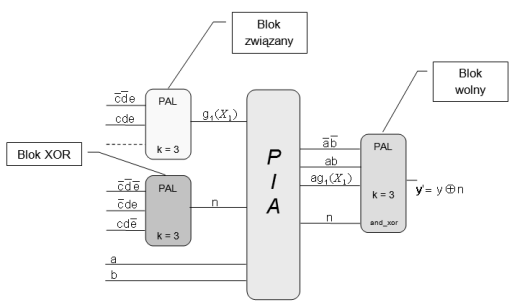
Dzięki tej właściwości możliwa jest modyfikacja siatki Karnaugha do postaci przedstawionej na rysunku 3. Efektem tych zmian jest zmniejszenie złożoności kolumnowej, która dla funkcji *f'*(*a*,*b*,*c*,*d*,*e*) wynosi 2 (rys. 3).

ab \ cde	000	001	011	010	110	111	101	100
00	1	1	1	1	1	1	1	1
01	0	0	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1
10	0	1	0	0	0	1	0	0
	$\overline{A}$	B	$\overline{A}$	A	$\overline{A}$	B	A	A

y = f'(a, b, c, d, e)

Rys. 3. Siatka Karnaugha funkcji *y=f(a,b,c,d,e)* z wzorcami dopełniającymi
Fig. 3. Karnaugh map of the function *y=f(a,b,c,d,e)* with complementary patterns

W takiej sytuacji, zgodnie z twierdzeniem Curtisa, blok związany może mieć tylko jedno wyjście (*p=1*). Fakt ten przekłada się bezpośrednio na brak potrzeby dokonywania ekspansji liczby iloczynów oraz umożliwia realizację funkcji $g(X_1)$ jedynie przy użyciu jednego bloku logicznego.



Rys. 4. Realizacja funkcji z przykładu 1
Fig. 4. A realization of the function presented in example 1

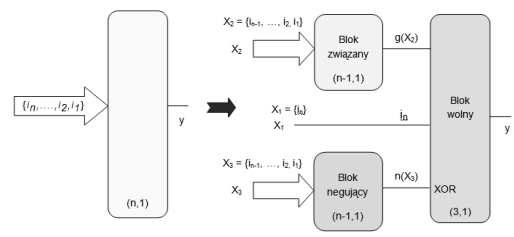
Zysk w postaci zmniejszenia liczby bloków logicznych typu PAL, niezbędnych do realizacji funkcji $g(X_1)$, obarczony jest potrzebą zanegowania funkcji f'(a,b,c,d,e) dla trzech kombinacji wektorów c,d,e = {000,011,110}. Ponieważ standardowo blok logiczny występujący w strukturach CPLD zawiera jeden iloczyn dołączony do bramki XOR, nie ma możliwości negacji funkcji f'(a,b,c,d,e) dla wektorów c,d,e={000,011,110} bez wykorzystania dodatkowych zasobów. Należy więc użyć kolejnego bloku logicznego do realizacji funkcji $n = \overline{c}\,\overline{d}\,\overline{e} + \overline{c}de + cd\overline{e}$. W takiej sytuacji funkcję f(a,b,c,d,e) należy wyrazić w następujący sposób: $f = f' \oplus n$. Realizację funkcji f przedstawia rysunek 4.

4. Dekompozycja kolumnowa ukierunkowana na wykorzystanie elementu XOR

Rozłączna dekompozycja Curtisa oraz jej rozszerzenie umożliwiające wykorzystanie elementu XOR stanowi sztandarowy przykład ukierunkowania znanych metod dekompozycji na syntezę przeznaczoną dla układów CPLD typu PAL. Wydaje się jednak, że takie podejście do problemu nie jest kompleksowe. Dekompozycja kolumnowa posiada ograniczenia opłacalności jej zastosowania, związane ze złożonością kolumnową matrycy podziałów. Tak więc, z definicji nie może być wykorzystywana w procesie syntezy pewnych funkcji logicznych. Ponadto jej ukierunkowanie na wykorzystanie elementu XOR, znajduje zastosowanie jedynie w przypadku syntezy funkcji logicznych zawierających wzorce dopełniające. Skutkuje to uniemożliwieniem wykorzystania bramek XOR w procesie syntezy części funkcji logicznych.

Alternatywną drogą rozwiązania problemu jest opracowanie metod, których architektura bazuje na elemencie XOR. Pozwoliłoby to na analizę pełnego spektrum funkcji logicznych, bez wprowadzania dodatkowych ograniczeń możliwości zastosowania metody. Propozycją takiego podejścia do zagadnienia, jest przedstawiona w dalszej części artykułu **dekompozycja kolumnowa ukierunkowana na wykorzystanie elementu XOR**.

Funkcja $y = f(i_n,....,i_2,i_1) = f(X_1,X_2,X_3)$ może zostać przedstawiona w postaci $y = F(g(X_2), n(X_3), X_1)$. Zbiory X_1, X_2 oraz X_3 nazywane są odpowiednio zbiorem wolnym, związanym oraz negującym, przy czym $X_1 = \{i_n\}$ oraz $X_2 \cap X_3 = \{ i_{n-1}, \ldots, i_2, i_1\}$. Przedstawiony model dekompozycji dzieli blok scharakteryzowany uporządkowaną parą liczb (liczba wejść, liczna wyjść) wynoszącą (n,1), na trzy inne o następujących parametrach: (n-1,1), (n-1,1) i (3,1) [1]. Rys. 5 odzwierciedla przedstawiony opis.



Rys. 5. Dekompozycja kolumnowa ukierunkowana na użycie bramki XOR
Fig. 5. Column decomposition oriented on the XOR use

W dekompozycji kolumnowej ukierunkowanej na wykorzystanie elementu XOR argumenty funkcji opisującej blok wolny stanowią złożenie argumentów funkcji dekomponowanej oraz części wyjściowej bloku związanego oraz negującego. Dla rozłącznej dekompozycji Curtisa wynikiem takiego złożenia jest funkcja *m*-argumentowa. W dekompozycji kolumnowej ukierunkowanej na wykorzystanie elementu XOR omawiana funkcja zawiera jedynie 3 argumenty (rys. 5). Zatem funkcja opisująca blok wolny wyraża się zależnością $y = y' \oplus n(X_3)$, przy czym funkcja y' jest funkcją dwuargumentową: $y' = f(i_n, g(X_2))$.

Dla rozłącznej dekompozycji Curtisa oraz jej zmodyfikowanej formy wykorzystującej bramkę XOR, liczba występujących wzorców zależy jedynie od sposobu przyporządkowania argumentów funkcji do zbiorów X_1 oraz X_2. Ponadto w procesie dekompozycji analizowane są wszystkie występujące wzorce. Dla dekompozycji kolumnowej ukierunkowanej na wykorzystanie elementu XOR podejście to zostało zmienione. Ponieważ liczba elementów zbiorów X_2 oraz X_3 jest określona w definicji jako *n-1* (gdzie *n* to liczba argumentów funkcji pierwotnej), mogą zostać wyznaczone najwyżej cztery rodzaje wzorców. Liczba wzorców jest więc wartością stałą. Pozwala to w sposób jednoznaczny stwierdzić wystąpienie danego wzorca na podstawie analizy pozostałych. To z kolei umożliwia wyłączenie takiego wzorca z procesu dekompozycji.

Przykład 2: Niech będzie dana funkcja f : $B^5 \rightarrow$ B przedstawiona za pomocą siatki Karnaugha (rys.6). Poszukujemy realizacji funkcji na blokach logicznych typu PAL zawierających 3 iloczyny.

Rys. 6. Siatka Karnaugha funkcji $f: B^5 \rightarrow B$
Fig. 6. Karnaugh map of the function $f: B^5 \rightarrow B$

Analizę przykładu należy rozpocząć od wyznaczenia kombinacji wektorów {bcde}, które odpowiadać będą wystąpieniom poszczególnych wzorców w siatce Karaugha z rys. 6. Wektory te przedstawiają się następująco:

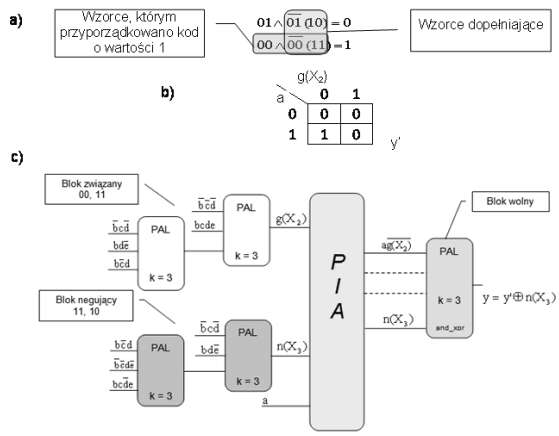
$$wzorzec\ \begin{matrix}0\\1\end{matrix} \rightarrow \{0\text{-}11, 011\text{-}, 1\text{-}00, 100\text{-}\},\ wzorzec\ \begin{matrix}0\\0\end{matrix} \rightarrow \{000\text{-}, 1111\},$$
$$wzorzec\ \begin{matrix}1\\1\end{matrix} \rightarrow \{010\text{-}, 1\text{-}10, 101\text{-}\}.\ wzorzec\ \begin{matrix}1\\0\end{matrix} \rightarrow \{0010, 1101\},$$

Blok związany oraz blok negujący realizują funkcje będące sumą wyznaczonych wektorów. Podstawowa metoda kojarzenia wektora z funkcją realizowaną w danym bloku związana jest z przyjętym kodowaniem. Funkcja realizowana w bloku związanym stanowi sumę wektorów, którym odpowiadają wzorce z przyporządkowanym kodem o wartości 1. Natomiast z funkcją realizowaną w bloku negującym związane są te wektory, które odpowiadają wzorcom dopełniającym.

Proces kodowania, w jego podstawowej formie, opiera się na analizie wyznaczonych wektorów w kontekście ich liczby dla danego wzorca. Jak wykazano, z wzorcem 01 skojarzona jest największa liczba wektorów. Wzorzec taki powinien zostać wyłączony z dalszej analizy. Odbywa się to poprzez przyporządkowanie mu odpowiedniej wartości kodowej – równej 0. Również wzorcowi stanowiącemu jego dopełnienie (wzorzec 10) jest przyporządkowana wartość 0. Należy pamiętać, iż wzorzec, któremu przyporządkowano kod o wartości 0 oraz, który nie został określony jako dopełniający, nie może zostać skojarzony z żadnym z bloków. Przyjęte kodowanie zostało przedstawione na rys. 7a.

Siatka Karnaugha, odzwierciedlająca funkcję y', tworzona jest w sposób następujący: w kolumnę, dla której argument $g(X_2)$ przyjmuje wartość 0 należy wpisać wzorzec, któremu przyporządkowano kod o wartości 0, natomiast dla kolumny $g(X_2) = 1$, wpisywanym wzorcem jest taki, któremu przyporządkowano kod o wartości 1 (rys. 7b).

Pozostałe wzorce – określone jako dopełniające – zostaną wyznaczone za pomocą $n(X_3)$, stanowiącego argument funkcji charakteryzującej blok wolny: $y = y' \oplus n(X_3)$. Realizację funkcji f przedstawia rysunek 7c. Pełna wersja prezentowanej metody dekompozycji przedstawiona została w [1].



Rys. 7. a) Przyjęte kodowanie wzorców, b) siatka Karnaugha opisująca funkcję $y' = f(i_n, g(X_2))$, c) realizacja funkcji z przykładu 2
Fig. 7. a) Proposed pattern encoding, b) Karnaugh map of the function $y' = f(i_n, g(X_2))$, c) A realization of the function presented in example 2

5. Przeprowadzone eksperymenty

Zaproponowaną metodę syntezy, oznaczoną jako DKUnWEX porównano z metodami zaimplementowanymi w systemie Quartus II firmy Altera. W tym celu wykorzystano 13 powszechnie znanych układów testowych. Uzyskane wyniki w postaci liczby bloków logicznych zostały przedstawione w tabeli 1. Syntezę w systemie Quartus II przeprowadzono dla rodziny MAX7000 [5], dla trzech różnych strategii: bez wykorzystania bramek XOR i ekspanderów (Metoda A), z wykorzystaniem bramek XOR (Metoda B) oraz z wykorzystaniem bramki XOR i ekspanderów (Metoda C).

Tab. 1. Wyniki eksperymentów dla wybranych układów testowych (liczba bloków)
Tab. 1. Experimental results of the selected benchmarks (the number of blocks)

	Quartus II			DKUnWEX
Układ testowy	Metoda A	Metoda B	Metoda C	
b12	14	12	17	13
bw	28	28	28	28
con1	2	2	2	2
ex1010	332	286	243	249
f51m	21	21	19	16
Inc	12	11	14	11
misex1	7	7	7	7
misex2	22	19	25	18
plce	9	9	9	12
soa2	17	17	13	13
squar5	9	9	9	9
x2	7	7	7	7
xor5	3	2	2	2

Największą trudność porównania z systemem Quartus II stanowił zupełny brak przewidywalności uzyskiwanych wyników, np. dla układu testowego b12 metoda nieuwzględniająca bramek XOR i ekspanderów prowadziła do użycia 14 bloków logicznych, metoda uwzględniająca bramki XOR już tylko 12 bloków, a metoda uwzględniająca zarówno bramki XOR, jak i ekspandery prowadziła do użycia aż 17 bloków logicznych.

Zastosowanie dekompozycji kolumnowej ukierunkowanej na wykorzystanie elementu XOR prowadzi do uzyskania takich samych wyników, w stosunku do najlepszych wyników uzyskanych za pomocą systemu Quartus II, aż w ośmiu przypadkach. W trzech przypadkach zaproponowana metoda prowadzi do uzyskania wyników gorszych, a w dwóch przypadkach lepszych od najlepszego wyniku systemu Quartus II. Dla układu testowego f51m wynik okazał się być lepszy o blisko 16%.

6. Wnioski

W artykule przedstawiono nową metodę dekompozycji kolumnowej ukierunkowaną na efektywne wykorzystanie bramek XOR, zawartych w blokach logicznych układów CPLD. W większości przypadków uzyskane wyniki eksperymentów potwierdzają skuteczność zaproponowanego podejścia. Proponowana metoda może stanowić alternatywę dla przedstawionych w artykule metod dekompozycji bazujących na rozłącznej dekompozycji Curtisa.

Należy wyraźnie podkreślić, iż prezentowana metoda jest rozwiązaniem nowym i posiada pewne ograniczenia. Niemniej jednak uzyskane wyniki skłaniają autorów do dalszych prac, m.in. nad takimi zagadnieniami jak: analiza stanów nieokreślonych czy przeprowadzenie dekompozycji w kontekście zespołu funkcji.

7. Literatura

[1] Ławrocki Ł.: Synteza zespołów funkcji w strukturach CPLD typu PAL wykorzystująca elementy XOR. Praca magisterska, Politechnika Śląska, Gliwice, 2008.
[2] Curtis H.A.: The Design of switching Circuits. Princeton: D. van Nostrand Company, 1962.
[3] Kania D., Grabiec W.: Synteza logiczna przeznaczona dla struktur CPLD z elementami XOR. Pomiary Automatyka Kontrola, 2007, ss. 54-56.
[4] Kania D., Grabiec W.: Dekompozycja zespołu funkcji wykorzystująca elementy XOR, Pomiary Automatyka Kontrola, 2008. ss. 502-504.
[5] Altera, Dokumentacja techniczna układów MAX7000.

Artykuł recenzowany