2019

Dekompozycja funkcji boolowskich

TADEUSZ ŁUBA

FUNKCJA BOOLOWSKA , UKŁADY KOMBINACYJNE, MIMIMALIZACJA, REDUKCJA ARGUMENTÓW, DEKOMPOZYCJA FUNKCJONALNA I LINIOWA, ANALIZA DANYCH, REDUKCJA ATRYBUTÓW

Rewolucyjny rozwój technologii mikroelektronicznych spowodował, że projektowanie systemów cyfrowych może być realizowane wyłącznie za pomocą komputerowych narzędzi projektowania przystosowanych do przetwarzania dużych ilości różnorodnych danych. Celem materiałów dydaktycznych jest omówienie zaawansowanych metod syntezy logicznej niezbędnych zarówno w projektowaniu systemów cyfrowych jak też w analizie i eksploracji danych. Dlatego głównymi zagadnieniami omawianymi w materiałach są metody minimalizacji i dekompozycji funkcji boolowskich, jak też redukcja atrybutów i indukcja reguł decyzyjnych. Takie ujęcie tych zagadnień jest zgodne z pilną potrzebą ważnych zastosowań takich jak: dystrybucja adresów IP, skanowanie wirusów, wykrywanie niepożądanych danych, itp. Nie mniejsze potrzeby stosowania zaawansowanych metod syntezy logicznej dotyczą analizy i eksploracji danych. Inaczej mówiąc celem tego podręcznika jest przygotowanie przyszłych inżynierów do umiejętnego wykorzystania ogromnego potencjału syntezy logicznej o czym świadczą wyniki prezentowanych metod i algorytmów.

Spis treści

1 Dekompozycja funkcji boolowskich

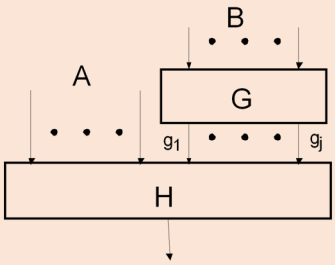
1.1 Metoda klasyczna

Niech będzie dana funkcja boolowska f: $\{0, 1\}^n \rightarrow \{0, 1\}$ oraz pewien podział zmiennych $X = \{x_1,..., x_n\}$ na dwa rozłączne zbiory B (*bound set*) i A (*free set*). Tablicą dekompozycji funkcji f (*decomposition chart*) nazywamy macierz dwuwymiarową o kolumnach etykietowanych wartościami zmiennych funkcji f ze zbioru B oraz o wierszach etykietowanych wartościami zmiennych funkcji f ze zbioru A. Elementami macierzy M są wartości, jakie przyjmuje funkcja f na wektorach złożonych z odpowiednich etykiet i-tego wiersza i j-tej kolumny. Liczbę istotnie różnych kolumn tej macierzy ze względu na ich zawartość oznaczamy symbolem $\nu(A|B)$ (*column multiplicity*).

TWIERDZENIE 3.1 [3.5].
Niech będzie dana funkcja boolowska f oraz podział zbioru zmiennych wejściowych funkcji f na dwa rozłączne zbiory A i B, wówczas:

$$f(A,B) = h(A,g_1(B),..., g_j(B)) \Leftrightarrow \nu(A|B) \leq 2^j$$

Schemat takiej dekompozycji jest przedstawiony na rys. 3.1.



Rys. 1.1. Schemat blokowy dekompozycji funkcjonalnej

Twierdzenie powyższe można uogólnić na przypadek zespołów funkcji. Niech będzie dana rodzina funkcji boolowskich F, wówczas:

$$F(A,B) = H(A,G(B)) \text{ dla } G(B) = (g_1(B),..., g_j(B)) \Leftrightarrow \nu(A|B) \leq 2^j$$

Dla funkcji z tablicy 3.1 kolumny są adresowane trzema zmiennymi $\{x_1, x_2, x_3\}$ względem których funkcja f jest symetryczna. Wiersze natomiast są adresowane zmiennymi $\{x_4, x_5\}$.

Tablica 1.1

x_4x_5 \ $x_1x_2x_3$	000	001	010	100	110	101	011	111
00	1	1	1	1	0	0	0	0
01	0	1	1	1	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	1	1	1	0

Grupy kolumn o adresach {(001), (010), (100)} i {(110), (101), (011)} są odpowiednio równe. Zatem istnieje dekompozycja, w której zmienne {x_1, x_2, x_3} są przeadresowywane na {g_1, g_2} (tablica 3.2a).

Wówczas funkcja *f* może być specyfikowana w tablicy, w której adresami kolumn są {g_1, g_2}, a wierszy odpowiednio wartości zmiennych {x_4, x_5} (patrz tablica 3.2b). Tablice te są jednocześnie tablicami prawdy funkcji reprezentujących składowe dekompozycji, tzn. bloki *G* i *H*.

Tablica 1.2

a)

x_1	x_2	x_3	g_1	g_2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0

b)

x_4x_5 \ g_1g_2	00	01	10	11
00	1	1	0	0
01	0	1	0	0
10	0	0	0	0
11	0	0	1	0

Niestety postępowanie takie w ogólnym przypadku funkcji nie w pełni określonych nie może być już tak bezpośrednie. Przykład takiej bardziej skomplikowanej sytuacji omówimy dokonując dekompozycji funkcji podanej w tabl. 3.4.

Tablica 1.3

ab \ *cde*	000	001	010	011	100	101	110	111
00	1	–	0	1	–	0	1	0
01	–	–	–	–	1	1	–	–
10	–	0	1	0	0	–	0	1
11	0	1	–	–	–	–	–	–
	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7

Otóż w tym przypadku, ze względu na nieokreśloności funkcji, nie można bezpośrednio podjąć decyzji, które kolumny w tablicy dekompozycji są takie same. Wynika to z faktu, że kreskom reprezentującym punkty nieokreśloności funkcji można przypisać wartość albo 0 albo 1, w rezultacie zaliczając odpowiednią kolumnę do – być może – innego zbioru kolumn identycznych. Dlatego w tym przypadku wygodnie będzie wszystkie

kolumny, dla których w poszczególnych wierszach tablicy dekompozycji nie występują sprzeczne wartości (0 i 1), nazywać kolumnami zgodnymi. Zgodne są na przykład kolumny K_0 i K_3, gdyż w ich poszczególnych wierszach, jako wartości funkcji, występują wyłącznie: 1 (w kolumnie K_0) i 1 (w kolumnie K_3), oraz odpowiednio, – i –; – i 0; wreszcie 0 i –. Natomiast kolumny K_0 i K_1 są niezgodne, gdyż w wierszu oznaczonym 11 mają wartości funkcji odpowiednio 0 i 1. Nietrudno zauważyć, że tak definiowane pary kolumn zgodnych tworzą relację zgodności, której własności: zwrotność, symetryczność i nieprzechodniość uprawniają do grupowania kolumn zgodnych w maksymalne klasy zgodności. Wypisując więc dla funkcji z tabl. 3.4 wszystkie pary zgodne: {K_0, K_3}, {K_0, K_4}, {K_0, K_6}, {K_1, K_3}, {K_1, K_4}, {K_1, K_5}, {K_1, K_6}, {K_2, K_5}, {K_2, K_7}, {K_3, K_4}, {K_3, K_6}, {K_4, K_5}, {K_4, K_6}, {K_5, K_7}, łatwo możemy obliczyć – stosując algorytm omówiony w rozdz. 1 – maksymalne klasy zgodności:

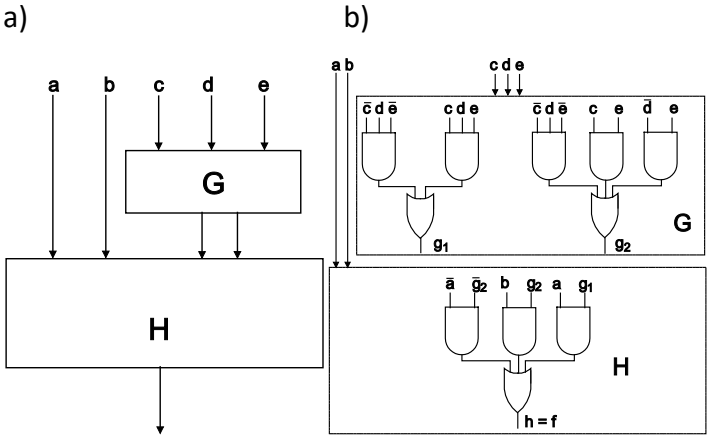
{K_0, K_3, K_4, K_6},
{K_1, K_3, K_4, K_6},
{K_1, K_4, K_5},
{K_2, K_5, K_7}.

Ze zrozumiałych względów niektóre kolumny występują w więcej niż jednej klasie zgodności. Musimy więc podjąć decyzję o wyborze rozłącznych klas zgodności (nie muszą być w tym przypadku maksymalne) pokrywających wszystkie kolumny tablicy dekompozycji, tzn. każda kolumna musi być reprezentowana w jednej klasie zgodności. Jedno z możliwych rozwiązań jest:{K_0, K_3, K_4, K_6}, {K_2, K_5}, {K_2, K_7}.

Po połączeniu kolumn należących do jednej klasy w jedną, uzyskuje się tablice funkcji *G* i *H* oraz wyrażenia boolowskie tych funkcji:

$$g_1 = \bar{c}d\bar{e} + cde$$
$$g_2 = \bar{c}d\bar{e} + ce + \bar{d}e$$
$$h = \bar{a}\bar{g}_2 + bg_2 + ag_1$$

Funkcjom *G* i *H* odpowiada schemat blokowy oraz logiczny przedstawiony odpowiednio na rys. 3.2a oraz 3.2b.

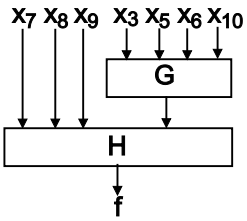


Rys. 1.2. Schemat blokowy (a) i logiczny (b) funkcji GiH

Przedstawiona wyżej metoda, jakkolwiek łatwa do interpretacji graficznej, w obliczeniach analitycznych bardziej skomplikowanych przykładów staje się zbyt pracochłonna.

Na przykład dla funkcji TL27 (patrz tab. 2.28) odpowiednia tablica dekompozycji – uzyskana dla reduktu $R_{10} = \{x_3, x_5, x_6, x_7, x_8, x_9, x_{10}\}$ – jest przedstawiona w tab. 3.4.

Tablicę tę skonstruowano przy założeniu, że zbiory zmiennych bezpośrednich oraz pośrednich są odpowiednio $A = \{x_7, x_8, x_9\}$, $B = \{x_3, x_5, x_6, x_{10}\}$. Spostrzeżenie, że w tablicy tej – w celu obliczenia składowych *G* i *H* dekompozycji – należy „skleić" kolumny oznaczone wektorami: 0000, 0011, 0100, 0101, 1000, 1001, 1010, 1011, 1111 oraz 0010, 0110, 0111, 1100, 1101, 1110, nie jest zadaniem łatwym. Spostrzeżenie to prowadzi do wniosku, że funkcję TL27 można zrealizować, jak na rys. 3.3.



Rys. 1.3. Realizacja funkcji TL27

Tablica 1.4

x_3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_5	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_6	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_{10}	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x_7x_8x_9$															
000	–	–	–	1	–	0	–	1	–	–	1	–	–	–	–
001	0	–	0	–	–	–	1	–	–	–	–	1	–	–	–
010	–	1	–	–	–	–	–	–	–	0	–	–	–	–	–
011	–	0	–	–	1	–	–	1	–	–	–	0	–	–	–
100	–	–	–	–	–	1	–	–	–	–	–	–	1	–	–
101	–	–	–	–	–	–	–	–	–	–	–	–	0	0	1
110	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1
111	–	–	1	–	–	–	–	–	1	–	–	–	–	–	1

1.2 Dekompozycja funkcjonalna metodą rachunku podziałów

1.2.1 Dekompozycja funkcjonalna – model podstawowy

W tym punkcie zajmiemy się analitycznym opisem dekompozycji funkcji boolowskich. Do tego celu posłużymy się omówionym w rozdz. 1 rachunkiem podziałów. W szczególności wyprowadzimy warunki dekompozycji dla funkcji boolowskich opisanych podziałami $P_1, P_2, \ldots, P_n, P_f$ określonymi na zbiorze K ponumerowanych wektorów tablicy prawdy funkcji f.

Na początek zajmiemy się poszukiwaniem realizacji $f = h(U, g(V, W))$, gdzie:

$$U = \{x_{i_1}, \ldots, x_{i_r}\},\ V = \{x_{i_{r+1}}, \ldots, x_{i_t}\},\ w = \{x_{j_1}, \ldots, x_{j_t}\}$$ [1)]

a ponadto

$$W \subseteq U, U \cup V \subseteq X, U \cap V = \Phi \text{ oraz } P(x_{i_1}) \cdot P(x_{i_2}) \cdots P(x_{i_t}) \leq P_f.$$

Sens powyższych ustaleń wynika z faktu, że dla danego U zbiór V może być wyznaczony bezpośrednio z reprezentacji argumentowej RA_f ograniczonej do par p, q: $\{p, q\} \subseteq$ BLOK(P_U), gdzie $P_U = P_{i_1} \ldots P_{i_r}$.

Oznaczając tak zdefiniowaną reprezentację przez RA_g, łatwo stwierdzić, że V jest rozwiązaniem równania $RA_g = 1$. Ponadto przez P_V, P_W będziemy oznaczać podziały na K określone zbiorami V i W, tzn.:

$$P_V = P(x_{i_{r+1}}) \cdot P(x_{i_{r+2}}) \cdot \ldots \cdot P(x_{i_t}) \qquad P_W = P(x_{j_1}) \cdot P(x_{j_2}) \cdot \ldots \cdot P(x_{j_p})$$

TWIERDZENIE 3.2

Jeżeli $f: \boldsymbol{B}^n \rightarrow \{0,1,-\}$ oraz $X = U \cup V$ jest zbiorem argumentów funkcji f, to:

$f = h(U, g(V, W))$

wtedy i tylko wtedy, gdy istnieje dwublokowy podział $\Pi_g \geq P_V \cdot P_W$ taki, że:

$P_U \cdot \Pi_g \leq P_f$,

gdzie $g: \boldsymbol{B}^{t-r+p} \rightarrow \{0,1\}$, przy czym $g(k) = c$, natomiast $k \in K$, $c \in \{0,1\}$, jeżeli $k \in \Pi_g^c$.

W szczególności $f = h(U, g(V))$ wtedy i tylko wtedy, gdy istnieje $\Pi_g \geq P_V$, dla którego $P_U \cdot \Pi_g \leq P_f$ (jest to tzw. dekompozycja rozłączna [3.15].

Uzasadnienie jest natychmiastowe, jeżeli zauważymy, że dwublokowy podział $\Pi_g \geq P_{V,W}$ jednoznacznie reprezentuje funkcję g, natomiast $P_U \cdot \Pi_g$ funkcję h. Otóż każdy blok H podziału $P_{V,W}$ jest wzajemnie jednoznaczną reprezentacją wektora o składowych $(x_{j_1}, \ldots, x_{j_p}, x_{i_{r+1}}, \ldots, x_{i_t})$, gdyż skoro $H \in P_V \cdot P_W$, to dla

---

1) **Uwaga:** w opisie tym zmieniamy oznaczenia zbiorów A i B z modelu Curtisa, wprowadzając oznaczenia typowe dla metod dekompozycji w ujęciu rachunku podziałów [3.3].

każdego P_a spośród $P_{j_1} \ldots P_{j_p}, P_{i_{r+1}} \ldots P_{i_t}$ blok *H* należy do ustalonego bloku P_a^c podziału P_a. Zapisując ten fakt w formie przyporządkowania:

$H \rightarrow$ wektor o składowych binarnych $c \in \{0,1\}$

i uwzględniając, że blokom tym są w podziale Π_g przyporządkowane:

$$0, \text{gdy } H \in \Pi_g^0$$
$$1, \text{gdy } H \in \Pi_g^1$$

uzyskujemy naturalną reprezentację *g*: $\boldsymbol{B}^{t\text{-}r+p} \rightarrow \{0,1\}$. Z kolei, skoro $P_U \cdot \Pi_g \leq P_f$, to $P_{i_1} \ldots P_{i_r} \cdot P(g) \leq P(f)$ i tym samym $f = h(x_{i_1}, \ldots, x_{i_r}, g)$ .Natomiast wartości *h* wynikają (analogicznie jak poprzednio) z przynależności bloków $P_U \cdot \Pi_g$ do bloków podziału P_f.

Jak wynika z twierdzenia 3.2 najistotniejsze w obliczeniu dekompozycji funkcji boolowskiej jest obliczenie podziału Π_G. Algorytm obliczania Π_G można sprowadzić do algorytmu obliczania MKZ. Oznaczmy bloki podziału P_V: P_V=(B_1,…,B_i,…,B_j,…,B_N) i przyjmijmy przez poddział γ_{ij} rozumieć podział uzyskany z P_V przez sklejenie B_i oraz B_j w jeden blok i pozostawienie pozostałych bloków bez zmiany, tzn.: γ_{ij} =(B_1,…,B_iB_j,…,B_N).

Dwa bloki B_i i B_j podziału P_V są zgodne, jeśli podział γ_{ij} spełnia warunek twierdzenia o dekompozycji: $P_U \bullet \gamma_{ij} \leq P_F$. W przeciwnym przypadku B_i oraz B_j są niezgodne.

W ten sposób obliczanie Π_G można sprowadzić do algorytmu obliczania MKZ. Wystarczy w tym celu dla obliczonych par zgodnych B_i, B_j (albo sprzecznych) obliczyć rodzinę maksymalnych zbiorów zgodnych bloków *B*, a następnie stosując algorytm wyznaczania minimalnego pokrycia skojarzyć zbiory bloków minimalnego pokrycia z blokami podziału Π_G.

PRZYKŁAD 3.1

Sposób postępowania wyjaśnimy na przykładzie funkcji podanej w tabl. 3.5.

Tablica 1.5

	x_1	x_2	x_3	x_4	x_5	f
1	0	0	0	0	0	0
2	0	1	0	1	1	0
3	0	0	1	0	1	0
4	0	1	1	1	1	0
5	0	0	1	1	0	0
6	0	1	0	0	1	1
7	0	0	1	0	0	1
8	0	1	1	1	0	1
9	1	0	1	0	1	1
10	1	1	0	0	1	1
11	1	0	0	0	1	1

Zakładając dekompozycję dla zbiorów: $U = \{x_1,x_2\}$, $V = \{x_3,x_4,x_5\}$ podziały $P_U = P_3 \bullet P_4$ i $P_V = P_1 \bullet P_2 \bullet P_5$ można wyznaczyć bezpośrednio z podziałów:

$$P_1 = \{\overline{1,2,3,4,5,6,7,8}; \overline{9,10,11}\}$$

$$P_2 = \{\overline{1,3,5,7,9,11}; \overline{2,4,6,8,10}\}$$

$$P_3 = \{\overline{1,2,6,10,11}; \overline{3,4,5,7,8,9}\}$$

$$P_4 = \{\overline{1,3,6,7,910,11}; \overline{2,4,5,8}\}$$

$$P_5 = \{\overline{1,5,7,8}; \overline{2,3,4,6,9,10,11}\}$$

$$P_f = \{\overline{1,2,3,4,5}; \overline{6,7,8,9,10,11}\}$$

Na tej podstawie obliczamy:

$$P_U = P_1 \bullet P_2 = (\overline{1,3,5,7}; \overline{2,4,6,8}; \overline{9,11}; \overline{10})$$

$$P_V = P_3 \bullet P_4 \bullet P_5 = (\overline{1}; \overline{2}; \overline{3,9}; \overline{4}; \overline{5,8}; \overline{6,10,11}; \overline{7})$$

Numerujemy bloki P_V:

$P_V = (B_1, B_2, B_3, B_4, B_5, B_6, B_7)$

$$\gamma_{12} = (\overline{1,2}; \overline{3,9}; \overline{4}; \overline{5,8}; \overline{6,10,11}; \overline{7})$$

$$\gamma_{57} = (\overline{1}; \overline{2}; \overline{3,9}; \overline{4}; \overline{5,7,8}; \overline{6,10,11})$$

$P_U \bullet \gamma_{12} \leq P_f$,

$P_U \bullet \gamma_{57} \not\leq P_f$,

czyli para (B_1, B_2) jest zgodna, natomiast para (B_5, B_7) jest sprzeczna.

Ale do wyznaczenia zgodnych (lub sprzecznych) par (B_i, B_j) niekoniecznie musimy się posługiwać skomplikowaną nierównością $P_U \bullet \gamma_{ij} \leq P_F$. Wystarczy w tym celu obliczyć iloczyn zbioru $B_i \cup B_j$ z blokami podziału P_U i sprawdzić, czy każdy „niepusty iloczyn" jest zawarty w jakimś bloku P_F.

Do obliczenia Π_G zastosujemy algorytm wyznaczania klas zgodnych wg par niezgodności: (B_1, B_7), (B_2, B_5), (B_2, B_6), (B_3, B_7), (B_4, B_5), (B_4, B_6), i (B_5, B_7). Utworzona według par niezgodnych formuła typu iloczyn sum i jej kolejne przekształcenia są następujące:

$(B_1 + B_7)\,(B_2 + B_5)\,(B_2 + B_6)\,(B_3 + B_7)\,(B_4 + B_5)\,(B_4 + B_6)\,(B_5 + B_7) =$

$= (B_7 + B_1B_3B_5)\,(B_2B_4 + B_5B_6) = B_2B_4B_7 + B_5B_6B_7 + B_1B_2B_3B_4\,B_5 + B_1B_3B_5B_6$

Klasy zgodne uzyskamy odejmując od zbioru $\{B_1,...,B_7\}$, zbiory tych B_i, które występują w jednym składniku obliczonego wyżej wyrażenia typu „suma iloczynów". Stąd:

$B_1B_3B_5B_6 + B_1B_2B_3B_4 + B_6B_7 + B_2B_4B_7$.

$\{B_1,..., B_7\} - \{B_2, B_4, B_7\} = \{B_1, B_3, B_5, B_6\}$

$\{B_1,...,B_7\} - \{B_5, B_6, B_7\} = \{B_1, B_2, B_3, B_4\}$

$\{B_1,...,B_7\} - \{B_1, B_2, B_3, B_4, B_5\} = \{B_6, B_7\}$

$\{B_1,...,B_7\} - \{B_1, B_3, B_5, B_6\} = \{B_2, B_4, B_7\}$

Z powyższych klas bloków zgodnych można wyselekcjonować dwie: $\{B_1, B_3, B_5, B_6\}$ oraz $\{B_2, B_4, B_7\}$, pokrywające zbiór $\{B_1,..., B_7\}$, czyli. $\Pi_g = \{\overline{B_1, B_3, B_5, B_6}; \overline{B_2, B_4, B_7}\}$.

W tym przypadku rozwiązanie jest trywialne. Wracając do pierwotnych kostek tworzących poszczególne bloki B_i otrzymujemy:

$$\Pi_g = \{\overline{1,3,5,6,8,10,11}; \overline{2,4,7}\}.$$

w którym od razu określamy kodowanie bloków.

Obliczanie podziału Π_G można uprościć rezygnując z możliwości uzyskania rozwiązania optymalnego tj. podziału Π_G z najmniejszą liczbą bloków, co w realizacji odpowiada najmniejszej liczbie wyjść z komponentu G. Metoda prowadząca do takiego rozwiązania polega na zastosowaniu podziału ilorazowego $P_U \,|P_U \cdot P_F$. Podział ilorazowy podaje informację, które bloki podziału P_V należy umieścić w różnych blokach tworzonego podziału Π_G. Dwa bloki B_i, B_j, podziału P_V należy umieścić w różnych blokach Π_G wtedy, gdy ich przecięcia ze zbiorem B_u podziału P_U należą do różnych elementów C_s, C_t, odpowiedniego bloku podziału $P_U \,|P_U \cdot P_F$. Taką metodę konstruowania podziału Π_G nazywać będziemy metodą rozdziału elementów podziału ilorazowego $P_U \,|P_U \cdot P_F$. Na przykład dla podziałów P_U, P_f, P_V z przykładu 3.1:

$$P_U = P_1 \bullet P_2 = \left(\overline{1,3,5,7}; \overline{2,4,6,8}; \overline{9,11}; \overline{10}\right)$$

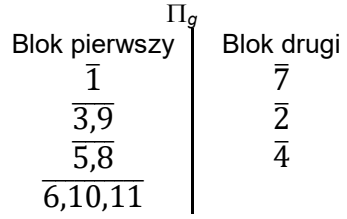
$$P_f = \left\{\overline{1,2,3,4,5}; \overline{6,7,8,9,10,11}\right\}$$

$$P_V = P_3 \bullet P_4 \bullet P_5 = (B_1, B_2, B_3, B_4, B_5, B_6, B_7) = \left(\overline{1}; \overline{2}; \overline{3,9}; \overline{4}; \overline{5,8}; \overline{6,10,11}; \overline{7}\right)$$

podział ilorazowy jest następujący:$P_U|P_U \bullet P_F = \left(\overline{(1,3,5)(7)}; \overline{(2,4)(6,8)}; \overline{(9,11)}; \overline{(10)}\right)$

Rozważmy bloki B_1 i B_7 podziału P_V. Ich przecięcia z blokiem $B_U = \overline{1,3,5,7}$, czyli 1 i 7 należą odpowiednio do C_s = (1,3,5) oraz C_t = (7), czyli dwóch różnych elementów P_U |P_U·P_f. Zatem konstruując podział Π_G należy B_1 i B_7 umieścić w różnych blokach Π_G.

Metodę rozdziału przy tworzeniu podziału Π_G dla funkcji z przykładu 3.1 wyjaśnimy posługując się interpretacją graficzną przedstawioną na rys. 3.4. Z podziału ilorazowego wnioskujemy, że Π_G może być dwublokowy i musi rozdzielać elementy (1,3,5) od (7), (2,4) od (6,8) itd. Dlatego (zaczynając od pierwszego bloku podziału ilorazowego) bloki P_V zawierające elementy 1; 3,9 oraz 5,8 umieszczamy w pierwszym bloku Π_G – na rys 3.3 z lewej strony pionowej kreski, natomiast blok zawierający 7 musimy umieścić w drugim – po prawej stronie pionowej kreski. W drugim bloku podziału ilorazowego należy rozdzielić (2,4) od (6,8). Ale element 8 został już wpisany do pierwszego bloku, zatem również do pierwszego bloku należy wpisać blok P_V = (6,10,11), natomiast do drugiego należy wpisać (2) i (4). W rezultacie uzyskujemy



Rys. 1.4. Konstrukcja podziału Π_g

$$\Pi_g = \left\{\overline{1,3,5,6,8,10,11}; \overline{2,4,7}\right\},$$

identyczny jak poprzednio.

PRZYKŁAD 3.2

Skuteczność wprowadzonego aparatu pokażemy na przykładzie funkcji TL27 (podroz. 2.6 – tab. 2.28).

Dla tej funkcji obliczyliśmy jeden z minimalnych zbiorów argumentów: $X = \{x_3, x_5, x_6, x_7, x_8, x_9, x_{10}\}$.

Korzystając z tej informacji załóżmy poszukiwanie dekompozycji dla zbiorów $U = \{x_7, x_8, x_9\}$ oraz $V = \{x_3, x_5, x_6, x_{10}\}$. Bezpośrednio z tablicy 2.28 zapisujemy podziały P_i utworzone na zbiorze K ponumerowanych wektorów $K = \{1, 2, \ldots, 25\}$:

$$P_1 = \{\overline{1,3,6,8,9,10,11,13,14,15,19,21,24,25}; \overline{2,4,5,7,12,16,17,18,20,22,23}\}$$

$$P_2 = \{\overline{1,2,4,11,15,19,21,23}; \overline{3,5,6,7,8,9,10,12,13,14,16,17,18,20,22,24,25}\}$$

$$P_3 = \{\overline{3,5,6,8,9,11,12,13,14,20,25}; \overline{1,2,4,7,10,15,16,17,18,19,21,22,23,24}\}$$

$$P_4 = \{\overline{1,2,3,5,6,7,8,11,13,14,15,19,21,23,24,25}; \overline{4,9,10,12,16,17,18,20,22}\}$$

$$P_5 = \{\overline{2,3,5,6,9,11,14,15,16,19,21,24}; \overline{1,4,7,8,10,12,13,17,18,20,22,23,25}\}$$

$$P_6 = \{\overline{4,7,9,13,15,17,19,20,21,22,24}; \overline{1,2,3,5,6,8,10,11,12,14,16,18,23,25}\}$$

$$P_7 = \{\overline{2,5,6,7,8,9,11,12,13,15,16,19,20,22,24}; \overline{1,3,4,10,14,17,18,21,23,25}\}$$

$$P_8 = \{\overline{1,4,5,8,9,10,12,13,16,17,19,22,25}; \overline{2,3,6,7,11,14,15,18,20,21,23,24}\}$$

$$P_9 = \{\overline{2,8,11,13,16,17,19,23,25}; \overline{1,3,4,5,6,7,9,10,12,14,15,18,20,21,22,24}\}$$

$$P_{10} = \{\overline{1,2,3,6,7,8,9,11,13,15,19,22,24,25}; \overline{4,5,10,12,14,16,17,18,20,21,23}\}$$

$$P_f = \{\overline{1,2,\ldots,9}; \overline{10,\ldots,25}\}$$

Kolejne obliczenia uzyskane bezpośrednio z podziałów P_i są następujące:

$$P_U = P_7 \bullet P_8 \bullet P_9 = (\overline{1,4,10}; \overline{2,11}; \overline{3,14,18,21}; \overline{5,9,12,22}; \overline{6,7,15,20,24}; \overline{8,13,16,19}; \overline{17,25}; \overline{23})$$

$$P_V = P_3 \bullet P_5 \bullet P_6 \bullet P_{10} = (\overline{1}; \overline{2}; \overline{3,6,11}; \overline{4,17}; \overline{5,14}; \overline{7,22}; \overline{8,25}; \overline{9}; \overline{10,18,23}; \overline{12}; \overline{13}; \overline{15,19,24}; \overline{16}; \overline{20}; \overline{21})$$

Dla wygody dalszych obliczeń podział P_U zapiszemy w postaci podziału ilorazowego:

$$P_U|P_f =$$

$$= \overline{(1,4)(10)}; \overline{(2)(11)}; \overline{(3)(14,18,21)}; \overline{(5,9)(12,22)}; \overline{(6,7)(15,20,24)}; \overline{(8)(13,16,19)}; \overline{(17)(25)}; \overline{(23)}$$

Najważniejszą czynnością jest obliczenie podziału Π_g . Aby to zrealizować wystarczy zauważyć, że:

a) podział Π_g musi rozdzielać elementy 1 i 4 od elementu 10; 2 od 11; 3 od 14 i 18 i 21 itp. Wynika to z podziału ilorazowego,

b) podział Π_g musi być zbudowany z bloków podziału P_V.

PRZYKŁAD 3.2 c.d.

Zatem, jeśli zaliczymy bloki $\overline{1}$ oraz $\overline{4,17}$ do pierwszego bloku Π_g, to blok $\overline{10,18,23}$ musimy zaliczyć do drugiego bloku Π_g (ponieważ zawiera element 10) itp. Reszta obliczeń jest pokazana na rys. 3.5.

Π_g

Blok pierwszy	Blok drugi
$\overline{1}$ $\overline{4,17}$	$\overline{10,18,23}$
$\overline{3,6,11}$	$\overline{2}$ $\overline{5,14}$ $\overline{21}$
$\overline{12}$ $\overline{7,22}$	$\overline{9}$
$\overline{8,25}$	$\overline{15,19,24}$ $\overline{20}$
	$\overline{13}$ $\overline{16}$

Rys. 1.5. Konstrukcja podziału Π_g

Na tej podstawie stwierdzamy, że:

$$\Pi_g = \{\overline{1,3,4,6,7,8,11,12,17,22,25}; \overline{2,5,9,10,13,14,15,16,18,19,20,21,23,24}\}.$$

Zastosowanie omówionych wyżej algorytmów dekompozycji do syntezy logicznej modułów wielowyjściowych (np. matryce PLA, moduły PLD, pamięci ROM) wymaga przede wszystkim uogólnienia podstawowego twierdzenia o dekompozycji na przypadek układów wielowyjściowych. Łączna (jednoczesna) dekompozycja wszystkich funkcji wchodzących w skład układu realizowanego z zastosowaniem modułu wielo-wyjściowego pozwala na wyodrębnienie wielowyjściowych podukładów *H*, *G*, których tablice prawdy stanowią bądź bezpośrednią informację, określającą na przykład zawartość pamięci ROM lub komórki typu LUT, bądź też są punktem wyjścia do dalszych etapów syntezy (np. minimalizacja układów wielowyjściowych) w przypadku matryc PLA.

Łatwo sprawdzić, że $P_V \cdot \Pi_g \leq P_f$. Zatem dekompozycja istnieje. Dalej, korzystając z podziałów P_V i Π_g obliczamy tablicę funkcji *g* (tab. 3.7). Obliczenia interpretujemy następująco. Dla przykładu: blok $\overline{1}$ jest zawarty w drugim bloku podziałów P_1, P_5, P_6, natomiast w pierwszym bloku podziału P_{10} oraz w pierwszym bloku podziału Π_g. Dlatego odpowiadający mu wektor jest 1110, ma wartość funkcji *g* = 0 (patrz tab. 3.6).

Podobnie postępujemy dla funkcji *h* (tab. 3.7), ale tu korzystamy z bloków iloczynu podziałów $P_U \cdot \Pi_g$ oraz ich przynależności do podziałów P_7, P_8, P_9, Π_g oraz P_f.

Tablica 1.6

	x_3	x_5	x_6	x_{10}	g
$\overline{1}$	1	1	1	0	0
$\overline{2}$	1	0	1	0	1
$\overline{3,6,11}$	0	0	1	0	0
$\overline{4,17}$	1	1	0	1	0
$\overline{5,14}$	0	0	1	1	1
$\overline{7,22}$	1	1	0	0	0
$\overline{8,25}$	0	0	1	0	0
$\overline{9}$	0	0	0	0	1
$\overline{10,18,23}$	1	1	1	1	1
$\overline{12}$	0	1	1	1	0
$\overline{13}$	0	1	0	0	1
$\overline{15,19,24}$	1	0	0	0	1
$\overline{16}$	1	0	1	1	1
$\overline{20}$	0	1	0	1	1
$\overline{21}$	1	0	0	1	1

Tablica 1.7

	x_7	x_8	x_9	g	f
$\overline{14}$	1	0	1	0	0
$\overline{10}$	1	0	1	1	1
$\overline{2}$	0	1	0	1	0
$\overline{11}$	0	1	0	0	1
$\overline{3}$	1	1	1	0	0
$\overline{14,18,21}$	1	1	1	1	1
$\overline{5,9}$	0	0	1	1	0
$\overline{12,22}$	0	0	1	0	1
$\overline{6,7}$	0	1	1	0	0
$\overline{15,20,24}$	0	1	1	1	1
$\overline{8}$	0	0	0	0	0
$\overline{13,16,19}$	0	0	0	1	1
$\overline{17,25}$	1	0	0	0	1
$\overline{23}$	1	1	0	1	1

1.2.2 Dekompozycja zespołów funkcji boolowskich

Ze względu na różne zastosowanie przedstawionego wyżej schematu, blok G będziemy nazywać albo układem składowych dekompozycji g_j, albo układem modyfikacji adresu. Pierwsza nazwa jest analogią do dekompozycji pojedynczej funkcji boolowskiej. Druga z kolei wiąże się z zastosowaniem dekompozycji w projektowaniu układów adresowania pamięci ROM, dla których to układów ograniczenia w liczbie wejść adresowych zmuszają do procesu modyfikacji adresu o n bitach na adres t-bitowy.

Oznaczmy jak poprzednio wektory z $\mathbf{B}^n$ liczbami naturalnymi $K = \{1,\ldots, |\boldsymbol{B}^n|\}$, a przez P_F podział wyjściowy funkcji F skonstruowany w następujący sposób:

$$(\mathbf{s}, \mathbf{t}) \in B_{P_F} \Leftrightarrow F(s) = F(t)$$

Inaczej mówiąc, wektory s i t należą do jednego bloku B podziału P_F tylko wtedy, gdy odwzorowanie F przyporządkowuje tym wektorom taki sam wektor z $\{0,1\}^m$. Na przykład, dla odwzorowania F określonego jak w tabl. 3.8 podział P_F (zapisany w postaci podziału na zbiorze $K = \{1,2,\ldots,15\}$) będzie miał postać:

$$P\_F = \backslash left(\backslash bar\{1,9,14\};\backslash bar\{5,7,8,13\};\backslash bar\{2,6,12\};$$
$$\backslash bar\{4,11\};\backslash bar\{3,10,15\}\backslash right)$$

Tablica 1.8

	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	0
3	0	0	0	1	0	1	0	0
4	0	1	1	0	0	0	1	1
5	0	1	1	0	1	0	0	1
6	0	1	1	1	0	0	1	0
7	0	1	0	0	0	0	0	1
8	1	1	0	0	0	0	0	1
9	1	1	0	1	0	0	0	0
10	1	1	1	0	0	1	0	0
11	1	1	1	1	1	0	1	1
12	1	1	1	1	0	0	1	0
13	1	0	0	0	1	0	0	1
14	1	0	0	1	1	0	0	0
15	1	0	0	1	0	1	0	0

Warunek istnienia dekompozycji o schemacie blokowym jak na rys. 3.6, gdzie U, V są rozłącznymi podzbiorami zbioru X funkcji $F(X)$ oraz $W \subseteq U$ formułuje następujące twierdzenie.

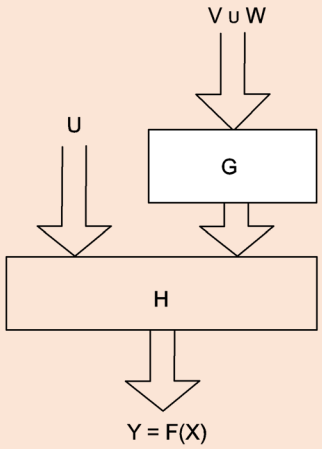
TWIERDZENIE 3.3

Funkcja $F : B^n \rightarrow \{0,1,-\}^m$ ma dekompozycję (rys. 3.6):

$$F = H(U, G(V, W))$$

wtedy i tylko wtedy, gdy istnieje podział $\Pi_G \geq P_{V \cup W}$ taki, że:

$$P_U \cdot \Pi_G \leq P_F \qquad (3\text{-}1)$$



1.6. Dekompozycja wg twierdzenia 3.3

Jak widać zastosowanie rachunku podziałów okazało się wyjątkowo wygodne, gdyż odpowiednie twierdzenie o dekompozycji układów wielowyjściowych traktuje zespół funkcji jako pojedynczy obiekt, podlegający zasadom dekompozycji identycznie jak pojedyncza funkcja boolowska.

PRZYKŁAD 3.3

Dla układu funkcji F z tabl. 3.9 przyjmijmy $U = \{x_1, x_3, x_4\}$ oraz z $V = \{x_2, x_5\}$ oraz $W = \phi$.

Tablica 1.9

	x_1	x_3	x_4	g	y_1	y_2	y_3
1	0	0	0	0	0	0	0
2	0	0	1	1	0	1	0
3	0	0	1	0	1	0	0
4	0	1	0	1	0	1	1
5	0	1	0	0	0	0	1
6	0	1	1	1	0	1	0
7	0	0	0	1	0	0	1
8,13	1	0	0	1	0	0	1
9,14	1	0	1	1	0	0	0
10	1	1	0	1	1	0	0
11	1	1	1	0	0	1	1
12	1	1	1	1	0	1	0
15	1	0	1	0	1	0	0

PRZYKŁAD 3.3. c.d.

Podział $P_U = P_1 \cdot P_3 \cdot P_4$ ($P_i = P(x_i)$), $P_V = P_2 \cdot P_5$. Zapisując P_U w postaci podziału ilorazowego mamy, że:

$$P_U|P_F = \overline{(1)(7)}; \overline{(8)(13)}; \overline{(2)(3)}; \overline{(9,14)(15)}; \overline{(4)(5)}; \overline{(10)}; \overline{(6)}; \overline{(11)(12)}$$
$$P_V = \{\overline{1,3,15}; \overline{2,13,14}; \overline{4,6,7,8,9,1012}; \overline{5,11}\}$$

Blokom $H_1,\ldots, H_4$ podziału P_V odpowiadają wektory $D(H_1) = 00$, $D(H_2) = 01$, $D(H_3) = 10$, $D(H_4) = 11$. W celu wyznaczenia dwublokowego podziału $\Pi_G \geq P_V$ spełniającego warunek dekompozycji (3-1) zauważmy, że jeśli blok H_1 przeznaczymy do pierwszego bloku podziału Π_G, to blok H_3 należy przeznaczyć do drugiego bloku tego podziału. Stwierdzenie to jest wynikiem faktu, że H_1 zawiera wektor 1, H_3 zawiera wektor 7, a ponadto wektory te należą do różnych bloków podziału wyjściowego P_F. Fakt ten jest zresztą zaznaczony w podziale ilorazowym $P_U \,|\, P_F$. Mamy więc $\Pi_G^0 = \{1,3,15\}$, więc $\Pi_G^1 = \{4,6,7,8,9,10,12\}$ .Następnie sprawdzamy kolejną parę wektorów z $P_U \,|\, P_F$, czyli 2 i 3. Skoro 3 jest w Π_G^0 , to $\Pi_G^1 = \Pi_G^1 \cup \{2,3,14\}$. Kolejna para (9,14) i (15) należy już do różnych bloków Π_G, a więc sprawdzamy 4 i 5. Stąd:

$$\Pi_G = \left(\overline{1,3,5,11,15}; \overline{2,4,6,7,8,9,10,12,13,14}\right)$$

Łatwo stwierdzić, że funkcja g przyporządkowuje wektorowi $(x_2, x_5) = (0,0)$ wartość 0, i analogicznie $g(0,1) = 1$, $g(1,0) = 1$, $g(1,1) = 0$. Ostatecznie $g = x_2 \oplus x_5$. W rezultacie, dla układu y_1, y_2, y_3 funkcji z tabl. 3.9 danego odwzorowaniem $F: (y_1, y_2, y_3) = F(x_1,\ldots, x_5)$ uzyskaliśmy dekompozycję: $F = H(x_1, x_3, x_4, g(x_2 \oplus x_5))$.

Tablicę prawdy funkcji H można wyznaczyć na podstawie przynależności bloków iloczynu $P_1 \cdot P_3 \cdot P_4 \cdot \Pi_G$ do bloków podziałów P_1, P_3, P_4, Π_G oraz P_F. Odpowiednie obliczenia podane są w tabl. 3.10.

Tablica 1.10

(2)	(9,14)	(3,15)
$\overline{2};$	$\overline{8,9,10,12}$ $\overline{13,14}$	$\overline{1,3}$ $\overline{15}$
$\overline{4,6,7};$		$\overline{5}$
		$\overline{11}$

Z kolei dla $U = \{x_3, x_4\}$ oraz $V = \{x_1, x_2, x_5\}$, podział $P_U = P_3 \cdot P_4$, $P_V = P_1 \cdot P_2 \cdot P_5$, a więc:

$$P_U = \left(\overline{1,7,8,13}; \overline{2,3,9,14,15}; \overline{4,5,10}; \overline{5,6,11,12}\right)$$
$$P_F = \left(\overline{1,9,14}; \overline{5,7,8,13}; \overline{2,6,12}; \overline{4,11}; \overline{3,10,15}\right)$$
$$P_U|P_F = \overline{(1)(7,8,13)}; \overline{(2)(9,14)(3,15)}; \overline{(4)(5)(10)}; \overline{(11)(6,12)}$$
$$P_V = \left(\overline{1,3}; \overline{2}; \overline{4,6,7}; \overline{5}; \overline{8,9,10,12}; \overline{11}; \overline{13,14}; \overline{15}\right)$$

gdzie bloki P_V są oznaczone kolejno $B_1, B_2,\ldots, B_8$.

PRZYKŁAD 3.3. c.d.

Z podziału ilorazowego $P_U \mid P_F$ wnioskujemy, że odpowiedni Π_G powinien mieć co najmniej 3 bloki. Wynika to z faktu, że elementy (2), (9,14) i (3,15) muszą należeć do trzech różnych bloków Π_G (to samo dotyczy (4), (5) i (10)). Ponadto pamiętamy, że $\Pi_G \geq P_V$. Zatem wprowadzenie bloków z P_V do tworzonego Π_G powinno przebiegać według schematu pokazanego w tablicy 3.10. Stąd:

$$\Pi_G = \left(\overline{2,4,6,7}; \overline{8,9,10,12,13,14}; \overline{1,3,5,11,15}\right)$$

Przyjmując, że kodowanie bloków Π_G jest: pierwszy blok – 01, drugi blok – 10, trzeci blok – 00 i uwzględniając, iż kolejnym blokom z podziału P_V odpowiadają wektory (zmienne x_1, x_2, x_5) odpowiednio: $B_1 = 000$, $B_2 = 001$, $B_3 = 010$, $B_4 = 011$, $B_5 = 110$, $B_6 = 111$, $B_7 = 101$ oraz $B_8 = 100$, wyznaczamy tablicę prawdy funkcji G (tab. 3.11).

Tablica 1.11

	x_1	x_2	x_5	g_1	g_2
$\overline{1,3}$	0	0	0	0	0
$\overline{2}$	0	0	1	0	1
$\overline{4,6,7}$	0	1	0	0	1
$\overline{5}$	0	1	1	0	0
$\overline{8,9,10,12}$	1	1	0	1	0
$\overline{11}$	1	1	1	0	0
$\overline{13,14}$	1	0	1	1	0
$\overline{15}$	1	0	0	0	0

Podobnie, po obliczeniu iloczynu: $P_U \cdot \Pi_G = \left(\overline{1}; \overline{7}; \overline{8,13}; \overline{3,15}; \overline{2}; \overline{9,14}; \overline{4}; \overline{5}; \overline{10}; \overline{6}; \overline{11}; \overline{12}\right)$ wyznaczamy tablicę prawdy funkcji H (tab. 3.12).

Tablica 1.12

	x_3	x_4	g_1	g_2	y_1	y_2	y_3
$\overline{1}$	0	0	0	0	0	0	0
$\overline{7}$	0	0	0	1	0	0	1
$\overline{8,13}$	0	0	1	0	0	0	1
$\overline{3,15}$	0	1	0	0	1	0	0
$\overline{2}$	0	1	0	1	0	1	0
$\overline{9,14}$	0	1	1	0	0	0	0
$\overline{4}$	1	0	0	1	0	1	1
$\overline{5}$	1	0	0	0	0	0	1
$\overline{10}$	1	0	1	0	1	0	0
$\overline{6}$	1	1	0	1	0	1	0
$\overline{11}$	1	1	0	0	0	1	1
$\overline{12}$	1	1	1	0	0	1	0

1.2.3 Pojęcie *r*-przydatności i dekompozycja nierozłączna

Istotną zaletą rachunku podziałów wprowadzonego w poprzednim rozdziale jest prosta metoda selekcji zbioru argumentów należących do zbioru *U*.

W selekcji argumentów x_i ze zbioru *U* spełniających warunek (3-1) z twierdzenia 3.3 pomocne jest pojęcie *r* - p r z y d a t n o ś c i zbioru podziałów względem podziału *P* [3.1], [3.15].

Oznaczmy przez $\gamma(\tau|\delta)$ liczbę elementów w największym bloku ilorazu podziałów τ i δ. Niech $\Gamma(\tau|\delta) = \log_2 \gamma(\tau|\delta)$.

Zbiór $\{P_1,..., P_k\}$ jest *r*-przydatny względem *P*, gdzie:

$$r = k + \Gamma(P_1...P_k \,|P \cdot P_1...P_k) \qquad (3\text{-}2)$$

Dla zmiennej x_1 z tab. 3.9 obliczymy *r* dla $P_1 = \left(\overline{1,2,3,4,5,6,7}; \overline{8,9,10,11,12,13,14,15}\right)$ oraz $P = \left(\overline{1,9,14}; \overline{5,7,8,13}; \overline{2,6,12}; \overline{4,11}; \overline{3,10,15}\right)$.

Mamy tu:

$$P \cdot P_1 = \left(\overline{1}; \overline{9,14}; \overline{5,7}; \overline{8,13}; \overline{2,6}; \overline{12}; \overline{4}; \overline{11}; \overline{3}; \overline{10,15}\right)$$

$$P|P \cdot P_1 = \left\{\overline{(1)(2,6)(3)(4)(5,7)}; \overline{(8,13)(9,14)(10,15)(11)(12)}\right\}$$

$\gamma(P_1|P \cdot P_1) = 5$, $\Gamma(P_1|P \cdot P_1) = 3$, czyli $r = 4$

Można więc *r*-przydatności dwublokowych podziałów P_i względem podziału *P* obliczać bezpośrednio z P_i oraz *P*, grupując po prostu elementy w bloku podziału P_i w podzbiory o maksymalnej liczności należące do ustalonego bloku podziału *P*. Dlatego też iloraz $P_i|P{\cdot}P_i$ oznaczać będziemy często po prostu przez P_i, zapisując elementy tego ilorazu w nawiasach.

Jeśli $\{P_1,..., P_k\}$ jest r-przydatny ($k < r$) względem P, to istnieje zbiór podziałów $\{P_{k+1},..., P_r\}$, dla których:

$$P_1...P_k \cdot P_{k+1}...P_r \leq P$$

natomiast nie istnieje $\{P_{k+1}',..., P_{r-1}'\}$ taki, że:

$$P_1...P_k \cdot P_{k+1}',..., P_{r-1}' \leq P$$

Inaczej mówiąc, warunkiem koniecznym i wystarczającym na to, aby:

$$P_1...P_k \cdot P_{k+1}...P_m \leq P$$

jest *m*-przydatność zbioru $\{P_1,..., P_k\}$ względem *P*.

Można również wykazać, że jeśli $P = \{P_1,..., P_k\}$ jest m-przydatny względem P, to każdy podzbiór z P jest m'-przydatny, gdzie $m' \leq m$. Tym samym warunkiem koniecznym na to, aby $P = \{P_1,..., P_k\}$ był m-przydatny względem P jest r-przydatność podzbiorów z P taka, że dla każdego P' zawartego w P, $r(P') \leq m$. To proste stwierdzenie pozwala w łatwy sposób grupować podziały 2-blokowe w zbiory o ustalonej przydatności.

Na przykład, jeśli w zbiorze podziałów $\{P_1,..., P_5\}$ następujące podzbiory są m-przydatne: $\{P_1, P_3\}$, $\{P_1, P_4\}$, $\{P_3, P_4\}$, $\{P_4, P_5\}$, to jedynym m-przydatnym zbiorem o liczności 3 może być tylko zbiór $\{P_1, P_3, P_4\}$.

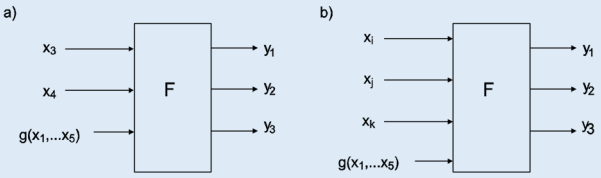
Zastosowanie r-przydatności w syntezie funkcji g, to jest składowych dekompozycji wyjaśnia następujący przykład.

PRZYKŁAD 3.4

Obliczymy r-przydatność podziałów $P(x_i)$ dla układu F funkcji z tab. 3.9. Mamy tu:

$$P(x_1) = \{\overline{(1)(2,6)(3)(4)(5,7)}; \overline{(8,13)(9,14)(10,15)(11)(12)}\}$$

czyli $P(x_1)$ jest 4-przydatny względem P_F. Dokonując analogicznych obliczeń dla pozostałych podziałów, stwierdzamy, że $P(x_3)$, $P(x_4)$ są 3-przydatne, natomiast $P(x_2)$, $P(x_5)$, 4-przydatne.



Rys. 1.7. Ilustracja obliczeń z przykładu 3.4: a) przy założeniu, że zbiór {P(x3), P(x4)} jest 3-przydatny; b) przy założeniu, że {P(xi), P(xj), P(xk)} jest 4-przydatny

Jeśli więc istnieje dekompozycja układu F taka, jaką pokazano na rys. 3.7a, to byłoby to możliwe tylko pod warunkiem, że zbiór $\{P(x_3), P(x_4)\}$ jest 3-przydatny. Ale

$$P(x_3) \cdot P(x_4) = \{\overline{(1)(7,8,13)}; \overline{(9,14)(2)(3,15)}; \overline{(4)(5)(10)}; \overline{(11)(6,12)};\}$$

czyli $\{P(x_3), P(x_4)\}$ jest 4-przydatny. Sprawdźmy więc, czy istnieje dekompozycja o schemacie blokowym jak na rys. 3.7b.

W tym celu obliczymy r dla każdego zbioru $\{P(x_i), P(x_j)\}$. W wyniku uzyskujemy, że 4-przydatnymi parami są tylko $\{P_1, P_3\}$[1)], $\{P_1, P_4\}$, $\{P_2, P_3\}$, $\{P_2, P_4\}$, $\{P_3, P_4\}$, $\{P_3, P_5\}$ oraz $\{P_4, P_5\}$. W związku z tym 4-przydatnymi trójkami mogą być tylko: a) $\{P_1, P_3, P_4\}$; b) $\{P_2, P_3, P_4\}$; c) $\{P_3, P_4, P_5\}$.

Następnie stwierdzamy, że wyłącznie trójki a) oraz c) są 4-przydatne. Dla trójek tych odpowiednie iloczyny podziałów są:

$$P(x_1) \cdot P(x_3) \cdot P(x_4) = \{\overline{(1)(7)}; \overline{(8,13)}; \overline{(2)(3)}; \overline{(9,14)(15)}; \overline{(4)(5)}; \overline{(10)}; \overline{(6)}; \overline{(11)(12)}\}$$
$$P(x_3) \cdot P(x_4) \cdot P(x_5) = \{\overline{(1)(7,8)}; \overline{(13)}; \overline{(9)(3,15)}; \overline{(14)(2)}; \overline{(4)(10)}; \overline{(5)}; \overline{(6)(12)}; \overline{(11)}\}$$

[1)] Stosujemy tu oznaczenie $P(x_i) = P_i$.

Z przykładu 3.4 wynika, że synteza składowych dekompozycji wymaga utworzenia wszystkich możliwych dwójek, trójek... podziałów o ustalonej *r*-przydatności. Przy dużej liczbie zmiennych wejściowych (dużym *n*) wymaga to obliczenia stosunkowo dużej liczby *r*-przydatności zbiorów $\{P_i, P_j\}$ $(i, j \in \{1,..., n\})$, gdyż liczba wszystkich różnych par $\{P_i, P_j\}$ jest $n!/2(n-2)!$.

Proces ten można uprościć, określając związki między *r*-przydatnością podziałów dwublokowych.

Jeżeli $r(P_a) = n$ i $r(P_b) = n$, to zbiór $\{P_a, P_b\}$ ma przydatność co najwyżej równą n+1, jeżeli natomiast $r(P_a) = n$ i $r(P_b) = n + 1$, to zbiór $\{P_a, P_b\}$ jest n+1-przydatny.

Zastosujemy powyższe związki w celu obliczenia 4-przydatnych par P_i, P_j dla podziałów $P_1,..., P_5$ z przykładu 3.4. Poprzednio obliczyliśmy, że *r*-przydatności tych podziałów wynoszą odpowiednio: 3 dla P_3, P_4 oraz 4 dla P_1, P_2, P_5. Na mocy powyższych związków, 4-przydatnymi zbiorami $\{P_i, P_j\}$ są: $\{P_1, P_3\}$, $\{P_1, P_4\}$, $\{P_2, P_3\}$, $\{P_2, P_4\}$, $\{P_3, P_5\}$, $\{P_4, P_5\}$, natomiast *r*-przydatność pary $\{P_3, P_4\}$ jest r': $3 \le r' \le 4$, a par $\{P_1, P_2\}$, $\{P_1, P_5\}$, $\{P_2, P_5\}$ jest r'': $4 \le r'' \le 5$. Stąd wniosek, że w celu wyselekcjonowania wszystkich 4-przydatnych par wystarczy obliczyć *r*-przydatność wyłącznie dla par $\{P_i, P_j\}$: $(i, j) \in \{(3,4),(1,2), (1,5),(2,5)\}$ – czyli czterech par. Poprzednio (w przykładzie 3.4) obliczenia te należało wykonać dla 10 par.

PRZYKŁAD 3.5

Dla funkcji F z tablicy 3.13 zbiory podziałów $\{P_1, P_2\}$, $\{P_1, P_4\}$, $\{P_1, P_5\}$, $\{P_2, P_3\}$, $\{P_2, P_4\}$, $\{P_3, P_4\}$,$\{P_3, P_5\}$,$\{P_4, P_5\}$ są 4-przydatne. Obliczyć wszystkie najlepsze dekompozycje szeregowe tej funkcji dla $U = \{x_i, x_j, x_k\}$, czyli $F = H(x_i, x_j, x_k, G_0)$. Wykazać, że dla żadnej z nich nie istnieje dekompozycja $F = H(G_1(x_i, x_j, x_k), G_0)$.

Tablica 1.13

	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3
1	0	0	0	1	1	1	0	0
2	0	0	0	1	0	1	0	1
3	0	1	1	0	0	0	1	1
4	0	1	1	0	1	0	1	0
5	1	1	0	0	0	0	0	1
6	1	1	0	1	0	0	0	0
7	1	1	1	0	0	1	0	1
8	1	1	1	1	0	0	1	0
9	1	0	0	0	1	1	1	1
10	1	0	0	1	1	1	0	0
11	1	0	0	1	0	1	0	1

Rozwiązanie

$$P_F = \left(\overline{1,10}; \overline{2,7,11}; \overline{3}; \overline{4,8}; \overline{5}; \overline{6}; \overline{9}\right).$$

Z podanych par $\{P_i, P_j\}$ tworzymy trójki podejrzane o $r = 4$

$A=\{x_1,x_2,x_4\}$, $B=\{x_1,x_4,x_5\}$ $C=\{x_2,x_3,x_4\}$ $D=\{x_3,x_4,x_5\}$

$$P(A)|P_F = P_1 \cdot P_2 \cdot P_4|P_F = \left\{\overline{(1)(2)}; \overline{(3)(4)}; \overline{(5)(7)}; \overline{(6)(8)}; \overline{(9)}; \overline{(10)(11)}\right\} \qquad r = 3+1$$

$$P(B)|P_F = P_1 \cdot P_4 \cdot P_5|P_F = \left\{\overline{(1)}; \overline{(2)}; \overline{(3)}; \overline{(4)}; \overline{(5,7)}; \overline{(6)(8)(11)}; \overline{(9)}; \overline{(10)}\right\} \qquad r = 3+2$$

$$P(C)|P_F = P_2 \cdot P_3 \cdot P_4|P_F = \left\{\overline{(1,10)(2,11)}; \overline{(3)(4)(7)}; \overline{(5)}; \overline{(6)}; \overline{(8)}; \overline{(9)}\right\} \qquad r = 3+2$$

$$P(D)|P_F = P_3 \cdot P_4 \cdot P_5|P_F = \left\{\overline{(1,10)}; \overline{(2,11)(6)}; \overline{(3)(7)}; \overline{(4)}; \overline{(5)}; \overline{(8)}; \overline{(9)}\right\} \qquad r = 3+1$$

Obliczenia dla $U = \{x_1, x_2, x_4\}$ (rys. 3.7)

$$P_V = P(x_3x_5) = \left(\overline{1,9,10}; \overline{2,5,6,11}; \overline{3,7,8}; \overline{4}\right).$$

Z podziału ilorazowego $P(A)\,|\,P_F$ wynika, że warunki dekompozycji będą spełnione, gdy rozdzielimy wektory: 1 od 2, 3 od 4, 5 od 7, 6 od 8 oraz 10 od 11. Uzyskamy to, gdy podział Π_{G_0} będzie 2-blokowy (tab. 3.14):

Tablica 1.14

	x_3x_5	g_0
1,9,10	0 1	0
2,5,6,11	0 0	1
3,7,8	1 0	0
4	1 1	1

PRZYKŁAD 3.5.c.d.

$$P_V = P(x_3x_5) = (\overline{1,9,10}; \overline{2,5,6,11}; \overline{3,7,8}; \overline{4}).$$

1. blok: $\overline{1,9,10}$ i $\overline{3,7,8}$
2. blok: $\overline{2,5,6,11}$ i $\overline{4}$

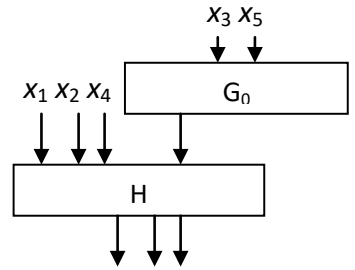
Zatem

$$\Pi_{G_0} = (\overline{1,3,7,8,9,10}; \overline{2,4,5,6,11})$$

$$P_H = P(U)\Pi_{G_0} = (\overline{1,2}; \overline{3,4}; \overline{5,7}; \overline{6,8}; \overline{9}; \overline{10,11}) \cdot (\overline{1,3,7,8,9,10}; \overline{2,4,5,6,11}) =$$

$$(\overline{1}; \overline{2}; \overline{3}; \overline{4}; \overline{5}; \overline{6}; \overline{7}; \overline{8}; \overline{9}; \overline{10}; \overline{11}) .$$

Pierwszy krok dekompozycji przedstawiono w tab. 3.14 i na rys. 3.8.



Rys. 1.8. Pierwszy krok dekompozycji

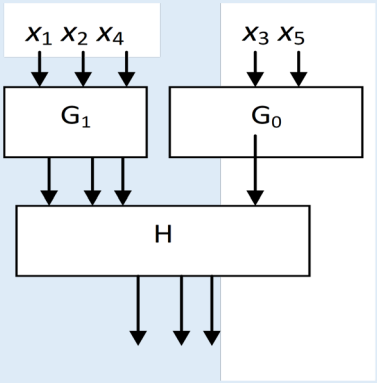
Sprawdzamy czy istnieje dekompozycja $F = H(G_1(x_i, x_j, x_k), G_0)$.

W tym celu liczymy *r*-przydatność Π_{G0}:

$$\Pi_{G_0}|P_F = \{\overline{(1,10)(3)(7)(8)(9)}; \overline{(2,11)(4)(5)(6)}\}$$

czyli $r = 1+3 = 4$, zatem wymagana dekompozycja nie istnieje, gdyż blok G_1 musiałby mieć 3 wyjścia (rys. 3.9).

Obliczenia dla $U = \{x_3, x_4, x_5\}$ są analogiczne. Funkcję H podano w tab. 3.15.



Rys. 1.9. Drugi krok dekompozycji

Tablica 1.15

	$x_1x_2x_4g_0$	$y_1y_2y_3$
1	0010	100
2	0011	101
3	0100	011
4	0101	010
5	1101	001
6	1111	000
7	1100	101
8	1110	010
9	1000	111
10	1010	100
11	1011	101

Dekompozycja nierozłączna

Zauważmy, że *r*-przydatność *k*-elementowego zbioru *U* nie oznacza, że istnieje dekompozycja:

$$F = H((U,G(V))$$

w której funkcja *G* ma $r - k$ wyjść. Sytuacja ta oznacza tylko, że istnieje dekompozycja z funkcją *G* o argumentach $V \cup W$ gdzie $W \subseteq U$.

Oczywiście w najlepszym przypadku *W* jest zbiorem pustym, a w najgorszym $W = U$ i wtedy dekompozycja taka nie ma sensu. Można się jednak spodziewać, że sytuacja najlepsza i najgorsza są najmniej prawdopodobne.

Z powyższego wynika, że w praktyce często stajemy przed problemem wyznaczania dekompozycji nierozłącznej. Podstawowym problemem w obliczaniu dekompozycji nierozłącznej jest wyznaczenie zbioru *W* o możliwie minimalnej liczności. Wybór argumentów do zbioru *W* jest przeprowadzany na podstawie analizy podziału P_V. Otóż, w przypadku, gdy nie istnieje dekompozycja rozłączna, nie istnieje również $\Pi_G \geq P_V$ spełniający nierówność $P_V \cdot \Pi_G \leq P_F$.

Jest zrozumiałe, że powiększenie zbioru *V* o argumenty z $W \subseteq U$ prowadzi w konsekwencji do utworzenia „drobniejszego" $\Pi'_G \geq P'_V$ (gdzie P'_V jest zbiorem indukowanym argumentami $V' = V \cup W$). Drobniejszy Π'_G może zapewnić spełnienie warunku:

$$P \cdot \Pi_G' \leq P_F$$

Sposób postępowania przy wyznaczaniu *W* wyjaśnimy na przykładzie.

Dla funkcji podanej w tabl. 3.16 poszukujemy dekompozycji, w której $U = \{x_1,x_2,x_3\}$, $V = \{x_4,x_5\}$.
Wyznaczamy kolejno:

$$\Pi_{G_0} = \left(\overline{1,3,7,8,9,10}; \overline{2,4,5,6,11}\right)$$

$$P_H = P(U)\Pi_{G_0} =$$

$$= \left(\overline{1,2}; \overline{3,4}; \overline{5,7}; \overline{6,8}; \overline{9}; \overline{10,11}\right)$$

$$\cdot \left(\overline{1,3,7,8,9,10}; \overline{2,4,5,6,11}\right) =$$

$$= \left(\overline{1}; \overline{2}; \overline{3}; \overline{4}; \overline{5}; \overline{6}; \overline{7}; \overline{8}; \overline{9}; \overline{10}; \overline{11}\right)$$

$$P_F = \left(\overline{1,7,10}; \overline{2}; \overline{3,8}; \overline{4}; \overline{5,69}\right)$$

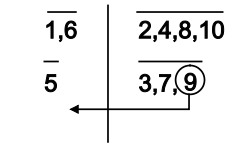
$$P_U = \left(\overline{1,2}; \overline{3,6}; \overline{4,5}; \overline{7}; \overline{8,9}; \overline{10}\right)$$

$$P_V = \left(\overline{1,6}; \overline{2,4,8,10}; \overline{3,7,9}; \overline{5}\right)$$

Tablica 1.16

	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	0
3	0	1	0	1	0	1	0	0
4	0	1	1	1	1	0	1	1
5	0	1	1	0	1	0	0	1
6	0	1	0	0	0	0	0	1
7	1	1	0	1	0	0	0	0
8	1	0	0	1	1	1	0	0
9	1	0	0	1	0	0	0	1
10	1	0	1	1	1	0	0	0

Podział Π_G konstruujemy w sposób pokazany na rys. 3.10. Niestety nie można spełnić warunków rozdziału z podziału ilorazowego: elementy 8 i 9 są w jednym bloku podziału Π_G.



Rys. 1.10. Konstrukcja podziału Π_G

Warunki te byłyby spełnione, gdybyśmy przenieśli element 9 do pierwszego bloku (strzałka na rys. 3.9). Byłoby to jednak możliwe, gdyby elementy 3, 7 były oddzielone od elementu 9. Z tablicy 3.17 wynika, że wektory 3 i 9 różnią się na pozycjach x_1 oraz x_2, natomiast wektory 7 i 9 różnią się na pozycji x_2. Zatem zmienna x_2 wystarczy do oddzielenia 3, 7 od 9. Stąd wniosek, że $W = \{x_2\}$, czyli istnieje dekompozycja nierozłączna. Jej wyznaczenie nie nastręcza trudności, gdyż dla nowego $P_{V'}$ obliczenie odpowiedniego Π_G jest już możliwe:

$P_{V'} = P_V \bullet P_2$

$P_V = \left(\overline{1}; \overline{6}; \overline{2,8,10}; \overline{4}; \overline{3,7}; \overline{9}; \overline{5}\right)$

$\Pi_{G_0} = \left(\overline{1,5,6,9}; \overline{2,3,4,7,8,10}\right)$

Pojęcie *r*-przydatności i dekompozycja nierozłączna ułatwiają znajdowanie najlepszych dekompozycji.

PRZYKŁAD 3.6

Dla funkcji z tablicy 3.17 podziały P_1, P_2, P_5 są 3-przydatne, a P_3, P_4 – 4-przydatne. Należy obliczyć wszystkie dekompozycje szeregowe z najmniejsza liczbą wejść do bloku *H* oraz najmniejszą liczbą wejść i wyjść bloku *G*.

Uwaga: wystarczy obliczyć odpowiednie podziały Π_G spełniające warunek wystarczający dekompozycji.

Tablica 1.17

	x_1	x_2	x_3	x_4	x_5	Y
1	0	0	0	0	0	D
2	0	0	0	0	1	A
3	0	0	0	1	0	C
4	1	0	0	1	0	D
5	1	0	0	1	1	B
6	0	0	0	1	1	B
7	0	0	1	0	0	D
8	1	0	1	0	1	B
9	0	0	1	0	1	A
10	0	1	1	0	0	C
11	1	1	1	0	0	E
12	1	1	1	0	1	A

W rozwiązaniu podać też schematy blokowe obliczonych dekompozycji, w szczególności wejścia i wyjścia bloków *G* i *H*.

Rozwiązanie:

$$P_F = \left(\overline{1,4,7}; \overline{2,9,12}; \overline{3,10}; \overline{4,8}; \overline{5,6,8}; \overline{11}\right).$$

$$P_1 \cdot P_5|P_F = \left\{\overline{(1,7)(3,10)}; \overline{(2,9)(6)}; \overline{(4)(11)}; \overline{(5,8)(12)}\right\} \qquad r = 3$$

$$P_1 \cdot P_2|P_F = \left\{\overline{(1,7)(2,9)(3)(6)}; \overline{(5,8)(4)}; \overline{(10)}; \overline{(11)(12)}\right\} \qquad r = 4$$

$$P_2 \cdot P_5|P_F = \left\{\overline{(1,4,7)(3)}; \overline{(2,9)(5,6,8)}; \overline{(10,11)}; \overline{(12)}\right\} \qquad r = 3$$

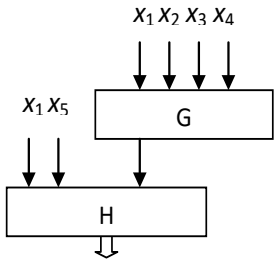
Dla $U = \{x_1, x_5\}$ (rys. 3.11)

$$P_1 \cdot P_5|P_F = \left\{\overline{(1,7)(3,10)}; \overline{(2,9)(6)}; \overline{(4)(11)}; \overline{(5,8)(12)}\right\}$$

$$P_V = P(x_2, x_3, x_4) = \left(\overline{1,2}; \overline{3,4,5,6}; \overline{7,8,9}; \overline{10,11,12}\right).$$

Tworzenie podziału Π_G:

1. blok $\overline{1,2}$ i $\overline{7,8,9}$
2. blok $\overline{3,4,5,6}$ i $\overline{10,11,12}$



Rys. 1.11. Dekompozycja funkcji z przykładu 3.6

Wektory 4, 5 trzeba oddzielić od 3, 6. Stąd dekompozycja nierozłączna z x_1, czyli $V' = x_1\, x_2\, x_3\, x_4$

PRZYKŁAD 3.6 c.d.

$$P_{V'} = \left(\overline{1,2};\overline{3,6};\overline{4,5};\overline{7,9};\overline{8};\overline{10};\overline{11,12}\right)$$

$$\Pi_{G'} = \left(\overline{1,2,4,5,7,8,9};\overline{3,6,10,11,12}\right)$$

$$P_H = P(U)\cdot\Pi_{G'} = \left(\overline{1,3,7,10};;\overline{2,6,9};\overline{4,11};\overline{5,8,12}\right)\cdot\left(\overline{1,2,4,5,7,8,9};\overline{3,6,10,11,12}\right)$$

$$= \left(\overline{1,7};\overline{3,10};\overline{2,9};\overline{6};\overline{4};\overline{11};\overline{5,8};\overline{12}\right)$$

Tablice bloków *G* i *H* przedstawiono odpowiednio w tab. 3.18. i tab. 3.19.

Tablica 1.18

	$x_1x_2x_3x_4$	g
1,2	0 0 0 0	0
3,6	0 0 0 1	1
4,5	1 0 0 1	0
7,9	0 0 1 0	0
8	1 0 1 0	0
10	0 1 1 0	1
11,12	1 1 1 0	1

Tablica 1.19

	x_1x_5g'	Y
1,7	0 0 0	D
2,9	0 1 0	A
3,10	0 0 1	C
4	1 0 0	D
5,8	1 1 0	B
6	0 1 1	B
11	1 0 1	E
12	1 1 1	A

Dla $U = \{x_2, x_5\}$ (rys. 3.12):

$$P_2 \cdot P_5 | P_F = \left\{\overline{(1,4,7)(3)};\overline{(2,9)(5,6,8)};\overline{(10)(11)};\overline{(12)}\right\}$$
$$P_V = P(x_1, x_3, x_4) = \left(\overline{1,2};\overline{3,6};\overline{4,5};\overline{7,9,10};\overline{8,11,12}\right).$$

Podział Π_G można utworzyć następująco:

1. blok $\overline{1,2};\overline{4,5};\overline{7,9,10}$
2. blok $\overline{3,6};\ \overline{8,11,12}$

Aby zlikwidować konflikt wektor 5 należy oddzielić od wektora 4. Stąd dodatkowo x_5, czyli $V' = \{x_1, x_3, x_4, x_5\}$

$$P_{V'} = \left(\overline{1};\overline{2};\overline{3};\overline{4};\overline{5};\overline{6};\overline{7,10};\overline{9};\overline{8,12};\overline{11}\right)$$

$$\Pi_{G'} = \left(\overline{1,2,4,7,9,10};\overline{3,5,6,8,11,12}\right)$$

Tablice bloków *H* i *G'* podano odpowiednio w tab. 3.20 i tab. 3.21.

PRZYKŁAD 3.6 c.d.

Tablica 1.20

	$x_1x_3x_4x_5$	g
1	0 0 0 0	0
2	0 0 0 1	0
3	0 0 1 0	1
4	1 0 1 0	0
5	1 0 1 1	1
6	0 0 1 1	1
7,10	0 1 0 0	0
9	0 1 0 1	0
8,12	1 1 0 1	1
11	1 1 0 0	1

Tablica 1.21

	x_2x_5g'	Y
1,7	0 0 0	D
2,9	0 1 0	A
3,10	0 0 1	C
4	1 0 0	D
5,8	1 1 0	B
6	0 1 1	B
11	1 0 1	E
12	1 1 1	A

Dekompozycja nierozłączna, mimo iż zwiększa liczbę wejść do komponentu G, może w rezultacie prowadzić do uzyskania realizacji z mniejszą liczbą komórek logicznych. Sytuację taką omawiamy w poniższym przykładzie.

PRZYKŁAD 3.7

Dla funkcji *F* z tab. 3.22a istnieje dekompozycja:

$$F = H(x_4, x_5, x_6, G_1(x_1, x_2, x_3))$$

dla której funkcja G_1 jest podana w tablicy 3.22b.

Tablica 1.22

a)

	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2
1	1	0	0	1	0	0	0	0
2	1	0	1	1	1	0	0	1
3	1	1	0	1	1	1	0	1
4	1	0	1	1	0	0	0	0
5	0	0	1	1	1	0	1	0
6	1	1	0	1	0	0	1	0
7	1	0	0	1	1	1	1	0
8	0	0	1	0	1	1	1	1
9	0	0	1	0	1	0	1	1

b)

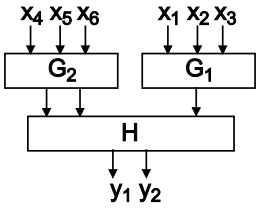
	x_1	x_2	x_3	g_1
1	1	1	0	0
2	0	0	1	0
3	1	0	1	1
4	1	0	0	1

Ponieważ $P_U = P(g_1) = (\overline{3,5,6,8,9}; \overline{1,2,4,7})$. jest 3-przydatny, możemy wyznaczyć:

$\Pi_G = (\overline{3,7}; \overline{2,5}; \overline{1,4,6}; \overline{8,9})$,

który zapewnia realizację funkcji $F = H(G_2(x_4, x_5, x_6), G_1(x_1, x_2, x_3))$

w strukturze przedstawionej na rys. 3.13.



Rys. 1.13.Schemat dekompozycji funkcji

Przy założeniu, że dysponujemy strukturami z komórkami o wymiarach 3/1, do realizacji tej funkcji jest potrzebnych 5 komórek.

Postaramy się znaleźć dekompozycję prowadzącą do oszczędniejszej realizacji. W tym celu obliczymy *r*-przydatność dla podziałów reprezentujących pary zmiennych: (g_1, x_4), (g_1, x_5), (g_1, x_6).

$P_V = (\overline{1,6}; \overline{2,4,8,10}; \overline{3,7,9}; \overline{5})$.

$P_U|P_F = \{\overline{(1)(2)}; \overline{(3)(6)}; \overline{(4)(5)}; \overline{(7)}; \overline{(8)(9)}; \overline{(10)}\}$

$U = \{g_1, x_4\}$

$P_U = (\overline{1,2,4,7}; \overline{3,5,6}; \overline{8,9})$.

$P_U|P_F = \{\overline{(1,4)}; \overline{(2)(7)}; \overline{(3)(5,6)}; \overline{(8,9)}\}$ $r = 4$

$U = \{g_1, x_5\}$

$$P_U = \left(\overline{1,4}; \overline{2,7}; \overline{3,5,8,9}; \overline{6}\right).$$

$$P_U|P_F = \left\{\overline{(1,4)}; \overline{(2)(7)}; \overline{(3)(5)(8,9)}; \overline{6}\right\} \qquad r = 4$$

$U = \{g_1, x_6\}$

$$P_U = \left(\overline{1,2,4}; \overline{3,8}; \overline{5,6,9}; \overline{7}\right).$$

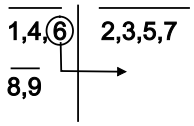
$$P_U|P_F = \left\{\overline{(1,4)(2)}; \overline{(3)(8)}; \overline{(5,6)(9)}; \overline{7}\right\} \qquad r = 3$$

Zatem tylko dla $U = \{g_1, x_6\}$ istnieje dekompozycja o strukturze, jak na rys. 3.13, której realizacja wymaga tylko czterech komórek typu 3/1. Obliczenia uzasadniające taką dekompozycję są podane poniżej.

Najpierw wyznaczamy podział P_V:

$$P_U = \left(\overline{1,4,6}; \overline{2,3,5,7}; \overline{8,9}\right).$$

$$P_U|P_F = \left\{\overline{(1)(2)}; \overline{(3)(6)}; \overline{(4,5)}; \overline{(7)(8)(9)}; \overline{(10)}\right\}$$



Rys. 1.14. Konstrukcja podziału Π_G

Do wyznaczenia Π_G dwa pierwsze bloki podziału P_V przeznaczamy do pierwszego bloku podziału Π_G, natomiast trzeci blok P_V przeznaczamy do drugiego bloku Π_G (rys. 3.14).

Ale tak utworzony Π_G jest sprzeczny z warunkami narzuconymi przez podział ilorazowy $P_U \mid P_F$, z którego wynika, że mintermy o numerach (5,6) oraz (9) muszą należeć do różnych bloków Π_G. Aby tak było, należy element 6 (rys. 3.14) „przenieść” do drugiego bloku. Będzie to możliwe, gdy mintermy 1,4 będą „oddzielone” od mintermu 6. Z tab. 3.22a wynika, że mintermy 1 i 6 różnią się na pozycji x_2, natomiast 4 i 6 różnią się na pozycjach x_2 i x_3. Zatem $W = \{x_2\}$, czyli $V' = \{x_2, x_4, x_5\}$. Na podstawie nowego $P_{V'} = P_V \cdot P(x_2)$:

$$P_{V'} = \left(\overline{1,4}; \overline{2,5,7}; \overline{8,9}\right).$$

łatwo wyznaczyć $\Pi_{G'}$:

$$\Pi_{G'} = (\overline{1,4,8,9}; \overline{2,3,6,7}),$$

spełniający warunek $P_U \cdot \Pi_{G'} \leq P_F$, co uzasadnia schemat blokowy dekompozycji pokazany na rys. 3.13.

1.3 Dekompozycja funkcjonalna – model ogólny

Niech $X = \{x_1,...,x_n\}$ będzie zbiorem zmiennych wejściowych funkcji $F(x_1,...,x_n)$ specyfikowanej zbiorem kostek typu *fr*. Niech U i V będą dwoma rozłącznymi podzbiorami zbioru X takimi, że $U \cup V = X$.

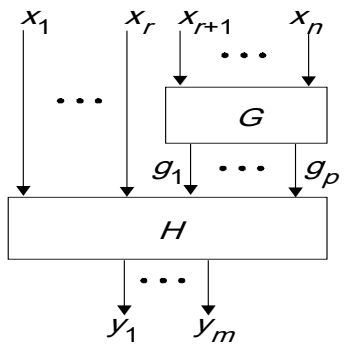
Załóżmy, że zbiór zmiennych $x_1,...,x_n$ został w odpowiedniej tablicy funkcji przenumerowany w taki sposób, że $U = \{x_1,...,x_r\}$ oraz $V = \{x_{r+1},...,x_n\}$. Niech dla dowolnej n-krotki x, pierwsze r składowe będą oznaczone przez x^U, a ostatnie $s = n - r$ składowe, przez x^V.

DEFINICJA 3.1

Niech F będzie funkcją *fr* posiadającą $n > 0$ wejść oraz $m > 0$ wyjść, w której para (U,V) jest określona jak wyżej. Załóżmy, że F jest specyfikowana zbiorem kostek K. Niech G będzie funkcją *fr* o $s = n - r$ wejściach i p wyjściach oraz H będzie funkcją *fr* o $r + p$ wejściach i $m > 0$ wyjściach.

Parę (G,H) nazywamy dekompozycją szeregową rozłączną funkcji F (rys. 3.15) ze względu na (U,V), jeśli, dla każdego mintermu b odpowiedniego do K, $G(b^V)$, jest zdefiniowana a ponadto $G(b^V) \in \{0,1\}^p$ oraz

$$F(b) \supseteq H(b^U, G(b^V)\}.$$



1.15. Dekompozycja szeregowa – model ogólny

TWIERDZENIE 3.4

Funkcja F typu *fr* określona zbiorem kostek K ma dekompozycję szeregową rozłączną ze względu na (U,V) wtedy i tylko wtedy, gdy istnieje nakrycie β_G na K takie, że:

$$\beta_V \leq \beta_G \text{ oraz}$$
$$\beta_U \bullet \beta_G \leq \beta_F,$$

gdzie β_U i β_V są nakryciami indukowanymi przez U i V oraz β_F jest nakryciem wyjściowym.

Analogiczne twierdzenie obowiązuje również w przypadku, gdy nakrycia β zostaną zastąpione systemami zbiorów σ.

TWIERDZENIE 3.5

Funkcja *F* typu *fr* określona zbiorem kostek *K* ma dekompozycję szeregową rozłączną ze względu na (*U*,*V*) wtedy i tylko wtedy, gdy istnieje system zbiorów σ_G na *K* taki, że:

$\sigma_V \leq \sigma_G$ oraz

$\sigma_U \bullet \sigma_G \leq \sigma_F$,

gdzie: $\sigma_U = max\, \beta_U$ oraz $\sigma_V = max\, \beta_V$ są systemami zbiorów indukowanymi przez *U* i *V* oraz $\sigma_F = max\, \beta_F$ jest odpowiednim systemem wyjściowym.

Jak wynika z powyższych twierdzeń głównym zadaniem w obliczeniu dekompozycji funkcji *F* przy danych zbiorach *U* i *V* jest znalezienie nakrycia β_G, które spełnia warunki twierdzenia 3.4. Podział ten nie tylko weryfikuje istnienie dekompozycji, ale również pozwala na wyznaczenie tablic *fr* dla funkcji *G* i *H*. Wynika to z faktu, że nakrycia β_V i β_G oraz $\beta_U \bullet \beta_G$ i β_F umożliwiają skonstruowanie odpowiednio tablic funkcji *G* i *H*. Konstrukcję tych tablic, a także metody obliczania β_G omówimy dokładnie w dalszej części rozdziału. Na razie zajmiemy się podstawowymi zasadami tych obliczeń.

Skoro β_G musi być konstruowany przez sklejanie bloków nakrycia β_V, to dwa bloki B_i i B_j nakrycia β_V są zgodne, jeśli nakrycie γ_{ij} uzyskane z β_V przez sklejenie B_i i B_j do jednego bloku i pozostawienie wszystkich pozostałych bloków bez zmiany, spełnia drugi warunek twierdzenia 3.4, czyli $\beta_U \bullet \gamma_{ij} \leq \beta_F$. W przeciwnym przypadku bloki B_i i B_j są niezgodne.

Podzbiór δ bloków nakrycia β_V jest klasą zgodną, jeśli bloki w zbiorze δ są parami zgodne. Klasa zgodna jest klasą maksymalną, jeśli nie jest podzbiorem żadnej innej klasy zgodnej. Maksymalne klasy zgodne umożliwiają konstrukcję β_G. *B*loki nakrycia β_G powstają z klas zgodności tworzących minimalne pokrycie zbioru wszystkich bloków β_V. Wyselekcjonowane według minimalnego pokrycia klasy zgodne są następnie redukowane o bloki powtarzające się i przyjmowane jako bloki nakrycia β_G (lub σ_G), po podstawieniu pierwotnych elementów w postaci numerów kostek funkcji *F* .

PRZYKŁAD 3.8

Dla funkcji *F* z tablicy 3.23 i zbiorów $U = \{x_1\}$ i $V = \{x_2,x_3,x_4\}$ nakrycia β_U, β_V są następujące:

$$\beta_V = \{\overline{1,2,3}; \overline{3,7}; \overline{6,7}; \overline{5}; \overline{4,6}; \overline{1,2}; \overline{4}\}$$

Tablica 1.23

x_1	x_2	x_3	x_4	y_1	y_2
0	0	*	0	1	1
1	0	*	0	1	0
*	0	0	*	1	–
*	*	1	1	0	–
*	1	1	0	0	0
*	1	*	1	–	1
0	*	0	1	1	–

Uwzględniając, że $\sigma_F = \{\overline{4,5}; \overline{4,6}; \overline{2,3,7}; \overline{1,3,6,7}\}$ możemy obliczyć, iż sklejenie bloków B_1 i B_2 prowadzi do γ_{12} = {1,2,3,7}, co po obliczeniu $\beta_U \cdot \gamma_{12} = \{\overline{1,3,7}; \overline{2,3}\}$ pozwala wnioskować, że $\beta_U \bullet \gamma_{12} \leq \sigma_F$, zatem ($B_1$, B_2) jest parą zgodną. Inaczej jest z parą (B_1, B_3), dla której γ_{13} = {1,2,3,6,7}, stąd, $\beta_U \cdot \gamma_{13} = \{\overline{1,3,6,7}; \overline{2,3,6}\} \not\leq \sigma_F$. ($B_1$, B_3) jest więc parą niezgodną. W rezultacie wszystkie pary zgodne są następujące: (B_1, B_2), (B_1, B_6), (B_2, B_3), (B_2, B_6), (B_4, B_7) i (B_5, B_7). Łatwo więc wyznaczyć maksymalne klasy zgodne: {B_1, B_2, B_6},{B_2, B_3}, {B_4, B_7} i {B_5,B_7}. Minimalne pokrycie zbioru {B_1,..., B_7} podanymi wyżej klasami prowadzi do rozwiązania: {B_1, B_2, B_6},{B_4}, {B_3} i {B_5,B_7}. Na tej podstawie nakrycie $\beta_G = \{\overline{1,2,3,7}; \overline{5}; \overline{6,7}; \overline{4,6}\}$ (po uwzględnieniu pierwotnych numerów kostek z bloków β_V). Łatwo sprawdzić, że $\beta_V \leq \beta_G$, ponadto $\beta_U = \{\overline{1,3,4,5,6,7}; \overline{2,3,4,5,6}\}$, $\beta_F = \{\overline{4,5}; \overline{4,6}; \overline{2,3,7}; \overline{1,3,6,7}\}$, czyli:

$$\beta_U \cdot \beta_G = \{\overline{1,3,7}; \overline{2,3}; \overline{5}; \overline{6,7}; \overline{6}; \overline{4,6}\} \leq \beta_F$$

Warunki twierdzenia 3.4 są więc spełnione i odpowiednia dekompozycja istnieje. Należy więc wyznaczyć tablice opisujące funkcje *G* i *H*.

W tym celu będziemy analizowali $\beta_V \leq \beta_G$, oraz $\beta_U \bullet \beta_G$ i β_F. Tablica funkcji *G* powstaje w następujący sposób. Dla każdego bloku *B* nakrycia β_V tworzymy kostkę, której poszczególne pozycje reprezentują wartości zmiennych wejściowych funkcji *G*. Wartości te dla zmiennej x_i są wyznaczane w zależności od przynależności bloku *B* do bloku nakrycia β_i. Z kolei wartość funkcji *G* dla obliczonej w ten sposób kostki wynika z przynależności bloku *B* do bloku β_G.

PRZYKŁAD 3.9

Dla funkcji *F* z przykładu 3.8 otrzymana w ten sposób tablica podana jest w tablicy 3.24a.

Tablica 1.24

a)

Blok	x_2	x_3	x_4	g_1	g_2
1,2,3	0	0	0	0	0
3,7	0	0	1	0	0
6,7	1	0	1	1	0
5	1	1	0	0	1
4,6	1	1	1	1	1
1,2	0	*	0	0	0
4	*	1	1	1	1

b)

Blok	x_1	g_1	g_2	y_1	y_2
1,3,7	0	0	0	1	1
2,3	1	0	0	1	0
5	*	0	1	0	0
6,7	0	1	0	1	1
6	*	1	*	–	1
4,6	*	1	1	0	1

W analogiczny sposób tworzymy tablicę funkcji *H*. Każdemu blokowi *B* z iloczynu:

$$\beta_U \cdot \beta_G = \{\overline{1,3,7}; \overline{2,3}; \overline{5}; \overline{6,7}; \overline{6}; \overline{4,6}\} \leq \beta_F$$

przyporządkowujemy kostkę o składowych x_1, g_1, g_2, przy czym wartości tych składowych wynikają z przynależności bloku *B* do bloków nakrycia β_1 oraz β_G. Należy więc określić kodowanie bloków $\beta_G = \{\overline{1,2,3,7}; \overline{5}; \overline{6,7}; \overline{4,6}\}$, które tu przyjmujemy następująco: 00 dla {1,2,3,7}, 01 dla {5}, 10 dla {6,7} oraz 11 dla {4,6}. Wartości funkcji *H* wynikają z przynależności bloków *B* do poszczególnych bloków z σ_F. Przypomnijmy, że bloki σ_F (por. przykład 3.8) są kodowane następująco: 4,5, jako 00; 4,6 jako 01; 2,3,7 jako 10 oraz 1,3,6,7 jako 11. W rezultacie tablica *H* jest taka, jak w tablicy 3.24b.

Obliczenia tablic funkcji *G* i *H* w przykładzie 3.8, jak też obliczenie β_G w przykładzie 3.9 są proste i nie wymagają stosowania wszystkich, potrzebnych do tego celu procedur. W bardziej skomplikowanych przypadkach obliczenia te wymagają stosowania dodatkowych procedur, których rolę wyjaśnimy poniżej.

Przy konstrukcji tablicy funkcji *G* pierwszą czynnością jest wyznaczanie kostki *C* odpowiadającej blokowi *B* z nakrycia β_G oraz wartości funkcji *G* dla tej kostki, tj. *G*(*C*). Kostkę C_i kojarzoną z B_i nazywać będziemy reprezentantem bloku B_i. W ogólnym przypadku może się zdarzyć, że dla dwóch różnych B_i, B_j, kostki C_i, C_j będą miały niepuste przecięcie, a jednocześnie odpowiadające im wartości funkcji *G* będą różne:

$$C_i \cap C_j \neq \phi,\ G(C_i) \neq G(C_j)$$

Jest to sytuacja niedopuszczalna z punktu widzenia niesprzeczności tablicy specyfikującej funkcję.

PRZYKŁAD 3.9.c.d.

Celem niedopuszczenia do takich sprzeczności należy odpowiednio modyfikować kostki reprezentujące bloki $B \in \beta_V$. Niech $C = r(B)$ oraz niech $B_1, ..., B_k$ oznaczają bloki z β_V, dla których:

$B \subset B_1$, $G(C_1) \neq G(C)$

$B \subset B_2$, $G(C_2) \neq G(C)$

-
-
-

$B \subset B_k$, $G(C_k) \neq G(C)$

gdzie: $C, C_1, ..., C_k$ są reprezentantami bloków $B, B_1,..., B_k$.

W takiej sytuacji sprzeczność w tablicy funkcji G usuniemy zastępując kostkę C kostką (lub kostkami) obliczonymi jako różnica: $C - \{C_1,..., C_k\}$. Różnicę taką najwygodniej jest obliczyć jako przecięcie C z uzupełnieniem $\{C_1,..., C_k\}$, czyli kostka zmodyfikowana:

$C' = C \cap \{C_1,..., C_k\}$.

PRZYKŁAD 3.10

Rozważmy dekompozycję funkcji F z tablicy 3.25 dla $U = \{x_1,x_2\}$ i $V = = \{x_3,x_4,x_5,x_6\}$. Odpowiednie nakrycie β_V jest tu następujące:

$$\beta_V = \{\overset{B_1}{\overline{7,8}}; \overset{B_2}{\overline{3,4,8}}; \overset{B_3}{\overline{8}}; \overset{B_4}{\overline{4,8}}; \overset{B_5}{\overline{1,5,8}}; \overset{B_6}{\overline{3,5,8}}; \overset{B_7}{\overline{6,8}}; \overset{B_8}{\overline{1,2,5,8}};\}$$

Tablica 1.25

	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3
1	*	0	1	*	0	0	1	–	–
2	0	*	1	1	0	0	1	–	–
3	1	*	*	*	0	1	–	1	–
4	0	*	0	*	*	1	–	1	–
5	1	*	1	*	0	*	–	–	1
6	*	*	1	0	1	*	0	0	0
7	*	*	0	*	0	0	–	–	0
8	1	1	*	*	*	*	0	–	–

PRZYKŁAD 3.10 c.d.

Łatwo sprawdzić, że nakrycie:

$$\sigma_G = \{\overline{3,4,7,8};\ \overline{3,5,8};\ \overline{6,8};\ \overline{1,2,5,8}\}^{\ 00\ \ 01\ \ 10\ \ 11}$$

spełnia warunki twierdzenia 3.4, zatem dekompozycja istnieje.

Dla kolejnych bloków β_V, ich reprezentantów obliczamy albo wg zawartości tych bloków w blokach nakryć β_i albo bezpośrednio z tablicy 3.25, korzystając z faktu, że reprezentant bloku *B* jest iloczynem kostek zawartych w *B*. Na przykład dla $B_1 = \{7,8\}$, $B_2 = \{3,4,8\}$, $B_3 = \{8\}$,

$C_1 = r(B_1) = 0{*}00 \cap {*}{*}{*}{*} = 0{*}00$

$C_2 = r(B_2) = {*}{*}01 \cap 0{*}{*}1 \cap {*}{*}{*}{*} = 0{*}01$

$C_3 = r(B_3) = {*}{*}{*}{*}$

$C_4 = r(B_3) = 0{*}{*}1$

Pełna tablica reprezentantów podana jest w tabl. 3.26.

Tablica 1.26

Blok β_V	Reprezentant	Wyjścia
B_1	0 * 0 0	0 0
B_2	0 * 0 1	0 0
B_3	* * * *	0 0
B_4	0 * * 1	0 0
B_5	1 * 0 0	1 1
B_6	1 * 0 1	0 1
B_7	1 0 1 *	1 0
B_8	1 1 0 0	1 1

Skoro $B_1 \subseteq \{3,4,7,8\}$, któremu w β_G został przyporządkowany kod 00, to $G(0{*}00) = 00$. Podobnie dla $B_2 \subseteq \{3,4,7,8\}$ mamy $G(0{*}01) = 00$. Zatem przyjmując, że $G(C_3) = 00$ ($B_3 \subseteq \{3,4,7,8\}$, można więc jako wartość *G* dla kostki C_3 przyjąć kod bloku {3,4,7,8}), konflikt znajdujemy dla kostek reprezentujących bloki B_5, B_6, B_7, B_8, stąd następująca modyfikacja C_3:

$$\{({*}{*}{*}{*})\} - \{1{*}00,\ 1{*}01,\ 101{*},\ 1100\} = \mathit{COMPLEMENT}\{1{*}00,\ 1{*}01,\ 101{*},\ 1100\} = \{0{*}{*}{*},\ {*}11{*}\}.$$

Omawiana wyżej dekompozycja, w której zbiory U i V są rozłączne, jest przypadkiem szczególnym dekompozycji, dla której założenie $U \cap V = \phi$ jest nieistotne. Istotne jest jedynie założenie dotyczące sensownej liczby wejść do poszczególnych komponentów.

Niech U i V będą podzbiorami X takimi, że $U \cup V = X$. Załóżmy, że zmienne $x_1,...,x_n$ są uporządkowane w taki sposób, iż $U = \{x_1,...,x_r\}$ i $V = \{x_{n-s+1},...,x_n\}$.

DEFINICJA 3.2

Niech F będzie funkcją o $n > 0$ wejściach i $m > 0$ wyjściach, i niech (U,V) będzie określone jak wyżej. Zakładamy też, że F jest specyfikowana zbiorem kostek K. Niech G będzie funkcją o s wejściach i p wyjściach, natomiast H funkcją fr o $r + p$ wejściach i m wyjściach. Parę (G,H) nazywamy dekompozycją funkcji F ze względu na (U,V), jeśli dla każdego mintermu b odpowiedniego do K, $G(b^V) \in \{0,1\}^p$ oraz

$$F(b) \supseteq H(b^U, G(b^V)).$$

TWIERDZENIE 3.6

Jeśli istnieje nakrycie β_G na K takie, że:

$$\beta_V \leq \beta_G, \text{ oraz}$$
$$\beta_U \bullet \beta_G \leq \beta_F,$$

to F ma dekompozycję szeregową ze względu na (U,V).

PRZYKŁAD 3.11

Dla funkcji z tablicy 3.23 rozważmy dekompozycję nierozłączną ze zbiorami:

$U = \{x_1, x_4\}$ i $V = \{x_2, x_3, x_4\}$. Mamy tu:

$$\beta_U = \{\overline{1,3,5}; \overline{3,4,6,7}; \overline{2,3,5}; \overline{3,4,6}; \},$$
$$\beta_V = \{\overline{1,2,3}; \overline{3,7}; \overline{1,2}; \overline{4}; \overline{6,7}; \overline{5}; \overline{4,6}\},$$

Do obliczenia β_G zastosujemy algorytm wyznaczania klas zgodnych wg par niezgodności: (B_1, B_4), (B_1, B_6), (B_1, B_7), (B_2, B_4), (B_2, B_6), (B_2, B_7), (B_3, B_6), (B_4, B_5), i (B_5, B_7). Utworzona według par niezgodnych formuła typu iloczyn sum i jej kolejne przekształcenia są następujące:

$$(B_1 + B_4)\,(B_1 + B_6)(B_1 + B_7)\,(B_2 + B_4)\,(B_2 + B_6)\,(B_2 + B_7)\,(B_3 + B_6)\,(B_4 + B_5)\,(B_5 + B_7) =$$
$$= (B_1 + B_4B_6B_7)(B_2 + B_4B_6B_7)(B_3 + B_6)\,(B_5 + B_4B_7) = B_1B_2B_3B_5 + B_1B_2B_5B_6 +$$
$$+ B_1B_2B_3B_4B_7 + B_4B_6B_7.$$

Klasy zgodne uzyskamy odejmując od zbioru $\{B_1,\ldots,B_7\}$, zbiory tych B_i, które występują w jednym składniku obliczonego wyżej wyrażenia typu „suma iloczynów". Stąd:

$$\{B_1,\ldots, B_7\} - \{B_1, B_2, B_3, B_5\} = \{B_4, B_6, B_7\}$$
$$\{B_1,\ldots,B_7\} - \{B_1, B_2, B_5, B_6\} = \{B_3, B_4, B_7\}$$
$$\{B_1,\ldots,B_7\} - \{B_1, B_2, B_3, B_4, B_7\} = \{B_5, B_6\}$$
$$\{B_1,\ldots,B_7\} - \{B_4, B_6, B_7\} = \{B_1, B_2, B_3, B_5\}$$

Z powyższych klas bloków zgodnych można wyselekcjonować dwie: $\{B_1, B_2, B_3, B_5\}$ oraz $\{B_4, B_6, B_7\}$, pokrywające zbiór $\{B_1,\ldots, B_7\}$, czyli $\beta_G = \{\overline{B_1,B_2,B_3,B_5}; \overline{B_4,B_6,B_7}\}$. W tym przypadku rozwiązanie jest trywialne. Wracając do pierwotnych kostek tworzących poszczególne bloki B_i otrzymujemy:

$$\beta_G = \{\overset{0}{\overline{1,2,3,6,7}}; \overset{1}{\overline{4,5,6}}\},$$

w którym od razu określamy kodowanie bloków.

PRZYKŁAD 3.11 c.d.

Identyczny wynik uzyskamy obliczając klasy zgodne algorytmem korzystającym z par zgodnych. Następujące pary są zgodne: (B_1, B_2), (B_1, B_3), (B_1, B_5), (B_2, B_3), (B_2, B_5), (B_3, B_4), (B_3, B_5), (B_3, B_7), (B_4, B_6), (B_4, B_7), (B_5, B_6), (B_6, B_7). Na tej podstawie wyznaczamy zbiory S_j i tworzymy kolejne listy (rodziny RKZ$_k$) klas zgodnych (porównaj algorytm z rozdz. 1), które dla uproszczenia zapisów podawać będziemy w postaci zbiorów indeksów:

$S_1 = \phi$	{1}
$S_2 = \{1\}$	{1, 2}
$S_3 = \{1, 2\}$	{1, 2, 3}
$S_4 = \{3\}$	{3, 4}, {1, 2, 3}
$S_5 = \{1, 2, 3\}$	~~{3, 5}~~, {1, 2, 3, 5}, {3, 4}
$S_6 = \{4, 5\}$	{5, 6}, {4, 6}, {1, 2, 3, 5}, {3, 4}
$S_7 = \{3, 4, 6\}$	~~{6, 7}~~, {4, 6, 7}, ~~{3, 7}~~, {3, 4, 7}, {5, 6}, {1, 2, 3, 5}

W obliczeniach powyższych przekreśleniem zaznaczyliśmy zbiory, które są usuwane na mocy definicji maksymalnych klas zgodnych. Dysponując nakryciem β_G, łatwo już wyznaczyć tablice funkcji G i H. Podane są one odpowiednio w tablicach 3.27a i 3.27b.

Tablica 1.27

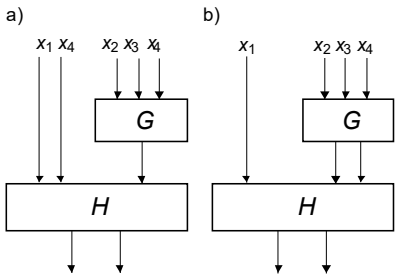
a)

x_2	x_3	x_4	g
0	0	0	0
0	0	1	0
0	*	0	0
*	1	1	1
1	0	1	0
1	1	0	1
1	1	1	1

b)

x_1	x_4	g	y_1	y_2
0	0	0	1	1
*	0	1	0	0
0	1	0	1	1
*	1	1	0	1
1	0	0	1	0
*	1	0	1	1

W wielu przypadkach dekompozycja szeregowa z przecinającymi się zbiorami U i V – zwana często dekompozycją nierozłączną – jest pożyteczna. Załóżmy realizację funkcji z powyższego przykładu na komórkach FPGA o strukturze: 3 wejścia, 1 wyjście. Obliczona dekompozycja nierozłączna (rys. 3.16a) wymaga do całkowitej realizacji tej zdekomponowanej funkcji 3 komórek: jedną do funkcji G i dwie do funkcji H (po jednej na każde wyjście). Dekompozycja rozłączna o strukturze $H(x_1,x_2,G(x_3,x_4))$, również wymagałaby do swojej realizacji 3 komórek; niestety, taka dekompozycja nie istnieje. Istnieje natomiast dekompozycja rozłączna $H(x_1,G(x_2,x_3,x_4))$ – rys. 3.16b, w której funkcja G ma 2 wyjścia, i dla tego jej realizacja wymaga aż 4 komórek (dwóch dla funkcji G i dwóch dla H).



Rys. 1.16. Dekompozycja szeregowa: a) nierozłączna; b) rozłączna

1.4 Dekompozycja liniowa

Dekompozycja liniowa polega na łączeniu zmiennych w pary i wprowadzaniu ich za pośrednictwem bramek EXOR do głównego układu realizującego detekcję i klasyfikację (czyli indeksowanie) danych. Jest to istotne, gdyż prowadzi do dalszego redukowania liczby wejść w generatorach indeksu. I pewnie z tych powodów dekompozycja liniowa jest intensywnie wykorzystywana w syntezie funkcji generowania indeksów [3.20].

W dotychczasowych rozwiązaniach problem dekompozycji liniowej ogranicza się do syntezy silnie nieokreślonych funkcji boolowskich, o specyficznej własności: $|D^n| = k$. Ograniczenie tego problemu do funkcji, w których wszystkie wektory wyjściowe są różne, jest niepotrzebne, gdyż problem ten można sprowadzić do ogólniejszego zadania syntezy systemów funkcji boolowskich o dowolnych wyjściach. Inaczej mówiąc lepiej przyjąć założenie, że liczność zbioru wektorów wyjściowych Y funkcji F jest $\leq |D^n|$.

Ogólnie rzecz biorąc problem można sprowadzić do wskazania warunków jakie muszą być spełnione, aby funkcja F posiadała dekompozycję z dwu-argumentowymi funkcjami G.

$$F = H(G_1(X_1), G_2(X_2),..., x_n),$$

gdzie: $X_1 = \{x_{i_1}, x_{i_2}\}$, $X_2 = \{x_{i_3}, x_{i_4}\}$,

Funkcję G nazywać można g-argumentem, stosownie do ogólnej teorii dekompozycji funkcji boolowskich. Przyjmujemy również założenie, że w dekompozycji liniowej stosujemy funkcje o zredukowanej liczbie argumentów, czyli tzw. minimalno-argumentowe reprezentacje funkcji F.

Przy tak postawionym zagadnieniu wygodne jest stosowanie uporządkowanego i zwartego sposobu reprezentowania funkcji, jakim jest rachunek podziałów. Szczegóły dotyczące tego sposobu przedstawiana funkcji zostały już omówione w rozdz. 1.4 i nie będą tu dyskutowane. Niezbędne jest jednak krótkie przypomnienie, gdyż własności podziałów będą wykorzystywane do skonstruowania metody syntezy układów bramkowych, omawianej w dalszej części artykułu.

Rozpatrzmy funkcję logiczną zdefiniowaną w tab. 3.28, w której symbole T: 1,2,...,11 reprezentują wektory, $x_1,...,x_5$ reprezentują zmienne wejściowe, zaś y – zmienną wyjściową.

Dla tego przykładu, różne wartości, przyjmowane przez zmienną wejściową x_4, wyznaczają podzbiory wektorów {1,5,7,8} i {2,3,4,6,9,10,11}. Dla wektorów 1,5,7 i 8, $x_4 = 0$, dla 2,3,4,6,9,10,11, $x_4 = 1$. Zatem wiedząc, że $x_4 = 0$, wiemy, że wybrany został wektor 1,5,7 lub 8, nie wiemy jednak który z nich. Analogicznie wiedząc, że $x_3 = 1$ wiemy, że wybrano symbol 2,4,5 lub 8, nie jesteśmy jednak wstanie wskazać, który z nich konkretnie. W ten sposób informacja modelowana jest przez podziały, pokrycia i systemy zbiorów.

Tablica 1.28

T	x_1	x_2	x_3	x_4	x_5	y
1	0	0	0	0	0	0
2	0	0	1	1	1	0
3	0	1	0	1	0	0
4	0	1	1	1	1	0
5	0	1	1	0	0	0
6	0	0	0	1	1	1
7	0	1	0	0	0	1
8	0	1	1	0	1	1
9	1	1	0	1	0	1
10	1	0	0	1	1	1
11	1	0	0	1	0	1

PRZYKŁAD 3.12

Dla funkcji z tab, 3.29 podziały wejściowe i wyjściowy są następujące

$$P(x_1) = \{\overline{1,2,3,4,5,6,7,8}; \overline{9,10,11}\},$$
$$P(x_2) = \{\overline{1,2,6,10,11}; \overline{3,4,5,7,8,9}\},$$
$$P(x_3) = \{\overline{1,3,6,7,9,10,11}; \overline{2,4,5,8}\},$$
$$P(x_4) = \{\overline{1,5,7,8}; \overline{2,3,4,6,9,10,11}\},$$
$$P(x_5) = \{\overline{1,3,5,7,9,11}; \overline{2,4,6,8,10}\},$$
$$P_F = \{\overline{1,\dots,5}; \overline{6,\dots,11}\},$$

W celu uproszczenia zapisów wektory $a,b \in \{0,1\}^n : F(a) \neq F(b)$, będą reprezentowane liczbami $p,q \in K = \{1,..., | D^n |\}$.

Niech F będzie wielowyjściową funkcją boolowską, gdzie zbiór zmiennych $X = (x_1,\dots, x_n)$, zbiór wektorów (mintermów) $M = (m_1,\dots, m_t)$. Przez C_{pq}, gdzie p, q są wektorami z M, dla których $F(p) \neq F(q)$ oznaczamy zbiór niezgodności, definiowany jak następuje:

$$C_{pq} = \{x \in X : x(p) \neq x(q)\} \text{ dla } p,q = 1,\dots,t \text{ and } p < q\}. \qquad (3\text{-}3)$$

Rodzinę zbiorów $C = (C_1,\dots, C_n)$, dla wszystkich C_{pq} oznaczać będziemy RC_{pq}. W obliczeniach par zmiennych tworzących dekompozycję liniową posługiwać się będziemy rodziną zbiorów C_{pq}.

Problem wyznaczenia odpowiednich warunków, oparty jest na pojęciu funkcji rozdzielającej. O funkcji g będziemy mówić, że jest funkcją rozdzielająca (dystrybucją) pary wektorów p,q wtedy i tylko wtedy, gdy p

i q należą do różnych bloków podziału $P(g)$. W uproszczeniu można powiedzieć, że w takim przypadku funkcja g umożliwia rozdzielenie wektorów p i q.

Z przyjętej definicji wynika, że każda jednoargumentowa funkcja $g(x_j) = x_j$, gdzie $x_j \in C_{pq}$ jest dystrybucją p, q. Ponadto jeśli g rozdziela parę p, q, to również parę tę rozdziela funkcja $\bar{g}$.

Wykorzystując powyższe ustalenia można teraz warunkowi dekompozycji nadać wygodniejszą postać: $F = H(g_1, g_2,..., g_t)$ wtedy i tylko wtedy, gdy dla każdej pary p, q: takiej, że (p,q) nie jest zawarta w jednym bloku podziału charakterystycznego P_F istnieje g_j rozdzielająca parę p, q. Zalety tego warunku zostaną obecnie wykorzystane do sformułowania metodyki syntezy układów w których argumenty z $X = (x_1,..., x_n)$ łączone będą w pary i wprowadzane do układu H za pośrednictwem funktorów $g(x_i, x_j)$ tzn. $F = H(g_1, g_2,..., g_t, x_a, x_b...)$.

Wykorzystując powyższe ustalenia można zaproponować sposób prowadzenia syntezy logicznej jako dekompozycję bezpośrednio na bramki, w odróżnieniu od dekompozycji funkcjonalnej, w której składowe są funkcjami opisanymi tablicami prawdy. Możliwe jest obliczenie podziału $P(g)$ dla dowolnej funkcji o skończonej liczbie zmiennych wejściowych i w następnym kroku usunięcie wszystkich zbiorów C_{pq}, dla których g nie jest dystrybucją. Obliczamy w ten sposób dekompozycję funkcji F i zapisujemy ją w postaci:

$$F = H\left(G\left(x_{i_1}, \dots, x_{i_t}\right), \dots, x_{i_n}\right).$$

Liczba argumentów funkcji H wynosi w tym przypadku co najwyżej $n - 1$, w związku z czym proces prowadzi do zmniejszenia złożoności funkcji.

TWIERDZENIE 3.7

Funkcja $g = x_i \oplus x_j$ jest funkcją rozdzielającą (dystrybucją) parę (p, q) wtedy i tylko wtedy gdy $\{x_i, x_j\} \notin C_{pq}$ i $\left|\{x_i, x_j\}\right| \cap C_{pq} = 1$.

Powyższe twierdzenie prowadzi do prostej metody dekomponowania funkcji i wyznaczania $g = g(x_i, x_j)$ bezpośrednio ze zbiorów C_{pq} i podziałów $P(x_i)$, np. podziałów generowanych przez funkcję jednoargumentową $g(x_i) = x_i$.

Zgodnie z koncepcją funkcji rozdzielającej, opracowany został prosty test weryfikujący istnienie dekompozycji liniowej.

TEST 3.1

$G = x_i \oplus x_j$ jest dekompozycją funkcji F, wtedy i tylko wtedy, gdy $\{x_i, x_j\} \notin RC_{pq}$, gdzie RC_{pq} jest rodziną zbiorów C_{pq}, dla których p oraz q należą do różnych bloków podziału wyjściowego P_F.

PRZYKŁAD 3.13

Dla funkcji F z tabeli 3.28 rodzinę RC_{pq} ograniczoną do zbiorów dwuelementowych zapisano w tabeli 3.29. Łatwo zauważyć, że RC_{pq} nie zawiera 3 następujących par: x_1, x_5; x_3, x_4; x_3, x_5.

Tablica 1.29

p,q	C_{pq}
1,6	x_4,x_5
1,11	x_1,x_4
2,8	x_2,x_4
2,10	x_1,x_3
3,6	x_2,x_5
3,11	x_1,x_2
4,6	x_2,x_3

Stąd wniosek, że możliwe są g-argumenty reprezentowane przez dekompozycje funkcji F:

$F = H(x_1 \oplus x_5, x_2, x_3, x_4)$; $F = H(x_3 \oplus x_4, x_1, x_2, x_5)$; $F = H(x_3 \oplus x_5, x_1, x_2, x_4)$.

Proces obliczania dekompozycji liniowej może być wykonywany iteracyjnie, aż do uzyskania dekompozycji $F = H(g_1, g_2,\ldots, X)$ z maksymalną liczbą funkcji G, co zapewnia minimalna liczbę argumentów dla funkcji H, oczywiście łącznie z g-argumentami typu EXOR.

Twierdzenie 3.7 i test 3.1 można uogólnić na przypadek tzw. dekompozycji wielokrotnej. Wtedy dodatkowym warunkiem na istnienie dekompozycji z parą funkcji $\{x_i \oplus x_j , x_k \oplus x_l\}$, jest sprawdzenie czy zbiór zmiennych $\{x_i, x_j, x_k, x_l\}$ jest zawarty w rodzinie RC_{pq}.

Dla uproszczenia zapisów uzupełnienie RC_{pq} względem rodziny wszystkich podzbiorów o liczności równej liczności analizowanych C_{pq} oznaczać będziemy COM(RC_{pq}).

Dla funkcji z przykładu 3.12 w celu sprawdzenia, które g-argumenty mogą tworzyć dekompozycję wielokrotną (w tym przypadku podwójną), należy obliczyć czteroelementowe zbiory RC_{pq}. W wyniku uzyskujemy, że wszystkie cztero-elementowe zbiory RC_{pq} są: $\{x_2\ x_3\ x_4\ x_5\}$, $\{x_1\ x_2\ x_3\ x_5\}$, $\{x_1\ x_2\ x_3\ x_4\}$. Zatem uzupełnienie COM(RC_{pq}) zawiera dwa zbiory $\{\{x_1\ x_2\ x_4\ x_5\}, \{x_1\ x_3\ x_4\ x_5\}\}$ i jeden z nich reprezentuje zmienne g-argumentów $x_1 \oplus x_5$, $x_3 \oplus x_4$. Na tej podstawie stwierdzamy, że funkcja z przykładu 3.12 ma dekompozycję: $F = H(x_1 \oplus x_5, x_3 \oplus x_4, x_2)$.

Inne uogólnienie Twierdzenia 3.7 dotyczy dekompozycji nierozłącznej. W tym przypadku dodatkowy warunek na istnienie dekompozycji z parą funkcji $\{x_i \oplus x_j , x_j \oplus x_k\}$ polega na sprawdzeniu, czy zbiór zmiennych $\{x_i, x_j, x_k\}$ jest zawarty w rodzinie RC_{pq}.

Omówioną metodykę obliczania dekompozycji liniowej skonfrontujemy teraz z algorytmem zaproponowanym w [3.20]. Obliczenia przeprowadzimy dla funkcji reprezentującej koder 1 z 10 – tab. 3.30.

Tablica 1.30

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	y_1	y_2	y_3	y_4	y_5	y_6
1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0

Dla funkcji tej w [3.20] obliczono dekompozycję liniową z 6 *g*-argumentami, które w tab. 3.30 oznaczono $y_1,\ldots, y_6$:

$$F = H(x_1 \oplus x_6,\ x_3 \oplus x_7,\ x_3 \oplus x_9,\ x_4 \oplus x_8,\ x_4 \oplus x_{10},\ x_5 \oplus x_6)$$

Obliczenia wykonano za pomocą heurystycznego algorytmu s-min. Ponieważ w uzyskanym rozwiązaniu nie ma zmiennej x_2, wygodnie będzie wprowadzić nowe oznaczenia zmiennych, zgodnie z tabelą 3.31.

Tablica 1.31

x_1	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9

Przy tych oznaczeniach uzyskuje się nowa tablicę kodera podaną w tabeli 3.32, jak też nowe wyrażenie na uzyskaną w [3.20] dekompozycję:

$$F = H(a_1 \oplus a_5,\ a_2 \oplus a_6,\ a_2 \oplus a_8,\ a_3 \oplus a_7,\ a_3 \oplus a_9,\ a_4 \oplus a_5)$$

Tablica 1.32

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	1

Dla uproszczonego wg tabeli 3.32 kodera 1 z 10 obliczamy rodzinę RC_{pq}, a następnie COM(RC_{pq}). W wyniku uzyskujemy: uzupełnienie rodziny RC_{pq} ma pusty zbiór par, ale zawiera wszystkie możliwe (84) podzbiory trój-elementowe zbioru $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$. Stąd wniosek, że dla kodera 1 z 10 istnieje dekompozycja nierozłączna wielokrotna reprezentowana trójelementowymi podzbiorami zbioru $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$, czyli

$$F = H(a_i \oplus a_j,\ a_j \oplus a_k,\ a_l \oplus a_m,\ a_m \oplus a_n,\ a_p \oplus a_q,\ a_q \oplus a_r) \qquad (3\text{-}4)$$

Tym samym potwierdza to istnienie dekompozycji:

$$F = H(a_1 \oplus a_5,\ a_2 \oplus a_6,\ a_2 \oplus a_8,\ a_3 \oplus a_7,\ a_3 \oplus a_9,\ a_4 \oplus a_5)$$

uzyskanej w [3.20], przy czym należy podkreślić, że dekompozycja taka istnieje dla wszystkich możliwych trójek:

$a_i\ a_j\ a_k$,

$a_l\ a_m\ a_n$,

$a_p\ a_q\ a_r$

Zatem przy zastosowaniu metody z [3.20] znalezienie jakiegokolwiek rozwiązania nie było trudne.

PRZYKŁAD 3.14

Wykazana własność dekompozycji (wg (3-4)) pozwala zapisać dekompozycję kodera 1 z 10 w naturalnym porządku zmiennych

$$F = H(a_1 \oplus a_2,\ a_2 \oplus a_3,\ a_4 \oplus a_5,\ a_5 \oplus a_6,\ a_7 \oplus a_8,\ a_8 \oplus a_9) = H(b_1, b_2, b_3, b_4, b_5, b_6), \qquad (3\text{-}5)$$

któremu odpowiada tablica indeksów przedstawiona w tabeli 3.33.

Tablica 1.33

b_1	b_2	b_3	b_4	b_5	b_6	f
1	0	0	0	0	0	1
0	0	0	0	0	0	2
1	1	0	0	0	0	3
0	1	0	0	0	0	4
0	0	1	0	0	0	5
0	0	1	1	0	0	6
0	0	0	1	0	0	7
0	0	0	0	1	0	8
0	0	0	0	1	1	9
0	0	0	0	0	1	10

W rozwiązaniu tym indeksy są reprezentowane wektorami 6-bitowymi $b_1\, b_2\, b_3\, b_4\, b_5\, b_6$.

Można przypuszczać, że stosując raz jeszcze metodykę zbiorów niezgodności uzyskamy rozwiązanie z mniejszą liczbą bitów. Ponieważ rodzina zbiorów niezgodności zawiera wszystkie możliwe pary $b_i\, b_j$, 12 zbiorów trójelementowych oraz żadnego zbioru 5 i 6 elementowego, znalezienie dekompozycji liniowej jest możliwe wyłącznie dla funkcji typu $\{x_i \oplus x_j\ ,\ x_j \oplus x_k\}$.

Z przeglądu w RC_{pq} zbiorów trójelementowych:

1. b_1, b_3, b_4;
2. b_1, b_5, b_6;
3. b_1, b_2, b_3;
4. b_1, b_2, b_4;
5. b_1, b_2, b_5;
6. b_1, b_2, b_6;
7. b_2, b_3, b_4;
8. b_2, b_5, b_6;
9. b_3, b_5, b_6;
10. b_3, b_4, b_5;
11. b_3, b_4, b_6;
12. b_4, b_5, b_6;

PRZYKŁAD 3.14 c.d.

wynika, że trójelementowymi zbiorami b_i, b_j, b_k możliwymi do utworzenia dekompozycji nierozłącznej są $\{b_1, b_3, b_5\}$, $\{b_1, b_3, b_6\}$, $\{b_1, b_4, b_5\}$, $\{b_1, b_4, b_6\}$, $\{b_2, b_3, b_5\}$, $\{b_2, b_3, b_6\}$, $\{b_2, b_4, b_5\}$, $\{b_2, b_4, b_6\}$. Ponieważ są wśród nich dwa zbiory rozłączne: $\{b_1, b_3, b_5\}$ i $\{b_2, b_4, b_6\}$, a ponadto w COM(RC_{pq}) jest zbiór 6 elementowy $b_1, b_2, b_3, b_4, b_5, b_6$, spełnione są warunki istnienia dekompozycji: $F = H(b_1 \oplus b_3, b_3 \oplus b_5, b_2 \oplus b_4, b_4 \oplus b_6)$. W rezultacie indeksy kodera 1 z 10 są 4-bitowe (tab. 3.34) i jest to rozwiązanie lepsze niż w [3.20].

Tablica 1.34

y_1	y_2	y_3	y_4	h
1	0	0	0	1
0	0	0	0	2
1	0	1	0	3
0	0	1	0	4
1	1	0	0	5
1	1	1	1	6
0	0	1	1	7
0	1	0	0	8
0	1	0	5	9
0	0	0	1	10

3.5 Dekompozycja równoległa

Dekompozycja funkcjonalna jest – podobnie jak redukcja argumentów – metodą syntezy logicznej, dostosowaną do implementacji na komórkach FPGA i pamięciach ROM. W związku z tym jej stosowanie w projektowaniu układów cyfrowych staje się coraz bardziej znaczące. W [3-20] problem dekompozycji określa się jako jedno z najpilniejszych zadań syntezy logicznej.

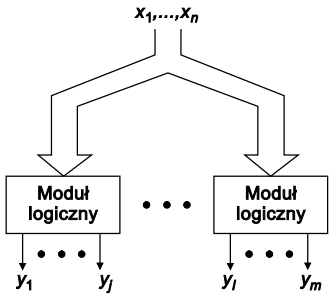
W szczególności proponuje się dalsze rozwijanie metody dekompozycji, polegającej na stosowaniu tzw. tablic dekompozycji (*Decomposition Chart*). Wydaje się jednak, że metoda zaproponowana w niniejszym rozdziale (podrozdz. 3.1, 3.2 oraz 3.3) jest znacznie skuteczniejsza. Jej zaletą jest stosowanie rachunku podziałów, co umożliwia sformułowanie wygodnego praktyce twierdzenia (Twierdzenie 3.3), w którym warunek wystarczający dekompozycji polega na wyznaczaniu podziałów na zbiorze ponumerowanych wektorów tablicy funkcji. Twierdzenie to wyraźnie wskazuje, że istotnym ograniczeniem dekompozycji jest podział wyjściowy funkcji (P_F) – im drobniejszy jest podział P_F, tym trudniejsze jest spełnienie warunku dekompozycji $P_U \cdot \Pi_G \leq P_F$. Stąd pomysł uzupełnienia procesu dekompozycji funkcjonalnej – dekompozycją równoległą.

D e k o m p o z y c j a r ó w n o l e g ł a w układzie opisanym przez:

$$(y_1,..., y_m) = F(x_1,..., x_n) \qquad (3\text{-}5)$$

to podział zbioru $\{y_1,..., y_m\}$ na takie rozłączne podzbiory $\boldsymbol{Y}_1,..., \boldsymbol{Y}_i,..., \boldsymbol{Y}_k$, z których każdy można zrealizować w układzie kombinacyjnym o argumentach odpowiednio $\boldsymbol{X}_1,..., \boldsymbol{X}_i,..., \boldsymbol{X}_k$, gdzie $\boldsymbol{X}_i \subseteq \{x_1,..., x_n\}$.

Sens powyższej dekompozycji wynika z ograniczeń konstrukcyjnych produkowanych modułów PLD i FPGA. Jeśli bowiem liczba wyjść, np. modułu PLD jest mniejsza od m, to do realizacji układu opisanego przez (3-5) można przeznaczyć (w zależności od m) kilka modułów, uzyskując w rezultacie układ pokazany na rys. 3.17. W układzie tym wyjścia y_i trzeba odpowiednio rozdzielić na poszczególne moduły. Wskazane jest, aby zbiór $\{y_1,..., y_m\}$ rozdzielić na takie podzbiory $\boldsymbol{Y}_i$, dla których odwzorowanie $\boldsymbol{Y}_i = h_i(\boldsymbol{X}_i)$ wymagałoby możliwie najmniejszej liczby argumentów $\boldsymbol{X}_i$, przy czym $\boldsymbol{X}_i \subseteq \{x_1,..., x_n\}$, $\boldsymbol{Y}_i \subseteq \{y_1,..., y_m\}$.



Rys. 1.17. Dekompozycja równoległa

Pełną informację o rozdziale zbioru $\boldsymbol{Y} = \{y_1,..., y_m\}$ na podzbiory $\boldsymbol{Y}_1,..., \boldsymbol{Y}_k$ uzyskać można z minimalno-argumentowych realizacji poszczególnych wyjść y_i.

PRZYKŁAD 3.15

Dla funkcji F z tablicy 3.35 redukty poszczególnych (pojedynczych) funkcji są odpowiednio:

y_1: $\{x_1, x_2, x_6\}$

y_2: $\{x_3, x_4\}$

y_3: $\{x_1, x_2, x_4, x_5, x_8\}$, $\{x_1, x_2, x_4, x_6, x_8\}$

y_4: $\{x_1, x_2, x_3, x_4, x_7\}$

y_5: $\{x_1, x_2, x_4\}$

y_6: $\{x_1, x_2, x_6, x_8\}$

Tablica 1.35

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y_1	y_2	y_3	y_4	y_5	y_6
1	0	0	0	1	1	1	0	0	0	0	0	0	–	0
2	1	0	1	0	0	0	0	0	0	0	–	1	0	1
3	1	0	1	1	1	0	0	0	0	1	1	0	1	1
4	1	1	1	1	0	1	0	0	0	1	1	1	1	0
5	1	0	1	0	1	0	0	0	0	0	0	–	0	1
6	0	0	1	1	1	0	0	0	1	1	0	1	0	0
7	1	1	1	0	0	0	0	0	1	0	–	0	1	0
8	1	0	1	1	0	1	0	0	1	1	0	0	–	1
9	1	0	1	1	0	1	1	0	–	1	0	1	–	1
10	0	0	0	1	1	1	0	1	0	0	1	0	–	1
11	0	0	0	1	1	0	0	1	–	–	1	0	0	0

Zatem rozsądnie jest funkcji G_1 przyporządkować wyjścia y_1, y_3, y_6, a funkcji G_2 wyjścia y_2, y_4, y_5 (rys. 3. 18), gdyż wtedy uzyska się minimalne zbiory argumentów dla komponentów G_1 i G_2 dekompozycji równoległej funkcji F, a mianowicie $\{x_1, x_2, x_4, x_6, x_8\}$ dla G_1 oraz $\{x_1, x_2, x_3, x_4, x_7\}$ dla G_2. Tablice funkcji G_1 i G_2 są podane w tablicach 3.36a i 3.36b.

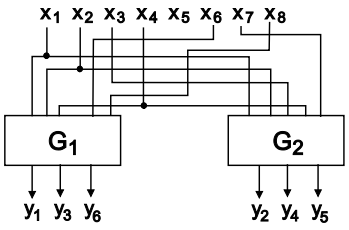
Tablica 1.36

a)

	x_1	x_2	x_4	x_6	x_8	y_1	y_3	y_6
1	0	0	1	1	0	0	0	0
2	1	0	0	0	0	0	0	1
3	1	0	1	0	0	0	1	1
4	1	1	1	1	0	0	1	0
5	0	0	1	0	0	1	0	0
6	1	1	0	0	0	1	1	0
7	1	0	1	1	0	1	0	1
8	0	0	1	1	1	0	1	1
9	0	0	1	0	1	–	1	0

b)

	x_1	x_2	x_3	x_4	x_7	y_2	y_4	y_5
1	0	0	0	1	0	0	0	0
2	1	0	1	0	0	0	1	0
3	1	0	1	1	0	1	0	1
4	1	1	1	1	0	1	1	1
5	0	0	1	1	0	1	1	0
6	1	1	1	0	0	0	0	1
7	1	0	1	1	1	1	1	–



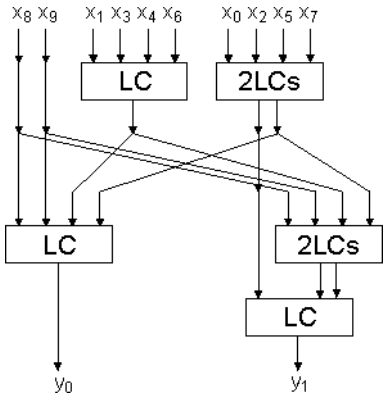
Rys. 1.18. Dekompozycja równoległa funkcji z przykładu 3.15

PRZYKŁAD 3.16

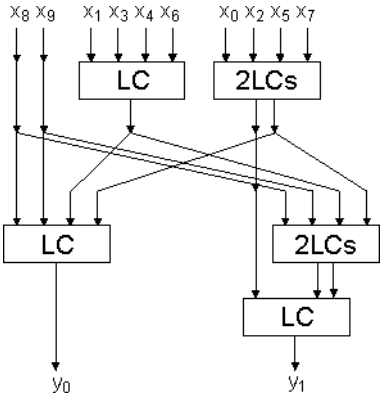
Wpływ dekompozycji zrównoważonej na ostateczny wynik syntezy w układzie FPGA zostanie pokazany na przykładzie pewnej funkcji *F* z 10 wejściami i 2 wyjściami (tab. 3.37), którą realizujemy w komórkach o 4 wejściach i 1 wyjściu (w uproszczeniu oznaczanych (4,1).

Tablica 1.37

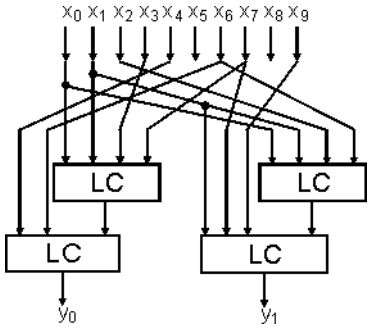
```
.type fr
.i 10
.o 2
.p 25
0101000000 00
1110100100 00
0010110000 10
0101001000 10
1110101101 01
0100010101 01
1100010001 00
0011101110 01
0001001110 01
0110000110 01
1110110010 10
0111100000 00
0100011011 00
0010111010 01
0110001110 00
0110110111 11
0001001011 11
1110001110 10
0011001011 10
0010011010 01
e.
```



Rys. 1.19. Dekompozycja funkcji F – strategia szeregowa



Rys. 1.21. Dekompozycja funkcji F – strategia szeregowa



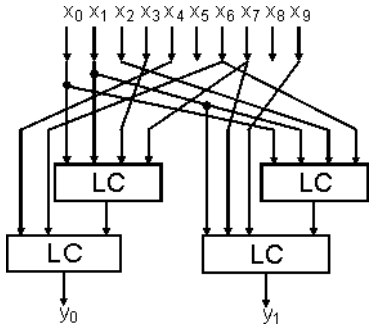
Rys. 1.20. Dekompozycja funkcji F – strategia równoległa

Skoro funkcja *F* ma 10 wejść i 2 wyjścia, pierwszym krokiem dekompozycji może być dekompozycja równoległa lub szeregowa. Jednak w tym przypadku zastosujemy dekompozycję szeregową. Algorytm wydziela funkcję *G* mającą wejścia x_1, x_3, x_4 i x_6,. Następny krok dotyczy więc funkcji 7-wejściowej *H*, dla której ponownie jest wykonywana dekompozycja szeregowa. Otrzymujemy blok *G* o 4 wejściach i 2 wyjściach (implementowany przez 2 komórki). Blok *G* ma na wejściu zmienne x_0, x_2, x_5, and x_7. Są to pierwotne zmienne wejściowe i na tym etapie nie jest zwiększona liczba poziomów, co widać na rys. 3.19.

W następnym kroku dokonujemy dekompozycji równoległej. Tworzy ona dwa komponenty, każdy o jednym wyjściu, natomiast odpowiednio o 4 i 5 wejściach. Pierwszy komponent realizuje pojedyncza komórka. Drugi komponent podlega dwupoziomowej dekompozycji szeregowej. Utworzona sieć może być zrealizowana za pomocą 7 komórek (4,1). Liczba poziomów ścieżki krytycznej wynosi 3.

PRZYKŁAD 3.16 c.d.

Ta sama funkcja zdekomponowana tak, że najpierw wykonuje się dekompozycję równoległą, jest realizowana w zupełnie innej strukturze (rys. 3.20).



Rys. 1.22 Dekompozycja funkcji F – strategia równoległa

Dekompozycja równoległa zastosowana bezpośrednio dla funkcji *F* tworzy 2 komponenty o 6 wejściach i jednym wyjściu. Każdy z nich podlega dwupoziomowej dekompozycji szeregowej. Dla pierwszego z nich możliwa do zastosowania jest rozłączna dekompozycja szeregowa z blokiem G_1 (4,1). Drugi komponent może być również zdekomponowany szeregowo, jednak liczba wyjść bloku G_2 wynosi 2. Aby zmniejszyć łączną liczbę komórek, można zastosować dekompozycję nierozłączną, uzyskując tylko 4 komórki. Tablice prawdy tych komórek podane są w tabeli 3.38. Ta znacząca zmiana struktury wynika z tego, że dekompozycja równoległa zmniejsza liczbę wejść obu komponentów, prowadząc do uproszczenia końcowej implementacji.

Tablica 1.38

a) funkcja G_1	b) funkcja H_1	c) funkcja G_2	d) funkcja H_2
0110 1	–01 0	0110 1	10–1 0
1101 1	011 1	0011 1	–101 1
1000 1	111 0	0100 1	–111 1
0010 1	100 1	1000 1	0011 0
0000 0	0–0 0	0101 1	0001 1
0101 0	110 0	1100 0	1–00 0
1100 0		0010 0	0000 0
0100 0		1010 0	1110 1
0011 0		1110 0	1010 0
1011 0		0001 0	0100 1
1111 0		0111 0	0010 1
		1111 0	

Z powyższego przykładu wynika, że skuteczność dekompozycji w transformacji funkcji boolowskich na sieci komórek FPGA silnie zależy od scenariusza całego procesu dekompozycji, a w szczególności od kolejności wykonywania poszczególnych procedur, a także od parametrów składowej *G* oraz od rozdziału wyjść na komponenty dekompozycji równoległej.

1.5 Zadania

ZADANIE 3.1.

Automat z tablicy 3.39 zrealizować na pamięci ROM o 4 wejściach adresowych. W rozwiązaniu podać tablicę prawdy UMA oraz organizację pamięci ROM.

Tablica 1.39

	v_1	v_2	v_3	v_4	v_5
a	*a*	*b*	*c*	*a*	*c*
b	–	–	*d*	*e*	*a*
c	*c*	*b*	*e*	–	*b*
d	*a*	*c*	*e*	*b*	*b*
e	*b*	–	*e*	*c*	*b*

ZADANIE 3.2.

W tablicy 3.40 dana jest funkcja $f(a,b,c,d,e)$:

$$P_F = (\overline{1,10,17}; \overline{5,7,19}; \overline{6,8,14}; \overline{3,12,16}; \overline{2,13}; \overline{4,15}; \overline{9,18}; \overline{11,20}).$$

Należy obliczyć dekompozycję nierozłączną dla $U = \{d, e\}$. W rozwiązaniu podać tablice funkcji *G* oraz *H*. Kodowanie bloków P_F przyjąć dowolne wg NKB.

Tablica 1.40

de / *a b c*	0 0	0 1	1 1	1 0
0 0 0	1	2	–	3
0 0 1	4	5	6	–
0 1 1	–	7	8	9
0 1 0	10	–	11	12
1 1 0	–	13	14	–
1 1 1	15	–	–	16
1 0 1	17	–	–	18
1 0 0	–	19	20	–

ZADANIE 3.3.

Wiedząc, że dla funkcji z tablicy 3.41 podziały P_1, P_3, P_5 są 3-przydatne, a P_2, P_4 – 4-przydatne, obliczyć najlepszą dekompozycję szeregową tej funkcji

Tablica 1.41

	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3
1	0	0	0	0	0	1	0	1
2	0	0	0	1	0	1	0	0
3	0	0	0	1	1	0	1	1
4	0	0	1	1	1	0	0	0
5	0	1	0	0	0	1	0	0
6	1	0	1	1	1	0	0	1
7	1	1	0	0	0	0	1	1
8	1	1	1	0	0	0	0	0
9	1	1	1	0	1	0	0	1
10	1	1	1	1	1	0	1	0

1.6 Bibliografia

[3.1] Borowik G., Łuba T., Tomaszewicz P.: On the Memory Capacity to Implement Logic Functions, Eds. F. Pichler, R. Moreno-Díaz, A. Quesada-Arencibia, Computer Aided Systems Theory - EUROCAST 2011, vol.6928: Springer-Verlag Berlin Heidelberg 2012, Lecture Notes in Computer Science, pp. 343-350, 2012.

[3.2] Brayton R., Hachtel G.D., McMullen C., Sangiovanni-Vincentelli A.: *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Boston, 1984.

[3.3] Brzozowski J. A., Łuba T.: *Decomposition of Boolean Functions Specified by Cubes.* Journal of Multiple-Valued Logic and Soft Computing, Vol. 9, Old City Publishing, Inc., Philadelphia, pp. 377–417, 2003.

[3.4] Cong J., Yan K.: *Synthesis for FPGAs with embedded memory blocks*. In: Proc. of the 2000 ACM/SIGDA 8th International Symposium on Field Programmable Gate Arrays, pp. 75 – 82, ACM Press NY, Monterey, California 2000.

[3.5] Curtis H.A.: *A New Approach to the Design of Switching Circuits*. D. Van Nostrand Company, 1962.

[3.6] Hartmanis J., Stearns R. E.: *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, Inc., Englewood Cliffs, N. J, 1966.

[3.7] Hassoun S., Sasao T., Brayton R. (ed.): *Logic Synthesis and Verification*. Kluwer Academic Publishers, New York 2002.

[3.8] Hrynkiewicz E., Kania D.: *Metody syntezy dedykowane dla struktur FPGA typu tablicowego*. Kwartalnik Elektroniki i Telekomunikacji, 50, z. 3, pp. 325-342, 2004.

[3.9] Kania D.: *Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL.* Politechnika Śląska. Zeszyty Naukowe. Nr 1619. Gliwice 2004.

[3.10] Kania D.: Układy logiki programowalnej. Podstawy syntezy i sposoby odwzorowania technologicznego. Wydawnictwo Naukowe PWN, Warszawa 2012.

[3.11] Łuba T.: *Multi-level logic synthesis based on decompos*ition. Microprocessors and Microsystems. 18, No 8, pp. 429-437, 1994.

[3.12] Łuba T., Lasocki R., Rybnik J.: *An Implementation of Decomposition Algorithm and its Application in Information Systems Analysis and Logic Synthesis*. In Rough Sets, Fuzzy Sets and Knowledge Discovery, W. Ziarko (Ed.). Workshops in Computing Series. Springer Verlag, pp. 458-465, 1994.

[3.13] Łuba T., Selvaraj H., Nowicka M., Kraśniewski A.: *Balanced multilevel decomposition and its applications in FPGA-based synthesis*, in Saucier G., Mignotte A. (ed.), Logic and Architecture Synthesis, Chapman&Hall, 1995.

[3.14] Łuba T.: *Decomposition of Multiple-Valued Functions*. 25th International Symposium on Multiple-Valued Logic. Bloomington, Indiana, pp. 256-261, 1995.

[3.15] Łuba T, Selvaraj H.: *A General Approach to Boolean Function Decomposition and its Applications in FPGA-based Synthesis*. VLSI Design, Special Issue on Decompositions in VLSI Design, Vol. 3, Nos. 3-4, pp. 289-300, 1995.

[3.16] Łuba T., Moraga C., Yanushkevich S., Opoka M., Shmerko V.: *Evolutionary Multi-Level Network Synthesis in Given Design Style*. In: Proc. IEEE 30th Int. Symp. on Multiple-Valued Logic, pp. 253-258, Portland 2000.

[3.17] Łuba T., Borowik G.: *Synteza logiczna*, Oficyna Wydawnicza PW, Warszawa 2015.

[3.18] De Micheli G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York 1994.

[3.19] Rawski M., Selvaraj H., Falkowski B.J., Łuba T.: Chapter XII: *Significance of Logic Synthesis in FPGA-Based Design of Image and Signal Processing Systems*, pp. 265–283, w B. Verma, M. Blumenstain (Ed.), Pattern Recognition Technologies and Applications: Recent Advanced, 2008.

[3.20] Sasao T.: Index Generation Functions, Logic Synthesis for Pattern Matching, EPFL Workshop on Logic Synthesis & Verification, Dec. 2015

[3.21] Sasao, T.: Memory-Based Logic Synthesis, Springer New York Dordrecht Heidelberg London, 2011.

[3.22] Selvaraj H., Sapiecha P., Łuba T.: *Functional decomposition and its applications in machine learning and neural networks*. International Journal of Computational Intelligence and Applications. World Scientific Publishing Company, Vol. 1, no. 3, pp. 259-271, Imperial College Press, 2001.

[3.23] Stańczyk U., Cyran K., Pochopień B.: *Theory of logic circuits – volume 1 – Fundamental issues*. Publishers of the Silesian University of Technology, Gliwice 2007.

[3.24] Stańczyk U., Cyran K., Pochopień B.: *Theory of logic circuits – volume 2 – Circuit design and analysis*. Publishers of the Silesian University of Technology, Gliwice 2007.

[3.25] Yanushkevich S, Shmerko V.: *Introduction to Logic Design*. CRC Press, 2008.