Przedstawiona wyżej procedura prowadzi do rozwiązań quasi-optymalnych, gdyż — dla uproszczenia — uwzględnia tylko minimalne zbiory liczb, występujących we wszystkich kolumnach albo wierszach (p. 3). Rozwiązanie optymalne uzyskuje się rozpatrując wszystkie takie zbiory i wybierając spośród nich minimalny zestaw, obejmujący wszystkie elementy F^1 albo F^0.

3.2. UKŁADY Z ELEMENTÓW STYKOWYCH ALBO Z ELEMENTÓW I, LUB, NIE

3.2.1. FAKTORYZACJA

Przy porównaniu urządzeń o zbliżonych parametrach technicznych, bierze się głównie pod uwagę dwa czynniki: niezawodność i koszt. Dobry projekt powinien być rozwiązany według tej wersji, która zabezpiecza kompromis między kosztami a niezawodnością. W typowych układach przełączających, w których niezawodność nie jest zwiększana przez dublowanie sprzętu, zwykle optymalną w rozważanym sensie wersję uzyskuje się przez minimalizację liczby elementów i ich wejść, a więc powstaje problem *minimalizacji układu.*

W realizacjach stykowych minimalnym będzie układ o najmniejszej liczbie zestyków, a zatem opisany funkcją o najmniejszej liczbie liter.

W realizacjach bezstykowych można wyodrębnić dwa przypadki:

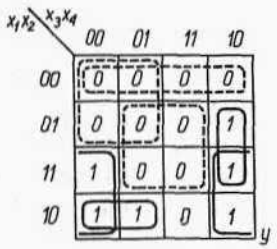
— gdy układ buduje się z elementów dyskretnych (diod, tranzystorów itp.) układ minimalny zawiera najmniejszą liczbę tych elementów (dla ułatwienia obliczeń wprowadza się niekiedy jednostkę porównawczą: diodę — $1D$, a wówczas tranzystor to np. 3 diody: $1T = 3D$ itd.);

— gdy układ buduje się z gotowych bloków, układ minimalny zawiera najmniejszą możliwą liczbę bloków, a przy równej ich liczbie — wybiera się układ mniej obciążający sygnały wejściowe lub wyjściowe.

Opisane wyżej metody minimalizacji funkcji umożliwiają uzyskanie funkcji o minimalnej liczbie liter, ale w postaci normalnej sumy lub postaci normalnej iloczynu. Tak określone funkcje są wygodną postacią dla dalszych modyfikacji, mających na celu uzyskanie postaci, opisujące układ minimalny. Sposób dalszego przekształcania wyrażeń zależy w dużym stopniu od rodzaju stosowanych elementów i opiera się głównie na wyłączaniu przed nawias części wspólnych i stosowaniu praw rozdziel-

ności i de Morgana. Czynności te są niekiedy nazywane *faktoryzacją*, a ich przydatność wyjaśni przykład. Na rys. 3-9 przedstawiono funkcję y. Różne jej przekształcanie pozwala uzyskać:

1) $y = x_1\bar{x}_4 + x_1\bar{x}_2\bar{x}_3 + x_2x_3\bar{x}_4$	8S	11D,3T	7	(4)E
2) $y = x_1(\bar{x}_4 + \bar{x}_2\bar{x}_3) + x_2x_3\bar{x}_4$	7S	11D,3T	8	(5)E
3) $y = x_1\overline{(\bar{x}_4 + x_2 + x_3)} + x_2x_3\bar{x}_4$	—	11D,2T	7	(6)E
4) $y = x_1x_4\overline{(x_2 + x_3)} + x_2x_3\bar{x}_4$	—	11D,2T	7	(6)E
lub				
5) $y = (x_1 + x_2)(x_1 + x_3)(\bar{x}_2 + \bar{x}_4)(\bar{x}_3 + \bar{x}_4)$	8S	12D,3T	8	(5)E
6) $y = (x_1 + x_2x_3)(\bar{x}_4 + \bar{x}_2\bar{x}_3)$	6S	10D,3T	8	(5)E
7) $y = (x_1 + x_2x_3)\overline{(\bar{x}_4 + x_2 + x_3)}$	—	10D,2T	7	(6)E
8) $y = (x_1 + x_2x_3)\overline{x_4(x_2 + x_3)}$	—	10D,1T	6	(6)E



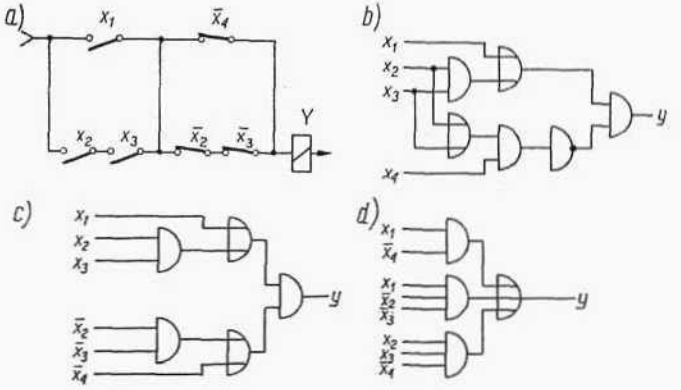
Rys. 3-9. Tablica funkcji faktoryzowanej

Koszt realizacji stykowych poszczególnych postaci liczy się łatwo, gdyż liczba zestyków odpowiada liczbie liter w formule. Negacje, obejmujące więcej niż jeden argument, nie są praktycznie realizowane, więc minimalny układ stykowy otrzyma się z postaci 6 (6 zestyków — 6S).

Przy ocenianiu liczby elementów dyskretnych (dla realizacji diodowych I oraz LUB oraz tranzystorowego NIE) funktory iloczynu i sumy dwu i więcej zmiennych określa się taką liczbą diod, ile argumentów ma funkcja, a funktor negacji — jednym tranzystorem. Jak widać, najlepsze jest rozwiązanie wg postaci 8. Gdyby jednak negacje argumentów były do dyspozycji, lepsza byłaby postać 6.

Dla układów, budowanych z gotowych bloków, minimalny układ jest opisany zależnością 8, a przy danych negacjach (liczby w nawiasie) — zależnością 1. Odpowiednie układy minimalne są pokazane na rys. 3-10.

Znając tylko postaci normalne minimalne trudno jest określić przekształcenia prowadzące do minimalnego układu, ale dalsze poszukiwania można skrócić, korzystając z następujących zasad:



Rys. 3-10. Układy minimalne z najmniejszą liczbą: a) zestyków; b) elementów dyskretnych i bloków (przy braku negacji); c) elementów (negacje dostępne); d) bloków (negacje dostępne)

1. W przypadku układu stykowego przekształca się postać normalną minimalną sumy przez wyciągnięcie przed nawias wszystkich możliwych członów wspólnych, a postać normalną minimalną iloczynu — przez stosowanie zasad rozdzielności wszędzie, gdzie to jest możliwe.

2. Przy budowaniu układu bezstykowego z elementów dyskretnych lub bloków, i danych negacjach argumentów, wystarczają zwykle przekształcenia jak w p. 1; jeśli negacje nie są dane, należy spróbować zastosować jeszcze prawa de Morgana w celu zmniejszenia liczby negatorów.

Stosowanie faktoryzacji w przypadku elementów I, LUB, NIE jest często ograniczone właściwościami elementów, które mogą np. dopuszczać jedynie struktury dwustopniowe, typu suma iloczynów. Pozostaje wtedy tylko możliwość stosowania praw de Morgana, gdyż wprowadzanie negacji jako dodatkowych stopni układu jest zwykle dopuszczalne.

3.2.2. UKŁADY WIELOWYJŚCIOWE

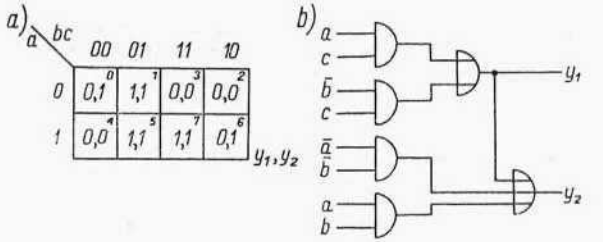
W praktyce stosunkowo rzadko występują układy kombinacyjne o jednym wyjściu, opisane jedną funkcją; zazwyczaj układy mają kilka wyjść, przy czym stan sygnałów na tych wyjściach zależy od tych samych sygnałów wejściowych. Wprawdzie każdy układ wielowyjściowy można uważać za połączenie odpowiednich układów jednowyjściowych, projektowanych niezależnie, ale takie rozwiązanie zwykle nie jest najlepsze, gdyż nie uwzględnia możliwości wielokrotnego wykorzystania niektórych elementów. Na przykład dwie funkcje w postaci minimalnej:

$$y_1 = \bar{x}_1 x_3 + x_1 \bar{x}_2$$
$$y_2 = \bar{x}_1 \bar{x}_2 x_3 + x_1 \bar{x}_2$$

wymagają zrealizowania czterech różnych iloczynów. Jeśli pierwszą z tych funkcji przekształci się w postać

$$y_1 = \bar{x}_1 \bar{x}_2 x_3 + x_1 \bar{x}_2$$

to, rezygnując z postaci minimalnej, ale nie zmieniając wartości funkcji (łatwo to sprawdzić np. w tablicy Karnaugha), upraszcza się realizację, gdyż pierwszy składnik jest wspólny dla obu funkcji.



Rys. 3-11. Układ dwuwyjściowy: a) tablica; b) schemat

Wydzielanie wspólnych członów w kilku wyrażeniach funkcyjnych jest łatwiejsze, gdy funkcje zadane są w postaci kanonicznej, w zapisie dziesiętnym. Można wyróżnić trzy szczególne przypadki relacji między dwiema funkcjami, wpływające na tok dalszego postępowania.

1. Jeśli funkcje y_1 i y_2 zadane są za pomocą zbiorów F_1^1 i F_2^1 (lub F_1^0 i F_2^0) i jeden z nich jest zawarty w drugim (np. $F_1^1 \subseteq F_2^1$, czyli $F_2^0 \subseteq F_1^0$), to jedną z tych funkcji można wyrazić za pomocą drugiej oraz członu uzupełniającego.

Na przykład funkcje z tablicy na rys. 3-11a można zapisać w postaci:

$$y_1 = \sum(1,5,7) \qquad y_2 = \sum(0,1,5,6,7)$$

czyli

$$F_1^1 = \{1,5,7\} \qquad F_2^1 = \{0,1,5,6,7\}$$

lub

$$F_1^0 = \{0,2,3,4,6\} \qquad F_2^0 = \{2,3,4\}$$

Wynika stąd, że $F_1^1 \subseteq F_2^1$ oraz $F_2^0 \subseteq F_0^1$, a więc można napisać

$$y_2 = \sum(1,5,7) + \sum(0,6) = y_1 + \sum(0,6)$$

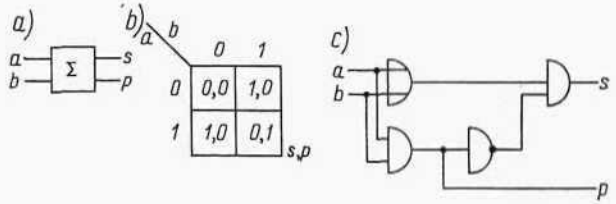
albo

$$y_1 = \prod(2,3,4) \cdot \prod(0,6) = y_2 \cdot \prod(0,6)$$

W pierwszej zależności trzeba wyrazić y_2 za pomocą y_1, więc:

$$y_1 = ac + \bar{b}c$$

$$y_2 = y_1 + \bar{a}\bar{b} + ab$$



Rys. 3-12. Półsumator: a) oznaczenie; b) tablica; c) schemat

Przy określaniu wyrażenia $\sum(0,6)$ odpowiadające tym kratkom jedynki można oczywiście sklejać z innymi jedynkami funkcji y_2. Otrzymany układ przedstawiono na rys. 3-11b.

Druga wersja miałaby postać:

$$y_2 = (a + \bar{b})(\bar{a} + b + c)$$

$$y_1 = y_2 \cdot (b + c)(\bar{b} + c)$$

Tu również, określając wyrażenie $\prod(0,6)$, można odpowiednie zera y_1 sklejać z innymi zerami, już objętymi funkcją y_2.

2. Jeśli zbiory F^1 obydwu rozważanych funkcji są rozłączne, to znaczy, że zbiór F^1 jednej z nich jest zawarty w F^0 drugiej:

$$F_1^1 \cap F_2^1 = \emptyset \quad \text{więc} \quad F_1^1 \subseteq F_2^0 \quad \text{i} \quad F_2^1 \subseteq F_1^0$$

wtedy jedną z tych funkcji można wyrazić za pomocą negacji drugiej funkcji oraz członu uzupełniającego.

Na przykład rys. 3-12a przedstawia symbol, a rys. 3-12b — tablicę tzw. *półsumatora*, czyli układu realizującego arytmetyczną sumę dwóch cyfr binarnych i przeniesienie do starszej pozycji, powstające przy dodawaniu. Z tablicy otrzymuje się:

$$s = \bar{a}b + a\bar{b} = \sum(1,2) = \prod(0,3)$$
$$p = ab \quad = \sum(3) \quad = \prod(0,1,2)$$

czyli

$$\bar{s} = \sum(0,3) \quad = \prod(1,2)$$
$$\bar{p} = \sum(0,1,2) = \prod(3)$$

wobec tego

$$s = \prod(3) \cdot \prod(0) = \bar{p} \cdot \prod(0)$$

albo

$$p = \prod(1,2) \cdot \prod(0) = \bar{s} \cdot \prod(0)$$

Ponieważ realizacja p jest prostsza, lepiej będzie wyrazić s za pomocą $\bar{p}$:

$$p = ab$$
$$s = \bar{p} \cdot (a+b)$$

Odpowiedni układ (rys. 3-12c) nie wymaga zanegowanych argumentów.

Podobnie postępuje się w przypadku, gdy $F_1^0 \cap F_2^0 = \emptyset$, a więc gdy $F_1^0 \subseteq F_2^1$ i $F_2^0 \subseteq F_1^1$. Jedną z funkcji można wówczas przedstawić jako sumę negacji drugiej funkcji oraz członu uzupełniającego.

3. Gdy nie zachodzi żaden z opisanych wyżej przypadków, istnieją dwie możliwości:

— sprowadzić zadanie (sztucznie) do któregoś z opisanych przypadków,

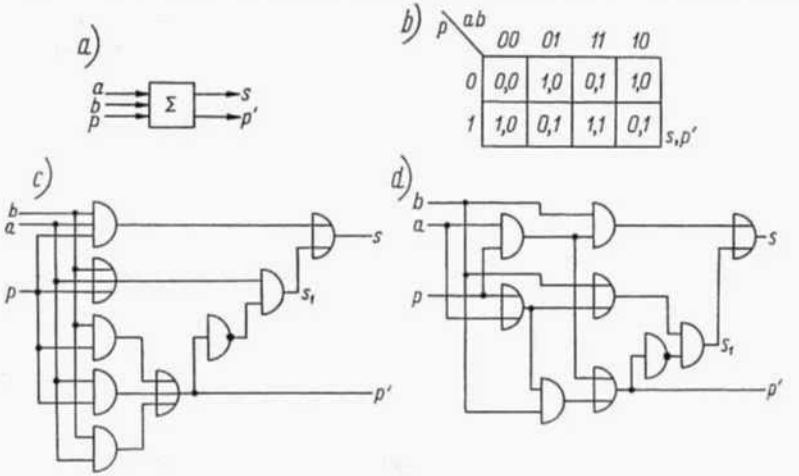
— zrezygnować z powtórnego wykorzystania całych funkcji i badać mniejsze wspólne człony.

Przykładem celowości pierwszego postępowania może być synteza *sumatora jednobitowego*, o tablicy z rys. 3-13b, realizującego sumę arytme-

tyczną trzech cyfr i przeniesienie do pozycji bardziej znaczącej. Z tablicy łatwo otrzymuje się:

$$s = \bar{a}b\bar{p} + ab p + a\bar{b}\bar{p} + \bar{a}\bar{b}p = \sum(1,2,4,7) = \prod(0,3,5,6)$$
$$p' = ab + ap + bp \qquad\qquad = \sum(3,5,6,7) = \prod(0,1,2,4)$$

Funkcja s jest złożona, więc celowe jest poszukiwanie metody wyznaczenia s za pomocą p'. Wprawdzie nie zachodzi tu żaden z opisanych



Rys. 3-13. Sumator jednobitowy: a) oznaczenie; b) tablica; c) schemat po faktoryzacji

wyżej przypadków, ale gdyby z wyrażeń dla s usunąć $\sum(7)$ albo $\prod(0)$ byłby typowy przypadek 2. Można więc napisać:

$$s = \sum(1,2,4) + \sum(7) = s_1 + \sum(7) = s_1 + abp$$
$$s_1 = \sum(1,2,4) = \prod(0,3,5,6,7) = \bar{p}' \cdot \prod(0) = \bar{p}' \cdot (a+b+p)$$

Wyrażenia $\sum(7)$ i $\prod(0)$ realizuje się bez użycia negacji, więc cały układ (rys. 3-13c) nie wymaga negowania argumentów. Wprowadzając dodatkową faktoryzację można przekształcić ten schemat w postać z rys. 3-13d.

To samo zadanie można wykonać również korzystając z drugiej postaci kanonicznej s:

$$s = \prod(3,5,6) \cdot \prod(0) = s_2 \cdot \prod(0) = s_2 \cdot (a+b+c)$$
$$s_2 = \prod(3,5,6) = \sum(0,1,2,4,7) = \bar{p}' + \sum(7) = \bar{p}' + abc$$

Tego rodzaju postępowanie jest celowe wówczas, gdy dwie rozważane funkcje mają niewiele wspólnych członów postaci kanonicznych, a w dodatku człony te realizuje się dość prosto. We wszystkich pozostałych przypadkach poszukiwanie wspólnych członów funkcji rozpoczyna się od wyznaczenia wspólnych składników postaci kanonicznych, a dalszą minimalizację, uwzględniającą człony wspólne, wykonuje się znanymi metodami. Najdogodniej jest posługiwać się odpowiednio zmodyfikowaną metodą Quine'a-Mc Cluskeya o następujących czynnościach:

1. Określa się indeksy liczb opisujących postaci kanoniczne wszystkich funkcji i wypisuje kolumnę z podziałem na grupy indeksowe. Każdej liczbie w kolumnie przypisuje się dodatkowy symbol funkcji, z której dana liczba pochodzi (np. litery *a*, *b*, *c*, ...).

2. Następne kolumny tworzy się z poprzednich w wyniku sklejania, przy którym — oprócz zasad opisanych poprzednio — obowiązuje jeszcze warunek, aby sklejane wyrażenia miały chociaż jeden wspólny symbol funkcji. Przy wyniku sklejania umieszcza się właśnie te symbole, a wyrażenie uważa się za objęte następną kolumną i odpowiednio to zaznacza tylko wówczas, gdy wszystkie jego symbole są umieszczone przy wyniku sklejania, np. 0 *bc* i 4 *abc* dają 0,4 (4) *bc*, ale tylko człon 0 *bc* jest pochłonięty wynikiem sklejenia.

3. Wyrażenia, których nie udało się skleić całkowicie odpowiadają implikantom (implicentom) funkcji i ich wspólnych członów. Tworzy się z nich tablicę, wyznacza implikanty (implicenty) zasadnicze, a następnie wybiera postacie minimalne, preferując te człony, które mają więcej symboli funkcji, tzn. które są wspólne dla większej liczby funkcji.

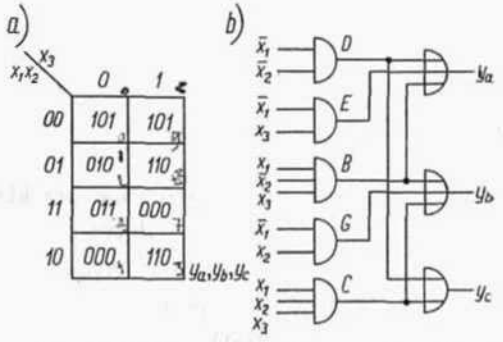
4. Zamiana wyrażeń na postać binarną i literową jest taka jak poprzednio.

Przykładem niech będą funkcje, zadane tablicą z rys. 3-14a, z której otrzymuje się:

$$y_a = \bar{x}_1\bar{x}_2 + \bar{x}_1x_3 + \bar{x}_2x_3 = (\bar{x}_1 + \bar{x}_2)(\bar{x}_2 + x_3)(\bar{x}_1 + x_3) =$$
$$= \sum(0,1,3,5) = \prod(2,4,6,7)$$
$$y_b = x_2\bar{x}_3 + \bar{x}_1x_2 + x_1\bar{x}_2x_3 = (x_1 + x_2)(x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3) =$$
$$= \sum(2,3,5,6) = \prod(0,1,4,7)$$
$$y_c = \bar{x}_1\bar{x}_2 + x_1x_2\bar{x}_3 = (x_1 + \bar{x}_2)(\bar{x}_1 + x_2)(\bar{x}_1 + \bar{x}_3) =$$
$$= \sum(0,1,6) = \prod(2,3,4,5,7)$$

Postacie minimalne mają tylko jeden wspólny człon ($\bar{x}_1\bar{x}_2$), układ będzie więc dość złożony. Stosując minimalizację kanonicznych postaci sumy uzyskuje się:

0	*ac*	V	0,1 (1) *ac*	*D*
1	*ac*	V	1,3 (2) *a*	*E*
2	*b*	V	1,5 (4) *a*	*F*
3	*ab*	*A*	2,3 (1) *b*	*G*
5	*ab*	*B*	2,6 (4) *b*	*H*
6	*bc*	*C*		



Rys. 3-14. Przykład układu o trzech wyjściach

Tablica 3-7

Tablica implikantów układu wielowyjściowego

	y_a				y_b				y_c		
	0	1	3	5	2	3	5	6	0	1	6
A 3 *ab*			×			×					
B 5 *ab**				×			×				
C 6 *bc**								×			×
D 0,1(1)*ac**	×	×							×	×	
E 1,3(2)*a*		×	×								
F 1,5(4)*b*		×		×							
G 2,3(1)*b*					×	×					
H 2,6(4)*b*					×			×			
	•	•		•			•	•	•	•	•

Wyrażenia nieredukowalne (implikanty) dla uproszczenia zapisu oznaczono literami. Tablica implikantów (tabl. 3-7) jest wypełniona z uwzględnieniem ich przynależności do konkretnych funkcji, a więc implikant z symbolem a ma krzyżyki tylko w kolumnach funkcji y_a itd. Pojedyncze krzyżyki w kolumnach wyznaczają zasadnicze proste implikanty B, C, D. Nie pochłaniają one wszystkich składników funkcji, więc trzeba wyznaczyć implikanty dodatkowe. Wspólny dla dwu funkcji jest implikant A, ale nie wystarcza on całkowicie, więc lepiej będzie przyjąć prostsze (bo dwuskładnikowe) implikanty E i G. W rezultacie otrzymuje się:

$$y_a = D+E+B = \bar{x}_1\bar{x}_2+\bar{x}_1x_3+x_1\bar{x}_2x_3$$
$$y_b = G+C+B = \bar{x}_1x_2+x_1x_2\bar{x}_3+x_1\bar{x}_2x_3$$
$$y_c = D+C \quad = \bar{x}_1\bar{x}_2+x_1x_2\bar{x}_3$$

i schemat jak na rys. 3-14b. Podobnie można przekształcić postacie kanoniczne iloczynu.

Rozważane wyżej przypadki były ilustrowane przykładami funkcji całkowicie określonych, lecz — oczywiście — dotyczą one także funkcji niepełnych. Ogólne zasady postępowania nie ulegają zmianie i tylko ostatni przedstawiony tu algorytm postępowania trzeba dostosować do istnienia pozycji nieokreślonych. Pozycje te wypisuje się w pierwszej kolumnie, tak jak pozostałe, ale symbol funkcji, gdzie ta pozycja (składnik postaci kanonicznej) ma wartości nieokreśloną, umieszcza się w nawiasie. Przy sklejaniu obowiązują zasady jak poprzednio, z tym że sklejenie wyrażenia o symbolu w nawiasie z wyrażeniem o symbolu bez nawiasu daje wynik z symbolem bez nawiasu, np.

$$2\ b(c) \quad \text{i} \quad 3\ a(bc) \quad \text{daje} \quad 2{,}3(1)\ b(c)$$

Po zakończeniu sklejania symboli w nawiasach nie bierze się pod uwagę.

Na przykład układ opisany funkcjami czterech zmiennych:

$$y_a = \sum[1{,}3{,}9\ (5{,}6{,}7{,}11{,}12{,}13{,}14{,}15)]$$
$$y_b = \sum[1{,}6\ \ (3{,}5{,}7{,}12{,}13{,}14{,}15)]$$
$$y_c = \sum[6{,}9{,}11(3{,}5{,}12{,}13{,}14{,}15)]$$

minimalizuje się w następujący sposób:

1 *ab* V	1, 3(2) *ab* V	1, 3, 5, 7(2,4) *ab* *C*
3 *a*(*bc*) V	1, 5(4) *ab* V	
5 (*abc*) V	1, 9(8) *a* V	1, 3, 9,11(2,8) *a* V
6 (*a*)*bc* V	3, 7(4) *a*(*b*) V	1, 5, 9,13(4,8) *a* V
9 *ac* V	3,11(8) *ac* *A*	
12 (*abc*) V	5, 7(2) (*ab*) V	3, 7,11,15(4,8) *a* V
7 (*ab*) V	5,13(8) (*abc*) V	
11 (*a*)*c* V	6, 7(1) (*a*)*b* V	5, 7,13,15(2,8) (*ab*)
13 (*abc*) V	6,14(8) (*a*)*bc* *B*	
14 (*abc*) V	9,11(2) *ac* V	6, 7,14,15(1,8) (*a*)*b* *D*
15 (*abc*) V	9,13(4) *ac* V	
	12,13(1) (*abc*) V	9, 11,13,15(2,4) *ac* *E*
	12,14(2) (*abc*) V	
		12,13,14,15(1,2) (*abc*)
	7,15(8) (*ab*) V	
	11,15(4) (*a*)*c* V	
	13,15(2) (*abc*) V	
	14,15(1) (*abc*) V	

1,3,5,7,9,11,13,15(2,4,8) *aF*

Tablica 3-8

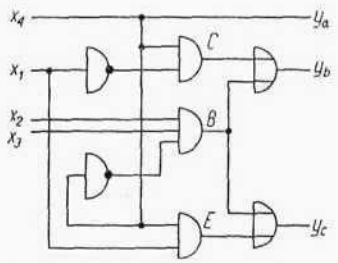
Tablica układu wielowyjściowego

	y_a			y_b		y_c		
	1	3	9	1	6	6	9	11
A		×						×
B •					×	×		
C •	×	×		×				
D					×			
E •			×				×	×
F	×	×	×					
	*	*	*	*	*	*	*	*

Z tablicy implikantów (tabl. 3-8) wynika, że zasadnicze proste implikanty B, C, E pochłaniają wszystkie składniki funkcji, więc:

$$y_a = C+E = \bar{x}_1 x_4 + x_1 x_4$$
$$y_b = B+C = x_2 x_3 \bar{x}_4 + \bar{x}_1 x_4$$
$$y_c = B+E = x_2 x_3 \bar{x}_4 + x_1 x_4$$

Jednakże y_a można wyrazić znacznie prościej przez $y_a = x_4$ (implikant F), więc nie ma potrzeby korzystania z implikantów C i E. Schemat układu przedstawiono na rys. 3-15. Przykład ten ilustruje dość często spotykany przypadek, gdy realizacja minimalnej postaci indywidualnej funkcji daje lepsze rezultaty niż realizacja członów wspólnych. Należy

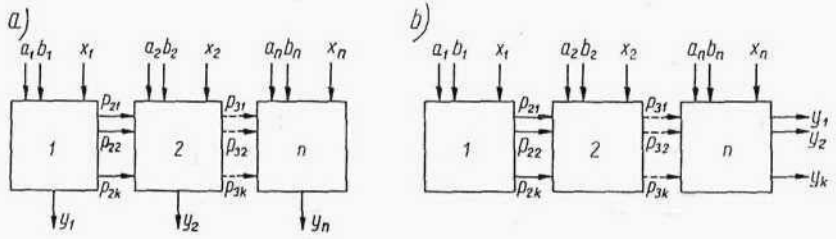


Rys. 3-15. Przykład układu o trzech wyjściach

więc zawsze porównywać wyniki syntezy zbiorowej i syntezy indywidualnej układów, wybierając rozwiązanie prostsze. Opisany algorytm zadanie to bardzo ułatwia.

3.2.3. UKŁADY ITERACYJNE

Projektowanie i budowę układu o wielu wejściach lub wyjściach można często znacznie uprościć przez podział układu na jednakowe lub podobne do siebie człony (segmenty). Tak budowane układy noszą nazwę



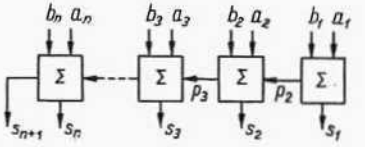
Rys. 3-16. Typowe struktury układów iteracyjnych

iteracyjnych (*kaskadowych*) i zwykle mają strukturę jak na rys. 3-16. Cechą charakterystyczną tych układów jest to, że przetwarzają one ciągi sygnałów binarnych (np. liczby): $A = (a_1, a_2, \ldots, a_n)$, $B =$

$= (b_1, b_2, \ldots, b_n)$ itd, a działania na i-tych sygnałach są określone jednoznacznie za pomocą pomocniczych sygnałów z $i-1$-go segmentu — sygnałów przeniesienia. Wyjścia układu mogą pochodzić z każdego członu (rys. 3-16a) lub tylko z ostatniego (rys. 3-16b) — i wówczas są to zwykle sygnały przeniesienia z n-tego członu. W szczególnym przypadku układ może mieć tylko jedno wyjście z n-tego członu.

Budowanie układu o postaci iteracyjnej umożliwia ujednolicenie sprzętu, zmniejsza liczbę wejść elementów (a często i liczbę elementów), pozwala na proste rozszerzenie liczby wejść układu, ale — niestety — nie wszystkie układy mogą być tak dzielone.

Typowym układem iteracyjnym jest sumator równoległy, budowany z opisanych wyżej sumatorów jednobitowych (rys. 3-17). Sygnał przenie-



Rys. 3-17. Sumator równoległy

sienia wynika tu w naturalny sposób z algorytmu dodawania. Ponieważ $p_1 = 0$, więc pierwszy człon układu jest prostszy (jest półsumatorem). Dotyczy to większości układów iteracyjnych.

W przypadkach mniej oczywistych należy przede wszystkim ustalić:

— kierunek przeniesień,

— liczbę sygnałów przeniesienia,

— sposób zakodowania tych sygnałów, tzn. przypisania im wartości 0 lub 1.

Po tych ustaleniach pozostaje zaprojektować i-ty człon, którego wejściami są odpowiednie wejścia układu i przeniesienia z poprzedniego członu, a wyjściami — przeniesienia do następnego członu i, ewentualnie, odpowiednie wyjście układu. Zasady postępowania wyjaśnią przykłady.

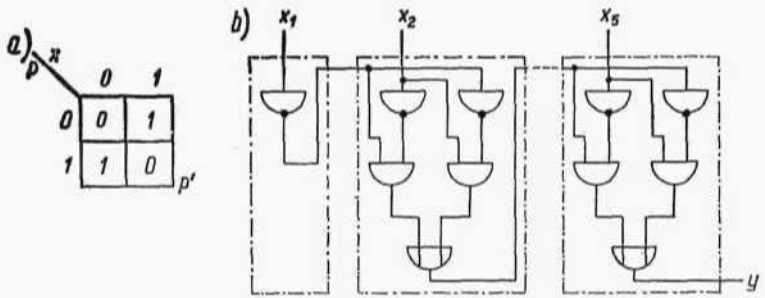
Układ o pięciu wejściach, który daje sygnał wyjściowy $y = 1$ tylko wówczas, gdy na parzystej liczbie wejść (lub żadnym) są sygnały $x = 1$, można łatwo zaprojektować za pomocą tablicy Karnaugha pięciu zmiennych, ale postać kanoniczna nie może być uproszczona i układ jest rozbudowany. Zadanie to może być rozwiązane iteracyjnie, gdyż w każdym

członie o jednym wejściu x można określić przeniesienie p informujące o parzystości sygnałów na poprzednich wejściach, a przeniesienie ostatniego członu będzie wyjściem układu. Kierunek przeniesień jest obojętny.

Przyjmując $p_i = 1$, gdy wśród $x_1, x_2, \ldots, x_{i-1}$ jest parzysta liczba jedynek, i $p_i = 0$ w przypadku przeciwnym, można zestawić tabelę (rys. 3-18a), określającą wartość p_{i+1} na podstawie p_i i x_i (dla uproszczenia — p', p i x). Z tablicy wynika, że

$$p' = x\bar{p} + \bar{x}p = (x+p)(\bar{x}+\bar{p})$$

Ponieważ $p_1 = 1$, więc $p_2 = x_1 \cdot 0 + \bar{x}_1 \cdot 1 = \bar{x}_1$ i układ ma postać jak na rys. 3-18b. Liczba wejść takiego układu może być zwiększona



Rys. 3-18. Jednowyjściowy układ iteracyjny

prawie dowolnie, ale — jeśli elementy są bierne — należy zapewnić regenerację sygnałów przeniesienia, które przechodzą przez dużą liczbę elementów.

Przykład bardziej złożony — to układ o n wejściach x i n wyjściach y, działający w ten sposób, że $y_i = 1$ tylko wówczas, gdy na wejściach $x_1, x_2, \ldots, x_i$ jest nie więcej niż jedna jedynka ($i = 1, 2, \ldots, n$).

W tym przypadku kierunek przeniesień jest określony zadaniem, a same przeniesienie do członu i powinno zawierać następujące informacje:

1) żaden z sygnałów $x_1, x_2, \ldots, x_{i-1}$ nie ma wartości 1,

2) jeden z tych sygnałów ma wartość 1,

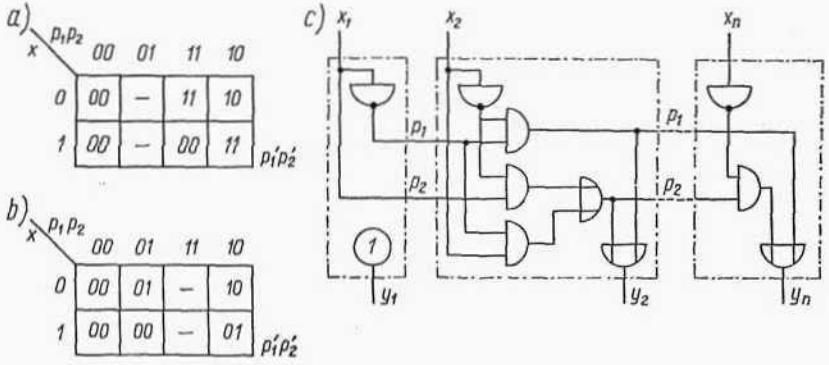
3) więcej niż jeden z tych sygnałów ma wartość 1.

Te trzy stany wymagają zastosowania dwóch bitów przeniesienia, (p_1, p_2),

a sposób w jaki poszczególnym stanom zostaną przypisane wartości bitów przeniesienia ma istotne znaczenie, gdyż wpływa na złożność realizacji. Stan wyjścia y ma wyróżniać pierwszy i drugi stan przeniesienia, istnieją więc dwie podstawowe możliwości:

— przyjąć dla stanu przeniesień *1* i *2* $p_1 = 1$, a dla stanu *3* — $p_1 = 0$, natomiast p_2 dowolnie (byle różniły się stany *1* i *2*); wówczas $y = p_1$,

— przyjąć dla stanów *1* i *2* taki kod, aby y mogło być prostą funkcją przeniesień (np. sumą).



Rys. 3-19. Wielowyjściowy układ iteracyjny

W pierwszym przypadku, przypisując stanom przeniesień wartości binarne p_1p_2 w sposób następujący:

$$1 — (10), \quad 2 — (11), \quad 3 — (00)$$

działanie i-tego segmentu można opisać tablicą z rys. 3-19a, skąd

$$p_1' = p_1\bar{p}_2 + \bar{x}p_1$$
$$p_2' = xp_1\bar{p}_2 + \bar{x}p_2 \qquad y = p_1'$$

W drugim przypadku, kodując

$$1 — (10), \quad 2 — (01), \quad 3 — (00)$$

otrzymuje się tablicę jak na rys. 3-19b, oraz

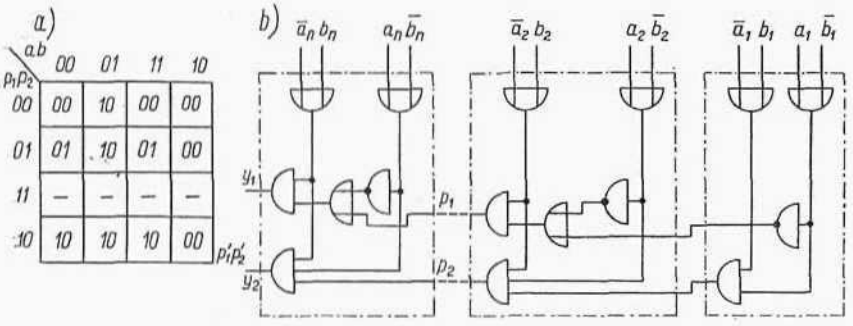
$$p_1' = \bar{x}p_1$$
$$p_2' = \bar{x}p_2 + xp_1 \qquad y = p_1' + p_2'$$

W pierwszym rozwiązaniu p_1' można wyrazić za pomocą p_2', ale mimo to rozwiązanie drugie jest prostsze. Przyjmując je do realizacji można zauważyć, że pierwszy człon otrzymuje stałe sygnały $p_1 = 1$, $p_2 = 0$ („brak sygnałów x o wartości 1"), więc dla niego $p_1' = \bar{x}$, $p_2' = x$. Ostatni człon nie musi generować sygnałów przeniesienia, więc można go uprościć, analizując warunki określające $y_n = 1$. Otrzymuje się $y_n = p_1 + p_2\bar{x}$, a schemat układu ma postać przedstawioną na rys. 3-19c.

Ważnym dla zastosowań praktycznych układem jest tzw. *komparator* — układ porównujący dwie liczby binarne. Najbardziej pełna odmiana komparatorów wyróżnia trzy możliwe relacje między liczbami wejściowymi A i B:

$$1.\ A > B, \quad 2.\ A = B, \quad 3.\ A < B.$$

Sygnalizowanie tych trzech stanów wymaga wyjścia o dwóch bitach, a łatwo zauważyć, że w rozwiązaniu iteracyjnym również przeniesienie musi zawierać podobny rodzaj informacji (dla odcinków ciągów liczbowych). Cechą charakterystyczną komparatora jest to, że przeniesienia



Rys. 3-20. Komparator równoległy

w układzie iteracyjnym mogą być przekazywane zarówno od pozycji bardziej znaczących do mniej znaczących jak i odwrotnie. Stosunkowo prostą realizację uzyskuje się, wybierając przeniesienie od mniej znaczących i kod dla p_1p_2 i y_1y_2:

$$1 — (00), \quad 2 — (01), \quad 3 — (10)$$

Typowy segment komparatora można wówczas opisać tablicą z rys. 3-20a, z której otrzymuje się:

$$p_1' = \bar{a}b + \bar{a}p_1 + bp_1 = (\bar{a}+b)(\bar{a}+p_1)(b+p_1)$$
$$p_2' = abp_2 + \bar{a}\bar{b}p_2 = p_2(a+\bar{b})(\bar{a}+b)$$

Poszukując wspólnych członów obu funkcji można napisać

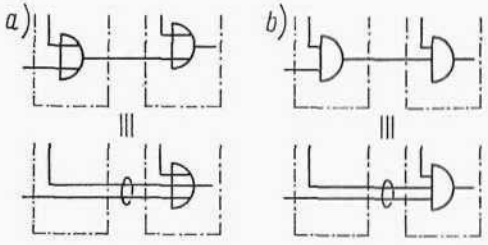
$$p_1' = (\bar{a}+b)(p_1+\bar{a}b) = (\bar{a}+b)(p_1+\overline{a+\bar{b}})$$

Schemat układu przedstawiony jest na rys. 3-20b. Dla pierwszego segmentu jest $p_1 = 0$. $p_2 = 1$ („poprzednie odcinki równe"), więc

$$p_1' = \bar{a}b = \overline{a+\bar{b}}$$
$$p_2' = ab + \bar{a}\bar{b} = (a+\bar{b})(\bar{a}+b)$$

Przeniesienia z ostatniego segmentu są wyjściami układu. Można zauważyć, że y_2 jest iloczynem członów typu $(a+\bar{b})(\bar{a}+b)$, więc realizację iteracyjną tego sygnału można zastąpić działaniem równoległym, a wówczas — przy np. $n = 5$ — zamiast pięciu elementów I 3-wejściowych potrzebny będzie jeden element 10-wejściowy.

Takie rozwiązanie dosyć często znajduje zastosowanie w układach i — ogólnie — polega na zastąpieniu jednego przewodu wiązką innych. Przykłady tworzenia wiązek przedstawia rys. 3-21.



Rys. 3-21. Przykłady stosowania wiązek

3.3. UKŁADY Z ELEMENTÓW NOR ALBO NAND

3.3.1. TRANSFORMACJA UKŁADÓW

Podane wyżej zasady budowania układów minimalnych z elementów I, LUB, NIE mają bardzo ograniczony zakres użyteczności w praktyce, gdyż prawie wszystkie systemy elementów logicznych wykorzystują