Część trzecia

UKŁADY PRZEŁĄCZAJĄCE

13. Funkcje i elementy logiczne

13.1. Określenia podstawowe

Układy przełączające, nazywano również układami cyfrowymi lub logicznymi *), operują sygnałami dyskretnymi, tzn. przyjmującymi tylko określoną liczbę wartości (najczęściej dwie).Sygnały takie występują w maszynach cyfrowych i niektórych urządzeń przeliczających, występują również w wielu procesach technologicznych, których sterowanie można sprowadzić do załączania i wyłączania poszczególnych urządzeń, oczywiście w określonych warunkach i w określonej kolejności. Ale oprócz tego naturalnego pola zastosowań, układy przełączające wykorzystywane są niekiedy w układach automatyzacji procesów ciągłych, przede wszystkim w części pomiarowej i centralnej. Decyduje o tym mała wrażliwość sygnałów dyskretnych (zwłaszcza dwustanowych) na zakłocenia i związana z tym możliwość uzyskania dużych dokładności pomiaru i przetwarzania informacji. Istotne są również takie cechy jak duża niezawodność działania i łatwość przesyłania sygnałów na znaczne odległości.

W każdym układzie przełączającym można wyróżnić trzy zasadnicze części:

1) urządzenia wejściowe (uzyskiwanie, przetwarzanie i ewentualnie utrwalanie danych),

2) część centralna, tzn. układ logiczny,

3) urządzenia wyjściowe (wzmacnianie i przetwarzanie sygnałów wykonawczych).

Niniejszy rozdział dotyczy części centralnej układów przełączających. Podane zostaną w nim podstawy matematyczne opisu układów logicznych oraz przykłady rozwiązań konstrukcyjnych elementów logicznych.

---

*) W niniejszej książce układem logicznym nazywać będziemy część centralną układu przełączającego.

Najczęściej stosowane są elementy logiczne dwustanowe, tzn. mające dwa stany stabilne, w których sygnały wejściowe i wyjściowe elementów przyjmują wartości oznaczone umownie 0 i 1. Operacjami matematycznymi na zmiennych przyjmujących tylko te wartości zajmują się algebry logiki, zwłaszcza zerojedynkowa (dwuelementowa) algebra Boole'a, a funkcje logiczne realizowane przez te elementy nazywane są funkcjami boolowskimi lub zerojedynkowymi.

Spotyka się również elementy logiczne trójstanowe, o trzech stanach stabilnych i odpowiadających tym stanom wartościach sygnałów oznaczonych 0, ½, 1 (lub –1, 0, +1). Użycie tych elementów może często znacznie uprościć budowę układu, a zwłaszcza zmniejszyć liczbę połączeń, jednak stosowane są rzadko, gdyż ich budowa jest bardziej złożona (stąd wyższy koszt elementu), a ponadto projektowanie układów, opartych na logice trójwartościowej, jest dużo trudniejsze.

W dalszej części książki rozważania zostaną ograniczone do układów zawierających dwustanowe elementy logiczne.

Układy przełączające można podzielić na kombinacyjne i sekwencyjne.

W układach kombinacyjnych wartości sygnałów sterujących poszczegdl-nymi urządzeniami procesu zależą, *tylko od bieżących wartości* sygnałów informu-jących o stanie procesu i sygnałów zewnętrznych. Procesy sterowane nazywa się wówczas jednotaktowymi; jako przykład przytoczyć można proces dekodo-wania sygnałów binarnych na dziesiętne.

W układach sekwencyjnych wartości sygnałów sterujących zależą *nie tylko od bieżących, lecz również od poprzednich wartości* sygnałów informujących o stanie procesu i sygnałów zewnętrznych. Procesy sterowane nazywa się wówczas wielotaktowymi, wymagają, one ściśle określonego programu (kolejności) czynności sterujących, narzuconego przez założony przebieg procesu.

Przykładem procesu wielotaktowego może być proces dozowania cieczy, przedstawiony schematycznie na rys.13.1. Przycisk ręczny *PR* służy do wprowadzania sygnału – „start”, a przekaźniki poziomu *P1* i *P2* sygnalizują osiągnięcie poziomów h_1 i h_2 przez ciecz znajdującą, się w zbiorniku. Wymagany program procesu jest następujący:

a) napełnianie zbiornika przez otwarcie zaworu *Z1* (y_1=1) powinno się rozpocząć, gdy wprowadzony zostanie ręcznie sygnał „start” (x_0 = 1), pod warunkiem, że poziom $h < h_1$ (x_1 = 0);

b) gdy puścimy przycisk ręczny (x_0 = 0), zawór *Z1* powinien być nadal otwarty, jeżeli $h < h_1$ (x_1 = 0);

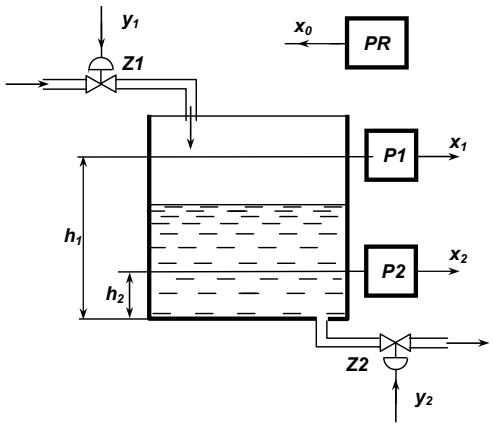
c) zawór *Z1* powinien zostać zamknięty, gdy $h = h_1$ (x_1 = 1);

d) zawór *Z2* powinien zostać otwarty, gdy $h \geq h_1$ (x_1 = 1); wówczas oczywiście $h > h_2$ (x_2 = l);

e) zawór *Z2* powinien pozostawać otwarty, gdy $h < h_1$ (x_1 = 0), jeżeli $h \geq h_1$ (x_2 = 1);

f) zawór *Z2* powinien zostać zamknięty, gdy $h < h_2$ (x_2 = 0).

Program taki zapewnia odmierzenie określonej porcji cieczy (objętości zawartej miedzy h_1 i h_2,) i jednocześnie zabezpiecza przed pomyłkami obsługi, gdyż np. naciśnięcie przycisku startowego *PR* nie spowoduje otwarcia zaworu *Z1,* jeżeli zbiornik jest pełny ($h = h_1, x_1 = 1$).



Rys. 13.1. Schomat procesu dozowania cieczy z urządzeniami wejściowymi i wyjściowymi: *PR* — przycisk ręczny, *P1, P2* — przekaźniki poziomu, *Z1, Z2* — zawory

Dwustanowy charakter działania układów przełączających wpływa zasadniczo na sposób opisywania i projektowania układu. Zwykle pomija się przy tym stany przejściowe występujące w chwilach przełączania elementów i formułuje się jodynie związki pomiędzy sygnałami wyjściowymi i wejściowymi w stanach ustalonych.

13.2. Funkcje logiczne

Dowolny układ logiczny można przedstawić schematycznie według rys. 13.2. Zakładamy, że sygnały występujące w tym układzie przyjmują, w stanach ustalonych tylko dwie wartości, oznaczane symbolami 0 i 1. Warunki pracy takiego układu będą, więc opisywane za pomocą funkcji, które przyjmować mogą tylko wartość 0 lub 1 i zależą od zmiennych (argumentów) mogących również przyjmować tylko wartości 0 lub 1.

Rys. 13.2. Schemat ogólny układu logicznego

Jeżeli założymy, że układ przedstawiony na rys.13.2 jest układem kombinacyjnym, to związki pomiędzy sygnałami wyjściowymi i wyjściowymi można zapisać jak następuje:

$$y_1 = f_1(x_1, x_2, \ldots, x_n),$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n),$$

gdzie $f_1, f_2, \ldots, f_m$, - *funkcje logiczne,* nazywane tez funkcjami boolowskimi, zerojedynkowymi lub przełączającymi, $x_1, x_2, \ldots, x_n$ - zmienne wejściowe (argumenty funkcji).

Zbiór wartości zmiennych wejściowych w danej chwili zapisywany jest w postaci ciągu cyfr dwójkowych 0 i 1

$$x = (x_1, x_2, \ldots, x_n)$$

i nazywany jest *stanem wejść.* Każderau stanowi odpowiada pewna liczba całkowita wyrażona w układzie dwójkowym, np. stan (1100) przedstawia liczbę $0\cdot2^0+0\cdot2^1+1\cdot2^2+1\cdot2^3 = 12$, a stan (101) liczbę $1\cdot2^0+0\cdot2^1+1\cdot2^2 = 5$. Liczbę przedstawioną przez dany stan nazywać będziemy numerem stanu.

Istnieje $k = 2^n$ stanow n zmiennych dwojkowych (n = l, 2, ...), które numeruje się od 0 (tzw. stan zerowy, kiedy wszystkie zmienne są zerami) do $2^n - 1$ (tzw. stan jedynkowy, kiedy wszystkie zmienne są jedynkami).

Każdemu stanowi wejść można przypisać wartość funkcji 0 lub 1, niezależnie od wartości funkcji dla innych stanów. Dla godnego stanu można, więc określić dwie różne funkcje logiczne dla k stanów – 2^k funkcji. Ogólnie, liczba różnych funkcji logicznych n zmiennych wejściowych (argumentów) wynosi 2^{2n}.

Funkcje logiczne można określić za pomocą tablic, w które wpisuje się wszystkie możliwe stany wieść (w kolejności zgodnej z numerem stanu) i odpowiadające tym stanom wartości funkcji.

W tablicy 13.1 zestawiono funkcje jednej zmiennej (jednoargumentowe). Liczba tych funkcji wynosi 4, ale tylko dwie z niech są realizowane przez elementy logiczne: f_1 - powtórzenie f_2 - negacja. Funkcja negacji ma duże znaczenie w teorii układów przełączających, a przykłady jej realizacji podane są w punkcie 13.6, natomiast funkcja powtórzenia jest typowa dla niektórych elementów wzmacniających (moc sygnału „1" na wyjściu może być znacznie większa niż moc sygnału „1" na wejściu). Funkcje f_0 i f_3 nie zależą od wartości zmiennej x nie wymagają użycia elementów logicznych do ich realizacji (patrz str. 331).

Funkcje dwu zmiennych (dwuargumentowe) zestawiono w tabl. 13.2. Ponieważ liczba możliwych stanów wejść wynosi 4, funkcji tych jest $2^4 = 16$. Jak łatwo spostrzec, numeracja funkcji jest zgodna z wartościami odpowiednich liczb całkowitych wyrażonych w układzie dwójkowym. Spośród wy-

Tablica 13.1

Funkcje logiczne jednej zmiennej

x / y	0	1	Nazwa funkcji	Równanie
f_0	0	0	stała zerowa	$y = 0$
f_1	0	1	powtórzenie	$y = x$
f_2	1	0	negacja	$y = \bar{x}$
f_3	1	1	stała jedynkowa	$y = 1$

mienionych funkcji największe znaczenie mają, (obok nazwy funkcji podano odpowiadający jej spójnik w rachunku zdań oraz oznaczenie angielskie):

f_1 — *koniunkcja* — „i", AND -funkcja ma wartość 1 tylko wtedy, gdy obydwa argumenty mają wartość 1,

f_7 — *alternatywa* — „lub", OR — funkcja ma wartość 1, gdy co najmniej jeden z argumentów ma wartość 1,

f_8 — *negacja alternatywy* — „ani", „nie lub", NOR (skrót od *not or*) — funkcja ma wartość 1 tylko wtedy, gdy ani jeden z argumentów nie ma wartości 1,

f_{14} — *negacja koniunkcji* — „nie i", NAND (skrót od *not and*) — funkcja ma wartość 1, gdy co najmniej jeden z argumentów ma -wartość 0.

Funkcja f_6 (nierównoważność) nazywana jest także alternatywą wyłączającą, lub sumą, modulo dwa, a w rachunku zdań odpowiada jej słowo „albo". Funkcja ta ma wartość 1, gdy tylko jeden z jej argumentów ma wartość 1, drugi musi mieć równocześnie wartość 0. Funkcja ta jest negacją funkcji f_9 (równoważności), która ma wartość 1, gdy obydwa argumenty mają równocześnie wartość 0 lub wartość 1.

Łatwo zauważyć również, że implikacja f_{13} jest negacją, funkcji f_2 zakazu przez x_2, a implikacja odwrotna f_{11} jest negacją funkcji f_4 zakazu przez x_1.

Funkcje f_0 i f_{16} nie zależą od wartości argumentów x_1, x_2, a funkcje f_3, f_5, f_{10}, i f_{12} zależą tylko od jednego argumentu.

W przypadku trzech argumentów liczba możliwych stanów wejść wynosi $2^3 = 8$, a liczba funkcji $2^8 = 256$. Rozważanie tych funkcji, a tym bardziej funkcji większej liczby argumentów, me jest celowe, gdyż każdą funkcję dwu argumentową można traktować jako argument innej funkcji, rozszerzając tym samym pojecie funkcji logicznej na większą liczbę argumentów (zmiennych).

Tablica 13.8

Funkcje logiczne dwu zmienonych

x_1, x_2 / y	0 0	0 1	1 0	1 1	Nazwa funkcji	Oznaczenie	Równanie
f_0	0	0	0	0	stała zerowa	0	$y=0$
f_1	0	0	0	1	Koniunkcja (iloczyn logiczny)	$x_1 \wedge x_2$, $x_1 \cdot x_2$	$y=x_1x_2$
f_2	0	0	1	0	zakaz przez x_2	$x_1 \Delta x_2$	$y=x_1\bar{x}_2$
f_3	0	0	1	1	powtórzenie x_1	x_1	$y=x_1$
f_4	0	1	0	0	zakaz przez x_1	$x_2 \Delta x_1$	$y=\bar{x}_1x_2$
f_5	0	1	0	1	powtórzenie x_2.	x_2	$y=x_2$
f_6	0	1	1	0	nierównoważność	$x_1 \oplus x_2$	$y=x_1\bar{x}_2+\bar{x}_1x_2$
f_7	0	1	1	1	Alternatywa (suma logiczna)	$x_1 \vee x_2$, x_1+x_2	$y=x_1+x_2$
f_8	1	0	0	0	negacja alternatywy funkcja Peirce'a, NOR)	$x_1 \downarrow x_2$	$y=\overline{x_1+x_2}=$ $=\bar{x}_1\bar{x}_2$
f_9	1	0	0	1	równoważność	$x_1 \Leftrightarrow x_2$	$y=x_1x_2+\bar{x}_1\bar{x}_2$
f_{10}	1	0	1	0	negacja x_2	$\bar{x}_2$	$y=\bar{x}_2$
f_{11}	1	0	1	1	implikacja odwrotna	$x_2 \Rightarrow x_1$	$y=x_1+\bar{x}_2$
f_{12}	1	1	0	0	negacja x_1	$\bar{x}_1$	$y=\bar{x}_1$
f_{13}	1	1	0	1	implikacja	$x_1 \Rightarrow x_2$	$y=\bar{x}_1+x_2$
f_{14}	1	1	1	0	negacja koniunkcji (funkcja Sheffera, NAND)	$x_1 \mid x_2$	$y=\overline{x_1x_2}=$ $=\bar{x}_1+\bar{x}_2$
f_{15}	1	1	I	1	stała jedynkowa	1	$y=1$

Jeżeli np.

$$y_1 = f_1(x_1, x_2), \qquad y_2 = f_2(x_3, x_4)$$

oraz

$$z = f_3(y_1, y_2),$$

to

$$z = f_3(f_1(x_1, x_2), f_2(x_3, x_4)) = f_4(x_1, x_2, x_3, x_4).$$

Funkcja czteroargumentowa f_4 jest pewną superpozycją funkcji dwuargumentowych f_1, f_2 i f_3.

13.3. Elementy algebry Boole'a

Podstawowym narzędziem analizy i syntezy dwuwartościowych układów przełączających jest zerojedynkowa algebra Boole'a, bazująca na operacjach negacji, koniunkcji i alternatywy. W algebrze tej rozpatruje się wyrażenia zbudowane za pomocą, wymienionych trzech operacji ze stałych 0 i 1 oraz zmiennych mogących przyjmować wartości 0 lub 1 (do zmiennych zalicza się dowolne funkcje logiczne, m. in. zwykłe zmienne dwójkowe).

Kolejność wykonywania działań, gdy wyrażenie nie zawiera nawiasów, jest taka: najpierw negacja, później koniunkcja, w końcu alternatywa. Gdy są, nawiasy, najpierw wykonuje się działania wewnątrz nich. W przypadku negacji wyrażeń złożonych często opuszcza się nawiasy; role, nawiasu odgrywa wówczas sam symbol negacji (zachowuje przy tym ważność poprzednie zdanie o zmianie kolejności wykonywania działań w obecności nawiasów).

Występujące w równaniach znaki „+” oraz „$\cdot$” należy traktować jako znaki alternatywy (sumy logicznej) oraz koniunkcji (iloczynu logicznego) i należy je czytać ,,lub" oraz ,,i”. Nie oznaczają one dodawania lub mnożenia w klasycznym sensie algebraicznym.

Dla uproszczenia zapisu znak „$\cdot$” będziemy opuszczać w każdym, przypadku, gdy nie, pociąga to za sobą możliwości nieporozumień. Pierwsze przykłady równań z opuszczonymi znakami koniunkcji znajdujemy już w tabl. 13.2.

Algebra Boole'a przyjmuje jako aksjomaty podane niżej tożsamości.

Prawa przemienności

$$x + y = y + x \tag{13.1}$$

$$xy = yx \tag{13.1a}$$

Prawa łączności

$$x + (y + z) = (x + y) + z = x + y + z \tag{13.2}$$

$$x(yz) = (xy)x = xyz \tag{13.2a}$$

Prawa rozdzielności

$$x(y+z) = xy + xz \tag{13.3}$$

$$x + yz = (x+y)(x+z) \tag{13.3a}$$

Prawa de Morgana:
a) dla dwóch zmiennych

$$\overline{x+z} = \bar{x}\bar{y} \tag{13.4}$$

$$\overline{xy} = \bar{x} + \bar{y} \tag{13.4a}$$

b) dla większej liczby zmiennych

$$\overline{x+z} = \bar{x}\bar{y} \tag{13.5}$$

$$\overline{xy} = \bar{x} + \bar{y} \tag{13.5a}$$

Prawo podwójnego zaprzeczenia (wynikające z definicji negacji)

$$\bar{\bar{x}} = x. \tag{13.6}$$

Z definicji funkcji alternatywy i koniunkcji wynikają, zależności

$$x+0=x, \quad (13.7) \qquad x\cdot 0=0, \quad (13.7a)$$

$$x+1=1, \quad (13.8) \qquad x\cdot 1=x, \quad (13.8a)$$

$$x+x=x, \quad (13.9) \qquad x\cdot x=x, \quad (13.9a)$$

$$x+\bar{x}=1, \quad (13.10) \qquad x\cdot\bar{x}=0. \quad (13.10a)$$

Operacje wykonywane na stałych zero i jeden:

$$0+0=0, \quad (13.11) \qquad 0\cdot 0=0, \quad (13.11a)$$

$$1+0=1, \quad (13.12) \qquad 1\cdot 0=0, \quad (13.12a)$$

$$1+1=1, \quad (13.13) \qquad 1\cdot 1=1, \quad (13.13a)$$

$$\bar{0}=1, \quad (13.14) \qquad \bar{1}=0. \quad (13.14a)$$

Podane zależności podstawowe algebry Boole'a pozwalają latwo wyprowadzić następne. Istotne znaczenie przy przekształcaniu funkcji logicznych miały zwłaszcza tzw. *prawa pochłaniania*

$$x + xy = x \tag{13.15}$$

$$x(x+y) = x \tag{13.15a}$$

Pierwsze z tych praw można udowodnić pisząc następujący łańcuch równań:

$$x + xy = x\cdot 1 + xy = x(1+y) = x\cdot 1 = x$$

Wykorzystano tu kolejno zależności (13.8a), (13.3), (13.8) i (13.8a).

Podobnie można udowodnić drugie prawo pochłaniania

$$x(x+y) = xx + xy = x + xy = x(1+y) = x$$

Wykorzystano tu kolejno zależności (13.3), (13.9), (13.15).

Oczywiście, identycznie przebiegałyby dowody tożsamości

$$x + x\bar{y} = x, \tag{13.16}$$

$$x(x+\bar{y}) = x. \tag{13.16a}$$

Niżej zestawiono kilka dalszych tożsamości ułatwiających upraszczanie równań; (13.17) i (13.18) nazywa się często „regułami sklejania”

$$(x+y)(x+\bar{y}) = x, \tag{13.17}$$

$$xy + x\bar{y} = x, \tag{13.18}$$

$$x + \bar{x}y = x + y, \tag{13.19}$$

$$\bar{x} + xy = \bar{x} + y. \tag{13.20}$$

Przeprowadzenie dowodów tych tożsamości zaleca się jako ćwiczenie sprawdzające opanowanie podanych wcześniej zależności podstawowych.

13.4. Rozkład wyrażeń strukturalnych. Postacie normalne

Każde wyrażenie strukturalne może być rozłożone na składniki jedynki lub czynniki zera (konstytuanty jedynki lub zera). Omówiony zostanie przede wszystkim rozkład jedynki i zera, a następnie rozkład dowolnej funkcji.

Rozkład jedynki na składniki zilustrowany zostanie na przykładzie dwu zmiennych x_1 i x_2. Zgodnie z zależnością (13.10) można napisać

$$x_1 + \bar{x} = 1, \quad x_2 + \bar{x}_2 = 1,$$

a wykorzystując (13.13a) otrzymamy

$$1 = 1 \cdot 1 = (x_1 + \bar{x}_1)(x_2 + \bar{x}_2) = x_1x_2 + x_1\bar{x}_2 + \bar{x}_1x_2 + \bar{x}_1\bar{x}_2. \tag{13.21}$$

Postępując podobnie dla trzech zmiennych, otrzymamy alternatywę (sumę logiczną) ośmiu koniunkcji elementarnych, przedstawiających wszystkie możliwe stany zmiennych. Koniunkcje te zestawiono w tabl. 13.3.

Zapis wyrażenia (13.21) oraz odpowiednich wyrażeń dla większej liczby zmiennych można uprościć, przypisując umownie zmiennym x symbol 1, a zmiennym $\bar{x}$ symbol 0 i oznaczając poszczególne koniunkcje K indeksem dziesiętnym odpowiadającym utworzonym przez nie liczbom binarnym. Na przykład wyrażeniu x_1x_2 odpowiada liczba binarna (11), a więc dziesiętna 3:

$$x_1x_2 = K_{(11)} = K_3, \quad \bar{x}_1x_2 = K_{(01)} = K_1,$$

$$x_1x_2x_3 = K_{(111)} = K_7, \quad \bar{x}_1x_2\bar{x}_3 = K_{(010)} = K_2, \quad ...$$

Tablica 13.3

Składniki jedynki i czynnika zera dla trzech zmiennych

Nr stanu	Stan wejść x_1	x_2	x_3	Składniki jedynki K	Czynniki zera D
0	0	0	0	$K_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$D_0 = x_1 + x_2 + x_3$
1	0	0	1	$K_1 = \bar{x}_1\bar{x}_2 x_3$	$D_1 = x_1 + x_2 + \bar{x}_3$
2	0	1	0	$K_2 = \bar{x}_1 x_2\bar{x}_3$	$D_2 = x_1 + \bar{x}_2 + x_3$
3	0	1	1	$K_3 = \bar{x}_1 x_2 x_3$	$D_3 = x_1 + \bar{x}_2 + \bar{x}_3$
4	1	0	0	$K_4 = x_1\bar{x}_2\bar{x}_3$	$D_4 = \bar{x}_1 + x_2 + x_3$
5	1	0	1	$K_5 = x_1\bar{x}_2 x_3$	$D_5 = \bar{x}_1 + x_2 + \bar{x}_3$
6	1	1	0	$K_6 = x_1 x_2\bar{x}_3$	$D_6 = \bar{x}_1 + \bar{x}_2 + x_3$
7	1	1	1	$K_7 = x_1 x_2 x_3$	$D_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

Wyrażenie (13.21) zapiszemy teraz

$$1 = K_3 + K_2 + K_1 + K_0 .$$

Ogó1nie, dla n zmiennych

$$1 = \sum_{i=0}^{2^n-1} K_i , \tag{13.22}$$

gdzie Σ - symbol alternatywy (sumy logicznej), K_i – składniki (konstytuenty) jedynki - koniunkcje elementarne zupełne, tzn. koniunkcje wszystkich n zmiennych lub ich negacji.

Rozkład zera na czynniki również zostanie pokazany najpierw na przykładzie dwu zmiennych. Wykorzystamy przy tym zależności (13.10a), (13.11) i dwukrotnie (13.3a).

$$\begin{gathered} x_1\bar{x}_1 = 0, \quad x_2\bar{x}_2 = 0, \\ 0 = 0 + 0 = x_1\bar{x}_1 + x_1\bar{x}_2 = (x_1 + x_2\bar{x}_2)(\bar{x}_1 + x_2\bar{x}_2) = \\ = (x_1 + x_2)(x_1 + \bar{x}_2)(\bar{x}_1 + x_2)(\bar{x}_1 + \bar{x}_2) \end{gathered} \tag{13.23}$$

Postępując podobnie dla trzech zmiennych, otrzymamy koniunkcje (iloczyn logiczny) ośmiu alternatyw elementarnych, przedstawiających wszystkie możliwe stany zmiennych (tabl. 13.3).

Uprościmy teraz zapis przypisując umownie zmiennym x symbol 0, a zmiennym $\bar{x}$ symbol 1 (dla jednolitości zapiau, widocznej w tabl. 13.3) i oznaczając poszczególne alternatywy (dysjunkcje) D indeksem dziesiętnym odpowiadającym utworzonym przez nie liczbom binarnym. Na przykład wyrażeniu x_1+x_2 odpowiada liczba binarna (00), a wiec dziesiętna 0:

$$\begin{gathered} x_1 + x_2 = D_{(00)} = D_0, \quad x_1 + \bar{x}_2 = K_{(01)} = D_1, \\ \bar{x}_1 x_2 x_3 = D_{(100)} = D_4, \quad \bar{x}_1\bar{x}_2\bar{x}_3 = D_{(111)} = D_7, \quad \ldots \end{gathered}$$

Wyrażenie (13.23) otrzyma postać

$$0=D_0D_1D_2D_3$$

Ogólnie dla *n* zmiennych

$$0=\prod_{i=0}^{2^n-1} K_i\,, \tag{13.24}$$

gdzie Π — symbol koniunkcji (iloczynu logicznego), D_i—czynniki (konstytuenty) zera — elementarne alternatywy zupełne, tzn. alternatywy wszystkich *n* zmiennych lub ich negacji.

Rozkład funkcji względem składników jedności. Rozważmy funkcję *n* zmiennych $f(x_1, x_2, ..., x_n)$. Funkcja ta może być rozłożona na składniki względem swoich zmiennych (argumentów), np. względem x_1. Ponieważ w danej funkcji mogą występować zarówno poszczegó1ne zmienne, jak i ich negacje, wyłączając, więc x_1 i $\bar{x}_1$ otrzymamy wyrażenie

$$f(x_1, x_2, ... ,x_n) = Ax_1+B\bar{x}_1\,, \tag{13.25}$$

w którym *A* i *B* — funkcje boolowskie pozostałych zmiennych. Podstawiając $x_1 = 1$, $\bar{x}_1 = 0$ wyznaczymy *A*

$$f(1, x_2, ..., x_n) = A\cdot 1+B\cdot 0=A,$$

natomiast podstawiając $x_1 = 0$, $\bar{x}_1 = 1$ wyznaczymy *B*

$$f(0, x_2, ... ,x_n) = A\cdot 0+B\cdot 1=B.$$

Wyrażenie (13.25) można teraz przedstawić w postaci

$$f(x_1, x_2, ... ,x_n) = f(1, x_2, ... ,x_n)x_1+f(0, x_2, ... ,x_n)\bar{x}_1\,, \tag{13.26}$$

Jeżeli następnie każdą funkcję z prawej strony wyrażenia (13.26) rozłożyć podobnie względem x_2, to otrzyma się.

$$f(x_1, x_2, ..., x_n) = f(1, 1, ..., x_n)x_1x_2+f(1, 0, ..., x_n)x_1\bar{x}_2 + \\ +f(0, 1, ..., x_n)\bar{x}_1x_2+f(0, 0, ..., x_n)\bar{x}_1\bar{x}_2\,. \tag{13.27}$$

Warto zauważyć, że prawa strona wyrażenia (13.27) jest alternatywy składni-ków, w których występują koniunkcje elementarne zupełne zmiennych x_1 i x_2, (por. wzory 13.21 i 13.22).

Rozkładając dalej prawą stronę wyrażenia (13.27) względem zmiennych x_3 , …, x_n, otrzymamy ostatecznie

$$f(x_1, x_2, ..., x_n) = f(1, 1, ..., 1)x_1x_2...x_n,+ f(1, 1, ..., 0)x_1x_2...\bar{x}_n + \\ +f(1, 0, ..., 0)x_1\bar{x}_2...\bar{x}_n+f(0, 0, ..., x_n)\bar{x}_1\bar{x}_2...\bar{x}_n\,, \tag{13.28}$$

Zapis ten można skrócić oznaczając poszczegó1ne koniunkcje elementarne zupełne symbolem K_i według (13.22), a funkcje symbolem f_i, gdzie indeks *i* jest

liczbą dziesiętną równoważna, liczbie binarnej utworzonej przez konkretne wartości argumentów funkcji. Na przykład

$$f(1, 1, 1) = f_7, \qquad f(1, 1, 0) = f_6, \qquad \ldots$$

Po wprowadzeniu podanych skrótowych oznaczeń ogólny wzór na rozkład funkcji n zmiennych przyjmie postać

$$f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^n-1} f_i K_i \tag{13.29}$$

gdzie Σ jest symbolem alternatywy (sumy logicznej).

Funkcje mają wartość 1 lub 0 jednoznacznie określona dla konkretnych wartości argumentów funkcji. Dla $f_i = 0$ również $f_i K_i = 0$, natomiast dla $f_i = 1$ mamy $f_i K_i = K_i$. Wynika z tego, że każda funkcja może być przedstawiona w postaci alternatywy tych składników jedynki (koniunkcji elementtarnych zupełnych), dla których funkcje f_i mają wartości 1. Postać taka jest tylko jedna dla danej funkcji i nosi nazwę *normalnej zupełnej postaci alternatywnej* lub *koniunkcyjnej postaci alternatywnej*.

Rozkład funkcji względem czynników zera. Funkcję $f(x_1, x_2, ..., x_n)$ można również rozłożyć względem x_1 według wyrażenia

$$f(x_1, x_2, \ldots, x_n) = (A + x_1)(B + \bar{x}_1), \tag{13.30}$$

Funkcje A i B wyznaczymy podstawiając a) $x_1 = 0$, $\bar{x}_1 = 1$, b) $x_1 = 1$, $\bar{x}_1 = 0$,

a) $f(x_1, x_2, \ldots, x_n) = (A+0)(B+1) = A \cdot 1 = A$,

b) $f(x_1, x_2, \ldots, x_n) = (A+1)(B+0) = B \cdot 1 = B$.

Podstawiając wyznaczone wartości A i B do (13.30) otrzymamy

$$f(x_1, x_2, \ldots, x_n) = [f(0, x_2, \ldots, x_n) + x_1)(f(1, x_2, \ldots, x_n) + \bar{x}_1] \tag{13.31}$$

Stosując dalej analogiczne postępowanie jak przy wyznaczaniu wzoru (13.29) otrzymujemy ostatecznie

$$f(x_1, x_2, ..., x_n) = \prod_{i=0}^{2^n-1} (f_i + D_i), \tag{13.32}$$

gdzie Π — symbol koniunkcji (iloczynu logicznego), f_i — określone przy (13.29), D_i — określone przy (13.24).

Funkcje f_i mają wartość 1 lub 0 jednoznacznie określoną dla konkretnych wartości argumentów funkcji. Dla $f_i = 0$ mamy $f_i + D_i = D_i$, natomiast dla $f_i = 1$ również $f_i + D_i = 1$. Zgodnie z (13.8a) w koniunkcji Π możemy opuścić wszystkie jedynki.

Każda funkcja może być, więc przedstawiona w postaci koniunkcji tych czynników zera (alternatyw elementarnych zupełnych), dla których funkcje f_i

mają wartość 0. Postać taka jest tylko jedna dla danej funkcji i nosi nazwę *normalnej zupełnej postaci koniunkcyjnej* lub *kanonicznej postaci koniunkcyjnej.*

Przypadki szczegó1ne. Jeżeli wszystkie $f_i = 1$, tzn. dla wszystkich stanów wejściowych wartość funkcji wynosi 1, to według (13.29)

$$f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^n-1} K_i ,$$

a według (13.32)

$$f(x_1, x_2, \ldots, x_n) = 1;$$

zatem

$$\sum_{i=0}^{2^n-1} K_i = 1 .$$

Otrzymany wynik przedstawia rozkład jedynki według (13.22).

Jeżeli wszystkie $f_i = 0$, tzn. dla wszystkich stanów wejść wartość funkcji wynosi 0 to według (13.29)

$$f(x_1, x_2, \ldots, x_n) = 0,$$

a według (13.32)

$$f(x_1, x_2, ..., x_n) = \prod_{i=0}^{2^n-1} D_i ,$$

zatem

$$\prod_{i=0}^{2^n-1} D_i = 0 .$$

Otrzymany wynik przedstawia rozkład zera według (13.24)

Inne postacie normalne. Oprócz postaci kanonicznych często spotykać będziemy normalne, ale nie zupełne, postacie alternatywną lub koniunkcyjną.

W normalnej postaci alternatywnej występować będzie zawsze alternatywa koniunkcji elementarnych, tzn. koniunkcji, w których nie ma powtarzających się liter. Mogą, to nie być koniunkcje zupełne, lecz koniunkcje tylko niektórych zmiennych lub ich negacji, np.

$$x_1, \quad x_1 \bar{x}_2, \quad \bar{x}_1 x_2 x_3.$$

Nie są natomiast koniunkcjami elementarnymi wyrażenia

$$\overline{x_1 x_2}, \quad x_1 x_2 x_1, \quad x_1 \bar{x}_1,$$

ponieważ w pierwszym z nich znak negacji znajduje się nad całym wyrażeniem (a nie nad tworzącymi je literami), a w drugim i trzecim występują dwukrotnie jednakowe litery.

W normalnej postaci koniunkcyjnej występować będzie zawsze koniunkcja alternatyw elementarnych, tzn. alternatyw, w których nie ma powtarzających

się liter. Mogą to nie być alternatywy zupełne, lecz alternatywy tylko niektórych zmiennych lub ich negacji, np.

$$x_1, \quad x_1+\bar{x}_2, \quad \bar{x}_1+\bar{x}_2+\bar{x}_3.$$

Nie są natomiast alternatywami elementarnymi wyrażenia

$$\overline{x_1+x_2}, \quad x_1+x_2+x_1, \quad x_1+x_2+\bar{x}_1$$

z tych samych powodów jak poprzednio.

13. Systemy funkcjonalnie pełne

Projektując układ logiczny (przełączający) trzeba znać zbiór elementów, które użyte zostaną. do budowy układu. Do zbudowania dowolnie złożonego układu wystarczy dysponować niewielką liczbą, typów elementów, realizujących funkcje logiczne tworzące tzw. system funkcjonalnie pełny.

Zbiór funkcji logicznych nazywa się systemem funkcjonalnie pełnym (bazą), jeżeli dowolną funkcję logiczną $f(x_1\ x_2,\ ...,\ x_n)$ można przedstawić jako pewną kombinację argumentów (lub funkcji) $x_1\ x_2,\ ...,\ x_n$ stałych 0 i 1 oraz funkcji należących do tego zbioru.

Wszystkie funkcje dwu argumentów mogą być przedstawione za pomocą operacji algebry Boole'a, tzn. negacji, koniunkcji i alternatywy oraz stałych 0 i 1. Ponieważ za pomocą funkcji dwu argumentów można utworzyć dowolną funkcję wieloargumentową, zatem za pomocą algebry Boole'a można wyrazić również dowolną funkcję wieloargumentową. Wynika z tego, że funkcje negacji, koniunkcji i alternatywy tworzą *podstawowy system funkcjonalnie pełny.* Każdy inny system jest tylko wtedy funkcjonalnie pełny, jeżeli jego funkcje pozwalają. Utworzyć negację, koniunkcję i alternatywę.

Łatwo stwierdzić, że podstawowy system funkcjonalnie pełny nie jest minimalny, gdyż np. alternatywy można utworzyć za pomocą negacji i koniunkcji

$$x_1+x_2=\overline{\bar{x}_1\cdot\bar{x}_2},$$

System zawierający tylko negację i koniunkcję jest, więc również funkcjonalnie pełny.

Podobnie można utworzyć koniunkcję za pomocą negacji i alternatywy

$$x_1\cdot x_2=\overline{\bar{x}_1+\bar{x}_2},$$

a więc system zawierający tylko negację i alternatywę jest także funkcjonalnie pełny.

Ostatnie dwa systemy są już minimalne, gdyż usunięcie z nich którejkolwiek funkcji sprawia, że system przestaje być funkcjonalnie pełny.

Ogólnie można powiedzieć, że *system* jest *minimalny funkcjonalnie pełny,* jeżeli jego wszystkie funkcje są niezbędne do spełnienia warunku pełności systemu.

Minimalne funkcjonalnie pełne są również systemy bazujące tylko na funkcji alternatywy (NOR) lub tylko na funkcji negacji koniunkcji (NAND).

Za pomocą. funkcji NOR, zwanej również funkcją Peirce'a, $y=\overline{x_1+x}$, otrzymuje się w następujący sposób negacje, koniunkcję i alternatywę:

a) negacja $\bar{x}_1=\overline{x_1+x_1}=\overline{x_1+0}$,

b) koniunkcja $x_1x_2=\overline{\bar{x}_1+\bar{x}_2}=\overline{\overline{x_1+0}+\overline{x_2+0}}$, (13.33)

c) alternatywa $x_1+x_2=\overline{\overline{x_1+x_2}}=\overline{\overline{x_1+x_2}+0}$,

natomiast za pomocą, funkcji NAND, zwanej również funkcją, Sheffera, $y=\overline{x_1x_2}$

a) negacja $\bar{x}_1=\overline{x_1x_1}=\overline{x_1\cdot 1}$,

b) koniunkcja $x_1x_2=\overline{\overline{x_1x_2}}=\overline{\overline{x_1x_2}\cdot 1}$, (13.34)

c) alternatywa $x_1+x_2=\overline{\bar{x}_1\bar{x}_2}=\overline{(\overline{x_1\cdot 1})(\overline{x_2\cdot 1})}$.

Uzyskanie stałych 1 i 0 nie sprawia żadnych trudności, gdyż na przykład w układach elektrycznych utożsamia się je z odpowiednimi napięciami zasilającymi, a w układach pneumatycznych z ciśnieniami (najczęściej 1 odpowiada ciśnieniu zasilania, a 0 ciśnieniu atmosferycznemu) itd. Dlatego też stałych 1 i 0 nie wymienia się wśród składników bazy.

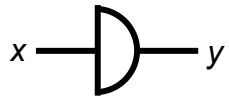
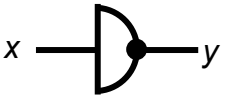
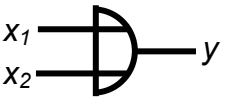
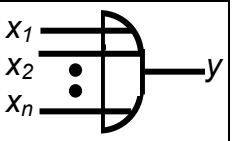
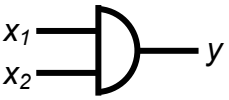
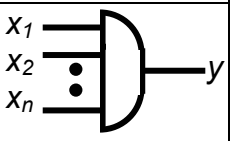
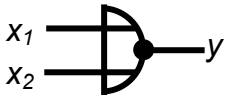
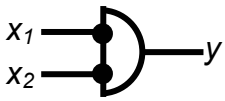
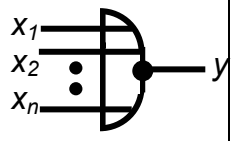
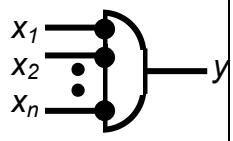
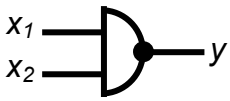
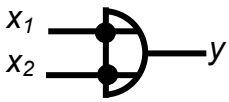
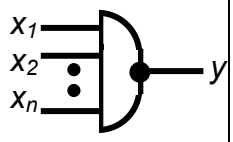
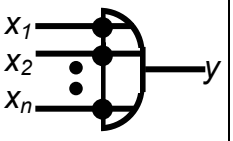
Wybór minimalnego systemu funkcjonalnie pełnego bazującego na jednym lub dwóch elementach podstawowych jest bardzo atrakcyjny dla producenta (duże serie identycznych elementów), ale często zwiększa liczbę elementów użytych do budowy układu w stosunku do systemów, nieminimalnych. Zwiększa się wtedy również liczba połączeń, możliwość uszkodzeń itd. Dlatego niekiedy rezygnuje się z minimalności systemu i wprowadza się elementy dodatkowe lub przynajmniej odmiany elementów podstawowych różniące się liczbą wejść, co pozwala zmniejszyć liczbę potrzebnych elementów i uprościć montaż układu.

13.6. Elementy logiczne

Przydatność praktyczna różnych systemów funkcjonalnie pełnych zależy w decydującym stopniu od budowy i własności elementów logicznych. W niniejszym punkcie przedstawione zostaną jedynie przykłady rozwiązań elementów elektronicznych (półprzewodnikowych) oraz pneumatycznych i hydraulicznych. Nie będą natomiast omawiane przykłady klasycznych rozwią-

Tablioa 13.4

Symbole graficzne typowych elementów logicznych

Liczba wejść / Nazwa i funkcja elementu	1 lub 2	*n*
porównanie $y = x$		---
negacja $y = \bar{x}$		---
alternatywa $y = x_1 + x_2 + \ldots + x_n$		
koniunkcja $y = x_1 x_2 \ldots x_n$		
negacja alternatywa NOR $y = \overline{x_1 + x_2 + \ldots + x_n} = \bar{x}_1 \bar{x}_2 \ldots \bar{x}$	 	 
negacja koniunkcji NAND $y = \overline{x_1 x_2 \ldots x_n} = \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_n$	 	 

zań przekaźnikowych (tzn. bazujących na przekaźnikach elektromagnetycznych), a także rozwiązań magnetycznych i innych. Czytelnik zainteresowany tym zagadnieniem znajdzie obszerne informacje w literaturze cytowanej na końcu części III książki.

Nazwy elementów logicznych utożsamia się zwykle z nazwami funkcji realizowanych przez nie. Poszczególne elementy oznacza się na schematach układów za pomocą symboli graficznych *) zestawionych w tabl. 13.4. Dla elementów NOR i NAND podano dwie wersje symboli, odpowiadające dwu postaciom funkcji według prawa de Morgana.

13.6.1. Elementy półprzewodnikowe

Do budowy tych elementów wykorzystywane są diody i tranzystory. Stosowane są rozwiązania konstrukcyjne:

— montaż płaski podzespołów na płytkach drukowanych,
— montaż przestrzenny i zalanie żywicą epoksydową; (kostki),
— układy scalone.

Zwłaszcza technika układów scalonych rozpowszechnia się obecnie coraz szerzej. Prowadzone są, prace nad tzw. scalaniem wielkoskalowym, polegającym na wykonywaniu na jednym podłożu całych bloków funkcjonalnych, zawierających od kilkudziesięciu do kilkuset elementów logicznych.

Zalety elementów półprzewodnikowych: bardzo duża szybkość działania, małe wymiary, stosunkowo duża niezawodność. Ostatnie dwie cechy dotyczą zwłaszcza układów scalonych.

Przy omawianiu przykładów budowy elementów półprzewodnikowych przyjmiemy umownie, że sygnał „0” reprezentowany jest przez napięcie o wartości bliskiej 0 V, a sygnał „1” przez ujemne lub dodatnie napięcie o wartości bliskiej napięcia zasilającego.

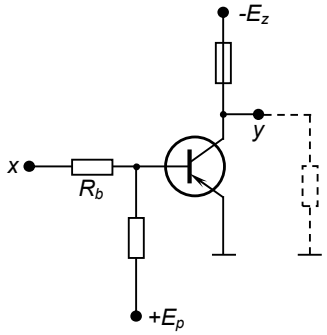
Element negacji. Realizowany jest w postaci tzw. negatora tranzystorowego o schemacie przedstawionym na rys. 13.3.

Przy podaniu na wejście x sygnału „0” baza tranzystora pozostaje spolaryzowana dodatnio względem emitera i tranzystor nie przewodzi (znajduje się w stanie odcięcia). Napięcie na wyjściu y jest wtedy równe „1”. Podanie na wejście sygnału „1” powoduje stan nasycenia tranzystora, kolektor jest wtedy praktycznie zwarty z emiterem (a więc z masą) i napięcie wyjściowe jest równe „0”.

Zwykle wymaga się, aby element logiczny mógł wysterować, co najmniej 3÷5 innych elementów. Wstępna polaryzacja bazy i dzielnik wejściowy są, tak dobierane, że tranzystor nasyca się przy napięciach wejściowych rzędu 50% napięcia zasilającego. Dzięki temu można dopuścić duże tolerancje sygnałów

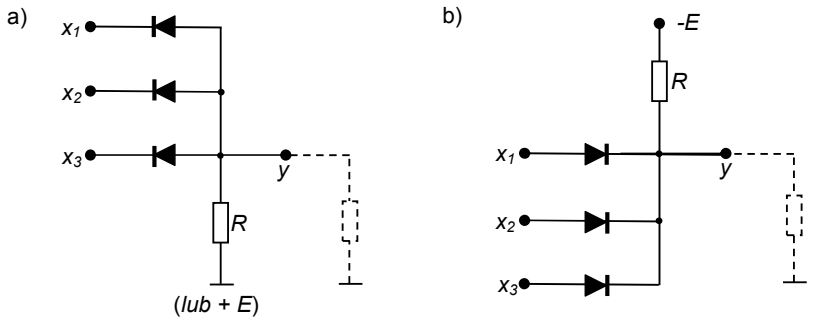
*) Według PN-64/M-42003. W literaturze spotyka się również inne symbole graficzne.

„1” i „0”, niezbędne ze względu na wpływ rezystancji obciążenia na sygnały wyjściowe układów. Rezystancja ta, zaznaczona linią kreskowaną, na rysunku, powoduje zmniejszanie napięcia wyjściowego w stosunku do napięcia zasilającego (tworzy się dzielnik wyjściowy).



Rys. 13.3. Schemat tranzystorowego elementu negacji z tranzystorem *pnp* (w przypadku tranzystora *npn* znaki napięć zasilających bądą przeciwne: $+E_z$, $-E_p$)

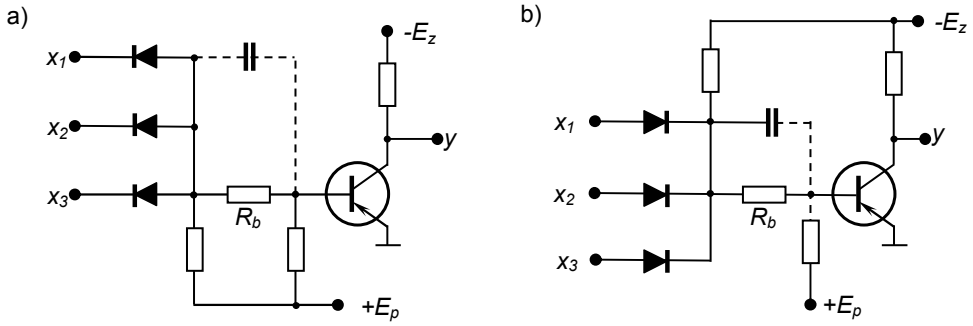
Elementy alternatywy i koniunkcji. Budową diodowych elementów alternatywy i koniunkcji, trójwejściowych, pokazano na rys. 13.4. W elemencie alternatywy, jeżeli na któreś z wejśc podane zostanie napięcie ujemne (sygnał ,,1”), to napięcie to pojawia się również na wyjściu, gdyż odpowiednia dioda przewodzi i zwiera *y* z tym wejściem. Tylko w przypadku, gdy na wszystkie trzy wejścia podany jest sygnał „0”, na wyjściu jest również „0”.



Rys. 13.4. Schematy diodowych elementów: a) alternatywy, b) koniunkcji

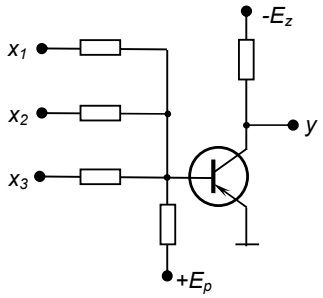
W elemencie koniunkcji, jeżeli na któreś z wejść podany jest sygnał „0”, to na wyjściu jest również „0”, gdyż odpowiednia dioda przewodzi i zwiera *y* *z* tym wejściem. Tylko w przypadku, gdy na wszystkie trzy wejścia podany jest sygnał „1”, na wyjściu jest również „1”.

Przy projektowaniu układów z elementów diodowych trzeba uwzględnić szereg ograniczeń, spowodowanych spadkiem napięcia na diodach przewodzących i prądem wstecznym diod nieprzewodzących Dlatego np. rezystor *R* w elemencie alternatywy przyłącza się niekiedy do napięcia dodatniego. Liczba wejść elementów diodowych może sięgać 10 obciążalność (zdolność wysterowania innych elementów) 2÷5 [3].



Rys. 13.5. Schematy elementów DTL: a) NOR, b) NAND

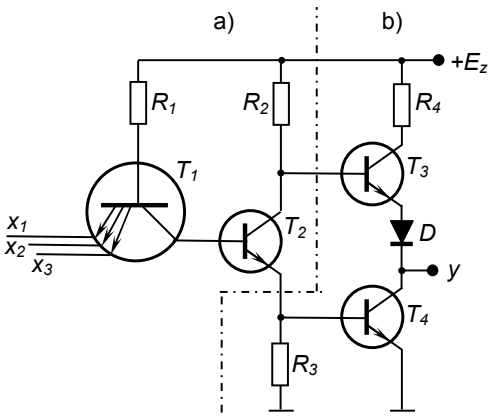
Elementy NOR i NAND. Przez szeregowe połączenie elementów alternatywy lub koniunkcji według rys. 13.4 z negatorem tranzystorowym według rys. 13.3 otrzymuje się odpowiednio elementy NOR lub NAND przedstawione na rys. 13.5. Są to elementy diodowo-tranzystorowe, oznaczane często skrótowo DTL *(Diode-Transistor Logic).* Wzmacniające działanie tranzystorów łagodzi ograniczenia dotyczące obciążalności (dopuszcza się 5÷6) i struktury połączeń między elementami. Dodanie kondensatora ma na celu zwiększenie szybkości działania elementu.



Rys. 13.6. Schemat elementu NOR typu RTL

Wiele systemów (np. krajowy Logister E-50) bazuje na elemencie NOR z wejściami oporowymi, według rys. 13.6. Elementy tego typu, oznaczane RTL (*Resistor-Transistor Logic*), mają wprawdzie mniejszą, szybkość działania, mniejszą obciążalność i mniejszą, liczbę wejść (zwykle 3 do 4) niż elementy DTL, ale są tańsze i bardziej niezawodne.

Jeżeli na wszystkich wejściach elementu z rys. 13.6 jest sygnał „0”, to baza tranzystora jest spolaryzowana dodatnio, tranzystor nie przewodzi i na wyjściu jest sygnał „1”. Pojawienie się na jednym z wejść sygnału „1” powoduje nasycenie tranzystora i zmianę wartości sygnału *y* na „0”. Pojawienie się sygnałów „1” na pozostałych wejściach wprowadza tranzystor głębiej w nasycenie, co właśnie jest główną przyczyną zmniejszenia szybkości działania elementu (bocznikowanie rezystorów kondensatorami jest tu niemożliwe, gdyż układ działałby wtedy w stanach przejściowych jak generator impulsu).



Rys. 13.7. Schemat układu TTL: a) element NAND, b) wzmacniacz wyjściowy. W układzie zastosowano tranzystory *npn;* stąd dodatnia wartość napięcia zasilającego i napięcia reprezentującego sygnał „1”.

W technice obwodów scalonych szeroko stosowane są obecnie elementy tranzystorowo-tranzystorowe, zwane skrótowo TTL (*Transistor-Transistor Logic*). Na rysunku 13.7 przedstawiono realizację elementu NAND, polegająca, na zastąpieniu diodowego elementu koniunkcji (występującego w rozwiązaniu DTL z rys. 13.5b) tranzystorem wieloemiterowym. Dla uzyskania dużej, obciążalności układu, do elementu NAND dołączony jest wzmacniacz wyjściowy sterowany będącymi w przeciwfazie sygnałami z emitera i kolektora tranzystora T_2. Gdy jeden z podłączonych przeciwsobnie tranzystorów T_3 i T_4 przewodzi, wówczas drugi znajduje się w stanie odcięcia. Wyjście *y* jest, więc dołączone do masy lub do napięcia zasilającego, co jest szczególnie korzystne przy obciążeniu pojemnościowym. Rezystancja R_4 jest niewielka i służy do zabezpieczenia tranzystorów w trakcie trwania procesów przejściowych, gdy przez moment mogą oba przewodzić.

Jeżeli chociaż jeden sygnał wejściowy ma wartość „0”, to tranzystor T_2 znajduje się w stanie odcięcia, tranzystor T_4 również, natomiast T_3 przewodzi dzięki dodatniej polaryzacji bazy poprzez rezystor R_2. Na wyjściu *y* jest wiec sygnał „1”.

Jeżeli na wszystkie wejścia podany jest sygnał „1", to tranzystor T_2 przewodzi, dzięki prądowi bazy płynącemu przez R_1. Napięcie na emiterze T_2 jest równe spadkowi napięcia baza-emiter przewodzącego tranzystora T_4 i wynosi kilka dziesiątych wolta. Napięcie na kolektorze T_2 jest niewiele wyższe. Aby napięcie to mogło odciąć tranzystor T_3, w jego emiter włączona jest dioda D. Spadek napięcia na tej diodzie powoduje, że potencjał bazy będzie w rozpatrywanym przypadku niższy od potencjału emitera. Tranzystor T_3 znajduje się więc w stanie odcięcia i na wyjściu jest sygnał „0".

Układy TTL charakteryzują. się dużą szybkością, działania, liczba wejść dochodzi do 8, a obciążalność do 10.

13.6.2. Elementy membranowe

Są to elementy pneumatyczne, których działanie przełączające wywoływane jest przez dwustanowe sygnały ciśnieniowe, podawane do komór wydzielonych przez poszczególne membrany.

Systemy membranowych elementów logicznych znalazły liczne zastosowania przemysłowe, zarówno, ze względu na ogólne zalety elementów płynowych (trwałość, bezpieczeństwo pracy w środowisku wybuchowym, niewrażliwość na pola magnetyczne, znikomy wpływ temperatury na własności elementów), jak i ze względu na dość dużą, moc sygnałów wyjściowych, pozwalającą, w niektórych przypadkach na bezpośrednie wysterowanie elementów wykonawczych.

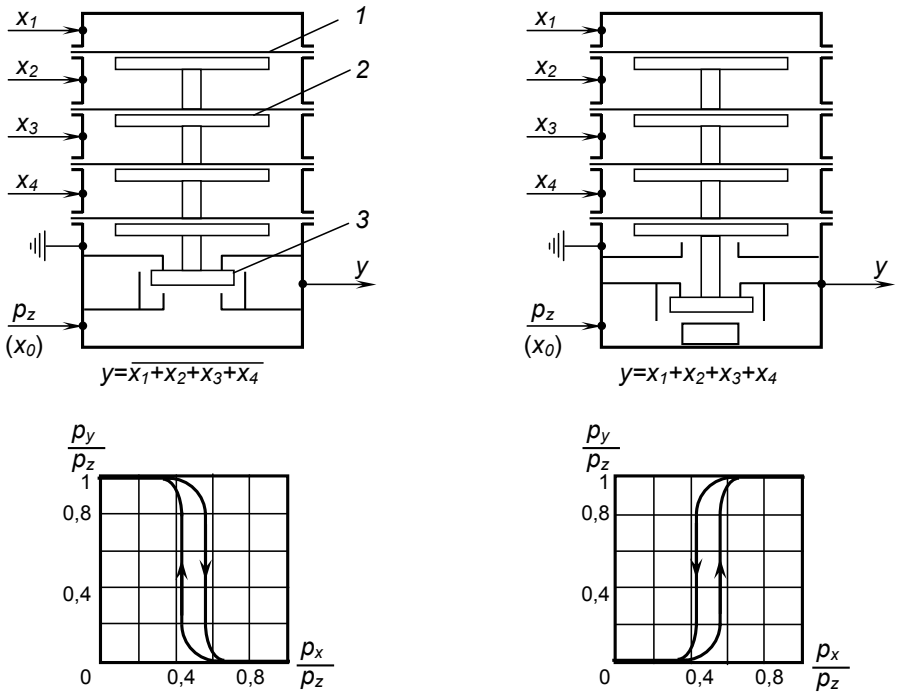
Na rysunku 13.8 przedstawiono schematy budowy membranowych elementów logicznych systemu MEEALOG*), produkowanego w kraju. Charakterystyczną. cechą, tych elementów jest zastosowanie tzw. stosu membran, w którym wiotkie membrany 1 nie są, trwale związane ze sztywnikami 2, lecz są swobodne i dopiero w przypadku podania do którejkolwiek komory sygnału jedynkowego odpowiednia membrana za pośrednictwem swojego sztywnika z popychaczem przesuwa do dołu znajdujące się niżej membrany i sztywniki, a popychacz ostatniego sztywnika przesuwa płytkę przełączającą 3. Wartość ciśnienia sygnału wejściowego, przy której zachodzi proces przełączania, zależy od stosunku powierzchni efektywnych płytki przełączającej i membrany i wynosi około 0,5 ciśnienia zasilania p_z.

W elemencie realizującym funkcję negacji alternatywy (rys. 13.8a) przy wszystkich sygnałach x równych zeru płytka 3 pod wpływem ciśnienia p_z znajduje się w górnym położeniu i na wyjściu y panuje ciśnienie p_z reprezentujące sygnał „1". Jeżeli którykolwiek sygnał (lub kilka sygnałów) x jest równy jeden, to płytka 3 zostaje przesunięta do dolnego położenia, w którym zamyka połączenie wyjścia y z ciśnieniem p_z, a łączy z ciśnieniem atmosferycznym reprezentującym sygnał „0". W elemencie realizującym funkcję alternatywy (rys. 13.8b) przy wszystkich sygnałach x równych zeru wyjście y jest połączone z atmo-

*) Opracowanego w Instytucie Automatyki Przemysłowej Politechniki Warszawakiej

sfera. („0”), a przy którymkolwiek sygnale (lub kilku sygnałach) x równym jeden otwarte zostaje połączenie wyjścia y z ciśnieniem p_z („1”).

Oprócz przedstawionych na rys. 13.8 elementów czterowejściowych produkowane są odmiany trój-, dwu- i jednowejściowe, te ostatnie realizują funkcje negacji i powtórzenia. Łatwo spostrzec, że obydwa typy elementów podstawowych jak i odmiany o mniejszej liczbie wejść budowane są z tych samych



Rys. 13.8. Schematy budowy, równania funkcji logicznych i charakterystyki statyczne elementów membranowych systemu MERALOG: a) negacji alternatywy, b) alternatywy

części składowych, a więc rozszerzenie asortymentu elementów w stosunku do wymogów minimalnego systemu funkcjonalnie pełnego w małym stopniu utrudnia produkcję, a ułatwia projektowanie i pozwala często zmniejszyć liczbę elementów potrzebnych do budowy bardziej złożonych układów logicznych.

Dodatkowe możliwości funkcjonalne pojawiają się, gdy zamiast ciśnienia zasilania p_z wprowadzony zostanie jeszcze jeden sygnał wejściowy x_0. Element z rys. 13.8a realizuje wówczas funkcje,

$$y = x_0 \overline{(x_1 + x_2 + x_3 + x_4)},$$

natomiast element z rys. 13.8b

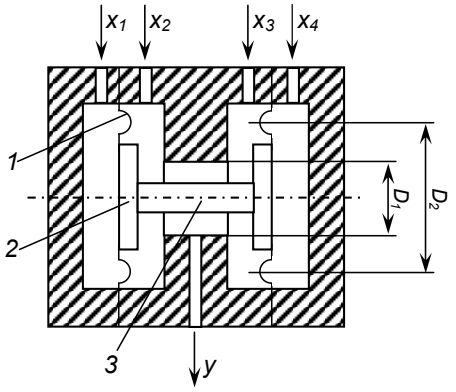
$$y = x_0(x_1 + x_2 + x_3 + x_4),$$

ale elementy stają się wówczas bierne (pasywne) i trzeba starannie przeanalizować, czy obciążenie sygnału x_0 jest dopuszczalne. W wersji podstawowej elementy pracują. jako czynne (aktywne), tzn. sygnały wejściowe x oddzielone są energetycznie od wyjścia, a sygnał y powstaje przez połączenie wyjścia z ciśnieniem zasilania lub z atmosferą.

Podstawowe dane techniczno elementów systemu MERALOG:

ciśnienie zasilania p_z= 1-1,4 kG/cm^2 (okolo 100÷140 kN/m^2),
wartość sygnału „0” 0÷0,2 p_z,
wartość sygnału „1” 0,8÷l p_z,
dopuszczalne przeciążenie 2 kG/cm^2 (około 200 kN/m^2),
wartość natężenia przepływu przy spadku ciśnienia Δp = 1 kG/cm^2 (około 100 kN/m^2) $Q_{wy} \geq 1000$ Nl/h,
zakres temperatury otoczenia -30° ÷ +50°C,
dopuszczalne zanieczyszczenia powietrza mniejsze lub równe 40 μm,
czas przeliczania jednego elementu poniżej 2 ms.

Ostatnia informacja pozwala również ocenić czas przełączania całego układu, w którym sygnał przechodzi zwykle przez kilka elementów i znajdujące się pomiędzy nimi linie łączące. Wielkość tego czasu stanowi główne ograniczenie zakresu zastosowań elementów membranowych.



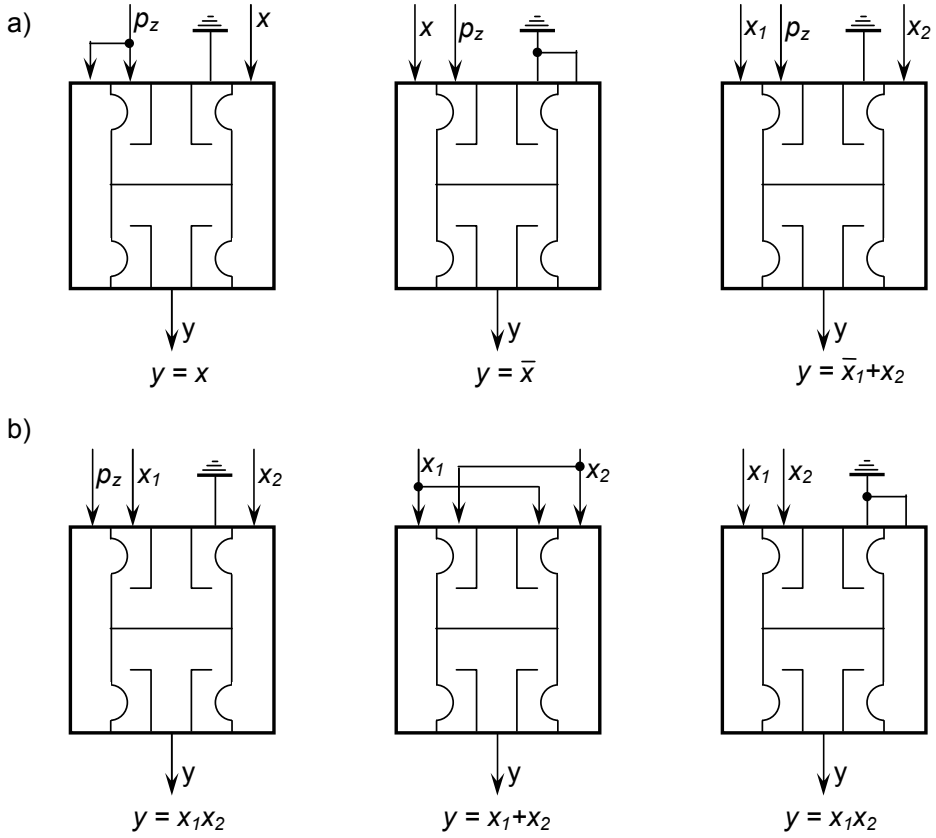
Rys. 13.9. Budowa elementu membranowego systemu DRELOBA

Inne rozwiązanie elementu podstawowego zastosowano w systemie DRDLOBA (XBD) (rys. 13.0). Membrany 1 ze sztywnikami 2 związane trzpieniem 3 tworzą blok wydzielający komory *a, b,* c, *d,* do których wprowadza się sygnały wejściowe oraz ewentualnie ciśnienia zasilania i atmosferyczne. Sygnał wyjściowy y pobierany jest z środkowej części korpusu. Wartość ciśnienia sygnału wejściowego, przy której zachodzi proces przełączania, zależy od sto-

sunku powierzchni efektywnych wytoczenia w środkowej części korpusu i membrany

$$\frac{\pi D_1^2 / 4}{\pi D_2^2 / 4} = 0{,}5$$

a więc wynosi około $0{,}5p_z$. Charakterystyki statyczne mają kształt podobny do przedstawionego na rys. 13.8.



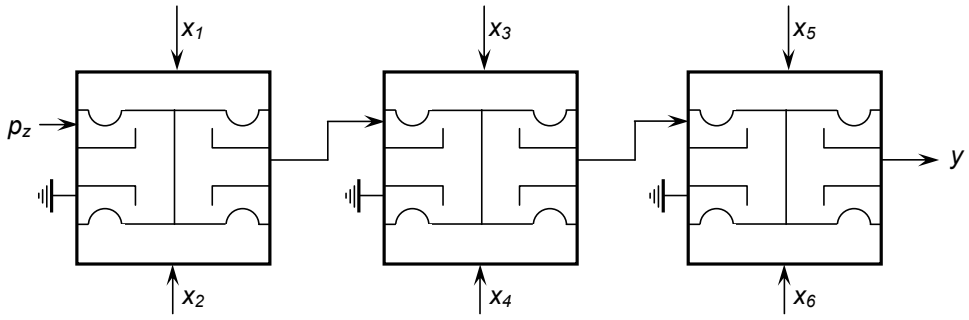
Rys. 13.10. Przykłady realizacji funkcji logicznych za pomocą elementu membranowego systemu DRELOBA

Pojedynczy element realizować może, zależnie od podłączenia, następujące funkcje jedno wyjściowe lub dwuwejściowe (rys. 13.10):

a) powtórzenie, negacja, implikacja — element czynny,

b) koniunkcja, alternatywa, zakaz — element bierny.

Funkcje większej liczby wejść, jak również inne funkcje złożone, otrzymać można łącząc kilka elementów podstawowych. W systemie DRELOBA przewidziano kilka typowych zestawów elementów, które stanowią konstrukcyjną całość i sprzedawane są w postaci gotowych bloków. Budowę jednego z tych



Rys. 13.11. Zestaw elementów systemu DRELOBA

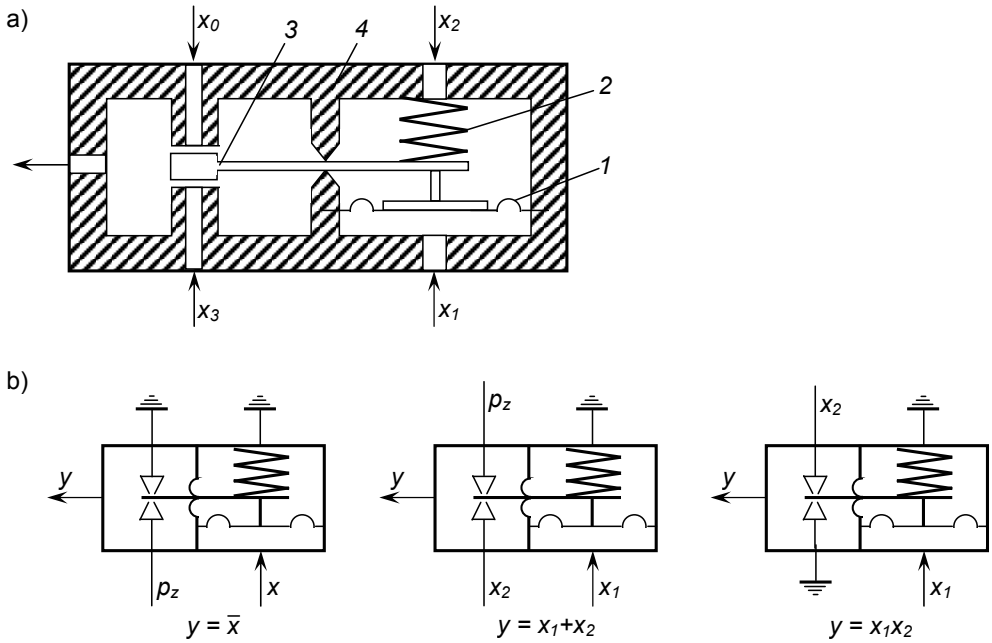
zestawów przedstawiono na rys. 13.11; realizowana przez niego funkcja ma postać

$$y = (\bar{x}_1 + x_2)(\bar{x}_3 + x_4)(\bar{x}_5 + x_6)$$

W przypadku połączenia komór x_2, x_3 *i* x_6 *z* atmosferą, tzn. dla $x_2 = x_4 = x_6 = 0$, zestaw ten realizuje funkcję NOR

$$y = \bar{x}_1 \bar{x}_3 \bar{x}_5 = \overline{x_1 + x_3 + x_5}$$

Jako trzeci przykład budowy elementów membranowych wybrano element podstawowy systemu SAMSOMATIC (rys. 13.12a). Membrana 1 ze sztywni-

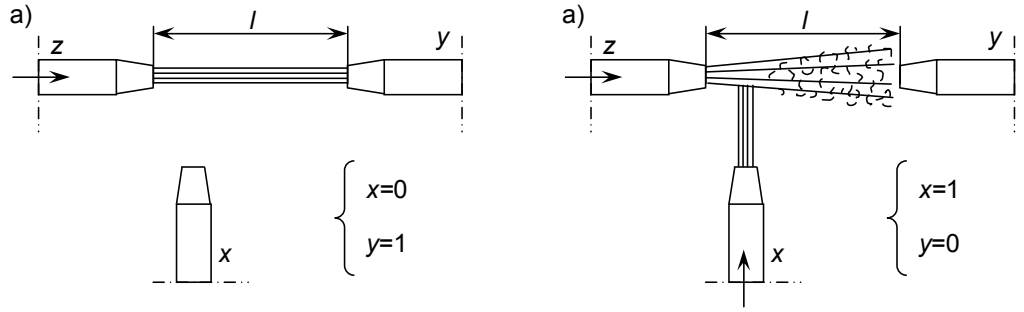


Rys. 13.12. Element membranowy systemu SAMSOMATIC: a) budowa, b) przykłady realizacji funkcji logicznych.

kiem podparta jest sprężyną 2, której napięcie wstępne wyznacza wartość ciśnienia przełączania. Dźwignia 3 ułożyskowana jest w szczelnej przegrodzie 4. Na końcu dźwigni znajduje się płytka rozdzielacza, łącząca wyjście *y z* kanałem x_0 lub x_3. Kilka możliwości realizacji funkcji logicznych za pomocą tego elementu przedstawiono na rys. 13.12b.

13.6.3. Elementy strumieniowe

Są to elementy płynowe (pneumatyczne lub hydrauliczne) bez ruchomych części mechanicznych, w których wszelkie zmiany wartości sygnałów wyjściowych następują wskutek wzajemnego oddziaływania strumieni płynu. Charakteryzują się dużą trwałością oraz większą prędkością działania niż elementy membranowe, natomiast mają mniejszą moc sygnałów wyjściowych i wymagają zawsze użycia wzmacniaczy przed elementami wykonawczymi.

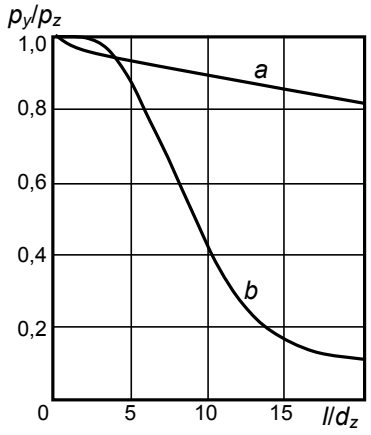


Rys. 13.13. Budowa i działanie elementu turbulentnego: *z* - zasilanie, *x* - sygnał sterujący, *y* - sygnał wyjściowy

Ważną grupę elementów strumieniowych stanowią tzw. elementy turbulentne, których zasada działania, przedstawiona poglądowo na rys. 13.13, polega na zmianie charakteru przepływu strumienia głównego laminarnego w turbulentny pod wpływem oddziaływania strumienia sterującego.

Na rysunku 13.14 podano wykresy odzysku ciśnienia p_y/p_z (przy zerowym natężeniu przepływu na wyjściu) w zależności od odległości l pomiędzy dyszami. Jak widać, przy odległości $l = (15\div20)d_z$, gdzie d_z jest średnicą otworu dyszy zasilania, odzysk ten jest bliski jedności dla przepływu laminarnego i bliski zera dla przepływu turbulentnego. Pozwala to wyraźnie rozróżnić zakres wartości ciśnienia, któremu przypiszemy wartość „1” sygnału wyjściowego, oraz zakres, któremu przypiszemy wartość „0” tego sygnału.

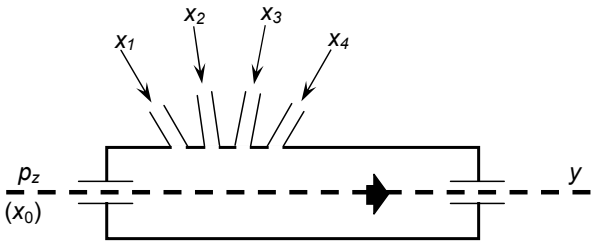
Wybór ciśnienia zasilania elementów turbulentnych jest problemem bardzo istotnym. Zwiększanie tego ciśnienia powoduje zmianę charakteru przepływu z laminarnego na turbulentny (bez oddziaływania strumienia sterującego), a ponadto zwiększa zużycie płyn, natomiast zbyt niskie ciśnienie zasilania



Rys. 13.14. Odzysk ciśnienia p_y/p_z (przy $Q_y = 0$) dla przepływu: *a* — laminarnego, *b* — turbulentnego

utrudnia wysterowanie dalszych elementów układu. Dla elementów turbulentnych pneumatycznych ciśnienia zasilania wynoszą średnio 100÷300 mm H_2O (1÷3 kN/m^2), a dla hydraulicznych 4÷10kG/cm^2 (0,4÷l MN/m^2).

W praktycznie wykorzystywanych turbulentnych elementach logicznych dysz sterujących jest zwykle kilka, co umożliwia realizację bardziej złożonych



Rys. 13.15. Schemat budowy strumieniowego elementu logicznego systemu MAXALOG

funkcji logicznych. Na rysunku 13.15 podano schemat budowy klasycznego elementu turbulentnego systemu MAXALOG, realizującego cztero wejściową funkcję NOR

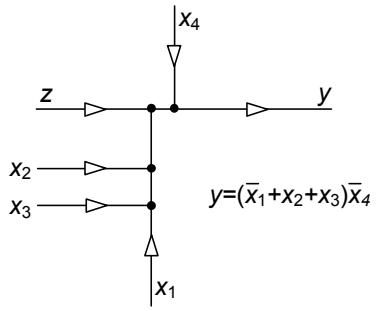
$$y = \overline{x_1 + x_2 + x_3 + x_4},$$

a w przypadku zastąpienia ciśnienia zasilania p_z dodatkowym sygnałem wejściowym x_0

$$y = x_0(\overline{x_1 + x_2 + x_3 + x_4}).$$

Jeszcze większe możliwości funkcjonalne wykazują tzw. kaskadowe strumieniowe elementy logiczne KSEL *), Przykładowy układ dysz elementu kaskadowe-

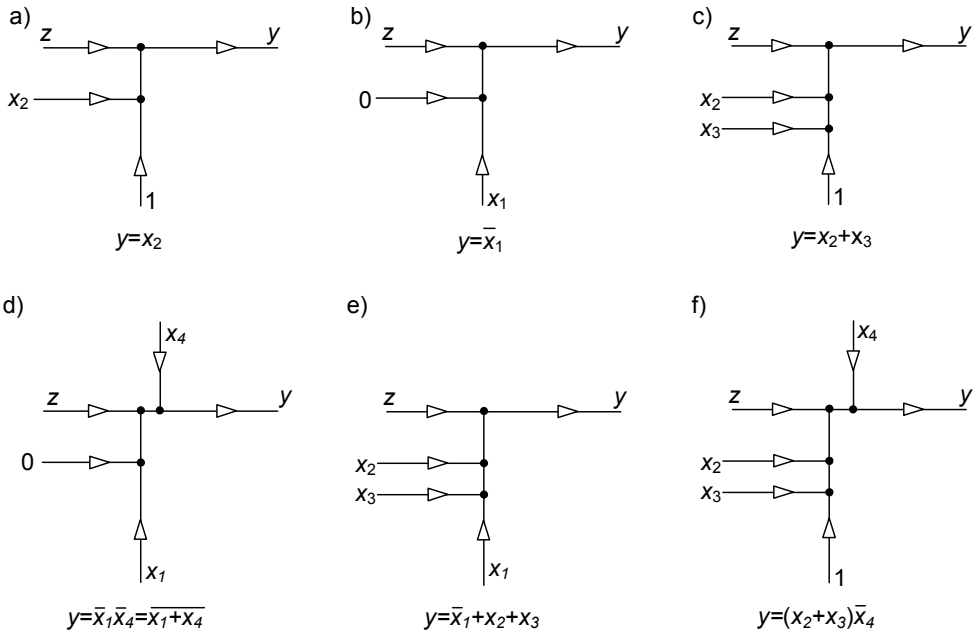
*) Opracowane w Instytucie Automatyki Przemysłowej Politechniki Warszawskiej.



Rys, 13.16. Schemat i równanie funkcji logicznej elementu kaskadowego

go przedstawiono na rys. 13.16. W elemencie tym część strumieni sterujących oddziałuje najpierw kolejno na siebie i dopiero wynik tego oddziaływania skierowany jest na strumień główny. Kilka możliwości realizacji funkcji logicznych za pomocą tego elementu pokazano na rys. 13.17.

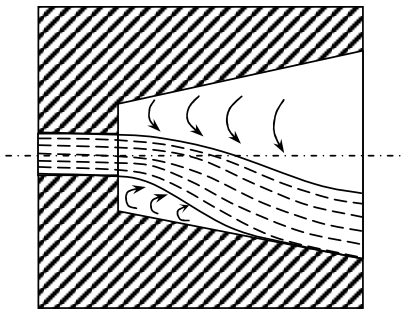
Drugą istotną grupę elementów strumieniowych stanowią, elementy z przylegającym strumieniem. Wykorzystany w nich został tzw. efekt Coandy, to znaczy zjawisko przylegania turbulentnego strumienia głównego do ścianki bocznej kanału elementu (rys. 13.18). Przyleganie to następuje w wyniku podciśnienia pojawiającego się w przestrzeni pomiędzy strumieniem a ścianką,



Rys. 13.17. Przykłady realizacji jeduo-, dwu- i trójargnmentowych funkcji logicznych za pomocą elementu kaskadowego

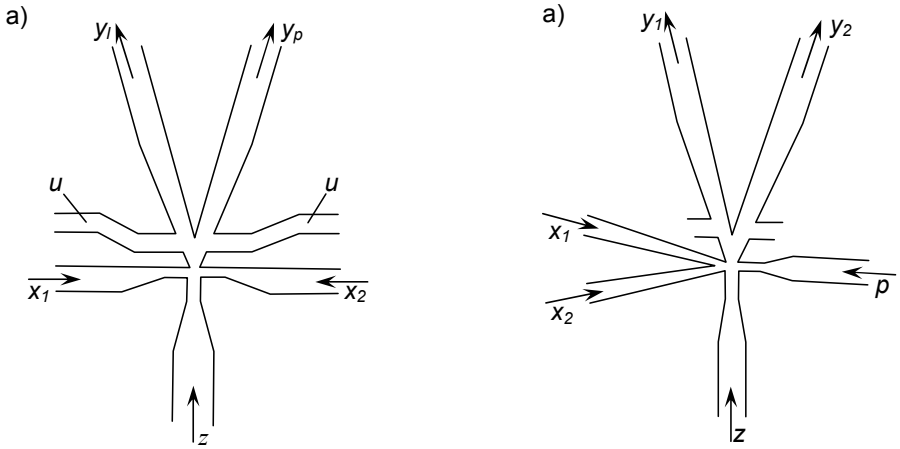
spowodowanego porywaniem cząstek płynu z tej przestrzeni przez strumień główny. Obniżenie ciśnienia jest tym większe, im bliższa jest ścianka, strumień przylgnie więc zawsze do ścianki bliższej.

Na rysunku 13.19a przedstawiono schemat elementu bistabilnego (przerzutnika), którego działanie opiera się na wykorzystaniu efektu Coandy. Kanały



Rys. 13.18. Zachowanie się strumienia turbulentnego w pobliżu ścianki

upustowe *u* pozwalają usunąć wpływ obciążenia na wyjście elementu; w przypadku braku tych kanałów wzrost oporności w jednym kanale wyjściowym może spowodować przerzucenie strumienia do drugiego kanału wyjściowego



Rys. 13.19. Schematy elementów strumieniowych z przylegającym strumieniem: a) przerzutnik, b) element OR/NOR

bez udziału strumieni sterujących. Rysunek 13.19b pokazuje dwuwejściowy element realizujący funkcje alternatywy i negacji alternatywy

$$y_1 = \overline{x_1 + x_2}\,, \qquad y_2 = x_1 + x_2\,.$$

W kanale *p* tego elementu panuje stałe ciśnienie podporowe (przy odpowiednim ukształtowaniu kanałów może to być ciśnienie atmosferyczne).

Ciśnienia zasilania w elementach z przylegającym strumieniem są wyższe niż w elementach turbulentnych i wynoszą (dla elementów pneumatycznych)

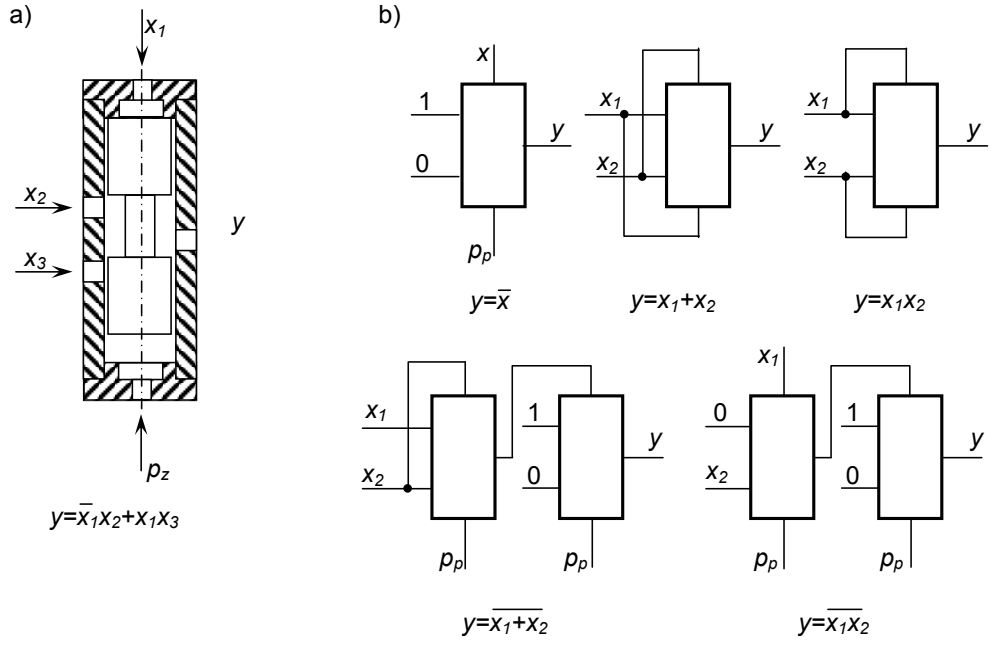
od kilkunastu do kilkudziesięciu KN/m^2. Czas przełączania jednego elementu $t_p = 0{,}5 \div 5$ ms.

Ważnym problemem przy projektowaniu układów złożonych z elementów strumieniowych jest dobór odpowiednich kanałów komunikacyjnych. Muszą one, z jednej strony, minimalizować straty energii strumienia, z drugiej, zapewniać możliwie największą prędkość przesyłania sygnałów, a w niektórych przypadkach ponadto muszą zapewniać określony charakter przepływu, np. laminarny. Zagadnienia te opisane są bliżej w książce [5].

13.6.4. Elementy tłoczkowe

Elementy tłoczkowe zaliczane są do szerszej grupy elementów suwakowych, nazywanych również rozdzielaczami suwakowymi, w których część przełączającą stanowi suwak cylindryczny (przypadek elementów tłoczkowych) lub płaski.

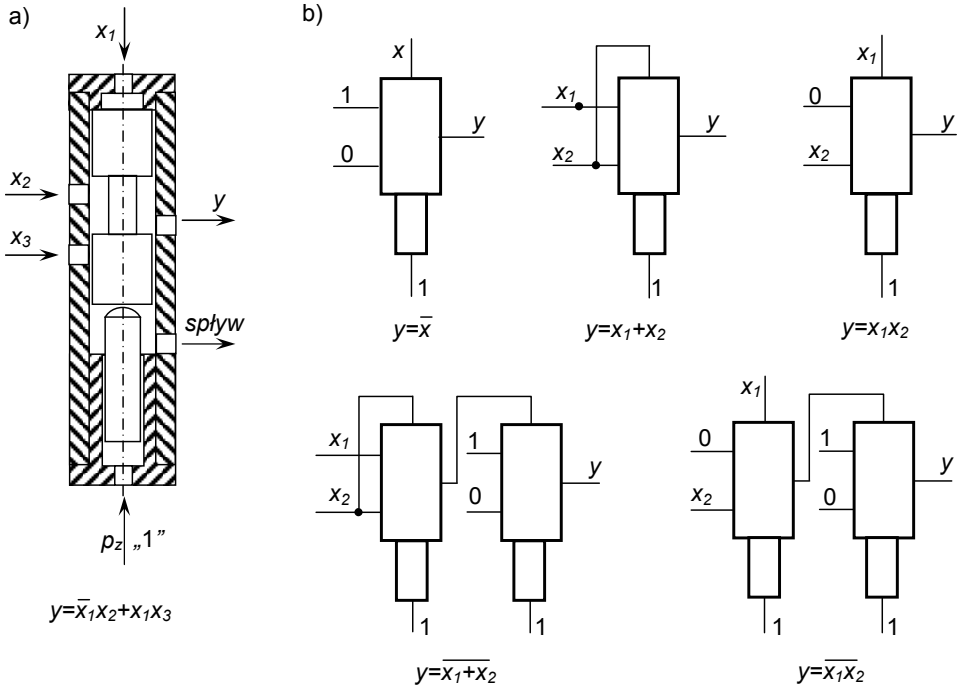
Są to elementy pneumatyczne lub hydrauliczne, które charakteryzują się dużą mocą sygnałów wyjściowych oraz dużą trwałością i niezawodnością działania. Zasadniczą wadą jest natomiast wysoki koszt *), wielokrotnie przekracza-



Rys. 13.20. Schemat budowy tłoczkowego elementu logicznego z ciśnieniem podporowym (a i układy połączeń realizujące podstawowe funkcje logiczne (b)

*) Koszt ten wynika z trudnej technologii: duża dokładność wykonania pary suwakowej, specjalne uszczelnienia, rowki odciążające tłoczek od nacieków promieniowych itd.

jacy koszt identycznych funkcjonalnie elementów półprzewodnikowych, membranowych lub strumieniowych. Wymienione cechy powodują, że zastosowania przemysłowe elementów tłoczkowych dotyczą zwykle układów, których pewność działania musi być bardzo duża, a ponadto wymagane są duże moce sygnałów wyjściowych.



Rys. 13.21. Schemat budowy tłoczkowego elementu logicznego z podwójnym tłoczkiem (a) i układy połączeń realizujące podstawowe funkcje logiczne (b)

Na rysunku 13.20a przedstawiono schemat elementu o jednym, dwóch lub trzech wejściach, z doprowadzonym ciśnieniem podporowym *) p_p równym połowie ciśnienia zasilania p_z, tzn. sygnału „l”. Kilka przykładów realizacji funkcji logicznych za pomocą tego elementu (lub dwóch elementów) pokazano na rys. 13.20b. Warto zwrócić uwagę, że funkcje alternatywy i koniunkcji mogą być realizowane bez doprowadzenia ciśnienia podporowego.

Inne rozwiązanie elementu tłoczkowego, w którym zamiast ciśnienia podporowego wprowadzono dwuczęściowy tłoczek o zróżnicowanych, powierzchniach, przedstawiono na rys. 13.21a. Powierzchnia dolnego tłoczka jest dwukrotnie mniejsza niż górnego, dzięki czemu każdej kombinacji sygnałów wejściowych odpowiada jednoznaczne położenie zespołu tłoczków i określona wartość sygnału

*) Zamiast ciśnienia podporowego można zastosować sprężynę.

wyjściowego. Przykłady realizacji kilku podstawowych funkcji logicznych za pomocą tego elementu pokazano na rys. 13.21 b.

Seria prototypowa tłoczkowych elementów logicznych o schematach według rys. 13.20 i 13.21 została wykonana i zbadana w Przemysłowym Instytucie Automatyki i Pomiarów w Warszawie. Ciśnienia zasilania p_z wynosiły l÷7 MN/m^2 (10 ÷ 70 kG/cm^2), a czas przełączania pojedynczego elementu, zależny od p_z zawierał się w granicach 40÷20 ms. Znane są rozwiązania elementów tłoczkowych, w których ciśnienia zasilania dochodzą do 20 MN/m^2 (200 kG/cm^2), a czas przełączania jest rzędu l÷5ms. Natężenia przepływu oleju na wyjściu elementu wynoszą zwykle od kilkudziesięciu do kilkuset cm^3/s.

Spotyka się również elementy tłoczkowe o większej liczbie przyłączy zewnętrznych i połączeń (dróg) wewnętrznych. Obszerny przegląd rozwiązań konstrukcyjnych i przykłady zastosowań przemysłowych znaleźć można w książce [5].

14. Synteza układów kombinacyjnych

14.1. Uwagi ogólne. Formułowanie zadania układu

W procesie projektowania układów przełączających można wyróżnić kilka zasadniczych etapów, zestawionych niżej.

1. Formułowanie zadania układu.
2. Synteza części centralnej-układu przełączającego, tzn. układu logicznego.
3. Dobór urządzeń wejściowych i wyjściowych, umożliwiających powiązanie układu logicznego z obiektem sterowania; są to urządzenia uzyskiwania, przetwarzania i utrwalania danych po stronie wejść a oraz wzmacniania i przetwarzania sygnałów wykonawczych po stronie wyjścia.
4. Opracowanie schematów montażowych wszystkich części układu.

W niniejszej książce omówione zostaną krótko dwie pierwsze części tego procesu, część trzecia zasygnalizowana zostanie jedynie w kilku przykładach. Pełniejsze omówienie całej problematyki znaleźć można w literaturze specjalistycznej, z tym, że do opanowania części czwartej nie wystarczają wiadomości literaturowe - konieczna jest praktyka projektowa.

Jeżeli układ logiczny jest układem kombinacyjnym, to ogólny przebieg syntezy można pokazać za pomocą schematu podanego na rys. 14.1. W zależności od sposobu sformułowania i złożoności zadania istnieje wiele wariantów prowadzenia syntezy; najbardziej typowe zaznaczono pogrubionymi liniami.

Przed przystąpieniem do syntezy układu należy sporządzić opis procesu sterowanego i określić założenia dla układu sterowania. Czynności te nazwane zostały łącznie *formułowaniem zadania układu;* powinny być wykonywane bardzo starannie i wielokrotnie weryfikowane, gdyż wszelkie błędy lub nieścisłości w założeniach doprowadzą oczywiście do zbudowania niewłaściwego układu.

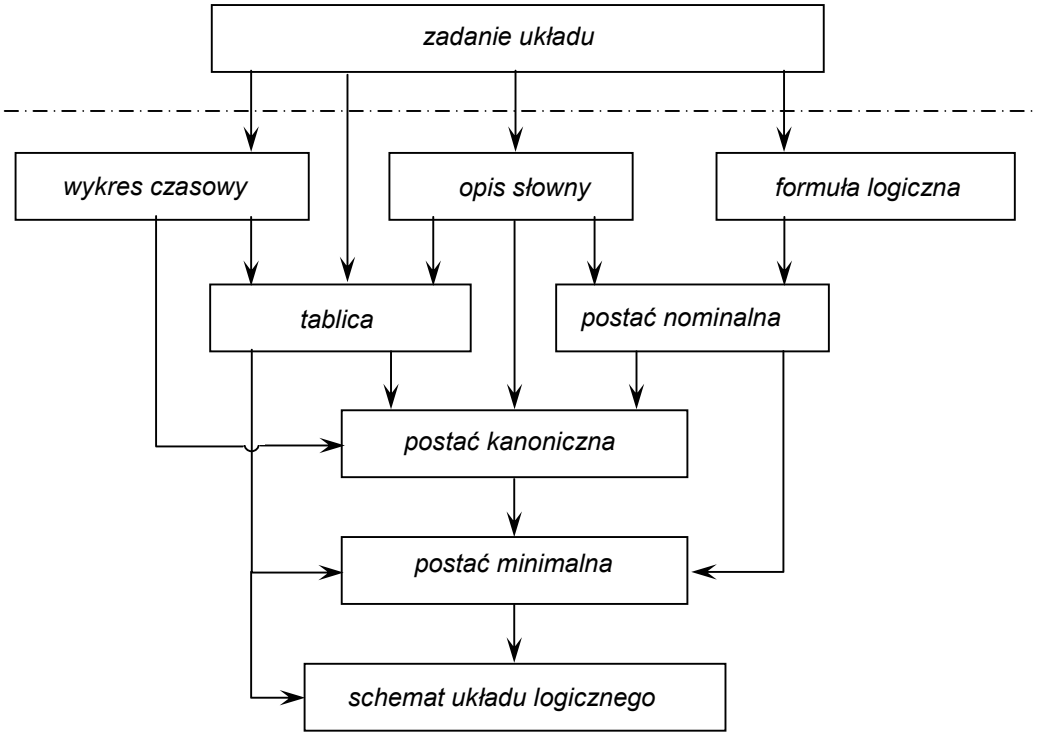
Informacje o procesie sterowanym powinny zawierać co najmniej następujące punkty:

— szkic (schemat) części aparaturowej procesu,

— dane techniczne dotyczące oprzyrządowania procesu (elementy pomiarowe, wykonawcze, ewentualne układy regulacji ciągłej itd.),

— dopuszczanie wartości czasu zadziałania układu (określenie szybkości działania układu),

— ogólne dane dotyczące warunków eksploatacji urządzeń.



Rys. 14.1. Ogólny schemat syntezy układów kombinacyjnych

Założenia dla układu sterowania mogą być sformułowane w następujących postaciach:

a) *opis słowny,* ujmujący zależności pomiędzy wyjściami i wejściami za pomocą zdań o określonej budowie; zwykle są to proste zdania połączone spójnikami „i", „lub" oraz „nie", co pozwala później zapisać zadanie układu za pomocą funkcji koniunkcji, alternatywy oraz negacji;

b) opis graficzny, określający żądany algorytm pracy układu za pomocąwykresów czasowych;

c) *tablica wartości funkcji,* obejmująca zestawienie wszystkich możliwychstanów wejść i odpowiadających tym stanom wartości funkcji;

d) *formuła logiczna* — przypadek rzadki przy projektowaniu układu, spotykany raczej przy analizie istniejących układów.

Przykłady korzystania z wymienionych postaci założeń podane zostaną w dalszych punktach rozdziału.

14.2. Minimalizacja formalna funkcji logicznych

Zasadniczą częścią syntezy kombinacyjnych układów logicznych jest minimalizacja formalna funkcji logicznych, tzn. doprowadzenie funkcji do postaci o możliwie najmniejszej liczbie symboli użytych do jej zapisania (przez symbole rozumiemy tu zarówno symbole zmiennych, jak i symbole działań logicznych). Zagadnienie to jest szczególnie ważne nie tylko, dlatego, że zmniejszając liczbę

elementów użytych do budowy układu obniża się koszt urządzenia, ale przede wszystkim dlatego, że każdemu zmniejszeniu złożoności układu towarzyszy wzrost trwałości i niezawodności.

Minimalizacja formalna musi być następnie uzupełniona minimalizacją układową, w której podstawowym kryterium jest uzyskanie minimalnej liczby elementów logicznych spośród określonego systemu (funkcjonalnie pełnego) postawionego do dyspozycji projektanta.

Należy podkreślić, że w ogólnym przypadku wynikiem minimalizacji formalnej nie jest otrzymanie jednej, określonej postaci funkcji. Postaci minimalnych może być kilka i wybór jednej z nich opiera się na dodatkowych przesłankach wynikających z wymagań minimalizacji układowej.

Omówione zostaną trzy częściej stosowane metody minimalizacji formalnej funkcji logicznych.

14.2.1. Metoda algebraiczna

Metoda ta polega na bezpośrednim przekształcaniu funkcji logicznych na podstawie wszystkich praw i tożsamości algebry Boole'a.

W ogólnym przypadku, dla ustalenia postaci minimalnej należałoby wyszukać najprostszy sposób przedstawienia danej funkcji w postaci superpozycji funkcji dowolnego systemu funkcjonalnie pełnego. Jest to jednak zbyt uciążliwe i najczęściej na etapie minimalizacji formalnej ograniczamy się do wyszukania najprostszej postaci wyrażonej za pomocą funkcji systemu podstawowego, tzn. negacji, alternatywy i koniunkcji.

Metoda algebraiczna pozostawia dość duże pole dla intuicji przeprowadzającego minimalizację, gdyż często mamy możliwość zastosowania różnych wzorów, których efektywność może być odmienna.

W przypadku bardziej złożonych funkcji logicznych algebraiczna metoda minimalizacji jest zbyt pracochłonna, dlatego stosuje się wówczas raczej inne metody, opisane w dalszych punktach. Natomiast ta metoda pozwala niekiedy na dalsze uproszczenie funkcji otrzymanej w wyniku minimalizacji innymi metodami.

Zastosowanie algebraicznej metody minimalizacji zostanie zilustrowane kilkoma przykładami.

Przykład 1. Pierwotną postać funkcji określa formuła

$$y = \overline{(x_1 + x_2)}x_3 + x_2(\bar{x}_1 x_3 + x_2)$$

Stosując prawo de Morgana (13.4) oraz prawa (13.9) i (13.15) możemy przekształcić funkcję do znacznie prostszej postaci

$$y = \bar{x}_1 x_2 x_3 + x_2 \bar{x}_1 x_3 + x_2 x_2 = \bar{x}_1 x_2 x_3 + x_2 x_2 = x$$

Jest rzeczą oczywistą, że wynik ten określa postać minimalną funkcji.

Przykład 2 [1]. Pierwotną postać funkcji określa formuła

$$y = (a+b)[x_1(a+x_2) + \bar{x}_1(a+c) + (x_2+x_3)\bar{x}_3 a]$$

Przekształcimy początkowo wyrażenie w nawiasie kwadratowym. Otrzymamy następujący ciąg przekształceń:

$$\begin{aligned}
&x_1(a+x_2) + \bar{x}_1(a+c) + (x_2+x_3)\bar{x}_3 a = \\
&= x_1 a + x_1 x_2 + \bar{x}_1 a + \bar{x}_1 c + x_2 \bar{x}_3 a + x_3 \bar{x}_3 a = \\
&= a(x_1 + \bar{x}_1) + x_1 x_2 + \bar{x}_1 c + a x_2 \bar{x}_3 + 0 \cdot a = \\
&= a + a x_2 \bar{x}_3 + x_1 x_2 + \bar{x}_1 c = \\
&= a(1 + x_2 \bar{x}_3) + x_1 x_2 + \bar{x}_1 c = \\
&= a + x_1 x_2 + \bar{x}_1 c
\end{aligned}$$

Uproszczone wyrażenie wstawiamy do pierwotnej postaci funkcji

$$y = (a+b)[a + x_1 x_2 + \bar{x}_1 c]$$

i wykonujemy dalsze przekształcenia

$$\begin{aligned}
&y = aa + a x_1 x_2 + a\bar{x}_1 c + ab + b x_1 x_2 + b\bar{x}_1 c = \\
&= a(1 + x_1 x_2 + \bar{x}_1 c + b) + b(x_1 x_2 + \bar{x}_1 c) = \\
&= a + b(x_1 x_2 + \bar{x}_1 c)
\end{aligned}$$

Wyrażenia powyższego nie da się już dalej uprościć, przedstawia ono zatem postać minimalną funkcji.

Przykład 3. Wyznaczyć postać minimalną funkcji określonej za pomocą tablicy przedstawionej na rys. 14.2.

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
y	0	1	1	1	0	0	1	1

Rys. 14.2. Tablica wartości funkcji do przykładu 3

Na podstawie podanej tablicy można napisać równanie funkcji w kanonicznej postaci alternatywnej

$$y = \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3 \qquad (14.1)$$

lub w kanonicznej postaci koniunkcyjnej

$$y = (x_1 + x_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3)\,. \qquad (14.2)$$

Uprościmy najpierw kanoniczną postać alternatywną (14.1). Otrzymamy następujący ciąg przekształceń:

$$y = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2(\bar{x}_3 + x_3) + x_1x_2(\bar{x}_3 + x_3) =$$
$$= \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2(\bar{x}_1 + x_1) = x_2 + \bar{x}_1\bar{x}_2x_3 =$$
$$x_2 + \bar{x}_1x_3$$

W końcowym przekształceniu wykorzystano tożsamość (13.19).

Przekształcając kanoniczną postać koni aukcyjną (14.2) otrzymamy

$$y = (x_1\bar{x}_1 + x_1x_2 + x_1x_3 + x_2x_2 + x_2x_3 + x_3\bar{x}_1 + x_3x_2 + x_3x_3)(\bar{x}_1 + x_2 + \bar{x}_3) =$$
$$= [0 + x_2(x_1 + \bar{x}_1 + x_2 + x_3 + x_3) + x_3(x_1 + \bar{x}_1 + x_3)](\bar{x}_1 + x_2 + \bar{x}_3) =$$
$$= [0 + x_2(x_1 + \bar{x}_1 + x_2 + x_3 + x_3) + x_3(x_1 + \bar{x}_1 + x_3)](\bar{x}_1 + x_2 + \bar{x}_3) =$$
$$= x_2(\bar{x}_1 + x_2 + \bar{x}_3) =$$
$$x_2 + \bar{x}_1x_3$$

W obu przypadkach uproszczone postacie funkcji są jednakowe i nie można ich już dalej skracać. Przedstawiają one postać minimalną funkcji.

14.2.2. Metoda tablic Karnaugha

Minimalizacja funkcji logicznych metodą Karnaugha opiera się na tzw. „regułach sklejania" (por. tożsamości 13.17 i 13.18)

$$Ax + A\bar{x} = A, \tag{14.3}$$

$$(B + x)(B - \overline{x}) = B, \tag{14.4}$$

gdzie *A i B* — zmienne (argumenty) lub funkcje logiczne.

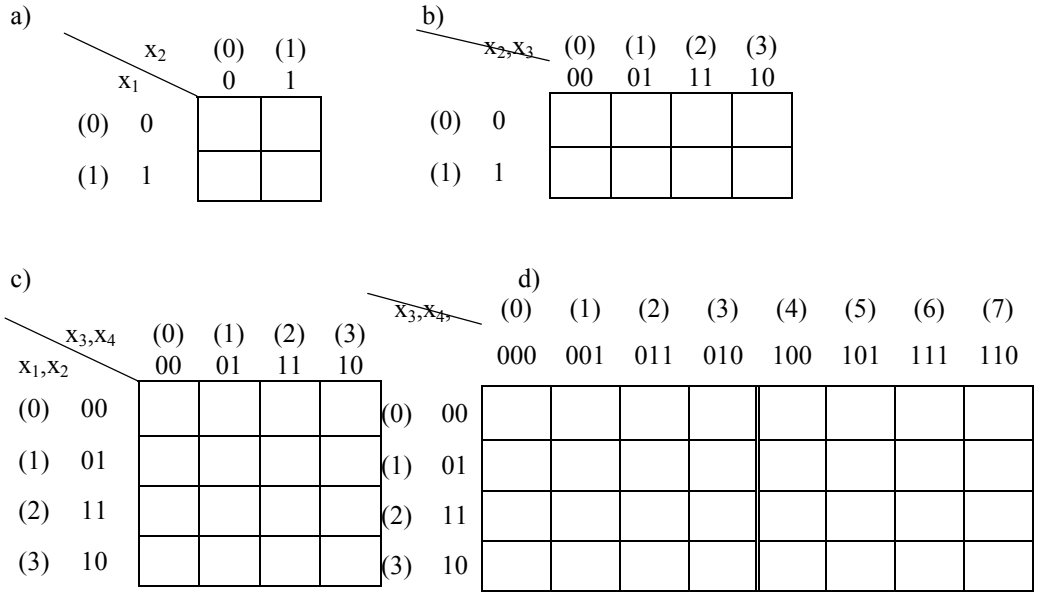
Reguły to wskazują, że jeżeli w normalnej postaci alternatywnej lub koniunkcyjnej funkcji dwa wyrażenia różnią się tylko negacją jednej zmiennej, to funkcja może być uproszczona przez odrzucenie tej zmiennej. Alternatywy lub koniunkcje zmiennych różniące się między sobą negacją jednej zmiennej nazywa się wyrażeniami sąsiednimi. Takimi wyrażeniami są np.

$$x_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 = x_1\bar{x}_2,$$
$$(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + x_3) = \bar{x}_1 + \bar{x}_2.$$

Zasada omawianej metody polega na wyszukaniu wszystkich wyrażeń sąsiednich i sklejaniu ich. Należy się przy tym posługiwać tablicami Karnangha, które są tak zbudowane, że wyrażenia sąsiednie zawsze znajdują się obok siebie lab symetrycznie względem siebie w tablicy (warto zwrócić uwagę, ze sąsiednie kwadraty tablicy różnią się tylko wartością jednej zmiennej).

Tablice Karnaugha stanowią właściwie pewną odmianę tablic wartości funkcji. Budowę tych tablic dla dwu, trzech, czterech i pięciu zmiennych pokazano na rys. 14.3.

W przypadku tablic funkcji dwu, trzech i czterech zmiennych można powiedzieć, że wyrażenia sąsiednie zawsze znajdują się w kwadratach tablicy



Rys. 14.3. Tablice Karnaugha dla funkcji: a) dwu zmiennych, b) trzech zmiennych (zaznaczono kolejność wpisywania wyrazów według ich indeksów dziesiętnych), c) czterech zmiennych, d) pięciu zmiennych

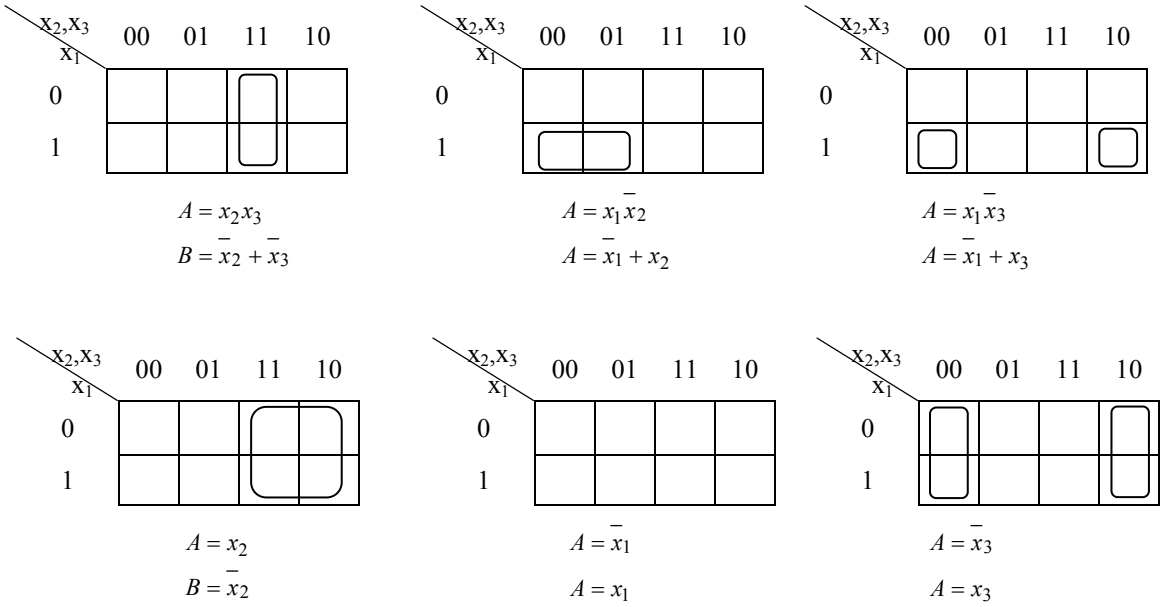
przylegających do siebie całymi bokami, przy czym przeciwległe krawędzie tablicy trzeba również traktować jako wspólną linię przylegania. W przypadku tablic funkcji pięciu zmiennych występuje dodatkowo symetria względem pionowej osi dzielącej tablicę.

Jeżeli funkcja zapisana jest w kanonicznej postaci alternatywnej, to w kwadratach tablicy odpowiadających kolejnym składnikom jedynki wpisuje się symbole l, a w pozostałych symbole 0. Jeżeli funkcja zapisana jest w kanonicznej postaci koniunkcyjnej, to w kwadratach tablicy odpowiadających kolejnym czynnikom zera wpisuje się symbole 0, a w pozostałych symbole 1.

Dla ułatwienia wpisywania funkcji w tablice Karnaugha często podaje się obok symboli zmiennych oznaczenia numeryczne, które pozwalają natychmiast umiejscowić składniki jedynki K_i (według wzoru 13.22) lub czynniki zera D_i (według wzoru 13.24) w tablicy: suma liczb określających wiersz i kolumnę musi być równa indeksowi „*i*”. Na rysunku 14.3 podano w nawiasach oznaczenia numeryczne kolumn i wierszy oraz przykładowo dla tablicy trzech zmiennych

wpisano odpowiednie sumy do poszczególnych kratek. Sumy te określają kolejność wpisywania wyrazów K_i lub D_i według wartości indeksów „i".

Na rysunku 14.4 pokazano możliwości sklejania w tablicach trzech zmiennych. Sklejane kratki tablicy obejmuje się linią, przy czym poszczególne kratki lub grupy kratek mogą być obejmowane wiele razy. Jeżeli wyróżnione kratki



Rys. 14.4. Możliwości sklejania w tablicach trzech zmiennych

zawierają jedynki, to zamiast odpowiadającego im wyrażenia $Ax+A\bar{x}$ można przyjąć A, jeżeli zawierają zera — zamiast $(B+x)(B+\bar{x})$ można przyjąć B. Wartości A i B wpisano obok odpowiednich tablic na rys. 14.4.

Udowodnimy przykładowo wynik odpowiadający rys. 14. 4c. Załóżmy, że w zakreślonych kwadratach występują jedynki. Reprezentują one wyrażenia:

$$x_1\bar{x}_2\bar{x}_3 \quad \text{oraz} \quad x_1x_2\bar{x}_3.$$

Alternatywa tych wyrażeń wyniesie

$$x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 = x_1\bar{x}_3(\bar{x}_2 + x_2) = x_1\bar{x}_3 \cdot 1 = x_1\bar{x}_3$$

Jeżeli w zakreślonych kwadratach występują zera, reprezentują one wyrażenia:

$$(\bar{x}_1 + x_2 + x_3) \quad \text{oraz} \quad (\bar{x}_1 + \bar{x}_2 + x_3).$$

Koniunkcja tych wyrażeń wyniesie

$$\begin{aligned}
&(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3) = \\
&= \bar{x}_1\bar{x}_1 + \bar{x}_1\bar{x}_2 + \bar{x}_2x_3 + x_2\bar{x}_1 + x_2\bar{x}_2 + x_2x_3 + x_3\bar{x}_1 + x_3\bar{x}_2 + x_3x_3 = \\
&= \bar{x}_1\underbrace{(\bar{x}_1 + \bar{x}_2 + x_3 + x_2)}_{=1} + 0 + x_3\underbrace{(x_2 + \bar{x}_1 + \bar{x}_2 + x_3)}_{=1} = \bar{x}_1 + x_3
\end{aligned}$$

Przykład. Za pomocą tablic Karnaugha przeprowadzić minimalizację funkcji trzech zmiennych podanej w przykładzie 3 punktu 14.2.1.

Kanoniczną postać alternatywną (14.1)

$$y = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1x_2\bar{x}_3 + x_1x_2x$$

można zapisać krócej

$$y(x_1, x_2, x_3) = \sum(K_1K_2K_3K_6K_7), \tag{14.5}$$

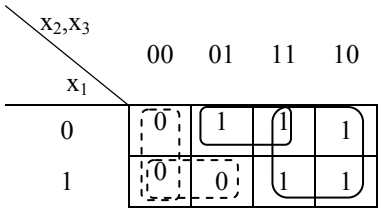
gdzie:

$$K_1 = \bar{x}_1\bar{x}_2x_3,$$
$$K_2 = \bar{x}_1x_2\bar{x}_3,$$
$$K_3 = \bar{x}_1x_2x_3,$$
$$K_6 = x_1x_2\bar{x}_3,$$
$$K_7 = x_1x_2x_3,$$

lub jeszcze krócej

$$y(x_1, x_2, x_3) = \sum(1,2,3,6,7) \tag{14.6}$$

Na podstawie każdej z postaci (14.1), (14.5) lub (14.6) można wpisać do tablicy trzech zmiennych jedynki zgodnie z rys. 14.5, a na pozostałych miejscach zera.



Rys. 14.5. Przykład minimalizacji za pomocą tablicy Karnaugha

Rozpatrzmy teraz kanoniczną postać koniunkcyjną (14.2)

$$y = (x_1 + x_2 + x_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + x_2 + \bar{x}_3)$$

którą można zapisać krócej

$$y(x_1, x_2, x_3) = \prod(D_0D_4D_5), \tag{14.7}$$

gdzie: $D_0 = (x_1 + x_2 + x_3)$,

$$D_4 = (\bar{x}_1 + x_2 + x_3),$$
$$D_5 = (\bar{x}_1 + x_2 + \bar{x}_3),$$

lub jeszcze krócej

$$y(x_1, x_2, x_3) = \prod(0,4,5), \tag{14.8}$$

Na podstawie każdej z postaci (14.2), (14.7) lub (14.8) można wpisać do tablicy zera zgodnie z rys. 14.5, a na pozostałych miejscach jedynki.

Właściwą minimalizację możemy teraz przeprowadzić sklejając jedynki albo zera. W ogólnym przypadku wybiera się zawsze tę możliwość, która daje prostszą postać końcową funkcji. W rozważanym przykładzie pokażemy obydwie możliwości.

Po sklejeniu jedynek (zaznaczonym linią ciągłą na rys. 14.5) funkcję zapiszemy w postaci

$$y = x_2 + \bar{x}_1 x_3,$$

zgodnej z postacią minimalną otrzymaną w punkcie 14.2.1.

Po sklejeniu zer (zaznaczonym linią kreskowaną) funkcję zapiszemy w postaci

$$y = (x_2 + x_3)(\bar{x}_1 + x_2)$$

którą można jeszcze uprościć stosując metodę algebraiczną (znajduje tu potwierdzenie uwaga podana w punkcie 14.2.1). Po wykonaniu elementarnych przekształceń otrzymamy

$$y = x_2\bar{x}_1 + x_2 + x_3\bar{x}_1 + x_3x_2 = x_2(1 + \bar{x}_1 + x_3) + \bar{x}_1x_3 = x_2 + \bar{x}_1x_3$$

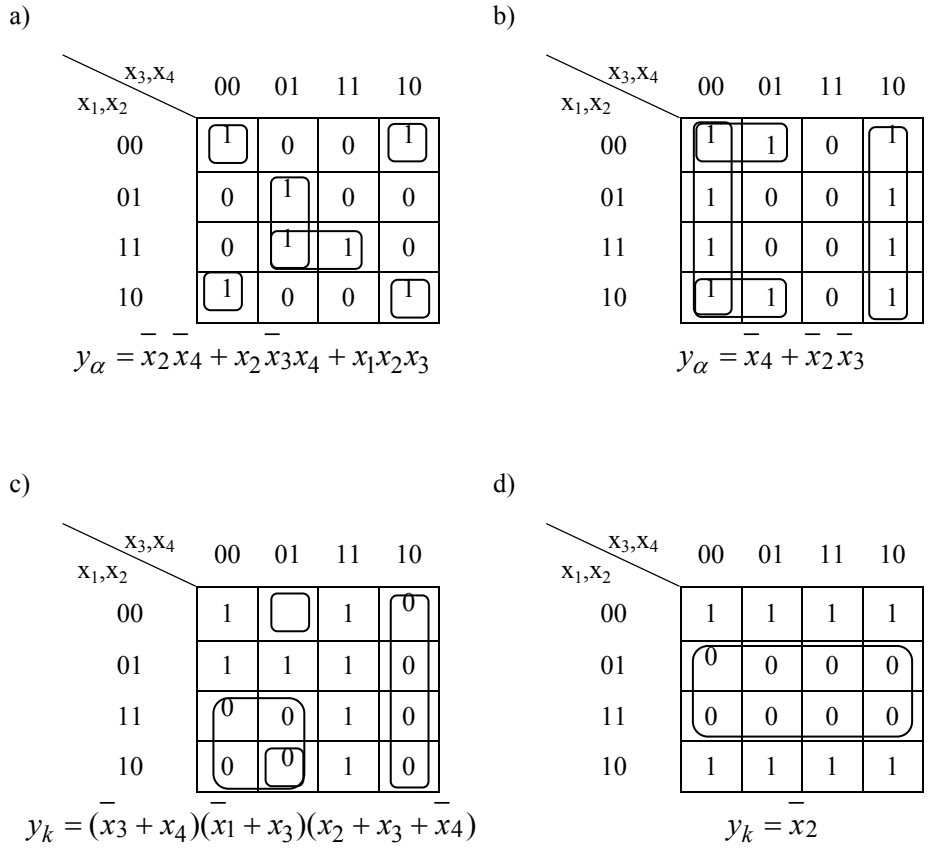
Na rysunku 14.6 zestawiono kilka przykładów obrazujących możliwości sklejania w tablicach czterech zmiennych. Podobnie jak poprzednio, grupę jedynek objętych linią opisuje się koniunkcją elementarną, a grupę zer alternatywą elementarną, w które wchodzą tylko litery o wartościach nie zmieniających się w ramach grupy. Grupa dwukratkowa pozwala usunąć z danego członu funkcji jedną zmienną, czterokratkowa — dwie zmienne, ośmiokratkowa — trzy zmienne itd. Pod każdą tablicą podano zminimalizowaną postać funkcji, przy czym jest to postać alternatywna y_α jeżeli, sklejane były jedynki, a koniunkcyjna y_k, jeżeli sklejane były zera.

Tablicę pięciu zmiennych można podzielić na dwie części, wewnątrz których obowiązują takie same zasady sklejania jak w omówionych tablicach czterech zmiennych, a ponadto sklejane mogą być wszystkie kratki położone symetrycznie względem pionowej osi dzielącej tablicę. Dwa przykłady sklejania pokazano na rys. 14.7. Dla ułatwienia wpisywania funkcji do tablicy i przechodzenia do postaci minimalnej zaznacza się niekiedy dodatkowo klamrami obszary jedynkowe poszczególnych zmiennych, jak na rys. 14.7 a.

Przy większej liczbie zmiennych nie korzysta się już z tablic Karnaugha, gdyż metoda staje się zbyt uciążliwa.

Wpisywanie funkcji do tablicy jest szczególnie proste, gdy jest ona dana w postaci kanonicznej. Jeżeli funkcja jest dana w postaci normalnej, ale nie kanonicznej, człony funkcji zawierające mniejszą liczbę zmiennych trzeba traktować jako grupy kratek, a nie jako pojedyncze kratki. Na przykład w wyrażeniu

$$y = f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_3$$



Rys. 14.6. Przykłady sklejania w tablicach czterech zmiennych

trzeci składnik reprezentuje dwie kratki w tablicy, a czwarty — cztery kratki (rys. 14.8a). Po sklejeniu jedynek według rys. 14.8b otrzymamy

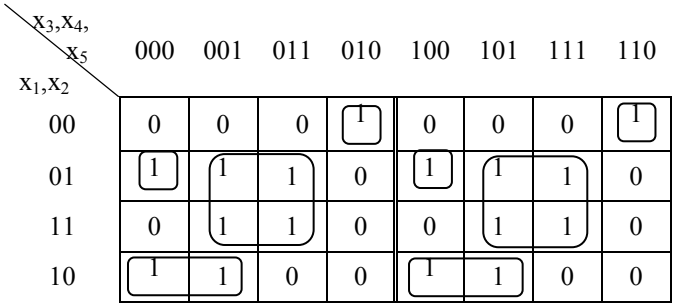
$$y = f(x_1, x_2, x_3, x_4) = \bar{x}_1x_3 + \bar{x}_1\bar{x}_4 + x_2\bar{x}_4 = \bar{x}_1(x_3 + \bar{x}_4) + x_2\bar{x}_4$$

W omawianych przykładach zakładano dotychczas, że wszystkie kratki tablicy są jednoznacznie określone za pomocą symboli 0 lub l i w tych kratkach, gdzie funkcja nie przyjmowała wartości l, wpisywano wartość 0 lub odwrotnie. Tymczasem spotyka się również układy sterowania, w których pewne kombinacje stanów elementów wejściowych nie mogą nigdy wystąpić. Takie kombinacje stanów nazywamy obojętnymi lub nieokreślonymi i zaznaczamy w tablicy za pomocą kresek, a funkcje noszą wówczas nazwę niepełnych. Symbole nieokreśloności można dowolnie zastępować jedynkami lub zerami (jest to obojętne, ponieważ odpowiednie kombinacje stanów i tak nie wystąpią w rozpatrywanym procesie), co umożliwia tworzenie większych grup jedynek lub zer, a przez to upraszcza postać minimalną.

Przykład tablicy funkcji niepełnej przedstawiono na rys. 14.9a. Bez uwzględnienia pozycji nieokreślonych funkcję tę można na podstawie rys. 14.9b zapisać w postaci

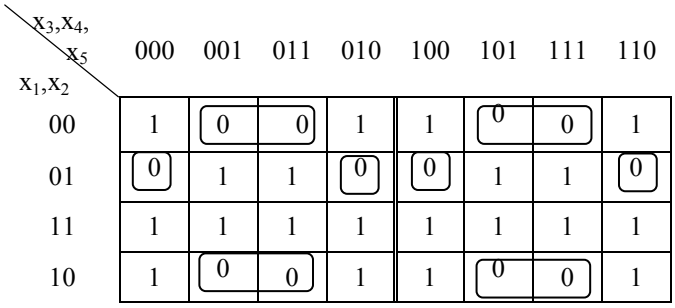
$$y = x_1x_2\bar{x}_4 + x_1\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1x_2x_3x\,.$$

a)



$$y_\alpha = x_2x_5 + x_1\bar{x}_2\bar{x}_4 + \bar{x}_1\bar{x}_2x_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_4\bar{x}_5$$

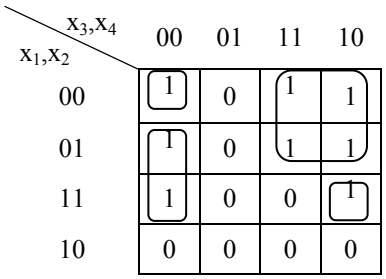
b)



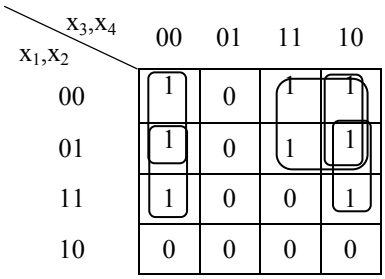
$$y_k = (x_2 + \bar{x}_5)(x_1 + \bar{x}_2 + x_5)$$

Rys. 14.7. Przykłady sklejania w tablicach pięciu zmiennych

a)



b)



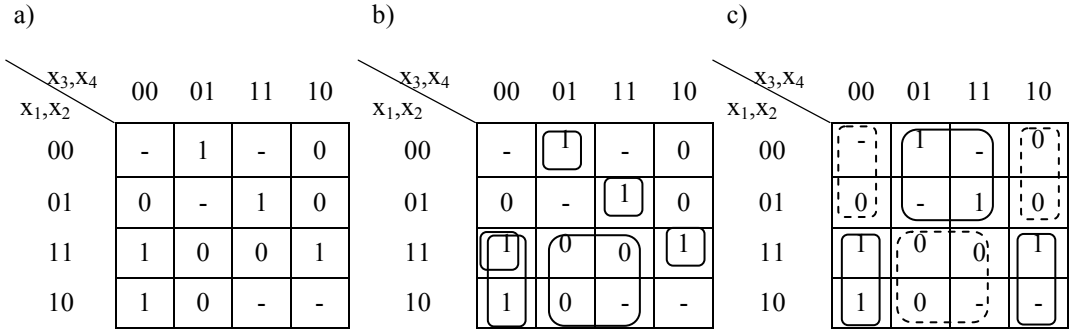
Rys. 14.8. Przykład wpisywania funkcji do tablicy (a) i przechodzenia do postaci minimalnej (b)

Traktując część pozycji nieokreślonych jako jedynki, według rys. 14.9c, otrzymamy znacznie prostszą postać funkcji

$$y = \bar{x}_1x_4 + x_1\bar{x}_4$$

Równie prosty wynik otrzymamy sklejając zera w sposób zaznaczony linią kreskowaną na rys. 14,9 c

$$y = (x_1 + x_4)(\bar{x}_1 + \bar{x}_4) = x_1\bar{x}_4 + x_4\bar{x}$$



Rys. 14.9. Przykład minimalizacji funkcji niepełnej

14.2.3. Metoda Quine'a-McCluskeya

Metoda Quine'a, a zwłaszcza wariant tej metody opracowany przez MeCluskeya, mają oprogramowanie pozwalające minimalizować funkcje logiczne przy użyciu komputera.

Omówione zostanie bliżej zastosowanie metody Quine'a do funkcji przedstawionych w normalnej postaci alternatywnej. Końcowym celem tej metody jest znalezienie minimalnego zestawu tzw. prostych implikantów funkcji.

Implikantem g funkcji *f* nazywa się funkcję mającą tę własność, że wszystkim możliwym kombinacjom argumentów, dla których 0=1, odpowiada również *f*=1. Implikant *g* pokrywa więc (część lub wszystkie) jedynki funkcji *f*.

Prostym implikantem funkcji *f* nazywamy elementarną, koniunkcję będącą implikantem, która zmniejszona o dowolną literę przestaje być implikantem.

Ponieważ każdą funkcję można przedstawić w kanonicznej postaci alternatywnej, obejmującej wszystkie kombinacje (koniunkcje) argumentów, dla których *f*=l, można więc również przedstawić ją w postaci alternatywy implikantów prostych, pokrywających wszystkie jedynki funkcji.

Algorytm metody Quine'a przewiduje następującą kolejność operacji:

1) doprowadzić funkcję do kanonicznej postaci alternatywnej;

2) wykonać wszystkie możliwe operacje niepełnego sklejania, tzn. operacje typu

$$Ax + A\bar{x} = Ax + A\bar{x} + A \tag{14.9}$$

3) wykonać wszystkie możliwe operacje pochłaniania

$$Ax + A\bar{x} = Ax + A\bar{x} + A \tag{14.10}$$

uzyskuje się wówczas tzw. postać skróconą, zawierającą wszystkie implikanty proste funkcji; zwykle nie jest to jednak postać minimalna;

4) za pomocą tablicy implikantów wyszukać minimalny zestaw implikantów prostych, pokrywający wszystkie jedynki funkcji.

Przykład 1. Przeprowadzić minimalizację funkcji

$$y = x_1\bar{x}_2 + x_2\bar{x}_3 + \bar{x}_2x_3 + x_1x_2x_3 + \bar{x}_1x_2x_3 .$$

Postępujemy zgodnie z podanym algorytmem metody Quine'a:

1) $y = x_1\bar{x}_2(x_3 + \bar{x}_3) + x_2\bar{x}_3(x_1 + \bar{x}_1) + \bar{x}_2x_3(x_1 + \bar{x}_1) + x_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 =$

$= x_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2x_3 =$

$\underbrace{x_1x_2\bar{x}_3}_{1} + \underbrace{x_1\bar{x}_2x_3}_{2} + \underbrace{x_1\bar{x}_2\bar{x}_3}_{3} + \underbrace{\bar{x}_1x_2x_3}_{4} + \underbrace{\bar{x}_1x_2\bar{x}_3}_{5} + \underbrace{\bar{x}_1\bar{x}_2x_3}_{6}$

2) $y = \underbrace{x_1x_2\bar{x}_3}_{1} + \underbrace{x_1\bar{x}_2\bar{x}_3}_{3} + x_1\bar{x}_3 + \underbrace{x_1x_2\bar{x}_3}_{1} + \underbrace{\bar{x}_1x_2\bar{x}_3}_{5} + x_2\bar{x}_3 +$

$+ \underbrace{x_1\bar{x}_2x_3}_{2} + \underbrace{x_1\bar{x}_2\bar{x}_3}_{3} + x_1\bar{x}_2 + \underbrace{x_1\bar{x}_2x_3}_{2} + \underbrace{\bar{x}_1\bar{x}_2x_3}_{6} + \bar{x}_2x_3 +$

$+ \underbrace{\bar{x}_1x_2x_3}_{4} + \underbrace{\bar{x}_1x_2\bar{x}_3}_{5} + \bar{x}_1x_2 + \underbrace{\bar{x}_1x_2x_3}_{4} + \underbrace{\bar{x}_1\bar{x}_2x_3}_{6} + \bar{x}_1x_3;$

3) $y = x_1\bar{x}_3 + x_2\bar{x}_3 + x_1\bar{x}_2 + \bar{x}_2x_3 + \bar{x}_1x_2 + \bar{x}_1x_3 ;$

4) Tablica implikantów podana jest na rys. 14.10, wybieramy z niej minimalną liczbę implikantów prostych pochłaniających wszystkie składniki kano-

	Proste implikanty	Składniki kanonicznej postaci alternatywnej					
		$x_1x_2\bar{x}_3$	$x_1\bar{x}_2x_3$	$x_1\bar{x}_2\bar{x}_3$	$\bar{x}_1x_2x_3$	$\bar{x}_1x_2\bar{x}_3$	$\bar{x}_1\bar{x}_2x_3$
I	$x_1\bar{x}_3$	×		×			
II	$x_2\bar{x}_3$	×				×	
II	$x_1\bar{x}_2$		×	×			
I	$\bar{x}_2x_3$		×				×
I	$\bar{x}_1x_2$				×	×	
II	$\bar{x}_1x_3$				×		×

Rys. 14.10. Tablica implikantów do przykładu l

nicznej postaci alternatywnej (tzn. wszystkie kolumny tablicy muszą być popokryte znakami x); w danym przykładzie istnieją dwie postaci minimalne:

$$y_I = x_1\bar{x}_3 + \bar{x}_1x_2 + \bar{x}_2x_3, \qquad y_{II} = x_1\bar{x}_2 + \bar{x}_1x_3 + x_2\bar{x}_3$$

Występowanie kilku równoważnych postaci minimalnych może mieć duże znaczenie w przypadku ograniczeń obciążalności niektórych elementów (wybieramy wówczas tę postać, w której elementy najbardziej wrażliwe na obciążenia są najmniejszą liczbę razy wykorzystywane), a także w przypadku układów wielowyjściowych, w których daje się niekiedy wykorzystać niektóre składniki

postaci minimalnej do realizacji innych funkcji. Gdyby np. oprócz funkcji *y* na-leżało zrealizować funkcję

$$z = \bar{x}_1 x_3 + x_2 \bar{x}_3 + \bar{x}_2 x_4,$$

korzystnie byłoby wybrać

$$y_{II} = x_1 \bar{x}_2 + \bar{x}_1 x_3 + x_2 \bar{x}_3,$$

gdyż dwa składniki funkcji *y* oraz *z* byłyby wspólne, co zmniejsza liczbę elementów użytych do budowy układu.

W przypadku funkcji przedstawionej w kanonicznej postaci koniankcyjnej tok postępowania według metody Quine'a będzie podobny. Szukamy wówczas minimalnej postaci koniunkcyjnej tzw. *prostych implicentów* funkcji, przy czym prosty implicent *h* jest to elementarna alternatywa mająca tę własność, że gdy *h*=0, wówczas i *f*=0. Niepełne sklejanie wykonuje się wtedy według zależności

$$(A + x)(A + \bar{x}) = (A + x)(A + \bar{x})A, \tag{14.11}$$

natomiast pochłanianie

$$A(A + x) = A(A + \bar{x}) = A, \tag{14.12}$$

Opracowana przez McCluskeya odmiana metody Qnine'a polega na wprowadzeniu oznaczeń binarnych i uproszczeniu zapisu kolejnych faz sklejania i pochłaniania.

Dla funkcji przedstawionych w kanonicznej postaci alternatywnej *algorytm metody McCluskeya* jest następujący:

1. Wszystkie koniunkcje elementarne (składniki jedynki) zapisuje się formie liczb binarnych, pisząc l zamiast x_i oraz 0 zamiast $\bar{x}_i$.

2. Uzyskane liczby binarne zestawia się w kolumnie, dzieląc je na grupy o jednakowych indeksach, przy czym indeks oznacza ilość jedynek w liczbie binarnej. Na przykład liczba 1000 ma indeks l, a liczba 1011 ma indeks 3.

3. Przeprowadza się pierwszą operację sklejania, przestrzegając następujących zasad:

a) sklejane mogą być wyrażenia należące do sąsiadujących grup (tzn. grup o indeksach różniących się o 1) i różniące się tylko na jednej pozycji,

b) każde wyrażenie może być sklejane dowolną liczbę razy,

c) sklejenie wyrażeń *a*0*c* i *a*1*c* daje w wyniku *a*-*c*,

d) wyniki sklejeń zestawia się w następnej kolumnie, grupując je ponownie według indeksów. Są to indeksy równe mniejszym indeksom sklejanych wyrażeń.

4. Przeprowadza się kolejne operacje sklejania, pamiętając o tym (oprócz zasad podanych w punkcie 3), że sklejane wyrażenia muszą mieć kreski na tych samych pozycjach.

5. Po wyczerpaniu wszystkich możliwych sklejeń otrzymane wyrażenia zamienia się ponownie na formę literową, opuszczając kreski. Na przykład (11-0) oznacza $x_1 x_2 \bar{x}_4$. Uzyskuje się w ten sposób wszystkie implikanty proste funkcji.

6. Za pomocą tablicy implikantów wyszukuje się minimalny zestaw, implikantów prostych, pokrywających wszystkie jedynki funkcji.

W przypadku funkcji przedstawionej w kanonicznej postaci koniunkcyjnej tok postępowania jest podobny. Wszystkie alternatywy elementarne (czynniki zera) zapisu je się również w formie liczb binarnych, ale wówczas l występuje zamiast $\bar{x}_i$ a 0 zamiast x_i. W końcowej fazie otrzymuje się wszystkie implicenty proste funkcji, a następnie, za pomocą tablicy implicentów, wyszukuje się postać minimalną (por. przykład 2).

Przykład 2. Przeprowadzić minimalizację funkcji

$$y = (x_1 + x_2 + x_3 + x_4)(x_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4) \cdot$$
$$\cdot (\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + x_4)(\bar{x}_1 + x_2 + \bar{x}_3 + x_4)$$

Wprowadzone niżej w operacjach 2 1 3 numerowanie liczb oraz wypisywanie indeksów mają na celu jodynie ułatwienie sprawdzenia zgodności postępowania z podanym algorytmem metody McCluskeya. Dalej opuszcza się je, zaznaczając jedynie znakiem ∨ fakt sklejenia liczby.

Operacja l

0000	
0011	
0010	
1111	
1011	
1110	
1010	

Operacja 2

nr	liczba	indeks
1	0000	0
2	0010	1
3	0011	2
4	1010	
5	1110	3
6	1011	
7	1111	4

Operacja 3

nr sklejanych liczb	wynik sklejania	indeks
1 + 2	00-0	0
2 + 3	001- V	1
2 + 4	-010 V	
3 + 6	-011 V	2
4 + 5	1-10 V	
4 + 6	101- V	
5 + 7	111- V	3
6 + 7	1-11 V	

Operacja 4 *)

00-0
-01-
1-1-

Operacja 5

$(x_1 + x_2 + x_4)$
$(x_2 + \bar{x}_3)$
$(\bar{x}_1 + \bar{x}_3)$

*) Pierwsze wyrażenie tej kolumny jest przepisane bez zmian z kolumny poprzedniej (nie dało się z niczym skleić), natomiast następne są wynikiem sklejania pozycji oznaczonych symbolem ∨. Jeżeli wyniki sklejania powtarzają się, należy wypisać je tylko raz.

Operacja 6

Tablicę implicentów przedstawiono na rys. 14.11. Wynika z niej, że wszystkie trzy implicenty są w tym przypadku niezbędne do pokrycia zer funkcji. Minimalna postać koniunkcyjna będzie więc następująca:

$$y = (x_1 + x_2 + x_4)(x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_3)$$

Gdyby minimalizowana funkcja zawierała stany obojętne (nieokreślone), tok postępowania byłby bardzo podobny. W pierwszą kolumnę należy wówczas wpisać również liczby binarne odpowiadające tym stanom, etapy 1-5 przebiegają zgodnie z podanym algorytmem i dopiero w tablicy implikantów (lub implicentów) pomija się składniki (lub czynniki) oznaczające stany obojętne.

Proste implikanty	Składniki kanonicznej postaci alternatywnej						
	$x_1 + x_2 + x_3 + x_4$	$x_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$x_1 + x_2 + \bar{x}_3 + x_4$	$\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4$	$\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + x_4$	$\bar{x}_1 + x_2 + \bar{x}_3 + x_4$
$x_1 + x_2 + x_4$	×		×				
$x_2 + \bar{x}_3$		×	×		×		×
$\bar{x}_1 + \bar{x}_3$				×	×	×	×

Rys. 14.11. Tablica implicentów do przykładu 2

Opracowana została również odmiana metody Mc CIuskeya, stosowana do minimalizacji funkcji przedstawionych w skróconym zapisie dziesiętnym, pokazanym już w punkcie 14.2.2 (wzory 14.6 i 14.8). Obowiązuje wówczas kolejność operacji podana niżej.

1. Poszczególnym liczbom dziesiętnym przypisuje się indeksy, oznaczające ilość jedynek w dwójkowym zapisie tych liczb. Liczby zestawia się w kolumnie, grupując je według indeksów.

2. Przeprowadza się pierwszą operację sklejania według następujących zasad:

a) sklejane mogą być tylko liczby o indeksach różniących się o l, same liczby muszą się różnić o 2^k (k = 0, l, 2, ...) oraz liczba o indeksie większym musi być większa;

b) wynikiem sklejenia *A i B* jest wyrażenie *A*, *B* (*C*), gdzie *C=B - A*; np. 2 i 10 daje 2,10 (8);

c) każda liczba może być sklejana dowolną ilość razy;

d) należy wyczerpać wszystkie możliwości sklejeń.

3. Przeprowadza się następne operacje sklejania, zgodnie z zasadami podanymi w punkcie 2 oraz dodatkowo:

a) różnice umieszczone w nawiasach muszą być przy sklejaniu jednakowe;

b) wynikiem sklejenia *A*, *B* (*C*) oraz *D, E* (*C*) jest *A, B, D, E* (*C, F*) lub *A, D, B, E* (*F, C*), gdzie *F = D-A = E-B*;

c) jeżeli jakieś wyrażenie powtarza się kilkakrotnie, to wpisuje się je tylko jeden raz.

4. Wyrażenia, których nie można już dalej sklejać, odpowiadają prostym implikantom (lub implicentom) funkcji. Na podstawie tablicy implikantów (lub implicentów) usuwa się spośród nich wyrażenia zbędne i wyszukuje się postaci minimalne.

5. Wyrażenia wchodzące do postaci minimalnej zamienia się na formę binarną, a następnie literową, zgodnie z następującymi regułami:

a) wypisuje się w formie binarnej pierwszą liczbę wchodzącą w skład implikantu (implicentu); np. dla implikantu 4, 5, 12, 13 (l, 8) należy wypisać w formie binarnej liczbę 4, tj. 0100;

b) na pozycjach określonych przez różnice podane w nawiasie, występujące jako wagi pozycji, zastępuje się zera kreskami, tj. -10- (gdyż l i 8 reprezentują pozycje pierwszą i czwartą, licząc od końca);

c) jedynkom na poszczególnych pozycjach przypisuje się odpowiednie litery, a zerom — negacje liter, natomiast kreski pomija się; w przytoczonym wyżej przykładzie: $x_2 \bar{x}_3$.

Przykład 3. Przeprowadzić minimalizację funkcji przedstawionej w skróconym zapisie dziesiętnym:

$$y(x_1, x_2, x_3, x_4) = \sum_{\text{indeksy}} (1, \underset{1}{4}, \underset{1}{5}, \underset{2}{6}, \underset{2}{7}, \underset{3}{8}, \underset{1}{9}, \underset{2}{12}, \underset{2}{(13)}, \underset{3}{})$$

przy czym składnik podany w nawiasie, (13), oznacza stan obojętny.

Operacja l

1	VV
4	VVV
8	VV
5	VVVV
6	V
9	VVV
12	V
7	VV
13	VVV

Operacja 2

1, 5	(4)	V
1, 9	(8)	V
4, 5	(1)	VV
4, 6	(2)	V
4, 12	(8)	V
8, 9	(1)	V
8, 12	(4)	V
5, 7	(2)	V
5, 13	(8)	V
6, 7	(1)	V
9, 13	(4)	VV
12,13	(1)	VV

Operacja 3

1, 5, 9, 13 (4,8)
4, 5, 6, 7 (1, 2)
4, 5, 12, 13 (1,8)
8, 9, 12, 13 (l, 4)

Uwaga: dla ułatwienia prześledzenia przykładu znak sklejenia V stawiano przy każdej liczbie tyle razy, ile razy była sklejana.

Operacja 4

Tablicę implikantów przedstawiono na rys. 14.12. Składniki postaci minimalnej:

l, 5, 9, 13 (4, 8)
4, 5, 6, 7 (l, 2)
8, 9, 12, 13 (l, 4)

Proste implikanty	Składniki kanonicznej postaci alternatywnej							
	1	4	5	6	7	8	9	12
1, 5, 9, 13 (4, 8)	×		×				×	
4, 5, 6, 7 (1, 2)		×	×	×	×			
4, 5, 12, 13 (1, 8)		×	×					×
8, 9, 12, 13 (1,4)						×	×	×

Rys. 14.12. Tablica impllkantów do przykładu 3

Operacja 5

a) 0 0 0 1
0 1 0 0
1 0 0 0

b) - - 0 1
0 1 - -
1 - 0 -

c) $\bar{x}_3 x_4$
$\bar{x}_1 x_2$
$x_1 \bar{x}_3$

Postać minimalna funkcji:

$$y(x_1, x_2, x_3, x_4) = x_1\bar{x}_3 + \bar{x}_1 x_2 + \bar{x}_3 x_4.$$

Jeżeli minimalizowana funkcja przedstawiona jest w postaci koniunkcyjnej $y = \Pi(a, b, c, \ldots)$, to w operacji 5c zastępuje się symbole 0 symbolem x oraz l symbolem $\bar{x}$, a wynik końcowy ma postać koniunkcyjną.

14.3. Minimalizacja układów wielowyjściowych

Opisane w poprzednim punkcie metody minimalizacji formalnej mają zastosowanie do układów opisywanych za pomocą jednej funkcji logicznej, tzn. układów, w których istnieje jeden sygnał wyjściowy zależny od stanów m elementów wejściowych.

W praktyce bardzo często spotyka się układy wici o wyjściowe, w których zespół m elementów wejściowych steruje pracą n elementów wyjściowych. Układ taki opisany jest za pomocą n funkcji logicznych m-argumentowych. Oddzielna minimalizacja każdej funkcji nie daje zwykle minimalnego wyniku dla całego zespołu n funkcji.

Główny problem minimalizacji układów wielowyjściowych polega na wyszukaniu wspólnych wyrażeń w poszczególnych funkcjach, gdyż wyrażenia te realizowane są później za pomocą jednego bloku elementów „obsługującego" kilka funkcji. Często opłaca się nawet rozbudować postać minimalną niektórych funkcji, jeżeli w rezultacie wystąpią w nich wyrażenia występujące już w innych funkcjach.

Przeanalizujmy na przykład prosty układ dwuwyjściowy, którego funkcje y_1 i y_2 zminimalizowane zostały oddzielnie i mają postać

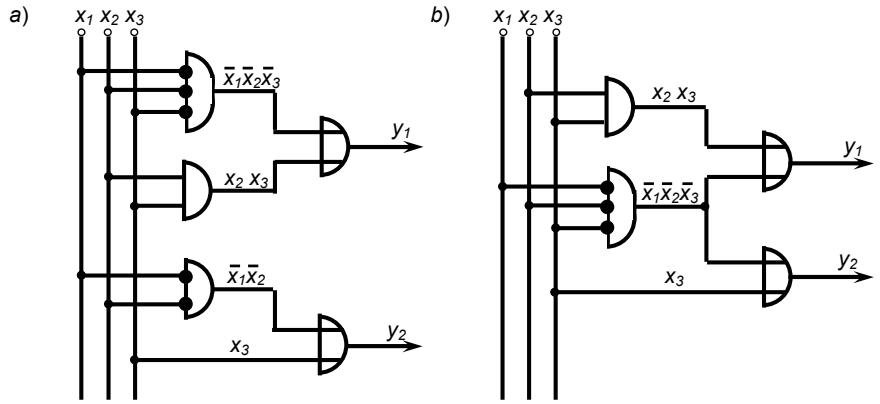
$$y_1 = \bar{x}_1\bar{x}_2\bar{x}_{3+}x_2x_3, \qquad y_2 = \bar{x}_1\bar{x}_2 + x_3.$$

Odpowiedni schemat logiczny układu przedstawiono na rys. 14.13a. Korzy-stając z tożsamości (13.19) możemy funkcję y_z zapisać w postaci

$$y_2 = \bar{x}_1 x_2 \bar{x}_{3+} x_3,$$

co pozwoli uprościć układ zgodnie z rys. 14.13b.

Minimalizację układów wielowyjściowych prowadzić można różnymi metodami ([6], [7]); najczęściej stosowane są pewne modyfikacje metod opisanych w punkcie 14.2. Bliżej przedstawimy obecnie zastosowanie do tego celu udoskonalonej metody McCłuskeya.



Rys. 14.13. Schematy logiczne układu dwuwyjściowego minimalizowanego: a) oddzielnie b) łącznie

Funkcje przedstawia się w skróconym zapisie dziesiętnym i przypisuje się każdej liczbie literowy symbol funkcji, z której pochodzi, np. funkcja

$$y_\alpha(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 = \sum(0,2,10)$$

ma składniki 0*a*, 2*a* i 10*a*. Składniki wszystkich funkcji zestawia się w kolumnie, z podziałem na grupy indeksowe.

Następne kolumny tworzy się ze sklejania liczb kolumny pierwszej, według następujących reguł:

a) w ramach jednej grupy indeksowej skleja się jednakowe liczby o różnych literach, np. 4*a* i 4*c* daje 4*ac*,

b) liczby z sąsiadujących ze sobą grup indeksowych skleja się według zasad metody McCluskeya opisanych w punkcie 14.2.3, jeżeli mają jednakowe symbole literowe.

Wyrażenia, których nie można sklejać, reprezentują proste implikanty, a ich symbol literowy pokazuje, której funkcji dotyczą. Na podstawie tablicy implikantów wyszukuje się postać minimalną i przechodzi na zapis literowy.

W przypadku minimalizacji funkcji przedstawionych w postaci koniunkcyjnej tok postępowania jest identyczny, jedynie w końcowej fazie przechodzenia na zapis literowy występują różnice opisane w punkcie 14.2.3.

Przykład. Przeprowadzić minimalizację układu o trzech wyjściach, opisanego funkcjami

$$y_a(x_1,x_2,x_3,x_4) = \sum(0,1,4,5,9,12,13),$$
$$y_b(x_1,x_2,x_3,x_4) = \sum(0,2,3,5,7,13,15),$$
$$y_c(x_1,x_2,x_3,x_4) = \sum(0,2,5,8,10).$$

Operacja 1		Operacja 2		Operacja 3
0a	∨∨∨	0 (abc)		0 (abc)
0b	∨∨	0, 1 (1a)	∨	0, 2 (2b)
0c	∨∨∨	0, 4 (4a)	∨	0, 1, 4, 5 (l, 4a)
l a	∨∨∨	0, 2 (2b)		0, 2, 8,10 (2, 8c)
4a	∨∨∨	0, 2 (2c)	∨	2 (bc)
2b	∨∨∨	0, 8 (8c)	∨	2, 3 (1b)
2c	∨∨∨	2 (bc)		1, 5, 9,13 (4, 8a)
8c	∨∨	l, 5 (4a)	∨∨	4, 5,12,13 (l,8a)
5a	∨∨∨∨	1, 9 (8a)	∨	5 (abc)
9a	∨∨	4, 5 (1a)	∨∨	3, 7 (4b)
12a	∨∨	4, 12 (8a)	∨	5, 7,13,15 (2, 86)
3b	∨∨	2, 3 (lb)		13 (ab)
5b	∨∨∨	2, 10 (8c)	∨	
5c	∨	8, 10 (2c)	∨	
l0c	∨∨	5 (abc)		
13a	∨∨	5, 13 (8a)	∨∨	
7b	∨∨∨	9, 13 (4a)	∨	
13b	∨∨∨	12,13 (la)	∨	
15b	∨∨	3, 7 (4b)		
		5, 7 (2b)	∨	
		5, 13 (88)	∨	
		13 (ab)		
		7, 15 (8b)	∨	
		13,15 (2b)	∨	

Operacja 4

Tablicę implikantów przedstawiono na rys. 14.14. Składniki postaci minimalnej

0 (abc)	— y_a, y_b
0, 2, 8,10 (2b, 8c)	— y_c
2, 3 (lb)	— y_b

1, 5, 9,13 (4, 8a) — y_a
4, 5,12,13 (1, 8a) — y_a
5 (abc) — y_c
5, 7,13,15 (2, 8b) — y_b

Proste implikanty	Składniki kanonicznej postoci alternatywnej																		
	y_a							y_b							y_c				
	0	y	4	5	9	12	13	0	2	3	5	7	13	15	0	2	5	8	10
0(*abc*)	⊗							⊗							×				
0,2(2*b*)								×	×										
0,1,4,5 (1,4*a*)	×	×	×	×															
0,2,8,10 (2,8*c*)															×	×		×	×
2(*bc*)									×							×			
2,3(1*b*)									×	×									
1,5,9,13 (4,8*a*)		×		×	×		×												
4,5,12,13 (1,3*a*)			×	×		×	×												
5(*abc*)				×							×						⊗		
3,7(4*b*)										×		×							
5,7,13,15(2.8*b*)											×	×	×	×					
13 (*ab*)							×						×						

Rys. 14.14. Tablica implikantów do przykładu

Operacja 5

a)
0000
0000
0010
0001
0100
0101
0101

b)
0000
–0–0
001–
–01–
–10–
0101
–1–1

c)
$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$
$\bar{x}_2\bar{x}_4$
$\bar{x}_1\bar{x}_2x_3$
$\bar{x}_3x_4$
$x_2\bar{x}_3$
$\bar{x}_1x_2\bar{x}_3x_4$
x_2x_4

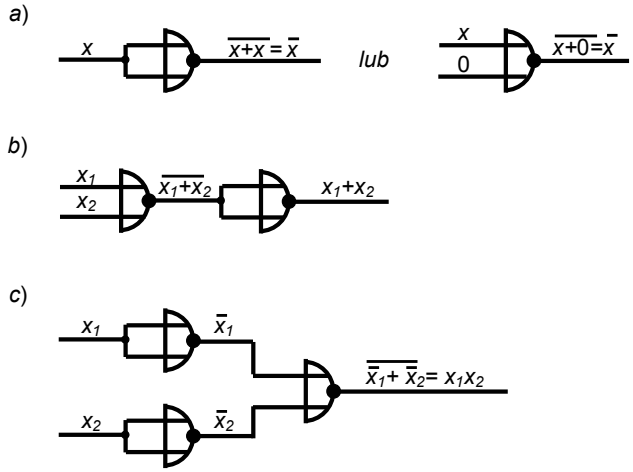
W danym przykładzie udało się znaleźć tylko jeden składnik, $\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$, który będzie jednocześnie wykorzystany do realizacji funkcji y_a i y_b. Równania zminimalizowane (łącznie) poszczególnych funkcji są następujące:

$$y_a = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_3x_4 + x_2\bar{x}_3,$$
$$y_b = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3 + x_2x_4,$$
$$y_c = \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_2\bar{x}_4.$$

14.4. Układy z elementów NOR albo NAND

Uzyskana w wyniku minimalizacji formalnej postać funkcji wyrażona jest za pomocą operacji podstawowego systemu funkcjonalnie pełnego, tzn. negacji, alternatywy i koniunkcji. Jeżeli dysponujemy elementami realizującymi te operacje, to przejście od postaci minimalnej do schematu logicznego jest oczywiste i nie wymaga żadnych przekształceń. Najbardziej rozpowszechnione na rynku są jednak elementy NOR i NAND, zarówno ze względu na swą uniwersalność (każdy z nich stanowi bazę systemu funkcjonalnie pełnego), jak i prostotę budowy.

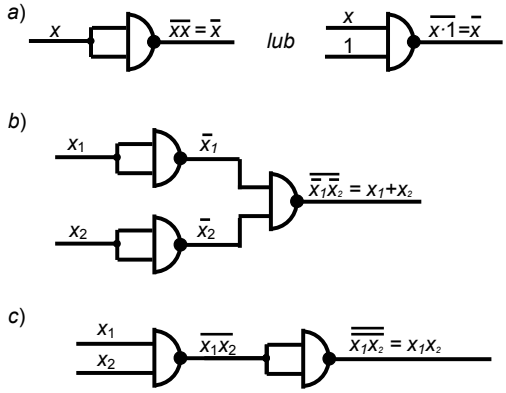
Realizacja operacji negacji, alternatywy i koniunkcji za pomocą elementów NOR albo NAND wynika z wzorów (13.33) lub (13.34) i została przedstawiona na rys. 14.15 i 14.16 na przykładzie elementów dwuwejściowych.



Rys. 14.15. Układy złożone z elementów NOR, realizujące funkcje: a) negacji, b) alternatywy, c) koniunkcji

Jednak bezpośrednie zastąpienie poszczególnych składników postaci minimalnej przez odpowiednie zestawy elementów NOR albo NAND według rys. 14.15 lub 14.16 (tzw. transformacja) zwykle nie prowadzi do rozwiązań minimalnych z punktu widzenia liczby elementów NOR albo NAND użytych do zbudowania układu. Często opłaca się poszukać takiego przekształcenia postaci minimalnej, aby uzyskać formułę zawierającą najmniejszą liczbę żądanych funkcji.

Należy więc przeprowadzić drugi etap minimalizacji, tzw. minimalizację układową. W prostych przypadkach można wykorzystać do tego celu metodę algebraiczną. Zilustruje to podany niżej przykład.



Rys. 14.16. Układy złożono z elementów NAND, realizujące funkcje: a) negacji, b) alternatywy, c) koniunkcji

Przykład. Przeprowadzić minimalizację funkcji

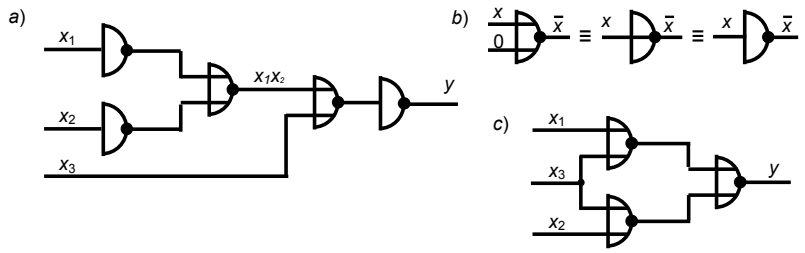
$$y_a(x_1, x_2, x_3) = x_1 x_2 + \overline{x}_1 x_3 + \overline{x}_2 x_3,$$

i narysować schemat układu logicznego realizującego tę funkcję, zbudowanego z elementów NOR.

Minimalizacja formalna funkcji jest bardzo prosta. Otrzymujemy postać mmimalna.

$$y = x_1 x_2 + x_3, \tag{14.13a}$$

której odpowiada schemat logiczny przedstawiony na rys. 14.17a. Na schemacie tym NOR-y realizujące funkcję negacji przedstawiono jako jednowejściowe negatory, gdyż drugie wejście równe stale zeru można traktować jako brak drugiego sygnału wejściowego (rys. 14.17b). Układ ten można jednak



Rys. 14.17. Schematy okładów logicznych do przykładu

zrealizować za pomocą, mniejszej liczby NOR-ów, jeżeli skorzysta się z prawa rozdzielności (13.3a) i wyrazi się postać minimalną za pomocą funkcji NOR:

$$y = x_1x_2 + x_3 = (x_1 + x_3)(x_2 + x_3) = \overline{\overline{(x_1 + x_3)(x_2 + x_3)}} =$$
$$= \overline{\overline{(x_1 + x_3)} + \overline{(x_2 + x_3)}}.$$

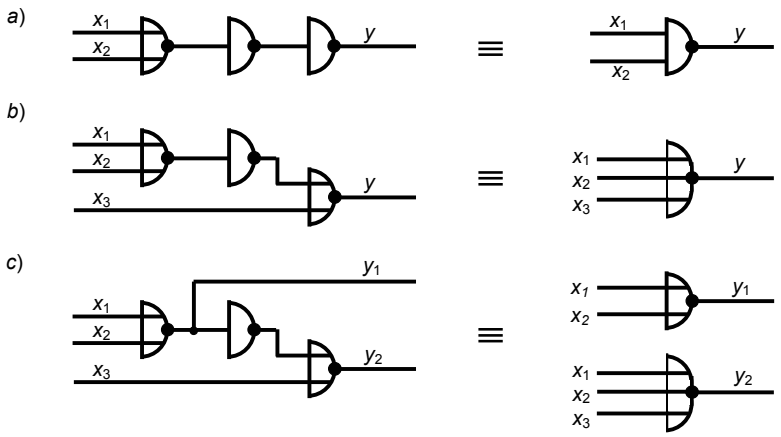
Do zbudowania układu wystarczają teraz trzy elementy NOR, zgodnie z rys. 14.17c.

Bezpośrednia transformacja postaci minimalnej na schemat układu logicznego z elementów NOR może dać dobre wyniki, jeżeli postępuje się według następujących reguł (niekiedy nazywa się to metodą schematów zastępczych):

1) znaleźć alternatywną i koniunkcyjną postać minimalną funkcji,

2) sporządzić schematy układów logicznych z elementów NOR, odpowiedające obu postaciom minimalnym,

3) uprościć obydwa schematy zgodnie z zasadami podanymi na rys. 14.18. (według [7]),



Rys. 14.18. Zasady upraszczania schematów układów logicznych z elementów NOR (identyczne obowiązują dla elementów NAND)

4) wybrać prostszy układ.

O potrzebie takiego właśnie postępowania najlepiej świadczy przykład *z* rys. 14.17. Podane zasady upraszczania nie znajdują zastosowania do schematu z rys. 14.17a i nie pozwalają na jego podstawie przejść do schematu z rys. 14.17c, ale gdybyśmy znaleźli minimalną postać koniunkcyjną odpowiadającą (14.13a)

$$y = (x_1 + x_3)(x_2 + x_3). \qquad (14.13b)$$

wówczas po wykonaniu uproszczeń otrzymamy właśnie schemat z rys. 14.17c.

Analogiczne reguły postępowania stosuje się dla układów budowanych z elementów NAND.

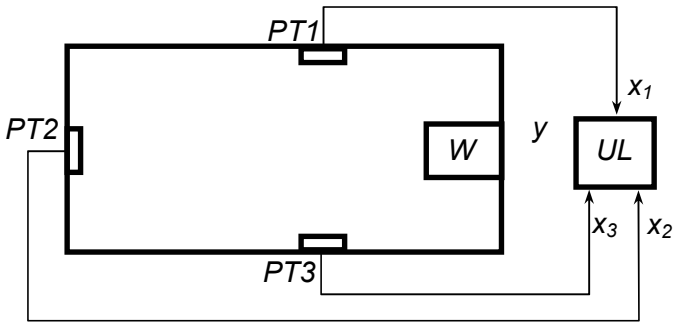
Należy pamiętać, że opisana metoda nie zawsze prowadzi do znalezienia minimalnego układu. Istnieją skuteczniejsze (ale bardziej złożone) metody minimalizacji układów z elementów NOR albo NAND, opisane w literaturze specjalistycznej, np. [4], [7],

14.5. Przykłady syntezy układów kombinacyjnych

14.5.1. Układ sterowania pracą wentylatora

Schemat procesu pokazany jest na rys. 14.19, a zadanie układu formułowano w postaci opisu słownego.

„Temperatura mierzona jest w trzech punktach hali maszyn. Wentylator powinien być uruchamiany, jeżeli co najmniej w dwóch punktach temperatura przekracza 25° C".



Rys. 14.19. Schemat procesu do przykładu z punktu 14.5.1. *PT* — przetworniki (dwustanowe) temperatury, *UL* — układ logiczny, *W* — wentylator z silnikiem i stycznikiem

Przyjmując, że temperaturom poniżej 25°C odpowiadają wartości 0 sygnałów wejściowych x_1, x_2, x_3, temperaturom powyżej 25°C wartości l, stanowi załączenia wentylatora odpowiada y = l, a stanowi spoczynku y = 0, sporządzimy tablicę wartości funkcji przedstawioną na rys. 14.20.

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
y	0	0	0	1	0	1	1	1

Rys. 14.20. Tablica wartości funkcji do przykładu z punktu 14.5.1

Na podstawie tej tablicy można napisać równanie funkcji logicznej w kanonicznej postaci alternatywnej:

$$y = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3. \tag{14.14}$$

lub w kanonicznej postaci koniunkcyjnej

$$y = (x_1 + x_2 + x_3)(x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)(\bar{x}_1 + x_2 + x_3) \qquad (14.15)$$

Zajmiemy się dalej tylko kanoniczną postacią alternatywną. Równanie (14.14) można uprościć dodając do prawej strony dwa wyrazy $x_1x_2x_3$ i zapisując to równanie w postaci

$$y = x_1x_2(x_3 + \bar{x}_3) + x_1x_3(x_2 + \bar{x}_2) + x_2x_3(x_1 + \bar{x}_1). \qquad (14.16)$$

Wyrażenia w nawiasach mają wartość ł, a zatem po uproszczeniu otrzymamy

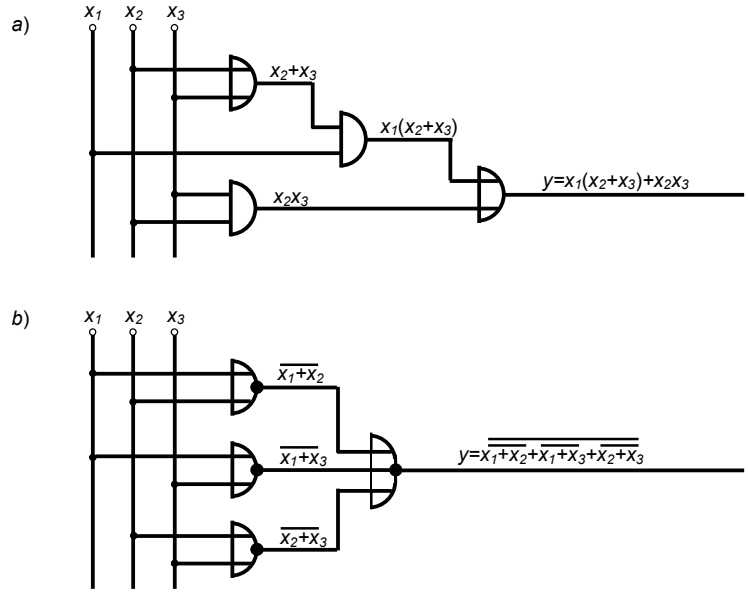
$$y = x_1x_2 + x_1x_3 + x_2x_3. \qquad (14.17)$$

Rozważany przykład jest tak prosty, że równanie (14.17) można byłoby napisać bezpośrednio na podstawie słownego opisu procesu.

Dalsze możliwe uproszczenia są już niewielkie i ostatecznie otrzymamy postać minimalną

$$y = x_1(x_2 + x_3) + x_2x_3. \qquad (14.18)$$

Jeżeli dysponujemy elementami podstawowego systemu funkcjonalnie pełnego, tzn. elementami negacji, alternatywy i koniunkcji, to przejście od wy-



Rys. 14.21. Schematy układów logicznych do przykładu z punktu 14.5.1: a) z elementów podstawowego systemu funkcjonalnie pełnego, b) z elementów NOR

znaczonej postaci minimalnej do schematu logicznego jest bardzo proste, gdyż w równaniu (14.18) występują tylko funkcje koniunkcji i alternatywy. Odpowiedni schemat przedstawiono na rys. 14.21a.

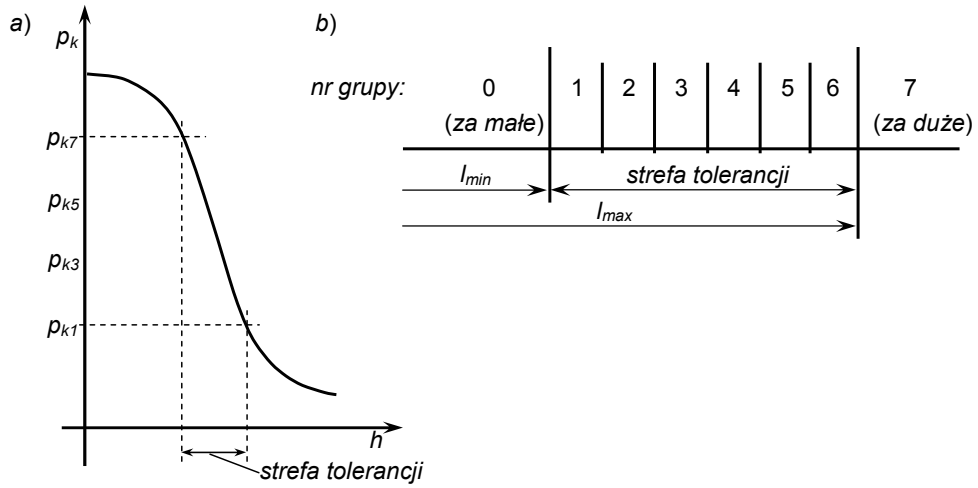
Jeżeli dysponujemy systemem zawierającym wyłącznie elementy NOR, to jej będzie wykonać następujący ciąg przekształceń równania (14.17):

$$\begin{aligned} y &= \overline{\bar{x}_1 + \bar{x}_2} + \overline{\bar{x}_1 + \bar{x}_3} + \overline{\bar{x}_2 + \bar{x}_3} = \overline{(\bar{x}_1 + \bar{x}_2)(\bar{x}_1 + \bar{x}_3)(\bar{x}_2 + \bar{x}_3)} = \\ &= \bar{x}_1\bar{x}_2 + \bar{x}_1\bar{x}_3 + \bar{x}_2\bar{x}_3 = \overline{\overline{x_1 + x_2}} + \overline{\overline{x_1 + x_3}} + \overline{\overline{x_2 + x_3}}. \end{aligned} \tag{14.19}$$

Prawa strona równania (14.19) zawiera wyłącznie funkcje NOR i odpowiada jej schemat logiczny przedstawiony na rys. 14.21b.

14.5.2. Układ sterowania procesem selekcji na grupy wymiarowe

Wymiary liniowe elementów wytwarzanych masowo często mierzone są za pomocą czujników pneumatycznych. Wielkością wyjściową czujnika jest wówczas ciśnienie panujące w kaskadzie pneumatycznej, które jest funkcją np. długości mierzonego elementu (rys. 14.22a).

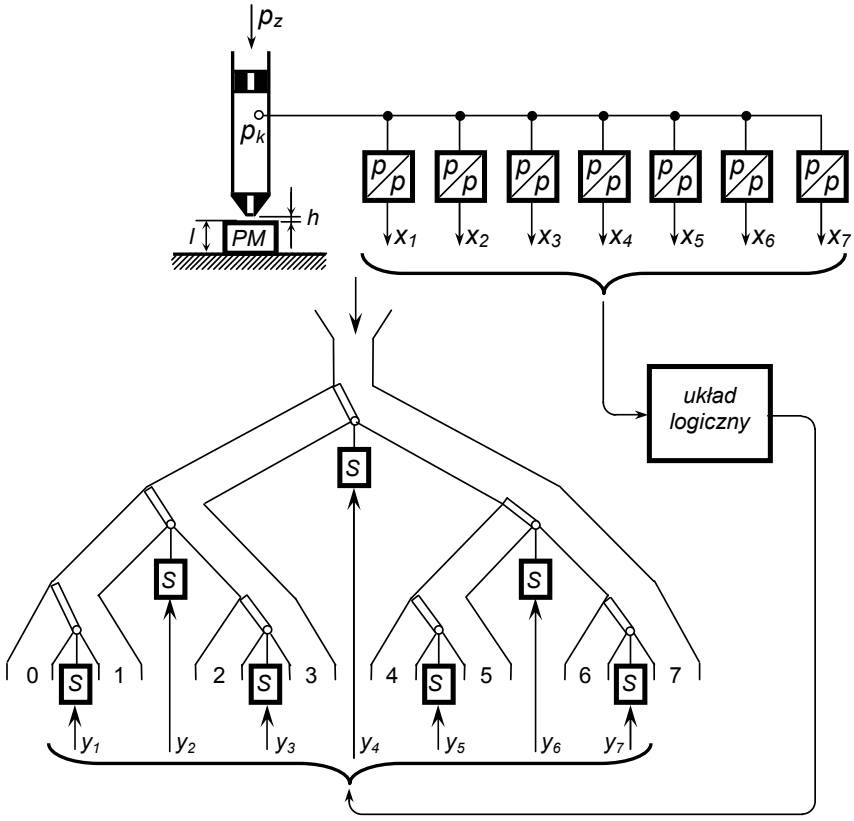


Rys. 14.22. a) charakterystyka kaskady pneumatycznej czujnika, b) podział na grupy wymiarowe

Strefę tolerancji możemy dodatkowo podzielić na szereg przedziałów (rys. 14.22b) i zadaniem układu utorowania będzie selekcja przedmiotów mieszczących się w strefie tolerancji na sześć grup wymiarowych oraz dodatkowo oddzielne grupowanie przedmiotów o długości mniejszej od $l_{\min}$ (braki nienaprawialne) i większej od $l_{\max}$ (braki naprawialne).

Schemat urządzenia do selekcji przedmiotów na grupy wymiarowe przed-

stawiono na rys. 14.23. Elementami wejściowymi układu sterowania są przekaźniki ciśnienia *p/p,* które przy określonej wartości ciśnienia wejściowego (jest to tzw. punkt przełączania, nastawiany) zmieniają wartość swojego sygnału wyjściowego z 0 na l przy wzroście ciśnienia, a z l na 0 przy spadku ciśnienia.



Rys. 14.23. Schemat urządzenia do selekcji przedmiotów na grupy wymiarowe; *PM* — przedmiot mierzony, *δ* — siłownik, p_z — ciśnienie zasilania, p_k — ciśnienie kaskadowe

Elementami wyjściowymi są siłowniki pneumatyczne, sterujące położeniami przegród kierujących przedmioty do kolejnych zasobników. Jeżeli wartości wszystkich sygnałów *y* są równe l, to położenia przegród są zgodne z rys. 14.23.

Wartości sygnałów wyjściowych przekaźników ciśnienia odpowiadające poszczególnym grupom wymiarowym zestawiono w tabl. 14.1.

Dzięki specjalnemu układowi kanałów przedstawionemu na rys. 14.23 synteza układu logicznego może być bardzo łatwo przeprowadzona metodą intuicyjną. Można więc zrezygnować z normalnego toku postępowania (układanie równań na podstawie tabl. 14.1, minimalizacja itd.) i napisać bezpośrednio wyrażenia logiczne opisujące poszczególne sygnały wyjściowe *y,* sterujące położeniami przegród w rozgałęzieniach kanałów:

$$y_1 = x_1,$$

$$y_2 = x_1 x_2,$$

$$y_3 = x_3,$$
$$y_4 = x_1x_2x_3x_4,$$
$$y_5 = x_5,$$
$$y_6 = x_5x_6,$$
$$y_7 = x_7.$$

Tablica 14.1

Wartości sygnałów *x* przekaźników ciśnienia dla poszczególnych grup wymiarowych

Sygnał / Grupa wymiarowa	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0 (za małe)	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	1	0	0	0	0	0
3	1	1	1	0	0	0	0
4	1	1	1	1	0	0	0
5	1	1	1	1	1	0	0
6	1	1	1	1	1	1	0
7 (za duże)	1	1	1	1	1	1	1

Łatwo sprawdzić, że gdy mierzony przedmiot należy np. do grupy 4, a więc zachodzi

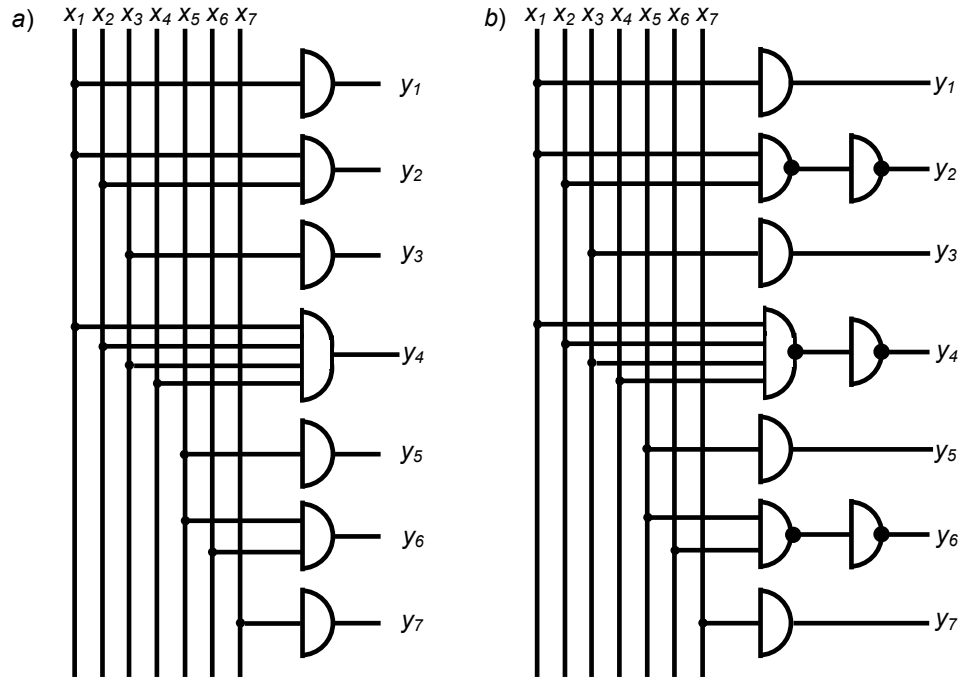
$$x_1x_2x_3x_4\bar{x}_5\bar{x}_6\bar{x}_7 = 1,$$

poszczególne sygnały wyjściowe przyjmą wartości:

$$y_4 = x_1x_2x_3x_4 = 1,$$
$$y_4 = x_5x_4 = 0,$$
$$y_4 = x_5 = 0.$$

Wartości pozostałych sygnałów y są w tym przypadku nieistotne.

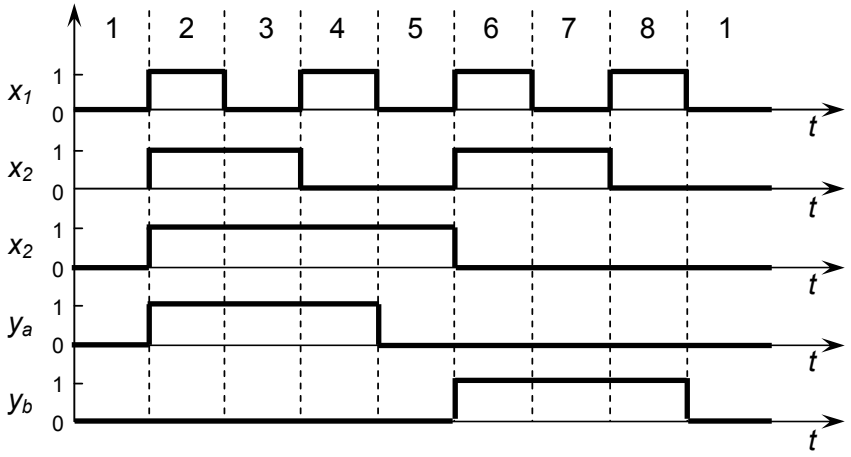
Schemat układu logicznego przedstawiony został na rys. 14.24. W układzie z rys. 14.24a występują tylko elementy koniunkcji i powtórzenia. Rolę tych ostatnich odgrywać będą wzmacniacze dwustanowe, w które wyposażony jest każdy system, gdyż sygnały wyjściowe elementów logicznych oraz przekaźników pomiarowych często mają za małą moc, aby bezpośrednio sterować siłownikami. Z tego względu w praktyce sygnały y_2, y_4, i y_5, również powinny zostać wzmocnione przed wejściem na siłowniki. Schemat układu z rys. 14.24b narysowano przy założeniu, że mamy do dyspozycji tylko elementy NAND (negacji koniunkcji) oraz wzmacniacze realizujące funkcję powtórzenia lub negacji.



Rys. 14.24. Schematy układów logicznych sterujących procesem selekcji na grupy wymiarowo

14.5.3. Synteza układu logicznego na podstawie wykresu czasowego

Wymagania stawiane układowi określane są często przez podanie wykresu czasowego, pokazującego przebiegi sygnałów wejściowych x (informujących o stanie procesu) i wyjściowych y (sterujących procesem). Przykładowy wykres czasowy przedstawiono na rys. 14.25.



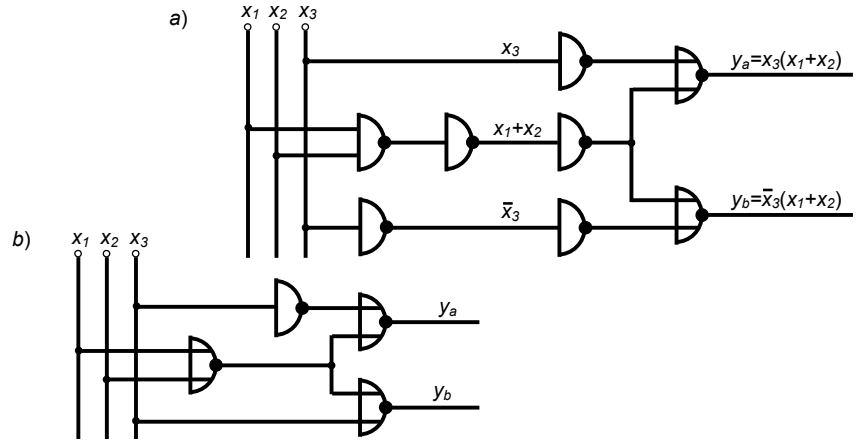
Rys. 14.25. Wykres czasowy

Zwykle można pominąć na wykresie przebiegi przejściowe występujące w chwilach przełączania i przyjąć, że czasy przełączania są zerowe. Przedział czasu, w którym wartości sygnałów wejściowych i wyjściowych pozostają stałe, nazywany jest taktem. W układach kombinacyjnych o n wejściach liczba wszystkich możliwych różnych taktów wynosi 2^n. Takty należy ponumerować według kolejności ich występowania lub za pomocą liczb dziesiętnych odpowiedających binarnemu zapisowi sygnałów wejściowych w danym takcie.

Na podstawie wykresu podanego na rys. 14.25 należy przeprowadzić syntezę układu logicznego z elementów NOR.

Nr	1	2	3	4	5	6	7	8
x_1	0	1	0	1	0	1	0	1
x_2	0	1	1	0	0	1	1	0
x_3	0	1	1	1	1	0	0	0
y_a	0	1	1	1	0	0	0	0
y_b	0	0	0	0	0	1	1	1

Rys. 14.26. Tablica wartości funkcji odpowiadająca wykresowi czasowemu z rys. 14.25



Rys. 14.27. Schematy układów logicznych z elementów NOR: a) postać pierwotna, b) postać uproszczona

Pierwszym krokiem będzie przejście od wykresu czasowego do tablicy wartości funkcji, którą przedstawiono na rys. 14.26. Kanoniczna postać alternatywna funkcji jest następująca:

Minimalizując oddzielnie obydwie funkcje, np. metodą tablic Karnaugha, otrzymamy

$$y_a = x_3(x_1 + x_2), \qquad y_b = \bar{x}_3(x_1 + x_2).$$

W obydwu funkcjach występuje wspólny składnik (x_1+x_2) co umożliwia zmniejszenie liczby elementów potrzebnych do realizacji tych funkcji.

Przechodząc bezpośrednio od postaci minimalnej do schematu układu z elementów NOR, otrzymamy wynik przedstawiony na rys. 14.27a. Po przeprowadzeniu możliwych uproszczeń ostateczna postać schematu będzie zgodna z rys. 14.27b.

15. Synteza układów sekwencyjnych

15.1. Struktury układów sekwencyjnych

W układach sekwencyjnych sygnały wyjściowe są uzależnione nie tylko od stanu wejść w danej chwili, lecz również od stanów poprzednich. Jest to zrealizowane przez zastosowanie tzw. elementów pamięci lub, inaczej mówiąc, przez wprowadzenie sprzężeń zwrotnych podających na wejście układu sygnały informujące o dotychczasowym stanic wyjścia.

Wprowadzimy następujące oznaczenia:

stan wejść X — zbiór wartości występujących w danej chwili sygnałów wejściowych

$$X = (x_1, x_2, ..., x_n),$$

stan wyjść T — zbiór wartości występujących w danej chwili sygnałów wyjściowych

$$Y = (y_1, y_2, ..., y_m),$$

stan wewnętrzny Q — zbiór wartości występujących w danej chwili sygnałów wyjściowych elementów pamięciowych

$$Q = (Q_1, Q_2, ..., Q_k),$$

stan wzbudzeń q — zbiór wartości występujących w danej chwili sygnałów wejściowych elementów pamięciowych

$$q = (q_1, q_2, ..., q_k),$$

Ogólną strukturę układów sekwencyjnych przedstawiono na rys. 15.la. W układach kombinacyjnych czas przejścia sygnału przez układ nie ma wpływu na realizowaną funkcję i zostaje pominięty. Natomiast w układzie pamięciowym uwzględnienie czasu jest bardzo istotne; opóźnienie sygnałów wyjściowych w stosunku do wejściowych oznaczone zostanie τ. Równania opisujące układ sekwencyjny można, więc podać w postaci:

$$Y^t = f(Q^t, X^t), \tag{15.1}$$

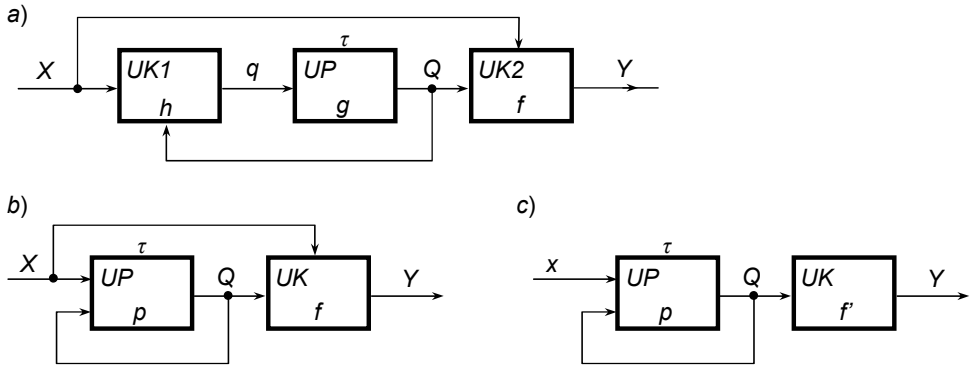
$$Q^{t+\tau} = g(q^t), \tag{15.2}$$

$$q^t = h(Q^t, X^t) \tag{15.3}$$

Stan wzbudzeń q^t można wyeliminować z opisu układu podstawiając (15.3) do (15.2). Otrzymamy

$$Q^{t+\tau} = p(Q^t, X^t) \tag{15.4}$$

Układ równań (15.1) i (15.4) odpowiada strukturze tzw. *układu Mealy'ego* (rys. 15.1b). Funkcja wyjść f określa stan wyjść Y na podstawie stanu wewnętrznego (pamięci) i stanu wejść, a funkcja przejść p określa nowy stan wewnętrzny na podstawie aktualnego stanu wewnętrznego i stanu wejść.



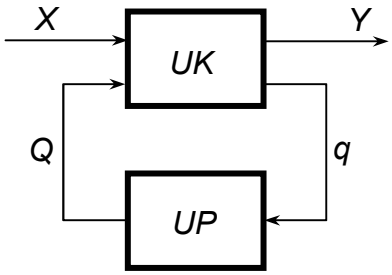
Rys. 15.1. Struktury układów sekwencyjnych: a) schemat ogólny, b) układ Mealy'ego, c) układ Moore'a; *UK* — układ kombinacyjny, *UP* — układ pamięciowy

Układ Moore'a (rys. 15.Ic) charakteryzuje się tym, że funkcja wyjść f' zależy tylko od stanu wewnętrznego Q^t

$$Y^t = f'(Q^t)\,. \tag{15.5}$$

Układy o tej strukturze opisują równania (15.4) i (15.5).

Spotyka się również przedstawianie ogólnej struktury układu sekwencyjnego w postaci pokazanej na rys. 15.2 ([1], [6]). Oznaczenia sygnałów wejściowych i wyjściowych przyjęto takie same jak na rys. 15.1.



Rys. 15.2. Inny sposób przedstawiania struktury układu sekwencyjnego

Układy sekwencyjne dzieli się także na asynchroniczne i synchroniczne. W układach asynchronicznych taktem nazywa się przedział czasu między kolejnymi zmianami wartości sygnałów, niezależnie od czasu trwania poszczę-

gólnych sygnałów, a więc długość taktów może być różna. W układach synchronicznych takt pracy jest stały; specjalny generator impulsów taktujących (zegar) wyznacza momenty, w których stan wejść wprowadzany jest do układu. Wówczas np. sygnał l trwający przez trzy kolejne takty wprowadzony zostanie jako trzy kolejne jedynki.

W niniejszej książce rozważania ograniczono do układów asynchronicznych statycznych, tzn. operujących wyłącznie stanami statycznymi X^t, Y^t, Q^t (w układach dynamicznych bierze się również pod uwagę sposób zmiany X oraz niekiedy Q, przez uwzględnienie tzw. stanu dynamicznego wejść X^{t-1}/X^t oraz stanu dynamicznego wewnętrznego Q^{t-1}/Q^t)

Formułowanie zadania układu sekwencyjnego najczęściej spotyka się w następujących postaciach:

a) opis słowny, w przypadkach bardziej złożonych podawany w formie tablicy opisującej poszczególne takty (etapy działania) układu,

b) wykres czasowy lub odpowiadające mu ciągi zerojedynkowe,

c) tablice przejść i wyjść.

Opis słowny jest najchętniej stosowany przy projektowaniu układów sekwencyjnych metodą intuicyjną, natomiast inne metody syntezy wymagają, zwykle sporządzenia tablic przejść i wyjść. Zagadnienia te zostaną opisane szerzej w dalszych punktach rozdziału.

15.2. Elementy pamięci

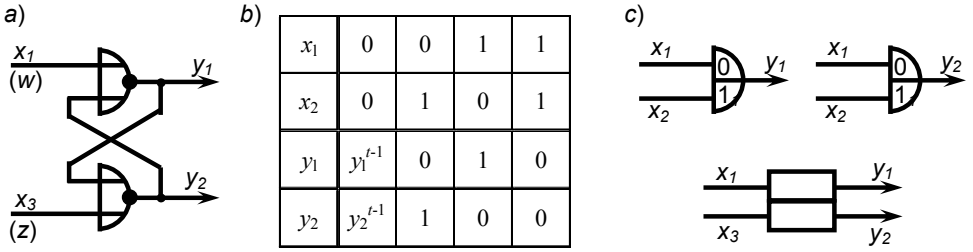
Najczęściej spotykana realizacja pamięci polega na połączeniu dwóch elementów NOR według rys. 15.3a. Obok schematu połączeń podano tablicę wartości funkcji (tablicę stanów), z której wynika, że pamiętanie poprzedniego stanu wyjścia (oznaczonego indeksem t - 1) zachodzi przy kombinacji sygnałów wejściowych x_1=0 *i* x_2=0. Dla wyraźniejszego zilustrowania działania elementu na rys. 15.3d wypisano wartości sygnałów w kolejnych chwilach czasu, (t - 1 oraz t, dla obydwu możliwych przypadków dojścia do stanu x_1=0 i x_2=0. Zakłada się, że wartości sygnałów x_1 i x_2 nie mogą zmieniać się równocześnie, a zatem nie może wystąpić przejście ze stanu x_1=l i x_2=l do stanu x_1=0 i x_2=0. Sygnał x_1 nazywany jest często wpisującym (w) jedynkę do pamięci albo włączającym, a x_2 zerującym (z) pamięć albo wyłączającym.

Wykorzystując tylko wyjście y_1 lub tylko wyjście y_2 otrzymuje się tzw. element pamięci z priorytetem wyłączania, przy czym sygnałem wyłączającym (zerującym pamięć) jest w pierwszym przypadku x_2, a w drugim przypadku x_1. Odpowiednie równania funkcji logicznych są następujące:

$$y_1^t = \bar{x}_2(x_1 + y_1^{t-1}),, \tag{15.6}$$

$$y_2^t = \bar{x}_1(x_2 + y_2^{t-1}).., \tag{15.7}$$

Wykorzystując równocześnie wyjścia y_1 i y_2 otrzymuje się tzw. przerzutnik *w-z*, nazywany także przerzutnikiem *r-s* lub tryggerem. Na podstawie podanej tablicy wartości funkcji (rys. 15.3b) łatwo zauważyć, że we wszystkich przy-



x_1	0	0	1	1
x_2	0	1	0	1
y_1	y_1^{t-1}	0	1	0
y_2	y_2^{t-1}	1	0	0

d)

	$t-1$	t
x_1	0	0
x_2	1	0
y_1	0	0
y_2	1	1

	$t-1$	t
x_1	1	0
x_2	0	0
y_1	1	1
y_2	0	0

Rys. 15.3. Podstawowy element pamięci, zbudowany z dwóch elementów NOR: a) schemat połączeń, b) tablica wartości funkcji, c) symbole graficzne, d) przykłady działania

padkach kiedy $x_1x_2=0$ otrzymujemy $y_2=\overline{y_1}$. Niekiedy nawet oznacza się dwa wyjścia przerzutnika symbolami y i $\overline{y}$ (lub Q i $\overline{Q}$), ale należy pamiętać, że negacja ta nie jest słuszna, jeżeli obydwa wejścia mają wartość 1.

Spotyka się również elementy pamięci z priorytetem włączania, które buduje się najczęściej z elementów NAND według schematu podanego na rys. 15.4a. Równania funkcji logicznych mają wówczas postać

$$y_1^t = x_1 + \overline{x}_2 y_1^{t-1}, \tag{15.8}$$

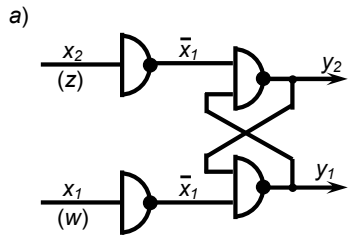
$$y_2^t = x_2 + \overline{x}_1 y_2^{t-1}. \tag{15.9}$$

Sygnałami włączającymi są: x_1 dla wyjścia y_1 oraz x_2 dla wyjścia y_2.

Porównując tablice funkcji z rys. 15.3b i 15.4b można zauważyć, że obydwa przedstawione elementy pamięci różnią się tylko stanem wyjść przy kombinacji sygnałów x_1=l i x_2=1. Przy wszystkich pozostałych kombinacjach sygnałów, tzn. gdy $x_1x_2=0$, działanie tych elementów jest identyczne.

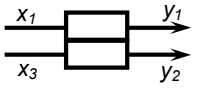
Elementy negacji występujące w schemacie z rys. 15.4a można oczywiście odrzucić, jeżeli dysponujemy w układzie sygnałami $\overline{x_1}$ i $\overline{x_2}$. W przypadku wprowadzenia sygnałów x_1 i x_2 bezpośrednio na elementy NAND (zamiast $\overline{x_1}$ i $\overline{x_2}$)

otrzymamy także układ pamięci, ale zniknie podobieństwo do układu z rys. 15.3. Na przykład stan pamiętania wystąpi wówczas przy kombinacji sygnałów x_1 = l i x_2 = l itd.



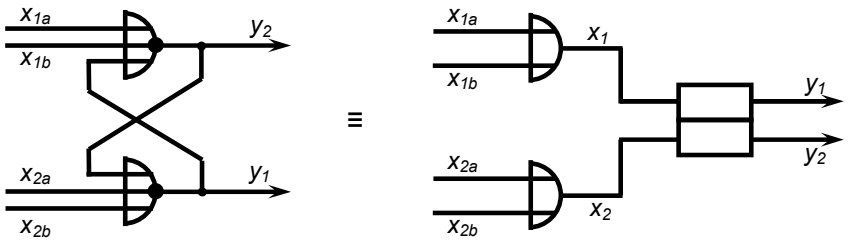
b)

x_1	0	0	1	1
x_2	0	1	0	1
y_1	y_1^{t-1}	0	1	1
y_2	y_2^{t-1}	1	0	1



Rys. 15 4. Element pamięci zbudowany z elementów NAND: a) schemat połączeń, b) tablica wartońci funkcji, c) symbol graficzny

Układy pamięciowe o większej liczbie wejść niż dwa można zwykle sprowadzić do podstawowego elementu dwu wejściowego (przerzutnika), poprzedzonego układem kombinacyjnym. Przykład takiego przekształcenia pokazano na rys. 15.5.

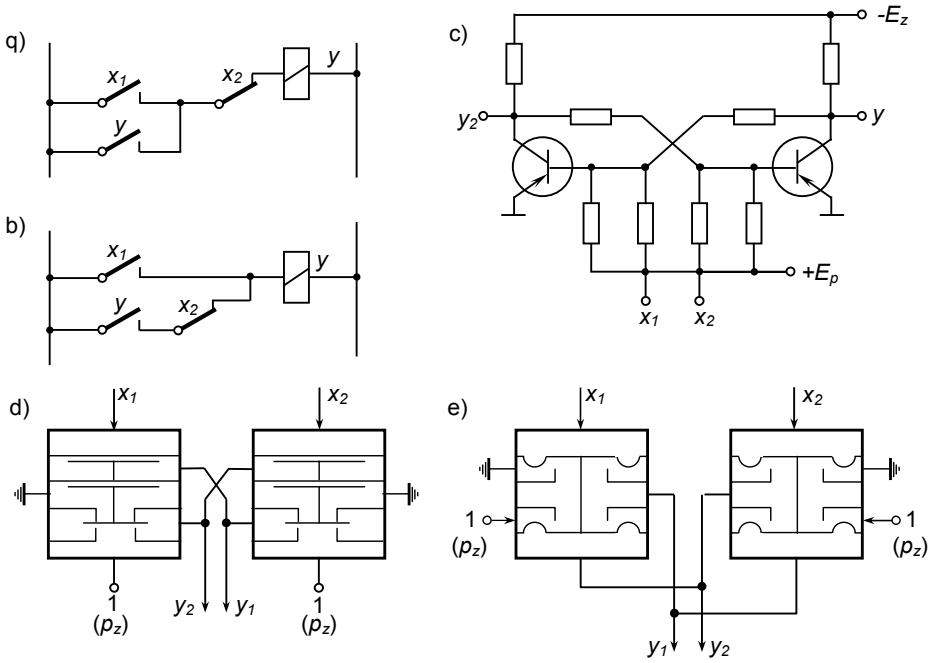


Rys. 15.5. Przekształcenie układu pamięci o czterech wejściach

Budowa elementów pamięci w układach przekaźnikowych, półprzewodnikowych i membranowych została zilustrowana kilkoma przykładami na rys. 15.6. Budowę najprostszego przerzutnika strumieniowego pokazano już wcześniej na rys. 13.19a.

W układach przekaźnikowych pamięć można zrealizować przez podtorzymanie działania przekaźnika jego własnym zestykiem. Schemat elementu pamięci z priorytetem wyłączania przedstawiono na rys. 15.6a, przy czym x_1 jest stanem zestyku załączającego, a x_2 stanem zestyku wyłączającego (normalnie zamkniętego). Zwarcie zestyku x_1 (tzn. x_1 = 1) powoduje zadziałanie przekaźnika i stan y=l utrzymuje się aż do chwili rozwarcia zestyku x_2 (tzn. do chwili x_2=1), niezależnie od stanu x_1, Stan x_2=l zawsze powoduje wyłączenie przekaźnika, tzn. y=0, zgodnie z równaniem (15.6). Na rysunku 15.6b podano schemat połączeń elementu pamięci z priorytetem załączania, o równaniu (15.8).

W układach półprzewodnikowych podstawowym rozważaniem jest połączenie dwóch elementów NOR według rys. 15.6c, co odpowiada równaniom (15.6) To samo rozwiązanie dla układów membranowych systemu MERALOG



Rys. 15.6. Przykłady budowy elementów pamięci: a), b) — przekaźnikowych, o) półprzewodnikowego, d), e) — membranowych

pokazano na rys. 15.6d, natomiast w systemie DEELOBA według podobnego schematu połączono dwa elementy implikacji, otrzymując pamięć z priorytetem załączania, o równaniach (15.8) i (15.9).

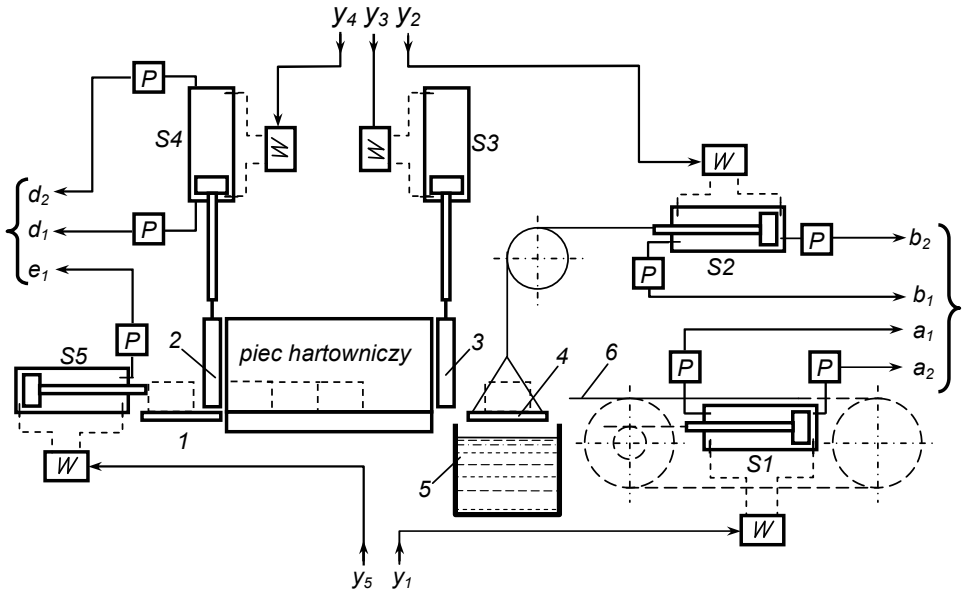
15.3. Intuicyjna metoda syntezy

Metoda intuicyjna polega na sporządzaniu schematów układów logicznych bezpośrednio na podstawie opisu (programu) procesu, zakładając umiejętność projektowania części składowych układu realizujących poszczególne fragmenty procesu i połączenia ich w jedną całość. Otrzymane tą drogą układy zwykle nie są minimalne, natomiast ich struktura jest często bardziej przejrzysta i wykrywanie uszkodzeń łatwiejsze. Zakres zastosowań tej metody zależy w decydującym stopniu od kwalifikacji i doświadczenia projektanta.

Niżej pokazane zostaną dwa przykłady projektowania układu logicznego metodą intuicyjną.

15.3.1. Przykład sterowania procesem hartowania

Schemat procesu przedstawiono na rys. 15.7. Ze względu na wymagane duże siły i przesunięcia, jako elementy wykonawcze najczęściej stosuje się w hartowni siłowniki hydrauliczne. Dysponujemy więc hydraulicznymi źród-



Rys. 15.7. Schemat procesu hartowania: *S* — siłownik hydrauliczny, *W* — wzmacniacz hydrauliczny, *P* — przekaźnik krańcowy położenia, *2* — stół, 2 oraz *3* — drzwi pieca, *4* — stół podnośnika, 5 — zbiornik cieczy chłodzącej, *6* — szyna z zaczepami

łami zasilania i najkorzystniej będzie zastosować hydrauliczny układ sterujący. W układzie tym możemy wyróżnić trzy grupy elementów:

a) elementy wejściowe — przekaźniki krańcowe położenia *P*,
b) część centralna układu sterowania — elementy logiczne,
c) elementy wyjściowe — wzmacniacze hydrauliczne *W*.

Na rysunku 15.7 oznaczono rozmieszczenie elementów wejściowych i wyjściowych, natomiast schemat logiczny części centralnej podany został na odrębnym rysunku.

Opis procesu. Detale ustawione na płytach hartowniczych powinny przebywać w piecu czas *T*. Po wyjęciu z pieca należy je zanurzyć w cieczy chłodzącej, a po wyjęciu z niej — odtransportować. Płyty hartownicze ustawione są w piecu w szeregu, tzn. gdy wsuwamy do pieca kolejną płytę, wówczas jednocześnie przesuwamy do przodu wszystkie płyty. Jeżeli liczba płyt hartowniczych mieszczących się w piecu wynosi «, to czas pomiędzy dwoma kolejnymi cyklami pracy wynosi T/N. Wartość tę nastawiamy na skali przekaźnika czasowego *PCz*, który w określonych nastawą chwilach czasowych wysyła sygnał uruchamiający kolejny cykl pracy układu.

Płyty hartownicze z detalami podawane są za pomocą transportera na stół 1, *z* którego wypychane są siłownikiem *S5* do wnętrza pieca. Piec ma drzwi 2 *i* 3 podnoszone siłownikami *S4* i *S3*.

Płyty wyciągane są z pieca za pomocą szyny 6, zaopatrzonej w odpowiednie zaczepy. Szyna 6 jest poruszana siłownikiem *S1*. Płyta wraz z detalami ustawiona na stole podnośnika 4 opuszczana jest do zbiornika cieczy chłodzącej 5. Stół podnośnika poruszany jest siłownikiem *S2*.

Przekaźniki *P* reagują na zmianę położenia odpowiednich tłoków. Jeżeli tłok zajmuje położenie krańcowe przy przekaźniku, to przekaźnik ten wysyła sygnał „1" i jeżeli tłoka tam nie ma — sygnał „0".

Zadaniem układu przełączającego sterującego procesem jest realizowanie podanego wyżej cyklu pracy, z zastrzeżeniem, aby nie były otwarte jednocześnie drzwi po obu stronach pieca.

Tablica 15.1

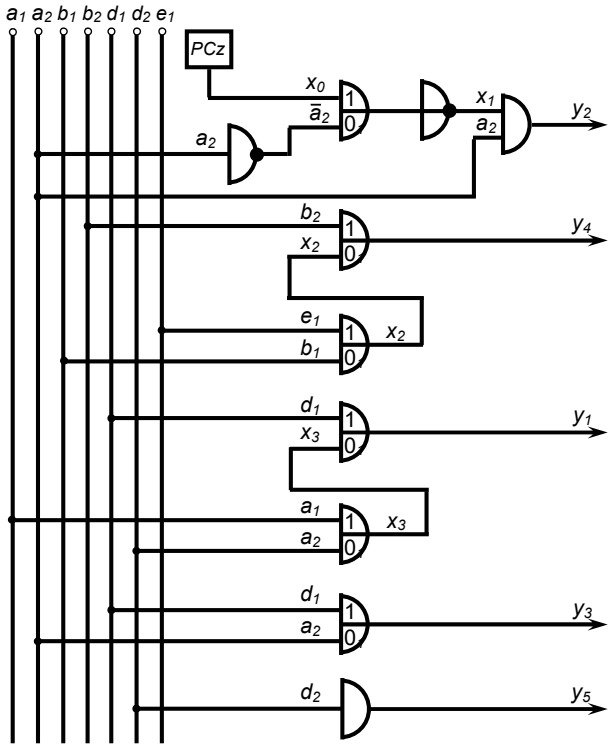
Program procesu hartowania

Etap cyklu	„Jeżeli"	„to"	
		sygnały	czynności
1	pojawia się impuls z przekaźnika *PCz*	$x_0 = 1$, $y_2 = 0$ $x_1 = 1$,	tłok *S2* przesuwa się w prawo ($b_1 = 0$)
2	tłok *S2* dojdzie do prawego skrajnego położenia	$b_2 = 1$, $y_4 = 1$ $x_2 = 0$,	tłok *S4* przesuwa się do góry ($d_1 = 0$)
3	tłok *S4* dojdzie do górnego skrajnego położenia	$d_2 = 1$, $y_5 = 1$ $x_3 = 0$,	tłok *S5* przesuwa się w prawo
4	tłok *S5* dojdzie do prawego skrajnego położenia	$e_1 = 1$, $b_1 = 0$, $y_4 = 0$ $x_2 = 1$,	tłok *S4* zaczyna przesuwać się w dół
5*)	tłok *S4* ruszy z górnego skrajnego położenia	$d_2 = 0$, $y_5 = 0$	tłok *S5* przesuwa się w lewo ($e_1 = 0$)
6	tłok *S4* dojdzie do dolnego skrajnego położenia	$d_1 = 1$, $y_1 = 1$ $x_3 = 0$,	tłok *S1* przesuwa się w lewo, napędzając szynę 6
7*)	tłok *S1* ruszy z prawego skrajnego położenia	$a_2 = 0$, $x_1 = 1$, $y_3 = 1$ $d_2 = 1$,	tłok *S3* przesuwa się do góry
8	dok *S2* dojdzie do lewego skrajnego położenia	$a_1 = 1$, $y_1 = 0$ $x_3 = 1$,	tłok *S1* przesuwa się w prawo; szyna 6 zabiera płytę znajdującą się na stole podnośnika z detalami już zahartowanymi i zostawia na tym stole płytę wyjętą s pieca
9	tłok *S1* dojdzie do prawego skrajnego położenia	$a_2 = 1$, $y_3 = 0$, $x_1 = 1$, $y_2 = 1$	tłok *S3* przesuwa się do dołu oraz tłok *S2* przesuwa się w lewo ($b_2 = 0$)
10	tłok *S2* dojdzie do lewego skrajnego położenia	$b_1 = 1$, $x_2 = 0$	koniec cyklu pracy; układ czeka na kolejny sygnał z przekaźnika *PCz*

*) Dzięki równoczesnemu zachodzeniu etapów 4 i 5 oraz etapów 6 i 7 uzyskujemy znaczne skrócenie czasu chłodzenia (otwarcia) pieca.

Stan wyjściowy procesu przed rozpoczęciem kolejnego cyklu pracy jest następujący: układ sterowania znajduje się w spoczynku; płyta hartownicza wyjęta w poprzednim cyklu i umieszczona na stole 4 zanurzona jest w zbiorniku 5; tłok siłownika *S2* znajduje się w lewym skrajnym położeniu, tłoki pozostałych siłowników znajdują się w położeniach przedstawionych na rys. 15.7; w piecu jest jedno wolne miejsce po płycie, która znajduje się w zbiorniku.

Założony program procesu zestawiono w tabl 15.1, a schemat układu logicznego realizującego ten program podano na rys. 15.8. Oznaczenia sygnałów wejściowych, wewnętrznych i wyjściowych w tabl. 15.1 są zgodne z przyjętymi na rys. 15.7 i 15.8.

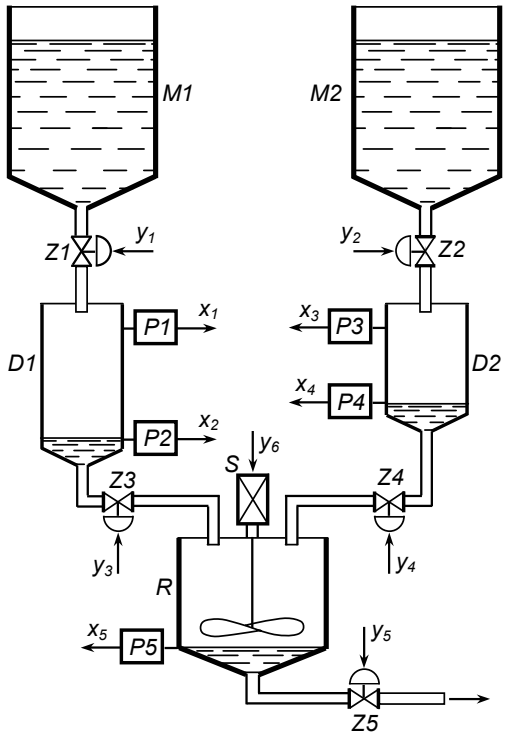


Rys. 15.8. Schemat logiczny części centralnej okładu sterowania procesem hartowania

Układ logiczny przedstawiony na rys. 15.8 uzupełniany jest zwykle dodatkowym układem sterowania ręcznego i elementami blokady, których dla uproszczenia rozważań nie analizowano.

15.3.2. Przykład sterowania procesem dozowania i mieszania dwóch substancji

Schemat części aparaturowej procesu podano na rys. 15.9. W zbiornikach magazynowych *M1* i *M2* zgromadzone są mieszane substancje, których żądane ilości, przed zmieszaniem w reaktorze *R,* odmierzane są w dozownikach *D1*



Rys. 15.9. Schemat procesu dozowania i mieszania dwóch substancji: *M* — zbiorniki magazynowe, *D* — dozowniki, *R* — reaktor, *Z* — zawory, *P* — przekaźniki poziomu, *S* — silnik elektryczny ze stycznikiem

i *D2.* Dozowniki te zaopatrzone są w dwustanowe przetworniki poziomu *P* (zwane krócej przekaźnikami poziomu), których sygnały wyjściowe *x* przyjmują wartości

$$x_i = 0, \quad gdy \quad h < h_i,$$
$$x_i = 1, \quad gdy \quad h < h_i.$$

Silnik elektryczny *8* służy do napędzania mieszadła uruchamianego w reaktorze podczas dolewania zawartości *D2* do *Dl.*

Przed rozpoczęciem kolejnego cyklu pracy poziomy substancji w zbiornikach są zgodne z rys. 15.9 i wszystkie zawory *Z* są zamknięte. Załóżmy, że wymagany jest następujący program procesu:

a) dozowanie obu substancji przez napełnianie zbiorników *D1 i D2* może się rozpocząć tylko wtedy, gdy wprowadzony zostanie ręcznie sygnał „start" ($x_0 = 0$), a zbiorniki nie są całkowicie napełnione: $x_1 = 0$ oraz $x_3 = 0$;

b) zakończenie napełniania każdego z dozowników *D* powinno nastąpić wówczas, gdy osiągnięty zostanie górny poziom: $x_1 = 1$ dla *D1* oraz $x_2 = 1$ dla *D2*;

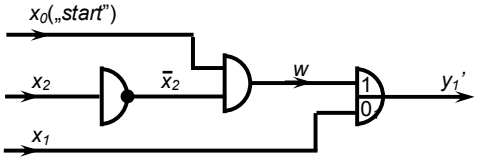
c) po napełnieniu dozownika *D1* rozpoczyna się przelewanie jego zawartości do reaktora *R,* które powinno zakończyć się wtedy, gdy zostanie osiągnięty dolny poziom: $x_2 = 0$;

d) po przelaniu zawartości *D1* do reaktora rozpoczyna się dolewanie do niej zawartości *D2 i* równocześnie uruchomiony zostaje silnik *S* napędzający mieszadło;

e) po zakończeniu dolewania zawartości *D2* do reaktora mieszanie powinno trwać jeszcze przez czas τ;

f) po wyłączeniu silnika *S* powinien zostać otwarty zawór $Z5_t$ który zamyka się po opróżnieniu reaktora, tzn. gdy $x_5 = 0$; aparatura gotowa jest do rozpoczęcia następnego cyklu pracy.

Rozważmy przykładowo część układu logicznego służącą do sterowania pracą zaworu *Z1* (rys. 15.10). Przed rozpoczęciem kolejnego cyklu dozownik *D1* jest opróżniony, zatem zarówno $x_1 = 0$, jak i $x_2 = 0$. Ponieważ $x_0 = 0$, zatem

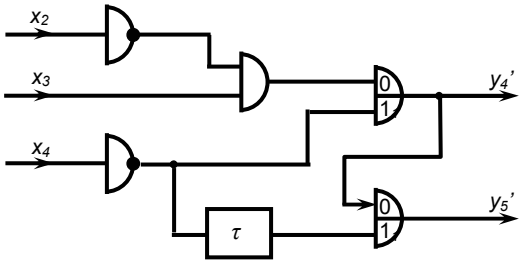


Rys. 15.10. Część układu sterująca zaworem *Z1*

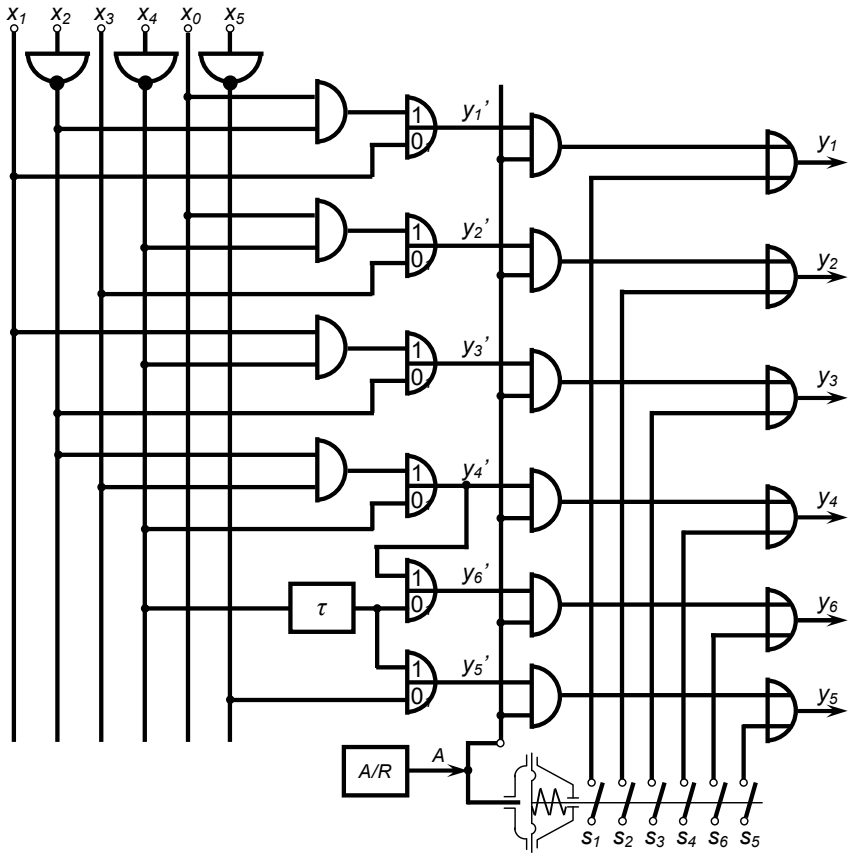
$w = x_0 \overline{x_2} = 0$ (stan pamiętania) i zawór *Z1* znajduje się w zajmowanym ostatnio położeniu w poprzednim cyklu pracy, tzn. jest zamknięty: $y_1 = 0$. Wprowadzenie sygnału „start" ($x_0 = 1$) powoduje $w = 1$ oraz $y_1 = 1$ i zawór zostaje otwarty. Stan $y_1 = 1$ utrzymuje się po przekroczeniu poziomu dolnego, tzn. gdy $x_2 = 1$ i $w=0$, a zamknięcie zaworu następuje dopiero wtedy, gdy osiągnięty zostanie górny poziom w dozowniku: wówczas $x_1 = 1$ i $y_1 = 0$.

Opóźnienie wyłączenia silnika o czas *r* w stosunku do momentu zamknięcia zaworu *Z4* uzyskane może być przez zastosowanie przekaźnika czasowego, którego sygnał wyjściowy przesunięty jest o τ w stosunku do wejścia. Odpowiedni fragment układu pokazano na rys. 15.11.

Schemat całego układu logicznego realizującego założony program procesu przedstawiono na rys. 15.12. Dodatkowo pokazano układ sterowania ręcznego (sygnały $s_1, \ s_1, \ldots, s_5$ mogą być wprowadzane np. za pomocą przycisków lub przełączników ręcznych), który wykorzystywany jest do sterowania pracą



Rys. 15.11. Część układu sterująca zaworem Z4 i silnikiem S



Rys. 15.12. Schemat części centralnej układu sterowania procesem dozowania i mieszania dwóch substancji; *A/R* — przełącznik rodzaju pracy (sterowanie automatyczne/sterowanie ręczne), τ — przekaźnik czasowy

zaworów Z i silnika S po przestawieniu przełącznika rodzaju pracy z położenia „sterowanie automatyczne” w położenie „sterowanie ręczne”; wówczas $A = 0$ i sygnały wyjściowe układu sterowania automatycznego $y'_1, y'_2, ..., y'_n$ nie mają wpływu na położenia zaworów i etan silnika. Przy $A = 1$ uniemożliwione jest oczywiście sterowanie ręczne.

15.4. Metoda tablic przejść i wyjść

Metoda ta pozwala prowadzić syntezę układów o złożonym algorytmie działania, a ponadto umożliwia wykrywanie i usuwanie zjawisk tzw. „wyścigów”, prowadzących do zawodności logicznej układów zbudowanych ze sprawnych technicznie elementów. Zalety te powodują, że jest jedną z częściej stosowanych metod syntezy układów sekwencyjnych, chociaż trudności w jej stosowaniu szybko rosną z liczbą sygnałów wejściowych układu. W przypadkach układów bardziej rozbudowanych powoduje to konieczność dekompozycji układu na bloki, których syntezę przeprowadza się oddzielnie.

Tablica przejść opisuje funkcję przejść *p,* tzn. określa nowy stan wewnętrzny $Q^{t+\tau}$ (skrócony zapis Q') na podstawie aktualnego stanu wewnętrznego Q^t i stanu wejść X^t (skrócony zapis: Q, X).

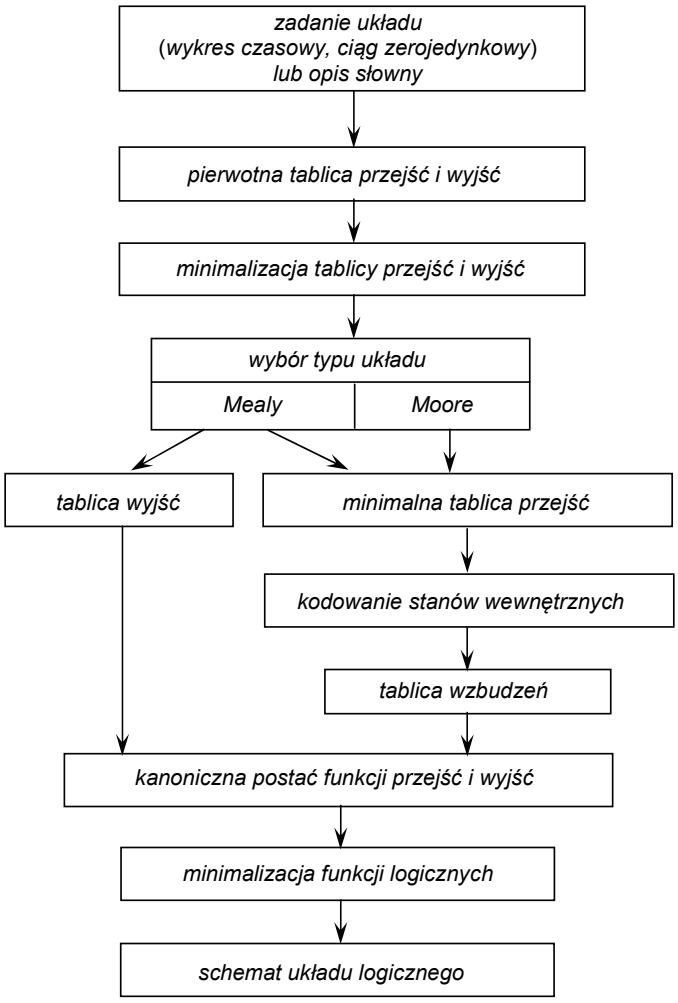
Tablica wyjść opisuje funkcję wyjść *f*, przy czym dla układu Moore'a upraszcza się ona do jednej kolumny podającej stany wyjść Y odpowiadające poszczególnym stanom wewnętrznym Q, natomiast dla układu Mealy'ego jest tablicą określającą stany wyjść Y na podstawie stanów wewnętrznych Q i stanów wejść X.

Ogólny zarys postępowania przy syntezie układów sekwencyjnych metodą tablic przejść i wyjść przedstawiono na rys. 15.13. Kolejne etapy syntezy omówione zostaną na dwóch przykładach.

Zadanie układów sekwencyjnych (zwłaszcza asynchronicznych statycznych) określane jest niekiedy za pomocą wykresu czasowego. Tworzenie tablic przejść i wyjść w takim przypadku zilustrowane zostanie przykładem układu bramkowania generatora (rys. 15.14). Zadanie tego układu polega na przepuszczaniu impulsów prostokątnych x_1 generatora, gdy sygnał bramkujący $x_2 = 0$ i nieprzepuszczaniu, gdy $x_2 = 1$, przy czym impulsy x_1 nic mogą ulegać zniekształceniom (obcięciom), niezależnie od momentów zmian x_2.

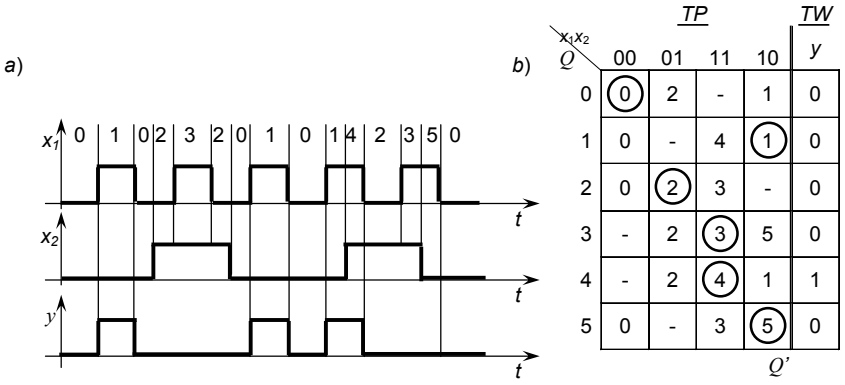
Na wykresie czasowym podane są informacje o stanach wejść X (w rozpatrywanym przykładzie x_1, x_2) i wymaganych odpowiednio stanach wyjść Y (w rozpatrywanym przykładzie y). Stany wewnętrzne Q należy wprowadzić analizując zachowanie się układu w kolejnych chwilach czasu — na rysunku oznaczono je 0, 1, 0, 2, 3, ..., 0 nad wykresem x_1 Po wprowadzeniu tych stanów łatwo ocenić, czy układ będzie kombinacyjny, czy sekwencyjny. Jeżeli identycznym stanom wejść nie są przypisane różne stany wyjść, to wystarczy przewidzieć układ kombinacyjny, którego synteza może być prowadzona meto-

darni opisanymi w rozdz. 14. Z rysunku 15.14 a wynika, że w danym przypadku identyczne stany wejść l i 5 (x_1 = l, x_2 = 0) oraz 3 i 4 (x_1 = l, x_2 = 1) mają przypisane y = 0 lub y = l, zależnie od stanów poprzednich, zatem układ będzie sekwencyjny.



Rys 15.13. Etapy syntezy okładów sekwencyjnych metodą tablic przejść i wyjść

Pierwotna tablica przejść, przedstawiona na rys. 15.14 b, ma tyle wierszy, ile stanów wewnętrznych wyróżniono w układzie, natomiast liczba kolumn odpowiada liczbie możliwych stanów wejść (stany te zapisuje się w takiej kolejności jak w tablicach Karnaugha, co ułatwi końcowy etap syntezy). W poszczególnych kratkach tablicy wpisuje się numery stanów wewnętrznych, do których przechodzi układ z danego stanu wewnętrznego pod wpływem stanu wejść zapisanego u góry kolumny.



Rys 15.14. Wykres czasowy (a) oraz pierwotna tablica przejść i wyjść (b) układu bramkowania generatora

Jeżeli stan wewnętrzny Q_i pod wpływem stanu wejść X_j nie zmienia się (pozostaje ten sam)

$$p(Q_i, X_j) = Q$$

to stan Q_i jest przy X_j stabilny, co oznacza się kółkiem w tablicy przejść.

Zakłada się, że równoczesna zmiana sygnałów x_1 i x_2 nie wystąpi (podobnie jak przy analizie elementów pamięci), a zatem nie jest możliwe np. przejście ze stanu (00) do stanu (11), lecz wystąpi zawsze (00)→(01)→(11) lub (00)→(10)→(11). Część miejsc w tablicy będzie, więc nieokreślona (wyklucza się odpowiednie przejścia), co oznacza się kreską.

Może się zdarzyć, że po wpisaniu wszystkich przejść wynikających z wykresu czasowego oraz stanów nieokreślonych w tablicy pozostaną jeszcze puste miejsca. Wypełnienie ich wymaga analizy pożądanej pracy układu. Na przykład przejście ze stanu wewnętrznego 4 pod wpływem stanu wejść X = (10) nie jest pokazane na wykresie, ale gdyby taki przypadek wystąpił (bardzo krótki impuls x_2), wówczas powinno nastąpić przejście do stanu l, ponieważ impuls x_l nie może zostać obcięty. Podobna analiza wskazuje, że ze stanu 5 pod wpływem X = (11) powinno nastąpić przejście do stanu 3.

Tablicę (kolumnę) wyjść sporządza się również na podstawie wykresu czasowego, wpisując stany wyjść Y odpowiadające poszczególnym stanom Q (rys. 15.14b, $Y = y$).

15.4.1. Minimalizacja tablic przejść i wyjść

Minimalizacja (skracanie) tablic przejść i wyjść polega na zmniejszaniu liczby stanów wewnętrznych, przez zastępowanie dwóch (lub więcej) wierszy tablicy jednym. Im mniejsza liczba stanów wystarczy do opisania działania układu, tym mniej elementów potrzeba zwykle do jego realizacji.

Zasady minimalizacji tablic podane zostaną dla układów asynchronicznych, w których obecność stanów stabilnych wpływa na sposób określania zgodności stanów.

Dwa stany można uznać za zgodne i zastąpić jednym, jeżeli pod wpływem każdego stanu wejść X przechodzą w stany niesprzeczne lub zgodne oraz odpowiadające im stany wyjść Y, przypisane stanom stabilnym przy tym samym X, są niesprzeczne. W wyniku łączenia otrzymuje się:

$$\left.\begin{matrix} Q \\ Q \end{matrix}\right\} \to Q, \quad \left.\begin{matrix} Q \\ - \end{matrix}\right\} \to Q, \quad \left.\begin{matrix} \textcircled{Q} \\ Q \end{matrix}\right\} \to \textcircled{Q}, \quad \left.\begin{matrix} \textcircled{Q} \\ - \end{matrix}\right\} \to \textcircled{Q}, \quad \left.\begin{matrix} - \\ - \end{matrix}\right\} \to -.$$

Przeszukiwanie maksymalnych zbiorów stanów zgodnych jest ułatwione dzięki wyraźnemu uporządkowaniu stanów w tablicach układów asynchronicznych. Jeżeli w tablicy pierwotnej w kolumnie X_i występuje stan Q_i, to wiadomo, że stany Q_i występują tylko w tej kolumnie, a jeden z nich jest stabilny (w wierszu Q_i).

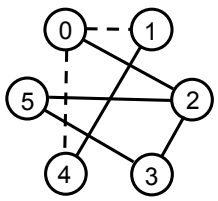
Na rysunku 15.15 pokazano przykłady minimalizacji, wychodząc z pierwotnej tablicy przejść i wyjść otrzymanej na rys. 15.14 dla układu bramkowania generatora.

a)

Q \ x_1x_2	00	01	11	10	y
0	(0)	2	-	1	0
1	0	-	4	(1)	1
2	0	(2)	3	-	0
3	-	2	(3)	5	0
4	-	2	(4)	1	1
5	0	-	3	(5)	0

Q'

b)



c)

Q \ x_1x_2	00	01	11	10	y
(0) - 0	(0)	2	-	1	0
(1,4) - 1	0	2	(1)	(1)	1
(2,3,5) - 2	0	(2)	(2)	(2)	0

d)

Q \ x_1x_2	00	01	11	10
(0,1,4) - 0	(0)	1	(0)	(0)
(2,3,5) - 1	0	(1)	(1)	(1)

Q'

e)

Q \ x_1x_2	00	01	11	10
0	0	0	1	1
1	0	0	0	0

y

Rys. 15.15. Minimalizacja (skracanie) tablicy przejąć układu bramkowania generatora: a) tablica pierwotna, b) wykres skracania, c) mimnalna tablica przejąć i wyjść układu More'a d) minimalna tablica przejść układa Mealy'a. e) tablica wyjść układu Mealy'ego

Najprostszym kryterium niezgodności stanów jest sprzeczność - wyjść Y dla stanów stabilnych przy tym samym X i od wyszukania takich stanów należy rozpocząć analizę. Dodatkową pomocą może być wykres skracania (graf), którego wierzchołki oznaczają poszczególne stany, a gałęzie oznaczają zgodność odpowiednich stanów. Maksymalny zbiór stanów zgodnych będzie wskazany przez największy zbiór nawzajem połączonych wierzchołków grafu. Wybierając najmniejszą liczbę takich zbiorów, obejmującą wszystkie stany bez ich powtarzania, uzyskuje się minimalny zestaw stanów zgodnych. Każdy zbiór tego zestawu można zastąpić jednym stanem, uzyskując w ten sposób tablicę skróconą (minimalną).

Tablica pierwotna jest zawsze tablicą Moore'a, natomiast skracanie można prowadzić dwiema drogami: dla uzyskania minimalnej tablicy Moore'a lub minimalnej tablicy Mealy'ego. Nie można z góry przewidzieć, która droga prowadzi do prostszego układu, należy, więc zbadać obydwie. Dlatego na wykresie skracania stany zgodne o wyjściach niesprzecznych łączy się linią ciągłą, a stany zgodne o wyjściach sprzecznych linią przerywaną.

Wykorzystywanie tylko połączeń liniami ciągłymi prowadzi do układu Moore'a, a wszystkich połączeń — do układu Mealy'ego.

W analizowanym przykładzie minimalna tablica Moore'a ma trzy stany {0;1,4;2,3,5}, a minimalna tablica Mealy'ego dwa stany {0,1,4;2,3,5}.

Tablicę wyjść układu Mealy'ego buduje się na podstawie tablicy pierwotnej, przypisując odpowiednie sygnały wyjściowe stanom stabilnym tablicy przejść. Wyznaczenie Y dla stanów niestabilnych wymaga analizy zachowania się układu:

a) niestabilnemu stanowi wewnętrznemu 1 w pierwszym wierszu tablicy (rys. 15.15d) odpowiada $y = 0$, ponieważ pod wpływem $X = 01$ układ przechodzi ze stanu 0 (tablica pierwotna, $y = 0$) do stanu 2 ($y = 0$), a więc musi być zachowana trwała wartość $y = 0$;

b) niestabilnemu stanowi wewnętrznemu 0 w drugim wierszu tablicy odpowiada $y = 0$, ponieważ pod wpływem $X = 00$ układ przechodzi ze stanu 5 (tablica pierwotna, $y = 0$) do stanu 0 ($y = 0$).

15.4.2. Kodowanie stanów wewnętrznych. Zjawisko wyścigu

Kodowanie minimalnej tablicy przejść Moore'a lub Mealy'ego polega na zastąpieniu stanów wewnętrznych Q, oznaczanych zwykle w procesie syntezy za pomocą kolejnych liczb naturalnych, konkretnymi ciągami sygnałów dwustanowych (Q_1, Q_2, ..., Q_k). Stany wejść X i wyjść Y od początku występują w postaci ciągów sygnałów (x_1, x_2, ..., x_n) oraz (y_1, y_2, ..., y_n), a więc są określone w sposób jednoznaczny dla projektanta.

Pierwszym problemem, który powstaje przy kodowaniu stanów we-

wnętrznych jest określenie liczby k sygnałów Q potrzebnych do opisania K stanów. Im więcej elementów pamięci (przerzutników) z wyjściami Q zawiera układ, tym bardziej jest skomplikowany i zawodny, należy więc dążyć do jak najmniejszej możliwej liczby k, którą można wyznaczyć z zależności

$$2^{k-1} < K \leq 2^k,$$

określającej minimalną liczbę bitów potrzebnych do przedstawienia liczby K w postaci dwójkowej.

Ustalenie relacji pomiędzy kolejnymi stanami Q a ciągami wartości (Q_1, Q_2, ..., Q_k) jest bardziej złożone. Należy pamiętać m. in., że tablica przejść i wyjść jest po zakodowaniu doprowadzana do postaci tablicy Karnaugha, z której otrzymywane są następnie pewne funkcje kombinacyjne (wzbudzenia). Na postać tych funkcji istotny wpływ ma to, czy sąsiadujące ze sobą w tablicy stany zakodowane zostaną jako np. 000 i 001, czy jako 000 i 111, gdyż w pierwszym przypadku istnieją większe możliwości sklejeń, a więc uproszczenia funkcji i układu.

Podobnych wymagań, prowadzących w rezultacie do jak najprostszego układu, jest więcej. Porównanie wszystkich możliwych sposobów kodowania jest jednak zbyt uciążliwe (dla $K = 5$ jest ich 140, dla $K = 9$ już 10 milionów [7]), w praktyce korzysta się, więc z pewnych metod nie gwarantujących wprawdzie zawsze najprostszego rozwiązania, ale uwzględniających wiele warunków mających wpływ na złożoność układu i poprawność jego działania.

Najczęściej stosowana jest metoda kodowania oparta na rachunku podziałów [7].

Podział π jest to pełny (tzn. zawierający wszystkie elementy rozpatrywanego zbioru) zbiór bloków (podzbiorów), które oznacza się za pomocą kreski nad ich elementami. Na przykład $\pi = (\overline{1,3,5}\,;\overline{2,4},\,6)$ jest trójblokowym podziałem zbioru {1,2,3,4,5,6}. Podziałem najmniejszym jest podział o blokach jednoelementowych oznaczany symbolem 0, np.

$$\{\bar{1};\bar{2};\bar{3};\bar{4};\bar{5};\bar{6}\} = 0,$$

a podziałem największym jest podział o jednym bloku, oznaczony symbolem l, np.

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

Dla dowolnego podziału n spełniony jest warunek

$$0 < \pi < 1.$$

Iloczyn podziałów π_1 i π_2 tego samego zbioru jest podziałem, którego blokami są przekroje (wspólne elementy) bloków π_1 z blokami π_2. Na przykład

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

$$\{\overline{1,2};\bar{3};\overline{4,5}\}\{\overline{1,3,4}\ \ \overline{2,5}\} = \{\bar{1};\bar{2};\bar{3};\bar{4};\bar{5};\bar{6}\} = 0.$$

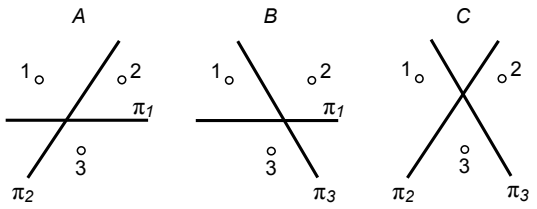
Jeżeli iloczyn podziałów jest równy 0, to kodowanie zgodne z podziałami przy-pisze każdemu stanowi inne oznaczenie kodowe.

Gdy układ ma trzy stany wewnętrzne, wówczas praktycznie mamy do wyboru trzy warianty kodowania, oznaczone *A, B, C* na rys. 15.16a (inne warianty kodowania nie różnią się niczym istotnym, mogą polegać jedynie na zamianie miejsc lub negacji *A, B, C,* co nie ma wpływu na złożoność układu). Na rysunku 15.16b pokazano odpowiadające wariantom *A, B*, *C* podziały π_1, π_2, π_3.

a)

Stan	A $\pi_1\pi_2$	B $\pi_1\pi_3$	C $\pi_2\pi_3$
1	00	00	00
2	01	01	11
3	11	10	10

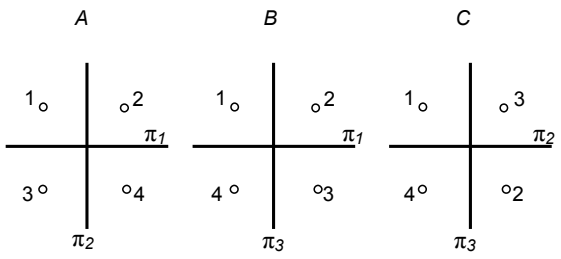
b)



c)

Stan	A $\pi_1\pi_2$	B $\pi_1\pi_3$	C $\pi_2\pi_3$
1	00	00	00
2	01	01	11
3	10	11	01
4	11	10	10

d)



Rys. 15.16. Sposoby kodowania układów: a), b) o trzech stanach wewnętrznych, c), d) o czterech stanach wewnętrznych

W wariancie *A* pierwszy bit swą, wartością 0 albo l wprowadza podział $\pi_1 = \{\overline{1,2}\,;\overline{3}\}$, a drugi bit podział $\pi_2 = \{^{1\,;\,2,\,3}\}$, w wariancie *B* mamy w, oraz $\pi_3 = \{\overline{1,3}\,;\overline{2}\}$ itd. Każda para stanów jest tu przedzielona linią podziału, co oznacza, że nie będzie dwóch stanów zakodowanych w taki sam sposób. Można to sprawdzić również za pomocą iloczynów podziałów; ponieważ $\pi_1\pi_2 = 0$, $\pi_1\pi_3 = 0$ oraz $\pi_2\pi_3 = 0$, zatem każde dwa z tych trzech podziałów mogą być wykorzystane do kodowania.

Gdy układ ma cztery stany wewnętrzne, wówczas istnieją również tylko trzy różne podziały spełniające warunek $k = 2$:

$$\pi_1 = \{\overline{1,2}\ \ \overline{3,4}\}, \quad \pi_1 = \{\overline{1,3}\ \ \overline{2,4}\}, \quad \pi_1 = \{\overline{1,4}\ \ \overline{2,3}\}.$$

Podobnie jak poprzednio, iloczyny $\pi_1\pi_2 = \pi_1\pi_3 = \pi_2\pi_3 = 0$, każde, więc dwa z tych trzech podziałów mogą być wykorzystane do kodowania — odpowiednie warianty kodowania przedstawiono na rys. 15.16 c, a ich ilustrację graficzną na rys. 15.16d.

Gdy układ ma liczbę stanów wewnętrznych większą od czterech, wówczas wyznaczenie podziałów służących do określenia nierównoważnych kodów jest trudniejsze i nie jest celowe wypisywanie wszystkich. Pamiętać natomiast należy, żo podziały, które mogą być użyteczne do kodowania, nazywane prawidłowymi, muszą spełniać dwa warunki:

a) liczba bloków wynosi 2,

b) liczba elementów w bloku nie przekracza 2^{k-1}.

Drugi warunek wynika z konieczności uzyskania iloczynu k podziałów równego 0.

Przykładem podziału prawidłowego dla $K = 6$ (wówczas $k = 3$) jest podział

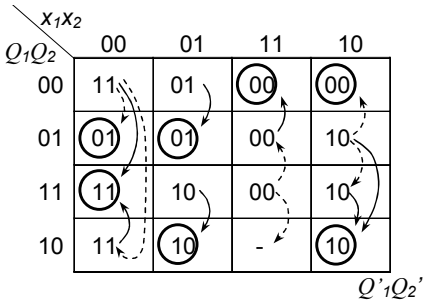
$$\{\overline{1, 2}; \overline{3, 4, 5, 6}\},$$

natomiast dla $K = 7$ (wówczas również $k = 3$) podział

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

nie jest prawidłowy.

Na przebieg kodowania układów asynchronicznych istotny wpływ ma możliwość powstawania tzw. wyścigów. Zjawisko to wytłumaczone zostanie na przykładzie zakodowanej tablicy przejść (stanom Q_1 *i* Q_2 przypisano już konkretne wartości), przedstawionej na rys. 15.17.



Rys. 15.17. Zakodowana tablica przejść układu, w którym mogą powstawać zjawiska wyścigów

Wyścig może wystąpić w tych przypadkach, kiedy przejście do nowego stanu wewnętrznego wymaga zmiany stanów obydwu elementów pamięci Q_1 i Q_2. Ponieważ nie istnieją dwa elementy fizyczne o identycznych właściwościach, zawsze więc jeden z elementów pamięci zareaguje wcześniej niż drugi i układ znajdzie się w stanie pośrednim, który może być stanem stabilnym. Jeżeli np. opisany tablicą układ jest w stanie $Q = (00)$, to pod wpływem $X = (00)$ powinien przejść do stanu $Q' = (11)$. W rzeczywistości mogą wystąpić dwa przypadki, pokazane liniami przerywanymi:

a) najpierw pojawi się stan pośredni $Q = (10)$; wówczas naturalne wymuszenia kierują układ do właściwego stanu $Q' = (11)$, gdyż ze stanu $Q = (10)$ pod wpływem $X = (00)$ następuje właśnie przejście do $Q' = (11)$;

b) najpierw pojawi się stan pośredni Q = (01); stan ten przy X = (00) jest stabilny i układ pozostanie w nim.

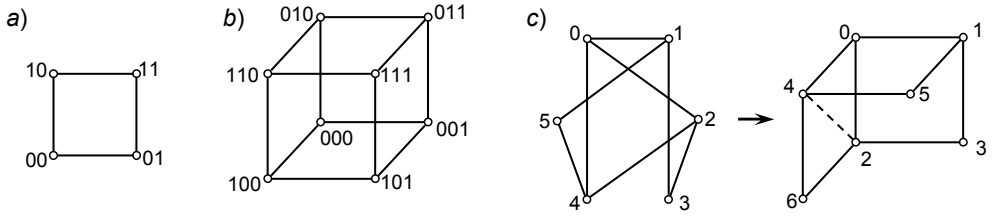
Opisane w punkcie b) zjawisko, polegające na uzyskiwaniu błędnego stanu stabilnego na skutek różnych czasów przełączania elementów lub opóźnień na drodze przekazywania sygnału, nazywa się *wyścigiem krytycznym*.

W drugiej kolumnie tablicy zmienia wartość najwyżej jeden sygnał Q, wyścigu więc nie ma. W trzeciej kolumnie, przy Q = (11), występuje wyścig elementów pamięci, ale wystarczy w miejsce stanu nieokreślonego wpisać (00), aby wszystkie możliwe przejścia prowadziły do właściwego celu. Wyścig taki nazywa się *niekrytycznym*.

W czwartej kolumnie tablicy występuje wyścig krytyczny przy Q = (0l), który przy opóźnionej reakcji Q_t prowadzi do błędnego stanu stabilnego Q' = (00), zamiast do Q' = (10). W tym przypadku niewłaściwe przejście można zlikwidować wykorzystując fakt, że nie tylko ze stanu Q = (01), lecz także z Q = (11) układ przechodzi do Q' = (10). Można zatem w kratce o współrzędnych Q = (01), X= (10) zamiast Q' = (10) napisać Q' = (11), co spowoduje, że ze stanu Q = (01) pod wpływem X= (10) układ przejdzie kolejno drogę (01) → (11) → (10). Ten sposób usuwania wyścigu nosi nazwę przejścia cyklicznego i związany jest z pewnym wydłużeniem czasu reakcji układu. Przejście cykliczne można wprowadzić tylko wtedy, gdy stan pośredni nie powoduje krótkotrwałych zmian sygnałów wyjściowych, które mogą być niedopuszczalne.

Jeżeli zlikwidowanie wyścigu przez usunięcie stanów nieokreślonych lub wprowadzenie przejścia cyklicznego nie jest możliwe, to należy zmienić kod stanów wewnętrznych.

Najprostszym sposobem uniknięcia wyścigów krytycznych jest takie kodowanie, aby każde dwa stany Q i Q', takie że Q przechodzi w Q', miały sąsiednie wyrażenia kodowe, tzn. aby odpowiadające im sygnały Q różniły się tylko w jed-



Rys. 15.18. Kodowanie układów zabezpieczające przed wystąpieniem wyścigów: a) dla 2-4 stanów wewnętrznych, bj dla 5-8 stanów wewnętrznych, c) usunięcie wyścigu przez wprowadzenie dodatkowego stanu 6

nym miejscu. Dla liczby stanów mniejszej lub równej 8 pomocne w tym mogą być uproszczone wykresy przejść o postaci kostki (rys. 15.18), zbudowane tak, że każde dwa połączone ze sobą stany spełniają podany wyżej warunek.

Przedstawiony sposób nie uwzględnia jednak możliwości uproszczenia układu, dlatego lepsze wyniki daje opisana wcześniej metoda kodowania oparta na ra-

chunku podziałów, którą należy uzupełnić dodatkowymi warunkami pozwalającymi wyeliminować zjawiska wyścigów. Warunki te polegają na przyjęciu do kodowania takich podziałów prawidłowych π, które separują wszystkie bloki podziałów wewnętrznych. Wówczas oznaczenia kodowe wszystkich stanów przechodzących do określonego stanu Q' będą miały część wspólną inną niż stany pozostałe (występujące w innym bloku n), co wystarcza do uniknięcia wyścigów. Bliższe omówienie postępowania podane jest w pracy [7],

15.4.3. Tablice wzbudzeń. Zjawisko hazardu

Po zakodowaniu tablicy przejść następnym etapem syntezy jest utworzenie tablicy wzbudzeń, która ma takie same współrzędne jak tablica przejść, ale wewnątrz zamiast stanów Q' należy wpisać sygnały wzbudzenia q, przeprowadzające układ ze stanu Q do Q'.

Wyznaczanie funkcji wzbudzeń zależy od rodzaju przyjętych do realizacji układu podstawowych elementów pamięci, nazywanych też automatami elementarnymi. W układach asynchronicznych wybiera się zwykle jedną z dwóch dróg realizacji pamięci:

a) wprowadzenie pętli sprzężenia zwrotnego,

b) zastosowanie przerzutnika *w-z*, opisanego w punkcie 15.2.

Najmniejszą częścią układu mającą własność pamięci jest sprzężenie zwrotne łączące wyjście układu z wejściem, tzw. przerzutnik (automat elementarny)

Q \ q_D	0	1
0	0	1
1	0	1

Rys. 15.19, Tablica przejść przerzutnika typu D (sprzężenie zwrotne)

typu D, którego tablica przejść podana jest na rys. 15.19. Wartość sygnału wejściowego (wzbudzenia) q_D przekazywana jest tu bez zmian na wyjście, występuje jedynie pewne opóźnienie τ i równanie przerzutnika można zapisać

$$Q' = q_D$$

Każdą zakodowaną tablicę przejść można więc w tym przypadku uważać za tablicę wzbudzeń, wyznaczającą sygnały wzbudzenia $q_D = Q'$ na podstawie Q *i* X. Jeżeli zakodowana tablica przejść ma postać tablicy Karnaugha, można z niej bezpośrednio (jak dla układów kombinacyjnych) wyznaczyć minimalne postacie funkcji określających $q_{Di} = Q'_i$. O tym, że jest to układ sekwencyjny, świadczą tylko sprzężenia zwrotne, powstałe przez połączenie wyjść Q'_i

z wejściami Q_i. Należy sobie zdawać sprawę, że są to te same sygnały, a indeks służy jedynie do rozróżnienia kolejności ich występowania.

Opisane postępowanie zilustrowane zostanie przykładem, będącym kontynuacją syntezy układu bramkowania generatora. Minimalna tablica przejść układu Mealy'ego (rys. 15.15 d) może w tym przypadku być traktowana równocześnie jako tablica wzbudzeń, ponieważ $q_D = Q'$. Po przeprowadzeniu minimalizacji funkcji metodą tablic Karnaugha według rys. 15.20a otrzymamy, w przypadku sklejania jedynek

$$Q' = \bar{x}_1 x_2 + x_1 Q,$$

a w przypadku sklejania zer

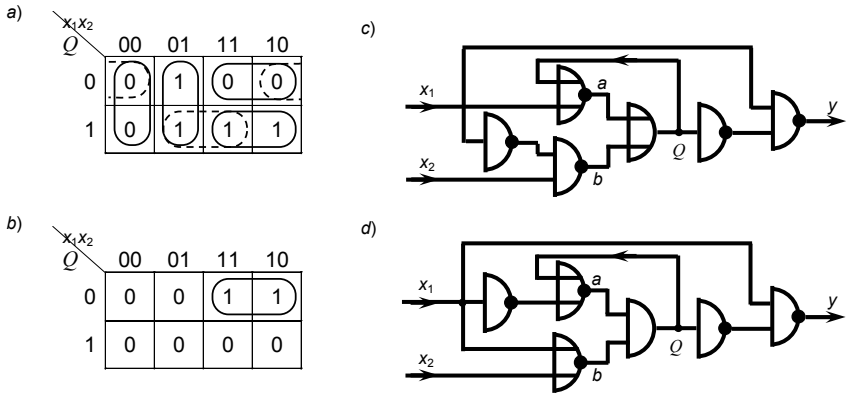
$$Q' = (x_1 + x_2)(\bar{x}_1 + Q)$$

Minimalizując funkcję wyjść według rys. 15.20b (na podstawie tablicy wyjść, rys. 15.15e) otrzymamy

$$y = x_1 \bar{Q}$$

Schemat układu logicznego odpowiadający równaniom (15.10) i (15.12) przedstawiono na rys. 15.20c, a odpowiadający równaniom (15.11) i (15 12) na rys. 15.20d.

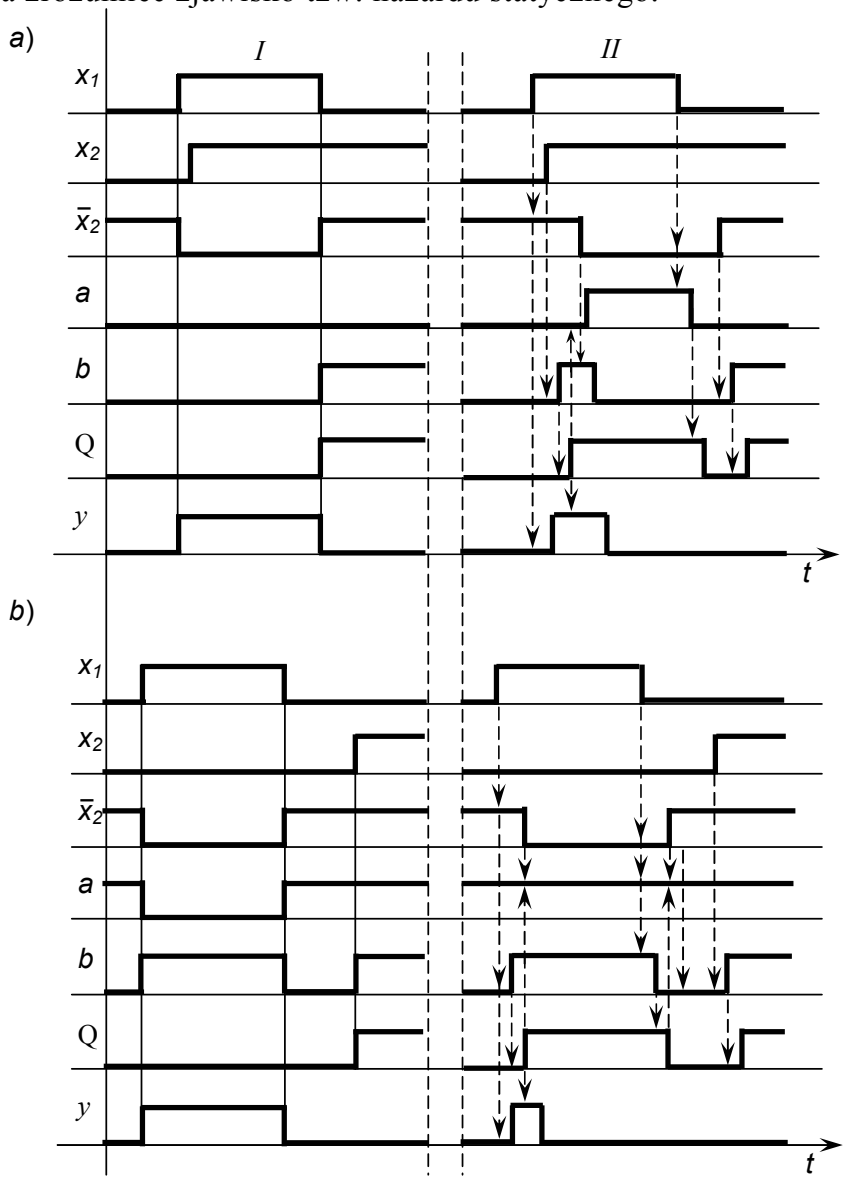
Gdyby opóźnienia wnoszone przez poszczególne elementy i linie łączące nie miały wpływu na pracę układów z rys. 15.20c,d, ich działanie byłoby poprawne.



Rys. 15.20. Układ bramkowania generatora: a), b) minimalizacja funkcji logicznych na podstawie tablicy wzbudzeń i tablicy wyjść układu Mealy'ego, c), d) dwa warianty schematu układu logicznego

Jednak w danym przypadku sygnał x_1 wpływa na stan Q dwiema drogami (zaznaczonymi liniami pogrubionymi na rysunku), przy czym na jednej z nich znajduje się dodatkowo element negacji, co pozwala przewidywać, że czas-

przesyłania sygnału tą drogą będzie dłuższy. Na rysunku 15.21 przedstawiono przykładowe przebiegi czasowe w przypadkach idealnym i rzeczywistym, co pozwala zrozumieć zjawisko tzw. hazardu statycznego.



Rys. 16.21. Przebiegi czasowe sygnałów idealne (7) i rzeczywiste (77) w układach bramkowania generatora: a) według rys. 15.20c, b) według rys. 15.20d

Hazardem statycznym nazywamy pojawienie się niewłaściwego stanu Q na skutek różnicy w czasie przesyłania sygnału po dwóch różnych drogach. Hazard pojawia się wówczas, gdy w funkcji logicznej opisującej Q jeden z argumentów występuje raz z negacją, a drugi raz bez negacji. Przypadki takie można rozpoznać już na podstawie tablic Karnawgha, gdyż odpowiada im stykanie się ze sobą dwóch grup sklejanych kratek, właśnie na linii zmiany wartości tego argumentu

(na rys. 15.20a linią tą jest środek tablicy w przypadku sklejania jedynek, a krawędź tablicy w przypadku sklejania zer — zmiana wartości x_1).

Jeżeli część układu realizująca funkcję odpowiadającą jednej grupie sklejanych kratek działa szybciej niż część układu realizująca funkcję odpowiadającą drugiej grupie, to hazard pojawia się wówczas, gdy zakończyło się już działanie pierwszej części układu, a nie zaczęło się jeszcze działanie drugiej części. Zjawisko takie ma miejsce również w układach kombinacyjnych, ale tam powodować może jedynie krótkotrwały niewłaściwy impuls na wyjściu *), natomiast w układach sekwencyjnych — ze względu na sprzężenie zwrotne — może nastąpić podtrzymanie niewłaściwego stanu Q *i* odpowiednio sygnału wyjściowego *y*.

Analiza wykresów podanych na rys. 15.21 wskazuje, że w układzie z rys. 15.20c hazard obserwować można tylko wtedy, gdy sygnał bramkujący x_2 pojawi się po wystąpieniu x_1 = l, ale przed wystąpieniem na wyjściu elementu negacji $\bar{x}_1 = 0$. Prawdopodobieństwo hazardu jest, więc bardzo małe, tym mniejsze, im mniejsze opóźnienia wprowadza element negacji. Natomiast układ z rys. 15.20d zupełnie nie spełnia założeń, gdyż nie przenosi bez zniekształcenia sygnałów rc_1 generatora przy zerowym sygnale bramkującym.

Dla usunięcia hazardu statycznego wystarczy dodatkowo skleić grupę kratek na styku wcześniej sklejonych grup (to dodatkowe sklejenie zaznaczono linią kreskowaną na rys. 15.20a), aby zapewniła ona utrzymanie właściwej wartości Q w stanie przejściowym, gdy nie działają dwie poprzednie grupy. Oczywiście układ rozbudowuje się wówczas, a wolna od hazardu funkcja Q' ma w przypadku sklejania jedynek postać

$$Q' = \bar{x}_1 x_2 + x_1 Q + x_2 Q, \tag{15.13}$$

a w przypadku sklejania zer

$$Q' = (x_1 + x_2)(\bar{x}_1 + Q)(x_2 + Q). \tag{15.14}$$

Schemat układu logicznego odpowiadający równaniom (15.12) i (15.13) przedstawiono na rys. 15.22a, a odpowiadający równaniom (15.12) i (15.14), po przekształceniu ich do postaci zawierającej elementy NOR

$$y = \overline{\bar{x}_1 + Q}, \tag{15.14}$$

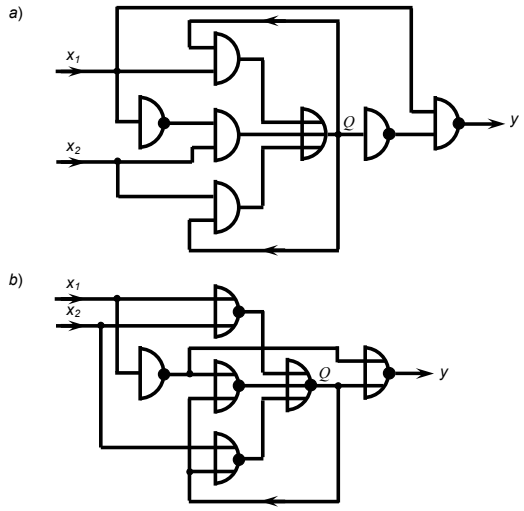
$$Q' = \overline{(\overline{x_1 + x_2}) + (\overline{\bar{x}_1 + Q}) + (\overline{x_2 + Q})}, \tag{15.14}$$

przedstawiono na rys. 15.14b.

Powstawanie hazardu można również niekiedy wyeliminować metodami układowymi, przez odpowiednie przyspieszenie lub opóźnienie reakcji poszczególnych elementów, kompensujące różnicę czasów działania z rozwiązania pierwotnego.

*) Niekiedy nieistotny, bo przy znikomym czasie trwania impulsu przetoczenie elementów odbierających go albo nie nastąpi, albo nie będzie mieć praktycznie żadnego wpływu na przebieg procesu sterowanego.

Schemat układu bramkowania generatora podany na rys. 15.22b wskazuje na możliwość zastosowania drugiej drogi realizacji pamięci, przez wydzielenie przerzutników *w-z*, zbudowanych z elementów NOR według rys. 15.3.



Rys. 15.22. Schematy układów bramkowania generatora *z* wyeliminowanym zjawiskiem hazardu

Tablica przejść przorzutnika *w-z* podana jest na rys. 15.23a, a jego tablica wzbudzeń na rys. 15.23b. W stosunku do rys. 15.3 oznaczenia są następujące: $w = x_1$ z= x_2, $Q = y_1$ $\overline{Q} = y_2$. Obowiązuje założenie $w \cdot z = 0$, tzn. obydwa sygnały wejściowe nie mogą być równocześnie równe 1.

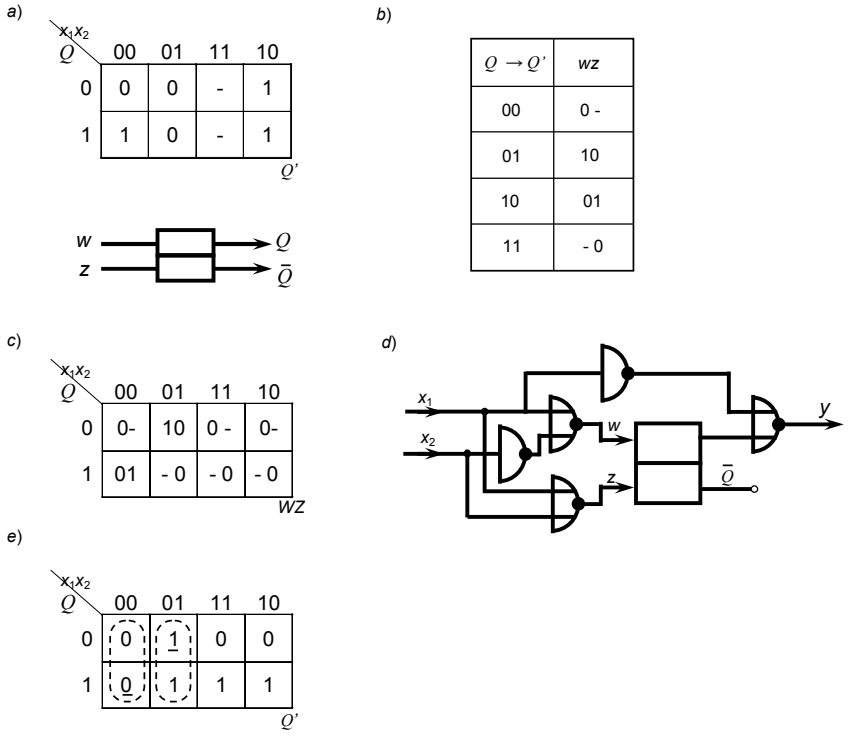
Skrócona (minimalna) tablica przejść układu Mealy'ego z rys. 15.20a może za pomocą tablicy z rys. 15.23b zostać przekształcona w tablicę wzbudzeń z rys. 15.23c, z której otrzymuje się

$$w = \overline{x}_1 x_2, \qquad z = \overline{x}_1 \overline{x}_2,$$

Schemat układu bramkowania generatora z przerzutnikiem *w-z* przedstawiono na rys. 15.23d.

W przypadkach bardziej złożonych układów korzystne jest przedstawienie tablicy wzbudzeń w postaci uwypuklającej te stany w tablicy przejść, które przy danym *X* ulegają zmianie. Tablica wzbudzeń odpowiadająca rys. 15.23a miałaby wówczas postać pokazaną na rys. 15.23e. Sygnał *w* musi mieć wartość l w tych kratkach, gdzie występuje l i musi mieć wartość 0 tam, gdzie występuje jakiekolwiek 0; kreski i l oznaczają dowolną wartość *w*. Sygnał *z* musi mieć wartość l w tych kratkach, gdzie występuje 0 oraz musi mieć wartość 0

tam, gdzie występuje jakiekolwiek 1; kreski i 0 oznaczają dowolną wartość *z*. Na podstawie tych reguł otrzymujemy identyczne wartości *w* i *z* jak poprzednio.



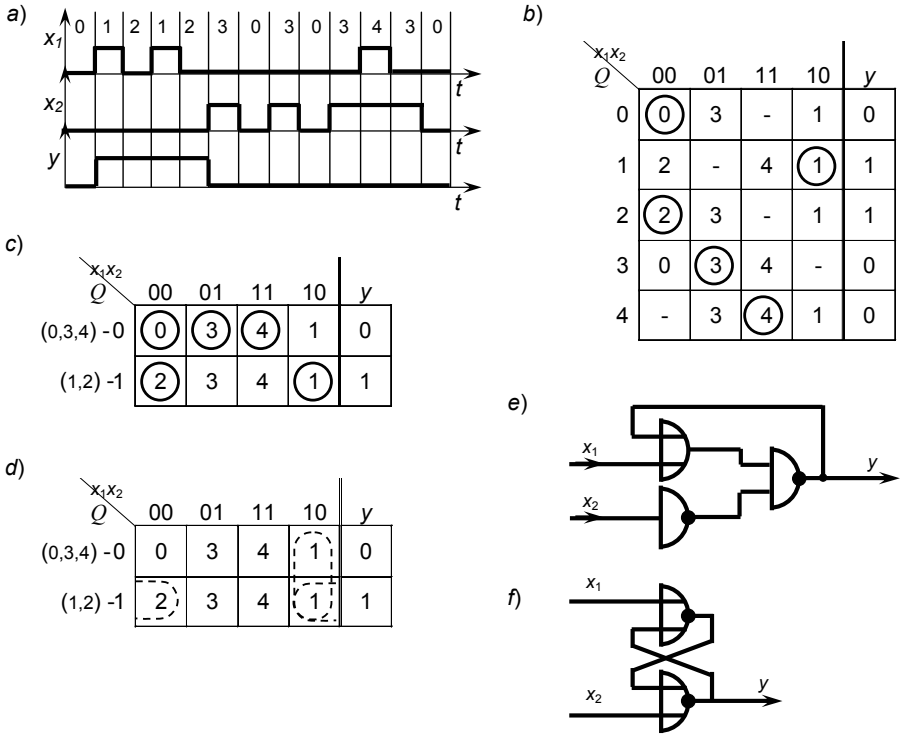
Rys. 15.23. Układ bramkowania generatora z przerzutnikiem *w-z:* a) tablica przejść przerzutnika, b) tablica wzbudzeń przerzutnika, c) tablica wzbudzeń układu Mealy'ogo, d) schemat układu logicznego, e) inny sposób przedstawienia tablicy wzbudzeń

Dla utrwalenia podanych zasad postępowania zaleca się przeprowadzić syntezę rozważanego układu bramkowania generatora na podstawie minimalnej tablicy przejść układu Moore'a (rys. 15.15c) — wynik powinien być taki sam.

15.5. Przykład syntezy przerzutnika

Niniejszy punkt traktowany jest jako krótkie podsumowanie zasad syntezy układów asynchronicznych statycznych na prostym przykładzie przerzutnika. Kolejne etapy postępowania przedstawiono na rys. 15.24.

Zadanie układu podano w formie wykresu czasowego. Z minimalnej tablicy przejść układu Moore'a wynika, że stan wewnętrzny układu da się zakodować stanem jednego automatu elementarnego, odpadają więc problemy



Rys. 16.24. Synteza przerzutnika: a) wykres czasowy, b) pierwotna tablica przejść i wyjść, c) minimalna tablica przejść i wyjść, d) tablica wzbudzeń, e), f) schematy układu logicznego

związane z wyścigiem. Jeżeli decydujemy się przeprowadzić syntezę z wykorzystaniem automatu elementarnego typu *D* (sprzężenie zwrotne), to tablicę z rys. 15.24d można traktować również jako tablicę wzbudzeń. Minimalizacja funkcji metodą tablic Karnaugha daje wynik

$$y = Q' = Q\bar{x}_2 + x_1\bar{x}_2 = \bar{x}_2(Q + x_1), \tag{15.15}$$

któremu odpowiada schemat układu logicznego według rys. 15.24e.

Korzystniej jest przekształcić równanie (15.15) do postaci

$$y = Q' == \overline{x_2 + \overline{(Q + x_1)}}. \tag{15.15a}$$

Otrzymuje się wówczas klasyczny schemat przerzutnika z dwóch elementów NOR (rys. 15.241).

Bibliografia

[1] Bromirski J.t *Teoria automatów*, Warszawa 1969, WNT.

[2] Głuszkow W. M., *Synteza automatów cyfrowych*, Warszawa 1968, WNT.

[3] Grzybek M., Misiurewicz P., *Wybrane tranzystorowe układy cyfrowe*, Warszawa 1969, WNT.

[4] Jaczewski J., *Układy logiczne dla zastosowań przemysłowych*, Warszawa 1970, PWN.

[6] Kaczanowski S., Olszewski M., Wański Z., *Plynowe elementy i układy logiczne*, Warszawa 1971. WKŁ.

[6] Siwiński J., *Układy przełączające w automatyce*, Warszawa 1968, WNT.

[7] Traczyk W., *Układy cyfrowe automatyki*. Warszawa 1974, WNT.

[8] Waligórski S., *Układy przełączające. Elementy teorii i projektowanie*. Warszawa 1971, WNT.