5. UKŁADY TYPOWE

5.1. WIADOMOŚCI OGÓLNE

Opisane wyżej metody syntezy układów kombinacyjnych i sekwencyjnych można stosować bez pomocy maszyny cyfrowej tylko wówczas, gdy liczba sygnałów wejściowych układu nie jest duża. W przypadku urządzeń złożonych, o licznych sygnałach wejściowych i wyjściowych, projektowanie całego urządzenia jako jednego układu sekwencyjnego byłoby bardzo trudne i dlatego zazwyczaj jest ono dzielone na mniejsze części, łatwiejsze do opisania i projektowania. Części te (zwane *blokami* lub *zespołami funkcjonalnymi*) są dobierane tak, by realizowana przez nie funkcja była możliwie prosto i jednoznacznie określona, co upraszcza budowanie odpowiedniego układu, a także ułatwia eksploatację całego urządzenia (wymianę zespołów, odnalezienie uszkodzenia, kontrolę działania itp.). Dodatkowa zaleta podziału na bloki wynika z faktu, że w urządzeniach cyfrowych (nawet wówczas, gdy służą one do bardzo różnych celów) niektóre bloki często powtarzają się, co umożliwia zbudowanie dowolnego urządzenia w 50...90% z kilku lub kilkunastu typowych bloków o standardowych funkcjach. Ta cecha urządzeń cyfrowych została wykorzystana w systemach średniej skali integracji (MSI), zawierających gotowe bloki, zamiast oddzielnych elementów, jak w małej skali integracji (MSI). Zalety takiego rozwiązania są oczywiste.

Najczęściej powtarzające się zespoły funkcjonalne urządzeń to liczniki, konwertery kodów, rejestry i sumatory. Każdy z tych zespołów może występować w licznych wersjach, wynikających z odmienności zadań lub elementów, z których jest budowany. Opisanie wszystkich stosowanych rozwiązań bloków wykracza poza ramy tej książki i nie jest celowe, gdyż wiele dalszych odmian można łatwo uzyskać, powtarzając czynności niżej opisane, lecz z zastosowaniem innego kodu lub innych

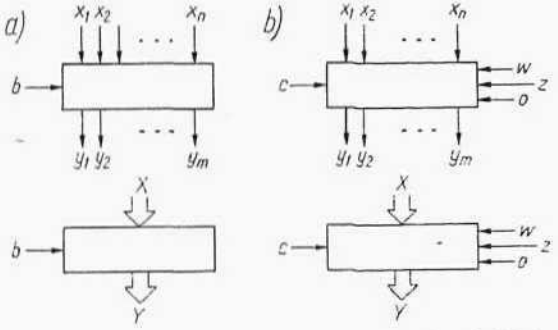
elementów. Dla zwiększenia praktycznej przydatności opisywanych schematów, będą one przedstawiane w trzech wersjach:

— z wykorzystaniem elementów I, LUB, NIE i dowolnych przerzutników — dla zilustrowania zasady działania lub w przypadkach, gdy realizacja z elementów NOR, NAND jest oczywista;

— z wykorzystaniem elementów NOR i przerzutników z elementami impulsowymi — gdy ich zastosowanie wpływa na strukturę układu;

— z wykorzystaniem elementów NAND i przerzutników synchronizowanych — gdy ich zastosowanie wpływa na strukturę układu lub gdy struktura ta jest przyjęta w realizacjach MSI.

Jak z tego wynika, pierwsza wersja to schematy ideowe, druga — to praktyczne schematy dla elementów z systemów automatyki przemysłowej, a trzecia wersja — to schematy dla układów scalonych. Inne odmiany (*np. NOR i JK*) *można tworzyć w sposób podobny do opisanych.*



Rys. 5-1. Schematowe oznaczenia bloków kombinacyjnych (a) i sekwencyjnych (b)

Przy rysowaniu schematów dużych układów, zestawianych z bloków, jest celowe używanie symboli graficznych dla oznaczenia tych bloków. Ogólny symbol bloku kombinacyjnego, przetwarzającego stany X w Y przedstawiono na rys. 5-1a, a bloku sekwencyjnego (z pamięcią) na rys. 5-1b. Oprócz *sygnałów informacyjnych* x i y występują tu jeszcze *sygnały sterujące*:

b — *bramkujący* — $Y = f(X)$, gdy $b = 1$ oraz $Y = \text{const}$, albo $Y = g(X)$, gdy $b = 0$; sygnałów tego typu może być wiele;

w — *wpisujący* — wprowadza stan X do pamięci układu;
o — *odczytujący* — wprowadza stan pamięci na wyjście Y układu;
z — *zerujący* — ustawia określony stan pamięci, np. same zera;
c — *taktujący* — zmienia stan pamięci na podstawie jej stanu poprzedniego (i ewent. sygnałów zewnętrznych x_0).

Sygnały *w*, *z* i *c* nie zawsze odpowiadają wejściom przerzutników o tych samych nazwach. Sygnały *x* mogą występować w kilku grupach (np. X_1 i X_2 albo X i S); niekiedy wprowadza się jeszcze dodatkowe sygnały x_0, y_0 i y_p. Wewnątrz prostokąta symbolizującego blok można umieszczać oznaczenia, wyróżniające rodzaj funkcji, stosowane kody itp.

5.2. KONWERTERY KODÓW I KOMUTATORY

5.2.1. KODERY

Układy służące do zamiany jednego kodu na drugi są nazywane *konwerterami kodów*, a ich szczególny przypadek — gdy kodem pierwotnym jest kod „1 z *n*" — to *kodery*. Jeśli zadaniem kodera jest zamiana kodu „1 z 4" na naturalny kod dwójkowy (kod „21"), to jego działanie można opisać tabelką

x_3	x_2	x_1	x_0	y_2	y_1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

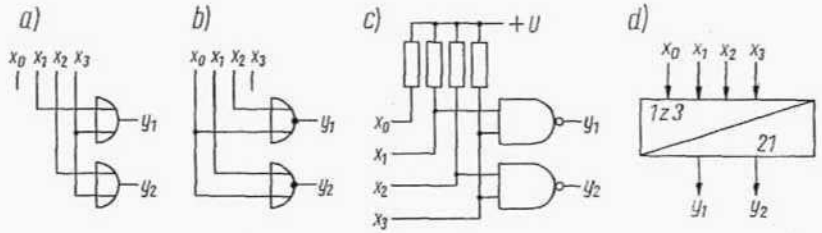
Nie trudno z niej wyznaczyć

$$y_1 = x_1 + x_3$$
$$y_2 = x_2 + x_3$$

Schemat tego prostego kodera jest przedstawiony na rys. 5-2a i wskazuje na sposób tworzenia innych układów tego typu: każdy sygnał x jest doprowadzany do wejścia tych elementów LUB, których sygnał wyjściowy y ma mieć wartość 1.

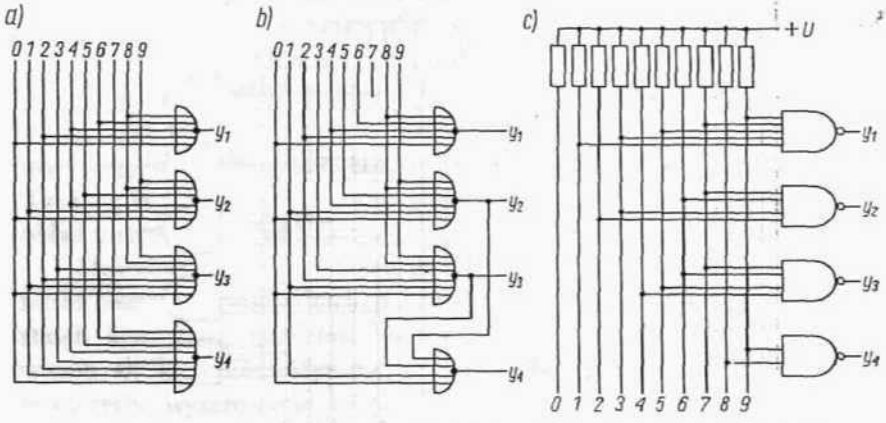
Jeśli koder ma być zbudowany z elementów NOR, przez zastosowanie połączeń z rys. 5-2a uzyskuje się sygnały $\bar{y}_1$ i $\bar{y}_2$, które mogą być użyteczne, gdy następne elementy NOR pełnią rolę iloczynów (np.

bramkujących wyjścia y sygnałem b). W przypadkach gdy jednak są potrzebne sygnały y_1 i y_2, stosuje się układ z rys. 5-2b, w którym każdy sygnał x jest doprowadzany do tych elementów NOR, których sygnał wyjściowy y ma mieć wartość 0.



Rys. 5-2. Schematy kodera z elementów LUB (a), NOR (b) i NAND (c) oraz jego symbol (d)

Jeśli koder jest budowany z elementów NAND, można wykorzystać połączenia z rys. 5-2a, ale z zanegowanymi sygnałami x. W wielu przypadkach sygnałowi $x = 1$ może odpowiadać dołączenie odpowiedniego przewodu do punktu o napięciu 0V (np. zestykiem przełącznika lub przycisku, elementem z otwartym kolektorem), a wówczas schemat kodera może mieć postać z rys. 5-2c. Zasady dołączania wejść są takie same jak w przypadku elementów LUB.



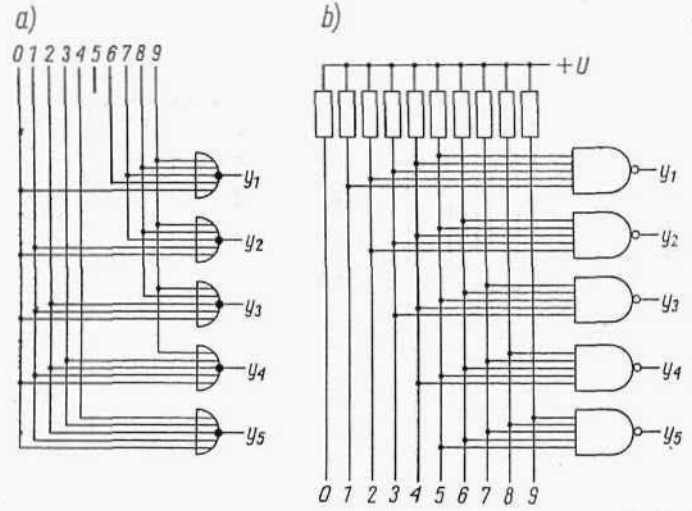
Rys. 5-3. Kodery kodu dwójkowo-dziesiętnego 8421: a) prosty z elementów NOR; b) optymalny z elementów NOR; c) specjalny z elementów NAND

Symbolem konwerterów kodów będzie prostokąt z jedną przekątną i wpisanymi nazwami kodów; dla rozpatrywanego kodera symbol jest przedstawiony na rys. 5-2d.

Najczęściej spotykaną grupą koderów są układy o dziesięciu wejściach, zamieniające sygnał z przełączników lub przycisków na jeden z kodów dwójkowo-dziesiętnych. Na rys. 5-3 przedstawiono przykłady takich koderów dla kodu naturalnego 8421, a więc układów zamieniających kod VIII w kod I (tabl. 1-2 i 1-3). Oznaczenia x_i zastąpiono dla uproszczenia cyfrą i; liczbę wyjściową tworzy ciąg $(y_4y_3y_2y_1)$. W rozwiązaniu z rys. 5-3b wykorzystano fakt (łatwy do zauważenia w tablicy 1-2), że gdy $y_2 = 1$ lub $y_3 = 1$ to $y_4 = 0$, czyli $F_1(y_2) \subseteq F^0(y_4)$ i $F^1(y_3) \subseteq F^0(y_4)$, co umożliwia realizację y_4 za pomocą y_2 i y_3. W otrzymanym układzie prostszy staje się jeden element, obciążenie sygnałów x jest mniejsze, ale należy uwzględniać stany przejściowe y_4.

Układy jednostopniowe (jak na rys. 5-3a) są niekiedy nazywane *prostymi*, a dwu- lub wielostopniowe (rys. 5-3b) — *optymalnymi*. Rozwiązanie kodera z rys. 5-3c zakłada uziemianie przewodów wejściowych.

Według opisanych wyżej zasad można zbudować kodery dowolnego



Rys. 5-4. Kodery kodu dwójkowo-dziesiętnego pseudopierścieniowego

innego kodu, z wykorzystaniem elementów LUB, NOR, albo NAND. Na rys. 5-4 pokazano jeszcze dwa przykłady dla dwójkowo-dziesiętnego kodu pseudopierścieniowego (VI). Zastosowanie układu wielostopniowego jest w tym przypadku niemożliwe.

5.2.2. DEKODERY

Konwerter, którego sygnały wyjściowe przedstawione są w kodzie „1 z n" nazywany jest *dekoderem*. Jeśli zadaniem dekodera jest zamiana dwubitowego kodu naturalnego „21" na kod „1 z 4", to jego działanie można opisać tabelką:

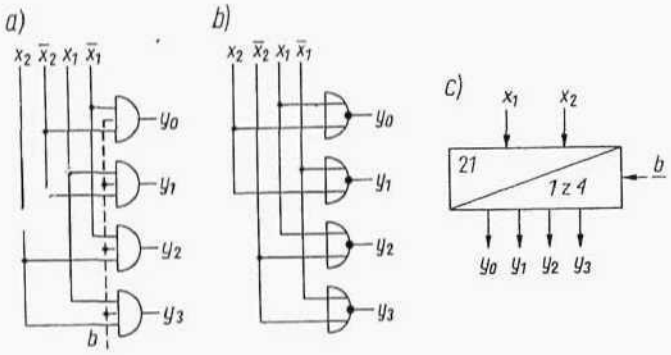
x_2	x_1	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

z której wyznacza się: $y_0 = \bar{x}_1\bar{x}_2, y_1 = x_1\bar{x}_2, y_2 = \bar{x}_1 x_2, y_3 = x_1 x_2$.

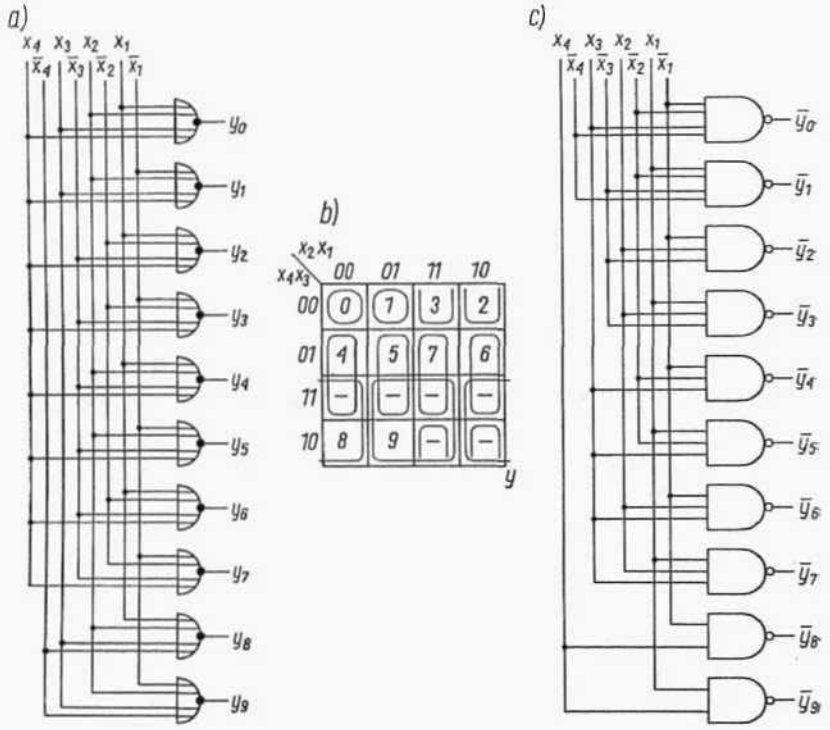
Schemat odpowiedniego układu jest przedstawiony na rys. 5-5a (z wykorzystaniem elementów I) i rys. 5-5b (z elementami NOR); Układy te różnią się negacjami sygnałów wejściowych. Realizacja iloczynu z elementów NAND wymaga użycia aż dwóch takich elementów i dlatego zazwyczaj dekodery z NAND'ów dostarczają sygnałów $\bar{y}$, które w miarę możności wykorzystuje się w takiej postaci, albo neguje.

Jak wykazuje powyższy przykład, dekoder można zbudować w postaci zespołu elementów realizujących pełne iloczyny zmiennych wejściowych; dekoder nazywany jest wówczas *pełnym*. Przykład dekodera pełnego kodu dwójkowo-dziesiętnego 8421 jest przedstawiony na rys. 5-6a. Taką samą strukturę, ale z zanegowanymi sygnałami x i y, będzie miał układ z elementów NAND.

Dekodery kodów dwójkowo-dziesiętnych (i innych, w których między liczbą wejść n i wyjść m zachodzi relacja $m < 2^n$) nie wykorzystują wszystkich kombinacji sygnałów wejściowych. W dekoderze pełnym pojawieniu się niewykorzystywanej (nieprawidłowej) kombinacji wartości x towarzyszy wyzerowanie sygnałów y, gdyż żaden z realizowanych iloczynów nie ma wartości 1. Jeśli można założyć, że nieprawidłowe kombinacje nigdy nie pojawią się na wejściu układu, to fakt ten trzeba wykorzy-

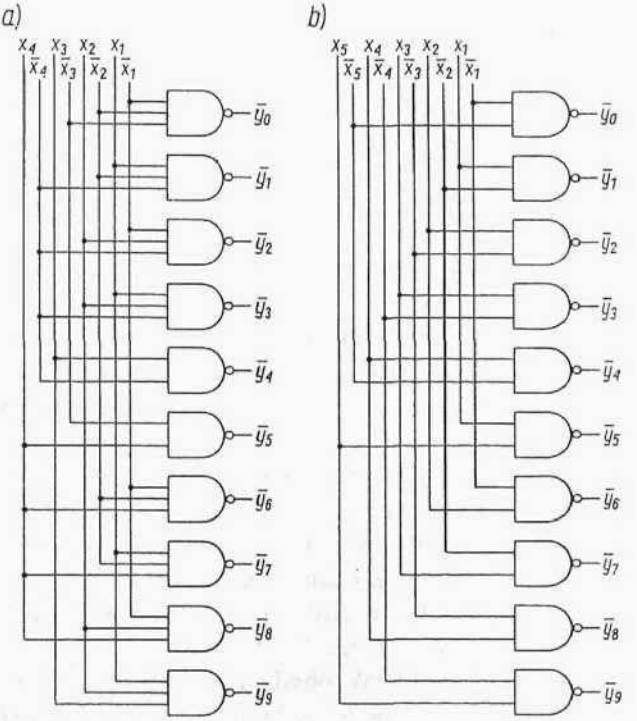


Rys. 5-5. Schematy dekodera z elementów I (a) i NOR (b) oraz jego symbol (c)



Rys. 5-6. Dekodery kodu dwójkowo-dziesiętnego 8421: a) pełny z elementów NOR; b) tablica Karnaugha; c) uproszczony z elementów NAND

stać dla uproszczenia układu. Możliwość uproszczenia wynika z wielofunkcyjnej tablicy Karnaugha, która dla kodu dwójkowo-dziesiętnego 8421 (I) ma postać jak na rys. 5-6b. Stan $y_i = 1$ odpowiada sytuacji, gdy w kratce i jest 1, a w pozostałych ponumerowanych — 0. Ponieważ



Rys. 5-7. Dekodery uproszczone kodów dwójkowo-dziesiętnych: a) Aikena; b) pseudopierścieniowego

równocześnie tylko dla jednego i może być $y_i = 1$, więc wszystkie funkcje dekodera można zapisywać tego typu tablicami i stosować znane metody minimalizacji funkcji. Na przykład z rys. 5-6b łatwo uzyskuje się funkcje:

$y_0 = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \qquad y_1 = x_1\bar{x}_2\bar{x}_3\bar{x}_4 \qquad y_2 = \bar{x}_1 x_2\bar{x}_3 \qquad y_3 = x_1 x_2\bar{x}_3$

itd., aż do $y_9 = x_1 x_4$.

Jedną z możliwych realizacji pokazano na rys. 5-6c.

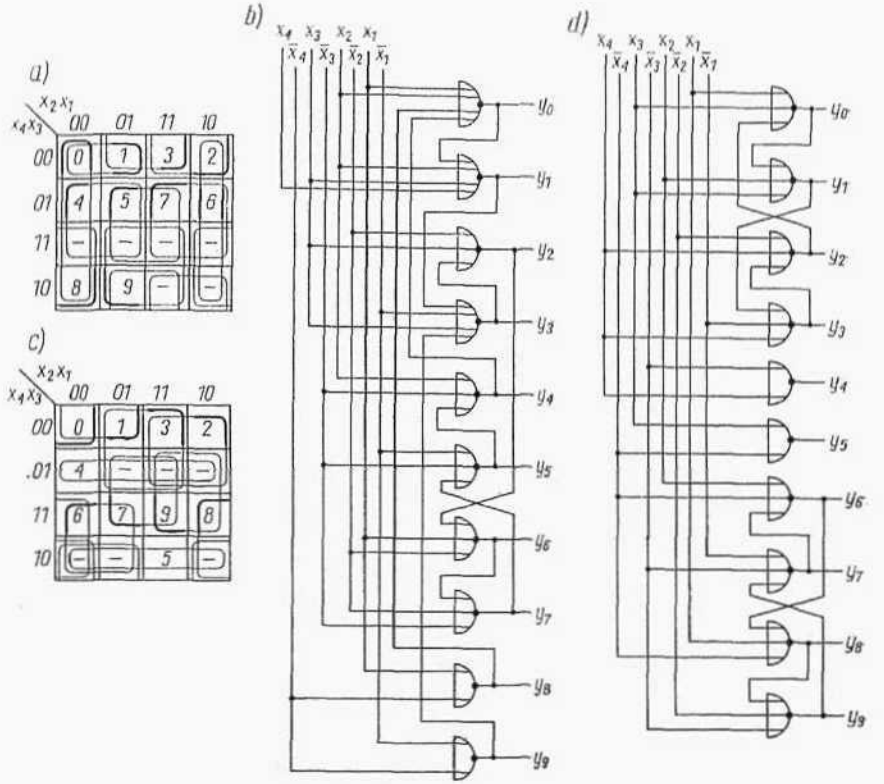
Dekodery, wykorzystujące nieokreślone kombinacje x dla minimalizacji funkcji y są nazywane *uproszczonymi*. W porównaniu z dekoderami

pełnymi zawierają prostsze elementy (z mniejszą liczbą wejść) i mniej obciążają sygnały x, ale pojawienie się nieprawidłowej kombinacji powoduje powstanie błędnych sygnałów wyjściowych. Dwa dalsze przykłady dekoderów uproszczonych są przedstawione na rys. 5-7 (dla kodów IV i VI). Szczególnie prosty jest dekoder kodu pseudopierścieniowego, gdyż budowa tego kodu umożliwia określenie całej kombinacji na podstawie wartości dwóch sąsiednich pozycji x.

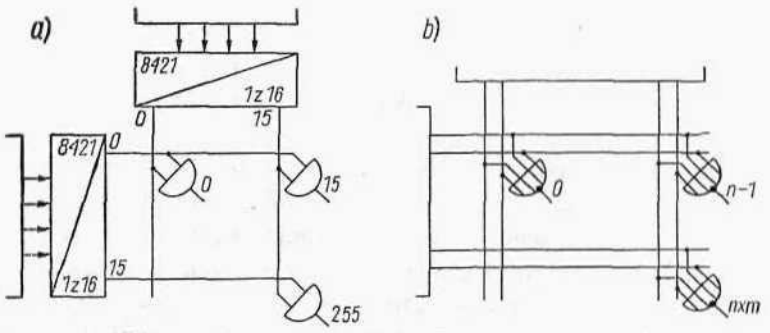
W dekoderach tylko jeden sygnał wyjściowy może mieć wartość 1, więc spełniona jest zależność $F^1(y_i) \subseteq F^0(y_j)$, dla $i \neq j$. Umożliwia to syntezę układów z elementów NOR i NAND, z wykorzystaniem działania zakazu (p. 3.3.4), czyli wymuszania sygnałem $y_i = 1$ — wartości $y_j = 0$, albo przeciwnie. Celem takiego postępowania może być wyrugowanie sygnałów wejściowych zanegowanych (rys. 3-40) albo zmniejszenie obciążeń sygnałów wejściowych. Dekodery wykorzystujące sygnały y_i do realizacji y_i, są nazywane *optymalnymi*, przy czym kryteria optymalności mogą być różne. Przykład optymalnego dekodera dwójkowo-dziesiętnego kodu 8421 pokazano na rys. 5-8b, a opisującą ten układ tablicę na rys. 5-8a. Z tablicy wynika, że postać funkcji y_8 i y_9 może być prosta, jeśli wykorzysta się pozycje nieokreślone. Funkcję y_0 można uzyskać z grupy (—00) przez wyrugowanie kratek y_4 i y_8, a więc sygnałami wejściowymi elementu NOR z wyjściem y_0 będą: x_1, x_2, y_4, y_8. Funkcję y_1 uzyskuje się z grupy (000–), odejmując od niej kratkę y_0, funkcję y_3 z grupy (–0–1), odejmując y_1 i y_9, itd. — zgodnie z rys. 5-8a. W podobny sposób, na podstawie tablicy z rys. 5-8c, utworzono schemat dekodera kodu Aikena (IV) — rys. 5-8d. W obydwu tych przykładach występują sprzężenia zwrotne tworzące pętle, w których pracuje kilka elementów. Jeśli liczba elementów w pętli jest parzysta, to równanie opisujące funkcję y_i można sprowadzić do postaci

$$y_i = A(B + Cy_i)$$

w której A, B, C są funkcjami x. Jest to typowe równanie elementu pamięci, a ponieważ w dekoderze podtrzymywanie stanu oznacza sprzeczność między X a Y, więc funkcję pamięci trzeba wykluczyć, przyjmując $AC = 0$. Spełnienie tego warunku można łatwo sprawdzić na schemacie układu, gdyż sprowadza się on do wymagania, aby w zamkniętej pętli elementy i oraz $i+2$ miały sygnały wejściowe x_j oraz $\bar{x}_j$. Na przykład w schemacie z rys. 5-8b występuje pętla z elementów (przy założeniu, że



Rys. 5-8. Tablice i schematy optymalnych dekoderów dwójkowo-dziesiętnych: a,b) kodu „8421"; c,d) kodu Aikena



Rys. 5-9. Struktury dekoderów: a) wielostopniowego; b) współrzędnościowego

sygnał y_i pochodzi z elementu i): 01326754_0, a więc należy sprawdzić sygnały wejściowe par (0,3), (1,2), (3,6) itd.

Jeśli liczba elementów w pętli jest nieparzysta, to równanie opisujące każdą z funkcji można sprowadzić do postaci

$$y_i = A(B + C\bar{y}_i)$$

Dla wyrugowania możliwości powstawania drgań należy dążyć do spełnienia warunku $AC = 0$. Sprowadza się on do wymagania, aby w zamkniętej pętli elementy i oraz $i+1$ miały sygnały wejściowe x_j i $\bar{x}_j$.

Stosowanie dekoderów optymalnych jest bardziej uzasadnione w przypadkach, gdy obciążalność elementów generujących sygnały x jest niewielka; w układach scalonych raczej nie są stosowane.

Dekodery z dużą liczbą wejść są niekiedy budowane w postaci matryc diodowych, stanowiących zestaw iloczynów diodowych (rys. 2-7b). W takich przypadkach jest celowe stosowanie układów wielostopniowych. Na przykład pełny dekoder 8-bitowy kodu naturalnego zawiera 256 8-wejściowych elementów I, czyli $256 \cdot 8 = 2048$ diod. Gdyby 8 wejść podzielić na dwie grupy i zbudować dwa dekodery 4-wejściowe, to każdy z nich zawierałby $16 \cdot 4 = 64$ diody, a sygnały wyjściowe można by było uzyskać z 2-wejściowych elementów I kosztem $256 \cdot 2 = 512$ diod, co łącznie stanowi $64+64+512 = 640$ diod. Struktura tego układu jest pokazana na rys. 5-9a. Zysk w porównaniu z dekoderem jednostopniowym jest więc bardzo duży. Gdyby jeszcze dekodery 4-wejściowe zbudować w postaci układu dwustopniowego (z dwóch dekoderów 2-bitowych i zespołu iloczynów) to liczbę diod w takim dekoderze można by ograniczyć do 48, a w całym układzie do 608. Zysk z wprowadzenia trzeciego stopnia nie jest więc duży, zwłaszcza jeśli się uwzględni tłumienie sygnału wnoszone przez każdy stopień układu. Podobne proste obliczenia dla innych kodów i różnej liczby wejść umożliwiają wybór struktury najprostszej.

Budowanie dekoderów wielostopniowych z elementów NOR lub NAND nie jest opłacalne, gdyż wymaga realizowania negacji sygnałów między stopniami. Z tego powodu występuje niekiedy konieczność stosowania układów jednostopniowych nawet w tych przypadkach, gdy sygnały wejściowe tworzą odrębne grupy, ułatwiające dekodowanie częściowe. Na przykład przy równoczesnym dekodowaniu stanu dwóch

liczników, układ jednostopniowy wymaga wprowadzenia na wejście elementów I sygnałów jednego i drugiego licznika (rys. 5-9b). Tworzy się w ten sposób charakterystyczny *dekoder współrzędnościowy*. Liczba sygnałów w grupach wejściowych ograniczona jest liczbą wejść elementów wyjściowych, ale i tu można stosować dekodowanie uproszczone. Szczególnie proste układy uzyskuje się, gdy sygnały wejściowe tworzą kody pseudopierścieniowe; element I o czterech wejściach może wówczas dekodować stan np. dwóch dowolnie długich liczników. Kody w różnych grupach sygnałów wejściowych mogą też być różne, a liczba współrzędnych układu nie musi być ograniczona do dwóch.

5.2.3. TRANSLATORY KODÓW I UKŁADY UZUPEŁNIAJĄCE

Te konwertery kodów, które nie są koderami i dekoderami, bywają nazywane *translatorami kodów* i w zasadzie mogą być budowane przez połączenie wyjść odpowiedniego dekodera z wejściami kodera. Taka realizacja w większości przypadków nie jest jednak optymalna. Najlepsze wyniki daje synteza tych układów jako typowych układów wielowyjściowych, metodami wprowadzonymi wyżej. Na przykład konwerter kodu Gray'a na naturalny kod dwójkowy, w przypadku trzech zmiennych można opisać tablicą z rys. 5-10, z której otrzymuje się

$$
\begin{aligned}
y_3 &= x_3 \\
y_2 &= x_2\bar{x}_3 + \bar{x}_2 x_3 = x_2 \oplus x_3 = x_2 \oplus y_3 \\
y_1 &= x_1\bar{x}_2\bar{x}_3 + \bar{x}_1 x_2\bar{x}_3 + \bar{x}_1\bar{x}_2 x_3 + x_1 x_2 x_3 = \\
&= x_1(\bar{x}_2\bar{x}_3 + x_2 x_3) + \bar{x}_1(\bar{x}_2 x_3 + x_3\bar{x}_3) = x_1 \oplus y_2
\end{aligned}
$$

Zaobserwowana w tych zależnościach prawidłowość umożliwia budowanie konwerterów o większej liczbie wejść, z zachowaniem iteracyjnej struktury układu z rys. 5-10b, tzn. ze związku

$$y_i = x_i \oplus y_{i+1}$$

W podobny sposób dla konwertera naturalnego kodu dwójkowego na kod Gray'a otrzymuje się z tablicy (rys. 5-10c)

$$
\begin{aligned}
y_3 &= x_3 \\
y_2 &= x_2\bar{x}_3 + \bar{x}_2 x_3 = x_2 \oplus x_3 \\
y_1 &= x_1\bar{x}_2 + \bar{x}_1 x_2 = x_1 \oplus x_2
\end{aligned}
$$