WOJCIECH GŁOCKI

UKŁADY CYFROWE

WYDAWNICTWA SZKOLNE I PEDAGOGICZNE

Wydawnictwa Szkolne i Pedagogiczne

polecają następujące podręczniki:

Józef Parchański

Miernictwo elektryczne i elektroniczne

Barbara Pióro, Marek Pióro

Podstawy elektroniki cz.1

Augustyn Chwaleba, Bogdan Moeschke, Grzegorz Płoszajski

Elektronika

ISBN 83-02-06242-1

9 788302 062421 02

WOJCIECH GŁOCKI

UKŁADY CYFROWE

Podręcznik dla technikum

Wydanie drugie

Warszawa 1998
Wydawnictwa Szkolne i Pedagogiczne

Recenzenci:
prof. dr hab. Wiesław Traczyk — pracownik Instytutu Automatyki i Informatyki Stosowanej PW
mgr inż. Ewa Kowalska — nauczyciel przedmiotu w Zespole Szkół Elektronicznych w Warszawie, współtwórczyni programów nauczania
mgr inż. Krzysztof Wojtuszkiewicz — nauczyciel Technikum Elektronicznego oraz pracownik Elektronicznych Zakładów Naukowych we Wrocławiu

Okładkę projektował: Krzysztof Demianiuk
Redaktor: Anna Moniuszko
Redaktor techniczny: Maria Dylewska

Książka dopuszczona do użytku szkolnego przez Ministra Edukacji Narodowej i wpisana do zestawu podręczników do nauczania przedmiotu *układy cyfrowe* w zawodzie *technik elektronik* na poziomie technikum i szkoły policealnej.
Numer w zestawie 9/96

Książka opracowana na podstawie programu 2105/MPiH/1993.02.15

Książka zawiera podstawowe pojęcia techniki cyfrowej i automatyki, których znajomość jest niezbędna do opisu działania układów cyfrowych. Scharakteryzowano w niej technologie układów cyfrowych (TTL oraz CMOS) oraz opisano podstawowe ich elementy, czyli bramki i przerzutniki. W książce omówiono wybrane zagadnienia projektowania układów cyfrowych. Rozpatrzono działanie bloków funkcjonalnych, takich jak multipleksery, demultipleksery, układy arytmetyczne, liczniki, pamięci i mikroprocesor. Liczne przykłady i zadania o dużym stopniu trudności ułatwiają zrozumienie i utrwalenie wiedzy teoretycznej.

Książka jest przeznaczona dla uczniów średnich szkół zawodowych oraz liceów technicznych o profilu elektronicznym specjalności elektronika ogólna, systemy komputerowe, elektryczna i elektroniczna automatyka przemysłowa.

ISBN 83-02-06242-1

Wydawnictwa Szkolne i Pedagogiczne, Warszawa 1998
Wydanie drugie. Ark. druk. 20,75+0,5 wkładka
Skład i łamanie: Oficyna Owczarnia
Druk i oprawa: Cieszyńska Drukarnia Wydawnicza
Zam. nr 1293/K-98

Spis treści

Przedmowa

Podręcznik jest przeznaczony dla uczniów średnich szkół zawodowych o profilu elektronicznym. Głównym adresatem są uczniowie kształcący się w specjalności „Elektronika ogólna" (program 210502). Dzięki pewnemu rozszerzeniu treści zawartych w podręczniku może on być także przydatny w specjalności „Elektryczna i elektroniczna automatyka przemysłowa" (program 210505). Treść podręcznika obejmuje także program nauczania w specjalności „Systemy komputerowe" (program 210503).

Zawartość podręcznika jest dostosowana do wdrażanych aktualnie, nowo opracowanych, programów nauczania przedmiotów zawodowych.

W strukturze podręcznika przyjęto zasadę stopniowego rozszerzania zasobu wiadomości w taki sposób, aby możliwe było rozwiązywanie przez uczącego się coraz bardziej złożonych problemów technicznych. Z punktu widzenia tematyki omawianych treści nauczania może to stanowić pewne przemieszanie i przeplatanie się tematów. Jednak taki podział treści nauczania umożliwia stopniowe rozbudowywanie możliwości pracy własnej ucznia. Wzrost zaangażowania się ucznia w aktywne przyswajanie nowych wiadomości można osiągnąć poprzez sformułowanie pewnych problemów technicznych do samodzielnego rozwiązania. Każdy podrozdział, w związku z tym, kończy się zbiorem pytań i zadań. Zakres tych zadań jest tak dobrany, aby na podstawie treści zawartych w podręczniku możliwe było ich rozwiązanie.

Niektóre treści podręcznika wykraczają poza zakres określony programem nauczania, niektóre zostały omówione sformalizowanym językiem (językiem matematyki) i jednocześnie językiem potocznym, pozwalając w ten sposób dostosować nauczycielowi formę oraz zakres nauczanych treści do możliwości i aktywności uczniów.

1

Wstęp

1.1. Wprowadzenie

W ostatnich latach coraz częściej urządzenia elektroniczne są realizowane przy użyciu techniki cyfrowej zamiast używanej wcześniej techniki analogowej. Samo wprowadzenie techniki cyfrowej stworzyło ponadto wiele nowych obszarów jej zastosowań i rozszerzyło możliwości oraz horyzonty elektroniki. Stało się tak dzięki takim zaletom techniki cyfrowej, jak: niezawodność, prostota i wygoda obsługi, odporność na zakłócenia, a także niewielki koszt, malejący wraz z rozwojem technologii wytwarzania scalonych układów cyfrowych.

W pojęciu **technika cyfrowa** przymiotnik **cyfrowa** bierze się stąd, że informacja wewnątrz urządzeń cyfrowych jest zakodowana za pomocą liczb. A liczba, jak wiemy, to pewien uporządkowany ciąg cyfr. Jest to naturalne w takich urządzeniach, jak na przykład kalkulator, kasa sklepowa, komputer wykorzystywany do obliczeń inżynierskich. Wielkości przetwarzane w tych urządzeniach mają z natury rzeczy postać cyfrową także w potocznym tego słowa znaczeniu.

Jak wiadomo komputer może także przetwarzać (tzw. edytory tekstowe) dane w postaci tekstów, czyli ciągów znaków graficznych zawartych w zbiorze zwanym alfabetem. Urządzenia cyfrowe mogą przetwarzać wyłącznie informację w postaci cyfrowej. Czy zatem alfabet jest wielkością cyfrową?

Aby odpowiedzieć na postawione powyżej pytanie zapoznajmy się z definicją tego pojęcia.

Wielkością cyfrową ***będziemy nazywać taką wielkość, która w danym przedziale swej zmienności przyjmuje skończoną liczbę wartości.***

Innymi słowy, zbiór wartości wielkości cyfrowej jest zbiorem przeliczalnym. W kontekście tej definicji alfabet jest wielkością cyfrową. Każdemu elementowi (znakowi graficznemu) alfabetu (liczba elementów jest skończona) możemy zatem przypisać jakąś liczbę (nazywaną kodem) i przetwarzać w urządzeniu cyfrowym. Gdyby zbiór wartości (elementów) był nieskończenie duży (nieprzeliczalny), wówczas powyższy zabieg nie byłby możliwy.

Cechą odróżniającą wielkość analogową od cyfrowej jest to, że jej zbiór wartości jest zbiorem nieprzeliczalnym.

Wielkością analogową ***będziemy nazywać taką wielkość, która w danym przedziale swej zmienności przyjmuje nieskończoną liczbę wartości.***

Wiadomo jednak, że układy cyfrowe przetwarzają także informację, która w postaci źródłowej jest wielkością analogową. Przykładem takim może być: czas, dźwięk, napięcie elektryczne czy inne wielkości fizyczne o charakterze analogowym. Zastanówmy się, jakie są różnice w przedstawianiu informacji wyjściowej przy pomiarze tych wielkości przyrządami analogowymi i cyfrowymi. Zegar analogowy (wskazówkowy) może wskazać w dowolnym przedziale czasu (1 godziny, minuty czy sekundy) nieskończenie wiele wartości, gdyż każda ze wskazówek może przyjąć nieskończenie wiele położeń. Wielkość analogową, jaką jest czas, możemy w dowolnym przedziale (godziny, minuty, sekundy itp.) podzielić na skończoną liczbę odcinków i w ten sposób nadać jej charakter wielkości cyfrowej. Zegar cyfrowy w danym przedziale czasu będzie mógł wyświetlić jedynie skończoną liczbę wskazań. Liczba tych wskazań zależy od liczby cyfr wyświetlacza. Jeśli to będzie 4-cyfrowy wyświetlacz (2 cyfry godzin, 2 cyfry minut), to w przedziale jednej godziny liczba wskazań wyniesie 60. Czas będzie pokazywany z dokładnością do 1 minuty.

Napięcie elektryczne (wielkość analogowa) może być mierzone woltomierzem analogowym bądź cyfrowym. Na przykład w zakresie od 0 do 10 V liczba różnych wskazań przyrządu analogowego będzie teoretycznie nieskończenie duża. W tym samym zakresie przyrząd cyfrowy o 3-polowym wskaźniku odczytowym wskaże jedynie 1000 różnych wartości (od 000 do 999).

Wydawałoby się więc, że wielkość analogowa po przetworzeniu na postać cyfrową zostaje obarczona pewną niedokładnością. A więc przejście na postać cyfrową wymaga rezygnacji z absolutnej dokładności odwzorowania i pogodzenia się z pewnym błędem tej transformacji, natomiast postać analogowa nie ma takich ograniczeń. Otóż jest to błędny wniosek.

Wzmiankowany powyżej woltomierz analogowy, którego wskazówka teoretycznie może przyjąć nieskończenie wiele położeń odwzorowujących napięcie mierzone, nie wskaże dwóch różnych napięć różniących się mniej niż wynosi czułość tego woltomierza (istnieje pewna progowa wartość zmiany napięcia mierzonego, która może wywołać zauważalny ruch wskazówki). Człowiek nie jest w stanie rozróżnić dwóch położeń wskazówki różniących się mniej niż 0,2 działki.

Wymieniony powyżej woltomierz cyfrowy może wskazywać z dokładnością do 10/1000 V (10 mV). Ale wystarczy użyć woltomierza o 4- czy 5-polowym wyświetlaczu, aby dokładność wyniosła odpowiednio 1 mV czy 0,1 mV. Zakres zmienności wielkości analogowej możemy podzielić na dostatecznie dużą liczbę przedziałów, aby uzyskać wymaganą dokładność.

Praktyczna wyższość reprezentacji cyfrowej nad analogową przejawia się w procesie przesyłania lub zapisywania informacji. Szczególnie proces transmisji

(przesyłania) jest narażony na czynniki zakłócające. Weźmy dla przykładu sygnał analogowy w postaci napięcia o zmienności z zakresu od 0 do 10 V (np. odwzorowujący wartość mierzonej temperatury). Sygnał ten z miejsca pomiaru jest przesyłany do odległego centrum kontroli i sterowania. Przesyłany sygnał podlega różnego rodzaju zniekształceniom, wywołanym przez niezerową rezystancję przewodów, obecność zewnętrznych pól elektromagnetycznych i niestałość parametrów urządzeń realizujących transmisję. Niewielka zmiana wartości napięcia — np. o 100 mV, wprowadzi do wyniku pomiaru błąd równy 1% zakresu.

Zupełnie inaczej wygląda sytuacja w systemie cyfrowym. Zmierzona wartość temperatury natychmiast jest zamieniana na postać cyfrową. Może to także być sygnał napięciowy, ale przekazywaniu na dużą odległość podlegają tu liczby dwójkowe. Cyfra **0** jest reprezentowana przez napięcie bliskie 0 V, a cyfra **1**[1)] przez napięcie bliskie 5 V (przykładowo). Teraz nawet kilkakrotnie większe niż w poprzednim przykładzie zniekształcenie sygnału nie spowoduje pomylenia **0** z **1** i zmierzona wartość zostanie przekazana w sposób dokładny, to znaczy tylko z błędem wynikającym z użycia ograniczonej liczby cyfr. Jeżeli dla przedstawienia wyników pomiarów użyjemy 10 cyfr dwójkowych (tzw. bitów — patrz p. 1.2), to błąd tego przedstawienia wyniesie mniej niż 0,1% zakresu; przy użyciu 16 bitów — mniej niż 0,002%.

Cyfrowe metody przetwarzania i przekazywania danych są coraz częściej stosowane również w takich urządzeniach i systemach, które dla sygnałów wejściowych i wyjściowych są i muszą pozostać urządzeniami analogowymi (np. magnetofon cyfrowy). W celu zwiększenia dokładności warto zamienić tu analogowy sygnał wejściowy na postać cyfrową, wykonać niezbędne przekształcenia danych (zapis) w sposób cyfrowy i z powrotem zamienić wyniki na postać analogową. Odporność sygnałów cyfrowych na wpływ czynników zakłócających zostanie omówiona w dalszej części podręcznika (p. 1.3; 4.2).

W przyrządach cyfrowych, w których postać źródłowa przetwarzanej informacji ma charakter analogowy, musi nastąpić proces jej zamiany na postać cyfrową.

***Proces zamiany wielkości analogowej na cyfrową będziemy nazywać* kwantowaniem.**

Kwantowanie jest niczym innym, jak podzieleniem pewnego ciągłego obszaru zmienności jakiejś wielkości na skończoną liczbę przedziałów. Każdemu takiemu przedziałowi możemy już przypisać pewną określoną liczbę. Im większa liczba przedziałów, tym uzyskana postać cyfrowa dokładniej odzwierciedla wielkość analogową.

Podany powyżej przykładowo zegar możemy wyposażyć w dodatkowe dwie cyfry i wtedy czas będzie wskazywany z dokładnością do 1 sekundy (pociąga to za sobą pewne zmiany w budowie wewnętrznej, ale zasada działania się nie zmienia). Można sobie także wyobrazić ograniczenie pola odczytowego naszego zegara do

[1)] W treści podręcznika przyjęto zasadę wyróżniania pismem półgrubym wartości i symbole o charakterze dwustanowym, czyli cyfrowe. Natomiast w tablicach i na rysunkach, gdzie zapis na ogół nie budzi wątpliwości, z wyróżnień takich zrezygnowano.

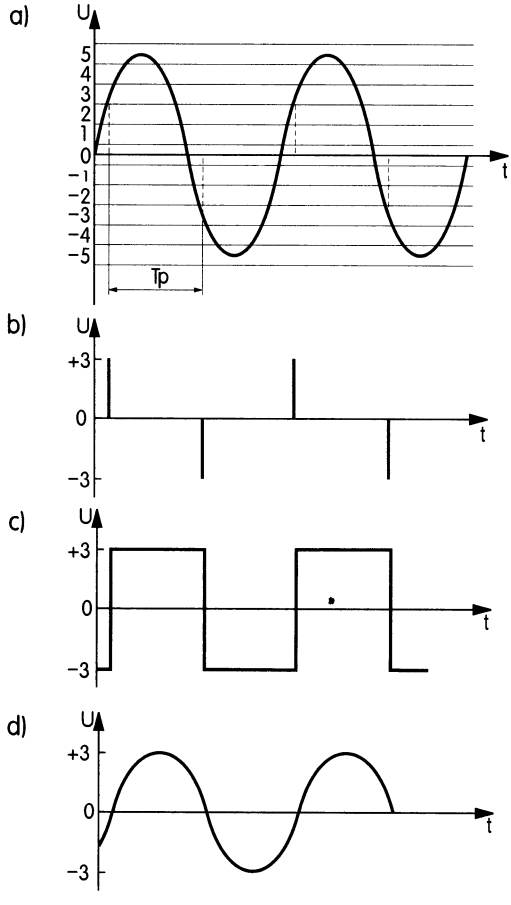
dwóch cyfr. Taki zegar wskazywałby czas z dokładnością do 1 godziny, co obecnie dyskwalifikowałoby go z użycia, ale jeszcze kilkaset lat temu byłby on bardzo użytecznym urządzeniem.

Wiele urządzeń cyfrowych przetwarza informacje analogowe zmienne w czasie, np. falę akustyczną, czyli dźwięk zapisywany na płycie kompaktowej czy w magnetofonie cyfrowym. Proces zamiany wielkości analogowej na cyfrową (czyli kwantowanie) musi wówczas zostać poprzedzony tak zwanym **próbkowaniem**.

Próbkowanie ***polega na obserwacji (pomiarze) zmiennej w czasie wielkości analogowej w pewnych (przeważnie regularnych) odstępach czasu.*** Obserwowana wartość tej wielkości jest następnie kwantowana.

Całość procesu zamiany wielkości analogowej na cyfrową (próbkowanie + kwantowanie) określa się mianem **dyskretyzacji (cyfryzacji).**

Intuicyjnie jest wyczuwalne, że jeżeli wielkość analogowa zmieniać się będzie powoli, to obserwacje powyższe (próbkowanie) możemy przeprowadzać niezbyt często (z niewielką **częstotliwością próbkowania**). Przyjęcie dużej częstotliwości



Rys. 1.1. Próbkowanie i kwantowanie przebiegu sinusoidalnego

próbkowania nie zawsze jest krokiem optymalnym. Przy dużej częstotliwości próbkowania ilość informacji, jaką trzeba będzie przetwarzać, radykalnie się zwiększa. Z drugiej strony częstotliwość ta nie może być zbyt mała, aby możliwe było wierne odtworzenie informacji wejściowej.

Załóżmy, że przebieg wielkości wejściowej ma kształt sinusoidalny. Przyjmijmy ponadto, że próbkujemy go (tzn. mierzymy jego wartość) z częstotliwością 2 razy większą niż jego własna. W wyniku takiego próbkowania otrzymamy ciąg wartości jak na rys. 1.1.

W wyniku podzielenia zakresu zmienności amplitudy napięcia na 11 przedziałów, każdej zmierzonej wartości została przyporządkowana odpowiednia liczba. Efekt dyskretyzacji (próbkowania i kwantowania) przedstawia rys. 1.1b.

Odzyskanie informacji wejściowej będzie polegać na odtworzeniu przebiegu poddanego przetwarzaniu. Najprostszy zabieg prowadzący do tego celu to przyjęcie, że pomiędzy kolejnymi próbkami wartość próbkowanego sygnału nie zmieniała się. Daje to w efekcie falę prostokątną jak na rys. 1.1c. Przepuszczenie jej przez filtr dolnoprzepustowy pozwala uzyskać przebieg jak na rys. 1.1d. Jeżeli nośnikiem informacji wejściowej był sinusoidalny kształt przebiegu, to jak widać z rys. 1.1, odzyskanie tej informacji jest możliwe. W przypadku na przykład dźwięku kształt przebiegu zawiera pełną o nim informację, gdyż natężenie dźwięku i tak jest dostosowywane do aktualnych potrzeb słuchacza poprzez dobranie odpowiedniego wzmocnienia w urządzeniu odtwarzającym.

Powyższa analiza pozwala sformułować wniosek, że ***dwukrotnie większa częstotliwość próbkowania niż częstotliwość przebiegu próbkowanego*** (sinusoidalnie zmiennego) ***jest wystarczająca do odtworzenia pełnej informacji zawartej w sygnale analogowym*** (ściślej w kształcie tego sygnału, bowiem nie pozwala na odtworzenie amplitudy i fazy). Dociekliwy Czytelnik zauważy pewnie, że przy dokładnie takiej częstotliwości próbkowania „możemy mieć pecha i trafić" w chwile przejścia sinusoidy przez zero. W wyniku takiego próbkowania nic nie zaobserwujemy, pomimo że informacja wejściowa będzie określona. W praktyce należy więc — dla zapewnienia możliwości odtworzenia informacji wejściowej — próbkować z częstotliwością większą niż dwukrotna częstotliwość przebiegu sinusoidalnego.

Przebiegi próbkowanych wielkości analogowych nie zawsze mają kształt sinusoidy, jednak dowolny przebieg w skończonym przedziale czasu można rozłożyć na sumę przebiegów sinusoidalnych (tzw. **harmonicznych**). Składowe harmoniczne różnią się między sobą amplitudą, fazą i częstotliwością. Im większa częstotliwość składowej sinusoidalnej, tym mniejsza jest jej amplituda. W przypadku ogólnym liczba harmonicznych jest nieskończenie duża, ale powyżej pewnej częstotliwości granicznej f_{gr} amplituda ich jest pomijalnie mała. Wówczas możemy przyjąć, że maksymalną częstotliwością, jaka występuje w widmie przebiegu wejściowego, jest ta częstotliwość graniczna. Korzystając z pojęcia częstotliwości granicznej warunek na częstotliwość próbkowania f_p można sformułować następująco:

$$f_p > 2f_{gr} \tag{1.1}$$

W praktyce znalezienie częstotliwości granicznej (a tym samym wyznaczenie częstotliwości próbkowania) jest często znacznie prostsze i nie wymaga analizy widma przebiegu poddawanego dyskretyzacji. W przypadku na przykład dźwięku częstotliwość graniczną determinują możliwości percepcyjne ucha ludzkiego. Teoretycznie pasmo częstotliwości odbieranych przez ucho ludzkie zawiera się w przedziale od 16 Hz do 20 kHz. W praktyce często przyjęcie maksymalnej częstotliwości równej 16 kHz jest całkowicie wystarczające.

W sprzęcie audio najwyższej klasy częstotliwość próbkowania musi przekraczać częstotliwość 2·20 kHz (np. f_p= 44 kHz).

W teletransmisji na przykład, gdzie nie zależy nam na absolutnej wierności (pełnej barwie) przesyłanego dźwięku, częstotliwość próbkowania może być znacznie mniejsza — co zdecydowanie zmniejsza ilość przesyłanej informacji i pozwala obniżyć koszty takiej transmisji.

Dla lepszej orientacji w ilości informacji powstałej w wyniku dyskretyzacji sygnału analogowego przeliczmy, jaką porcją informacji będzie 1 (jedna) sekunda muzyki. Przyjmijmy, że amplituda sygnału jest skwantowana na 2^{16} przedziałów. Odpowiada to użyciu szesnastobitowego przetwornika AC (analogowo-cyfrowego). Każda próbka to 16 bitów (2 bajty) informacji. Jeżeli próbkowanie wykonujemy z częstotliwością f_p= 44 kHz, to na 1 sekundę muzyki złoży się 44 tysiące takich próbek — co daje w sumie 2·44 000, czyli ok. 86 KB (kilobajtów; 1 KB = 1024 bajty). Mnożąc uzyskany rezultat przez 3600 (tyle sekund ma 1 godzina), liczba bitów wyrazi się w miliardach. Dla lepszego wyobrażenia sobie, jaką to stanowi porcję informacji, należy pamiętać, że tekst zapisany na 1 stronie arkusza formatu A4 to około 2 KB informacji.

Ilość przetwarzanej informacji jest bardzo istotna i stwarza określone problemy techniczne. Często jest ona jedynym ograniczeniem możliwości zastosowania techniki cyfrowej. Obecnie coraz wyższe stopnie scalenia układów elektronicznych oraz coraz większe dopuszczalne częstotliwości przetwarzanych przez nie sygnałów elektrycznych sprawiają, że dziedziny zastosowań techniki cyfrowej ulegają stałemu poszerzaniu.

1.2. Systemy pozycyjne

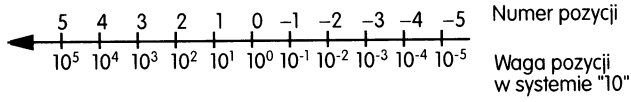
Znany i powszechnie używany **system dziesiętny** liczenia jest tak zwanym **systemem pozycyjnym**. Spójrzmy na zapis liczby dziesiętnej pod kątem związków z podstawą systemu dziesiętnego — liczbą 10. Zauważmy, że system dziesiętny operuje dziesięcioma znakami graficznymi (od 0 do 9) zwanymi cyframi. Każda cyfra reprezentuje pewną wartość w zapisie liczby dziesiętnej. Ale, czy wartość jej zależy wyłącznie od niej samej? Popatrzmy na cyfrę 7 w trzech następujących liczbach:

a) 137,25 b) **7**13,48 c) 342,**7**2

W przykładzie a) wartość wskazywana przez cyfrę siedem jest równa siedmiu jednostkom, w przykładzie b) siedmiuset jednostkom i w przykładzie c) siedmiu dziesiątym jednostki. Spostrzeżenia powyższe są tak oczywiste, że zapewne wzbudziły uśmiech Czytelnika. Ale, co zatem decyduje o wartości cyfry występującej w zapisie dziesiętnym? Jak zapewne wszystkim wiadomo, jest to jej pozycja — stąd nazwa: **system pozycyjny.** Każdej pozycji jest przypisana jej **waga** (czyli znaczenie). W systemie dziesiętnym mówimy o pozycji jednostek, dziesiątek, setek, tysięcy itd. (analizując liczbę na lewo od przecinka) lub dziesiętnych części, setnych itd. (analizując liczbę na prawo od przecinka). Jaki jest związek między podstawą systemu (10) i wagą danej pozycji? Jeżeli poszczególnym pozycjom przyporządkujemy liczby zgodnie z odwróconą osią liczbową (rys. 1.2), to związek ten wyrazi się zależnością

$$w = p^k \tag{1.2}$$

gdzie: w — waga pozycji k; p — podstawa systemu; k — pozycja.



Rys. 1.2. Numery pozycji cyfr w systemach pozycyjnych

Zapis liczby w systemie pozycyjnym jest więc umownym zapisem współczynników (cyfr) przy odpowiednich potęgach podstawy systemu. Dla systemu dziesiętnego postać liczby będzie więc następująca:

$$L_{10} = a_n a_{n-1} \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-k} = \sum_{i=-k}^{i=n} a_i 10^i \tag{1.3}$$

przy czym współczynniki (cyfry)

$$a_i \in \{0,1,\ldots,9\}$$

Przyjęcie za podstawę systemu liczby 10 jest kwestią czysto umowną. Równie dobrze może to być każda inna liczba p. Wszystkie powyższe spostrzeżenia poczynione dla systemu dziesiętnego odpowiednio przenoszą się na dowolny system pozycyjny o podstawie p. Podstawa p określa w sposób kompletny system. I tak w systemie pozycyjnym o podstawie p:

- liczba znaków graficznych (cyfr) potrzebnych do zapisu dowolnej liczby wynosi p,
- wagi poszczególnych pozycji mają wartość p^k, gdzie k jest pozycją cyfry liczoną zgodnie z osią liczbową, jak na rys. 1.2.

Tak więc dowolna liczba zapisana w dowolnym systemie pozycyjnym będzie miała postać

$$L_p = \mathbf{a}_n\mathbf{a}_{n-1}\ldots\mathbf{a}_1\mathbf{a}_0, \mathbf{a}_{-1}\mathbf{a}_{-2}\ldots\mathbf{a}_{-k} = \sum_{i=-k}^{i=n} \mathbf{a}_i p^i \qquad (1.4)$$

przy czym współczynniki a_i należą do p-elementowego zbioru znaków graficznych (cyfr).

To stałe zaznaczanie, że współczynniki są znakami graficznymi (cyframi), ma na celu odejście od stereotypu istnienia tylko dziesięciu cyfr. Dla potrzeb systemu dziesiętnego i każdego innego o podstawie $p < 10$ ten zbiór znaków jest wystarczający. Chcąc zbudować np. **system szesnastkowy** (**heksadecymalny**), musimy zdefiniować 16 symboli — cyfr. Zwykle wykorzystuje się 10 cyfr systemu dziesiętnego, a dla reprezentacji **cyfr** 10, 11, 12, 13, 14 i 15 przyjęto używać kolejne litery alfabetu A, B, C, D, E i F.

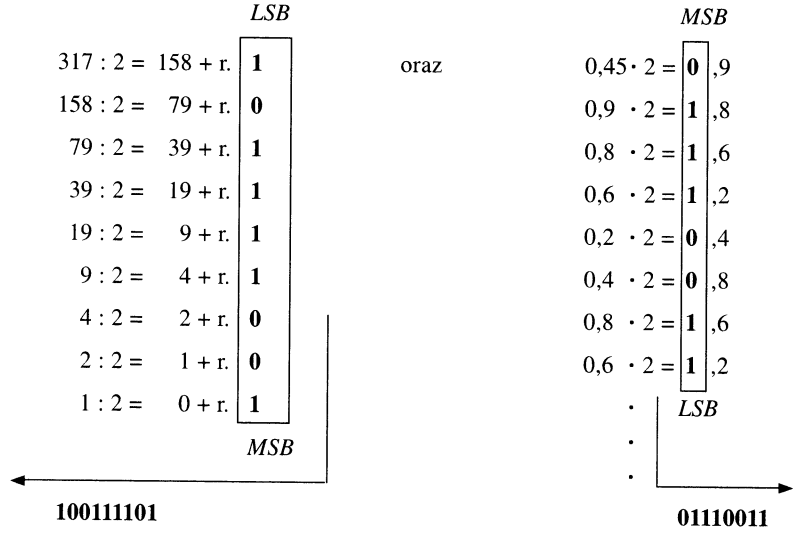
Istnieje wiele sposobów konwersji liczb dziesiętnych na liczby zapisane w systemie pozycyjnym o podstawie p. Jeden z nich zostanie przedstawiony poniżej na przykładzie **systemu dwójkowego (binarnego),** czyli systemu o podstawie $p = 2$. Warto jeszcze zaznaczyć, że zastępując w poniższym algorytmie liczbę 2 liczbą p uzyskamy zapis w systemie o podstawie p.

Przykład 1.1

Dokonać konwersji dziesiętno-dwójkowej liczby 317,45.

Rozwiązanie

Zamiany dokonujemy oddzielnie dla części całkowitej i oddzielnie dla części ułamkowej. Tak więc:



Wynik: $317{,}45_{10} = \mathbf{100111101{,}01110011}_2$ ■

Zamiana części całkowitej polega na dzieleniu jej przez podstawę systemu (dla systemu dwójkowego podstawa $p = 2$). Wynik dzielenia zapisujemy w posta-

ci: część całkowita + reszta z dzielenia. W kolejnym kroku z częścią całkowitą ilorazu postępujemy identycznie jak w kroku poprzednim.

Zauważmy, że reszta z dzielenia będzie zawsze mniejsza niż liczba p. W niniejszym przykładzie reszty te mogą przyjmować tylko dwie wartości: **0** lub **1**. Ciąg tych reszt jest zapisem dwójkowym liczby dziesiętnej poddawanej konwersji. Należy jedynie zapamiętać, że pierwsza reszta jest cyfrą z najmłodszej pozycji (pozycji o najmniejszej wadze dla części całkowitej). Jest to tak zwany **najmniej znaczący bit — LSB** (ang. *Least Significant Bit*).

Zamiana części ułamkowej polega na mnożeniu jej przez podstawę systemu p. Wynik mnożenia zawiera zawsze część całkowitą mniejszą od p oraz jakąś część ułamkową. Z częścią ułamkową postępujemy dalej tak, jak w kroku poprzednim. Uzyskany ciąg części całkowitych jest zapisem dwójkowym ułamkowej części liczby dziesiętnej poddawanej konwersji. Należy jedynie zapamiętać, że część całkowita uzyskana w pierwszym mnożeniu odpowiada **najbardziej znaczącemu bitowi — MSB** (ang. *Most Significant Bit*), a uzyskana w ostatnim bitowi LSB. Pomiędzy liczbą $0{,}45_{10}$ a liczbą $0{,}01110011\ldots_2$ nie można postawić znaku równości, gdyż ciąg części całkowitych generowanych przez powyższy algorytm jest ciągiem nieskończonym. Nie jest to cechą tego właśnie algorytmu, lecz wynika z wybranej do przykładu liczby dziesiętnej. Zauważmy jednak, że kolejna jedynka, która pojawi się na pozycji „–11" będzie miała wagę $2^{-11} = 1/2048$. Uzyskany wynik jest więc pewnym przybliżeniem liczby dziesiętnej. Ta „niedokładność" nie jest jednak cechą systemu dwójkowego czy każdego innego niż dziesiętny. W systemie dziesiętnym także bardzo często musimy zadowolić się jedynie przybliżeniem pewnych liczb (np. pierwiastek kwadratowy z liczby 2), gdyż ich dokładny reprezentant dziesiętny nie istnieje (zawiera nieskończenie wiele pozycji).

Liczby dwójkowe są zawsze „dłuższe" (wymagają większej liczby pozycji) niż odpowiadające im liczby dziesiętne. Jednak w systemach cyfrowych pamiętanie, przetwarzanie i przesyłanie nawet licznych, ale prostych bo dwustanowych sygnałów, jest zawsze łatwiejsze niż realizowanie tych samych operacji z mniejszą liczbą sygnałów wielowartościowych.

Liczby dwójkowe są zwykle długie, co utrudnia ich zapis, zwiększa możliwość pomyłek i wydłuża czas przy opisywaniu sygnałów w fizycznym systemie cyfrowym. Aby uniknąć tych wad, wprowadza się niekiedy grupowanie trzech lub czterech cyfr dwójkowych i oznacza je jednym symbolem. Łatwo zauważyć, że sprowadza się to do wyrażenia danej liczby w systemie **ósemkowym** lub **szesnastkowym**. Na przykład

$$87_{10} = \mathbf{1010111}_2 = \mathbf{001\ 010\ 111} = 127_8$$
$$\Downarrow\ \ \Downarrow\ \ \Downarrow$$
$$1\ \ 2\ \ 7$$
$$87_{10} = \mathbf{1010111}_2 = \mathbf{0101\ 0111} = 57_{16} = 57\text{H}$$
$$\Downarrow\ \ \Downarrow$$
$$5\ \ 7$$

W wielu urządzeniach cyfrowych operuje się liczbami dwójkowymi o 8 cyfrach dwójkowych. Grupa takich ośmiu cyfr to tak zwany **bajt** (ang. *byte*). Cyfra

dwójkowa jest zaś często nazywana **bitem** (ang. *bit*), czyli bajt to 8 bitów. ***Bit jest najmniejszą porcją informacji.***

W układach cyfrowych występuje często konieczność wykonywania operacji arytmetycznych na sygnałach, przedstawionych w postaci liczb dwójkowych. Reguły obowiązujące przy realizacji podstawowych działań arytmetycznych (takich jak: dodawanie, odejmowanie, mnożenie czy dzielenie) na liczbach dwójkowych są bardzo proste i nie różnią się niczym od tych stosowanych w systemie dziesiętnym, jeżeli uwzględnimy tylko wpływ innej podstawy systemu. Popatrzmy tylko na poniższe przykłady.

Realizacja działań arytmetycznych na liczbach w zapisie dwójkowym jest, jak widać, banalna. Zbędną staje się także znajomość tabliczki mnożenia, która w swoim czasie sprawiała wiele trudu uczącym się liczyć w systemie dziesiętnym.

```
  9        1001          9        1001
+ 5      + 0101        - 5      - 0101
 14         1110          4        0100

  9        1001
· 5      · 0101          7,5          111,1
 45         1001        45 :6       101101 :110
         + 0000         -42          -110
           1001          =30         =1010
         0000            -30          -110
         0101101          ==          =1001
                                       -110
                                       ==110
                                        -110
                                        ===
```

Mimo to ludzie na pewno nie zaczną posługiwać się systemem dwójkowym — ze względu na rozwlekłość takiej reprezentacji liczb.

Pewne komplikacje w realizacji powyższych działań pojawią się, jeżeli zechcemy uwzględnić liczby ujemne, a także gdy będziemy odejmować liczbę większą od mniejszej. Powodem prezentacji powyższych przykładów jest zwrócenie Czytelnikowi uwagi na prostotę wykonywanych operacji, co w praktycznej realizacji będzie po prostu wymagało nieskomplikowanych rozwiązań układowych. Popatrzmy jeszcze raz na operację mnożenia. Łatwo zauważyć, że sprowadza się ona do dwóch czynności: sumowania i przesuwania. Jeżeli bit mnożnika, przez który aktualnie mnożymy mnożną, jest równy **0**, to tylko przesuwamy mnożną o jedno miejsce w kierunku starszych bitów, a jeśli jest równy **1**, to poza przesunięciem wykonujemy jeszcze sumowanie z wynikiem uzyskanym w poprzednich krokach.

Pytania i zadania

1. Wykonaj konwersję dziesiętno-dwójkową liczb:
   a) 267,39
   b) 427,83
   c) 184,52

2. Wykonaj konwersję dwójkowo-dziesiętną liczb binarnych:
   a) **110011001,110011**
   b) **101010101,10101**
3. Znajdź postać piątkową (zapis w systemie o podstawie 5) liczb dziesiętnych jak w zadaniu 1 (będzie to więc konwersja dziesiętno-piątkowa).
4. Wykonaj konwersję szóstkowo-dziesiętną liczb:
   a) 415,41
   b) 130,52
   c) 55,45
5. Zamień na postać dwójkową, a następnie ósemkową i szesnastkową liczbę dziesiętną:
   a) 1548
   b) 741
   c) 194
6. Zamień na postać dwójkową liczbę w zapisie heksadecymalnym:
   a) 1A9F
   b) FFBC
   c) ABBA
7. Zamień liczby w zapisie szesnastkowym z zadania 6. na postać dziesiętną bezpośrednio z zapisu heksadecymalnego.
8. Wykonaj następujące operacje arytmetyczne w zapisie dwójkowym (podane liczby są zapisane w systemie dziesiętnym):
   a) 75 + 112; 109 – 88; 19 · 5; 49 : 6
   b) 67 + 76; 92 – 71; 14 · 6; 65 : 6
   Uzyskane wyniki sprawdź przeliczając je na system dziesiętny.
9. Wykonaj następujące operacje arytmetyczne w zapisie szóstkowym (podane liczby już są zapisane w systemie szóstkowym):
   a) 45 + 13; 53 – 34; 34 · 15; 44 : 5
   b) 52 + 14; 42 – 34; 54 · 3; 35 : 4
   Uzyskane wyniki sprawdź, przeliczając dane oraz wyniki na system dziesiętny.

Wskazówka. Zasady wykonywania operacji arytmetycznych w systemach pozycyjnych są jednakowe. Wymagają jedynie uwzględnienia aktualnej podstawy systemu p. Sumowanie liczb dziesiętnych wygląda następująco: 8 + 7 = 15. Cyfra 1 w wyniku wskazuje, że suma zawiera jedną dziesiątkę (p = 10) i pozostaje 5 jednostek. Sumowanie liczb np. siódemkowych przebiega analogicznie: 5 + 3 = 11. Cyfra 1 w wyniku wskazuje, że suma zawiera jedną siódemkę (p = 7) i pozostaje jeszcze 1 jednostka. Mnożenie 4 · 8 = 32 (w zapisie dziesiętnym) i analogicznie 4 · 6 = 33 (w zapisie siódemkowym). W obu przykładach mechanizm powstawania iloczynu jest taki sam. W pierwszym mnożeniu (4 · 8) uzyskujemy 3 dziesiątki (p = 10) oraz 2 jednostki, w drugim zaś (4 · 6) 3 siódemki (p = 7) oraz 3 jednostki.

1.3. Kody

Czynność przypisywania różnym informacjom pewnych symboli jest nazywana* kodowaniem, *a zestaw symboli przypisany danej informacji* — kodem *tej informacji.

Opisany wcześniej system dwójkowy zapisu liczb jest więc **naturalnym kodem dwójkowym**. System dziesiętny jest **kodem dziesiętnym** itp.

Jednak naturalny kod dwójkowy w wielu zastosowaniach nastręcza bardzo dużo problemów technicznych przy realizacji pewnych operacji. Taką trudność stanowi na przykład konwersja na system dziesiętny. Większość systemów cyfrowych komunikuje się z człowiekiem i informacja wyprowadzana przez te urządzenia musi być przedstawiona w systemie dziesiętnym (np. wskaźniki cyfrowe, drukarki). Bezpośrednie przejście z naturalnego kodu dwójkowego na system dziesiętny jest technicznie trudne, gdyż cyfry dziesiętne nie mają żadnego stałego odpowiednika w ciągu symboli binarnych. Realizacje są znacznie prostsze, gdy każdej cyfrze dziesiętnej przyporządkuje się na stałe określoną liczbę binarną. Koduje się po prostu każdą cyfrę oddzielnie, a nie całą liczbę dziesiętną. Takie kody są znane pod nazwą **kodów dwójkowo-dziesiętnych (2/10)** lub **kodów BCD** (ang. *Binary Coded Decimal*).

Na przykład

$317_{10} = \mathbf{100111101}_2 =$ **0011** **0001** **0111** $_{2/10}$

oddzielnie kodowana cyfra 3 1 7

W powyższym przykładzie cyfry 3, 1 i 7 zostały zakodowane 4-bitowym kodem naturalnym. Taki kod BCD jest nazywany w związku z tym **kodem naturalnym BCD** lub **kodem BCD 8421.** Druga nazwa pochodzi od pierwszych czterech wag w systemie dwójkowym naturalnym.

Z porównania zapisu dwójkowego i dwójkowo-dziesiętnego wynika, że zwiększa się liczba bitów (a tym samym i sygnałów w układzie cyfrowym), lecz prostota przetwarzania całkowicie to rekompensuje.

Tablica 1.1. **Kody dwójkowo-dziesiętne (BCD)**

Kod Nazwa	I 8421	II Aikena	III Johnsona	IV 2 z 5	V 1 z 10
Bity	D C B A	D C B A	E D C B A	E D C B A	K I H G F E D C B A
Wagi	8 4 2 1	2 4 2 1	– – – – –	– – – – –	9 8 7 6 5 4 3 2 1 0
0	**0 0 0 0**	**0 0 0 0**	**0 0 0 0 0**	**0 0 0 1 1**	**0000000001**
1	**0 0 0 1**	**0 0 0 1**	**0 0 0 0 1**	**0 0 1 0 1**	**0000000010**
2	**0 0 1 0**	**0 0 1 0**	**0 0 0 1 1**	**0 1 0 0 1**	**0000000100**
3	**0 0 1 1**	**0 0 1 1**	**0 0 1 1 1**	**1 0 0 0 1**	**0000001000**
4	**0 1 0 0**	**0 1 0 0**	**0 1 1 1 1**	**0 0 1 1 0**	**0000010000**
5	**0 1 0 1**	**1 0 1 1**	**1 1 1 1 1**	**0 1 0 1 0**	**0000100000**
6	**0 1 1 0**	**1 1 0 0**	**1 1 1 1 0**	**1 0 0 1 0**	**0001000000**
7	**0 1 1 1**	**1 1 0 1**	**1 1 1 0 0**	**0 1 1 0 0**	**0010000000**
8	**1 0 0 0**	**1 1 1 0**	**1 1 0 0 0**	**1 0 1 0 0**	**0100000000**
9	**1 0 0 1**	**1 1 1 1**	**1 0 0 0 0**	**1 1 0 0 0**	**1000000000**

Dziesięć cyfr dziesiętnych można zakodować binarnie na wiele różnych sposobów. Liczba możliwych kodów BCD jest olbrzymia. Nie wszystkie oczywiście znalazły praktyczne zastosowanie.

Przykłady kodów 2/10 (BCD) przedstawiono w tabl. 1.1.

Kod I — utworzony poprzez przyporządkowanie kolejnym cyfrom dziesiętnym ich reprezentacji dwójkowych jest wspomnianym wyżej kodem **naturalnym BCD**.

Kod II — zwany **kodem Aikena** — jest kodem wagowym (tzn. każdej pozycji można przypisać odpowiednią wagę, podobnie jak w naturalnym kodzie binarnym z tym, że waga nie jest prostą funkcją pozycji — nie jest to więc kod pozycyjny). Jego szczególną cechą jest oś „antysymetrii". Oś ta przebiega pomiędzy kodem cyfry 4 i cyfry 5. Wyrazy tego kodu leżące w jednakowej odległości od tej osi różnią się negacją wszystkich bitów. Na przykład kod cyfry 3 — **0011** po zanegowaniu wszystkich bitów staje się kodem cyfry 6 — **1100**. Kod ten można także otrzymać poprzez dodanie liczby 6 (**0110**) do kodowanej cyfry, poczynając od cyfry 5. Pierwszych pięć wyrazów tego kodu jest bowiem identycznych jak w kodzie **8421**. Ta cecha „antysymetrii" jest szczególnie przydatna przy realizacji operacji arytmetycznych.

Kod III — zwany **kodem Johnsona** (lub inaczej **kodem pseudopierścieniowym**) — wymaga pięciu bitów do zakodowania każdej cyfry dziesiętnej. Charakteryzuje się specyficznym rozkładem zer i jedynek. Cecha ta znacznie upraszcza dekodowanie. Kod ten nie jest kodem wagowym (a tym bardziej pozycyjnym).

Kod IV jest reprezentantem całej rodziny **kodów ze stałym indeksem,** tzn. ze stałą liczbą jedynek w zapisie binarnym. Nazwę kodu **2 z 5** należy więc rozumieć tak, że każdy 5-bitowy wyraz tego kodu ma 2 jedynki i same zera. Kod „2 z 5 z negacją" (oznaczany: $\overline{\mathbf{2\ z\ 5}}$) jest także kodem ze stałym indeksem, ale w tym kodzie każdy 5-bitowy wyraz zawiera 2 zera i same jedynki. Liczba 2 dotyczy tym razem zer, a nie jedynek.

Wśród kodów ze stałym indeksem szczególne znaczenie ma kod **1 z *n*** lub ($\overline{\mathbf{1\ z\ n}}$). Jest on bowiem często **kodem pierwotnym,** to znaczy kodem wejściowym urządzenia. Wprowadzanie informacji do systemu cyfrowego często jest realizowane za pośrednictwem klawiatury z przyciskami. Jeden z przycisków jest wciśnięty, a pozostałe zwolnione — co odpowiada kodowi **1 z *n***. Wprowadzanie cyfr z klawiatury polega na wciśnięciu jednego z dziesięciu klawiszy przypisanych kolejnym cyfrom dziesiętnym. Informacja wejściowa ma więc postać „1 klawisz z 10 jest wciśnięty". (Jeżeli klawiatura jest wyposażona w większą liczbę przycisków, to zwykle są stosowane bardziej złożone układy identyfikacji wciśniętego klawisza.)

Innym przykładem może być sterowanie ruchem jakiegoś urządzenia: „w lewo", „w prawo", „stop" — najdogodniej jest zakodować wstępnie jako **100**, **010** i **001** (trzy przyciski będą służyły do wprowadzania tych poleceń). Dopiero potrzeba dalszego przetwarzania informacji wejściowej powoduje, że warto zamienić ją na krótsze wyrażenie. Kod pierwotny **1 z 10** (dziesięć sygnałów) można więc zamienić np. na kod **naturalny BCD** (tylko cztery sygnały).

Jak już wspomniano wcześniej, wprowadzenie zapisu liczb w kodzie **BCD** (zamiast zapisu naturalnego) wydłuża ten zapis. Użycie kodu **BCD** 5-bitowego zamiast 4-bitowego (np. kodu **2 z 5**) powoduje dalsze wydłużenie tego zapisu. W technicznej realizacji przetwarzanie takiej informacji wymaga zwiększenia liczby sygnałów, a więc komplikuje urządzenie lub wydłuża czas przetwarzania. Jaki jest więc cel stosowania np. kodu **2 z 5**?

Przyjmijmy, że naszym zadaniem jest przesłanie na pewną odległość informacji zakodowanej w postaci kodu **2 z 5**. Zatem wysyłamy słowa 5-bitowe o postaci jak w tabl. 1.1. Wiemy, że proces transmisji jest szczególnie narażony na wpływ czynników zakłócających. Mimo, że dwustanowe sygnały są dość odporne na zakłócenia (była o tym mowa w p. 1.1), to może się zdarzyć, że jakiś nadany bit o wartości **1** zostanie odebrany jako bit o wartości **0** (lub na odwrót). Otrzymane wówczas na wejściu urządzenia, odbierającego przesyłaną informację, słowo binarne będzie miało jedną jedynkę lub trzy jedynki. Jeżeli urządzenie odbiorcze wyposażymy w układ kontrolujący liczbę jedynek w odbieranym słowie, to każdą taką sytuację natychmiast zidentyfikujemy. Układ sterujący procesem transmisji może w takiej sytuacji ponowić przesyłanie informacji albo zatrzymać przesyłanie i zasygnalizować wystąpienie błędu (zakłócenia). W pierwszym przypadku (powtórzenia transmisji) użytkownik nawet nie zauważy, że wystąpiło zakłócenie i zinterpretuje to jako „odporność" na zakłócenia.

Można sobie wyobrazić, że w nadanym słowie 5-bitowym w wyniku zakłócenia jeden bit zmieni wartość z **0** na **1**, drugi zaś z **1** na **0** i tak „zdeformowana" informacja dotrze do układu odbiorczego. Układ kontrolujący odbieraną informację nic wówczas „nie zauważy". Rzeczywiście, taki podwójny błąd może w praktyce wystąpić i nie zostać zauważony. Ale prawdopodobieństwo, że dwa impulsy zakłócające (i to o przeciwnych kierunkach oddziaływania) wystąpią w tym samym czasie jest dużo, dużo mniejsze. Próg „odporności" na zakłócenia informacji kodowanej przy użyciu kodu **2 z 5** jest więc bardzo wysoki.

Kod mający cechę sobie właściwą, która może być identyfikowana, jest nazywany **kodem z zabezpieczeniem** lub **kodem detekcyjnym**. Kod taki jest kodem o zwiększonej odporności na zakłócenia. Chociaż, jak już wyjaśniono, „odporność" nie jest tu adekwatnym określeniem, bowiem kod daje się zakłócać jak każdy inny — ale istnieje możliwość detekcji zakłócenia. Istnieją także kody rzeczywiście odporne na zakłócenia. Konstrukcja takiego kodu pozwala na odtworzenie pierwotnej informacji w urządzeniu odbierającym nawet wówczas, gdy wystąpił błąd (zakłócenie), np. **kod Hamminga.**

Odporność na zakłócenia kodu **2 z 5** (czy generalnie kodu ze stałym indeksem) jest jego cechą „wrodzoną", to znaczy już sama zasada konstrukcji kodu sprawia, że ma on cechę charakterystyczną, która może być identyfikowana i sprawdzana. Wiele kodów nie ma jednak takiej odporności na zakłócenia. Najprostszą metodą zwiększenia odporności kodu na zakłócenia jest dodanie tak zwanego **bitu parzystości**. Na przykład: kod **naturalny BCD** uzupełniony o bit parzystości będzie miał następującą postać:

	D	C	B	A	P	
0	**0**	**0**	**0**	**0**	**0**	
1	**0**	**0**	**0**	**1**	**1**	
2	**0**	**0**	**1**	**0**	**1**	Bit dodatkowy P pełni rolę
3	**0**	**0**	**1**	**1**	**0**	bitu parzystości.
4	**0**	**1**	**0**	**0**	**1**	
5	**0**	**1**	**0**	**1**	**0**	
6	**0**	**1**	**1**	**0**	**0**	
7	**0**	**1**	**1**	**1**	**1**	
8	**1**	**0**	**0**	**0**	**1**	
9	**1**	**0**	**0**	**1**	**0**	

Zauważmy, że dodatkowy piąty bit ma taką wartość, że liczba jedynek w nowo powstałym słowie 5-bitowym jest liczbą parzystą. Kod więc uzyskuje w ten sposób pewną cechę charakterystyczną i staje się kodem detekcyjnym.

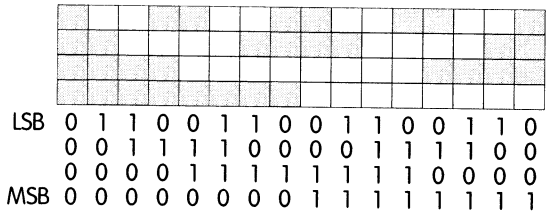
W taki sposób (za pomocą bitu parzystości) można zabezpieczyć dowolny kod. Zazwyczaj nie dodaje się bitu parzystości, jeżeli informacja jest przetwarzana wewnątrz urządzenia i przesłania dotyczą modułów znajdujących się w jednej obudowie. Jeżeli jednak transmisja ma być realizowana pomiędzy różnymi urządzeniami, a odległość między nimi jest liczona w metrach, to wówczas nadawana informacja jest uzupełniana o dodatkowy bit — bit parzystości. Urządzenie odbiorcze kontroluje, czy w odbieranych słowach binarnych jest parzysta liczba jedynek czy nieparzysta. Pojedyncze zakłócenie zostanie więc łatwo zidentyfikowane. Zakłócenie podwójne, podobnie jak w przypadku kodu **2 z 5**, nie zostanie rozpoznane. Także zakłócenie podwójne jednokierunkowe (tzn. jednoczesna zmiana dwóch bitów z **0** na **1** lub na odwrót) nie zostanie zidentyfikowane. Kod **2 z 5** jest natomiast odporny na takie zakłócenia. Prawdopodobieństwo wystąpienia podwójnego błędu jednokierunkowego jest większe niż prawdopodobieństwo wystąpienia podwójnego błędu dwukierunkowego. Odporność kodów zaopatrzonych w bit parzystości jest więc niższa niż kodów ze stałym indeksem.

Zamiast parzystości można także kontrolować nieparzystość. Dodatkowy bit — **bit nieparzystości** — powinien mieć wówczas taką wartość, aby liczba jedynek w słowie kodowym była liczbą nieparzystą. Nie ma to jednak wpływu na poziom odporności kodu na zakłócenia.

Pobieranie informacji wejściowej przez urządzenia cyfrowe bardzo często dotyczy wartości przesunięć liniowych lub kątowych. Są to wielkości analogowe. Mogą być one bezpośrednio zamieniane na postać cyfrową, kodowaną dwójkowo za pomocą liniałów lub tarcz kodowych.

Na rysunku 1.3 przedstawiono poglądowo liniał kodowy o czterech ścieżkach. Nad każdą ścieżką jest umieszczone źródło światła. Pod każdą ścieżką liniału są elementy fotoczułe (fotodiody, fotorezystory, fototranzystory). Jeżeli strumień światła przechodzi do elementu światłoczułego, to generuje on bit informacji o wartości **1**, natomiast element nie oświetlony — bit o wartości **0**. Układ przesłoniętych i otwartych okienek określa kod przypisany kolejnym położeniom (przemieszcza się liniał lub układ elementów fotooptycznych). Dokładniejsze odwzorowanie wymaga użycia większej liczby ścieżek (bitów), czyli skwantowania

określonego przedziału zmienności na większą liczbę odcinków. W celu uniknięcia dużych błędów przy odczytywaniu wartości przesunięć, szczególnie w przypadku gdy odczyt występuje na granicy dwóch kolejnych stref o różnych kombinacjach kodowych, stosuje się kombinacje różniące się na jednej pozycji. Warunku takiego nie spełnia żaden z poznanych do tej pory kodów. Najczęściej stosowanym kodem mającym cechę: ***dwa sąsiednie wyrazy kodowe różnią się tylko jednym bitem*** — jest **kod Graya,** zwany też **refleksyjnym** ze względu na sposób konstruowania jego wyrazów.



Rys. 1.3. Liniał kodowy

Kod Graya jednobitowy ma oczywiście postać jednoznaczną. Aby rozszerzyć go do dwóch bitów, należy jeszcze raz przepisać ciąg wyrazów kodu jednobitowego, ale w odwrotnej kolejności (znaleźć symetryczne odbicie). Następnie do początkowych wyrazów dopisać dodatkowy bit **0**, a do dopisanych (lustrzanych) — dodatkowy bit **1**. Na przykład

A		B A		C B A
0	**0**	**0 0**	**0 0**	**0 0 0**
1	**1**	**0 1**	**0 1**	**0 0 1**
	—	**1 1**	**1 1**	**0 1 1**
	1	**1 0**	**1 0**	**0 1 0**
kod	**0**		— —	**1 1 0**
1 bitowy	symetr.		**1 0**	**1 1 1**
	odbicie	kod 2-bitowy	**1 1**	**1 0 1**
		(po dopisa-	**0 1**	**1 0 0**
		niu 0 i 1)	**0 0**	
			symetr.	kod 3-bitowy itd.
			odbicie	

Kod Graya *n*-bitowy tworzy się analogicznie, to znaczy powtarzając (w odwrotnej kolejności) $n-1$ bitowy kod Graya i dopisując dodatkowy bit (**0** w pierwotnej części i **1** w części dopisanej). Zauważmy, że kod ten jest **kodem cyklicznym** — ***to znaczy, że pierwszy wyraz kodu i ostatni również spełniają zasadę na jakiej skonstruowano ten kod.***

W złożonych systemach cyfrowych (systemy mikroprocesorowe, komputery) najbardziej rozpowszechnionym kodem jest kod ASCII (ang. *American Standard Code for Information Interchange* — standardowy 8-bitowy kod do przesyłania informacji między komputerami a urządzeniami peryferyjnymi). W zasadzie w tym kodzie informacja została przypisana jedynie połowie słów 8-bitowych, czyli wystarczy 7 bitów do jego zapisu (patrz dodatek), ***a ósmy bit jest wykorzystywany jako bit parzystości.***

Pytania i zadania

1. Co to jest kod **dwójkowo-dziesiętny** (inaczej **BCD**)? Jakie są powody stosowania kodów **BCD** zamiast naturalnego kodu dwójkowego?
2. Zapisz w kodzie **BCD 8421** liczby:
   a) 173,
   b) 391,
   c) 97.

   Wykonaj konwersję dziesiętno-dwójkową tych liczb i porównaj długość (w bitach) obu zapisów.
3. Co to jest bit parzystości i kontrola parzystości? Kiedy słowa kodowe uzupełnia się o bit parzystości? Zapisz kod Aikena, uzupełniając go bitem parzystości.
4. Który z kodów jest bardziej „odporny" na zakłócenia: kod **2 z 5** czy kod wyposażony w bit parzystości? Wyjaśnij na czym polega „odporność na zakłócenia" kodów detekcyjnych.
5. Zapisz 4-bitowy kod Graya. Jaką cechę ma ten kod?

2

Podstawy matematyczne

2.1. Wprowadzenie

Jak już wiemy, technika cyfrowa jest realizowana jako dwustanowa. Oznacza to, że wszelkie sygnały mogą przyjmować dwie wartości umownie oznaczane przez **0** i **1**. Wszelka informacja — występująca i przetwarzana w układach cyfrowych — w zapisie symbolicznym będzie się sprowadzała do tych dwóch wartości. Sygnałów może być oczywiście wiele. Będą to więc słowa binarne, które po przetworzeniu także będą miały postać słów zero-jedynkowych.

Aby opisać w sposób sformalizowany (językiem matematyki) funkcje, których zarówno argumenty, jak i same wartości funkcji należą do zbioru **{0,1}**, nie wystarczy klasyczny aparat matematyczny (algebra). Funkcje arytmetyczne w wyniku dodawania wprowadzają liczbę 2 (1+1), w wyniku odejmowania liczby ujemne. Do opisu działania układów cyfrowych jest zaś potrzebny taki aparat matematyczny, który ***argumenty o wartościach ze zbioru* {0,1} *przetwarza w wartości ze zbioru* {0,1}**. Takim narzędziem matematycznym przydatnym do opisu układów cyfrowych jest dział logiki matematycznej — **dwuelementowa algebra Boole'a.** Ponieważ układy cyfrowe są opisywane językiem logiki matematycznej, przeto są nazywane także **układami logicznymi**.

2.2. Algebra Boole'a

Algebra Boole'a operuje zmiennymi dwuwartościowymi (**a**,**b**,**c**,..) o wartościach **0** oraz **1**. Wprowadza ona trzy nowe operacje (funkcje), których argumentami i wynikami są zawsze elementy **0** lub **1**. Operacje te są zdefiniowane w sposób następujący:

Suma logiczna (alternatywa, dysjunkcja) ***jest równa jedności, gdy którykolwiek ze składników jest równy jedności. Sumę argumentów*** **a** ***i*** **b** ***oznacza się jako*** **a + b.**

Iloczyn logiczny (koniunkcja) *jest równy jedności, gdy wszystkie czynniki są równe jedności. Iloczyn argumentów* **a** *i* **b** *oznacza się przez* **a·b** *lub krócej* **ab.**

Negacja (dopełnienie) *jest działaniem jednoargumentowym i jest równa jedności, gdy argument ma wartość zero. Negację oznacza się przez* $\bar{a}$ *i czyta* **„nie a".**

Stosowane są także i inne operatory działań bulowskich, np. suma: $a \vee b$, iloczyn: $a \wedge b$, a&b, negacja: a′, ~a, ¬a, a#, /a. W technice zazwyczaj stosuje się symbole zwykłej sumy i iloczynu z uprzednim zaznaczeniem, czy chodzi o operacje arytmetyczne czy logiczne. W podręczniku pisząc krótko „suma" czy „iloczyn" będziemy rozumieć sumę logiczną oraz iloczyn logiczny; wyjątkowo umowa powyższa zostanie zawieszona, co będzie wyraźnie zaznaczone.

Z podanych definicji wynika, że:

$$\left.\begin{array}{llll} 0+0=0 & 0\cdot 0=0 & & \\ 0+1=1 & 0\cdot 1=0 & \bar{0}=1 & \\ 1+0=1 & 1\cdot 0=0 & \bar{1}=0 & \\ 1+1=1 & 1\cdot 1=1 & & \end{array}\right\} \quad (2.1)$$

Zapisane powyżej operacje sumy i iloczynu dotyczą działań dwuargumentowych, chociaż definicja odnosi się do operacji o dowolnej liczbie argumentów. Działania te można przedstawić w postaci następującej tablicy:

Tablica 2.1. **Definicje operacji bulowskich**

a	b	a + b	ab	$\bar{a}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Z powyższych zapisów wynikają pewne ogólniejsze prawa, które zapiszemy w dwóch grupach:

$$\left.\begin{array}{lll} \text{A1:} & a+b=b+a & \text{B1:} \quad ab=ba \\ \text{A2:} & a(b+c)=ab+ac & \text{B2:} \quad a+bc=(a+b)(a+c) \\ \text{A3:} & (a+b)+c=a+(b+c) & \text{B3:} \quad (ab)c=a(bc) \\ \text{A4:} & a+0=a & \text{B4:} \quad a1=a \\ \text{A5:} & a+1=1 & \text{B5:} \quad a0=0 \\ \text{A6:} & a+\bar{a}=1 & \text{B6:} \quad a\bar{a}=0 \\ & & \bar{\bar{a}}=a \end{array}\right\} \quad (2.2)$$

Powyższe tożsamości można łatwo sprawdzić, podstawiając w nich wszystkie możliwe wartości argumentów (po stronie lewej i prawej) i porównując obie strony. Korzystając z zależności (2.2) albo podstawiając wszystkie możliwe wartości argumentów, można wykazać słuszność następujących twierdzeń:

Tw. 1	$\mathbf{a + ac = a}$	$\mathbf{a(a + b) = a}$	(2.3)
Tw. 2	$\mathbf{a + \bar{a}b = a + b}$	$\mathbf{a(\bar{a} + b) = ab}$	(2.4)
Tw. 3	$\mathbf{a + a = a}$	$\mathbf{aa = a}$	(2.5)
Tw. 4	$\mathbf{\overline{a + b} = \bar{a}\bar{b}}$	$\mathbf{\overline{ab} = \bar{a} + \bar{b}}$	(2.6)

Szczególne znaczenie oraz przydatność przy przekształceniach wyrażeń bulowskich ma twierdzenie 4, które nosi nazwę **praw de Morgana.**

Ponieważ w wyniku działań sumy, iloczynu i negacji otrzymuje się wartość **0** lub **1**, więc wyniki można uważać za argumenty innych działań, tworząc w ten sposób formuły wielostopniowe, np.

$$\mathbf{ab + \bar{a}c(\overline{b + de})}$$

Oprócz sumy, iloczynu i negacji w zastosowaniach praktycznych duże znaczenie mają również inne funkcje, które wprawdzie można zapisać za pomocą operacji bulowskich $\{+, \cdot, \bar{}\}$, ale dogodniej jest je nazwać i oznaczyć odrębnie.

Funkcja Pierce'a $\mathbf{a \downarrow b}$ ma wartość **1**, jeśli $\mathbf{a = 0}$ i $\mathbf{b = 0}$. W zapisie bulowskim funkcja ta wyrazi się zależnością

$$\mathbf{a \downarrow b = \bar{a}\bar{b} = \overline{a + b}}$$

Funkcja ta w zapisie bulowskim jest „negacją sumy" stąd ona jak i jej **funktor** (element realizujący funkcję) występuje zwykle pod nazwą **NOR** (ang. **N**ot-**OR**).

Funkcja Sheffera $\mathbf{a \mid b}$ ma wartość **1**, jeśli $\mathbf{a = 0}$ lub $\mathbf{b = 0}$. W zapisie bulowskim funkcja ta wyrazi się zależnością

$$\mathbf{a \mid b = \bar{a} + \bar{b} = \overline{ab}}$$

Funkcja ta w zapisie bulowskim jest „negacją iloczynu" stąd ona i jej funktor występuje zwykle pod nazwą **NAND** (ang. **N**ot-**AND**).

Suma modulo 2 (różnica symetryczna, nierównoważność) $\mathbf{a \oplus b}$ $(\mathbf{\overline{a \equiv b}})$ ma wartość **1**, gdy tylko jeden argument ma wartość **1**. W zapisie bulowskim

$$\mathbf{a \oplus b = \overline{a \equiv b} = a\bar{b} + \bar{a}b}$$

Funkcja ta, jak i funktor ją realizujący, jest nazywana **Ex-OR** (ang. **Ex**clusive **OR**).

Równoważność $\mathbf{a \otimes b}$ $(\mathbf{a \equiv b}, \mathbf{a \sim b})$ ma wartość **1**, gdy argumenty mają jednakowe wartości. W zapisie bulowskim

$$\mathbf{a \otimes b = a \equiv b = ab + \bar{a}\bar{b} = \overline{a \oplus b}}$$

Funkcja ta, jak i funktor ją realizujący, jest nazywana **Ex-NOR** (ang. **Ex**clusive **N**ot **OR**).

Powody, dla których powyżej wymienione funkcje zostały uznane za najważniejsze, wyjaśniono w rozdz. 3.

Liczba N_f wszystkich funkcji n-argumentowych jest określona zależnością

$$N_f = 2^{2^n} \tag{2.7}$$

Dla liczby zmiennych $n = 1$ liczba N_f wyniesie 4. Oznacza to, że jednoargumentowych funkcji bulowskich jest dokładnie 4. Są to:

funkcja stała **1**, $\mathbf{f(a) = 1}$;
funkcja stała **0**, $\mathbf{f(a) = 0}$;
oraz $\mathbf{f(a) = a}$ i $\mathbf{f(a) = \bar{a}}$.

Z zależności (2.7) wynika, że dwuargumentowych funkcji może być 16, trzyargumentowych 256 itd. W tablicy 2.2 sporządzono wykaz wszystkich 16 funkcji dwóch zmiennych.

Tablica 2.2. **Funkcje logiczne dwóch zmiennych**

Nr	**a 0 0 1 1** **b 0 1 0 1**	Symbol (operator)	Zapis bulowski	Nazwa funkcji
0	**0 0 0 0**		**0**	stała **0**
1	**0 0 0 1**	·, *, &, ∧,	**ab**	iloczyn logiczny
2	**0 0 1 0**	Δ, −	$\mathbf{a\Delta b = a\bar{b}}$	iloczyn zakazu na **b**, różnica niesymetryczna
3	**0 0 1 1**		**a**	
4	**0 1 0 0**	Δ, −	$\mathbf{b\Delta a = b\bar{a}}$	iloczyn zakazu na **a**, różnica niesymetryczna
5	**0 1 0 1**		**b**	
6	**0 1 1 0**	⊕	$\mathbf{a \oplus b = a\bar{b} + \bar{a}b}$	suma modulo 2, Ex-OR
7	**0 1 1 1**	+, ∨	**a + b**	suma logiczna
8	**1 0 0 0**	↓	$\mathbf{a \downarrow b = \overline{a + b} = \bar{a}\bar{b}}$	funkcja Pierce'a. NOR
9	**1 0 0 1**	⊗, ≡, ~	$\mathbf{a \otimes b = ab + \bar{a}\bar{b}}$	rownoważność, Ex-NOR
10	**1 0 1 0**	¯, ′, ¬	$\mathbf{\bar{b}}$	negacja **b**
11	**1 0 1 1**	→	$\mathbf{b \rightarrow a = a + \bar{b}}$	implikacja **a** przez **b**
12	**1 1 0 0**		$\mathbf{\bar{a}}$	negacja **a**
13	**1 1 0 1**	→	$\mathbf{a \rightarrow b = \bar{a} + b}$	implikacja **b** przez **a**
14	**1 1 1 0**	|	$\mathbf{a \mid b = \overline{ab} = \bar{a} + \bar{b}}$	funkcja Sheffera, NAND
15	**1 1 1 1**		**1**	stała **1**

Opisany aparat matematyczny będzie przydatny w procesie syntezy układów cyfrowych, jeśli będziemy umieli się nim posługiwać. W tym celu należy dokładnie przerobić pytania i zadania zamieszczone na końcu tego rozdziału.

Dla ułatwienia tej pracy przeanalizujmy sposób postępowania przy rozwiązywaniu kilku poniższych zadań.

Przykład 2.1

Udowodnić równość
$\overline{w} + \overline{z} + (x+wz)(y+z) = 1$

Rozwiązanie

$L = \overline{w} + \overline{z} + (x + wz)(y + z) = \overline{w} + \overline{z} + xy + xz + wzy + wzz = \overline{w} + \overline{z} + xy + xz +$
$+ wzy + wz = \overline{w} + \overline{z} + xy + xz + wz(y + 1) = \overline{w} + \overline{z} + xy + xz + wz = \overline{w} + \overline{z} +$
$+ xy + x + w = 1 + \overline{z} + x(y + 1) = 1$ c.b.d.u.

W powyższych przekształceniach korzystano (kolejno) z następujących zależności: (2.2) A2, (2.5) Tw.3, (2.2) A2, (2.2) A5, 2·(2.4) Tw.2, (2.2) A6, (2.2) A2, (2.2) A5. Przekształcenia te pokazano bez żadnych skrótów (myślowych), starając się maksymalnie ograniczać liczbę wykonywanych operacji w jednym kroku, aby ułatwić ich analizę. W praktyce liczba kroków będzie mniejsza wskutek grupowania w jednym kroku kilku przekształceń. Nie jest to oczywiście jedyna sekwencja przekształceń prowadząca do celu. Nie można jej traktować jako jedynej drogi. Dla potwierdzenia powyższych słów popatrzmy na poniższe przekształcenia danego wyrażenia

$L = \overline{w} + \overline{z} + (x + wz)(y + z) = \overline{wz} + xy + xz + wzy + wz = 1$ c.b.d.u.

W pierwszym kroku zastosowano prawo de Morgana do wyrażenia $\overline{w} + \overline{z}$ oraz wymnożono wyrażenia w nawiasach. Uzyskany wynik pozwolił natychmiast stwierdzić słuszność tezy na podstawie zapisu A6 i A5 w zależności (2.2) ($a + \overline{a} = 1$, u nas $wz + \overline{wz} = 1$). ■

Przykład 2.2

Udowodnić równość
$ab + a\overline{b} + bc = ab + a\overline{b} + \overline{a}c$

Rozwiązanie

$L = ab + a\overline{b} + bc = ab + a\overline{b} + bc(a + \overline{a}) = ab + a\overline{b} + abc + \overline{a}bc = ab(1 + c) +$
$+ \overline{a}(\overline{b} + bc) = ab + \overline{a}(\overline{b} + c) = ab + a\overline{b} + \overline{a}c$ c.b.d.u.

W pierwszym kroku pomnożono jeden ze składników lewej strony (wybierając do tej operacji ten składnik, który nie występuje po stronie prawej) przez $1 = a + \overline{a}$, na co pozwalają tożsamości B4, A6 z zależności (2.2). Następnie po wymnożeniu, pogrupowaniu i skorzystaniu z Tw. 2 — wzór (2.4) uzyskano prawą stronę tezy. ■

Tezy z powyższych dwóch przykładów można było udowodnić także poprzez podstawienie wszystkich możliwych wartości argumentów. Równość lewej i prawej strony wyrażenia dla każdej kombinacji argumentów byłaby wystarczającym dowodem słuszności postawionej tezy. Byłby to sposób znacznie bardziej pracochłonny i znacznie mniej elegancki. W podrozdziale 3.3 poznamy jeszcze jedną metodę dowodzenia prawdziwości powyższych równości. Sposoby takie nie mogą jednak być zastosowane w kolejnym przykładzie.

Przykład 2.3

Doprowadzić do prostszej postaci następujące wyrażenie:
$(a + \bar{b})[abc + \bar{b}(a + c)] + ab\bar{c}(a + \bar{a}b)$

Rozwiązanie

$(a + \bar{b})[abc + \bar{b}(a + c)] + ab\bar{c}(a + \bar{a}b) = (a + \bar{b})(abc + a\bar{b} + \bar{b}c) + ab\bar{c}(a + b) =$
$= aabc + aa\bar{b} + a\bar{b}c + \bar{b}abc + \bar{b}a\bar{b} + \bar{b}\,\bar{b}c + ab\bar{c}a + ab\bar{c}b = abc + a\bar{b} + a\bar{b}c +$
$+ 0 + a\bar{b} + \bar{b}c + ab\bar{c} + ab\bar{c} = abc + a\bar{b} + a\bar{b}c + \bar{b}c + ab\bar{c} = ab(c + \bar{c}) + a\bar{b}(1 +$
$+ c) + \bar{b}c = ab + a\bar{b} + \bar{b}c = a(b + \bar{b}) + \bar{b}c = a + \bar{b}c.$ ■

Pytania i zadania

1. Sprawdzić, czy następujące równości są prawdziwe (zadanie zrealizować metodą przekształceń algebraicznych, a nie obliczania wartości wyrażeń dla wszystkich możliwych wartości argumentów):
   a) $(a + \bar{b} + ab)(a + \bar{b})\bar{a}b = 0$
   b) $(a + \bar{b} + a\bar{b})(ab + \bar{a}c + bc) = ab + \bar{a}c\bar{b}$
   c) $(ab + c + d)(\bar{c} + d)(\bar{c} + d + e) = ab\bar{c} + d$
   d) $\bar{a} + b + \bar{c} = \bar{a}(\bar{b} + \bar{c}) + bc + a\bar{c}$
   e) $ab + ac + bc = (a + b)(a + c)(b + c)$
   f) $b\bar{c} + \bar{a}c + \bar{a}b = b\bar{c} + \bar{a}c$
2. Udowodnić równości za pomocą przekształceń algebraicznych
   a) $ab + \bar{a}b\bar{c} + bc = b$
   b) $(a\bar{b} + c)(\bar{a} + b)\bar{c} = 0$
   c) $a\bar{b} + c + (\bar{a} + b)\bar{c} = 1$
   d) $(a + b)[x(a + y) + \bar{x}(a + c) + (y + z)\bar{z}a] = a + b(xy + \bar{x}c)$
   e) $ab(z + x) + \bar{a}b(\bar{x} + z) + x(\bar{a} + b) = \bar{a}(b + x) + b(x + z)$
   f) $x[(b + y)x + y(z + \bar{x}y)] + \bar{x}[(c + y)x + (a + \bar{x})y] = bx + y$
   g) $a\bar{b}(xy + \bar{x}y\bar{z}) + \bar{x}[\bar{a}y + \bar{b}(x + y)a] = (a\bar{b} + \bar{a}\,\bar{x})y$
3. Doprowadzić do prostszej postaci wyrażenia:
   a) $(a + \bar{a}\bar{b})[ac + a\bar{c}(b + \bar{b})]$
   b) $ab\bar{c} + (ab\bar{c} + \bar{a}c)[b(a + c) + \bar{b}c + \bar{b}a\bar{c}]$
   c) $(a \downarrow b) \mid [(a + (\bar{a} + xz)\bar{a})(\bar{z} + \bar{a})]$
4. Udowodnić równości metodą przekształceń algebraicznych:
   a) $a\bar{b} + ac + bc = a\bar{b} + bc$
   b) $(a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c)$
   c) $ab + \bar{a}\bar{b} + bcd = ab + \bar{a}\bar{b} + \bar{a}cd$
   d) $\bar{a}d(\bar{b} + c) + (b + \bar{c})\bar{a}\bar{d} + (\bar{b} + c)(b + \bar{c}) = (\bar{a}\bar{d} + \bar{b} + c)(\bar{a}d + b + \bar{c})$
   e) $bc + abd + a\bar{c} = bc + a\bar{c}$
   f) $abc + \bar{b}a\bar{c} + b\bar{c}d + \bar{b}cd + ad = abc + \bar{b}a\bar{c} + \bar{b}cd + b\bar{c}d$
   g) $ab + \bar{c}\bar{d} + \bar{a}bc\bar{d} + \bar{b}a\bar{c}\bar{d} = (a + \bar{d})(b + \bar{c})$
5. Sprawdzić, czy podane równości są prawdziwe:
   a) $a(b \oplus c) = (ab) \oplus (ac)$
   b) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
   c) $a \mid (b \mid c) = (a \mid b) \mid c$
   d) $a \downarrow (b \downarrow c) = (a \downarrow b) \downarrow c$

3

Układy kombinacyjne

3.1. Wprowadzenie

Układy cyfrowe *dzieli się na dwie podstawowe grupy:*
— układy kombinacyjne,
— układy sekwencyjne.

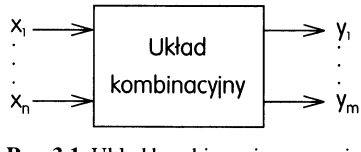
***W układzie kombinacyjnym każda kombinacja sygnałów wejściowych określa jednoznacznie kombinację sygnałów wyjściowych. Kombinacja sygnałów wejściowych jest nazywana* stanem wejść *układu* (*lub* słowem wejściowym), *natomiast kombinacja sygnałów wyjściowych* — stanem wyjść *układu* (słowem wyjściowym).**

Układ kombinacyjny w sposób ogólny przedstawiono na rys. 3.1. Schemat logiczny takiego układu można jednoznacznie opisać rodziną **funkcji przełączających (funkcji logicznych, funkcji bulowskich)**

$$y_i = f_i(x_1, x_2, ..., x_n) \qquad i = 1,2,...,m \tag{3.1}$$

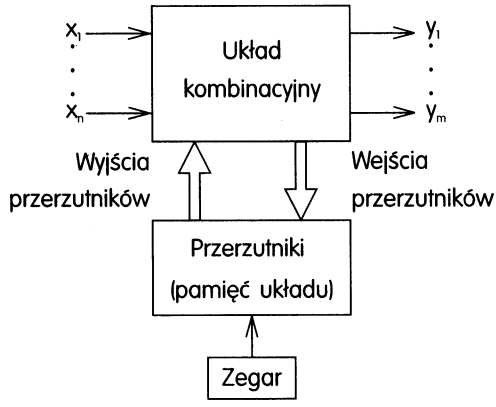
***Do realizacji fizycznej kombinacyjnych układów cyfrowych są stosowane* bramki logiczne (funktory)**. Postęp w zakresie scalania układów cyfrowych spowodował, że ostatnio wykorzystuje się także generatory funkcji logicznych zbudowane z multiplekserów lub pamięci stałej.

W **układzie sekwencyjnym** stan wejść nie określa w sposób jednoznaczny stanu wyjść. Słowo wyjściowe zależy także od poprzednich stanów wejściowych oraz ich kolejności występowania (sekwencji słów wejściowych — stąd nazwa: układy sekwencyjne). W sposób ogólny układ sekwencyjny można przedstawić tak jak na rys. 3.1 (tzn. tak samo, jak układ kombinacyjny). Ponieważ układ sekwencyjny uzależnia swe działanie także od wcześniej występujących stanów wejściowych, przeto musi być wyposażony dodatkowo w pa-



Rys. 3.1. Układ kombinacyjny o *n* wejściach i *m* wyjściach

mięć. Dlatego układy sekwencyjne nazywa się także **układami kombinacyjnymi z pamięcią. Układy sekwencyjne** dzieli się na **synchroniczne** i **asynchroniczne**. Niniejszy podręcznik ogranicza się głównie do sekwencyjnych układów synchronicznych. Schemat poglądowy takiego układu (z uwzględnieniem bloku pamięci) przedstawiono na rys. 3.2.



Rys. 3.2. Schemat poglądowy sekwencyjnego układu synchronicznego

Zadaniem zegara jest synchronizacja pracy układu. Rola zegara zostanie omówiona dokładniej w dalszej części podręcznika.

Podstawowymi elementami sekwencyjnych układów cyfrowych są funktory (które umożliwiają budowę układu kombinacyjnego) oraz tzw. **przerzutniki** (służące do budowy pamięci układu). Przerzutniki omówiono w rozdz. 7. podręcznika.

Ponieważ przerzutniki można budować z bramek (funktorów), przeto układ zawierający tylko bramki nie musi być układem kombinacyjnym. Zwykle jednak do budowy układów sekwencyjnych (w szczególności synchronicznych) wykorzystuje się gotowe przerzutniki (scalone).

Pytania i zadania

1. Jaka jest różnica między układem kombinacyjnym a sekwencyjnym?
2. Wymień podstawowe elementy układów kombinacyjnych i sekwencyjnych.

3.2. Podstawowe funktory układów kombinacyjnych

Funktorem (bramką) ***będziemy nazywać podstawowy układ kombinacyjny realizujący funkcję logiczną jednej, dwu lub wielu zmiennych.*** Są to kombinacyjne układy cyfrowe, realizujące elementarne funkcje logiczne: AND, OR, NOT oraz ich niezbyt złożone kombinacje, np.: NAND, NOR, ExOR.

Zmienną logiczną w układach półprzewodnikowych jest sygnał elektryczny (napięcie). Sygnał ten jest dwustanowy. Niski poziom napięcia jest oznaczany literą **L** (ang.: *Low* — niski, dolny). Wysoki poziom napięcia oznaczany jest literą **H** (ang.: *High* — wysoki). Zwykle poziomowi **L** przyporządkowuje się stan logiczny **0**, a poziomowi **H** stan logiczny **1**. Takie przyporządkowanie nosi nazwę **konwencji dodatniej**. W całym podręczniku za obowiązującą przyjęto właśnie tę konwencję. W **konwencji ujemnej** poziom **L** przyjmuje się za **1**, a poziom **H** za **0**.

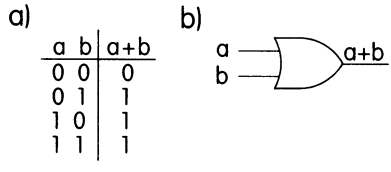
Do opisu działania logicznego bramek stosuje się tablice prawdy. Jest to ujęty w tablicę zbiór wszystkich kombinacji sygnałów wejściowych oraz odpowiadające im sygnały wyjściowe. Oczywiście każdy funktor ma także opisującą go funkcję logiczną wyrażoną językiem algebry Boole'a. Tablice prawdy operują abstrakcyjnymi stanami logicznymi **0** i **1**, lub dwoma poziomami **L** i **H**. Nazwy funktorów pochodzą od nazw operacji logicznych realizowanych przez nie. Częściej są używane (np. w katalogach) nazwy angielskie. W podręczniku będziemy używać nazw angielskich.

- **Bramka OR (LUB)**

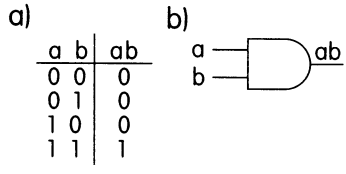
Bramka ta jest układem o dwu lub większej liczbie wejść. Bramka OR realizuje funkcję sumy logicznej zmiennych wejściowych (rys. 3.3).

- **Bramka AND (I)**

Bramka ta jest układem o dwu lub większej liczbie wejść. Realizuje ona funkcję iloczynu logicznego zmiennych wejściowych (rys. 3.4).

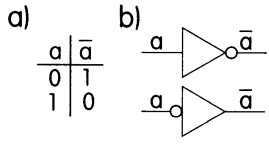


Rys. 3.3. Dwuwejściowa bramka OR (LUB): a) tablica prawdy; b) symbol graficzny



Rys. 3.4. Dwuwejściowa bramka AND (I): a) tablica prawdy; b) symbol graficzny

- **Bramka NOT (NIE)**



Rys. 3.5. Bramka NOT (NIE): a) tablica prawdy; b) symbol graficzny

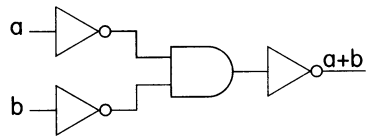
Bramka ta (rys. 3.5) jest układem o jednym wejściu. Realizuje ona funkcję negacji zmiennej wejściowej. Symbol graficzny tej bramki należy traktować jako złożenie symbolu oznaczającego wzmacniacz logiczny oraz kółka reprezentującego realizację inwersji. Kółko może być umieszczone po stronie wejścia lub wyjścia. Wzmacniaczem logicznym jest układ nie zmieniający wartości logicznej sygnału, a jedynie zwiększający obciążalność wyjścia.

Opisane powyżej funktory logiczne realizują trzy podstawowe, zdefiniowane w algebrze Boole'a, operacje logiczne. Oczywisty jest fakt, że każdą funkcję przełączającą można przedstawić za pomocą argumentów stałych **0**, **1** oraz operacji logicznych sumy, iloczynu i negacji. Gdyby bowiem to nie było możliwe, to należałoby wówczas zrezygnować z algebry Boole'a jako narzędzia przydatnego do opisu układów kombinacyjnych. Innymi słowy, odnosząc powyższe stwierdzenie do technicznej realizacji układów logicznych, można powiedzieć, że wystarczy dysponować funktorami AND, OR, NOT, aby zrealizować dowolny układ kombinacyjny.

W rzeczywistości wystarczy dysponować dwoma rodzajami funktorów np. AND, NOT (lub OR, NOT), aby zapewnić realizowalność dowolnego układu. Dowód powyższej tezy jest bardzo prosty. Należy wykazać, że dysponując funktorami AND, NOT jesteśmy w stanie zrealizować operację sumy logicznej (lub, że dysponując funktorami OR, NOT jesteśmy w stanie zrealizować iloczyn logiczny). Na przykład

$$a + b = \overline{\overline{a + b}} = \overline{\bar{a}\,\bar{b}}$$

co graficznie można przedstawić jak na rys. 3.6.



Rys. 3.6. Realizacja funkcji OR przy użyciu bramek AND, NOT

Taki *zbiór funktorów (czy operacji logicznych), który pozwala zrealizować dowolną funkcję logiczną nazywamy* systemem funkcjonalnie pełnym (w skrócie SFP).

Oczywiście zbiór {AND, OR, NOT} jest SFP, ale nie minimalnym. Zbiór {AND, NOT} (podobnie jak zbiór {OR, NOT}) jest SFP minimalnym.

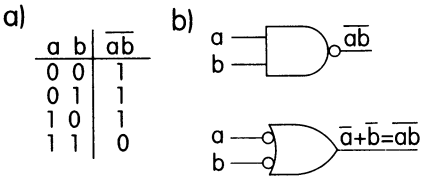
- **Bramka NAND (NIE I)**

Bramka NAND (ang. *Not AND*) jest układem o dwu lub większej liczbie wejść. Realizuje ona funkcję negacji iloczynu zmiennych wejściowych (rys. 3.7).

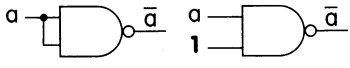
Bramka NAND jest funkcjonalnie pełna, bowiem stosując ją można zrealizować zarówno operację iloczynu logicznego (AND), jak i operację negacji (NOT) — czyli zgodnie z wcześniejszymi wnioskami dowolną funkcję logiczną.

Realizacja negacji przy użyciu bramki NAND (rys. 3.8) wykorzystuje zależność

$$\bar{a} = \overline{a\bar{a}} \qquad \text{lub} \qquad \bar{a} = \overline{a \cdot 1}$$



Rys. 3.7. Dwuwejściowa bramka NAND (NIE I): a) tablica prawdy; b) symbol graficzny



Rys. 3.8. Realizacja negacji przy użyciu bramek NAND

Rys. 3.9. Realizacja iloczynu logicznego przy użyciu bramek NAND

Realizacja iloczynu logicznego przy użyciu bramek NAND (rys. 3.9) wykorzystuje zależność

$$ab = \overline{\overline{ab}} = \overline{\overline{ab}\,\overline{ab}} \qquad \text{lub} \qquad ab = \overline{\overline{ab}} = \overline{\overline{ab} \cdot 1}$$

Ta cecha (że NAND jest SFP) sprawiła, iż funktor ten jest podstawową bramką w kilku klasach scalonych układów cyfrowych (np. TTL, ECL, CMOS).

- **Bramka NOR (NIE LUB)**

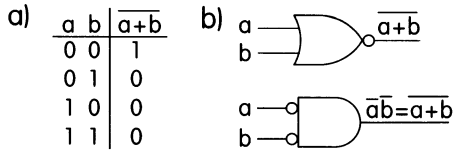
Bramka NOR (ang. ***Not OR***) jest układem o dwu lub większej liczbie wejść (rys. 3.10). Realizuje ona funkcję negacji sumy. Można wykazać, że jest ona funkcjonalnie pełna.

- **Bramka Ex-OR (XOR, ALBO)**

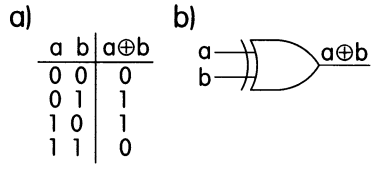
Bramka Ex-OR (rys. 3.11) (ang. ***Exclusive OR*** — LUB z wyłączeniem) realizuje funkcję

$$f(a,b) = a\bar{b} + \bar{a}b = a \oplus b$$

która jest także nazywana **sumą modulo 2.**



Rys. 3.10. Dwuwejściowa bramka NOR (NIE LUB): a) tablica prawdy; b) symbol graficzny



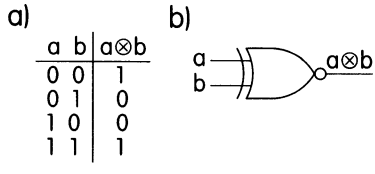
Rys. 3.11. Dwuwejściowa bramka Ex-OR: a) tablica prawdy; b) symbol graficzny

Funktor Ex-OR nie stanowi SFP, ale ma on jednak duże znaczenie praktyczne. Umożliwia on bowiem, w dość szerokiej klasie układów, bardzo oszczędną (liczba elementów i połączeń) relizację układu. Dotyczy to zwłaszcza realizacji: operacji arytmetycznych, konwersji kodów, korekcji błędów i innych.

- **Bramka Ex-NOR (NIE ALBO)**

Bramka Ex-NOR (rys. 3.12) (ang. ***Exclusive — Not OR***) realizuje funkcję

$$f(a,b) = ab + \bar{a}\bar{b} = a \otimes b$$



Rys. 3.12. Dwuwejściowa bramka Ex-NOR: a) tablica prawdy; b) symbol graficzny

Pytania i zadania

1. Zrealizuj przy użyciu bramek OR, NOT iloczyn logiczny **ab**.
2. Wykaż, że funktor NOR jest funkcjonalnie pełny.
3. Jakie operacje logiczne będą realizować funktory AND, OR,NOT, NAND, NOR przystosowane do pracy w konwencji dodatniej, jeżeli zastosujemy je w układzie o konwencji ujemnej?
4. Wykaż, że $a \oplus b = \overline{a \otimes b}$.
5. Wykaż, że $\overline{a} \oplus b = a \otimes b$.
6. Sprawdź, czy $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
7. Oblicz $a \oplus 0$, $a \oplus 1$, $a \oplus a$, $a \oplus \overline{a}$.
8. Zrealizuj funktor NAND, wykorzystując jedynie funktory NOR.
9. Zrealizuj funktor NOR, wykorzystując jedynie funktory NAND.

3.3. Metody opisu układów kombinacyjnych

W celu opisania działania cyfrowego układu kombinacyjnego wykorzystuje się kilka sposobów. Pierwotną postacią opisu jest zwykle **opis słowny**. Bardzo przejrzystą i jednoznaczną formą opisu jest tak zwana **tablica prawdy**. Najbardziej zwięzłą i uproszczoną formą opisu jest **postać kanoniczna**. Niezależnie od postaci opisu układu docelową formą jest taki opis funkcji logicznej, który można bezpośrednio przetworzyć na schemat logiczny układu elektronicznego.

Poniżej zostaną omówione i scharakteryzowane poszczególne sposoby opisu układów przełączających.

- **Opis słowny**

Najczęściej pierwotna informacja na tamat funkcjonowania układu cyfrowego ma charakter opisowy. Jednak język naturalny nie zawsze jest jednoznaczny. Przystąpienie do projektowania układu na podstawie opisu słownego wymaga przeważnie uściśleń, bądź przyjęcia określonej interpretacji, być może nie zawsze zgodnej z intencją autora opisu. Tak więc formułując taki opis, należy dążyć do tego, aby był on możliwie jednoznaczny.

Przykład 3.1

Zaprojektować układ z elementów AND, OR, NOT o trzech wejściach **c, b, a,** wyróżniający sygnałem wyjściowym **y = 1** przypadki, gdy na wejściu pojawi się liczba dwójkowa nieparzysta lub podzielna przez 3. Sygnał **a** odpowiada najmłodszemu bitowi (LSB) słowa wejściowego. W każdej kombinacji wejściowej co najmniej jeden z sygnałów wejściowych (**cba**) jest różny od zera. ■

- **Tablica prawdy (tablica wartości funkcji, tablica wierności)**

Ten sposób opisu był już wykorzystywany do opisu działania podstawowych funktorów logicznych. W wierszach tablicy wypisuje się wszystkie kombinacje zero-jedynkowe zmiennych niezależnych. Wierszy takich jest 2^n, gdzie n jest liczbą zmiennych. Ostatnia kolumna jest przeznaczona do wpisania wartości funkcji dla poszczególnych słów wejściowych. Zwykle wszystkie możliwe kombinacje zmiennych wypisujemy tak, aby stanowiły one kolejne liczby dziesiętne zapisane w systemie dwójkowym.

Przykład 3.2

Zapisać tablicę prawdy dla układu z przykładu 3.1.

Tablica 3.1

	c	b	a	y
0	**0**	**0**	**0**	—
1	**0**	**0**	**1**	**1**
2	**0**	**1**	**0**	**0**
3	**0**	**1**	**1**	**1**
4	**1**	**0**	**0**	**0**
5	**1**	**0**	**1**	**1**
6	**1**	**1**	**0**	**1**
7	**1**	**1**	**1**	**1**

Uwaga. W praktyce bardzo często mamy do czynienia z sytuacją, że fizyczne warunki działania urządzenia nie dopuszczają wystąpienia pewnych kombinacji zmiennych. (W naszym przykładzie stwierdzenie zawarte w ostatnim zdaniu opisu działania układu eliminuje możliwość jednoczesnego wystąpienia zer na wszystkich trzech wejściach układu). Jeśli dana kombinacja zero-jedynkowa nigdy się na wejściu układu nie pojawia, to dla takiej kombinacji wartość funkcji może wynosić równie dobrze **0** jak i **1**. I taką, jedną z tych dwóch wartości, można wpisać w powyższą tablicę. Możliwość przyjęcia dowolnej wartości funkcji dla pewnych kombinacji jest bardzo przydatna w procesie **minimalizacji**. W tablicach w miejsca gdzie wartość funkcji może być dowolna, będziemy wpisywać umowny symbol „—"(lub „x"). ■

- **Postać kanoniczna**

Postacią kanoniczną w matematyce nazywa się pewien umowny sposób opisu obiektów matematycznych. Postać taka ułatwia porównania opisywanych obiektów lub uwypukla pewne ich charakterystyczne cechy. Postać kanoniczna funkcji logicznej zawdzięcza swą uprzywilejowaną rolę prostocie i zwięzłości zapisu. Aby uzyskać postać kanoniczną dowolnej funkcji $\mathbf{f}(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n)$ należy skorzystać z twierdzenia o rozkładzie.

Twierdzenie o rozkładzie

Dowolną funkcję przełączającą można rozłożyć na dwa składniki:

$$\mathbf{f(x_1,x_2,\ldots,x_n) = x_1 f(1,x_2,\ldots,x_n) + \bar{x}_1 f(0,x_2,\ldots,x_n)} \tag{3.2}$$

lub dwa czynniki

$$\mathbf{f(x_1,x_2,\ldots,x_n)=[x_1+f(0,x_2,\ldots,x_n)][\bar{x}_1+f(1,x_2,\ldots,x_n)]} \tag{3.3}$$

Dowód można przeprowadzić poprzez porównanie lewej i prawej strony tezy dla $\mathbf{x_1 = 0}$ i dla $\mathbf{x_1 = 1}$.

Zastosujmy pierwszą część twierdzenia (o rozkładzie na składniki) trzykrotnie, względem kolejnych zmiennych, wobec funkcji **f(c,b,a)**:

$$\begin{aligned}&\mathbf{f(c,b,a) = cf(1,b,a) + \bar{c}f(0,b,a) = c[bf(1,1,a) + \bar{b}f(1,0,a)] +}\\&\mathbf{+ \bar{c}[bf(0,1,a) + \bar{b}f(0,0,a)] = cbf(1,1,a) + c\bar{b}f(1,0,a) + \bar{c}bf(0,1,a) +}\\&\mathbf{+ \bar{c}\bar{b}f(0,0,a) = cbaf(1,1,1) + cb\bar{a}f(1,1,0) + c\bar{b}af(1,0,1) +}\\&\mathbf{+ c\bar{b}\bar{a}f(1,0,0) + \bar{c}baf(0,1,1) + \bar{c}b\bar{a}f(0,1,0) + \bar{c}\bar{b}af(0,0,1) + \bar{c}\bar{b}\bar{a}f(0,0,0)}\end{aligned} \tag{3.4}$$

W wyniku trzykrotnego zastosowania twierdzenia uzyskaliśmy postać (3.4) zawierającą osiem składników. Zwróćmy uwagę, że każdy składnik zawiera element złożony z iloczynu wszystkich argumentów funkcji (**cba**).

Iloczyn wszystkich argumentów funkcji będziemy nazywać* iloczynem pełnym *i oznaczać dużą literą K z indeksem i, tzn. $\boldsymbol{K_i}$.

Indeks ***jest liczbą dwójkową (lub jej odpowiednikiem dziesiętnym) utworzoną poprzez przyporządkowanie zmiennej*** $\mathbf{x_i}$ **— 1,** ***a zmiennej*** $\bar{\mathbf{x}}_\mathbf{i}$ **— 0.**

Na przykład:

Pełny iloczyn	**c b a**	**c $\bar{\mathbf{b}}$ a**	**$\bar{\mathbf{c}}$ $\bar{\mathbf{b}}$ $\bar{\mathbf{a}}$**
	↓ ↓ ↓	↓ ↓ ↓	↓ ↓ ↓
Indeks dwójkowy	**1 1 1**	**1 0 1**	**0 0 0**
	⇓	⇓	⇓
Indeks dziesiętny	7	5	0
	↓	↓	↓
Zapis symboliczny	K_7	K_5	K_0

Potraktujmy ponadto kombinację wejściową jako liczbę binarną i następnie przejdźmy na jej odpowiednik dziesiętny. Pozwoli to nam na wprowadzenie następujących oznaczeń:

$$\begin{aligned}&\mathbf{f(0,0,0) = f(000) = f}(0)\\&\mathbf{f(0,0,1) = f(001) = f}(1)\\&\mathbf{f(0,1,0) = f(010) = f}(2)\\&\ldots\ldots\ldots\ldots\\&\mathbf{f(1,1,1) = f(111) = f}(7)\end{aligned}$$

Korzystając z pojęcia pełnego iloczynu oraz powyższych oznaczeń możemy zależność (3.4) zapisać w sposób bardziej zwięzły

$$\mathbf{f(c,b,a) = K_0 f(0) + K_1 f(1) + \ldots + K_7 f(7)}$$

Uogólnijmy te rozważania na przypadek funkcji n zmiennych. Łatwo zauważyć, że funkcja n zmiennych będzie miała 2^n składników, gdyż rozkład będzie wykonywany n razy, a każdy kolejny rozkład podwaja liczbę składników. Zatem

$$\mathbf{f(x_1,x_2,\ldots,x_n)} = \sum_{i=0}^{2^{n-1}} \mathbf{K_i f}(i) \tag{3.5}$$

Znak Σ we wzorze (3.5) oznacza sumę logiczną, a postać (3.5) będziemy nazywać **kanoniczną postacią sumy** (w skrócie **KPS**).

Dla przykładu z tablicy 3.1 otrzymamy więc pełne iloczyny(tabl. 3.2):

Tablica 3.2

	c b a	y	f(i)	Pełne iloczyny
0	0 0 0	—	f(0)	$\bar{c}\,\bar{b}\,\bar{a} = K_0$
1	0 0 1	1	f(1)	$\bar{c}\,\bar{b}\,a = K_1$
2	0 1 0	0	f(2)	$\bar{c}\,b\,\bar{a} = K_2$
3	0 1 1	1	f(3)	$\bar{c}\,b\,a = K_3$
4	1 0 0	0	f(4)	$c\,\bar{b}\,\bar{a} = K_4$
5	1 0 1	1	f(5)	$c\,\bar{b}\,a = K_5$
6	1 1 0	1	f(6)	$c\,b\,\bar{a} = K_6$
7	1 1 1	1	f(7)	$c\,b\,a = K_7$

a zatem funkcja

$$\begin{aligned}\mathbf{f(c,b,a)} &= \mathbf{K_0 f(0) + K_1 f(1) + \ldots + K_7 f(7)} = \\ &= \mathbf{(K_0) + K_1 1 + K_2 0 + K_3 1 + K_4 0 + K_5 1 + K_6 1 + K_7 1} = \\ &= \mathbf{K_1 + K_3 + K_5 + K_6 + K_7 + (K_0)}\end{aligned}$$

lub w zapisie skróconym

$$\mathbf{f(c,b,a)} = \Sigma[1,3,5,6,7,(0)] \tag{3.6}$$

Postać (3.6) jest postacią kanoniczną zapisaną w sposób uproszczony, z pominięciem symboli **K** pełnych iloczynów i zastąpieniem sumowania jednym znakiem sumy umieszczonym przed nawiasem zawierającym indeksy pełnych iloczynów. Jak widać z tablicy 3.2, indeksy te odpowiadają dokładnie tym kombinacjom wejściowym, dla których funkcja ma wartość **1** (pozostałe składniki znikają, ponieważ są mnożone przez **0**). Z tego powodu liczby te są nazywane **jedynkami funkcji**. W zapisie tym uwzględnia się przypadki, dla których wartość funkcji jest nieokreślona, poprzez wzięcie ich w nawias.

Popatrzmy ponownie na tablicę prawdy funkcji z przykładu 3.1 (tabl. 3.1), uzupełniając ją o niektóre pełne iloczyny (tabl. 3.3). Zauważmy, że każdy z wypi-

sanych iloczynów (pełnych) ma wartość **1** tylko i wyłącznie dla takiej kombinacji wejściowej, która odpowiada wierszowi, w którym ten iloczyn został zapisany. Innymi słowy taki pełny iloczyn „dostarcza" wyłącznie jednej jedynki funkcji.

Tablica 3.3

	c b a	y	Pełne iloczyny	Pełne sumy
0	0 0 0	—	$\bar{c}\,\bar{b}\,\bar{a} = K_0$	$c + b + a = D_0$
1	0 0 1	1	$\bar{c}\,\bar{b}\,a = K_1$	
2	0 1 0	0		$c + \bar{b} + a = D_2$
3	0 1 1	1	$\bar{c}\,b\,a = K_3$	
4	1 0 0	0		$\bar{c} + b + a = D_4$
5	1 0 1	1	$c\,\bar{b}\,a = K_5$	
6	1 1 0	1	$c\,b\,\bar{a} = K_6$	
7	1 1 1	1	$c\,b\,a = K_7$	

Aby funkcja miała wartość **1** także dla kombinacji z wierszy 1, 3, 5, 6 i 7 należy zsumować odpowiadające im pełne iloczyny, a zatem

$$\mathbf{f(c, b, a)} = (\bar{c}\bar{b}\bar{a}) + \bar{c}\bar{b}a + \bar{c}ba + c\bar{b}a + cb\bar{a} + cba =$$
$$= (K_0) + K_1 + K_3 + K_5 + K_6 + K_7 = \Sigma[1,3,5,6,7,(0)] \qquad (3.7)$$

Tak więc, w sposób bardziej poglądowy, uzyskaliśmy wynik identyczny jak w (3.6).

W tablicy 3.3 pojawiły się sumy zawierające wszystkie argumenty funkcji **f(c, b, a)**, które przez analogię do pełnych iloczynów, nazwiemy **pełnymi sumami**. Zwróćmy jedynie uwagę, że indeksy tworzy się teraz odwrotnie, to znaczy zmiennej x_i przyporządkowujemy **0**, a zmiennej $\bar{x}_i$ — **1**. Pełną sumę oznaczamy dużą literą **D**. Na przykład:

Pełna suma	c+b+a	c+$\bar{b}$+a	$\bar{c}$+b+a
	↓ ↓ ↓	↓ ↓ ↓	↓ ↓ ↓
Indeks dwójkowy	0 0 0	0 1 0	1 0 0
	⇓	⇓	⇓
Indeks dziesiętny	0	2	4
	↓	↓	↓
Zapis symboliczny	D_0	D_2	D_4

Łatwo sprawdzić, że pełna suma ma wartość **0** tylko dla kombinacji wejściowej jak w wierszu, w którym została zapisana. We wszystkich pozostałych przypadkach jej wartość jest równa **1**. Zatem, jeżeli chcemy, aby funkcja przyjęła wartość zero dla kombinacji z wierszy 2 i 4 należy je pomnożyć przez siebie, czyli

$$\mathbf{f(c,b,a)} = [(c + b + a)](c + \bar{b} + a)(\bar{c} + b + a) = (D_0)D_2D_4 = \Pi[2, 4, (0)] \quad (3.8)$$

Uzyskany w ten sposób zapis funkcji będziemy nazywać **kanoniczną postacią iloczynu** (w skrócie **KPI**), a liczby występujące w tym zapisie — zerami funk-

cji. Oczywiście, kanoniczną postać iloczynu możemy uzyskać w sposób analogiczny jak KPS wykorzystując w tym celu drugą część twierdzenia o rozkładzie, a mianowicie rozkład na czynniki. Uzyskamy wtedy (po uogólnieniu) zapis

$$\mathbf{f}(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n) = \prod_{i=0}^{2^{n-1}} [\mathbf{D}_i + \mathbf{f}(i)] \tag{3.9}$$

Z zależności (3.9) widać, że z postaci kanonicznej iloczynu znikają (po podstawieniu wartości) te czynniki, dla których $\mathbf{f}(i) = \mathbf{1}$, a zatem zostają same zera funkcji.

W podsumowaniu należy zwrócić uwagę na następujące spostrzeżenia:

1. Postać kanoniczna jest bardzo dogodna w zapisie oraz łatwo ją uzyskać na podstawie tablicy prawdy. Odwrotny zabieg, to znaczy przejście z zapisu kanonicznego na tablicę prawdy, jest także bardzo prosty.
2. Jeżeli dysponujemy zapisem funkcji w KPS, to łatwo możemy przejść na KPI lub na odwrót (należy tylko pamiętać, że kombinacje wejściowe, dla których funkcja ma wartość nieokreśloną, występują w obydwóch postaciach).
3. Każda funkcja ma jedną KPS i jedną KPI, co można wykorzystać przy porównywaniu różnych form.

Spostrzeżenie 3. wykorzystajmy do rozwiązania przykładu 2.2 (z podrozdziału 2.1). Polecenie w tym zadaniu było następujące:

udowodnić równość

$$\mathbf{ab + \bar{a}\bar{b} + bc = ab + \bar{a}\bar{b} + \bar{a}c}$$

Zadanie wykonamy sprowadzając każdą ze stron równości do postaci kanonicznej sumy (KPS).

$$\mathbf{L = ab + \bar{a}\bar{b} + bc = ab(c + \bar{c}) + \bar{a}\bar{b}(c + \bar{c}) + (a + \bar{a})bc = abc + ab\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}\bar{c} + }$$
$$\mathbf{+ abc + \bar{a}bc = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + abc + abc = K_{000} + K_{001} + K_{011} + K_{110} + }$$
$$\mathbf{+ K_{111} = K_0 + K_1 + K_3 + K_6 + K_7 = \Sigma(0, 1, 3, 6, 7)}$$

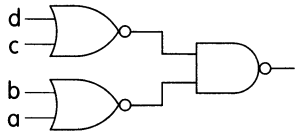
$$\mathbf{P = ab + \bar{a}\bar{b} + \bar{a}c = ab(c + \bar{c}) + \bar{a}\bar{b}(c + \bar{c}) + \bar{a}(b + \bar{b})c = abc + ab\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}\bar{c} + }$$
$$\mathbf{+ \bar{a}bc + \bar{a}\bar{b}c = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}bc + ab\bar{c} + abc = K_{000} + K_{001} + K_{011} + K_{110} + }$$
$$\mathbf{+ K_{111} = K_0 + K_1 + K_3 + K_6 + K_7 = \Sigma(0, 1, 3, 6, 7)}$$

Z porównania wynika, że $\mathbf{L = P}$. ■

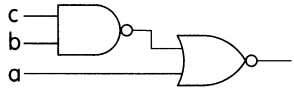
Pytania i zadania

1. Podaj i udowodnij twierdzenie o rozkładzie na składniki.
2. Podaj i udowodnij twierdzenie o rozkładzie na czynniki.
3. Zdefiniuj pojęcie pełnego iloczynu oraz podaj zasady tworzenia jego indeksu.
4. Zdefiniuj pojęcie pełnej sumy oraz podaj zasady tworzenia jej indeksu.

5. Zastosuj trzykrotnie, względem funkcji **f(c,b,a)**, twierdzenie o rozkładzie na czynniki.
6. Zapisz funkcję **f(d,c,b,a)** = Σ[1, 3, 4, 5, 9, 11, 12, (7, 13)] w KPI i zapisz jej tablicę prawdy.
7. Zapisz funkcję **f(d,c,b,a)** = Π[2, 4, 6, 7, 9, 12, 14, (0, 10)] w KPS i zapisz jej tablicę prawdy.
8. Zapisz funkcję z zadania 6. w KPS, ale jako funkcję **f(a,b,c,d)**, a nie **f(d,c,b,a)**.
9. Zapisz funkcję z zadania 7. w KPI, ale jako funkcję **f(a,b,c,d)**, a nie **f(d,c,b,a)**.
10. Znajdź KPS i KPI funkcji **f(d,c,b,a) = cba + db**.

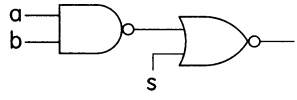


Rys. 3.13. Schemat układu do zadania 12.

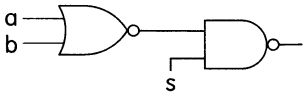


Rys. 3.14. Schemat układu do zadania 13.

11. Znajdź KPS i KPI funkcji **f(d,c,b,a) =(b + a)(d + c + b)**.
12. Zapisz tablicę prawdy oraz podaj KPS i KPI dla układu z rys. 3.13.



Rys. 3.15. Schemat układu do zadania 14.

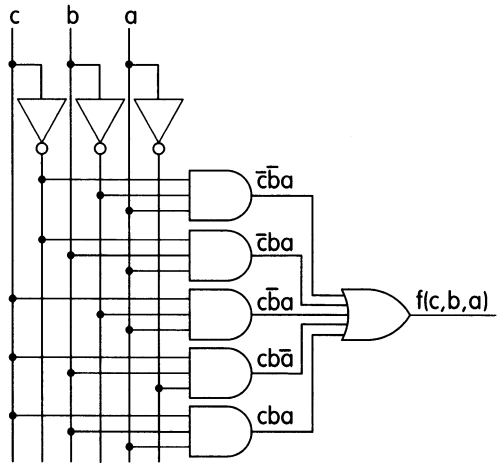


Rys. 3.16. Schemat układu do zadania 15.

13. Zapisz tablicę prawdy oraz podaj KPS i KPI dla układu z rys. 3.14.
14. Określ stan wyjścia układu z rys. 3.15 dla wejścia sterującego **S**: a) **S = 0**; b) S = **1**.
15. Określ stan wyjścia układu z rys. 3.16 dla wejścia sterującego **S**: a) **S = 0**; b) S = **1**.
16. Sprawdzić, czy następujące równości są prawdziwe (zadanie zrealizować, sprowadzając obie strony równania do postaci kanonicznej sumy — KPS):
    a) $(a + \bar{b} + a\bar{b})(ab + \bar{a}c + bc) = ab + \bar{a}c\bar{b}$,
    b) $(ab + c + d)(\bar{c} + d)(\bar{c} + d + e) = ab\bar{c} + d$,
    c) $(\bar{a} + b + \bar{c}) = \bar{a}(\bar{b} + \bar{c}) + bc + a\bar{c}$,
    d) $ab + ac + bc = (a + b)(a + c)(b + c)$,
    e) $b\bar{c} + \bar{a}c + \bar{a}b = b\bar{c} + \bar{a}c$,
    f) $ab + \bar{a}b\bar{c} + bc = b$,
    g) $a\bar{b} + ac + bc = a\bar{b} + ac$,
    h) $(a + b)(\bar{a} + c)(b + c) = (a + b)(\bar{a} + c)$,
    i) $ab + \bar{a}b + bcd = ab + \bar{a}\bar{b} + \bar{a}cd$.
17. Wykazać, że równości z zad.16 są prawdziwe (zadanie zrealizować sprowadzając obie strony równania do postaci kanonicznej iloczynu — KPI).
18. Co nazywamy **1**, a co **0** funkcji?

3.4. Realizacja układów kombinacyjnych przy użyciu bramek

Zapis funkcji w postaci kanonicznej umożliwia jej realizację z bramek logicznych bezpośrednio na podstawie tego zapisu (chodzi o zapis kanoniczny, w którym pełne sumy lub pełne iloczyny są zapisane w sposób jawny). Jeśli układ ma być realizowany z bramek AND, OR, NOT, to do negacji zmiennych używamy elementów NOT w celu uzyskania pełnego iloczynu (lub iloczynu pełnych sum w KPI) bramek AND, a do uzyskania pełnej sumy (lub sumy pełnych iloczynów w KPS) — funktorów OR.



Rys. 3.17. Realizacja funkcji w postaci kanonicznej sumy z przykładu 3.1

Realizacja układu z przykładu 3.1 przy wykorzystaniu kanonicznej postaci sumy

$$\mathbf{f(c,b,a) = \bar{c}\bar{b}a + \bar{c}ba + c\bar{b}a + cb\bar{a} + cba} \tag{3.10}$$

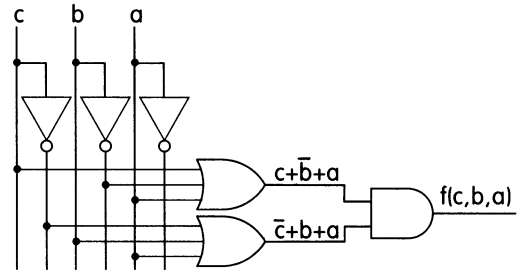
będzie taka jak na rys. 3.17.

Możemy zrealizować tę samą funkcję na podstawie postaci kanonicznej iloczynu (rys. 3.18)

$$\mathbf{f(c,b,a) = \Pi(2,4) = (c + \bar{b} + a)(\bar{c} + b + a)} \tag{3.11}$$

W zapisie (3.10) pominięto pełny iloczyn $\mathbf{K_0}$, ponieważ może on być **1** funkcji, ale nie musi. Podobnie uczyniliśmy pomijając pełną sumę $\mathbf{D_0}$ w formule (3.11), ponieważ może być ona **0** funkcji, ale nie musi. Wynika to z zapisu tej funkcji — w tablicy prawdy wartość funkcji to „—", co interpretujemy jako wartość dowolną. Czy obie realizacje są sobie równoważne? Otóż każda z nich wygeneruje na swym wyjściu stan **L** i stan **H** dokładnie zgodnie z zapisem kanonicznym (czy tablicą prawdy). Stan wyjścia będzie różny w obu układach jedynie przy stanie wejść **cba = 000**. Pierwszy z nich będzie miał na wyjściu stan **L**, a drugi stan **H**. W eksploatacji tych układów taka sprzeczność nigdy nie wystąpi, bo zgodnie z opisem działania układu stan wejść **cba = 000** nigdy się nie pojawi.

Porównajmy złożoność obu realizacji pod względem liczby bramek i liczby połączeń. Jak widać, druga wersja (rys. 3.18) jest prostsza, a zatem tańsza, czyli generalnie lepsza. Pomijając szczegółowe problemy optymalizacji układu można



Rys. 3.18. Realizacja funkcji w postaci kanonicznej iloczynu z przykładu 3.1

stwierdzić, że na ogół ***układ o mniejszej liczbie elementów jest tańszy i bardziej niezawodny. Z dwóch układów o takiej samej liczbie elementów logicznych lepszy jest ten, w którym występuje mniej połączeń*** (czyli mniej wejść mają w sumie wszystkie jego elementy). Dlatego bardzo ważnym etapem syntezy układu logicznego jest poszukiwanie takiej ***postaci*** funkcji opisującej jego działanie, w której występuje minimalna liczba liter (tzn. zmiennych lub ich negacji). Proces poszukiwania takiej postaci minimalnej będziemy nazywać **minimalizacją formuły funkcji**. Potocznie mówi się często „minimalizacja funkcji", ale należy sobie zdawać sprawę z tego, że nie funkcja podlega minimalizacji (bo przed i po minimalizacji jest to ta sama funkcja), lecz jedynie formuła opisująca ją.

Upraszczanie zapisu funkcji przeprowadza się przy zastosowaniu tzw. **reguł sklejania.**

$$\begin{aligned} &\mathbf{A}x + \mathbf{A}\bar{x} = \mathbf{A} \\ &(\mathbf{A} + x)(\mathbf{A} + \bar{x}) = \mathbf{A} \end{aligned} \tag{3.12}$$

gdzie: **x** — zmienna, **A** — zmienna lub funkcja logiczna.

Formułę funkcji danej w postaci kanonicznej sumy (3.10) można zminimalizować w następujący sposób:

$$\begin{aligned} \mathbf{f(c,b,a)} = \Sigma[1, 3, 5, 6, 7] \; &= \mathbf{\bar{c}\bar{b}a + \bar{c}ba + c\bar{b}a + cba + cb\bar{a} + cba} = \\ &= \mathbf{\bar{c}a(\bar{b} + b) + ca(\bar{b} + b) + cb(\bar{a} + a)} = \\ &= \mathbf{\bar{c}a + ca + cb = cb + a(\bar{c} + c) = cb + a} \end{aligned} \tag{3.13}$$

Aby utworzyć pary (do sklejania) wpisano w (3.13) dwukrotnie iloczyn **cba.** Jest to dozwolone, gdyż w algebrze Boole'a **A + A = A**, więc każdy składnik można powtórzyć (nawet wielokrotnie), bez zmiany wartości funkcji. Realizacja takiego układu będzie bardzo prosta (rys. 3.19).

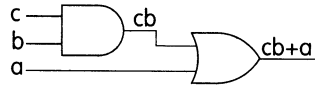
Przeprowadzając sklejanie dla postaci kanonicznej iloczynu (dla tego samego przykładu), otrzymamy

$$\mathbf{f(c,b,a)} = \Pi(2, 4) = \mathbf{(c + \bar{b} + a)(\bar{c} + b + a)}$$

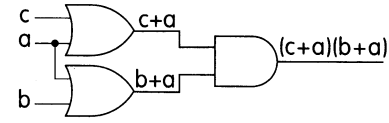
Jak widać nie można wykonać żadnego sklejenia. Uwzględnijmy w zapisie tej funkcji także tę kombinację wejściową, dla której wartość funkcji jest dowolna. Otrzymamy:

$$\mathbf{f(c,b,a)} = \Pi[2, 4, (0)] = (\mathbf{c} + \bar{\mathbf{b}} + \mathbf{a})(\bar{\mathbf{c}} + \mathbf{b} + \mathbf{a})(\mathbf{c} + \mathbf{b} + \mathbf{a})(\mathbf{c} + \mathbf{b} + \mathbf{a}) = (\mathbf{c} + \mathbf{a})(\mathbf{b} + \mathbf{a}) \tag{3.14}$$

Dopisanie więc jeszcze jednej pełnej sumy (dwukrotnie, ale $\mathbf{A} \cdot \mathbf{A} = \mathbf{A}$) umożliwiło sklejenia i w konsekwencji uproszczenie (minimalizację formuły) funkcji. Jej realizacja będzie teraz taka, jak na rys. 3.20.



Rys. 3.19. Realizacja układu z przykładu 3.1 na podstawie alternatywnej postaci minimalnej (3.13)



Rys. 3.20. Realizacja układu z przykładu 3.1 na podstawie koniunkcyjnej postaci minimalnej (3.14)

Różnica między realizacją postaci kanonicznej i postaci minimalnej jest więc bardzo duża. Porównaj schematy ideowe z rysunków 3.17 i 3.19 oraz 3.18 i 3.20.

W wyniku sklejania uzyskuje się wyrażenia, które już nie są postaciami kanonicznymi, ale zachowują postać sumy iloczynów (przy minimalizacji KPS, czyli sumy jedynek funkcji) oraz iloczynu sum (przy minimalizacji KPI, czyli iloczynu zer funkcji). Wyrażenia tego typu przyjęto nazywać odpowiednio: **postacią normalną sumy** (**alternatywną postacią normalną — APN**) i **postacią normalną iloczynu** (**koniunkcyjną postacią normalną — KPN**).

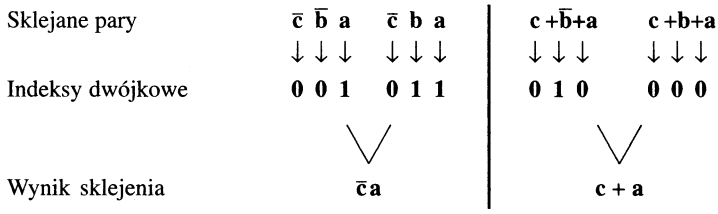
W przypadku złożonych funkcji wielu zmiennych wyszukiwanie wyrażeń podlegających sklejaniu oraz ich sklejanie w sposób zaprezentowany powyżej (tzn. metodą przekształceń algebraicznych) staje się bardzo uciążliwe, zwłaszcza, że dla otrzymania postaci minimalnej trzeba dokonać wszystkich możliwych sklejeń.

Istnieją metody upraszczające procedurę minimalizacji. Do najbardziej rozpowszechnionych (przy projektowaniu bez użycia komputera) należą:

1) **Metoda tablic Karnaugha** (**metoda graficzna**). Ma zastosowanie do minimalizacji funkcji maksymalnie 6 zmiennych. Jest prosta, szybka i łatwa w stosowaniu, szczególnie gdy liczba zmiennych nie przekracza 4.
2) **Metoda Quine'a-Mc Cluskeya.** Może być stosowana dla dowolnej liczby zmiennych. Łatwa do algorytmizacji i realizacji komputerowej.
3) **Metoda minimalizacji funkcji silnie nieokreślonych.** Ma zastosowanie do minimalizacji wieloargumentowych funkcji silnie nieokreślonych. **Funkcja silnie nieokreślona** ***to taka funkcja, której zbiór jedynek i zer jest w sumie nieliczny w stosunku do liczby wszystkich kombinacji wejściowych.***

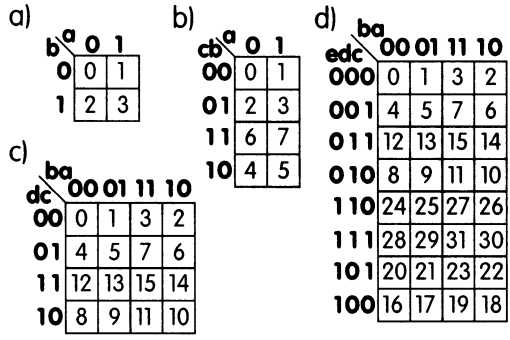
W podręczniku zostanie omówiona i będzie stosowana jedynie **metoda graficzna.**

Zauważmy, że indeksy (dwójkowe) sklejanych pełnych iloczynów lub sklejanych pełnych sum różnią się wyłącznie na jednej pozycji. Na przykład:



Kodem dwójkowym (binarnym), którego kolejne słowa różnią się tylko na jednej pozycji, jest omówiony wcześniej (podrozdział 1.3) kod Graya. Skonstruowanie tablicy, której kolejne wiersze i kolumny będą opisane w kodzie Graya, zapewni nam sąsiadowanie ze sobą tych jedynek (zer) funkcji, które podlegać będą sklejaniu. Tablica prawdy o takiej konstrukcji nazywa się **tablicą Karnaugha.**

Przykłady tablic Karnaugha przedstawiono na rys. 3.21. Każde pole tablicy odpowiada jednej kombinacji wartości zmiennych wejściowych (tak jak jeden wiersz tablicy prawdy). Dlatego postacią zapisu funkcji dogodną do utworzenia odpowiadającej jej tablicy Karnaugha jest tablica wartości funkcji. Bardzo często



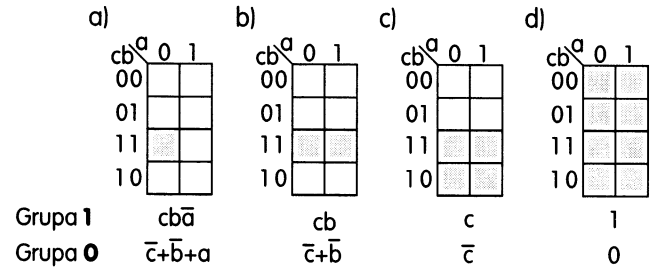
Rys. 3.21. Tablice Karnaugha: a) dwóch zmiennych; b) trzech zmiennych; c) czterech zmiennych; d) pięciu zmiennych

funkcja jest zadana w postaci kanonicznej (indeksy dziesiętne jedynek lub zer funkcji). Zapisanie tak przygotowanych funkcji w tablicy Karnaugha ułatwiają wpisane w poszczególne pola liczby dziesiętne. ***Należy pamiętać, aby kolejność zmiennych występujących w zapisie funkcji była identyczna z kolejnością zmiennych opisujących współrzędne tablicy Karnaugha.*** W przeciwnym razie „ściągawki" z rys. 3.21 będą miały inną postać.

Proces minimalizacji za pomocą tablicy Karnaugha składa się z trzech etapów. Etap pierwszy polega na przygotowaniu tablicy dla danej liczby zmiennych i wpisaniu w jej pola wartości funkcji. Następnie należy narysować obwiednie (połączyć w grupy — skleić) możliwie największych obszarów, które obejmują wyłącznie jedynki (dla postaci alternatywnej), albo wyłącznie zera (dla postaci koniunkcyjnej) sąsiadujące ze sobą.

Jeżeli w dwóch sąsiednich polach wypełnionej tablicy Karanugha znajdują się jednakowe symbole (**1** lub **0**), to odpowiadające tym jedynkom (zerom) pełne iloczyny (pełne sumy) można skleić — co odpowiada usunięciu litery, która

w ramach sklejanej grupy zmienia swą wartość. Gdy zakreślone pola zawierają jedynki, wówczas zamiast odpowiadającego im wyrażenia $\mathbf{Ax + A\bar{x}}$ można przyjąć **A**, natomiast gdy zawierają zera, wówczas zamiast $\mathbf{(B+x)(B+\bar{x})}$ można przyjąć **B**. Wzięcie grupy jedynek (lub zer) złożonej z czterech pól elementarnych usuwa kolejną literę z jej opisu. W stosunku do pełnego iloczynu (czy pełnej sumy) opis takiej „czwórki" będzie zawierał o dwie litery mniej. Generalnie można stwierdzić, że ***każde zwiększenie zakreślanej grupy zmniejsza opis tej grupy o jedną literę.*** Powyższe stwierdzenie zostało zilustrowane na rys. 3.22.



Rys. 3.22. Wielkość grupy i jej opis bulowski

Zakreślania grup należy dokonywać zgodnie z następującymi zasadami:

1) Liczba pól elementarnych łączonych ze sobą musi być potęgą liczby 2 (1, 2, 4, 8,...,2^n).
2) Łączone pola muszą być polami sąsiadującymi ze sobą, tzn. oddzielonymi od siebie linią pionową lub poziomą, albo krawędzią tablicy.
3) Połączone pola muszą mieć kształt symetryczny względem swych osi (kwadraty lub prostokąty).
4) Dla tablic 5 zmiennych obowiązuje jeszcze następująca zasada: jeżeli zakreślane pola znajdują się w obu połówkach tablicy, to w wyniku złożenia tej tablicy względem osi dzielącej ją na dwie symetryczne części zakreślony obszar powinien się dwukrotnie zmniejszyć i spełniać zasadę określoną w p. 3.

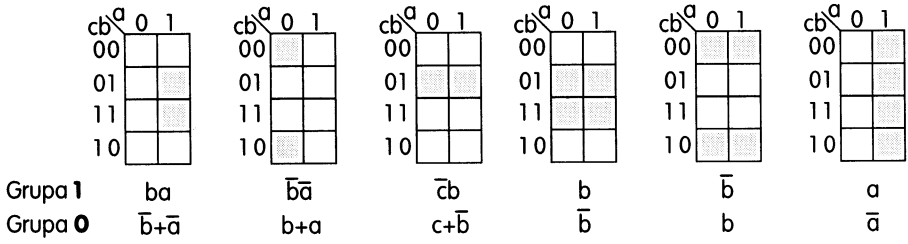
Jeśli w tablicy występują znaki „—" (tzn. funkcja jest nie w pełni określona), to pola elementarne zawierające takie znaki można łączyć z jedynkami albo zerami. Takie dołączenie ma sens wówczas, gdy pozwala nam zakreślić większą grupę zamiast mniejszej.

Na rysunkach 3.23÷3.25 przedstawiono przykłady sklejeń w tablicach Karnaugha, przy czym: jeżeli pod tablicą umieszczono wyrażenie opisujące zakreśloną grupę, to na pierwszym miejscu dotyczy ono jedynek, a na drugim — zer funkcji.

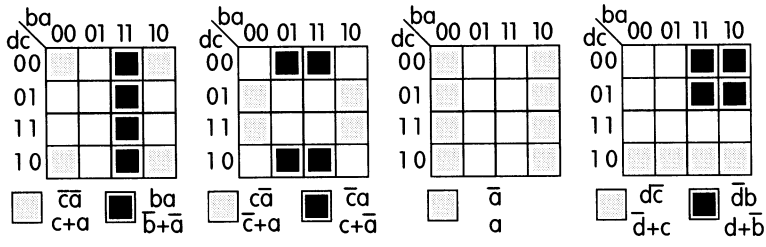
Należy pamiętać, że przeciwległe krawędzie tablicy można uważać za jedną linię. Każde pole (elementarne) w tablicy trzech zmiennych ma trzech „sąsiadów", w tablicy czterech zmiennych — czterech, pięciu zmiennych — pięciu, itd. W tablicy pięciu zmiennych, piątym „sąsiadem" jest pole leżące symetrycznie względem poziomej osi symetrii dzielącej tablicę na dwie równe części.

Formułę łączenia w grupy można więc wyrazić także w następujący sposób:

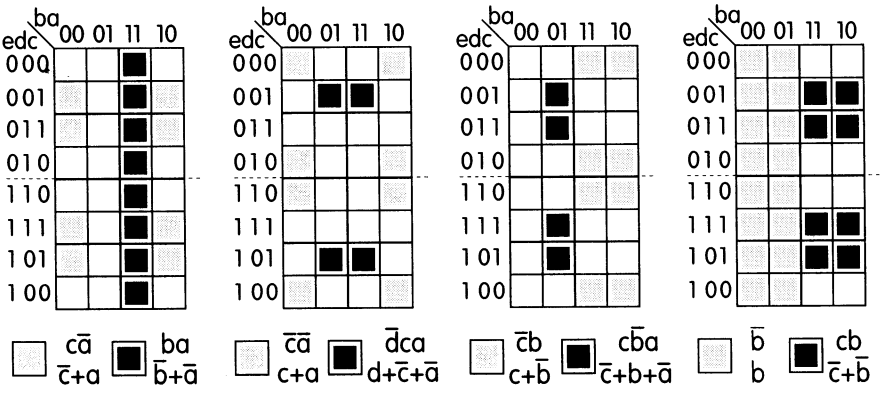
— w grupie obejmującej dwa pola elementarne, każde pole musi mieć jednego sąsiada (jedno sklejanie);
— w grupie obejmującej cztery pola elementarne, każde pole musi mieć dwóch sąsiadów (dwa sklejania);
— w grupie obejmującej osiem pól elementarnych, każde pole musi mieć trzech sąsiadów, itd.



Rys. 3.23. Przykłady sklejeń w tablicach trzech zmiennych



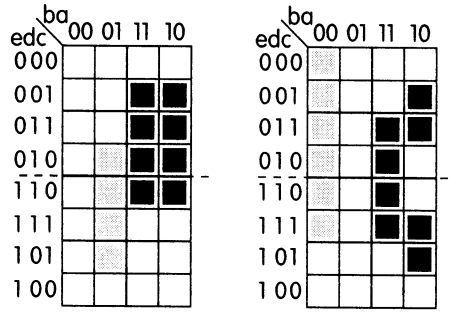
Rys. 3.24. Przykłady sklejeń w tablicach czterech zmiennych



Rys. 3.25. Przykłady sklejeń w tablicach pięciu zmiennych

Pokazane na rysunkach 3.23÷3.25 przykłady sklejania umożliwiają sformułowanie ogólnego wniosku: grupa dwupolowa zmienia dwa człony w postaci kanonicznej — jeden usuwa, a drugi zmniejsza o jedną literę; grupa czteropolowa zmienia cztery człony postaci kanonicznej — trzy usuwa, a czwarty zmniejsza o dwie litery itd. Przykłady, by nie zaciemniać rysunku, pokazują tylko grupy rozłączne, ale nie jest to konieczne, bowiem grupy mogą mieć pola wspólne.

Na rysunku 3.26 pokazano kilka przykładów sklejeń, które nie spełniają wymagań sformułowanych powyżej. Pozostawia się Czytelnikowi identyfikację warunków, które nie są spełnione przez poszczególne grupy.



Rys. 3.26. Przykłady **niedopuszczalnych** sklejeń w tablicach pięciu zmiennych

Trzeci etap procesu minimalizacji będzie zawierał więc następujące kroki:

1) Wybór do zakreślania jedynek albo zer. Decyzja ta jest przeważnie uzależniona od posiadanych elementów (będzie to wyjaśnione poniżej). Jeżeli elementy nie wprowadzają ograniczeń, to należy łączyć w grupy te symbole, które dają prostsze rozwiązanie. Niekiedy można to przewidzieć, ale przeważnie należy sprawdzić obie możliwości.
2) Zakreślenie wybranego rodzaju symboli w możliwie największe grupy, przy minimalnej liczbie tych grup. Aby osiągnąć ten cel, należy rozpocząć zakreślanie od takich grup, które zawierają co najmniej jeden symbol nie wchodzący do jakiejkolwiek innej grupy. Oczywiście, zgodnie z wcześniej sformułowaną zasadą, musi to być grupa, która nie da się skleić w większą grupę.
3) Wyodrębnione w tablicy grupy opisuje się funkcją w postaci normalnej, redukując wyrażenia wg podanych wyżej zasad.

Zasady postępowania w procesie poszukiwania postaci minimalnej funkcji logicznych zilustrujemy kilkoma przykładami.

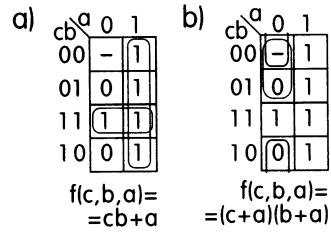
Przykład 3.3

Znaleźć postać minimalną funkcji zapisanej w postaci kanonicznej (3.7):

$\mathbf{f(c, b, a)} = \Sigma[1, 3, 5, 6, 7, (0)]$ (patrz przykład 3.1).

Jak widać (rys. 3.27), efekt minimalizacji metodą graficzną jest identyczny, jak uzyskany metodą przekształceń algebraicznych (wzory (3.13) i (3.14)).

Łatwo też zauważyć, że w przypadku zakreślania zer (tzn. poszukiwania minimalnej normalnej postaci iloczynu) jedynie wykorzystanie kombinacji wejściowej, dla której wartość funkcji jest dowolna, umożliwia sklejenia. „Kreski" w tablicy Karnaugha możemy sklejać dowolnie, ale zawsze mając na celu maksymalizację liczby pól wchodzących w skład jednej grupy. ***Jeżeli takich „kresek" jest więcej, to zakreślenie jednej z nich nie oznacza, że i pozostałe powinny zostać zakreślone.***

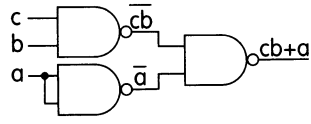


Rys. 3.27. Minimalizacja: a) poprzez poszukiwanie minimalnej postaci sumy (zakreślanie jedynek funkcji); b) poprzez poszukiwanie minimalnej postaci iloczynu (zakreślanie zer funkcji) dla funkcji z przykładu 3.3

Zrealizujmy ten układ korzystając z minimalnej normalnej postaci sumy

$$\mathbf{f(c,b,a) = cb + a = \overline{\overline{cb} + \overline{a}} = \overline{\overline{cb}\,\overline{a}}}$$

Podwójne zanegowanie postaci normalnej sumy oraz zastosowanie prawa de Morgana, przekształca zapis w postać bezpośrednio odzwierciedlającą strukturę realizowalną przy użyciu funktorów NAND (rys. 3.28).



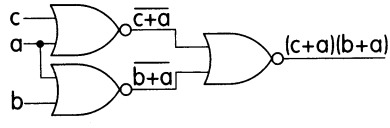
Rys. 3.28. Realizacja postaci normalnej sumy przy użyciu bramek NAND funkcji z przykładu 3.3

Dalej będziemy przyjmować, że ***w celu uzyskania minimalnej realizacji z bramek* NAND *będziemy poszukiwać minimalnej normalnej postaci sumy*** (tzn. będziemy zakreślać **1** funkcji).

Wykonajmy podobny zabieg na minimalnej postaci koniunkcyjnej.

$$\mathbf{f(c,b,a) = \overline{\overline{(c + a)(b + a)}} = \overline{\overline{c + a} + \overline{b + a}}}$$

Podwójne zanegowanie postaci normalnej iloczynu oraz zastosowanie prawa de Morgana przekształca zapis w postać bezpośrednio odzwierciedlającą strukturę realizowalną przy użyciu funktorów NOR (rys. 3.29).

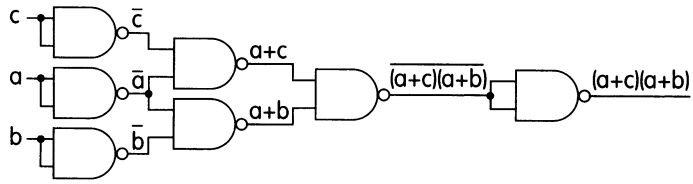


Rys. 3.29. Realizacja postaci normalnej iloczynu przy użyciu bramek NOR funkcji z przykładu 3.3

Dalej będziemy przyjmować, że ***w celu uzyskania minimalnej realizacji z bramek* NOR *poszukiwać będziemy minimalnej normalnej postaci iloczynu*** (tzn. zakreślać będziemy **0** funkcji).

Możliwa jest oczywiście realizacja funkcji z bramek NAND na podstawie postaci normalnej iloczynu (uzyskanej poprzez zakreślanie zer funkcji); np.:

$$\mathbf{f(c,b,a) = \overline{\overline{(c + a)}\;\overline{(b + a)}} = \overline{\overline{c}\,\overline{a}\,\overline{b}\,\overline{a}} = \overline{\overline{\overline{c}\,\overline{a}\,\overline{b}\,\overline{a}}}}$$



Rys. 3.30. Realizacja postaci normalnej iloczynu przy użyciu bramek NAND

Z porównania schematów na rys. 3.29 i 3.30 widać, że takie postępowanie jest dalekie od optymalnego. Do realizacji należało użyć znacznie więcej bramek, wzrosła liczba połączeń, co w sumie wpływa negatywnie zarówno na koszt układu, jak i jego niezawodność. ■

Przykład 3.4

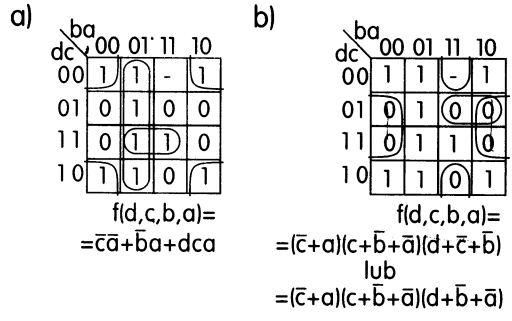
Narysować schemat układu opisanego funkcją:

$\mathbf{f(d,c,b,a)} = \Sigma[0, 1, 2, 5, 8, 9, 10, 13, 15, (3)]$

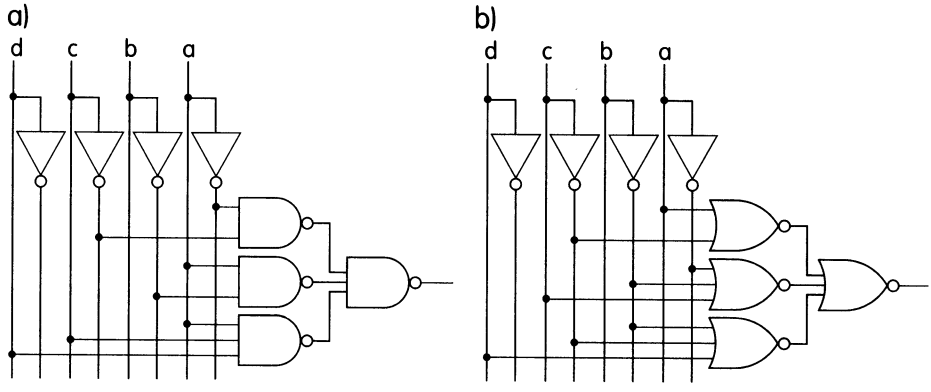
a) używając bramek NAND, b) używając bramek NOR.

Rozwiązanie

Tablice Karnaugha oraz sposób minimalizacji przedstawiono na rys. 3.31, a odpowiednie schematy logiczne na rys. 3.32.



Rys. 3.31. Minimalizacja: a) poprzez poszukiwanie minimalnej postaci sumy (zakreślanie jedynek funkcji); b) poprzez poszukiwanie minimalnej postaci iloczynu (zakreślanie zer funkcji) dla funkcji z przykładu 3.4



Rys. 3.32. Realizacja układu z przykładu 3.4: a) przy użyciu bramek NAND; b) przy użyciu bramek NOR

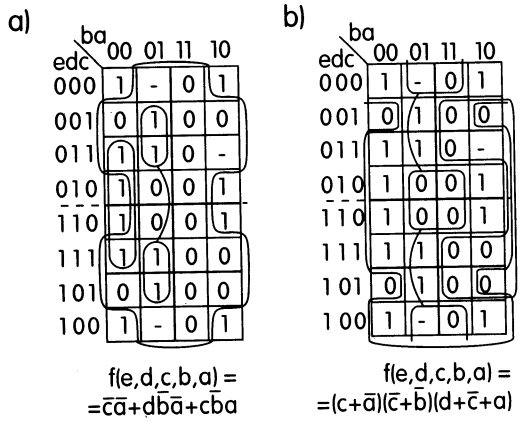
Funkcja ma dwie minimalne normalne postacie iloczynu. Sposób zakreślenia grup zawierających **0**, dla drugiej z nich, pozostawia się Czytelnikowi. ■

Przykład 3.5

Znaleźć minimalną:
a) postać normalną sumy, b) postać normalną iloczynu
funkcji: **f(e,d,c,b,a)** = Π[3,4,6,7,9,11,15,19,20,22,23,25,27,30,31,(1,14,17)].

Rozwiązanie

Tablice Karnaugha oraz sposób minimalizacji pokazano na rys. 3.33. ■



Rys. 3.33. Minimalizacja: a) poprzez poszukiwanie minimalnej postaci sumy (zakreślanie jedynek funkcji); b) poprzez poszukiwanie minimalnej postaci iloczynu (zakreślanie zer funkcji) dla funkcji z przykładu 3.5

Pytania i zadania

1. Zapisz tablice Karnaugha dla funkcji realizowanych przez bramkę:
   a) AND,
   b) NAND,
   c) OR,
   d) NOR.
2. Narysuj tablice Karnaugha dla funkcji realizowanych przez bramkę:
   a) Ex-OR,
   b) Ex-NOR.
   Narysuj schematy tych funkcji używając bramek NAND.
3. Jakie są zasady zakreślania grup w procesie minimalizacji przy użyciu tablic Karnaugha?
4. Używając bramek: a) NAND, b) NOR, zrealizuj funkcję:
   f(d,c,b,a) = Σ[0,2,6,7,8,9,10,15,(1)].
5. Używając bramek: a) NAND, b) NOR, zrealizuj funkcję:
   f(d,c,b,a) = Π[2,3,9,11,12,13,14,(10)].
6. Wyznacz minimalną normalną postać: a) sumy, b) iloczynu
   funkcji: **f(e,d,c,b,a)** = Σ[1,2,3,10,11,15,18,21,23,27,31,(5,7,25,26,29)].
7. Wyznacz minimalną normalną postać: a) sumy, b) iloczynu
   funkcji: **f(e,d,c,b,a)** = Π[0,3,6,8,11,14,16,22,24,(1,7,15,17,30)].

3.5. Realizacja funkcji logicznych przy użyciu elementów stykowych

Technika cyfrowa jest techniką dwustanową. Oznacza to — jak wiemy — że sygnały mogą przyjmować dwa różne stany i w konsekwencji elementy, z których budujemy układy cyfrowe, muszą także być elementami dwustanowymi. Bramki logiczne to dwustanowe elementy półprzewodnikowe. W elektrotechnice natomiast jako elementy dwustanowe są stosowane przekaźniki elektromechaniczne. Przekaźnik to elektromagnes, który w chwili jego wzbudzenia (przepływu prądu przez cewkę) powoduje przyciągnięcie ruchomego fragmentu obwodu magnetycznego (tzw. kotwicy). Przerwanie przepływu prądu sprawia, że w wyniku działania sprężyn zwrotnych kotwica powraca do poprzedniego stanu. Ruch kotwicy jest wykorzystywany do zamykania i otwierania zestyków. Przekaźnik może być wyposażony w **zestyki zwierne** (***normalnie otwarte, normalnie rozwarte***) lub w **zestyki rozwierne** (***normalnie zwarte, normalnie zamknięte***) lub w oba te rodzaje zestyków. Można jeszcze spotkać tzw. **zestyki przełączające**.

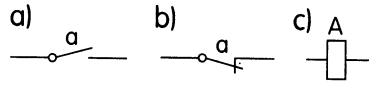
***Zestyki zwierne** (rys. 3.34a) są otwarte w stanie bezprądowym przekaźnika. Przepływ prądu przez cewkę przekaźnika powoduje przyciągnięcie kotwicy, a ta z kolei zamknięcie zestyków normalnie otwartych.*

***Zestyki rozwierne** (rys. 3.34b) są zwarte w stanie bezprądowym przekaźnika. Przepływ prądu przez cewkę przekaźnika powoduje przyciągnięcie kotwicy, a ta z kolei otwarcie zestyków normalnie zwartych.*

Przyjmijmy następującą konwencję oraz symbolikę:

- Duża litera będzie oznaczać cewkę przekaźnika (rys. 3.34c),
  A = 1 przez cewkę przekaźnika **A** płynie prąd, kotwica przyciągnięta,
  A = 0 przez cewkę przekaźnika **A** nie płynie prąd, kotwica zwolniona;

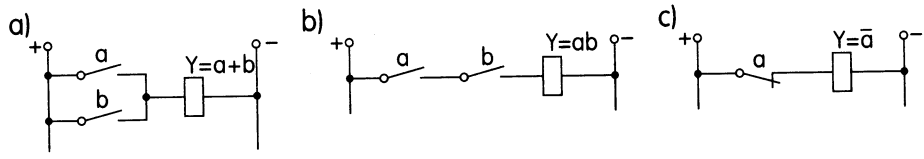
- Mała litera będzie oznaczać zestyk przekaźnika,
  a = 1 ($\bar{a} = 0$) — zestyk zwierny (rozwierny) zamknięty,
  a = 0 ($\bar{a} = 1$) — zestyk zwierny (rozwierny) otwarty.



Rys. 3.34. Symbole elementów stykowych: a) zestyk zwierny, b) zestyk rozwierny; c) cewka przekaźnika

Aby z danych elementów można było zbudować dowolną funkcję logiczną, to zbiór tych elementów musi być SFP (systemem funkcjonalnie pełnym — patrz p. 3.2). Zbiór trzech podstawowych operacji logicznych (suma, iloczyn, negacja) jest takim SFP. (Jak pamiętamy wystarczy tylko suma i negacja bądź iloczyn i negacja, aby był to SFP.)

Sumę (logiczną) **f(a,b) =Y = a+b** *realizujemy z elementów stykowych jako połączenie równoległe zestyków — rys. 3.35a, iloczyn (logiczny)* **f(a,b) = Y = ab** *realizujemy z elementów stykowych jako połączenie szeregowe zestyków — rys. 3.35b, a do realizacji negacji* **f(a) =Y = $\bar{a}$** *używamy zestyku normalnie zwartego — rys. 3.35c. (Uwaga. Na schematach nie rysujemy negacji przy nazwie zestyku rozwiernego.)*



Rys. 3.35. Realizacja podstawowych operacji bulowskich z elementów stykowych: a) suma logiczna; b) iloczyn logiczny; c) negacja

Układy logiczne budowane z elementów stykowych z natury rzeczy są nazywane **układami przełączającymi**. Nazwa ta jest używana czasami także do układów budowanych z bramek logicznych, czyli układów kombinacyjnych. Określenia: układ kombinacyjny i układ przełączający należy traktować jako synonimy. Do opisu układów przełączających używa się bowiem tego samego aparatu matematycznego, jakim jest dwuelementowa algebra Boole'a. W sensie funkcjonalnym dowolny układ cyfrowy możemy zbudować więc z elementów stykowych. Zwykle nie będzie to rozwiązaniem racjonalnym, ze względu na takie wady przekaźników, jak: duże wymiary, duży pobór mocy, mała szybkość działania. Jednak wiele prostych układów logicznych łatwiej i z mniejszym nakładem finansowym można zbudować przy zastosowaniu stykowych elementów „cyfrowych".

Przykład 3.6

Zrealizować z elementów stykowych układ logiczny opisany kanoniczną postacią sumy

$\mathbf{Y = f(d,c,b,a) = \Sigma[2, 4, 6, 11, 12, 14, (1, 15)]}$.

Będziemy poszukiwać realizacji minimalnej. Dlatego wykorzystamy metodę graficzną minimalizacji do znalezienia postaci minimalnej opisującej ten układ. Tablicę Karnaugha oraz minimalizację pokazano na rys. 3.36.

dc \ ba	00	01	11	10
00	0	-	0	1
01	1	0	0	1
11	1	0	-	1
10	0	0	1	0

Y= $c\bar{a}+\bar{d}b\bar{a}+dba$

Rys. 3.36. Tablica Karnaugha do przykładu 3.6

Minimalna normalna postać sumy jest następująca:

$$\mathbf{Y = c\bar{a} + \bar{d}b\bar{a} + dba} \qquad (3.15)$$

Dodatkowym etapem poszukiwania stykowego układu minimalnego (w porównaniu do układu z bramek NAND lub NOR) jest tzw. **faktoryzacja**. Polega ona na wyprowadzeniu przed nawias wspólnych liter w normalnej postaci sumy $\mathbf{aB + aC = a(B + C)}$ lub częściowym wymnożeniu w przypadku nor-

malnej postaci iloczynu $(\mathbf{a} + \mathbf{B})(\mathbf{a} + \mathbf{C}) = \mathbf{a} + \mathbf{BC}$. Faktoryzacja jest także pożądana w przypadku realizacji układu z bramek AND, OR, NOT. Na ogół jednak do realizacji używa się jednego typu bramek, najczęściej bramek NAND (NOR) i wówczas faktoryzacja nie prowadzi do zmniejszenia złożoności układowej.

Minimalna normalna postać sumy funkcji z przykładu 3.6 poddana faktoryzacji przyjmie następującą postać:

$$\mathbf{Y = c\bar{a} + \bar{d}b\bar{a} + dba = \bar{a}(c + \bar{d}b) + dba} \tag{3.16}$$

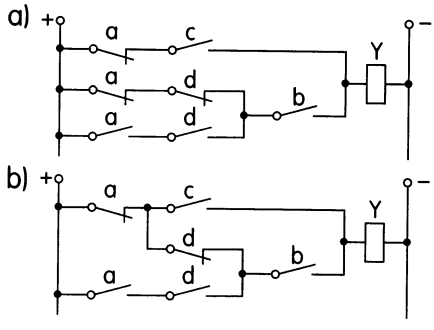
lub

$$\mathbf{Y = c\bar{a} + \bar{d}b\bar{a} + dba = c\bar{a} + b(\bar{d}\bar{a} + da) = \bar{a}c + (\bar{a}\bar{d} + ad)b} \tag{3.17}$$

Układ stykowy narysujemy wykorzystując postać (3.17), która została „uporządkowana" w określonym celu, jaki poniżej zostanie wyjaśniony.

Schemat układu stykowego będącego rozwiązaniem zadania z przykładu 3.6, przedstawiono na rys. 3.37a. ■

Na rysunku 3.37b pokazano dalszą możliwość zmniejszenia liczby zestyków potrzebnych do realizacji analizowanej funkcji. Pomogło w tym wspomniane wyżej „uporządkowanie" zapisu. Sposobem weryfikacji poprawności takich (i każdych innych) przekształceń układów przełączających jest zapisanie wszystkich



Rys. 3.37. Realizacja stykowa funkcji z przykładu 3.6

możliwych połączeń realizujących przepływ prądu przez cewkę na podstawie przekształconego schematu i porównanie uzyskanego zapisu z postacią minimalną (przed lub po faktoryzacji) albo z postacią kanoniczną, co wymaga dodatkowych przekształceń opisu uzyskanego na podstawie schematu.

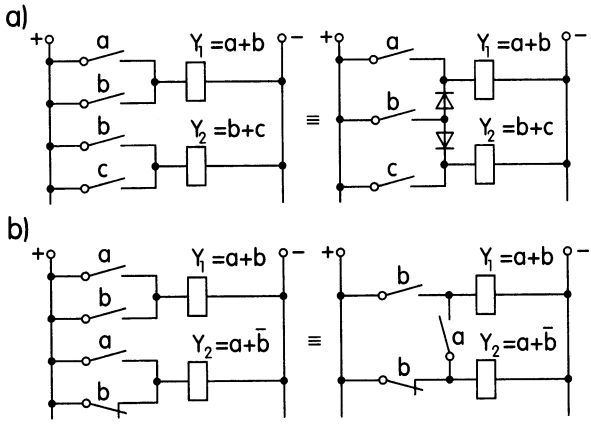
Dla układu przedstawionego na rys. 3.37a opis sporządzony na podstawie schematu będzie następujący:

$$\mathbf{\bar{a}c + \bar{a}\bar{d}b + adb}$$

Z analizy schematu na rys. 3.37b otrzymamy

$$\mathbf{\bar{a}c + \bar{a}\bar{d}b + ad\bar{d}c + adb = \bar{a}c + \bar{a}\bar{d}b + adb}$$

Jak łatwo zauważyć oba powyższe opisy są całkowicie równoważne postaci (3.15). Czasami jednak nie uda się tak prosto zastąpić dwóch zestyków jednym, nie



Rys. 3.38. Metody minimalizacji liczby zestyków w układach przełączających

zmieniając realizowanej funkcji. Wówczas pomocna może okazać się dioda półprzewodnikowa. Przykład takiej modyfikacji układu stykowego z wykorzystaniem diod przedstawiono na rys. 3.38a.

Jeszcze jeden (dość ciekawy) sposób przekształcania, prowadzący do uproszczenia układu przełączającego, pokazano na rys. 3.38b.

Pytania i zadania

1. Narysuj schemat układu przełączającego zbudowanego z elementów stykowych, realizującego funkcję NAND.
2. Narysuj schemat układu przełączającego zbudowanego z elementów stykowych, realizującego funkcję NOR.
3. Zbuduj funktor ExOR z elementów stykowych.
4. Przeprowadź faktoryzację funkcji: **Y = abdfh + cdfh + efh + ghkl**. Oblicz potrzebną do realizacji stykowej liczbę zestyków przed i po faktoryzacji.
5. Wykonaj zadanie 4. dla funkcji: **Y = (a + b)(a + c)(c + d)(b̄ + d)**.
6. Zrealizuj z elementów stykowych, używając minimalnej liczby zestyków funkcję: **f(d,c,b,a)** = Σ[0, 2, 6, 7, 8, 9, 10, 15, (1)].
7. Zrealizuj z elementów stykowych, używając minimalnej liczby zestyków funkcję: **f(d,c,b,a)** = Π[2, 3, 9, 11, 12, 13, 14, (10)].
8. Zrealizuj z elementów stykowych, używając minimalnej liczby zestyków, zespół funkcji (układ wielowyjściowy) opisany:
   X = ac + ad + bc + b̄d;
   Y = ce + de;
   Z = df̄ + af̄
9. Zrealizuj z elementów stykowych, używając minimalnej liczby zestyków, zespół funkcji (układ wielowyjściowy) opisany:
   P1 = ac + bc;
   P2 = ad + bd;
   P3 = cf + df;
   P4 = de + ae
10. Jakie znasz rodzaje zestyków?

3.6. Synteza cyfrowych układów kombinacyjnych

W celu dokonania syntezy układu kombinacyjnego należy:

1. Określić funkcję logiczną (lub zespół funkcji w przypadku układu wielowyjściowego) rozpatrywanego problemu (np. w postaci tablicy prawdy, kanonicznej).
2. Przeprowadzić minimalizację formuły opisującej tę funkcję, wykorzystując do tego celu tablice Karnaugha lub inną znaną sobie metodę minimalizacji.
3. Sporządzić schemat układu logicznego.

Przy realizacji zespołu funkcji logicznych (układu wielowyjściowego) należy dążyć do realizacji minimalnej (najtańszej), biorąc pod uwagę możliwość wykorzystania wspólnych iloczynów (sum) wchodzących w skład rozpatrywanych funkcji. Można dokonać syntezy, realizując każdą funkcję oddzielnie. Jednak układ zrealizowany w ten sposób na ogół nie spełnia wymagania minimalnej złożoności układowej.

Niekiedy korzystnie jest (upraszcza się synteza) dokonać podziału zadania na powiązane ze sobą mniejsze problemy lub — w przypadku układów wielowejściowych — dokonać podziału układu na jednakowe (lub podobne) segmenty, które połączone ze sobą realizują zadanie.

Poniższe przykłady zostały tak dobrane, aby zaprezentować kilka możliwości realizacji układów (z wykorzystaniem różnorodnych elementów), a także różne sposoby syntezy.

Przykład 3.7

Zbudować z dowolnych elementów logicznych sumator dwóch liczb dwubitowych.

Przyjmijmy następujące oznaczenia A + B = C, przy czym A = $\mathbf{a_2a_1}$, B = $\mathbf{b_2b_1}$ i C = $\mathbf{c_3c_2c_1}$. Wówczas możemy zapisać tablicę prawdy, a następnie tablice Karnaugha w taki sposób, jak na rys. 3.39. (**Uwaga.** W powyższym zapisie A + B = C znak „+" oznacza sumę arytmetyczną, a nie logiczną.) Tak więc:

$$\mathbf{c_3 = a_2b_2 + a_2a_1b_1 + a_1b_2b_1 = a_2b_2 + (a_2 + b_2)a_1b_1}$$

$$\mathbf{c_2 = \bar{a}_2\bar{a}_1b_2 + \bar{a}_2b_2\bar{b}_1 + a_2\bar{b}_2\bar{b}_1 + a_2\bar{a}_1\bar{b}_2 + \bar{a}_2a_1\bar{b}_2b_1 + a_2a_1b_2b_1 = \bar{a}_1(\bar{a}_2b_2 + a_2\bar{b}_2) + \bar{b}_1(\bar{a}_2b_2 + a_2\bar{b}_2) + a_1b_1(a_2b_2 + \bar{a}_2\bar{b}_2) = \bar{a}_1(a_2 \oplus b_2) + \bar{b}_1(a_2 \oplus b_2) + a_1b_1(\overline{a_2 \oplus b_2}) = (\bar{a}_1 + \bar{b}_1)(a_2 \oplus b_2) + a_1b_1(\overline{a_2 \oplus b_2}) = \overline{a_1b_1}(a_2 \oplus b_2) + a_1b_1(\overline{a_2 \oplus b_2}) = (a_1b_1) \oplus a_2 \oplus b_2}$$

$$\mathbf{c_1 = \bar{a}_1b_1 + a_1\bar{b}_1 = a_1 \oplus b_1}$$

Schemat logiczny układu przedstawiono na rys. 3.40.

a)

a_2	a_1	b_2	b_1	c_3	c_2	c_1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

b)

a_2a_1 \ b_2b_1	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

c_3

c)

a_2a_1 \ b_2b_1	00	01	11	10
00	0	0	1	1
01	0	1	0	1
11	1	0	1	0
10	1	1	0	0

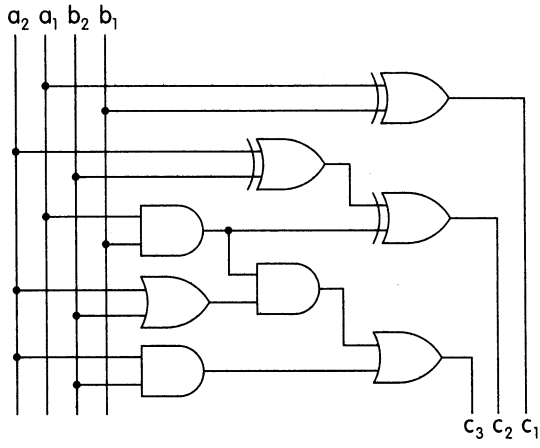
c_2

d)

a_2a_1 \ b_2b_1	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

c_1

Rys. 3.39. Tablica prawdy (a) oraz tablice Karnaugha (b ,c , d) do przykładu 3.7

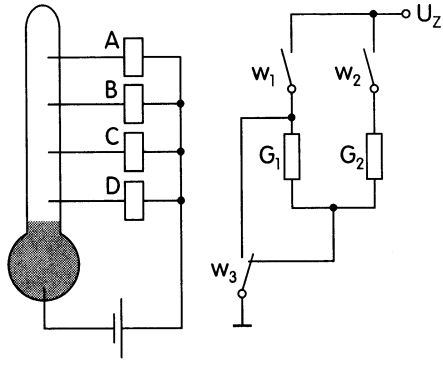


Rys. 3.40. Schemat zasadniczy układu do przykładu 3.7 ■

Przykład 3.8

Termometr kontaktowy dostarcza sygnały **a**, **b**, **c**, **d**, gdy temperatura przekroczy odpowiednie wartości. Zaprojektować układ z elementów stykowych, sterujący zestykami $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ na rys. 3.41 tak, aby przy temperaturze:

$\vartheta < \vartheta_d$ były włączone równolegle grzejniki G_1, G_2,
$\vartheta_d \le \vartheta < \vartheta_c$ — włączony grzejnik G_1,
$\vartheta_c \le \vartheta < \vartheta_b$ — włączony grzejnik G_2,
$\vartheta_b \le \vartheta < \vartheta_a$ — włączone szeregowo grzejniki G_1, G_2,
$\vartheta_a \le \vartheta$ — grzejniki wyłączone.



Rys. 3.41. Rysunek poglądowy do zadania z przykładu 3.8

Rozwiązanie

Tablice prawdy oraz tablice Karnaugha do przykładu 3.8 przedstawiono na rys. 3.42. Rozwiązaniem zadania z przykładu 3.8 jest schemat przedstawiony na rys. 3.43.

a)

a	b	c	d	W_1	W_2	W_3
0	0	0	0	1	1	0
0	0	0	1	1	0	0
0	0	1	0	-	-	-
0	0	1	1	0	1	0
0	1	0	0	-	-	-
0	1	0	1	-	-	-
0	1	1	0	-	-	-
0	1	1	1	0	1	1
1	0	0	0	-	-	-
1	0	0	1	-	-	-
1	0	1	0	-	-	-
1	0	1	1	-	-	-
1	1	0	0	-	-	-
1	1	0	1	-	-	-
1	1	0	0	-	-	-
1	1	1	1	0	0	-

b)

ab \ cd	00	01	11	10
00	1	1	0	-
01	-	-	0	-
11	-	-	0	-
10	-	-	-	-

$W_1 = \bar{c}$

c)

ab \ cd	00	01	11	10
00	1	0	1	-
01	-	-	1	-
11	-	-	0	-
10	-	-	-	-

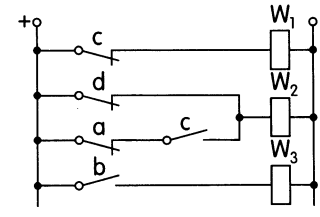
$W_2 = \bar{d} + \bar{a}c$

d)

ab \ cd	00	01	11	10
00	0	0	0	-
01	-	-	1	-
11	-	-	-	-
10	-	-	-	-

$W_3 = b$

Rys. 3.42. Tablica prawdy (a) oraz tablice Karnaugha (b, c, d) do przykładu 3.8



Rys. 3.43. Schemat zasadniczy układu sterowania do zadania z przykładu 3.8 ■

W przykładzie 3.7 zaprojektowano sumator liczb dwubitowych. Zwróćmy uwagę, że układ ten nie może być wykorzystany do sumowania liczb o większej liczbie bitów. Chcąc zbudować układ sumatora liczb np. czterobitowych, należałoby cały proces syntezy powtórzyć. Byłoby to znacznie bardziej skomplikowane. Układ miałby bowiem osiem wejść i pięć wyjść.

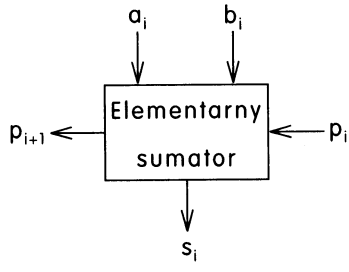
Przyjrzyjmy się poszczególnym krokom wykonywanym w procesie sumowania dwóch (przykładowych) liczb binarnych:

$A = \mathbf{a}_n\mathbf{a}_{n-1}\ldots\mathbf{a}_i\ldots\mathbf{a}_1$ i $B = \mathbf{b}_n\,\mathbf{b}_{n-1}\ldots\mathbf{b}_i\ldots\mathbf{b}_1$ (znak „+" oznacza tutaj dodawanie arytmetyczne).

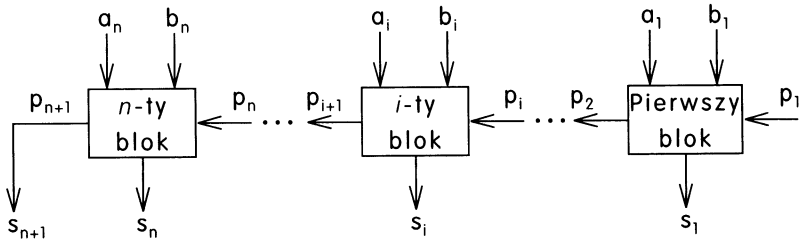
	1	0	1	1	1	1	0	0	0	0	Przeniesienie
	←	←	←	←	←	←	←	←	←	←	
A	1	0	1	1	0	1	1	0	0		Składnik A
+ B	1	0	1	0	1	1	0	1	0		Składnik B
	1	0	1	1	0	0	0	1	1	0	Suma
	MSB								LSB		

Zauważmy, że w każdym kroku sumujemy dwa kolejne bity $\mathbf{a}_i + \mathbf{b}_i$, uwzględniając w procesie sumowania wartości przeniesień $\mathbf{p}_i$ uzyskane w poprzednim $(i-1)$ kroku oraz określamy wynik sumowania $\mathbf{S}_i$ i nową wartość przeniesienia $\mathbf{p}_{i+1}$. Elementarny blok realizujący te operacje możemy nazwać sumatorem elementarnym i w sposób blokowy przedstawić jak na rys. 3.44.

Łącząc ze sobą szeregowo (kaskadowo) n takich elementarnych sumatorów (rys. 3.45) możemy zrealizować sumator liczb n-bitowych.



Rys. 3.44. Sumator elementarny — schemat funkcjonalny



Rys. 3.45. Sumator liczb n-bitowych zbudowany z elementarnych sumatorów

Proces wyodrębniania podstawowej struktury realizującej elementarne, powtarzające się operacje, będziemy nazywać **dekompozycją układu.** W wyniku dekompozycji rozwiązanie zadania sprowadza się do zaprojektownia i-tego bloku, a zadanie zostaje zrealizowane za pomocą n takich bloków połączonych kaskadowo. Układy tak projektowane i budowane będziemy nazywać **układami iteracyjnymi**. Cechą charakterystyczną tych układów jest to, że przetwarzają one ciągi sygnałów binarnych, a działania na i-tych sygnałach są określone jednoznacznie przy udziale pomocniczych sygnałów z $i-1$ (lub $i+1$) segmentu — tak zwanych **sygnałów przeniesienia**. Wyjścia układu mogą pochodzić z każdego członu lub tylko z ostatniego.

Budowanie układu o postaci iteracyjnej umożliwia ujednolicenie sprzętu, zmniejsza liczbę wejść elementów (a często liczbę elementów), pozwala na proste rozszerzenie liczby wejść układu, sprowadza syntezę wielowejściowego układu do syntezy prostego układu o kilku wejściach, ale oznacza także zwiększenie czasu potrzebnego na ustalenie się wyniku, co jest niestety wadą tego rozwiązania.

Najtrudniejszym etapem projektowania układu iteracyjnego jest **dekompozycja układu,** w ramach której należy ustalić:

1) kierunek przeniesień,
2) liczbę sygnałów przeniesienia (liczbę informacji przekazywanych między blokami),
3) logikę działania segmentu (*i*-tego bloku).

Po tych ustaleniach należy opisać (tablica prawdy, tablica Karnaugha) i zaprojektować *i*-ty blok. Zwykle w procesie syntezy kombinacyjnego układu iteracyjnego można pierwszy i ostatni blok nieco zmodyfikować. Przeważnie prowadzi to do uproszczenia układu. Zasady projektowania układów iteracyjnych zilustrujemy na kolejnych przykładach.

Przykład 3.9

Zaprojektować sumator dwóch liczb *n*-bitowych:

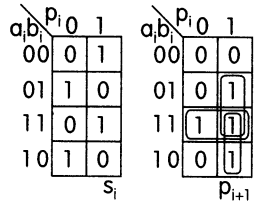
$A = \mathbf{a}_n\mathbf{a}_{n-1}\ldots\mathbf{a}_i\ldots\mathbf{a}_1$ i $B = \mathbf{b}_n\mathbf{b}_{n-1}\ldots\mathbf{b}_i\ldots\mathbf{b}_1$

W rozwiązaniu wykorzystamy przeprowadzony powyżej proces dekompozycji układu. Należy zatem opisać jedynie działanie *i*-tego bloku np. w postaci tablicy prawdy (rys. 3.46).

a)

a_i	b_i	p_i	s_i	p_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b)

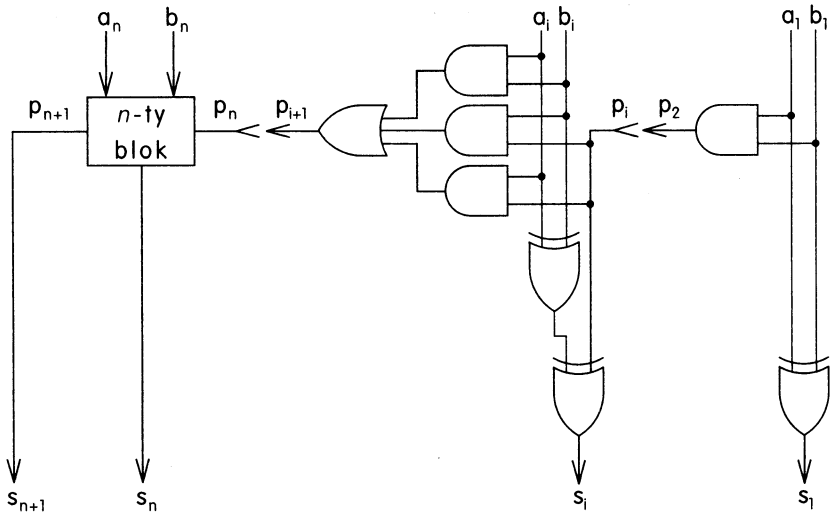


Rys. 3.46. Opis działania elementarnego sumatora: a) tablica prawdy; b) tablice Karanaugha

W wyniku minimalizacji (rys. 3.46), a następnie przekształceń algebraicznych otrzymujemy:

$$\mathbf{s}_i = \bar{\mathbf{a}}_i\bar{\mathbf{b}}_i\mathbf{p}_i + \bar{\mathbf{a}}_i\mathbf{b}_i\bar{\mathbf{p}}_i + \mathbf{a}_i\mathbf{b}_i\mathbf{p}_i + \mathbf{a}_i\bar{\mathbf{b}}_i\bar{\mathbf{p}}_i = \bar{\mathbf{a}}_i(\bar{\mathbf{b}}_i\mathbf{p}_i + \mathbf{b}_i\bar{\mathbf{p}}_i) + \mathbf{a}_i(\mathbf{b}_i\mathbf{p}_i + \bar{\mathbf{b}}_i\bar{\mathbf{p}}_i) =$$
$$= \bar{\mathbf{a}}_i(\mathbf{b}_i \oplus \mathbf{p}_i) + \mathbf{a}_i\overline{(\mathbf{b}_i \oplus \mathbf{p}_i)} = \mathbf{a}_i \oplus \mathbf{b}_i \oplus \mathbf{p}_i$$

$$\mathbf{p}_{i+1} = \mathbf{a}_i\mathbf{b}_i + \mathbf{b}_i\mathbf{p}_i + \mathbf{a}_i\mathbf{p}_i$$



Rys. 3.47. Schemat zasadniczy *n*-bitowego sumatora

Powyższe równania opisują strukturę *i*-tego bloku. Opis bloku pierwszego otrzymamy, uwzględniając w powyższych równaniach, że: $\mathbf{a}_i = \mathbf{a}_1$; $\mathbf{b}_i = \mathbf{b}_1$; $\mathbf{p}_i = \mathbf{p}_1 = \mathbf{0}$. A zatem

$$\mathbf{s}_1 = \mathbf{a}_1 \oplus \mathbf{b}_1 \oplus \mathbf{0} = \mathbf{a}_1 \oplus \mathbf{b}_1$$

$$\mathbf{p}_2 = \mathbf{a}_1\mathbf{b}_1 + \mathbf{b}_1\mathbf{0} + \mathbf{a}_1\mathbf{0} = \mathbf{a}_1\mathbf{b}_1$$

Struktura *n*-tego bloku będzie identyczna jak bloku *i*-tego. Należy jedynie uwzględnić, że przeniesienie p_{n+1} będzie n+1 bitem sumy. Schemat ideowy *n*-bitowego sumatora przedstawiono na rys. 3.47. ■

Przykład 3.10

Zaprojektować komparator dwóch liczb *n*-bitowych:

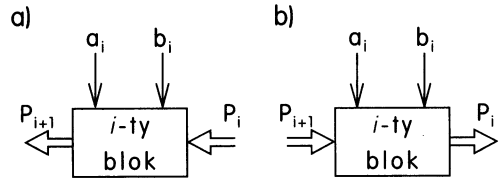
$$A = \mathbf{a}_n\mathbf{a}_{n-1}\ldots\mathbf{a}_i\ldots\mathbf{a}_1 \text{ i } B = \mathbf{b}_n\mathbf{b}_{n-1}\ldots\mathbf{b}_i\ldots\mathbf{b}_1$$

Układ powinien mieć trzy wyjścia $Y_{A>B}$, $Y_{A=B}$, $Y_{A<B}$.

Rozwiązanie

W wyniku dekompozycji układu stwierdzamy, że *i*-ty blok powinien analizować bity $\mathbf{a}_i$, $\mathbf{b}_i$ słów wejściowych *A*, *B*, przekazywać do kolejnego bloku (i pobierać z poprzedzającego) trzy różne informacje: ($A>B$), ($A = B$) i ($A<B$). Wyjście układu należy zrealizować na podstawie informacji przeniesień pochodzącej z ostatniego (*n*-tego) bloku. Kierunek przekazywanych informacji może być dowolny. Jednak wybór określonego kierunku będzie rzutował na algorytm działania *i*-tego bloku. W związku z tym komparator, w którym prze-

niesienia będą przekazywane w kierunku od młodszych (LSB) bitów do starszych (MSB), nazwijmy komparatorem LSB, a komparator o przeciwnym kierunku przeniesień komparatorem MSB (rys. 3.48).



Rys. 3.48. Blok *i*-ty komparatora: a) LSB; b) MSB

Przyjmijmy do dalszych etapów syntezy komparator LSB. Działanie *i*-tego bloku komparatora LSB możemy przedstawić za pomocą tablicy pokazanej na rys. 3.49. W tablicy tej zapisano, jaką powinien generować blok *i*-ty informację przeniesienia P_{i+1}, gdy na jego wejściu znajduje się para bitów a_ib_i oraz jest wprowadzana informacja przeniesienia P_i. Zbiór informacji P_i= {(*A*>*B*), (*A* = *B*), (*A*<*B*)} przekazywanych pomiędzy blokami zawiera trzy różne informacje. Aby zakodować trzy różne informacje, potrzeba w tym celu co najmniej dwóch sygnałów binarnych. Sposób kodowania jest dowolny. Złożoność projektowanego układu zależy od sposobu kodowania. Z góry jednak nie można przewidzieć, jakie kodowanie zapewni lepszy rezultat. Przyjmijmy więc jeden z możliwych sposobów kodowania (rys. 3.50).

P_i \ a_ib_i	00	01	11	10
A>B	A>B	A<B	A>B	A>B
A=B	A=B	A<B	A=B	A>B
A<B	A<B	A<B	A<B	A>B

P_{i+1}

Rys. 3.49. Przetwarzanie informacji wejściowej przez *i*-ty blok komparatora LSB

P_i	p_i q_i
A>B	1 0
A=B	0 0
A<B	0 1

Rys. 3.50. Kodowanie informacji przeniesień p_i, q_i — sygnały przenoszące informacje

Wykorzystując wcześniej sporządzoną tablicę opisującą działanie komparatora LSB oraz przyjęty sposób kodowania, możemy sporządzić zapis w układzie tablicy Karnaugha jak na rys. 3.51.

a)

p_iq_i \ a_ib_i	00	01	11	10
A=B 00	00	01	00	10
A<B 01	01	01	01	10
— 11	--	--	--	--
A>B 10	10	01	10	10

p_{i+1}, q_{i+1}

b)

p_iq_i \ a_ib_i	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	-	-	-	-
10	1	0	1	1

p_{i+1}

c)

p_iq_i \ a_ib_i	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	-	-	-	-
10	0	1	0	0

q_{i+1}

Rys. 3.51. Zakodowana tablica z rys. 3.49 (a) oraz minimalizacja postaci funkcji opisujących sygnały przeniesień (b) i (c)

Aby ułatwić minimalizację, należy rozdzielić pary sygnałów $\mathbf{p_{i+1}}$, $\mathbf{q_{i+1}}$ na dwie odrębne tablice Karnaugha. W wyniku minimalizacji otrzymamy:

$$\mathbf{p_{i+1} = a_i\bar{b}_i + p_ia_i + p_i\bar{b}_i = a_i\bar{b}_i + p_i(a_i + \bar{b}_i) = a_i\bar{b}_i + \overline{p_i\bar{a}_ib_i}}$$

$$\mathbf{q_{i+1} = \bar{a}_ib_i + q_i\bar{a}_i + q_ib_i = \bar{a}_ib_i + q_i(\bar{a}_i + b_i) = \bar{a}_ib_i + \overline{q_ia_i\bar{b}_i}}$$

Opis bloku pierwszego otrzymamy uwzględniając w powyższych równaniach, że informacja $P_1 = (A = B)$, czyli sygnały ją przenoszące $\mathbf{p_1 = q_1 = 0}$. A zatem

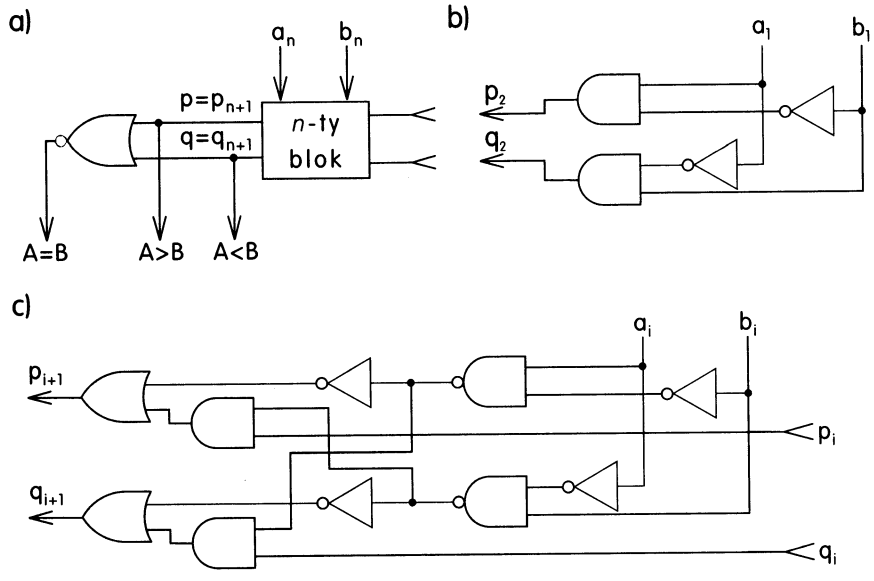
$$\mathbf{p_2 = a_1\bar{b}_1} \quad \text{oraz} \quad \mathbf{q_2 = \bar{a}_1b_1}$$

Aby układ miał trzy wyjścia $\mathbf{y_{A>B}}$, $\mathbf{y_{A=B}}$, $\mathbf{y_{A<B}}$ należy sygnały przeniesień $\mathbf{p_{n+1} = p}$, $\mathbf{q_{n+1} = q}$ odpowiednio zdekodować na kod **1 z 3** (rys. 3.52).
Schemat logiczny komparatora sporządzony na podstawie powyższych zależności przedstawiono na rys. 3.53.

p	q	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$
0	0	0	1	0
0	1	0	0	1
1	1	–	–	–
1	0	1	0	0

Po minimalizacji
$Y_{A>B} = p$
$Y_{A=B} = \bar{p}\bar{q} = \overline{p+q}$
$Y_{A<B} = q$

Rys. 3.52. Tablica prawdy opisująca działanie bloku *n*+1



Rys. 3.53. Schemat logiczny komparatora: a) blok *n*-ty; b) blok pierwszy; c) blok *i*-ty ■

Pytania i zadania

1. Zaprojektuj układ podnoszący do kwadratu cyfry dziesiętne zapisane w naturalnym kodzie dwójkowym. Wynik ma być także w kodzie dwójkowym. Użyć elementów NAND.
2. Zaprojektuj układ podnoszący do kwadratu cyfry dziesiętne zapisane w naturalnym kodzie dwójkowym. Wynik ma być w kodzie **BCD 8421.** Użyć elementów NAND.
3. Zaprojektuj, używając elementów NOR, układ do mnożenia liczb dwubitowych.
4. Zaprojektuj, używając elementów NAND, układ do mnożenia liczb trzybitowych przez liczby dwubitowe.
5. Zaprojektuj konwerter kodu **naturalnego BCD** na kod **2 z 5.**
6. Zaprojektuj konwerter kodu **2 z 5** na kod **naturalny BCD.**
7. W banku jest kasa pancerna, do której klucze mają dyrektor i trzech kierowników. Kasę można otworzyć w dni robocze między godziną 8^{00} a 15^{00} kluczami dwóch spośród trzech kierowników lub kluczem dyrektora. W niedzielę w tych samych godzinach do otwarcia kasy konieczne są klucze wszystkich trzech kierowników lub klucz dyrektora. O innej porze dnia do otwarcia kasy konieczne są wszystkie cztery klucze. Zakładając, że są dostępne potrzebne sygnały z zegara, zaprojektuj układ logiczny sterujący zamkiem kasy.
8. Zaprojektuj komparator MSB (patrz przykład 3.10).
9. Zaprojektuj układ kontroli kodu. Jeżeli n-bitowe słowo wejściowe należy do kodu **1 z *n*** to na wyjściu **y** układu powinien być stan wysoki **H**, a w pozostałych przypadkach stan niski **L**.
10. Rozwiąż zadanie 9. dla kodu $\overline{\mathbf{1\ z\ }\boldsymbol{n}}$.
11. Zaprojektuj układ o n wejściach i 1 wyjściu dający sygnał na wyjściu **y** = **1**, jeżeli liczba jedynek K na wejściu spełnia warunek $K = 3m + 1$; przy czym $m = 0, 1, 2, \ldots$.
12. Zaprojektuj układ przesyłający na wyjście grupy dokładnie trzech jedynek ze słowa wejściowego. Na pozostałych wyjściach mają być zera. Na przykład:
    Słowo wejściowe **1 0 1 1 0 1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 1**
    Słowo wyjściowe **0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0**
13. Co nazywamy dekompozycją układu?
14. Jakie są wady i zalety układów iteracyjnych?

4

Techniki realizacyjne układów cyfrowych

4.1. Klasyfikacja cyfrowych układów scalonych

W zależności od przyjętego kryterium układy scalone można podzielić na różne klasy.

Ze względu na postać przetwarzanych sygnałów układy scalone dzieli się na:

— cyfrowe,

— liniowe (analogowe).

Kolejnym kryterium jest złożoność układu określana mianem **stopnia scalenia.** ***Miarą* stopnia scalenia *jest liczba bramek elementarnych tworzących dany układ scalony lub liczba elementów.*** Określenia dotyczące stopnia scalenia odnoszą się tylko do półprzewodnikowych scalonych układów cyfrowych.

Tablica 4.1. **Miary stopnia scalenia cyfrowych układów scalonych**

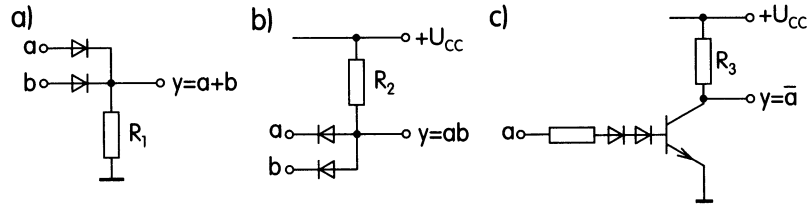
Stopień scalenia	Liczba bramek	Liczba elementów
SSI — mały stopień scalenia (ang. *Small Scale Integration*)	do 10	do 100
MSI — średni stopień scalenia (ang. *Medium Scale Integration*)	10÷100	100÷1000
LSI — duży stopień scalenia (ang. *Large Scale Integration*)	100÷10000	1000÷100000
VLSI — bardzo duży stopień scalenia (ang. *Very Large Scale Integration*)		>100000

Następnym kryterium klasyfikacji układów cyfrowych jest struktura elektroniczna (schemat zasadniczy) podstawowego funktora lub **technologia ich wytwarzania.** Zgodnie z powyższym kryterium rozróżnia się następujące układy: DTL, RTL, DCTL, TTL, ECL, MOS, I^2L, CTD. Większość z nich zostanie poniżej krótko scharakteryzowana, natomiast układy TTL oraz CMOS (należące do grupy układów MOS) zostaną dokładnie opisane w kolejnych rozdziałach.

Nazewnictwo literowe klas układów cyfrowych wywodzi się z terminologii angielskiej i jest związane z charakterystycznymi cechami konfiguracji elektronicznej podstawowego funktora logicznego lub zastosowanej technologii.

- **Technika DTL**

Technika DTL (ang. *Diode Transistor Logic*) jest najprostszą z technik półprzewodnikowych. Ze względu na swoje wady, takie jak mała obciążalność wyjść i wrażliwość na zakłócenia, technika ta została wyparta przez układy TTL i CMOS. Układy te w praktyce nie są obecnie używane. Na rysunku 4.1 przedstawiono realizację podstawowych funkcji logicznych w technice DTL dla wykazania, jak łatwo można zrealizować podstawowe funktory logiczne. Ten sposób realizacji układu logicznego może znaleźć zastosowanie w układach analogowych do realizacji prostych operacji (logicznych) sumy czy iloczynu.



Rys. 4.1. Realizacja podstawowych funktorów w technice DTL: a) suma logiczna; b) iloczyn logiczny; c) negacja

Realizacja negatora (rys.4.1c) wymaga użycia elementu odwracającego fazę (tranzystora). Na wejściu zastosowano dodatkowo diody, aby podnieść poziom napięcia odpowiadającego logicznemu **0**. Przy braku tych diod napięcie ok. 0.6 V (napięcie przewodzenia złącza baza-emiter) odpowiadałoby już poziomowi wysokiemu (**1** logicznej).

- **Technika TTL**

Technika TTL (ang. *Transistor-Transistor Logic*) jest zmodyfikowaną techniką DTL, w której elementy diodowe zastąpiono tranzystorem wieloemiterowym. Jest ona najbardziej rozpowszechnioną techniką wytwarzania cyfrowych układów scalonych, szczegółowo omówioną w rozdz. 5.

- **Techniki MOS**

W omawianych dotychczas technikach były stosowane wyłącznie tranzystory bipolarne. W układach o dużym stopniu scalenia są obecnie stosowane najczęściej tranzystory unipolarne MOS (ang. *Metal-Oxide-Semiconductor*). Jak wiadomo z podstaw elektroniki tranzystory MOS mogą być budowane z kanałem typu P (gdzie nośnikami ładunku elektrycznego są dziury) lub z kanałem typu N (gdzie nośnikami ładunku elektrycznego są elektrony). Stąd w ramach techniki MOS można rozróżnić technikę PMOS i NMOS.

Technika PMOS (ang. *P-channel Metal-Oxide-Semiconductor*) jest techniką MOS z kanałem typu P. Jej podstawową wadą jest konieczność stosowania kilku źródeł zasilania.

Technika NMOS (ang. *N-channel Metal-Oxide-Semicondudtor*) jest techniką MOS z kanałem typu N. Jest ona dogodniejsza w stosowaniu, gdyż wymaga mniejszej liczby napięć zasilających niż technika PMOS.

Zazwyczaj nie produkuje się układów cyfrowych NMOS czy PMOS w postaci podstawowych bramek logicznych lub przerzutników, lecz od razu całe bloki funkcjonalne (układy realizujące złożone operacje logiczne). Są to więc wyłącznie układy MSI i LSI (średniego i dużego stopnia scalenia).

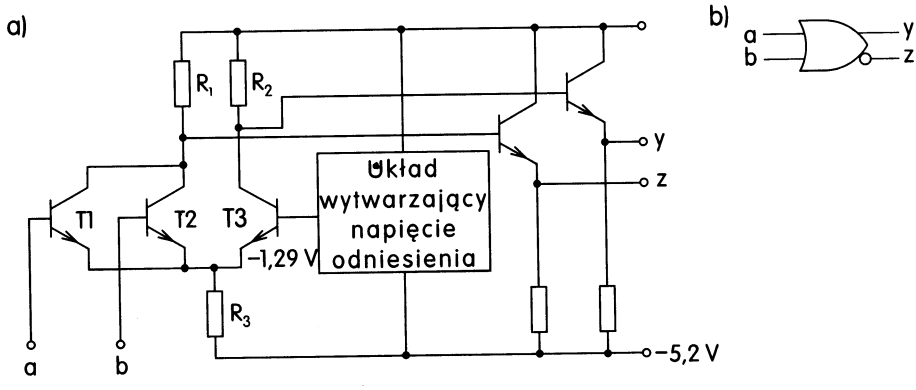
Technika CMOS należy także do technik MOS. Wykorzystuje ona zarówno tranzystory z kanałem typu P, jak i tranzystory z kanałem typu N. Technika CMOS (ang. *Complementary MOS*) nazywana także techniką COSMOS (ang. *COmplementary Symmetry MOS*) realizuje układy logiczne zawierające wyłącznie tranzystory przeciwstawne (komplementarne). Technika CMOS jest techniką równie popularną, jak technika TTL (w ostatnich latach nawet zaczyna zajmować pozycję dominującą) i zostanie jej poświęcony rozdz. 6.

Wszystkie techniki MOS wykorzystują wyłącznie tranzystory, dzięki czemu są łatwe do scalania.

- **Technika ECL**

Technika ECL (ang. *Emitter Coupled Logic*) należy do najszybszych technik półprzewodnikowych, gdyż tranzystory (bipolarne) podczas pracy nie wchodzą w obszar nasycenia.

Schemat elementu OR/NOR ECL przedstawia rys. 4.2. Emitery tranzystorów *T1* i *T2* są połączone z emiterem tranzystora odniesienia *T3*, a baza tego tranzystora jest przyłączona do układu wytwarzającego napięcie odniesienia (−1,29 V w elementach serii 10000). Sygnałem o wartości logicznej **1** jest tu napięcie −0,9 V, sygnałem o wartości logicznej **0** napięcie −1,75 V.



Rys. 4.2. Element OR/NOR ECL: a) schemat; b) symbol

- **Technika I^2L**

Układy bipolarne zwane I^2L (ang. *Integrated Injection Logic*) charakteryzują się bardzo małą powierzchnią zajmowaną przez pojedynczą bramkę, co umożliwia osiągnięcie dużej gęstości upakowania w strukturze scalonej. Złożoność technologii wytwarzania układów I^2L jest porównywalna z tą, jaka jest potrzebna do realizacji układów TTL.

- **Technika CTD**

Układy z transmisją ładunku (CTD — ang. *Charge Transfer Devices*) stanowią klasę elementów półprzewodnikowych, których zasada działania polega na zjawisku magazynowania i transportu ładunku, reprezentującego informację. Technika wytwarzania układów CTD jest oparta na technologii MOS. Podstawowym elementem takiego układu jest kondensator MOS. Technika ta jest wykorzystywana do budowy pamięci półprzewodnikowych.

4.2. Oznaczenia cyfrowych układów scalonych

Układy cyfrowe są produkowane przez wiele firm na świecie. Niektóre z tych firm stosują własne, odmienne oznaczenia swoich wyrobów. Oznaczenie SN74 jest stosowane przez firmę Texas Instruments. Układy firmy Fairchild są określane jako układy serii 9000. Firma Signetic produkuje serie 74 i 8200, a firma Motorola układy serii MC3000. Układy polskiej produkcji mają oznaczenia zgodne z normą branżową BN-78-3375-21.

Symbolika stosowana w oznaczaniu układów polskiej produkcji jest tworzona zgodnie z zasadami podanymi poniżej.

- Pierwszy znak — litera — określa wykonanie:
  U — układ scalony półprzewodnikowy monolityczny wykonany w technologii bipolarnej,
  M — układ scalony półprzewodnikowy monolityczny wykonany w technologii unipolarnej.

- Drugi znak — litera — określa spełnianą funkcję:
  C — układy cyfrowe,
  L — układy liniowe (analogowe).

- Trzeci znak — litera — określa zastosowanie:
  X — wykonanie prototypowe, doświadczalne,
  Y — wykonanie do zastosowań w sprzęcie profesjonalnym,
  A — do zastosowań specjalnych.
  Brak litery oznacza wyrób do zastosowań w sprzęcie powszechnego użytku.

Uwaga. Każda firma produkująca układy cyfrowe wprowadza na wyżej wymienione pozycje własne oznaczenia. Dalsze informacje dotyczące kolejnych pozycji są już bardziej uniwersalne i odnoszą się do produktów wielu firm.

- Kolejny znak — cyfra — określa zakres dopuszczalnej temperatury otoczenia.
  1 — zakres inny niż wymienione poniżej,
  4 — od −55 do +85°C,
  5 — od −55 do +125°C,
  6 — od −40 do +85°C,
  7 — od 0 do +70°C,
  8 — od −25 do +85°C.

- Kolejny znak — cyfra — określa numer serii. Dodatkowo mogą wystąpić jedna lub dwie litery określające rodzaj serii:

  Dla układów TTL:

  Brak litery — seria standardowa,
  L — seria małej mocy,
  H — seria o zwiększonej szybkości,
  S — seria Schottky'ego (bardzo szybka),
  LS — seria Schottky'ego o małym poborze mocy,
  F — seria szybka,
  ALS — ulepszona Schottky'ego małej mocy,
  AS — ulepszona Schottky'ego (najszybsza).

  Dla układów CMOS:

  HC — szybkie układy CMOS,
  HCT — szybkie układy CMOS kompatybilne z układami TTL,
  AC — ulepszone szybkie układy CMOS (można spotkać także oznaczenie ACL),
  ACT — ulepszone szybkie układy CMOS kompatybilne z układami TTL.

- Kolejne znaki — dwie lub trzy cyfry — są liczbą porządkową określającą rodzaj elementu.

- Na końcu może wystąpić jeszcze litera określająca rodzaj obudowy (F, S, H, J, N, L, K, M, P, R, T). Na przykład:
  N — obudowa dwurzędowa plastykowa (typu DIL — ang. *Dual In Line*),
  K — obudowa czterorzędowa plastykowa.

Na przykład oznaczenie UCY74LS132N należy rozumieć jako: układ scalony półprzewodnikowy monolityczny wykonany w technologii bipolarnej (U), cyfrowy (C), do zastosowań w sprzęcie profesjonalnym (Y), o dopuszczalnych temperaturach otoczenia w zakresie od 0°C do +70°C, o numerze seryjnym 4, serii układów Schottky'ego (bardzo szybkich) małej mocy. Jest to układ zawierający w dwurzędowej obudowie plastykowej (N) cztery dwuwejściowe bramki NAND z przerzutnikiem Schmitta (132).

Pytania i zadania

1. Wymień podstawowe zalety scalania układów elektronicznych.
2. Wymień powody, dla których technologię scalania wprowadzono najpierw do techniki układów cyfrowych.
3. Jakie wady mają elementy bierne wykonywane w układach scalonych?
4. Jakie elementy jest najłatwiej wykonać w strukturze układu scalonego?
5. Wyjaśnij pojęcie stopnia scalenia.
6. Jakie kryteria są stosowane przy klasyfikacji scalonych układów cyfrowych?
7. Jaka jest podstawowa różnica między techniką TTL a technikami MOS?
8. Jaka jest podstawowa różnica między układami NMOS, PMOS a CMOS?
9. Wyjaśnij znaczenie następujących symboli:
   a) UCY74H00N,
   b) UCY64L00N,
   c) UCY84LS00N,
   d) UCY44S00N,

   wiedząc, że układ UCY7400N to: układ cyfrowy scalony półprzewodnikowy wykonany w technice bipolarnej, przeznaczony do zastosowań w sprzęcie profesjonalnym i pracy w temperaturze otoczenia od 0°C do +70°C, serii standardowej, zawierający w plastykowej obudowie dwurzędowej (typu DIL) cztery dwuwejściowe bramki NAND.

4.3. Podstawowe parametry scalonych układów cyfrowych

Do podstawowych parametrów półprzewodnikowych scalonych układów cyfrowych należą:

Czas propagacji t_p, ***określający szybkość działania układu.***
Straty mocy (moc pobierana, moc rozpraszana), ***określające moc pobieraną z zasilacza.***
Margines zakłóceń ΔU, ***określający odporność układu na zakłócenia.***

Wartości typowe tych parametrów są najczęściej wykorzystywane do analizy porównawczej serii układów wykonanych w różnych technologiach, przez różnych producentów. Parametry te mogą być przydatne użytkownikowi przy wyborze serii układów.

Projektanta systemów cyfrowych interesują jeszcze takie parametry, jak: napięcia i prądy zasilania, napięcia i prądy wejściowe/wyjściowe, obciążalność wyjść, asortyment układowy i inne.

Znajomość tych wszystkich parametrów umożliwia świadomy i optymalny wybór określonej klasy układów do konkretnych zastosowań, a także właściwą ich eksploatację.

Na oznaczenie poszczególnych parametrów przyjęto używać pewne symbole literowe, których znaczenie staje się oczywiste, jeśli jest znane ich pochodzenie.

Jak większość tego typu określeń i symboli (stosowanych w technice cyfrowej), tak i te mają swoje źródło w języku angielskim.

Litera **I** pochodzi od angielskiego słowa *Input* — wejście, litera **O** od słowa *Output* — wyjście, litera **H** od *High* — wysoki, litera **L** od słowa *Low* — niski, litera **S** od *Shorting* (*Short-circuit*) — zwarcie, litera **P** od *Propagation* — propagacja. Uwzględniając powyższe uwagi znaczenie niżej podanych symboli staje się oczywiste i łatwiejsze do zapamiętania.

- **Parametry statyczne**

  U_{CC} — napięcie zasilania,
  U_{IH} — napięcie wejściowe w stanie wysokim,
  U_{IL} — napięcie wejściowe w stanie niskim,
  U_{OH} — napięcie wyjściowe w stanie wysokim,
  U_{OL} — napięcie wyjściowe w stanie niskim,
  I_{IH} — prąd wejściowy w stanie wysokim,
  I_{IL} — prąd wejściowy w stanie niskim,
  I_{OH} — prąd wyjściowy w stanie wysokim,
  I_{OL} — prąd wyjściowy w stanie niskim,
  I_{CCH} — prąd zasilania układu w stanie wysokim na wyjściu,
  I_{CCL} — prąd zasilania układu w stanie niskim na wyjściu,
  I_{OS} — wyjściowy prąd zwarciowy,
  ΔU_L — margines zakłóceń w stanie niskim,
  ΔU_H — margines zakłóceń w stanie wysokim.

- **Parametry dynamiczne**

  t_{pHL} — czas propagacji przy zmianie stanu logicznego na wyjściu z wysokiego (**H**) na niski (**L**), tj. czas upływający między występowaniem na wejściu i na wyjściu napięcia $(U_{IH\,min} + U_{IL\,max})/2$ przy zmianie stanu logicznego na wyjściu z **H** na **L**.

  t_{pLH} — czas propagacji przy zmianie stanu logicznego na wyjściu z niskiego (**L**) na wysoki (**H**), tj. czas upływający między występowaniem na wejściu i na wyjściu napięcia $(U_{IH\,min} + U_{IL\,max})/2$ przy zmianie stanu logicznego na wyjściu z **L** na **H**,

  t_p — czas propagacji, tj. średnia arytmetyczna czasów t_{pLH} i t_{pHL} lub niekiedy wartość większa spośród czasów t_{pLH}, t_{pHL}.

Wszystkie te parametry będą w podręczniku sukcesywnie definiowane i komentowane przy okazji omawiania odpowiednich charakterystyk statycznych. Obecnie ograniczymy się do zdefiniowania podstawowych parametrów.

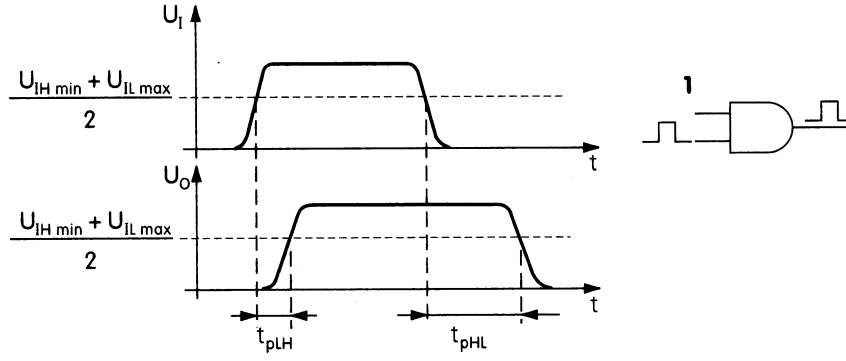
- ***Czas propagacji t_p***

Formalna definicja czasu propagacji została już powyżej sformułowana, ale ponieważ jest dość skomplikowana, przeto zostanie przedstawiona dodatkowo bardziej czytelna definicja graficzna (za pomocą odpowiednich przebiegów czasowych). Ale przedtem kilka uwag natury ogólnej.

Opóźnienia sygnałów wyjściowych w stosunku do sygnałów wejściowych elementów cyfrowych są konsekwencją określonego czasu trwania procesu przełączania. Czas ten jest sumą czasu przeładowania pojemności pasożytniczych i czasu przejścia tranzystora ze stanu przewodzenia do stanu blokowania. Opóźnienie pojawienia się odpowiedzi układu jest określane w technice układów cyfrowych jako **czas propagacji t_p**.

Procesy przełączania wyjścia mogą przebiegać z różnymi szybkościami w zależności od kierunku przełączania. Jeżeli zmiana sygnału wejściowego spowoduje zmianę stanu wyjścia z **H** na **L** (zbocze ujemne), to odcinek czasu zawarty pomiędzy zboczem sygnału wejściowego a zboczem sygnału wyjściowego oznacza się jako t_{pHL}. Czas t_{pHL} jest nazywany **czasem propagacji do stanu niskiego.** Jeżeli w odpowiedzi na sygnał wejściowy stan na wyjściu zmieni się z **L** na **H** to opóźnienie po jakim wystąpi zbocze dodatnie na wyjściu oznacza się przez t_{pLH}. Czas t_{pLH} jest nazywany **czasem propagacji do stanu wysokiego.**

Czas propagacji t_p definiuje się jako średnią arytmetyczną czasów t_{pHL} i t_{pLH} (niekiedy jako t_p przyjmuje się większą wartość spośród tych dwóch wartości). Definicję graficzną czasów propagacji przedstawiono na rys. 4.3.



Rys. 4.3. Graficzna definicja czasów propagacji

Czas propagacji jest istotnym parametrem dla projektanta i użytkownika systemów cyfrowych. Wynika z niego minimalny czas trwania impulsu wejściowego, który spowoduje odpowiedź układu. Impulsy wejściowe o czasie trwania krótszym od czasu propagacji nie zostaną przez układ „zauważone". Jeśli to będą impulsy zakłócające, to dłuższy czas propagacji będzie je skuteczniej eliminował (filtrował). Jednak w praktyce wydłużenie czasu propagacji nie jest sposobem na zwiększenie odporności na zakłócenia i przeważnie dąży się do tego, aby parametr ten miał małą wartość. Pozwala to na pracę z większymi częstotliwościami, czyli na przetwarzanie większej ilości informacji w jednostce czasu.

Ścisły związek między czasem propagacji i dopuszczalną częstotliwością pracy sprawia, że zamiast parametru t_p określa się niekiedy **częstotliwość maksymalną f_{max}** (np. w odniesieniu do przerzutników czy liczników).

Złożone układy cyfrowe zawierają zwykle wiele elementów o różnych czasach propagacji. Sygnał wejściowy przechodzi co najmniej przez kilka z nich.

Szybkość działania takiego układu charakteryzuje się maksymalną dopuszczalną częstotliwością pracy.

- ***Straty mocy P_s***

Straty mocy P_s (moc pobierana przez układ, moc rozpraszana) — podawana w katalogach — jest to moc tracona w układzie przy przełączaniu tego układu przebiegiem prostokątnym o wypełnieniu 1/2 i częstotliwości 100 kHz. Dla użytkownika jest to istotny parametr, bo pozwala mu określić zapotrzebowanie układu na energię, a tym samym dobrać odpowiedni zasilacz.

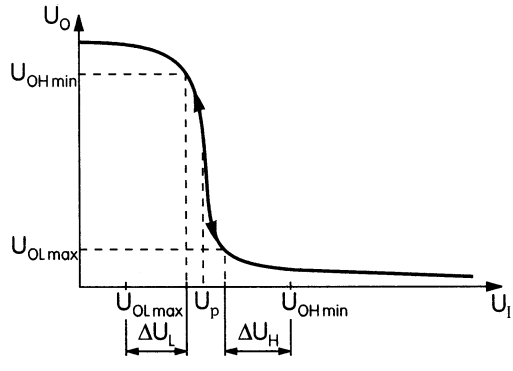
- ***Współczynnik dobroci D***

Dla danej techniki realizacyjnej zwiększenie szybkości pracy odbywa się zwykle kosztem wzrostu mocy traconej w elemencie i odwrotnie — zmniejszenie strat mocy odbywa się kosztem szybkości działania. Iloczyn czasu propagacji t_p i strat mocy P_s jest więc dla danej klasy układów wielkością w przybliżeniu stałą. Iloczyn ten określa się mianem **współczynnika dobroci** i oznacza literą D ($D = t_p P_s$). Umożliwia on porównanie układów należących do różnych klas.

Jeżeli P_s wyrazimy w miliwatach, a t_p w nanosekundach, to D będzie miało wymiar pikodżuli (pJ). Nie jest to jednak parametr charakteryzujący jakąś cechę fizyczną układu, lecz jedynie współczynnik służący do porównywania różnych serii układów cyfrowych.

- ***Margines zakłóceń***

Zmiany sygnału wyjściowego układu cyfrowego mogą być wywołane nie tylko sygnałem użytecznym, ale również sygnałem zakłócającym. W katalogach odporność na zakłócenia charakteryzuje parametr nazywany **marginesem zakłóceń ΔU. Margines zakłóceń ΔU *jest to maksymalna wartość sygnału, która dodana do sygnału wejściowego elementu* (pochodzącego z wyjścia poprzedniego elementu) *nie spowoduje przekroczenia przez sygnał wyjściowy dopuszczalnych granic.*** Marginesy zakłóceń można określić na podstawie charakterystyki przejściowej, jeżeli znamy przyjęte dla danej klasy układów wartości progowe napięć wejściowych i wyjściowych w stanie **H** i **L**. Na rysunku 4.4 przedstawiono sposób



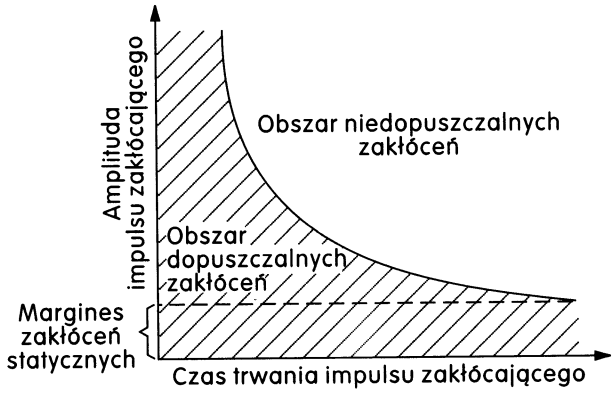
Rys. 4.4. Charakterystyka przejściowa negatora. Sposób wyznaczenia marginesów zakłóceń

U_I — napięcie wejściowe, U_O —napięcie wyjściowe, $U_{OL\,max}$ — maksymalne napięcie wyjściowe w stanie niskim, $U_{OH\,min}$ — minimalne napięcie wyjściowe w stanie wysokim

wyznaczania marginesów zakłóceń na podstawie charakterystyki przejściowej negatora. Aby tak wyznaczone parametry odnieść do całej klasy układów, należałoby przeanalizować całą rodzinę charakterystyk i przyjąć minimalne wartości jako ostateczne. W rzeczywistości bowiem charakterystyki przełączania bramki zawierają się w pewnych obszarach. Zależą one od takich czynników, jak: napięcie zasilania, temperatura, starzenie się elementów. Marginesy zakłóceń należy zatem określić dla najbardziej niekorzystnych warunków działania bramki.

Parametrem impulsu zakłócającego decydującym o jego „skuteczności" jest nie tylko jego amplituda, ale także i czas trwania. Ze względu na czas trwania impulsów zakłócających można zakłócenia podzielić na statyczne i dynamiczne. Jako granicę pomiędzy nimi przyjmuje się czas propagacji sygnału przez bramkę. **Zakłócenia dynamiczne** ***to takie, których czas trwania nie przekracza wartości*** t_p. ***Impulsy trwające dłużej to*** **zakłócenia statyczne.**

Jeżeli czas trwania impulsu zakłócającego jest mniejszy od czasu propagacji sygnału przez bramkę, to dopuszczalna wartość maksymalna tego impulsu może być większa niż w przypadku zakłóceń statycznych.



Rys. 4.5. Zależność amplitudy od czasu trwania impulsu zakłócającego

Uogólniając powyższe uwagi można powiedzieć, że o zakłócającym działaniu danego impulsu decyduje jego energia. Wartość tej energii jest proporcjonalna do pola powierzchni zawartej pod krzywą przebiegu czasowego impulsu zakłócającego.

Zależność dopuszczalnej amplitudy impulsu zakłócającego od jego czasu trwania przedstawiono na rys. 4.5.

- ***Obciążalność N***

Obciążalność ***N wyjścia układu cyfrowego jest to dopuszczalna liczba wejść innych elementów (tej samej lub określonej serii), które mogą być z tego wyjścia prawidłowo sterowane (tzn. bez przekroczenia katalogowych wartości prądów i napięć).*** Z powyższej definicji wynika, że jest to wielkość bezwymiarowa.

Znajomość parametrów charakteryzujących układy cyfrowe pozwala na porównanie kilku najbardziej rozpowszechnionych klas (tabl. 4.2).

Tablica 4.2. **Parametry wybranych układów cyfrowych**

Parametr \ Klasa		TTL	ECL	PMOS	NMOS	CMOS
Czas propagacji t_p	ns	3÷33	1÷2	35÷300	15÷150	5÷50
Straty mocy P_s	mW	1÷23	25÷60	0,5÷1,5	1	0,00001 1 (f=1MHz)
Współczynnik dobroci $D = t_p P_s$	pJ	19÷138	50÷60	50÷150	15÷150	0,00001 50 (f=1MHz)
Margines zakłóceń ΔU	V	1	0,2	0,7÷1,5	1	$0{,}3U_{CC}$
Częstotliwość f_{max}	MHz	3÷100	150÷550	2÷8	7÷18	5÷125
Liczba nap. zas.		1	1	2÷3	1÷3	1 2÷6
U_{zas}	V	5±5%	−5÷2	−27÷+5	−15÷+15	3÷18
Asortyment układowy	—	bardzo duży	średni	mały LSI	mały LSI	bardzo duży

Pytania i zadania

1. Wyjaśnij znaczenie następujących symboli: U_{CC}, U_{IH}, U_{IL}, U_{OH}, U_{OL}, I_{IH}, I_{IL}, I_{OH}, I_{OL}, I_{CCH}, I_{CCL}, I_{OS}, t_{pHL}, t_{pLH}, t_p.
2. Wymień podstawowe parametry scalonych układów cyfrowych.
3. Jakim parametrem jest charakteryzowana szybkość działania układu?
4. Czy czas opóźnienia przełączenia bramki zależy od stanu w jakim się ona znajduje w chwili zmiany sygnału wejściowego?
5. Zdefiniuj pojęcie czasu propagacji t_{pHL}.
6. Zdefiniuj pojęcie czasu propagacji t_{pLH}.
7. Jak jest definiowany średni czas propagacji t_p?
8. Dlaczego czas propagacji powinien być możliwie mały?
9. Czy istnieją jakieś powody, dla których dłuższy czas propagacji byłby korzystniejszy?
10. Jakim parametrem (zamiast czasu t_p) może być charakteryzowana szybkość działania układu?
11. Do czego jest potrzebna projektantowi systemu cyfrowego znajomość strat mocy P_s?
12. W jakim celu jest używany parametr zwany współczynnikiem dobroci?
13. Co to jest margines zakłóceń?
14. Jakie parametry impulsu zakłócającego decydują o skuteczności jego oddziaływania na układ?
15. Narysuj charakterystykę określającą dopuszczalną amplitudę impulsu zakłócającego w zależności od czasu jego trwania.
16. Co oznacza obciążalność $N = 10$?

5

Technika TTL

5.1. Wprowadzenie

Do najbardziej rozpowszechnionych układów cyfrowych wykonanych w monolitycznej technice bipolarnej należą układy zrealizowane w technice tranzystorowo-tranzystorowej TTL (ang. *Transistor Transistor Logic*).

W technice TTL są produkowane obecnie następujące serie:

- podstawowa — 74,
- Schottky'ego — 74S (ang. *Schottky*),
- Schottky'ego małej mocy — 74LS (ang. *Low power Schottky*),
- szybka — 74F (ang. *Fast*),
- ulepszona Schottky'ego małej mocy — 74ALS (ang. *Advanced Low power Schottky*),
- ulepszona Schottky'ego — 74AS (ang. *Advanced Schottky*).

Można jeszcze spotkać układy serii o dużej szybkości (ale wolniejsze od serii 74F) — 74H (ang. *High speed*) oraz układy serii małej mocy 74L (ang. *Low power*). Serii tych obecnie już się nie produkuje.

Podstawowe parametry techniczne układów TTL różnych serii zestawiono w tabl. 5.1.

Tablica 5.1. **Podstawowe parametry techniczne bramek TTL**

Parametr \ Seria		74	74S	74LS	74F	74ALS	74AS
Czas propagacji t_p	ns	10	3	9	3,5	5	1,7
Straty mocy P_s	mW	10	19	2	5	1	8
Współczynnik dobroci $D = t_p P_s$	pJ	100	57	18	18	5	13,6
Częstotliwość f_{max}	MHz	35	125	40	125	50	150

W skład poszczególnych serii układów wchodzą układy o różnym stopniu scalenia. Są produkowane układy: małego stopnia scalenia (SSI) — zawierające bramki logiczne i przerzutniki; układy średniego stopnia scalenia (MSI) — zawierające układy kombinacyjne (multipleksery, demultipleksery, dekodery, sumatory, komparatory itp.) i układy sekwencyjne (liczniki, rejestry itp.), a także układy dużego stopnia scalenia (LSI), np. pamięci, mikroprocesory.

Aktualnie najbardziej są rozpowszechnione serie 74, 74S i 74LS. Parametry techniczne tych serii zestawiono w tabl. 5.2.

Tablica 5.2. **Parametry techniczne układów TTL**

Parametr \ Seria		74	74S	74LS
U_{CC}	V	5±5%		
$U_{CC\,max}$	V	7		
U_{IL}	V	−0,5÷+0,8	−1,2÷+0,8	−1,5÷+0,8
U_{OL}	V	0÷0,4	0÷0,5	0÷0,5
U_{IH}	V	2÷5,5		
U_{OH}	V	2,4÷5	2,7÷5	2,7÷5
$I_{OL\,max}$	mA	16	20	8
t_{pLH}	ns	8	3	10
t_{pHL}	ns	12	3	9
N	—	10	10	20
ΔU_L	V	0,4	0,3	0,3
ΔU_H	V	0,4	0,7	0,7

Są to parametry typowe czyli występujące przy napięciu zasilania $U_{CC} = 5$ V i temperaturze otoczenia 25°C. Dokładne wartości tych parametrów oraz układy do ich pomiaru znajdują się w katalogach dostarczanych przez firmy produkujące te elementy.

Na schematach omawianych układów będą podawane wartości znamionowe rezystancji. Należy jednak pamiętać, że rezystory wykonywane technologią monolityczną mają rozrzut wartości rezystancji dochodzący do 20%.

W analizie pracy tych układów będziemy zakładać, że spadek napięcia na krzemowym złączu PN[1)] spolaryzowanym w kierunku przewodzenia wynosi 0,7 V, a spadek napięcia między kolektorem a emiterem tranzystora nasyconego 0,2 V.

[1)] Według PN-88/T-01102 *Przyrządy półprzewodnikowe i układy scalone. Terminologia* obowiązują oznaczenia: półprzewodnika typu N (uprzednio *n*), półprzewodnika typu P (uprzednio *p*), złącza PN (uprzednio *p-n*).

Przyjmujemy również, że prądy wpływające do bramki (do wejścia lub do wyjścia) mają zwrot dodatni (znak +), a prądy wypływające z bramki (z wejścia lub wyjścia) mają zwrot ujemny (znak –).

W każdej serii układów cyfrowych są budowane bramki:

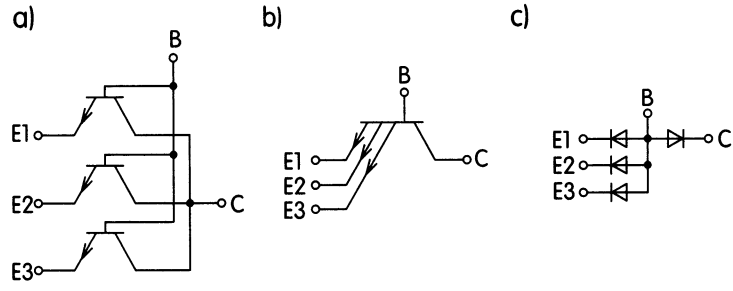
1) standardowe z wyjściem:
   a) przeciwsobnym (ang. *totem pole*),
   b) OC — otwarty kolektor (ang. *Open-Collector*),
2) buforowe (mocy) z wyjściem:
   a) przeciwsobnym,
   b) OC — otwarty kolektor.

5.2. Podstawowa bramka TTL serii standardowej 74

Podstawową i powszechnie stosowaną bramką jest bramka NAND (jest ona przecież SFP — patrz p. 3.2, pozwala w sposób bezpośredni zrealizować układ na podstawie minimalnej normalnej postaci sumy — patrz p. 3.4). Dlatego opis typowych cech, budowy obwodów wejściowych czy wyjściowych, charakterystyk statycznych itp. będzie dotyczyć w niniejszym podręczniku właśnie tego typu bramek. A najczęściej opisywaną bramką będzie dwuwejściowa bramka NAND.

- ***Układ wejściowy bramki logicznej TTL serii standardowej***

Układ wejściowy bramki logicznej iloczynowej (AND, NAND) zrealizowanej w technice TTL jest zbudowany przy użyciu tzw. **tranzystora wieloemiterowego**. Jest to układ tylu tranzystorów o połączonych bazach oraz kolektorach, ile wynosi liczba wejść. W wersji scalonej takiego układu odpowiednie obszary baz i kolektorów są także połączone, co daje w efekcie strukturę określaną jako tranzystor wieloemiterowy. Na rysunku 5.1 przedstawiono schemat tranzystora wieloemiterowego jako układu połączeń zwykłych tranzystorów, jego symbol graficzny oraz schemat zastępczy ułatawiający zrozumienie zasady jego działania.

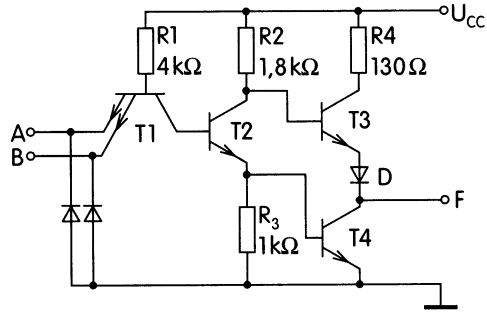


Rys. 5.1. Układ wejściowy bramki logicznej iloczynowej TTL: a) układ połączeń; b) symbol graficzny; c) schemat zastępczy

- ***Podstawowa bramka TTL (7400)***

Układ 7400 zawiera w swej obudowie cztery dwuwejściowe bramki NAND. Jest to układ 14-nóżkowy. Po trzy (dwie wejściowe, jedna wyjściowa) nóżki na każdą z czterech bramek plus dwie na doprowadzenie napięcia zasilającego.

Schemat ideowy bramki NAND pokazano na rys. 5.2.



Rys. 5.2. Schemat bramki NAND (7400)

Stopień wejściowy bramki stanowi tranzystor wieloemiterowy *T1*. Tranzystor *T2* jest podstawowym elementem wzmacniacza pośredniczącego (inwertera), a tranzystory *T3* i *T4* — stopnia wyjściowego (tzw. **wzmacniacza przeciwsobnego**). Do wejść bramki są dołączone diody, które tłumią oscylacje powstałe w liniach łączących bramki w czasie ich przełączania i zapobiegają powstawaniu ujemnych napięć o wartości większej niż ok. 0,7 V.

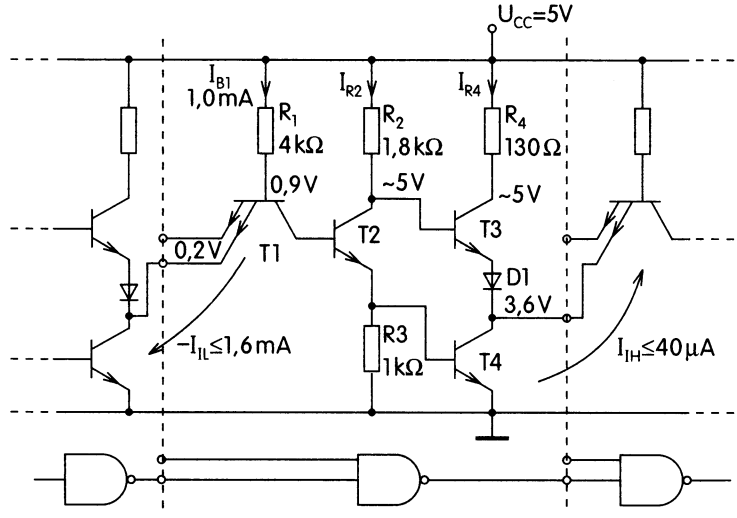
Wszystkie tranzystory (poza tranzystorem *T3*) w tym układzie — w stanach ustalonych, czyli przy nie zmieniających się sygnałach wejściowych i wyjściowych o wartościach zawartych w dopuszczalnych przedziałach — pracują w charakterze kluczy tranzystorowych, tzn. każdy z nich znajduje się w jednym z dwóch stanów pracy: w stanie nasycenia lub w stanie odcięcia (zatkania). Pamiętanie o tym bardzo ułatwia analizę pracy układu.

- **Bramka TTL NAND w stanie H (bramka wyłączona)**

Na rysunku 5.3 przedstawiono schemat bramki w stanie wysokim **H** na wyjściu. Dla uproszczenia pominięto diody przyłączone do wejść.

Przy napięciu wejściowym (co najmniej jednym) o wartości odpowiadającej poziomowi niskiemu **L** (U_{IL}= −0,5÷+0,8V) z wejścia bramki wypływa prąd o wartości typowej 1 mA ($I_{IL\,max}$ = −1,6 mA). Prąd ten wpływa do elementu, z którego jest sterowana analizowana bramka. Tranzystor *T1* znajduje się w stanie nasycenia. Na bazie tranzystora *T2* występuje napięcie wejściowe powiększone o napięcie $U_{CE\,nas}$ (0,2 V) nasyconego tranzystora *T1*, czyli $U_{B(T2)} = U_{IL} + U_{CE\,nas} \leq$ 1V. Napięcie to jest wystarczające, aby wprowadzić tranzystor *T2* w stan pracy aktywnej, ale jednocześnie zbyt małe, aby uaktywnić także tranzystor *T4*, do czego potrzeba napięcia $U_{B(T2)}$ większego niż 1V (ponieważ spolaryzować musi ono dwa złącza krzemowe: BE_{T2} i BE_{T4}). Praca aktywna tranzystora *T2* sprawia, że napięcie $U_{C(T2)} = U_{B(T3)}$ maleje, co zmniejsza wysterowanie tranzystora *T3*, a w konsek-

wencji prowadzi do obniżenia napięcia wyjściowego. Jednak napięcie wyjściowe U_O jest nadal większe niż 2,4 V, co oznacza, że bramka jest w stanie wysokim. Dla typowych napięć U_I (o poziomie **L**, $U_{IL} < 0{,}4$ V) napięcie bazy tranzystora *T3* jest wysokie (zbliżone do napięcia zasilającego $U_{CC} = 5$ V), co zapewnia dobre wysterowanie tranzystora *T3* w kierunku przewodzenia. Tranzystor *T3* jest w stanie aktywnym (przewodzenia) i na wyjściu ustala się napięcie, którego typowa wartość wynosi 3,6 V.



Rys. 5.3. Napięcia i prądy bramki TTL NAND w stanie H

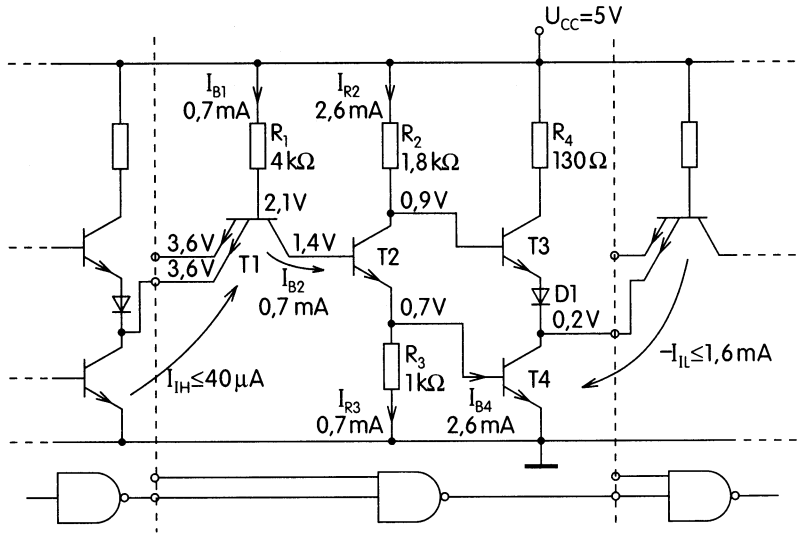
Analiza pracy bramki nie ulegnie zmianie, jeżeli oba wejścia będą w stanie niskim lub tylko jedno w stanie wysokim, a drugie w stanie niskim.

- **Bramka TTL NAND w stanie L (bramka włączona)**

Na rysunku 5.4 przedstawiono schemat bramki w stanie niskim **L** na wyjściu.

Przy napięciach wejściowych (obu wejść) o wartości odpowiadającej poziomowi wysokiemu **H** ($U_{IH} = 2 \div 5$V) do wejścia (każdego) bramki wpływa prąd o wartości 40 μA ($I_{IH} \leq 40$ μA). Jest to prąd kolektora tranzystora *T1* pracującego w połączeniu inwersyjnym. Złącze *BE* tranzystora *T1* jest spolaryzowane zaporowo, a złącze *BC* w kierunku przewodzenia. Prąd I_{IH} (czyli prąd kolektora tranzystora *T1*) ma niewielką wartość, ponieważ $\beta_{inw} \ll \beta$.

Prąd złącza *BC* tranzystora *T1* plus prądy wejściowe stanowi prąd wpływający do bazy tranzystora *T2*, który dzięki temu znajduje się w stanie nasycenia. Część prądu emiterowego tranzystora *T2* wpływa do bazy tranzystora *T4*, nasycając go. Na bazie tranzystora *T3* występuje napięcie $U_{B(T3)} = 0{,}9$V ($U_{B(T3)} = U_{BE(T4)} + U_{CE\,nas(T2)} = 0{,}7 + 0{,}2$). Napięcie to jest za małe, aby wysterować tranzystor *T3*, który znajduje się wobec tego w stanie zatkania. Stan odcięcia tranzystora *T3* uzyskujemy dzięki diodzie *D1*. Konsekwencją umieszczenia diody *D1* jest po prostu konieczność spolaryzowania w kierunku przewodzenia dwóch (BE_{T3} i AK_{D1}), a nie jednego (BE_{T3}) złącza w celu wprowadzenia tranzystora *T3* w stan



Rys. 5.4. Napięcia i prądy bramki TTL NAND w stanie L

aktywny. Napięcie $U_{B(T3)}$ = 0,9 V jest zbyt małe, aby spolaryzować te dwa złącza w kierunku przewodzenia.

Opisane powyżej dwa stany pracy (stan włączenia i wyłączenia) bramki TTL NAND dotyczą stanów ustalonych. To znaczy sygnał wejściowy nie ulega zmianie, a sygnał wyjściowy już się ustalił po ostatniej zmianie sygnału wejściowego. W każdym z tych stanów jeden z tranzystorów (*T3* lub *T4*) stopnia końcowego jest w stanie przewodzenia, drugi zaś w stanie zatkania.

Stopień końcowy jest **przeciwsobnym wzmacniaczem wyjściowym** (ang. *totem-pole output*) charakteryzującym się małą statyczną i dynamiczną rezystancją wyjściową zarówno w stanie **H**, jak i **L**. Wzmacniacz taki zapewnia dużą obciążalność oraz krótki czas narastania sygnału przy obciążeniu pojemnościowym (przy małej rezystancji wyjściowej może popłynąć duży prąd ładowania kondensatora).

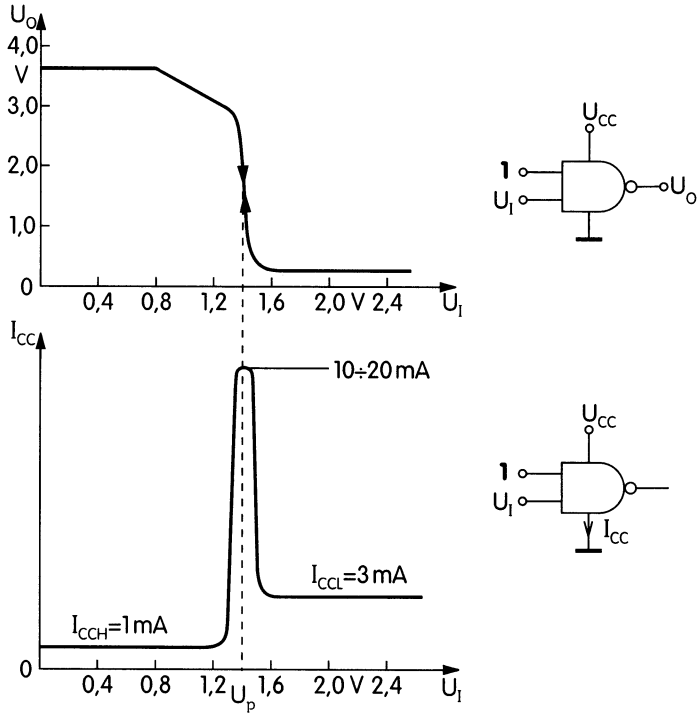
- **Przełączanie bramki TTL NAND**

Przy analizowaniu procesu przełączania bramki należy pamiętać, że tranzystor krzemowy zaczyna przewodzić przy napięciu U_{BE} wynoszącym ok. +0,5÷+0,6 V. W stanie nasycenia napięcie to wynosi ok. +0,7 V.

Załóżmy, że na jednym wejściu bramki poziom napięcia jest wysoki (**H**), a na drugim niski (**L**) czyli, że bramka jest w stanie wysokim (na wyjściu poziom **H**). Przeanalizujmy zachowanie się układu, gdy wartość sygnału wejściowego odpowiadająca poziomowi **L** zmienia się powoli w kierunku poziomu **H**. Założono powolną zmianę sygnału wejściowego, aby można było pominąć w analizie wpływ opóźnień wnoszonych przez poszczególne elementy.

Punktem wyjścia jest zatem bramka w stanie **H** przedstawiona na rys. 5.3. Tranzystor *T1* jest nasycony, a tranzystory *T2* i *T4* są w stanie zatkania. Aby tranzystory *T2* i *T4* były w stanie aktywnym, to napięcie $U_{BE(T4)} + U_{BE(T2)} = U_{B(T2)}$

musi osiągnąć wartość co najmniej +1,0 V. Wynika stąd, że napięcie wejściowe U_I musi wzrosnąć do wartości około +0,8 V, gdyż $U_I + U_{CE(T1)nas} = U_{B(T2)}$, czyli 0,8 + 0,2 = 1. Przy takim właśnie napięciu zaczyna najpierw przewodzić tranzystor *T2*, który przechodzi w obszar pracy aktywnej. Pracuje on jako prosty wzmacniacz w układzie wspólnego emitera OE z ujemnym prądowym sprzężeniem zwrotnym o wzmocnieniu napięciowym $k_u = R_2/R_3$ = 1,6 V/V. Powoduje to niewielkie nachylenie charakterystyki przejściowej (rys. 5.5). Wzrost napięcia wejściowego do wartości ok. +1,2V powoduje wprowadzenie tranzystora *T4* w stan przewodzenia. Przewodzące złącze BE tranzystora *T4* bocznikuje rezystancję R_3, powodując gwałtowne zmniejszenie się rezystancji emiterowej tranzystora *T2* i równie gwałtowny wzrost jego wzmocnienia napięciowego k_u. Odpowiada to fragmentowi charakterystyki (rys. 5.5) o dużej stromości.



Rys. 5.5. Charakterystyka przejściowa $U_O = f(U_I)$ oraz charakterystyka poboru prądu przez bramkę $I_{CC} = f(U_I)$

Następuje zmiana polaryzacji tranzystora *T1*, który zaczyna pracować w trybie inwersyjnym. Tranzystory *T2* i *T4* wchodzą w stan nasycenia, powodując odcięcie tranzystora *T3*. W czasie przełączania tranzystorów *T3* i *T4* występuje stan ***jednoczesnego*** przewodzenia obu tranzystorów. Jest to przyczyną gwałtownego wzrostu poboru prądu ze źródła. Prąd płynący w stopniu końcowym jest ograniczony jedynie niewielką rezystancją $R_4 \approx 130\ \Omega$ oraz rezystancjami dynamicznymi przewodzących tranzystorów *T3*, *T4* i może osiągnąć w szczycie wartość ok. 20 mA.

Opisane procesy będą przebiegać identycznie (tylko w odwrotnej kolejności) przy zmianach sygnału wejściowego z poziomu **H** na **L**.

Omówione zjawiska można przedstawić graficznie za pomocą charakterystyk: **przejściowej (przełączania)** $U_O = f(U_I)$ oraz **poboru prądu** $I_{CC} = f(U_I)$. Dla podkreślenia ich współzależności podano obie na jednym rysunku (rys. 5.5), sytuując je nad sobą.

Przebieg charakterystyki przełączania po tych samych punktach zarówno przy wzrastających, jak i malejących wartościach napięcia wejściowego U_I ***ma bardzo istotne konsekwencje praktyczne.*** Załóżmy, że sygnał wejściowy zmienia się powoli i rozpatrywany przez nas przedział czasu to ten, w którym napięcie wejściowe ma wartość ok. +1,4 V. Pamiętamy, że tranzystor *T2* znajduje się wówczas w obszarze pracy aktywnej. Z analizy procesu przełączania bramki, jak i z przebiegu charakterystyki przejściowej wynika, że dla takiego napięcia wejściowego stan wyjścia zmienia się na niski **L**. W wyniku dodatniego sprzężenia zwrotnego poprzez niewielkie pojemności pasożytnicze (około 1 pF) oraz indukcyjność (nie do uniknięcia zarówno w samej strukturze scalonej TTL, jak i przy montażu) napięcie wejściowe maleje, a wyjście ponownie jest przestawiane w stan **H**. To samo sprzężenie zwrotne powoduje teraz wzrost napięcia wejściowego, przejście bramki w stan **L** itd. Efektem są oscylacje o dużej częstotliwości (około 20 MHz), a o bramce mówimy, że się **wzbudziła** (zachowuje się jak generator).

Zastanówmy się jeszcze, jak należy rozumieć mało precyzyjne określenie „powolna zmiana sygnału". Przyjmijmy, że sygnał wejściowy zmienia swą wartość od 0 V do +5 V (liniowo) w czasie 0,1 ms. Z prostych wyliczeń wynika, że zmiana napięcia wejściowego o 0,2 V (np. od +1,3 do +1,5 V) trwa 4 µs. Bramka znajduje się więc w stanie aktywnym (przełączania) co najmniej kilka mikrosekund, a w rzeczywistości zakres napięć wejściowych, przy których bramka jest w stanie aktywnym, jest szerszy niż przyjęte przez nas 0,2 V. W czasie tych 4 µs bramka może (teoretycznie) przełączyć ok. 400 razy (przy t_p= 10 ns; 4 µs/10 ns = 400). W praktyce liczba tych przełączeń będzie znacznie mniejsza, ale i tak będzie ona uniemożliwiać poprawną pracę układu.

Doświadczalnie stwierdzono, że efektu generacji można uniknąć przestrzegając warunku, by szybkość zmian sygnału wejściowego była nie mniejsza niż **1 V/µs**.

W przypadku wolniejszych przebiegów wejściowych należy korzystać z układów z wejściami charakteryzującymi się histerezą przy przełączaniu, czyli bramek Schmitta (np. 74132). Będzie jeszcze o nich mowa w dalszej części podręcznika.

Również istotny, z uwagi na praktyczne skutki, jest **przebieg prądu pobieranego przez bramkę** — charakterystyka $I_{CC} = f(U_I)$ na rys. 5.5. Wynika z niej że każdorazowemu przełączeniu bramki TTL towarzyszy impulsowy (o względnie dużej amplitudzie) pobór prądu ze źródła. Skutki, jakie powoduje, są analogiczne do tych, jakie w skali makro można zaobserwować w chwili załączenia silnika indukcyjnego do sieci. Niemal każdy z nas miał chyba okazję zaobserwować w takim momencie przygaśnięcie żarówek na chwilę i ich powrót do normalnej jasności, gdy silnik zwiększył obroty (wartość prądu rozruchowego zmalała kilkakrotnie w stosunku do początkowej wartości). Mechanizm powstania tego zakłóce-

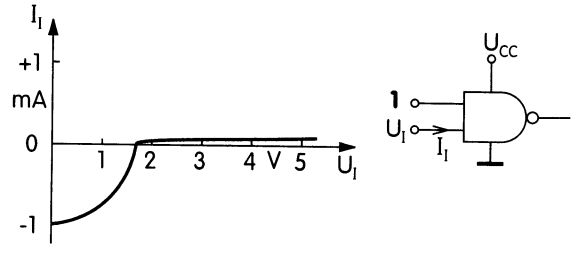
nia pracy żarówek jest następujący: duży impuls prądowy powoduje duży spadek napięcia na impedancji źródła zasilania, w efekcie czego maleje napięcie zasilające żarówkę. Wszystkie odbiorniki podłączone do tego samego źródła zasilania będą narażone na skutki obniżenia się napięcia zasilającego. Elementem pośredniczącym w przenoszeniu się zaburzeń w funkcjonawaniu odbiorników jest źródło zasilania.

Zjawisko takie (również w odniesieniu do układów TTL) określamy mianem **sprzęgania się poprzez źródło** lub krótko **sprzęganiem** się układów. Skutki sprzęgania się układów TTL są bardzo niekorzystne. Objawiają się one przypadkową zmianą stanu bramek, zasilanych z tego samego źródła co przełączana bramka. Taka przypadkowa zmiana stanu w przypadku przerzutnika ma jeszcze ten mankament, że zostaje w nim zapamiętana (przerzutnik jest elementem pamięciowym). ***Aby zapobiec skutkom sprzęgania się układów stosuje się tzw.* kondensatory odsprzęgające (blokujące)** — zwane tak od funkcji jaką spełniają w układzie. ***Są to bezindukcyjne kondensatory ceramiczne o niewielkiej pojemności. Powinny być one przylutowywane do ścieżek masy i źródła napięcia U_{CC} jak najbliżej elementów blokowanych.*** Zazwyczaj każde pięć układów SSI blokuje się kondensatorem o pojemności 0,01÷0,1 μF. Każdy układ MSI przeważnie wymaga własnego kondensatora blokującego. Kondensatory te podczas zmian stanów bramek dostarczają wymaganego prądu, który dzięki temu nie musi być pobierany ze źródła, co eliminuje możliwość sprzęgania się układów poprzez źródło.

● Charakterystyka wejściowa bramki TTL NAND

Zachowanie się obwodu wejściowego bramki NAND przy różnych napięciach wejściowych zostało już częściowo omówione przy okazji analizowania wszystkich trzech stanów pracy: stanu **H**, stanu **L** i stanu przełączania. Obecnie uzupełnimy te informacje o sytuacje wcześniej nie uwzględnione oraz zbierzemy w całość.

Charakterystykę wejściową $I_I = f(U_I)$ przedstawiono na rys. 5.6.



Rys. 5.6. Charakterystyka wejściowa $I_I = f(U_I)$

Jeżeli na wejściu bramki poziom napięcia jest niski, to złącze *BE* tranzystora wejściowego *(T1* na rys. 5.3) jest spolaryzowane w kierunku przewodzenia. Obwodem zastępczym układu wejściowego jest wówczas dioda spolaryzowana w kierunku przewodzenia z połączonym szeregowo rezystorem R_1. Wartość płynącego prądu jest ograniczona przez ten właśnie rezystor. Wartość znamionowa

tego prądu $I_{IL} = -1$ mA, natomiast jego wartość maksymalna (w odniesieniu do wartości bezwzględnej) $I_{IL\,max} = -1{,}6$ mA.

Ze wzrostem wartości napięcia wejściowego bezwzględna wartość prądu wejściowego maleje tak długo, aż zostanie osiągnięty próg przełączania bramki. Wówczas złącze *BE* tranzystora *T1* zostaje spolaryzowane zaporowo (potencjał kolektora jest mniejszy niż potencjał emitera) i tranzystor *T1* przechodzi do pracy w trybie inwersyjnym. Prąd I_I zmienia swój kierunek i obecnie wpływa do bramki, ale jest to niewielki prąd, którego wartość nie przekracza 40 μA. W zakresie dopuszczalnych napięć odpowiadających stanowi wysokiemu wartość tego prądu jest praktycznie stała. Dopiero przekroczenie przez napięcie wejściowe poziomu 5,5 V powoduje istotny wzrost wartości prądu wejściowego, ale grozi także uszkodzeniem obwodu wejściowego bramki (złącza PN).

W podsumowaniu warto zwrócić uwagę, że wejście bramki TTL może pobierać prąd z układu sterującego tym wejściem albo go do niego wysyłać. Układ sterujący musi więc umożliwiać przepływ tego prądu w obu kierunkach. Czy powyższe warunki spełnia układ wyjściowy bramki TTL, zostanie rozstrzygnięte przy omawianiu charakterystyk wyjściowych bramki.

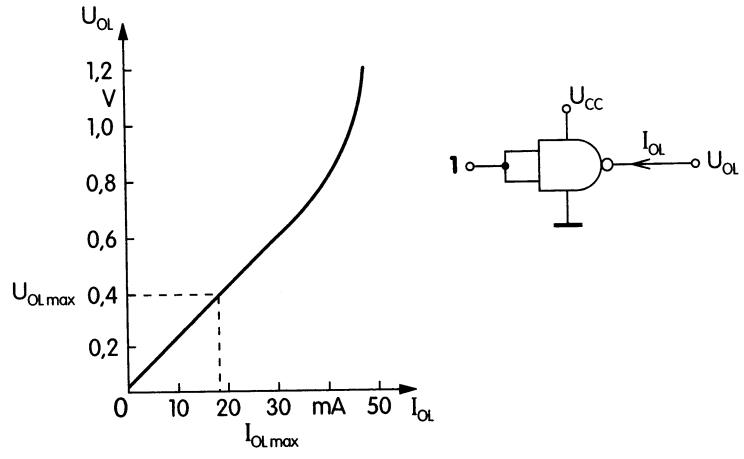
• Charakterystyki wyjściowe bramki TTL NAND

Bramka TTL ma dwie charakterystyki wyjściowe, dla każdego stanu oddzielnie, określające zależność napięcia wyjściowego U_O od prądu wyjściowego I_O. Są to zależności $U_{OL} = f(I_{OL})$ oraz $U_{OH} = f(I_{OH})$.

1. Charakterystyka $U_{OL} = f(I_{OL})$

Charakterystykę wyjściową bramki TTL NAND w stanie **L** pokazano na rys. 5.7. Do wyjścia bramki (dokładnie do nasyconego tranzystora *T4*, patrz rys. 5.4) wpływa prąd o wartości określonej przez układ dołączony do tego wyjścia.

Prąd wpływający do wyjścia bramki w stanie **L** (I_{OL}) nie powinien spowodować wzrostu napięcia wyjściowego powyżej wartości 0,4 V (dla układów TTL serii 74 $U_{OL\,max} = 0{,}4$ V — patrz tablica 5.2). Maksymalna wartość tego prądu

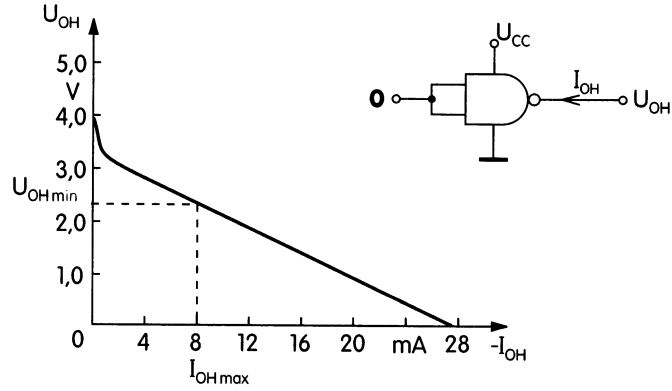


Rys. 5.7. Charakterystyka wyjściowa $U_{OL} = f(I_{OL})$

$I_{OL\,max}$ = 16 mA, co odpowiada obciążeniu wnoszonemu przez 10 standardowych bramek TTL (10 · $I_{IL\,max}$ = 10 · 1,6 mA = 16 mA). Przy zwiększaniu wartości prądu wpływającego do bramki powyżej 16 mA należy liczyć się z tym, że przy pewnej wartości tego prądu (przekraczającej 16 mA) tranzystor *T4* wyjdzie ze stanu nasycenia i napięcie wyjściowe wzrośnie powyżej dopuszczalnej wartości 0,4 V.

2. Charakterystyka $U_{OH} = f(I_{OH})$

Na rysunku 5.8 przedstawiono charakterystykę $U_{OH} = f(I_{OH})$ standardowej bramki TTL z wyjściem przeciwsobnym. Bramka znajduje się w stanie wysokim (rys. 5.3) i wobec tego tranzystor *T3* jest aktywny, a tranzystor *T4* jest zatkany. Napięcie wyjściowe jest napięciem U_{CC} pomniejszonym o: spadek napięcia na rezystancji R_4 (≈ 130 Ω), napięcie U_{CE} tranzystora *T3* oraz napięcie anoda-katoda diody *D1*.



Rys. 5.8. Charakterystyka $U_{OH} = f(I_{OH})$

Ze wzrostem prądu I_{OH} maleje $U_{CE\,T3}$, aż do wejścia tranzystora *T3* w nasycenie (odpowiada to początkowemu fragmentowi charakterystyki o dużej stromości). Wówczas napięcie U_{CE} (tranzystora *T3*) ustala się (na wartości ok. 0,2 V). Ponieważ także napięcie U_{AK} (diody *D1*) jest w przybliżeniu stałe (niezależne od prądu I_O, przeto wyjście układu zachowuje się analogicznie jak wyjście rzeczywistego źródła napięcia, co potwierdza przebieg charakterystyki $U_{OH} = f(I_{OH})$ na rys. 5.8. Z jej przebiegu można odczytać także, że w zakresie zmian prądu $-I_{OH}$ = = 0÷8 mA napięcie wyjściowe bramki nie maleje poniżej dopuszczalnej w stanie **H** wartości 2,4 V ($U_{OH\,min}$ = +2,4 V — patrz tablica 5.2). Jednocześnie pamiętajmy, że jedno wejście bramki przy wysokim poziomie napięcia wejściowego pobiera prąd o wartości $I_{IH} \leq 40\,\mu A$. Aby więc zapewnić obciążalność N = 10, wyjście rozpatrywanego układu powinno umożliwiać pobór prądu o wartości − 400 μA. (***Pamiętajmy, że zgodnie z przyjętą na początku tego rozdziału konwencją prądy wypływające z bramki mają zwrot ujemny.***) Występuje tu więc olbrzymi zapas mocy wyjściowej. Obciążalność N = 10 (tablica 5.2) jest więc determinowana przez obciążalność w stanie niskim.

Zauważmy, że połączenie ze sobą wejść bramki, na które jest podawane napięcie w stanie **L**, nie zwiększa pobieranego z tej bramki prądu (rys. 5.4). Jednocześnie dysponujemy zapasem mocy wyjściowej w stanie **H**. Dzięki temu możemy sformułować bardzo ważny wniosek: ***Łączenie ze sobą kilku wejść tej samej bramki nie zwiększa pobieranego prądu z elementu sterującego. Należy jednak pamiętać, że mówimy o bramce mającej wejście w postaci tranzystora wieloemiterowego, co w układach TTL dotyczy jedynie bramek iloczynowych.*** Wprawdzie w stanie wysokim każde dodatkowe wejście to zapotrzebowanie na dodatkowe 40 μA, ale jest to tak mała wartość w odniesieniu do $I_{OH\,max} = 8$ mA, że może zostać pominięta. Fakty te mają duże znaczenie praktyczne. Uzyskuje się dzięki temu możliwość dołączania nie wykorzystywanych wejść do wejść wykorzystywanych. Takie połączenie nadal możemy traktować jako obciążenie równe 1, a obciążalność $N = 10$ należy teraz rozumieć jako obciążenie dziesięcioma wejściami ***różnych*** bramek.

Charakterystyka $U_{OH} = f(I_{OH})$ przedstawiona na rys. 5.8 obejmuje pełny zakres obciążeń, od braku obciążenia (prąd równy zeru), aż do maksymalnego obciążenia (stanu zwarcia — prąd maksymalny I_{OS}). W stanie zwarcia płynie prąd ograniczony rezystancją R_4 (oraz niewielką impedancją tranzystora *T3* i diody *D*), a jego bezwzględna maksymalna wartość wynosi ok. 25 mA. Pamiętamy, że rozrzut wartości rezystancji w układach scalonych jest dość duży. Stąd i wartość tego prądu (jako zależna od wartości rezystancji R_4) może zawierać się w granicach $18 \div 55$ mA.

Stan zwarcia wyjścia bramki nie grozi jej uszkodzeniem, jeżeli jest to jedyna bramka w tym układzie scalonym, której wyjście zwarto do masy. Powyższa uwaga ma istotne znaczenie praktyczne. Wynika z niej, że ***wolno zewrzeć w testowanym układzie dowolne wyjście do masy (jedno na układ scalony)***. Potrzeba taka często zachodzi w procesie uruchamiania układów cyfrowych.

Pytania i zadania

1. Wymień zalety układów TTL.
2. Wymień i nazwij produkowane serie układów TTL.
3. Jaki jest dopuszczalny zakres zmienności napięcia wejściowego w stanie niskim **L** dla układów TTL?
4. Jaki jest dopuszczalny zakres zmienności napięcia wejściowego w stanie wysokim **H** dla układów TTL?
5. Jaki jest dopuszczalny zakres zmienności napięcia wyjściowego w stanie niskim **L** dla układów TTL?
6. Jaki jest dopuszczalny zakres zmienności napięcia wyjściowego w stanie wysokim **H** dla układów TTL?
7. Jakim napięciem należy zasilać układy TTL?
8. Jak jest zbudowany obwód wejściowy bramki iloczynowej TTL?
9. Na podstawie schematu ideowego bramki TTL NAND wymień podstawowe obwody i elementy wchodzące w jej skład.
10. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) omów rozpływ prądów i rozkład napięć, gdy bramka znajduje się w stanie wysokim (na wyjściu). Oblicz typową wartość prądu I_{IL}.

11. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) omów rozpływ prądów i rozkład napięć, gdy bramka znajduje się w stanie niskim (na wyjściu).
12. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) określ, jaki będzie stan na wyjściu układu przy braku sygnałów wejściowych (tzn. wejścia „w powietrzu"). Jakiemu stanowi wejść odpowiada ta sytuacja?
13. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) omów i uzasadnij przebieg charakterystyki przejściowej $U_O = f(U_I)$.
14. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) omów i uzasadnij przebieg charakterystyki poboru prądu przez bramkę ($I_{CC} = f(U_I)$). Oblicz prądy I_{CC} dla bramki włączonej i wyłączonej oraz oszacuj wartość maksymalną tego prądu, jaka wystąpi w chwili przełączania.
15. Jakie skutki może mieć impulsowy pobór prądu ze źródła przez bramkę w momencie jej przełączania?
16. Omów zjawisko sprzęgania się układów TTL.
17. W jaki sposób można przeciwdziałać sprzęganiu się układów?
18. Na podstawie przykładowej charakterystyki $U_O = f(U_I)$ (np. z rys. 5.5) oraz danych zawartych w tablicy 5.2 wyznacz marginesy zakłóceń bramki TTL NAND w stanie **H** i **L**.
19. Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) określ wpływ połączenia ze sobą obu wejść bramki na wartość prądu wypływającego z bramki, gdy wejście jest w stanie **L** (czyli wpływ na wartość prądu I_{IL}). Odpowiedź uzasadnij.
20. Czy połączenie ze sobą wejść jednej bramki zwiększa obciążenie wyjścia bramki sterującej tymi wejściami, gdy bramka sterująca znajduje się w stanie: a) **H**; b) **L**?
21. W jakim stanie (**H** czy **L**) prąd wyjściowy może mieć większą wartość (w odniesieniu do wartości bezwzględnej)?
22. Jakie wartości (bezwzględne) mają prądy $I_{IH\,max}$, $I_{IL\,max}$, $I_{OH\,max}$, $I_{OL\,max}$?
23. Jakie skutki może wywołać sterowanie bramki TTL zbyt wolno zmieniającym się sygnałem wejściowym? Jaką szybkość narastania (opadania) powinien mieć sygnał wejściowy?
24. Duża stromość charakterystyki przełączania bramki TTL może powodować (przy wolno narastających/opadających sygnałach wejściowych) zjawisko wzbudzania się bramki. Pogorszenie jakich parametrów spowodowałoby dużo mniej strome ukształtowanie tej charakterystyki?
25. Czy jest dopuszczalne zwarcie wyjścia bramki TTL do masy układu? Na podstawie schematu ideowego bramki TTL NAND (rys. 5.2) oszacuj wartość prądu zwarciowego I_{OS}.
26. Wyjaśnij precyzyjnie, jak w odniesieniu do iloczynowych (AND, NAND) bramek TTL należy rozumieć parametr: $N = 10$.

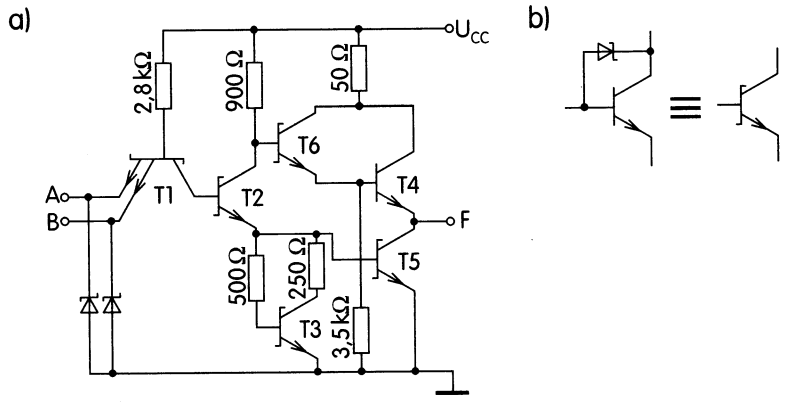
5.3. Bramki podstawowe innych serii

Schematy ideowe podstawowych bramek serii: 74S, 74LS, 74F, 74ALS, 74AS różnią się od schematu ideowego bramki standardowej (rys. 5.2) i to (szczególnie w najnowszych seriach) w sposób bardzo istotny.

- **Seria Schottky'ego 74S**

W standardowej bramce TTL istotny wpływ na ograniczenie szybkości przełączania ma praca tranzystorów w stanie nasycenia. Uniemożliwienie pracy tran-

zystorów w obszarze nasycenia pozwala znacznie przyspieszyć działanie bramki. Niedopuszczenie do pracy tranzystorów w stanie nasycenia osiąga się za pomocą diod Schottky'ego. Diodę taką charakteryzuje bardzo krótki czas przełączania oraz napięcie przewodzenia ok. 0,4 V. Połączenie diody Schottky'ego z tranzystorem w układzie jak na rys. 5.9b zapobiega nasyceniu tranzystora. Konfiguracja taka jest nazywana **tranzystorem Schottky'ego.** Schemat ideowy bramki TTL-S przedstawiono na rys. 5.9a. W układzie zastosowano diody Schottky'ego na wejściach oraz w obwodzie baza-kolektor wszystkich tranzystorów z wyjątkiem tranzystora *T4*, który nie wchodzi w nasycenie.



Rys. 5.9. Schemat zasadniczy bramki TTL NAND serii Schottky'ego (a) oraz tranzystor Schottky'ego (b)

Istotną zmianą w układzie z rys. 5.9 jest zastosowanie układu aktywnego z tranzystorem *T3* zamiast rezystora. Dzięki tej zmianie charakterystyka przejściowa (przełączania) bramki jest bardziej prostokątna, co zwiększa odporność bramki na zakłócenia. Zwiększenie odporności na zakłócenia odnosi się jedynie do stanu wysokiego. W stanie niskim odporność ta nieco maleje, gdyż napięcie wyjściowe $U_{OL\,max}$ wynosi 0,5 V (tranzystor *T5* nie wchodzi w nasycenie), a bramka zaczyna przełączać już przy napięciu wejściowym wynoszącym ok. 1,2 V. Tylko dzięki stromej charakterystyce przełączania znamionowa odporność na zakłócenia w stanie niskim pozostaje bez zmian. Tranzystor *T3* zwiększa ponadto szybkość przełączania tranzystora *T5*, co powoduje skrócenie czasu trwania impulsu prądowego pobieranego ze źródła w momencie przełączania bramki. Przy dużych częstotliwościach pracy ma to istotny wpływ na ograniczenie poboru prądu przez bramkę.

Układy TTL z diodami Schottky'ego mają następujące zalety:

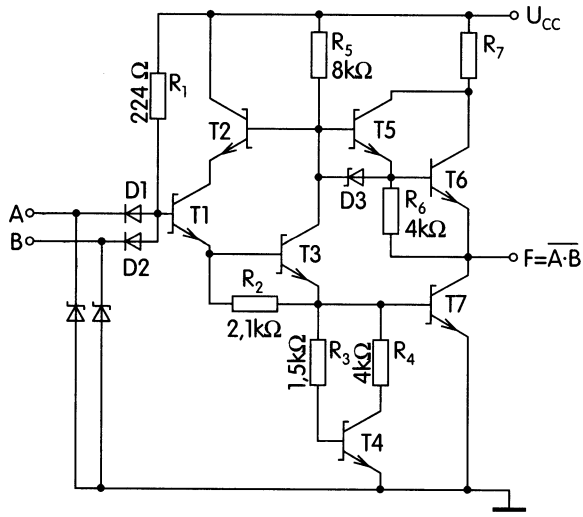
1. Czas propagacji (2÷3)-krotnie mniejszy niż w układach serii standardowej. Czas t_p serii 74S wynosi ok. 3÷4 ns. Jest to wartość porównywalna z wartością czasu t_p szybkich układów ECL.
2. Mniejszy pobór mocy podczas przełączania. Tranzystory nie pracują w stanie nasycenia, dzięki czemu ich przełączanie wymaga mniej energii. Jednak pobór mocy w stanach statycznych jest dla bramek serii 74S większy niż dla

serii standardowej. Wynika to z mniejszych wartości rezystancji R wchodzących w skład bramek serii S.

3. Obcinanie ujemnych napięć (wejściowych) na poziomie 0,4 V, co skuteczniej eliminuje przepięcia niż w układach z serii standardowej.

- **Seria Schottky'ego małej mocy 74LS**

Schemat ideowy bramki TTL NAND serii 74LS przedstawiono na rys. 5.10. W układzie bramki Schottky'ego małej mocy dwuemiterowy tranzystor wejściowy został zastąpiony dwiema diodami (w niektórych wykonaniach — diodami Schottky'ego) *D1*, *D2*. Diody te wraz z rezystorem R_1 tworzą klasyczną diodową bramkę iloczynu (patrz rys. 4.1b). Podobnie jak we wcześniej omówionej bramce TTL-S, także i ta bramka ma aktywny układ R_3, R_4, *T4* w obwodzie emitera tranzystora *T3*, zapewniający prostokątną charakterystykę przełączania bramki oraz zwiększający szybkość przełączania tranzystora *T7*. Układ Darlingtona w obwodzie wyjściowym (tranzystory *T5*, *T6*) zapewnia dobre parametry dynamiczne układu.



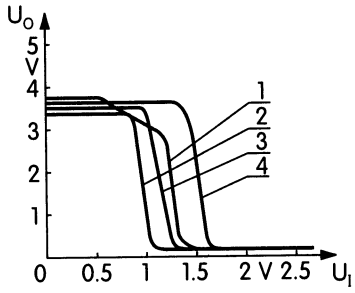
Rys. 5.10. Schemat zasadniczy bramki TTL NAND serii Schottky'ego małej mocy

Bramka ta zaczyna przełączać przy napięciu wejściowym wynoszącym ok. 1 V. Tranzystor *T2* i dioda *D3* zwiększają szybkość działania układu. Charakterystyki wejściowe i wyjściowe bramki są podobne do odpowiednich charakterystyk bramek serii standardowej. Prąd $I_{IL\,max}$ = 0,36 mA, a $I_{IH\,max}$ = 20 μA. Średni czas propagacji bramki t_p = 7 ns. Pobór mocy przez bramkę P_s= 2 mW. Efektem zastąpienia tranzystora wieloemiterowego układem diodowym jest, poza zwiększeniem szybkości przełączania bramki i zmniejszeniem wartości prądów wejściowych, zwiększenie napięcia przebicia na wejściu układu do wartości ok. 15 V. Dlatego nie używane wejścia można bezpośrednio łączyć ze źródłem napięcia zasilania.

Wszystkie parametry bramek serii 74LS są lepsze od odpowiednich parametrów bramek seri standardowej 74. Biorąc pod uwagę, że ceny obu serii są porównywalne, staje się oczywiste coraz częstsze stosowanie układów serii 74LS.

- **Serie ALS, AS i FAST**

Obecnie wszystkie nowo opracowane serie bipolarnych układów TTL bazują na tranzystorach i diodach Schottky'ego. Celem tych opracowań jest zwiększenie szybkości działania układów i zmniejszenie poboru mocy. Nowe udoskonalone serie to 74AS, 74ALS i 74F (FAST). Schematy elektryczne podstawowych bramek tych serii są dość złożone i nie będą tu prezentowane ani szczegółowo analizowane. Zainteresowani mogą je znaleźć w katalogach lub np. w publikacji [5]. Podstawowe parametry tych serii zestawiono w tabl. 5.1. Na rysunku 5.11 zestawiono charakterystyki przełączania podstawowych bramek serii 74, 74LS, 74F i 74ALS. Wynika z nich, że napięcie przełączania układów TTL wynosi 1 V ÷ 1,7 V, co stanowi 20 ÷ 35% U_{CC}.



Rys. 5.11. Charakterystyki przejściowe (przełączania) podstawowych bramek TTL serii:
1 — 74; *2* — 74LS; *3* — 74ALS; *4* — 74F

Pytania i zadania

1. Czym różni się dioda Schottky'ego od zwykłej diody krzemowej?
2. Jak jest zbudowany i czym się charakteryzuje tranzystor Schottky'ego?
3. Wymień zalety układów z diodami Schottky'ego.
4. Wymień i omów podstawowe różnice w budowie obwodów wejściowych i wyjściowych bramek TTL NAND serii 74 i 74LS.
5. Porównaj parametry techniczne bramek serii 74, 74LS, 74ALS, 74AS, 74F. Wyciągnij wnioski z tego porównania.

5.4. Inne rodzaje bramek TTL

5.4.1. Wprowadzenie

Jak wiadomo (patrz p. 3.2) bramka NAND jest funktorem funkcjonalnie pełnym. Oznacza to, że dowolny układ kombinacyjny możemy zbudować używając wyłącznie tych bramek. Jednak w praktyce możliwość korzystania również z bramek logicznych realizujących inne funkcje ułatwia projektowanie i niejednokrotnie umożliwia uzyskiwanie lepszych rozwiązań — układów oszczędniejszych, o mniejszej liczbie połączeń, o większej szybkości działania. Dlatego też opracowano wiele innych bramek, takich jak: NOT, NOR, AND, OR, Ex-OR. Liczba wejść bramek może być inna niż 2. Dotyczy to także bramki NAND. Bramki wie-

lowejściowe mogą mieć 2, 3, 4 lub 8 wejść. Bramki NOT mają oczywiście tylko jedno wejście, natomiast bramki Ex-OR i EX-NOR są budowane wyłącznie jako dwuwejściowe.

Na przykład:

7404 — układ zawierający 6 bramek NOT,
7410 — układ zawierający 3 trójwejściowe bramki NAND,
7420 — układ zawierający 2 czterowejściowe bramki NAND,
7430 — układ zawierający 1 ośmiowejściową bramkę NAND,
7408 — układ zawierający 4 dwuwejściowe bramki AND,
7402 — układ zawierający 4 dwuwejściowe bramki NOR,
7427 — układ zawierający 3 trójwejściowe bramki NOR,
7486 — układ zawierający 4 dwuwejściowe bramki Ex-OR,

oraz wiele innych, jednak liczba wejść może wynosić: 2, 3, 4 lub 8 (i 1, ale oczywiście tylko dla negatora).

Schematy ideowe tych bramek różnią się oczywiście od schematu bramki NAND, ale obwody wejściowe oraz wyjściowe są identyczne. To znaczy obwód wejściowy (dla bramek iloczynowych) jest wykonany jako tranzystor wieloemiterowy z diodami ograniczającymi napięcia ujemne i tłumiącymi oscylacje, natomiast obwód wyjściowy w postaci wzmacniacza przeciwsobnego. Charakterystki wejściowe, wyjściowe i przejściowe będą więc identyczne lub analogiczne. Inna będzie jedynie realizowana funkcja logiczna. Zasady wykorzystywania (łączenia, obciążania, blokowania) będą takie jak dla bramki NAND.

5.4.2. Rodzaje bramek z serii standardowej (bramki specjalne)

W ramach serii standardowej UCY74 są wytwarzane także bramki specjalne, różniące się od powyżej opisanej bramki podstawowej budową obwodów wejściowych lub wyjściowych. Rola jaką spełniają one w układach cyfrowych TTL wynika z ich budowy i zostanie omówiona poniżej.

- **Bramka z układem Schmitta**

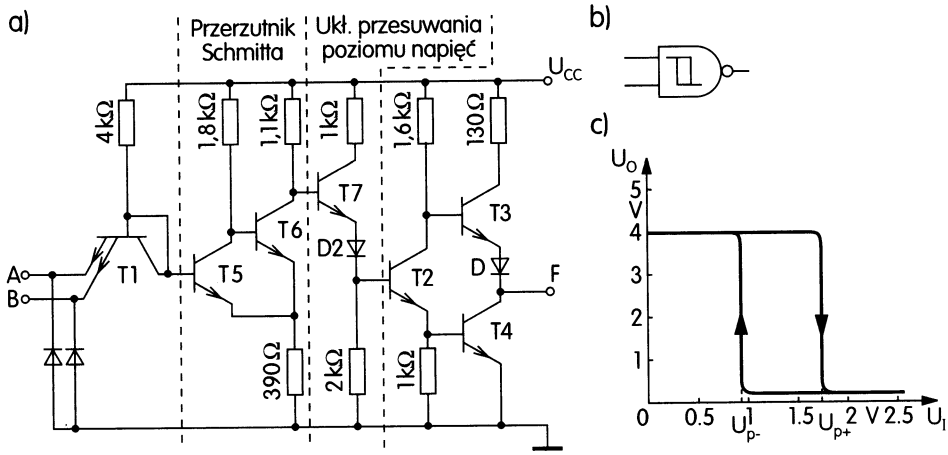
Bramka z układem Schmitta zostanie omówiona na przykładzie bramki NAND (np. 74132). Produkowane są także i inne bramki logiczne zawierające przerzutnik Schmitta (np. 7414 — 6 bramek NOT).

Schemat elektryczny bramki NAND z układem Schmitta ('132) przedstawiono na rys. 5.12. W jednej obudowie znajdują się cztery identyczne (dwuwejściowe) bramki NAND.

Na schemacie z rysunku 5.12 można wyodrębnić następujące układy:
— układ wejściowy (prawie identyczny jak w zwykłej bramce TTL),
— układ wyjściowy (identyczny jak w zwykłej bramce TTL)

oraz dodatkowe obwody (w odniesieniu do układu 7400):

— przerzutnik Schmitta,
— układ przesuwania poziomu napięcia.



Rys. 5.12. Bramka NAND z układem Schmitta: a) schemat zasadniczy; b) symbol graficzny; c) charakterystyka przełączania
U_{p+} — napięcie przełączania przy rosnącym napięciu wejściowym, U_{p-} — napięcie przełączania przy malejącym napięciu wejściowym

Ze względu na analogię do układu z rys. 5.2 zachowano numerację tranzystorów w obwodzie wejściowym i wyjściowym, oznaczając zaś kolejnymi liczbami tranzystory występujące w dodatkowych obwodach.

Przełączanie bramki następuje przy przekraczaniu napięcia ok. 1,7 V ***przy narastaniu sygnału wejściowego*** i przy napięciu ok. 0,9 V ***przy opadaniu sygnału.*** Widać to na rys. 5.12c, na którym pokazano charakterystykę przejściową tej bramki.

Histerezę przełączeniową uzyskuje się dzięki temu, że omawiana bramka (rys. 5.12) w porównaniu z bramką podstawową (rys. 5.2) zawiera dodatkowo układ przerzutnika Schmitta (tranzystory *T5*, *T6*).

Układ przesuwania poziomu (tranzystor *T7* i dioda *D2*) dopasowuje poziomy napięć pomiędzy wyjściem układu Schmitta a układem wyjściowym bramki. Ponadto, ze względu na dopasowanie poziomów i dla zmniejszenia progu przełączania, na wejściu układu zwarto złącze *BC* tranzystora *T1*.

Analizę pracy układu przy stanie **H** lub **L** na wyjściu pozostawia się Czytelnikowi. Analogicznie jak dla bramki podstawowej można i tutaj przyjąć, że tranzystory znajdują się w jednym z dwóch stanów pracy: nasycenia lub odcięcia.

Najważniejszą jednak cechą bramki z układem Schmitta jest jej histereza przełączeniowa. Zwykła bramka nie ma takiej histerezy i jak wykazano uprzednio, przy wolno zmieniającym się napięciu wejściowym bramka taka może generować falę o częstotliwości ok. 20 MHz. Bramka Schmitta przełącza przy napięciu przełączania $U_{p+} \approx 1{,}7$ V. Występujące w niej dodatnie sprzężenie zwrotne (podobnie jak w bramce podstawowej) zmniejsza sygnał wejściowy. Ale, aby bramka ponownie przełączyła, napięcie wejściowe musiałoby się zmniejszyć do wartości 0,9 V, czyli aż o ok. 0,8 V. Dzięki bowiem pętli histerezy próg przełączania przy zboczu opadającym przebiegu wejściowego U_{p-} to ok. 0,9 V. Wynika z tego, że napięcie wejściowe może się długo utrzymywać na poziomie 1,7 V i nie będzie to grozić wzbudzeniem się bramki, jeśli zawiera ona układ Schmitta.

Bramki z układem Schmitta są stosowane na wejściach układów cyfrowych współpracujących z sygnałami wolnozmiennymi o parametrach (amplituda, stromość zboczy) wykraczających poza standard TTL. Sygnały w **standardzie TTL** to takie, których szybkość narastania bądź opadania napięcia jest większa niż 1 V/μs, a amplituda zawiera się w granicach 2 ÷ 5 V.

Warto jednak zauważyć, że czas propagacji bramki Schmitta jest znacznie dłuższy niż analogiczny czas bramki standardowej. Zwraca się Czytelnikowi uwagę na ten fakt, gdyż często większą stromość charakterystyki przejściowej bramki Schmitta utożsamia się z większą szybkością działania — co, jak widać, jest błędnym wnioskiem. Wolniejsze działanie bramki z układem Schmitta jest spowodowane tym, że jest ona zbudowana z większej liczby elementów. Ta właśnie wada bramki Schmitta sprawia, że nie jest ona stosowana powszechnie, a jedynie w obwodach wejściowych.

Duża stromość charakterystyki przełączania bramki Schmitta (rys. 5.12) w połączeniu z histerezą przełączeniową jest cechą korzystną i pożądaną, ponieważ pozwala uzyskać większe marginesy zakłóceń. Jest to dodatkowa (poza odpornością na wzbudzanie się) zaleta tej bramki.

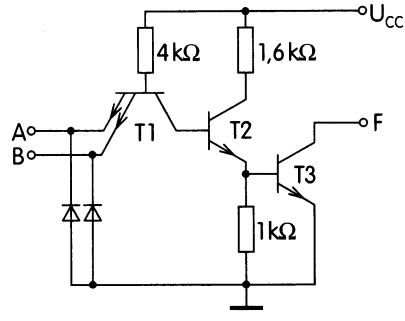
● Bramka z otwartym kolektorem

Schemat ideowy bramki z otwartym kolektorem (typu OC — ang. *Open Collector*)) z układu scalonego 7401 przedstawiono na rys. 5.13.

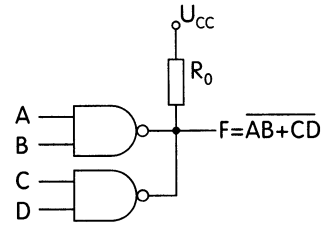
Gdy wyjście F zostanie dołączone do napięcia zasilającego U_{CC} przez rezystor R, wówczas układ realizuje funkcję NAND ($\mathbf{F = \overline{AB}}$). Bramka typu OC z dołączonym rezystorem nie różni się funkcjonalnie niczym od bramki standardowej TTL NAND. Jej czas propagacji t_{pLH} jest jednak znacznie dłuższy, natomiast czas t_{pHL} jest taki sam jak w bramce standardowej. Spowodowane jest to (wydłużenie) ładowaniem pojemności obciążenia poprzez rezystor R.

Jedną z zalet bramek z otwartym kolektorem jest możliwość łączenia ze sobą ich wyjść. Z bramkami standardowymi tak postępować nie wolno.

Układ pracy dwóch bramek typu OC z połączonymi wyjściami przedstawiono na rys. 5.14. Takie połączenie do wspólnego rezystora daje w efekcie iloczyn doprowadzonych sygnałów. A zatem $\mathbf{\overline{AB}\,\overline{CD} = \overline{AB+CD}}$.

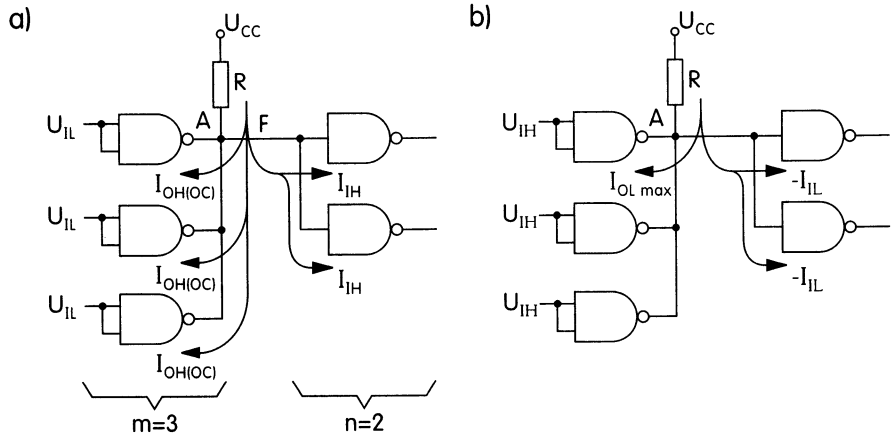


Rys. 5.13. Schemat zasadniczy bramki TTL NAND typu OC (7401)



Rys. 5.14. Układ pracy dwóch bramek NAND typu OC z połączonymi wyjściami

Liczba m bramek typu OC, których wyjścia będą ze sobą łączone, może być w dość szerokim zakresie dowolna ($m \leq 7$). Także liczba wejść n sterowanych przez to wyjście ($n \leq 10$). Dla każdej takiej konfiguracji (pary m, n) należy dobrać inną wartość rezystora obciążającego R. Wartość tej rezystancji powinna być tak dobrana, aby w stanie wysokim na wyjściu napięcie wyjściowe było nie mniejsze niż $U_{OH\,min} = 2{,}4$ V, natomiast w stanie niskim nie większe niż $U_{OH\,max} = 0{,}4$ V. W każdym z tych przypadków należy — w celu wyznaczenia wartości tej rezystancji — oszacować prąd przez nią płynący, przyjmując najbardziej niekorzystne warunki pracy. Rozpływ prądów w przypadku, gdy $m = 3$ i $n = 2$ przedstawiono na rys. 5.15.



Rys. 5.15. Rozpływ prądów w układzie bramek typu OC połączonych do wspólnego rezystora: a) w stanie **H** na wyjściu; b) w stanie **L**

Całkowity prąd płynący przez rezystor R w stanie **H** na wyjściu będzie sumą prądów wejściowych sterowanych bramek i prądów płynących przez zatkane tranzystory wyjściowe bramek z otwartym obwodem kolektora. Wartość prądu wpływającego do bramki TTL w stanie **H** wynosi $I_{IH} = 40$ μA. Największa wartość (gwarantowana przez producenta) prądu upływności tranzystora wyjściowego (*T3* na rys. 5.13) wynosi $I_{OH(OC)} = 250$ μA. W tym przykładzie wartość (maksymalna) prądu płynącego przez rezystor R wynosi więc $mI_{OH(OC)} + nI_{IH} = (3 \cdot 250 + 2 \cdot 40)$ μA = = 0,83 mA. Prąd ten wywoła spadek napięcia na rezystancji R. Spadek tego napięcia nie może być zbyt duży, aby napięcie wyjściowe nie zmalało poniżej dopuszczalnej wartości $U_{OH\,min} = 2{,}4$ V. Dopuszczalna wartość spadku tego napięcia wyniesie więc $U_{CC} - U_{OH\,min} = 5 - 2{,}4 = 2{,}6$ V. A zatem $R_{max} \cdot 0{,}83$ mA = 2,6 V, stąd $R_{max} = 3{,}13$ kΩ. Ogólnie, zależność na rezystancję R_{max} można przedstawić w postaci

$$R_{max} = \frac{U_{CC} - U_{OH\,min}}{mI_{OH(OC)} + nI_{IH}} \tag{5.1}$$

Przy obliczaniu wartości minimalnej rezystancji R najmniej korzystny przypadek występuje wówczas, gdy tylko jedna bramka typu OC jest w stanie niskim, z czego wynika maksymalna wartość prądu wpływającego do węzła A (niezależna od liczby m tych bramek) $I_{\mathrm{OL\,max}}$= 16 mA (wartość identyczna jak dla standardowych bramek TTL). Z bramki TTL, w stanie niskim na wejściu, wypływa prąd o wartości maksymalnej (w odniesieniu do wartości bezwzględnej) $I_{\mathrm{IL\,max}}$= 1,6 mA. Zwrot tego prądu jest ujemny (wypływa z bramki), a zatem minimalna wartość prądu płynącego przez rezystor R wynosi $I_{\mathrm{OL\,max}} - nI_{\mathrm{IL\,max}}$ = 16 mA –2 · 1,6 mA = =12,8 mA. Prąd ten powinien wywołać spadek napięcia na rezystancji R co najmniej taki, aby napięcie wyjściowe nie wzrosło powyżej wartości dopuszczalnej dla układów TTL $U_{\mathrm{OL\,max}}$ = 0,4 V, czyli $U_{\mathrm{CC}} - U_{\mathrm{OL\,max}}$ = 5 V – 0,4 V = 4,6 V. Warunek zostanie spełniony, jeżeli tak dobierzemy rezystancję R_{min}, aby R_{min} · 12,8 mA = = 4,6 V, a zatem R_{min} = 640 Ω. W sposób ogólny zależność na rezystancję minimalną można przedstawić następująco:

$$R_{\mathrm{min}} = \frac{U_{\mathrm{CC}} - U_{\mathrm{OL\,max}}}{I_{\mathrm{OL\,max}} - nI_{\mathrm{IL\,max}}} \tag{5.2}$$

Dla konkretnego układu połączeń (danych m i n) należy przeliczyć ponownie zależności (5.1) i (5.2), wyznaczając w ten sposób zakres rezystancji, które mogą być zastosowane w danym przypadku. Przy wyborze rezystancji z zakresu od R_{min} do R_{max} należy kierować się takimi przesłankami, jak: moc tracona w rezystancji R oraz czas propagacji sygnału.

W praktyce często zachodzi konieczność współpracy różnych układów cyfrowych, których wyjścia są połączone wspólnym przewodem — zwanym **magistralą (szyną).** Jednym ze sposobów umożliwiających taką współpracę jest zastosowanie bramek z otwartym obwodem kolektora.

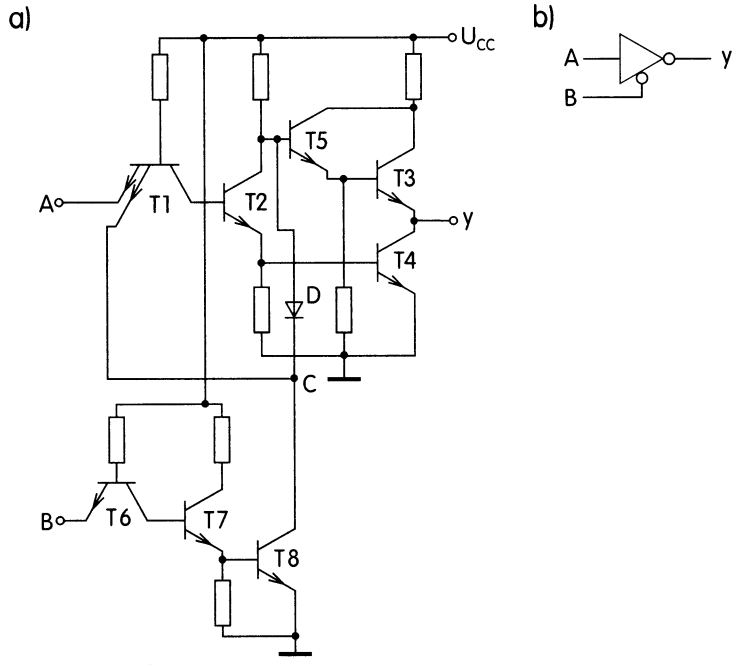
Produkowane są także bramki typu OC o zwiększonej wartości napięcia przebicia tranzystora wyjściowego do 15 V albo 30 V. Bramki te zwykle są bramkami o zwiększonej obciążalności prądowej w stanie niskim **L** na wyjściu. Na przykład układ 7406 zawiera sześć inwerterów typu OC, przy czym maksymalna wartość napięcia wyjściowego w stanie **H** wynosi 30 V, a maksymalna wartość prądu wyjściowego w stanie **L** wynosi 30 mA. Bramki tego typu umożliwiają współpracę typowych układów TTL z układami (niekoniecznie cyfrowymi) o innym napięciu zasilania niż 5 V. Pozwalają także sterować elementami wyjściowymi (np. przekaźnikami) o większym poborze prądu i większym napięciu zasilania niż bramka TTL.

Są produkowane także bramki z otwartym obwodem kolektora, które nie realizują żadnej funkcji logicznej (tzw. **wzmacniacze logiczne, bufory**), których zadaniem jest zwiększenie mocy sygnału wyjściowego i umożliwienie sprzęgnięcia z układami wysokonapięciowymi (np. układ 7407 zawierający sześć wzmacniaczy-buforów z otwartym kolektorem). Symbol graficzny takiej bramki jest identyczny jak symbol negatora pozbawiony kółka na wyjściu.

- **Bramka trójstanowa**

Przyłączanie wyjść elementów do wspólnej magistrali (szyny) umożliwiają również (poza bramkami OC) tzw. **bramki trójstanowe.** Bramka trójstanowa, oprócz dwóch normalnych stanów pracy (**H**, **L**) ma dodatkowy trzeci stan, charakteryzujący się wielką impedancją wyjściową (wyłączenie bramki). Z tego powodu element taki ma dodatkowe wejście sterujące trybem jego pracy.

Schemat ideowy negatora z wyjściem trójstanowym przedstawiono na rys. 5.16. Jeśli na wejściu sterującym **B** jest niski potencjał (stan **L**), to tranzystory *T7*, *T8* układu sterującego są zatkane i bramka zachowuje się tak, jak zwykła bramka negacji.



Rys. 5.16. Negator z wyjściem trójstanowym: a) schemat; b) symbol graficzny

Jeżeli na wejściu sterującym poziom napięcia jest wysoki (stan **H**), to tranzystory *T7*, *T8* są w stanie nasycenia. Potencjał punktu *C* wynosi 0,2 V. W tej sytuacji dioda *D* jest spolaryzowana w kierunku przewodzenia i na bazie tranzystora *T5* występuje potencjał o wartości 0,9 V, co sprawia, że tranzystor *T3* jest zatkany. Odcięte są również tranzystory *T2*, *T4* ze względu na to, że na wejściu tranzystora *T1* połączonym z punktem *C* poziom napięcia jest niski. W efekcie obydwa tranzystory (*T3*, *T4*) w przeciwsobnym wzmacniaczu wyjściowym są w stanie zatkania, co powoduje, że na wyjściu występuje nieokreślony stan charakteryzujący się dużą impedancją wyjściową. Tablicę stanów bramki trójstanowej z rys. 5.16 przedstawiono jako tabl. 5.3.

Tablica 5.3. **Tablica stanów układu z rys. 5.16**

Wejścia		Wyjście
B	**A**	**Y**
L	**L**	**H**
L	**H**	**L**
H	—	(Z)

Stan (Z) oznacza stan wielkiej impedancji. Warto zauważyć, że bramka zachowuje się jak zwykła bramka dwustanowa dla wejścia sterującego **B = 0**, co można poznać po symbolu graficznym tej bramki — kółeczko przy wejściu **B**. Brak takiego kółeczka oznaczałby, że należy doprowadzić do wejścia sterującego poziom **H**, aby bramka taka działała jak zwykła bramka dwustanowa.

Bramki trójstanowe mają dodatkowe parametry dynamiczne (poza takimi, jak dla bramek dwustanowych) charakteryzujące ich działanie. Są to:

1) **Czas zablokowania wyjścia (czas blokowania).** Ponieważ blokowanie takiej bramki może nastąpić zarówno wtedy, gdy znajduje się ona w stanie **H**, jak i w stanie **L**, rozróżnia się:
   - t_{HZ} — czas zablokowania wyjścia przy przejściu ze stanu o poziomie wysokim do stanu wielkiej impedancji,
   - t_{LZ} — czas zablokowania wyjścia przy przejściu ze stanu o poziomie niskim do stanu wielkiej impedancji.
2) **Czas odblokowania wyjścia.** Analogicznie jak dla czasu blokowania:
   - t_{ZL} — czas odblokowania wyjścia przy przejściu ze stanu wielkiej impedancji do stanu o poziomie niskim,
   - t_{ZH} — czas odblokowania wyjścia przy przejściu ze stanu wielkiej impedancji do stanu o poziomie wysokim.

Obwody bramek trójstanowych projektuje się tak, aby czas blokowania bramki (przełączenia w stan wielkiej impedancji) był krótszy od czasu jej otwierania (odblokowania), co jest istotne przy współpracy wielu bramek z magistralą.

- **Bramki z wyjściem mocy**

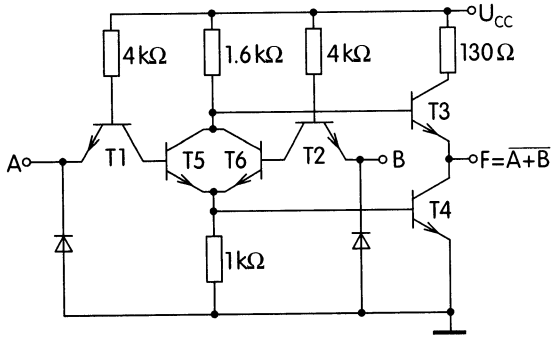
Przy okazji charakteryzowania bramek z otwartym obwodem kolektora wspomniano już o bramkach mocy. Przykładem bramki o zwiększonej obciążalności wyjścia może być układ 7440, zawierający dwie czterowejściowe bramki NAND z wyjściem mocy. W stopniu końcowym bramki zastosowano układ Darlingtona, w wyniku czego zmniejszono rezystancję wyjściową układu i zwiększono prąd obciążenia I_{OH}, przy zachowaniu dużej szybkości przełączania. Obciążalność tej bramki wynosi $N = 30$.

- **Bramka NOR (oraz OR)**

Schemat ideowy bramki NOR — 7402 przedstawiono na rys. 5.17. Wejścia bramki stanowią dwa tranzystory jednoemiterowe *T1* i *T2*, które sterują równolegle połączonymi tranzystorami *T5* i *T6*. Jeżeli na dowolne z wejść (**A** lub **B**) doprowadzimy poziom wysoki, wówczas jeden z tranzystorów *T3* lub *T4* przewodzi,

wprowadzając tranzystor *T6* w stan przewodzenia, co odpowiada stanowi niskiemu na wyjściu. Jeżeli obydwa wejścia (**A** i **B**) są w stanie niskim, to żaden z tranzystorów *T1* i *T2* nie przewodzi, co powoduje, że są zatkane tranzystory *T3* i *T4* oraz tranzystor *T6* i wyjście jest w stanie wysokim.

Obwody wejściowe oraz wyjściowe bramki OR są identyczne jak w bramce NOR. Schemat elektryczny bramki OR różni się (od schematu bramki NOR) jedynie obecnością stopnia negującego, umieszczonego przed wzmacniaczem wyjściowym. Układ wyjściowy w obu bramkach jest taki sam jak w bramce NAND.



Rys. 5.17. Schemat zasadniczy bramki TTL NOR

Technika TTL jest ciągle jeszcze w stadium rozwoju i można oczekiwać nowych rozwiązań. Obecnie produkowany asortyment bipolarnych układów cyfrowych obejmuje ponad 500 pozycji — od prostych układów funkcjonalnych po układy systemów mikroprocesorowych. Mimo tak szerokiej gamy produktów, zakres wykorzystywania bipolarnych układów scalonych (nie tylko TTL) jest ograniczony głównie dużym poborem mocy ze źródła zasilania. Pobór mocy ma szczególnie duże znaczenie przy budowie urządzeń przenośnych i przewoźnych. Istotne oszczędności energii uzyskano dzięki zastosowaniu w układach scalonych tranzystora unipolarnego MOSFET (patrz układy CMOS, rozdz. 6.).

Pytania i zadania

1. Jaka jest różnica w budowie bramki standardowej TTL NAND a bramki TTL NAND z układem Schmitta?
2. Czym różni się charakterystyka przejściowa $U_O = f(U_I)$ bramki Schmitta od charakterystyki przełączania bramki standardowej?
3. Jakie zastosowanie ma bramka z układem Schmitta?
4. Wymień wady i zalety bramki Schmitta.
5. Na jaki parametr układów cyfrowych ma wpływ stromość charakterystyki przełączania?
6. Jaka jest różnica w budowie bramki standardowej TTL NAND a bramki TTL NAND z otwartym kolektorem? Porównaj ich charakterystyki przełączania.
7. Jaka funkcja jest realizowana przez dwie bramki dwuwejściowe NOR typu OC o wyjściach przyłączonych do wspólnego rezystora?

8. Jaka funkcja jest realizowana przez trzy bramki NOT typu OC o wyjściach przyłączonych do wspólnego rezystora?
9. Wyznacz zakres rezystancji obciążenia dla układu trzech ($m = 3$) bramek typu OC, sterujących pięcioma ($n = 5$) wejściami bramek TTL.
10. Jaką wartość rezystancji należy wybrać z zakresu wyznaczonego w zadaniu 9., aby:
    a) uzyskać najmniejsze straty mocy w rezystorze R,
    b) uzyskać najkrótszy czas propagacji?
11. Jakie zastosowanie mają bramki typu OC?
12. Scharakteryzuj bramkę trójstanową i podaj przykład jej zastosowania.

6

Technika CMOS

6.1. Wprowadzenie

Układy CMOS (ang. *Complementary MOS*) zawierają w swej strukturze tranzystory MOSFET (zwane w skrócie tranzystorami MOS) zarówno z kanałem typu N, jak i typu P. Tranzystory te tworzą tak zwane **pary komplementarne**. Każda taka para zawiera tranzystor MOS o przewodnictwie N i tranzystor MOS o przewodnictwie P.

Obecnie wytwarzany asortyment układów CMOS jest podobny do asortymentu układów bipolarnych TTL. Są produkowane serie układów CMOS, będące funkcjonalnymi zamiennikami odpowiednich układów TTL. Topografia wyprowadzeń tych układów jest taka sama (jak w układach TTL), co pozwala stosować je zamiennie. Przykładem takiej serii jest seria 74HC. Seria 74HCT jest całkowicie kompatybilna (zgodna) z serią układów bipolarnych TTL-LS. Tak więc jest możliwa zamiana układów TTL-LS na układy 74HCT — w celu redukcji poboru mocy bez zmniejszenia szybkości działania. Krajowa seria MCY74/64 nie daje takich możliwości ze względu na inną konfigurację wyprowadzeń.

Technologia CMOS jest obecnie dynamicznie rozwijającą się techniką scalania układów cyfrowych (a także i analogowych). Powstało w ostatnich latach wiele odmian tych układów. Mają one coraz lepsze parametry dynamiczne. Dzięki nowym technologiom wytwarzania, umożliwiającym większe upakowanie elementów, są produkowane już układy CMOS wielkiego (LSI) i bardzo wielkiego (VLSI) stopnia scalenia.

W podrozdziale 6.2 scharakteryzowano serię 4000B (produkowaną przez firmy: RCA, Philips oraz Motorola), której krajowym odpowiednikiem jest seria MCY74 produkcji CEMI. Obecnie jednak coraz powszechniej są stosowane serie HC/HCT i AC/ACT, których charakterystykę podano w p. 6.4.

Dla porównania zestawiono w tablicy 6.1 parametry najbardziej obecnie rozpowszechnionych rodzin układów CMOS. W ostatniej kolumnie tablicy przypomniano parametry układów TTL serii 74LS.

Tablica 6.1. **Parametry techniczne różnych serii układów CMOS**

Parametr \ Technologia			4000B (MCY74)	HC HCT	AC ACT	TTL-LS
Napięcie zasilania		V	3÷18	2÷6 HC 5±10% HCT	3÷5,5 AC 5±10% ACT	5±5%
Prąd wejściowy		mA	0,001	0,001	0,001	0,4
Moc statyczna		nW	10	2,5	2,5	2 000 000
Współczynnik dobroci przy f_{CL} = 100 kHz		pJ	0,2	0,02	0,01	20
Czas propagacji przy C_L = 15 pF (poj. obc.)		ns	90	9	5	10
Maksymalna częstotliwość pracy		MHz	12	40	125	40
Obciążalność (liczba wejść bramek TTL-LS)		—	2	10	10	20
Margines zakłóceń	ΔU_L	$\%U_z$	30	28 HC 14 HCT	28 AC 14 ACT	8
	ΔU_H		30	28 HC 58 HCT	28 AC 58 ACT	14

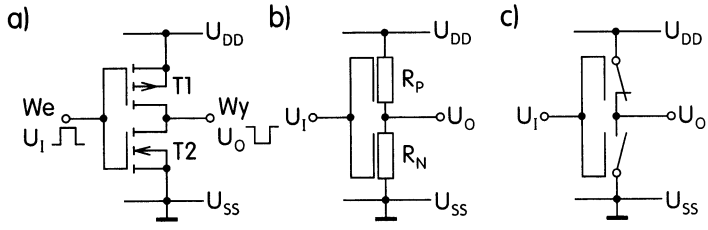
Układy MOS są wrażliwe na przebicia ładunkiem elektrostatycznym. Wynika to ze względnie dużych pojemności występujących na wejściach układów. Izolowana bramka wraz z kanałem stanowi kondensator o dość dużej pojemności ze względu na bardzo małą odległość „okładek" kondensatora. Warstwa tlenku, izolująca bramkę, ma grubość ok. 120 nm. Napięcie wywoływane przez ładunek Q zgromadzony w takim kondensatorze może osiągnąć wartość wystarczającą do przebicia warstwy tlenku i doprowadzić do zniszczenia układu. Stawia to określone wymagania odnośnie do zasad przechowywania i obchodzenia się z układami MOS. ***Nie należy dotykać ich wyprowadzeń. Powinno się je przechowywać z połączonymi ze sobą końcówkami (np. owinięte w folię aluminiową).***

Obecnie buduje się układy, których obwody wejściowe są zabezpieczone przed przebiciem ładunkiem elektrostatycznym. Do takich należą np. układy 4000B (MCY74/64) i oczywiście wszystkie nowsze serie (HC, HCT, AC, ACT). Zabezpieczenia te są jednak skuteczne tylko podczas normalnej eksploatacji układów CMOS, tzn. gdy końcówki zasilania układu są przyłączone do źródła napięcia zasilania. Zatem wyposażenie układów CMOS w obwody zabezpieczenia wejść nie zwalnia nas z obowiązku przestrzegania powyżej wymienionych zasad obchodzenia się z układami CMOS.

6.2. Właściwości układów CMOS. Inwerter CMOS

Podstawową strukturą każdego układu CMOS jest para komplementarna, która jednocześnie stanowi układ realizujący negację sygnału wejściowego, czyli jest funktorem NOT. Bardziej złożone bramki logiczne (NAND, NOR) są kombinacją odpowiednio połączonych ze sobą inwerterów (par komplementarnych). Podstawowe cechy dotyczące działania takiego inwertera są zatem wspólne dla wszystkich serii układów CMOS. Różnice dotyczą wartości parametrów, ale zachowanie się układów oraz ich charakterystyki są podobne. Tak więc uwagi i wnioski sformułowane w p. 6.2 odnoszą się do wszystkich serii układów CMOS.

Schemat inwertera CMOS przedstawiono na rys. 6.1. Para komplementarna elementu CMOS (jest ona podstawową strukturą w tej technologii) połączona jak na rys. 6.1, realizuje funkcję negacji. Jeżeli napięcie wejściowe jest odpowiednio małe (ma wartość U_{IL}), to tranzystor *T2* (z kanałem N) jest zablokowany, a przewodzi tranzystor *T1* (z kanałem P). Napięcie wyjściowe jest w przybliżeniu równe napięciu zasilania. Jeżeli natomiast napięcie wejściowe ma wartość U_{IH}, to tran-

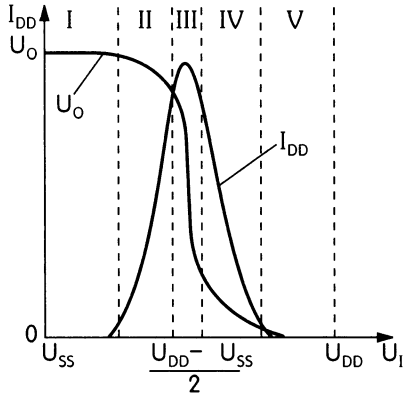


Rys. 6.1. Inwerter CMOS: a) schemat zasadniczy; b) zastępczy układ rezystorowy; c) zastępczy układ stykowy

zystor *T2* jest odblokowany, a tranzystor *T1* zablokowany. Napięcie wyjściowe jest w przybliżeniu równe 0 V. Struktura inwertera jest układem symetrycznym. W warunkach ustalonych, przy dopuszczalnych napięciach wejściowych w stanie **L** bądź **H**, jeden z tranzystorów przewodzi drugi zaś jest zatkany. Tranzystor CMOS w stanie zablokowanym ma bardzo dużą rezystancję (rzędu kilkuset kiloomów), w stanie przewodzenia natomiast niewielką (rzędu kilkuset omów). Pracę inwertera CMOS w stanie ustalonym bardzo dobrze modeluje układ zastępczy przedstawiony na rys. 6.1c. W układzie tym nie można jednak wyjaśnić zachowania się bramki w stanie przejściowym (przełączania).

- **Charakterystyka przełączania oraz poboru prądu**

Działanie negatora można opisać za pomocą charakterystyk statycznych: przejściowej $U_O = f(U_I)$, poboru prądu $I_{DD} = f(U_I)$. Przedstawiono je na rys. 6.2 w jednym układzie współrzędnych, aby lepiej zobrazować ich współzależność. Można w nich wyróżnić pięć obszarów określonych przez stany tranzystorów *T1* i *T2* (tablica 6.2).



Rys. 6.2. Charakterystyka przejściowa $U_O = f(U_I)$ oraz poboru prądu $I_{DD} = f(U_I)$ inwertera CMOS

W celu dokonania analizy poszczególnych obszarów pracy inwertera CMOS (rys. 6.2) przyjmijmy, że jest on zasilany napięciem $U_z = U_{DD} - U_{SS} = 10\,V - 0\,V = = 10\,V$. W obszarze I tranzystor *T2* nie przewodzi, ponieważ napięcie bramki $U_I = U_{GS(T2)}$ jest bliskie zeru. Napięcie bramka-źródło tranzystora *T1* (z kanałem P) wynosi zatem ok. $U_{GS(T1)} = -10\,V$. Oznacza to, że tranzystor *T1* przewodzi, ale ponieważ tranzystor *T2* jest odcięty, więc prąd I_{DD} nie płynie (płynie niewielki prąd upływnościowy o wartości rzędu kilku dziesiątych lub kilku mikroamperów w zależności od wartości napięcia zasilającego). Po wzroście napięcia wejściowe-

Tablica 6.2. **Obszary pracy inwertera CMOS**

Nr obszaru	Zakres napięcia wejściowego	Stan tranzystora	
		T1	*T2*
I	$0 \le U_I \le U_{TN}$	nienasycony	zablokowany
II	$U_{TN} \le U_I \le U_O - \lvert U_{TP} \rvert$	nienasycony	nasycony
III	$U_O - \lvert U_{TP} \rvert \le U_I \le U_O + UTN$	nasycony	nasycony
IV	$U_O + U_{TN} \le U_I \le U_{DD} - \lvert U_{TP} \rvert$	nasycony	nienasycony
V	$U_{DD} - \lvert U_{TP} \rvert \le U_I \le U_{DD}$	zablokowany	nienasycony

Oznaczenia:

U_I — napięcie wejściowe,

U_O — napięcie wyjściowe,

U_{TN} — napięcie progowe tranzystora CMOS z kanałem N (wartość znamionowa dla tranzystorów MOSFET wynosi ok. +2 V),

U_{TP} — napięcie progowe tranzystora CMOS z kanałem P (wartość znamionowa dla tranzystorów MOSFET wynosi ok. −3 V),

U_{GS} — napięcie bramka-źródło,

U_{DD} — napięcie doprowadzone do drenu,

U_{SS} — napięcie doprowadzone do źródła (zwykle $U_{SS} = 0\,V$).

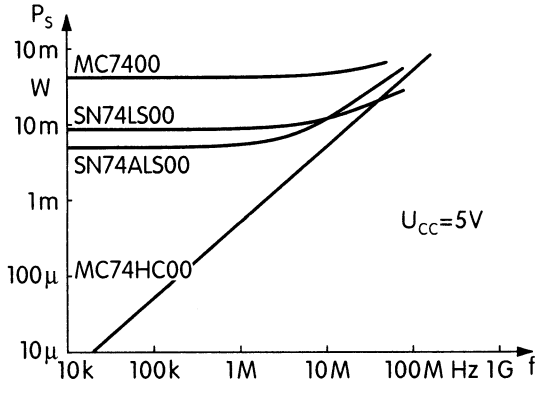
go do wartości równej napięciu progowemu U_{TN} tranzystora $T2$ następuje odblokowanie obu tranzystorów (obszar II) i przepływ względnie dużego prądu I_{DD} (obszar III), kiedy to obydwa tranzystory znajdują się w stanie nasycenia. Przez dalsze zwiększanie napięcia wejściowego inwerter doprowadza się do stanu V, w którym tranzystor $T2$ jest w stanie przewodzenia, a tranzystor $T1$ — zablokowany.

Bramka w stanie przełączania pobiera znacznie większy prąd ze źródła niż w stanie ustalonym. Dotyczy to szczególnie obszarów II, III i IV (rys. 6.2). W obszarze III (gdy oba tranzystory znajdują się w stanie nasycenia) amplituda tego prądu osiąga maksimum (od kilku do kilkunastu miliamperów w zależności od wartości napięcia zasilającego układ). Prąd pobierany ze źródła jest ograniczony jedynie niewielkimi rezystancjami tranzystorów w stanie nasyconym. Skutki takiego impulsowego poboru prądu ze źródła mogą być identyczne, jak opisane w p. 5.2 w odniesieniu do układów TTL. Zapobieganie sprzęganiu się układów polega na stosowaniu kondensatorów odsprzęgających o pojemnościach 0,01÷0,1 mF (kondensatorów bezindukcyjnych).

- **Pobór mocy**

Z porównania charakterystyk poboru prądu przez bramkę TTL i bramkę CMOS wynika, że każde przełączenie bramki CMOS wymaga poboru znacznie większej mocy niż w przypadku bramki TTL. Miarą tej mocy może być pole powierzchni pod krzywą przebiegu pobieranego prądu. Spostrzeżenie to ma istotne znaczenie, gdyż pozwala zrozumieć silną zależność poboru mocy przez układy CMOS od częstotliwości przełączania bramki. Potoczne przeświadczenie o znikomym zapotrzebowaniu na moc przez układy CMOS, w praktyce znajduje potwierdzenie jedynie przy niewielkich częstotliwościach pracy układu. Zależność mocy traconej w bramce NAND od częstotliwości jej przełączania dla układów TTL (74, 74LS, 74ALS) i CMOS (74HC) przedstawiono na rys. 6.3. Charakterystyka ta dla układów CMOS serii 4000B będzie miała podobny przebieg, a przy napięciach zasilających większych niż 5V będzie położona odpowiednio wyżej.

Jak widać z charakterystyki na rys. 6.3 pobór mocy przez bramki CMOS jest porównywalny z poborem mocy przez układy TTL już przy częstotliwościach ok. 10 MHz. Przy kilkunastu megahercach moc pobierana przez układy CMOS zaczy-



Rys. 6.3. Zależność poboru mocy przez bramki CMOS i TTL od częstotliwości

na przewyższać nawet moc pobieraną przez układy TTL. Wzrost poboru mocy przez układ w zależności od częstotliwości przełączania dotyczy także układów TTL. Jest on jednak znacznie mniejszy i wynosi ok. 0,15÷0,3 mW/MHz.

- **Parametry wejściowe i wyjściowe. Obciążalność**

Rezystancja wejściowa układów CMOS jest tak duża, że jej wpływ na pracę układów można pominąć. Typowa wartość prądu wejściowego zawiera się w przedziale 10÷100 pA (układy serii 4000B), natomiast wartość $I_I < 100$ nA dla szybkich układów CMOS (serie HC, HCT, AC, ACT). Jego maksymalna (dopuszczalna) wartość wynosi odpowiednio 10 mA i 20 mA.

Prąd wyjściowy charakteryzuje możliwości obciążenia układu. Wartość maksymalna tego prądu jest uwarunkowana wartością dopuszczalną prądu przepływającego w obszarze źródło-dren. W stanie wysokim przewodzi tranzystor *T1* z kanałem typu P. Jest to prąd I_{OH}, płynący ze źródła zasilania U_{DD} do obciążenia. W stanie niskim przewodzi tranzystor *T2* z kanałem typu N. Jest to prąd płynący od obciążenia do źródła U_{SS}. Wartością dopuszczalną prądu wyjściowego jest wartość prądu mierzona wówczas, gdy spadek napięcia na przewodzącym tranzystorze wynosi 0,5V. Oznacza to, że w chwili pomiaru tego prądu na wyjściu otrzymuje się napięcie $U_{DD} = -0{,}5$ V (w stanie wysokim) oraz napięcie $U_{SS} = +0{,}5$ V (w stanie niskim). Wartości prądów wyjściowych (I_{OH}, I_{OL}) zależą nie tylko od napięcia zasilania, ale także od temperatury otoczenia. Dokładne informacje na ten temat można znaleźć w katalogach wydawanych przez producentów układów CMOS. Z powodu symetrii układu wartości bezwzględne tych prądów są jednakowe w obu stanach pracy. Różnią się one tylko zwrotami. Prąd I_{OH} wypływa z bramki (przypisujemy mu zgodnie z przyjętą konwencją zwrot ujemny), natomiast prąd I_{OL} wpływa do niej (ma zwrot dodatni). Typowe wartości prądów wyjściowych I_{OH}, I_{OL} zestawiono w tablicy 6.3.

Tablica 6.3. **Prądy wyjściowe układów CMOS (wartości znamionowe)**

Prąd wyjściowy		Układy 4000B			Układy HC			Układy AC		
		Napięcie zasilania, V								
		+5	+10	+15	+2	+4,5	+6	+3,0	+4,5	+5,5
$-I_{OH}$	mA	1,0	2,6	6,8	2,5	12	18	4	24	24
I_{OL}	mA	1,0	2,6	6,8	4,5	15	23	12	24	24

Wartości te są wielokrotnie większe niż znamionowa wartość prądu wejściowego. ***Ze względu na wydajność prądową i pobór prądu w stanach statycznych można wyjście układu CMOS obciążyć praktycznie dowolną liczbą wejść.***

Jedynym czynnikiem ograniczającym dopuszczalną obciążalność w układach CMOS jest **obciążenie pojemnościowe.** Wartość pojemności wejściowej układów CMOS wynosi 5 ÷ 10 pF. Obciążenie pojemnościowe układu ma istotny wpływ na czasy narastania, opadania i propagacji sygnałów. Wydłużony czas propagacji sprawia, że każde przełączenie bramki wymaga zwiększonego poboru energii ze źródła (przez dłuższy czas oba tranzystory są w stanie przewodzenia i nasycenia). ***Im większe jest obciążenie pojemnościowe, tym dłuższe są czasy narastania, opadania i propagacji sygnałów oraz tym większa jest moc pobierana ze źródła, a zatem — tym mniejsza dopuszczalna częstotliwość przełączania układów. Na podstawie maksymalnej częstotliwości pracy układu — z jaką chcemy, aby układ pracował — można określić dopuszczalne obciążenie pojemnościowe, czyli liczbę sterowanych układów CMOS.***

Pytania i zadania

1. Wymień różnice między układami TTL a CMOS.
2. Podaj zasady obchodzenia się z układami CMOS. Określ powód obowiązywania takich zasad.
3. Narysuj charakterystykę przejściową $U_O = f(U_I)$ oraz poboru prądu $I_{DD} = f(U_I)$ inwertera CMOS. Uzasadnij ich przebieg. Określ związek między nimi.
4. Omów pobór mocy przez bramkę CMOS. Uzasadnij zależność strat mocy od częstotliwości przełączania bramki.
5. Jaka jest obciążalność bramek CMOS? Od czego przede wszystkim zależy dopuszczalna liczba wejść CMOS sterowanych z jednej bramki?
6. W jakim celu stosuje się kondensatory blokujące (odsprzęgające)?

6.3. Wybrane układy SSI serii 4000B

Oznaczenia układów CMOS serii 4000B produkcji CEMI omówiono w rozdz. 5. Także ich parametry (i ich symbole) są definiowane analogicznie, jak dla wcześniej omówionych układów TTL. Podstawowe parametry techniczne układów CMOS serii MCY74 produkcji CEMI zestawiono w tabl. 6.1. ***Napięcie zasilania*** $U_z = U_{DD} - U_{SS}$ ***tych układów może przyjmować wartości z przedziału*** $+2 \div +18\,V$ (***maksymalnie*** $+20\,V$). ***Napięcie to — w przeciwieństwie do napięcia zasilającego układy TTL — nie musi być stabilizowane. Na ogół punkt dołączenia potencjału niskiego jest uziemiony*** ($U_{SS} = 0\,V$) ***i napięcie zasilające*** $U_z = U_{DD}$. Dla układów CMOS zgodnych z TTL przyjęto oznaczać napięcie zasilania symbolem U_{CC}. W tablicy 6.2 podano parametry układów MCY74 przy trzech różnych napięciach zasilających: +5 V, +10 V i +15 V. Wartości parametrów przy innych napięciach zasilających można uzyskać metodą aproksymacji.

Podstawowymi układami SSI w serii 4000B (MCY74) są: inwerter, bramka NAND i NOR, bramka trójstanowa oraz bramka transmisyjna.

Asortyment układowy elementów wytwarzanych w technice CMOS jest porównywalny z asortymentem układowym układów TTL. Uwaga ta dotyczy jednak

serii 4000B wytwarzanej przez zachodnie firmy elektroniczne. Polski odpowiednik tych układów, seria MCY74, jest jak dotychczas asortymentowo znacznie uboższa.

W ramach serii MCY74 były produkowane: inwertery, dwuwejściowe bramki NAND i NOR, bramki z przerzutnikiem Schmitta, także 2-, 3-, 4-wejściowe bramki NAND i NOR, bufory (inwertery), bufory (wzmacniacze logiczne), bramki z otwartym drenem (analogi bramek TTL z otwartym kolektorem), bramki z wyjściem trójstanowym, bramki transmisyjne. Niektóre z tych układów zostaną opisane w tym rozdziale. Oczywiście asortyment układowy elementów CMOS jest szerszy i obejmuje także układy MSI oraz LSI — zarówno kombinacyjne, jak i sekwencyjne. Niektóre z nich zostaną omówione w następnych rozdziałach.

Tablica 6.4. **Parametry techniczne układów 4000B (MCY74)**

Parametr		Napięcie zasilania, V		
		+5	+10	+15
U_I	V	−0,5 ÷ +5,5	−0,5 ÷ +10,5	−0,5 ÷ +15,5
U_{IL}	V	0 ÷ +1,5	0 ÷ +3	0 ÷ +4
U_{IH}	V	+3,5 ÷ +5	+7 ÷ +10	+11 ÷ +15
U_{OH}	V	+4,95 ÷ +5	+9,95 ÷ +10	+14,95 ÷ +15
U_{OL}	V	0 ÷ +0,05	0 ÷ +0,05	0 ÷ +0,05
$t_{pLH} = t_{pHL} = t_p$ (bramka)	ns	125 (do 250)	60 (do 120)	40 (do 80)
$t_{pLH} = t_{pHL} = t_p$ (przerzutnik)	ns	150 (do 300)	65 (do 130)	45 (do 90)
P_s (na układ scalony)	mW	dop. 500 (stat. 2,5 nW)	500	500
obciążalność N (bramkami TTL-LS)	—	2	—	—
$D = t_p P_s$ (przy f_{CL} = 100 kHz)	pJ	0,2	0,2	0,2
ΔU (30% U_z)	V	1,5	3	4,5
ΔU_{min}	V	1	2	2,5

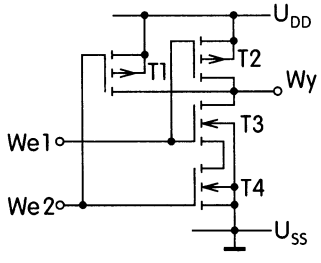
• Bramka NAND CMOS 4011B (MCY74011)

Schemat bramki realizującej negację iloczynu sygnałów wejściowych przedstawiono na rys. 6.4.

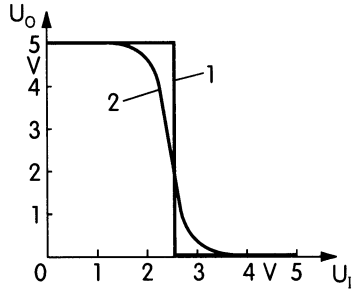
Podstawową strukturą budowy elementów logicznych w technice CMOS pozostaje nadal inwerter, czyli para komplementarna tranzystorów MOSFET. Inwerter tworzy para tranzystorów o połączonych bramkach. Układ na rysunku 6.4 zawiera dwa takie inwertery: jeden zbudowany z tranzystorów *T1*, *T4*, a drugi z tranzystorów *T2*, *T3*. Jeżeli na jednym z wejść układu jest niski poziom napięcia, to przewodzi jeden z tranzystorów o kanale P (*T1* lub *T2*), a zatkany jest odpowiadający mu tranzystor o kanale N. Na wyjściu układu poziom napięcia jest wysoki. Prąd wyjściowy płynie przez przewodzący tranzystor *T1* lub *T2*. Rezystancję wyjściową układu determinuje rezystancja tego tranzystora. Jeżeli na obydwu wejściach będzie stan niski, to w stanie przewodzenia będą obydwa tranzystory *T1*, *T2*. Nie zmieni to poziomu napięcia wyjściowego, ale rezystancja wyjściowa będzie teraz dwa razy mniejsza (równoległe połączenie rezystancji przewodzących tranzystorów *T1*, *T2*). Także czas opadania (zmiany stanu na wyjściu z **H** na **L**) jest krótszy przy jednoczesnym pobudzeniu obydwu wejść niż przy pobudzeniu tylko jednego z nich.

Zależność rezystancji wyjściowej układu oraz szybkości przełączania od liczby pobudzanych wejść sprawiła, że nie produkuje się podstawowych bramek logicznych o większej liczbie wejść niż cztery. ***Jednak stałość omawianych parametrów (rezystancja wyjściowa, czasy narastania, opadania i propagacji) najskuteczniej zapewnia stosowanie na wyjściu*** **bufora** ***w postaci dodatkowego inwertera. Układy takie noszą nazwę*** **układów buforowanych.** W niektórych seriach układów CMOS stosuje się także buforowanie wejść. Układy buforowane są jednak wolniejsze od układów niebuforowanych. Omawiana seria układów cyfrowych 4000B (MCY74) ma buforowane zarówno wejścia, jak i wyjścia. Rzeczywisty schemat bramki NAND (MCY74011) zawiera więc dodatkowo (w porównaniu do schematu z rys. 6.4) inwertery (bufory) na wejściu i wyjściu układu oraz wspomniane wcześniej układy zabezpieczające wejścia przed przebiciem ładunkiem elektrostatycznym.

Wracając do opisu działania bramki NAND (rys. 6.4), należy jeszcze omówić jej zachowanie w sytuacji, gdy na obu wejściach poziom napięcia jest wysoki. Wówczas oba tranzystory z kanałem P (*T1*, *T2*) są w stanie zatkania. Tranzystory



Rys. 6.4. Schemat zasadniczy dwuwejściowej bramki NAND CMOS



Rys. 6.5. Zestawienie charakterystyk przejściowych bramki buforowanej *1* i niebuforowanej *2*

T3, *T4* są w stanie przewodzenia i na wyjściu poziom napięcia jest niski. Z opisu działania układu z rys. 6.4 wynika więc, że realizuje on funkcję negacji iloczynu w konwencji dodatniej.

Charakterystyka przejściowa bramki buforowanej różni się od takiej charakterystyki bramki niebuforowanej (rys. 6.2). Porównanie obu charakterystyk przedstawiono na rys. 6.5. Kształt charakterystyki przejściowej bramki buforowanej odpowiada całkowicie charakterystyce idealnego elementu przełączającego.

- **Bramka z układem Schmitta**

Z tych samych przyczyn, co dla układów TTL, są produkowane bramki CMOS z przerzutnikiem Schmitta. Jeżeli zachodzi potrzeba kształtowania sygnałów cyfrowych (z przebiegów wolnozmiennych lub analogowych) lub jest pożądana duża odporność na zakłócenia, to należy stosować układ Schmitta. Układ Schmitta w wersji CMOS (4000B) znajduje nawet szersze zastosowanie niż układ TTL ze względu na szeroki zakres napięcia zasilania, dużą rezystancję wejściową, jednakowe rezystancje wyjściowe (zarówno w stanie **H**, jak i **L**).

Układy CMOS serii 4000B z powodu małej szybkości (duży czas propagacji) są obecnie serią nie produkowaną. Ich istotną wadą jest także niezgodność końcówkowa z układami TTL oraz niewielki prąd wyjściowy. Coraz powszechniej są produkowane i używane tzw. **szybkie układy CMOS** do których należą serie HC/HCT i AC/ACT.

Pytania i zadania

1. Jakim napięciem można zasilać układy CMOS serii 4000B (MCY74)?
2. Wymień przyczyny buforowania wejść i wyjść bramek CMOS. Podaj zalety i wady takiego rozwiązania.
3. Jakie mogą być powody stosowania bramek z przerzutnikiem Schmitta?

6.4. Układy CMOS serii HC/HCT i AC/ACT

Podstawowy kierunek ulepszania układów CMOS to zwiększanie szybkości działania. Obecnie produkowane **szybkie układy CMOS** mają czasy propagacji porównywalne z szybkimi układami TTL (patrz tabl. 6.1). Biorąc jeszcze pod uwagę, że te **szybkie układy CMOS** mogą sterować dużymi obciążeniami (ich prądy wyjściowe są nawet większe niż w układach TTL), dominacja tych układów nad układami TTL staje się coraz bardziej oczywista.

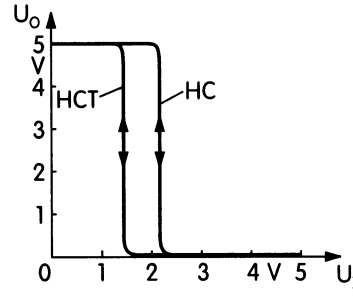
Serie HC (ang. High speed CMOS) i AC (ang. Advanced CMOS) są w pełni zgodne końcówkowo, oznaczeniowo i funkcjonalnie z układami TTL. Serie HCT i ACT ponadto są kompatybilne z układami TTL LS (stąd litera T w nazwie serii). Z powyższych uwag wynika więc, że możemy zamienić np. układ 74LS00 na układ 74HCT00 (lub 74ACT00), ponieważ są one całkowicie zgodne końcówkowo i funkcjonalnie oraz ich parametry wejściowe i wyjściowe nie są gorsze niż układu 74LS.

Tablica 6.5. **Parametry techniczne szybkich układów CMOS**

Parametr		HC	HCT	AC	ACT	TTL-LS
Napięcie zasilania	V	2 ÷ 6	5±10%	3 ÷ 5,5	5±10%	5±5%
Prąd wejściowy (przy $U_I = 0{,}4$ V)	mA	0,001	0,001	0,001	0,001	0,4
$I_{OL\,max}$	mA	25*	25*	50*	50*	8*
$I_{OH\,max}$						0,4*
$U_{IH\,min}$	V	3,5	2,0	3,5	2,0	2,0
$U_{IL\,max}$		1,5	0,8	1,5	0,8	0,8
$U_{OH\,min}$		4,5	4,5	4,5	4,5	2,7
$U_{OL\,max}$		0,26	0,26	0,36	0,36	0,5

* Są to wartości prądu przy napięciu $U_{OL} = 0{,}5$ V oraz $U_{OH} = U_{CC} - 0{,}5$ V, $U_{CC} = 5$ V (przy $U_{CC} < 5$ V prądy te są odpowiednio mniejsze)

Wprawdzie również układy HC i AC są zgodne końcówkowo, oznaczeniowo i funkcjonalnie z układami TTL, ale taka zamiana nie zawsze będzie dopuszczalna, gdyż ich parametry wejściowe różnią się od parametrów wyjściowych układów TTL. Natomiast parametry wyjściowe szybkich układów CMOS są lepsze niż wymagane parametry sygnałów sterujących bramką TTL i takie połączenie (TTL → CMOS) jest dozwolone. W celu porównania zestawiono w tabl. 6.5 parametry układów CMOS — serii szybkich.



Rys. 6.6. Typowe charakterystyki przejściowe szybkich układów CMOS

Szybkie układy CMOS są układami buforowanymi. Ich charakterystyki przejściowe są więc zbliżone do idealnych charakterystyk przełączania. Jednak napięcie przełączania układów HCT i ACT wynosi ok. 1,4 V (jak TTL), natomiast układów HC i AC wynosi ok. 2,3 V (przy napięciu $U_{CC} = 5$ V). Obie charakterystyki przejściowe pokazano na rys. 6.6.

7

Przerzutniki

7.1. Wprowadzenie

Na lekcjach z podstaw elektroniki podano wiadomości na temat przerzutników: monostabilnych, bistabilnych i astabilnych. Omówiono zasadę ich działania, elementy z jakich są zbudowane, przebiegi czasowe i rozkłady potencjałów w różnych punktach układu przerzutnika. Z uwagi na użyteczność tych układów w systemach cyfrowych interesować nas będą jedynie przebiegi czasowe na wejściach i wyjściu przerzutnika. W tym rozdziale zajmiemy się wyłącznie przerzutnikami bistabilnymi, a w następnym monostabilnymi i astabilnymi.

Przerzutniki, oprócz bramek logicznych, są podstawową grupą elementów stosowanych w technice cyfrowej. W podrozdziale 3.1 wprowadzono podział układów cyfrowych na kombinacyjne i sekwencyjne (rys. 3.2). Układy sekwencyjne mają pamięć i z tego powodu są nazywane układami kombinacyjnymi z pamięcią. Przerzutniki bistabilne pełnią w nich rolę pamięci. ***Jednocześnie sam przerzutnik jest najprostszym układem sekwencyjnym.***

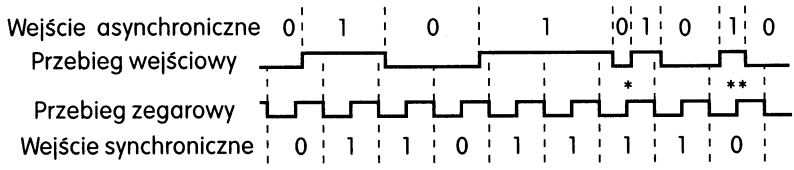
Układy sekwencyjne dzieli się na synchroniczne i asynchroniczne. Do budowy pamięci układów sekwencyjnych synchronicznych używa się przerzutników synchronicznych, natomiast do budowy pamięci układów sekwencyjnych asynchronicznych używa się przerzutników asynchronicznych.

W układach synchronicznych występuje pewien (co najmniej jeden) wyróżniony sygnał — zwany **przebiegiem zegarowym, taktującym** lub **synchronizującym.** Przebieg ten wyznacza cykl pracy układu, a jego okres stanowi umowną jednostkę czasu. Sygnał zegarowy określa chwile, w których stany wejść oddziałują na układ. Chwile te są wyznaczane przez zbocze dodatnie bądź ujemne przebiegu taktującego, dlatego mówimy o synchronizacji układu zboczem narastającym lub opadającym. ***W chwilach tych stan innych wejść nie powinien się zmieniać.*** Odcinek czasu pomiędzy dwoma kolejnymi zboczami aktywnymi sygnału zegarowego jest nazywany **taktem.**

Są produkowane także przerzutniki synchroniczne wyzwalane poziomem. Przerzutnik taki również zostanie omówiony.

W układach asynchronicznych każda zmiana stanu wejść układu oddziałuje na układ, powodując jego reakcję.

Aby zrozumieć istotę oddziaływania wejścia na układ synchroniczny i asynchroniczny posłużmy się dowolnym przebiegiem czasowym i określmy sekwencję wejściową w obu przypadkach. Pokazano to na rys.7.1.



Rys. 7.1. Ilustracja oddziaływania przebiegu wejściowego na układ asynchroniczny i synchroniczny

Sekwencja wejściowa „zaobserwowana" przez układ asynchroniczny to **010101010**. W tym samym czasie układ synchroniczny zinterpretuje ten przebieg jako sekwencję wejściową **011011110**. Zauważmy przy tym, ***że sekwencję wejściową układu synchronicznego jesteśmy w stanie określić dopiero po naniesieniu na wykres przebiegu zegarowego.*** Przyjęcie innej częstotliwości sygnału taktującego sprawi natychmiast, że sekwencja wejściowa układu synchronicznego będzie zupełnie inna, chociaż dla układu asynchronicznego nic się nie zmieni. Zwróćmy jeszcze uwagę na impulsy oznaczone *. Impuls * o poziomie **L** oraz impuls ** o poziomie **H** przez układ synchroniczny nie został w ogóle „zauważony". Czy zatem układy synchroniczne gubią informację wejściową? Odpowiedzi na to pytanie mogą być dwie:

1. Jeżeli wzmiankowane impulsy niosą określoną informację wejściową i powinny mieć wpływ na działanie układu, to oznacza, że częstotliwość przebiegu zegarowego jest zbyt mała. W takiej sytuacji należy zwiększyć częstotliwość przebiegu taktującego.
2. Jeżeli zaś częstotliwość przebiegu synchronizującego jest właściwa, to wzmiankowane impulsy są impulsami zakłócającymi. Impulsy zakłócające z natury rzeczy są impulsami krótkotrwałymi i jeżeli czas ich trwania jest dużo mniejszy niż czas jednego taktu (okres przebiegu zegarowego), to większość z nich zostanie „nie zauważona", czyli nie zakłóci pracy układu. Jest to niewątpliwą zaletą układów sekwencyjnych synchronicznych. Naturalnie impuls zakłócający może wystąpić w chwili pojawienia się aktywnego zbocza sygnału zegarowego, ale prawdopodobieństwo takiej sytuacji jest mniej więcej takie, jak iloraz czasu trwania zbocza aktywnego do okresu taktowania. Powyższe uwagi są prawdziwe przy założeniu, że wejścia zegarowe są wolne od zakłóceń. Wymaga to stosowania odpowiednich zabezpieczeń przed pojawieniem się zakłóceń na tych właśnie wejściach, np. poprzez ekranowanie doprowadzeń sygnałów taktujących.

Układy asynchroniczne są zatem bardziej wrażliwe na zakłócenia. Ponadto trudniejsze jest ich projektowanie. Szczególną trudność stanowi tzw. zjawisko wyścigów, spowodowane niejednoczesnością (brakiem synchronizacji) przełącza-

nia elementów pamięciowych układu. Zaletą układów asynchronicznych jest ich prostsza budowa i mniejsza ilość elementów potrzebna do ich budowy. W świetle ostatnich osiągnięć w technologii scalania układów zaleta ta jest mało istotna. W praktyce więc większość układów cyfrowych sekwencyjnych to układy synchroniczne. W podręczniku będą więc omawiane układy sekwencyjne budowane przy użyciu synchronicznych elementów pamięciowych (przerzutników).

Działanie przerzutnika można opisać za pomocą tzw. **tablicy przejść, tablicy wzbudzeń, tablicy charakterystycznej** lub **wykresu czasowego.** Wszystkie te sposoby będą wykorzystywane w niniejszym podręczniku.

Przerzutnik używany w technice cyfrowej jest układem o co najmniej dwóch wejściach i z reguły dwóch wyjściach. **Wejścia** mogą być:

— **zegarowe** (ang. *Clock*), zwane również **synchronizującymi** albo **wyzwalającymi**. Wejście to oznaczać będziemy literą **C**. Używa się także oznaczeń **CK, CL, CP, T, CLK**;
— **informacyjne**;
— **programujące, przygotowujące**.

Wejście zegarowe mają wyłącznie przerzutniki synchroniczne. ***Przerzutniki takie reagują na informację podawaną na* wejścia informacyjne *tylko w obecności impulsu zegarowego. Stan wejść informacyjnych powinien być wówczas już ustalony i nie zmieniać się.***

Przerzutnik może być wyposażony w dwa **wejścia programujące**: wejście ustawiające w stan wysoki, zwane krótko **wejściem ustawiającym** oznaczane **S** lub **PR** (ang. *Set* lub *PReset*) oraz wejście ustawiające w stan niski, zwane **wejściem zerującym** i oznaczane **R** lub **CLR** (ang. *Reset* lub *CLeaR*). Wejścia te są wejściami asynchronicznymi i nadrzędnymi w stosunku do pozostałych wejść. Nadrzędność ta wyraża się tym, że przy sterowaniu przerzutnika od strony jednych i drugich wejść (synchronicznych i asynchronicznych) o stanie przerzutnika decydują wejścia asynchroniczne.

Istnieje wiele typów przerzutników. Podstawowymi przerzutnikami asynchronicznymi są przerzutniki typu *rs* oraz $\overline{rs}$. Zasadniczymi przerzutnikami synchronicznymi są przerzutniki typu *JK*, *D* i D_{LATCH} (*D* „zatrzask"). Używane są także przerzutniki typu *T* i *RS*.

Zastosowania przerzutników są bardzo szerokie. Są one wykorzystywane do budowy np. liczników, rejestrów. Układy te omówiono w następnych rozdziałach podręcznika.

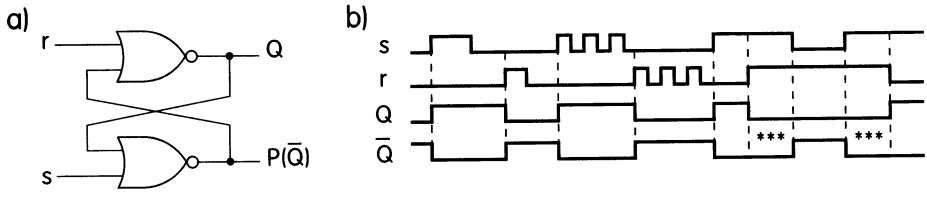
7.2. Przerzutniki asynchroniczne

Przerzutniki asynchroniczne — w celu odróżnienia ich od przerzutników synchronicznych — będziemy oznaczać małymi literami. W literaturze technicznej częściej jednak można spotkać oznaczenie pisane dużymi literami. Poniżej omówiono dwa typy przerzutników asynchronicznych: przerzutniki *rs* i $\overline{rs}$.

Najbardziej czytelną postacią opisu działania przerzutnika asynchronicznego są przebiegi czasowe. ***Należy jednak pamiętać, że sygnały wejściowe nie powinny zmieniać się jednocześnie***. Przebiegi czasowe rysowane z pominięciem tej zasady byłyby nieczytelne (niejednoznaczne). W praktyce, nawet jednoczesna zmiana dwóch sygnałów wejściowych będzie interpretowana przez układ jako dwie odrębne zmiany stanu wejść. Układ zareaguje jak na zmiany niejednoczesne i to z przypadkową (nie zawsze tą samą) kolejnością. Powodem tego jest duża szybkość działania układów cyfrowych, różne czasy propagacji przy pobudzaniu układu od strony różnych wejść, różne poziomy napięć, na które reagują poszczególne wejścia (nawet ta sama bramka sterowana z dwóch wejść może mieć na każdym z nich inny próg napięcia przełączającego), a ponadto parametry te mogą się zmieniać pod wpływem temperatury i na skutek starzenia się.

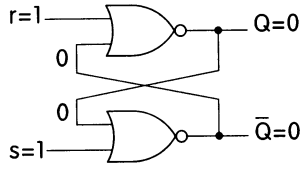
- **Przerzutnik *rs***

Przerzutnik asynchroniczny *rs* jest zbudowany z dwóch bramek NOR (rys. 7.2a). Przerzutnik ma dwa wejścia: ustawiające **s** (ang. *set*) i zerujące **r** (ang. *reset*) oraz dwa wyjścia: jedno oznaczone **Q** i drugie oznaczone wstępnie **P**. Przeanalizujmy możliwe stany wyjść tego układu przy stanie wejść **rs = 00**. Zauważmy, że **Q** może być równe **1** i wówczas **P = 0** lub **Q = 0**, a wtedy **P = 1**. W obu więc przypadkach $\mathbf{P = \overline{Q}}$, co ma istotne zalety i będziemy się starać ten warunek spełnić.



Rys. 7.2. Przerzutnik asynchroniczny *rs*: a) schemat logiczny, b) przebiegi czasowe

Umożliwi to uzyskanie tzw. **wyjść komplementarnych** i oznaczenie ich przez **Q** oraz $\overline{\mathbf{Q}}$. Ale przypomnijmy sobie, jaki jest stan na wyjściu bramki NOR, jeżeli jedno z jej wejść jest w stanie **H**. Otóż niezależnie od stanu drugiego wejścia (innych wejść) bramka ta będzie w stanie niskim **L**. Zatem ustawienie obu wejść tego przerzutnika w stan wysoki sprawi, że oba wyjścia będą w stanie niskim (rys. 7.3). Wówczas $\mathbf{P \neq \overline{Q}}$ i oznaczenie tych wyjść: jednego jako proste, drugiego jako zanegowane prowadzi do konkluzji, że $\mathbf{Q = \overline{Q}}$, co naturalnie nie jest prawdą. **Dlatego dla przerzutnika *rs* stan wejść rs = 11 określamy jako logicznie zabroniony**. Fizyczny zakaz oznaczałby potencjalną możliwość uszkodzenia układu. Takiego niebezpieczeństwa w tym przypadku nie ma. Uzyskujemy jedynie sprzeczność polegającą na tym, że $\mathbf{Q = \overline{Q}}$. Jest to sprzeczność logiczna i dlatego mówimy jedynie o zakazie logicznym. Pamiętajmy także, że wejście **s**



Rys. 7.3. Przerzutnik asynchroniczny *rs* przy rs = 11

jest wejściem ustawiającym, a **r** — zerującym. Wymuszenie na obu wejściach stanu **1** to próba wykonania akcji „ustaw i jednocześnie wyzeruj". W tym kontekście niepożądane zachowanie się przerzutnika jest całkowicie usprawiedliwione.

Z powyższego wstępu wynika praktyczny wniosek nie tylko odnośnie do użytkowania tego przerzutnika, ale także jego opisywania. Otóż należy tak budować układy współpracujące (sterujące) wejściami przerzutnika, aby wyeliminować możliwość pojawienia się stanu **1** jednocześnie na obu wejściach — szczególnie wtedy, kiedy korzystamy z obu wyjść przerzutnika.

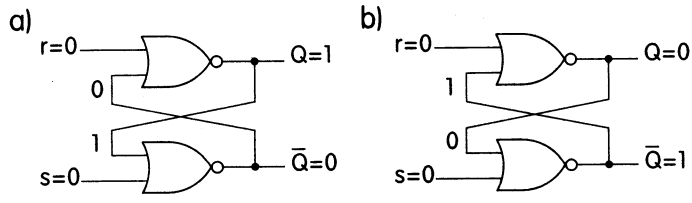
Wykres czasowy opisujący działanie przerzutnika powinien także uwzględniać powyższe wymagania, więc należy go tak rysować, aby stan wejść **11** nie pojawiał się co chwilę, bo wykres przestanie być wówczas komunikatywny. Podstawowym stanem wejść przerzutnika powinien być stan **00.** Kolejne fragmenty wykresu powinny przedstawiać zachowanie się przerzutnika przy pobudzaniu (ustawianiu w stan **1**) raz wejścia **r** raz **s**, czy też kilkakrotnego pobudzenia wejścia **r** lub **s**. Naturalnie wykres objaśniający działanie przerzutnika powinien zawierać także fragment ilustrujący stan logicznie zabroniony, ale jako przypadek szczególny i odosobniony. Wykres czasowy na rys. 7.2b skonstruowano zgodnie z powyższymi zaleceniami. Możemy z niego odczytać, że: impuls na wejściu **s** (przy **r = 0**) ustawia przerzutnik w stan **1**, ponowienie impulsów wpisujących jedynkę nie ma już żadnego wpływu na zachowanie się przerzutnika. Podobnie działa przerzutnik przy doprowadzeniu do jego wejścia **r** sygnału **1** (przy **s = 0**). Tym razem jest on zerowany i kolejne impulsy zerujące nie zmieniają już stanu przerzutnika. Przy stanie wejść **00** przerzutnik może być zarówno w stanie **1**, jak i w stanie **0**. Stan ten nazywamy **stanem pamiętania informacji wpisanej do przerzutnika.** Jednoczesne ustawianie (**s = 1**) oraz zerowanie (**r = 1**) prowadzi do tego, że $\mathbf{P} \neq \overline{\mathbf{Q}}$ i sytuację tę demonstruje ostatni fragment przebiegu oznakowany ***.

Przeanalizujmy jeszcze stan przerzutnika (czyli jego wyjście) w przypadku różnych stanów wejść. ***Układ asynchroniczny bowiem nie zmienia swego stanu tak długo, jak długo nie zmienia się stan jego wejść. Mówimy wówczas, że układ asynchroniczny jest w*** **stanie stabilnym**. Przeanalizujmy zatem wszystkie jego stany stabilne.

Rozważyliśmy już szczegółowo sytuację, w której oba wejścia przerzutnika są ustawione w stan **1** (rys. 7.3). Przejdźmy zatem do omówienia pozostałych stanów stabilnych.

Jeżeli na obu wejściach jest stan niski **0**, to przerzutnik może się znajdować zarówno w stanie pamiętania **1** (wyjście **Q = 1**), jak i w stanie pamiętania **0** (wyjście **Q = 0**). Mówimy wówczas, że przerzutnik jest odpowiednio: w stanie wysokim — **włączony** (**ustawiony w stan 1**) lub niskim — **wyłączony** (**zgaszony, wyzerowany, ustawiony w stan 0**). Obie te sytuacje przedstawiono na rys. 7.4.

Normalny stan pracy przerzutnika to oba wejścia w stanie niskim i oczekiwanie na pojawienie się **1** na jednym z wejść. Zauważmy, że podanie **1** na wejście ustawiające **s** wówczas, gdy przerzutnik jest w stanie pamiętania **1** (rys. 7.4a), nie



Rys. 7.4. Przerzutnik asynchroniczny *rs* przy rs = 00: a) w stanie pamiętania 1; b) w stanie pamiętania 0

zmienia stanu tego przerzutnika. Na drugim wejściu bramki NOR, do której doprowadzamy sygnał **s**, panuje bowiem poziom **1**. Podanie kolejnej **1** na kolejne wejście bramki NOR nie zmienia przecież stanu jej wyjścia.

Podobnie, podanie **1** na wejście zerujące **r** wówczas, gdy przerzutnik jest w stanie pamiętania **0** (rys. 7.4b) nie zmienia stanu tego przerzutnika. Na drugim wejściu bramki NOR, do której doprowadzamy sygnał **r**, panuje już poziom **1**. Podanie kolejnej **1** na kolejne wejście bramki NOR nie zmienia przecież stanu jej wyjścia.

Tablica 7.1. **Stany stabilne przerzutnika *rs***

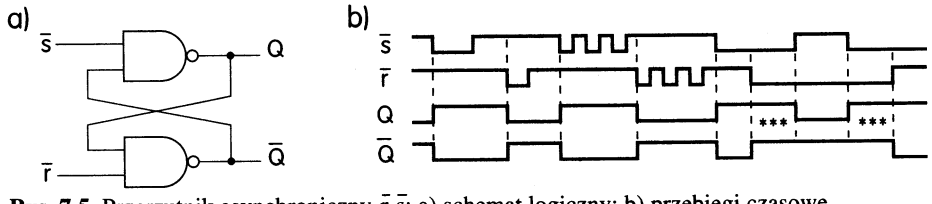
r	*s*	Q $\overline{Q}$	Stan stabilny
1	1	0 0	stan wejść logicznie zabroniony
1	0	0 1	przerzutnik ustawiony w stan 0
0	1	1 0	przerzutnik ustawiony w stan 1
0	0	0 1	pamiętanie stanu 0 } stan pamiętania
		1 0	pamiętanie stanu 1 }

Zmiana stanu przerzutnika nastąpi więc po podaniu **1** na wejście ustawiające **s**, gdy przerzutnik jest w stanie pamiętania **0**, albo po podaniu **1** na wejście zerujące **r**, gdy przerzutnik jest w stanie pamiętania **1**.

Wszystkie stany stabilne przerzutnika *rs* zestawiono w tabl. 7.1.

- **Przerzutnik $\overline{r}\,\overline{s}$**

Do budowy przerzutnika asynchronicznego można użyć bramek NAND zamiast NOR (rys. 7.5). Taki przerzutnik jest włączany (wyłączany) wówczas, gdy napięcie na wejściu ustawiającym (zerującym) przyjmie poziom logiczny **0**. Jest to dokładnie odwrotnie niż w przerzutniku zbudowanym z bramek NOR, w którym



Rys. 7.5. Przerzutnik asynchroniczny $\overline{r}\,\overline{s}$: a) schemat logiczny; b) przebiegi czasowe

ten efekt osiągano przez doprowadzenie sygnału **1** do odpowiednich jego wejść. Stąd nazwa tego przerzutnika (NIE *r* NIE *s*).

Także pozostałe cechy przerzutnika $\overline{r}\,\overline{s}$ można wywieść przez analogię do przerzutnika *rs*. Tak więc stan wejść **00** jest w tym przerzutniku logicznie zabroniony, a stan wejść **11** oznacza pozostawanie przerzutnika w stanie pamiętania. Dalszą analizę pracy przerzutnika (wzorowaną na analizie pracy przerzutnika *rs*) pozostawia się Czytelnikowi.

Odmienne oznaczenie wejść w obu przerzutnikach ma określony cel, mimo że zarówno wejście **s**, jak i $\overline{\mathbf{s}}$ są to wejścia ustawiające, a wejście **r** i $\overline{\mathbf{r}}$ są to wejścia zerujące. Dla przerzutnika *rs* **poziomem aktywnym** sygnałów wejściowych jest poziom wysoki **H**, natomiast **poziomem aktywnym** na wejściach przerzutnika $\overline{r}\,\overline{s}$ jest poziom niski **L**.

Poziomem aktywnym nazywamy poziom, który powoduje działanie układu określone przez rodzaj wejścia, do którego jest on doprowadzony. Aby więc wyzerować (ustawić) przerzutnik, należy na wejściu zerującym (ustawiającym) ustawić **poziom aktywny.** Dla przerzutnika *rs* będzie to poziom **H**, a dla przerzutnika $\overline{r}\,\overline{s}$ — poziom **L**.

Powyższa zasada oznaczania wejść jest obowiązująca dla wszystkich układów cyfrowych.

Zestawienie stanów stabilnych przerzutnika zbudowanego z bramek NAND zawiera tabl. 7.2.

Tablica 7.2. **Stany stabilne przerzutnika zbudowanego z bramek NAND**

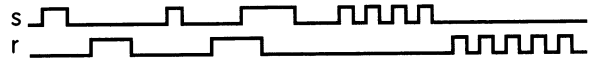
$\overline{r}$	$\overline{s}$	Q $\overline{Q}$	Stan stabilny	
0	0	1 1	stan wejść logicznie zabroniony	
0	1	0 1	przerzutnik ustawiony w stan 0	
1	0	1 0	przerzutnik ustawiony w stan 1	
1	1	0 1	pamiętanie stanu 0	stan pamiętania
		1 0	pamiętanie stanu 1	

Przerzutniki asynchroniczne są używane w układach wejściowych jako elementy pośredniczące pomiędzy zestykiem a wejściami układów cyfrowych. Rolą ich jest filtracja drgań zestyków. Przykłady takich zastosowań omówiono w p. 9.3.

Pytania i zadania

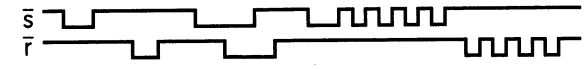
1. Narysuj układ przerzutnika asynchronicznego zbudowanego z bramek:
   a) NOR,
   b) NAND.
2. Wyjaśnij, co oznacza sformułowanie, że stan wejść **11** jest w przerzutniku *rs* logicznie zabroniony.

3. Narysuj przerzutnik zbudowany z bramek NAND i opisz sygnały na jego wejściach i wyjściach w stanie pamiętania:
   a) **1**,
   b) **0**.
4. Na podstawie analizy schematu logicznego przerzutnika zbudowanego z bramek NAND narysuj przykładowe przebiegi czasowe w tym układzie.
5. Dane są przebiegi czasowe (rys. 7.6) sygnałów na wejściach przerzutnika zbudowanego z bramek NOR. Narysuj przebiegi czasowe na obu wyjściach przerzutnika.



Rys. 7.6. Przebiegi czasowe do zadania 5.

6. Dane są przebiegi czasowe (rys. 7.7) sygnałów na wejściach przerzutnika zbudowanego z bramek NAND. Narysuj przebiegi czasowe na obu wyjściach przerzutnika.



Rys. 7.7. Przebiegi czasowe do zadania 6.

7. Zbuduj z elementów stykowych układ pamięciowy (przerzutnik), którego działanie można opisać równaniem
   a) $Q = (z + q)\bar{w}$,
   b) $Q = z + q\bar{w}$.

   Uwaga. W przyjętej przez nas konwencji duża litera oznacza cewkę przekaźnika, a mała litera — jego zestyk.

   Na podstawie analizy obu układów określ, który z nich ma priorytet załączania (z), a który priorytet wyłączania (w).

7.3. Przerzutniki synchroniczne

7.3.1. Rodzaje przerzutników

Przerzutniki synchroniczne scalone mają: **wejścia informacyjne, wejścia programujące i wejścia synchronizujące.** W odróżnieniu od przerzutników asynchronicznych oddziaływanie stanu wejść informacyjnych na stan przerzutnika jest możliwe tylko w obecności impulsu synchronizującego (zegarowego) doprowadzonego do wejścia **C**.

Stany wejść programujących oddziałują asynchronicznie, tzn. niezależnie od obecności czy też braku jakichkolwiek innych sygnałów wejściowych, także sygnału zegarowego. Od strony tych wejść przerzutnik zachowuje się dokładnie tak, jak omówiony wcześniej przerzutnik asynchroniczny.

Działanie przerzutnika będziemy opisywać za pomocą tzw. **tablicy wzbudzeń** lub **tablicy przejść**. Używa się także opisu za pomocą przebiegów czasowych, podobnie jak w przypadku przerzutników asynchronicznych.

Tablica wzbudzeń ***określa, jaki powinien być stan wejść informacyjnych, aby przerzutnik przeszedł z jednego stanu do drugiego.***

Tablica przejść ***określa, jaki będzie kolejny stan przerzutnika w zależności od aktualnego stanu przerzutnika i od aktualnego stanu jego wejść.***

W opisach tych stan aktualny (bieżący, występujący przed pojawieniem się kolejnego impulsu zegarowego) będziemy oznaczać przez **Q**, a stan kolejny (następny — ustawiany w chwili kolejnego impulsu synchronizującego) przez $\mathbf{Q}^+$. Można spotkać także inne oznaczenia, np.: **Q** i **Q'** lub $\mathbf{Q}^n$ i $\mathbf{Q}^{n+1}$.

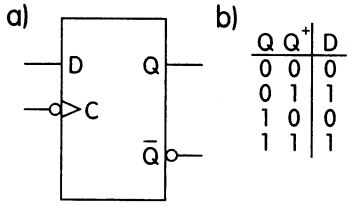
W opisie działania poszczególnych typów przerzutników pominięto wejścia programujące. Uczyniono tak z dwóch powodów:

1. Ponieważ przerzutnik synchroniczny np. typu *JK* musi mieć wejścia synchronizowane **J** i **K** oraz wyjście **Q**, natomiast nie musi mieć jakichkolwiek wejść programujących.
2. Oddziaływanie wejść programujących (o ile takowe przerzutnik ma) na stan przerzutnika jest niezależne od sygnału synchronizującego. Uwzględnianie ich w podstawowym opisie działania przerzutnika zbytnio skomplikowałoby ten opis. A ponadto są produkowane przerzutniki mające wejścia ustawiające oraz zerujące, albo tylko zerujące. Dla tych wejść stanem aktywnym może być stan niski lub wysoki, co odczytujemy z symboli graficznych.

Opis działania przerzutnika nie zależy także (poza opisem za pomocą przebiegów czasowych) od rodzaju aktywnego zbocza sygnału synchronizującego. Dlatego przyjęto, że przerzutniki działają na zbocze ujemne, o czym informuje nas kółko+trójkąt rysowane przy wejściu zegarowym. Dokładne zasady oznaczania wejść zegarowych zostaną podane później, przed omówieniem konkretnych przerzutników scalonych. Oczywiście przy opisie określonych układów scalonych będą uwzględniane wszystkie wejścia i wyjścia, jakie dany przerzutnik ma oraz będzie stosowana odpowiednia symbolika.

- **Przerzutnik synchroniczny typu *D***

Przerzutnik ma jedno **wejście informacyjne D** i **wejście zegarowe C**(rys. 7.8a). W opisie działania przerzutnika nie występuje w sposób jawny sygnał zegarowy (rys. 7.8b). Jest to cechą wszystkich metod opisu w postaci tablic układów synchronicznych, a przerzutnik jest najprostszym układem synchronicznym. ***Istnienie i oddziaływanie* impulsu zegarowego *jest ukryte w zapisie* $\mathbf{Q} \rightarrow \mathbf{Q}^+$. *Przejście to bowiem dokonuje się właśnie synchronicznie z przebiegiem zegarowym.*** Poszczególne wiersze tablicy na rys. 7.8b należy czytać następująco: Przerzutnik pozostaje w stanie **0**, gdy na wejściu **D** jest stan **0**; przerzutnik przechodzi ze stanu **0** do stanu **1**, gdy na wejściu **D** jest **1**, itd. Określenia te odnoszą się do chwili wyznaczonej przez przebieg synchronizujący.



Rys. 7.8. Przerzutnik synchroniczny typu *D*: a) symbol graficzny; b) tablica wzbudzeń

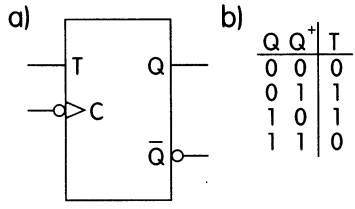
Z tablicy wynika, że $Q^+ = D$. Na wyjściu przerzutnika pojawia się to, co jest na jego wejściu, ale dopiero w chwili wystąpienia impulsu zegarowego. Dlatego przerzutnik *D* jest nazywany **elementem opóźniającym.**

- **Przerzutnik synchroniczny typu *T***

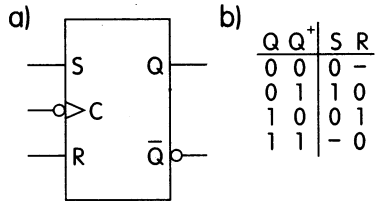
Symbol graficzny oraz tablicę wzbudzeń przerzutnika przedstawiono na rys. 7.9. Przerzutnik ma jedno **wejście informacyjne** oznaczane literą **T** i **wejście zegarowe C**.

- **Przerzutnik synchroniczny typu *RS***

Symbol graficzny oraz tablicę wzbudzeń przerzutnika przedstawiono na rys. 7.10. Przerzutnik ma dwa **wejścia informacyjne** oznaczane literami **R** i **S** oraz **wejście zegarowe C**. W przerzutniku tym stan wejść **11** jest logicznie zabroniony, podobnie jak w przerzutniku asynchronicznym *rs* zbudowanym z bramek NOR.



Rys. 7.9. Przerzutnik synchroniczny typu *T*: a) symbol graficzny; b) tablica wzbudzeń

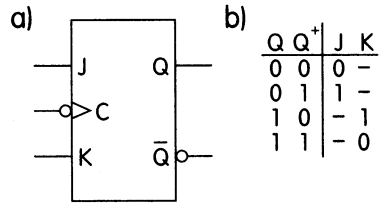


Rys. 7.10. Przerzutnik synchroniczny typu *RS*: a) symbol graficzny; b) tablica wzbudzeń

- **Przerzutnik synchroniczny typu *JK***

Symbol graficzny oraz tablicę wzbudzeń przerzutnika przedstawiono na rys. 7.11.

Przerzutnik ma dwa **wejścia informacyjne** oznaczane literami **J** i **K** oraz **wejście zegarowe C**. Wejście **J = 1** ustawia przerzutnik w stan **1**, a wejście **K = 1** ustawia przerzutnik w stan **0**. Przerzutnik ten jest inną (poprawioną) wersją przerzutnika *RS*. Stan wejść **11** nie jest w nim zabroniony. Przerzutnik ten przy stanie wejść **11** zmienia swój stan na przeciwny.



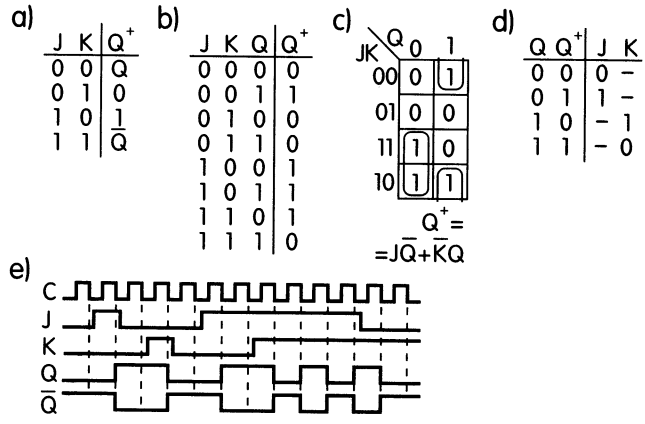
Rys. 7.11. Przerzutnik synchroniczny typu *JK*: a) symbol graficzny; b) tablica wzbudzeń

- **Metody opisu działania przerzutników**

Poznaliśmy już dwie metody opisu działania przerzutników: przebiegi czasowe — opisujące działanie przerzutników asynchronicznych oraz tablice wzbudzeń — opisujące działanie przerzutników synchronicznych. W literaturze technicznej można spotkać również **tablicę przejść** oraz tzw. **tablicę charakterystyczną**. Jest to pewna odmiana tablicy przejść, ale w postaci bardziej skondensowanej.

Ponadto tablica przejść jest rysowana dość często w układzie tablicy Karnaugha i na jej podstawie jest sporządzany opis bulowski przerzutnika.

Oczywiście wszystkie te sposoby opisu są sobie równoważne, czyli dysponując jednym z nich, możemy określić każdy inny. W celu porównania przedstawiono na rys. 7.12 opis synchronicznego przerzutnika typu *JK* przy użyciu wszystkich



Rys. 7.12. Przerzutnik typu *JK*: a) tablica charakterystyczna; b) tablica przejść; c) tablica przejść w układzie tablicy Karnaugha; d) tablica wzbudzeń; e) przykładowe przebiegi czasowe (zboczem aktywnym jest zbocze ujemne)

metod. W powyższych opisach **Q** oznacza stan aktualny przerzutnika, a $\mathbf{Q}^+$ stan kolejny. Zgodnie z tymi oznaczeniami pierwszy wiersz tablicy charakterystycznej (rys. 7.12a) czytamy następująco: Gdy stan wejść informacyjnych **JK = 00**, wówczas przerzutnik nie zmienia swego stanu ($\mathbf{Q}^+ = \mathbf{Q}$, czyli stan następny jest taki sam jak aktualny).

Natomiast pierwszy wiersz tablicy przejść (rys. 7.12b) czytamy: Jeżeli na wejściach **JK** jest stan **00** i przerzutnik jest w stanie **0**, to nie zmienia on swego stanu.

Sformułowania powyższe odnoszą się zawsze do chwili wyznaczonej przez przebieg zegarowy przerzutnika.

Tablica przejść w układzie tablicy Karnaugha umożliwia ponadto uzyskanie równania logicznego opisującego dany przerzutnik (rys. 7.12c).

7.3.2. Konwersja[1)] przerzutników

Jak już wspomniano, przerzutnik jest drugim podstawowym elementem (oprócz bramek) koniecznym do budowy układów sekwencyjnych. Jednocześnie sam przerzutnik też jest układem sekwencyjnym. Budując pewien układ synchroniczny sekwencyjny, możemy posłużyć się dowolnym rodzajem przerzutnika synchronicznego. Wniosek z powyższych spostrzeżeń jest więc następujący: ***Dowolny przerzutnik synchroniczny (układ synchroniczny sekwencyjny) możemy zbudować, wykorzystując do tego celu inny dowolny przerzutnik synchroniczny.***

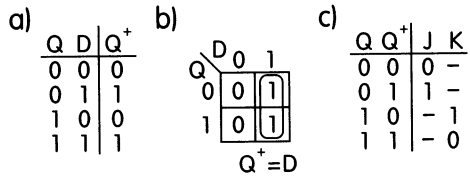
[1)] Inaczej: przekształcenie, zmiana.

Przykład 7.1

Dany jest przerzutnik typu *JK*. Zbudować z niego przerzutnik typu *D*.

Rozwiązanie

Pierwszym krokiem jest opisanie działania przerzutnika *D* (czyli projektowanego układu sekwencyjnego) za pomocą **tablicy przejść** (rys. 7.13a). Kolejnym — przekształcenie jej w **tablicę Karnaugha** (rys. 7.13b).

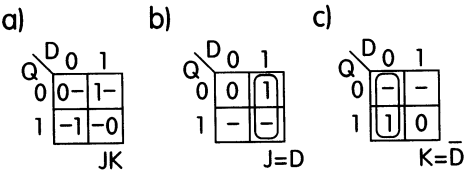


Rys. 7.13. Opis działania przerzutnika typu *D*: a) tablica przejść; b) tablica Karnaugha; c) tablica wzbudzeń przerzutnika *JK*

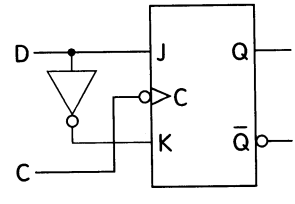
Tablica przejść określa, jak powinien zmienić się stan projektowanego układu. Zamieszczone na rys. 7.13c **tablica wzbudzeń** przerzutnika *JK* odpowiada na pytanie: Jakie sygnały należy doprowadzić do jego wejść, aby w określony sposób zmieniał się jego stan? Połączmy obie te informacje.

Z pierwszego elementu tablicy na rys. 7.13b wynika, że projektowany układ powinien w aktualnym stanie $\mathbf{Q = 0}$ i dla aktualnego stanu wejścia $\mathbf{D = 0}$ pozostać w stanie $\mathbf{Q = 0}$. Takie działanie przerzutnika *JK* (będącego podstawą budowanego układu) zapewnia doprowadzenie do jego wejść sygnałów $\mathbf{JK} = (\mathbf{0}-)$ (co odczytujemy z tablicy wzbudzeń). Wynik tego spostrzeżenia notujemy w tablicy przejść w miejsce rozważanego elementu, tworząc w ten sposób tablicę wzbudzeń projektowanego układu (rys. 7.14a). Z tablicy na rys. 7.13b wynika, że projektowany układ powinien przy sygnale wejściowym $\mathbf{D = 1}$ przejść ze stanu $\mathbf{Q = 0}$ do stanu **1**. Takie zachowanie się przerzutnika *JK* zapewnia doprowadzenie do jego wejść sygnałów $\mathbf{JK} = (\mathbf{1}-)$. Notujemy to w tworzonej tablicy (rys. 7.14a) itd.

W każdej kratce tablicy wzbudzeń (rys. 7.14a) mamy zapisaną parę bitów odpowiadających wejściom informacyjnym przerzutnika *JK*. Aby ułatwić minimalizację, dogodnie jest rozdzielić tę tablicę na dwie: jedną dla wejścia **J**, drugą dla wejścia **K** (rys. 7.14b, c). Z tablic tych wyznaczamy funkcje wzbudzeń wejścia **J** i wejścia **K** przerzutnika *JK*. Możemy już narysować schemat logiczny układu będącego rozwiązaniem zadania (rys. 7.15).



Rys. 7.14. Tablice wzbudzeń do przykładu 7.1



Rys. 7.15. Przerzutnik typu *D* zbudowany z przerzutnika typu *JK*

7.3.3. Konwersja przerzutnika w dwójkę liczącą

Jednymi z najczęściej budowanych układów sekwencyjnych są liczniki. **Licznikiem *nazywamy układ cyfrowy służący do zliczania i pamiętania liczby impulsów wejściowych.*** Najczęściej buduje się liczniki, które zliczają impulsy wejściowe w sposób określany jako zliczanie **modulo *n*** (w skrócie **mod *n*).** Tak zlicza licznik, który powraca do swojego stanu początkowego po *n* impulsach wejściowych. Nazwa pochodzi od operacji matematycznej o tej samej nazwie. W matematyce ***a* mod *b*** jest resztą z dzielenia (całkowitego) liczb naturalnych ***a*** przez ***b***. Na przykład: 24 mod 7 = 3. Podobnie licznik **mod 7** wskaże 3 po podaniu na jego wejście 24 impulsów (o ile zliczanie rozpoczęto po wyzerowaniu licznika).

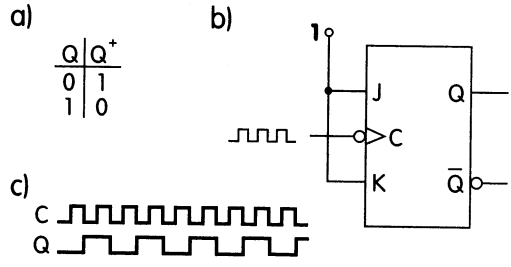
Licznik zliczający **mod 2** jest nazywany **dwójką liczącą.** Licznik taki ma dwa stany (**0** i **1**), w których na przemian się znajduje.

Przykład 7.2

Zbudować dwójkę liczącą z przerzutnika *JK*.

Rozwiązanie

Działanie takiego licznika można opisać za pomocą **tablicy przejść** (rys. 7.16a). Przejścia powinny dokonywać się w takt zliczanych zboczy sygnału wejściowego. Dlatego dwójkę liczącą będziemy budować wykorzystując przerzutnik synchroniczny, przy czym przebieg impulsów zliczanych będzie jednocześnie sygnałem zegarowym przerzutnika.



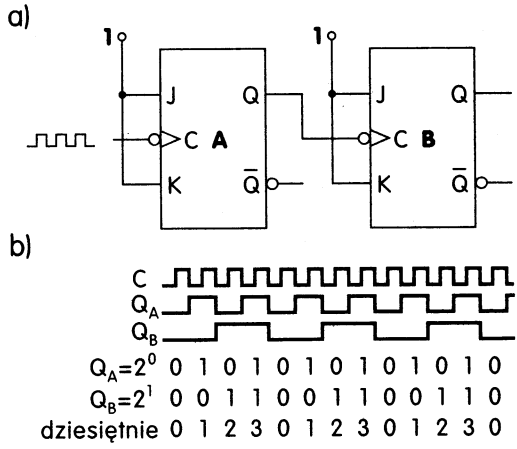
Rys. 7.16. Rysunek do przykładu 7.2: a) tablica przejść licznika modulo 2; b) schemat logiczny dwójki liczącej zbudowanej z przerzutnika *JK*; c) przebiegi czasowe na wejściu i na wyjściu układu (przy założeniu, że zboczem aktywnym jest zbocze ujemne)

Łatwo zauważyć w tablicy przejść licznika (rys. 7.16a), że $\mathbf{Q^+ = -Q}$. Porównajmy ten zapis z czwartym wierszem tablicy na rys. 7.12a. Wniosek jest następujący: ***Aby przerzutnik JK zachowywał się jak dwójka licząca, należy oba jego wejścia ustawić w stan wysoki*** **H**. Na rysunku 7.16b przedstawiono schemat logiczny dwójki liczącej zbudowanej z przerzutnika *JK*, a na rys. 7.16c — przebiegi czasowe. ■

Zauważmy, że częstotliwość przebiegu wyjściowego jest dwukrotnie mniejsza niż częstotliwość wejściowa. Licznik **mod 2** jest więc także dzielnikiem częstotliwości przez dwa. Uogólniając: **Licznik mod *n* jest dzielnikiem częstotliwości przez *n*.**

Zbudujmy układ z dwóch takich dwójek liczących. Niech pierwsza z nich będzie źródłem sygnału wejściowego dla drugiej. Drugi przerzutnik będzie więc dzielił częstotliwość przebiegu wyjściowego pierwszego przerzutnika. Odpowiedni schemat logiczny oraz przebiegi czasowe przedstawiono na rys. 7.17.

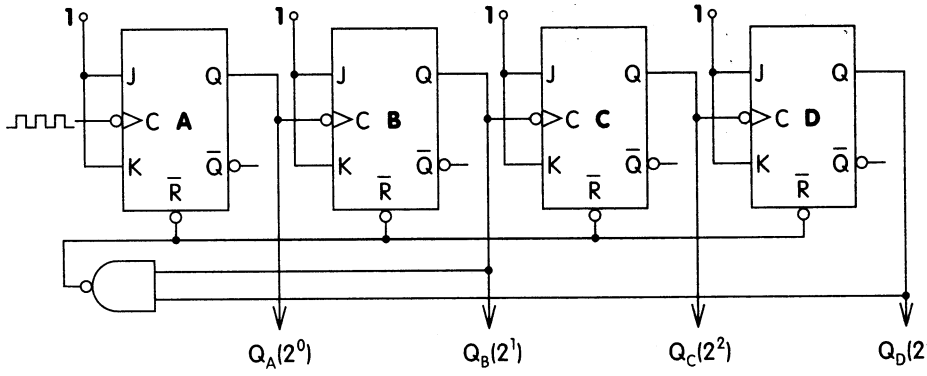
Łatwo zauważyć (rys. 7.17b), że taki układ zlicza impulsy wejściowe w trybie **mod 4**. Rozbudowując układ o kolejną dwójkę liczącą otrzymamy licznik **mod 8**, a dodając czwartą — licznik **mod 16**.



Rys. 7.17. Licznik modulo 4 (a) oraz przebiegi czasowe (b)

Częściej jednak korzysta się z liczników **mod 10** (tzw. **dekad**). Licznik taki można zbudować z licznika **mod 16,** stosując odpowiednie sprzężenie zwrotne zerujące licznik (rys. 7.18).

Licznik **mod 16** ze stanu $\mathbf{Q_DQ_CQ_BQ_A = 1001}$ (dziesiętnie 9) przechodzi do stanu $\mathbf{Q_DQ_CQ_BQ_A = 1010}$ (dziesiętnie 10). Licznik dziesiętny powinien zaś ze stanu $\mathbf{Q_DQ_CQ_BQ_A = 1001}$ przejść do stanu $\mathbf{Q_DQ_CQ_BQ_A = 0000}$. W układzie jak na rys. 7.18, jeżeli na wyjściach $\mathbf{Q_D}$ i $\mathbf{Q_B}$ pojawią się jedynki, to spowodują, że na wyjściu bramki NAND pojawi się poziom niski **0**. Sygnał ten wyzeruje natych-



Rys. 7.18. Licznik modulo 10 (dekada) zbudowany z licznika modulo 16

miast (z opóźnieniem wynikającym z czasu propagacji przerzutnika) wszystkie przerzutniki, ustawiając tym samym licznik w stan $\mathbf{Q_DQ_CQ_BQ_A = 0000}$. W efekcie na wyjściu bramki NAND ponownie zostanie ustawiony stan **1**, a kolejny impuls zerujący wystąpi po następnych dziesięciu impulsach. W taki sposób można uzyskać dowolny licznik **mod** k (dzielnik częstotliwości przez k) z licznika **mod** n (dla $k < n$).

Licznik taki ma jednak następujące wady:

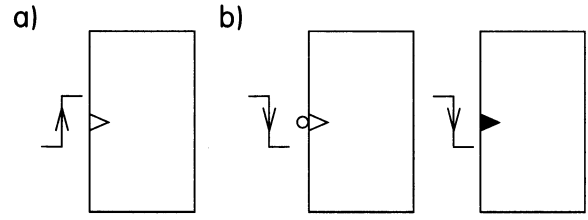
1. Przy przejściu ze stanu $\mathbf{Q_DQ_CQ_BQ_A = 1001}$ do stanu $\mathbf{Q_DQ_CQ_BQ_A = 0000}$ na pewien czas (czas propagacji bramki plus czas propagacji przerzutnika) pojawi się stan $\mathbf{Q_DQ_CQ_BQ_A = 1010}$. Na wyjściu $\mathbf{Q_B}$ pojawi się więc krótki impuls dodatni (hazard dynamiczny — patrz p. 9.1), który w pewnych zastosowaniach może być źródłem nieprawidłowej pracy systemu cyfrowego.
2. Szybkość (maksymalna częstotliwość przebiegu wejściowego) takiego licznika jest nieduża. Na przykład przy przejściu licznika ze stanu $\mathbf{Q_DQ_CQ_BQ_A =}$ $\mathbf{= 0111}$ do stanu $\mathbf{Q_DQ_CQ_BQ_A = 1000}$ kolejny zliczany impuls sprawi, że przerzutnik A zmieni swój stan (z **1** na **0**). Sygnał $\mathbf{Q_A}$ jest jednocześnie przebiegiem zegarowym przerzutnika B. Z opóźnieniem (równym czasowi propagacji) zmieni swój stan przerzutnik B itd. Ostatecznie kolejny stan licznika ustali się po czasie równym sumie czasów propagacji wszystkich przerzutników.
3. Niekiedy, aby impuls wyjściowy z bramki (zerujący licznik) był zawsze skuteczny, należy dodać jeszcze odpowiedni układ formujący, co dodatkowo komplikuje układ. Sytuacja taka może wystąpić wówczas, gdy zerowanie dotyczy kilku przerzutników i wskutek różnych czasów propagacji jeden z nich ustawi się wcześniej w stan niski. Zniknie wtedy impuls zerujący i stan licznika (różny jeszcze od stanu wyzerowania) może się utrwalić.

Z wymienionych powyżej powodów nie buduje się liczników w sposób pokazany na rys. 7.18.

Liczniki budowane jako połączenie dwójek liczących są nazywane **licznikami szeregowymi** ***lub*** **asynchronicznymi.** Druga nazwa może być nieco myląca, biorąc pod uwagę, że licznik jest zbudowany z przerzutników synchronicznych. **Liczniki scalone** są budowane jako **szeregowe** (asynchroniczne) lub jako **równoległe**. ***W liczniku równoległym sygnał zegarowy (będący dla licznika zawsze przebiegiem impulsów zliczanych) jest doprowadzony jednocześnie do wejść synchronizujących wszystkich przerzutników.*** Pojawienie się kolejnego impulsu zliczanego sprawia, że wszystkie przerzutniki jednocześnie (współbieżnie) przetwarzają informację wejściową i czas ustalania się kolejnego stanu licznika wyznacza przerzutnik o najdłuższym czasie propagacji. Licznik taki jest znacznie szybszy od licznika szeregowego, jednak jego struktura jest bardziej złożona. Liczniki scalone omówiono w rozdz. 12.

7.3.4. Przerzutniki scalone serii 74

Do tej pory przyjmowaliśmy, że przerzutniki synchroniczne są wyzwalane zboczem ujemnym, co znajdowało swój wyraz w rysowaniu kółeczka i trójkąta przy wejściu zegarowym. ***W praktyce możemy spotkać przerzutniki wyzwalane zboczem ujemnym, dodatnim lub poziomem.*** Dla oznaczenia sposobu wyzwalania danego przerzutnika stosuje się symbole przedstawione na rys. 7.19.



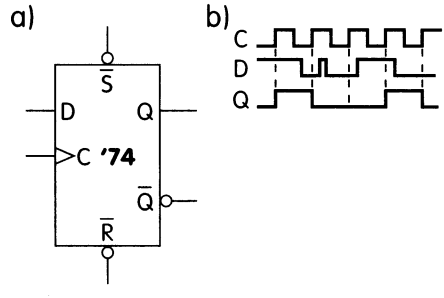
Rys. 7.19. Oznaczanie dynamicznego wejścia zegarowego przerzutnika wyzwalanego zboczem: a) dodatnim; b) ujemnym

Mały trójkącik rysowany na wejściu oznacza **wyzwalanie przerzutnika zboczem.** Jeśli zboczem synchronizującym jest zbocze ujemne (opadające), to jest rysowane dodatkowo kółeczko lub trójkącik ten jest zaczerniony. **Przerzutnik wyzwalany poziomem** będzie rysowany bez trójkącika. Jeżeli poziomem aktywnym będzie poziom niski, to na takim wejściu rysowane będzie kółeczko.

- **Przerzutniki scalone typu *D***

Układ scalony **'74** (np. UCY7474) zawiera dwa przerzutniki typu *D*, wyzwalane zboczem dodatnim. Każdy z przerzutników ma dwa **wejścia przygotowujące,** oddziałujące na przerzutnik asynchronicznie. Wejście $\overline{\mathbf{S}}$ — ustawiające przerzutnik w stan wysoki **H** oraz wejście $\overline{\mathbf{R}}$ — ustawiające przerzutnik w stan niski **L**. Wejścia te są opisywane symbolami **S** i **R** z negacjami dla zaznaczenia, że odpowiadająca danemu wejściu funkcja jest realizowana po doprowadzeniu do niego sygnału **0**. Aby ustawić przerzutnik w stan **L**, należy doprowadzić sygnał **0** do wejścia $\overline{\mathbf{R}}$, natomiast doprowadzenie **0** do wejścia $\overline{\mathbf{S}}$ ustawia przerzutnik w stan wysoki **H**. **Stan wejść $\overline{\mathbf{R}}\overline{\mathbf{S}}$ = 00 jest logicznie zabroniony.** Przerzutnik ten od strony wejść $\overline{\mathbf{R}}\overline{\mathbf{S}}$ zachowuje się dokładnie tak, jak przerzutnik asynchroniczny zbudowany z bramek NAND. Na obu wyjściach komplementarnych jest wówczas (przy $\overline{\mathbf{R}}\overline{\mathbf{S}}$ = **00**) stan wysoki **H**. Wejścia $\overline{\mathbf{R}}\overline{\mathbf{S}}$ są dominujące i przerzutnik zachowuje się w ten sposób niezależnie od stanu wejść **D** i **C**.

Symbol graficzny przerzutnika oraz przykładowe przebiegi czasowe przedstawiono na rys. 7.20. Opis działania przerzutnika w postaci tablicy wzbudzeń ilustruje rys. 7.8.

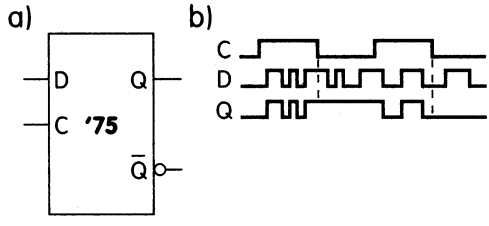


Rys. 7.20. Przerzutnik scalony typu '74: a) symbol graficzny; b) przebiegi czasowe (przykładowe)

Zwróćmy uwagę na zasady obowiązujące przy rysowaniu przebiegów czasowych w układach synchronicznych. Częstym błędem jest rysowanie zmiany sygnału wejściowego w tym samym momencie, co zmiana sygnału synchronizującego (co zbocze aktywne). Jest to bardzo istotny błąd, gdyż ***uniemożliwia określenie zachowania się układu — nie wiadomo bowiem, jaki stan wejść oddziałuje na układ, skoro się on zmienia.*** Z parametrów dynamicznych (patrz p. 7.4) wynika bowiem, że stan wejść synchronizowanych powinien być ustalony na pewien czas przed wystąpieniem zbocza zegarowego, aby układ ten stan zaobserwował. Wynika z tego, że stan wejścia zmieniającego się wraz ze zboczem aktywnym przebiegu zegarowego będzie odczytany jako stan sprzed zmiany, czyli jak gdyby zmiana nastąpiła dopiero po impulsie zegarowym. Ze względów praktycznych, aby nie utrudniać interpretacji przebiegów czasowych, lepiej unikać takich sytuacji i przyjmować, iż wszelkie zmiany stanów wejść synchronizowanych dokonują się poza aktywnymi zboczami przebiegu zegarowego.

Zmiany sygnałów wyjściowych* Q *natomiast należy przyjmować jako jednoczesne ze zboczem synchronizującym przebiegu zegarowego. Wprawdzie są one opóźnione o czas propagacji, ale przy rozpatrywaniu przebiegów czasowych, opisujących działanie układu synchronicznego, zwykle się je pomija. W przypadkach szczególnych, gdy jest wymagana pogłębiona analiza działania układu — ponieważ szybkość jego pracy jest na tyle duża, że opóźnienia mogą mieć wpływ na jego pracę — uwzględnia się także te opóźnienia.

Układ scalony **'75** zawiera cztery przerzutniki *D* wyzwalane poziomem. Na zewnątrz są wyprowadzone dwa wejścia zegarowe, przy czym każde z nich jest połączone z wejściami zegarowymi dwóch przerzutników. W układzie nie ma wejść



Rys. 7.21. Przerzutnik typu D „zatrzask" ('75): a) symbol graficzny; b) przebiegi czasowe (przykładowe)

przygotowujących. Działanie przerzutnika jest następujące: Dopóki na wejściu zegarowym jest stan wysoki **H**, dopóty sygnał z wejścia **D** bezpośrednio oddziałuje na wyjście (**Q** = **D**). Jeżeli na wejściu zegarowym jest stan niski **L**, to wyjście **Q** jest w takim stanie, w jakim było wejście **D** w chwili zmiany sygnału zegarowego z **1** na **0**. Przerzutnik taki jest nazywany **przerzutnikiem typu „zatrzask"** (ang. *latch*). Jego symbol graficzny oraz przebiegi czasowe pokazano na rys. 7.21.

Układ scalony **'013** (np. MCY74013) (CMOS) zawiera dwa przerzutniki typu *D-MS* (sposób działania przerzutnika określanego mianem *MS* zostanie wyjaśniony poniżej — przy opisie przerzutników *JK*). Przerzutnik ten (**'013**) zmienia swój

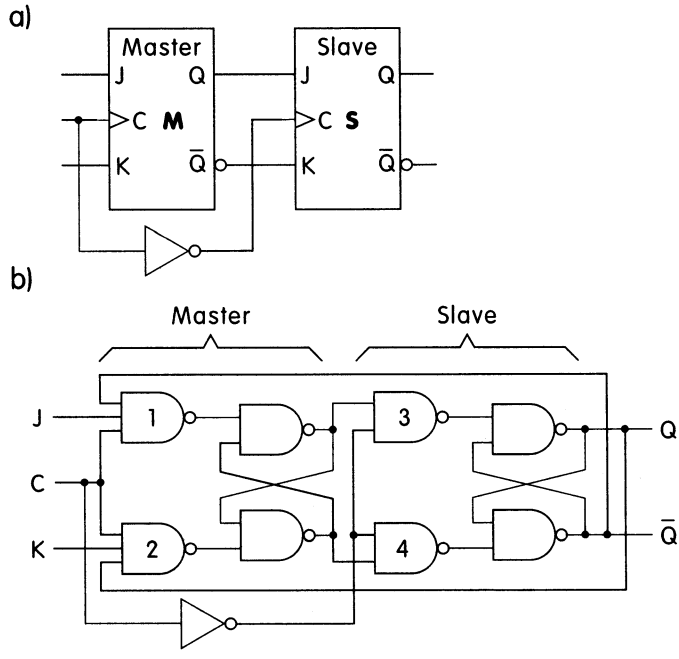
stan przy dodatnim zboczu sygnału zegarowego. Ma on dwa wejścia asynchroniczne przygotowujące typu **RS**. Jedynka na jednym z wejść ustawia przerzutnik w stan **0** (**R** = **1**) lub w stan **1** (**S** = **1**). Stan wejść **RS** = **11** jest logicznie zabroniony.

- **Przerzutniki scalone typu *JK***

Przerzutnik synchroniczny może być synchronizowany zboczem dodatnim lub ujemnym, poziomem (typu „zatrzask") lub może być **przerzutnikiem dwutaktowym, *co oznacza, że do ustawienia stanu przerzutnika są wymagane dwa kolejne zbocza impulsu zegarowego (tzn. pojedynczy impuls prostokątny).***

Przerzutnik dwutaktowy *JK* działa w ten sposób, że w czasie pierwszego zbocza (narastającego) są próbkowane stany wejść **J** i **K**, drugie zbocze (opadające) powoduje zgodną z tablicą przejść zmianę stanu przerzutnika. W rezultacie zmianę stanu obserwujemy przy opadającym zboczu impulsu zegarowego i dlatego symbol graficzny takiego przerzutnika ma na wejściu zegarowym trójkącik i symbol negacji (kółeczko). Kolejność zboczy może być odwrócona, czyli czytanie wejść może być przy zboczu ujemnym, a zmiana stanu wyjść przy zboczu dodatnim. Przykładem takiego przerzutnika może być wspomniany powyżej, wykonany w technice CMOS, przerzutnik *D-MS* typu **'013**. Wejście zegarowe tego przerzutnika (na symbolu graficznym) będziemy więc rysować bez kółeczka. Nie wprowadzono odrębnego symbolu graficznego dla wejść zegarowych dwuzboczowych.

Przerzutnik dwutaktowy *składa się z dwu przerzutników połączonych kaskadowo. Pierwszy z nich nosi nazwę Master (M), drugi Slave (S).* (Po angielsku *Master-Slave*, to po polsku mistrz (pan)-sługa). Schemat zasadniczy i logiczny przerzutnika *JK* przedstawiono na rys. 7.22.



Rys. 7.22. Przerzutnik typu *JK-MS*: a) schemat zasadniczy; b) schemat logiczny

Zmiana stanu przerzutnika *M* odbywa się podczas zmiany poziomu z **0** na **1** na wejściu zegarowym, natomiast przepisanie informacji z przerzutnika *M* do *S* zachodzi podczas zmiany poziomu na tym wejściu z **1** na **0**. ***Jednakże — dla zapewnienia poprawnej pracy przerzutnika — sygnały na wejściach informacyjnych powinny być ustalone przez cały czas trwania impulsu zegarowego.*** Zmiana stanu wejść *JK* podczas wysokiego poziomu sygnału na wejściu zegarowym może spowodować niezgodne z tablicą przejść działanie przerzutnika.

Układ scalony UCY7473 zawiera dwa niezależne przerzutniki typu *JK-MS*, bez wyprowadzonych wejść ustawiających **S**. Układ UCY74107 różni się od poprzedniego jedynie konfiguracją wyprowadzeń. Układ scalony UCY7476 zawiera także dwa przerzutniki *JK-MS*, ale ma wyprowadzone obydwa wejścia przygotowujące. Układ UCY7472 zawiera tylko jeden przerzutnik *JK-MS*, ale jest to przerzutnik wielowejściowy. Przerzutnik ma potrójne wejścia **J** (**J**1, **J**2, **J**3) i **K** (**K**1, **K**2, **K**3). Działanie układu określają sygnały **JK**, będące iloczynami określonymi następująco: **J** = **J**1**J**2**J**3 i **K** = **K**1**K**2**K**3.

Układ scalony MCY74027 zawiera dwa przerzutniki typu *JK-MS*. Zmiana stanu przerzutnika następuje przy dodatnim zboczu sygnału zegarowego. Wejścia przygotowujące (asynchroniczne) są typu **RS**.

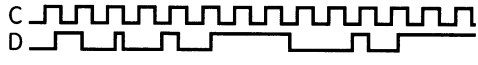
W układach cyfrowych zawierających przerzutniki (tzn. w układach sekwencyjnych) należy się liczyć z możliwością przypadkowego ustalenia stanu układu (stanów poszczególnych przerzutników) po włączeniu zasilania. Wobec tego na ogół wymaga się sprowadzenia układu do stanu początkowego przed rozpoczęciem pracy. Także w czasie pracy rozpoczęcie nowego zliczania przez licznik powinno być poprzedzone jego wyzerowaniem. Takie wstępne przygotowanie układu do pracy osiąga się zwykle za pomocą pojedynczego impulsu podanego do wejść asynchronicznych (przygotowujących) przerzutników i wymuszenie pożądanego stanu na ich wyjściach. Układy generujące takie impulsy omówiono w rozdz. 8.

Pytania i zadania

1. Opisz za pomocą tablicy charakterystycznej, tablicy przejść i tablicy przejść w układzie tablicy Karnaugha przerzutnik:
   a) typu *D*, b) typu *T*, c) typu *RS*.

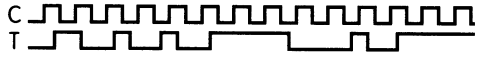
   Na podstawie tablicy przejść w układzie tablicy Karnaugha zapisz równanie charakterystyczne przerzutnika.

2. Narysuj przebiegi czasowe na wyjściu przerzutnika typu *D* dla danego przebiegu sygnału **D** i sygnału zegarowego (rys. 7.23). Przerzutnik jest synchronizowany dodatnim zboczem przebiegu zegarowego.



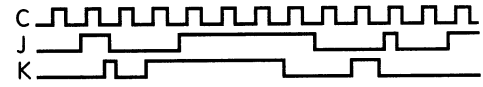
Rys. 7.23. Przebiegi czasowe do zadania 2.

3. Dane są przebiegi sygnału **T** oraz sygnału zegarowego **C** przerzutnika typu *T* (rys. 7.24). Narysuj przebieg czasowy na wyjściu **Q**. Przerzutnik zmienia swój stan przy opadającym zboczu sygnału zegarowego.



Rys. 7.24. Przebiegi czasowe do zadania 3.

4. Dane są sygnały **J** i **K** oraz przebieg zegarowy **C** przerzutnika typu *JK-MS* (rys. 7.25). Przerzutnik przełącza przy ujemnym zboczu sygnału synchronizującego **C**. Narysuj przebieg czasowy na wyjściu **Q** przerzutnika.



Rys. 7.25. Przebiegi czasowe do zadania 4.

5. Dokonaj konwersji przerzutnika *D* w przerzutnik *JK*.
6. Dokonaj konwersji przerzutnika *T* w przerzutnik *JK*.
7. Dokonaj konwersji przerzutnika *JK* w przerzutnik *T*.
8. Dany jest synchroniczny przerzutnik typu *D*. Zbuduj z niego przerzutnik typu *T*.
9. Dany jest synchroniczny przerzutnik typu *T*. Zbuduj z niego przerzutnik typu *D*.
10. Zbuduj z przerzutnika *D* dwójkę liczącą.
11. Zbuduj z przerzutnika *T* dwójkę liczącą.
12. Na podstawie szeregowego licznika **mod 16**, zbudowanego z przerzutników *JK*, zbuduj licznik **mod 11**. Narysuj przebiegi czasowe w tym liczniku. Na którym wyjściu wystąpi hazard dynamiczny?
13. Zbuduj dzielnik częstotliwości przez 7. Narysuj przebiegi czasowe w tym układzie.
14. Narysuj układ czterech dwójek liczących (z przerzutników *JK*) połączonych szeregowo, ale pobierających sygnał zegarowy z wyjścia $\overline{\mathbf{Q}}$ (zamiast **Q**) przerzutnika z młodszej pozycji. Narysuj przebiegi czasowe w tym układzie. Zapisz kolejne stany licznika. Skomentuj uzyskane wyniki.
15. Zaprojektuj schemat licznika równoległego (synchronicznego) zliczającego **mod 10** w kodzie **naturalnym BCD.**

Tablica przejść

S	S^+
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	1

Zakodowana tablica przejść

$Q_DQ_CQ_BQ_A$	$Q_D^+Q_C^+Q_B^+Q_A^+$
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 0 0 0

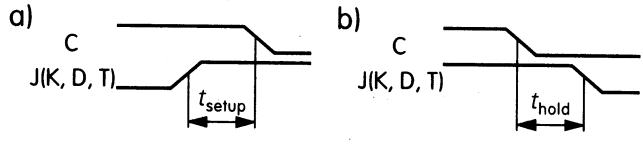
Wskazówka. Licznik **mod 10** ma 10 stanów (S). Numerujemy je np.: 1, 2, ..., 10. Każdy ze stanów licznika musi być pamiętany i aby było to możliwe, należy zbudować pamięć z 4 przerzutników. Ponieważ licznik ma zliczać w kodzie **BCD 8421**, więc należy je zakodować w taki sposób, jak w tablicy powyżej.

Dalej postępować podobnie, jak przy konwersji przerzutników. Należy jedynie określić funkcje wzbudzeń wszystkich czterech przerzutników.

7.4. Parametry dynamiczne przerzutników synchronicznych

Przerzutniki scalone — oprócz **wejść informacyjnych**, które oddziałują na stan przerzutnika tylko w przypadku doprowadzenia impulsu zegarowego — mają dodatkowo **wejścia przygotowujące**, umożliwiające ustawienie dowolnego stanu przerzutnika asynchronicznie (bez impulsu zegarowgo). Dlatego też, mówiąc o czasie propagacji przerzutnika scalonego, należy rozróżniać dwa parametry. Jeden będzie liczony od wejścia zegarowego, drugi — od wejść przygotowujących.

Jak już wspomniano w p. 7.1, z uwagi na poprawną pracę przerzutnika wymaga się, aby stan wejść informacyjnych przerzutnika był już ustalony zanim doprowadzimy impuls zegarowy. Także w czasie trwania zbocza aktywnego impulsu zegarowego stan tych wejść nie powinien się zmieniać, nawet przez pewien czas po wystąpieniu aktywnego zbocza impulsu synchronizującego. Wymagania sformułowane powyżej są określane w katalogach poprzez wartości takich parametrów, jak: **czas ustalania** — t_{setup} (krócej t_s) oraz **czas przetrzymywania** — t_{hold} (krócej t_h). **Czas ustalania** t_{setup} ***określa minimalną wartość opóźnienia zbocza wyzwalającego impulsu zegarowego w odniesieniu do zbocza ustalającego stan wejścia informacyjnego***. Innymi słowy, jest to parametr, który wskazuje, o ile wcześniej należy ustawić stan wejść (synchronizowanych) przerzutnika przed wystąpieniem zbocza aktywnego impulsu synchronizującego, aby przerzutnik zareagował na ten właśnie stan. Graficzną definicję tego parametru przedstawiono na rys. 7.26a.



Rys. 7.26. Ilustracja graficzna: a) czasu ustalania; b) czasu przetrzymywania

Czas przetrzymywania t_{hold} ***określa niezbędne minimalne opóźnienie zbocza impulsu zmieniającego stan wejścia informacyjnego w odniesieniu do aktywnego zbocza wyzwalającego impulsu synchronizującego.*** Innymi słowy, jest to parametr, który wskazuje, jak długo należy utrzymywać stan wejść informacyjnych po wystąpieniu impulsu synchronizującego, aby przerzutnik zareagował na ten właśnie stan. Graficzną definicję tego parametru przedstawiono na rys. 7.26b.

Dla przerzutnika typu *MS* (z wejściem aktywnym rysowanym jako ujemne) czas przetrzymywania będzie odmierzany od zbocza ujemnego impulsu zegarowego. Tak więc, dla tego typu przerzutników stan wejść powinien być niezmienny przez przedział czasu będący sumą: czasu ustalania, czasu trwania impulsu zegarowego i czasu przetrzymywania.

Przy omawianiu przerzutnika dwutaktowego (typu *MS*) zwrócono uwagę Czytelnika, że dla zapewnienia poprawnej pracy takiego przerzutnika wymaga się, aby stan jego wejść nie zmieniał się w czasie występowania pełnego impulsu ze-

Tablica 7.3. **Parametry dynamiczne wybranych przerzutników synchronicznych**

Rodzaj przerzutnika			74LS76[1]	7476[2]	74013[3]	74027[4]
f_{max} [MHz] (co najmniej)			30	15	3,5	3,5
C → Q	t_{pHL}		30	40	150	150
	t_{pLH}		20	40	150	150
R, S → Q	t_{pHL}		30	40	200	200
	t_{pLH}		20	40	150	150
C	t_w(H)	ns	20	30	70	70
	t_w(L)		13	20	—	—
R, S	t_w		25	35	90	100
	t_s		20	20	20	100
	t_h		0	5	—	—

[1] UCY74LS76 (lub UCY74LS112); 2 przerzutniki typu *JK* wyzwalane zboczem ujemnym.
[2] UCY7476; 2 przerzutniki typu *JK-MS*, zmiana stanu wyjść przy zboczu ujemnym.
[3] MCY74013 (U_{DD} = 5 V); 2 przerzutniki typu *D-MS*, zmiana stanu wyjść przy zboczu dodatnim.
[4] MCY74027 (U_{DD} = 5 V); 2 przerzutniki typu *JK-MS*, zmiana stanu wyjść przy zboczu dodatnim.

garowego. Z tej uwagi wysnuto wniosek, że impulsy zegarowe powinny mieć kształt szpilek (o krótkim czasie trwania poziomu wysokiego t_w(**H**) lub niskiego t_w(**L**)). Jednak czas trwania t_w takiego impulsu zegarowego nie może być dowolnie krótki. W katalogach jest podawany minimalny czas trwania takich impulsów. Także określa się minimalny czas trwania t_w impulsów doprowadzanych do wejść przygotowujących. Przerzutniki jednotaktowe również wymagają impulsów o określonym (minimalnym) czasie trwania.

Wartości parametrów dynamicznych wybranych przerzutników synchronicznych zestawiono w tabl. 7.3.

8

Układy czasowe

8.1. Wprowadzenie

Układy cyfrowe bardzo często wymagają sygnałów, których parametry czasowe, takie jak: czas trwania impulsu, częstotliwość, opóźnienie impulsu są istotne z uwagi na poprawność działania układu. Na przykład przygotowanie licznika do zliczania impulsów wymaga, aby został on wcześniej wyzerowany. Istnieje tu ściśle określone następstwo czasowe — najpierw zerowanie, a następnie otwarcie bramki dla przebiegu zliczanego. Czas otwarcia bramki w wielu przypadkach (np. w przyrządach pomiarowych) powinien być dokładnie wyznaczony. **Układami czasowymi** ***są więc układy elektroniczne przeznaczone do generacji pojedynczych impulsów lub fali prostokątnej. Są to — znane z podstaw elektroniki — przerzutniki (multiwibratory) monostabilne i astabilne.***

W rozdziale niniejszym omówiono następujące układy wykonane w postaci scalonej:

— przerzutniki monostabilne **'121** i **'123**,
— układ ULY7855 (tajmer 555),
— multiwibrator monostabilny/astabilny **'047**,
— programowany układ czasowy **'541**.

Układy uzależnień czasowych oraz generatory stanowią bardzo ważną grupę układów, występujących praktycznie w każdym urządzeniu cyfrowym, ***zwłaszcza w jego części sterującej.*** Podstawowymi układami stosowanymi do budowy jednostek sterujących są układy monostabilne i astabilne. Dlatego — po zapoznaniu się z przerzutnikami scalonymi — zajmiemy się w kolejnych podrozdziałach układami uzależnień czasowych, generatorami pojedynczych impulsów i generatorami fali prostokątnej.

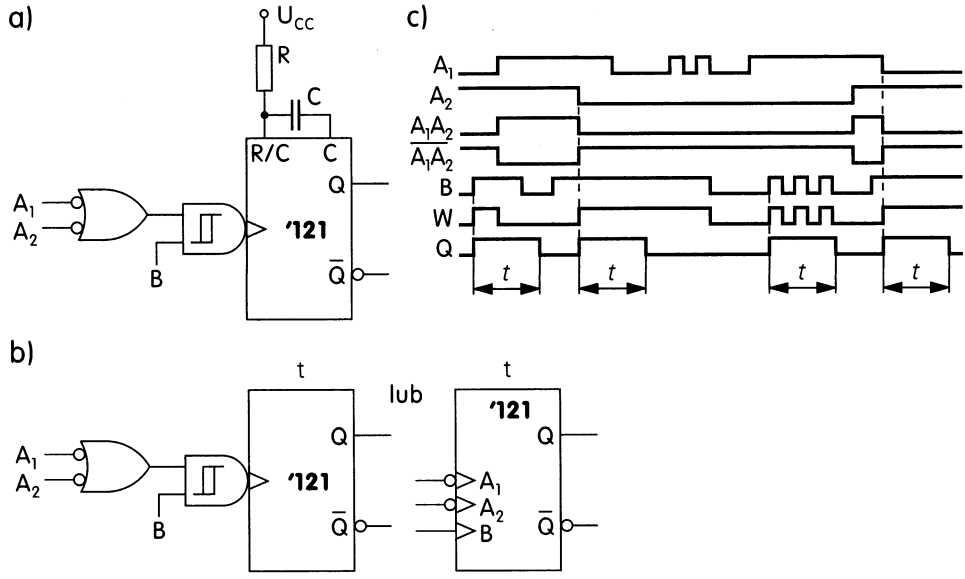
8.2. Przerzutniki monostabilne i inne elementy czasowe

Przerzutniki monostabilne charakteryzują się ***jednym stanem równowagi trwałej***. Przerzutnik taki może się znajdować dowolnie długo np. w stanie niskim.

Dopiero ingerencja z zewnątrz (doprowadzenie tzw. sygnału wyzwalającego) sprawia, że przechodzi on w stan wysoki. Jednak stan ten nie jest stanem stabilnym. Przerzutnik samoczynnie powraca do stanu początkowego. Czas, po jakim następuje ten powrót, jest zależny od parametrów *R*, *C* elementów dołączanych z zewnątrz przez użytkownika takiego układu.

8.2.1. Przerzutnik monostabilny '121

Układ scalony **'121** zawiera jeden multiwibrator monostabilny. Umożliwia on generowanie impulsów o czasie trwania od ok. 30 ns do 28 s. Układ ma dwa komplementarne wyjścia **Q** i $\overline{\mathbf{Q}}$. W stanie równowagi trwałej (stanie spoczynkowym) poziom logiczny na wyjściu **Q** jest równy **0**. Do wyzwalania przerzutnika służą trzy wejścia sterujące (zwane wejściami wyzwalającymi): wejścia **A1** i **A2** — do wyzwalania ujemnymi oraz wejście **B** — do wyzwalania dodatnimi zboczami impulsów. Minimalny czas trwania impulsu wyzwalającego wynosi 50 ns. Schemat funkcjonalny, symbol graficzny przerzutnika oraz przebiegi czasowe przedstawiono na rys. 8.1.



Rys. 8.1. Przerzutnik monostabilny UCY74121: a) schemat funkcjonalny; b) symbol graficzny; c)przebiegi czasowe
W — przebieg pomocniczy na wyjściu bramki AND $W = \overline{A1A2} \cdot B$

Układ logiczny doprowadzający sygnał do wejścia wyzwalającego przerzutnika (wejście oznaczone trójkątem niezamalowanym, co oznacza, że zboczem aktywnym jest zbocze dodatnie) realizuje funkcję **W** o postaci

$$\mathbf{W = B\,(\overline{A1} + \overline{A2}) = B\,\overline{A1A2}}$$

Narastające zbocze sygnału **W** sprawia, że na wyjściu **Q** przerzutnika jest generowany impuls o czasie trwania *t* (rys. 8.1c). Z przebiegów czasowych przedstawionych na rys. 8.1c widać ponadto, że wystąpienie kolejnych impulsów wyzwalających w czasie trwania impulsu generowanego *t* nie ma żadnego wpływu na czas impulsu wyjściowego **Q**. Mówimy, że taki przerzutnik jest **nieretrygerowalny**.

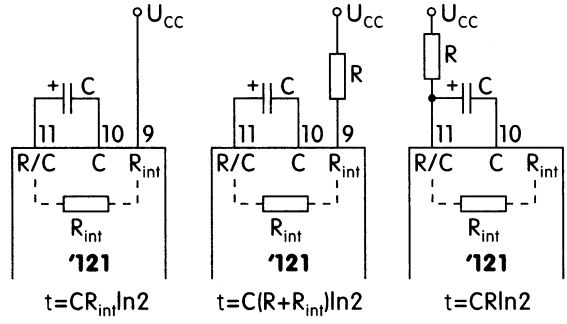
Użytkownik układu scalonego ma jednak do dyspozycji wejścia **A1**, **A2** i **B**. Analizując wyzwalanie przerzutnika od strony tych właśnie wejść, na podstawie przebiegów czasowych z rys. 8.1c można sformułować następujące wnioski:

— można wyzwolić przerzutnik ujemnym zboczem sygnału **A**, gdy na drugim wejściu **A** oraz na wejściu **B** jest stan **H**;

— można wyzwolić przerzutnik dodatnim zboczem sygnału **B**, gdy co najmniej na jednym z wejść **A** jest poziom niski **L**.

Ponieważ wejście **B** jest doprowadzone do bramki z przerzutnikiem Schmitta, zatem sygnał **B** nie musi być w standardzie TTL. Od strony tego wejścia układ **'121** można wykorzystać do kształtowania sygnałów wejściowych wolnozmiennych. Na przykład można dołączyć układ *RC* i uzyskać opóźnienie generowanego impulsu. Sygnał ukształtowany w przerzutniku monostabilnym będzie miał więc nie tylko odpowiednie zbocza, ale także pożądany czas trwania impulsów.

Czas trwania impulsu wyjściowego jest określony wyłącznie przez obwód *RC*. Przy braku zewnętrznych elementów *R*, *C* układ generuje impuls o czasie trwania ok. 30 ns. Czas ten jest uwarunkowany wartościami wewnętrznych elementów *R*, *C*. Pojemność własna układu między końcówkami 10 i 11 wynosi ok. 20 pF, a rezystancja wewnętrzna ok. 2 kΩ. Sposoby dołączenia zewnętrznych elementów *R*, *C* do końcówek przerzutnika przedstawiono na rys. 8.2.

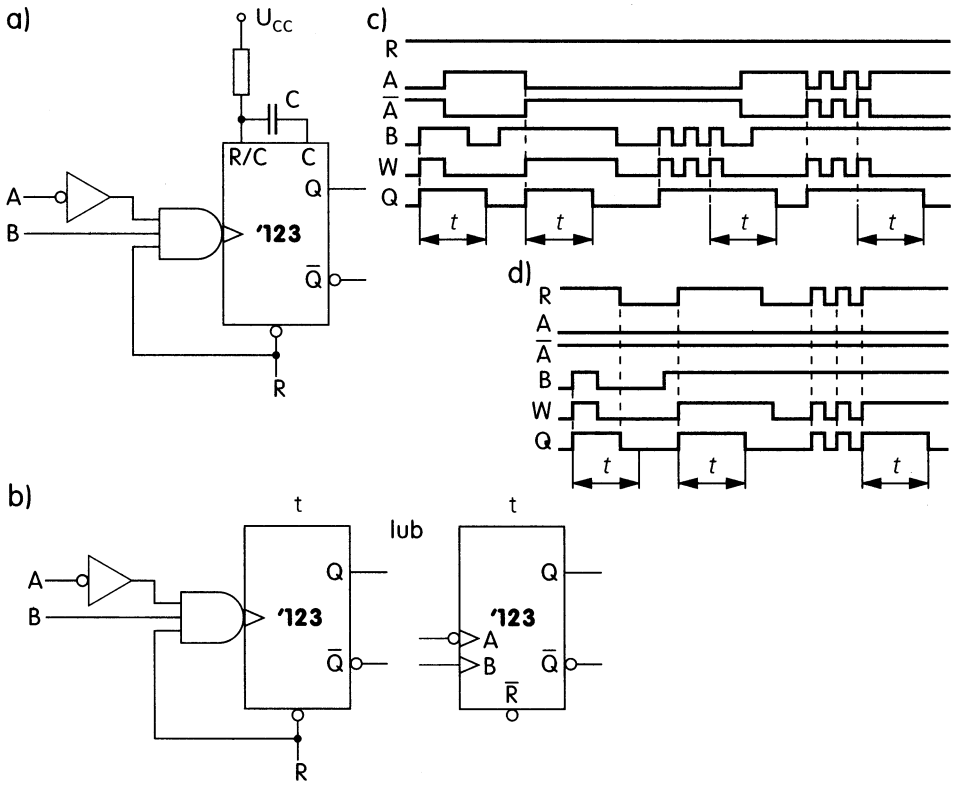


Rys. 8.2. Sposoby dołączenia zewnętrznych elementów *RC* do końcówek przerzutnika

Maksymalna szerokość impulsu wyjściowego, jaką można uzyskać w układzie **'121**, wynosi ok. 28 s. Wynika to z ograniczeń nałożonych przez producenta układu na wartości elementów zewnętrznych. Wartości pojemności powinny się zawierać w przedziale 10 pF ÷ 1000 μF, a wartości rezystancji 1,4 kΩ ÷ 40 kΩ.

8.2.2. Przerzutnik monostabilny '123

Układ scalony **'123** zawiera dwa jednakowe przerzutniki monostabilne. Do wyzwalania przerzutnika służą dwa wejścia sterujące: wejście **A** — do wyzwalania ujemnymi oraz wejście **B** — do wyzwalania dodatnimi zboczami impulsów. Układ ma dodatkowe wejście $\overline{\mathbf{R}}$ (ang. *Reset* — zeruj) pozwalające wyzerować przerzutnik w dowolnej chwili. Schemat funkcjonalny, symbol graficzny przerzutnika oraz przebiegi czasowe przedstawiono na rys. 8.3.



Rys. 8.3. Przerzutnik monostabilny '123: a) schemat funkcjonalny; b) symbol graficzny; c), d) przebiegi czasowe
W — przebieg pomocniczy na wyjściu bramki AND $W = \overline{A} \cdot B \cdot R$

Układ logiczny doprowadzający sygnał do wejścia wyzwalającego przerzutnika (wejście oznaczone trójkątem niezamalowanym, co oznacza, że zboczem aktywnym jest zbocze dodatnie) realizuje funkcję **W** o postaci

$$\mathbf{W = \overline{A} \cdot B \cdot R}$$

Narastające zbocze sygnału **W** sprawia, że na wyjściu **Q** przerzutnika jest generowany impuls o czasie trwania *t* (rys. 8.3c). Z przebiegów czasowych przedstawionych na rys. 8.3c widać ponadto, że wystąpienie kolejnych impulsów wyzwalających w czasie trwania *t* impulsu generowanego wydłuża czas trwania im-

pulsu wyjściowego **Q**. Można powiedzieć, że każdy kolejny impuls wyzwalający sprawia, że odliczanie czasu t rozpoczyna się na nowo. O tak działającym przerzutniku mówimy, że jest **retrygerowalny**.

Użytkownik układu scalonego ma jednak do dyspozycji jedynie wejścia **A** i **B** oraz **R**. Analizując wyzwalanie przerzutnika od strony tych właśnie wejść, na podstawie przebiegów czasowych z rys. 8.3c można sformułować następujące wnioski:
— można wyzwolić przerzutnik ujemnym zboczem sygnału **A**, gdy na wejściu **B** jest stan **1**;
— można wyzwolić przerzutnik dodatnim zboczem sygnału **B**, gdy na wejściu **A** jest poziom niski **0**.

Oba powyżej wymienione sposoby wyzwalania zachodzą wówczas, gdy na wejściu **R** jest stan wysoki **1**. Wpływ wejścia **R** na zachowanie się przerzutnika przedstawiono na rys. 8.3d. Dla ułatwienia przyjęto, że **A = 0**. Doprowadzenie do wejścia zerującego **R** poziomu logicznego **0** podczas generowania impulsu powoduje natychmiastowe ustalenie się poziomu logicznego **0** na wyjściu **Q**, czyli skrócenie czasu trwania impulsu. Sygnał **R** może również służyć do wyzwalania układu **'123**. Jeżeli w czasie zmiany poziomu sygnału **R** z **0** na **1** są spełnione warunki: **A = 0** i **B = 1**, to zostanie wygenerowany impuls wyjściowy o pełnej szerokości t, o ile sygnał **R** nie zmieni się w tym czasie.

Porównując przerzutnik **'123** z przerzutnikiem **'121** można zauważyć, że podobieństwo dotyczy jedynie sposobu ich wyzwalania. Oba przerzutniki są wyzwalane ujemnym zboczem sygnału **A** (przy **B = 1**) lub dodatnim zboczem sygnału **B** (przy **A = 0**). Pozostałe cechy obu przerzutników są różne. Kolejne impulsy wyzwalające w czasie generowania impulsu wyjściowego wydłużają czas generowanego impulsu w przerzutniku **'123**, natomiast nie mają żadnego wpływu na impuls generowany przez przerzutnik **'121**. Ponadto przerzutnik **'123** różni się od przerzutnika **'121** tym, że:
— nie ma wejściowej bramki Schmitta;
— nie ma wewnętrznego rezystora;
— wartość dołączanej pojemności jest nieograniczona;
— wartość rezystancji musi się zawierać w granicach $5 \div 50\ \text{k}\Omega$;
— minimalny czas trwania impulsu wyjściowego wynosi 40 ns.

Na rysunku 8.4 przedstawiono sposoby dołączania elementów zewnętrznych. Podstawowy układ podano na rys. 8.4a. Czas trwania impulsu t_i przy wartości $C > 1000$ pF oblicza się ze wzoru

$$t_i = 0{,}32\, RC\left(1 + \frac{0{,}7}{R}\right)$$

(przy czym: t_i [ns], R [kΩ], C [pF]),
chociaż często wystarczy wzór przybliżony

$$t_i \approx 0{,}3RC.$$

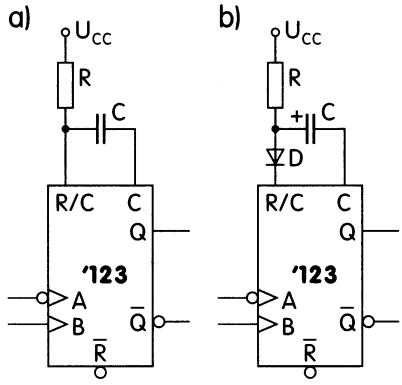
Jeżeli $C < 1000$ pF, to należy korzystać z nomogramów podanych w literaturze technicznej (np. [21], [6]).

W przypadku stosowania kondensatorów elektrolitycznych lub wykorzystywania wejść zerujących należy dodatkowo włączyć do układu diodę krzemową (rys. 8.4b). Czas trwania impulsu wyjściowego t_i należy wówczas wyznaczyć z zależności

$$t_i = 0{,}28\,RC\left(1+\frac{0{,}7}{R}\right)$$

(przy czym: t_i [ns], R [kΩ], C [pF]).

Także i w tym przypadku można korzystać z podanego powyżej przybliżonego wzoru na czas trwania generowanego impulsu wyjściowego.



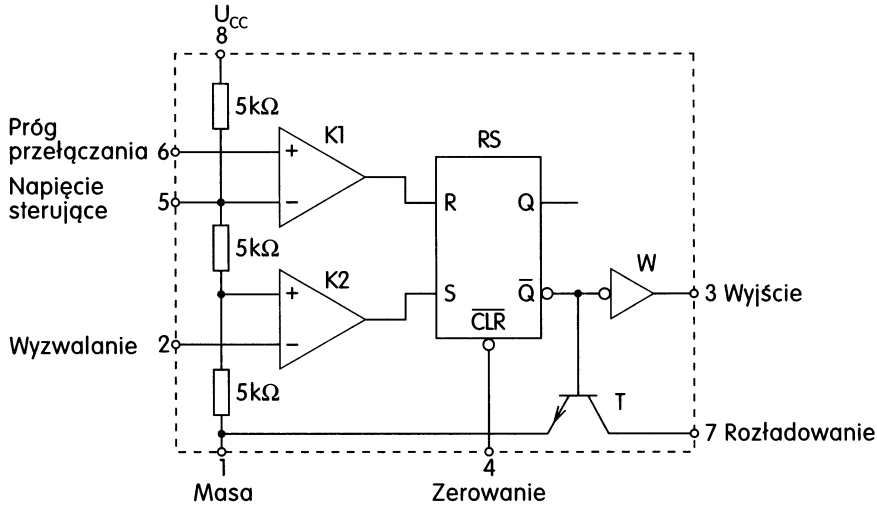
Rys. 8.4. Sposoby dołączenia elementów zewnętrznych *R*, *C*: a) kondensator nieelektrolityczny; b) kondensator elektrolityczny

8.2.3. Układ ULY7855 (555)

Układ **ULY7855** stanowi odpowiednik popularnego układu czasowego (tzw. tajmera) 555 produkowanego w krajach Europy Zachodniej. Jego schemat funkcjonalny przedstawiono na rys. 8.5. Zawiera on następujące elementy:

— dwa komparatory *K1* i *K2*,
— przerzutnik prosty (asynchroniczny) *RS* z dodatkowym wejściem zerującym $\overline{\mathbf{CLR}}$,
— wzmacniacz wyjściowy *W*,
— tranzystor rozładowujący *T*.

Przerzutnik *RS* ma dwa wejścia zerujące: **R** i $\overline{\mathbf{CLR}}$. Wejście **R** jest sterowane z wyjścia komparatora *K1*, natomiast wejście $\overline{\mathbf{CLR}}$ jest wyprowadzone na zewnątrz układu jako końcówka przeznaczona do doprowadzenia zewnętrznego sy-

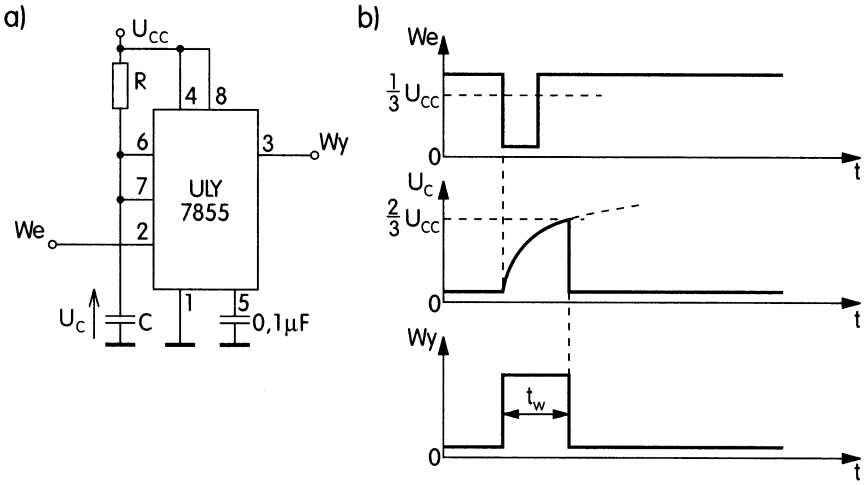


Rys. 8.5. Schemat funkcjonalny układu ULY7855

gnału zerującego. Symbol negacji oznacza (zgodnie ze stosowaną przez nas symboliką), że zerowanie nastąpi po ustawieniu poziomu logicznego **0**. Napięcie zasilające układ doprowadza się między końcówki 8 (U_{CC}) i 1 (masa). Jego wartość powinna się zawierać w przedziale 3 ÷ 18 V. W celu uzyskania poziomów logicznych TTL i umożliwienia współpracy tego układu z układami TTL, należy zasilać układ napięciem 5 V.

Producent przewidział dwa podstawowe układy pracy tajmera ULY7855: układ monostabilny i astabilny.

Na rysunku 8.6 pokazano układ generatora monostabilnego oraz przebiegi czasowe w charakterystycznych punktach układu.



Rys. 8.6. Generator monostabilny: a) schemat; b) przebiegi czasowe

Układ jak na rys. 8.6 działa w następujący sposób.

W stanie spoczynkowym (stabilnym) napięcie na wyjściu *Wy* układu ma poziom niski, czyli na wyjściu $\overline{\mathbf{Q}}$ (rys. 8.5) występuje poziom wysoki. Wysterowuje on tranzystor *T* w stan przewodzenia, co sprawia, że zbocznikowany tym tranzystorem kondensator *C* jest rozładowany i napięcie na nim jest bliskie 0. Jest to napięcie doprowadzone do wejścia nieodwracającego komparatora *K1*. Na drugim wejściu komparatora *K1* (odwracającym) napięcie ma wartość równą $2U_{CC}/3$ (ustaloną przez wewnętrzny dzielnik napięcia). W efekcie na wyjściu komparatora *K1* (i jednocześnie wejściu **R** przerzutnika) jest poziom niski.

Na wejściu wyzwalającym *We* (końcówka 2) napięcie powinno mieć wartość większą niż $U_{CC}/3$. Napięcie na wejściu nieodwracającym komparatora *K2*, ustalone przez wewnętrzny dzielnik napięcia, ma wartość równą $U_{CC}/3$. W efekcie na wyjściu komparatora *K2* (i jednocześnie wejściu **S** przerzutnika) jest poziom niski. Przerzutnik jest więc w stanie pamiętania sygnału logicznego **0**.

Doprowadzenie do wejścia wyzwalającego (końcówka 2) zbocza ujemnego powoduje, że napięcie na wejściu odwracającym komparatora *K2* jest niższe niż na wejściu nieodwracającym, pod warunkiem, że poziom sygnału wyzwalającego

zmalał poniżej $U_{CC}/3$.) Wyjście komparatora *K2* zostaje przestawione w stan wysoki, co sprawia, że przerzutnik *RS* zostaje ustawiony w stan **1** i na wyjściu układu pojawia się wysoki poziom napięcia. Na wyjściu $\overline{\mathbf{Q}}$ występuje wówczas poziom niski i dzięki temu tranzystor *T* zostaje odcięty. Kondensator *C* jest teraz ładowany przez rezystor *R*. Gdy napięcie na nim osiągnie wartość $U_{CC}/3$ (czyli wartość napięcia odniesienia na wejściu odwracającym komparatora *K1*) stan wyjścia komparatora *K1* ulegnie zmianie i wysoki poziom na jego wyjściu ustawi ponownie w stan niski przerzutnik *RS*, o ile na jego wejściu **S** pojawił się ponownie stan niski. ***Tak więc, działanie generatora monostabilnego będzie poprawne pod warunkiem, że czas trwania impulsu wyzwalającego będzie krótszy niż czas trwania impulsu generowanego.*** W przeciwnym razie — pomimo że wejście **R** przerzutnika zostanie wysterowane przez wyjście komparatora *K1* — przerzutnik nie zmieni stanu dopóty, dopóki wejście **S** nie znajdzie się w stanie wysokim.

W chwili powrotu stanu wyjścia $\overline{\mathbf{Q}}$ do poziomu niskiego **L** tranzystor *T* zacznie przewodzić i rozładuje szybko kondensator *C*. Układ powróci do stanu początkowego.

Czytelnikowi pozostawia się do przeanalizowania pytanie, czy kolejne impulsy wyzwalające (doprowadzane w czasie trwania generowanego impulsu) mają wpływ na czas impulsu wyjściowego?

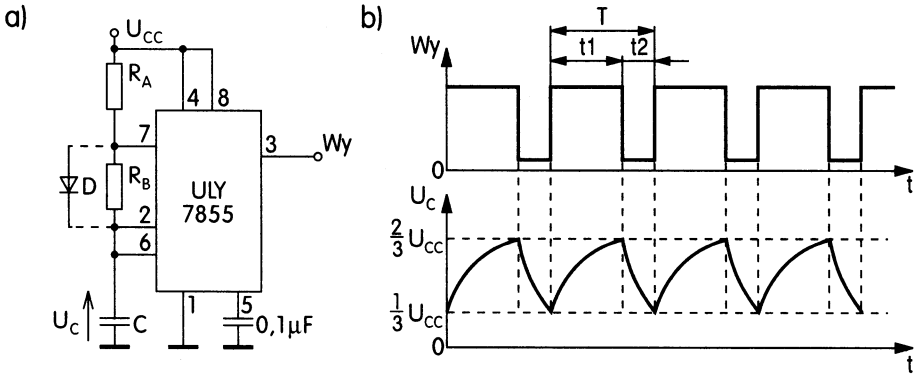
Szerokość impulsu wyjściowego można określić w przybliżeniu z zależności

$$t \approx 1{,}1\ RC.$$

Maksymalna wartość rezystancji *R* wynosi ok. 20 MΩ. Stosując zatem kondensatory elektrolityczne, można uzyskać czas trwania impulsu wyjściowego rzędu kilku godzin. Jest to nieosiągalne przy zastosowaniu układów **'121** i **'123**.

Kondensator 0,1 μF, przyłączony do końcówki 5, służy do tłumienia tętnień występujących na wejściu odwracającym komparatora *K1*.

Na rysunku 8.7 przedstawiono układ połączeń tajmera ULY7855 przy pracy astabilnej. W układzie tym kondensator *C* jest ładowany przez dwa rezystory R_A i R_B. Gdy napięcie na kondensatorze osiągnie wartość $2U_{CC}/3$, wówczas rozpo-



Rys. 8.7. Generator astabilny: a) schemat; b) przebiegi czasowe

czyna się proces rozładowania przez rezystor R_B oraz tranzystor T. Ponieważ końcówka 2 jest połączona z kondensatorem, więc rozładowanie trwa dopóty, dopóki napięcie na kondensatorze nie będzie mniejsze niż $U_{CC}/3$. Wówczas ponownie rozpoczyna się proces ładowania. Kondensator jest ładowany ze stałą czasową $(R_A + R_B)C$, a rozładowywany ze stałą czasową R_BC. Czas trwania ładowania t_1 i rozładowania t_2 można wyznaczyć z przybliżonych zależności:

$$t_1 \approx 0{,}7(R_A + R_B)C; \quad t_2 \approx 0{,}7R_BC.$$

Zatem wartość **okresu generowanego przebiegu**

$$T = t_1 + t_2 \approx 0{,}7(R_A + 2R_B)C.$$

Z podanych zależności wynika, że zawsze $t_1 > t_2$ i współczynnik wypełnienia przebiegu wyjściowego będzie większy niż 50%. Aby mieć możliwość uzyskania mniejszych współczynników wypełnienia, należy dołączyć diodę D tak, jak to pokazano na rys. 8.7 linią przerywaną. W takim przypadku ładowanie kondensatora odbywa się wyłącznie przez rezystor R_A oraz diodę D, a rozładowanie przez rezystor R_B. Można więc, niezależnie i dowolnie dobierać czasy t_1 i t_2, a tym samym i współczynnik wypełnienia.

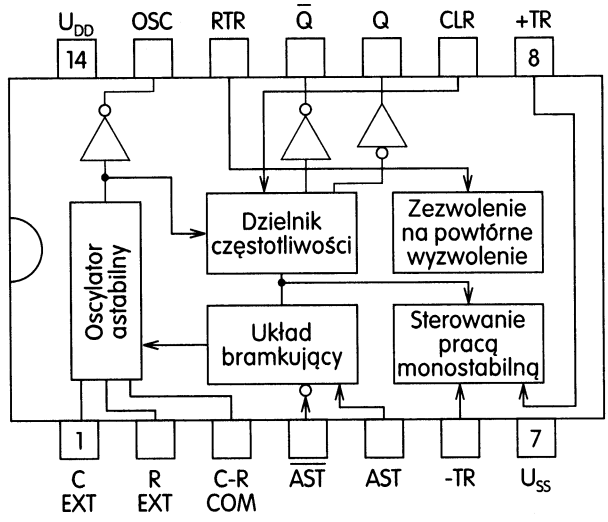
Obciążalność prądowa wyjścia układu 555 jest bardzo duża (w porównaniu z układami TTL czy CMOS) i wynosi ±200 mA (± oznacza, że zarówno w stanie wysokim, jak i niskim prąd wyjściowy może mieć wartość 200 mA). Ponieważ kierunki prądów są różne, przeto i znak przy wartości prądu raz będzie „+" (w stanie **L**), raz „–" (w stanie **H**).

Możliwości zastosowań tajmera **ULY7855** są bardzo duże. W literaturze można znaleźć wiele praktycznych układów, w których wykorzystano ten układ. Mogą to być np. dzielniki częstotliwości, przetworniki *U/f*, autoalarmy, itp. Znaczenie tego układu dla techniki cyfrowej wynika chociażby z możliwości uzyskania bardzo długich czasów trwania generowanych impulsów (nieosiągalnych w układach **'121** i **'123**). Ponadto czas trwania generowanych impulsów jest bardzo stabilny — niezależny od temperatury i napięcia zasilania. Użycie stabilnych elementów zewnętrznych zapewnia uzyskanie prawie wzorcowych impulsów czasowych.

8.2.4. Monostabilny/astabilny multiwibrator '047 (MCY74047)

Schemat funkcjonalny układu scalonego CMOS **'047** przedstawiono na rys. 8.8. Układ ma sześć wejść. Dwa z nich **AST** i $\overline{\textbf{AST}}$ służą do bramkowania pracy oscylatora astabilnego. Dwa kolejne: **+TR** i **–TR** są przeznaczone do wyzwalania przerzutnika odpowiednio zboczem dodatnim lub ujemnym. Wejście **RTR** pozwala ustawić przerzutnik w tryb pracy przerzutnika **'123**, tzn. w tryb wydłużania impulsu wyjściowego (**przerzutnik retrygerowalny**). Wejście **CLR** jest wejściem zerującym układ.

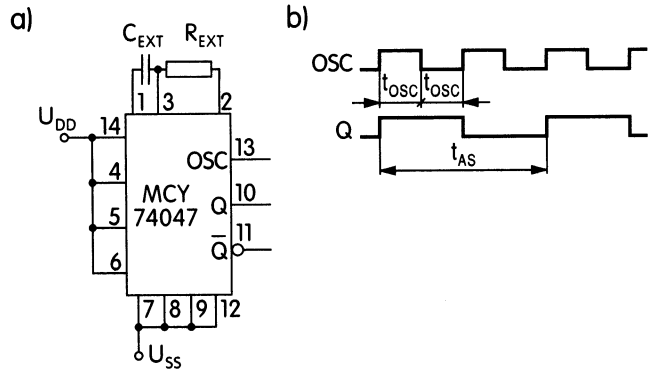
Układ ma wyjście komplementarne **Q** i $\overline{\textbf{Q}}$ oraz wyprowadzony sygnał wewnętrznego generatora — wyjście **OSC**.



Rys. 8.8. Schemat funkcjonalny układu scalonego MCY74047

Końcówki układu opisane: C_{EXT}, R_{EXT} i C–R_{COM} są przeznaczone do podłączenia zewnętrznych elementów R, C. Układ połączeń przy **pracy astabilnej** przedstawiono na rys. 8.9. Wartości rezystancji i pojemności dobiera się zgodnie z następującymi zależnościami:

$$t_{OSC} = 1{,}1\, R_{EXT} C_{EXT}; \qquad t_{AS} = 4{,}4\, R_{EXT} C_{EXT}.$$



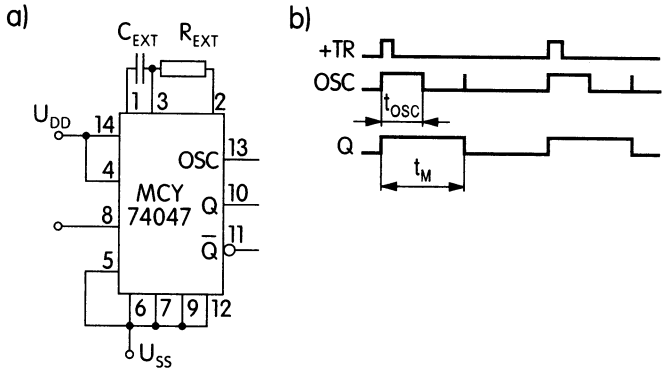
Rys. 8.9. Multiwibrator astabilny: a) układ połączeń; b) przebiegi czasowe

Układ połączeń przy **pracy monostabilnej** przedstawiono na rys. 8.10.

Wartości rezystancji i pojemności dobiera się zgodnie z następującymi zależnościami:

$$t_{OSC} = 1{,}38\, R_{EXT} C_{EXT}; \qquad t_{M} = 2{,}48\, R_{EXT} C_{EXT}.$$

W układzie jak na rys. 8.10 przerzutnik jest wyzwalany zboczem dodatnim (sygnał wyzwalający jest doprowadzony do wejścia **+TR** (8), a do wejścia **–TR** (7) poziom niski). Jeżeli chcemy wyzwolić układ zboczem opadającym, to należy go



Rys. 8.10. Multiwibrator monostabilny: a) układ połączeń; b) przebiegi czasowe

podać na wejście **−TR** (7), a wejście **+TR** (8) powinno być w stanie wysokim. W celu wydłużenia impulsu wyjściowego (praca w trybie przerzutnika **'123 — przerzutnik retrygerowalny**) należy wejście **+TR** (8) połączyć z wejściem **RTR** (12).

W obu rodzajach pracy (praca astabilna, praca monostabilna) jest wymagane dołączenie elementów zewnętrznych *R* i *C*. Właściwie nie ma ograniczeń co do granicznych wartości parametrów tych elementów. Jednak zaleca się stosowanie kondensatorów nieelektrolitycznych o rezystancji rezystora upływowego (równoległego) 10 razy większej niż rezystancja współpracującego z nim rezystora. Właściwą pracę układu producent gwarantuje przy stosowaniu elementów zewnętrznych o następujących wartościach:

$C > 100$ pF do pracy w trybie astabilnym,

$C > 1000$ pF do pracy w trybie monostabilnym,

$10\ \text{k}\Omega < R < 1\ \text{M}\Omega$ do pracy w obu trybach.

Generowane w tym układzie impulsy mają następujące parametry dynamiczne:

— czas narastania $t_{TLH} = 100$ ns (przy $U_z = 5$ V),

— czas opadania $t_{THL} = 100$ ns (przy $U_z = 5$ V).

Minimalny czas trwania impulsu wejściowego (wyzwalającego) wynosi $t_w = 500$ ns (przy $U_z = 5$ V).

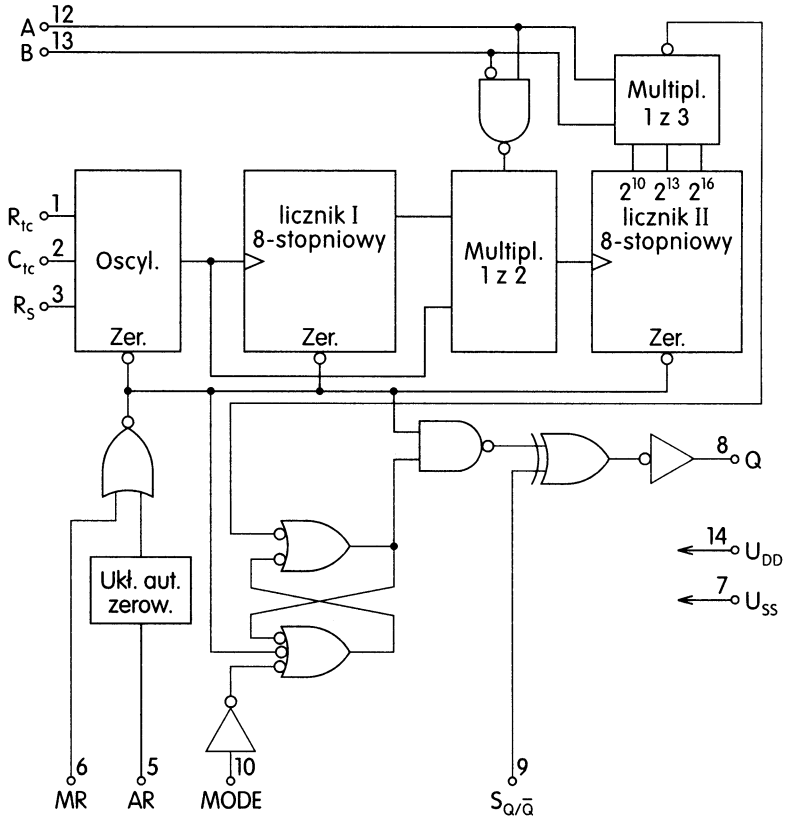
8.2.5. Programowany układ czasowy CMOS '541 (MCY74541)

Programowany układ czasowy '541 (rys. 8.11) zawiera wewnętrzny generator (oscylator), dwa liczniki 8-bitowe (mod 256), dwa multipleksery oraz układy sterujące. Częstotliwość wewnętrznego oscylatora jest określona przez wartości zewnętrznych elementów *RC* i może się zawierać w przedziale $0 \div 500$ kHz. Sygnał wyjściowy z oscylatora może być pobrany z wyprowadzenia *1* (R_{tc}) lub *2* (C_{tc}). Są one względem siebie w przeciwfazie. Obciążenie tych wyjść może jednak spowodować zmianę częstotliwości generowanej fali prostokątnej.

W zakresie $1\ \text{kHz} \leq f \leq 100\ \text{kHz}$ częstotliwość jest określona wzorem

$$f = \frac{1}{2{,}3\, C_{tc} R_{tc}}$$

przy czym: $R_s \approx 2R_{tc}$, $R_s \geq 10\ \text{k}\Omega$.



Rys. 8.11. Programowalny układ czasowy MCY74541 — schemat funkcjonalny

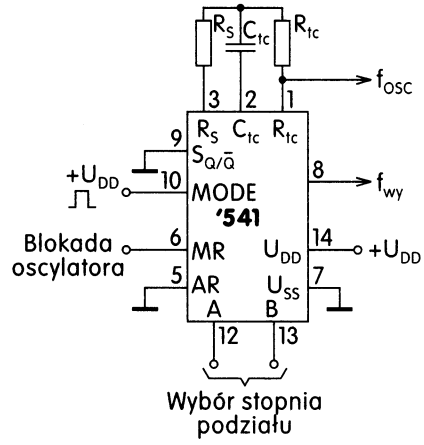
Częstotliwość oscylatora może być stabilizowana za pomocą rezonatora kwarcowego.

Układ **'541** może być sterowany także zewnętrznym sygnałem prostokątnym, który należy doprowadzić do końcówki *3* (R_S). Zbędne są wówczas elementy C_{tc}, R_{tc} i R_S, bo jest niewykorzystywany wewnętrzny oscylator.

Układ '541 może generować pojedyncze impulsy (praca monostabilna) lub falę prostokątną (praca astabilna).

Schemat układu **'541** wykorzystywanego jako **układ astabilny (generator)** przedstawiono na rys. 8.12.

Sygnał wyjściowy wewnętrznego oscylatora jest podawany na wejście licznika **mod 256**. Zmianę stanu licznika powoduje do-



Rys. 8.12. Schemat oscylatora

datnie zbocze impulsu wejściowego. Częstotliwość przebiegu generowanego w oscylatorze jest dzielona poprzez oba liczniki binarne zgodnie z tabl. 8.1.

Sygnały **A**, **B** doprowadzamy z zewnątrz do wyprowadzeń odpowiednio *12* (**A**) i *13* (**B**). Ponadto do dyspozycji mamy jeszcze cztery wejścia sterujące, określające rodzaj pracy układu.

Tablica 8.1

A	B	Stopień podziału	
0	0	8192	8 stopni 1. licznika + 5 stopni 2.
0	1	1024	8 stopni 1. licznika + 2 stopnie 2.
1	0	256	1. licznik jest pomijany.
1	1	65536	Pełny podział przez oba liczniki.

Wejście:

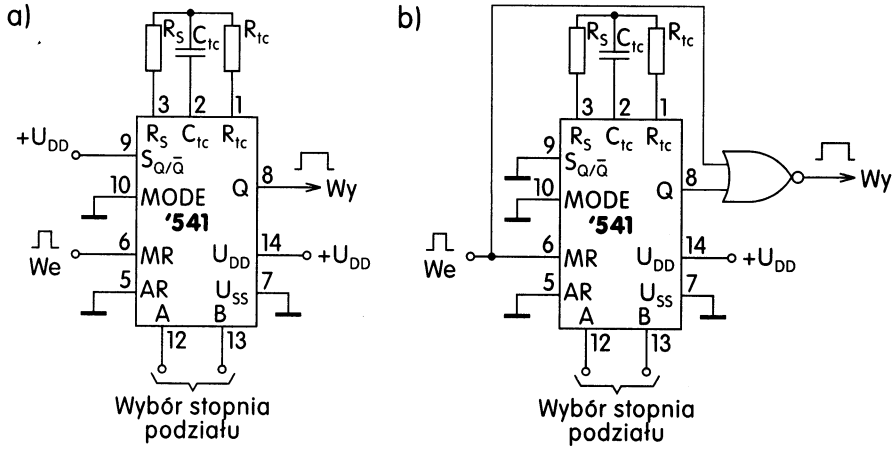
AR (*5*): **0** — automatyczne zerowanie układu w chwili włączenia zasilania (napięcie zasilania musi mieć wartość z przedziału 7,5 ÷ 18 V);
1 — brak automatycznego zerowania (zmniejszenie poboru mocy przez układ);

MR (*6*): **0** — generator i liczniki pracują;
1 — generator, liczniki i przerzutnik (asynchroniczny, zbudowany z bramek NAND) są wyzerowane;

MODE (*10*): **0** — jest generowany pojedynczy impuls (praca monostabilna);
1 — praca astabilna;

$S_{Q/\bar{Q}}$ (9): **0** — po wyzerowaniu układu na wyjściu **Q** (*8*) jest stan **0**;
1 — po wyzerowaniu układu na wyjściu **Q** (*8*) jest stan **1**.

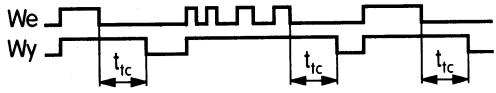
Podstawowe **układy pracy monostabilnej** przedstawiono na rys. 8.13. Czas generowanego w tych układach impulsu zależy nie tylko od elementów zewnętrz-



Rys. 8.13. Generatory monostabilne zbudowane z układu MCY74541

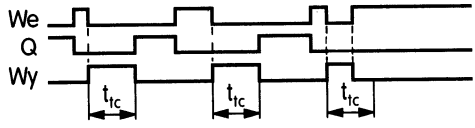
nych R, C, ale także od stopnia podziału (n), określonego za pośrednictwem wejść **A** i **B** zgodnie z tabl. 8.1.

W rezultacie są to więc przerzutniki monostabilne o programowanym czasie trwania impulsu wyjściowego. W układzie z rys. 8.13a impuls wyjściowy jest dodatkowo wydłużony o czas trwania impulsu wejściowego doprowadzonego do wejścia **MR** (*6*). Pojawienie się kolejnego impulsu wejściowego (przed zanikiem sygnału na wyjściu) sprawi, że na wyjściu **Q** będzie się utrzymywał stan **1** (cecha charakterystyczna przerzutnika **'123**). Właściwości te obrazują przebiegi czasowe przedstawione na rys. 8.14.



Rys. 8.14. Przebiegi czasowe w układzie z rys. 8.13a

Układ jak na rys. 8.13b generuje impuls wyjściowy wyzwalany ujemnym zboczem sygnału wejściowego. Czas jego trwania może zostać skrócony wówczas, gdy wcześniej zostanie ustawiony stan wysoki na wejściu wyzwalającym, co uwidoczniono na rys. 8.15.

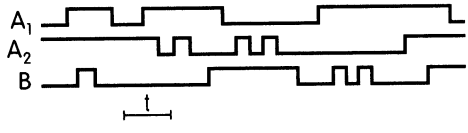


Rys. 8.15. Przebiegi czasowe w układzie z rys. 8.13b

Układ **'541** ma wiele możliwości zastosowań i dwa układy przedstawione na rys. 8.13 nie wyczerpują wszystkich jego zalet. W bardzo prosty sposób można uzyskać programowany generator fali prostokątnej, programowany dzielnik częstotliwości i wiele innych. Niektóre z nich będą jeszcze przedstawione w dalszej części podręcznika.

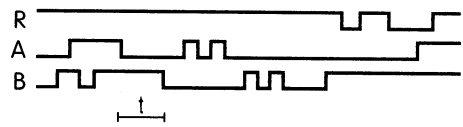
Pytania i zadania

1. Co to jest przerzutnik? Wyjaśnij takie pojęcia, jak: przerzutnik monostabilny, przerzutnik bistabilny, przerzutnik astabilny.
2. Narysuj przebiegi czasowe na wyjściu przerzutnika **'121** dla danych (rys. 8.16) przebiegów wejściowych.



Rys. 8.16. Przebiegi czasowe do zadania 2.

3. Narysuj przebiegi czasowe na wyjściu przerzutnika **'123** dla danych (rys. 8.17) przebiegów wejściowych.

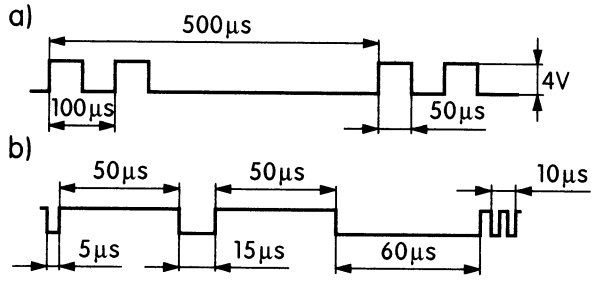


Rys. 8.17. Przebiegi czasowe do zadania 3.

4. Narysuj schemat zasadniczy oraz dobierz elementy *R*, *C* układu umożliwiającego uzyskanie na wyjściu impulsu o czasie trwania równym 1 s. Do budowy generatora wykorzystaj układ:
   a) **'121**, wyzwalanie zboczem dodatnim,
   b) **'123**, wyzwalanie zboczem ujemnym,
   c) 555 (ULY7855),
   d) MCY74047, wyzwalanie zboczem ujemnym,
   e) MCY74541, wyzwalanie zboczem ujemnym.

   Narysuj przykładowe przebiegi czasowe w tym układzie.

5. Narysuj i zwymiaruj przebiegi czasowe przerzutnika **'121**. Dane: C = 10 nF = const, a) R = 10 kΩ, b) R = 20 kΩ, c) R = 47 kΩ, **A1** = **A2** = **0**, do wejścia **B** doprowadzono przebieg jak na rys. 8.18a.



Rys. 8.18. Przebiegi czasowe do zadania 5.

6. Wykonaj zadanie 5. gdy **A1** = **B** = **1**, a do **A2** doprowadzono przebieg jak na rys. 8.18.
7. Wykonaj zadanie 5. dla przerzutnika **'123**.
8. Wykonaj zadanie 6. dla przerzutnika **'123**.
9. Wykonaj zadanie 5. dla układu 555 i przebiegu z rys. 8.18b.
10. Wykonaj zadanie 5. dla przebiegu wejściowego w postaci fali prostokątnej o częstotliwości f = 10 kHz i współczynniku wypełnienia 1/2.
11. Wykonaj zadanie 6. dla przebiegu wejściowego w postaci fali prostokątnej o częstotliwości f = 10 kHz i współczynniku wypełnienia 1/2.

12. Wykonaj zadanie 7. dla przebiegu wejściowego w postaci fali prostokątnej o częstotliwości $f = 10$ kHz i współczynniku wypełnienia 1/2.
13. Wykonaj zadanie 8. dla przebiegu wejściowego w postaci fali prostokątnej o częstotliwości $f = 10$ kHz i współczynniku wypełnienia 1/2.

8.3. Układy uzależnień czasowych

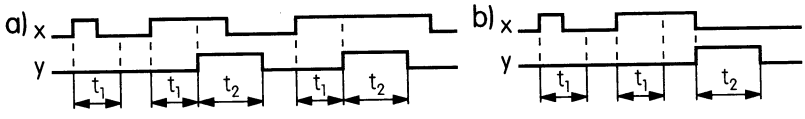
Wiele układów tego rodzaju (typowych dla najczęściej budowanych systemów cyfrowych) można znaleźć np. w publikacji [21]. Z tego też powodu w niniejszym podrozdziale przedstawiono jedynie pewien sposób postępowania, umożliwiający zaprojektowanie układu uzależnień czasowych na podstawie pożądanych przebiegów wyjściowych lub słownego opisu działania układu.

Przykład 8.1

Zbudować układ czasowy generujący na wyjściu impuls o czasie trwania t_2, jeżeli na wejściu **x** pojawi się impuls dłuższy niż t_1. Kolejne impulsy wejściowe pojawiają się nie wcześniej niż po czasie t_2 liczonym od zaniku impulsu wejściowego. Zanim przystąpimy do rozwiązania, zwróćmy uwagę na praktyczny sens takiego układu. Ma to być bowiem układ wejściowy, który będzie tłumił impulsy krótsze niż t_1 (np. **zakłócające**), a na każdy **użyteczny** impuls wejściowy będzie generował standardowy impuls o czasie trwania t_2.

Rozwiązanie

Pierwszym etapem projektowania układu uzależnień czasowych jest narysowanie przebiegów czasowych sygnałów wejściowych i wyjściowych, obrazujących wszystkie możliwe sytuacje. W tym przypadku należy więc uwzględnić odpowiedź układu na impuls wejściowy: krótszy niż t_1, dłuższy niż t_1, ale krótszy niż $t_1 + t_2$ i dłuższy niż $t_1 + t_2$ (rys. 8.19a).

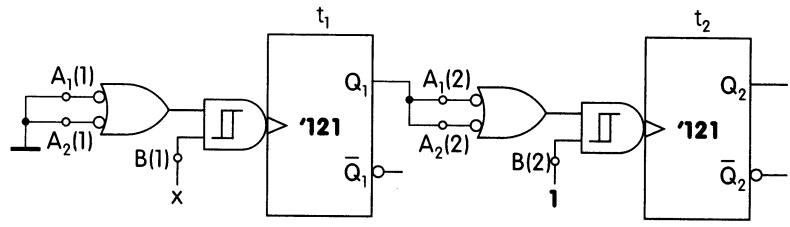


Rys. 8.19. Przebiegi czasowe do przykładu 8.1

Z treści zadania nie wynika, czy impuls wyjściowy ma się pojawić już w chwili, gdy wiadomo jest, że impuls wejściowy jest dłuższy niż t_1, czy dopiero po zaniku impulsu wejściowego. Z tego powodu są możliwe dwa warianty przebiegów czasowych spełniających warunki zadania. Przyjmijmy do dalszej analizy przebiegi z rys. 8.19a, pozostawiając Czytelnikowi do samodzielnego rozwiązania wersję z rys. 8.19b.

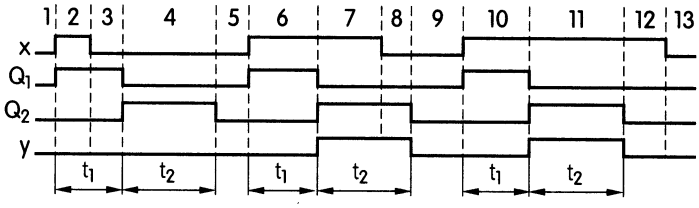
Dalsze postępowanie wymaga przyjęcia odpowiednich układów przerzutników monostabilnych, generujących impulsy o czasie trwania t_1 i t_2 w odpo-

wiednich chwilach. Z rysunku 8.19a widać, że w chwili pojawienia się impulsu wejściowego **x** należy rozpocząć generowanie impulsu o czasie trwania t_1, aby móc stwierdzić, czy impuls wejściowy jest dłuższy, czy krótszy niż t_1. Można to uzyskać, wyzwalając np. pierwszy przerzutnik **'121** z wejścia **B**, gdy na wejściach **A** jest poziom logiczny **0** (rys. 8.20). Po czasie t_1 powinno się rozpocząć generowanie impulsu t_2. Można to uzyskać, wyzwalając kolejny przerzutnik **'121** ujemnym zboczem impulsu t_1.



Rys. 8.20. Generatory impulsów o czasie trwania t_1 i t_2 do przykładu 8.1

Zakładając, że zbudowaliśmy już układ jak na rys. 8.20, możemy uzupełnić przebiegi czasowe z rys. 8.19a o sygnały uzyskane na wyjściach przerzutników **'121**. Aktualne przebiegi czasowe przedstawiono na rys. 8.21. Dzięki temu zadanie nasze możemy sformułować następująco: Zbudować układ, który przetworzy sygnały wejściowe **x**, $\mathbf{Q}_1$, $\mathbf{Q}_2$ w sygnał wyjściowy **y** zgodnie z opisem w postaci przebiegów czasowych, przedstawionym na rys. 8.21.



Rys. 8.21. Przebiegi czasowe do przykładu 8.1

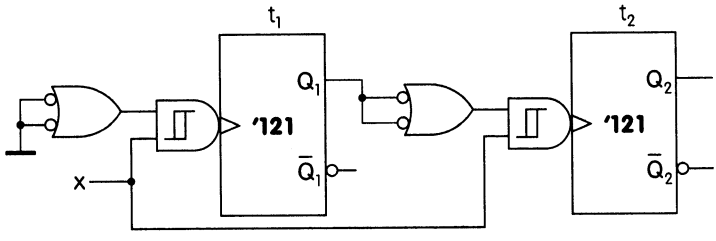
Poszukujemy więc układu kombinacyjnego, którego sygnałami wejściowymi są sygnały: **x**, $\mathbf{Q}_1$ i $\mathbf{Q}_2$, a wyjściowym **y**. W tym celu dzielimy przebiegi czasowe na odcinki czasowe, w których żaden z sygnałów wejściowych nie zmienia się (rys. 8.21). Konstruujemy tablicę prawdy (tabl. 8.2) na podstawie wykresów czasowych.

W drugim wierszu tablicy otrzymaliśmy różne stany wyjść. Dotyczy to przedziałów czasowych o numerach *4* i *8*. ***Układ cyfrowy, w którym stan wyjść jest***

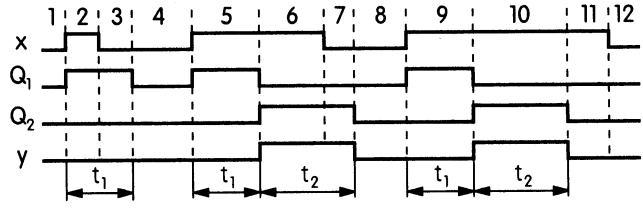
Tablica 8.2. **Tablica prawdy układu opisanego przebiegami czasowymi na rys. 8.21**

x	Q_1	Q_2	y
0	0	0	0
0	0	1	0/1
0	1	0	0
0	1	1	—
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	—

różny mimo takiego samego stanu wejść, jest układem sekwencyjnym. Jego realizacja wymaga użycia pamięci. Ale zanim przystąpimy do projektowania układu sekwencyjnego, należy rozważyć możliwość usunięcia sprzeczności w tablicy prawdy. Można to niekiedy osiągnąć poprzez zmianę sposobu wyzwalania przerzutników. Zauważmy, że zbędne jest generowanie impulsu t_2 wówczas, gdy impuls wejściowy jest krótszy niż t_1 (przedział czasowy *4*). Gdyby zmienić sposób wyzwalania przerzutnika Q_2 tak, aby generował on impuls wyjściowy tylko w przypadku, gdy impuls wejściowy jest dłuższy niż t_1, wówczas zostałaby usunięta sprzeczność uniemożliwiająca rozwiązanie zadania w klasie układów kombinacyjnych. Z przebiegów czasowych na rys. 8.21 wynika, że impuls t_2 powinien być wygenerowany, jeżeli ujemne zbocze impulsu $\mathbf{Q_1}$ pojawi się w czasie, gdy sygnał wejściowy **x** ma poziom **1**. Wystarczy zatem sygnał **x** doprowadzić do wejścia **B** przerzutnika Q_2. Zmodyfikowany układ przedstawiono na rys. 8.22, a przebiegi czasowe w tym układzie — na rys. 8.23.



Rys. 8.22. Zmodyfikowany układ z rys. 8.20



Rys. 8.23. Przebiegi czasowe w układzie z rys. 8.22

Tablica 8.3. **Tablica prawdy układu opisanego przebiegami czasowymi na rys. 8.23**

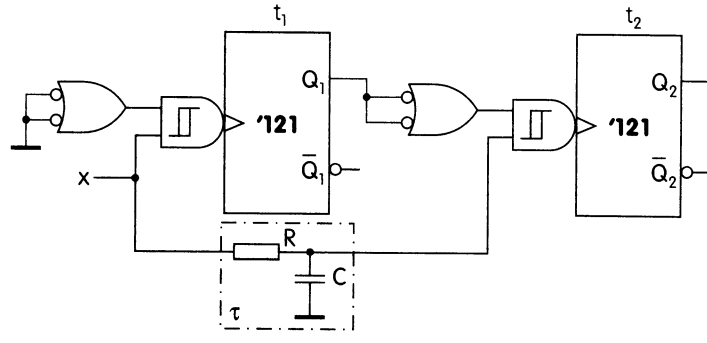
x	Q_1	Q_2	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	—
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	—

xQ_1 \ Q_2	0	1
00	0	1
01	0	—
11	0	—
10	0	1

$y = Q_2$

Tablica prawdy (tabl. 8.3), uzyskana na podstawie przebiegów czasowych z rys. 8.23, nie zawiera już sprzeczności.

Po przekształceniu jej w tablicę Karnaugha i minimalizacji otrzymujemy wyrażenie opisujące sygnał wyjściowy $\mathbf{y} = \mathbf{Q}_2$. W zasadzie zapisywanie tablicy prawdy i minimalizacja były zbędne, ponieważ z rys. 8.23 widać, że przebiegi czasowe **y** i $\mathbf{Q}_2$ są identyczne. Czy wobec tego układ z rys. 8.22 stanowi rozwiązanie zadania z przykładu 8.1? Aby odpowiedzieć na to pytanie, należy jeszcze przeanalizować drugi możliwy sposób wyzwalania przerzutnika **'121**. W projekcie przyjęliśmy, że jest on wyzwalany ujemnym zboczem sygnału **A** w obecności poziomu **1** na wejściu **B**. Oba wejścia wyzwalające są wobec tego podłączone i należy sprawdzić, czy nie zachodzi sytuacja, że pojawia się dodatnie zbocze na wejściu **B**, gdy na wejściu **A** jest poziom niski **0**. Otóż w stanie spoczynku (oba przerzutniki w stanie niskim) pojawienie się narastającego zbocza sygnału **x** wzbudza oba przerzutniki jednocześnie. Wprawdzie przerzutnik Q_1 jest ustawiany w stan wysoki (co powinno zablokować możliwość wyzwolenia przerzutnika Q_2 z wejścia **B**), ale dzieje się to z pewnym opóźnieniem (wynikającym z czasu propagacji przerzutników) i w rzeczywistym układzie przerzutnik Q_2 zostanie jednak wyzwolony. Aby się zabezpieczyć przed taką ewentualnością, należy w tor sygnału **x** wprowadzić układ opóźniający τ, jak np. na rys. 8.24.



Rys. 8.24. Rozwiązanie zadania z przykładu 8.1

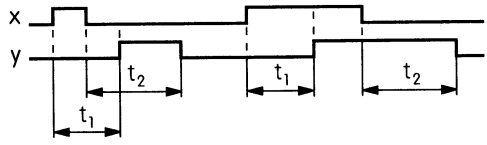
Należy jednak pamiętać, że ostatecznym środkiem weryfikacji poprawności zaprojektowanego układu czasowego powinno być zawsze zbudowanie jego prototypu i wszechstronne przetestowanie. ■

Przykład 8.2

Zaprojektować układ czasowy opóźniający zbocze dodatnie impulsu wejściowego **x** o czas t_1, a zbocze ujemne o czas t_2 (przy czym $t_1 < t_2$).

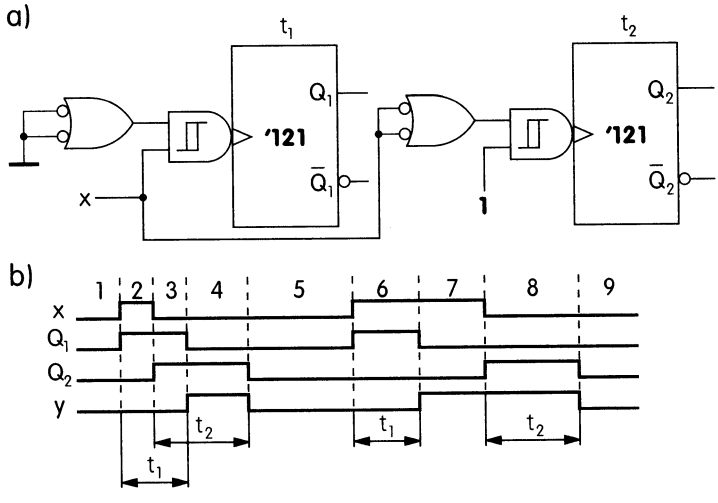
Rozwiązanie

Przebiegi czasowe, obrazujące wszystkie możliwe sytuacje, przedstawiono na rys. 8.25. (Zakładamy, że kolejne impulsy wejściowe pojawiają się nie wcześniej niż po zaniku sygnału wyjściowego **y**).



Rys. 8.25. Przebiegi czasowe do przykładu 8.2

Źródłem impulsu o czasie trwania t_1 będzie przerzutnik **'121** (Q_1) wyzwalany sygnałem **x** z wejścia **B**, a źródłem sygnału o czasie trwania t_2 będzie przerzutnik **'121** (Q_2) wyzwalany sygnałem **x** z wejścia **A** (rys. 8.26a). Przebiegi czasowe z rys. 8.25 uzupełnione o sygnały wyjściowe $\mathbf{Q}_1$, $\mathbf{Q}_2$ przedstawiono na rys. 8.26b. Na tej podstawie sporządzono tablicę prawdy oraz tablicę Karnaugha (rys. 8.27).



Rys. 8.26. Generatory impulsów o czasie trwania t_1 i t_2 (a) oraz przebiegi czasowe (b) do przykładu 8.2

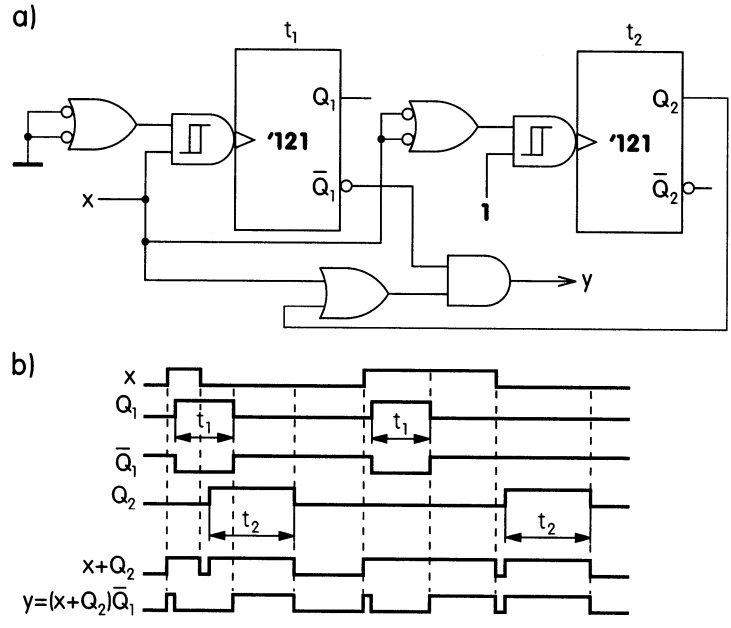
x	Q_1	Q_2	y
0	0	0	0
0	0	1	1
0	1	0	–
0	1	1	0
1	0	0	1
1	0	1	–
1	1	0	0
1	1	1	–

xQ_1 \ Q_2	0	1
00	0	1
01	–	0
11	0	–
10	1	1

$y = x\bar{Q}_1 + \bar{Q}_1 Q_2 = \bar{Q}_1(x + Q_2)$

Rys. 8.27. Tablica prawdy oraz tablica Karnaugha do przykładu 8.2

Rozwiązaniem zadania z przykładu 8.2 jest układ jak na rys. 8.28a. Rzeczywiste przebiegi czasowe w projektownym układzie, tzn. z uwzględnieniem czasu propagacji przerzutników, przedstawiono na rys. 8.28b. Zauważmy, że przebieg wyjściowy różni się w sposób zasadniczy od żądanego. Jeżeli sygnał wyjściowy tego układu steruje jakimś wolnym lub pozbawionym pamięci elementem (przekaźnik, dioda LED), to dodatkowe impulsy (hazard dynamiczny), które pojawiły się z powodu opóźnień wprowadzanych przez elementy, nie będą miały ne-



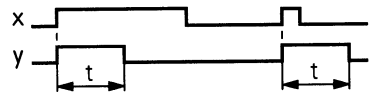
Rys. 8.28. Rozwiązanie zadania z przykładu 8.2: a) schemat logiczny; b) rzeczywiste przebiegi czasowe

gatywnego wpływu. W pozostałych przypadkach impulsy takie mogą uniemożliwić właściwą pracę układu sterowanego z wyjścia tego układu. Trzeba wówczas rozważyć inne wersje rozwiązań, jak np. dodatkowy filtr krótkich impulsów lub dodatkowy przerzutnik asynchroniczny itp. ■

Należy więc jeszcze raz podkreślić, że projektowanie układów uzależnień czasowych zalicza się do trudnych problemów technicznych, a przygotowany projekt należy zawsze zweryfikować poprzez badania układu prototypowego.

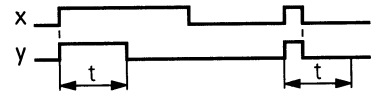
Pytania i zadania

1. Zaprojektuj układ o działaniu jak na rys. 8.29.



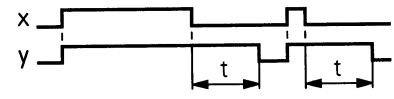
Rys. 8.29. Przebiegi czasowe do zadania 1.

2. Zaprojektuj układ o działaniu jak na rys. 8.30.



Rys. 8.30. Przebiegi czasowe do zadania 2.

3. Zaprojektuj układ o działaniu jak na rys. 8.31.

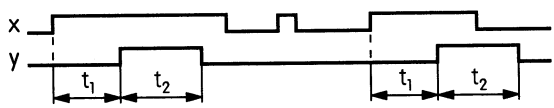


Rys. 8.31. Przebiegi czasowe do zadania 3.

4. Zaprojektuj układ o działaniu jak na rys. 8.32.

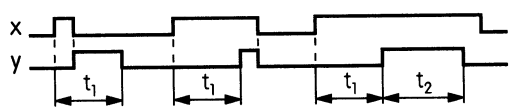
Rys. 8.32. Przebiegi czasowe do zadania 4.

5. Zaprojektuj układ o działaniu jak na rys. 8.33.



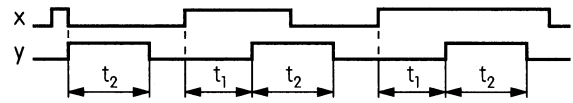
Rys. 8.33. Przebiegi czasowe do zadania 5.

6. Zaprojektuj układ o działaniu jak na rys. 8.34.



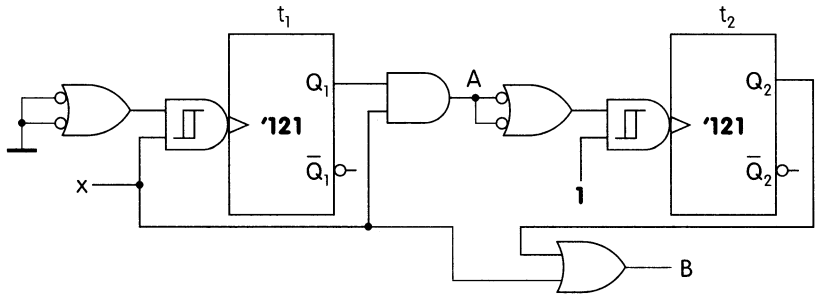
Rys. 8.34. Przebiegi czasowe do zadania 6.

7. Zaprojektuj układ o działaniu jak na rys. 8.35.



Rys. 8.35. Przebiegi czasowe do zadania 7.

8. Narysuj przebiegi czasowe w punktach *A* i *B* układu przedstawionego na rys. 8.36.



Rys. 8.36. Schemat układu do zadania 8.

Uwaga. Rysuj na papierze kratkowanym, przyjmując następujące długości impulsów: t_1 = 2 kr., t_2 = 3 kr. (gdzie kr. — kratka). Uwzględnij impulsy wejściowe **x** o długościach 1 kr., 2,5 kr., 6 kr..

9. Rozwiąż zadanie 8., zastępując wyjściową bramkę OR bramką AND.

10. Rozwiąż zadanie 8., doprowadzając do wejścia bramki AND sygnał $\overline{\mathbf{Q}}_1$ zamiast sygnału $\mathbf{Q}_1$.

11. Dane są dwa przerzutniki **'123** — Q_1 i Q_2. Impuls generowany przez przerzutnik Q_1 ma długość t_1 = 2 kr. (gdzie kr. — kratka na papierze kratkowanym), a przez przerzutnik Q_2 — t_2 = 4 kr. Impuls wejściowy może mieć długość 1 kr., 3 kr. i 7 kr. Narysuj układ oraz przebiegi czasowe w układzie, jeżeli:

    a) $\mathbf{A_1 = x, B_1 = 1, A_2 = 0, B_2 = x, R_1 = R_2 = 1, y = Q_1Q_2}$.
    b) $\mathbf{A_1 = x, B_1 = 1, A_2 = 0, B_2 = x, R_1 = R_2 = 1, y = Q_1 + Q_2}$.
    c) $\mathbf{A_1 = x, B_1 = 1, A_2 = 0, B_2 = x, R_1 = R_2 = 1, y = Q_1 + xQ_2}$.
    d) $\mathbf{A_1 = 0, B_1 = x, A_2 = Q_1, B_2 = 1, R_1 = R_2 = 1, y = xQ_2}$.
    e) $\mathbf{A_1 = 0, B_1 = x, A_2 = Q_1, B_2 = 1, R_1 = R_2 = 1, y = x\overline{Q}_1Q_2}$.
    f) $\mathbf{A_1 = 0, B_1 = x, A_2 = Q_1, B_2 = 1, R_1 = R_2 = 1, y = x\overline{Q}_1 + Q_2}$.

12. Dlaczego projektowanie układów uzależnień czasowych zalicza się do trudnych problemów w technice cyfrowej?

8.4. Układy wyzwalające

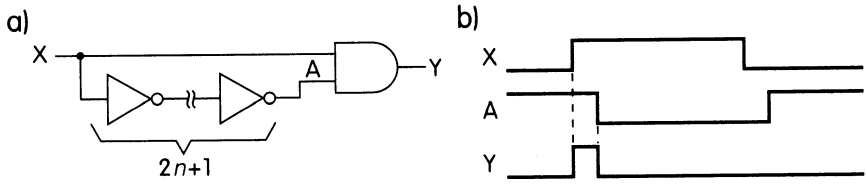
W układach cyfrowych krótkie (rzędu nanosekund) impulsy są zwykle potrzebne do ustawienia układu w stan początkowy. Polega to na wpisaniu do przerzutników, rejestrów czy liczników pożądanych informacji. Informacją tą jest przeważnie **0** logiczne. Dlatego określenie: **ustawianie w stan początkowy** jest utożsamiane z **zerowaniem układu**, pomimo że nie zawsze musi to odpowiadać fizycznemu ustawieniu w stan niski.

Wejścia zegarowe (wyzwalające) przerzutników wymagają także krótkich impulsów wyzwalających (dotyczy to zwłaszcza przerzutników typu MS (ang. *Master Slave*)). Dlatego układy generujące krótkie impulsy przy zmianie sygnału wejściowego są nazywane — **układami wyzwalającymi.** W matematyce natomiast operacją, która w wyniku daje funkcję zmieniającą skokowo swoją wartość przy zmianie argumentu tej operacji jest różniczkowanie. Stąd też układy wyzwalające są także nazywane **układami różniczkującymi.**

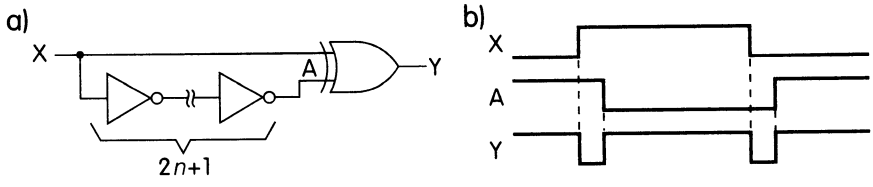
Do wytwarzania impulsów wyzwalających można zastosować opisane wcześniej scalone przerzutniki monostabilne albo układy wykorzystujące naturalne opóźnienia, wnoszone przez funktory logiczne (bramki), przerzutniki lub obwody *RC*.

Proste układy różniczkujące, zbudowane z samych bramek, przedstawiono na rys. 8.37 i rys. 8.38. W układach tych czas trwania impulsu wyjściowego zależy od opóźnienia wnoszonego przez szeregowo połączone negatory. Liczba negatorów w układzie z rys. 8.37 musi być nieparzysta.

Zauważmy jeszcze, że sygnał wyjściowy układu z rys. 8.38 ma częstotliwość dwa razy większą niż częstotliwość przebiegu wejściowego. Układ ten może być wykorzystywany więc jako tzw. **podwajacz częstotliwości.**

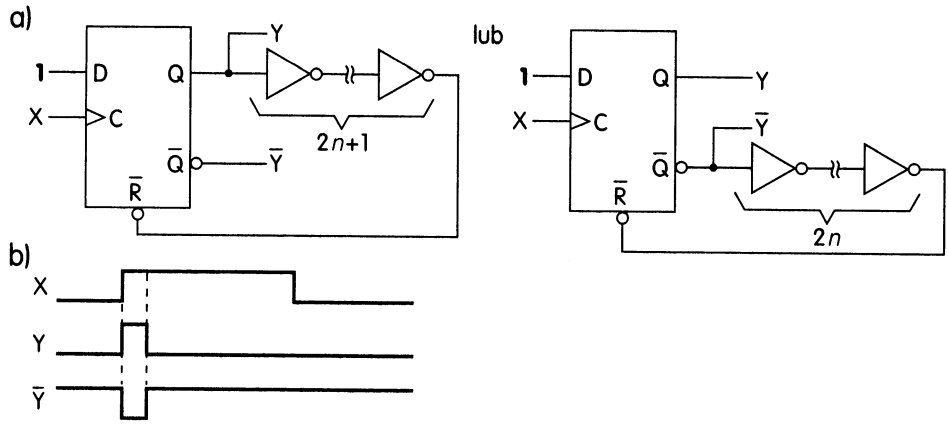


Rys. 8.37. Układ różniczkujący zbocze narastające: a) schemat logiczny; b) przebiegi czasowe

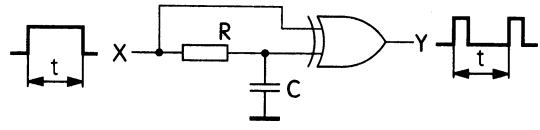


Rys. 8.38. Układ różniczkujący zbocza narastające i opadające: a) schemat logiczny; b) przebiegi czasowe

Układ wyzwalający zbudowany z wykorzystaniem przerzutnika przedstawiono na rys. 8.39, a na rys. 8.40 — układ wyzwalający zbudowany z wykorzystaniem układu opóźniającego *RC*.



Rys. 8.39. Układ różniczkujący zbocze dodatnie: a) schemat logiczny; b) przebiegi czasowe



Rys. 8.40. Układ różniczkujący oba zbocza (podwajacz częstotliwości): a) schemat logiczny; b) przebiegi czasowe

Należy pamiętać, że sygnał na wyjściu układu *RC* (przy dużych opóźnieniach — duża stała czasowa *RC*) może się zmieniać zbyt wolno. W takim przypadku należy użyć bramki Schmitta.

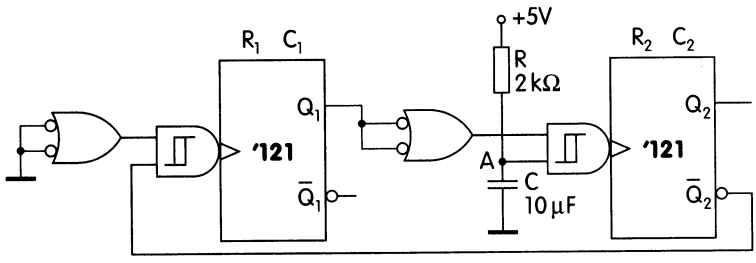
8.5. Generatory przebiegu prostokątnego

Różnorodność funkcji spełnianych przez przebiegi prostokątne w układach cyfrowych sprawia, że w zależności od przeznaczenia należy uwzględniać różne parametry i właściwości. Należą do nich:

— niezmienność częstotliwości generowanego przebiegu,
— zakres ustawiania generowanej częstotliwości,
— współczynnik wypełnienia (zakres jego zmian),
— pewność startu (wzbudzenie generatora), start z określoną fazą w chwili załączenia napięcia zasilającego.

Generator fali prostokątnej zawierający układ MCY74047 przedstawiono na rys. 8.9, natomiast zbudowany z dwóch przerzutników **'121** — na rys. 8.41.

Układ z rysunku 8.41 charakteryzuje się możliwością uzyskania częstotliwości przebiegu wyjściowego w zakresie 0,01 Hz ÷ 10 MHz i dowolnego współczynnika wypełnienia. Oba te parametry zależą od wartości stałych czasowych

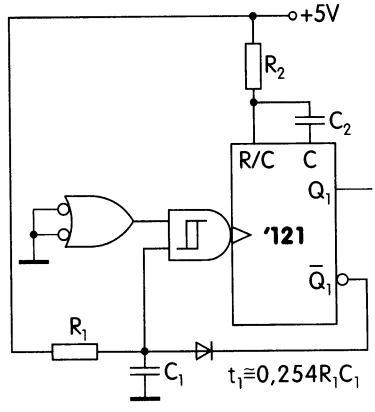


Rys. 8.41. Generator fali prostokątnej zbudowany z dwóch przerzutników monostabilnych '121

$t_1 = R_1C_1$ i $t_2 = R_2C_2$. Obwód *RC* zapewnia wzbudzenie się generatora po włączeniu napięcia zasilania. Wartości elementów *RC* podane na rys. 8.41 zapewniają niezawodne wzbudzenie się generatora w większości typowych zastosowań.

W wielu urządzeniach cyfrowych stosuje się generatory fali prostokątnej wyzwalane sygnałem zewnętrznym. W układzie z rys. 8.41 zamiast układu startowego *RC* możemy doprowadzić zewnętrzny sygnał, który będzie sygnałem bramkującym generator. Gdy sygnał bramkujący będzie miał poziom **H**, wówczas generator będzie generował przebieg prostokątny na wyjściu. Poziom niski **L** sygnału bramkującego spowoduje zablokowanie generowania drgań — na wyjściu generatora ustali się poziom niski **L**.

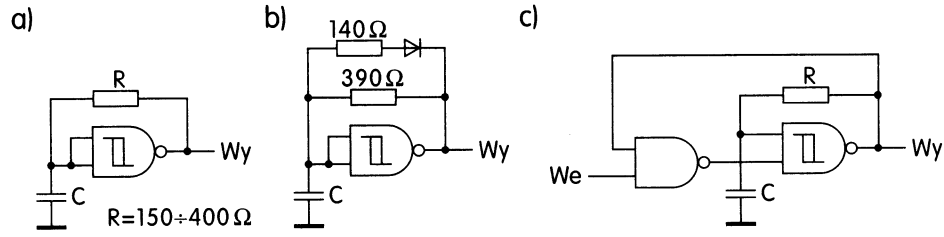
Schemat zasadniczy generatora zbudowanego z jednego tylko przerzutnika monostabilnego **'121** pokazano na rys. 8.42. W układzie tym kondensator C_1 ładuje się przez rezystor R_1 (dioda nie przewodzi, gdyż jej katoda jest dołączona do potencjału **H**). Na wyjściu **Q** jest stan niski **L**. W chwili, gdy napięcie na kondensatorze C_1 osiągnie napięcie progowe bramki Schmitta (1,6 ÷ 1,7 V), nastąpi wyzwolenie przerzutnika i jego wyjście zostanie ustawione w stan **H** na czas określony przez stałą czasową $t_2 = R_2C_2$ (patrz p. 8.2). Kondensator *C* rozładowuje się przez diodę, której anoda jest teraz dołączona do potencjału **L**. Napięcie to powinno zmaleć co najmniej do wartości ok. 0,9 V, aby bramka Schmitta mogła przejść w stan niski. Wynika z tego, że diodę należy dobrać tak, aby $U_{OL} + U_F < 0{,}9$ V (U_{OL} — napięcie wyjściowe w stanie niskim, U_F — napięcie przewodzenia diody).



Rys. 8.42. Generator fali prostokątnej zbudowany z jednego przerzutnika '121

Stałość częstotliwości drgań i współczynnika wypełnienia w obu tych układach nie jest zbyt duża. Generatory takie nie mogą być używane jako generatory wzorcowe.

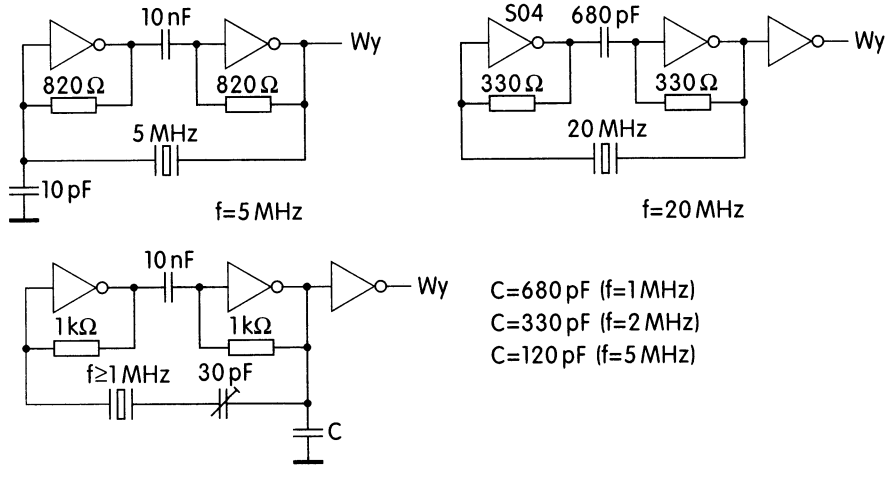
Bardzo proste układy generatorów można zbudować, stosując bramki z układem Schmitta (rys. 8.43). Generatory te umożliwiają uzyskanie częstotliwości generowanych przebiegów w przedziale 1 ÷ 10 MHz. W układzie z rysunku 8.43b otrzymuje się przebieg prostokątny, którego współczynnik wypełnienia $t_H/(t_H + t_L)$



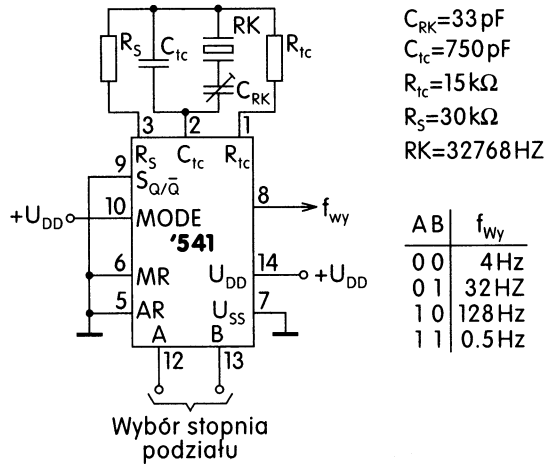
Rys. 8.43. Układy generatorów fali prostokątnej zbudowane z bramek Schmitta

wynosi ok. 1:2. W pozostałych układach współczynnik ten ma wartość ok. 1:3. Na rysunku 8.43c pokazano generator wyzwalany zewnętrznym sygnałem, doprowadzonym do wejścia bramki NAND.

Bardzo często w układach cyfrowych jest wymagana duża stałość częstotliwości generowanej fali impulsów prostokątnych. Do takich układów należą **częstościomierze, czasomierze, generatory wzorcowe, programowane generatory impulsów itp.** Wysokie wymagania odnośnie do stałości generowanych drgań spełniają **generatory z rezonatorami kwarcowymi** (**generatory kwarcowe**). Dodatkowe zwiększenie stałości generowanych drgań (zmniejszenie zakresu niepożądanych zmian częstotliwości przebiegu wyjściowego) uzyskuje się dzięki zastosowaniu generatora kwarcowego o częstotliwości znacznie przekraczającej



Rys. 8.44. Układy generatorów fali prostokątnej z rezonatorami kwarcowymi



Rys. 8.45. Programowany generator stabilizowany rezonatorem kwarcowym

częstotliwość pożądaną. Właściwą częstotliwość otrzymujemy wówczas, stosując odpowiedni dzielnik częstotliwości. Podział ten bowiem dotyczy również przedziału zmienności generowanego przebiegu (w generatorze kwarcowym), dzięki czemu przedział zmienności przebiegu wyjściowego (po podzieleniu) jest odpowiednio mniejszy. Do budowy generatorów z rezonatorami kwarcowymi o częstotliwościach generowanej fali przekraczających wartość 5 MHz należy stosować bramki serii szybkich S, F, AS. Kilka prostych układów generatorów kwarcowych, zbudowanych przy użyciu bramek, przedstawiono na rys. 8.44.

Przykład generatora kwarcowego zawierającego układ scalony CMOS MCY74541 pokazano na rys. 8.45. Elementy C_{tc}, R_{tc} i R_s dobiera się w taki sposób, aby uzyskać częstotliwość nieco większą niż częstotliwość stosowanego rezonatora kwarcowego (RK). Po dołączeniu rezonatora, trymerem C_{RK} dokładnie dostraja się oscylator do żądanej częstotliwości. Stosując rezonator o częstotliwości 32768 Hz oraz sterując układ od strony wejść **A** i **B**, można uzyskać na wyjściu częstotliwości 1/2, 4, 32 i 128 Hz.

- **Generatory przestrajane napięciem**

Generatory przestrajane napięciem **VCO** (ang. *Voltage Controlled Oscillator*) są to **przetworniki napięcia na częstotliwość.**

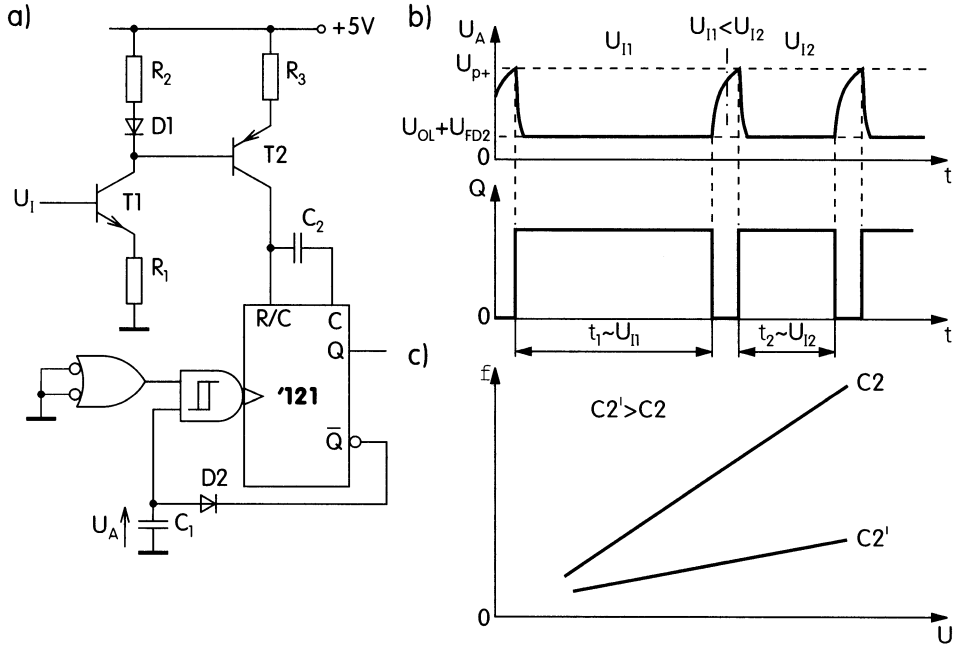
Charakterystyczną cechą przetworników jest tzw. **czułość.** Jest ona definiowana jako iloraz zmiany wielkości wyjściowej do zmiany wielkości wejściowej, która tę zmianę (na wyjściu) spowodowała. Czułość generatorów **VCO** można wyrazić wzorem

$$K = \frac{\Delta f}{\Delta U}$$

gdzie: Δf — zmiana częstotliwości wywołana zmianą ΔU napięcia wejściowego.

Od przetworników (tym samym i od generatorów **VCO**, które także są przetwornikami) wymaga się, aby były one **liniowe,** to znaczy, aby ich czułość była stała w całym zakresie zmian wielkości wejściowej. Pożądaną cechą jest także ***niezależność częstotliwości przebiegu wyjściowego od temperatury i napięcia zasilania.***

Istnieje wiele układów generatorów przestrajanych napięciem. Często są one budowane z przerzutników monostabilnych lub układów czasowych (patrz p. 8.1 oraz publikacja [21]). W tym miejscu omówimy jedynie bardzo prosty generator (rys. 8.46) zbudowany z przerzutnika **'121**, charakteryzujący się dobrą liniowością i dość dużą czułością (64 kHz/V).



Rys. 8.46. Generator VCO: a) schemat logiczny; b) przebiegi czasowe; c) charakterystyka przetwarzania $f = f(U_I)$

Zauważmy podobieństwo tego generatora do układu z rys. 8.42. Kondensator C_1 jest tutaj ładowany prądem I_{IL} bramki AND, jeżeli katoda diody *D2* ma potencjał wysoki, czyli w czasie, gdy przerzutnik jest w stanie stabilnym i na jego wyjściu **Q** jest stan niski **L**, a na wyjściu zanegowanym stan **H**. Gdy napięcie na kondensatorze osiągnie wartość napięcia przełączania bramki Schmitta, wówczas przerzutnik wygeneruje impuls. Podczas generowania impulsu wyjście $\overline{\mathbf{Q}}$ jest ustawiane w stan niski, co sprawia, że kondensator C_1 jest teraz rozładowywany przez diodę *D2* i tranzystor w stopniu wyjściowym przerzutnika. Aby uzyskać pewność, że napięcie na kondensatorze obniży się do poziomu, przy którym bramka ponownie przełączy, należy użyć diody germanowej o napięciu przewodzenia ok. 0,2 V. Po wygenerowaniu impulsu przez przerzutnik cykl pracy się powtarza. Ładowanie

kondensatora odbywa się za każdym razem w tych samych warunkach i trwa zawsze taki sam odcinek czasu. Odpowiada on stanowi niskiemu przebiegu wyjściowego. Parametrem regulowanym (zależnym od napięcia wejściowego) przebiegu wyjściowego jest czas trwania stanu wysokiego. W obwodzie ustalającym stałą czasową przerzutnika użyto bowiem źródła prądu stałego zamiast włączanej tam zwykle rezystancji. Szybkość ładowania kondensatora C_2 zależy liniowo od wartości prądu tego źródła. Prąd jest zaś proporcjonalny do napięcia wejściowego U_I. W ten sposób częstotliwość fali prostokątnej na wyjściu przerzutnika jest proporcjonalna do napięcia wejściowego.

- **Generatory o programowanej częstotliwości**

Istnieje bardzo wiele rozwiązań generatorów o programowanej częstotliwości przebiegu wyjściowego. Jednak najczęściej stosowana metoda polega na podziale jednej częstotliwości przez różne współczynniki podziału (rys. 8.47). Sposób ten jest wykorzystywany np. w układzie czasowym **'541** (opisanym w p. 8.2.5). Przebieg prostokątny, którego częstotliwość podlega podziałowi przez ten układ, może być generowany w wewnętrznym generatorze lub pobierany z zewnątrz.

Rys. 8.47. Schemat funkcjonalny generatora o programowanej częstotliwości

Jak już wyjaśniono w p. 7.3.3 licznik **mod *n*** to jest dzielnik częstotliwości przez *n*. Konstruowanie programowalnego dzielnika częstotliwości polegać zatem będzie na budowie programowalnego licznika. Jeden ze sposobów omówiono w p. 7.3.3, gdzie pokazano, jak zrobić licznik **mod 10** z licznika **mod 16**. Polegał on na zdekodowaniu odpowiedniego stanu licznika i wyzerowaniu go sygnałem pochodzącym z dekodera (rolę dekodera pełniła bramka NAND — rys. 7.16). Zamiast dekodera można wstawić komparator. W chwili gdy stan licznika będzie taki, jak słowo sterujące, na wyjściu komparatora pojawi się sygnał, który może zostać wykorzystany do zerowania licznika. Innym sposobem jest zastosowanie licznika liczącego wstecz i mającego możliwość wpisania do niego dowolnej licz-

Rys. 8.48. Schemat funkcjonalny programowanego dzielnika częstotliwości z symetrycznym przebiegiem wyjściowym

by. Liczba ta jest wpisywana każdorazowo po osiągnięciu przez licznik stanu zerowego. Metody te zostaną omówione dokładniej przy okazji opisu liczników scalonych (rozdz. 12.).

Bardzo często wymaga się, aby przebieg wyjściowy dzielnika (:*n*) miał współczynnik wypełnienia $t_H/(t_H + t_L) = 1/2$. Taki przebieg nazywamy **przebiegiem symetrycznym**. Sposób pozwalający spełnić ten wymóg przedstawiono na rys. 8.48.

Jako podwajacz częstotliwości można wykorzystać układ różniczkujący oba zbocza sygnału wejściowego (rys. 8.38 lub 8.29). Jako dzielnika przez dwa można użyć licznika **mod 2**, czyli dwójkę liczącą. Układy dwójek liczących (zbudowanych z przerzutników) omówiono w p. 7.3.3.

Pytania i zadania

1. Narysuj przebiegi czasowe w układzie z rys. 8.37, zmieniając w nim nieparzystą liczbę negatorów na parzystą.
2. Wykonaj zadanie 1. dla układu z rys. 8.38.
3. Narysuj przebiegi czasowe w układzie z rys. 8.41. Rozpocznij ich rysowanie od chwili określonej włączeniem zasilania układu.
4. Narysuj przebiegi czasowe w układzie z rys. 8.41, zastępując w nim obwód startowy *RC* sygnałem bramkującym **B.** Przeanalizuj przypadki różnych chwil zaniku (przejścia z **B** = **1** na **B** = **0**) sygnału bramkującego **B**.
5. Narysuj przebiegi czasowe w układzie z rys. 8.42.
6. Wyszukaj w literaturze (np. w publikacji [21]) układ generatora **VCO**. Opisz jego działanie. Narysuj przebiegi czasowe w charakterystycznych punktach układu.
7. Przedstaw schemat funkcjonalny i opisz zasadę działania generatora fali prostokątnej z kodowym wybieraniem częstotliwości.
8. Przedstaw schemat funkcjonalny i opisz zasadę działania programowanego dzielnika częstotliwości z przebiegiem wyjściowym o współczynniku wypełnienia równym 1/2.
9. Co to są układy wyzwalające (różniczkujące)? Jakie jest ich zastosowanie w technice cyfrowej?

9

Wybrane zagadnienia projektowania układów cyfrowych

9.1. Wprowadzenie

W rozdziale 3. podręcznika poznaliśmy metody opisu, minimalizacji i projektowania układów kombinacyjnych. Schematy logiczne tych układów rysowaliśmy, zakładając, iż dysponujemy dowolnymi bramkami lub bramkami jednego typu (np. NAND), ale o dowolnej liczbie wejść. Elementy te traktowaliśmy jako idealne, to znaczy takie, które nie wnoszą żadnych opóźnień, a zmiana stanu na wyjściu odbywa się zgodnie z przebiegiem charakterystyki przejściowej elementu idealnego.

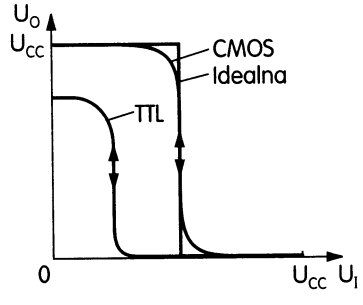
W tym rozdziale zajmiemy się ograniczeniami nałożonymi na te układy, wynikającymi ze stosowania rzeczywistych (a nie wyidealizowanych) elementów. Poruszone zostaną także zagadnienia współpracy układów TTL i CMOS.

Informacja z wyjścia układów cyfrowych służy do sterowania elementów wykonawczych lub jest wyświetlana (wizualizowana) w sposób czytelny dla człowieka. Elementami sterowanymi bezpośrednio z wyjść cyfrowych będą więc najczęściej takie elementy, jak: dioda świecąca LED, tranzystor (wzmacniacz), przekaźnik (p. 9.5).

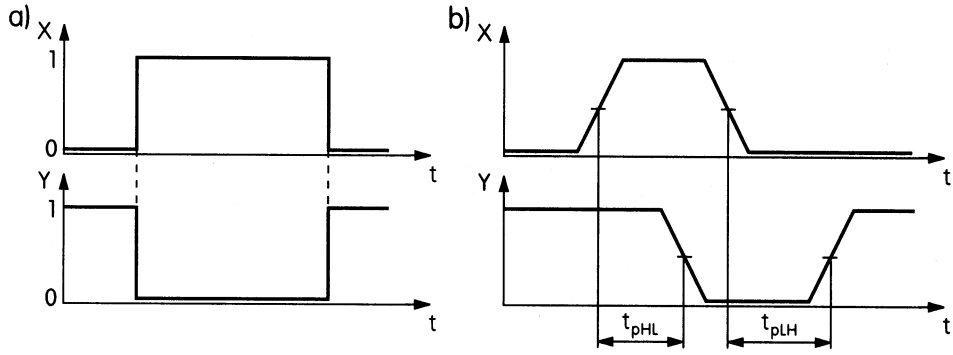
9.2. Zjawiska szkodliwe w układach kombinacyjnych

Charakterystyki przejściowe (charakteryzujące właściwości przełączające) bramek TTL lub CMOS różnią się od charakterystyki przejściowej elementu idealnego, co obrazuje rys. 9.1.

Idealny element nie wprowadza w torze przetwarzanego sygnału żadnego opóźnienia. Jak jednak już wiemy, rzeczywiste właściwości transmisyjne (przenoszenia sygnału przez element) odbiegają od idealnych. Każdy rzeczywisty ele-



Rys. 9.1. Charakterystyki przełączania



Rys. 9.2. Przebiegi czasowe przełączania: a) bramki idealnej; b) bramki rzeczywistej

ment TTL czy CMOS wprowadza pewne opóźnienie, którego miarą jest **czas propagacji** t_p. Przebiegi czasowe odwzorowujące przełączanie bramki idealnej i rzeczywistej przedstawiono na rys. 9.2.

Nieidealne właściwości przełączające i transmisyjne powodują, że układy działają w sposób odbiegający od ich matematycznego opisu, którym posługuje się projektant systemu cyfrowego.

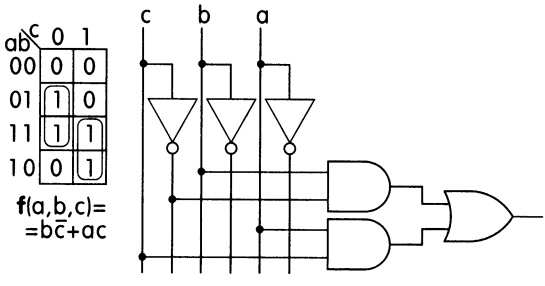
Zjawisko powstawania błędnych stanów na wyjściu układu w stanach przejściowych nosi nazwę **hazardu.** Błędne stany na wyjściu powstają w stanach przejściowych, gdyż wtedy dają o sobie znać rzeczywiste właściwości przełączające i transmisyjne. ***Jeżeli źródłem takiego niepożądanego stanu na wyjściu układu są nieidealne właściwości przełączające, to taki hazard będziemy nazywać*** **hazardem statycznym;** ***jeżeli natomiast — nieidealne właściwości transmisyjne, to*** **— hazardem dynamicznym.**

9.2.1. Hazard statyczny

Zminimalizujmy i zrealizujmy z dowolnych bramek funkcję przełączającą $\mathbf{f(a,b,c)} = \Sigma(2,5,6,7)$ — rys. 9.3.

Przyjmijmy, że w rozpatrywanej chwili sygnały wejściowe $\mathbf{a = b = 1}$. Wówczas wyrażenie opisujące działanie układu przyjmie postać

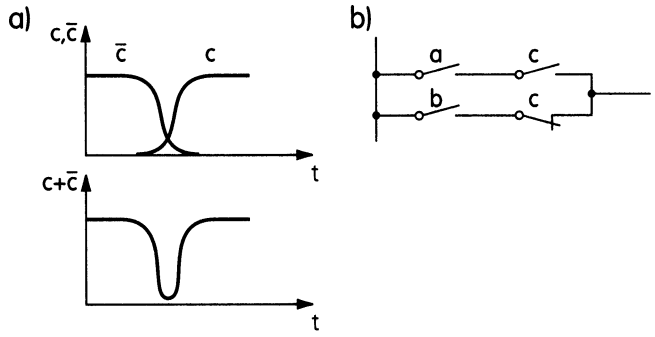
$$\mathbf{f(a,b,c) = b\bar{c} + ac = 1\bar{c} + 1c = \bar{c} + c = 1} \tag{9.1}$$



Rys. 9.3. Tablica Karnaugha funkcji $f(a,b,c) = \Sigma(2,5,6,7)$ i jej minimalna realizacja

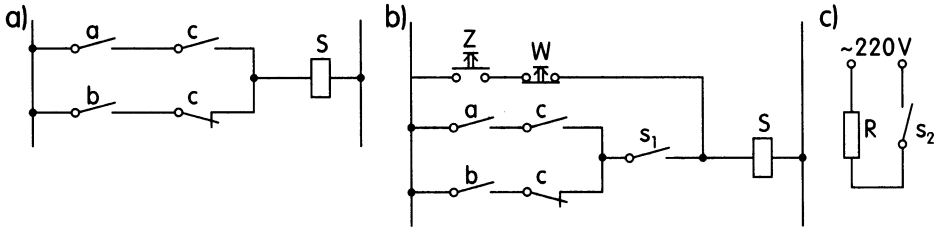
Z zapisu (9.1) wynika, że sygnał wyjściowy powinien mieć stałą wartość **1**, niezależnie od tego, co będzie się działo na wejściu **c** (o ile tylko **a** = **b** = **1**). Przeanalizujmy teraz, jak zachowa się rzeczywisty układ, gdy sygnał **c** będzie zmieniał swą wartość np. z **0** na **1**. Pomijamy opóźnienia wnoszone przez poszczególne elementy, uwzględniając jedynie rzeczywiste właściwości przełączające. Na wejścia bramki OR oddziałują dwa, jednocześnie zmieniające się, sygnały. Jeden z nich zmienia swój stan z **0** na **1** (sygnał **c**), drugi z **1** na **0** (sygnał $\bar{\mathbf{c}}$). Sumowane przez bramkę OR przebiegi wejściowe przedstawiono na rys. 9.4.

Jak widać na rys. 9.4 przebieg na wyjściu bramki OR nie spełnia zależności (9.1). Na wyjściu poja wia się błędny sygnał o poziomie **L**.



Rys. 9.4. Przebiegi czasowe (a) oraz realizacja stykowa (b) układu z rys. 9.3

Zjawisko to łatwiej jest zrozumieć, analizując stykową realizację układu (rys. 9.4b). Jeżeli **a** = **b** = **1** to zestyki **a** i **b** są zamknięte. Zmiana sygnału **c** z **0** na **1** oznacza (w tym układzie), że zestyk zwierny **c** zamyka się, a rozwierny **c** otwiera. Nietrudno jest wyobrazić sobie taką sytuację, że zanim zestyk zwierny zamknie obwód, to zestyk rozwierny już go otworzy. (Jeżeli do realizacji tego układu użyjemy zestyku przełączającego, to zjawisko powstawania przerwy w obwodzie w chwili jego przełączania staje się jeszcze bardziej ewidentne.) Przez chwilę obwód wyjściowy jest w stanie bezprądowym. Gdy elementem sterowanym jest np. stycznik załączający element grzejny (rys. 9.5c), wówczas efekt hazardu będzie niezauważalny (rys. 9.5a). Jeżeli jednak analizowany układ znajdzie się w obwo-

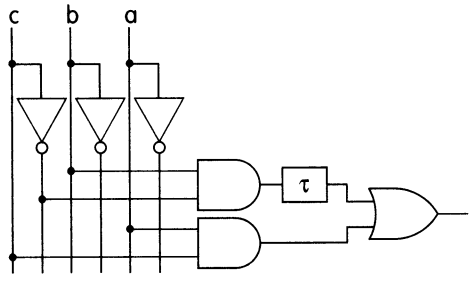


Rys. 9.5. Układ z hazardem nie powodujący zauważalnych skutków jego wystąpienia (a); układ, w którym hazard może uniemożliwić jego poprawną pracę (b); obwód sterowany z grzejnikiem *R* (c)

dzie podtrzymania stycznika, to hazard taki może spowodować trwałe odłączenie elementu grzejnego załączanego tym stycznikiem (rys. 9.5b).

W odniesieniu do układów kombinacyjnych zbudowanych z bramek wnioski powyżej sformułowane zachowują swą moc. Jeżeli obwód z hazardem steruje układem kombinacyjnym lub elementami wyjściowymi takimi, jak: dioda LED, tranzystor, przekaźnik, to skutki wystąpienia hazardu są pomijalne. ***Jeżeli jednak obwód sterowany jest układem sekwencyjnym (mającym pamięć), to hazard należy wyeliminować.***

W układzie już zbudowanym hazard możemy wyeliminować poprzez wprowadzenie w torze sygnału $\overline{\mathbf{c}}$ opóźnienia. Przedstawiono to na rys. 9.6. Opóźnienie takie można uzyskać wstawiając np. dwa negatory lub układ *RC*. Opóźnienie takie nie likwiduje hazardu (a wręcz go pogłębia) występującego przy zmianie sygnału **c** z **H** na **L**.



Rys. 9.6. Eliminacja hazardu za pomocą opóźnienia τ

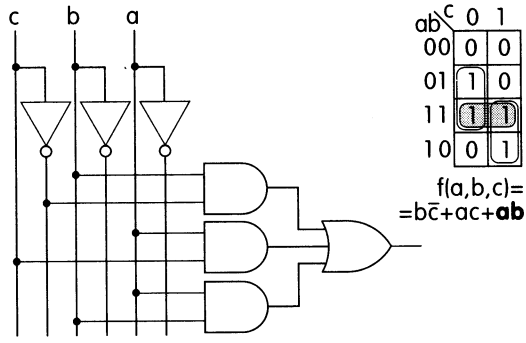
Określenie wpływu opóźnienia τ na przebiegi czasowe sygnałów wejściowych i sygnału wyjściowego bramki OR pozostawia się Czytelnikowi.

Jednak eliminacja hazardu w już zbudowanym układzie to ostateczność. ***Należy tak projektować układy kombinacyjne, aby nie było możliwe wystąpienie hazardu.*** Wyjątkiem od tej reguły są sytuacje, gdy skutki hazardu w żadnym przypadku nie będą prowadziły do niewłaściwej pracy układu.

Eliminacja hazardu już na etapie projektowania układu jest istotna z tego powodu, iż skutki hazardu w działającym układzie mogą mieć charakter przypadkowy. Oznacza to, że układ z hazardem może działać poprawnie przez pewien czas, a już po chwili (np. wskutek zmiany temperatury otoczenia) działanie układu będzie błędne. Lokalizacja takiego „uszkodzenia" jest szczególnie trudna.

Zakreślmy w tablicy Karnaugha na rys. 9.3 dodatkową grupę jedynek w taki sposób, aby znalazły się w niej jedynki z obu wcześniej zakreślonych grup (patrz rys. 9.7).

W poprzednim układzie zjawisko hazardu wystąpiło w sytuacji, gdy $\mathbf{a} = \mathbf{b} = \mathbf{1}$. Obecnie dodatkowa bramka, realizująca iloczyn sygnałów **a** i **b**, doprowadza sygnał **1** do wejścia bramki OR. Teraz więc zmiana sygnału **c** nie może mieć żadnego wpływu na sygnał wyjściowy, gdyż wystarczy, aby jedno z wejść bramki OR było w stanie **H**, by także wyjście było w stanie **H** niezależnie od tego, co się



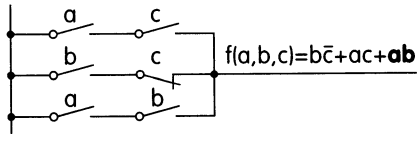
Rys. 9.7. Tablica Karnaugha funkcji f(a,b,c) = Σ(2,5,6,7) i jej realizacja pozbawiona hazardu

dzieje na pozostałych wejściach. Na wyjściu układu jest więc stale sygnał **1** zgodnie z zależnością opisującą jego działanie. ***Hazard został wyeliminowany.***

Możemy się o tym przekonać, analizując odpowiedni układ stykowy realizujący funkcję

$$\mathbf{f(a,b,c) = b\bar{c} + ac + \mathit{ab}} \tag{9.2}$$

Schemat układu stykowego pozbawionego hazardu przedstawiono na rys. 9.8. Jeżeli **a** = **b** =**1** (a w takiej sytuacji wystąpił hazard), to zestyki **a** i **b** są zamknięte. Dzięki dodatkowej gałęzi utworzonej przez szeregowo połączone zestyki **a** i **b** obwód prądu będzie stale zamknięty, nawet jeżeli zestyki **c** będą zmieniać swoje położenie.



Rys. 9.8. Układ stykowy po wyeliminowaniu hazardu

Powyższe rozważania, przeprowadzone na konkretnym przykładzie, możemy oczywiście uogólnić na całą klasę układów kombinacyjnych.

Zjawisko* hazardu statycznego *będzie występowało zawsze wtedy, kiedy w tablicy Karnaugha funkcji, dla której poszukujemy postaci minimalnej, wystąpią grupy sąsiadujące ze sobą.

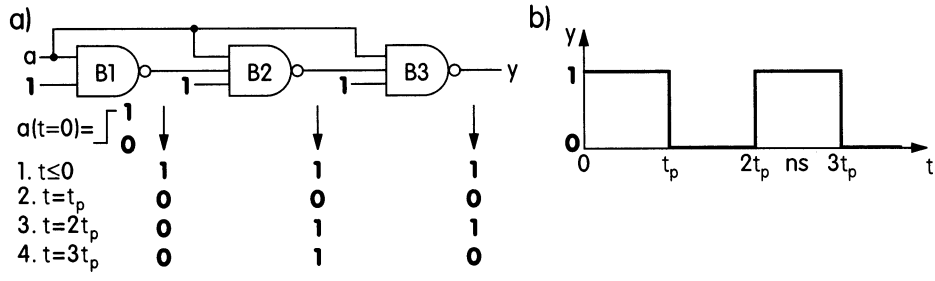
Oczywiście mogą to być grupy jedynek, ale także i grupy zer, jeżeli to je wybraliśmy do sklejania. Sąsiedztwo należy rozumieć w ten sposób, że grupy takie mają co najmniej jedną krawędź wspólną. Grupy stykające się jedynie rogami nie sąsiadują ze sobą. Żadnego wpływu na występowanie hazardu lub jego eliminację nie będzie miał także rodzaj bramek, jakich użyjemy do realizacji funkcji (tzn. AND, OR, NOT, NAND, NOR).

Hazard ***eliminuje się, wprowadzając dodatkowe grupy w tablicy Karnaugha, zawierające pola wchodzące w skład sąsiadujących ze sobą grup.***

Z tego, co powiedziano powyżej odnośnie do identyfikacji występowania hazardu, wynika, że proponowany sposób likwidacji hazardu może być zawsze zastosowany. Hazard występuje wówczas, gdy są stykające się grupy. A jeżeli grupy stykają się (sąsiadują ze sobą), to można zawsze wprowadzić dodatkową grupę eliminującą hazard.

9.2.2. Hazard dynamiczny

Nieidealne właściwości transmisyjne są przyczyną powstawania hazardu dynamicznego. Jego powstawanie pokażemy na przykładzie fragmentu układu zbudowanego z trzech bramek NAND, połączonych jak na rys. 9.9. Są to bramki wielowejściowe, przy czym wejścia współpracujące z innymi fragmentami układu (nie uwidocznionymi na rysunku) mają wartość **1**, co symbolicznie zaznaczono na schemacie. Dla uproszczenia analizy przyjmiemy, że bramki są idealnymi elementami przełączającymi (patrz idealna charakterystyka przejściowa na rys. 9.1). Przyjmijmy także, że każda z nich ma taki sam czas propagacji t_p.



Rys. 9.9. Powstawanie hazardu dynamicznego: a) schemat przykładowego układu z hazardem; b) przebieg czasowy sygnału wyjściowego *y*

W stanie ustalonym dla sygnału wejściowego **a** = **0** wyjścia wszystkich trzech bramek są w stanie **1**. Przyjmijmy, że sygnał **a** zmienia swój stan z **0** na **1** w chwili $t = 0$. Ponieważ sygnał **a** jest doprowadzony do wszystkich trzech bramek, przeto w chwili $t = 0$ stan **1** zacznie oddziaływać na wszystkie trzy bramki jednocześnie. Pozostałe wejścia bramek są w stanie **1**. W wyniku opóźnienia wprowadzanego przez bramki odpowiedź na zmianę sygnału z **0** na **1** pojawi się na wyjściach tych bramek dopiero po czasie t_p (krok 2. na rys. 9.9). W chwili $t = t_p$ na wyjściach wszystkich trzech bramek jest stan niski **0**. Ponieważ na wejściu bramki *B1* nic się już nie zmienia, więc wyjście pozostaje w stanie **0**. Na wejścia bramek *B2* i *B3* oddziałuje stan niski, pochodzący z wyjść poprzedzających je bramek NAND. Stan **0** pojawił się tam w czasie $t = t_p$, a zatem jego skutek przeniesie się na wyjście bramek po kolejnym czasie t_p, czyli w chwili $t = 2t_p$ (krok 3. na rys. 9.9). Wyjścia bramek *B2* i *B3* zostaną ustawione w stan **1**. Stany wejść bramek *B1* i *B3* nie ulegają więc zmianie. Zmienia się tylko stan wejść bramki *B3*, gdyż jest ona sterowana z wyjścia bramki *B2*. Po kolejnym czasie t_p, czyli w chwili $t = 3t_p$ spowoduje

to, że wyjście bramki *B3* zostanie ponownie ustawione w stan **0**. Ponieważ stany wejść bramek NAND nie zmieniają się już, przeto osiągnięty stan jest stanem ustalonym. W stanie ustalonym przed zmianą sygnału **a** wyjście było w stanie **1**. W stanie ustalonym po zmianie sygnału **a** wyjście jest w stanie **0**. Teoretycznie więc na wyjściu powinno się pojawić jedynie zbocze ujemne sygnału wyjściowego **y**. Przebiegi czasowe (rys. 9.9b) obrazują, jak zmienia się sygnał wyjściowy bramki *B3*. Widzimy, że rzeczywisty przebieg sygnału wyjściowego różni się od oczekiwanego (teoretycznego). Dodatkowy impuls o czasie trwania t_p to właśnie hazard dynamiczny.

Skutki hazardu dynamicznego mogą być identyczne, jak hazardu statycznego. Należy zatem tak projektować układy logiczne, aby hazard ten nie występował. Hazard dynamiczny powstaje wówczas, gdy sygnał jest przenoszony przez nieparzystą liczbę bramek (jak np. w przykładzie powyżej). Zlikwidować go można, wprowadzając opóźnienie lub zmieniając konfigurację układu.

***Likwidacja* hazardu statycznego *(poprzez wprowadzenie dodatkowych grup) likwiduje jednocześnie* hazard dynamiczny.**

Budowane są również układy, w których opóźnianie sygnału wnoszone przez element cyfrowy odgrywa pozytywną rolę. Są nimi na przykład układy różniczkujące, opisane w p. 8.3.

Pytania i zadania

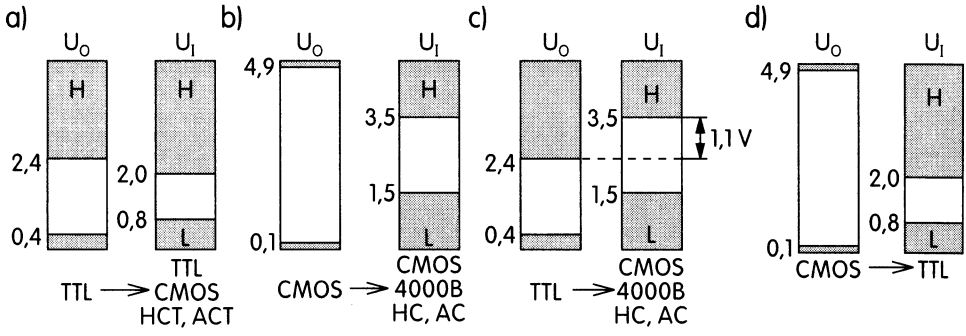
1. Porównaj właściwości przełączające elementu idealnego i elementu TTL oraz CMOS.
2. Porównaj właściwości transmisyjne elementu idealnego i elementu TTL oraz CMOS.
3. Co to jest hazard statyczny i dynamiczny?
4. Określ sposób identyfikacji występowania hazardu na etapie projektowania układu kombinacyjnego.
5. Jakie mogą być objawy występowania hazardu? W jakich układach występowanie hazardu jest szczególnie szkodliwe? Omów sposoby eliminacji hazardu.
6. Znajdź postać minimalną funkcji $\mathbf{f(a,b,c,d)} = \Sigma(0,2,3,4,8,10,11,12)$. Zrealizuj ją, używając bramek NAND. Jeżeli występuje hazard to wyeliminuj go. Narysuj schemat logiczny układu pozbawionego hazardu (także używając bramek NAND).
7. Dana jest funkcja $\mathbf{f(a,b,c,d)} = \Pi(0,1,4,5,6,14)$ i bramki NOR. Wykonaj polecenie jak w zad. 6.
8. Wykonaj zadanie 6., używając elementów stykowych.
9. Wykonaj zadanie 7., używając elementów stykowych.
10. Podziel niżej wymienione elementy sterowane przez układ z hazardem na dwie grupy: I grupa — ma zawierać te elementy, dla których zjawisko hazardu nie będzie miało istotnego wpływu na ich funkcjonowanie; II grupa — ma zawierać te elementy, które będą funkcjonować nieprawidłowo, gdy pojawi się hazard. Wykaz elementów: dioda LED, licznik, tranzystor sterujący przekaźnikiem, przerzutnik, rejestr, tranzystor włączający element grzejny.

9.3. Współpraca układów TTL i CMOS

Często, aby w pełni wykorzystać wszystkie zalety poszczególnych układów cyfrowych, buduje się systemy cyfrowe z elementów należących do różnych rodzin czy serii układów. W takich systemach problemem staje się zapewnienie współpracy pomiędzy elementami wykonanymi różnymi technikami. Należy osiągać zgodność takich parametrów technicznych, jak: napięcia zasilania, poziomy logiczne, prądy wejściowe i wyjściowe (obciążalność wejść i wyjść). Jeżeli zagadnienie współpracy pomiędzy jakimiś dwoma rodzinami jest typowe, to sprzężenie pomiędzy tymi układami będzie można zrealizować za pomocą specjalnych (produkowanych w tym celu) układów — nazywanych **translatorami** lub **konwerterami.** Dla danej pary rodzin układów cyfrowych mogą być potrzebne dwa rodzaje translatorów. Jeden, umożliwiający sprzężenie wyjść rodziny I z wejściami rodziny II, drugi zaś realizujący odwrotny kierunek sprzężenia. Budowane są także **konwertery uniwersalne,** umożliwiające uzyskanie dowolnego kierunku sprzężenia w zależności od stanu wejść sterujących.

Ze względu na największe rozpowszechnienie ma zasadnicze znaczenie współpraca układów TTL i CMOS.

Układy CMOS serii 4000B (MCY74) mogą pracować przy zasilaniu napięciem o wartości 3 ÷ 18 V, układy CMOS serii HC i AC przy zasilaniu napięciem o wartości 2 ÷ 6 V, natomiast układy TTL (czy CMOS serii HCT i ACT) przy zasilaniu napięciem równym 5 V. Czy zatem wystarczy zasilać je z jednego źródła (5 V), aby można było je ze sobą dowolnie łączyć? Sformułowanie odpowiedzi na to pytanie ułatwi graficzne przedstawienie zakresów napięć wejściowych i wyjściowych układów TTL i CMOS przy zasilaniu z tego samego źródła o napięciu $U = 5$ V (rys. 9.10).



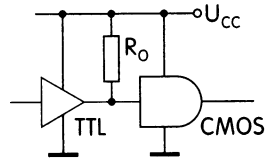
Rys. 9.10. Porównanie poziomów napięć wejściowych i wyjściowych układów TTL i CMOS

Rysunek ten obrazuje jedynie dopasowanie poziomów napięć przy różnych wariantach współpracy. Zauważmy, że występuje odpowiedniość poziomów napięć (w obie strony) dla układów TTL oraz CMOS HCT, ACT. Ponieważ i obciążalność wyjść układów HCT i ACT jest odpowiednio duża (patrz p. 6.4), więc sprzężenie ich ze sobą (niezależnie od kierunku) jest całkowicie dozwolone. Mówimy, że są one **kompatybilne.**

Zagadnienie współpracy obu rodzin (CMOS jedynie serie 4000B, HC, AC) zostanie omówione dokładniej — z uwzględnieniem obciążalności wyjść.

- **Sprzężenie TTL ⇒ CMOS**

Prąd wejściowy bramki CMOS w porównaniu z obciążalnością bramki TTL jest pomijalnie mały. Ze względu na obciążalność prądową do wyjścia bramki TTL można dołączyć dowolną liczbę wejść CMOS. Jednak, jak widać z rys. 9.10c, mamy tu do czynienia z niedopasowaniem poziomów napięciowych układów 4000B, HC i AC. Minimalne napięcie wyjściowe układu TTL w stanie **H** jest o 1,1V mniejsze niż minimalne napięcia wejściowe układu CMOS w stanie **H**. Dlatego przy sprzęganiu TTL → CMOS należy stosować dodatkowy **rezystor podciągający** R_O (rys. 9.11).



Rys. 9.11. Sprzężenie TTL ⇒ CMOS z użyciem rezystora podciągającego R_0

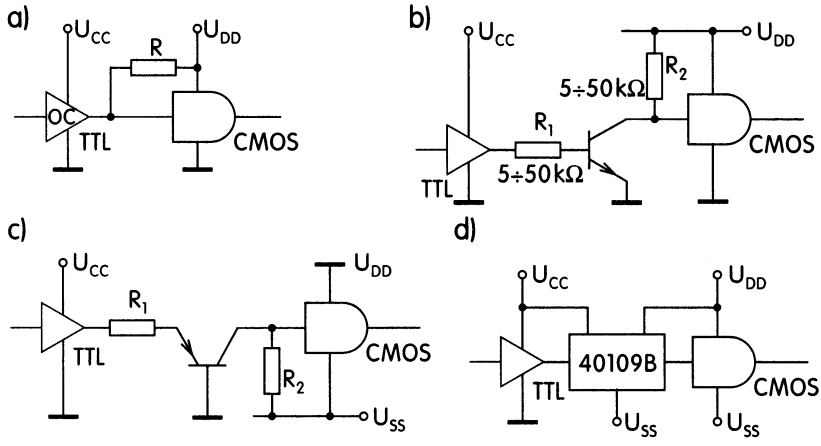
Zadaniem rezystora podciągającego jest zwiększenie napięcia wyjściowego z układu TTL w stanie **H**. W tablicy 9.1 przedstawiono wartości minimalne i maksymalne rezystora podciągającego. ***Zaleca się stosować rezystory o wartościach* 1,5 ÷ 4,7 kΩ. *Ponadto do wyjścia bramki TTL sterującej układami CMOS nie należy dołączać wejść TTL.***

Tablica 9.1. **Wartości rezystora podciągającego**

Rezystancja		Seria	
		74	74LS
$R_{O\,min}$	Ω	390	810
$R_{O\,max}$	kΩ	4,7	12

Układy CMOS mają lepsze parametry dynamiczne (krótszy czas propagacji) i większą odporność na zakłócenia (większy margines zakłóceń) przy wyższych napięciach zasilania. Aby więc wykorzystać te zalety, należy układy CMOS zasilać napięciami większymi niż 5 V. Wówczas do wysterowania układu CMOS należy zastosować układ transformujący poziomy logiczne TTL na odpowiednio wyższe poziomy logiczne CMOS. Sprzężenie takie można zrealizować, wykorzystując bramki TTL z otwartym obwodem kolektora (typu OC). Można też stosować układy prostych kluczy tranzystorowych albo specjalizowane układy sprzęgające (translatory). Przykłady takich układów sprzęgających przedstawiono na rys. 9.12.

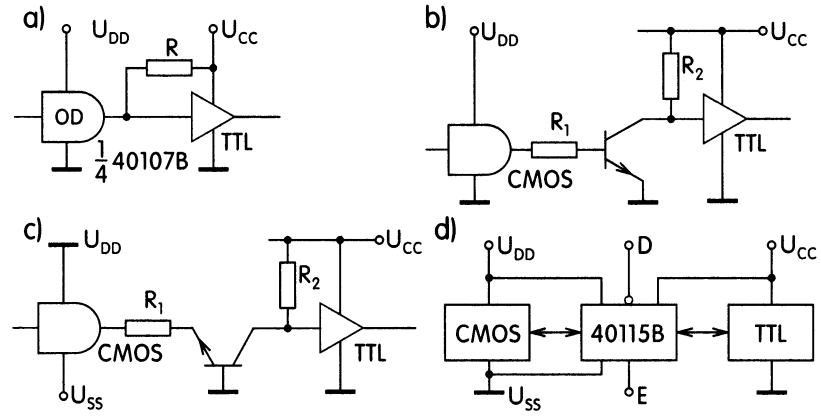
Układ 40109B zawiera cztery konwertery (translatory) poziomów logicznych TTL na poziomy CMOS. Jest on elementem trójstanowym. Podanie **0** na wejście **EA** (ang. *Enable A* — nie pokazane na rys. 9.12c) ustawia go w stan wielkiej impedancji.



Rys. 9.12. Sprzężenie TTL ⇒ CMOS przy różnych napięciach zasilania: a) za pomocą bramki TTL typu OC; b, c) za pomocą klucza tranzystorowego w układzie ze wspólnym emiterem oraz w układzie ze wspólną bazą; d) za pomocą translatora 40109B

- **Sprzężenie CMOS ⇒ TTL**

Przy stosowaniu jednego źródła zasilania nie istnieje problem dopasowania napięciowego (rys. 9.10d), zatem jest możliwe bezpośrednie sprzężenie CMOS ⇒ TTL. Przeciwwskazaniem bezpośredniego łączenia jest mała wydajność prądowa układów CMOS (nie dotyczy to układów HC i AC). Mała wydajność prądowa ma znaczenie w stanie niskim, w którym prąd I_{IL} wejścia TTL wynosi 1,6 mA i wejścia TTL LS wynosi 0,4 mA (wartości maksymalne), a wydajność prądowa standardowego wyjścia CMOS 4000B nie przekracza 1 mA (na ogół wynosi 0,4 mA). Wystarcza więc na wysterowanie wejścia TTL LS, ale jest za mała w przypadku serii standardowej TTL. W stanie wysokim każda bramka CMOS jest zdolna wysterować wejście TTL, gdyż prąd wejściowy (wejścia TTL) nie przekracza warto-



Rys. 9.13. Sprzężenie CMOS ⇒ TTL przy różnych napięciach zasilania: a) za pomocą bramki CMOS z otwartym drenem; b, c) za pomocą klucza tranzystorowego w układzie ze wspólnym emiterem oraz w układzie ze wspólną bazą; d) za pomocą translatora 40115B

ści 40 μA. Bramka buforowa CMOS (z końcówkami mocy) umożliwia wysterowanie nawet 2 do 4 wejść TTL (standardowych).

Zwiększenie napięcia zasilania powoduje zwiększenie obciążalności prądowej wyjść układów CMOS, ale jednocześnie likwiduje dopasowanie napięciowe. Przy różnych napięciach zasilania sprzężenie CMOS ⇒ TTL można zrealizować w układach analogicznych do układów sprzężenia TTL ⇒ CMOS. Mogą to być układy wykorzystujące bramki CMOS z otwartym drenem. Można też stosować układy prostych kluczy tranzystorowych albo specjalizowane układy sprzęgające (konwertery). Przykłady takich układów sprzęgających przedstawiono na rys. 9.13.

Tablica 9.2. **Funkcje realizowane przez układ 40115B**

Wejście		Funkcja
E	D	
0	0	CMOS ⇒ TTL
1	0	TTL ⇒ CMOS
0	1	TTL ⇒ CMOS-(Z)
1	1	Zabroniony
(Z) — stan wielkiej impedancji.		

Układ 40115B zawiera osiem dwukierunkowych konwerterów napięcia sterowanych dwoma sygnałami: **D** (ang. *Disable*) i **E** (ang. *Enable*). W tablicy 9.2 zestawiono funkcje realizowane przez układ.

Pytania i zadania

1. Omów zagadnienie dopasowania poziomów napięciowych sygnałów wyjściowych i wejściowych układów TTL i CMOS.
2. Omów współpracę układów TTL ⇒ CMOS.
3. Omów współpracę układów CMOS ⇒ TTL.
4. Jakie układy nazywamy translatorami (konwerterami)? Do czego służą te układy?
5. Czy można dokonać zamiany elementu TTL LS na element HCT (ACT)? Czy zawsze jest dopuszczalna zamiana elementu HCT (ACT) na element TTL LS?

9.4. Układy wejściowe

Źródłem sygnałów wejściowych najczęściej są:

— zestyki przełączników, przekaźników, wyłączników krańcowych (sygnały dwuwartościowe);

— przetworniki położenia liniowego lub kątowego na sygnały cyfrowe (przedstawione np. w kodzie Graya);

— czujniki z analogowymi sygnałami wyjściowymi;
— klawiatury;
— dalekopisy, czytniki urządzeń przechowujących informację (takich jak np. taśma magnetyczna, dysk magnetyczny lub optyczny).

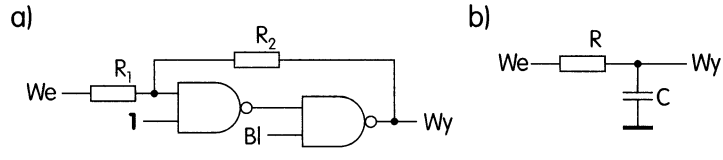
Sygnały otrzymywane z powyższych źródeł nie zawsze można wykorzystać w sposób bezpośredni. Często wymagają one wzmocnienia, translacji poziomów, galwanicznej separacji, wytłumienia zakłóceń (oscylacji), czy obróbki kształtu (głównie dotyczy to zboczy sygnałów wejściowych). Każde urządzenie cyfrowe ma układy wejściowe, przy czym typowe układy to: układy formowania i regeneracji sygnałów, układy współpracy z zestykami, układy rozdzielenia galwanicznego. Do układów wejściowych zalicza się także układy translacji poziomów. Niektóre z nich omówiono w p. 9.3. Natomiast układy separacji galwanicznej są stosowane zarówno w obwodach wejściowych, jak i wyjściowych.

9.4.1. Układy formowania i regeneracji sygnałów

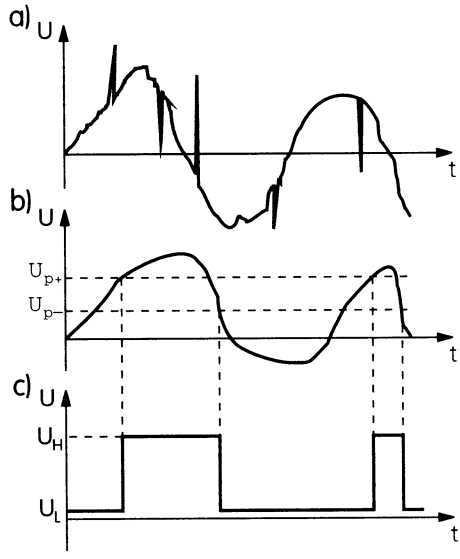
Układy formowania i regeneracji sygnałów stosuje się w celu:
— kształtowania przebiegów prostokątnych z przebiegów sinusoidalnych, trójkątnych lub innego kształtu (np. o dużych czasach narastania i opadania sygnałów);
— tłumienia zakłóceń.

Źródła sygnałów wejściowych są często dość odległe od urządzeń cyfrowych. Sygnały wysyłane przez te źródła mogą zatem ulegać zniekształceniu wskutek zakłóceń. Nawet jeśli nadany sygnał ma właściwe parametry (amplituda, strome zbocza), to w miejscu jego odbioru może on wymagać regeneracji (tj. odtworzenia jego pierwotnego kształtu). ***Układy detekcji sygnałów użytecznych zawierają zwykle układy progowe (dyskryminatory amplitudy) oraz obwody całkujące o dość dużej stałej czasowej.*** Dyskryminator nie przepuszcza sygnałów, których amplituda nie osiągnie odpowiednio dobranego poziomu (progu), czyli filtruje zakłócenia o amplitudzie mniejszej niż ten próg. Układ całkujący natomiast tłumi sygnały krótkotrwałe (krótsze niż sygnał użyteczny).

Układy formowania i regeneracji sygnałów są najczęściej realizowane z wykorzystaniem przerzutnika Schmitta. Bramki logiczne zawierające taki przerzutnik omówiono w rozdz. 5. Dyskryminator może być zbudowany także ze standardowych bramek, jak na rys. 9.14a — zaletą tego układu jest możliwość blokady dyskryminatora przez podanie sygnału **0** na wejście **Bl**.



Rys. 9.14. Schematy: a) dyskryminatora (przerzutnika Schmitta) zbudowanego z bramek NAND; b) układu całkującego



Rys. 9.15. Przykładowe przebiegi czasowe: a) na wejściu układu całkującego; b) na wyjściu układu całkującego (przebieg wejściowy dyskryminatora); c) na wyjściu dyskryminatora
U_{p+} — wartość progowa narastającego napięcia wejściowego; U_{p-} — wartość progowa malejącego napięcia wejściowego

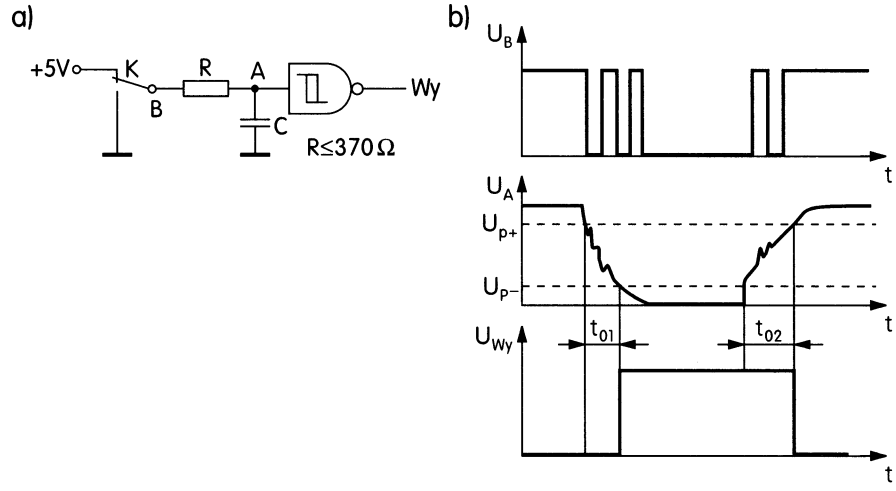
Przykładowe przebiegi czasowe w układzie zawierającym człon całkujący oraz dyskryminator amplitudy przedstawiono na rys. 9.15.

9.4.2. Układy współpracy z zestykami

Zestyki przekaźników, przełączników, wyłączników krańcowych itp. często są wykorzystywane jako źródła informacji wejściowej cyfrowych urządzeń automatyki. Wadą zestyków jest wytwarzanie drgań w czasie przełączania. W zależności od typu zestyków drgania te mogą trwać od kilku mikrosekund do kilku milisekund, zwykle od 5 μs ÷ 5 ms. Gdyby sygnał z przełącznika bezpośrednio doprowadzić do wejścia bramki, wówczas na jej wyjściu pojawiłby się przypadkowy ciąg zero-jedynkowy. W przypadku współdziałania zestyków z układami kombinacyjnymi na ogół drgania zestyków nie odgrywają istotnej roli. Natomiast przy współpracy z układami sekwencyjnymi należy zawsze zastosować układy, które wytłumią te drgania.

Prosty układ współdziałający z zestykiem przełącznym przedstawiono na rys. 9.16. Układ zawiera filtr dolnoprzepustowy (układ całkujący) oraz dyskryminator (bramkę z przerzutnikiem Schmitta). Stała czasowa *RC* układu powinna być tak dobrana, aby przełączenie bramki nastąpiło po zaniku drgań zestyku.

Jeden z najprostszych sposobów wyeliminowania drgań zestyków polega na użyciu przerzutnika asynchronicznego (patrz p. 7.2). Można także zastosować przerzutnik synchroniczny (scalony), wykorzystując go od strony wejść przygotowujących. Układy współpracy zestyków z przerzutnikami asynchronicznymi przedstawiono na rys. 9.17. Po zmianie położenia przełącznika następuje zmiana stanu przerzutnika i wówczas drgania zestyków nie mają już wpływu na działanie układu.

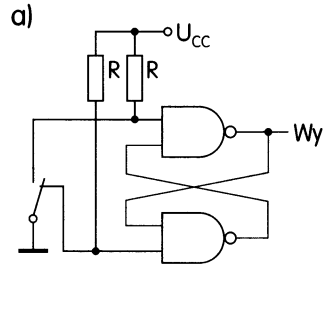


Rys. 9.16. Układ z zestykiem przełącznym: a) schemat zasadniczy; b) przebiegi czasowe

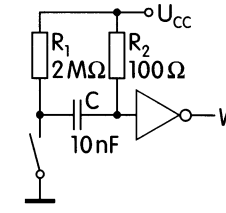
Jeżeli przełączniki są połączone długimi przewodami z wejściami bramek, to zakłócenia będą miały duży wpływ na pracę układu. Aby zwiększyć odporność na zakłócenia układów z rys. 9.17, należy bramki standardowe (użyte do budowy przerzutników) zastąpić bramkami z układem Schmitta oraz zastosować na ich wejściach filtry dolnoprzepustowe (jak na rys. 9.16).

W niektórych układach wymaga się, aby odpowiedź na przełączenie zestyku miała postać impulsu, a nie jedynie zmiany poziomu sygnału wyjściowego, jak w układach omówionych powyżej. Przykłady takich rozwiązań przedstawiono na rys. 9.18 i na rys. 9.19.

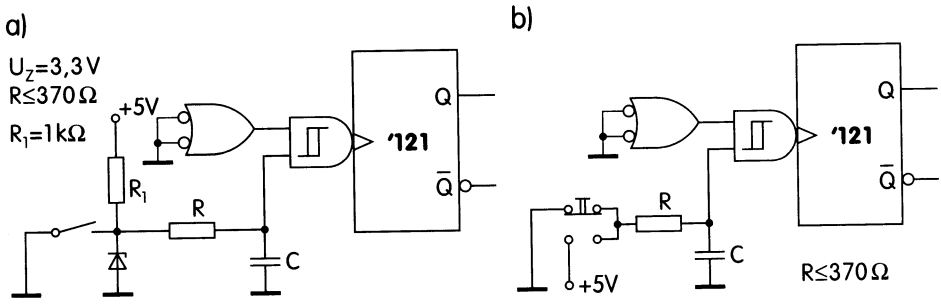
W układzie jak na rys. 9.18, gdy zestyk jest otwarty, wówczas na wyjściu bramki jest stan niski, a kondensator C jest rozładowany. Zamknięcie zestyku powoduje szybkie ładowanie kondensatora ze stałą czasową $R_2C = 1$ µs. Na wyjściu bramki pojawia się dodatni impuls o czasie trwania $300 \div 500$ ns (czas ładowania się kondensatora C do napięcia przełączania bramki). Drgania zestyku nie mają wpływu na działanie układu, gdyż kolejne wytworzenie impulsu może nastą-



Rys. 9.17. Układy współpracy z zestykiem przełącznym: a) z przerzutnikiem $\bar{r}\,\bar{s}$; b) z przerzutnikiem rs



Rys. 9.18. Układ z zestykiem zwiernym generujący impuls o czasie trwania $300 \div 500$ ns



Rys. 9.19. Układy z zestykami generujące impuls: a) z zestykiem zwiernym; b) z zestykiem przełącznym

pić dopiero po rozładowaniu kondensatora. A to wymaga otwarcia zestyku na czas dłuższy niż czas drgania zestyków, bowiem stała czasowa rozładowania kondensatora jest względnie duża i wynosi $(R_1 + R_2)C \approx 20$ ms.

W układach jak na rys. 9.19, zastosowano przerzutniki monostabilne **'121** (opisane w p. 8.2). Dzięki użyciu układów całkujących, których sygnał wyjściowy jest doprowadzany do wejścia bramki Schmitta, układy te poprawnie współpracują z zestykami umieszczonymi z dala od urządzenia.

9.4.3. Układy rozdzielenia galwanicznego

W wielu systemach elektronicznych, w tym także cyfrowych, istnieje potrzeba galwanicznego rozdzielenia poszczególnych części układu. ***Galwaniczna separacja układów oznacza brak metalicznego połączenia między nimi. Eliminuje ona zakłócenia powstające we wspólnych obwodach zasilania, umożliwiając zasilanie obu obwodów z odrębnych źródeł (brak sprzęgania się poprzez źródło).*** W przypadku gdy jeden z obwodów jest obwodem wysokonapięciowym, wówczas separacja galwaniczna zabezpiecza obwód niskonapięciowy przed przedostaniem się wysokiego napięcia i uszkodzeniem go.

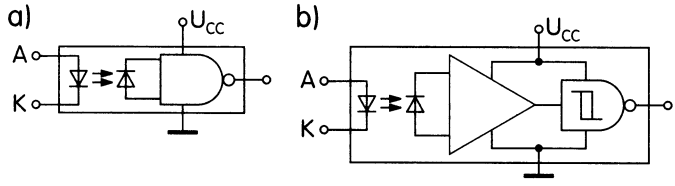
Elementami umożliwiającymi rozdzielenie galwaniczne obwodów są: **przekaźniki, transformatory** i, obecnie coraz częściej używane, optoizolatory zwane **transoptorami.**

Transoptorem *jest nazywany* układ scalony *zawierający sprzężony optycznie, a odizolowany galwanicznie układ. Składa się on* z diody elektroluminescencyjnej (LED), detektora promieniowania i ośrodka przewodzącego światło, *znajdującego się między tymi elementami. Elementem służącym jako* detektor promieniowania *może być:* fotorezystor, fotodioda, fototranzystor *lub* fototyrystor. *Rodzaj użytego detektora określa typ transoptora.*

Transoptory zapewniają doskonałą izolację wejścia od wyjścia układu, charakteryzują się małymi opóźnieniami sygnału i nadają się do budowy prostych układów współpracujących z elementami i układami półprzewodnikowymi.

Wprawdzie przekaźniki również zapewniają dobrą izolację wejścia od wyjścia, lecz ich czas odpowiedzi jest rzędu milisekund. Jako elementy elektromechaniczne są bardziej zawodne w działaniu. Transformatory natomiast mają ograniczone pasmo przenoszenia i nie przenoszą sygnałów wolnozmiennych i stałoprądowych.

Budowane są także transoptory z fototranzystorem pracującym w układzie Darlingtona, a także wyposażone w wewnętrzny wzmacniacz sygnału wytwarzanego przez detektor (np. fotodiodę) połączony z bramką zawierającą układ Schmitta (rys. 9.20).

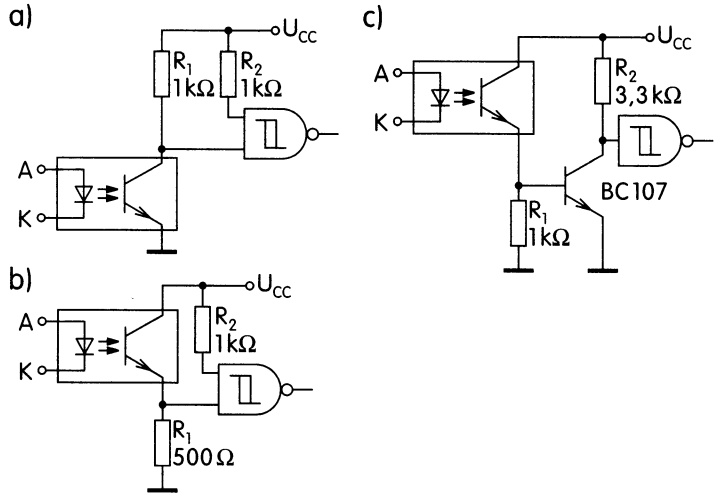


Rys. 9.20. Transoptor typu LED-bramka logiczna: a) schemat zasadniczy; b) schemat funkcjonalny

Transoptor przedstawiony na rys. 9.20 jest powszechnie stosowany w systemach cyfrowych. Jest on produkowany jako układ scalony np. MCL600, MCL601, MCL610, MCL611 — firmy Motorola.

Przykładowe układy wejściowe z transoptorem typu LED-fototranzystor przedstawiono na rys. 9.21.

Separacja galwaniczna jest stosowana także w obwodach wyjściowych systemów cyfrowych. Biorąc pod uwagę fakt, że elementem wejściowym każdego transoptora jest dioda LED, sterowanie transoptorem z wyjścia układu cyfrowego sprowadza się do sterowania diodą elektroluminescencyjną (patrz p. 9.5).



Rys. 9.21. Transoptor LED-fototranzystor z układami wyjściowymi: a) w układzie odwracającym fazę; b) w układzie wtórnika emiterowego; c) z dodatkowym (tranzystorowym) stopniem wzmacniającym

Pytania i zadania

1. Jakie zadania realizują układy wejściowe (czyli układy sprzęgające źródła sygnałów z wejściami systemów cyfrowych)?
2. Wyjaśnij rolę dyskryminatora i układu całkującego w detektorach sygnałów wejściowych.
3. Narysuj układ wejściowy z zestykiem przełącznym i przerzutnikiem asynchronicznym zbudowanym z bramek NAND. Określ poziomy logiczne na wyjściu przerzutnika przy różnych położeniach zestyku.
4. Wykonaj zadanie 3. dla przerzutnika zbudowanego z bramek NOR.
5. Czy układ stykowy z zadania 3. może współpracować z przerzutnikiem zbudowanym z bramek NOR? Odpowiedź uzasadnij.
6. Czy układ stykowy z zadania 4. może współpracować z przerzutnikiem zbudowanym z bramek NAND? Odpowiedź uzasadnij.
7. Narysuj przebiegi czasowe w układzie z rys. 9.18.
8. Narysuj przebiegi czasowe w układach z rys. 9.19.
9. Co to jest i jakie spełnia zadania izolacja galwaniczna obwodów?
10. Wymień elementy zapewniające separację galwaniczną. Omów ich wady i zalety.
11. Jak jest zbudowany transoptor? Jakie znasz typy transoptorów?

9.5. Układy wyjściowe

W układach cyfrowych informacja wejściowa jest przetwarzana i po przetworzeniu najczęściej służy do sterowania pewnych obiektów albo jest wizualizowana w postaci czytelnej dla człowieka.

Informacja wyjściowa układów cyfrowych zwykle jest obrazowana przy użyciu wskaźnika cyfrowego zbudowanego z diod świecących typu LED lub wskaźnika ciekłokrystalicznego. W tym drugim przypadku jest potrzebna znikomo mała moc sygnałów sterujących w porównaniu z mocą sygnałów wyjściowych TTL czy CMOS.

Moc sygnałów wyjściowych z elementów TTL czy CMOS jest niewielka i przeważnie niewystarczająca do właściwego wysterowania obwodów wyjściowych innych niż wskaźniki elektroluminescencyjne czy ciekłokrystaliczne. Z tego powodu elementami bezpośrednio sterowanymi z wyjść układów cyfrowych będą często wzmacniacze mocy (tranzystor, tyrystor, przekaźnik).

9.5.1. Sterowanie wskaźników elektroluminescencyjnych z wyjść układów TTL

Wskaźniki elektroluminescencyjne stosowane w układach cyfrowych to najczęściej pojedyncze diody świecące (LED) lub zespoły diod świecących, zgrupowane w postaci cyfrowych wskaźników siedmiosegmentowych. Wskaźnik siedmiosegmentowy zawiera 8 (tak, osiem) diod świecących, z których 7 tworzy układ umożliwiający wyświetlenie dowolnej cyfry dziesiętnej, natomiast ósma odpowiada za świecenie kropki dziesiętnej. Diody we wskaźniku mogą być połączone

ze sobą anodami (tzw. wskaźnik ze wspólną anodą) lub katodami (tzw. wskaźnik ze wspólną katodą). Matryca diodowa zawierająca np. po 5 diod w 7 wierszach (5x7) pozwala zobrażować nie tylko cyfry, ale i litery alfabetu. W związku z tym jest nazywana **wskaźnikiem alfanumerycznym.** Jednak podstawowym elementem zarówno wskaźnika siedmiosegmentowego, jak i wskaźnika alfanumerycznego — wymagającym odrębnego sterowania — jest **dioda świecąca typu LED.** Wystarczy zatem dobrze poznać zagadnienie sterowania diody LED, aby poradzić sobie z wysterowaniem pozostałych wskaźników elektroluminescencyjnych.

Jak wiadomo z podstaw elektroniki, dioda świecąca zachowuje się jak zwykła dioda prostownicza. Dodatkowym efektem jest jej świecenie wówczas, gdy jest spolaryzowana w kierunku przewodzenia. Diody świecące emitują promieniowanie w kolorach: czerwonym, pomarańczowym, żółtym i zielonym. Prąd znamionowy diody LED przeważnie zawiera się w przedziale $10 \div 20$ mA. Dioda będąca elementem wskaźnika siedmiosegmentowego lub alfanumerycznego potrzebuje zwykle 10 razy mniejszego prądu ($1 \div 2$ mA). Spadek napięcia na przewodzącej diodzie LED najczęściej ma wartość $1{,}5 \div 3$ V.

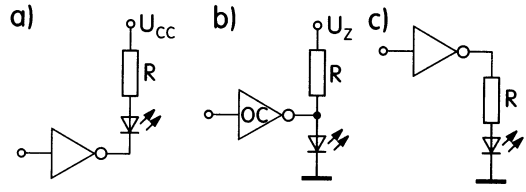
W przypadku sterowania diody LED ze źródła napięcia należy włączyć szeregowo z nią rezystor ograniczający prąd. Rezystor należy tak dobrać, aby prąd płynący przez diodę nie przekraczał wartości prądu przewodzenie diody I_F oraz dopuszczalnej wartości ($I_{OL\,max}$, $I_{OH\,max}$) prądu wyjściowego bramki sterującej. Spadek napięcia na rezystorze jest równy napięciu zasilania U_z pomniejszonemu o napięcie przewodzenia diody U_F. Z prawa Ohma wynika więc zależność na minimalną wartość rezystancji rezystora ograniczającego prąd diody.

$$R = \frac{U_z - U_F}{I_F} \tag{9.3}$$

Na przykład przy napięciu zasilania $U_z = U_{CC} = 5$ V, prądzie przewodzenia diody $I_F = 10$ mA i napięciu przewodzenia diody $U_F = 1{,}8$ V wartość rezystancji obliczona ze wzoru (9.3) wynosi 320 Ω.

Do sterowania diod świecących można użyć dowolnych bramek TTL z wyjściem przeciwsobnym lub z otwartym obwodem kolektora (OC). Sposoby sterowania diod świecących za pomocą układów TTL przedstawiono na rys. 9.22.

W układzie jak na rys. 9.22a świecenie diody uzyskuje się wówczas, gdy bramka sterująca jest w stanie niskim (niezależnie od tego, czy jest to bramka



Rys. 9.22. Sterowanie diod LED z bramek TTL: a) z wyjściem przeciwsobnym lub z otwartym obwodem kolektora; b) z otwartym obwodem kolektora; c) z wyjściem przeciwsobnym

z wyjściem przeciwsobnym czy typu OC). Maksymalna wartość prądu $I_{OL\,max}$ dla bramki standardowej z wyjściem przeciwsobnym ('00) czy typu OC ('01) wynosi 16 mA, bramki mocy z wyjściem przeciwsobnym ('37) wynosi 48 mA, a bramki mocy z otwartym obwodem kolektora ('07) wynosi 30 mA. Tak więc w układzie tym bramka standardowa bądź typu OC może wysterować dowolną diodę LED, której prąd przewodzenie jest nie większy niż 16 mA. Przy sterowaniu z bramki buforowej TTL-OC prąd ten nie powinien przekraczać 30 mA.

W układzie jak na rys. 9.22b dioda świeci wówczas, gdy bramka jest w stanie wysokim. Maksymalny prąd przewodzenia diody w tym układzie powinien być mniejszy niż 16 mA. Dokładną analizę tego układu pozostawia się Czytelnikowi. Proponuje się oszacować wartość maksymalną prądu przewodzenia diody, jaka może być wysterowana przez ten układ, jeżeli napięcie przewodzenia diody wynosi np. $U_F = 1{,}8$ V. Wskazane byłoby także porównanie mocy pobieranej ze źródła w obu układach, przy założeniu oczywiście sterowania identycznej diody w obu przypadkach. Uzasadnij, dlaczego z tego wyjścia co sterowana dioda nie można będzie sterować innego układu TTL. Wynik powyższej analizy będzie jednoznaczny — układ ten jest zdecydowanie gorszy od opisanego uprzednio i należy unikać jego stosowania.

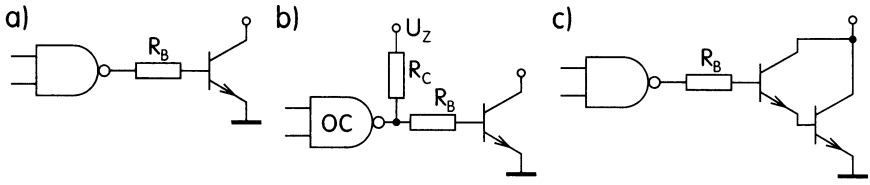
Także w układzie jak na rys. 9.22c świecenie diody uzyskuje się wówczas, gdy bramka jest w stanie wysokim. Jednak należy pamiętać, że wartość maksymalna prądu I_{OH} wynosi tylko 8 mA. Ogranicza to dość znacznie zakres stosowania tego sposobu do diod, których prąd przewodzenia nie przekracza wartości 8 mA. Także sterowanie wskaźników siedmiosegmentowych czy alfanumerycznych tym sposobem nie jest zalecane, mimo że w takim wskaźniku dioda pobiera jedynie prąd 1÷2 mA. Wynika to z faktu, że charakterystyki wyjściowe bramek TTL mają dość duży rozrzut i przy takich samych rezystorach ograniczających prąd, intensywność świecenia poszczególnych diod w segmencie może być zróżnicowana. Jest to uciążliwe dla użytkowników takich wskaźników. W celu wyeliminowania tej niedogodności należałoby dobierać rezystory indywidualnie do każdej diody (segmentu). Byłoby to w praktyce zbyt kłopotliwe. Każdorazowa wymiana układu scalonego, zawierającego bramki sterujące segmentami, wymagałaby ponownego doboru rezystorów.

Z analizy przedstawionych układów wynika więc, ***że najwłaściwszym sposobem sterowania diody LED z bramki TTL jest układ przedstawiony na rys. 9.22a.***

9.5.2. Współpraca układów TTL (CMOS) z tranzystorem

Typowe układy TTL i CMOS nie są przystosowane do sterowania odbiorników dużej mocy (rzędu kilku watów lub większej) — o dużym prądzie (rzędu kilku amperów) lub napięciu (większym niż kilkadziesiąt woltów). Układy buforowe umożliwiają wprawdzie wysterowanie niektórych odbiorników (bufory z wyjściem typu OC, OD), ale nie takich, jak: grzałka, silnik, żarówka itp. W takich przypadkach elementem pośredniczącym między wyjściem układu cyfrowego a odbiornikiem jest wzmacniacz tranzystorowy.

Układy sterowania tranzystora mocy (o prądzie rzędu kilku amperów) z bramki z wyjściem przeciwsobnym przedstawiono na rys. 9.23a. Tranzystor przewodzi wówczas, gdy bramka sterująca jest w stanie wysokim. Rezystor R umieszczony w obwodzie bazy tranzystora pozwala ustalić prąd bazy na pożądanej wartości. Dobiera się go w taki sposób, aby prąd bazy I_B był nie więcej niż β razy mniejszy od prądu kolektora tranzystora I_C (β jest wzmocnieniem prądowym tranzystora).



Rys. 9.23. Sterowanie tranzystora z bramki TTL (CMOS): a) bramka z wyjściem przeciwsobnym; b) bramka typu OC (OD); c) sterowanie wzmacniacza tranzystorowego w układzie Darlingtona

Od wartości tego prądu zależy, czy tranzystor znajdzie się w obszarze pracy aktywnej czy w obszarze nasycenia. Większy prąd bazy (ale nie większy niż dopuszczalny prąd bazy) to praca w obszarze nasycenia i mniejsze straty mocy w układzie. Maksymalną wartość prąd ten osiąga przy zerowej wartości rezystancji — bramka sterująca jest wówczas w stanie zwarcia. Prąd osiąga wartość nieco mniejszą niż prąd zwarciowy I_{OS}, ponieważ nie jest to stan idealnego zwarcia (na wyjściu jest napięcie przewodzenia złącza baza-emiter). ***Należy jednak pamiętać, że stan zwarcia jest dopuszczalny tylko na jednym z wyjść układu scalonego — w przeciwnym przypadku grozi zniszczenie układu.***

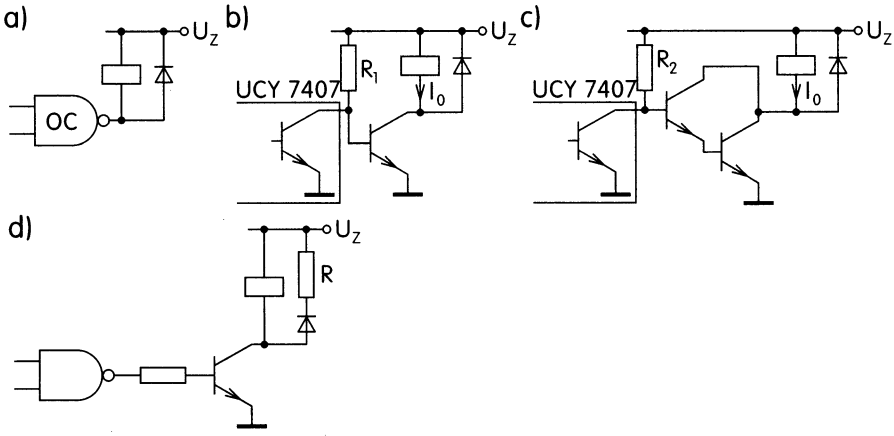
W układzie jak na rys. 9.23b, w którym zastosowano bramkę z otwartym kolektorem, można uzyskać większe prądy bazy, co pozwala na wysterowanie odbiorników o większych mocach niż w układzie jak na rys. 9.23a.

Znaczne prądy wyjściowe można także uzyskać, stosując układ Darlingtona jak na rys. 9.23c.

9.5.3. Współpraca układów TTL (CMOS) z przekaźnikiem

Bardzo często elementem sterowanym z wyjść układów cyfrowych jest przekaźnik. Zwarcie jego zestyków roboczych powoduje bezpośrednie załączenie odbiornika mocy lub np. stycznika sterującego odbiornikami dużej mocy. Podstawową zaletą przekaźnika jest **separacja galwaniczna** (brak połączenia metalicznego) obwodu sterującego od obwodu sterowanego. Inną coraz częściej stosowaną metodą oddzielenia galwanicznego obwodów jest stosowanie elementów optoelektronicznych (jak np. transoptor).

Niewielkie przekaźniki (o niedużym napięciu i prądzie cewki) można wysterować bezpośrednio, przy użyciu bufora typu OC (rys. 9.24a). Przekaźniki o większej mocy obwodu sterującego będą załączane za pośrednictwem dodatkowego tranzystora rys. 9.24b.

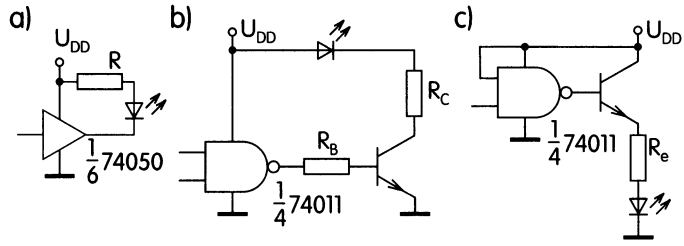


Rys. 9.24. Współpraca bramki TTL z przekaźnikiem: a), b), c) bramka typu OC; d) bramka z wyjściem przeciwsobnym sterująca wzmacniaczem tranzystorowym

Równolegle z cewką przekaźnika należy zawsze włączyć diodę, której zadaniem jest likwidowanie przepięć powstających w chwili wyłączania prądu cewki. W polu magnetycznym cewki jest zmagazynowana energia o wartości $LI^2/2$ (gdzie L — indukcyjność cewki, I — prąd cewki). W chwili wyłączenia prądu cewki indukuje się sem (siła elektromotoryczna), usiłująca podtrzymać przepływ prądu. Przy braku diody przeciwprzepięciowej ta sem (plus energia zgromadzona w polu magnetycznym) może być przyczyną uszkodzenia elementów załączających prąd cewki, a w najlepszym przypadku — źródłem zakłóceń działania układu cyfrowego. Opisane powyżej zjawisko może być pożyteczne, jak np. w układzie zapłonowym samochodu. Jednak w układach cyfrowych należy je zawsze wyeliminować. Zadanie to doskonale spełnia dioda, włączona tak jak na rys. 9.24. W czasie normalnej pracy, gdy przez cewkę przekaźnika płynie prąd, dioda jest spolaryzowana zaporowo i nie ma żadnego wpływu na pracę układu. W chwili wyłączania prądu cewki napięcie indukujące się w niej ma zwrot zgodny z prądem (usiłując podtrzymać przepływ prądu), czyli polaryzuje diodę w kierunku przewodzenia. Napięcie to nie może teraz osiągnąć dużych wartości, gdyż dioda ogranicza je do napięcia przewodzenie $U_F = 0{,}6 \div 0{,}7$ V. Energia zgromadzona w polu magnetycznym rozładowuje się w obwodzie cewka-dioda, nie powodując dzięki temu negatywnych zjawisk.

9.5.4. Sprzężenie układów CMOS z elementami sygnalizacyjnymi

Prąd potrzebny do rozświecenia elementów sygnalizacyjnych (diody LED, wyświetlacze LED, żarówki) na ogół przekracza wartość 1 mA. Układy CMOS serii 4000B charakteryzują się bardzo małym prądem wyjściowym, który dla typowych układów wynosi ok. 1 mA. Aby więc wysterować bezpośrednio z wyjścia takiego układu diodę LED, należy użyć bufora, np. MCY74050. Schemat takiego układu przedstawiono na rys. 9.25a. Dioda świeci wówczas, gdy bramka sterująca jest w stanie niskim. Rezystor R ogranicza prąd diody do wartości prądu przewo-



Rys. 9.25. Współpraca układów CMOS z diodami LED: a) sterowanie diodą LED z bufora MCY74050; b) sterowanie diodą LED za pomocą wzmacniacza tranzystorowego; c) sterowanie diodą LED za pomocą wtórnika emiterowego

dzenia I_F. Sposób doboru jego wartości jest identyczny, jak omówiono wcześniej (zależność (9.3)). Zamiast bufora można (bez zmiany konfiguracji układu) zastosować układ z otwartym drenem.

Na rysunku 9.25b przedstawiono sposób sterowania diodą LED za pomocą wzmacniacza tranzystorowego. Układ taki można wykorzystać do sterowania wskaźników LED (np. siedmiosegmentowych) o wspólnej anodzie. Wyświetlacz o wspólnej katodzie może być sterowany w układzie jak na rys. 9.25c. Oczywiście każdy segment takiego wskaźnika wymaga odrębnego układu sterującego.

9.5.5. Przekaźniki półprzewodnikowe

W przekaźnikach półprzewodnikowych elementem przełączającym jest: tranzystor bipolarny, tranzystor MOS, tyrystor lub triak. W obwodzie wejściowym przekaźnika znajduje się dioda świecąca. ***Sprzężenie obu obwodów jest zatem realizowane za pośrednictwem światła, dzięki czemu przekaźniki te zapewniają jednocześnie separację galwaniczną pomiędzy obwodem sterującym a obwodem sterowanym.*** Spełniają one takie same funkcje jak przekaźnik elektromechaniczny, ale są pozbawione wielu jego wad.

Do sterowania przekaźników półprzewodnikowych jest wymagany sygnał prądowy od 2 do kilkudziesięciu miliamperów, w zależności od typu przekaźnika. Pozwala to na ich bezpośrednie sterowanie z układów CMOS lub TTL. Sposób sterowania przekaźników półprzewodnikowych jest więc identyczny jak diod typu LED. Poniżej omówiono podstawowe zalety i wady tych przekaźników oraz ich parametry, co ułatwi określenie obszarów ich zastosowań.

Przekaźniki półprzewodnikowe można podzielić na kilka grup [2]:

1. **Przekaźniki małosygnałowe** — do przełączania sygnałów stało- i zmienno-prądowych lub tylko stałoprądowych, w których elementem przełączającym jest tranzystor MOSFET lub bipolarny.
2. **Przekaźniki średniej mocy** — do przełączania tylko prądu przemiennego lub tylko stałego o wartości mniejszej niż 3 A. Elementem przełączającym jest w nich tyrystor, triak lub tranzystor mocy MOSFET.
3. **Przekaźniki dużej mocy** — do przełączania tylko prądu przemiennego do 50 A — z tyrystorem lub triakiem jako elementem przełączającym.

Dioda wejściowa w tych przekaźnikach steruje albo bezpośrednio elementem wykonawczym, albo innym elementem światłoczułym, a ten dopiero elementem wykonawczym. W pierwszym przypadku elementami przełączającymi będą fototranzystory lub fototyrystory.

Przekaźniki półprzewodnikowe są zamknięte w hermetycznych obudowach: małej mocy — w obudowach np. typu DIL, czyli identycznych jak układy scalone TTL lub CMOS; dużej mocy — w obudowach przystosowanych do zamocowania na radiatorze; bardzo dużych mocy — w obudowach z wyprowadzeniami dostosowanymi do połączeń płaskich nasadowych lub śrubowych.

Szybkość działania przekaźników półprzewodnikowych jest dużo większa niż przekaźników elektromechanicznych. Przekaźniki elektromechaniczne z „zestykiem zwiernym" mają czas włączania ok. 1 ms, a czas wyłączania ok. 0,5 ms. W przekaźnikach elektromechanicznych z „zestykiem rozwiernym" wartości tych czasów kształtują się odwrotnie. Szybkie przekaźniki półprzewodnikowe mają czas włączania rzędu kilkudziesięciu mikrosekund, ale za to większe prądy sterujące i większą rezystancję przewodzenia, która może osiągać wartość kilkuset omów, podczas gdy dla wolniejszych wynosi ona kilkadziesiąt miliomów (dla połączeń stałoprądowych).

Przekaźniki półprzewodnikowe zwykle mają tylko jeden „zestyk". Produkowane są jednak i takie, które w jednej obudowie mają dwa zestyki sterowane niezależnymi diodami lub sterowane tą samą diodą.

Przekaźniki półprzewodnikowe do przełączania obwodów stało- i zmiennoprądowych mają następujące **z a l e t y**:

— bardzo dużą liczbę zadziałań (w związku z brakiem elementów ruchomych), która osiąga wartość 15 bilionów ($15 \cdot 10^9$); warto zauważyć, że zadziałania te są bezgłośne;
— brak odbić i zawieszeń „zestyków";
— bardzo dużą niezawodność działania;
— dużą szybkość działania;
— stałość rezystancji przejścia;
— niewrażliwość na zewnętrzne zakłócenia elektryczne;
— odporność na wstrząsy, wibracje i wpływy czynników zewnętrznych;
— małe wymiary i niewielka masa;
— proste układy sterowania;
— małą moc sterowania.

Przekaźniki półprzewodnikowe mają też pewne **w a d y**, a należą do nich:

— duża rezystancja przejścia w chwili przewodzenia w porównaniu z przekaźnikami mechanicznymi oraz jej zależność od temperatury;
— mała moc przełączana;
— mała liczba zestyków;
— prądy upływu, tzn. przewodzenie niewielkich prądów w stanie wyłączenia.

Przekaźniki półprzewodnikowe mogą mieć także inne zalety, nie wymienione powyżej, ponieważ dotyczą one wykonań specjalnych. Należy do nich odporność na zwarcia, przy czym tę zaletę mogą mieć tylko przekaźniki stałoprądowe. Przekaźniki zmiennoprądowe mogą natomiast być wyposażone w układy synchroniza-

cji, sprawiające, że załączanie obwodów zmiennoprądowych jest dokonywane w chwili przejścia ***sinusoidy napięcia*** przez zero (w praktyce przy napięciu ok. 4 V) i ich wyłączanie w chwili, gdy ***prąd*** osiągnie wartość bliską zera. Dzięki takiemu załączaniu (i wyłączaniu) obciążeń o charakterze indukcyjnym nie występują stany przejściowe i związane z nimi przepięcia, co mogłoby być przyczyną uszkodzeń lub zakłóceń. Więcej informacji na temat przekaźników półprzewodnikowych znajdzie Czytelnik w publikacji [2].

Pytania i zadania

1. Oblicz wartość rezystora ograniczającego prąd diody świecącej, zasilanej ze źródła napięcia $U = 10\,V$. Prąd przewodzenia diody $I_F = 15\,mA$, a jej napięcie przewodzenia $U_F = 1{,}2\,V$.
2. Dobierz wartość rezystora R dla diody LED pracującej w układzie jak na rys. 9.22a (bramka '00). Prąd przewodzenia diody $I_F = 15\,mA$, a jej napięcie przewodzenia $U_F = 2{,}2\,V$.
3. Oblicz energię pobraną przez rezystor R w czasie 1 min, gdy łączny czas świecenia diody wyniósł 30 s. Obliczenia wykonaj dla układów jak na rys. 9.22a i 9.22b. Parametry diody przyjmij takie, jak w zad. 2. Porównaj otrzymane wyniki i skomentuj je.
4. Oblicz maksymalny prąd, jaki może płynąć przez diodę w układzie na rys. 9.22b (układ sterujący '07), jeżeli napięcie przewodzenia diody $U_F = 2{,}4\,V$, a dopuszczalny prąd $I_{OL\,max} = 30\,mA$. Wykonaj obliczenia dla napięcia $U_z = 10\,V$ oraz $U_z = 5\,V$.
5. Oblicz rezystancję R w układzie jak na rys. 9.22b dla diody, której prąd przewodzenia $I_F = 20\,mA$, a napięcie przewodzenia $U_F = 1{,}8\,V$. Obliczenia wykonaj dla dwóch napięć: $U_z = 10\,V$ oraz $U_z = 5\,V$. Oblicz, jaki prąd popłynie przez rezystor R, gdy dioda nie będzie świeciła (układ sterujący '07).
6. Oblicz wartość rezystora R_B w układzie jak na rys. 9.23a, jeżeli prąd kolektora tranzystora powinien mieć wartość $I_C = 0{,}5\,A$, a jego wzmocnienie prądowe wynosi $\beta = 100$.
7. Dla danych z zadania 6. oblicz wartość rezystora R_B w układzie jak na rys. 9.23b.
8. Dobierz rezystor R_B w układzie jak na rys. 9.23c, jeżeli załączany prąd ma mieć wartość 5A, a wzmocnienia prądowe tranzystorów wynoszą odpowiednio $\beta_1 = 50$ i $\beta_2 = 100$.
9. Narysuj schemat sterowania przekaźnika. Uzasadnij potrzebę włączenia diody równolegle z cewką przekaźnika.
10. Dany jest przekaźnik o napięciu znamionowym $U = 12\,V$ i prądzie 100 mA (obwodu sterowania) oraz tranzystor o wzmocnieniu prądowym $\beta = 100$. Dobierz rezystor do sterowania tego przekaźnika w układzie jak na rys. 9.24b. Czy można ten przekaźnik wysterować w układzie jak na rys. 9.24a?
11. Oblicz rezystancję R w układzie jak na rys. 9.25a dla diody, której prąd przewodzenia $I_F = 5\,mA$, a napięcie przewodzenia $U_F = 1{,}4\,V$. Obliczenia wykonaj dla dwóch napięć: $U_z = 10\,V$ oraz $U_z = 5\,V$ ($U_z = U_{DD} - U_{SS}$).
12. Oblicz wartości rezystorów R_B, R_C w układzie jak na rys. 9.25b, jeżeli prąd przewodzenia diody $I_F = 20\,mA$, a jej napięcie przewodzenia $U_F = 2{,}2\,V$. Wzmocnienie prądowe tranzystora wynosi $\beta = 50$. Obliczenia wykonaj dla dwóch napięć: $U_z = 10\,V$ oraz $U_z = 5\,V$ ($U_z = U_{DD} - U_{SS}$).
13. Oblicz wartość rezystora R_E w układzie jak na rys. 9.25c, jeżeli prąd przewodzenia diody $I_F = 16\,mA$, a jej napięcie przewodzenia $U_F = 2{,}5\,V$. Obliczenia wykonaj dla dwóch napięć: $U_z = 10\,V$ oraz $U_z = 5\,V$ ($U_z = U_{DD} - U_{SS}$). Dlaczego układu takiego nie stosuje się dla elementów TTL?

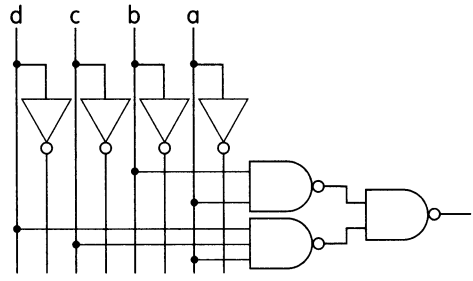
9.6. Minimalizacja liczby układów scalonych

Rysując schematy logiczne układów kombinacyjnych, nie analizowaliśmy do tej pory powiązania liczby bramek z liczbą układów scalonych. Wymogi optymalizacji układów pod względem liczby połączeń (niezawodność) oraz liczby użytych układów (koszty) wymagają od projektanta uwzględnienia ograniczeń, związanych z liczbą bramek wchodzących w skład jednego układu scalonego.

Przeanalizujmy prosty układ kombinacyjny, opisany następującym równaniem:

$$y = ab + acd \tag{9.4}$$

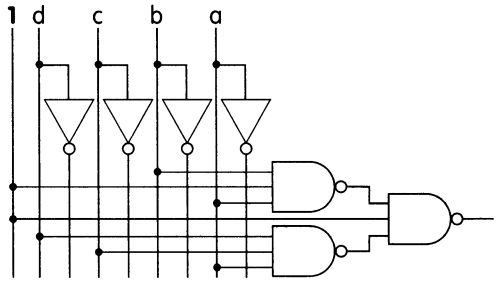
Schemat tego układu, zbudowanego z bramek NAND, przedstawiono na rys. 9.26. Zawiera on dwie bramki dwuwejściowe i jedną trójwejściową.



Rys. 9.26. Schemat logiczny układu opisanego równaniem (9.4)

Wybranie do realizacji dwóch układów scalonych, np.:
— UCY7400 — zawierającego 4 bramki dwuwejściowe NAND,
— UCY7410 — zawierającego 3 bramki trójwejściowe NAND,
jest postępowaniem nieoptymalnym. Wystarczy bowiem zmodyfikować układ z rys. 9.26, aby zaistniała możliwość zbudowania powyższego układu z wykorzystaniem tylko jednego układu scalonego UCY7410, zawierającego trzy bramki trójwejściowe (rys. 9.27).

W praktyce analiza projektowanych układów pod względem minimalizacji liczby układów scalonych może być bardziej złożona. Warto ją jednak przepro-



Rys. 9.27. Schemat logiczny układu opisanego równaniem (9.4) zbudowanego wyłącznie z bramek trójwejściowych

wadzić, bo uzyskiwane efekty to: niższy koszt układu, większa niezawodność i mniejsze wymiary.

Z poruszonym powyżej problemem wiążą się dwa nowe, nie omówione do tej pory, zagadnienia:

— przekształcanie schematów logicznych,
— zasady postępowania z niewykorzystywanymi wejściami scalonych układów cyfrowych.

- **Przekształcanie schematów logicznych**

Przekształcanie schematów układów logicznych może polegać na zamianie funktorów realizujących układ — zarówno ze względu na liczbę ich wejść, jak i realizowaną funkcję. Na przykład zamiana funktorów NAND na NOR lub na funktory AND, OR, NOT. Zamiana funktorów NAND o różnych liczbach wejść na funktory NAND wyłącznie dwuwejściowe itp.

Sposobów postępowania może być wiele i wszystkie trudno byłoby opisać. Jedynym (i wystarczającym) kryterium poprawności poczynionych przekształceń jest niezmienność realizowanej funkcji logicznej przez dany układ. Poniżej zostanie opisana jedna z metod przekształcania schematów logicznych.

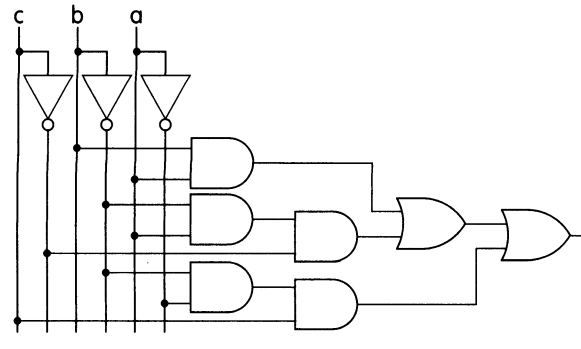
Przykład 9.1

Narysować schemat logiczny układu, przy założeniu dostępności jedynie dwuwejściowych bramek NAND. Układ jest opisany równaniem (alternatywna postać minimalna)

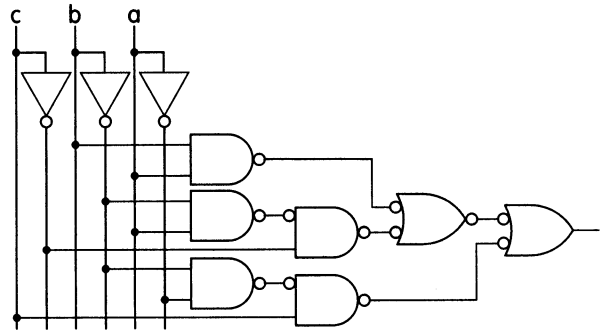
$$y = ab + a\bar{b}\bar{c} + \bar{a}\bar{b}c \tag{9.5}$$

Pierwszym krokiem jest narysowanie schematu tego układu przy wykorzystaniu bramek dwuwejściowych AND, OR i NOT (NOT — oczywiście jednowejściowe). Można to uczynić bezpośrednio na podstawie wyrażenia (9.5) lub przekształcając je do następującej postaci:

$$y = ab + a\bar{b}\bar{c} + \bar{a}\bar{b}c = ((ab + (a\bar{b})\bar{c})) + (\bar{a}\bar{b})c \tag{9.6}$$



Rys. 9.28. Schemat układu opisanego równaniem (9.6)

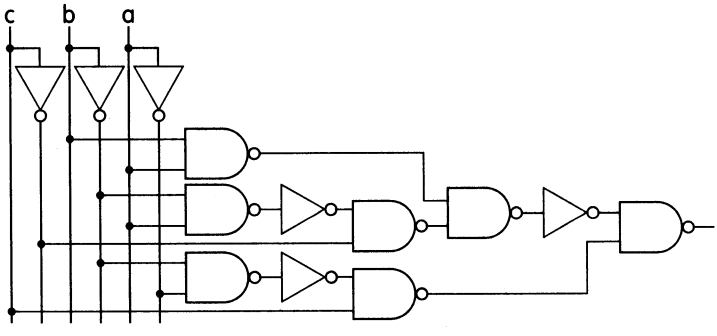


Rys. 9.29. Zmodyfikowany schemat z rys. 9.28

W zapisie (9.6) wszystkie operacje (sumy i iloczynu logicznego) są operacjami dwuargumentowymi, co oznacza, że mogą być zrealizowane przy użyciu funktorów dwuwejściowych. Strukturę tego układu przedstawiono na rys. 9.28.

Ponieważ podwójna negacja nie zmienia wartości logicznej sygnału, więc możemy w dowolnym miejscu w torze dowolnego sygnału umieścić parę negatorów. Na schemacie zaznaczać to będziemy kółeczkami, umieszczanymi na wyjściu bądź na wejściu bramki. Tak zmodyfikowany schemat przedstawiono na rys. 9.29.

Jak łatwo zauważyć, niektóre bramki na rys. 9.29 to już dwuwejściowe bramki NAND. Przyjrzyjmy się bliżej bramkom OR z negacjami sygnałów wejściowych. Przypomnijmy sobie drugi sposób rysowania bramki NAND (p. 3.2, rys. 3.7). Te spostrzeżenia pozwalają natychmiast przekształcić nasz schemat do wymaganej postaci (z dwuwejściowych bramek NAND), co przedstawiono na rys. 9.30. Układ na rysunku 9.30 zawiera także negatory, ale można je przecież zrealizować z bramek NAND (z czego dla większej przejrzystości rysunku, tutaj zrezygnowano).



Rys. 9.30. Rozwiązanie zadania 9.1 ■

- **Wejścia i bramki nie wykorzystane**

Konsekwencją minimalizacji liczby układów scalonych jest niekiedy użycie bramki o większej niż potrzeba liczbie wejść. Bywa też, że potrzebujemy użyć bramki o pięciu wejściach, a takie nie są produkowane. Użycie układu scalonego zawierającego np. 4 bramki, gdy są potrzebne tylko 2 sprawia, że pozostają w układzie jeszcze dwie bramki nie używane. Wreszcie korzystanie z bloków funkcjonalnych (układów scalonych o średnim i wysokim stopniu scalenia, realizujących złożone operacje) wiąże się z pozostawieniem pewnych wejść jako nie wykorzystanych. Poniżej podano zasady postępowania w takich właśnie sytuacjach.

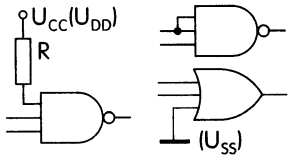
Z analizy obwodów wejściowych układów TTL wynika, że układ z wejściem nie podłączonym zachowuje się tak, jak gdyby to wejście było podłączone do źródła sygnału o poziomie wysokim **H**. Potocznie używa się określenia, że „wejście w powietrzu to jedynka logiczna". Układ taki będzie więc działał poprawnie, jeżeli na takim wejściu powinien być właśnie poziom wysoki. Sytuacja taka wystąpi np. w przypadku pozostawienia nie podłączonego wejścia bramki NAND. Pozostawienie jednak nie wykorzystanego wejścia bramki OR (czy NOR) „w powietrzu" sprawi, że bramka ta zostanie zablokowana (wejścia podłączone nie będą miały wpływu na jej stan i na wyjściu będzie cały czas poziom **H** (**L**)).

Inaczej to wygląda w przypadku bramek CMOS. Bardzo duża rezystancja wejściowa tych układów sprawia, że na wejściu pozostawionym w powietrzu ustala się napięcie, którego wartość jest wyższa niż wartość przyjęta dla poziomu **L**, a niższa niż poziom **H**. Zwykle jest ono bliskie napięciu przełączania U_p. Sprawia to, że bramka się wzbudza, pobiera duży prąd, a poziom logiczny wyjścia jest nieokreślony.

Jednak w żadnym przypadku nie należy pozostawiać wejść układów TTL, a szczególnie CMOS nie podłączonych!

Powody wyżej sformułowanej zasady postępowania są bowiem następujące:

1. Z powodu bardzo dużej impedancji wejściowej (dotyczy to zwłaszcza układów CMOS) poziom napięcia na wolnym wejściu nie jest stały. Jeżeli napięcie to jest bliskie napięcia przełączania, to układ pobiera znacznie większy prąd ze źródła zasilania, co może doprowadzić do przegrzania układu (patrz rys. 6.2). Grozi wzbudzeniem się bramki, impulsowym poborem prądu i w konsekwencji sprzęganiem się układów poprzez źródło.
2. Pozostawienie nie wykorzystanego wejścia „w powietrzu" powoduje zwiększenie czasu propagacji — układ działa wolniej. (Na przykład w układach TTL czas ten ulega wydłużeniu o ok. 1 ns na każde nie podłączone wejście.)
3. Zmniejsza się odporność takiego układu na zakłócenia. (Nie podłączone wejście zachowuje się tak, jak antena odbierająca impulsy zakłócające.)



Rys. 9.31. Podłączanie nie wykorzystanych wejść

Na rysunku 9.31 przedstawiono przykładowe schematy połączeń nie wykorzystanych wejść bramek TTL czy CMOS.

W przypadku układów TTL, jeżeli istnieje ryzyko, że napięcie U_{CC} przekroczy wartość 5,5V, to podłączenie nie wykorzystanego wejścia do źródła napięcia U_{CC} należy wykonać poprzez rezystor o wartości $1 \div 5k\Omega$.

Taki rezystor może być wspólny nawet dla 25 wejść. Uwaga ta odnosi się do tych serii, w których na wejściu znajduje się tranzystor (wieloemiterowy). Tam gdzie został on zastąpiony diodami, można takie wejścia łączyć bezpośrednio do U_{CC}. Podłączenie do masy można wykonać poprzez rezystor o wartości nie większej niż 400 Ω.

Na zakończenie warto jeszcze wspomnieć, że połączenie wejść powoduje zmniejszenie czasu propagacji bramki TTL, co jest niewątpliwą zaletą tego sposobu. Jednak sposób taki nie jest zalecany do bramek sumacyjnych TTL (OR, NOR), gdyż powoduje zwiększenie obciążenia bramki sterującej.

W układach CMOS łączenie wejść wydłuża czas propagacji bramki sterującej, bowiem każde dodatkowe wejście to dodatkowe obciążenie (pojemnościowe). Jeżeli więc zależy nam na szybkości działania, to należy unikać takich połączeń.

Wyjścia nie używanych bramek logicznych (TTL) powinno się ustawiać w stan wysoki. Osiągamy to w bramkach NAND czy NOR poprzez dołączenie ich wejść do masy. Takie postępowanie ma na celu zmniejszenie prądu zasilania (patrz charakterystyka poboru prądu) oraz umożliwienie wykorzystania ich wyjść jako źródeł sygnałów o poziomie logicznym **1**.

Wejścia nie używanych bramek CMOS należy koniecznie podłączyć do jednego z biegunów źródła napięcia zasilania (dowolnego).

Pytania i zadania

1. Zrealizuj z bramek:
   a) NAND,
   b) NOR,

   układ opisany funkcją: **f(d,c,b,a)** = Σ[0, 2, 6, 7, 8, 9, 10, 15, (1)]. Jakich i ile układów scalonych trzeba użyć do jego realizacji?
2. Zrealizuj z bramek
   a) NAND,
   b) NOR,

   układ opisany funkcją: **f(d,c,b,a)** = Π[2,3,9,11,12,13,14,(10)]. Jakich i ile układów scalonych trzeba użyć do jego realizacji?
3. Znajdź alternatywną postać minimalną funkcji **f(e,d,c,b,a)** = Σ[1,2,3,10,11,15,18, 21,23,27,31,(5,7,25,26,29)]. Narysuj schemat układu, wykorzystując bramki NAND. Jakich i ile układów scalonych trzeba użyć do realizacji tego układu?
4. Znajdź alternatywną postać minimalną funkcji **f(e,d,c,b,a)** = Π[0,3,6,8,11,14,16,22, 24,(1,7,15,17,30)]. Narysuj schemat układu, wykorzystując bramki NAND. Jakich i ile układów scalonych trzeba użyć do realizacji tego układu?
5. Zrealizuj zadanie 1. wyłącznie przy użyciu bramek dwuwejściowych.
6. Zrealizuj zadanie 2. wyłącznie przy użyciu bramek dwuwejściowych.
7. Zrealizuj zadanie 3. wyłącznie przy użyciu bramek dwuwejściowych.
8. Zrealizuj zadanie 4. wyłącznie przy użyciu bramek dwuwejściowych.
9. Dlaczego nie powinno się pozostawiać nie podłączonych wejść układów scalonych?
10. Jak będzie się zachowywać bramka trójwejściowa NAND, jeżeli do jej dwóch wejść doprowadzono sygnały **a** i **b**, natomiast trzecie wejście pozostawiono „w powietrzu". Rozważ dwa przypadki:
    a) bramka TTL,
    b) bramka CMOS.

11. Jak będzie się zachowywać bramka trójwejściowa NOR, jeżeli do jej dwóch wejść doprowadzono sygnały **a** i **b**, natomiast trzecie wejście pozostawiono „w powietrzu". Rozważ dwa przypadki:
    a) bramka TTL,
    b) bramka CMOS.
12. Podaj sposoby postępowania z nie wykorzystanymi wejściami układów TTL i CMOS.
13. Dlaczego korzystniej jest doprowadzić do wejść bramek nie wykorzystanych sygnały o takich poziomach, które ustawią je w stan wysoki, a nie niski?

9.7. Układy transmisji sygnałów cyfrowych

Jeszcze niedawno zagadnienie przesyłania (transmisji) sygnałów cyfrowych nie nastręczało tak wielu problemów technicznych jak obecnie. Czas przesyłania sygnałów między układami był pomijalnie mały w porównaniu do czasu potrzebnego na zmianę sygnału (tzw. **czas narastania, czas opadania** — ogólnie zwane **czasem przełączania**). Obecnie czas przełączania może być krótszy niż czas propagacji sygnału przez przewody. Przykładowe opóźnienia wprowadzane przez typowe rodzaje połączeń zestawiono w tabl. 9.3.

Typowe układy przesyłania sygnałów cyfrowych to:
— przesyłanie jednoprzewodowe (przewody nie ekranowane — np. ścieżki drukowane, przewody izolowane);
— przesyłanie skręconą parą przewodów (tzw. skrętką — przewody ekranowane częściowo);
— przesyłanie kablem koncentrycznym (przewody ekranowane).

Tablica 9.3. **Parametry linii transmisyjnych**

Rodzaj połączenia	Impedancja falowa, Ω	Opóźnienie ns/m
Dwa przewody 0,38 mm (tzw. skrętka)	110	6,2
Obwód drukowany	100	6
Kabel koncentryczny 75 Ω	75	5
Kabel koncentryczny 50 Ω	50	3,9
Linia paskowa	170	5,6

Z tablicy 9.3 wynika, że prędkość propagacji sygnałów w przewodach (odwrotność opóźnienia) zawiera się w przedziale 16 ÷ 25 cm/ns.

Połączenie *jest nazywane* długim, *gdy czas przejścia sygnału tam i z powrotem jest dłuższy niż czas narastania lub opadania (niż krótszy z tych czasów) w stosowanych elementach.*

Dla układów cyfrowych TTL serii 74 czas trwania zbocza opadającego wynosi ok. $t_{THL} = 5$ ns, a czas trwania zbocza narastającego wynosi $t_{TLH} = 11$ ns. W takim

przypadku połączenie o długości większej niż 42,5 cm należy uważać za długie (obwód drukowany). Wynika to z następujących obliczeń: Prędkość 17 cm/ns (100 cm/6ns) pomnożona przez 5 ns daje w wyniku 17 cm/ns · 5 ns = 85 cm. Odległość to połowa drogi (droga liczona jest tam i z powrotem), czyli 85 cm/2 = 42,5 cm. Dla kabla 50-omowego krytyczna długość połączenia wynosi ok. 64 cm. Wartości te mogą być inne w różnych książkach, gdyż autorzy przyjmują różne kryteria podziału linii na długie i krótkie, np. publikacje [21; 8].

W elektrycznie długich przewodach występują natomiast oscylacje. Oscylacje powstają wskutek odbić od końców przewodu z uwagi na brak dopasowania między impedancją falową linii a rezystancjami: wejściową i wyjściową układów. Oscylacje mogą nie tylko zakłócić funkcjonowanie układu logicznego, ale także ograniczyć jego szybkość działania, gdyż sygnał na końcu linii osiąga właściwą wartość dopiero po zaniku stanu przejściowego w linii.

Amplituda i czas trwania oscylacji w **linii długiej** zależą od:
— długości przewodów,
— konfiguracji przewodów,
— impedancji falowej przewodów,
— rezystancji wyjściowej układu nadawczego,
— rezystancji wejściowej układu odbiorczego.

Wyróżnia się dwa podstawowe ***sposoby transmisji sygnałów***:
— za pośrednictwem linii niesymetrycznych,
— za pośrednictwem linii symetrycznych.

Warto jeszcze przypomnieć, że dla zwiększenia pewności przekazywania informacji są stosowane dodatkowo specjalne kody detekcyjne lub korekcyjne (p. 1.3).

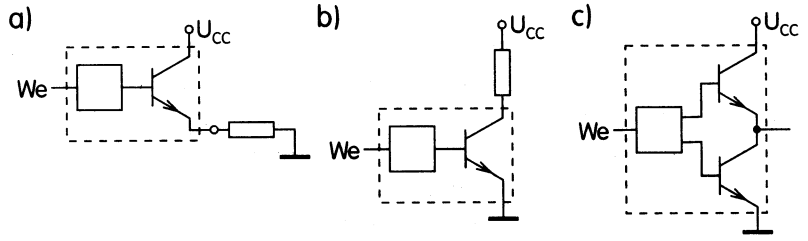
- **Transmisja sygnałów cyfrowych liniami niesymetrycznymi**

Układy z liniami niesymetrycznymi można stosować do transmisji sygnałów cyfrowych tylko na małe odległości, ze względu na ich wrażliwość na zakłócenia. Zastosowanie przewodów koncentrycznych lub częściowo ekranowanych (skrętki) powoduje zwiększenie odporności układu na działanie zakłóceń, ale także zwiększenie kosztu układu. Mimo to układy z liniami niesymetrycznymi są i tak tańsze niż z liniami symetrycznymi.

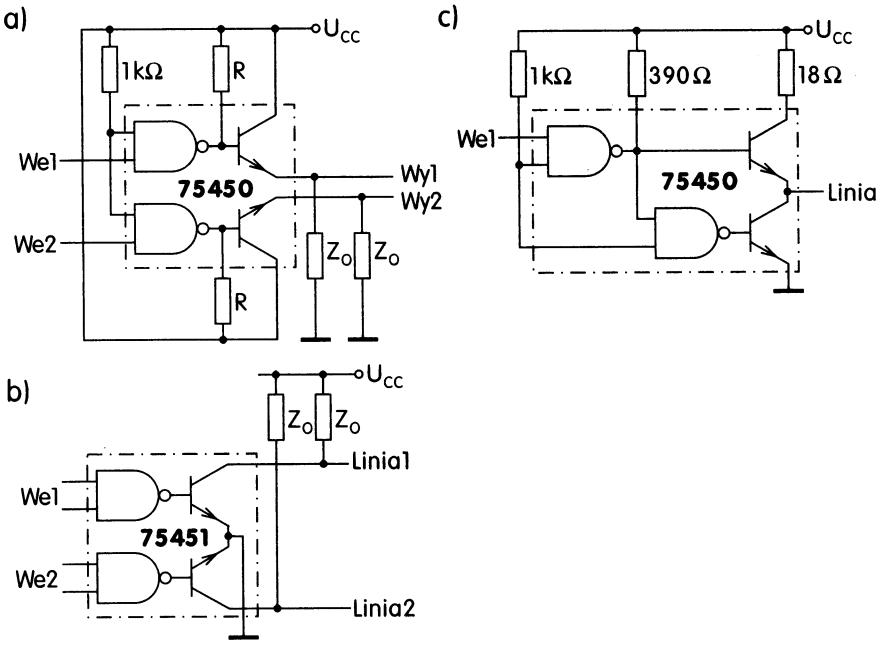
Bramka standardowa TTL może pracować jako nadajnik linii (źródło sygnału przesyłanego). Zalecana wartość impedancji falowej wynosi 100 Ω. Bramki nadawczej nie należy obciążać dodatkowo żadną inną bramką. Stosując skrętki, można do bramki standardowej dołączyć linię o długości do kilku metrów. Jednak w przypadku stosowania linii o impedancjach falowych 50 Ω i 75 Ω należy użyć bramki mocy.

Wiele typów układów scalonych może być wykorzystywanych w układach transmisji sygnałów cyfrowych. Należą do nich m.in. bramki z otwartym kolektorem, a także bramki z tranzystorem mocy na wyjściu (seria 75).

Na rysunku 9.32 przedstawiono trzy rodzaje układów wyjściowych nadajników, natomiast przykłady takich nadajników pokazano na rys. 9.33.



Rys. 9.32. Układy wyjściowe nadajników: a) z rezystorem dołączonym do masy; b) z rezystorem dołączonym do źródła zasilania; c) z wyjściem przeciwsobnym



Rys. 9.33. Schematy nadajników zbudowanych z zastosowaniem: a) układu 75450 (z rezystorami dołączonymi do masy); b) układu 75451 (z rezystorami dołączonymi do źródła zasilania); c) układu 75450 (z wyjściem przeciwsobnym)

• Transmisja sygnałów cyfrowych liniami symetrycznymi

W liniach symetrycznych sygnały są przesyłane za pośrednictwem dwóch przewodów. Odbiornik w takim systemie musi mieć wejście różnicowe. Linii symetrycznych używa się przy przesyłaniu sygnałów na duże odległości oraz wówczas, gdy oba systemy (nadawczy i odbiorczy) mają różne potencjały odniesienia (masy).

Sygnały zakłócające indukują się w obu przewodach i na wejściu różnicowym odbiornika odejmują się, dzięki czemu są eliminowane. Można zatem linie symetryczne stosować w warunkach oddziaływania silnych zakłóceń. Są to jednak systemy drogie i dlatego na ogół zamiast przewodów współosiowych używa się znacznie tańszych (i całkowicie tu wystarczających) skrętek.

Szczegółowe omówienie problemów transmisji wykracza poza ramy niniejszego podręcznika. Zainteresowani mogą je znaleźć w publikacji [8]. Przykładó-

we układy transmisji sygnałów cyfrowych na różne odległości można znaleźć: dla układów TTL — w pracy [21], dla układów CMOS — w pracy [3]. Przesyłanie sygnałów cyfrowych w warunkach oddziaływania silnych zakłóceń lub na znaczne odległości wymaga jednak stosowania specjalnych nadajników i odbiorników oraz właściwego doboru linii przesyłowych — patrz publikacje [8; 21]. W układach tych sygnały wejściowe nadajnika i sygnały wyjściowe odbiornika mają poziomy sygnałów TTL (CMOS), mimo że poziomy sygnałów w linii przesyłowej mogą się od nich znacznie różnić.

Pytania i zadania

1. Wyszukaj w katalogu odpowiednie parametry układu TTL 75450 i oblicz odległość, powyżej której połączenie wykonane:
   a) obwodem drukowanym,
   b) skrętką,
   c) kablem koncentrycznym 75 Ω,
   d) kablem koncentrycznym 50 Ω,
   e) linią paskową,

   będzie połączeniem długim.
2. Wykonaj zadanie 1. dla nadajnika zbudowanego z układu:
   a) TTL 75451,
   b) TTL 7406,
   c) TTL 74LS05,
   d) CMOS 74049,
   e) CMOS 74HC00,
   f) CMOS 74HCT00.
3. Kiedy połączenie nazywamy długim?
4. Jakie znasz sposoby transmisji sygnałów? Wymień ich wady i zalety.

10

Układy komutacyjne

10.1. Wprowadzenie

Układy kombinacyjne, umożliwiające przełączanie (komutację) sygnałów cyfrowych, nazywa się **układami komutacyjnymi.** Do podstawowych układów komutacyjnych zalicza się **multipleksery** i **demultipleksery**. Podobnie działają niektóre dekodery i z tego powodu zalicza się je także do układów komutacyjnych.

W niniejszym rozdziale omówiono zasadę działania multipleksera i demultipleksera. Podano przykłady zastosowań tych układów oraz opisano wybrane układy scalone zrealizowane w technice TTL i CMOS. Podobnie w odniesieniu do **koderów** i **dekoderów.**

10.2. Multipleksery i demultipleksery

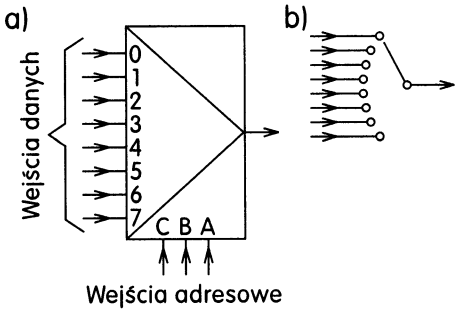
10.2.1. Multipleksery

Multiplekser *(zwany też* **selektorem danych***) służy do wyboru jednego z kilku sygnałów wejściowych i przekazania go na wyjście układu.*

Na rysunku 10.1 pokazano symbol graficzny multipleksera i model wyjaśniający zasadę jego działania. Działanie multipleksera odpowiada działaniu przełącznika wielopozycyjnego. Do nieruchomych styków przełącznika są doprowadzane sygnały wejściowe, a z suwaka sygnał jest przekazywany do wyjścia. Położenie suwaka określa, który sygnał wejściowy zostanie przełączony na wyjście układu. W przełączniku cyfrowym (multiplekserze) „ustawianie suwaka" odbywa się za pośrednictwem **wejść sterujących** — zwanych też **wejściami adresowymi.** Sygnały doprowadzane do wejść adresowych określają numer wejścia, z którego sygnał wejściowy przeniesiony zostanie na wyjście. Adresowanie jest realizowa-

ne w naturalnym kodzie binarnym i numer wejścia to dziesiętny odpowiednik adresu. Wejście **A** ma wagę 2^0 (doprowadzamy do tego wejścia bit LSB słowa adresowego), wejście **B** wagę 2^1 itd.

W związku z tym istnieje ścisły związek między liczbą wejść adresowych i liczbą **wejść danych** — zwanych też **wejściami informacyjnymi**. W multiplekserze o jednym wejściu adresowym można zaadresować dwa wejścia informacyjne, w multiplekserze o dwóch wejściach adresowych — cztery wejścia informacyjne, itd. Ogólnie liczba wejść informacyjnych wyrazi się zależnością $N = 2^n$ (gdzie n jest liczbą wejść adresowych). Najwyższy adres będzie miał wartość $2^n - 1$, bowiem numerację wejść rozpoczyna się od liczby 0. Liczba wejść informacyjnych może być oczywiście mniejsza niż N, ale nie może być od niej większa. Aktualnie są produkowane układy multiplekserów o $n = 1, 2, 3$ i 4 wejściach adresowych i odpowiednio o $N = 2^1, 2^2, 2^3$ i 2^4 wejściach informacyjnych. Multipleksery te mają wyjścia dwustanowe (stan wysoki, stan niski) lub wyjścia trójstanowe (stan wysoki, stan niski, stan wielkiej impedancji).



Rys. 10.1. Multiplekser: a) symbol graficzny; b) model mechaniczny wyjaśniający zasadę działania

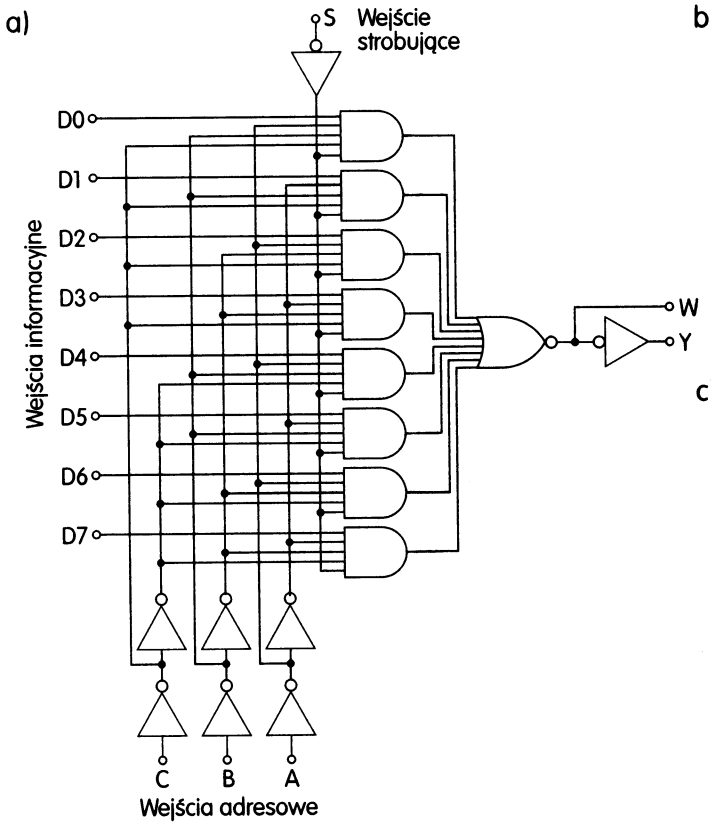
Większość multiplekserów (o wyjściu dwustanowym) ma dodatkowe wejście sterujące, zwane **wejściem strobującym** lub **zezwalającym** (ang. *strobe, enable*). Jeśli wejście to jest w stanie niskim, to multiplekser działa tak, jak podano w jego określeniu; natomiast jeśli jest w stanie wysokim, to niezależnie od stanu wejść informacyjnych i adresowych stan wyjścia jest stały i równy **0** (wyjście proste) lub **1** (wyjście zanegowane). W przypadku multiplekserów z wyjściem trójstanowym zamiast wejścia strobującego jest wejście sterujące wyjściem układu (przełącza z trybu pracy dwustanowej w stan wielkiej impedancji i na odwrót).

W technice TTL i CMOS są wytwarzane multipleksery scalone o liczbie wejść adresowych $1 \div 4$ i odpowiednio wejść informacyjnych $2 \div 16$.

Na rysunku 10.2 przedstawiono schemat logiczny oraz tablicę opisującą działanie multipleksera **'151**. Jego odpowiednik **'F251** różni się jedynie tym, że ma wyjście trójstanowe.

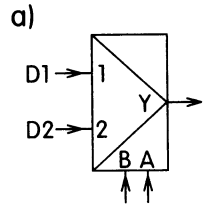
Łatwo zauważyć (rys. 10.2), że tylko ta bramka iloczynowa ma wszystkie wejścia w stanie **1** (sterowane z wejść adresowych), do której jest doprowadzone wejście informacyjne o numerze odpowiadającym ustawionemu adresowi.

Zauważmy jeszcze, że każdy sygnał wejściowy steruje wejściem tylko jednej bramki. Dzięki temu osiąga się standaryzację obciążeń wnoszonych przez dowolne wejście układu TTL. Każde wejście pobiera prąd odpowiadający jednemu wejściu TTL niezależnie od tego, do ilu wejść bramek sygnał z tego wejścia jest wewnątrz układu wykorzystywany.



S	C	B	A	Y	W
0	0	0	0	D0	$\overline{D0}$
0	0	0	1	D1	$\overline{D1}$
0	0	1	0	D2	$\overline{D2}$
0	0	1	1	D3	$\overline{D3}$
0	1	0	0	D4	$\overline{D4}$
0	1	0	1	D5	$\overline{D5}$
0	1	1	0	D6	$\overline{D6}$
0	1	1	1	D7	$\overline{D7}$
1	x	x	x	0	1

Rys. 10.2. Multiplekser scalony '151: a) schemat logiczny; b) symbol graficzny; c) tablica działania



B	A	Y
0	0	0
0	1	D1
1	0	D2
1	1	D1+D2

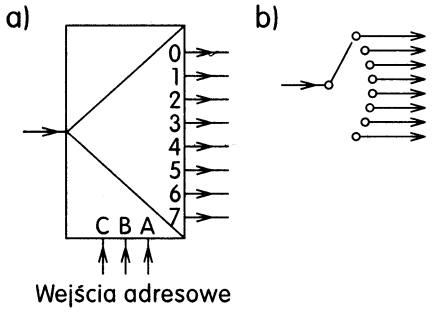
Rys. 10.3. Układ scalony MCY74019: a) symbol graficzny; b) tablica działania

Układ scalony MCY74019 wykonany w technice CMOS serii 4000B (MCY74) zawiera cztery multipleksery o dwóch wejściach informacyjnych. Ze względu na inny sposób sterowania zastosowany w tym układzie jego symbol graficzny oraz tablicę działania przedstawiono na rys. 10.3.

10.2.2. Demultipleksery

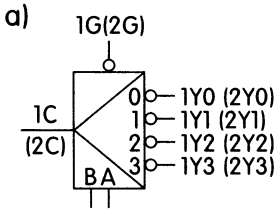
Demultiplekser ***umożliwia przesłanie do jednego z wyjść układu sygnału doprowadzonego do jego wejścia.*** Pozostałe wyjścia pozostają w jednym ze stanów (**L** lub **H**) w zależności od konkretnego typu demultipleksera.

Na rysunku 10.4 pokazano symbol graficzny demultipleksera i model wyjaśniający zasadę jego działania. Działanie demultipleksera odpowiada działaniu przełącznika wielopozycyjnego. Do ruchomego styku (suwaka) przełącznika jest doprowadzony sygnał wejściowy i w zależności od położenia suwaka sygnał ten jest przenoszony na odpowiednie wyjście. W przełączniku cyfrowym (demultiplekserze) „nastawianie suwaka" odbywa się za pośrednictwem **wejść sterujących** — zwanych też **wejściami adresowymi.** Sygnały doprowadzane do wejść adresowych określają numer wyjścia, na które zostanie przeniesiony sygnał wejściowy . Adresowanie i zasady z nim związane są identyczne, jak w przypadku omówionych wcześniej multiplekserów.



Rys. 10.4. Demultiplekser: a) symbol graficzny b) model mechaniczny wyjaśniający zasadę działania

Układ scalony **'155** zawiera dwa czterowyjściowe demultipleksery (rys. 10.5). Oba są sterowane tym samym dwubitowym słowem adresowym, mają natomiast niezależne sygnały strobujące. Różnica między nimi polega na tym, że w jednym z nich na wyjściu pojawia się zanegowany sygnał wejściowy. Umożliwia to w prosty



b)

B	A	1G	1C	1Y0	1Y1	1Y2	1Y3
-	-	1	-	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	1	0	1
1	1	0	1	1	1	1	0
-	-	-	0	1	1	1	1

c)

B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
-	-	1	-	1	1	1	1
0	0	0	0	0	1	1	1
0	1	0	0	1	0	1	1
1	0	0	0	1	1	0	1
1	1	0	0	1	1	1	0
-	-	-	1	1	1	1	1

Uwaga: Opisy w nawiasach odnoszą się do drugiego demultipleksera

Rys. 10.5. Demultiplekser '155: a) symbol graficzny; b, c) tablice stanów pierwszego i drugiego demultipleksera

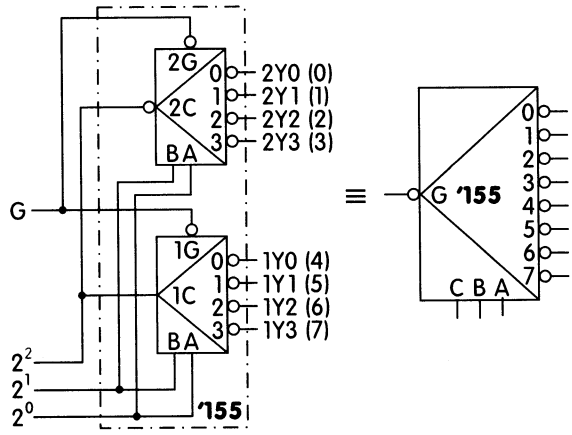
sposób otrzymanie z układu **'155** jednego demultipleksera 8-wyjściowego (rys. 10.6). Opis działania takiego demultipleksera przedstawia tabl. 10.1 po następującej modyfikacji: **G1 = G2 = G**, pominięcie wejścia adresowego **D**, pominięcie wyjść o numerach 8 ÷ 15.

Innym przykładem demultipleksera o 8 wyjściach może być układ **'138.**

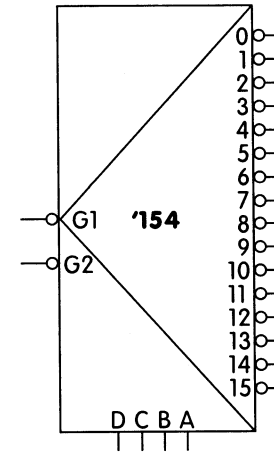
Demultiplekser **'154** (rys. 10.7) jest układem 16-wyjściowym o 4-bitowym słowie adresowym. W tablicy 10.1 przedstawiono opis działania tego demultipleksera.

Na wybranym (przez wejścia adresowe) wyjściu pojawia się stan niski **0** tylko wówczas, gdy do obu wejść **G1** i **G2** jest doprowadzony sygnał o poziomie logicznym **0**.

Jeżeli do wejścia (np.) **G1** doprowadzimy stan **L**, a do wejścia **G2** falę prostokątną, to pojawi się ona na wyjściu określonym przez wejścia adresowe. Zmie-



Rys. 10.6. Demultiplekser 8-wyjściowy zbudowany z układu scalonego '155



Rys. 10.7. Symbol graficzny demultipleksera '154

niając stan wejść adresowych, możemy taki przebieg prostokątny (np. zegarowy) rozdzielać na poszczególne układy sterowane z poszczególnych wyjść demultipleksera. Wejście **G1** pozwala wówczas na blokowanie pracy takiego rozdzielacza.

Tablica 10.1. **Tablica działania demultipleksera '154**

Wejścia						Wyjścia															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	–	–	–	–	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	–	–	–	–	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	–	–	–	–	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Najczęściej demultiplekser **'154** pracuje z obydwoma wejściami (**G1**, **G2**) połączonymi ze sobą i traktowanymi jako jedno wejście informacyjne lub jedno z wejść **G** stanowi wejście informacyjne, a drugie wejście strobujące.

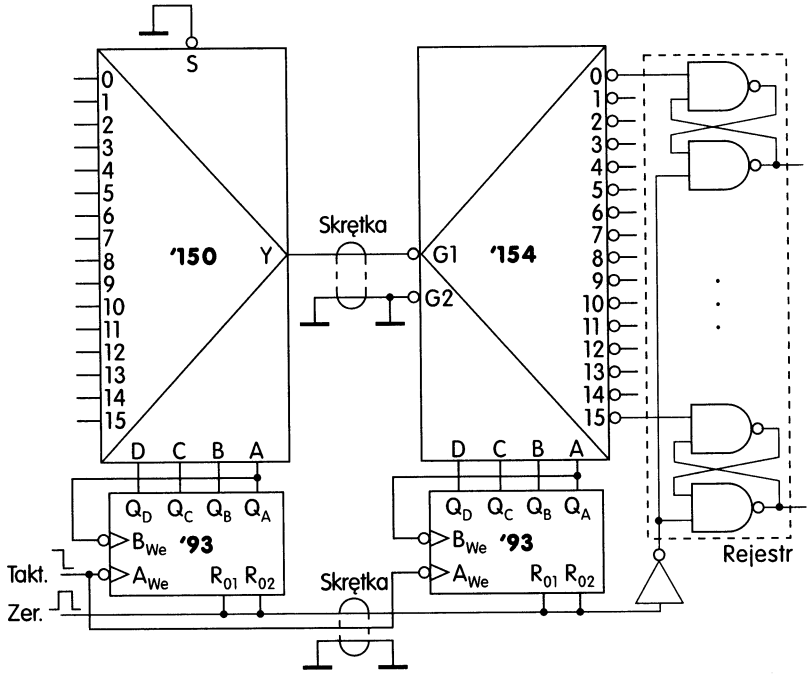
Zauważmy, że na wyjściu demultipleksera **'154** występuje słowo 16-bitowe (w demultiplekserze **'155** — 4-bitowe lub 8-bitowe), w którym tylko jeden z bitów ma wartość **0**, a pozostałe wartość **1**. Kod, w którym wyróżniony bit ma wartość **0**, a pozostałe wartość **1** jest nazywany kodem $\overline{\mathbf{1}}\mathbf{z}\boldsymbol{n}$ (patrz p. 1.3). Demultiplekser **'154** jest więc także dekoderem **naturalnego kodu dwójkowego** na kod $\overline{\mathbf{1}}\mathbf{z}\overline{\mathbf{16}}$. Jeżeli demultiplekser ten będziemy adresować słowami kodu **naturalnego BCD** i wykorzystywać pierwsze dziesięć wyjść, to taki układ będzie dekoderem kodu **BCD 8421** na kod $\overline{\mathbf{1}}\mathbf{z}\overline{\mathbf{10}}$. Dekoderem kodu **BCD 8421** na kod **1z10** jest np. układ scalony CMOS MCY74028. Może on być wykorzystywany także jako demultiplekser. Dokładniejszy jego opis znajduje się w p. 10.3. Dekoderem takim jest także układ **'537**, który ma wyjścia trójstanowe.

Uogólniając powyższe uwagi, należy stwierdzić, że każdy demultiplekser może pracować jako strobowany dekoder.

10.2.3. Przykłady zastosowań multiplekserów i demultiplekserów

- **Multipleksowy system przesyłania danych**

W celu uproszczenia i obniżenia kosztów systemu transmisji danych cyfrowych stosuje się technikę multipleksową. Technika ta umożliwia przesyłanie wielobitowych słów binarnych jedną linią zamiast wieloma przewodami. Schemat takiego układu przedstawiono na rys. 10.8.



Rys. 10.8. Multipleksowy system transmisji danych

Multiplekser pełni w nim rolę przetwornika, który zamienia format słów z równoległego na szeregowy. Demultiplekser dokonuje konwersji odwrotnej, tzn. zamienia informację szeregową na równoległą. Warunkiem koniecznym, aby w danej chwili sygnał z *i*-tego wejścia multipleksera był przenoszony na *i*-te wyjście demultipleksera, jest ustawienie identycznych słów adresowych w obu układach. Warunek ten jest spełniony poprzez adresowanie multipleksera i demultipleksera sygnałami z wyjść liczników binarnych (**mod 16**), zliczających impulsy tego samego przebiegu taktującego pracę układu. Liczniki te pracują dzięki temu współbieżnie (synchronicznie). Sposób ten pozwala na dalsze zmniejszenie liczby przewodów potrzebnych do transmisji informacji.

W trakcie pracy układu informacja z określonego wejścia pojawia się na odpowiednim wyjściu tylko na czas jednego taktu przebiegu zegarowego (taktującego). Aby uzyskać na wyjściu układu pełne słowo 16-bitowe, konieczne jest zastosowanie układu pamięci. Został on zbudowany z przerzutników asynchronicznych typu $\overline{r}\,\overline{s}$. Cykl pracy układu jest zatem następujący:

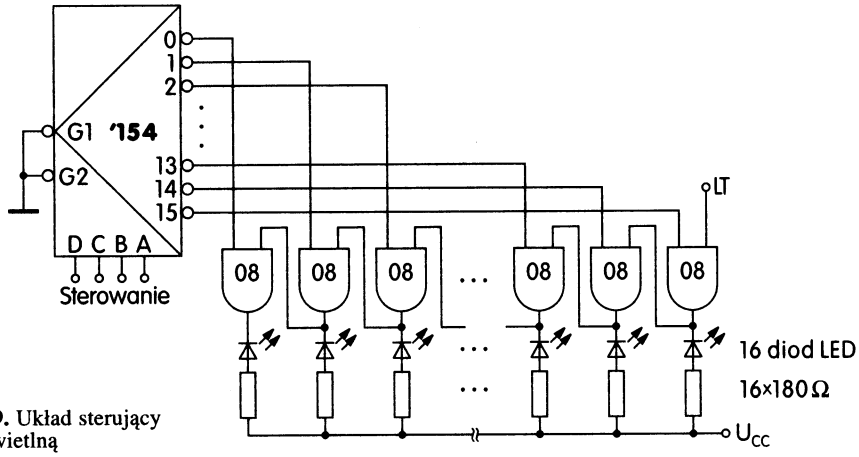
1. Wyzerowanie układu (przerzutniki wyjściowe ustawione w stan **1**, liczniki w stan **0**).
2. Ustawienie na wejściu informacji przeznaczonej do transmisji.
3. Podanie 16 impulsów na wejścia zliczające liczników (z każdym kolejnym impulsem sygnał z jednego z wejść jest przenoszony na odpowiednie wyjście i jeśli jest to **1**, to przerzutnik nie zmienia swego stanu, a jeśli **0**, to jest ustawiany w stan niski **L**).
4. Informacja z wejścia znajduje się na wyjściu układu i można przejść do początku kolejnego cyklu (p. 1. cyklu pracy).

Problem transmisji szeregowo-równoległej można także rozwiązać przy użyciu rejestrów np. **'164** i **'165** (rozdz. 13).

- **Linijka świetlna**

W układach cyfrowych wskaźnikiem wyjściowym jest często dioda typu LED. Wyświetlacz cyfrowy można zastąpić tanim układem wskaźnika diodowego utworzonego z szeregu diod. W takim szeregu świeci jedna dioda lub dodatkowo (oprócz tej jednej) wszystkie, które wskazują mniejszą wartość wielkości wyjściowej. W tym drugim przypadku obserwujemy „linijkę świetlną", której długość jest wskaźnikiem wartości wielkości wyjściowej. Układ taki może w pewnych zastosowaniach zastąpić analogowy wskaźnik wychyłowy. Układ linijki świetlnej zbudowanej przy użyciu demultipleksera **'154** przedstawiono na rys. 10.9.

Wybranie dowolnego wyjścia (słowem adresowym) powoduje świecenie sterowanej przez nie (za pośrednictwem bramki AND) diody oraz wszystkich diod przyłączonych do wyjść o numerach mniejszych od wybranego. Wejście **LT** pozwala skontrolować świecenie diod. Doprowadzenie do niego poziomu logicznego **0** powoduje świecenie wszystkich diod, niezależnie od stanu wejść adresowych.



Rys. 10.9. Układ sterujący linijką świetlną

• Realizacja układów kombinacyjnych

W bardzo prosty sposób można realizować funkcje logiczne, wykorzystując multiplekser. W przypadku natomiast układu kombinacyjnego wielowyjściowego korzystniejsze jest użycie demultipleksera. Sposoby realizacji układów kombinacyjnych zbudowanych z multiplekserów i demultiplekserów zostaną przedstawione na kilku kolejnych przykładach.

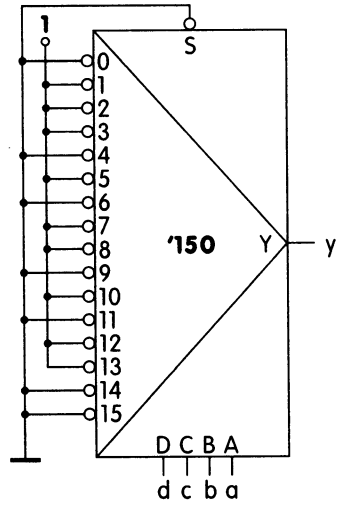
Przykład 10.1

Zbudować układ kombinacyjny opisany następującą funkcją logiczną:
$\mathbf{y = f(d,c,b,a)} = \Sigma(1,2,3,5,7,8,10,12,13)$.

Rozwiązanie

Zauważmy, że doprowadzając sygnały **d**, **c**, **b**, **a** do wejść adresowych multipleksera tak, jak to pokazano na rys. 10.10, adresujemy go zgodnie z tablicą

	d	c	b	a	y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0



Rys. 10.10. Tablica prawdy oraz schemat logiczny układu do przykładu 10.1

prawdy. Jeżeli w *i*-tym wierszu tablicy prawdy funkcja ma wartość **1**, to wejście informacyjne multipleksera o numerze „*i*" należy połączyć ze źródłem poziomu logicznego **1**. Jeżeli natomiast w *i*-tym wierszu tablicy prawdy funkcja ma wartość **0**, to wejście informacyjne multipleksera o numerze „*i*" należy połączyć ze źródłem poziomu logicznego **0**. ■

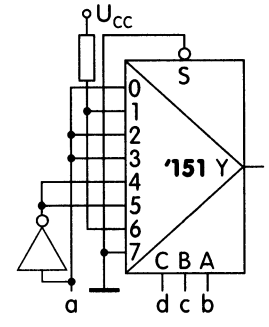
Przykład 10.2

Zbudować układ kombinacyjny opisany jak w przykładzie 10.1, korzystając z multipleksera o trzech wejściach adresowych.

Rozwiązanie

Doprowadzając sygnały **d**, **c**, **b** do wejść adresowych multipleksera tak, jak to pokazano na rys. 10.11, sprawiamy, że sygnał **b** steruje wejściem adresowym **A** (o wadze 2^0), sygnał **c** steruje wejściem adresowym **B** (o wadze 2^1), sygnał **d** steruje wejściem adresowym **C** (o wadze 2^2). Zatem, w przypadku dwóch

	d	c	b	a	y
Adres 0	0	0	0	0	0
0	0	0	0	1	1
Adres 1	0	0	1	0	1
1	0	0	1	1	1
Adres 2	0	1	0	0	0
2	0	1	0	1	1
Adres 3	0	1	1	0	0
3	0	1	1	1	1
Adres 4	1	0	0	0	1
4	1	0	0	1	0
Adres 5	1	0	1	0	1
5	1	0	1	1	0
Adres 6	1	1	0	0	1
6	1	1	0	1	1
Adres 7	1	1	1	0	0
7	1	1	1	1	0



Rys. 10.11. Tablica prawdy oraz schemat logiczny układu do przykładu 10.2

kolejnych wierszy tablicy prawdy ustawiany jest taki sam adres, bowiem trójki bitów **dcb** są w nich takie same. Wiersze te wyróżniono, zwiększając odstęp pomiędzy tymi parami i obok (po lewej stronie) zapisano odpowiadający im wspólny adres.

W obrębie dowolnych wyróżnionych dwóch wierszy tablicy wartość funkcji jest albo stała **1**, albo stała **0**, albo przyjmuje wartości takie jak sygnał **a**, albo wreszcie takie jak zanegowany sygnał **a** (czyli $\overline{\mathbf{a}}$). Uogólniając, można powiedzieć, że sygnał wejściowy multipleksera jest funkcją tej zmiennej (zmiennych), która nie została użyta do adresowania multipleksera. W tym przykła-

dzie liczba zmiennych nie wykorzystywanych do adresowania jest równa 1. Sygnał wejściowy multipleksera będzie zatem funkcją jednej zmiennej. Takich funkcji jest tylko cztery (patrz p. 2.2, wzór (2.7)). Spostrzeżenia te pozwalają sformułować wniosek, że ***w przypadku realizacji funkcji logicznej n-zmiennych za pomocą multipleksera o liczbie wejść adresowych równej n–1 potrzeba (w najgorszym przypadku) użyć dodatkowo tylko jednego negatora. Taki sposób realizacji należy zatem przyjąć jako zasadę, gdyż jest on tańszy i bardziej niezawodny (mniejsza liczba połączeń) od przedstawionego powyżej.*** Zwykle jednak w układach cyfrowych dysponujemy sygnałami wejściowymi w pozycji i negacji. Wówczas użycie negatora staje się zbędne i przewaga układu jak na rys. 10.11 nad układem jak na rys. 10.10 jest jeszcze bardziej ewidentna. ■

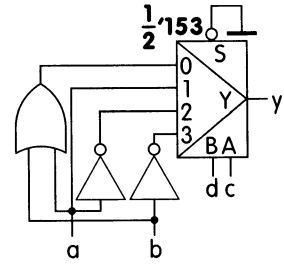
Przykład 10.3

Zbudować układ kombinacyjny opisany jak w przykładzie 10.1, korzystając z multipleksera o dwóch wejściach adresowych i dowolnych bramek logicznych.

Rozwiązanie

Doprowadzając sygnały **c**, **d** do wejść adresowych multipleksera tak, jak to pokazano na rys. 10.12 sprawiamy, że sygnał **c** steruje wejściem adresowym **A** (o wadze 2^0), sygnał **d** steruje wejściem adresowym **B** (o wadze 2^1). Zatem w przypadku czterech kolejnych wierszy tablicy prawdy jest ustawiany taki sam adres, bowiem dwójki bitów **dc** są w nich takie same. Wiersze te wyróżniono, zwiększając odstęp pomiędzy tymi czwórkami i obok (po lewej stronie) zapisano odpowiadający im wspólny adres.

	d	c	b	a	y	
	0	0	0	0	0	
Adres 0	0	0	0	1	1	a+b
	0	0	1	0	1	
	0	0	1	1	1	
	0	1	0	0	0	
Adres 1	0	1	0	1	1	a
	0	1	1	0	0	
	0	1	1	1	1	
	1	0	0	0	1	
Adres 2	1	0	0	1	0	$\bar{a}$
	1	0	1	0	1	
	1	0	1	1	0	
	1	1	0	0	1	
Adres 3	1	1	0	1	1	$\bar{b}$
	1	1	1	0	0	
	1	1	1	1	0	



Rys. 10.12. Tablica prawdy oraz schemat logiczny układu do przykładu 10.3

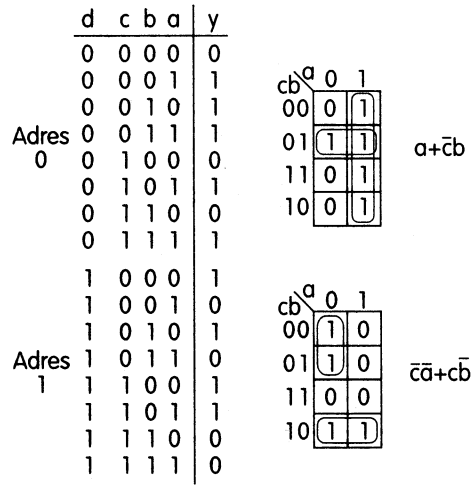
W obrębie dowolnych wyróżnionych czterech wierszy tablicy wartość funkcji (a jednocześnie sygnał wejściowy odpowiedniego wejścia informacyjnego multipleksera) jest funkcją tych zmiennych, które nie zostały użyte do adresowania multipleksera. W niniejszym przykładzie są to zmienne **b** i **a**. Funkcje te określamy bezpośrednio na podstawie tablicy prawdy lub zapisujemy dla każdej odpowiednią tablicę Karnaugha (dwóch zmiennych) i poszukujemy postaci minimalnej. Jak wiadomo (patrz p. 2.2, wzór (2.7)) wszystkich funkcji dwóch zmiennych jest 16. Zatem realizacja funkcji kombinacyjnej n-zmiennych za pomocą multipleksera o liczbie wejść adresowych równej n–2 może wymagać dodatkowego użycia różnych bramek i w efekcie niezbędna liczba układów scalonych potrzebnych do realizacji takiego układu uczyni ten sposób nieopłacalnym. ■

Przykład 10.4

Zbudować układ kombinacyjny opisany jak w przykładzie 10.1, korzystając z multipleksera o jednym wejściu adresowym i dowolnych bramek logicznych.

Rozwiązanie

Rozwiązanie zadania sprowadza się do realizacji za pomocą bramek dwóch funkcji zmiennych **c**, **b**, i **a** (każda), doprowadzeniu wyjść tych układów do wejść multipleksera oraz przesłaniu jednej z nich do wyjścia multipleksera poprzez adresowanie go sygnałem wejściowym **d**. Odpowiednio podzieloną tablicę prawdy oraz tablice Karnaugha, pozwalające zminimalizować poszukiwane funkcje, przedstawiono na rys. 10.13. Narysowanie samego układu pozostawia się Czytelnikowi.



Rys. 10.13. Tablica prawdy oraz pomocnicze tablice Karnaugha do przykładu 10.4 ■

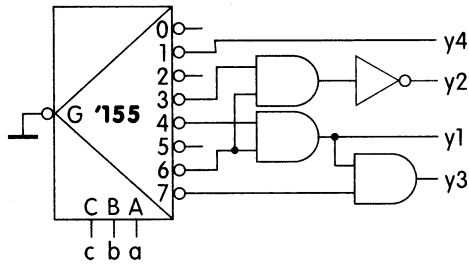
Przykład 10.5

Zbudować układ kombinacyjny czterowyjściowy opisany funkcjami:
$\mathbf{y}_1 = \mathbf{f}_1(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[1,2,5,7(0,3)]$
$\mathbf{y}_2 = \mathbf{f}_2(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[3,6(2,7)]$
$\mathbf{y}_3 = \mathbf{f}_3(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[4,6,7(0,3)]$
$\mathbf{y}_4 = \mathbf{f}_4(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[1(2,5)]$
korzystając z demultipleksera o trzech wejściach adresowych (np. **'155**) i dowolnych bramek logicznych.

Rozwiązanie

Rozwiązanie zadania z przykładu 10.5 przedstawiono na rys. 10.14. Doprowadzenie sygnałów wejściowych **c**, **b**, i **a** do wejść adresowych demultipleksera tak, jak pokazano to na rys. 10.14, spowodowało, że adresowanie jest zgodne z tablicą prawdy.

c	b	a	y1	y2	y3	y4
0	0	0	-	0	-	1
0	0	1	1	0	1	0
0	1	0	1	-	1	-
0	1	1	-	1	-	1
1	0	0	0	0	0	1
1	0	1	1	0	1	-
1	1	0	0	1	0	1
1	1	1	1	-	0	1



Rys. 10.14. Tablica prawdy oraz schemat logiczny układu do przykładu 10.5

Dalsze postępowanie odnośnie do poszczególnych funkcji było następujące.

Funkcja $\mathbf{y}_1$

Do realizacji wybrano zera funkcji, gdyż ich liczba jest mniejsza niż liczba jedynek funkcji. Gdy na wejściu adresowym zostanie ustawiony stan **100** albo **110** (adres 4 albo 6), wówczas na wyjściu 4 (6) demultipleksera wystąpi stan logiczny **0** (a jak wynika z opisu działania demultipleksera **'155** na pozostałych wyjściach stan wysoki **1**). Funkcja $\mathbf{y}_1$ w obu przypadkach powinna mieć wartość **0**, dlatego wyjścia te należy połączyć z wejściami bramki iloczynowej (rys. 10.14). Na wyjściu tej bramki stan **0** wystąpi tylko w obu tych przypadkach, czyli będzie realizowana funkcja $\mathbf{y}_1$.

Funkcja $\mathbf{y}_2$

Do realizacji wybrano jedynki funkcji (jest ich 2), gdyż ich liczba jest mniejsza niż liczba zer funkcji (jest ich 4). Jeżeli wyjścia 3 i 6 podłączymy do wejść bramki AND, to na jej wyjściu przy słowach wejściowych **abc = 011** (adres 3) lub **abc = 110** (adres 6) otrzymamy stan logiczny **0** (patrz realizacja funkcji $\mathbf{y}_1$), a w pozostałych przypadkach stan logiczny **1**. Funkcja $\mathbf{y}_2$ powinna mieć wartości przeciwne (patrz tablica prawdy), dlatego należy sygnał wyjściowy z bramki AND zanegować (lub zamiast niej użyć bramki NAND).

Naturalnie, można funkcję $\mathbf{y}_2$ zrealizować wykorzystując bramkę AND, ale musiałaby to być bramka czterowejściowa, gdyż realizacja zer funkcji wymagałaby połączenia wyjść 0, 1, 4, 5 demultipleksera z wejściami takiej bramki.

Funkcja $\mathbf{y}_3$

Do realizacji tej funkcji można wybrać zarówno zera, jak i jedynki funkcji (zbiór zer jest tak samo liczny, jak zbiór jedynek). Realizacja zer wymagałaby użycia trójwejściowej bramki AND i połączenia jej wejść z wyjściami 4, 6, 7 demultipleksera. Realizacja jedynek wymagałaby użycia trójwejściowej bramki NAND i połączenia jej wejść z wyjściami 1, 2, 5 demultipleksera.

Zwróćmy jednak uwagę, że realizując zera funkcji, mnożymy w bramce iloczynowej sygnały z wyjść 4, 6 i 7. Natomiast realizując funkcję $\mathbf{y}_1$, mnożyliśmy sygnały z wyjść 4 i 6. Wystarczy zatem sygnał $\mathbf{y}_1$ pomnożyć przez sygnał z wyjścia 7, aby otrzymać pożądany efekt. Dzięki temu spostrzeżeniu zamiast bramki trójwejściowej użyjemy bramki tylko o dwóch wejściach.

Funkcja $\mathbf{y}_4$

Funkcja ta ma wartość **0** tylko dla kombinacji wejściowej **abc = 001**, a przy pozostałych słowach wejściowych ma wartość **1** (lub może mieć wartość **1**). Tak działa właśnie demultiplekser adresowany w taki sposób, że **CBA = cba** i sygnał wyjściowy jest pobierany z wyjścia **1**. ■

Uogólniając, możemy stwierdzić, że każde wyjście demultipleksera realizuje dokładnie jedno zero funkcji, odpowiadające takiej kombinacji sygnałów wejściowych, która adresuje analizowane wyjście. Realizacja funkcji to iloczyn jej zer (patrz postać kanoniczna — p. 3.3). Jednak, aby zminimalizować liczbę połączeń projektowanego układu, gdy liczba zer jest większa niż liczba jedynek funkcji, wówczas realizujemy funkcję odwrotną $\overline{\mathbf{y}}$ (czyli **1** funkcji), a następnie dodajemy negator, co w efekcie daje pożądany wynik.

Tak więc realizacja funkcji za pomocą demultipleksera o trzech wejściach adresowych (8 wyjściach) wymaga, w ogólnym przypadku, użycia bramek co najwyżej czterowejściowych. Odnosi się to do sytuacji, gdy liczba zer jest równa liczbie jedynek. W każdym innym przypadku liczba zer jest mniejsza niż 4, albo liczba jedynek jest mniejsza niż 4. Uogólniając powyższy wniosek, powiemy, że realizacja funkcji za pomocą demultipleksera o m wyjściach będzie wymagać użycia bramek o liczbie wejść co najwyżej równej $m/2$.

Przykład 10.6

Zbudować układ kombinacyjny o jednym wyjściu opisany funkcją
$\mathbf{y}_1 = \mathbf{f}_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e}) = \Sigma(0,1,3,4,8,11,12,13,17,19,21,22,24,25,26,27)$
korzystając z demultipleksera o dwóch wejściach adresowych, multipleksera o trzech wejściach adresowych oraz dowolnych bramek logicznych.

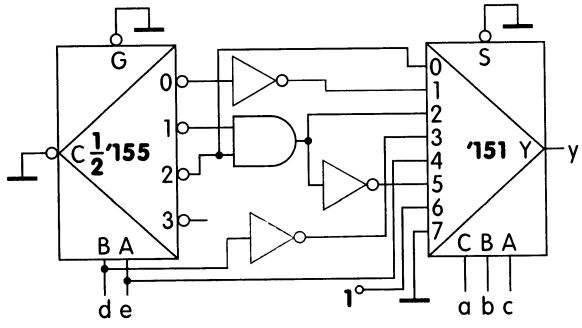
Rozwiązanie

Rozwiązanie zadania rozkładamy na dwa etapy. Pierwszy to realizacja ośmiu funkcji dwóch zmiennych **d** i **e**. Funkcje te przedstawiono na rys. 10.15, wy-

różniając odpowiednie fragmenty tablicy prawdy liniami przerywanymi. Obok zamieszczono zapis tych funkcji (w postaci kanonicznej iloczynu), numerując je od 0 do 7. Funkcje te realizujemy przy użyciu demultipleksera o dwóch wejściach adresowych (rys. 10.16), doprowadzając do tych wejść odpowiednio: sygnał **e** — wejście **A**, sygnał **d** — wejście **B**.

	a	b	c	d	e	y	
	0	0	0	0	0	1	
(0)	0	0	0	0	1	1	$y0=f_0(d,e)=\Pi(2)$
	0	0	0	1	0	0	
	0	0	0	1	1	1	
	0	0	1	0	0	1	
(1)	0	0	1	0	1	0	$y1=f_1(d,e)=\Pi(1,2,3)=\overline{\Pi(0)}$
	0	0	1	1	0	0	
	0	0	1	1	1	0	
	0	1	0	0	0	1	
(2)	0	1	0	0	1	0	$y2=f_2(d,e)=\Pi(1,2)$
	0	1	0	1	0	0	
	0	1	0	1	1	1	
	0	1	1	0	0	1	
(3)	0	1	1	0	1	1	$y3=f_3(d,e)=\Pi(2,3)=\bar{d}$
	0	1	1	1	0	0	
	0	1	1	1	1	0	
	1	0	0	0	0	0	
(4)	1	0	0	0	1	1	$y4=f_4(d,e)=\Pi(0,2)=e$
	1	0	0	1	0	0	
	1	0	0	1	1	1	
	1	0	1	0	0	0	
(5)	1	0	1	0	1	1	$y5=f_5(d,e)=\Pi(0,3)=\overline{\Pi(1,2)}=\overline{y2}$
	1	0	1	1	0	1	
	1	0	1	1	1	0	
	1	1	0	0	0	1	
(6)	1	1	0	0	1	1	$y6=f_6(d,e)=1$
	1	1	0	1	0	1	
	1	1	0	1	1	1	
	1	1	1	0	0	0	
(7)	1	1	1	0	1	0	$y7=f_7(d,e)=\Pi(0,1,2,3)=0$
	1	1	1	1	0	0	
	1	1	1	1	1	0	

Rys. 10.15. Tablica prawdy do przykładu 10.6



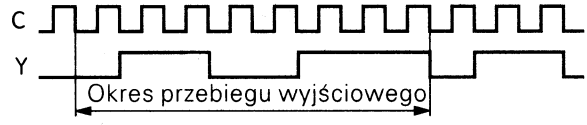
Rys. 10.16. Schemat logiczny układu do przykładu 10.6

Drugi etap to przesłanie do wyjścia układu jednej z ośmiu funkcji. O tym, która funkcja ma zostać przełączona na wyjście decydują sygnały **abc**, którymi adresujemy multiplekser. Zauważmy, że w obszarze każdej z funkcji $\mathbf{y}_0$ do $\mathbf{y}_7$ sygnały **abc** nie zmieniają się, czyli adresują określone (i to samo) wejście multipleksera (zapisano je w nawiasach po lewej stronie tablicy prawdy — rys. 10.15). ■

Przykład 10.6 pokazuje, że można realizować układ kombinacyjny *n*-wejściowy przy użyciu multi- i demultipleksera o łącznej liczbie wejść adresowych równej liczbie zmiennych *n*. Realizacja taka jest względnie prosta. Jednak przykład ten należy potraktować raczej jako ćwiczenie utrwalające wiadomości dotyczące multiplekserów i demultiplekserów oraz rozwijające umiejętność stosowania tych układów. W praktyce bowiem układy kombinacyjne wielowejściowe są obecnie budowane przy zastosowaniu pamięci typu ROM lub tzw. układów PLD (patrz rozdz. 14).

Pytania i zadania

1. Zaprojektuj multiplekser o adresie:
   a) jednobitowym,
   b) dwubitowym.
2. Dany jest przebieg zegarowy **C** oraz multiplekser o trzech wejściach adresowych. Zbuduj układ generatora przebiegu jak na rys. 10.17.



Rys. 10.17. Przebiegi czasowe do zadania 2

Wskazówka. Użyj licznika **mod 8** zliczającego impulsy przebiegu zegarowego **C**. Wyjścia licznika wykorzystaj do adresowania multipleksera.

3. Zbuduj układ kombinacyjny $\mathbf{y} = \Sigma(0,3,4,6,7,8)$ przy użyciu:
   a) multipleksera o 3 wejściach adresowych,
   b) multipleksera o 2 wejściach adresowych,
   c) multipleksera o 1 wejściu adresowym,
   używając dodatkowo (w razie konieczności) dowolnych bramek logicznych.
4. Wykonaj zadanie 3. dla funkcji:
   $\mathbf{y} = \Sigma(2,3,5,6,8,9,10)$.
5. Wykonaj zadanie 3. dla funkcji:
   $\mathbf{y} = \Sigma(0,2,4,5,9,15)$.
6. Wykonaj zadanie 3. dla funkcji:
   $\mathbf{y} = \Sigma(0,4,6,7,8,9,12,14)$.
7. Zrealizuj, korzystając z demultipleksera o 3 wejściach adresowych, układ kombinacyjny czterowyjściowy opisany:
   $\mathbf{y}_1 = \mathbf{f}_1(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[0,1,4(6)]$,
   $\mathbf{y}_2 = \mathbf{f}_2(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[2,4,5(6,7)]$,
   $\mathbf{y}_3 = \mathbf{f}_3(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[2,3(4)]$,
   $\mathbf{y}_4 = \mathbf{f}_4(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[1,2,3(5,6,7)]$.

8. Zrealizuj przy użyciu demultipleksera o 3 wejściach adresowych układ kombinacyjny czterowyjściowy opisany:
   $\mathbf{y}_1 = \mathbf{f}_1(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[6,7(0,1,3)]$,
   $\mathbf{y}_2 = \mathbf{f}_2(\mathbf{c},\mathbf{b},\mathbf{a}) = \Sigma[1,4,7(0,2,5)]$,
   $\mathbf{y}_3 = \mathbf{f}_3(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[4,6(3,5)]$,
   $\mathbf{y}_4 = \mathbf{f}_4(\mathbf{c},\mathbf{b},\mathbf{a}) = \Pi[0,3,5(1,4,7)]$.
   Określ, ile układów scalonych należy użyć do jego realizacji? Jakie to mogą być układy?
9. Zaprojektuj układ sterowania wskaźnikiem siedmiosegmentowym (o wspólnej anodzie). Wskaźnik ma wyświetlać cyfry w kodzie szesnastkowym.
10. Dysponując demultiplekserem o 2 wejściach adresowych, multiplekserem o 3 wejściach adresowych oraz dowolnymi bramkami, zbuduj układ kombinacyjny:
    $\mathbf{y} = \Sigma(4,5,6,7,10,12,13,15,18,19,20,21,24,26,29,31)$.
11. Rozwiąż zadanie 10. dla funkcji:
    $\mathbf{y} = \Sigma(0,1,2,3,4,5,12,14,16,18,19,22,23,25,29,31)$.
12. Rozwiąż zadanie 10. dla funkcji:
    $\mathbf{y} = \Sigma(0,1,3,7,8,9,13,15,20,22,26,27,28,29,30,31)$.
13. Dysponując demultiplekserem o 3 wejściach adresowych, multiplekserem o 2 wejściach adresowych oraz dowolnymi bramkami, rozwiąż zadanie z przykładu 10.6.
14. Dysponując demultiplekserem o 3 wejściach adresowych, multiplekserem o 2 wejściach adresowych oraz dowolnymi bramkami, rozwiąż zadanie 10.
15. Dysponując demultiplekserem o 3 wejściach adresowych, multiplekserem o 2 wejściach adresowych oraz dowolnymi bramkami, rozwiąż zadanie 11.
16. Dysponując demultiplekserem o 3 wejściach adresowych, multiplekserem o 2 wejściach adresowych oraz dowolnymi bramkami, rozwiąż zadanie 12.
17. Zbuduj z dwóch multiplekserów o dwóch wejściach adresowych jeden multiplekser o trzech wejściach adresowych.
18. Rozwiąż zadanie 17., ale w odniesieniu do demultiplekserów.

10.3. Przetworniki kodów

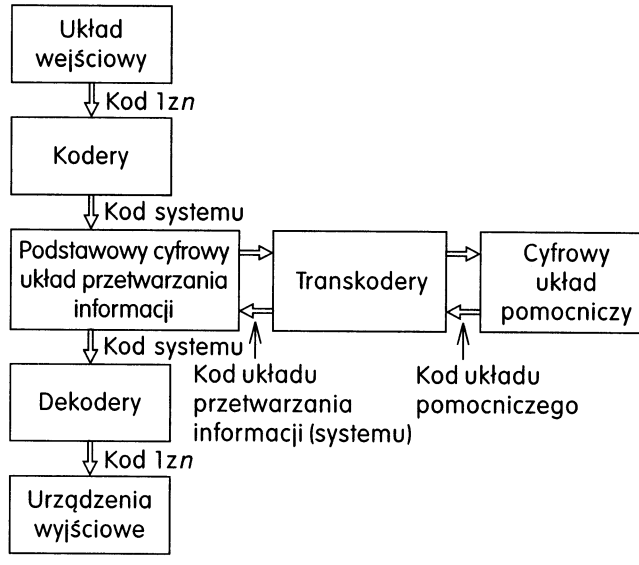
10.3.1. Pojęcia podstawowe

Komunikacja człowieka z systemem cyfrowym odbywa się najczęściej za pomocą klawiatury z przyciskami (np. kalkulator, nowoczesny telewizor, magnetowid, komputer). Oznacza to, że informacja wejściowa to: „jeden z przycisków wciśnięty", czyli ma ona postać słowa kodu **1 z *n*** (albo $\overline{\mathbf{1\,z\,}\boldsymbol{n}}$). Natomiast wewnątrz systemu cyfrowego są używane inne kody dwójkowe.

***Układy realizujące proces zamiany informacji kodowanej w kodzie* 1 z *n* *na kod wewnętrzny urządzenia (kod, w którym pracuje system) nazywamy* **koderami** *(lub* **enkoderami**). System cyfrowy (pracujący w jakimś kodzie) czasami współpracuje z innym systemem, w którym informacja jest kodowana za pomocą innego kodu wewnętrznego. *Układ umożliwiający współpracę takich systemów (czyli zamieniający jeden kod wewnętrzny na inny, z których żaden nie jest kodem typu* 1 z *n*) *nazywamy* **transkoderem.**

Informacja wyjściowa zwykle powinna mieć postać dostosowaną do urządzeń ją obrazujących lub sterujących. Dawniej, gdy używano wyświetlaczy neonowych (cyfr dziesiętnych) powinien to być kod **1 z 10**, przy sterowaniu *n* urządzeń wykonawczych w trybie „włącz — wyłącz" jest użyteczny kod **1 z *n***. Zwykle więc informacja wyjściowa miała postać kodu **1 z *n***. Dlatego ***układy zamieniające kod wewnętrzny na kod* 1 z *n* *nazwano* dekoderami**.

Obecnie kody wyjściowe są często inne niż kod **1 z *n*** i pojęcie dekodera jest szersze. Takim przykładem jest układ zamieniający kod **naturalny BCD** na kod **wskaźnika siedmiosegmentowego**. Mimo, że kodem wyjściowym nie jest tutaj kod **1 z *n***, to układ ten jest częściej nazywany dekoderem niż transkoderem. Ogólnie, ***układy zamiany kodów nazywa się* przetwornikami *lub* konwerterami kodów**. Na rysunku 10.18 pokazano usytuowanie poszczególnych rodzajów konwerterów w systemie cyfrowym.



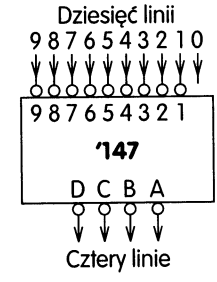
Rys. 10.18. Usytuowanie konwerterów kodu w systemie cyfrowym

Poniżej zostaną opisane wybrane układy konwerterów kodu wykonane w technice TTL lub CMOS. Ponieważ liczba bitów kodu wejściowego (liczba wejść), może różnić się od liczby bitów kodu wyjściowego (liczby wyjść), przeto często ***konwerter kodu określa się mianem* przetwornika typu „*n* linii na *m* linii"**, gdzie *n* oznacza liczbę bitów kodu wejściowego, a *m* liczbę bitów kodu wyjściowego.

10.3.2. Kodery

Układ scalony TTL typu **'147** jest koderem, którego kodem wejściowym może być kod $\overline{\mathbf{1z10}}$, a kodem wyjściowym jest kod **naturalny BCD**, przy czym wszystkie wyjścia są zanegowane. Inaczej można powiedzieć, że kodem wyjściowym jest kod naturalny BCD, ale przedstawiony w logice ujemnej (czyli takiej, w której

9 8 7 6 5 4 3 2 1	D C B A
1 1 1 1 1 1 1 1 1	1 1 1 1
1 1 1 1 1 1 1 1 0	1 1 1 0
1 1 1 1 1 1 1 0 -	1 1 0 1
1 1 1 1 1 1 0 - -	1 1 0 0
1 1 1 1 1 0 - - -	1 0 1 1
1 1 1 1 0 - - - -	1 0 1 0
1 1 1 0 - - - - -	1 0 0 1
1 1 0 - - - - - -	1 0 0 0
1 0 - - - - - - -	0 1 1 1
0 - - - - - - - -	0 1 1 0



Rys. 10.19. Koder scalony TTL '147: a) tablica działania; b) symbol graficzny

poziomowi niskiemu **L** przypisuje się logiczną wartość **1**). Jest on konwerterem typu „10 linii na 4 linie". Symbol graficzny oraz tablicę działania układu przedstawiono na rys. 10.19.

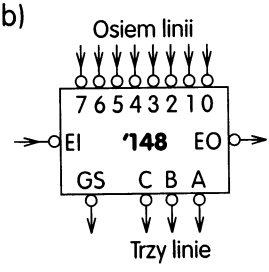
Normalna praca kodera to wyróżnienie jednego z jego wejść. W koderze **'147** wejście jest wyróżnione, gdy znajduje się w stanie niskim. Na wyjściu pojawia się słowo czterobitowe, które jest zapisem binarnym (w kodzie **naturalnym BCD**, logika ujemna) numeru wejścia, na którym jest poziom niski **L**. Układ działa jednoznacznie także wówczas, gdy jest wyróżnione więcej niż jedno wejście (stan **L** na kilku wejściach). Wyjście wskazuje wówczas najwyższy numer spośród wyróżnionych wejść. ***Mówimy, że wejście o wyższym numerze ma wyższy priorytet, a tak działające kodery nazywamy*** **koderami priorytetowymi**. Fizycznie koder **'147** ma 9, a nie 10 wejść, mimo że jest on układem typu „10 linii na 4 linie". Linia o numerze 0 nie musi być bowiem doprowadzona do układu. Jeżeli na dziewięciu liniach jest stan wysoki, to (przy kodzie wejściowym $\overline{1z10}$) na linii **0** (domyślnie) musi być stan **L**. Gdy oprócz **0** na linii zerowej jest jeszcze **0** na innej, wówczas o stanie wyjść decyduje i tak to inne wejście, bo jest to koder priorytetowy.

Znacznie bardziej wszechstronny jest koder priorytetowy **'148** typu „8 linii na 3 linie". Układ ten ma dodatkowo wejście **EI** (ang. *Enable Input*) oraz dwa wyjścia — **EO** (ang. *Enable Output*) i **GS** (ang. *Group Strobe*), które umożliwiają łączenie tych koderów w układy wielopoziomowe. Działanie logiczne układu oraz symbol graficzny pokazano na rys. 10.20.

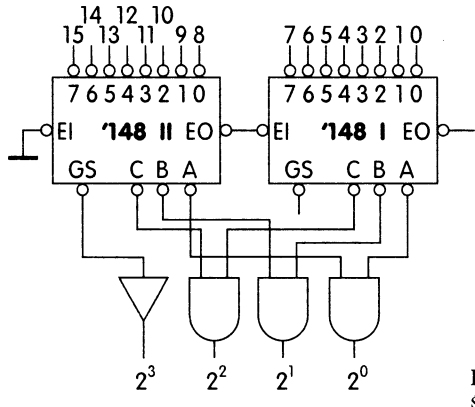
a)

EI	7 6 5 4 3 2 1 0	C B A	GS EO
0	1 1 1 1 1 1 1 0	1 1 1	0 1
0	1 1 1 1 1 1 0 -	1 1 0	0 1
0	1 1 1 1 1 0 - -	1 0 1	0 1
0	1 1 1 1 0 - - -	1 0 0	0 1
0	1 1 1 0 - - - -	0 1 1	0 1
0	1 1 0 - - - - -	0 1 0	0 1
0	1 0 - - - - - -	0 0 1	0 1
0	0 - - - - - - -	0 0 0	0 1
1	- - - - - - - -	1 1 1	1 1
0	1 1 1 1 1 1 1 1	1 1 1	1 0

b)



Rys. 10.20. Koder scalony TTL '148: a) tablica działania; b) symbol graficzny



Rys. 10.21. Koder priorytetowy o 16 liniach wejściowych zbudowany z koderów '148

Na rysunku 10.21 pokazano sposób połączenia dwóch układów **'148** w układ typu „16 linii na 4 linie".

Na rysunku 10.22 pokazano tablicę działania układu z rys. 10.21. Na jej podstawie można zauważyć, że wyjście $2^i = A_{i(II)}A_{i(I)}$ ($i = 0,1,2$) jest iloczynem logicznym i-tych wyjść obu układów, a wyjście 2^3 jest pobrane z wyjścia **GS** drugiego układu. Tak połączony układ nadal jest koderem priorytetowym. Załóżmy bowiem, że na liniach np. 6 i 11 jest poziom niski **L**. Poziom niski na linii 11 sprawia, że sygnał **EO** z drugiego układu ma poziom wysoki. Sygnał ten jest doprowadzony do wejścia **EI** pierwszego układu. Z tablicy działania układu **'148** (rys. 10.20) wynika, że przy **EI = 1** stan wejść nie ma wpływu na wyjścia, które są ustawiane w stan **HHH**, co odpowiada (pamiętamy, że są one zanegowane) liczbie binarnej **000**.

Układ:	drugi											pierwszy									II			I				
EI	15	.	.	.	.	.	.	8	GS	EO	=EI	7	6	5	4	3	2	1	0	C	B	A	C	B	A	2^3	2^2	2^3	2^2	
0	1	1	1	1	1	1	1	1	1	0		1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0		1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0		1	1	1	1	1	1	0	-	1	1	1	1	0	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	0		1	1	1	1	1	0	-	-	1	1	1	1	0	0	1	1	0	0	2
0	1	1	1	1	1	1	1	1	1	0		1	1	1	1	0	-	-	-	1	1	1	1	0	1	1	1	0	1	3
0	1	1	1	1	1	1	1	1	1	0		1	1	1	0	-	-	-	-	1	1	1	0	1	1	1	0	1	1	4
0	1	1	1	1	1	1	1	1	1	0		1	1	0	-	-	-	-	-	1	1	1	0	1	0	1	0	1	0	5
0	1	1	1	1	1	1	1	1	1	0		1	0	-	-	-	-	-	-	1	1	1	0	0	1	1	0	0	1	6
0	1	1	1	1	1	1	1	1	1	0		0	-	-	-	-	-	-	-	1	1	1	0	0	0	1	0	0	0	7
0	1	1	1	1	1	1	1	0	0	1		-	-	-	-	-	-	-	-	1	1	1	1	1	1	0	1	1	1	8
0	1	1	1	1	1	1	0	-	0	1		-	-	-	-	-	-	-	-	1	1	0	1	1	1	0	1	1	0	9
0	1	1	1	1	1	0	-	-	0	1		-	-	-	-	-	-	-	-	1	0	1	1	1	1	0	1	0	1	10
0	1	1	1	1	0	-	-	-	0	1		-	-	-	-	-	-	-	-	1	0	0	1	1	1	0	1	0	0	11
0	1	1	1	0	-	-	-	-	0	1		-	-	-	-	-	-	-	-	0	1	1	1	1	1	0	0	1	1	12
0	1	1	0	-	-	-	-	-	0	1		-	-	-	-	-	-	-	-	0	1	0	1	1	1	0	0	1	0	13
0	1	0	-	-	-	-	-	-	0	1		-	-	-	-	-	-	-	-	0	0	1	1	1	1	0	0	0	1	14
0	0	-	-	-	-	-	-	-	0	1		-	-	-	-	-	-	-	-	0	0	0	1	1	1	0	0	0	0	15

Rys. 10.22. Tablica opisująca działanie układu z rys. 10.21

10.3.3. Dekodery

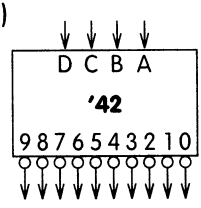
Zgodnie z klasyfikacją konwerterów kodów, dekoder zamienia jeden kod (inny niż **1 z *n***) na kod **1 z *n***. Jak już wspomniano powyżej, obecnie pojęcie to jest rozumiane szerzej i odnosi się do konwerterów używanych w obwodach wyjściowych układów cyfrowych. Omówione w podrozdziale 10.2 demultipleksery są także dekoderami kodu **dwójkowego naturalnego** na kod $\overline{\mathbf{1z}n}$ (przy stanie **L** na wejściach). Na przykład demultiplekser **'154** może być wykorzystany jako dekoder kodu dwójkowego naturalnego (czterobitowego) na kod $\overline{\mathbf{1z16}}$. Jeżeli do wejść adresowych doprowadzimy słowa w kodzie **BCD 8421** i wykorzystamy 10 pierwszych wyjść demultipleksera, to układ taki będzie pełnił rolę dekodera kodu **naturalnego BCD** na kod $\overline{\mathbf{1z10}}$.

W technice TTL jest produkowany dekoder kodu **naturalnego BCD** na kod $\overline{\mathbf{1z10}}$. Jest to układ **'42**. Na rysunku 10.23 przedstawiono symbol graficzny oraz tablicę działania tego układu.

a)

D C B A	9 8 7 6 5 4 3 2 1 0
0 0 0 0	1 1 1 1 1 1 1 1 1 0
0 0 0 1	1 1 1 1 1 1 1 1 0 1
0 0 1 0	1 1 1 1 1 1 1 0 1 1
0 0 1 1	1 1 1 1 1 1 0 1 1 1
0 1 0 0	1 1 1 1 1 0 1 1 1 1
0 1 0 1	1 1 1 1 0 1 1 1 1 1
0 1 1 0	1 1 1 0 1 1 1 1 1 1
0 1 1 1	1 1 0 1 1 1 1 1 1 1
1 0 0 0	1 0 1 1 1 1 1 1 1 1
1 0 0 1	0 1 1 1 1 1 1 1 1 1
1 0 1 0	
1 0 1 1	Na wszystkich wyjściach stan wysoki; stany wejść niedozwolone
1 1 0 0	
1 1 0 1	
1 1 1 0	
1 1 1 1	

b)

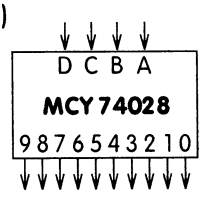


Rys. 10.23. Dekoder '42 kodu naturalnego BCD na kod $\overline{1z10}$: a) tablica prawdy; b) symbol graficzny

a)

D C B A	9 8 7 6 5 4 3 2 1 0
0 0 0 0	0 0 0 0 0 0 0 0 0 1
0 0 0 1	0 0 0 0 0 0 0 0 1 0
0 0 1 0	0 0 0 0 0 0 0 1 0 0
0 0 1 1	0 0 0 0 0 0 1 0 0 0
0 1 0 0	0 0 0 0 0 1 0 0 0 0
0 1 0 1	0 0 0 0 1 0 0 0 0 0
0 1 1 0	0 0 0 1 0 0 0 0 0 0
0 1 1 1	0 0 1 0 0 0 0 0 0 0
1 0 0 0	0 1 0 0 0 0 0 0 0 0
1 0 0 1	1 0 0 0 0 0 0 0 0 0
1 0 1 0	
1 0 1 1	Na wszystkich wyjściach stan niski; stany wejść niedozwolone
1 1 0 0	
1 1 0 1	
1 1 1 0	
1 1 1 1	

b)



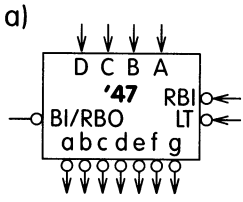
Rys. 10.24. Dekoder MCY74028 kodu naturalnego BCD na kod 1 z 10: a) tablica prawdy; b) symbol graficzny

Układ scalony CMOS MCY74028 zawiera dekoder kodu **naturalnego BCD** na kod **1 z 10**. Jego działanie różni się tylko tym od układu **'42**, że wyjście wyróżnione ma stan wysoki, a pozostałe wyjścia są w stanie niskim, co znalazło swój wyraz w postaci symboli graficznych. Jego tablicę prawdy oraz symbol graficzny pokazano na rys. 10.24.

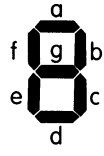
10.3.4. Transkodery

Najczęściej stosowanym transkoderem jest układ **'47** zamieniający kod **BCD 8421** na kod **wskaźnika siedmiosegmentowego**. Jak już wspomniano wcześniej, częściej jest on nazywany dekoderem. Tablicę działania układu oraz jego symbol graficzny przedstawiono na rys. 10.25.

a)



b)



c)

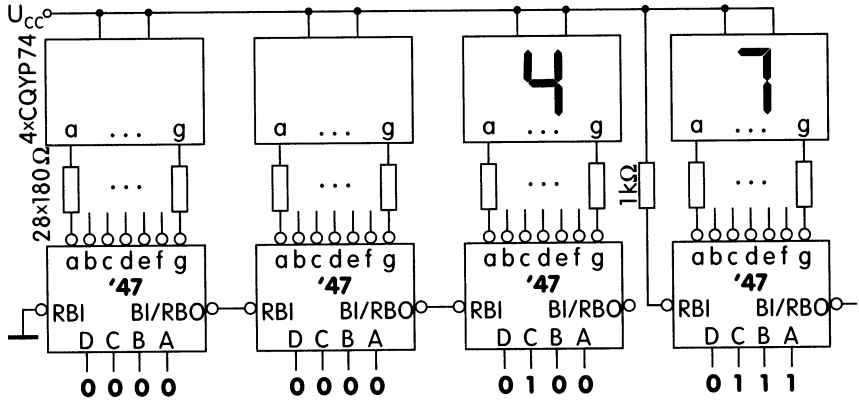
Wejścia						Wyjścia							Liczba dziesiętna lub realizowana funkcja
LT	RBI	D	C	B	A	BI/RBO	a	b	c	d	e	f	g	
1	1	0	0	0	0	1	0	0	0	0	0	0	1	0
1	-	0	0	0	1	1	1	0	0	1	1	1	1	1
1	-	0	0	1	0	1	0	0	1	0	0	1	0	2
1	-	0	0	1	1	1	0	0	0	0	1	1	0	3
1	-	0	1	0	0	1	1	0	0	1	1	0	0	4
1	-	0	1	0	1	1	0	1	0	0	1	0	0	5
1	-	0	1	1	0	1	1	1	0	0	0	0	0	6
1	-	0	1	1	1	1	0	0	0	1	1	1	1	7
1	-	1	0	0	0	1	0	0	0	0	0	0	0	8
1	-	1	0	0	1	1	0	0	0	1	1	0	0	9
1	-	1	0	1	0	1	1	1	1	0	0	1	0	(10)
1	-	1	0	1	1	1	1	1	0	0	1	1	0	(11)
1	-	1	1	0	0	1	1	0	1	1	1	0	0	(12)
1	-	1	1	0	1	1	0	1	1	0	1	0	0	(13)
1	-	1	1	1	0	1	1	1	1	0	0	0	0	(14)
1	-	1	1	1	1	1	1	1	1	1	1	1	1	(15)
-	-	-	-	-	-	0	1	1	1	1	1	1	1	wygaszenie wskaź.
1	0	0	0	0	0	0	1	1	1	1	1	1	1	wygaszanie zera
0	-	-	-	-	-	1	0	0	0	0	0	0	0	test wskaźnika

Rys. 10.25. Transkoder '47: a) symbol graficzny; b) oznaczenie segmentów; c) tablica działania

Układ **'47** jest przeznaczony do sterowania wskaźnika 7-segmentowego ze wspólną anodą (anody wszystkich diod w takim wskaźniku są ze sobą połączone i stanowią jedną końcówkę). Wyjścia dekodera są wysokonapięciowymi (do 15 V) wyjściami typu OC (otwarty obwód kolektora). Każdy tranzystor wyjściowy może przełączać prąd o maksymalnej wartości 40 mA. Rezystory ograniczające prąd włącza się pomiędzy wyjścia układu a segmenty (katody) wskaźnika. Dobór rezystancji został omówiony w p. 9.5.1. W przypadku gdy słowo wejściowe przybiera wartość zabronioną (z przedziału od 10 do 15), wówczas na wskaźniku jest wyświetlany symbol nie odpowiadający żadnej cyfrze dziesiętnej.

Dodatkowo układ scalony **'47** ma dwa wejścia oznaczone jako **LT** i **RBI** oraz wyprowadzenie oznaczone **BI/RBO**, które może być wykorzystywane zarówno

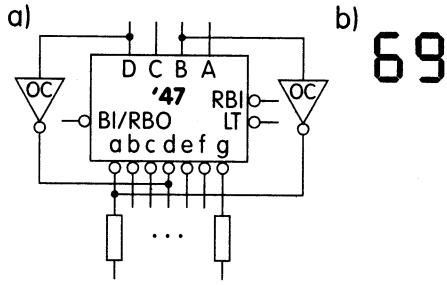
jako wejście, jak i wyjście. Wejście **LT** (ang. *Lamp Test* — kontrola świecenia) pozwala przy stanie **L** skontrolować świecenie wszystkich segmentów wskaźnika — wyjścia dekodera są w stanie niskim i jeżeli segmenty są sprawne, to jest wyświetlana cyfra 8 (rys. 10.25b). W czasie normalnej pracy na wejściu tym powinien być poziom wysoki **H**. Wejście **RBI** (ang. *Ripple Blanking Input* — kaskadowe wejście wygaszające) służy do wygaszenia wskaźnika, jeżeli ten wskazuje 0. Przyjmijmy, że na 4-cyfrowym polu odczytowym jest wyświetlana liczba 0047. Dwa pierwsze zera są **zerami nieznaczącymi** i można (a czasami nawet trzeba) je wygasić. Mniej męczące wzrok człowieka jest odczytywanie wskazań z wygaszonymi zerami nieznaczącymi. W sytuacji, gdy wskaźnik jest oddalony (nastawnia, dyspozytornia), takie nieznaczące zero może prowadzić do błędnych odczytów (może zostać odczytane jako 8). Oczywiście dopuszczalne jest wygaszenie tylko zer nieznaczących, czyli takich, względem których na starszych pozycjach nie ma innych cyfr niż zero. Na rysunku 10.26 pokazano układ wyświetlacza 4-polowego z wygaszaniem zer nieznaczących.



Rys. 10.26. Wyświetlacz 4-polowy z wygaszaniem zer nieznaczących

Układ wygaszania działa następująco. Jeżeli poziom logiczny na wejściu **RBI** ma wartość **0** i słowo wejściowe podane na wejścia **DCBA** odpowiada cyfrze 0, to taki segment zostaje wygaszony. Jednocześnie na wyjściu **RBO** (ang. *Ripple Blanking Output* — kaskadowe wyjście wygaszające) jest ustawiany stan niski **L**. Wyjście to dołączone jest do wejścia **RBI** kolejnego dekodera (na pozycji mniej znaczącej). Jeżeli i na jego wejściach **DCBA** są same **0**, to i ten kolejny wskaźnik zostanie wygaszony. Jeżeli jednak słowo wejściowe jest niezerowe, to wygaszenie nie nastąpi, a wyjście **RBO** tego transkodera zostanie ustawione w stan **1**. Nie pozwoli to na wygaszenie następnego zera, o ile pojawi się takie na młodszej pozycji. Wejście **RBI** dekodera najmłodszej cyfry dołącza się do poziomu wysokiego **H**, aby nie wygasić całego pola odczytowego przy wyświetlaniu liczby 0.

Wyjście **RBO** może zostać wykorzystane jako wejście **BI** (ang. *Blanking Input* — wejście wygaszające), jeżeli doprowadzimy do niego sygnał zewnętrzny o poziomie niskim **L**. Wygaszenie następuje niezależnie od innych sygnałów wej-



Rys. 10.27. Układ do poprawy kształtu cyfr (a) oraz (b) kształt poprawionych cyfr

ściowych. Sterując dekoder z tego wejścia (**BI/RBO**) falą prostokątną o regulowanym współczynniku wypełnienia, możemy regulować jaskrawość świecenia wskaźnika. Wskaźnik świeci dopóty, dopóki na wejściu **BI/RBO** jest poziom wysoki. Przez pozostałą część okresu przebiegu prostokątnego sterującego wskaźnik jest on wygaszany. Celem takiego sposobu postępowania jest dostosowanie jaskrawości świecenia wskaźników do zewnętrznych warunków oświetlenia. Ponadto uzyskuje się także zmniejszenie poboru prądu przez wskaźnik. Ma to istotne znaczenie w cyfrowych przyrządach przenośnych, zasilanych z baterii lub akumulatorów.

Istnieje także inny sposób ograniczenia poboru prądu przez wskaźnik. Polega on na tzw. **multipleksowaniu**. Upraszczając w dużym stopniu, można powiedzieć, że anody kolejnych wskaźnikow 7-segmentowych są zasilane cyklicznie tak, że w danej chwili świeci tylko jeden wskaźnik. Oczywiście szybkość przełączeń musi być na tyle duża, aby dzięki bezwładności diody i oka ludzkiego uzyskać wrażenie świecenia ciągłego. Rozbudowując układ sterowania wyświetlania multipleksowego, można zastosować tylko jeden dekoder. Jego wejścia muszą być dołączane do kolejnych cyfr, jakie mają być wyświetlane i synchronicznie z dołączeniem kolejnej cyfry powinien być zasilany odpowiedni wskaźnik.

Niekiedy dla poprawienia czytelności cyfr 6 i 9 stosuje się modyfikacje układowe, dzięki którym kształty tych cyfr przybierają taką postać jak na rys. 10.27b.

Dołączone dodatkowo bramki NOT (rys. 10.27a) powinny mieć wyjście typu otwarty kolektor (OC), np. **'06** lub **'16**. W układzie jak na rys. 10.27 niemożliwe jest jednak stosowanie wygaszania wskaźnika od strony wejścia **BI/RBO**. Aby zachować taką możliwość, należy zamiast bramek NOT użyć dwuwejściowe bramki NAND, łącząc drugie końcówki tych bramek z wyprowadzeniem **BI/RBO**.

Nowsze wersje dekoderów kodu **naturalnego BCD** na kod **wskaźnika siedmiosegmentowego** (układy typu **'246**, **'247**) nie mają już tej wady i kształty cyfr uzyskiwane przy ich stosowaniu są takie jak na rys. 10.27b.

Pytania i zadania

1. Zbuduj z bramek NAND dekoder służący do konwersji dwubitowego kodu dwójkowego na kod **1 z 4**.
2. Zbuduj z bramek NOR dekoder służący do konwersji dwubitowego kodu dwójkowego na kod **1 z 4**.

3. Zbuduj z układów **'42** (dekoder typu 4 linie na 10 linii) demultiplekser o 4 wejściach adresowych.
4. Narysuj układ z dekoderem **'47**, w którym uzyskuje się poprawienie kształtu cyfr 6 i 9 oraz istnieje możliwość wygaszania wskaźnika od strony wejścia **BI/RBO**.
5. Zaprojektuj transkoder kodu **naturalnego BCD** na kod **wskaźnika 7-segmentowego,** korzystając z demultipleksera. Stany wejść odpowiadające liczbom od 10 do 15 powinny powodować wyświetlanie symbolu E (ang. *Error* — błąd).
6. Co nazywamy koderem (enkoderem)?
7. Co to jest transkoder?
8. Co nazywamy dekoderem?
9. Co to jest koder priorytetowy?
10. Jakie jest miejsce poszczególnych konwerterów kodów w układach cyfrowych?

11

Układy arytmetyczne

11.1. Wprowadzenie

***Cyfrowe układy scalone umożliwiające realizację podstawowych działań arytmetycznych, takich jak: dodawanie, odejmowanie, mnożenie i dzielenie — nazywa się* układami arytmetycznymi.**

Podstawowym układem arytmetycznym jest **sumator**, który dodaje dwie liczby dwójkowe. Za pośrednictwem operacji dodawania wielokrotnie powtórzonej jest dokonywane mnożenie, a przy zastosowaniu dodatkowych przekształceń również odejmowanie i dzielenie. Do układów arytmetycznych zaliczamy także: **multiplikatory** — realizujące mnożenie, **komparatory** — porównujące liczby dwójkowe i układy kontroli parzystości.

Uniwersalną grupę układów arytmetycznych stanowią **arytmometry**, czyli **jednostki arytmetyczno-logiczne** (**ALU** — ang. *Arithmetic Logic Unit*). Mogą one realizować nie tylko operacje arytmetyczne, ale również logiczne. Arytmometr to także jeden z podstawowych bloków mikroprocesora.

Przykładem sumatora może być układ **'83**, komparatora — układ **'85**, a arytmometru układ — **'181**, będący czterobitową jednostką arytmetyczno-logiczną.

Liczby, jak i każda inna informacja, są w układach cyfrowych reprezentowane przez słowa binarne (zero-jedynkowe). W rozdziale 1. podręcznika poznaliśmy sposób konwersji zapisu dziesiętnego liczb na zapis dwójkowy. Zapis taki nazywa się **naturalnym kodem dwójkowym** . Działania arytmetyczne wykonywane w tym zapisie realizowaliśmy analogicznie jak w systemie dziesiętnym zapisu liczb. Poznaliśmy także kody **BCD**, umożliwiające łatwą konwersję zapisu dziesiętnego na dwójkowy oraz dwójkowego na dziesiętny. Wiadomości te oczywiście nie wyczerpują wszystkich problemów związanych z realizacją operacji arytmetycznych w systemach cyfrowych. Informacje te obecnie zostaną rozszerzone, w szczególności o metody zapisu liczb ze znakiem.

Przy omawianiu operacji arytmetycznych musimy założyć określoną i jednakową dla wszystkich liczb długość ich reprezentacji binarnej. Jest to uwarun-

kowane technicznie (konkretny układ cyfrowy zawsze ma określoną długość przetwarzanych słów binarnych) i pociąga za sobą określone skutki, wymagające właściwego postępowania.

UWAGA
W tym podrozdziale (11.2) oraz następnym (11.3) operator „+" będzie oznaczał sumowanie algebraiczne, a nie logiczne.

11.2. Metody zapisu liczb ze znakiem

11.2.1. Zapis „znak-moduł" (ZM)

Najprostszym sposobem kodowania liczb z uwzględnieniem ich znaku jest zapis **znak-moduł**. Jest to **naturalny kod dwójkowy** uzupełniony o dodatkowy bit — bit znaku. ***Przyjęto oznaczać liczbę dodatnią bitem znaku o wartości* 0, *liczbę ujemną bitem znaku* 1**. W przykładach poniżej bit ten będziemy oddzielać od reszty zapisu (modułu) kropką — dla ustalenia uwagi Czytelnika. W technicznej realizacji nic takiego nie występuje, a bit znaku ma tylko określoną pozycję. Na przykład

Liczba dziesiętna	Zapis **ZM** (znak-moduł)
9	**0.1001**
−9	**1.1001**

Zapis ten jednak nie daje poprawnych wyników przy wykonywaniu odejmowania wówczas, gdy odjemnik jest większy od odjemnej. Także zastąpienie odejmowania dodawaniem prowadzi do niewłaściwych wyników. Na przykład

7	**0.0111**	7	**0.0111**
−9	**−0.1001**	+(−9)	**+1.1001**
−2	(1) **1.0110**	−2	(1) **0.0000**

W obu przypadkach wynik jest niewłaściwy, gdyż powinien on mieć postać **1.0010**. W przykładzie odejmowania na pozycji bitu znaku pojawiła się **1** jako wynik pożyczki.

Należy zwrócić uwagę, że zapis **znak-moduł** nie pozwala zastąpić odejmowania dodawaniem. Wyników otrzymywanych w tym zapisie nie można jednoznacznie interpretować.

11.2.2. Zapis „znak-uzupełnienie do 1" (U1)

Zapis **U1** jest także dwuczęściowy. Zapis liczby dodatniej niczym nie różni się od zapisu w metodzie **ZM**. Dla liczb ujemnych (**BZ = 1**) zapis w kodzie **U1** uzyskuje się, negując każdy bit reprezentacji binarnej modułu zapisanego w kodzie naturalnym. Na przykład

Liczba dziesiętna	Zapis **ZM**	Zapis **U1**
9	**0.1001**	**0.1001**
–9	**1.1001**	**1.0110**

Działania na liczbach przedstawionych w kodzie **U1** wykonuje się łącznie z bitem znaku. Wykonajmy w zapisie **U1** wyżej zrealizowane obliczenia (w zapisie **ZM**) oraz dodatkowo działanie 9 + (–3)

7	**0.0111**	7	**0.0111**	9	**0.1001**
–9	**–0.1001**	+(–9)	**+1.0110**	+(–3)	**+1.1100**
–2	**(1)1.1110**	–2	**1.1101**	6	**(1)0.0101**
korekcja	↳ **–1**			korekcja	↳ **+1**
	1.1101				**0.0110**

W zapisie **znak-uzupełnienie do 1** (**U1**) wynik uzyskuje się zawsze w zapisie **U1**. Jednak gdy (po wykonaniu działań) pojawi się przed bitem znaku jedynka (tzw. **pożyczka cykliczna** lub **przeniesienie**), wówczas należy przeprowadzić korekcję polegającą na przesunięciu jej na pozycję najmniej znaczącą i ***powtórzeniu działania*** (tzn. jeśli było nim dodawanie, to należy dodać; jeśli odejmowanie, to odjąć). Poza tym zero ma w tym sposobie kodowania podwójną reprezentację +0 (**0.000**) lub –0 (**1.1111**), co jest dość istotną niedogodnością tej reprezentacji liczb. Z tych dwóch powodów znacznie częściej korzysta się z zapisu **znak-uzupełnienie do dwóch** (**U2**).

11.2.3. Zapis „znak-uzupełnienie do 2" (U2)

Zapis **U2** liczby dodatniej niczym nie różni się od zapisu w metodzie **ZM** (czy **U1**). Dla liczb ujemnych (**BZ** = **1**) przedstawia natomiast dopełnienie modułu tej liczby do wartości 2^n, gdzie n jest pozycją bitu (**uwaga:** pozycje numerujemy rozpoczynając od pozycji o numerze 0). Wartość liczby w zapisie **U2** można interpretować jako $-\mathbf{BZ} \cdot 2^n + D$ (gdzie: **BZ** — bit znaku, n — numer pozycji bitu znaku, D — dopełnienie modułu do liczby 2^n). Zauważmy, że powyższa uwaga jest prawdziwa także dla liczb dodatnich. Wówczas **BZ** = **0**, a dopełnienie D jest równe modułowi M.

Praktycznie, zapis liczby ujemnej w kodzie **U2** uzyskuje się, negując każdy bit reprezentacji binarnej modułu zapisanego w kodzie naturalnym, a następnie dodając liczbę 1. Inaczej mówiąc, zapis **U2** liczby ujemnej uzyskujemy dodając do jej zapisu **U1** liczbę 1. Na przykład

Liczba dziesiętna	Zapis **ZM**	Zapis **U1**	Zapis **U2**
9	**0.1001**	**0.1001**	**0.1001**
–9	**1.1001**	**1.0110**	**1.0111**

Sprawdźmy zależność $-\mathbf{BZ} \cdot 2^n + D$ dla liczby –9. Otrzymamy:
$-2^4 + 7 = -16 + 7 = -9$.

Wykonując w zapisie **U2** powyżej realizowane operacje arytmetyczne, otrzymamy:

7	**0.0111**	7	**0.0111**	9	**0.1001**
−9	**−0.1001**	+(−9)	**+1.0111**	+(−3)	**+1.1101**
−2	**1.1110**	−2	**1.1110**	6	**0.0110**

Zauważmy, że wszystkie otrzymane wyniki mają postać zapisu **U2**. Nie występuje także potrzeba wprowadzania korekcji. Z tego punktu widzenia ta metoda zapisu jest najkorzystniejsza. Jedyną jej wadą jest względnie złożony proces uzyskiwania zapisu, chociaż nie stanowi on istotnego problemu technicznego.

Zestawiając wady i zalety opisanych metod zapisu, możemy je porównać, co ułatwi wybór metody odpowiedniej do konkretnego zastosowania.

1. W zapisie modułowym (**ZM**) liczby występują w najprostszej postaci, ale niezbędna jest realizacja układowa zarówno dodawania, jak i odejmowania. Pewne trudności może sprawiać określenie znaku wyniku. Wynik nie zawsze ma postać zapisu **ZM**.
2. W zapisie **U1** generowanie reprezentacji liczby ujemnej jest relatywnie proste, ale zachodzi konieczność przeprowadzania korekcji w przypadku wystąpienia pożyczki lub przeniesienia. Pewne kłopoty sprawiać też będzie podwójna reprezentacja zera.
3. W zapisie **U2** wszystkie działania arytmetyczne są proste, ale samo utworzenie zapisu liczby ujemnej wymaga pewnych dodatkowych zabiegów.

11.2.4. Zapis „znak-uzupełnienie do 9(10)"

Jak wiemy, w urządzeniach cyfrowych wykonujących operacje arytmetyczne dogodnie jest używać zapisu **dwójkowo-dziesiętnego** (kodów **BCD**). Kody te — podobnie jak naturalne kody dwójkowe — mogą być przedstawiane w trzech podstawowych zapisach:

— znak-moduł (**ZM**),
— znak-uzupełnienie do 9 (**U9**),
— znak-uzupełnienie do 10 (**U10**).

Zapisy te są analogiczne (jeśli chodzi o ich wady i zalety) do opisanych powyżej metod zapisu **ZM**, **U1** i **U2**. We wszystkich tych zapisach liczby dodatnie są przedstawiane w taki sam sposób, ujemne zaś tworzy się na podstawie modułu.

***Aby uzyskać uzupełnienie do* 9 (U9), *należy każdą cyfrę modułu odjąć od* 9. *Uzupełnienie do* 10 (U10) *jest większe o* 1 *na najmniej znaczącej pozycji od* U9.

Aby otrzymać uzupełnienie do 10, należy określić uzupełnienie do 9 i dodać na najmniej znaczącej pozycji liczbę 1. Można je uzyskać także w inny sposób, mianowicie odejmując najmniej znaczącą pozycję modułu od liczby 10, a pozostałe pozycje od liczby 9. Każda cyfra dziesiętna w dowolnym z tych trzech zapisów jest oczywiście kodowana za pomocą jednego z kodów **BCD**.

Dla uproszczenia zapisy te przedstawmy przy użyciu cyfr dziesiętnych, bez przechodzenia na reprezentacje dwójkowe. Przykładowe zapisy będą więc miały następującą postać

Liczba dziesiętna	Zapis **U9**	Zapis **U10**
+47	**0**.47	**0**.47
−47	**1**.52	**1**.53

Przykład 10.1

Dane są dwie liczby zapisane w kodzie dziesiętnym naturalnym (**BCD 8421**). Wykonać odejmowanie i dodawanie na tak zapisanych liczbach. Działania wykonać w zapisie **U10.**

Oczywiście odejmowanie zrealizujemy jako dodanie liczby ujemnej.

Przykład A	w **U10**	w **U10** po zakodowaniu
15	**0**.15	**0.** **0001** **0101**
+32	+**0**.32	+**0.** **0011** **0010**
47	**0**.47	**0.** **0100** **0111**

Przykład B		w **U10**	w **U10** po zakodowaniu	
46	46	**0**.46	**0.** **0100** **0110**	
−28 czyli	+(−28)	+**1**.72	+ **1.** **0111** **0010**	
18	18	**0**.18	**1.** **1011** **1000**	
			+ **0110** **0000**	korekcja dziesiętna
			(1) **0.** **0001** **1000**	

na tych pozycjach dodawane są bity

W przypadku, gdy wynik nie jest liczbą w kodzie **naturalnym BCD**, przeprowadza się tzw. korekcję dziesiętną. Polega ona na dodaniu do wyniku 6, 60 lub 600 (itd.) w zależności od tego, która cyfra jest błędnie przedstawiona (cyfra jedności — dodajemy 6, cyfra dziesiątek — dodajemy 60, itd.). Przeniesienie cykliczne, pojawiające się przed bitem znaku w zapisie **U10**, pomijamy.

Sprawdzenie, czy w wyniku odejmowania liczby 28 od liczby 46 otrzymamy identyczny wynik, pozostawia się Czytelnikowi. Tutaj natomiast wykonajmy jeszcze odejmowanie liczby 46 od liczby 28.

Przykład C	w **U10**	w **U10** po zakodowaniu	
28	**0**.28	**0.** **0010** **1000**	
−46	−**0**.46	−**0.** **0100** **0110**	
−18	**1**.82	**1.** **1110** **0010**	
		− **0110** **0000**	korekcja dziesiętna
		1. **1000** **0010**	

Z przykładu tego widać, że korekcja dziesiętna polegała tutaj na ***odjęciu*** liczby 60 od uzyskanego wyniku. Uogólniając powiemy, że korekcja dziesiętna jest wykonywana przy użyciu ***takiej samej operacji, jak samo działanie*** (tzn. jeżeli wykonywaliśmy dodawanie, to należy dodać; jeżeli odejmowanie, to należy odjąć).

Interpretując zaś wynik, należy zauważyć, że na pozycji bitu znaku pojawiła się cyfra **1**, która wskazuje, że wynik jest liczbą ujemną. Przedstawiona jest ona w zapisie **U10**. Aby otrzymać moduł tej liczby, należy postąpić identycznie jak przy tworzeniu zapisu **U10** dla liczby ujemnej. Czyli

1000		**0010**	
⇓		⇓	
8		2	
⇓		⇓	
1	(bo 9 – 8)	8	(bo 10 – 2)

Sprawdzenie, czy zastąpienie w tym działaniu odejmowania dodawaniem (28 + (–46)) doprowadzi do takiego samego wyniku, pozostawia się Czytelnikowi. ■

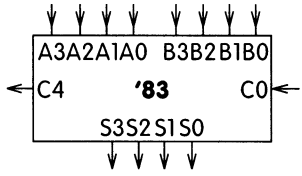
Pytania i zadania

1. Wykonaj następujące działania w zapisie **U1** oraz **U2**, posługując się jedynie operacją sumowania:
   a) 13 + 7, 14 – 8, 6 – 9;
   b) 15 + 11, –13 + 7, –12 – 13;
   c) 25 – 30, 19 + 21, –17 – 9.
2. Wykonaj następujące działania w zapisie **U9** oraz **U10** posługując się jedynie operacją dodawania:
   a) 25 + 48, 37 – 68, –49 – 21, –24 – 71
   b) 45 – 90, 8 + 9, –20 – 50, 38 – 27
   c) 87 + 91, 78 – 35, 68 – 93
3. Dane są cztery bity liczby wejściowej A = **A3 A2 A1 A0** oraz bit znaku **BZ**. Zaprojektuj układ, na wyjściu którego będzie liczba wejściowa w zapisie **U1**.
4. Dane są cztery bity liczby wejściowej A = **A3 A2 A1 A0** oraz bit znaku **BZ**. Zaprojektuj układ, na wyjściu którego będzie liczba wejściowa w zapisie **U2**. Zadanie rozwiąż jako iteracyjne. Wykorzystaj rozwiązanie uzyskane w zadaniu 3.
5. Jakie znasz metody zapisu liczb ze znakiem? Jakie są ich wady i zalety?

11.3. Sumatory

11.3.1. Sumator równoległy

Podstawowym układem arytmetycznym jest sumator, służący najczęściej do wykonywania operacji dodawania. W podrozdziale 3.6 projektowaliśmy sumator jako układ iteracyjny. Sumator taki sumował dwie liczby n-bitowe, jeżeli użyliśmy do jego budowy n elementarnych sumatorów połączonych kaskadowo (rys. 3.45).



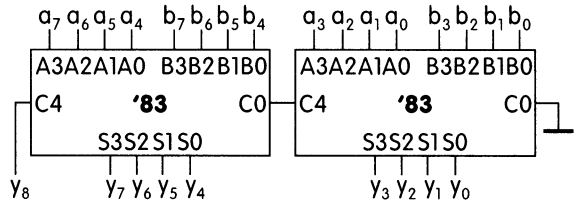
Rys. 11.1. Symbol graficzny sumatora scalonego '83

Monolityczny układ scalony **'83** (rys. 11.1) zawiera cztery takie bloki (sumatory elementarne), dzięki czemu od strony wejść i wyjść jest on **czterobitowym sumatorem dwójkowym równoległym.** Nazwa ta oznacza, że sumuje on liczby czterobitowe dwójkowe (A = **A3 A2 A1 A0**; B = **B3 B2 B1 B0**) podawane jednocześnie na jego wejścia. Układ ma jedno wejście przeniesienia **C0** i wyjście przeniesienia **C4**, co umożliwia łączenie ze sobą kilku takich sumatorów. Można wtedy sumować liczby 8-, 16-bitowe, itd.

Działanie sumatora można zapisać w sposób następujący:

	A3	**A2**	**A1**	**A0**
	B3	**B2**	**B1**	**B0**
+				**C0**
C4	**S3**	**S2**	**S1**	**S0**

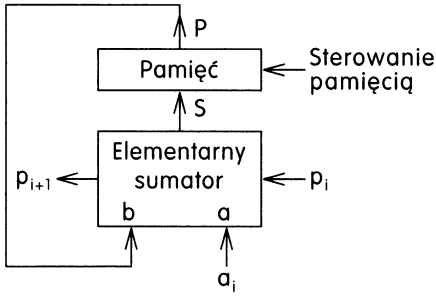
Sumator dodający dwie liczby 8-bitowe (lub $k \cdot$ 4-bitowe) można bardzo łatwo zbudować z 2 (k) sumatorów 4-bitowych. Układ taki przedstawiono na rys. 11.2.



Rys. 11.2. Sumator liczb 8-bitowych Y = A + B, gdzie:
$A = a_7a_6a_5a_4a_3a_2a_1a_0$, $B = b_7b_6b_5b_4b_3b_2b_1b_0$,
$Y = y_8y_7y_6y_5y_4y_3y_2y_1y_0$

11.3.2. Sumator szeregowy — akumulator

Sumator elementarny przedstawiony w p. 3.6 możemy wykorzystać do sumowania w sposób całkowicie odmienny od powyżej omówionego „równoległego", w którym oba składniki są jednocześnie obecne na wejściach sumatora. W tym celu dodajmy jeszcze element pamięciowy (np. przerzutnik) i przeanalizujmy działanie układu przedstawionego na rys. 11.3.



Rys. 11.3. Schemat funkcjonalny (uproszczony) sumatora elementarnego w układzie akumulatora

W sumatorze jak na rys. 11.3 dodanie dwóch bitów (np. **x** + **y**) będzie realizowane wg następującego podstawowego cyklu pracy:

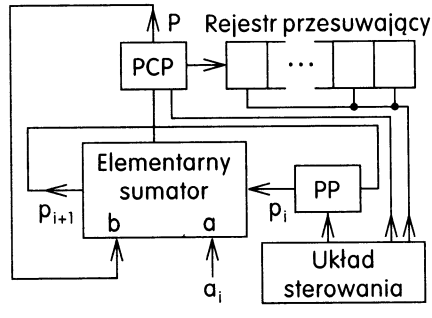
1. Zerowanie pamięci (przerzutnika).
2. Ustawienie na wejściu **a** pierwszego składnika sumy — **x**.
3. Zapamiętanie wyniku sumowania **S** = **x** + **b** = **x** (stan wejścia **a** jest sumowany ze stanem wejścia **b**, które teraz jest równe **0**, ponieważ **b** = **P**, a pamięć została wyzerowana). Wynik **S** zostaje zapamiętany i w efekcie wyjście **P** = **x**, co z kolei sprawia, że od tej chwili wejście sumatora **b** = **P** = **x**.
4. Ustawienie na wejściu **a** kolejnego składnika sumy — **y**.
5. Zapamiętanie wyniku sumowania, obecnie **P** = **S** = **x** + **y**.

Cały taki podstawowy cykl pracy jest równoważny jednej operacji sumatora elementarnego (pracującego w układzie równoległym), czyli polega na dodaniu dwóch bitów i określeniu wyniku sumowania oraz bitu przeniesienia.

***W takim sumatorze dodawane składniki są podawane na wejście kolejno jeden po drugim, czyli szeregowo, stąd nazwa tego sumatora* — sumator szeregowy.**

W porównaniu z sumatorem równoległym układ sumatora szeregowego jest o wiele bardziej skomplikowany (pamięć, układ sterowania). Jakie więc ma on zalety w porównaniu z sumatorem równoległym?

Otóż rozbudujmy pamięć w taki sposób (rys. 11.4), aby kolejne wyniki dodawania (czyli wyniki uzyskiwane na zakończenie każdego podstawowego cyklu pracy) były zapamiętywane w kolejnych przerzutnikach. Wówczas — nie zmieniając struktury samego sumatora elementarnego — możemy dodawać dowolne



Rys. 11.4. Schemat funkcjonalny sumatora elementarnego w układzie akumulatora
PCP — pamięć cyklu podstawowego, PP — pamięć przeniesienia

liczby n-bitowe, o ile użyjemy odpowiednią liczbę przerzutników i wykonamy n cykli podstawowych składających się na pełny cykl pracy. W każdym podstawowym cyklu pracy sumatora szeregowego dodajemy dwa bity z kolejnych pozycji sumowanych liczb.

Zespół przerzutników realizujących pamięć nazywamy **rejestrem** (patrz rozdz. 13). W rzeczywistości kolejne wyniki nie są zapamiętywane w kolejnych przerzutnikach, lecz używa się tzw. **rejestrów przesuwających**, do których informację wprowadza się stale do tego samego przerzutnika, ale każde takie wpisanie przesuwa całą informację w rejestrze o jedną pozycję.

Dodatkowy przerzutnik trzeba przeznaczyć na zapamiętywanie bitu przeniesienia, a wyjście tego przerzutnika należy połączyć z wejściem przeniesienia sumatora elementarnego. Zauważmy, że mając układ jednowejściowy, możemy sumować liczby n-bitowe. Zmiana długości sumowanych liczb będzie wymagać jedynie wymiany rejestru, a podstawowy układ i podstawowy cykl pracy nie ulegnie zmianie. Jest to niewątpliwą zaletą tego układu.

Zamiast sumatora elementarnego możemy użyć pełnego sumatora (np. **'83**) i dodawać w każdym cyklu liczby czterobitowe — wówczas wejścia **a** i **b** byłyby wejściami czterobitowymi, pamięć cyklu podstawowego (PCP) byłaby czterobitowa i także czterobitowe musiałyby być komórki rejestru przesuwającego.

Dodawane liczby mogą być w kodzie **naturalnym BCD** i jeżeli sumator będzie sumatorem dziesiętnym (np. sumator **'83** z układem korekcji dziesiętnej), to wówczas układ taki będzie można wykorzystać do dodawania liczb dziesiętnych przedstawianych w kodzie **BCD 8421**.

I wreszcie, układ taki umożliwia dodawanie dowolnej liczby składników (wymaga to jedynie odpowiednio pojemnej pamięci i układu sterowania, który zerowanie pamięci (PCP) przeprowadzałby dopiero po dodaniu wszystkich składników). ***Sumator taki może zatem dodawać ciągle nowe składniki (czyli gromadzić — akumulować), stąd jego nazwa*** — **akumulator**.

Pytania i zadania

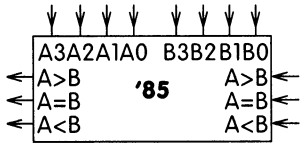
1. Narysuj układ sumatora 12-bitowego.
2. Dane są cztery bity liczby wejściowej A = **A3 A2 A1 A0** oraz bit znaku **BZ**. Zaprojektuj układ, na wyjściu którego będzie liczba wejściowa w zapisie **U2**. Układ zbuduj z wykorzystaniem sumatora **'83**.
3. Zbuduj układ mnożący przez 3 czterobitową liczbę wejściową. Zadanie rozwiąż, używając wyłącznie sumatorów.
4. Zbuduj układ mnożący przez 5 czterobitową liczbę wejściową. Zadanie rozwiąż, używając wyłącznie sumatorów.
5. Zbuduj układ mnożący przez 1,5 czterobitową liczbę wejściową. Zadanie rozwiąż, używając wyłącznie sumatorów.
6. Zbuduj układ mnożący przez 1,25 czterobitową liczbę wejściową. Zadanie rozwiąż, używając wyłącznie sumatorów.
7. Zaprojektuj układ generujący zapis **U9**. Dany bit znaku oraz cztery bity liczby w kodzie **naturalnym BCD.**
8. Zaprojektuj układ generujący zapis **U10**. Dany bit znaku oraz cztery bity liczby w kodzie **naturalnym BCD.**
9. Zaprojektuj układ generujący zapis **U10.** Dany bit znaku oraz osiem bitów liczby w kodzie **naturalnym BCD.**
10. Mając sumator **'83**, zbuduj sumator dziesiętny. Dane wejściowe oraz wynik sumowania muszą być przedstawione w kodzie **naturalnym BCD.**
11. Mając sumator **'83**, zbuduj komparator.
12. Narysuj symbol graficzny sumatora **'83**. Zaznacz wszystkie stany wejść i wyjść przy dodawaniu liczb dziesiętnych: 13, 12. Ile różnych stanów wejść można ustawić dla tego sumatora? Ile różnych stanów wyjść ma ten sumator?

11.4. Komparator '85

***Układ do porównywania dwóch liczb jest nazywany* komparatorem.**

Wynikiem porównania liczb A i B może być jedna z trzech relacji: A > B, A = B i A < B. Niekiedy wystarczy komparator umożliwiający odróżnienie jedynie relacji A = B i A ≠ B lub np. A ≥ B i A < B. W podrozdziale 3.6 zaprojektowano komparator (jako układ iteracyjny), który rozróżniał wszystkie trzy relacje, jakie mogą wystąpić pomiędzy dwoma liczbami.

Komparator scalony **'85** ma trzy wyjścia: **(A=B)**, **(A>B)**, **(A<B)**. Na jednym z nich jest poziom logiczny **1**, określający zależność między dwoma czterobitowymi słowami wejściowymi A i B. Układ ma ponadto trzy wejścia oznaczone również **(A=B)**, **(A>B)**, **(A<B)**, służące do współpracy z innymi układami scalonymi **'85** w celu porównywania liczb o długości większej niż czterobitowa. Wejścia te należy wówczas połączyć z odpowiednimi wyjściami komparatora młodszych bitów. W komparatorze najmłodszych bitów lub pracującym pojedynczo powinna być zapewniona następująca kombinacja sygnałów wejściowych: **(A=B) = 1**, **(A>B) = 0**, **(A<B) = 0**. Na rysunku 11.5 przedstawiono symbol graficzny sumatora **'85**.



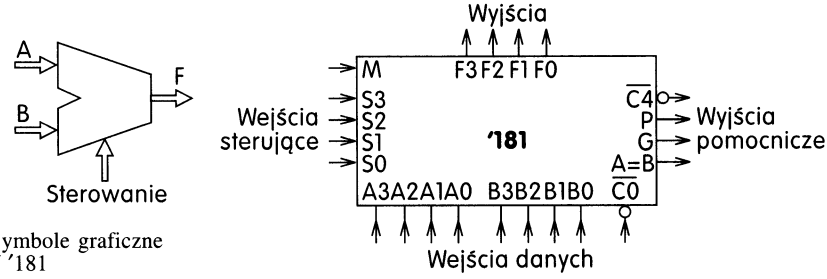
Rys. 11.5. Symbol graficzny komparatora scalonego '85

Pytania i zadania

1. Narysuj układ komparatora ośmiobitowego. Ile ma on stanów wejść?
2. Mając sumator **'83** oraz komparator **'85**, zbuduj transkoder kodu **naturalnego BCD** na kod **Aikena.**
3. Mając sumatory **'83** oraz komparatory **'85**, zbuduj sumator dziesiętny. Dane wejściowe oraz wynik sumowania muszą być przedstawione w kodzie **naturalnym BCD.**

11.5. Jednostka arytmetyczno-logiczna '181 (ALU)

Układ scalony **'181** jest blokiem funkcjonalnym, przeznaczonym do wykonywania operacji logicznych i arytmetycznych na dwóch czterobitowych słowach wejściowych A i B. Jest to układ 24-końcówkowy. Jego symbole graficzne przedstawiono na rys. 11.6.



Rys. 11.6. Symbole graficzne układu ALU '181

Ze względu na realizowane funkcje układ ten jest nazywany **jednostką arytmetyczno-logiczną**, w skrócie **ALU** (ang. *Arithmetic Logic Unit*). Jednostka taka zwykle współpracuje z zespołem rejestrów, służących do przechowywania argumentów oraz wyniku operacji.

UWAGA
Od tego podrozdziału (11.5) ponownie operator „+" będzie oznaczał sumowanie logiczne. Na oznaczenie działań arytmetycznych będziemy zaś stosować postać opisową tych operacji.

Wejścia sterujące **S0÷S3** są to **wejścia wyboru funkcji.** Po ustawieniu słowa na tym wejściu, układ realizuje odpowiadającą mu funkcję (np. słowu **1001** odpowiada funkcja arytmetyczna „A plus B", lub logiczna $\overline{A \oplus B}$).

Wejście rodzaju funkcji M — za pomocą tego wejścia wybieramy, czy układ ma realizować funkcje logiczne (**M = 1**), czy mieszane (arytmetyczne i logiczne, **M = 0**).

Wejścia danych to dziewięć wejść: **A0÷A3** oraz **B0÷B3** i **C0**. A = **A3A2A1A0** oraz B = **B3B2B1B0** to liczby czterobitowe, na których są przeprowadzane operacje. Wśród tych wejść **A0** i **B0** — to dwa najmniej znaczące bity (LSB), a **A3** i **B3** — dwa najbardziej znaczące bity (MSB). Wejście **C0** — to wejście bitu przeniesienia o wadze bitu LSB (zanegowane).

Wyjście wyniku F3÷F0, gdzie **F0** to LSB, **F3** — odpowiednio MSB. Na wyjściu tym jest generowany wynik operacji.

Wyjście przeniesienia C4 — na wyjściu tym jest generowany zanegowany bit przeniesienia (o wadze 2-krotnie większej niż waga **F3**) przy operacjach arytmetycznych.

Wyjście komparatora A = B jest to wyjście z otwartym obwodem kolektora (typu OC). Na wyjściu tym jest ustawiana jedynka wówczas, gdy na wyjściach wyniku są same jedynki. Wyjście to służy do porównywania liczb. Wykorzystując je razem z wyjściem **C4** można używać ALU jako komparatora. Należy w tym celu ustawić słowo sterujące: **S3S2S1S0 = 0110**, wejście **M = 0**, $\overline{\mathbf{C0}}$ **= 1**. Przy powyżej wymienionych stanach wejść jednostka realizuje funkcję arytmetyczną o postaci „A minus B minus 1". Działanie układu jako komparatora obrazuje tablica 11.1.

Wyjście przeniesienia generowanego — G.
Wyjście przeniesienia propagowanego — P.

Tablica 11.1

Wejścia	Wyjścia	
	A=B	C4
A=B	1	1
A>B	0	0
A<B	0	1

Wyjścia **G** i **P** przy wykorzystywaniu pojedynczego układu lub łączeniu ze sobą szeregowo kilku układów **'181** nie są wykorzystywane. Jednak kaskadowe (szeregowe) łączenie sprawia, że czas propagacji takiego rozbudowanego układu zwiększa się tyle razy, ile układów w nim zastosowano. Aby przyspieszyć działanie, wykorzystuje się wyjścia **G** i **P** oraz dodatkowo specjalnie do tego celu przeznaczony układ **'182** — tzw. **generator przeniesień jednoczesnych.**

Dla **M = 1** układ wykonuje wyłącznie funkcje logiczne, jak np. NOT, NAND, NOR, ExOR i inne. Jest to pełny zestaw funkcji dwóch zmiennych (patrz p. 2.2, tablica 2.2). Wyjaśnienia wymaga jedynie sposób realizacji tych funkcji, gdyż ich argumentami są tutaj słowa czterobitowe. ***Wszystkie operacje logiczne są realizowane na odpowiadających sobie parach bitów słów wejściowych*** A *i* B. Dla A = **1100** i B = **0010** operacje sumy logicznej oraz iloczynu logicznego będą się przedstawiać następująco (tu „+" oznacza działanie logiczne):

A = **1100**	**1 1 0 0**	**1 1 0 0**
B = **1010**	**+1 0 1 0**	**· 1 0 1 0**
	1 1 1 0	**1 0 0 0**

Podczas wykonywania operacji logicznych wyjścia przeniesień (**C4**, **A = B**) są blokowane i nie niosą żadnej informacji dotyczącej wyniku wykonywanego działania.

Dla **M = 0** jednostka arytmetyczno-logiczna realizuje funkcje logiczne, funkcje arytmetyczne oraz mieszane (arytmetyczno-logiczne). Jeżeli chodzi o funkcje logiczne, to są one powtórzeniami kilku z tych, jakie są realizowane dla **M = 1**.

Funkcja arytmetyczna „A plus B" jest wykonywana jako zwykłe dodawanie dwóch liczb czterobitowych. Przykładem funkcji mieszanej może być funkcja o postaci: „$(A + \overline{B})$ plus $\overline{A}$".

Pytania i zadania

1. Zbuduj z jednostki arytmetyczno-logicznej **'181** układ realizujący tzw. następnik, tj. liczbę o 1 większą od wejściowej.
2. Zbuduj z ALU **'181** układ realizujący tzw. poprzednik, tj. liczbę o 1 mniejszą od wejściowej.
3. Zbuduj z ALU **'181** układ realizujący zapis:
   a) **U1**,
   b) **U2**,
   c) **U9**,
   d) **U10**.
4. Rozwiąż zadanie 3. dla ośmiobitowych słów wejściowych.
5. Zbuduj z ALU **'181** sumator liczb czterobitowych, działający:
   a) w kodzie **binarnym naturalnym**,
   b) w kodzie **BCD naturalnym**.
6. Jaka jest różnica między dodawaniem arytmetycznym liczb (np. czterobitowych) a dodawaniem logicznym?
7. Wykonaj poniższe działania dla A = **1001**, B = **0110**.
   $\overline{A}B$, $A + \overline{B}$, $(A + B)$ plus $A\overline{B}$.
8. Wykonaj zadanie 7. dla danych: A = **1011**, B = **1000**.

12

Liczniki scalone

12.1. Wiadomości podstawowe

W podrozdziale 7.3 wprowadzono już pewne pojęcia odnoszące się do liczników oraz pokazano sposób budowy liczników asynchronicznych (szeregowych). W niniejszym rozdziale wiadomości te zostaną rozszerzone i wzbogacone o opis wybranych liczników scalonych zrealizowanych w technice TTL i CMOS.

Licznik ***jest to układ cyfrowy sekwencyjny, służący do zliczania i pamiętania liczby impulsów podawanych na jego wejście zliczające.*** Oprócz wejścia dla impulsów zliczanych, licznik ma zazwyczaj wejście ustawiające jego stan początkowy. ***Ustawienie wszystkich przerzutników, z których jest zbudowany licznik, w stan*** **0** ***nazywa się*** **zerowaniem licznika.**

Liczniki **dodające (zliczające w przód, zliczające w górę)** ***po każdym impulsie wejściowym zwiększają liczbę pamiętaną w liczniku o jeden. Natomiast*** **liczniki odejmujące (zliczające w tył, zliczające w dół)** ***zmniejszają o jeden zawartość licznika.*** W przypadku konieczności dodawania i odejmowania impulsów w jednym liczniku, są używane liczniki **rewersyjne (dwukierunkowe).**

Podstawowym elementem liczników jest przerzutnik synchroniczny. Liczniki są budowane w ten sposób, że wyjście przerzutnika **Q** jest jednocześnie wyjściem licznika. ***Liczba wyjść licznika jest równa liczbie przerzutników i określana mianem*** **długości licznika.**

Określona kombinacja stanów przerzutników, z których zbudowano licznik jest nazywana **stanem licznika**. Jeżeli licznik zbudowano z n przerzutników, to maksymalna liczba stanów licznika wynosi $N_{max} = 2^n$. Rzeczywista liczba stanów licznika musi więc spełniać nierówność $N \leq 2^n$. Liczba N jest nazywana **pojemnością licznika.** Na przykład licznik zbudowany z 4 przerzutników może mieć nie więcej niż 16 stanów.

Jeżeli licznik przechodzi przez wszystkie stany cyklicznie (tzn. po przejściu N stanów cykl jest powtarzany), to licznik taki nazywamy **licznikiem modulo *N* (*w skrócie* mod *N*).** Po podaniu na jego wejście zliczające *K* impulsów, licznik taki wskaże zliczenie *L* impulsów, gdzie $L = K \bmod N$ jest resztą z dzielenia całkowitego liczby *K* przez *N*. ***Jeżeli licznik przechodzi przez wszystkie stany jednokrotnie (i po osiągnięciu ostatniego pozostaje w nim), to taki licznik nazywamy*** **licznikiem do *N*.** Ponowne użycie takiego licznika wymaga wcześniejszego ustawienia go w stan początkowy (wyzerowania).

W praktyce mamy do czynienia przeważnie z licznikami zliczającymi w trybie **mod *N*.**

Liczniki są szczególnym rodzajem układów sekwencyjnych synchronicznych. Ta szczególność polega na tym, że przebieg zegarowy jest jednocześnie jedynym sygnałem wejściowym licznika, a zliczane impulsy są impulsami przebiegu synchronizującego pracę przerzutników. W opisie układów sekwencyjnych synchronicznych przebieg zegarowy w sposób jawny nie występuje (zobacz opis działania przerzutników synchronicznych — p. 7.3.1). Opis działania licznika nie zawiera więc sygnałów wejściowych (zobacz opis działania dwójki liczącej — p. 7.3.3).

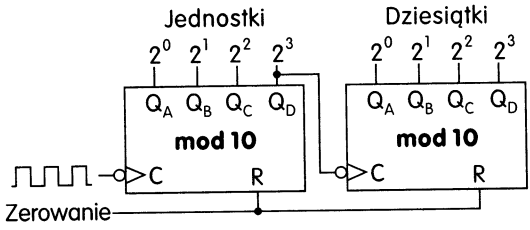
W podrozdziale 7.3.3 opisano liczniki budowane z przerzutników w taki sposób, że wyjście przerzutnika jest źródłem sygnału zegarowego (wejściowego) dla kolejnego przerzutnika. Liczniki takie są nazywane **licznikami szeregowymi** lub **asynchronicznymi.** Druga nazwa może być nieco myląca, wziąwszy pod uwagę fakt, że licznik jest zbudowany z przerzutników synchronicznych. Ale jak już wspomniano powyżej, liczniki są szczególnym przypadkiem układów sekwencyjnych synchronicznych. Przy połączeniu szeregowym przerzutników, zmianę stanu przerzutnika następnego powoduje przerzutnik poprzedni. Zliczane impulsy są podawane tylko do jednego przerzutnika. Stąd też nazwa **liczniki asynchroniczne** — nie wszystkie przerzutniki działają synchronicznie ze zliczanym impulsem.

Liczniki scalone są budowane zarówno jako asynchroniczne (szeregowe) lub jako **synchroniczne (równoległe).** ***W liczniku równoległym sygnał zegarowy (będący dla licznika zawsze przebiegiem impulsów zliczanych) jest doprowadzony jednocześnie do wejść synchronizujących wszystkich przerzutników.*** Pojawienie się kolejnego impulsu zliczanego sprawia, że wszystkie przerzutniki jednocześnie (współbieżnie) przetwarzają informację wejściową i czas ustalania się kolejnego stanu licznika wyznacza przerzutnik o najdłuższym czasie propagacji. Licznik taki jest znacznie szybszy od licznika szeregowego, jednak jego struktura jest bardziej złożona. Liczniki asynchroniczne są prostsze w budowie od liczników synchronicznych, lecz szeregowe działanie przerzutników sprawia, że ustalenie się nowej liczby w liczniku szeregowym następuje po dłuższym czasie niż w liczniku równoległym. Mimo tej wady liczniki szeregowe są często stosowane w układach automatyki, gdzie szybkość działania nie jest parametrem limitującym zastosowanie licznika, a najważniejszą sprawą jest prostota działania i mała liczba elementów z jakich zbudowano licznik.

W wielu licznikach scalonych część przerzutników pracuje synchronicznie (impuls zliczany jest doprowadzony jednocześnie do ich wejść zegarowych), a część asynchronicznie (ich wejścia zegarowe są sterowane z wyjść innych przerzutników). Nazywane są one **licznikami asynchroniczno-synchronicznymi** lub mniej precyzyjnie **licznikami asynchronicznymi.**

Najczęściej stosowanymi licznikami są liczniki zliczające **mod 10** i **mod 16.** Liczniki **mod 10** są zwane **licznikami dziesiętnymi** lub **dekadami.** Liczniki **mod 16** nazywa się **licznikami dwójkowymi (binarnymi**). Zwraca się uwagę Czytelnika na to nazewnictwo z tego względu, że w obu licznikach liczba zliczonych impulsów jest reprezentowana w zapisie binarnym, a nie tylko (jak sugerowałaby nazwa) w liczniku **mod 16.**

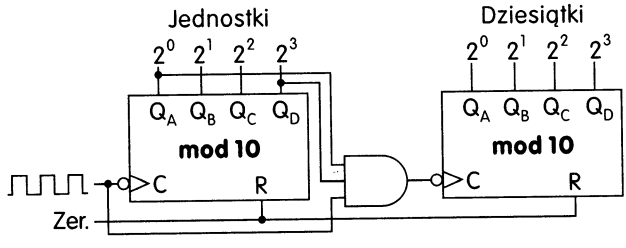
Aby rozszerzyć pojemność licznika możemy połączyć go z innymi licznikami bądź przerzutnikami. Ponieważ w tym rozdziale będziemy zajmować się wyłącznie licznikami scalonymi, to takie rozszerzenia będziemy realizować tylko przy użyciu liczników. Liczniki łączyć możemy równolegle lub szeregowo. Przy połączeniu szeregowym (rys. 12.1) sygnał zliczany doprowadzamy do pierwszego licz-



Rys. 12.1. Licznik modulo 100 zbudowany z dwóch liczników modulo 10 połączonych szeregowo

nika, a wyjście pierwszego licznika (to o najwyższej wadze) łączymy z wejściem zliczającym drugiego licznika. Połączenie szeregowe ma tę samą wadę co liczniki asynchroniczne, mianowicie taki układ działa wolniej. Jeżeli zależy nam na szybkości zliczania, to należy stosować połączenie równoległe.

Połączenie równoległe (rys. 12.2) polega na tym, że sygnał zliczany doprowadzamy do pierwszego licznika i jednocześnie do bramki AND, której wyjście podłączamy do wejścia drugiego licznika. Oprócz impulsów zliczanych do bramki AND należy podłączyć te wyjścia licznika pierwszego, które po zliczeniu $N - 1$ impul-



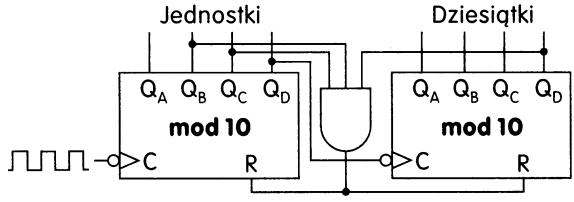
Rys. 12.2. Licznik modulo 100 zbudowany z dwóch liczników modulo 10 połączonych równolegle

sów (gdzie N — pojemność licznika pierwszego) mają stan wysoki. Dla licznika **mod 10** będzie to 9 impulsów. Po ich zliczeniu stan licznika wynosi więc **1001** i oba sygnały ($\mathbf{Q_D}$ i $\mathbf{Q_A}$) doprowadzone do wejść bramki AND mają poziom wysoki. Kolejny impuls przebiegu zliczanego (dziesiąty) zostanie zatem przepuszczony przez bramkę AND. Licznik następny zliczy więc jeden impuls. Ponieważ pierwszy licznik zostanie tym samym impulsem (zbocze opadające) przestawiony w stan **0000**, przeto bramka AND ponownie zablokuje przepuszczanie impulsów do drugiego licznika. Zliczenie kolejnego impulsu przez drugi licznik nastąpi po kolejnych dziesięciu impulsach wejściowych. Licznik ten będzie więc zliczał tylko dziesiątki impulsów wejściowych.

Połączenie liczników o pojemnościach N_1, N_2 daje w rezultacie licznik o pojemności $N = N_1N_2$. Jeżeli łączonymi licznikami będą liczniki **mod 10** (dekady), to wypadkowa pojemność wyniesie 100. Jeżeli będą to liczniki **mod 16**, to po połączeniu dwóch takich liczników uzyskamy licznik **mod 256** (o pojemności 256). Liczba łączonych liczników może być oczywiście większa niż dwa.

W przypadku, kiedy jest potrzebny licznik o pojemności innej niż jego własna lub wynikająca z pomnożenia pojemności liczników składowych, wtedy możemy zmniejszyć pojemność licznika za pomocą sprzężenia zwrotnego (patrz p. 7.3.3 — realizacja licznika **mod 10** z licznika **mod 16**). W tym celu należy zdekodować określony stan licznika za pomocą bramki AND (NAND — jeżeli licznik jest zerowany stanem niskim) i sygnałem z wyjścia tej bramki wyzerować licznik.

Załóżmy, że z liczników **mod 10** chcemy zbudować licznik **mod 86.** W tym celu łączymy dwa liczniki **mod 10** (np. szeregowo). Aby taki licznik zliczał **mod 86**, to jego najwyższym wskazaniem powinna być liczba 85. Następny, to jest 86. impuls powinien ustawić licznik w stan początkowy 0. W tym celu wyjścia licznika, które mają stan **1** w chwili wystąpienia 86. impulsu wejściowego, łączymy z wejściami bramki AND (86_{10}= **1000 0110**$_{2/10}$). Następnie wyjście tej bramki łączymy z wejściami zerującymi liczników (rys. 12.3).



Rys. 12.3 Licznik modulo 86

Gdyby licznik był zerowany niskim poziomem sygnału na wejściu zerującym **R**, wówczas należałoby zanegować sygnał z bramki AND lub zamiast niej zastosować bramkę NAND. Pozostawia się Czytelnikowi do przemyślenia, w jaki sposób zrealizować powyższe zadanie, aby nadal istniała możliwość zerowania licznika za pomocą zewnętrznego sygnału zerującego. Należy rozpatrzyć dwa przypadki. Jeden — gdy licznik jest zerowany poziomem wysokim, drugi — gdy jest zerowany poziomem niskim.

Taki sposób (rys. 12.3) uzyskiwania skróconego cyklu zliczania jest bardzo prosty, ale ma jedną wadę, która niekiedy może uniemożliwić jego wykorzystanie. Mianowicie, na jednym z wyjść takiego licznika pojawia się krótki impuls (hazard dynamiczny), który nie powinien wystąpić (w tym przykładzie dotyczy to wyjścia $\mathbf{Q_B}$ licznika jednostek). Zastosowanie tego typu układów lepiej więc ograniczyć jedynie do dzielników częstotliwości.

Aktywnym (**zliczanym,** czyli takim, w wyniku którego zmienia się stan licznika) **zboczem** przebiegu wejściowego może być zarówno zbocze narastające, jak i opadające. Zależy to od rodzaju przerzutników użytych do budowy licznika. Mogą to być także przerzutniki typu MS. Rodzaj zliczanego zbocza obrazuje symbol graficzny danego licznika.

Oprócz wejścia zerującego licznik (ustawiającego wszystkie przerzutniki w stan **L**) liczniki scalone mogą jeszcze mieć dodatkowe wejścia ustawiające w inny (określony) stan, np. w stan 9 = **1001** w liczniku **'90**. Budowane są także liczniki (**'192, '193, '029**) z wejściami równoległymi, których stan może być przepisany do licznika, co pozwala ustawić taki licznik w dowolny stan. Do przepisania stanu wejść równoległych do licznika służy dodatkowe wejście sterujące, zwane **wejściem przepisującym.** Licznik taki może nie mieć wejścia zerującego. Zerowanie licznika osiąga się wówczas poprzez wpisanie z wejść równoległych stanów niskich **L**.

Kierunek zliczania w licznikach rewersyjnych jest określany w dwojaki sposób: albo przez dwa niezależne wejścia sygnałów zliczanych $\mathbf{C_+}$ i $\mathbf{C_-}$ (jak np. w licznikach scalonych **'192, '193**), albo przez odrębne wejście sterujące **U/D** (+/–) (jak np. w liczniku **'029**).

W następnych podrozdziałach omówiono wybrane liczniki scalone. Ich zestaw tak wybrano, aby zaprezentować możliwie wiele różnych rozwiązań.

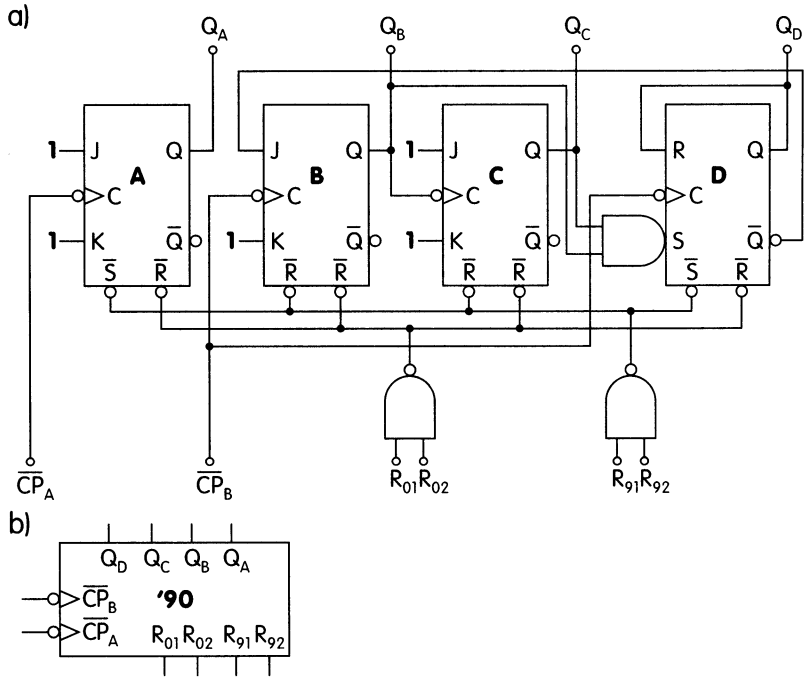
12.2. Scalone liczniki asynchroniczne

12.2.1. Licznik scalony '90

Schemat logiczny licznika oraz jego symbol graficzny przedstawiono na rys. 12.4. Układ ten zawiera cztery przerzutniki synchroniczne typu *MS* (dwuzboczowe), z których pierwszy (A) jest jednobitowym licznikiem **mod 2**, a trzy pozostałe (D, C, B) tworzą licznik **mod 5.**

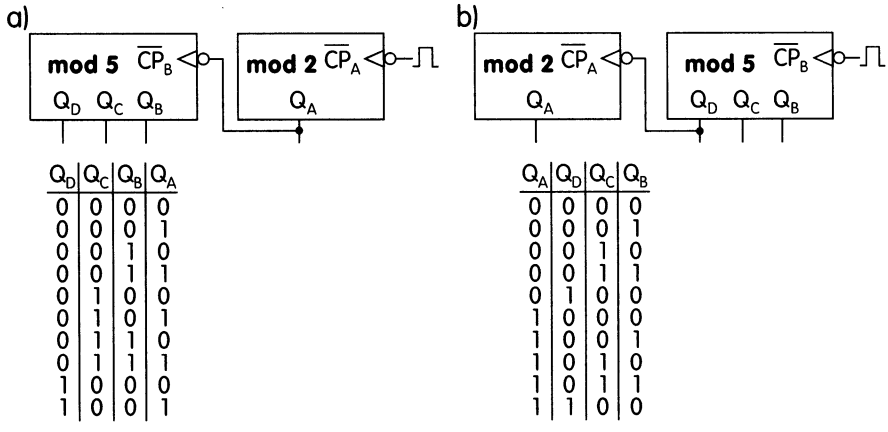
Trzy przerzutniki (A, B, C) są przerzutnikami typu *JK*, a czwarty przerzutnik (D) jest przerzutnikiem *RS*.

Układ scalony **'90** ma dwa wejścia zliczające $\overline{\mathbf{CP}}_A$ i $\overline{\mathbf{CP}}_B$, cztery wejścia sterujące $\mathbf{R}_{0(1)}$, $\mathbf{R}_{0(2)}$, $\mathbf{R}_{9(1)}$, $\mathbf{R}_{9(2)}$, oraz cztery wyjścia $\mathbf{Q_A}$, $\mathbf{Q_B}$, $\mathbf{Q_C}$, $\mathbf{Q_D}$. Układ może być wykorzystywany jako licznik **mod 2.** Wówczas wejściem zliczającym będzie wejście $\overline{\mathbf{CP}}_A$, a wyjściem tego licznika wyjście $\mathbf{Q_A}$. Wykorzystanie licznika jako zliczającego **mod 5** wymaga doprowadzenia przebiegu impulsów zliczanych do wejścia $\overline{\mathbf{CP}}_B$, a wyjściami takiego licznika są wyjścia $\mathbf{Q_B}$, $\mathbf{Q_C}$ i $\mathbf{Q_D}$ (o wagach odpowiednio 2^0, 2^1, 2^2).



Rys. 12.4. Licznik '90: a) schemat logiczny; b) symbol graficzny

Aby otrzymać licznik dziesiętny (dekadę), należy wykonać zewnętrzne połączenie wyjścia $\mathbf{Q_A}$ z wejściem $\overline{\mathbf{CP}}_\mathbf{B}$. Wejściem licznika jest wówczas $\overline{\mathbf{CP}}_\mathbf{A}$, a wyjściami wyprowadzenia $\mathbf{Q_A}$, $\mathbf{Q_B}$, $\mathbf{Q_C}$ i $\mathbf{Q_D}$ (o wagach odpowiednio 2^0, 2^1, 2^2 i 2^3). Połączenie takie jest szeregowym połączeniem licznika **mod 2** z licznikiem **mod 5**, co w efekcie prowadzi do uzyskania licznika **mod 10**. W podrozdziale 7.3.3 zwrócono już uwagę Czytelnika, że licznik **mod** *N* jest jednocześnie dzielnikiem przez *N*. Tak więc licznik **mod 10** może być wykorzystany jako dzielnik częstotliwości

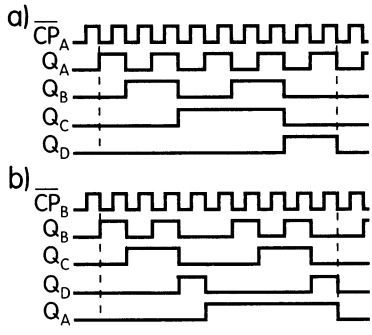


Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

Q_A	Q_D	Q_C	Q_B
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

Rys. 12.5. Asynchroniczny licznik mod10 zbudowany z układu '90: a) zliczający w kodzie naturalnym BCD; b) z przebiegiem wyjściowym o współczynniku wypełnienia równym 0,5 (zliczający w kodzie BCD 5421)

przez 10. Również w połączeniu szeregowym, ale w innej kolejności (tzn. najpierw **mod 5,** potem **mod 2**), uzyskamy licznik **mod 10**, ale nie będzie on zliczał w kodzie **naturalnym BCD (BCD 8421)**, lecz w kodzie **BCD**, który przedstawiono na rys. 12.5b, a który jest nazywany kodem **BCD 5421.** Przebiegi czasowe w obu licznikach pokazano na rys. 12.6.



Rys. 12.6. Przebiegi czasowe w liczniku '90 przy połączeniu: a) jak na rys. 12.5a; b) jak na rys. 12.5b

Drugi sposób połączenia licznika **'90** w dekadę liczącą ma tą zaletę, że przebieg wyjściowy z wyprowadzenia $\mathbf{Q_A}$ ma wypełnienie 0,5 (rys. 12.6b). Licznik taki będzie zatem wykorzystywany częściej jako dzielnik częstotliwości przez 10, bowiem w układach realizujących podział częstotliwości przebiegu wejściowego wymaga się często, aby przebieg wyjściowy miał współczynnik wypełnienia równy 0,5.

W celu uzupełnienia omówionych powyżej układów połączeń licznika scalonego **'90** należy jeszcze określić stany wejść ustawiających. Oddziaływanie sygnałów doprowadzonych do wejść ustawiających na pracę licznika zestawiono w tabl. 12.1.

Tablica 12.1

Wejścia ustawiające				Stan wyjść				Realizowana funkcja
$R_{0(1)}$	$R_{0(2)}$	$R_{9(1)}$	$R_{9(2)}$	Q_D	Q_C	Q_B	Q_A	
1	1	0	—	0	0	0	0	zerowanie
1	1	—	0	0	0	0	0	zerowanie
—	—	1	1	1	0	0	1	ustawianie liczby 9
—	0	—	0					zliczanie
0	—	0	—					zliczanie
0	—	—	0					zliczanie
—	0	0	—					zliczanie

Z tablicy 12.1 wynika, że licznik można wyzerować poprzez doprowadzenie do obu wejść $\mathbf{R_{0(1)}}$, $\mathbf{R_{0(2)}}$ poziomu wysokiego, pod warunkiem, że co najmniej do jednego z wejść $\mathbf{R_9}$ zostanie doprowadzony sygnał o poziomie niskim **L**. Na wyjściu licznika można ustawić stan wyjść $\mathbf{Q_DQ_CQ_BQ_A = 1001}$ (co odpowiada liczbie dziesiętnej 9), jeżeli do obu wejść ustawiających $\mathbf{R_9}$ (tzn. $\mathbf{R_{9(1)}}$, $\mathbf{R_{9(2)}}$) doprowadzimy sygnał o poziomie **1**, niezależnie od stanu wejść ustawiających $\mathbf{R_0}$. Inaczej mówiąc: wejście ustawiające stan 9 na wyjściu jest dominujące, a jednoczesne zerowanie i ustawianie licznika prowadzi do ustawienia jego wyjść w stan 9.

Warunkiem koniecznym do tego, aby licznik zliczał impulsy przebiegu wejściowego, jest doprowadzenie do jednego z wejść $\mathbf{R_0}$ i jednego z wejść $\mathbf{R_9}$ poziomu niskiego **L**.

Tablica 12.2. **Dane techniczne licznika 'LS90**

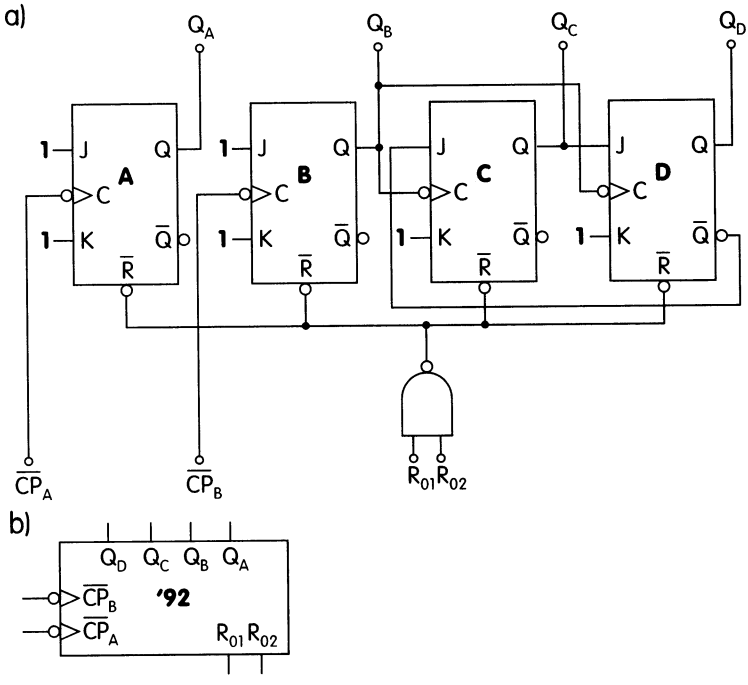
Oznaczenie	Wartość	We/Wy	Jednostka
I_{ILmax}	−0,4 −2,4 −3,2	R0, R9 $\overline{CP}_A$ $\overline{CP}_B$	mA
I_{IHmax}	20 100 200 400	R0 R9 CP_A CP_B	μA
$I_{OS}(I_{OH})$	min −20 max −100	— —	mA
I_{CC}	max15 typ 9	—	mA
N	10	—	—
t_{pHL} t_{pLH}	18 16	$\overline{CP}_A \rightarrow Q_A$	ns
t_{pHL} t_{pLH}	21 16	$\overline{CP}_B \rightarrow Q_B$	
t_{pHL} t_{pLH}	35 32	$\overline{CP}_B \rightarrow Q_C$	
t_{pHL} t_{pLH}	35 32	$\overline{CP}_B \rightarrow Q_D$	
t_{pHL} t_{pLH}	50 48	$\overline{CP}_A \rightarrow Q_D$	
t_{pHL}	40	$R0 \rightarrow Q_{A(BCD)}$	
t_{pHL} t_{pLH}	40 18	$R9 \rightarrow Q_{B(C)}$ $R9 \rightarrow Q_{A(D)}$	
f_{max}	32 16	CP_A CP_B	MHz

W tablicy 12.2 zestawiono parametry techniczne licznika **'LS90**. Dane pozostałych liczników (opisanych poniżej) nie będą prezentowane, gdyż podstawowym źródłem takich informacji powinien być katalog wydany przez producenta układu. Dla licznika **'LS90** uczyniono wyjątek, aby zwrócić uwagę Czytelnika na rodzaj parametrów, jakie charakteryzują podstawowe właściwości liczników scalonych.

Parametry te są analogiczne do parametrów podstawowych bramek TTL. Ponieważ licznik ma więcej różnych wejść (w sensie funkcji realizowanych, a nie liczebnie) i wyjść niż bramka, przeto wartości podawanych parametrów są odnoszone do konkretnych wejść i wyjść. Z parametrów dynamicznych bardziej istotna (niż czas propagacji) jest wartość maksymalnej częstotliwości impulsów zliczanych. Określa ona maksymalną częstotliwość przebiegu wejściowego, przy której producent zapewnia jeszcze poprawną pracę licznika.

12.2.2. Licznik scalony '92

Licznik scalony **'92** jest asynchronicznym licznikiem, zbudowanym z czterech przerzutników typu *JK-MS* (rys. 12.7). Jeden przerzutnik (A) jest licznikiem **mod 2** (dzielnikiem przez 2), a trzy pozostałe (B, C, D) stanowią licznik **mod 6.** Oba liczniki mogą być wykorzystywane niezależnie. Wszystkie cztery przerzutniki są zerowane tym samym sygnałem, pochodzącym z wyjścia bramki NAND. Doprowadzenie do obu wejść tej bramki ($\mathbf{R}_{0(1)}$, $\mathbf{R}_{0(2)}$) sygnałów o poziomie **1** zeruje licznik. Zliczanie jest możliwe, jeśli co najmniej na jednym z tych wejść jest poziom niski **L**. Połączenie szeregowe licznika **mod 2** z licznikiem **mod 6** (wyjście $\mathbf{Q}_A$ z wejściem $\mathbf{CP}_B$) stanowi licznik **mod 12**, zliczający w kodzie naturalnym. Połączenie w odwrotnej kolejności (wyjście $\mathbf{Q}_D$ z wejściem $\mathbf{CP}_A$) jest także licznikiem **mod 12**, ale przeznaczeniem takiej konfiguracji połączeń jest praca układu jako dzielnika częstotliwości przez 12. Przebieg wyjściowy (wyjście $\mathbf{Q}_A$) ma wówczas współczynnik wypełnienia równy 0,5.



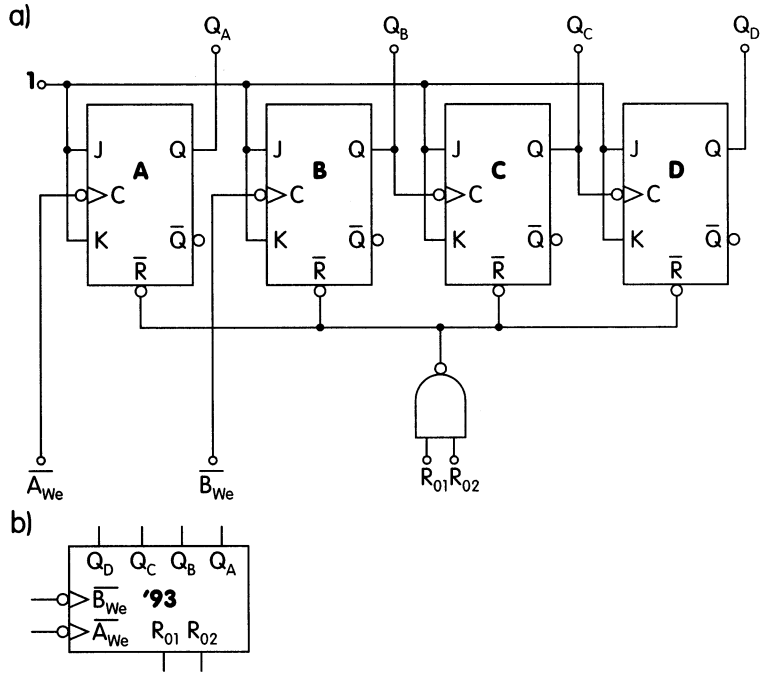
Rys. 12.7. Licznik '92: a) schemat logiczny; b) symbol graficzny

Liczniki **mod 6** i **mod 12** wraz z licznikami **mod 10** dogodnie jest stosować w układach zegarowych do zliczania sekund i minut (licznik **mod 60**) oraz godzin (licznik **mod 12**). Jednak ich istotną wadą w takich zastosowaniach jest brak możliwości ustawienia dowolnego stanu początkowego.

12.2.3. Licznik scalony '93

Układ scalony **'93** zawiera cztery przerzutniki *JK-MS* (rys. 12.8). Trzy przerzutniki (D, C i B) są połączone szeregowo, tworząc licznik **mod 8,** a czwarty przerzutnik (A) jest dwójką liczącą, która może być wykorzystana dwojako: wspólnie z licznikiem **mod 8** tworzy licznik **mod 16**, albo oddzielnie jako licznik **mod 2.**

Łącząc wyjście $\mathbf{Q}_A$ pierwszego przerzutnika z wejściem licznika 3-bitowego ($\overline{\mathbf{B}}_{We}$), uzyskujemy licznik **mod 16** zliczający w kodzie **naturalnym dwójkowym** (patrz p. 7.3.3), przy czym wyjściami licznika są wyprowadzenia $\mathbf{Q_AQ_BQ_CQ_D}$, gdzie $\mathbf{Q}_A$ — LSB i $\mathbf{Q}_D$ — MSB. Łącząc wyjście czwartego przerzutnika ($\mathbf{Q}_D$) z wejściem pierwszego przerzutnika ($\overline{\mathbf{A}}_{We}$) uzyskujemy także licznik **mod 16**, który również zlicza w naturalnym kodzie dwójkowym, przy czym wyjściami licznika są wyprowadzenia $\mathbf{Q_BQ_CQ_DQ_A}$, gdzie $\mathbf{Q}_B$ — LSB i $\mathbf{Q}_A$ — MSB. W obu tych przypadkach przebieg czasowy na wyjściu MSB ma częstotliwość 16-krotnie mniejszą niż przebieg zliczany oraz współczynnik wypełnienia równy 0,5. Dzieje się tak dlatego, że wszystkie przerzutniki w tym liczniku pracują w układzie dwójek liczących i kolejność ich połączeń nie ma znaczenia.



Rys. 12.8. Licznik '93: a) schemat logiczny; b) symbol graficzny

Zerowanie licznika **'93** odbywa się identycznie, jak opisanego wcześniej licznika **'92.** Wszystkie funkcje realizowane przez licznik (jako wynik odpowiednich połączeń i stanów wejść sterujących) można zestawić w postaci tablicy działania, w której

Tablica 12.3. **Tablica działania licznika '93**

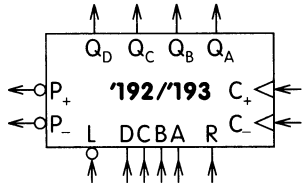
A_{We}	B_{We}	R0(1)	R0(2)	Realizowana funkcja
—	—	1	1	zerowanie licznika
⌐	— —	0 —	— 0	zliczanie **mod 2,** wyjście licznika Q_A
— —	⌐	0 —	— 0	zliczanie **mod 8,** wyjścia licznika $Q_BQ_CQ_D$, Q_B — LSB, Q_D — MSB
⌐	Q_A Q_A	0 —	— 0	zliczanie **mod 16** w naturalnym kodzie dwójkowym, Q_A — LSB, Q_D — MSB
Q_D Q_D	⌐	0 —	— 0	zliczanie **mod 16** w naturalnym kodzie dwójkowym, Q_B — LSB, Q_A — MSB

należy uwzględnić wszystkie wejścia i wszystkie możliwe stany pracy układu. Tablicę taką można utworzyć dla dowolnego licznika lub dowolnego układu sekwencyjnego. Na przykład dla licznika **'93** powinna mieć ona postać taką jak tabl. 12.3.

12.3. Scalone liczniki synchroniczne

12.3.1. Liczniki scalone '192 i '193

Liczniki scalone **'192** i **'193** są synchronicznymi licznikami rewersyjnymi, mającymi możliwość ustawienia w dowolny stan początkowy. Oba mają identyczne wejścia, wyjścia, a także konfigurację wyprowadzeń. Dlatego zostaną omówione wspólnie. Jedyna różnica między nimi jest taka, że licznik **'192** jest licznikiem **mod 10** (**dziesiętnym**), a licznik **'193** jest licznikiem **mod 16** (**binarnym**). Oba mają dość złożoną budowę, więc pominięto przedstawienie schematu logicznego obu liczników, zwłaszcza że znajomość schematu logicznego dla użytkownika układu nie jest niezbędna.



Rys. 12.9. Licznik '192 ('193) — symbol graficzny

Symbol graficzny licznika **'192** (**'193**) przedstawiono na rys. 12.9. Oba układy mają następujące **wejścia**:

— **zliczające C_+** i **C_-** (ang. *Count up, Count down*); impulsy doprowadzane do wejścia **C_+** są dodawane do aktualnej zawartości licznika, a impulsy doprowadzane do wejścia **C_-** są odejmowane od aktualnej zawartości licznika; aktywnym (zliczanym) zboczem jest zbocze dodatnie;

— **zerujące R** (ang. *Reset*); doprowadzenie poziomu **1** do tego wejścia powoduje wyzerowanie licznika; wejście działa asynchronicznie, tzn. niezależnie od impulsu zliczanego;

— **równoległe (informacyjne) D, C, B, A**; stan tych wejść jest wpisywany do licznika (tzn. stan wyjść $Q_DQ_CQ_BQ_A$ staje się taki, jak stan wejść **DCBA**) po doprowadzeniu do wejścia przepisującego poziomu logicznego **0**;

— **przepisujące, wprowadzające L** (ang. *Load*); doprowadzenie do tego wejścia poziomu logicznego **0** powoduje wpisanie informacji z wejść równoległych do licznika; zmiana stanu wejść równoległych przy **L = 0** jest natychmiast przenoszona do wyjścia układu.

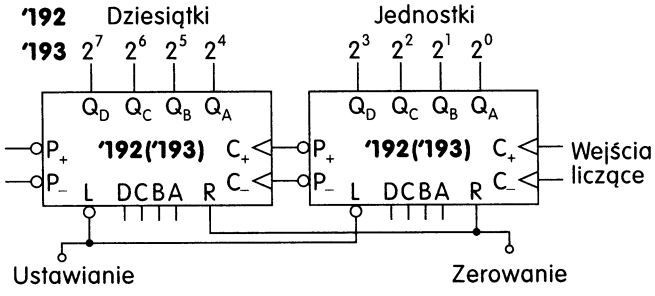
Wyjściami układów są wyprowadzenia:

— Q_D, Q_C, Q_B, Q_A, na których występuje słowo wyjściowe, przy czym wyjście Q_A odpowiada najmłodszemu bitowi (LSB), a wyjście Q_D najstarszemu (MSB);

— **przeniesienia P_+** (ang. *carry*); na wyjściu tym pojawia się poziom niski ***synchronicznie ze zboczem ujemnym przebiegu wejściowego***, gdy licznik znajdzie się w stanie **1001** (**1111**) i wyjście to ponownie jest ustawiane w stan **1** po przejściu do stanu **0000**, gdy licznik pracuje jako dodający; wyjście to jest przeznaczone do współpracy z kolejnymi licznikami w celu zwiększenia pojemności przy liczeniu w przód;

— **pożyczka P_-** (ang. *borrow)*; na wyjściu tym pojawia się poziom niski ***synchronicznie ze zboczem ujemnym przebiegu wejściowego***, gdy licznik znajdzie się w stanie **0000** i wyjście to ponownie jest ustawiane w stan **1** po przejściu do stanu **1001** (**1111**), gdy licznik pracuje jako odejmujący; wyjście to jest przeznaczone do współpracy z kolejnymi licznikami w celu zwiększenia pojemności przy liczeniu w tył. (Dane w nawiasach odnoszą się do licznika **'193**).

W bardzo prosty sposób można łączyć liczniki ze sobą, zwiększając ich pojemność. Na rysunku 12.10 pokazano schemat licznika **mod 100** (**mod 256**).



Rys. 12.10. Zwiększenie pojemności liczników rewersyjnych '192 ('193)

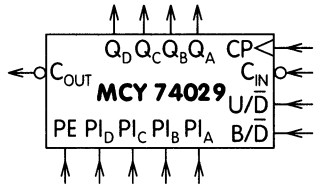
12.3.2. Licznik scalony '029 (CMOS)

Licznik scalony **'029**, wykonany w technice CMOS, jest licznikiem synchronicznym rewersyjnym. Może on zliczać impulsy wejściowe zarówno w trybie **mod 10**, jak i **mod 16**. Ma następujące **wejścia**:

— **zegarowe CP** (ang. *Clock Pulse*); jest to wejście sygnału impulsów zliczanych;
— **równoległe PI_D, PI_C, PI_B, PI_A** (ang. *Parallel Input*); stan tych wejść może być przepisany do licznika;
— **przepisujące PE** (ang. *Parallel Entry*, *Parallel Enable*); podanie **1** na to wejście powoduje przepisanie stanu wejść równoległych do licznika;
— **przeniesienia C_{IN}** (ang. *Carry INput*); umożliwia równoległe łączenie ze sobą kilku liczników **'029**; licznik zlicza impulsy z wejścia zegarowego, gdy na tym wejściu jest poziom niski (wejście może być także wykorzystywane do bramkowania licznika);
— **sterujące B/$\overline{D}$** (ang. *Binary/Decimal*); pozwala ustawić tryb zliczania na **mod 10** (**B/$\overline{D}$ = 0**) lub **mod 16** (**B/$\overline{D}$ = 1**);
— **sterujące U/$\overline{D}$** (ang. *Up/Down*); umożliwia wybór rodzaju pracy, **U/$\overline{D}$ = 0** — zliczanie w tył, **U/$\overline{D}$ = 1** — zliczanie w przód;

Układ ma 5 **wyjść**:

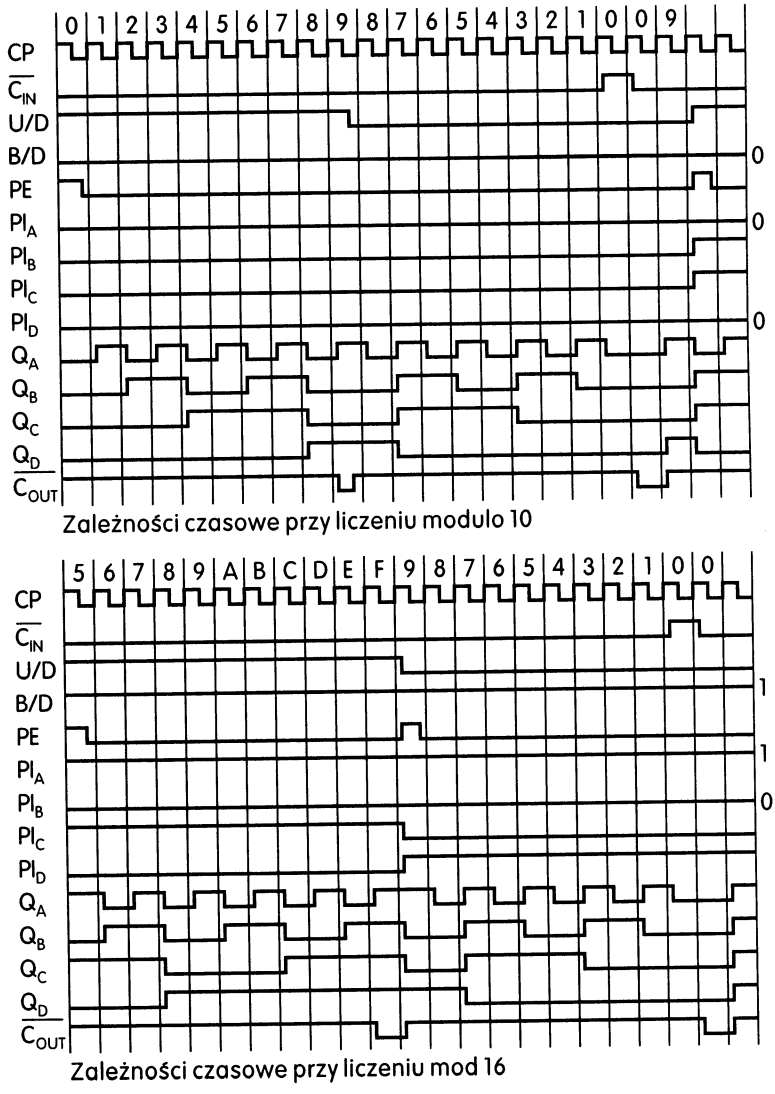
— **Q_D, Q_C, Q_B, Q_A**, na których występuje słowo wyjściowe, przy czym wyjście **Q_D** odpowiada bitowi MSB, a wyjście **Q_A** bitowi LSB;
— **przeniesienia C_{OUT}**(ang. *Carry OUTput*); pojawia się na nim niski poziom (**L**), gdy licznik liczący w przód znajdzie się w stanie **1001** (9) (albo **1111** (15) przy zliczaniu **mod 16**) lub gdy licznik liczący w tył znajdzie się w stanie **0000** (0).



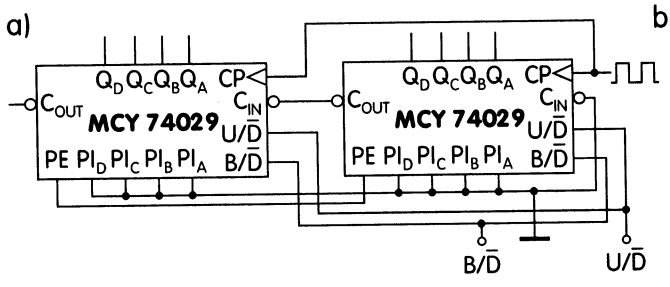
Rys. 12.11. Symbol graficzny licznika '029

Symbol graficzny licznika przedstawiono na rys. 12.11. Poprzednio (dla licznika **'93**) przedstawiono opis licznika w postaci tablicy działania. Jednak ***najpełniejszym, najbardziej precyzyjnym i jednoznacznym opisem działania licznika są przebiegi czasowe***. Ten sposób opisu dla licznika **'029** przedstawiono na rys. 12.12. Przebiegi te sporządzono z uwzględnieniem czasów propagacji elementów, z których jest zbudowany licznik (niekiedy rysuje się przebiegi czasowe tak, jak by nie było żadnych opóźnień) i dlatego zmiany stanów **Q** pojawiają się z opóźnieniem w stosunku do zliczanego zbocza. Opóźnienia te są jednakowe (w przybliżeniu), co świadczy o tym, że jest to licznik synchroniczny (równoległy).

Licznik ten nie ma oddzielnego wejścia zerującego (jak liczniki **'192** i **'193**). Jego wyzerowanie można uzyskać, wpisując z wejść równoległych słowo binarne **0000**. Schemat logiczny licznika o długości 8 bitów pokazano na rys. 12.13.



Rys. 12.12. Przebiegi czasowe w liczniku '029



b)

B/D	U/D	Funkcja
0	0	mod 10 w tył
0	1	mod 10 w przód
1	0	mod 16 w tył
1	1	mod 16 w przód

Rys. 12.13. Licznik 8-bitowy zbudowany z liczników '029: a) schemat zasadniczy; b) tablica sterowania

Liczniki wykonywane w technice CMOS (seria 4000B) są budowane przeważnie jako synchroniczne. Wynika to z mniejszej szybkości działania układów CMOS. Opisany powyżej licznik **'029** ma maksymalną częstotliwość przebiegu zliczanego nie mniejszą niż $f = 5{,}5$ MHz (przy $U_z = U_{DD} - U_{SS} = 15$ V). Liczniki CMOS w wersji asynchronicznej są wykonywane raczej z przeznaczeniem do pracy w charakterze dzielników częstotliwości. Ostatnie uwagi nie dotyczą serii układów szybkich CMOS (AC, ACT, HC, HCT).

Pytania i zadania

1. Omów takie pojęcia, jak: licznik, licznik asynchroniczny, licznik synchroniczny, licznik dodający, licznik odejmujący, stan licznika, zerowanie licznika, licznik **mod *N***, dzielnik przez *N*, licznik rewersyjny, pojemność licznika, licznik do *N*.
2. Uzasadnij, dlaczego w liczniku **'90** jednoczesne zerowanie i ustawianie wyjścia w stan 9 prowadzi do ustawienia wyjść w stan 9, a nie do wyzerowania układu.

   Wskazówka. Należy sobie przypomnieć działanie scalonych przerzutników przy sterowaniu ich od strony wejść przygotowujących.
3. Zapisz tablicę działania licznika:
   a) **90**,
   b) **92**,
   c) **192/193**,
   d) **'049**,

   wzorując się na tabl. 12.3.
4. Mając licznik **'90** zbuduj licznik do 9, tzn. taki, który po zliczeniu 9 impulsów pozostanie w stanie 9 (**1001**).

   Wskazówka. Bramkować odpowiednim sygnałem przebieg impulsów zliczanych.
5. Z licznika **'90** zbuduj dzielnik przez 7, korzystając:
   a) z licznika **mod 10** w połączeniu **mod 2** → **mod 5**,
   b) z licznika **mod 10** w połączeniu **mod 5** → **mod 2**.

   Który z tych układów nie wymaga użycia dodatkowych elementów (bramek)?
6. Narysuj (wykorzystując licznik **'93**) schemat zasadniczy dzielnika częstotliwości przez:
   a) 11,
   b) 13,
   c) 14.
7. Narysuj schemat licznika **mod 12** (wykorzystując układ scalony **'92**) w połączeniu: **mod 6** → **mod 2.** Określ kod, w jakim będzie zliczał licznik. Narysuj przebiegi czasowe w takim liczniku.
8. Narysuj schemat licznika **mod 16** (wykorzystując układ scalony **'93**) w połączeniu: **mod 8** → **mod 2.** Określ kod, w jakim będzie zliczał licznik. Narysuj przebiegi czasowe w takim liczniku.
9. Narysuj przebiegi czasowe w liczniku **'192**, liczącym w przód. Uwzględnij na wykresie przebieg wyjściowy $\mathbf{P}_+$.
10. Narysuj przebiegi czasowe w liczniku **'192**, liczącym w tył. Uwzględnij na wykresie przebieg wyjściowy $\mathbf{P}_-$.
11. Mając licznik **'192** zbuduj licznik do 9, tzn. taki, który po zliczeniu 9 impulsów pozostanie w stanie 9 (**1001**):
    a) zadanie rozwiąż analogicznie jak 4.,
    b) zadanie rozwiąż z wykorzystaniem wejść równoległych.

12. Zaprojektuj obwód wejściowy (dla liczników **'192**, **'193**) przełączający kierunek zliczania. Do wejścia **Z** jest doprowadzony przebieg impulsów zliczanych, a wejście **S** jest wejściem sterującym. Przy **S** = **1** układ ma zliczać w przód, a przy **S** = **0** w tył.
13. Narysuj schemat zasadniczy licznika zliczającego **mod 60**.
14. Narysuj (wykorzystując licznik **'193** i jego wejście zerujące) schemat zasadniczy dzielnika częstotliwości przez:
    a) 11,
    b) 13,
    c) 14.
15. Narysuj (wykorzystując licznik **'193** i jego wejścia równoległe wraz z przepisującym) schemat zasadniczy dzielnika częstotliwości przez:
    a) 11,
    b) 13,
    c) 14.
16. Narysuj (wykorzystując licznik **'029**) schemat zasadniczy dzielnika częstotliwości przez:
    a) 11,
    b) 13,
    c) 14.
17. Z licznika **'192** zbuduj licznik **mod 7**, liczący w przód. Narysuj przebiegi czasowe w układzie oraz określ wyjście, na którym wystąpi hazard dynamiczny.
    a) wykorzystaj wejście zerujące,
    b) wykorzystaj wejście przepisujące.
18. Z licznika **'192** zbuduj licznik **mod 7**, liczący w tył. Narysuj przebiegi czasowe w układzie.

    Uwaga. Licznik ze stanu **0000** powinien przechodzić do stanu **0110**. Zwykłe wpisanie (z wejść równoległych stanu **0110**) sygnałem **P_** spowoduje, że zgubimy stan **0000**, który będzie trwał tylko przez cza strwania impulsu zliczanego — dodatnie zbocze bowiem ustawi stan **0000** i w chwili pojawienia się zbocza ujemnego sygnał **P_** przyjmie wartość **0**. Jeśli tym sygnałem wpiszemy z wejść równoległych stan **0110**, to w jednym takcie licznik przejdzie obydwa stany. Układ, jaki należy dobudować, powinien więc opóźniać o takt proces przepisywania informacji z wejść równoległych. Zamiast sygnału **P_** (sterującego przepisywaniem) można użyć sygnału wyjściowego z układu dekodującego stan **1001** licznika, wówczas zbędne będzie używanie układu opóźniającego.
19. Z licznika **'90** zbuduj dzielnik przez 7, nie korzystając z innych (dodatkowych) elementów.
20. Jak jest różnica między licznikiem **mod *N*** a dzielnikiem częstotliwości przez *N*?
21. Jaka jest różnica między licznikiem **mod *N*** a licznikiem do *N*?

13

Rejestry

13.1. Wiadomości podstawowe

Rejestrem ***nazywamy układ zbudowany z przerzutników, służący do przechowywania informacji.*** Rejestry należą do sekwencyjnych układów cyfrowych. Zbudowane z przerzutników asynchronicznych należą do sekwencyjnych układów asynchronicznych. Przykładem takiego rejestru może być układ przerzutników asynchronicznych wykorzystany do budowy systemu transmisji szeregowo-równoległej przedstawiony w p. 10.2.3 (rys. 10.8). Rejestry zbudowane z przerzutników synchronicznych są sekwencyjnymi układami synchronicznymi.

W technice cyfrowej częściej mamy do czynienia z rejestrami synchronicznymi i takim jest poświęcony niniejszy rozdział.

Liczba bitów informacji, jaka może być przechowywana w rejestrze, jest nazywana* długością rejestru *i odpowiada zawsze liczbie przerzutników, z których jest zbudowany rejestr.

Informacja może być wprowadzana do rejestru **szeregowo**, tzn. bit po bicie w takt sygnału synchronizującego (zegarowego), lub **równolegle**, tzn. całe słowo wejściowe jest zapisywane jednocześnie w chwili wyznaczonej przez sygnał taktujący lub asynchronicznie przez wysterowanie odpowiedniego wejścia przepisującego. Także wyprowadzanie informacji przechowywanej w rejestrze może odbywać się szeregowo bądź równolegle. W związku z powyższym rozróżnia się następujące rodzaje **rejestrów:**

— **szeregowo-szeregowy** (nazywany też krócej — **szeregowy**) (SISO, ang. *Serial Input-Serial Output*); zapis i odczyt jest realizowany szeregowo;
— **szeregowo-równoległy** (SIPO, ang. *Serial Input-Parallel Output*); informacja jest wprowadzana szeregowo, a wyprowadzana równolegle;
— **równoległo-szeregowy** (PISO, ang. *Parallel Input-Serial Output*); zapis jest realizowany równolegle, a odczyt szeregowo;
— **równoległo-równoległy** (nazywany też krócej — **równoległy**) (PIPO, ang. *Parallel Input-Parallel Output*); zapis i odczyt jest realizowany równolegle.

Budowane są także rejestry łączące cechy kilku rejestrów wymienionych powyżej. Taki rejestr może być np. uniwersalny, jeśli chodzi o sposób wprowadzania do niego informacji, to znaczy informacja może być wpisywana zarówno równolegle, jak i szeregowo. W takich rejestrach jeden ze sposobów wprowadzania informacji (zwykle szeregowy) jest realizowany synchronicznie, a drugi (równoległy) asynchronicznie. Taka uniwersalność może także dotyczyć wyjścia rejestru.

Rejestry mające w swojej nazwie określenie „szeregowo (-y)" charakteryzują się koniecznością przesuwania wprowadzonej (wprowadzanej) informacji. W **rejestrach jednokierunkowych** informacja może być przesuwana w prawo (w kierunku starszych bitów) lub w lewo (w kierunku młodszych bitów). **Rejestry dwukierunkowe (rewersyjne)** umożliwiają przesuwanie wprowadzonej informacji zarówno w prawo, jak i w lewo. Rejestry z możliwością przesuwania zapisanej w nich informacji noszą nazwę **rejestrów przesuwających.**

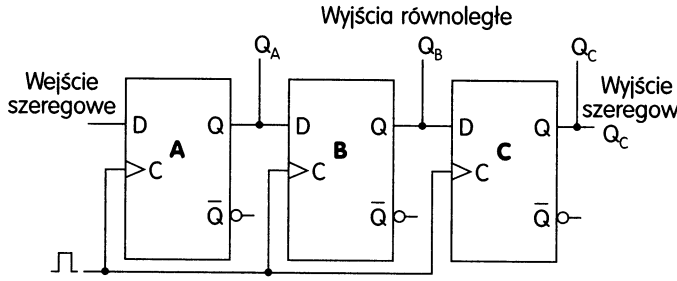
13.2. Budowa i zasada działania rejestrów

Najprostszym rejestrem jest synchroniczny przerzutnik typu *D* (patrz p. 7.3.1). Jedynka na jego wejściu **D** w chwili wyznaczonej przez przebieg zegarowy ustawia przerzutnik w stan **1**, co obecnie określimy jako wpisanie **1** do rejestru. Analogicznie **0** na wejściu **D** ustawia przerzutnik w stan niski **L**, czyli powiemy, że zostaje wpisane **0**. Zestawienie kilku takich przerzutników (np. 4), bez żadnych połączeń pomiędzy nimi poza wspólnym sygnałem zegarowym, będzie więc 4-bitowym **rejestrem równoległo-równoległym (równoległym).** Z rejestrów równoległych są budowane tzw. **pamięci buforowe.** Pamięć buforowa pośredniczy między układami cyfrowymi, które działają z różnymi szybkościami. Często także jest wykorzystywana w cyfrowych przyrządach pomiarowych, w których pomiar sprowadza się do zliczania impulsów w określonym czasie. Pełnią one wtedy rolę bufora pomiędzy licznikiem a transkoderem wskaźników. Po zakończeniu cyklu pomiarowego (zliczania) następuje przepisanie stanu liczników do rejestru, zdekodowanie i wysterowanie wskaźnika. Kolejny cykl pomiarowy to wyzerowanie liczników i ponowne zliczanie. Dzięki pamięci buforowej zostaje wyeliminowane migotanie cyfr, gdyż nie obserwujemy procesu zliczania. Po zliczeniu nowa wartość zostaje przepisana do rejestru i wyświetlona na wskaźniku.

Na przykład łącząc trzy (A, B, C) przerzutniki typu *D* w sposób pokazany na rys. 13.1, otrzymamy rejestr:

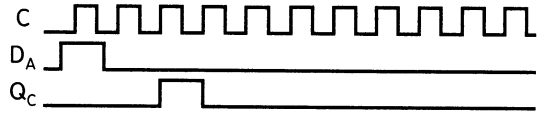
a) szeregowy, jeżeli wyjście będziemy pobierać tylko z $\mathbf{Q_C}$,
b) szeregowo-równoległy, jeżeli wyjście pobierać będziemy ze wszystkich przerzutników.

Przeanalizujmy pracę układu przy pobieraniu sygnału wyjściowego z wyjścia przerzutnika C. Załóżmy, że rejestr jest wyzerowany, a na jego wejściu jest stan wysoki podczas wystąpienia pierwszego impulsu zegarowego, a potem już cały czas stan niski. Pierwszy impuls zegarowy sprawi, że **1** z wejścia rejestru zostanie wpisana do przerzutnika A. W drugim takcie zegara przerzutnik znów



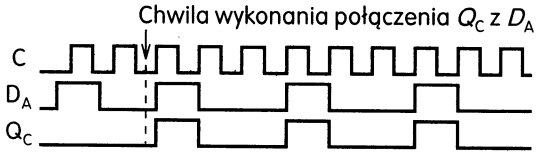
Rys. 13.1. Rejestr zbudowany z trzech przerzutników typu *D*

przyjmie stan niski (na jego wejściu zgodnie z założeniem jest już **0**), ale **1** z wyjścia przerzutnika A zostanie teraz wpisana do przerzutnika B. Kolejny impuls zegarowy przesunie jedynkę do przerzutnika C, czyli do wyjścia układu. Oczywiście w rozpatrywanym rejestrze jest przesuwana cała informacja trzybitowa, a nie tylko ta jedna **1**. Takie działanie nazywamy przesuwaniem informacji w rejestrze, a sam rejestr nazywamy **rejestrem przesuwającym**. Przebiegi czasowe w tym układzie przedstawiono na rys. 13.2.



Rys. 13.2. Przebiegi czasowe w rejestrze z rys. 13.1 pracującym jako rejestr szeregowo-szeregowy

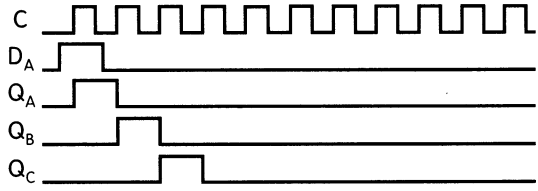
Załóżmy teraz, że w analizownym układzie po wystąpieniu drugiego impulsu zegarowego połączono wejście $\mathbf{Q_C}$ z wejściem **D** przerzutnika A. (W rejestrze mającym także wejście równoległe połączenie takie można wykonać wcześniej, a następnie wpisać do rejestru z tych wejść słowo kodu **1 z *n***). Przebiegi czasowe w tak zmodyfikowanym układzie podano na rys. 13.3. Jedynka



Rys. 13.3. Przebiegi czasowe w rejestrze szeregowym z rys. 13.1 przy połączeniu wyjścia Q_C z wejściem D_A

w rejestrze będzie krążyła i na wyjściu będziemy ją obserwować co trzeci takt. Przez dwa takty na wyjściu będzie poziom niski (wówczas jedynka będzie w przerzutniku A — pierwszy takt, potem w przerzutniku B — drugi takt). Zauważmy, że częstotliwość przebiegu wyjściowego w stosunku do przebiegu zegarowego jest trzy razy mniejsza — układ taki może być wykorzystany jako dzielnik częstotliwości. Współczynnik wypełnienia przebiegu wyjściowego wynosi 1/3. Wpisując do rejestru dwie jedynki (zamiast jednej), otrzymalibyśmy współczynnik równy 2/3.

Obserwując zachowanie tego układu od strony wyjść $\mathbf{Q_A Q_B Q_C}$ powiemy, że mamy do czynienia z rejestrem szeregowo-równoległym. W pierwszym przypadku (bez połączenia $\mathbf{Q_C}$ z $\mathbf{D_A}$) przebiegi czasowe mają postać jak na rys. 13.4.Na kolejnym rysunku (rys. 13.5) przebiegi czasowe uwzględniają wykonanie połączenia $\mathbf{Q_C}$ z $\mathbf{D_A}$.



Rys. 13.4. Przebiegi czasowe w rejestrze z rys. 13.1 pracującym jako rejestr szeregowo-równoległy

Rys. 13.5. Przebiegi czasowe w rejestrze z rys. 13.1 pracującym jako rejestr szeregowo-równoległy, przy połączeniu wyjścia Q_C z wejściem D_A

Zapiszmy kolejne stany wyjść równoległych na podstawie rys. 13.5. Otrzymamy

Q_C	Q_B	Q_A	
0	**0**	**1**	
0	**1**	**0**	
1	**0**	**0**	
0	**0**	**1**	itd.

Zauważmy, że powtarzają się cyklicznie trzy różne stany wyjść. Układ taki możemy wykorzystać jako licznik **mod 3** liczący w kodzie **1 z 3**. Liczniki tego typu są nazywane **licznikami pierścieniowymi.**

Za pomocą rejestrów przesuwających możemy realizować operację mnożenia przez 2 lub dzielenia przez 2 liczby przedstawionej w naturalnym kodzie dwójkowym. Mnożenie przez dwa (liczby dwójkowej) to po prostu przesunięcie jej w rejestrze w kierunku starszych bitów (w prawo) o jedną pozycję. Dzielenie przez dwa (liczby dwójkowej) to przesunięcie jej o jedną pozycję w kierunku młodszych bitów (w lewo). Przesunięcie w prawo (w lewo) o n pozycji to pomnożenie (podzielenie) liczby binarnej wpisanej do rejestru przez 2^n.

W rejestrach PISO (równoległo-szeregowych) można dokonać zamiany formatu słów binarnych z równoległego na szeregowy. W rejestrach SIPO (szeregowo-równoległych) można dokonać zamiany formatu słów binarnych z szeregowego na równoległy. Para takich rejestrów umożliwia więc budowę układu transmisji szeregowej, działającego analogicznie do opisanego w p. 10.2.3 (rys. 10.8), a zbudowanego z multi- i demultipleksera.

13.3. Rejestry scalone

W niniejszym podrozdziale omówiono wybrane rejestry scalone produkowane w ramach układów TTL i CMOS.

13.3.1. Rejestr scalony '174

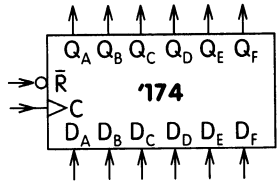
Jako rejestr równoległy, szeregowy bądź szeregowo-równoległy może być wykorzystany układ scalony **'174**, zawierający 6 przerzutników synchronicznych typu *D*. Jego symbol graficzny przedstawiono na rys. 13.6.

Bez dodatkowych połączeń zewnętrznych można go wykorzystać jako rejestr równoległy. Informacja z wejść jest przepisywana dodatnim zboczem sygnału zegarowego, który jest wspólny dla wszystkich przerzutników. Układ ma wyprowadzenie (też wspólne dla wszystkich przerzutników) umożliwiające wyzerowanie rejestru (przerzutników) po doprowadzeniu do niego sygnału o poziomie niskim **L**. Wykonując zewnętrzne połączenia: $\mathbf{Q_A} \rightarrow \mathbf{D_B}$, $\mathbf{Q_B} \rightarrow \mathbf{D_C}$, $\mathbf{Q_C} \rightarrow \mathbf{D_D}$, $\mathbf{Q_D} \rightarrow \mathbf{D_E}$, $\mathbf{Q_E} \rightarrow \mathbf{D_F}$ oraz doprowadzając informację wejściową do wejścia $\mathbf{D_A}$, otrzymujemy **rejestr**:

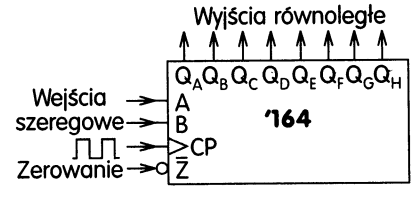
— **szeregowy**, gdy pobieramy informację wyjściową tylko z wyjścia $\mathbf{Q_F}$;
— **szeregowo-równoległy**, gdy pobieramy informację wyjściową ze wszystkich wyjść **Q**.

13.3.2. Rejestr scalony '164

Układ scalony **'164** zawiera 8-bitowy rejestr z wejściem szeregowym i wyjściem równoległym. Jest on zbudowany z 8 synchronicznych przerzutników *RS*, bramki NAND i 3 negatorów. Przerzutniki są tak sterowane, że działają jak przerzutniki typu *D* (patrz p. 7.3.2, rys. 7.15). Są połączone ze sobą kaskadowo. Symbol graficzny rejestru przedstawiono na rys. 13.7.



Rys. 13.6. Symbol graficzny rejestru scalonego '174



Rys. 13.7. Symbol graficzny rejestru scalonego '164

Tablica 13.1. **Rejestr '164**

$\overline{Z}$	CP	A	B	Realizowana funkcja
0	—	—	—	zerowanie rejestru
1	⎍	1	1	przesuwanie w prawo z wpisaniem stanu **1** do przerzutnika A
1	⎍	0	—	przesuwanie w prawo z wpisaniem stanu **0** do przerzutnika A
		—	0	

Rejestr ma wejście zegarowe (**CP**), wejście zerujące ($\overline{\mathbf{Z}}$), bramkowane wejścia szeregowe (**A**, **B**) oraz 8 wyjść. Doprowadzenie impulsów do wejścia zegarowego powoduje, że informacja zapisana w rejestrze jest przesuwana w prawo i jednocześnie z każdym taktem do przerzutnika **A** jest wprowadzana informacja będąca iloczynem logicznym sygnałów **A** i **B** ($\mathbf{Q_A = AB}$). Zmiana stanu rejestru następuje w odpowiedzi na narastające (dodatnie) zbocze przebiegu synchronizującego. Rejestr można wyzerować (asynchronicznie), doprowadzając do wejścia zerującego ($\overline{\mathbf{Z}}$) niski poziom logiczny.

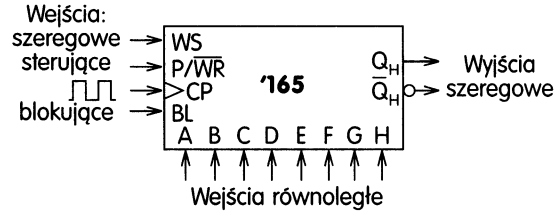
W tablicy 13.1 zestawiono stany poszczególnych wejść tego rejestru oraz odpowiadające tym stanom realizowane przez układ funkcje (czyli jest to tablica działania).

13.3.3. Rejestr scalony '165

Układ scalony **'165** (rys. 13.8) zawiera 8-bitowy rejestr z wejściem szeregowym i równoległym oraz z komplementarnym wyjściem szeregowym. Zawiera on 8 synchronicznych przerzutników *D* (zbudowanych z przerzutników *RS*) i połączonych kaskadowo (podobnie jak w rejestrze **'164**).

Układ ma następujące wejścia:

— równoległe **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**;
— szeregowe **WS**;
— sterujące **P/WR**: $\mathbf{P/\overline{WR} = 0}$ — informacja z wejść równoległych jest wpisywana do rejestru (asynchronicznie); $\mathbf{P/\overline{WR} = 1}$ — informacja jest przesuwana (synchronicznie z przebiegiem zegarowym);
— zegarowe **CP**; zmiana stanu rejestru następuje w odpowiedzi na dodatnie zbocze sygnału taktującego;
— blokujące **BL**; blokuje sygnał zegarowy, gdy jest na nim poziom **1**.



Rys. 13.8. Symbol graficzny rejestru scalonego '165

Tablica 13.2. **Rejestr '165**

BL	CP	P/$\overline{WR}$	Realizowana funkcja
—	—	0	wpisywanie do rejestru stanu wejść równoległych
1	—	1	blokowanie zegara — stan wyjść rejestru (stan przerzutników) nie zmienia się
1	⌋	1	przesuwanie w prawo z wpisaniem stanu z wejścia szeregowego WS do przerzutnika A

Układ ma komplementarne wyjście szeregowe Q_H i $\overline{Q}_H$.

Na rysunku 13.8 pokazano symbol graficzny rejestru **'165**, a w tablicy 13.2 zestawiono stany poszczególnych wejść tego rejestru oraz odpowiadające tym stanom, realizowane przez układ, funkcje.

Rejestr **'165** najczęściej służy do zamiany informacji równoległej na szeregową. W połączeniu z rejestrem **'164** (który zamienia informację szeregową na równoległą) umożliwia realizację transmisji szeregowo-równoległej.

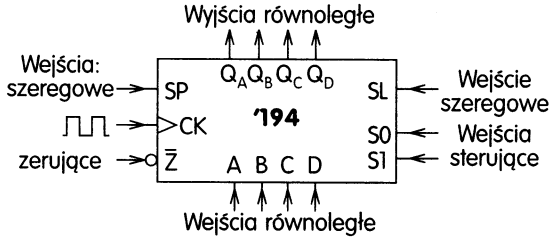
13.3.4. Rejestr scalony '194

Układ scalony **'194** jest 4-bitowym uniwersalnym rejestrem rewersyjnym, wytwarzanym jako układ bipolarny TTL lub unipolarny CMOS (serie HC i HCT). Uniwersalność odnosi się do wprowadzania i wyprowadzania informacji, które to operacje mogą być realizowane zarówno szeregowo jak i równolegle. Nazwa „rewersyjny" określa, że rejestr może przesuwać zapisaną informację w prawo i w lewo, bez wykonywania dodatkowych zewnętrznych połączeń.

Rejestr ma następujące wejścia:
- równoległe **A**, **B**, **C**, **D**;
- zerujące $\overline{\mathbf{Z}}$, zerowanie rejestru następuje, gdy $\overline{\mathbf{Z}} = \mathbf{0}$;
- szeregowe **SP** przy przesuwaniu w prawo;
- szeregowe **SL** przy przesuwaniu w lewo;
- sterujące rodzajem pracy **S0** i **S1**, (patrz tabl. 13.3);
- zegarowe **CK**, synchronizujące dodatnim zboczem pracę rejestru.

Układ ma wyjścia równoległe $Q_AQ_BQ_CQ_D$.

Na rysunku 13.9 pokazano symbol graficzny rejestru **'194**, a w tabl. 13.3 zestawiono stany poszczególnych wejść tego rejestru oraz odpowiadające tym stanom, realizowane przez układ, funkcje.



Rys. 13.9. Symbol graficzny rejestru scalonego '194

Tablica 13.3. **Rejestr '194**

$\overline{Z}$	CK	S0	S1	Realizowana funkcja
0	—	—	—	zerowanie rejestru
1	—	0	0	blokada pracy rejestru
1	⌐	0	1	przesuwanie w prawo z wpisaniem do przerzutnika A stanu wejścia **SP** (Q_A = **SP**)
1	⌐	1	0	przesuwanie w lewo z wpisaniem do przerzutnika D stanu wejścia **SL** (Q_D = **SL**)
1	⌐	1	1	wprowadzanie informacji z wejść równoległych, synchronizowane zegarem

13.3.5. Rejestr scalony '198

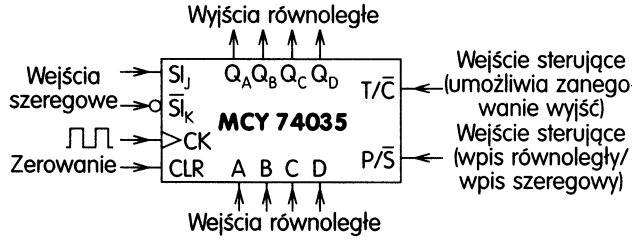
Rejestr ten funkcjonalnie jest identyczny jak rejestr **'194.** Jedyna różnica dotyczy długości rejestru. Układ **'198** jest bowiem rejestrem 8-bitowym. Jest on wytwarzany jako układ bipolarny TTL lub unipolarny CMOS (serie HC i HCT).

Asortyment rejestrów produkowanych w technice CMOS jest nawet bogatszy niż w technice TTL. Zawiera bowiem nawet rejestry 18-, 32- i 64-bitowe. Nie wszystkie one mają polskie odpowiedniki w postaci układów MCY. W podręczniku omówiono tylko jeden rejestr scalony MCY.

13.3.6. Rejestr scalony '035

Układ **'035** (np. MCY74035) jest 4-bitowym rejestrem uniwersalnym. Informacja może być do niego wprowadzana i z niego wyprowadzana zarówno szeregowo, jak i równolegle. Przystosowany jest do realizacji przesuwania w kierunku starszych bitów (MSB). Przesuwanie w kierunku przeciwnym wymaga wykonania odpowiednich zewnętrznych połączeń. Symbol graficzny rejestru przedstawiono na rys. 13.10.

Rejestr jest zbudowany z przerzutników synchronicznych typu *D*, przy czym pierwszy z nich (A) jest przekształcony w przerzutnik typu *JK* i jest wyprowadzone wejście **J** ($\mathbf{SI_J}$) oraz zanegowane wejście **K** ($\overline{\mathbf{SI}}_K$). Taka budowa pierwszego stopnia rejestru umożliwia realizację uzależnień logicznych w przypadku, gdy re-



Rys. 13.10. Symbol graficzny rejestru scalonego MCY74035

Tablica 13.4. **Rejestr MCY74035 (SI = SI_J = $\overline{SI}_K$; T/$\overline{C}$ = 1)**

CLR	CP	P/$\overline{S}$	Realizowana funkcja
1	—	—	zerowanie rejestru (asynchroniczne)
0	⌐	0	przesuwanie w prawo z wpisaniem do przerzutnika A stanu wejścia SI (Q_A = SI)
0	⌐	1	wprowadzanie informacji z wejść równoległych (synchronizowane zegarem)

jestr jest wykorzystywany jako licznik lub generator. Jeżeli wejścia te zostaną połączone, to pierwszy stopień rejestru staje się ponownie przerzutnikiem typu *D* (patrz p.7.3.2, rys. 7.15) i jest to typowe połączenie przy wykorzystywaniu układu jako rejestru.

W tablicy 13.4 zestawiono stany poszczególnych wejść tego rejestru oraz odpowiadające tym stanom, realizowane przez układ, funkcje.

Tablica 13.4 została sporządzona przy założeniu, że oba wejścia szeregowe są ze sobą połączone oraz wejście T/$\overline{C}$ jest w stanie **H**. Zmiana stanu wejścia T/$\overline{C}$ na **L** spowoduje jedynie zanegowanie wszystkich wyjść rejestru. Dotyczy to także operacji zerowania rejestru, która w tej sytuacji spowoduje ustawienie wszystkich wyjść w stan **1**.

Wykonując odpowiednie połączenia zewnętrzne można uzyskać rejestr przesuwający w kierunku młodszych bitów (LSB) informację zapisaną w rejestrze.

13.3.7. Rejestr scalony '373

Rejestr scalony **'373** zawiera 8 przerzutników typu *D*, wyzwalanych poziomem (ang. *latch* — zatrzask) o wyjściach trójstanowych. Układ ten jest wykorzystywany w technice komputerowej.

Pytania i zadania

1. Dany jest rejestr **'164**. Jego wejścia **A**, **B** są połączone ze sobą i sterowane z wyjścia Q_H poprzez negator. Narysuj odpowiedni układ oraz przebiegi czasowe w tym układzie. Wypisz (kolejno) wszystkie stany wyjść i je policz, określając w ten sposób pojemność licznika. (Licznik taki jest nazywany **licznikiem Johnsona**.)
2. Analizując schemat zasadniczy rejestru scalonego **'165** odpowiedz na pytanie, czy zmiana stanu wejść równoległych będzie przepisywana natychmiast do przerzutników, jeżeli wejście sterujące będzie miało cały czas poziom **0**.
3. Wykorzystując rejestry **'164** i **'165** narysuj schemat zasadniczy układu transmisji szeregowo-równoległej.
4. Zbuduj z rejestru **'164** dzielnik częstotliwości przez 10. Narysuj przebiegi czasowe w tym układzie.

   Wskazówka. Zbuduj najpierw licznik Johnsona (zad. 1.), a następnie, dekodując odpowiedni stan wyjść, wyzeruj nim rejestr.

5. Rozwiąż zadanie 4., ale konstruując licznik **mod 12**.
6. Z rejestrów **'194** zbuduj rejestr **'198**.
7. Narysuj układ połączeń rejestru **'194** pracującego jako rejestr szeregowy z przesuwaniem informacji w prawo. Wykreśl przykładowe przebiegi czasowe w tym układzie.
8. Narysuj układ połączeń rejestru **'194** pracującego jako rejestr szeregowy z przesuwaniem informacji w lewo. Wykreśl przykładowe przebiegi czasowe w tym układzie.
9. Narysuj układ połączeń rejestru MCY74030 umożliwiający przesuwanie informacji w lewo. Określ wejście i wyjście szeregowe takiego rejestru. Narysuj przykładowe przebiegi czasowe w tym układzie.
10. Jakie znasz kryteria klasyfikacji rejestrów?
11. Czym różni się szeregowe wprowadzenie informacji do rejestru od równoległego?

14

Pamięci

14.1. Wprowadzenie

Pamięci ***są układami służącymi do przechowywania informacji. Informacja ta jest pamiętana w postaci ciągu słów binarnych.***

Pamięci półprzewodnikowe dzieli się na :

— **pamięci odczyt-zapis** (ang. *Read-Write Memory* — RWM), zwane również **pamięciami o dostępie bezpośrednim** (ang. *Random-Access Memory* — **RAM**):

— **pamięci stałe** — (ang. *Read-Only Memory* — **ROM**), umożliwiające jedynie odczyt zapisanej w niej informacji.

Pojęcie „dostęp bezpośredni" oznacza, że czas dostępu do komórek pamięci jest niezależny od adresu, czyli miejsca informacji w matrycy. Taką cechę mają także pamięci stałe (ROM) i w związku z tym podział na pamięci RAM i ROM nie jest w pełni właściwy. Tym niemniej w literaturze bardziej rozpowszechniona jest nazwa — pamięć o dostępie bezpośrednim (RAM) niż pamięć typu odczyt-zapis (RWM).

Pamięci są budowane jako układy o dużym (LSI) czy nawet bardzo dużym (VLSI) stopniu scalenia. Są wykonywane w technice bipolarnej TTL i ECL oraz unipolarnej CMOS (CMOS-4000B, CMOS-HC) i NMOS. W ostatnich latach obserwuje się szczególnie dynamiczny rozwój pamięci MOS.

Podstawowym parametrem pamięci jest jej **pojemność**, określająca, jak wiele informacji można w niej przechować. Pojemność określa się w bitach [b], kilobitach [Kb], a w ostatnio wytwarzanych — w megabitach [Mb].

Najczęściej pojemność pamięci określa się w bitach. W przypadku dużych pojemności dodajemy przedrostek K — oznaczający liczbę 1024 (2^{10}) i czytany „kilo" przez analogię do przedrostka dodawanego do nazwy jednostek (np. kilo-gram). Ponieważ to kilo nie jest dokładnie równe 1000, więc dla podkre-

ślenia różnicy oznacza się je dużą literą K. Tak więc 1 Kb czytamy jako jeden kilobit. 1Mb czytamy — jeden megabit ($M = K \cdot K = 2^{20} = 1048576b$). Niekiedy pojemność podaje się w bajtach i wtedy B piszemy dużą literą. Na przykład 1KB to jeden kilobajt.

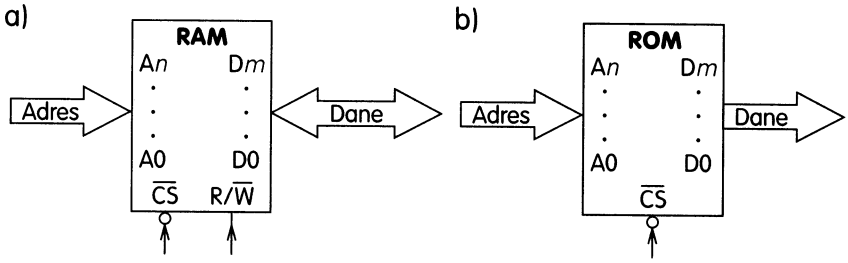
Każda pamięć ma tzw. **wejścia adresowe (A).** Można dzięki nim określić, w którym rejestrze (komórce) będzie zapisywana informacja lub z którego będzie ona odczytywana; n wejść adresowych umożliwia zaadresowanie $N = 2^n$ komórek pamięci. Jeżeli pamięć ma komórki jednobitowe, to jej pojemność jest taka sama jak liczba komórek ($P = N$).

Liczba bitów w komórce świadczy o **organizacji pamięci**. Przez organizację pamięci należy rozumieć sposób dostępu do informacji. Zapis lub odczyt może odbywać się pojedynczymi bitami (tzn., że pamięć ma komórki jednobitowe) i o takiej pamięci powiemy, że ma organizację bitową, a jej pojemność wynosi $N \cdot 1$ b. Komórki mogą zawierać 4 lub 8 (rzadko inną liczbę) bitów. Wówczas mówimy o organizacji słowowej typu $N \cdot 4$ czy $N \cdot 8$. Z organizacją $N \cdot 8$ wiąże się określanie pojemności w bajtach [B], kilobajtach [KB] lub megabajtach [MB]. Na przykład pamięć o 10 wejściach adresowych i komórkach 8-bitowych ma pojemność $P = N \cdot B = 2^{10} B = 1$ KB. Ta sama pojemność wyrażona w bitach, a nie w bajtach (czyli pojemność pamięci o organizacji bitowej) to $P = 8$ Kb.

Każda pamięć ma **wejścia/wyjścia danych (informacyjne D).** Wyjątek stanowią pamięci ROM, które mają jedynie wyjścia. Liczba tych wejść jest równa liczbie bitów komórki pamięci. Taka sama jest liczba wyjść **Y**. Wyjścia w pamięciach są zawsze trójstanowe (TS) lub (w starszych typach pamięci) typu OC. Dzięki temu możliwe jest łączenie ze sobą takich wyjść (różnych modułów) w celu zwiększenia ich pojemności (patrz p. 14.6). Wyjścia takie (TS, OC) można łączyć z magistralą. Układy pamięci o organizacji słowowej mają najczęściej wyprowadzenia danych dwukierunkowe, tzn., że dana końcówka jest wejściem lub wyjściem w zależności od stanu wejść sterujących. W pamięciach o organizacji bitowej wejście i wyjście danych jest najczęściej rozdzielone.

Pamięci, oprócz wejść adresowych oraz wejść informacyjnych, mają także **wejścia sterujące.** Najważniejsze z nich (mają je wszystkie pamięci) to **wejścia uaktywniające pamięć CS** (ang. *Chip Select*) albo **CE** (ang. *Chip Enable*) oraz **wejście zezwalające na zapis WE** (ang. *Write Enable*) albo **WR** (ang. *WRite*). Wiele pamięci ma również wejścia zezwalające na odczyt **OE** (ang. *Output Enable*) albo **RD** (ang. *ReaD*). Niektóre pamięci mają ponadto **wejścia strobujące adresów ALE** (ang. *Addres Latch Enable*), **RAS** (ang. *Row Addres Select*), **CAS** (ang. *Column Addres Select*).

Wejścia sterujące umożliwiają wybór funkcji realizowanej przez układ, np. można przełączyć pamięć w tryb odczytu lub zapisu. Wejście sterujące typu $\overline{\text{CS}}$ pozwala „wyłączyć" pamięć, co jest realizowane poprzez ustawienie wyjść w stan wielkiej impedancji (zmniejsza się również w sposób istotny pobór prądu). Dzięki temu wejściu jest możliwe przyłączanie wielu modułów pamięci do jednej szyny danych (rozbudowa pojemności) lub budowanie pamięci o dłuższych pamiętanych słowach (patrz p. 14.6).



Rys. 14.1. Symbol graficzny pamięci: a) RAM; b) ROM

Symbole graficzne pamięci RAM oraz ROM przedstawiono na rys. 14.1.

Zapis przy wejściu **R/$\overline{\text{W}}$** litery W z negacją oznacza, że tryb „zapis" wymaga ustawienia tego wejścia w stan niski. Stan wysoki na tym wejściu to tryb „odczyt".

14.2. Pamięci typu RAM

Pamięć typu RAM można traktować jako zespół rejestrów równoległych. Informację można zapisać w dowolnym rejestrze i z dowolnego rejestru odczytać (stąd nazwa „pamięć o dostępie bezpośrednim"), dla odróżnienia od pamięci o dostępie sekwencyjnym, utworzonej np. z rejestrów przesuwających.

Każdy taki rejestr jest nazywany **komórką pamięci**. W komórce takiej jest zapamiętywane jedno słowo zwykle o długości 8 bitów (1 bajt). Są budowane także pamięci o komórkach 1- lub 4-bitowych.

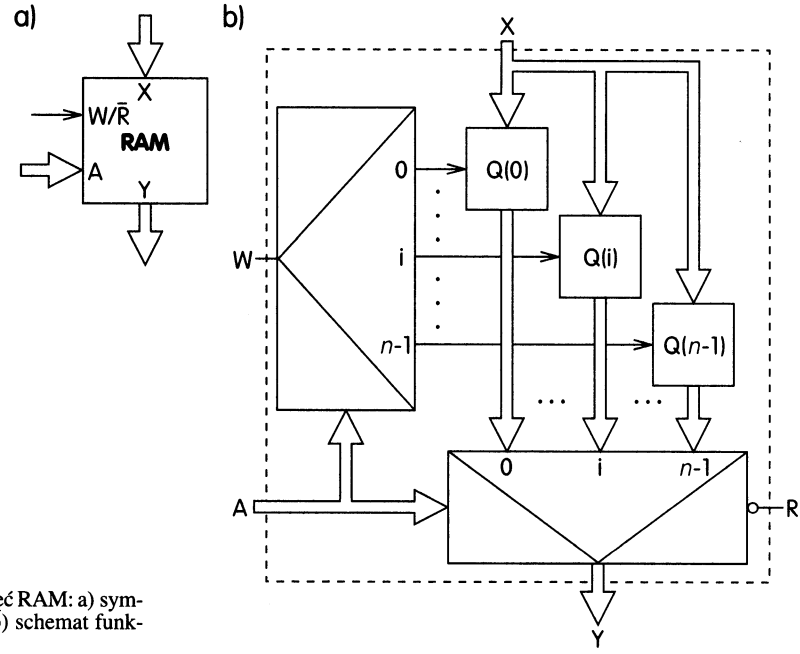
Schemat funkcjonalny pamięci RAM przedstawiono na rys. 14.2. Nie odzwierciedla on rzeczywistej struktury układu, ale ułatwia zrozumienie zasady działania.

Adres A steruje multiplekserem wyjściowym, przesyłając informację z wyjścia wybranego rejestru do wyjścia pamięci. Jednocześnie adres steruje demultiplekserem, dołączając wejście sterujące zapisem do wejścia wpisującego wybranego rejestru.

Pamięci RAM buduje się jako układy bipolarne (TTL, ECL) lub unipolarne (NMOS, CMOS). Pamięci bipolarne mają niewielkie pojemności, potrzebują dużych prądów zasilających i są wolniejsze (TTL) lub tylko nieznacznie szybsze (ECL) od najszybszych pamięci CMOS. Obszar zastosowań pamięci bipolarnych jest zatem coraz mniejszy.

Pamięci unipolarne mogą być **statyczne SRAM** (ang. *Static Random Access Memory*) lub **dynamiczne DRAM** (ang. *Dynamic Random Access Memory*). W pamięciach typu SRAM elementem pamiętającym jest przerzutnik bistabilny, zbudowany z tranzystorów MOS. Elementem pamiętającym w pamięci DRAM jest kondensator podłoże-dren tranzystora MOS. Oczywiście kondensator ten rozładowuje się przez istniejące w układzie upływności i musi być w związku z tym

okresowo doładowywany w celu utrzymania ładunku. Pamięci dynamiczne wymagają więc — w czasie ich użytkowania — „odświeżania" zawartej w nich informacji poprzez okresowe odczytywanie i zapisywanie poszczególnych komórek pamięci (uzupełnianie ładunku elektrycznego). W konsekwencji są potrzebne dodatkowe układy realizujące takie „odświeżanie".



Rys. 14.2. Pamięć RAM: a) symbol graficzny; b) schemat funkcjonalny

Pamięci statyczne są jednak znacznie droższe od pamięci dynamicznych (ok. 5-krotnie) i dlatego systemy cyfrowe wymagające pamięci o dużych pojemnościach są budowane z pamięci dynamicznych.

Technologie budowy pamięci półprzewodnikowych są obecnie chyba najdynamiczniej rozwijającą się dziedziną techniki. Dlatego z każdym rokiem należy oczekiwać nowych, coraz doskonalszych rozwiązań.

Na przykład wada pamięci dynamicznych, polegająca na konieczności jej „odświeżania", została wyeliminowana w niektórych typach pamięci DRAM poprzez wykonanie wewnętrznego układu „odświeżania".

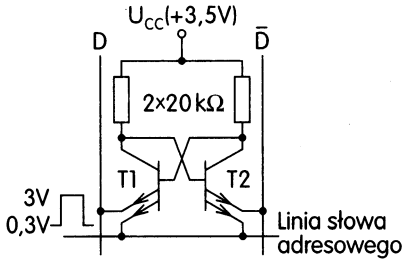
Także wadą pamięci RAM jest utrata informacji w niej zgromadzonej w chwili pozbawienia jej zasilania. Istnieją już rozwiązania pozbawione tej wady. Są to tak zwane **pamięci RAM nieulotne** NOVRAM (ang. *NOn Volatile RAM*).

Często stosowaną grupą pamięci dynamicznych RAM są pamięci przeznaczone do przechowywania obrazów. Są to tak zwane **pamięci VideoRAM**. Podział tych pamięci na dwie grupy jest determinowany przez ich przeznaczenie: jedna — do zastosowań w grafice komputerowej, druga — w cyfrowych odbiornikach TV.

Rzeczywista struktura pamięci typu RAM nie zawiera rejestrów, jakie poznaliśmy w rozdz. 13. Rejestry zastąpiono tak zwaną **matrycą pamięciową**, złożoną z możliwie prostych elementów pamięciowych. Element taki jest nazywany **pod-**

stawową komórką pamięci i służy do zapamiętania 1 bitu informacji. Na rysunku 14.3 przedstawiono schemat zasadniczy takiej komórki wykonanej w technice TTL.

Elementy, z których jest zbudowana podstawowa komórka pamięci RAM wykonana w technice TTL (najczęściej TTL-S lub TTL-LS) tworzą układ przerzutnika bistabilnego. Jeden z dwóch tranzystorów (*T1* lub *T2*) jest w stanie blokowania, drugi natomiast w stanie przewodzenia, a na linii słowa występuje potencjał około 0,3 V. Operacja zapisu wymaga doprowadzenia do linii słowa



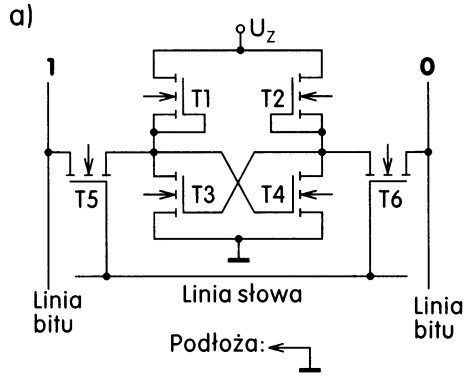
Rys. 14.3. Podstawowa komórka pamięci RAM (TTL)

napięcia +3 V. Aby zapisać stan **1** w przerzutniku, należy potencjał linii *D* obniżyć do około 2 V. Powoduje to przewodzenie tranzystora *T1*. Aby zapisać **0**, należy obniżyć potencjał linii $\overline{D}$.

Operacja odczytu jest podobna do operacji zapisu. Doprowadzamy napięcie +3 V do linii słowa. Wzmacniacz różnicowy, dołączony do linii bitowych, odczytuje stan przerzutnika. Jeżeli tranzystor *T1* przewodzi, to potencjał linii *D* jest niższy, co wzmacniacz różnicowy odczytuje jako stan **1**.

Budowę podstawowej komórki pamięci wykonanej w technice MOS przedstawiono na rys. 14.4. Podstawowa komórka pamięci statycznej MOS (rys. 14.4a) zawiera przerzutnik zbudowany z tranzystorów *T1* ÷ *T4*. Tranzystory *T1* i *T2* pełnią rolę rezystorów i jest to układ całkowicie analogiczny do układu z rys. 14.3.

W przypadku dynamicznej pamięci MOS (rys. 14.4b) elementem pamięciowym jest pojedynczy kondensator. Informacja jest przechowywana w postaci ładunku zmagazynowanego w kondensatorze *C*. Dostęp do informacji uzyskuje się poprzez wybranie linii słowa. Powoduje to przewodzenie tranzystora *T*, co umożliwia odczyt informacji pamiętanej w kondensatorze lub jej zmianę (zapis).



Rys. 14.4. Podstawowa komórka pamięci RAM (MOS): a) statycznej; b) dynamicznej

14.3. Pamięci typu ROM

Pamięć stała (typu ROM) jest to pamięć, której zawartość w czasie normalnej eksploatacji jest niezmienna. Raz zapisana informacja jest trwale przechowywana przez długi czas i może być wielokrotnie odczytywana.

Tego rodzaju pamięci są stosowane np. do: konwersji kodów, realizacji funkcji logicznych, realizacji funkcji arytmetycznych, tablicowania funkcji, budowy generatorów znaków, mikroprogramowania. W klasyfikacji układów cyfrowych pamięć ROM zalicza się do układów kombinacyjnych, w których jak wiemy stan wyjść zależy wyłącznie od stanu wejść. W taki sposób zachowuje się właśnie zaprogramowana pamięć typu ROM. (Podobnie także działa zaprogramowana pamięć typu RAM, ale w niej można zmienić zapisaną wcześniej informację i wówczas takiemu samemu stanowi wejść (adresowi) będzie odpowiadać już inny stan wyjść.)

Pod względem **sposobu programowania** pamięci stałe można ogólnie podzielić na trzy grupy:

— zapisywane przez producenta podczas produkcji pamięci — ROM (dokładniej MROM — ang. *Mask ROM*);

— pamięci programowane w sposób trwały przez użytkownika (przy użyciu specjalnych programatorów) bez możliwości wymazania raz zapisanej informacji — **PROM** (ang. *Programmable ROM*);

— pamięci programowane w sposób prawie trwały z możliwością wymazania (za pomocą specjalnych kasowników) i ponownego zapisu informacji (pamięci reprogramowalne) — **EPROM** (ang. *Erasable PROM*); programowanie i kasowanie odbywa się poza systemem, w którym pracują te pamięci.

Dalszy podział pamięci EPROM wynika ze sposobu **wymazywania informacji**. Pamięci reprogramowalne można podzielić na dwa rodzaje:

— programowane metodą elektryczną, kasowane metodą nieelektryczną przez naświetlanie promieniami *X* (jeżeli obudowa jest nieprzezroczysta) lub promieniami ultrafioletowymi (jeżeli w obudowie jest okienko kwarcowe) — pamięci typu EPROM;

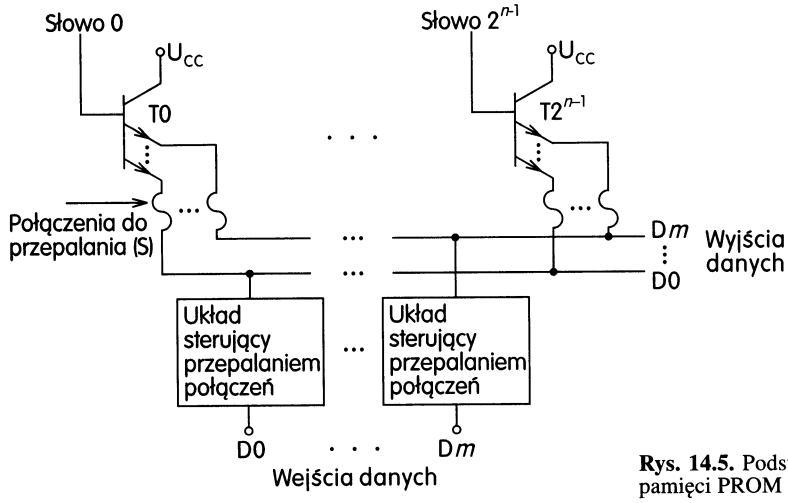
— programowane i kasowane metodą elektryczną — pamięci typu **EEPROM** (ang. *Electrical EPROM*); funkcjonalnie niemal takie jak pamięci RAM, a dodatkowo nie tracą informacji przy wyłączeniu zasilania.

Pamięci PROM to zwykle pamięci bipolarne, natomiast pamięci reprogramowalne (EPROM) są wykonywane w technice MOS.

Nie zaprogramowany układ pamięci ma wszystkie bity ustawione w stan **1** lub w stan **0**. Proces programowania polega na doprowadzeniu krótkiego impulsu programującego do wejścia zapisującego. Podczas programowania szczególne znaczenie — ze względu na możliwość uszkodzenia układu — ma kształt impulsu programującego. Dotyczy to w szczególności takich parametrów impulsu programującego jak: czas trwania, wartość maksymalna (napięcia).

W najprostszej postaci pamięć stała składa się z dekodera adresu wybierającego odczytywane słowo oraz z matrycy pamięciowej. Matryca pamięciowa pod względem realizowanych operacji logicznych stanowi układ bramek OR, z wyjść których jest odczytywane wybrane słowo.

W pamięciach stałych rolę elementów pamięciowych pełnią zwykle pojedyncze tranzystory bipolarne lub MOS. Budowę elementarnej komórki pamięci PROM w technice TTL przedstawiono na rys. 14.5. Wybranie słowa „0" powoduje przewodzenie wieloemiterowego tranzystora *T0*. Emitery tranzystora są dołączone do linii bitów poprzez np. nichromowe połączenia *S* o grubości ok. 20 nm. W celu



Rys. 14.5. Podstawowa komórka pamięci PROM (TTL)

zaprogramowania poszczególnych bitów w słowie (są zapisane same zera) należy wybrać słowo przez podanie odpowiedniego adresu; natomiast do wyjść $0 \div N$, które mają być zaprogramowane w stan **1**, należy doprowadzić napięcie o wartości takiej, aby komparator w układzie sterowania spowodował dołączenie emitera tranzystora do wspólnego potencjału. Jednocześnie należy podwyższyć napięcie U_{CC}, powodując zwiększenie wartości prądu płynącego w wybranych połączeniach *S* i ich przepalenie (przerwanie).

Do grupy pamięci zaliczamy także **programowalne struktury logiczne,** tzw. układy **PLD** (ang. *Programmable Logic Devices*) — patrz p. 14.7. Można je bowiem uważać za odmianę pamięci stałych. Dlatego uwagi dotyczące budowy pamięci typu PROM, EPROM odnoszą się także do układów PLD.

14.4. Parametry dynamiczne pamięci

Właściwości dynamiczne pamięci półprzewodnikowych są określane za pomocą parametrów czasowych (dynamicznych), które charakteryzują zachowanie się tych układów podczas sterowania. Definiowane są one zazwyczaj na podsta-

wie odpowiednich przebiegów czasowych, a ich oznaczenia są niestety odmienne u różnych producentów.

Podstawowymi parametrami dynamicznymi są: **czas dostępu** (t_A — ang. *Access time*) i **czas cyklu** (t_{CY} — *ang. CYcle time*).

Najważniejszym parametrem dynamicznym jest* czas dostępu (t_{AA}), *definiowany jako czas liczony od wystąpienia nowego adresu do pojawienia się na wyjściach układu zawartości komórki pamięci o tym adresie.

Czas dostępu może być definiowany także w odniesieniu do innych wejść niż adresowe. Liczony od wejść wybierających (CS — ang. *Chip Select*) to t_{ACS}, od sygnału odczytu (R — ang. *Read*) — t_{AR}.

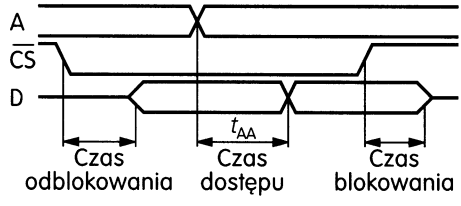
Czasem cyklu ***jest nazywany minimalny odstęp czasu między kolejnymi prawidłowymi zapisami i/lub odczytami pamięci. Niekiedy rozróżnia się czasy: cyklu zapisu, cyklu odczytu i cyklu odczyt-zapis.***

Dla pamięci są podawane (w katalogach) parametry dynamiczne charakterystyczne dla bramek trójstanowych, takie jak czas blokowania (ustawiania wyjść w stan wielkiej impedancji) czy czas odblokowania.

Czas blokowania ***to odstęp czasu liczony od zmiany stanu wejścia wybierającego (CS) do chwili przejścia wyjść/wejść danych w stan wielkiej impedancji.*** **Czas odblokowania** ***jest liczony od zmiany stanu wejścia wybierającego (CS) do chwili przejścia wyjść/wejść danych w tryb pracy dwustanowej.***

Pamięci o wyjściach TS (trójstanowych) powinny być tak wykonane, aby czas blokowania był krótszy niż czas odblokowania. Spełnienie tego warunku zapewnia bezkolizyjną współpracę kilku modułów pamięci w przypadku połączenia ze sobą ich wyjść.

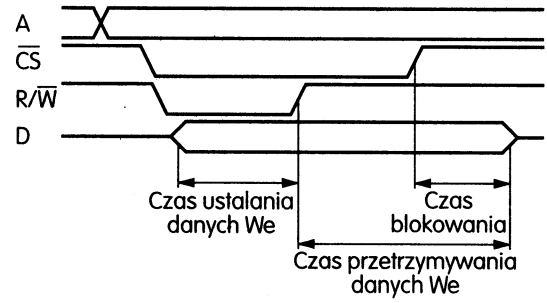
Na rysunku 14.6 przedstawiono przebiegi czasowe odczytu z pamięci oraz zdefiniowano niektóre parametry dynamiczne pamięci.



Rys. 14.6. Odczyt informacji z pamięci

Na rysunku 14.7 przedstawiono przebiegi czasowe zapisu do pamięci i na ich podstawie zdefiniowano wybrane parametry dynamiczne pamięci.

Straty mocy nie są wprawdzie parametrem dynamicznym, ale warto o nich



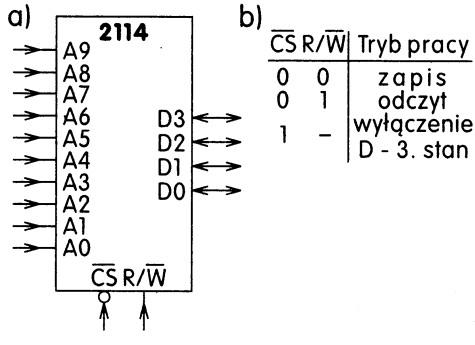
Rys. 14.7. Zapis informacji do pamięci

wspomnieć, bowiem dla pamięci mogą być określane w sposób globalny (dla danego układu) lub w przeliczeniu na bit. Pamięć może mieć dodatkowe wejście zezwalające (ENABLE), ustawiające pamięć w stan spoczynkowy (ang. *standby*), który charakteryzuje się zmniejszonym poborem mocy. Niekiedy ten sam efekt uzyskuje się od strony wejścia CS. Ustawieniu (wyjść) pamięci w stan wielkiej impedancji towarzyszy zmniejszony pobór mocy.

14.5. Charakterystyka wybranych pamięci półprzewodnikowych

Pamięci należą do układów, których rozwój w ostatnich latach jest szczególnie dynamiczny. Podstawowe kierunki rozwoju tych układów to: zwiększenie pojemności, zwiększenie szybkości, zmniejszenie poboru mocy. Spośród wielu układów poniżej przedstawiono jedynie dwa: pamięć typu RAM oraz pamięć typu ROM.

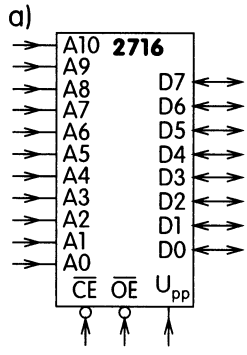
Układ **2114** firmy Intel (produkowany przez CEMI pod nazwą MCY 7114) jest statyczną pamięcią typu RAM o pojemności 4 Kb i organizacji 1024 · 4 (1 K · 4). Ma on 10 wejść adresowych, umożliwiających zaadresowanie 1024 słów czterobitowych. Wyprowadzenia **D** są trójstanowymi wejściami i wyjściami danych. Wejście **R/$\overline{W}$** pozwala ustawić pamięć w tryb pracy odczyt lub zapis. Wejście **$\overline{CS}$** umożliwia odłączenie pamięci (ustawienie w stan wielkiej impedancji). Symbol graficzny pamięci przedstawiono na rys. 14.8.



Rys. 14.8. Pamięć RAM 2114: a) symbol graficzny; b) tablica działania

Parametry pamięci to: czas dostępu ok. 250 ns, czas odblokowania maksimum 85 ns. Impuls zapisujący (poziom niski na wejściu **R/$\overline{W}$**) powinien być nie krótszy niż 135 ns.

Układ **2716** (produkowany przez CEMI pod nazwą MCY 7716) jest pamięcią EPROM o pojemności 16 Kb i organizacji 2 K · 8. Jedenaście wejść adresowych **A0 ÷ A10** umożliwia zaadresowanie jednej z 2048 komórek pamięci, której zawartość jest wysyłana do wyprowadzeń **D0 ÷ D7**. Wyprowadzenia te służą również jako wejścia danych przy programowaniu. Układ ma wejścia $\overline{CE}$ i $\overline{OE}$ wyboru trybu pracy oraz dodatkowe wejście U_{pp}, służące do programowania pamięci. Symbol graficzny pamięci przedstawiono na rys. 14.9.



b)

$\overline{CE}$	$\overline{OE}$	U_{pp}	Tryb pracy
0	0	+5V	odczyt
0	1	+5V	3. stan
1	0	+5V	„standby"
⎍	0	+25V	zapis
0	0	+5V lub +25V	weryfikacja

Rys. 14.9. Pamięć EPROM 2716: a) symbol graficzny; b) tablica działania

Gdy na wejściu $\overline{CE}$ jest poziom wysoki, wówczas układ jest w stanie zmniejszonego poboru mocy (*standby*). Pobór prądu maleje z wartości ok. 60 mA do ok. 10 mA.

Układ jest wyposażony w okienko kwarcowe, przez które można wykasować pamięć za pomocą naświetlania jej promieniami ultrafioletowymi. Pamięć wykasowana ma wszystkie bity ustawione w stan **1**.

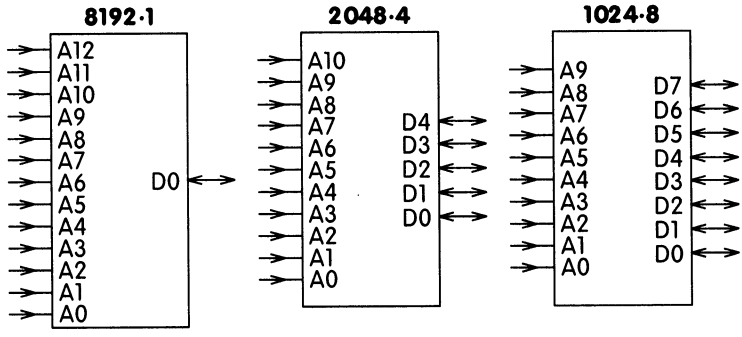
Zapis pamięci jest możliwy po doprowadzeniu do wyprowadzenia U_{pp} napięcia +25 V. Impuls (dodatni) na wejściu $\overline{CE}$ powinien być nie krótszy niż 45 ns, ale nie dłuższy niż 55 ns. Dłuższy impuls może uszkodzić pamięć. Krótszy impuls może sprawić, że zapis nie nastąpi. W trakcie operacji zapisywania można dokonać na bieżąco (bez odłączania napięcia +25 V) weryfikacji dokonanego zapisu — poprzez ustawienie wejść sterujących w taki stan, jak przy odczycie. Aby zapewnić poprawny zapis, należy ustawić adres komórki oraz wejście w stan aktywny co najmniej 2 ms przed podaniem impulsu zapisującego.

Czas dostępu tej pamięci ma wartość ok. 450 ns. Czas blokowania wynosi 100 ns, a czas odblokowania 150 ns.

14.6. Łączenie modułów pamięci

Określenie pojemności należy uzupełnić informacją o **organizacji pamięci,** tzn. liczbie wejść adresowych czy długości pamiętanego słowa (liczbie bitów dostępnych pod jednym adresem). Tak więc pamięć o pojemności 8 Kb może mieć

organizację: 1024 · 8, 2048 · 4 czy 8192 · 1. Pierwsza liczba w takim zapisie oznacza liczbę pamiętanych słów, druga zaś długość pamiętanego słowa wyrażoną w bitach. Określenie organizacji pamięci w powyższy sposób jednoznacznie informuje o liczbie wejść adresowych oraz o liczbie wyjść. Symbole ilustrujące różne sposoby organizacji pamięci o pojemności 8 Kb przedstawiono na rys. 14.10.

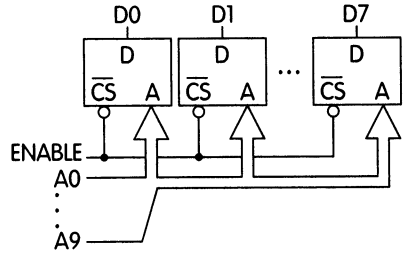


Rys. 14.10. Przykłady organizacji pamięci 8 Kb

Poniżej rozpatrzymy dwa typowe problemy techniczne związane z wykorzystaniem pamięci. Pierwszy z nich to zwiększenie pojemności pamięci poprzez zwiększenie długości pamiętanego słowa. Drugi to zwiększenie pojemności pamięci bez zmiany liczby zapamiętywanych bitów w jednej komórce.

Przykład 14.1

Dysponując pamięciami 1 Kb o organizacji 1024 · 1, zbudować blok pamięci o pojemności 8 Kb (1 KB) i organizacji 1024 · 8.



Rys. 14.11. Powiększanie pojemności pamięci poprzez zwiększenie długości pamiętanego słowa

Rozwiązanie tego zadania przedstawiono na rys. 14.11. Jak widać, wystarczy połączyć ze sobą odpowiednie wejścia adresowe oraz sterujące (także **R/$\overline{\mathbf{W}}$**, jeżeli miałaby to być pamięć typu RAM). W ten sposób można uzyskać dowolną długość pamiętanego słowa. Należy jedynie pamiętać, aby nie przeciążyć wyjść układów, które sterują wejściami tych pamięci. ■

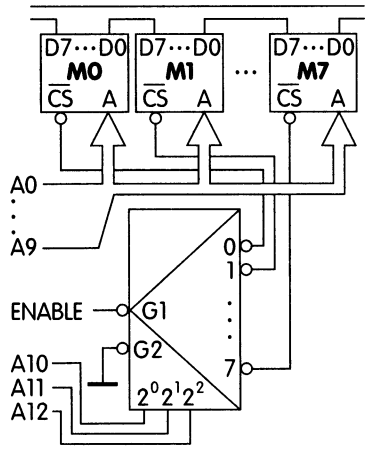
Przykład 14.2

Dysponując pamięciami 8 Kb o organizacji 1024 · 8 zbudować blok pamięci o pojemności 64 Kb (8 KB) i organizacji 8192 · 8.

Wejścia $\overline{\text{CS}}$ poszczególnych pamięci (rys. 14.12) są sterowane w tym układzie z wyjść demultipleksera (dekodera). Tylko jedno z wyjść demultipleksera jest w danej chwili w stanie niskim. Moduły pamięci podłączone do pozosta-

łych wyjść demultipleksera są odłączone (są w stanie wielkiej impedancji). Dzięki temu jest możliwe połączenie ze sobą odpowiednich wyjść danych. Ustawienie w układzie sygnału **ENABLE** (doprowadzonego do wejścia G_1 demultipleksera) w stan **1** blokuje cały blok pamięci — wszystkie moduły są ustawione w stan wielkiej impedancji.

Wejścia adresowe demultipleksera są dodatkowymi wejściami adresującymi blok pamięci. Najstarsze bity adresu wybierają w ten sposób moduł pamięci, który ma być aktywny. Najniższe i najwyższe adresy poszczególnych modułów zestawiono w tabl. 14.1. (**Uwaga**. Adresy w zapisie szesnastkowym.)



Rys. 14.12. Powiększanie pojemności pamięci poprzez zwiększenie liczby pamiętanych słów

Tablica 14.1. **Mapa pamięci układu z rys. 14.12**

Moduł	Adres początkowy	Adres końcowy
M0	0000	03FF
M1	0400	07FF
M2	0800	0BFF
M3	0C00	0FFF
M4	1000	13FF
M5	1400	17FF
M6	1800	1BFF
M7	1C00	1FFF

W obu powyżej opisanych układach rozszerzania pojemności pamięci użycie modułów typu RAM wymagałoby jedynie połączenia ze sobą wejść **R/$\overline{\text{W}}$**. ■

14.7. Programowalne struktury logiczne

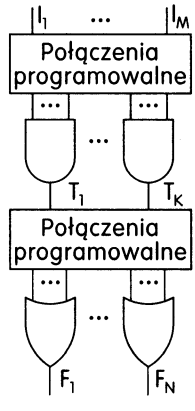
Projektowanie systemów cyfrowych może być realizowane przy zastosowaniu standardowych, uniwersalnych elementów wielkiego stopnia scalenia, uzupełnianych elementami małego i średniego stopnia scalenia. Układ tak zbudowany składa się z wielu odrębnych, aczkolwiek typowych elementów.

Olbrzymi postęp technologiczny w wytwarzaniu układów scalonych pozwala obecnie na stosowanie specjalizowanych układów scalonych, wytwarzanych na zamówienie projektanta systemów cyfrowych, tzw. **układów ASIC** (ang. *Aplication Specific Integrated Circuits*). Jeden taki układ umożliwia realizację całego, nawet bardzo złożonego systemu cyfrowego. Koszt zaprojektowania układu ASIC jest jednak nadal dość wysoki i rozwiązanie takie znajduje ekonomiczne uzasadnienie wówczas, gdy serie urządzeń zawierających te układy są wystarczająco duże. Poza względami ekonomicznymi istotny wpływ na decyzję zamówienia układów ASIC może mieć ich duża niezawodność oraz lepsze parametry techniczne (np. dynamiczne) w porównaniu z klasycznym rozwiązaniem.

Rozwiązaniem pośrednim pomiędzy wymienionymi powyżej technikami (elementy standardowe — elementy ASIC) są **programowalne moduły logiczne PLD** (ang. *Programmable Logic Devices*). Są to układy o standardowej strukturze, które można jednak dostosować do potrzeb użytkownika poprzez ingerencję w tę strukturę. Za pierwowzór tego typu układów można uznać pamięci PROM, które umożliwiają jedynie budowę układu kombinacyjnego. Niektóre rodzaje programowalnych modułów logicznych wyposażono w przerzutniki, dzięki czemu można z nich budować dowolne układy sekwencyjne.

Programowalne moduły logiczne są wytwarzane w technice TTL lub CMOS. Jednym z zasadniczych elementów konstrukcyjnych tych układów jest matryca tranzystorów (bi- lub unipolarnych). Tranzystory te są ułożone w postaci dwóch matryc: matrycy bramek AND i matrycy bramek OR. Schemat poglądowy takich matryc przedstawiono na rys. 14.13. Bramki AND umożliwiają realizację dowolnych iloczynów sygnałów wejściowych. Iloczyny te, sumowane w bramkach OR, umożliwiają realizację dowolnej funkcji kombinacyjnej. (Patrz: kanoniczna postać sumy — p. 3.3 oraz alternatywna postać minimalna — p. 3.4.)

Przystosowanie takiej struktury, jak przedstawiona na rys. 14.13, do realizacji określonego układu cyfrowego polega na przerwaniu (przepaleniu) odpowiednich połączeń (w przypadku układów programowalnych) lub na wprowadzeniu ładunku w obszar dielektryka bramki tranzystora MOS (w przypadku układów reprogramowalnych). Występuje tu analogia do pamięci PROM i EPROM.



Rys. 14.13. Struktura układów PLD

Moduły PLD mogą być dodatkowo wyposażone w przerzutniki, układy wejściowo-wyjściowe, bufory, sprzężenia zwrotne, wyjścia trójstanowe, ale zasadniczą ich częścią są matryce AND i OR. Są budowane także układy PLD o innej strukturze, tzw. komórkowej, czyli analogicznej do budowy pamięci. Poniżej przedstawiono jedynie krótką charakterystykę układów typu matrycowego. Więcej informacji na ten temat Czytelnik może znaleźć w publikacji [9].

W układach PLD typu matrycowego mogą być programowalne obie matryce (AND, OR) bądź tylko jedna z nich. Zestawienie takich układów podano w tabl. 14.2.

Tablica 14.2. **Układy PLD typu matrycowego**

Matryca	Typ modułu		
	PAL	PLA	PLE
AND	programowalna	programowalna	nieprogramowalna
OR	nieprogramowalna	programowalna	programowalna

- **Układy PAL**

Układy **PAL** (ang. *Programmable Array Logic*) były początkowo wykonywane wyłącznie jako układy bipolarne. Ich niska cena oraz duża szybkość działania sprawia, że są nadal produkowane i stosowane.

Obecnie jednak coraz szerzej są stosowane układy PAL wykonane w technice CMOS. Należą do nich układy **EPLD** (ang. *Erasable Programmable Logic Devices*) kasowalne promieniowaniem ultrafioletowym, co wymaga wykonania w nich okienka kwarcowego (analogicznie jak w pamięci EPROM). Układy **GAL** (ang. *Generic Array Logic*) są wykonane w technologii EECMOS (ang. *Electrically Erasable CMOS*). Moduły te są kasowane elektrycznie, dzięki czemu nie wymagają okienka kwarcowego, co obniża koszt ich wytwarzania. Jak wszystkie układy wykonane w technice CMOS, charakteryzują się niewielkim poborem prądu. Ich szybkość działania jest natomiast porównywalna z układami wykonanymi w technice bipolarnej. Najszybsze produkowane obecnie układy wykonane w technice EECMOS mają czas propagacji ok. 12 ns.

Moduły GAL 16V8 firmy LATTICE, oprócz matrycy bramek logicznych, zawierają przerzutniki synchroniczne typu D oraz multipleksery. Układy GAL były pierwszymi układami PLD typu PAL zrealizownymi w technice EECMOS. Układy PAL wykonane w technice EECMOS produkują także i inne firmy. Na przykład firma AMD swój produkt oznacza symbolem PALCE, natomiast firma ICT symbolem PEEL. Niektóre produkty tych firm są ze sobą zgodne i wówczas ich oznaczenia cyfrowo-literowe (następujące po symbolach: GAL, PALCE, PEEL) są identyczne.

- **Układy PLA**

Podstawowym wyposażeniem układów typu PLA są dwie programowalne matryce: AND i OR. Proste struktury PLA (np. Signetics PLS100, PLS101, PLS161), oprócz podstawowych matryc AND, OR, zawierają: programowalny układ polaryzacji wyjść (proste, zanegowane) oraz sterowany z zewnątrz trójstanowy bufor wyjściowy. Są to układy o kilkunastu wejściach, kilkudziesięciu liniach iloczynu i kilku wyjściach. Są produkowane w wersjach z wyjściami trójstanowymi (TS) lub wyjściami typu otwarty kolektor (OC).

Moduły PLA wyposażone w przerzutniki są nazywane układami **PLS** (ang. *Programmable Logic Sequencer*), albo **sekwenserami**. Przykładem takiego sekwensera może być układ PLS155 firmy Signetics.

- **Układy PLE**

Programowalne matryce **PLE** (ang. *Programmable Logic Element*) to nowa generacja szybkich pamięci PROM przystosowanych do syntezy układów logicznych. Od pamięci PROM różnią się większymi możliwościami funkcjonalnymi, a także większą szybkością działania (czas propagacji 15 ÷ 35 ns).

Prawie wszystkie układy PLE mają wyjścia trójstanowe (TS). Prostsze moduły PLE (np. PLE5P8) mają 5 wejść i 8 wyjść, duże (np. PLE12P8) — 12 wejść i 8 wyjść.

Bardziej rozbudowane stuktury PLE są wyposażone w buforowe rejestry wyjściowe, zbudowane z synchronicznych przerzutników typu D z trójstanowymi wyjściami sterowanymi bezpośrednio (asynchronicznie) lub pośrednio, za pomocą dodatkowego przerzutnika (synchronicznie). Ponadto układy te mają dodatkowe wejścia, służące do programowania tzw. inicjalizacji. Umożliwia ono zapisanie w określonym obszarze matrycy PLE sekwencji 16 słów, a następnie ich wygenerowanie w rejestrze wyjściowym.

Do optymalnego wykorzystania układów PLD nie wystarczają jednak klasyczne metody projektowania, stosowane w przypadku użycia układów małego i średniego stopnia scalenia. Niezbędny jest tu komputerowy system projektowania **CAD** (*ang. Computer-Aided Design* — komputerowe wspomaganie projektowania). Wraz z rozwojem układów PLD powstało i nadal powstaje wiele komputerowych systemów ich projektowania i programowania. Należą do nich między innymi: system ABEL firmy Data I/O Corp., CUPL firmy Logical Devices, PALASM firmy MMI, czy opracowany w Instytucie Telekomunikacji Politechniki Warszawskiej system PLATO.

Pytania i zadania

1. Pamięć jest układem sekwencyjnym czy kombinacyjnym? Co to jest pamięć RAM i pamięć ROM? Co oznacza pojęcie: pamięć o dostępie bezpośrednim? Czy pamięci RAM, ROM należą do klasy pamięci o dostępie bezpośrednim?
2. Jakie wejścia i jakie wyjścia mają pamięci? Czym, pod tym względem, różnią się pamięci RAM i ROM?
3. Zdefiniuj takie parametry, jak: czas dostępu, czas blokowania, czas odblokowania. Do jakiej niepożądanej sytuacji może doprowadzić większy czas blokowania niż czas odblokowania?
4. Narysuj symbol graficzny pamięci RAM o organizacji:
   a) 256 · 1,
   b) 1024 · 4,
   c) 16384 · 8.
5. Wykonaj zadanie 4. dla pamięci ROM.
6. Dysponując dowolną liczbą układów pamięci UCY780101 (publikacja [1]), zbuduj pamięć o organizacji:
   a) 16 słów 8-bitowych,
   b) 32 słowa 4-bitowe,
   c) 32 słowa 8-bitowe.

   Schematy uzupełnij bramkami, umożliwiającymi uzyskanie na wyjściach informacji w postaci prostej, a nie zanegowanej.
7. Dysponując pamięciami o pojemności 8 KB i organizacji 8192 · 8, zbuduj pamięć o pojemności 32 KB i organizacji 32768 · 8. Określ mapę pamięci dla tego układu.

15

Przykłady złożonych układów cyfrowych

15.1. Wprowadzenie

Poznane układy MSI (średniego stopnia scalenia) umożliwiają budowę złożonych systemów cyfrowych. Bloki funkcjonalne wchodzące w skład takich systemów mogą być różne, a działanie takiego systemu nie jest prostą wypadkową (sumą funkcji) bloków składowych, lecz zazwyczaj stanowi nową, odrębną funkcjonalnie jakość.

W tym rozdziale przedstawiono przykład projektowania układu zawierającego bloki funkcjonalne oraz przykład analizy działania złożonego układu cyfrowego.

15.2. Stoper (czasomierz)

Przyjmijmy, że naszym zadaniem jest zbudowanie stopera cyfrowego z możliwością pomiaru tzw. **międzyczasu**. Obsługa tego stopera ma być identyczna jak obsługa stopera mechanicznego, który jest wyposażony jedynie w dwa przyciski monostabilne, tzn. powracające samoczynnie do położenia spoczynkowego po zwolnieniu nacisku. Stoper jest przeznaczony do użytku w warunkach laboratoryjnych. Powinien być wyposażony we własny zasilacz sieciowy.

Przyciski stopera powinny realizować następujące funkcje:

— przycisk **S** — start/stop/zerowanie,
— przycisk **M** — wskazanie międzyczasu.

W założeniach musimy jeszcze określić, z jaką rozdzielczością ma być mierzony czas — tzn. minimalny przedział czasu, jaki chcemy, aby był możliwy do zmierzenia oraz wskazania przez stoper. Jeżeli przyjęliśmy, że obsługa stopera ma być ręczna, to wystarczy ograniczyć się do rozdzielczości wynoszącej 0,1 s. Musimy także określić maksymalny przedział czasu, jaki chcemy mierzyć. Przyjmijmy, że będzie to 1 godzina. Wyświetlacz stopera (który zrealizujemy ze wskaźników 7-segmentowych) powinien więc zawierać 5 pól: dwa do wskazywania minut ($0 \div 59$), dwa do wskazywania sekund ($0 \div 59$) i jedno do wskazywania dziesiątych części sekundy ($0 \div 9/10$).

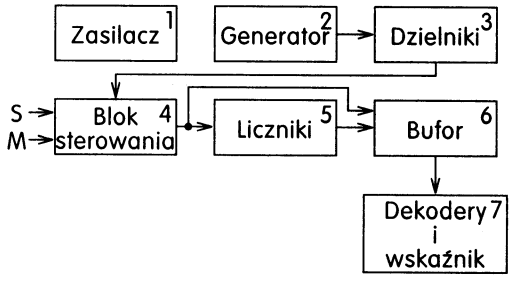
Cykl podstawowy pracy stopera powinien być następujący (stan początkowy — stoper wyzerowany):

1. **Start** (naciśnięcie przycisku **S**) — rozpoczęcie zliczania, wskaźnik wskazuje bieżący upływ czasu;
2. **Stop** (ponowne naciśnięcie przycisku **S**) — zatrzymanie zliczania, wskazanie odmierzonego czasu;
3. **Zerowanie** (kolejne naciśnięcie przycisku **S**) — wyzerowanie wskazania, a tym samym powrót do stanu początkowego.

Podstawowy cykl pracy może zostać „zaburzony" kilkakrotnym (liczba ta może być dowolna) użyciem przycisku **M** w dowolnym stanie pracy stopera. Dlatego musimy określić zachowanie się stopera w wyniku użycia przycisku **M** we wszystkich trzech stanach pracy:

1. Stoper wyzerowany (w stanie początkowym) — naciśnięcie przycisku **M** nie ma żadnego wpływu na stan stopera.
2. Stoper w stanie zliczania — naciśnięcie przycisku **M** powoduje wskazanie aktualnego upływu czasu (od naciśnięcia przycisku **S** do naciśnięcia przycisku **M**) bez przerywania jednak odliczania czasu.
   a) Kolejne naciśnięcie przycisku **M** sprawia, że wskaźnik przechodzi ponownie do wskazywania bieżącego upływu czasu.
   b) Naciśnięcie przycisku **S** — zatrzymanie zliczania i wyświetlenie na wskaźniku odmierzonego czasu (od chwili startu do zatrzymania).
3. Zatrzymanie zliczania — stoper wskazuje odmierzony czas — naciśnięcie przycisku **M** nie ma żadnego wpływu na stan stopera.

Schemat układu stopera będzie dość złożony i dlatego określimy najpierw jego schemat funkcjonalny, definiując dokładnie funkcje poszczególnych bloków. Schemat funkcjonalny stopera przedstawiono na rys. 15.1.



Rys. 15.1. Schemat funkcjonalny stopera

Funkcje poszczególnych bloków oraz współpraca pomiędzy nimi powinna przedstawiać się następująco:

- Zasilacz (blok 1.)

Zadaniem zasilacza jest dostarczenie energii elektrycznej do wszystkich układów (bloków). Aby nie zaciemnić schematu funkcjonalnego zrezygnowano z narysowania połączeń bloku zasilacza z pozostałymi blokami stopera.

- Generator (blok 2.)

Generator ma być źródłem przebiegu prostokątnego o stałej, dokładnie określonej częstotliwości. Przebieg wyjściowy z generatora jest doprowadzony do bloku dzielników (3).

- Dzielniki (blok 3.)

Zadaniem bloku dzielników jest taki podział częstotliwości przebiegu doprowadzonego do jego wejścia (sygnału uzyskanego w generatorze), aby na wyjściu bloku uzyskać przebieg wzorcowy o częstotliwości 10 Hz (czyli okresie równym 0,1 s).

- Blok sterowania (4.)

Zadaniem bloku sterowania jest generowanie impulsów zerujących liczniki w bloku liczników, sterowanie procesem zliczania impulsów pochodzących z bloku dzielników, sterowanie buforem.

- Liczniki (5.)

Liczniki zliczają impulsy przebiegu wzorcowego, przy czym za początek, koniec zliczania oraz zerowanie liczników jest odpowiedzialny blok sterowania.

- Bufor (6.)

Jest to blok pamięci, służący do pamiętania wartości chwilowej upływu czasu, czyli tzw. międzyczasu.

- Dekodery i wskaźnik (7.)

W dekoderach kod **naturalny BCD** jest zamieniany na kod **wskaźnika 7-segmentowego**. Wskaźnik zaś służy do wizualizacji odmierzonego czasu (bądź międzyczasu).

Projektowanie poszczególnych bloków stopera rozpoczniemy od bloku 2., czyli bloku generatora.

- **Generator (blok 2.)**

Generator jest jednym z elementów odpowiedzialnych za dokładność odmierzanego czasu. Im bardziej stabilna będzie częstotliwość generowanego przebiegu, tym dokładniej będzie wyznaczony przez niego wzorcowy odcinek czasu (okres). Stabilność generowanego przebiegu może zapewnić generator kwarcowy (patrz rys. 8.44), w którym element piezoelektryczny (kwarc) stabilizuje w dość wąskim zakresie częstotliwość generowanego przebiegu. Na rysunku 8.44 pokazano kilka generatorów kwarcowych o różnych częstotliwościach przebiegu generowanego. Naszym celem jest uzyskanie wzorcowego przebiegu o częstotliwości 10 Hz, czyli odcinka czasu o długości 0,1 s. Im większą częstotliwość będzie miał generator, tym bardziej złożony będzie blok dzielników.

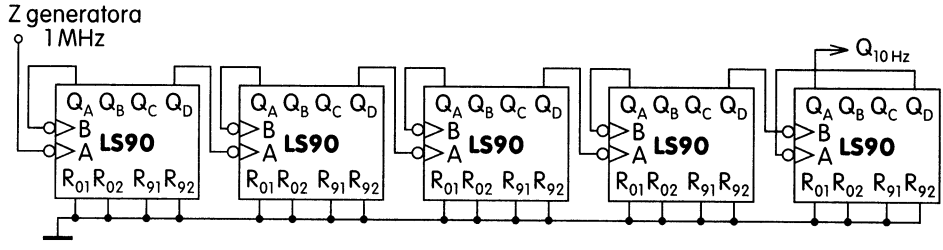
Jakie natomiast przesłanki przemawiają za tym, aby użyć generatora kwarcowego o większej częstotliwości? Załóżmy, że rozważamy budowę generatora o $f_1 = 100$ kHz lub $f_2 = 10$ MHz. Przyjmijmy, że oba generatory mają podobną zmienność częstotliwości generowanego przebiegu określoną jako Δf, przy czym Δf to maksymalne zmiany częstotliwości generowanego przebiegu. Aby z przebiegu o częstotliwości $f_1 = 100$ kHz uzyskać przebieg o $f = 10$ Hz, należy dopro-

wadzić go do dzielnika przez 10 000. Aby z przebiegu o częstotliwości $f_1 = 10$ MHz uzyskać przebieg o $f = 10$ Hz, należy doprowadzić go do dzielnika przez 1 000 000. W obu przypadkach zmniejsza się w tej samej proporcji zmienność częstotliwości Δf. Zmienność ta będzie zdecydowanie mniejsza (dokładnie 100 razy) dla przebiegu o $f = 10$ Hz uzyskanego z przebiegu o większej częstotliwości. W powyższych rozważaniach przyjęliśmy, że Δf jest jednakowa w obu generatorach, co w ogólnym przypadku nie musi być spełnione. Tym niemniej, jeżeli nawet Δf dla generatora o $f_2 = 10$ MHz jest większe niż dla generatora o $f_1 = 100$ kHz, to w praktyce w wyniku podziału przez większą liczbę ostateczny wynik jest korzystniejszy dla generatora o większej częstotliwości.

Dla naszych potrzeb wybierzmy generator, którego schemat zasadniczy przedstawiono na rys. 8.44a. Częstotliwość generowanego przebiegu ustawimy na wartości $f = 1$ MHz, regulując w tym celu trymerem.

- **Dzielniki (blok 3.)**

Do budowy dzielnika użyjemy liczników asynchronicznych np. 74LS90. Potrzebujemy częstotliwość 1 MHz podzielić przez 100 000, aby otrzymać przebieg o częstotliwości $f = 10$ Hz. W tym celu należy zbudować dzielnik złożony z pięciu dekad. Schemat dzielnika przedstawiono na rys. 15.2. Ostatnią dekadę dzielnika połączono tak, aby uzyskać przebieg wyjściowy o wypełnieniu 0,5. Sygnał wyjściowy bloku dzielników oznaczmy przez $\mathbf{Q}_{10Hz}$.



Rys. 15.2. Układ dzielnika przez 100 000

- **Liczniki (blok 5.)**

Zadaniem bloku liczników będzie zliczanie impulsów przebiegu wzorcowego ($\mathbf{Q}_{10Hz}$) o częstotliwości 10 Hz, uzyskanego na wyjściu bloku dzielników. Procesem zliczania (początek, koniec, zerowanie liczników) będzie sterować układ sterowania. Każdy zliczony impuls to 0,1 s, zatem 10 impulsów to 1 s, 600 impulsów to 1 minuta i 36 000 impulsów to 1 godzina. Zgodnie z założeniem maksymalny odcinek czasu, jaki ma mierzyć stoper, to 1 godzina.

Dla uproszczenia procesu dekodowania blok 5. zbudujemy z następujących liczników:

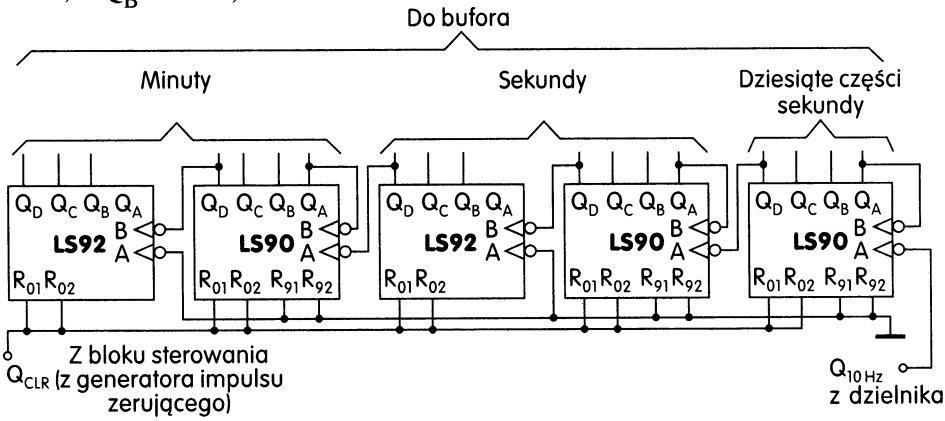
I. Licznik dziesiątych części sekundy liczący w trybie **mod 10**, np. dekada 74LS90. Przebiegiem wejściowym (zliczanym) tego licznika będzie przebieg wzorcowy pobierany z wyjścia bloku dzielników, ale za pośrednictwem blo-

ku sterowania, który określi chwilę rozpoczęcia i zakończenia procesu zliczania. Jeden pełny cykl pracy tego licznika to 1 sekunda.

II. Licznik sekund liczący w trybie **mod 60**. Przebiegiem wejściowym (zliczanym) tego licznika będzie przebieg pochodzący z wyjścia licznika I. Jeden pełny cykl pracy tego licznika to 1 minuta. Licznik ten zbudujemy z dwóch liczników: **mod 10** i **mod 6.** Jako licznika **mod 10** użyjemy układu np. 74LS90, a jako licznika **mod 6** — część układu 74LS92, który zawiera dwa liczniki: **mod 2** i **mod 6.**

III. Licznik minut liczący w trybie **mod 60**. Przebiegiem wejściowym (zliczanym) tego licznika będzie przebieg pochodzący z wyjścia licznika II. Jeden pełny cykl pracy tego licznika to 1 godzina. Licznik ten zbudujemy identycznie jak licznik II. Można go także zbudować z dwóch dekad i wówczas nasz stoper będzie odliczał maksymalnie 100 minut.

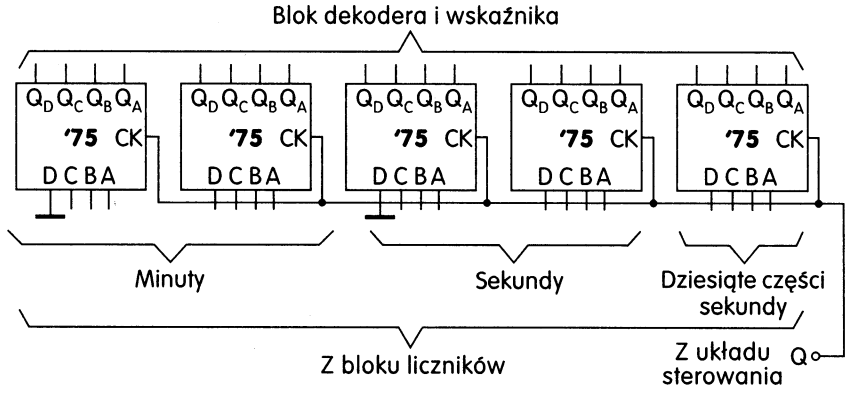
Schemat zasadniczy bloku liczników przedstawiono na rys. 15.3. Pamiętajmy, że wejście licznika **mod 6** w układzie 74LS92 to wejście oznaczone u nas przez **B** oraz, że wyjściami licznika **mod 6** są wyjścia $\mathbf{Q_D Q_C Q_B}$ (przy czym $\mathbf{Q_D}$ to MSB, a $\mathbf{Q_B}$ to LSB).



Rys. 15.3. Blok liczników

- **Bufor (blok 6.)**

Bufor będzie pełnił rolę pamięci międzyczasu. Układ sterowania pod wpływem sygnału generowanego przyciskiem **M** powinien spowodować zapamiętanie aktualnego stanu liczników w buforze. W pozostałych stanach pracy stopera stan liczników powinien być przenoszony przez blok bufora. Do tego celu najlepiej nadają się przerzutniki typu „zatrzask". Układ **7475** zawiera 4 takie przerzutniki. Dla każdego licznika (z bloku liczników) zliczającego **mod 10** są potrzebne 4 przerzutniki (1 układ **7475**), a dla każdego licznika **mod 6** są potrzebne 3 przerzutniki (3/4 układu **7475**). W rezultacie do budowy bloku 6. musimy użyć pięciu układów **7475**. Schemat bloku 6. pokazano na rys. 15.4. Zapamiętajmy (będzie to potrzebne przy projektowania bloku sterującego), że bufor zbudowany z układów **7475** wymaga sygnału sterującego (wejście zegarowe **C**) o poziomie **H** dla jego otwar-



Rys. 15.4. Układ bufora

cia oraz zmiany tego sygnału z **H** na **L** dla „zatrzaśnięcia" informacji wejściowej w buforze (zamknięcia bufora). Bufory **7475** współpracujące z wyjściami liczników **mod 6** są wykorzystywane jedynie w 3/4.

- **Dekodery i wskaźnik (blok 7.)**

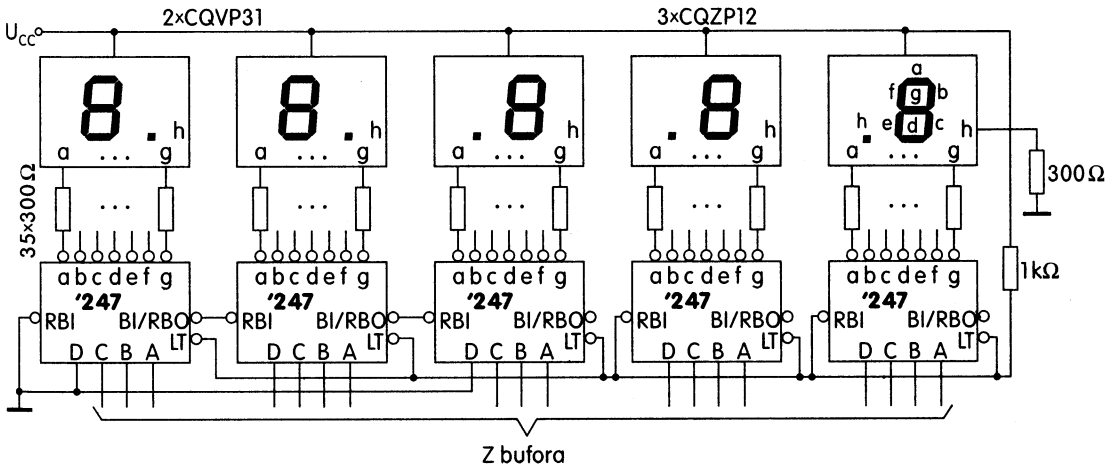
W dekoderach naturalny kod dwójkowy BCD będzie zamieniany na kod wskaźnika 7-segmentowego. Jako dekoderów użyjemy układów **74247**, które od dekoderów **7447** opisanych w p.10.2 różnią się jedynie tym, że nie wymagają korekty cyfr 6 i 9, bowiem są one w postaci już skorygowanej. Poziomem aktywnym na wyjściu tych dekoderów jest poziom niski. Oznacza to, że są one przeznaczone do sterowania wskaźnikami o wspólnej anodzie. Układ dekoderów (5 szt.) połączymy tak, aby wygasić nieznaczące zera, a w przypadku wyzerowania stopera, aby wskazanie jego miało postać „0.0". Ten sposób połączeń zapewni od razu, że zmierzony odcinek czasu krótszy niż 1s będzie wskazywany łącznie z zerem, mimo że jest ono nieznaczące.

Do budowy wyświetlacza użyjemy wskaźnika 7-segmentowego, np. typu CQVP31. Jest on wskaźnikiem o wysokości cyfry 12 mm, w kolorze czerwonym, o wspólnej anodzie. Napięcie przewodzenia segmentu wynosi 2 V, a prąd powinien być mniejszy lub równy 20 mA. W celu ograniczenia prądu każdego segmentu zastosujemy rezystor 300 Ω, co sprawi, że prąd płynący przez segment będzie się zawierać w przedziale 8 ÷ 10 mA. Wskaźnik ten zastosujemy do wyświetlania liczby minut. Natomiast do wyświetlania sekund i dziesiątych części sekund użyjemy identycznego wskaźnika pod względem parametrów elektrycznych i koloru, a jedynie mniejszego — o wysokości cyfr 8 mm (np. CQZP12).

Musimy jeszcze pamiętać o wysterowaniu segmentu *h* (kropki dziesiętnej) we wskaźniku dziesiątych części sekundy. Schemat bloku 7. przedstawiono na rys. 15.5.

- **Blok sterowania (blok 4.)**

Blok sterowania musi zawierać obwody wejściowe, które będą przetwarzać położenie zestyków przycisków **S** i **M** na sygnały o poziomach TTL. Pamiętając



Rys. 15.5. Blok dekodera i wskaźnika

o drganiach zestyku, użyjemy przerzutnika **asynchronicznego** zbudowanego z bramek NAND, aby je odfiltrować. Przerzutników takich użyjemy do współpracy z przyciskiem **S**, a także z przyciskiem **M** (rys. 15.6a, str. 284).

Układ sterowania powinien rozróżniać, czy naciśnięcie przycisku **S** ma oznaczać start, stop czy zerowanie. Ponieważ dopuszczamy jedynie sekwencję start--stop-zerowanie to wystarczy zliczać naciśnięcia przycisku **S**, aby właściwie zinterpretować informację wprowadzaną do układu. Do zliczania (przyciśnięć **S**) zastosujemy licznik **mod 3,** którego zadaniem będzie właśnie zliczanie i pamiętanie liczby naciśnięć przycisku **S** (rys. 15.6b, str. 284).

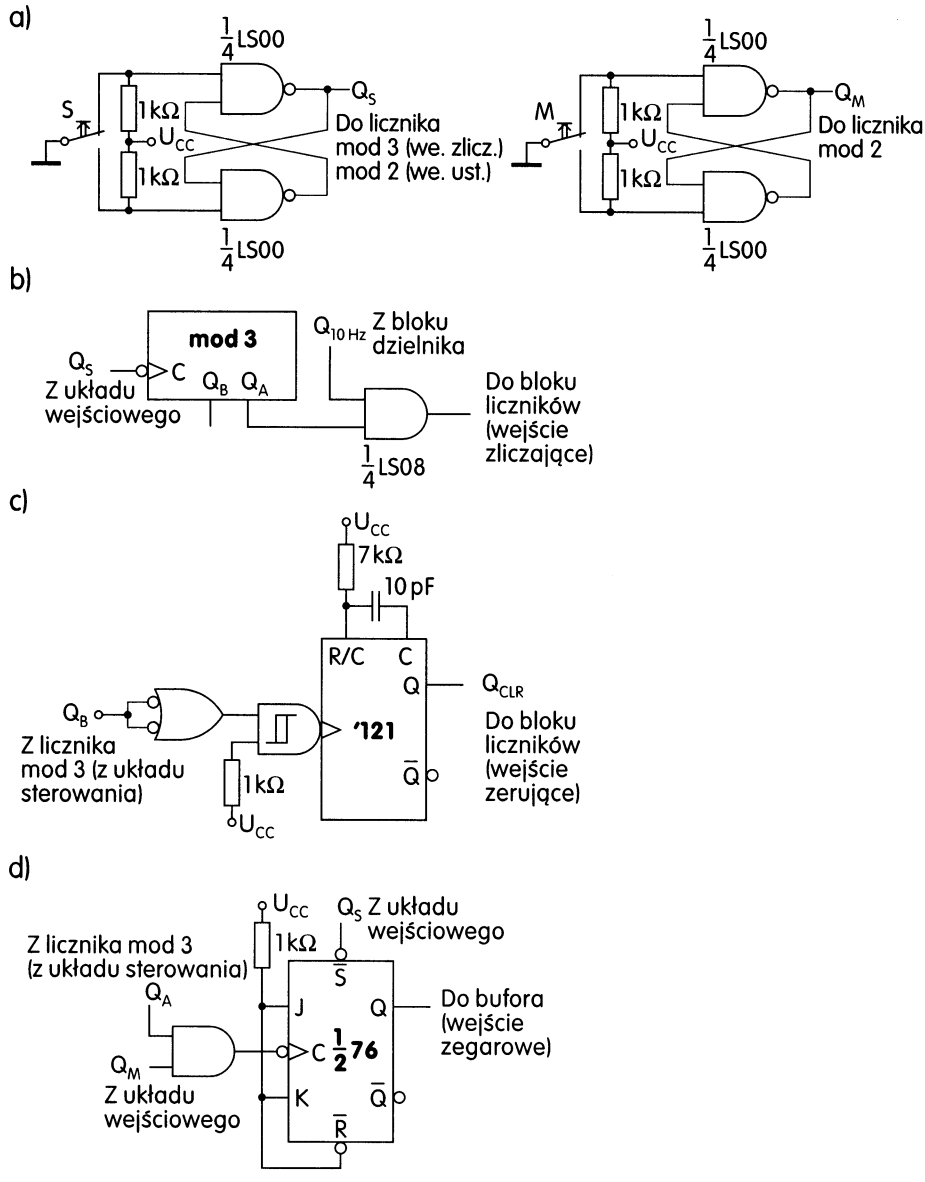
Przyjmijmy kodowanie stanów pracy stopera stanami licznika **mod 3** wg tabl. 15.1.

Tablica 15.1. **Kodowanie stanów pracy stopera stanami licznika mod 3**

Stan licznika **mod 3** Q_B	Q_A	Stan pracy stopera
0	0	stoper wyzerowany (stan początkowy)
0	1	zliczanie czasu
1	0	stop zliczania, wyświetlanie wyniku

Zauważmy, że sygnał z wyjścia $\mathbf{Q_A}$ możemy wykorzystać do bramkowania przebiegu zliczanego: $\mathbf{Q_A = 0}$ przebieg zliczany ($\mathbf{Q_{10\,Hz}}$) zablokowany, $\mathbf{Q_A = 1}$ przebieg zliczany jest podawany na wejście licznika. W tym celu należy $\mathbf{Q_A}$ oraz wyjście bloków dzielników połączyć z wejściami bramki AND (dwuwejściowej). Wyjście bramki AND połączymy z wejściem bloku liczników.

Zakładamy, że bufor jest cały czas otwarty. Sterowaniem bufora zajmiemy się przy realizacji międzyczasu.



Rys. 15.6. Schematy funkcjonalne: a) obwodów wejściowych; b) licznika modulo 3 (układ sterujący podstawowym cyklem pracy stopera); c) generatora impulsu zerującego liczniki; d) licznika modulo 2 zbudowanego z przerzutnika *JK* (układ sterujący buforem)

Trzecie naciśnięcie przycisku **S** powinno spowodować powrót układu do stanu początkowego — wyzerowania liczników z bloku 5. oraz licznika **mod 3** w bloku sterowania. Obojętnie jak zrealizujemy licznik **mod 3** (jego realizacją zajmiemy się nieco później), to trzecie naciśnięcie przycisku **S** będzie powodowało zmianę stanu wyjścia $\mathbf{Q_B}$ tego licznika z **H** na **L** (przejście ze stanu $\mathbf{Q_B Q_A = 10}$ do stanu $\mathbf{Q_B Q_A = 00}$). To ujemne zbocze możemy wykorzystać do pobudzenia przerzutni-

ka monostabilnego (np. **74121**), który wygeneruje odpowiedni impuls zerujący (rys. 15.6c). Impuls ten doprowadzimy do wejść zerujących liczniki w bloku liczników. Z danych katalogowych (dla licznika LS90) wynika, że czas trwania impulsu zerującego nie powinien być krótszy niż 15 ns. Dla zapewnienia skuteczności zerowania przyjmijmy elementy R, C tak, aby czas trwania impulsu generowanego przez przerzutnik monostabilny **'121** wyniósł ok. 50 ns. Z nomogramów zamieszczonych np. w publikacji [21] można odczytać, że należy użyć rezystora $R = 7\ \text{k}\Omega$ oraz kondensatora $C = 10\ \text{pF}$.

Koncepcja układu sterowania opracowana do tej pory zapewnia realizację podstawowego cyklu pracy stopera, tj.: start, stop, zerowanie. Pozostaje więc umożliwić jeszcze realizację funkcji wyświetlania międzyczasu.

Musimy przyjąć (aby rozwiązanie było eleganckie), że taki międzyczas możemy uchwycić wielokrotnie w procesie jednego pomiaru. To znaczy zatrzymać na wyświetlaczu aktualny wynik, wznowić wyświetlanie bieżącego upływu czasu, ponownie zatrzymać itd.. Pierwsze naciśnięcie przycisku **M** powinno zatem „zatrzasnąć" w buforze aktualnie zliczony czas, a kolejne wznowić wyświetlanie bieżącego czasu. Aby to było możliwe układ musi rozróżniać pierwsze wciśnięcie przycisku **M** od drugiego. W tym celu sygnał wyjściowy przerzutnika prostego, współpracującego z zestykiem monostabilnym **M**, podamy na wejście licznika **mod 2** (dwójki liczącej zbudowanej np. z przerzutnika typu *JK* — rys. 15.6d). Zauważmy, że funkcję „międzyczas" stoper powinien realizować tylko wtedy, kiedy znajduje się on w stanie zliczania, czyli $\mathbf{Q_A = 1}$. Przyciśnięcia przycisku **M** (zliczane w liczniku **mod 2**) możemy zatem zablokować sygnałem $\mathbf{Q_A}$, wykorzystując do tego celu bramkę AND. Dzięki temu uzyskamy od razu spełnienie warunku, aby wejście **M** było nieaktywne w stanie wyświetlania wyniku oraz w stanie wyzerowania stopera.

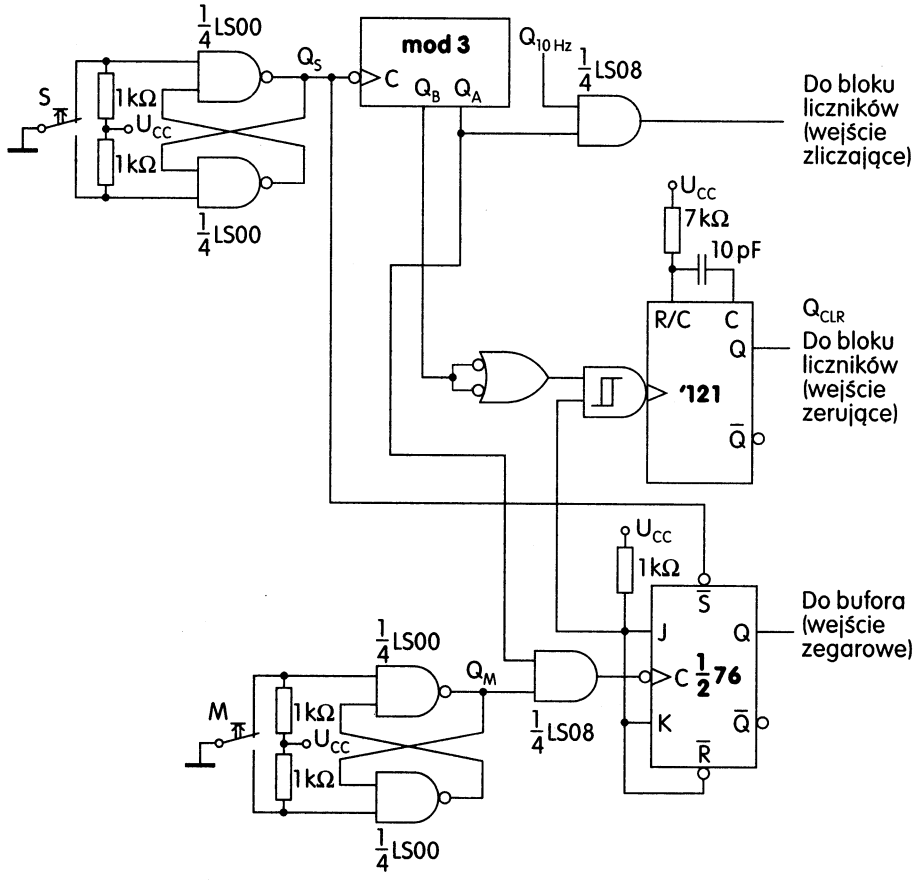
Bufor (zbudowany z przerzutników typu „latch" (zatrzask)) przenosi stany wejść na wyjście przy $\mathbf{C = 1}$ oraz zatrzaskuje informację obecną w chwili ujemnego zbocza sygnału wejściowego **C**. Aby sterować buforem bezpośrednio z wyjścia **Q** licznika **mod 2** należy zapewnić, by jego podstawowym stanem był stan wysoki **H**. Naciśnięcie przycisku **M** (podczas zliczania, czyli kiedy $\mathbf{Q_A = 1}$) spowoduje zmianę stanu licznika z **H** na **L**. Zmiana ta zamknie bufor. Ponowne naciśnięcie przycisku **M** otworzy bufor, a kolejne zamknie. Mamy w ten sposób zapewnioną możliwość pomiaru kilku międzyczasów podczas jednego zliczania. (Można naturalnie przyjąć za stan podstawowy licznika **mod 2** stan **L** i wówczas do sterowania bufora użyć sygnału zanegowanego $\overline{\mathbf{Q}}$.)

Pozostaje jeszcze rozważyć przypadek, gdy stoper jest w stanie wskazywania międzyczasu i wówczas następuje zatrzymanie zliczania czasu (przyciśnięcie przycisku **S** oznaczające funkcję stop). Aby stoper wskazał odmierzony czas, powinno nastąpić otwarcie bufora — wówczas aktualny stan liczników zostanie zdekodowany i wyświetlony na wskaźniku. W niniejszym układzie cel taki zostanie osiągnięty, jeżeli przyciśnięcie przycisku **S** spowoduje ustawienie licznika **mod 2** w stan **H**. Jeżeli do budowy dwójki liczącej użyjemy przerzutnika *JK* z układu np. **74LS76**, to możemy wykorzystać jego wejście asynchroniczne, ustawiające przerzutnik w stan wysoki **H** (rys. 15.6d). Poziomem aktywnym tego wejścia jest stan niski **L**.

Stan taki uzyskujemy także na wyjściu Q_S układu współpracującego z zestykiem przycisku **S** w chwili jego naciśnięcia. Sygnał wyjściowy z tego układu (jest to ten sam sygnał, który jest zliczany w liczniku **mod 3**) doprowadzimy zatem do wejścia ustawiającego w stan **H** licznik **mod 2.** Rozważmy jedynie, czy połączenie takie (czyli każdorazowe ustawianie licznika **mod 2** w stan wysoki) nie spowoduje niewłaściwego działania układu w pozostałych stanach pracy stopera. Z analizy pozostałych stanów wynika, że nie zaburzy to cyklu pracy stopera.

Schemat zasadniczy układu sterowania przedstawiono na rys. 15.7. W celu weryfikacji poprawności jego działania przeanalizujmy ten schemat. Zakładamy, że liczniki są wyzerowane oraz (na początek), że realizujemy jedynie cykl podstawowy (bez pomiaru międzyczasu).

Stan początkowy układu sterowania to: stan $Q_BQ_A = 00$ licznika **mod 3** oraz stan $Q = 1$ licznika **mod 2.** Bramka przebiegu zliczanego jest zamknięta (bo $Q_A = 0$) i do bloku liczników nie są podawane impulsy przebiegu wzorcowego. Bufor jest otwarty (sygnał sterujący buforem ma stan $Q = 1$). Ponieważ liczniki są wyzerowane, przeto wskaźnik stopera pokazuje czas „0.0". Naciśnięcie przycisku **M** nie zmienia stanu układu, co jest zgodne z założeniami.



Rys. 15.7. Schemat układu sterowania

Naciśnięcie przycisku **S** (pierwsze) powoduje zmianę stanu Q_S z **H** na **L**. Zbocze to zostaje zliczone w liczniku **mod 3**, co sprawia, że Q_A przechodzi w stan wysoki ($Q_A = 1$). Otwiera to bramkę przebiegu zliczanego $Q_{10\,Hz}$ i liczniki (w bloku liczników) rozpoczynają zliczanie. Bufor jest otwarty i obserwujemy na wskaźniku bieżący upływ czasu, liczony od chwili wciśnięcia przycisku **S**.

Kolejne naciśnięcie przycisku **S** sprawia, że Q_A przechodzi w stan niski (jest zliczona kolejna zmiana Q_S), co powoduje zamknięcie bramki przebiegu zliczanego. Bufor podczas realizacji podstawowego cyklu pracy stopera jest cały czas otwarty. Wskaźnik pokazuje odmierzony czas. Zauważmy, że przy realizacji podstawowego cyklu pracy bufor jest w zasadzie zbędny.

Przycisk **M** w stanie wyzerowania stopera oraz w stanie wskazywania wyniku jest nieaktywny ($Q_A = 0$ i zmiany położenia zestyku **M** są zablokowane przez bramkę iloczynową). Informacja wprowadzana przyciskiem **M** może mieć wpływ na pracę stopera jedynie przy $Q_A = 1$, czyli w stanie zliczania. Wówczas zmiana położenia (pierwsza) zestyku przycisku **M** powoduje zmianę stanu Q_M z **H** na **L**, i licznik **mod 2,** który zlicza te zbocza sygnału Q_M, przechodzi w stan **L**. Ujemne zbocze sygnału **Q** zamyka bufor. Kolejne naciśnięcie przycisku **M** otwiera bufor (licznik **mod 2** jest ustawiany w stan **H**). Także naciśnięcie przycisku **S** otwiera bufor poprzez ustawienie licznika **mod 2** w stan **H**.

Analiza pracy układu sterowania na podstawie schematu prowadzi do wniosku, że wszystkie funkcje stopera są realizowane poprawnie. Ostateczną weryfikacją działania układu będzie testowanie gotowego stopera.

Pozostał do opracowania schemat logiczny licznika **mod 3.** Możemy rozważyć dwa warianty budowy takiego licznika: asynchroniczny i synchroniczny. Zanim podejmiemy decyzję, rozpatrzmy wszystkie za i przeciw obu wersji.

Wersja I — **licznik asynchroniczny**

Zbudować go możemy z dwóch liczników **mod 2** połączonych szeregowo. Układ taki to licznik **mod 4.** Aby skrócić jego cykl zliczania należy wykonać zerujące sprzężenie zwrotne sygnałem $Q_A \cdot Q_B$ co wymaga użycia dodatkowej bramki AND (jeżeli poziomem aktywnym wejścia zerującego jest poziom wysoki **H**). Sygnał wyjściowy z bramki AND można wykorzystać także do zerowania wszystkich liczników w bloku liczników.

Zalety:

- Mała liczba układów potrzebnych do jego realizacji. Jeżeli wykorzystamy dwie dwójki liczące z układów **74LS92** (z których do tej pory użyto jedynie liczniki **mod 6** w bloku liczników) i sygnały Q_A, Q_B tak zbudowanego licznika **mod 4** doprowadzimy do wejść **R0**(1) i **R0**(2) ***wszystkich*** liczników w bloku liczników, to taka realizacja tego fragmentu układu sterowania nie wymaga użycia dodatkowych elementów. Dodatkowo możemy jeszcze zrezygnować z układu **74121** jako generatora impulsu zerującego.

Wady:

- Mała pewność (niezawodność) działania. Impuls zerujący jest bardzo krótki (określony przez czas propagacji zerowanych dwójek liczących w liczniku

mod 3). Może się zdarzać, że jedna dwójka zostanie wyzerowana szybciej i impuls zerujący zaniknie bez wyzerowania drugiej dwójki, a nawet pozostałych liczników w bloku liczników.

- Mała szybkość działania licznika. Ta wada jest nieistotna przy budowie stopera, ale jeżeli zechcemy z tego układu zbudować (np.) czasomierz do pomiaru czasów trwania impulsów (modyfikację taką rozważymy nieco później), to szybkość działania tego licznika będzie szczególnie istotna.
- Układ przestaje być modułowy. Przy budowie modułowej możemy łatwo zmieniać pojedyncze moduły, uzyskując różne układy. Oczywiście do budowy licznika asynchronicznego możemy użyć odrębnych układów, aby zachować budowę modułową, ale tracimy wówczas jedyną zaletę tego rozwiązania, jaką jest mała liczba elementów.

Wersja II — **licznik synchroniczny**

Zbudować go możemy z dwóch przerzutników synchronicznych (np. *JK*), wykonując odpowiedni projekt. Nie uda się w tym przypadku (prawdopodobnie) wykorzystać przerzutników (dwójek liczących) z układów **74LS92**, bowiem musimy dysponować dostępem do wejść **J**, **K** tych przerzutników. Zacznijmy tym razem od wad tego rozwiązania.

Wady:

- Konieczność użycia dodatkowych układów. Do budowy licznika są potrzebne: np. układ **74LS76** oraz — jako źródło sygnału zerującego liczniki w bloku liczników — układ **74121**. Zerowanie samego licznika **mod 3** jest zbędne, bowiem licznik taki (z definicji) wraca do stanu **00** po trzech impulsach.

Zalety:

- Pewność działania — właściwe ukształtowanie impulsu zerującego liczniki gwarantuje ich poprawną pracę.
- Szybkość działania — mniejsze opóźnienie przy otwieraniu i zamykaniu bramki przebiegu zliczanego.
- Modułowa budowa układu stopera.

Przewaga zalet wariantu II skłania więc do jego wyboru. Musimy zatem zaprojektować licznik synchroniczny liczący w trybie **mod 3.** Licznik taki będzie mieć trzy stany **S**, które ponumerujemy kolejno 0, 1 i 2. Działanie licznika opisuje tablica przejść w postaci jak na rys. 15.8a. Stany licznika zakodujmy w kodzie dwójkowym naturalnym (rys. 15.8b). Uwzględniając to kodowanie, możemy zapisać zakodowaną tablicę przejść naszego licznika (rys. 15.8c). Na jej podstawie oraz tablicy wzbudzeń przerzutnika *JK* — rys. 15.9a (patrz także rys. 7.12d) określimy tzw. **funkcje wzbudzeń** obu przerzutników użytych do budowy licznika (rys. 15.9b i rys. 15.9c). Wyznaczenie tych funkcji na podstawie tablic Karnaugha umożliwia narysowanie schematu projektowanego licznika — rys. 15.10.

a)

S	S^+
0	1
1	2
2	0

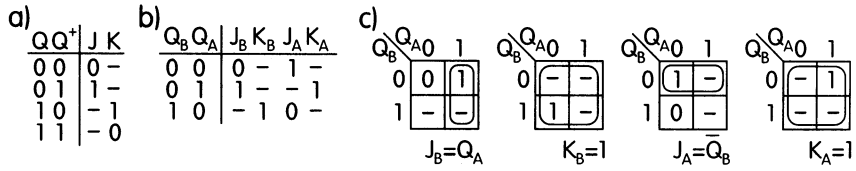
b)

S	$Q_B Q_A$
0	0 0
1	0 1
2	1 0

c)

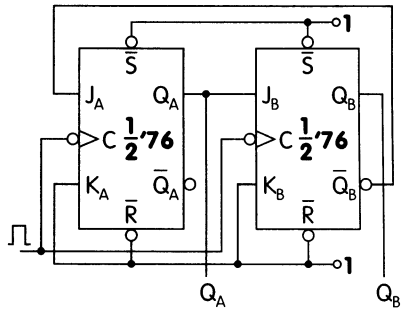
$Q_B Q_A$	$Q_B^+ Q_A^+$
0 0	0 1
0 1	1 0
1 0	0 0

Rys. 15.8. Tablica przejść licznika modulo 3 (a); kodowanie stanów licznika (b); zakodowana tablica przejść licznika modulo 3 (c)



Rys. 15.9. Tablica wzbudzeń przerzutnika *JK* (a); funkcje wzbudzeń (b); tablice Karnaugha dla funkcji wzbudzeń (c)

Układy cyfrowe zawierające przerzutniki wymagają zwykle ustawienia ich w określony stan w chwili załączenia napięcia zasilającego. W literaturze technicznej można znaleźć układy generatorów impulsów, które występują jedynie w chwili załączenia układu do napięcia zasilającego (czy samego zasilacza). Impulsy te wykorzystuje się do zerowania bądź ustawiania określonych przerzutników. Musimy także rozważyć potrzebę (lub jej brak) użycia takich układów.



Rys. 15.10. Schemat synchronicznego licznika modulo 3 w układzie sterowania

Na początek zauważmy, że wystarczy zapewnić, aby niezależnie od stanu, w jakim ustawią się przerzutniki (w układzie sterowania, w bloku liczników), można było sprowadzić nasz układ do stanu początkowego za pomocą dostępnych przycisków (**S** i **M**). Stan początkowy liczników w bloku dzielników nie ma wpływu na pracę stopera. O ile więc podstawowy licznik w układzie sterowania (czyli licznik **mod 3**) ustawi się w jeden z jego stanów pracy, to poprzez jedno, dwa lub trzy kolejne naciśnięcia przycisku **S** możemy doprowadzić stoper do stanu początkowego. Przypomnijmy, że stan początkowy stopera to: — w bloku liczników liczniki wyzerowane, a w układzie sterowania — licznik **mod 3** w stanie **00** i licznik **mod 2** w stanie **1**.

Jedynym zagrożeniem jest ustawienie się obu przerzutników w liczniku **mod 3** w stan **1**, czyli stan tego licznika **11**. Przeanalizujmy, na podstawie schematu tego licznika (rys. 15.10), jego zachowanie się w takiej sytuacji. Ze schematu wynika, że $\mathbf{J_A K_A = 01}$ i $\mathbf{J_B K_B = 11}$. Z zasady działania przerzutnika *JK* wiemy, że najbliższy impuls zegarowy (naciśnięcie przycisku **S**) spowoduje przejście obu przerzutników w stan **L**. Zmiana $\mathbf{Q_B}$ z **H** na **L** wzbudzi przerzutnik monostabilny **74121**, który wygeneruje impuls zerujący licznik. Układ stopera znajdzie się w stanie początkowym.

Nie ma więc potrzeby budowy dodatkowego układu ustawiającego stoper w stan początkowy. Jedyną wadą pozostawienia takiego rozwiązania jest brak jednoznacznej sygnalizacji stanu pracy stopera. Może bowiem stoper wskazywać 0.0, co sugerowałoby jego stan początkowy, a jednocześnie licznik **mod 3** może być w stanie **11**. Wówczas naciśnięcie przycisku **S** (start) nie spowoduje zliczania. Należy zatem w instrukcji obsługi takiego stopera zamieścić wskazówkę, aby przed każdym użyciem stopera (po włączeniu zasilania) wykonać kilka kolejnych naciśnięć przycisku **S** w celu jego sprawdzenia i ustawienia w stan początkowy.

Można także łatwo wykonać sygnalizację stanu pracy stopera. Na przykład sygnałem $\mathbf{Q_B+Q_A+\overline{Q}}$ (gdzie $\mathbf{Q_B}$, $\mathbf{Q_A}$ to sygnały wyjściowe licznika **mod 3,** a **Q** to sygnał wyjściowy licznika **mod 2**) sterujmy świeceniem kropki we wskaźniku minut (dokładniej dziesiątek minut). Kropka ta będzie świecić (powyższa suma będzie równa **0**) tylko wtedy, gdy $\mathbf{Q_BQ_A = 00}$ i $\mathbf{Q = 1}$. Stoper jest wtedy w stanie początkowym, o ile liczniki są wyzerowane. Jeśli liczniki nie są wyzerowane, to zobaczymy to na wskaźniku, ponieważ bufor jest otwarty ($\mathbf{Q = 1}$).

- **Zasilacz (blok 1.)**

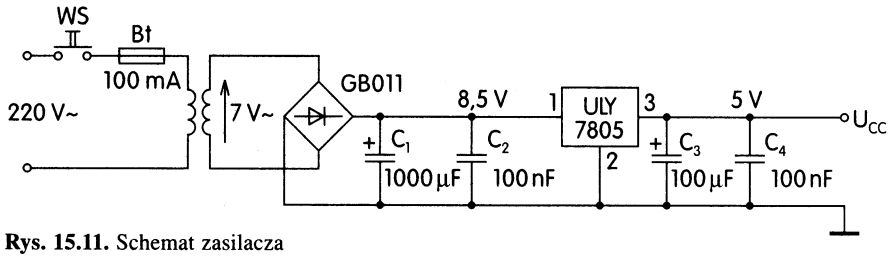
Znając budowę poszczególnych bloków stopera oraz elementy z jakich chcemy je zbudować, możemy oszacować ich pobór prądu (na podstawie danych katalogowych) i dzięki temu określić wymaganą wydajność prądową zasilacza (tabl. 15.2). Ułatwi to nam dobór odpowiedniego stabilizatora oraz transformatora sieciowego.

Tablica 15.2. **Pobór prądu przez stoper**

Układ	Element		Pobór prądu, mA	
	rodzaj	liczba		
Generator	74LS04	1	3	
Dzielniki	74LS90	5	45	
Blok sterowania	74LS00	1	2	
	74LS08	1	2	
	74LS76	2	8	
	74121	1	40	
	R = 1 kΩ	4	20	Σ = 625
Liczniki	74LS90	3	27	
	74LS92	2	18	
Bufor	74LS75	5	50	
Dekodery	74LS247	5	50	
Wskaźnik	LED	36	360	

Ponieważ do obliczeń były brane wartości znamionowe oraz należy zapewnić pewien zapas mocy, przeto przyjmiemy, że zasilacz powinien dostarczyć prądu o wartości minimum 0,8A. Można więc jako stabilizatora użyć typowego układu, np. **ULY7805**, którego napięciem wyjściowym jest U_O = 5V oraz, który ma wewnętrzne zabezpieczenie ograniczające prąd zwarciowy. Schemat zasadniczy zasilacza pokazano na rys. 15.11. Prostownikiem może być scalony mostek Graetza, np. typu **GB011** (lub podobny). Napięcie jest prostowane w mostku, a następnie filtrowane przez filtr pojemnościowy, zbudowany z kondensatorów C_1 i C_2. Kondensator elektrolityczny C_1 o pojemności ok. 1000μF ma za zadanie tłumić tętnienia napięcia wyprostowanego. Rolą kondensatora C_2 o pojemności ok. 100nF jest filtrowanie zakłóceń o częstotliwości dużo większej niż sieciowa, tzw. wyż-

szych harmonicznych. Na wyjściu stabilizatora znajduje się filtr, zbudowany z kondensatorów C_3 i C_4, którego zadaniem jest tłumienie zakłóceń pochodzących od zasilanego układu. Kondensator C_4 o pojemności ok. 100nF jest kondensatorem odsprzęgającym. Pamiętajmy, że układy cyfrowe (TTL, CMOS) wymagają stosowania kondensatorów odsprzęgających (blokujących). Należy przewidzieć już na etapie projektowania blokowanie oddzielnym kondensatorem każdego układu scalonego MSI (liczniki, dekodery, bufory) i dodatkowym każde 5 układów SSI.



Rys. 15.11. Schemat zasilacza

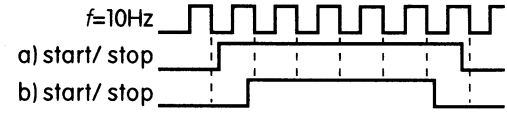
Aby określić napięcie uzwojenia wtórnego transformatora sieciowego należy pamiętać, że poprawna praca stabilizatora jest możliwa przy napięciu wejściowym co najmniej 8V. Jednocześnie spadek napięcia na mostku Graetza wynosi ok. 1,4 V (zawsze przewodzą dwie diody w mostku). Tak więc napięcie wtórne transformatora sieciowego powinno wynosić ok. 7 V. Pamiętajmy, że jest to wartość skuteczna. Po wyprostowaniu i odfiltrowaniu otrzymamy napięcie $7 \cdot \sqrt{2} \approx 9{,}9$ V. Uwzględniając spadek napięcia na diodach będzie $9{,}9 - 1{,}4 = 8{,}5$ V. Zatem moc pobierana z uzwojenia wtórnego transformatora wyniesie ok. $9{,}9\ \text{V} \cdot 1\ \text{A} \approx 10$ W (do obliczeń przyjęto prąd 1 A, aby w pełni wykorzystać możliwości stabilizatora). Uwzględniając straty mocy w transformatorze oraz pobór prądu magnesującego, należy określić moc transformatora sieciowego na poziomie ok. 15 V·A. Prąd pobierany z sieci będzie więc miał wartość $I = 15\ \text{V} \cdot \text{A}/220\ \text{V} = 0{,}068$ A. Wystarczy zatem bezpiecznik topikowy o prądzie znamionowym 100 mA.

Kompletny schemat całego układu stopera przedstawiono w Dodatku 2. zamieszczonym na końcu podręcznika.

Przedyskutujmy jeszcze niektóre aspekty przyjętego rozwiązania.

Dokładność pomiaru

Ten problem był już sygnalizowany przy omawianiu budowy generatora przebiegu wzorcowego. Tym razem jednak przeanalizujmy źródło błędów zlokalizowane w układzie bramkowania generatora. Przyjrzyjmy się rys. 15.12, na którym pokazano przebiegi czasowe, uzyskane w tym układzie przy pomiarze różnych przedziałów czasowych, które jednak spowodują jednakowe wskazanie naszego stopera.



Rys. 15.12. Bramkowanie przebiegu wzorcowego ($f = 10$ Hz) w układzie stopera sygnałem start/stop (Q_A)

W obu przypadkach układ stopera zliczy 5 impulsów. Zmierzony i wskazany czas wyniesie 0,5 s. W przypadku pierwszym (rys. 15.12a) rzeczywisty czas może mieć wartość prawie 0,6s. W przypadku drugim (rys. 15.12b) rzeczywisty czas może mieć wartość nawet ok. 0,4 s. Powiemy, że błąd pomiaru czasu wyniesie więc ±0,1 s, co oznacza, że wskazany wynik może być za mały lub za duży właśnie o wartość 0,1 s. W bardzo prosty sposób, bez dodatkowej rozbudowy układu stopera, możemy zmniejszyć wartość tego błędu. Przesuńmy jeden z liczników z układu dzielników do bloku liczników. Wówczas zliczany przebieg będzie miał częstotliwość równą 100Hz. Dodatkowy licznik (**mod 10**) w bloku liczników będzie zliczał zatem setne części sekundy. Bramkowany przebieg będzie miał okres równy 0,01 s. Błąd wynikający z niewłaściwej chwili otwarcia/zamknięcia bramki wyniesie tym razem ±0,01 s, czyli będzie o rząd mniejszy. Należy dodać jeszcze dekoder i wskaźnik 7-segmentowy zliczanych setnych części sekundy, o ile jest wymagana większa dokładność odczytu.

Pobór mocy

W założeniach przyjęto zasilanie układu stopera z sieci. Chcąc jednak dysponować przyrządem przenośnym, należałoby istotnie ograniczyć pobór mocy. Popatrzmy jeszcze raz na wykaz elementów i wartości prądu pobieranego przez te elementy — tabl. 15.2. Największy pobór mocy wykazuje wskaźnik zbudowany z diod typu LED. Zastąpienie go wskaźnikiem ciekłokrystalicznym zredukuje pobór mocy o połowę. Dalsze zmniejszenie poboru mocy osiągniemy, zamieniając układy TTL na układy CMOS.

Jednak najmniejszy pobór mocy uzyskamy, biorąc do budowy naszego stopera specjalizowany układ scalony LSI (dużego stopnia scalenia), realizujący wszystkie funkcje projektowanego stopera. Wiele elektronicznych zegarków jest wyposażonych w funkcje, jakie realizuje nasz stoper i do budowy stopera może być użyty układ podobnego typu, co w takim zegarku.

- **Czasomierz**

Zaprojektowany układ można wykorzystać do pomiaru czasu trwania impulsów elektrycznych. Wówczas sygnałem bramkującym, zamiast sygnału Q_A z bloku sterowania, będzie po prostu mierzony impuls, o ile poziomy jego są już w standardzie TTL. W przeciwnym przypadku trzeba będzie zbudować dodatkowe obwody wejściowe kształtujące ten impuls.

Rozbudowując obwody wejściowe czasomierza, można uzyskać pomiar odcinka czasu, którego początek jest wyznaczony dowolnym zboczem jednego sygnału wejściowego, a koniec dowolnym zboczem drugiego sygnału wejściowego. Tak zmodyfikowany układ byłby przydatny np. do pomiaru czasu propagacji. Jednak w takim przypadku rozdzielczość czasomierza powinna być dwa, trzy rzędy mniejsza niż mierzone czasy. Chcąc mierzyć odcinki czasu o długości nanosekund, należałoby zliczać przebieg wzorcowy o częstotliwości co najmniej rzędu kilku gigaherców. Istotny wpływ na dokładność pomiaru miałyby też opóźnienia wnoszone przez elementy czasomierza. W technice TTL czy CMOS wykonanie takiego czasomierza nie jest obecnie możliwe. Zresztą nigdy nie będzie możliwe

zbudowanie układu do pomiaru czasów propagacji z elementów wykonanych w tej samej technice co badane. Aby mierzyć parametry dynamiczne pewnych elementów, trzeba dysponować elementami 100, a nawet 1000 razy szybszymi od badanych.

15.3. Układ do badania pamięci EEPROM (generator znaków)

Program nauczania (nr 210502) „Pracowni układów i urządzeń elektronicznych" w IV klasie 5-letniego Technikum Elektronicznego, w zawodzie *technik elektronik* o specjalności elektronika ogólna, przewiduje badanie półprzewodnikowych pamięci typu ROM i RAM. Zakres badań obejmuje: odczyt/zapis informacji z/do pamięci, pomiar czasu dostępu, pomiar poboru prądu w różnych trybach pracy pamięci, badanie pamięci jako generatora znaków.

W Dodatku 2. przedstawiono schemat (rys. D.2.3) układu umożliwiającego wykonanie powyższych badań pamięci EEPROM 2817. Jest to pamięć o pojemności 16 Kb (2 KB), organizacji 2048·8 (2048 słów 8-bitowych). Zastosowanie pamięci EEPROM pozwala na uatrakcyjnienie punktu ćwiczenia: badanie pamięci jako generatora znaków. Dzięki temu, że jest ona programowana i kasowana elektrycznie, uczeń może w trakcie ćwiczenia zdefiniować własne znaki i odpowiednio zaprogramować pamięć. Może to być np. ciąg znaków składający się na nazwisko ucznia. Układ umożliwia bowiem wyświetlenie, przy użyciu matrycy diodowej, sekwencji znaków w takt naciskania jednego z przycisków. Można oczywiście układ rozbudować w taki sposób, aby określony ciąg znaków pojawiał się samoczynnie. Przedstawiony układ należy bowiem traktować jako pewną propozycję budowy stanowiska do badania pamięci. Zastosowane rozwiązania pewnych fragmetów układu są kontrowersyjne, ale pozostawiono je w wersji, jaka została wykonana przez uczniów klasy maturalnej jako praca dyplomowa. Była to praca prototypowa i dlatego nie trudno będzie wskazać jej mankamenty i zaproponować lepsze rozwiązania.

Na podstawie zamieszczonego schematu (patrz Dodatek 2. — rys. D.2.3) przeprowadzimy analizę pracy układu. Naszym podstawowym celem będzie określenie sposobu realizacji wszystkich badań, jakie w tym układzie można przeprowadzić. Na rysunku D.2.2 przedstawiono widok płyty czołowej stanowiska do badania pamięci EEPROM i generatora znaków, a na rys. D.2.3 schemat połączeń tego układu.

Zanim jednak przejdziemy do omówienia realizacji badań w prezentowanym układzie, określmy, w jaki sposób można ustawiać wejścia adresowe, danych oraz sterujące badanej pamięci US5 (US — Układ Scalony).

Wejścia adresowe pamięci są sterowane z wyjść liczników binarnych (**mod 16**): układy US7A, US7B, US6B. Są to układy MCY74520 (wykaz elementów na końcu niniejszego rozdziału). Każdy z obu układów scalonych US7, US8 zawiera dwa liczniki synchroniczne (**mod 16**) zliczające w górę. Symbole USA, USB oznacza-

ją części (funkcjonalnie identyczne) tego samego układu scalonego. Zamiast powyższych oznaczeń używa się także opisu w postaci 1/2US na oznaczenie części układu scalonego. W przypadku na przykład układu scalonego (US) zawierającego dwuwejściowe bramki NAND opis każdej bramki wyglądałby: USA (B, C lub D) lub 1/4US. Szczegółową identyfikację konkretnej części układu scalonego ułatwiają numery wyprowadzeń, które powinny być na schematach zaznaczane. Inny sposób zaznaczania na schematach scalonego układu wieloczęściowego to obrysowanie linią przerywaną wszystkich części tego układu. Jednak ten sposób jest możliwy do zastosowania tylko wówczas, gdy części te są (na schemacie) usytuowane blisko siebie.

Z wyjść liczników US7, US6B są sterowane także wejścia negatorów (układy US15, US16). Z wyjść tych negatorów są sterowane diody świecące typu LED ($D5 \div D15$) służące do wskazywania stanów wejść adresowych. Dioda świeci, gdy na wyjściu sterującego nią negatora jest stan niski. Odpowiada to wysokiemu stanowi na wejściu negatora. Tak więc przyjęto tutaj (najczęściej stosowaną) konwencję, że dioda świecąca sygnalizuje nam stan wysoki, dioda zgaszona — stan niski. Taki sam sposób sterowania zastosowano również do diod sygnalizujących stany wejść sterujących pamięcią (diody $D16, D17, D18$) oraz stany wejść adresowych układu US4 (diody $D19$, $D20$, $D21$).

Zauważmy, że sygnał $\mathbf{Q}_A$ z licznika US7A dochodzi tylko do zacisku laboratoryjnego G_1. Aby sterować nim wejście adresowe $\mathbf{A}_0$ należy połączyć zaciski G_1 i G_2. I takie połączenie należy wykonać przy badaniu pamięci. Jednak przy pomiarze czasu dostępu należy to połączenie usunąć, a wykonać połączenie G_2-G_3. Wówczas do gniazda typu BNC (B_1) doprowadzamy sygnał TTL z generatora i jest on tym samym doprowadzony do wejścia adresowego $\mathbf{A}_0$. Drugie gniazdo BNC (B_2) pozwala sygnał ten podać na wejście oscyloskopu. Wcześniej należy pamięć tak zaprogramować, aby dwie sąsiednie komórki pamięci (różniące się bitem $\mathbf{A}_0$ adresu) miały różne wartości bitu $\boldsymbol{D}_0$ słowa danych. Podając sygnał $\boldsymbol{D}_0$ (gniazdo B_3) na drugi kanał oscyloskopu, dokonamy pomiaru czasu dostępu poprzez pomiar przesunięcia pomiędzy przebiegiem wejściowym, a wyjściowym. Oczywiście, wejścia sterujące pamięcią powinny być ustawione tak, aby pamięć pracowała w trybie „odczyt".

Aby zakończyć identyfikację roli układów US7 i US6B, należy jeszcze określić wpływ diod D_3 i D_4 na pracę licznika US6B. Zauważmy, że diody te (wraz z rezystorem R_{41}) tworzą bramkę iloczynową (AND), której wyjście jest doprowadzone do wejścia **CLR** zerującego licznik (stanem wysokim). Zerowanie takie wystąpi w chwili, gdy licznik zliczy 5 impulsów (licznik osiągnie stan $\mathbf{Q_C Q_B Q_A = 101}$) i natychmiast (z opóźnieniem wynikającym z czasów propagacji) zostanie wyzerowany. Będzie więc liczył w trybie **mod 5.** Jest to pewne ograniczenie przestrzeni adresowej dostępnej dla użytkownika, ale uzasadnienie celowości takiego rozwiązania podamy dalej. Dokończmy najpierw analizę układu umożliwiającego adresowanie pamięci. Aby zaadresować te wejścia, należy ustawić liczniki US7, US6B w pożądany stan. Wejścia (zliczające) tych liczników są sterowane z wyjść układu US4. Układ ten jest ośmiobitowym zatrzaskiem adreso-

walnym. Zawiera on dekoder (3 linie na 8) oraz osiem przerzutników *D* typu zatrzask. W takim układzie połączeń jak na analizowanym schemacie (**WD = 0**, **R = 0**) przerzutnik adresowany powtarza stan wejścia **D**, a pozostałe przerzutniki pamiętają ostatni swój stan. Wejście **R** (ang. *Reset*) jest wejściem zerującym przerzutniki, a wejście **WD** (ang. *Write Disable*) jest wejściem zakazującym zapisu (lub zezwalającym na zapis dla **WD = 0**). Pełny opis działania podobnego układu można znaleźć np. w publikacji [3]. Użycie tego układu powoduje, że w danej chwili możemy zmieniać stan jednego z liczników — licznika określonego stanem wejść adresowych układu US4. Wejścia te są sterowane z licznika US6A połączonego identycznie jak licznik US6B, a zatem pracującego także w trybie **mod 5.** Jest to wystarczające, ponieważ wykorzystujemy tylko 5 wyjść 8-bitowego adresowalnego zatrzasku US4. Stan wejść adresowych (stan licznika US6A) możemy zmieniać, podając na jego wejście impulsy. Źródłem tych impulsów jest przerzutnik monostabilny (US1) wzbudzany za pomocą przycisku „WYBÓR". Stan wejść adresowych adresowalnego zatrzasku (US4) wskazują diody *D*19, *D*20, *D*21 i na tej podstawie możemy określić aktywne (wybrane) wyjście. Z pierwszych trzech wyjść układu US4 ($\mathbf{Q}_0$, $\mathbf{Q}_1$, $\mathbf{Q}_2$) sterujemy licznikami adresującymi pamięć.

Adresowanie pamięci wygląda więc następująco:

1. Naciskając przycisk „WYBÓR" dokonujemy uaktywnienia odpowiedniego wyjścia układu US4, a tym samym wyboru licznika ustawiającego określone wejścia adresowe badanej pamięci. Niech będzie to np. wyjście $\mathbf{Q}_2$ układu US4 (wybrany licznik US6B). Stan wejść adresowych układu US4 (wskazywany przez diody LED) powinien być nastujący: $\mathbf{A_2A_1A_0 = 010}$.
2. Naciskając przycisk „+1" wzbudzamy przerzutnik monostabilny US2. Sygnał wyjściowy tego przerzutnika podawany na wejście adresowalnego zatrzasku US4 jest przekazywany na jego wyjście $\mathbf{Q}_2$ i zliczany przez licznik **mod 5** (US6B), co umożliwia ustawienie trzech najstarszych bitów wejść adresowych pamięci w zakresie od $\mathbf{A_{10}A_9A_8 = 000}$ do $\mathbf{A_{10}A_9A_8 = 100}$.

W podobny sposób możemy ustawić dowolny stan pozostałych wejść adresowych badanej pamięci.

Uaktywniając wyjście $\mathbf{Q}_3$ lub $\mathbf{Q}_4$ układu US4 spowodujemy, że naciśnięcie przycisku „+1" będzie zliczane przez licznik US8B lub US8A. Możliwe więc będzie ustawienie dowolnego słowa zapisywanego do pamięci. Wyjścia liczników US8 nie mogą być jednak bezpośrednio dołączone do wyjść/wejść danych pamięci US5. Elementami pośredniczącymi są bufory US9 o wyjściach trójstanowych. Bufory te powinny być aktywne (przenosić stan wejść na wyjścia) tylko wówczas, gdy pamięć znajduje się w trybie „zapis" lub jest wyłączona (wyjście/wejście danych w stanie wielkiej impedancji). Stan wyjść danych (przy odczycie) lub wejść danych (przy zapisie) jest wskazywany przez jedną z kolumn matrycy diodowej.

Stany wejść sterujących **W/$\overline{\mathbf{R}}$**, **$\overline{\mathbf{OE}}$**, **$\overline{\mathbf{CE}}$** są ustawiane za pomocą, oznaczonych podobnie jak wejścia, przełączników **W/R**, **OE**, **CE**. Stany tych wejść są sygnalizowane przez diody LED (*D*16, *D*17, *D*18) oraz dodatkowo jest sygnalizowany (diodą *D*22) stan we/wy danych; dioda *D*22 świeci — we/wy danych są

w stanie wielkiej impedancji. Pamięć (US5) znajduje się w trybie pracy „odczyt", gdy wszystkie wejścia sterujące są w stanie niskim, co można poznać po odpowiedniej symbolice opisu tych wejść. Wszystkie stany pracy pamięci ujmuje tabl. 15.3.

Tablica 15.3. **Stany pracy pamięci EEPROM 2317**

$\overline{W/R}$	$\overline{OE}$	$\overline{CE}$	Tryb pracy pamięci	Stan we/wy danych
0	0	0	odczyt	wyjście
—	—	1	wyłączenie pamięci	wielka impedancja
⌡	1	0	zapis	wejście

Układ do badania pamięci jest wyposażony w generator fali prostokątnej o częstotliwości ok. 1 kHz zbudowany z bramek NOT (1/6US11, 1/6US16) i elementów R_{37}, R_{38}, C_3. Za pomocą przełącznika bistabilnego „GENERATOR" można spowodować, że wyjście tego generatora zostanie dołączone do wejścia zliczającego licznika US6B, który jest odpowiedzialny za ustawianie trzech najstarszych bitów słowa adresowego. Jeżeli pozostałe wejścia adresowe nie zmieniają się, to zachodzi przeglądanie zawartości pięciu komórek pamięci (licznik US6B liczy **mod 5**), o ile pamięć jest ustawiona w tryb odczytu. Na wyjściu danych występują cyklicznie te same słowa binarne. Zauważmy, że wyjście licznika **mod 5** (US6B) steruje także wejściami adresowymi demultipleksera US10. Z wyjść demultipleksera jest sterowana matryca diodowa. Jedno z tych wyjść jest w stanie niskim. Po zanegowaniu w bramce NOT (US11) stan wysoki steruje odpowiednim tranzystorem, powodując, że katody jednej kolumny diod w matrycy diodowej zostają dołączone do masy. Świecą te diody, które są sterowane przez wyjścia danych pamięci stanem wysokim, a pozostałe są wygaszone. Stany wejść adresowych pamięci $\mathbf{A}_{10}\mathbf{A}_9\mathbf{A}_8$ i stany wejść adresowych demultipleksera US10 zmieniają się współbieżnie (synchronicznie), co sprawia, że każda z kolumn matrycy diodowej jest sterowana każdorazowo tym samym słowem danych wyjściowych. Dzięki dużej częstotliwości oraz bezwładności diod i oka ludzkiego obserwujemy na matrycy stały układ diod świecących i wygaszonych. Odpowiednio programując zawartość przeglądanych komórek pamięci, możemy uzyskać dowolny znak graficzny złożony z 40 elementów ułożonych w 5 kolumn zawierających po 8 diod.

Jeżeli w tym stanie pracy układu będziemy, za pomocą przełącznika „+1", generować impulsy, a adresowalny zatrzask (US4) będzie je kierował do jednego z liczników US7, to na matrycy diodowej będą się pojawiać kolejne znaki graficzne w takt naciskanego przełącznika „+1", o ile wcześniej pamięć została odpowiednio zapisana.

Budowy zasilacza sieciowego nie będziemy analizować, ponieważ jest on zbudowany w sposób typowy (podobnie zasilacz układu stopera — p. 15.1). Zauważmy jedynie, że aby doprowadzić napięcie zasilające do pamięci, należy połączyć ze sobą gniazda (laboratoryjne) G_4, G_5. Jeżeli uczynimy to za pomocą amperomierza, to

zmierzymy prąd pobierany przez badany układ. Ustawiając następnie różne tryby pracy pamięci, zmierzymy wartość prądu pobieranego w tych stanach pracy.

Układ można uprościć, stosując zamiast monowibratorów (US1, US2) dwa przerzutniki asynchroniczne, do czego wystarczy np. jeden układ scalony zawierający 4 dwuwejściowe bramki NAND.

Wartości elementów użytych do budowy układu (R, C) nie są krytyczne i mogą być zmieniane w pewnym zakresie. Także można dobrać inne rodzaje układów scalonych i elementów dyskretnych.

Kompletny zestaw elementów zastosowanych w opisywanym układzie przedstawia poniższy wykaz.

Układy scalone:
US1, US2 — MCY74047 (mono-/astabilny multiwibrator),
US3 — L7805 (stabilizator),
US4 — MCY74099 (adresowalny zatrzask),
US5 — DQ2817 (pamięć EEPROM),
US6, US7, US8 — MCY74520 (dwukrotny, synchroniczny licznik binarny zliczający w górę),
US9 — 74HC241 (bufor z wyjściem trójstanowym),
US10 — 74HCT138 (dekoder/demultiplekser 3 linie na 8 linii),
US11, US15, US16, US17 — 74HC04 (6 bramek NOT),
US12, US13 — 74HC07 (6 buforów z otwartym kolektorem),
US14 — 74HCT00 (4 dwuwejściowe bramki NAND),
MP — RS202L (mostek prostowniczy).

Tranzystory:
T1÷T5 — BD127.

Diody:
D1÷D4 — KA 225 (BAVP 17),
D5÷D15 — CQP443 (diody typu LED — żółte, $U_F = 3{,}2$ V),
D16÷D21 — CQP442 (diody typu LED — zielone, $U_F = 3$ V),
D22, *D23* — CQP441 (diody typu LED — czerwone, $U_F = 2$ V),
matryca diodowa — LTP 2158A.

Rezystory:
R_1, R_3 — 470 kΩ/0,125 W,
R_2, R_4 — 22 kΩ/0,125 W,
$R_5 \div R_{15}$ — 180 Ω/0,125 W,
$R_{16} \div R_{21}$ — 200 Ω/0,125 W,
R_{22}, R_{23} — 300 Ω/0,125 W,
$R_{24} \div R_{28}$ — 4,4 kΩ/0,125 W,
$R_{29} \div R_{36}$ — 300 Ω/0,125 W,
R_{37} — 390 kΩ/0,125 W,
R_{38} — 820 kΩ/0,125 W,
$R_{39} \div R_{41}$ — 4,7 kΩ/0,125 W,

Kondensatory:

C_1, C_2 — 0,22 μF/10 V,

C_3 — 1 nF,

C_4 — 1000 μF/10 V,

C_5 — 100 nF,

C_6 — 100 μF/10 V,

$C_7 \div C_{16}$ — 100 nF (kondensatory blokujące, bezindukcyjne).

Inne:

Tr — Ts 15/18 (transformator sieciowy),

Bt — 63 mA (bezpiecznik topikowy),

izostaty monostabilne — 2szt. („+1", „WYBÓR"),

izostaty bistabilne — 5szt. („SIEĆ", „R/W", „OE", „CE", „GENERATOR"),

B_1, B_2, B_3 — gniazda typu BNC,

$G_1 \div G_5$ — zaciski laboratoryjne.

Pytania i zadania

1. Zaprojektuj układ sterowania czasomierza. Dodatnie zbocze mierzonego impulsu rozpoczyna zliczanie, które obserwujemy na wskaźniku, a ujemne zbocze powoduje wyświetlenie wyniku pomiaru i jednoczesne wyzerowanie liczników — przygotowanie układu do kolejnego cyklu pomiarowego.
2. Rozwiąż zadanie 1. przy założeniu, że przez cały czas kolejnego pomiaru obserwujemy wynik pomiaru poprzedniego. Zakończenie kolejnego pomiaru powoduje przepisanie do bufora stanu liczników i ich wyzerowanie — przygotowanie układu do kolejnego pomiaru.
3. Zaprojektuj obwody wejściowe czasomierza, umożliwiające za pomocą dwóch przycisków bistabilnych dokonywanie niezależnego wyboru dowolnego zbocza przebiegu wejściowego rozpoczynającego i drugiego przebiegu wejściowego kończącego zliczanie.
4. Rozwiąż zadanie 3. przy założeniu, że początek i koniec zliczania określa jeden i ten sam przebieg wejściowy.
5. Rozważ zastąpienie w układzie do badania pamięci (Dodatek 2.) adresowalnego zatrzasku (US4) demultiplekserem (np. takim jak US10). Wymień wady i zalety takiego rozwiązania.
6. Zaprojektuj układ adresowania pamięci za pomocą dwóch przycisków monostabilnych: „+1" — zwiększającego adres, „–1" — zmniejszającego adres. Każde chwilowe naciśnięcie jednego z przycisków zwiększa/zmniejsza adres o 1. Naciśnięcie i przytrzymanie (dłużej niż 1 s) powoduje, że adres zwiększa/zmniejsza się cyklicznie z częstotliwością ok. 4 ÷ 5 Hz. Jeżeli czas przyciśnięcia któregoś z przycisków przekroczy 4 s, to szybkość zmiany adresu zwiększa się do częstotliwości ok. 20 Hz.
7. Zakładając, że dysponujemy już układem działającym jak układ z zadania 6., umożliwiającym ustawienie adresu 8-bitowego, zaprojektować układ umożliwiający wykorzystanie go zarówno do adresowania pamięci, jak i do niezależnego zadawania słowa zapisywanego do pamięci.
8. Jakie czynniki mają wpływ na dokładność pomiaru czasu za pomocą czasomierza cyfrowego? Wymień sposoby pozwalające zwiększyć dokładność pomiaru?

16

Mikroprocesory

16.1. Budowa i zasada działania

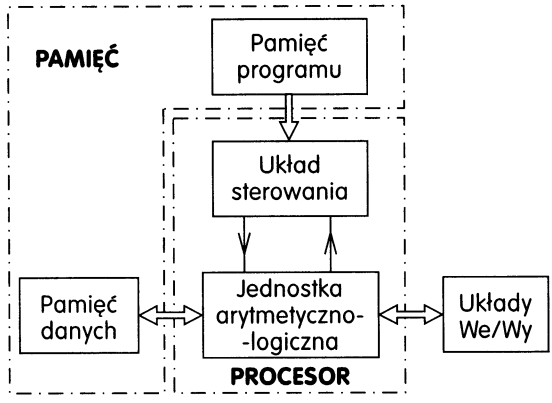
Poznaliśmy do tej pory półprzewodnikowe układy scalone, tj.bramki i przerzutniki budowane jako układy małego stopnia scalenia (SSI), układy kombinacyjne, tj. multipleksery, demultipleksery, kodery, dekodery, komparatory, sumatory, ALU, a także układy sekwencyjne, tj. liczniki i rejestry, ogólnie zwane blokami funkcjonalnymi, wykonane jako układy średniego stopnia scalenia (MSI). Umożliwiają one budowę dowolnie złożonego systemu cyfrowego, realizującego dowolnie złożony algorytm działania. Przykłady złożonych układów cyfrowych przedstawiono w rozdz. 15.

Ogromnym postępem w dziedzinie rozwoju techniki (a techniki cyfrowej w szczególności) okazała się koncepcja opracowana przez amerykańskiego matematyka Johna von Neumanna (na podstawie tej koncepcji pierwszy komputer zbudowano w Wielkiej Brytanii w roku 1949). Koncepcja ta polega na budowie układu realizującego pewien skończony zbiór operacji (jak np. jednostka arytmetyczno-logiczna ALU), a następnie zapewnieniu realizacji dowolnego algorytmu działania poprzez określoną sekwencję tych operacji. Zbiór tych operacji powinien być funkcjonalnie pełny, tzn. zapewniający realizację dowolnie złożonej operacji przetwarzania informacji. Istnieje tu całkowita analogia do pojęcia SFP (systemu funkcjonalnie pełnego) wprowadzonego w p. 3.2, a odnoszącego się do funkcji logicznych (funktorów). Tam SFP (np. bramka NAND) zapewniał realizację dowolnej funkcji logicznej, tutaj zbiór tych operacji musi zapewnić realizację dowolnego algorytmu przetwarzania informacji. Taki układ wraz z układem sterowania jest nazywany **procesorem** i stanowi podstawowy blok komputera. Zadaniem jego jest wykonywanie zawsze takich samych operacji (ze zbioru jakie jest on w stanie realizować) w określonej (w zależności od potrzeb) sekwencji. Gdy procesor jest wykonany jako pojedynczy układ scalony, wówczas nazywamy go **mikroprocesorem** — w skrócie **μP**.

Procesor (czy mikroprocesor) jest także nazywany **jednostką centralną** (**CPU** — ang. *Central Processing Unit* — główna jednostka przetwarzająca), gdyż stanowi on podstawowy blok komputera czy systemu mikroprocesorowego.

Wykonanie określonej operacji przez procesor wymaga ustawienia wejść sterujących w jego układzie sterowania w odpowiedni stan. Stan wejść sterujących jest nazywany **rozkazem.** Zbiór wszystkich rozkazów, jakie może wykonać procesor, jest nazywany **listą rozkazów** procesora. Ciąg rozkazów określający sekwencję wykonywanych operacji jest nazywany **programem**.

Zmieniając tę sekwencję (program), uzyskujemy jakościowo inny system cyfrowy, zdolny realizować zupełnie inne zadania. Ta uniwersalność jest podstawową cechą komputera. Jednocześnie oznacza to potrzebę wyposażenia takiego systemu cyfrowego w pamięć, w której będzie zapisywany aktualnie realizowany algorytm przetwarzania informacji (program). Zatem podstawowymi blokami komputera są: procesor oraz pamięć. System taki może już działać, ale jego użyteczność dla człowieka jest niewielka, ponieważ brak jest komunikacji z takim systemem. Potrzebne są ponadto specjalne układy wejścia/wyjścia, umożliwiające wprowadzanie danych (informacji, jakie mają być przetwarzane) oraz układy, które przekształcą informację wyjściową (uzyskaną poprzez przetwarzanie informacji wejściowej) na postać dogodną dla człowieka, albo na sygnały sterujące urządzeniami zewnętrznymi. Schemat funkcjonalny komputera przedstawiono na rys. 16.1.



Rys. 16.1. Schemat funkcjonalny komputera

Ze względu na lepsze zrozumienie schematu funkcjonalnego pamięć podzielono na dwa obszary, chociaż fizycznie (co zaznaczono linią przerywaną) może to być jedna całość i w dowolnej komórce pamięci może się znajdować rozkaz programu lub przetwarzana informacja.

Wyjaśnijmy nieco szerzej pojęcie **informacji**, które już pojawiało się wcześniej, a które należy rozumieć znacznie szerzej niż fizyczny stan wejść jakiegoś układu cyfrowego. Dla układu cyfrowego informacja wejściowa będzie zawsze w postaci słów binarnych. Ale będzie to tylko kod dwójkowy informacji przetwarzanej w systemie cyfrowym. Jak wiemy (patrz p. 1.3) zakodować binarnie możemy dowolną informację cyfrową. Mogą to być np. litery alfabetu, liczby, temperatury w charakterystycznych punktach sterowanego procesu technologicznego itp. W każdym z tych przypadków musi być inny sposób (program) przetwarzania

informacji wejściowej. W pierwszym przypadku będzie to przetwarzanie tekstu — tzw. edytory tekstowe. W drugim — będą wykonywane obliczenia matematyczne. W trzecim — przetworzenie informacji o temperaturach obiektu sterowanego ma dostarczyć właściwych sygnałów sterujących procesem technologicznym. Wynika z tego, że komputer bez oprogramowania jest narzędziem bezużytecznym. Ale jednocześnie, dzięki możliwości zmiany oprogramowania, jego zastosowania mogą być bardzo różnorodne. Nie należy jednak przeceniać możliwości tych **maszyn cyfrowych**. Warto pamiętać, że nie ma takiego problemu rozwiązanego przez maszynę cyfrową, którego człowiek nie mógłby rozwiązać samodzielnie — pod warunkiem, że miałby na to dostatecznie dużo czasu i możliwość prowadzenia odpowiednio obszernych notatek. Przewaga maszyny cyfrowej nad człowiekiem polega na większej szybkości przetwarzania informacji. Rozwiązanie wielu problemów, które zajmuje maszynie parę godzin czy minut, człowiekowi zajęłoby kilkadziesiąt (a nawet kilkaset) lat.

Wracając do zasadniczego tematu tego rozdziału, przypomnijmy, co to jest mikroprocesor (μP). ***Otóż* mikroprocesor *jest to procesor wykonany w postaci pojedynczego układu scalonego o wielkim stopniu scalenia (LSI lub VLSI)***. Możliwości technologiczne budowy układu scalonego o tak złożonej architekturze pojawiły się na początku lat siedemdziesiątych. Jednym z podstawowych parametrów, charakteryzujących możliwości obliczeniowe μP, jest długość przetwarzanego w nim słowa. Jest to liczba bitów, jaka jest przetwarzana podczas wykonywania pojedynczej operacji (opisana w p. 11.5 jednostka arytmetyczno-logiczna jest jednostką czterobitową). Pierwsze mikroprocesory były μP czterobitowymi. Opracowanie mikroprocesora ośmiobitowego spowodowało ogromny rozwój tej techniki. Jednym z pierwszych μP ośmiobitowych, który chyba został najszerzej rozpowszechniony na świecie, jest μP 8080 firmy Intel. W Polsce jest on budowany jako układ MOS typu MCY7880. Na przykładzie właśnie tego μP poznamy architekturę i zasadę działania mikroprocesorów. Obecnie są budowane μP 16-, 32- i 64-bitowe. Komputer zawierający taki μP ma możliwości obliczeniowe porównywalne z dużymi komputerami lat sześćdziesiątych, których procesory były zbudowane z wielu układów. Ale głównym zastosowaniem μP nie jest produkcja komputerów osobistych czy systemów wieloprocesorowych realizujących równoległe przetwarzanie informacji (systemy z równoległym przetwarzaniem informacji są obecnie najnowszą dziedziną techniki cyfrowej). Mikroprocesor spowodował bowiem zmianę „filozofii" projektowania i budowy układów cyfrowych. Zanim nastała era mikroprocesorów, rozwiązanie dowolnego problemu technicznego polegało na zbudowaniu układu o odpowiedniej strukturze (zbudowanego z elementów dobranych pod kątem realizacji określonego problemu technicznego). Inny problem wymagał zupełnie odrębnej struktury układu. Układy w ten sposób budowane określa się mianem **logiki sprzętowej** (lub **układowej**).

Mikroprocesor umożliwił budowę układów o stałej, uniwersalnej architekturze, których działanie określa informacja zapisana w pamięci układu. Jak już wiemy informacją tą jest sekwencja rozkazów wykonywanych przez μP, nazywana programem, a układy tego typu będziemy nazywać **układami programowanymi** (**logika**

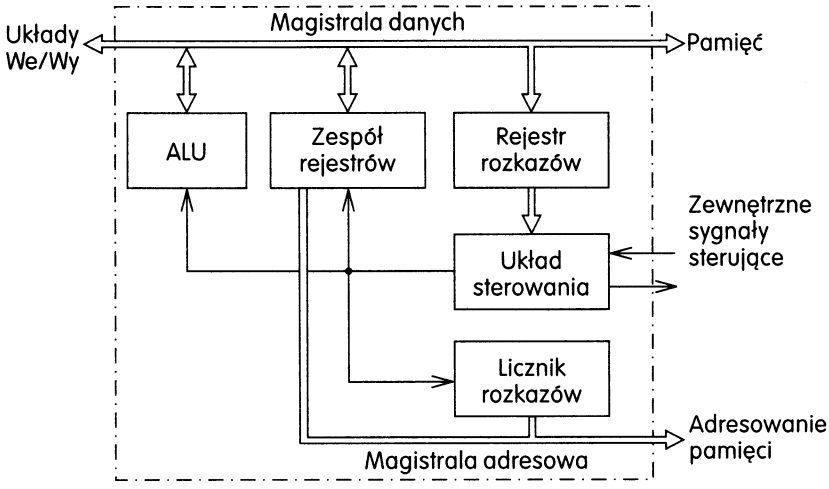
programowa). Realizacja odrębnego problemu technicznego przy wykorzystaniu układu programowanego (systemu mikroprocesorowego) polega więc jedynie na wymianie oprogramowania i ewentualnie układów otoczenia (we/wy). Jak wiemy, minimalną konfiguracją sprzętową zapewniającą działanie takiego systemu jest: μP, pamięć oraz układy wejścia/wyjścia. Najnowszą generacją mikroprocesorów (przeznaczonych do budowy systemów cyfrowych, ale nie komputerów) są mikroprocesory zawierające wszystkie te elementy i dlatego są one nazywane **mikrokomputerami jednoukładowymi** lub **mikrokontrolerami** (chociaż pojemność pamięci wewnętrznej oraz liczba wejść i wyjść jest w takich układach niewielka).

System mikroprocesorowy przeznaczony do określonych zastosowań (np. w pralce automatycznej, magnetowidzie, obrabiarce itp.) realizuje określony i zawsze taki sam program, który w związku z tym może być zapisany w pamięci stałej typu ROM. Układ taki potrzebuje jeszcze nieco pamięci RAM na wprowadzanie przez użytkownika aktualnych danych (np. w pralce automatycznej: wybranie temperatury prania, liczby obrotów wirowania bębna itp.). System taki, zbudowany i zainstalowany w urządzeniu, staje się układem specjalizowanym i nie ma podstawowej cechy komputera, jaką jest wymienność oprogramowania. I jedynie to różni komputer od systemu mikroprocesorowego. Architektura obu systemów jest bowiem identyczna. Mogą być różne układy wejścia/wyjścia, różne proporcje pojemności pamięci ROM i RAM, ale funkcjonalnie będą to takie same bloki.

W komputerze zasadniczą pamięcią będzie zatem pamięć typu RAM, umożliwiająca dowolne zmiany oprogramowania i zmianę danych. W specjalizowanym układzie mikroprocesorowym będzie natomiast dominować pamięć ROM (z zapisanym programem działania). Podobne proporcje (odnośnie do pojemności pamięci) mają właśnie wspomniane powyżej komputery jednoukładowe (przez analogię do μP nazywane mikrokomputerami — μK). Na przykład μK 80C50 (firmy Intel) ma 4 KB (kilobajty — 1024 słów 8-bitowych) pamięci ROM i 256 · 8 (256 słów 8-bitowych) pamięci RAM. Istnieje oczywiście możliwość dołączenia pamięci zewnętrznej, podobnie jak do μP. Ale w wielu zastosowaniach ta ilość pamięci w zupełności wystarcza. Zapisując w pamięci ROM odpowiedni program, uzyskujemy z takiego μK specjalizowany system cyfrowy.

Poznaliśmy już pojęcie procesora, mikroprocesora oraz rolę tych układów w systemach cyfrowych. Zajmiemy się teraz dokładniejszym omówieniem budowy μP i jego działaniem. Schemat funkcjonalny μP (w dużym uproszczeniu) przedstawiono na rys. 16.2.

Zwróćmy uwagę, że wszystkie bloki wchodzące w skład μP zostały już opisane w niniejszym podręczniku. ALU jest bowiem jednostką arytmetyczno-logiczną, która wprawdzie może wykonywać więcej operacji i przetwarzać dłuższe słowo, ale której zasada działania jest identyczna jak opisanej w p. 11.5. Rejestry zostały opisane w rozdz. 13., a liczniki w rozdz. 12. Układ sterowania natomiast jest po prostu generatorem przebiegów uzależnień czasowych. Generuje on impulsy w ściśle określonych sekwencjach w zależności od wejść sterujących, na które jest podawany kod wykonywanej operacji.

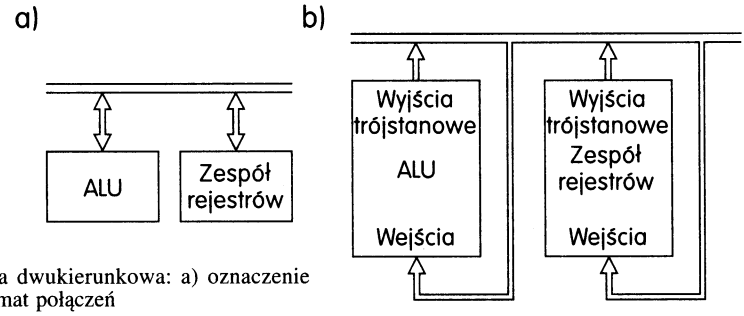


Rys. 16.2. Schemat funkcjonalny μP (uproszczony)

Nieco mniej wiemy na temat magistrali i zanim przejdziemy do opisania cyklu pracy μP, wyjaśnijmy dokładniej to pojęcie.

W systemach mikroprocesorowych przesyłanie informacji odbywa się za pośrednictwem magistrali. **Magistrala** (inaczej **szyna**) jest to wiązka przewodów, do których są dołączone bloki uczestniczące w wymianie informacji. Odbiorniki informacji (ich wejścia) mogą być bezpośrednio dołączone do magistrali, ponieważ liczba odbiorników może być dowolna (podobnie jak z wyjść np. jednego licznika możemy sterować wejściami adresowymi kilku np. multiplekserów).

Inaczej jest ze źródłem informacji. W danej chwili na magistralę może być wysłana informacja pochodząca tylko z jednego źródła. Fizycznie oznacza to, że ***w danej chwili można dołączyć wyjścia tylko jednego bloku współpracującego z magistralą, a wyjścia pozostałych powinny zostać odłączone.*** Wynika z tego, że sygnały wyjściowe bloków dołączanych do magistrali powinny być do niej dołączane za pośrednictwem bramek trójstanowych (patrz p. 5.4 i 6.3) lub bloki te powinny mieć wyjścia trójstanowe. Fragment μP obejmujący dwa bloki: ALU i zespół rejestrów, współpracujących za pośrednictwem wspólnej magistrali, pokazano na rys. 16.3, aby wyjaśnić rzeczywisty sposób połączeń symbolicznie zaznaczany magistralą dwukierunkową.



Rys. 16.3. Magistrala dwukierunkowa: a) oznaczenie symboliczne; b) schemat połączeń

W układzie jak na rys. 16.3b każdy z przedstawionych tam bloków może być zarówno źródłem informacji, jak i odbiornikiem informacji obecnej na magistrali. W danej chwili źródłem informacji może być jeden z nich lub żaden, jeżeli np. informacja jest pobierana z pamięci lub układów wejścia/wyjścia.

Wracając do schematu funkcjonalnego μP (rys. 16.2) zauważmy, że oprócz zespołu rejestrów (tzw. **rejestrów roboczych**, pełniących funkcję pamięci pomocniczej) wyróżniono jeden rejestr, zwany **rejestrem rozkazów.** Jego zadaniem jest pamiętanie operacji, jaka jest wykonywana w jednym cyklu pracy μP. Precyzując, w rejestr ten jest wpisywany (z pamięci programu) **kod tej operacji (rozkazu).** Ponadto stan tego rejestru jest podawany na wejścia układu sterującego, który na jego podstawie generuje odpowiednią sekwencję przebiegów sterujących wykonaniem tej operacji.

Układ sterowania μP działa cyklicznie, wykonując tzw. **cykl rozkazowy**, w którym można wyróżnić dwie fazy:

— pobranie kolejnego rozkazu z pamięci;
— wykonanie tego rozkazu, tj. wykonanie operacji określonej przez jej kod zapisany w pamięci.

Pobieranie rozkazów przebiega w sposób następujący: Na magistralę adresową jest wysyłana zawartość licznika rozkazów. Ta zawartość określa **adres komórki pamięci**, w której jest zapisany kod kolejnej operacji (rozkaz). Zawartość tej komórki (kod binarny rozkazu) jest wpisywana do rejestru rozkazów. Ponieważ przeważnie kolejne rozkazy są pamiętane w kolejnych komórkach pamięci, przeto układ sterujący powoduje zwiększenie stanu licznika rozkazów o 1. Przygotowany jest w ten sposób kolejny adres, spod którego będzie pobierany kolejny rozkaz.

W drugiej fazie jest wykonywany rozkaz określony przez kod operacji, który jest wpisany w rejestr rozkazów. Po zakończeniu drugiej fazy układ jest gotowy do następnego cyklu.

Spróbujmy — nie znając jeszcze nawet przykładowej listy rozkazów, jakie może wykonywać μP — określić na podstawie jego schematu funkcjonalnego (rys. 16.2) możliwe do wykonania operacje, wynikające z samych połączeń i kierunków przepływu informacji.

Najpierw operacje dotyczące wyłącznie bloków przedstawionych na rysunku. Ze schematu wynika, że magistralę adresową możemy zaadresować zawartością licznika rozkazów lub stanem rejestru z zespołu rejestrów. Do zespołu rejestrów może być wpisany stan wyjść jednostki arytmetyczno-logicznej (ALU) lub ALU może wykonać operację na zawartości rejestru roboczego.

Oba bloki (ALU i zespół rejestrów) współpracują z magistralą danych, do której podłącza się zewnętrzne układy (pokazane na schemacie komputera — rys. 16.1). Są to pamięć i układy wejścia/wyjścia. Ze schematu wynika zatem, że informacja zapisana w pamięci (lub wprowadzana przez układy we/wy) może zostać przesłana do:

— rejestru rozkazów,
— do zespołu rejestrów,

— do jednostki arytmetyczno-logicznej (argumentem operacji wykonywanej przez ALU może być więc albo zawartość rejestru z zespołu rejestrów, albo zawartość komórki pamięci, albo informacja wprowadzona z układu we/wy).

Analizując dalej schemat funkcjonalny możemy zauważyć, że magistrala danych jest dwukierunkowa, z czego wynika, że zawartości rejestrów roboczych lub wynik operacji (stan wyjść jednostki ALU) może zostać przesłany do pamięci, układów we/wy lub w kierunku przeciwnym.

Operacje powyższe można podzielić na dwie zasadnicze grupy:

1. **Rozkazy przesłania** — umożliwiające przekazywanie informacji pomiędzy rejestrami, pamięcią i układami we/wy.
2. **Operacje na zawartościach rejestrów** — realizowane przez ALU.

Istnieje jeszcze trzecia grupa rozkazów i to bardzo istotna, ale jest ona po prostu podzbiorem grupy 2. Jak wspomniano powyżej, program realizowany przez μP jest to zbiór rozkazów (operacji) zapisanych w kolejnych komórkach pamięci, za adresowanie których odpowiada licznik rozkazów sterowany przez układ sterujący. Z omówionego cyklu pracy μP wynika więc, że wykonanie takiego programu będzie przebiegało zawsze tak samo.

Wyobraźmy sobie, że tak działający system mikroprocesorowy steruje pracą magnetowidu. Wkładamy więc kasetę i uruchamiamy magnetowid. Rozpoczyna się realizacja programu zapisanego w pamięci, czyli wprawienie w ruch obrotowy głowicy wirującej, nawinięcie taśmy na głowicę, uruchomienie przesuwu taśmy itd. (każda z tych czynności wykonywanych przez magnetowid to szereg operacji realizowanych przez μP). A co, jeżeli chcemy tylko przewinąć taśmę? Powinien wtedy być zrealizowany inny program. Zatem musi istnieć możliwość zmiany (programowo) zawartości licznika rozkazów. Taka zmiana sprawi, że rozpocznie się realizacja programu zapisanego w innym obszarze pamięci. Rozkaz pozwalający zrealizować taką zmianę zawartości licznika rozkazów jest nazywany **rozkazem skoku**. Rozkaz skoku może być **bezwarunkowy,** tzn. „skocz i już" albo **warunkowy** „skocz pod warunkiem, że". Który z nich byłby bardziej właściwy w tym przypadku? Na pewno warunkowy, gdyż dobrze byłoby sprawdzić (za pomocą czujników) przed uruchomieniem przewijania taśmy, czy jest co przewijać (tzn. czy kaseta jest w magnetowidzie).

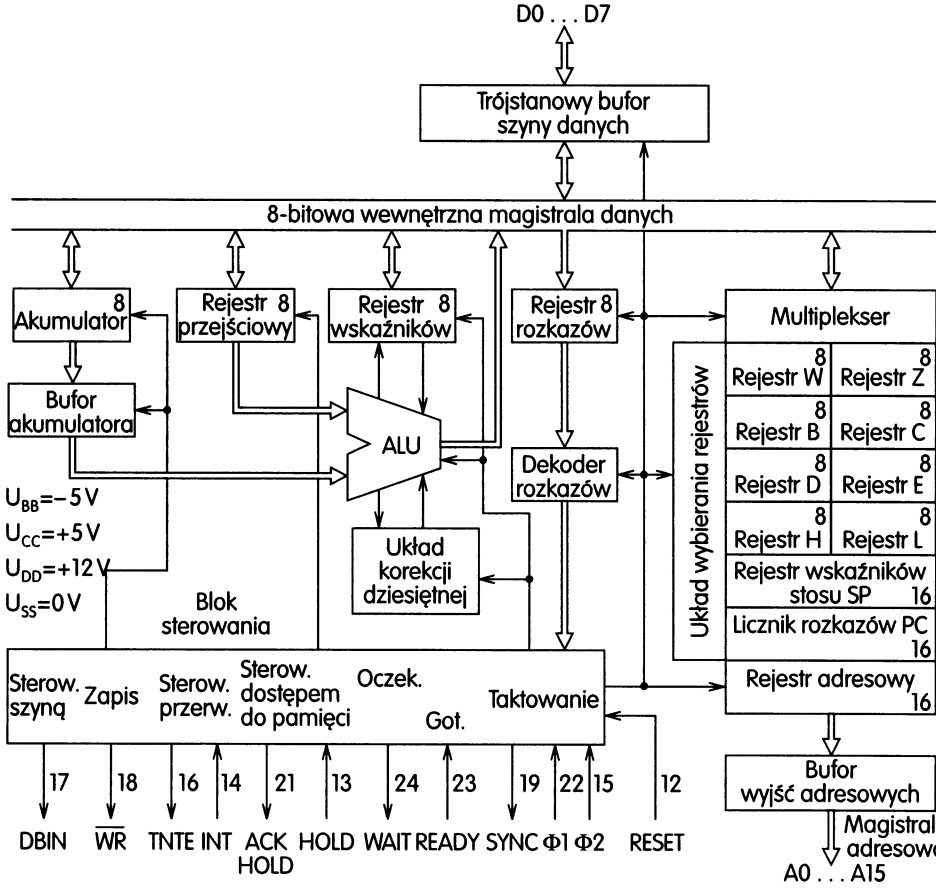
Rozkazy skoku pozwalają więc sterować wykonywaniem programu, co jest niezbędnym warunkiem efektywnego wykorzystania cyfrowych systemów programowanych. Rozkazy te należą właśnie do trzeciej grupy rozkazów mikroprocesora — do **grupy rozkazów sterujących**. Wprawdzie polegają one na zmianie zawartości pewnych rejestrów (i zaliczyć je można do drugiej grupy), ale ze względu na ich duże znaczenie są wyodrębniane często jako oddzielna grupa.

16.2. Mikroprocesor 8080

Architekturę (schemat funkcjonalny) μP 8080 (CPU) pokazano na rys. 16.4. Mikroprocesor 8080 (polski odpowiednik MCY7880) jest μP 8-bitowym, co oznacza, że wykonuje operacje na słowach 8-bitowych. Konsekwencją 8-bitowych re-

jestrów roboczych są 8-bitowe słowa zapamiętywane w komórkach pamięci RAM i ROM, z którymi współpracuje taki μP oraz 8-bitowa magistrala danych. Gdyby i licznik rozkazów był 8-bitowy, to pozwoliłoby to na zaadresowanie jedynie 256 (2^8) komórek pamięci. Dla większości zastosowań w systemach mikroprocesorowych jest to za mało (a tym bardziej do budowy komputera), dlatego μP 8080 ma licznik rozkazów 16-bitowy, co pozwala zaadresować pamięć o pojemności 64 KB (2^{16}). Jest to pamięć wystarczająca dla większości zastosowań w systemach mikroprocesorowych. W związku z tym rejestr adresowy oraz magistrala adresowa jest 16-bitowa. Pamiętamy z poprzedniego rozdziału, że istnieje możliwość adresowania (ustawiania stanu szyny adresowej) za pomocą zespołu rejestrów. Ponieważ rejestry te są 8-bitowe, więc adresowanie można zrealizować, używając dwóch takich rejestrów, poprzez wpisanie do nich dwóch słów 8-bitowych (najczęściej używa się do tego pary rejestrów HL). Rejestry robocze, znajdujące się obok siebie (rys. 16.4), mogą być łączone w pary i używane jako rejestry 16-bitowe.

Układ wybierania rejestrów (zwany też selektorem rejestrów) wraz z multiplekserem pozwalają na sterowanie komunikowaniem się poszczególnych rejestrów z magistralą danych.



Rys. 16.4. Schemat funkcjonalny mikroprocesora MCY7880

W zespole rejestrów roboczych występuje rejestr nazwany **rejestrem wskaźnika stosu** (lub krócej **wskaźnikiem stosu**), który — podobnie jak rejestr adresowy — jest rejestrem 16-bitowym. Ponieważ oba te rejestry pełnią inne funkcje niż pozostałe, przeto do wykonywania operacji na ich zawartości służą odrębne rozkazy (patrz p. 16.3). Jednak w tym miejscu zajmiemy się jedynie przeznaczeniem tego rejestru.

Wskaźnik stosu SP jest to rejestr 16-bitowy, w którym jest przechowywany adres komórki pamięci. Wyjaśnijmy zatem, co jest szczególnego w jakiejś komórce pamięci, że powinna ona być pamiętana w specjalnie do tego przeznaczonym rejestrze.

Posłużmy się ponownie przykładem programu sterującego działaniem magnetowidu. Stwierdziliśmy już, że do obsługi różnych poleceń użytkownika magnetowidu należy zapisać szereg programów, rozmieszczonych w różnych obszarach pamięci. Proponowana liczba programów była równa liczbie poleceń. Byłoby to w praktyce postępowanie dalekie od optymalnego. Szczególnie, jeśli weźmiemy pod uwagę, że programy te musiałyby zawierać jednakowe sekwencje poleceń. Na przykład program obsługi polecenia „przewiń taśmę" i program obsługi polecenia „odtwarzaj nagranie" będą mieć identyczną sekwencję rozkazów mikroprocesora, potrzebną do sprawdzenia obecności kasety w magnetowidzie. Sensowniej jest więc taką sekwencję zapisać raz w pamięci i odwoływać się do niej w tych miejscach (**programu głównego**), w których będzie to konieczne. Taką sekwencję rozkazów, która jest zapisana tylko raz w pamięci, lecz wykonywana wielokrotnie w różnych miejscach programu, nazywamy **podprogramem**. Odwołanie się do podprogramu jest realizowane także przez rozkaz skoku (zmiany zawartości licznika rozkazów), lecz rozkaz skoku do podprogramu tym się różni od zwykłego rozkazu skoku, że zostaje zapamiętana ostatnia (przed skokiem do podprogramu) zawartość licznika rozkazów. Umożliwia to powrót z podprogramu do programu głównego poprzez proste wpisanie do licznika rozkazów (po zakończeniu realizacji podprogramu) zapamiętanej zawartości licznika rozkazów. W rejestrze wskaźnika stosu nie jest jednak pamiętany ten ostatni (przed skokiem do podprogramu) stan licznika rozkazów. Wynika to stąd, że aby kontynuować realizację programu głównego dokładnie od tego miejsca, z którego było odwołanie do podprogramu, trzeba zapamiętać nie tylko stan licznika rozkazów, ale także rejestrów pomocniczych i rejestru akumulatora (o rejestrze akumulatora powiemy trochę dalej). Do tego celu potrzebny jest więc pewien fragment pamięci. W praktyce może to wyglądać na przykład tak:

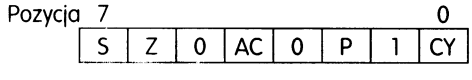
1. Pobieramy rozkaz z komórki pamięci o adresie **100**, który jest rozkazem skoku do podprogramu (w tym momencie licznik rozkazów jest ustawiony już na wartości **101**).
2. Jeszcze więc przed zmianą skokową licznika rozkazów (tzn. skokiem do podprogramu) jest zapamiętywany stan licznika rozkazów (ten **101**) oraz stany rejestrów. Przyjmijmy, że informacje te są zapisywane w kolejnych komórkach pamięci o adresach 1, 2, 3 i ostatnia w komórce o adresie 10.
3. I właśnie ten adres (10) zostanie zapamiętany w rejestrze wskaźnika stosu.

4. Wykonanie podprogramu jest kończone zawsze rozkazem powrotu do programu głównego. Wykonanie tego powrotu to wpisanie do rejestrów μP informacji z obszaru pamięci wskazywanej przez wskaźnik stosu w odwrotnej kolejności, jak była ona tam zapisywana.

Fragment pamięci, w której są zapisywane stany rejestrów obecne w chwili skoku do podprogramu, jest nazywany **stosem.** Wyjaśnijmy, dlaczego użyto takiego określenia. Otóż skok do podprogramu może wystąpić w dowolnym miejscu programu, ***także w trakcie wykonywanego podprogramu***. W niniejszym przykładzie, gdyby w podprogramie znalazło się odwołanie do kolejnego podprogramu, wówczas stany rejestrów zostałyby wpisane do komórek pamięci 11, 12,... i 20. Wskaźnik stosu przyjąłby więc wartość równą 20. Kolejne odwołanie to zapis w komórkach 21, 22,... aż do 30, wskaźnik stosu 30 itd. Informacja w obszarze stosu jest więc zapisywana coraz wyżej (w obszar o wyższych adresach), a dostęp do niej jest jednostronny — tylko od „góry", podobnie jak do ***stosu*** talerzy ułożonych jeden na drugim.

Na schemacie funkcjonalnym μP (rys. 16.4) można zauważyć jeszcze 8-bitowy rejestr zwany **akumulatorem** (oznaczany literą *A*). Jest to jeden z podstawowych rejestrów μP. Wszystkie operacje arytmetyczne i logiczne (realizowane przy użyciu ALU) są wykonywane na zawartości tego rejestru, a wynik takiej operacji jest umieszczany zawsze w tym rejestrze. Na przykład rozkaz *ADD B* oznacza: dodaj do zawartości rejestru akumulatora zawartość rejestru B i umieść wynik w akumulatorze. Symbolicznie operacja ta jest zapisywana $A := A + B$. Znak równości jest tu poprzedzony dwukropkiem dla odróżnienia czysto matematycznego zapisu, który byłby przecież sprzeczny, bowiem $A \neq A + B$. Zapis $A := A + B$ czytamy w sposób następujący: do *A* dodaj *B* i zapamiętaj pod (w) *A*.

Na schemacie funkcjonalnym (rys. 16.4) został wyróżniony **rejestr wskaźników** zwany także **rejestrem flagowym** (ang. *flag* — znak sygnalizujący) lub **rejestrem bitów warunkowych**. Jest on ściśle związany z jednostką ALU, gdyż zawiera przerzutniki, do których są wpisywane stany wyjść pomocniczych jednostki arytmetyczno-logicznej (patrz p. 11.5, rys. 11.6). Wyjścia te sygnalizują stany szczególne wyników operacji realizowanych przez ALU, takie jak np.: wynik równy zero, pojawienie się przeniesienia, bit znaku itp. Znaczniki te ustawiane przez ALU (czyli poszczególne przerzutniki w rejestrze flagowym) pozwalają na dokonywanie rozgałęzień w programie w zależności od wyniku poprzednio wykonanej operacji. Na przykład program, który oblicza pierwiastki równania kwadratowego (o danych współczynnikach *a*, *b* i *c*), powinien sprawdzić znak $\Delta = b^2 - 4ac$, aby odpowiednio poprowadzić dalsze obliczenia. Do tego celu służą specjalne rozkazy sterujące wykonywaniem programu, zwane rozkazami skoku. Rejestr wskaźników (flagowy) jest rejestrem 8-bitowym, w którym μP 8080 wykorzystuje jedynie 5 bitów — wskaźników. Rozmieszczenie tych wskaźników w rejestrze flagowym pokazano na rys. 16.5. Znaczenie poszczególnych wskaźników zestawiono w tabl. 16.1.



Rys. 16.5. Rejestr wskaźników (flagowy) mikroprocesora MCY7880

Tablica 16.1. **Zawartość rejestru wskaźników (flagowego)**

Wskaźnik	Wartość	Wskazywana informacja
Z	1 0	wynik operacji równy 0 wynik operacji nie równy 0
S	1 0	bit znaku — liczba ujemna bit znaku — liczba dodatnia
CY	1 0	przeniesienie (przeniesienie = 1) brak przeniesienia (przeniesienie = 0) (uwaga: jeżeli operacją było odejmowanie, to jest to pożyczka)
P	1 0	gdy w wyniku jest parzysta liczba „1" gdy w wyniku jest nieparzysta liczba „1"
AC		pomocniczy bit przeniesienia sygnalizujący wystąpienie przeniesienia z pozycji D_3 na D_4 (nie wykorzystywany przez rozkazy sterujące)

Wskaźnik **AC** dotyczy operacji arytmetycznych, które są wykonywane w kodzie BCD: $\mathbf{D_7D_6D_5D_4}$ | $\mathbf{D_3D_2D_1D_0}$.

Poniżej zamieszczono opis wyprowadzeń μP MCY7880 pokazanych na rys. 16.4 oraz końcówek zasilających (nie pokazanych na rys. 16.4). Może on być w wielu miejscach niezrozumiały, ale znaczenie i rolę nie omówionych do tej pory wyprowadzeń znajdzie Czytelnik w dalszej części podręcznika.

$\mathbf{A_{15} \div A_0}$ — wyjście trzystanowej magistrali adresowej adresującej pamięć 64 KB (2^{16}) lub wskazującej numer urządzenia we/wy (512 urządzeń we/wy);

$\mathbf{D_7 \div D_0}$ — wejścia/wyjścia trzystanowej dwukierunkowej szyny (magistrali) danych, umożliwiające komunikowanie się μP z pamięcią i urządzeniami we/wy;

SYNC — (ang. *SYNChronization*) wyjście sygnału synchronizacji; sygnał określa początek każdego cyklu maszynowego;

DBIN — (ang. *Data Bus INput)* wyjście sygnału wskazującego zewnętrznym urządzeniom stan oczekiwania μP na dane; magistrala danych ustawiona w stan wejścia;

READY — wejście sygnału wskazującego jednostce centralnej obecność danych na magistrali; sygnał jest używany do synchronizacji CPU z pamięcią lub urządzeniami we/wy, jeżeli po wysłaniu adresu jednostka centralna (μP) nie otrzyma sygnału na wejściu **READY**, wejdzie w stan oczekiwania (**WAIT**) i pozostanie w nim do chwili pojawienia się stanu wysokiego na linii **READY**;

WAIT — wyjście sygnału potwierdzającego stan oczekiwania jednostki centralnej na dane; **WAIT = 1**, to μP oczekuje na zgłoszenie gotowości do obsługi układu zewnętrznego (pamięć, układy we/wy);

$\overline{\mathbf{WR}}$ — (ang. *WRite*) wyjście sygnału zapisu do pamięci lub transmisji danych do urządzeń we/wy; gdy dane są wysyłane przez μP, wówczas wyjście to jest ustawiane w stan niski;

HOLD — wejście sygnału żądającego od μP oddania kontroli nad zewnętrzną magistralą adresową, szyną danych i magistralą sterującą urządzeniem zewnętrznym (zwykle źródłem tego sygnału jest sterownik DMA lub inny procesor); w następstwie tego sygnału jednostka centralna zawiesza swoją pracę, wprowadza magistralę adresową i danych w stan wielkiej impedancji i wysyła sygnał **HLDA**;

HLDA — (ang. *HoLD Acknowledge*) wyjście sygnału potwierdzającego możliwość przejęcia kontroli nad szynami danych i adresową np. przez sterownik DMA (realizacja bezpośredniego dostępu do pamięci); wyjście sygnału — patrz **HOLD**;

INT — (ang. *INTerrupt*) wejście sygnału przerwania, określa żądanie przerwania wykonywanego programu; sygnał nie jest akceptowany wówczas, gdy system jest w stanie **HLDA** lub przerzutnik stanu przerwania jest wyzerowany;

INTE — (ang. *INTerrupt Enable*) wyjście sygnału określającego stan przerzutnika przerwania; przerzutnik ten jest zerowany samoczynnie po otrzymaniu sygnału **RESET** lub rozpoczęciu obsługi przerwania;

RESET — wejście sygnału zerowania; μP po odebraniu tego sygnału zeruje licznik rozkazów i przerzutniki sygnałów **INTE** i **HLDA**; nie są zerowane: akumulator, rejestr flagowy, rejestr wskaźnika stosu, rejestry ogólnego przeznaczenia; sygnał RESET musi być w stanie aktywnym przez minimum 3 cykle zegarowe;

Φ_1, Φ_2 — wejścia sygnałów zegarowych (jedyne wejścia nie współpracujące z układami TTL).

Wszystkie końcówki μP Intel 8080 (MCY7880) (z wyjątkiem wejść Φ_1, Φ_2) przyjmują lub generują sygnały typu TTL. Obciążalność wyjść jest jednak mniejsza niż standardowego wyjścia TTL i wynosi 1,7 mA. Dlatego często wykorzystuje się wzmacniacze buforowe na magistrali danych i na magistrali adresowej.

Układ ma 4 końcówki, służące do doprowadzenia napięć zasilających układ. Są to:
U_{CC} (20) — do której doprowadza się napięcie +5 V (TTL),
U_{DD} (28) — zasilana napięciem +12 V,
U_{BB} (11) — zasilana napięciem −5 V,
U_{SS} (2) — masa układu.

Do mikroprocesora MCY7880 dołącza się dwa układy pomocnicze. Są to:
— układ **zegara** 8224 (polski odpowiednik 'S424),
— tzw. **sterownik (kontroler) systemu** 8228 (polski odpowiednik 'S428).

Ponieważ obecnie μP 8080 nie jest produkowany i nie wykorzystuje się go już do budowy nowych systemów mikroprocesorowych, więc w podręczniku zrezygnowano z opisu tych układów. Jest to jednak jeden z najprostszych mikroprocesorów, a podstawowe zasady ich programowania są podobne dla wszystkich μP, więc na jego przykładzie w p.16.3 zostaną omówione rozkazy μP 8080 oraz programowanie mikroprocesora.

16.3. Programowanie mikroprocesorów

Z dotychczasowych rozważań wynika, że μP jest zdolny wyłącznie do realizacji sekwencji ściśle zdefiniowanych operacji arytmetyczno-logicznych (ALU). Aby możliwe więc było zastosowanie go do realizacji określonych zadań (np. sterowanie pracą magnetowidu, pralki), należy rozwiązanie tego zadania sprowadzić do elementarnych operacji wykonywanych przez μP. Proces taki nazywa się **algorytmizacją** problemu. Przez **algorytm** należy rozumieć przepis postępowania pozwalający osiągnąć cel (rozwiązanie jakiegoś problemu).

Przepis ten należy następnie sformułować w „języku" zrozumiałym przez μP. Języki służące do zapisu programów nazywa się **językami programowania**. Jak wiemy, jedynym takim językiem dla μP jest język operujący słowami dwójkowymi, które należy zapisać w pamięci programu i które są kodami rozkazów wykonywanych przez μP. Jest to tzw. **język wewnętrzny systemu** (komputera). Zapisanie algorytmu w takim języku byłoby dla człowieka niezwykle skomplikowane, nawet jeżeli są to słowa tylko 8-bitowe (jak dla MCY7880). Zamiana ich na zapis szesnastkowy, czy nawet dziesiętny też nie upraszcza wystarczająco tego problemu. Dlatego do zapisywania programów używa się zawsze **języków symbolicznych**. Zamiast pisać kod binarny (szesnastkowy lub dziesiętny) jakiegoś rozkazu μP zapisujemy jego symbol słowny, którego znaczenie odpowiada realizowanej operacji. Zamiast więc pisać **1000 0110** (lub szesnastkowo 86H, albo dziesiętnie 134), co jest rozkazem dla μP (8080) „dodaj do akumulatora zawartość komórki pamięci, której adres znajduje się w rejestrze HL" (czyli $A := A + M(HL)$), zapiszemy symbolicznie: *ADD M*, gdzie słowo *ADD* oznacza — dodaj (ang. *add* — dodaj), a litera *M* jest pierwszą literą słowa *memory* (ang. *memory* — pamięć).

Ale używanie języka symbolicznego wymaga napisania programu tłumaczącego język symboliczny na język słów binarnych (język wewnętrzny). Nawet gdybyśmy używali zapisu szesnastkowego (lub dziesiętnego), to i tak potrzebny byłby taki program tłumaczący.

***Proces tłumaczenia języka symbolicznego na język wewnętrzny nazywa się* translacją, *a program tłumaczący* translatorem. *Program wyrażony w języku symbolicznym jest wówczas* programem źródłowym, *a program uzyskany w wyniku translacji (tłumaczenia) jest programem* wynikowym.**

Wprowadzenie języków symbolicznych jest działaniem racjonalnym, mimo że wymaga opracowania translatora (programu tłumaczącego). Program tłumaczący opracowuje się raz, a można korzystać z niego wielokrotnie. Jest to po prostu jedno z możliwych zastosowań komputera, dla którego pisze się raz program, a wykorzystuje go wiele razy dla różnych danych wejściowych (w tym przypadku różnych programów źródłowych).

Najprostszym językiem symbolicznym jest tzw. **język asemblera** (potocznie, krócej, nazywany asemblerem), w którym jednemu rozkazowi μP odpowiada jeden zapis symboliczny. Taki zapis symboliczny operacji wykonywanych przez μP

nazywa się **instrukcją**. Istnieją języki (symboliczne) programowania, w których jednej instrukcji odpowiada bardzo wiele operacji μP (czyli tym samym instrukcji asemblera). Są to tak zwane języki wyższego poziomu. Należą do nich np.: Pascal, Cobol, język C, Basic i wiele innych.

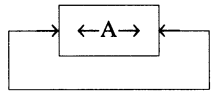
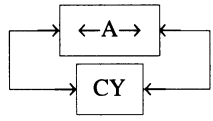
Asemblerem nazywamy nie tylko język symboliczny (w którym jedna instrukcja odpowiada dokładnie jednej operacji μP), ale także program tłumaczący (czyli translator) tego języka na język wewnętrzny. W związku z tym proces tłumaczenia programu źródłowego napisanego w asemblerze nazywa się niekiedy **asemblacją**. (Dla procesu odwrotnego, który jest także możliwy, używa się określenia **disasemblacji**.) Symbole w asemblerze utworzono zgodnie z prawidłami mnemotechniki, czyli w taki sposób, aby sama forma zapisu pomagała zapamiętać, jaką operację oznacza dany symbol. Dlatego są to słowa (lub skróty tych słów) odpowiadające realizowanej operacji (wzięte z języka angielskiego, jak w przykładzie powyżej). Z tego powodu symbol przypisany poszczególnym operacjom nazywa się często **mnemonikiem** (skrótem mnemonicznym).

Jedynym warunkiem, jaki powinna spełniać lista rozkazów μP, jest to, że powinna być funkcjonalnie pełna. Oznacza to możliwość zapisu dowolnego algorytmu w ramach danej listy rozkazów. Przeważnie lista tych rozkazów nie jest zbiorem minimalnym. To znaczy możliwe byłoby usunięcie z niej jeszcze wielu rozkazów, a i tak pozostałaby ona funkcjonalnie pełna. Jednak, ze względu na wygodę programowania, lista tych rozkazów jest dłuższa. Pełną listę rozkazów μP 8080 przedstawiono w tabl. 16.2.

Tablica 16.2. **Lista rozkazów μP 8080**

Mnemonik	Nazwa	Operacja
Operacje przesłania x = r, M p = BC, DE		
MOV r,x	przesłanie	r := x
MOV x,r	przesłanie	x := r
MVI x,n	przesłanie argumentu bezpośredniego	r := x
STAX p	pamiętaj akumulator	M(p) := A
LDAX p	ładuj akumulator	A := M(p)
STA aa	pamiętaj akumulator	M(aa) := A
LDA aa	ładuj akumulator	A := M(aa)
Operacje na zawartości rejestrów 8-bitowych x = r, M		
ADD x	dodaj	A :=A + x
ADI n	dodaj argument bezpośredni	A := A + n
ADC x	dodaj z przeniesieniem	A := A + x + CY
ACI n	dodaj argument bezpośredni i przeniesienie	A := A + n + CY
SUB x	odejmij	A :=A − x
SUI n	odejmij argument bezpośredni	A :=A − n
SBB x	odejmij z pożyczką	A := A − x − CY
SBI n	odejmij argument bezpośredni i pożyczkę	A := A − n − CY
ANA x	iloczyn logiczny	A := A AND x
ANI n	iloczyn logiczny	A := A AND n

Tablica 16.2. cd.

Mnemonik	Nazwa	Operacja
ORA x	suma logiczna	A := A OR x
ORI n	suma logiczna	A := A OR n
XRA x	suma modulo dwa	A := A⊕x
XRI n	suma modulo dwa	A := A⊕n
CMP x	porównaj	A − x
CPI n	porównaj	A − n
INR x	zwiększ o jeden	x := x + 1
DCR x	zmniejsz o jeden	x := x − 1
CMA	neguj akumulator	A := $\overline{A}$
DAA	korekcja dziesiętna akumulatora	
RLC	przesuw A w lewo	
RRC	przesuw A w prawo	
RAL	przesuw A w lewo z bitem przeniesienia	
RAR	przesuw A w prawo z bitem przeniesienia	
STC	ustaw CY	CY := 1
CMC	neguj CY	CY := $\overline{CY}$
Operacje na parach rejestrów p = BC, DE, HL, SP		
LXI p,nn	ładuj parę rejestrów	p := nn
INX p	zwiększ parę rejestrów	p := p + 1
DCX p	zmniejsz parę rejestrów	p := p − 1
DAD p	dodaj parę do HL	HL := HL + p
SHLD aa	zapamiętaj HL	M(aa) := HL
LHLD aa	ładuj HL	HL := M(aa)
XCHG	wymiana HL i DE	HL↔DE
SPHL	prześlij HL do SP	SP := HL
Skoki w = Z, NZ, S, NS, P, NP n = 0, 1, ..., 7		
JMP aa	skok bezwarunkowy	PC := aa
Jw aa	skok warunkowy, Z, N, P bity rejestru flagowego (NZ = NOT Z, itd.) skok jest wykonywany gdy w = 1	jeżeli w, to PC := aa
CALL aa	skok do podprogramu	M(SP − 2) := PC SP := SP − 2 PC := aa
Cw aa	warunkowy skok do podprogramu	
RST n	restart	jak CALL, lecz PC := 8n
PCHL	skok pośredni	PC := HL
RET	powrót z podprogramu	PC := M(SP) SP := SP + 2
Rw	warunkowy powrót z podprogramu	

Tablica 16.2. cd.

Mnemonik	Nazwa	Operacja
Operacje na zawartości stosu p = BC, DE, HL, PSW		
PUSH p	zapisz na stos	M(SP – 2) := p SP := SP – 2
POP p	odczytaj ze stosu	p := M(SP) SP := SP + 2
XTHL	wymiana wierzchołka stosu i HL	M(SP)↔HL
Operacje wejścia/wyjścia n – 8-bitowy adres układu we/wy		
IN n	wejście	A := we(n)
OUT n	wyjście	wy(n) := A
Inne		
EI	włącz system przerwań	
DI	wyłącz system przerwań	
NOP	nic nie rób	
HLT	stop	

Rozkaz w języku asemblera ma następujący format:

<etykieta:> symbol operacji <argument>,<argument>

przy czym te pola, które ujęto w nawias < > są opcjonalne, tzn. mogą wystąpić lub nie. Zapis powyższy oznacza więc, że zawsze musi wystąpić symbol operacji. Operacja może być bez-, jedno- lub dwuargumentowa. Argument operacji wykonywanej przez μP jest często nazywany **operandem**. Dla wygody programisty asembler dopuszcza zamieszczanie komentarzy. Powinno się je pisać po znaku średnika.

● **Zasady pisania programów w asemblerze** μP 8080 prześledzimy na kilku poniższych przykładach. W tym celu należy sobie przypomnieć zapis heksadecymalny (szesnastkowy). Asembler dopuszcza pisanie danych w postaci dziesiętnej, szesnastkowej lub binarnej. Liczba szesnastkowa jest kończona literą H, a liczba dwójkowa literą B. Liczba dziesiętna jest pisana bez żadnego dodatkowego symbolu. Dopuszczalne rodzaje zapisu zostaną przedstawione w poniższych przykładach, ale podstawowym zapisem będzie zapis szesnastkowy. W każdym z zapisów dopuszcza się pomijanie zer nieznaczących. Tak więc adres, który jest liczbą binarną 16-bitową powinien być zapisany przy użyciu 4 lub mniej cyfr szesnastkowych, np. adres 0100H wolno zapisać jako 100H. Dane dwójkowe mogą zawierać 8 bitów, czyli np. liczba dziesiętna 3 może mieć postać **00000011B**, ale wolno skrócić ten zapis do postaci **11B**.

Przykład 16.1

```
                ;SUMOWANIE TRZECH LICZB (1,2,3)
ORG 100H        ;POCZĄTEK OBSZARU PAMIĘCI PROGRAMU
ANA 0H          ;ZEROWANIE AKUMULATORA
ADI 1H          ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY 1
                ;(w zapisie szesnastkowym)
ADI 2           ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY 2
                ;(w zapisie dziesiętnym)
ADI 11B         ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY 3
                ;(w zapisie binarnym)
STA 200H        ;PRZESŁANIE ZAWARTOŚCI AKUMULATORA DO KOMÓRKI
                ;PAMIĘCI O ADRESIE 200H
END
```

W powyższym przykładzie wystąpiły tzw. **pseudoinstrukcje** (ORG, END), które nie są rozkazami μP. Potrzebne są one translatorowi (programowi tłumaczącemu) do sterowania procesem translacji. Pseudoinstrukcja ORG ustawia adres, od którego będą wpisywane kolejne rozkazy lub dane. Pseudoinstrukcja END może być pominięta, ale jeżeli wystąpi, to jest ostatnią instrukcją programu. Wszystkie instrukcje, które wystąpią po niej są ignorowane.

Tablica 16.3. **Mapa pamięci programu z przykładu 16.1**

Adres komórki pamięci	Zawartość komórki	Komentarz
0100	A0	kod rozkazu μP — ANA — iloczyn logiczny
0101	C6	kod rozkazu μP — ADI — dodaj argument bezpośredni
0102	01	argument rozkazu ADI, dodawana liczba 01H
0103	C6	kod rozkazu μP — ADI — dodaj argument bezpośredni
0104	02	operand rozkazu ADI, dodawana liczba 02H
0105	C6	kod rozkazu μP — ADI — dodaj argument bezpośredni
0106	03	operand rozkazu ADI, dodawana liczba 03H
0107	32	kod rozkazu μP — STA — pamiętaj akumulator
0108	00	młodszy bajt adresu, pod którym ma zostać zapamiętany akumulator
0109	02	starszy bajt adresu, pod którym ma zostać zapamiętany akumulator

W tablicy 16.3 przedstawiono, jakie będą zawartości poszczególnych komórek pamięci w wyniku przetłumaczenia programu źródłowego w kod maszynowy (język wewnętrzny). Adresy komórek pamięci oraz ich zawartość jest podana w zapisie szesnastkowym. Kody operacji μP 8080 w zapisie szesnastkowym podano w Dodatku 4.

W komórce o adresie 0100H jest zapisany kod operacji mnożenia logicznego zawartości akumulatora przez 0 (należy pamiętać, że operacje logiczne są wyko-

nywane na odpowiadających sobie parach bitów — tak więc, mnożenie przez 0 (dokładniej **00000000**) oznacza wymnożenie każdego bitu akumulatora przez **0** — w efekcie jego wyzerowanie). W komórce 0101H jest pamiętany kod operacji dodawania do akumulatora **argumentu bezpośredniego**. Przymiotnik „bezpośredni" informuje, że wartość operandu (argumentu) jest zawarta bezpośrednio w rozkazie. Jak wiemy, wszelkie przetwarzane dane są zapisywane w pamięci. Pobranie takiej informacji z pamięci (czy zapisanie do niej) wymaga określenia adresu komórki pamięci, co nazywamy krótko **adresowaniem pamięci**. Z tablicy 16.3 wynika, że operand bezpośredni został ulokowany w komórce pamięci o adresie 0102H. W komórce 0101H jest kod tej operacji, której argumentem jest operand bezpośredni z komórki 0102H. Po pobraniu kodu tej operacji licznik rozkazów jest ustawiany na wartości 0102H, czyli ***natychmiast*** wskazuje adres operandu. Taki sposób adresowania pamięci danych został więc nazwany **adresowaniem natychmiastowym**.

Zawartości komórek pamięci do adresu 0106H chyba nie wymagają komentarza.

Natomiast należy zwrócić uwagę, że instrukcja „pamiętaj zawartość akumulatora w komórce o adresie 0200H" (STA 200H) zajmuje w pamięci trzy kolejne komórki. W pierwszej jest zapisany kod rozkazu (kod operacji STA), w drugiej młodszy bajt, a w trzeciej starszy bajt adresu komórki pamięci, w której ma zostać zapisany wynik (w takiej kolejności są zapisywane wszystkie dane dwubajtowe). Zatem adres komórki pamięci znajduje się ***bezpośrednio*** za komórką z rozkazem (jej kodem). Taki sposób adresowania pamięci nazywamy więc **adresowaniem bezpośrednim**.

Z powyższego przykładu widać, że instrukcje asemblera μP 8080 mogą być 1-, 2- i 3-bajtowe (zajmować w pamięci 1, 2 lub 3 komórki). W tym miejscu należy więc zmodyfikować (podany wcześniej) opis cyklu rozkazowego. Dokładnie tak, jak to opisano wcześniej, jest realizowany rozkaz zerowania akumulatora. To znaczy: licznik rozkazów PC ma stan 0100H, pobrany jest rozkaz, zwiększony stan licznika rozkazów o 1, czyli ustawiony na adresie kolejnej operacji PC = 0101H. Po wykonaniu operacji jest wykonywany kolejny cykl rozkazowy.

W przypadku jednak wykonania operacji zapisanej w komórce 0101H cykl ten wygląda już nieco inaczej. Zostaje pobrany kod operacji spod adresu PC = 0101H i układ sterujący zwiększa licznik rozkazów o 1, PC = 0102H, ale nie jest to adres kolejnego rozkazu. Kod pobranego rozkazu jest dekodowany w układzie sterowania i wykonywany. Jego wykonanie wymaga pobrania zawartości kolejnej komórki pamięci. Operacja ta jest powiązana ze zwiększeniem licznika rozkazów o 1 i teraz licznik jest ustawiony już na adresie kolejnego rozkazu. Tak więc każde odwołanie do pamięci programu (a nie tylko pobranie kodu operacji) zwiększa licznik rozkazów o 1.

Zwróćmy uwagę, że przy adresowaniu natychmiastowym nie występuje podział na pamięć programu i pamięć danych. Ma to bardzo niekorzystne skutki praktyczne. Chcąc bowiem zmienić dane (sumowane liczby), musimy od nowa zapisać nasz program lub zmienić zawartość ***konkretnych*** komórek pamięci, których ad-

resy są dla nas nieznane, gdyż rzadko analizuje się tak dokładnie mapę pamięci programu, jak to zrobiliśmy w niniejszym przykładzie. Zwykle program zajmuje kilka lub kilkanaście tysięcy komórek i operuje na wielu danych, więc analiza obszaru pamięci pod kątem adresów komórek, w których zostały zapisane dane, byłaby zajęciem niezwykle żmudnym. Dlatego ten typ adresowania jest stosowany wyłącznie wówczas, gdy dane są stałe. Zaletą jego jest szybkość pobierania argumentu operacji, który znajduje się w sąsiedniej komórce pamięci (za kodem rozkazu) i której adres wskazuje licznik PC w chwili pobrania kodu rozkazu.

W przykładzie 16.2 wykażemy, że nasz program może zostać zapisany w taki sposób, aby nie miał tej wady.

Przykład 16.2

```
            ;SUMOWANIE TRZECH LICZB Z KOMÓREK PAMIĘCI O ADRE-
            ;SACH 201H, 202H, 203H
ORG 100H    ;POCZĄTEK OBSZARU PAMIĘCI PROGRAMU
XRA A       ;ZEROWANIE AKUMULATORA
LXI H,201H  ;WPISANIE DO REJESTRU HL ADRESU PIERWSZEJ Z SUMO-
            ;WANYCH LICZB
ADD M       ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
            ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
            ;PRZEZ ZAWARTOŚĆ REJESTRU HL — M(HL)
INX H       ;ZWIĘKSZENIE ZAWARTOŚCI REJESTRU HL O 1 (AKTUAL-
            ;NIE WYNIESIE ONA 202H)
ADD M       ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
            ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
            ;PRZEZ ZAWARTOŚĆ REJESTRU HL — M(HL) = 202H
INX H       ;ZWIĘKSZENIE ZAWARTOŚCI REJESTRU HL O 1
            ;(AKTUALNIE WYNIESIE ONA 203H)
ADD M       ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
            ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
            ;PRZEZ ZAWARTOŚĆ REJESTRU HL — M(HL) = 203H
STA 200H    ;PRZESŁANIE ZAWARTOŚCI AKUMULATORA DO KOMÓRKI
            ;PAMIĘCI O ADRESIE 200H
ORG 201H    ;POCZĄTEK OBSZARU PAMIĘCI DANYCH
DB 1,2,3
END
```

W programie z przykładu 16.2 pojawiła się nowa pseudoinstrukcja sterująca procesem translacji — DB (ang. *Data Byte*), która definiuje bajt. Informuje ona translator, że na dane (wypisane po niej) ma rezerwować po jednym bajcie pamięci. Oznacza to, że możemy tu używać wyłącznie liczb od 0H do FFH (od 0 do 255 w zapisie dziesiętnym) lub innych danych jednobajtowych, jak np. kodów ASCII. Użycie w tym miejscu pseudoinstrukcji DW (ang. *Data Word)* rezerwuje po dwie komórki pamięci na kolejne dane. Możliwe byłoby używanie liczb z zakresu od 0 do 65535 (dziesiętnie), ale musiałby zostać zmieniony także program.

W programie tym został użyty inny rozkaz zerujący akumulator (XRA A), aby pokazać inną możliwość osiągnięcia zerowego stanu akumulatora. Po asemblacji program ten zostanie wpisany w następujące komórki pamięci:

		OBSZAR PAMIĘCI PROGRAMU
0100	AF	kod rozkazu XRA A
0101	210102	kod rozkazu LXI, młodszy bajt, starszy bajt adresu wpisywanego do rejestru HL
0104	86	kod rozkazu ADD M
0105	23	kod rozkazu INX H
0106	86	kod rozkazu ADD M
0107	23	kod rozkazu INX H
0108	86	kod rozkazu ADD M
0109	320002	kod rozkazu STA, młodszy bajt, starszy bajt adresu
		OBSZAR PAMIĘCI DANYCH I WYNIKU
0200		komórka zarezerwowana na wynik
0201	01	pierwsza z sumowanych liczb
0202	02	druga z sumowanych liczb
0203	03	trzecia z sumowanych liczb

Dla skrócenia zapisu każdą instrukcję umieszczono w jednym wierszu. Zapis 0101 210102 należy więc rozumieć, że w komórce 0101 jest kod rozkazu (21), w komórce 0102 jest młodszy bajt adresu danych (01), w komórce 0103 jest starszy bajt adresu danych (02). Należy pamiętać, że operujemy zapisem szesnastkowym. Każda cyfra tego zapisu to cztery bity, a każda komórka pamięci (słowo μP 8080) to 8 bitów, czyli w zapisie szesnastkowym jedna komórka pamięci to dwie cyfry szesnastkowe.

Zawartość komórki o adresie 0200H jest nieokreślona. Dopiero wykonanie programu spowoduje, że zostanie tam wpisana liczba 06H (wynik sumowania liczb 1, 2 i 3). Aby uruchomić wykonanie programu, należy ustawić licznik rozkazów μP na wartość 0100H (patrz rozkazy skoku).

W tej wersji programu do zawartości akumulatora dodajemy dane (ADD M), wskazując za pomocą rejestru HL adres komórki pamięci, w której ta dana się znajduje.

Ten sposób adresowania, za pomocą zawartości rejestru HL, jest nazywany **adresowaniem zawartością wskaźnika danych** lub **adresowaniem rejestrowym pośrednim**. Rejestr HL pełni tu rolę **wskaźnika danych** lub — przez analogię do licznika rozkazów — **licznika danych.** Cała instrukcja dodania do akumulatora (ADD M) zajmuje teraz tylko jedną komórkę pamięci.

Zwróćmy uwagę, że aby teraz zmienić dane (sumowane liczby), należy jedynie wpisać w określony obszar pamięci (komórki 201H, 202H i 203H) nowe wartości. Obszar ten jest określony przez programistę, a nie translator, jak to miało miejsce przy adresowaniu natychmiastowym. Jedyną wadą tego rozwiązania jest konieczność pamiętania, jaki obszar jest zarezerwowany na dane i pod jakie adresy trzeba je wpisywać. Translator asemblera umożliwia tu symboliczny zapis tych adresów, co znacznie ułatwia korzystanie z tak opracowanych programów. Sposób ten przedstawiono w przykładzie 16.3, który jest drobną modyfikacją zapisu z przykładu 16.2.

Przykład 16.3

```
                ;SUMOWANIE TRZECH LICZB, PIERWSZA Z NICH UMIESZ-
                ;CZONA W KOMÓRCE PAMIĘCI O ADRESIE SYMBOLICZNYM
                ;"DANE", WYNIK SUMOWANIA W KOMÓRCE O ADRESIE SYM-
                ;BOLICZNYM „WYNIK"
ORG 100H        ;POCZĄTEK OBSZARU PAMIĘCI PROGRAMU
MVI A,0H        ;ZEROWANIE AKUMULATORA
LXI H,DANE      ;WPISANIE DO REJESTRU HL ADRESU, KTÓREMU ZOSTAŁA
                ;PRZYPORZĄDKOWANA SYMBOLICZNA NAZWA ;"DANE"
ADD M           ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
                ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
                ;PRZEZ ZAWARTOŚĆ REJESTRU HL - M(HL)
INX H           ;ZWIĘKSZENIE ZAWARTOŚCI REJESTRU HL O 1
ADD M           ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
                ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
                ;PRZEZ ZAWARTOŚĆ REJESTRU HL
INX H           ;ZWIĘKSZENIE ZAWARTOŚCI REJESTRU HL O 1
ADD M           ;DODANIE DO ZAWARTOŚCI AKUMULATORA LICZBY ZAPISA-
                ;NEJ W KOMÓRCE PAMIĘCI O ADRESIE WSKAZYWANYM
                ;PRZEZ ZAWARTOŚĆ REJESTRU HL
STA WYNIK       ;PRZESŁANIE ZAWARTOŚCI AKUMULATORA DO KOMÓRKI
                ;PAMIĘCI O ADRESIE SYMBOLICZNYM „WYNIK"

ORG 200H        ;POCZĄTEK OBSZARU DANYCH I WYNIKU
WYNIK: DS 1
DANE: DB 1,2,3
END
```

W przykładzie 16.3 pokazano jeszcze inny sposób zerowania akumulatora. Wykorzystano do tego celu rozkaz przesłania (MVI) do akumulatora liczby 0. Sposób ten jest jednak mniej korzystny od poprzednich, ponieważ potrzebuje dwóch komórek pamięci (patrz niżej) do zapisania rozkazu MVI.

W przykładzie tym wystąpiła nowa, kolejna pseudoinstrukcja sterująca translacją. Jest to dyrektywa DS 1, która poleca zarezerwować jedną komórkę pamięci (w tym przypadku z przeznaczeniem na wynik sumowania trzech liczb).

Należy zwrócić uwagę na sposób używania etykiet w asemblerze i ich wykorzystanie w tym przykładzie do symbolicznego określenia adresów komórek przeznaczonych na dane i wyniki. Kolejne zastosowania etykiet zostaną pokazane w następnych przykładach.

Po wpisaniu tego programu do pamięci (po uprzednim przetłumaczeniu na język maszynowy) zawartość poszczególnych komórek pamięci programu i pamięci danych przedstawiać się będzie następująco:

		OBSZAR PAMIĘCI PROGRAMU
0100	3E00	kod rozkazu MVI, parametr tego rozkazu
0102	210102	kod rozkazu LXI, młodszy bajt, starszy bajt adresu
0105	86	kod rozkazu ADD M
0106	23	kod rozkazu INX H
0107	86	kod rozkazu ADD M
0108	23	kod rozkazu INX H
0109	86	kod rozkazu ADD M
010A	320002	kod rozkazu STA, młodszy bajt, starszy bajt adresu.
		OBSZAR PAMIĘCI DANYCH I WYNIKU
0200	—	wartość przypadkowa, wynik pojawi się tu dopiero po uruchomieniu i wykonaniu programu
0201	01	pierwsza
0202	02	druga
0203	03	trzecia z sumowanych liczb

Zawartość komórek pamięci programu jest więc identyczna jak w poprzednim przykładzie z tym, że nastąpiło przesunięcie adresów o 1 ze względu na użycie w tym przypadku dwubajtowego rozkazu zerowania akumulatora. Po uruchomieniu i wykonaniu tego programu zawartość komórek pamięci o adresach 0201H do 0203H nie ulegnie zmianie, natomiast zawartość komórki o adresie 0200H (symbolicznym „WYNIK") wyniesie 06H.

Symboliczne określenie adresów obszaru pamięci danych i wyniku pozwala łatwo zmienić ten obszar w zależności od aktualnych potrzeb, bez dodatkowej ingerencji w sam program. Wystarczy bowiem w przykładzie 16.3 zmienić fragment od pseudoinstrukcji ORG na następujący zapis (a nie jest to program, lecz dyrektywy asemblera — asemblera tłumacza):

```
ORG 400H    ;POCZĄTEK OBSZARU DANYCH I WYNIKU
WYNIK: DS 1
DANE : DB 1,2,3
```

aby został zmieniony obszar pamięci danych.

Łatwo zauważyć, że w powyższych programach występuje powtarzająca się sekwencja instrukcji asemblera: ADD M, INX H. Można je zapisać w postaci podprogramu (raz) i wykorzystywać wielokrotnie. W naszym przykładzie nie jest to posunięcie racjonalne (wydłuża się czas wykonywania programu), ale gdyby sekwencja taka powtarzała się np. kilkadziesiąt razy (gdybyśmy sumowali kilkadziesiąt liczb), to chociażby ze względu na oszczędność pamięci należy zapisać ją jako podprogram. Zapiszmy zatem program z przykładu 16.3, definiując taki podprogram, aby poznać zasady konstrukcji programu z użyciem podprogramów.

Przykład 16.4

```
                ;SUMOWANIE TRZECH LICZB, PROGRAM WYKORZY-
                ;STUJĄCY MOŻLIWOŚĆ TWORZENIA PODPROGRAMÓW
ORG 100H        ;POCZĄTEK OBSZARU PAMIĘCI PROGRAMU
XRA A           ;ZEROWANIE AKUMULATORA
LXI H,DANE      ;WPISANIE DO REJESTRU HL ADRESU, KTÓREMU
                ;ZOSTAŁA PRZYPORZĄDKOWANA SYMBOLICZNA
                ;NAZWA "DANE"
```

```
CALL PODPROGRAM    ;WYWOŁANIE PODPROGRAMU O NAZWIE „PODPRO-
                   ;GRAM"
CALL PODPROGRAM    ;WYWOŁANIE PODPROGRAMU O NAZWIE „PODPRO-
                   ;GRAM"
CALL PODPROGRAM    ;WYWOŁANIE PODPROGRAMU O NAZWIE „PODPRO-
                   ;GRAM"
STA WYNIK          ;PRZESŁANIE ZAWARTOŚCI AKUMULATORA DO
                   ;KOMÓRKI PAMIĘCI O ADRESIE SYMBOLICZNYM
                   ;„WYNIK"
JMP KONIEC         ;SKOK DO INSTRUKCJI OPATRZONEJ ETYKIETĄ KO-
                   ;NIEC
ORG 150H           ;POCZĄTEK PAMIĘCI PODPROGRAMU
PODPROGRAM: ADD M  ;DODANIE DO ZAWARTOŚCI AKUMULATORA ZAWAR-
                   ;TOŚCI M(HL)
            INX H  ;ZWIĘKSZENIE STANU REJESTRU HL O 1
            RET    ;POWRÓT Z PODPROGRAMU

ORG 200H           ;POCZĄTEK OBSZARU DANYCH I WYNIKU
WYNIK: DS 1
DANE : DB 1,2,3
KONIEC: END
```

Po wpisaniu tego programu do pamięci (po uprzednim przetłumaczeniu na język maszynowy) zawartość poszczególnych komórek pamięci będzie się przedstawiać następująco:

0100	AF	kod operacji XRA A
0101	210102	kod operacji LXI, młodszy, starszy bajt adresu wpisywanego w rejestr HL
0104	CD5001	kod rozkazu CALL, młodszy, starszy bajt adresu startowego podprogramu
0107	CD5001	kod rozkazu CALL, młodszy, starszy bajt adresu startowego podprogramu
010A	CD5001	kod rozkazu CALL, młodszy, starszy bajt adresu startowego podprogramu
010D	320002	kod rozkazu STA, młodszy, starszy bajt adresu
0110	C3 0402	kod rozkazu JMP, młodszy, starszy bajt adresu końca programu
0150	86	kod operacji ADD M
0151	23	kod rozkazu INX H
0152	C9	kod rozkazu RET
0200		komórka zarezerwowana na wynik
0201	010203	sumowane liczby (pierwsza, druga i trzecia)
0204		adres końca programu

Podprogram został umieszczony w kolejnych trzech komórkach pamięci, począwszy od adresu 150H. Takie rozmieszczenie spowodowała pseudoinstrukcja ORG 150H. Pominięcie jej spowodowałoby umieszczenie podprogramu w komórkach 113, 114 i 115, czyli bezpośrednio następujących po ostatniej komórce zajętej przez program.

Liczba podprogramów może być dowolna, a także dopuszczalne jest odwoływanie się z jednego podprogramu do innego, a nawet wywoływanie samego siebie.

W niniejszym przykładzie odwoływaliśmy się trzykrotnie do podprogramu. Gdyby ilość sumowanych liczb była większa, wówczas także liczba tych odwołań musiałaby zostać zwiększona. Niekiedy liczba takich odwołań nie jest z góry znana, albo jest daną pobieraną przez program. Wówczas odwołania do podprogramu realizuje się jako cykliczne, aż do spełnienia pewnego warunku. Konstrukcję taką określa się mianem **pętli**. Konstrukcja taka jest bardzo użytecznym narzędziem programisty. Prześledźmy ją na przykładzie sumowania, liczb przy czym pierwsza pobierana przez program dana określa liczbę sumowanych liczb.

Przykład 16.5

```
                        ;SUMOWANIE N LICZB, LICZBA N JEST
                        ;UMIESZCZONA W PIERWSZEJ KOMÓRCE
                        ;OBSZARU DANYCH
            MVI C,0     ;ZEROWANIE REJESTRU C
            LXI H,DANE  ;WPISANIE DO REJESTRU HL ADRESU, KTÓREMU
                        ;ZOSTAŁA PRZYPORZĄDKOWANA SYMBOLICZNA
                        ;NAZWA "DANE"
            MOV B,M     ;WPISANIE DO REJESTRU B LICZBY SUMOWANYCH
                        ;LICZB (w tym przykładzie będzie to 5)
            INX H       ;ZWIĘKSZENIE O 1 ZAWARTOŚCI REJESTRU HL
JESZCZE:    CALL PODPROG ;WYWOŁANIE PODPROGRAMU O NAZWIE
                        ;"PODPROG"
            DCR B       ;ZMNIEJSZENIE ZAWARTOŚCI REJESTRU B
                        ;(TERAZ JEST 4 i po każdym powrocie
                        ;z podprogramu zawartość ta
                        ;zostanie zmniejszona o 1)
            JNZ JESZCZE ;JEŻELI (ZAWARTOŚĆ B) NIE JEST RÓWNA 0,
                        ;TO IDŹ DO INSTRUKCJI O ETYKIECIE
                        ;JESZCZE
            MOV A,C     ;B=0 CO OZNACZA ZSUMOWANIE WSZYSTKICH
                        ;LICZB, WIĘC WYNIK Z C PRZEPISUJEMY DO A
            STA WYNIK   ;ZAPAMIĘTANIE WYNIKU W KOMÓRCE WYNIK
            JMP KONIEC  ;SKOK DO INSTRUKCJI OPATRZONEJ ETYKIETĄ
                        ;KONIEC
PODPROG:    XRA A       ;ZERUJ AKUMULATOR
            ADD C       ;DODAJ DO AKUMULATORA ZAWARTOŚĆ
                        ;REJESTRU C
            ADD M       ;DODAJ DO AKUMULATORA ZAWARTOŚĆ KOMÓRKI
                        ;PAMIĘCI O ADRESIE PRZECHOWYWANYM
                        ;W REJESTRZE HL
            INX H       ;ZWIĘKSZ O 1 ZAWARTOŚĆ REJESTRU HL
            MOV C,A     ;PRZEPISZ DO C ZAWARTOŚĆ AKUMULATORA
            RET         ;POWRÓT Z PODPROGRAMU
ORG 200H                ;POCZĄTEK OBSZARU DANYCH I WYNIKU
WYNIK: DS 1
DANE: DB 5, 1, 2, 3, 4, 5
KONIEC: END
```

Na zakończenie jeszcze raz przyjrzyjmy się wykorzystywanym w powyższych przykładach metodom adresowania pamięci. Przyjmijmy, że naszym zadaniem jest umieszczenie w akumulatorze danej, która w zapisie szesnastkowym ma postać F1H. Możemy to osiągnąć różnymi sposobami, wykorzystując do tego celu np. następujące operacje μP:

```
1. MVI A, F1H  ;prześlij do A (akumulatora) daną F1H, lub
2. LDA 201H    ;prześlij do A daną zapisaną w komórce pamięci
               ;o adresie 201H, lub
3. MOV A,M     ;prześlij do A daną, której adres jest określony zawartością
               ;rejestru HL
4. LDAX B      ;prześlij do A daną, której adres jest określony zawartością
               ;rejestru BC.
```

Obraz pamięci zawierający te instrukcje (po przetłumaczeniu) będzie w poszczególnych przypadkach następujący:

ad. 1. Dla rozkazu MVI będzie: 3E F1, czyli instrukcja ta zajmie dwie komórki pamięci (w pierwszej kod operacji, w drugiej argument). Ten sposób lokalizacji danych nazywamy **adresowaniem natychmiastowym.**

ad. 2. Dla rozkazu LDA 201H będzie: 3A 01 02, czyli instrukcja ta zajmie trzy komórki pamięci (w pierwszej kod operacji, w drugiej młodszy bajt, w trzeciej starszy bajt argumentu). Ten sposób lokalizacji danych nazywamy **adresowaniem bezpośrednim.**

ad. 3. Dla rozkazu MOV A,M będzie: 7E, czyli instrukcja ta zajmie jedną komórkę pamięci (kod operacji). Adres argumentu operacji wskazuje zawartość rejestru HL. Ten sposób lokalizacji danych jest nazywany **adresowaniem zawartością wskaźnika danych (adresowaniem rejestrowym pośrednim).**

ad. 4. Jest to identyczny sposób adresowania jak w p. 3., z tym, że adres jest przechowywany w rejestrze BC (kod tej operacji 0A).

Poznane metody adresowania pamięci (adresowanie natychmiastowe, pośrednie, adresowanie zawartością wskaźnika danych) nie wyczerpują wszystkich metod adresowania pamięci. Metody te nie pozwalają na proste relokowanie (przesuwanie) obszarów pamięci. Bardzo często przetwarzana informacja jest zebrana w postaci tablic. Dogodną formą adresowania tak zorganizowanych danych jest połączenie adresowania bezpośredniego z adresowaniem zawartością rejestru. Polega ono na wyznaczeniu adresu jako sumy adresu bezpośredniego *a,* (umieszczonego za kodem operacji), i zawartości pewnego rejestru μP, zwanego w takim przypadku **rejestrem indeksowym**. Relokacja tak adresowanego obszaru danych może być łatwo osiągnięta poprzez zmianę zawartości rejestru indeksowego. Ponieważ taki sposób adresowania jest bardzo przydatny, przeto nowsze μP (np. μP Z80, który jest ulepszoną wersją μP 8080) ma odrębne rejestry indeksowe. Zadaniem μP jest więc nie tylko wykonywanie operacji na danych, lecz także obliczanie adresów.

Pytania i zadania

1. Narysuj schemat funkcjonalny komputera i omów rolę poszczególnych jego bloków. Określ kierunki przepływu informacji pomiędzy blokami.
2. Co to jest procesor? Czym różni się on od mikroprocesora.
3. Narysuj uproszczony schemat funkcjonalny μP i omów rolę poszczególnych jego bloków.
4. Narysuj symbol magistrali dwukierunkowej. Wyjaśnij, co nazywamy magistralą oraz w jaki sposób współpracuje z danym blokiem magistrala dwukierunkowa.
5. Wymień i omów zasadnicze różnice pomiędzy techniką cyfrową układową a techniką cyfrową programowaną.
6. Ilu bitowy jest μP MCY7880? Jakie są tego konsekwencje? Ilu bitowe jest adresowanie pamięci? Jaki obszar pamięci można zaadresować tą liczbą bitów?
7. Jaką rolę pełni w μP licznik rozkazów? Jak przebiega cykl pracy μP?
8. Co to jest stos (dokładniej pamięć stosu) i jaka jest rola wskaźnika stosu? Jaka jest praktyczna przydatność tych elementów?
9. Co to jest i do czego służy rejestr flagowy?
10. Zapisz program w asemblerze μP Intel 8080, który będzie mnożył przez 4 daną liczbę jednobajtową.
11. Zapisz program w asemblerze μP Intel 8080, który będzie dzielił przez 4 daną liczbę jednobajtową.
12. Zapisz program w asemblerze μP Intel 8080, który będzie mnożył przez siebie dwie liczby jednobajtowe.
13. Zapisz program w asemblerze μP Intel 8080, który będzie przepisywał obszar pamięci od adresu 200H do 20FH w obszar od adresu 400H do 40FH.

Dodatek 1.

Tablica kodu ASCII

ZD	ZH	Znak	ZD	ZH	Znak	ZD	ZH	Znak	ZD	ZH	Znak
32	20	spacja	61	3D	=	90	5A	Z	119	77	w
33	21	!	62	3E	>	91	5B	[	120	78	x
34	22	"	63	3F	?	92	5C	\	121	79	y
35	23	#	64	40	@	93	5D	]	122	7A	z
36	24	$	65	41	A	94	5E	'	123	7B	{
37	25	%	66	42	B	95	5F	_	124	7C	|
38	26	&	67	43	C	96	60	`	125	7D	}
39	27	'	68	44	D	97	61	a	126	7E	~
40	28	(	69	45	E	98	62	b	127	7F	<delete>
41	29	)	70	46	F	99	63	c			
42	2A	*	71	47	G	100	64	d			
43	2B	+	72	48	H	101	65	e			
44	2C	,	73	49	I	102	66	f			
45	2D	-	74	4A	J	103	67	g			
46	2E	.	75	4B	K	104	68	h			
47	2F	/	76	4C	L	105	69	i			
48	30	0	77	4D	M	106	6A	j			
49	31	1	78	4E	N	107	6B	k			
50	32	2	79	4F	O	108	6C	l			
51	33	3	80	50	P	109	6D	m			
52	34	4	81	51	Q	110	6E	n			
53	35	5	82	52	R	111	6F	o			
54	36	6	83	53	S	112	70	p			
55	37	7	84	54	T	113	71	q			
56	38	8	85	55	U	114	72	r			
57	39	9	86	56	V	115	73	s			
58	3A	:	87	57	W	116	74	t			
59	3B	;	88	58	X	117	75	u			
60	3C	<	89	59	Y	118	76	v			

ZD — zapis dziesiętny, ZH — zapis szesnastkowy. Kody 0...31 (dziesiętnie) odpowiadają znakom sterującym nie mającym odpowiedników graficznych (nie mogą być wydrukowane). Stosowane są do realizacji transmisji znaków. Ósmy bit (nie wykorzystywany przez kod ASCII) jest zwykle bitem parzystości.

Dodatek 3.

Nazwy angielskie operacji mP 8080

MOV — MOVe
MVI — MoVe Immediate
STAX — STore Accumulator
LDAX — LoaD Accumulator
STA — STore Accumulator
LDA — LoaD Accumulator
ADD — ADD
ADI — ADd Immediate
ADC — ADd with Carry
ACI — Add with Carry Immediate
SUB — SUBstract
SUI — SUbstract Immediate
SBB — SuBstract with Borrow
SBI — Substract with Borrow Immediate
ANA — ANd Accumulator
ANI — ANd Immediate
ORA — OR Accumulator
ORI — OR Immediate
XRA — eXclusive oR Accumulator
XRI — eXclusive oR Immediate
CMP — CoMPare
CMI — CoMpare Immediate
INR — INcRement
DCR — DeCRement
CMA — CoMplement Accumulator
DAA — Decimal Adjust A
RLC — Rotate Left without Cy
RRC — Rotate Right without Cy
RAL — Rotate Accumulator Left
RAR — Rotate Accumulator Right
STC — SeT Carry
CMC — CoMplement Carry
LXI — Load Immediate
INX — INcrement
DCX — DeCrement
DAD — Double ADd
SHLD — Store HL Direct
LHLD — Load HL Direct
XCHG — eXCHanGe
JMP — JuMP
CALL — CALL
RST — ReSTart
RET — RETurn
PUSH — PUSH register pair
POP — POP register pair
XTHL — eXchange Top of stock with HL
IN — INput
PUT — outPUT
EI — Enable Interrupts
DI — Disable Interrupts
NOP — No OPeration
HLT — HaLT

Dodatek 4.

Kody maszynowe mikroprocesora Intel 8080

DAD B	09	CMP E	BB	LDAX D	1A	MOV M,C	71	RNC	D0
DAD D	19	CMP H	BC	LDA nn	3A	MOV M,D	72	RNZ	C0
DAD H	29	CMP L	BD			MOV M,E	73	RP	F0
DAD SP	39	CMP M	BE	MOV B,B	40	MOV M,H	74	RPE	E8
		CMP A	BF	MOV B,C	41	MOV M,L	75	RZ	C8
ANA B	A0	CPI n	FE	MOV B,D	42	MOV M,A	77	RAL	17
ANA C	A1			MOV B,E	43			RLC	07
ANA D	A2	CMA	2F	MOV B,H	44	MOV A,B	78	RAR	1F
ANA E	A3	DAA	27	MOV B,L	45	MOV A,C	79	RRC	0F
ANA H	A4	DCR M	35	MOV B,M	46	MOV A,D	7A	RP0	E0
ANA L	A5	DCR A	3D	MOV B,A	47	MOV A,E	7B	RST 0	C7
ANA M	A6	DCR B	05			MOV A,H	7C	RST 1	CF
ANA A	A7	DCR C	0D	MOV C,B	48	MOV A,L	7D	RST 2	D7
ANI n	E6	DCR D	15	MOV C,C	49	MOV A,M	7E	RST 3	DF
		DCR E	1D	MOV C,D	4A	MOV A,A	7F	RST 4	E7
ADD B	80	DCR H	25	MOV C,E	4B			RST	5 EF
ADD C	81	DCR L	2D	MOV C,H	4C	MVI B,n	06	RST 6	F7
ADD D	82	DCX B	0B	MOV C,L	4D	MVI C,n	0E	RST 7	FF
ADD E	83	DCX D	1B	MOV C,M	4E	MVI D,n	16		
ADD H	84	MOV E,D	5A	MOV C,A	4F	MVI E,n	1E	SUB B	90
ADD L	85	DCX H	2B			MVI H,n	26	SUB C	91
ADD M	86	DCX SP	3B	MOV D,B	50	MVI L,n	2E	SUB D	92
ADD A	87	DI	F3	MOV D,C	51	MVI A,n	3E	SUB E	93
ADI n	C6	EI	FB	MOV D,D	52	MVI M,nn	36	SUB H	94
		XTHL	E3	MOV D,E	53			SUB L	95
ADC B	88	XCHG	EB	MOV D,H	54	LXI B,nn	01	SUB A	97
ADC C	89	HLT	76	MOV D,L	55	LXI D,nn	11	SUB M	96
ADC D	8A			MOV D,	56	LXI H,nn	21	SUI nn	D6
ADC E	8B	INR B	04	MOV D,A	57	LXI SP,nn	31		
ADC H	8C	INR C	0C					SBB B	98
ADC L	8D	INR D	14	MOV E,B	58	LHLD nn	42	SBB C	99
ADC M	8E	INR E	1C	MOV E,C	59	SPHL	F9	SBB D	9A
ADC A	8F	INR H	24	OUT nn	D3	NOP	00	SBB E	9B
ACI n	CE	INR L	2C	MOV E,E	5B			SBB H	9C
		INR M	34	MOV E,H	5C	ORA B	B0	SBB L	9D
		INR A	3C	MOV E,L	5D	ORA C	B1	SBB M	9E
AND B	A0	IN n	DB	MOV E,M	5E	ORA D	B2	SBB A	9F
AND C	A1			MOV E,A	5F	ORA E	B3	SBI nn	DE
AND D	A2	INX B	03			ORA H	B4		
AND E	A3	INX D	13	MOV H,B	60			STC	37
AND H	A4	INX H	23	MOV H,C	61	ORA L	B5		
AND L	A5	INX SP	33	MOV H,D	62	ORA M	B6	XRA B	A8
AND nn	E6	PCHL	E9	MOV H,E	63	ORA A	B7	XRA C	A9
		JC nn	DA	MOV H,H	64	ORI n	F6	XRA D	AA
CC nn	DC	JM nn	FA	MOV H,L	65			XRA E	AB
CM nn	FC	JNC nn	D2	MOV H,M	66	POP B	C1	XRA H	AC
CNC nn	D4	JNZ nn	C2	MOV H,A	67	POP D	D1	XRA L	AD
CNZ nn	C4	JP nn	F2			POP H	E1	XRA M	AE
CP nn	F4	JPE nn	EA	MOV L,B	68	POP PSW	F1	XRA A	AF
CPE nn	EC	JPO nn	E2	MOV L,C	69			XRI nn	EE
CPO nn	E4	JZ nn	CA	MOV L,D	6A	PUSH B	C5		
CZ nn	CC	JMP nn	C3	MOV L,E	6B	PUSH D	D5		
CALL nn	CD	STAX B	02	MOV L,H	6C	PUSH H	E5		
CMC	3F	STAX D	12	MOV L,L	6D	PUSH PSW	F5		
				MOV L,M	6E				
CMP B	B8	STA nn	32	MOV L,A	6F	RET	C9		
CMP C	B9	SHLD nn	22			RC	D8		
CMP D	BA	LDAX B	0A	MOV M,B	70	RM	F8		

Literatura

1. Borczyński J., Dumin P., Mliczewski A.: **Podzespoły elektroniczne, półprzewodniki. Poradnik.** Warszawa, WKiŁ 1990.
2. Dras M.: Przetworniki półprzewodnikowe. ***Elektronizacja.*** 1994. Nr 1
3. Gajewski P., Turczyński J.: **Cyfrowe układy scalone CMOS.** Warszawa, WKiŁ 1990.
4. Kalinowski S.: Programowany układ czasowy CMOS MCY74541. ***Radioelektronik.*** 1992. Nr 1. (s. 9-12).
5. Kalisz J.: **Podstawy elektroniki cyfrowej.** Warszawa, WKiŁ 1991.
6. Kręciejewski M.: **Układy cyfrowe.** Wydawnictwo Czasopism i Książek Technicznych NOT SIGMA, 1988.
7. Kruszyński H., Misiurewicz P., Perkowski M., Rydzewski A.: **Zbiór zadań z teorii układów logicznych.** Wydawnictwa Politechniki Warszawskiej 1976.
8. Łakomy M., Zabrodzki J.: **Cyfrowe układy scalone.** Warszawa, PWN 1986.
9. Łuba T., Markowski M.A., Zbierzchowski B.: **Komputerowe projektowanie układów cyfrowych w strukturach PLD**. Warszawa, WKiŁ 1993.
10. Misiurewicz P., Perkowski M.: **Teoria automatów.** Wydawnictwa Politechniki Warszawskiej 1976.
11. Misiurewicz P.: **Podstawy techniki cyfrowej.** Warszawa, WNT 1985.
12. Misiurewicz P.: **Układy automatyki cyfrowej.** Warszawa, WSiP 1987.
13. Misiurewicz P.: **Systemy mikrokomputerowe.** Warszawa, WSiP 1991.
14. Misiurewicz P., Grzybek M.: **Półprzewodnikowe układy logiczne TTL.** Warszawa, WNT 1982.
15. Motorola High-Speed CMOS Logic Data. 1989. (Katalog)
16. Niederliński A.: **Mikroprocesory, mikrokomputery, mikrosystemy.** Warszawa, WSiP 1988.
17. Nuhrman D.: **Elektronika łatwiejsza niż przypuszczasz — technika cyfrowa.** Warszawa, WKiŁ 1986.
18. Orłowski H.: **Komputerowe układy automatyki.** Warszawa, WNT 1987.
19. Philips. Integrated Circuits. Catalogue 1990. (Katalog).

20. Pieńkos J., Turczyński J.: **Układy scalone TTL serii UCY74 i ich zastosowanie.** Warszawa, WKiŁ 1976.
21. Pieńkos J., Turczyński J.: **Układy scalone TTL w systemach cyfrowych.** Warszawa, WKiŁ 1986.
22. Rydzewski A., Sacha K.: **Mikrokomputer — elementy, budowa, działanie.** Wydawnictwa Czasopism i Książek Technicznych NOT SIGMA, 1987.
23. Sacha K.: **Systemy techniki cyfrowej.** Warszawa, WSiP 1988.
24. Sacha K., Rydzewski A.: **Mikroprocesor w pytaniach i odpowiedziach.** Warszawa, WNT 1985.
25. Sasal W.: **Układy scalone serii UCA64/UCY74. Parametry i zastosowania.** Warszawa, WKiŁ 1985.
26. Sasal W.: **Układy scalone serii UCY74LS i UCY74S. Parametry i zastosowania.** Warszawa, WKiŁ 1993.
27. Stolarski E., Zając Jerzy M.**: Pamięci półprzewodnikowe.** Warszawa, Wydawnictwo Naukowe Semper 1993.
28. Traczyk W.: **Układy cyfrowe automatyki.** Warszawa, WNT 1976.
29. Turczyński J., Maksymowicz R., Malec B., Ponikiewski J.: **Wybrane układy z techniki cyfrowej.** Warszawa, WKiŁ 1983.

Skorowidz

R

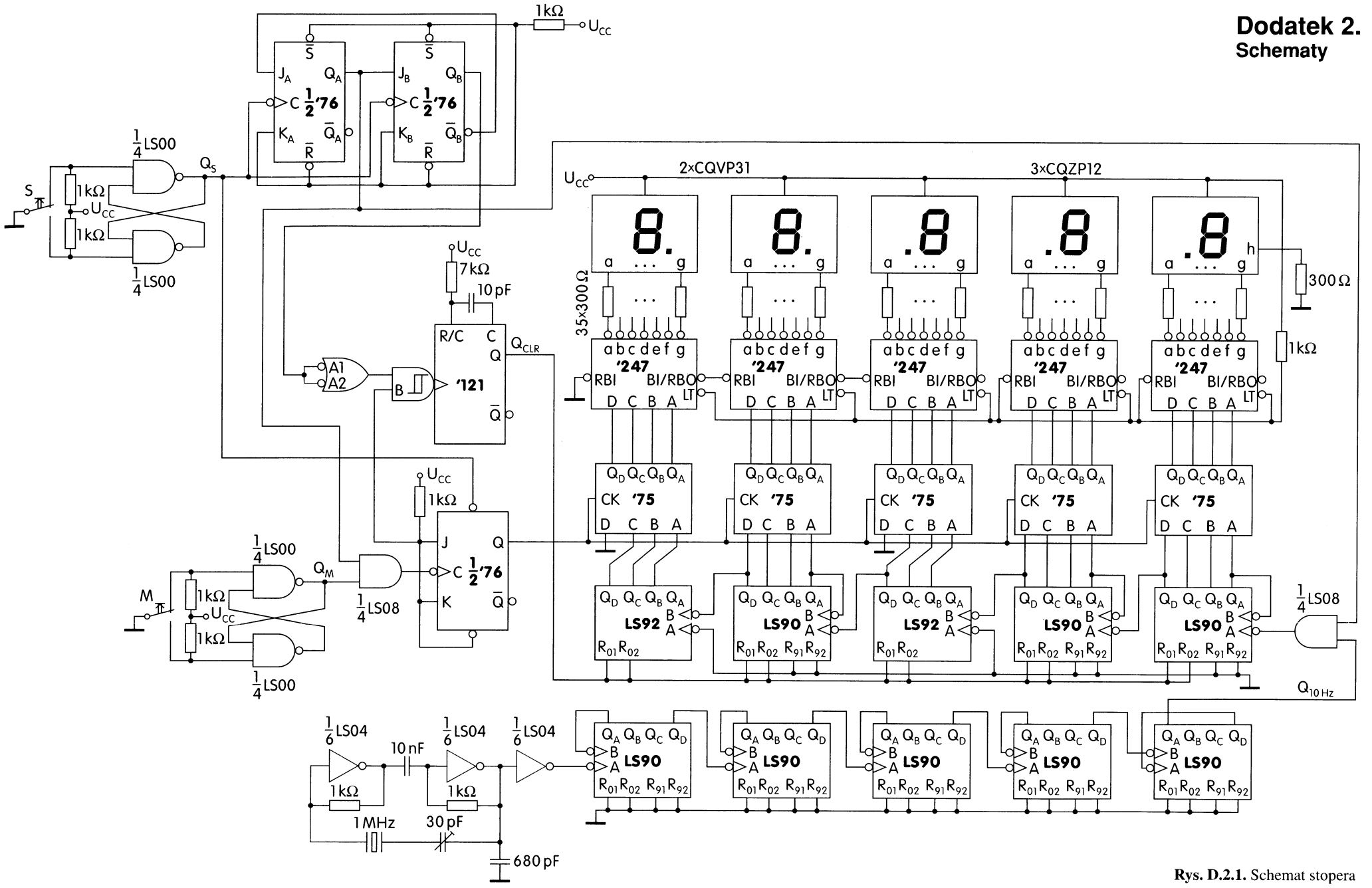
S

T

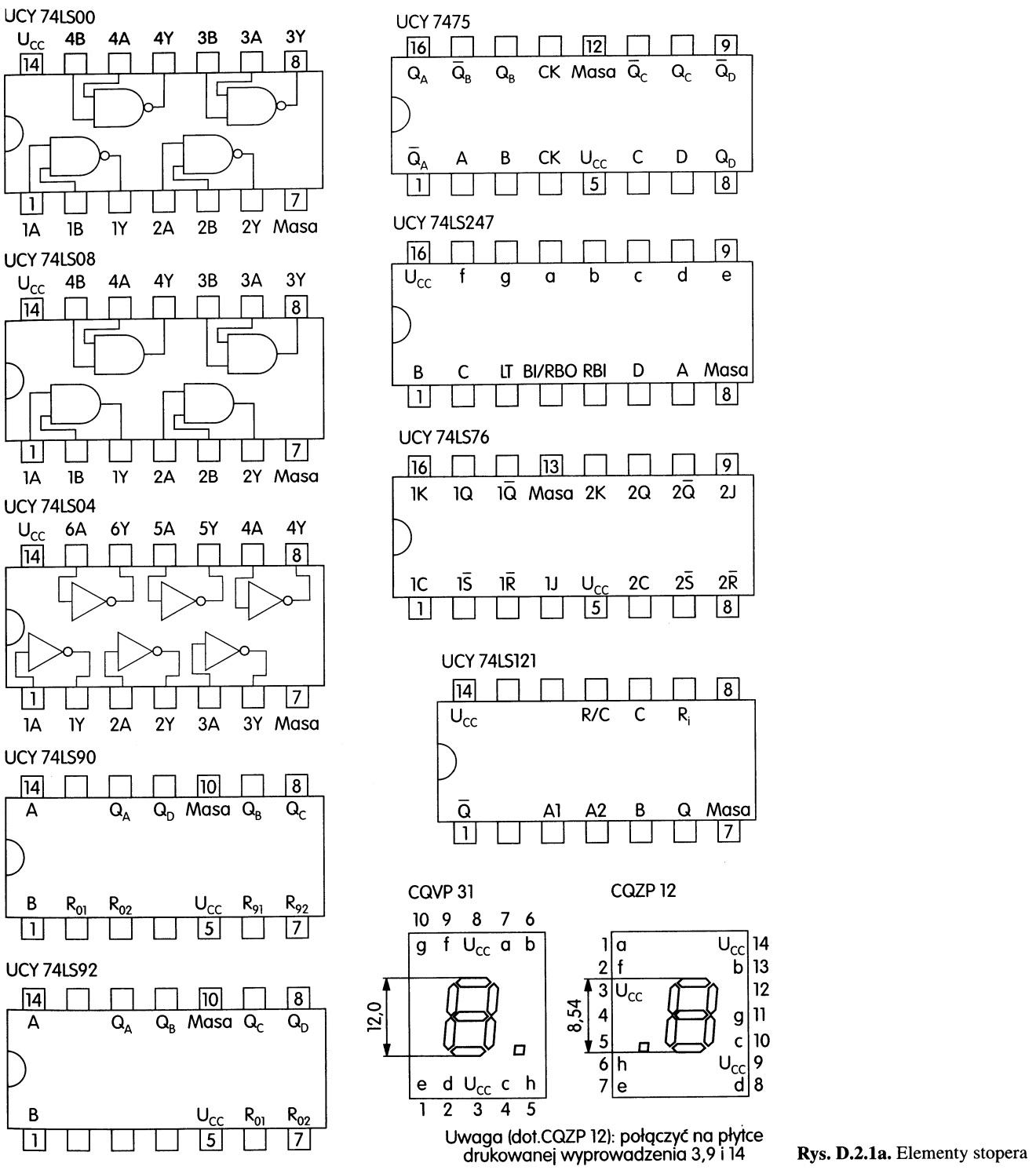
U

W

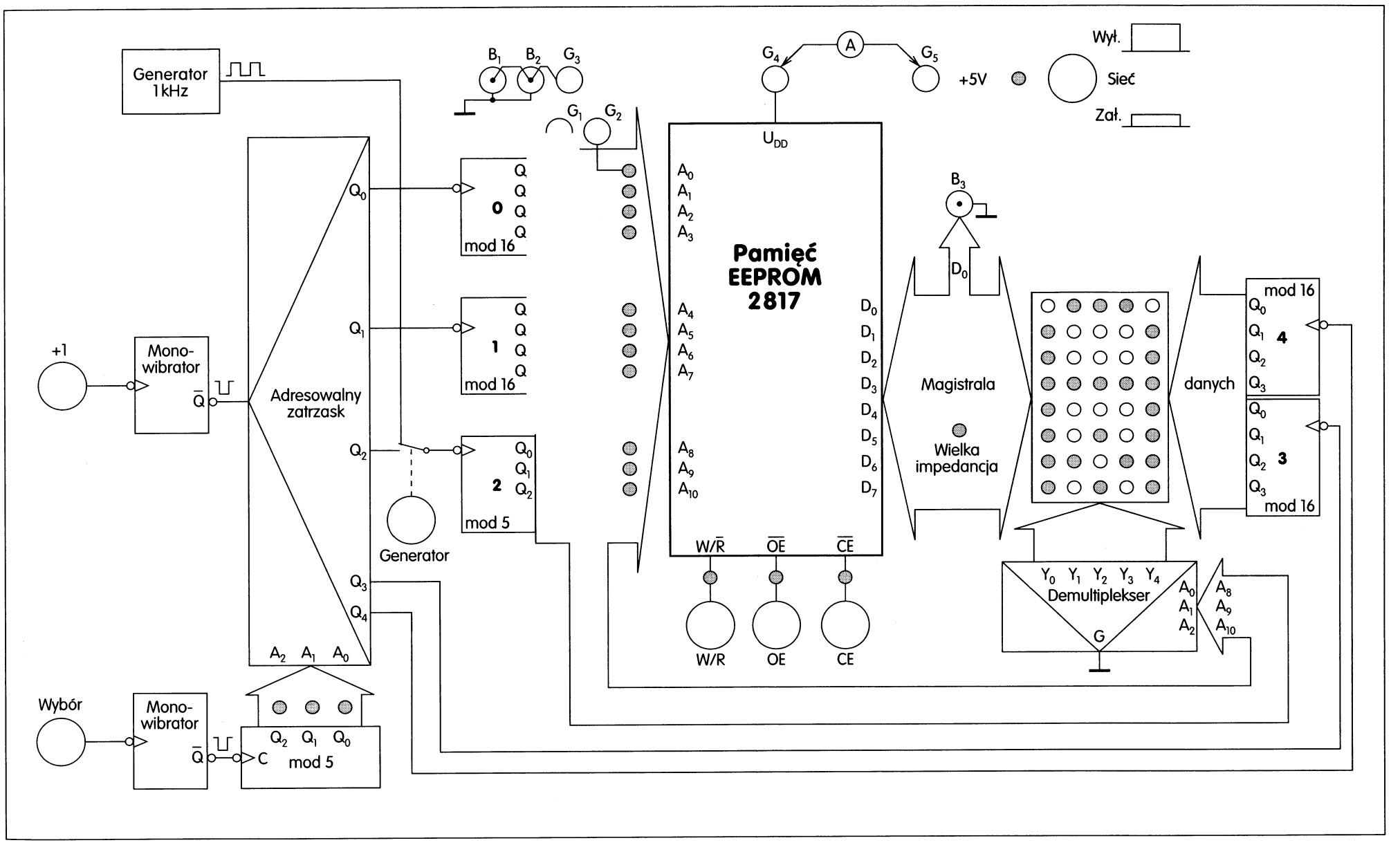
Z



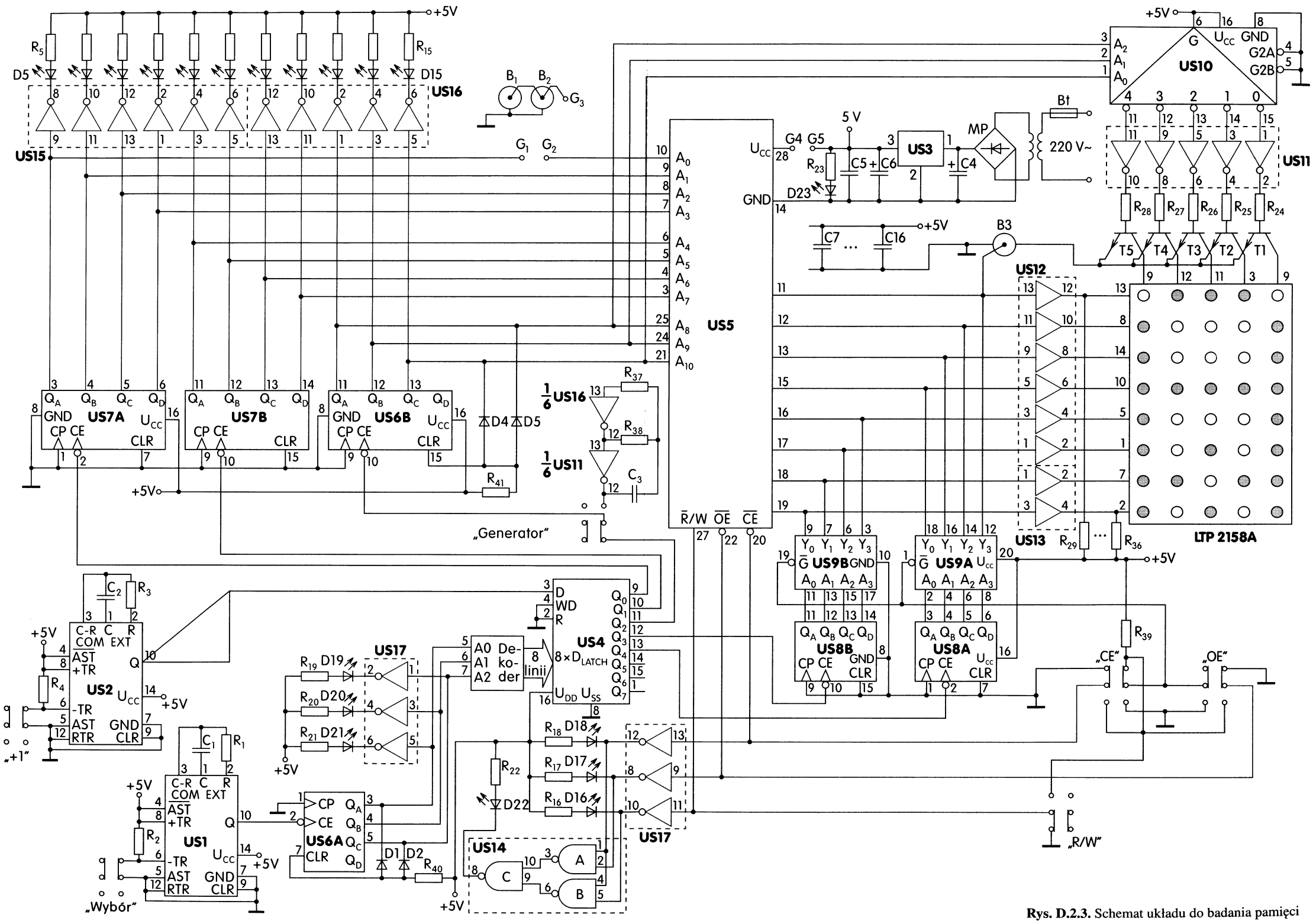
Rys. D.2.1. Schemat stopera



Rys. D.2.1a. Elementy stopera



Rys. D.2.2. Widok płyty czołowej stanowiska do badania pamięci



Rys. D.2.3. Schemat układu do badania pamięci