ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1979

Seria: AUTOMATYKA z. 49 Nr kol. 612

Ferdynand WAGNER

Instytut Elektroniki
Politechniki Śląskiej

PROJEKTOWANIE UKŁADÓW SEKWENCYJNYCH JAKO AUTOMATÓW MIKROPROGRAMOWANYCH

Streszczenie. W artykule zaprezentowano dwie podstawowe struktury układów sekwencyjnych budowanych jako automaty mikroprogramowane. Proponując graf przejść jako zasadniczą formę opisu działania tych automatów, pokazano możliwości ich przekształcania do postaci możliwej do realizacji w proponowanej strukturze z pamięcią stałą. Przedyskutowano możliwości popełnienia błędów w grafie przejść. Ustosunkowano się do zakresu zastosowań proponowanych struktur z punktu widzenia właściwości stosowanych w nich elementów (pamięci stałej i multipleksera). Zwrócono uwagę na szereg problemów związanych z projektowaniem automatu mikroprogramowanego, takich jak np. kodowanie sygnałów wyjściowych czy optymalne wykorzystanie multipleksera. Przeprowadzone w artykule rozważania zostały zilustrowane trzema przykładami prostych układów sekwencyjnych.

1. WPROWADZENIE

Koncepcja mikroprogramowanej struktury automatu znana jest od początku lat pięćdziesiątych, kiedy to M. Wilkes określił jej zasady. Jednakże wprowadzenie tej koncepcji do praktyki projektowania automatów napotykało na duże trudności, których źródłem był z jednej strony początkowy brak tanich i niezawodnych pamięci stałych, a z drugiej - brak metod projektowania tego rodzaju struktur. Dlatego też pierwsze mikroprogramowane automaty pojawiły się w latach sześćdziesiątych i to w komputerach, gdzie korzyści z ich zastosowania były najbardziej wyraźne. Natomiast wprowadzanie struktur mikroprogramowanych w urządzeniach mniej skomplikowanych aniżeli komputer jest rzadkie i spotykane np. w jednostkach sterujących urządzeń peryferyjnych.

Celem artykułu jest zaproponowanie dwóch podstawowych struktur automatów synchronicznych, które mogą z powodzeniem zastąpić stosowane najczęściej konstrukcje sekwencyjne, powstające czy to z wykorzystaniem znanych metod syntezy układów sekwencyjnych [3, 4, 5, 6], czy też intuicyjnie. Obie konstrukcje wywodzą się z podstawowej koncepcji układu mikroprogramowanego, opartej na pamięci stałej.

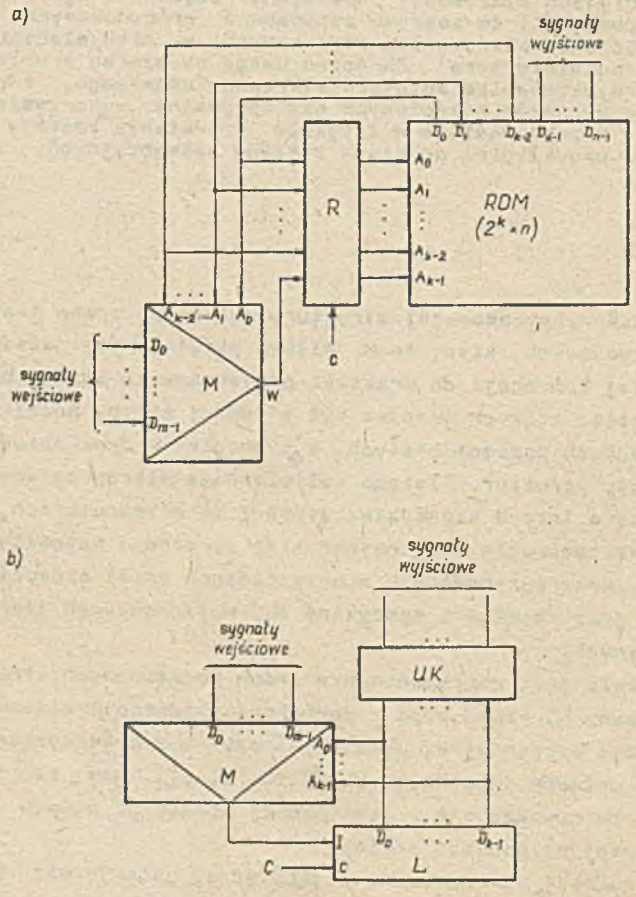
Podobne struktury są rozważane w [6], gdzie opisuje się działanie automatów sterujących za pomocą sieci działań i ich różne realizacje z wy-

korzystaniem m.in. liczników i rejestrów. Celem niniejszego artykułu jest zaproponowanie dwóch podstawowych struktur tego rodzaju układów, podanie sposobu dochodzenia do tych struktur i określenie możliwości i zakresu ich zastosowań.

2. AUTOMATY SEKWENCYJNE

2.1. Opis działania i schematy automatów

Dyskutowane w artykule struktury automatów przedstawia rys. 1. Pierwsza z nich (rys. 1a) składa się z rejestru R, pamięci stałej ROM oraz multipleksera M. W pamięci stałej zapamiętane są sygnały wyjściowe oraz adresy komórek pamięci, z których należy pobrać następne słowo. Układ jest synchroniczny, taktowany sygnałami zerowymi c, wpisującymi do rejestru R



Rys. 1. Struktury automatów synchronicznych
a) z pamięcią stałą; b) z licznikiem

kolejny adres komórki pamięci stałej. Adres ten jest modyfikowany sygnałem warunku W z multipleksera M, wybierającego w funkcji następnego adresu (stanu układu) jeden z sygnałów wejściowych. Struktura ta zakłada więc, że w każdym stanie jest sprawdzany co najwyżej jeden sygnał wejściowy. W przypadku gdy działanie automatu ma charakter licznikowy, tzn. że przejścia pomiędzy dwoma kolejnymi stanami są jednoznaczne, niezależne od sygnałów wejściowych, można zastosować strukturę przedstawioną na rys. 1b, w której rejestr jest zastąpiony licznikiem L, a pamięć stała układem kombinacyjnym UK typu dekoder. W tym rozwiązaniu stan licznika L określa stan układu, którego dekodowanie umożliwia wygenerowanie sygnałów sterujących, natomiast sygnał warunku W, odpowiadający wybranemu w danym stanie sygnałowi wejściowemu, decyduje jedynie o momencie zmiany stanu licznika; jest on w tym celu wprowadzony na wejście I, blokujące sygnały zegarowe c licznika.

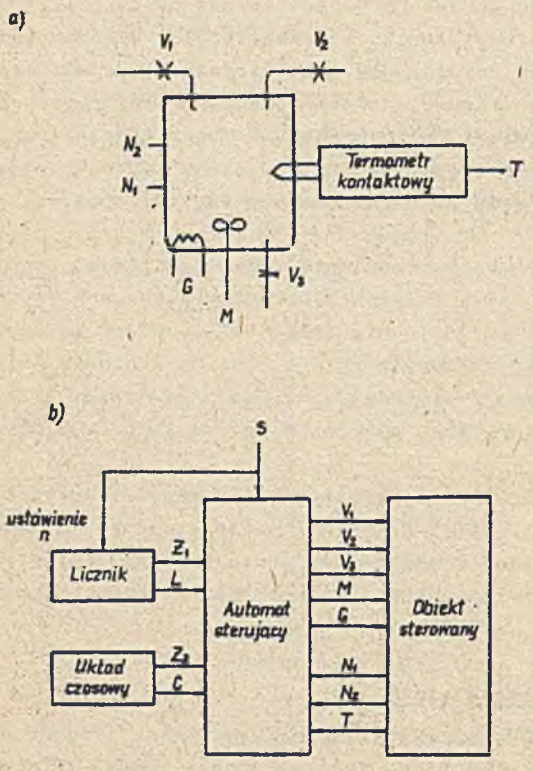
Sposób realizacji automatu wynika bezpośrednio z grafu diagramu przejść który proponuje się stosować do opisu działania układów synchronicznych. Można również stosować sieć działań [6]. Obie formy opisu układu są sobie równorzędne. W przypadku bardziej złożonych układów można stosować obie, przy czym punktem wyjścia jest wtedy sieć działań, na podstawie której można narysować graf przejść, który jest po prostu uproszczoną, bardziej czytelną formą tego schematu.

Dla prostszych układów, w których nie istnieje obawa zagubienia szczegółów działania układu, ujętych w sieci działań, można stosować wyłącznie graf przejść. Problemy związane z tworzeniem grafu przejść oraz przechodzeniem na program pamięci stałej zostaną zilustrowane dwoma przykładami.

2.2. Przykład automatu z pamięcią stałą

Omawiany w tym punkcie przykład dotyczy układu sterowania przygotowywaniem ciekłej mieszanki w zbiorniku (rys. 2a)i sam temat został wzięty z [8]. Mieszanka jest sporządzona z dwóch cieczy wlewanych do zbiornika rurami zamykanymi zaworami V_1 i V_2. Jeżeli zbiornik jest pusty, następuje zamknięcie zaworu V_3 na rurze wypływowej i otwarcie zaworu V_1. Po napełnieniu się zbiornika do poziomu wykrywanego czujnikiem N_1 następuje zamknięcie zaworu V_1 i dalsze napełnianie zbiornika drugą cieczą dzięki otwarciu zaworu V_2. Od momentu rozpoczęcia wlewania drugiej cieczy powinno pracować mieszadło M. Wlewanie cieczy, a więc zamknięcie zaworu V_2 kończy się po osiągnięciu przez ciecz w zbiorniku poziomu N_2. W tym momencie powinien zostać załączony grzejnik G przy nadal obracającym się mieszadle. Grzejnik nagrzewa mieszankę do temperatury T_d, osiągnięcie której jest sygnalizowane sygnałem T=1 z termometru kontaktowego. Nagrzanie mieszanki do wymaganej temperatury T_d oznacza, że jest ona przygotowana i można ją pobrać przez otwarcie zaworu V_3. Czynność ta powinna nastąpić równolegle z wyłączeniem grzejnika i mieszadła. Czas wypływu mieszanki ze zbiornika

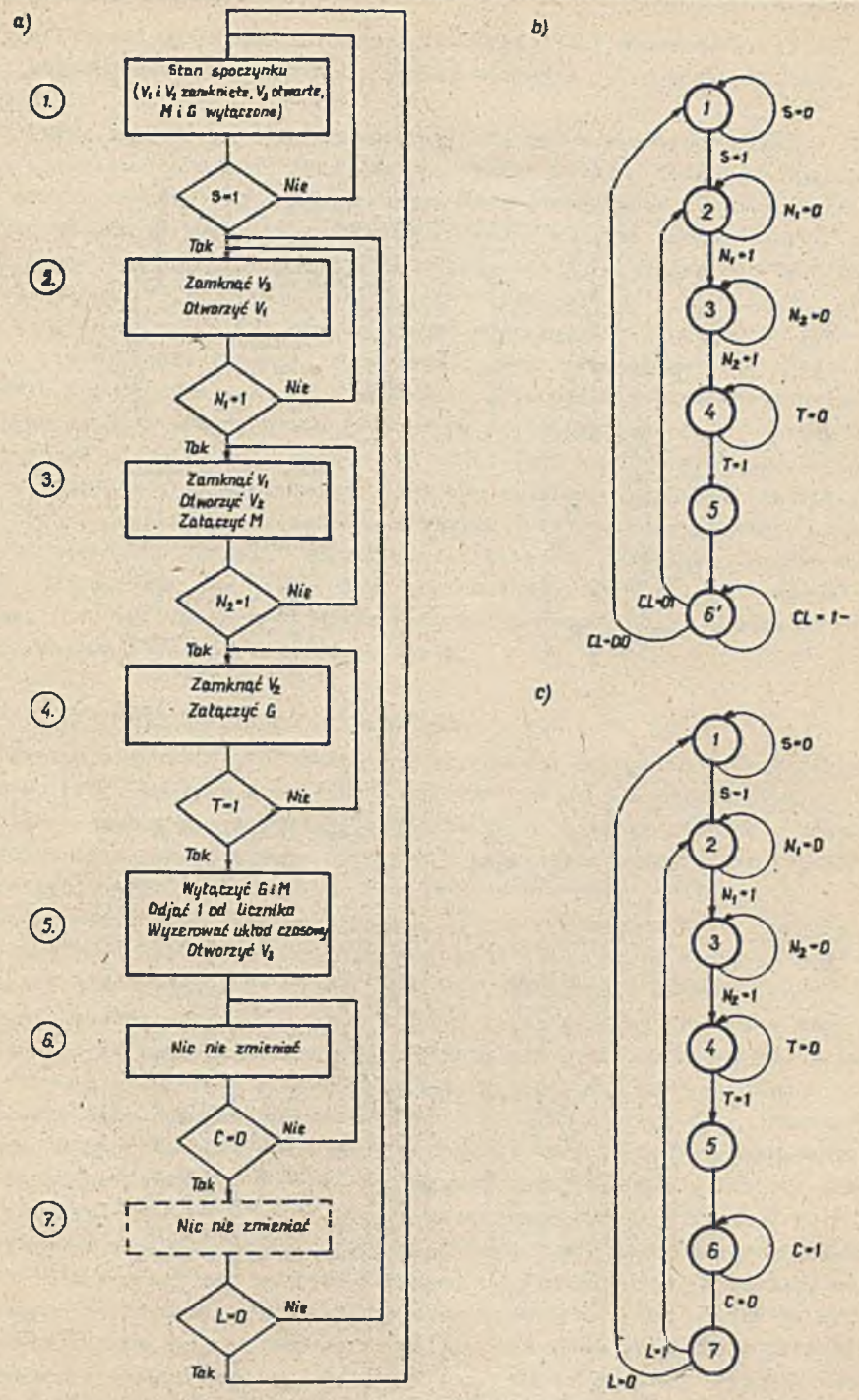
powinien być kontrolowany odpowiednim układem czasowym. Po odmierzeniu ustawionego w układzie czasowym okresu cykl przygotowania mieszanki powinien zostać powtórzony, przy czym liczbę powtórzeń n można nastawiać za pomocą odpowiedniego zadajnika cyfrowego.



Rys. 2. Układ przygotowywania mieszanki w zbiorniku
a) szkic zbiornika; b) schemat blokowy układu

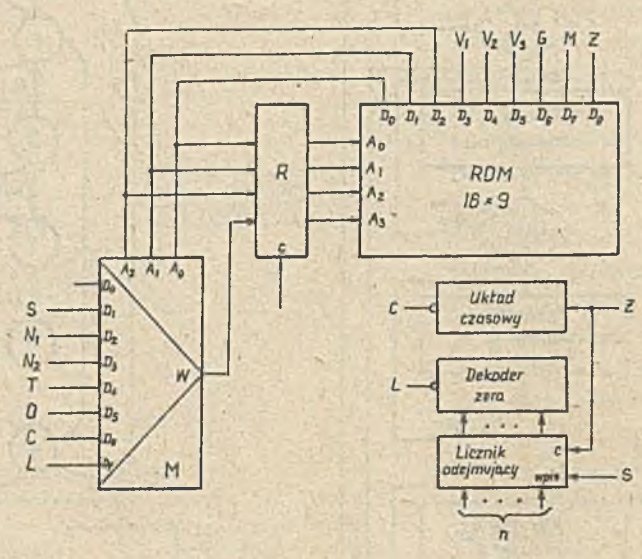
Na rys. 2b przedstawiono schemat blokowy, w którym wyodrębniono wszystkie sygnały wejściowe, przychodzące do automatu sterującego ze sterowanego obiektu (N_1, N_2, T) i sygnały wyjściowe generowane przez automat i służące do sterowania poszczególnymi elementami obiektu (V_1, V_2, V_3, M, G). Ponadto automat steruje pracą układu czasowego i licznikiem odliczającym liczbę cykli, generując odpowiednie sygnały (Z_1,Z_2) i sprawdzając stan sygnałów wyjściowych (L, C) tych układów. Pracę całości uruchamia sygnał startu S, inicjując proces n-krotnego przygotowywania mieszanki.

Sformułowane wyżej warunki działania automatu można opisać siecią działań pokazaną na rys. 3a. Prostokątne klatki operacyjne opisują operacje wykonywane w poszczególnych stanach automatu, a klatki warunkowe określają warunek, jaki musi być spełniony przy przejściu do następnego stanu. Przyjęta na rys. 1a struktura układu narzuca konieczność takiego naryeowania sieci działań, by w każdym stanie był sprawdzany co najwyżej jeden warunek - stąd wprowadzenie klatki operacyjnej nr 7. Równoważny do sieci działań jest graf przejść w którym klatki operacyjne zostały zastąpione wierzchołkami grafów (kółkami) i wyeliminowano klatki warunkowe. Wartości sygnałów warunkujących przejścia pomiędzy stanami automatu zostały napisane obok gałęzi (strzałek) grafu. Rysunek 3b przedstawia graf przejść bez stanu 7, co powoduje konieczność sprawdzania dwóch sygnałów C i L w stanie 6, z którego możliwe są trzy różne przejścia. Natomiast graf przejść na rys. 3c opisuje automat,który moż-



Rys. 3. Sieć działań (a) i grafy przejść (b,c) dla automatu sterującego z punktu 2.3

a)



b)

Stan	$A_0A_1A_2A_3$ (W)	$D_0D_1D_2$ ($\bar{A}_0\bar{A}_1\bar{A}_2$)	$D_3D_4D_5D_6D_7D_8$ ($V_1V_2V_3GMZ$)
1	1 0 0 0 1 0 0 1	1 0 0 0 1 0	0 0 1 0 0 0 0 0 1 0 0 0
2	0 1 0 0 0 1 0 1	0 1 0 1 1 0	1 0 0 0 0 0 1 0 0 0 0 0
3	1 1 0 0 1 1 0 1	1 1 0 0 0 1	0 1 0 0 1 0 0 1 0 0 1 0
4	0 0 1 0 0 0 1 1	0 0 1 1 0 1	0 0 0 1 1 0 0 0 0 1 1 0
5	1 0 1 0	0 1 1	0 0 1 0 0 1
6	0 1 1 0 0 1 1 1	1 1 1 0 1 1	0 0 1 0 0 0 0 0 1 0 0 0
7	1 1 1 0 1 1 1 1	1 0 0 0 1 0	0 0 1 0 0 0 0 0 1 0 0 0

Rys. 4. Schemat automatu z punktu 2.3
a) struktura układu; b) program w pamięci stałej

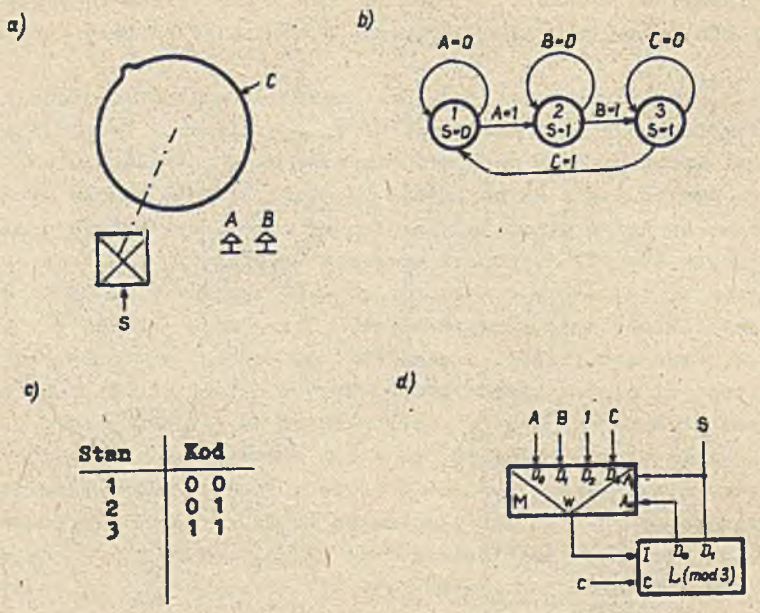
na zrealizować wg schematu na rys. 1a. Graf ten charakteryzuje się tym, że z każdego stanu (wierzchołka grafu) wychodzą tylko dwie strzałki (gałęzie).

Z grafu przejść wynika, że automat może być w siedmiu stanach. Stany tego automatu można więc zakodować na trzech bitach. Uwzględniając jeszcze sygnał warunku można narysować schemat układu przedstawiony na rys.4a, na którym pamięć stała ma pojemność 16 słów. Długość słowa wynosi dziewięć bitów, na które składają się: 3 bity adresu następnego oraz 6 bitów, określających sygnały wyjściowe automatu (sygnały Z_1 i Z_2 zastąpiono jednym Z, gdyż są one sobie równe). Chcąc wykorzystać w układzie typowe pamięci stałe, które mają słowa 8-bitowe, można zrezygnować z pamiętania sygnału Z w pamięci stałej, a generować go poprzez zdekodowanie stanu 5. Program pamięci stałej przedstawiono na tym samym rysunku. Otrzymuje się go poprzez przepisanie treści grafu przejść do tablicy programu. Wymagało to zakodowania stanów automatu, co w tym rozwiązaniu odpowiada przypisaniu poszczególnym stanom adresów w pamięci stałej; w przykładzie stanom przyporządkowano adresy, których postać binarna odpowiada numerom stanów (zakładając, że bitem najmłodszym jest bit A_0). Warunek W, modyfikujący adres, jest wprowadzany na najstarszy bit adresowy (A_3). W pamięci stałej wykorzystano 13 z 16 możliwych komórek.

2.3. Przykład automatu z licznikiem

Tematem przykładu jest układ [4] sterowania stycznika S, załączającego silnik, który poprzez przekładnię napędza tarczę z występem (rys. 5a). W stanie spoczynku występ dotyka czujnika C. Załączenie silnika następuje przyciskiem A, wyłączenie - w momencie dotknięcia występem czujnika C po naciśnięciu przycisku B.

Rozumowanie, jakie przeprowadza się przy rysowaniu grafu przejść na rys. 5b, jest oczywiste - silnik może stać (stan 1) i wtedy należy kontrolować stan sygnału A, silnik może być załączony i obracać tarczę, przy czym nie naciśnięto jeszcze przycisku B (stan 2) i należy wobec tego kontrolować stan sygnału B i wreszcie silnik może być załączony po naciśnięciu przycisku B (stan 3) i wtedy należy kontrolować stan sygnału C. Na podstawie tego rozumowania powstał graf przejść na rys. 5b. Rozwiązanie układu, wykorzystujące strukturę wg rys. 1b, przy przyjętym kodowaniu stanów licznika pokazanym na rys. 5c, przedstawia rys. 5d. Zastosowanie w tym przypadku układu o strukturze z licznikiem wynika z tego, że graf przejść nie zawiera pętli, obejmującej więcej niż jeden wierzchołek (stan) Innymi słowy przejścia pomiędzy stanami są jednoznaczne, a sygnały warunku decydują jedynie o momencie realizacji przejść. Kod licznika wynosi: 00-01-11. Na wejście D_2 podano sygnał 1, co zapewnia wyjście licznika z niewykorzystanego stanu 10, w którym może się on znaleźć, np. po załączeniu napięcia zasilającego. Jeżeli porównać sposób rozwiązywania takiego prostego układu w postaci asynchronicznej metodą np. tablic przejść [4],



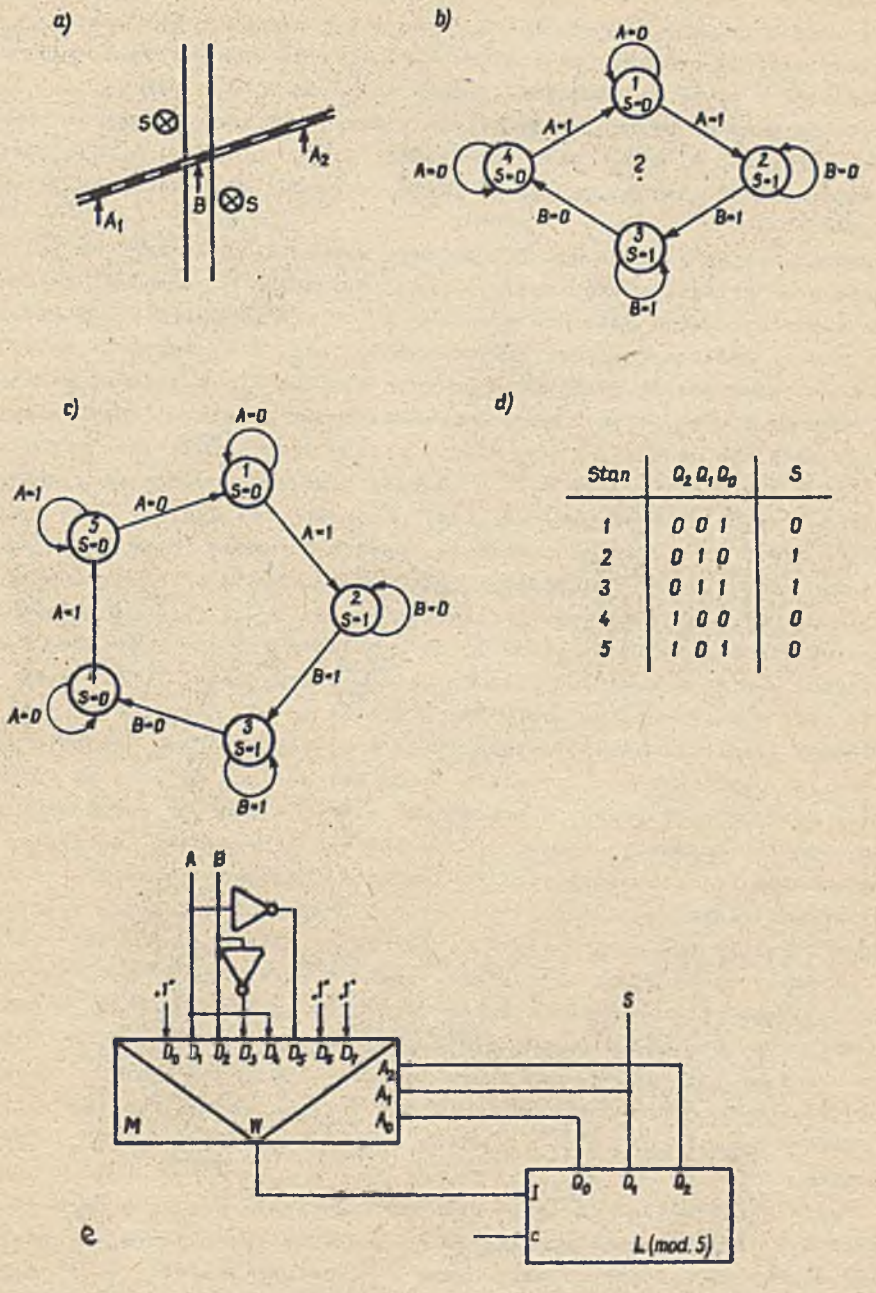
Rys. 5. Automat sterujący z punktu 2.4
a) szkic mechanizmu; b) graf przejść automatu; c) schemat automatu

a w szczególności sposób minimalizacji stanów (pierwotna tablica programu zawiera 10 stanów), to widać wyraźnie zalety przedstawionego na rys. 5 podejścia. Możliwość otrzymywania grafów przejść o minimalnej lub zbliżonej do minimalnej liczbie stanów wynika z przyjętego sposobu rozwiązywania automatu, charakteryzującego się tym, że dla każdego stanu kontroluje się tylko ten sygnał, który decyduje o przejściu do innego stanu; eliminuje to z góry konieczność analizy zachowania się automatu przy zmianach innych sygnałów, co jest charakterystyczne dla tablicy przejść.

3. PROBLEMY WYSTĘPUJĄCE PRZY PROJEKTOWANIU

3.1. Tworzenie grafu przejść

Utworzenie grafu przejść nie zawsze jest tak proste jak w poprzednio rozwiązywanych przykładach. W pierwszym rzędzie należy zdawać sobie sprawę, że automat o strukturze mikroprogramowanej wg rys. 1a jest tzw. automatem Moore'a, tzn. automatem, którego stan wyjść zależy tylko od stanu wewnętrznego. Sposób tworzenia grafu przejść umożliwia powstanie błędów, polegających na tym, że zawiera on za mało stanów, a praca zbudowanego wg niego automatu będzie błędna. Dlatego też graf przejść musi być analizowany czy rzeczywiście oddaje on poprawnie pracę projektowanego urządzenia. Ilustracją tego problemu niech będzie poniższy przykład.

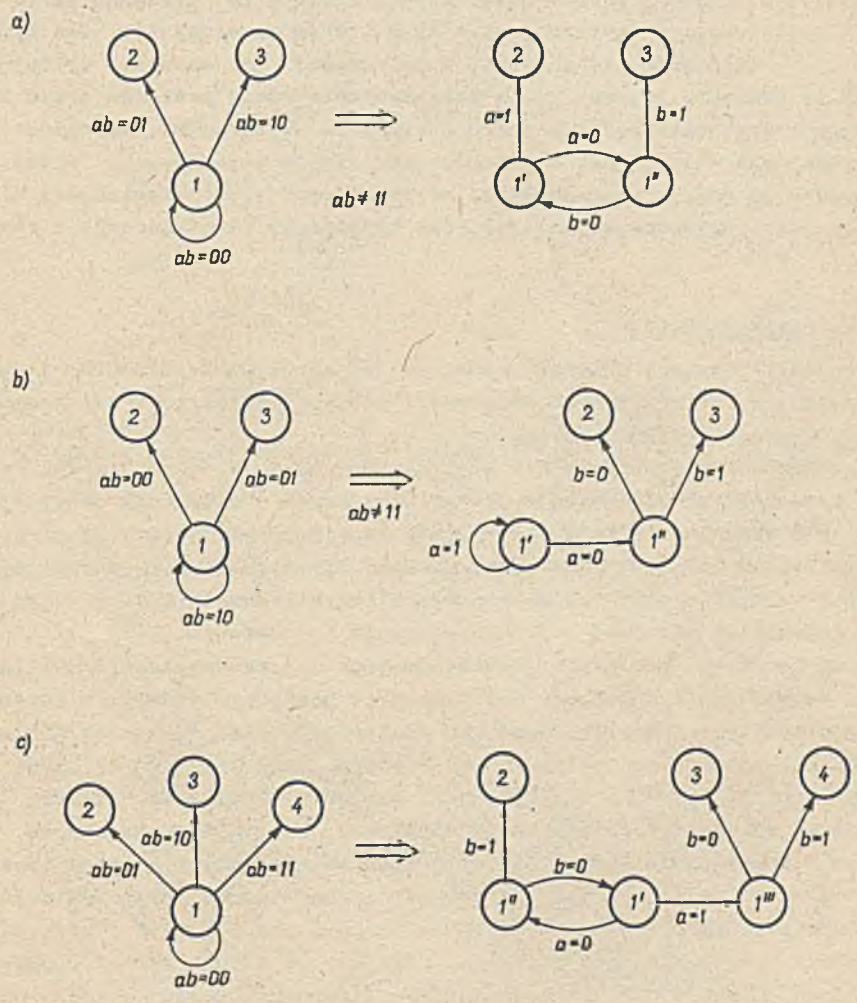


Rys. 6. Automat sterujący zapaleniem świateł na przejeździe kolejowym a) szkic rozmieszczenia czujników; b) błędny graf przejść; c) prawidłowy graf przejść; d) kod licznika mod 5; e) schemat automatu sterującego

Jest to znany przykład [3, 4] sterowania światłami względnie zaporą na przejeździe kolejowym (rys. 6a) w funkcji sygnałów z czujników A_1, A_2 i B. Zakłada się, że pomiędzy czujnikami A_1 i A_2 może się znajdować tylko jeden pociąg o dowolnej długości, jadący od A_1 do A_2 lub odwrotnie. Sposób zapalania świateł ostrzegawczych jest powszechnie znany. Zamiast operować dwoma sygnałami A_1 i A_2, można wprowadzić jeden $A = A_1 + A_2$, co wynika z symetrii układu (zachowuje się on identycznie dla obu kierunków jazdy pociągów).

Zastanawiając się nad działaniem tego automatu i tworząc na tej podstawie graf przejść można popełnić błąd, dochodząc do wniosku, że automat może się znajdować w czterech stanach: 1 - w którym światła się nie palą, tzn. nie ma pociągu pomiędzy czujnikami A_1 i A_2, 2 - w którym światła są zapalone, gdyż pociąg przejechał czujnik A_1 (lub A_2) i zmierza w kierunku czujnika B, 3 - w którym pociąg przejeżdża przez przejazd (światła ostrzegawcze palą się nadal) i 4 - w którym pociąg przejechał już przejazd i zmierza w kierunku A_1 (lub A_2). W efekcie tego rozumowania otrzymuje się diagram przejść pokazany na rys. 6b. Automat realizujący ten diagram przejść nie będzie działał poprawnie, gdyż w momencie najechania pociągu na czujnik A_1 (lub A_2) w sytuacji, gdy automat był w stanie 4 (pociąg odjeżdża od przejazdu) nastąpi przeskok do stanu 1, z którego nastąpi natychmiastowe przejście do stanu 2, co spowoduje zapalenie świateł ostrzegawczych, a więc błędne zadziałanie układu sterującego. Przyczyną tego błędu jest występienie po sobie dwóch stanów (4 i 1), w których jest kontrolowany stan tego samego sygnału (A), przy czym warunek wyjścia z tych stanów jest taki sam (A=1). Jeżeli natomiast warunki wyjścia są różne, w grafie przejść mogą po sobie występować stany, w których sprawdzane są te same sygnały (porównaj stany 2 i 3, w których sprawdzany jest sygnał B). Rozwiązaniem tej trudności jest wprowadzenie dodatkowego stanu 5 (rys. 6c), przedzielającego stany 1 i 4; stan 5 odpowiada sytuacji, w której pociąg przejeżdża nad czujnikiem A, odjeżdżając od niego. Przeskok ze stanu 5 do 1 następuje dopiero w momencie zaniku sygnału A=1. Schemat układu, realizującego program działania przedstawiony grafem przejść z rys. 6c, pokazuje rys. 6e. Układ składa się z multipleksera M i licznika L, liczącego mod 5 wg kodu z rys. 6d, obejmującego 5 kolejnych liczb naturalnych kodu dwójkowego, poczynając od 1. Dzięki takiemu kodowi nie jest potrzebny dekoder sygnału s, który odpowiada bezpośrednio drugiej (Q_1) pozycji licznika.

Drugim problemem, który może utrudnić tworzenie diagramu przejść, jest przyjęte w rozpatrywanej strukturze sprawdzanie tylko jednego sygnału warunku dla każdego stanu. Jeżeli nawet w pierwszej wersji jest wygodnie narysować diagram przejść, zawierający wierzchołki, które posiadają więcej niż dwie wychodzące strzałki, to następnie należy dokonać przekształcenia tego grafu tak, by występowały w nim jedynie wierzchołki z jedną lub dwiema strzałkami wyjściowymi. Przykład takiego postępowania został



Rys. 7. Przykłady zamiany wierzchołków grafu z trzema lub czterema strzałkami wyjściowymi na wierzchołki z dwiema strzałkami

zademonstrowany w przykładzie na rys. 3, w którym graf ze stanem 6' został przekształcona na graf ze stanami 6 i 7, które zastąpiły stan 6'.

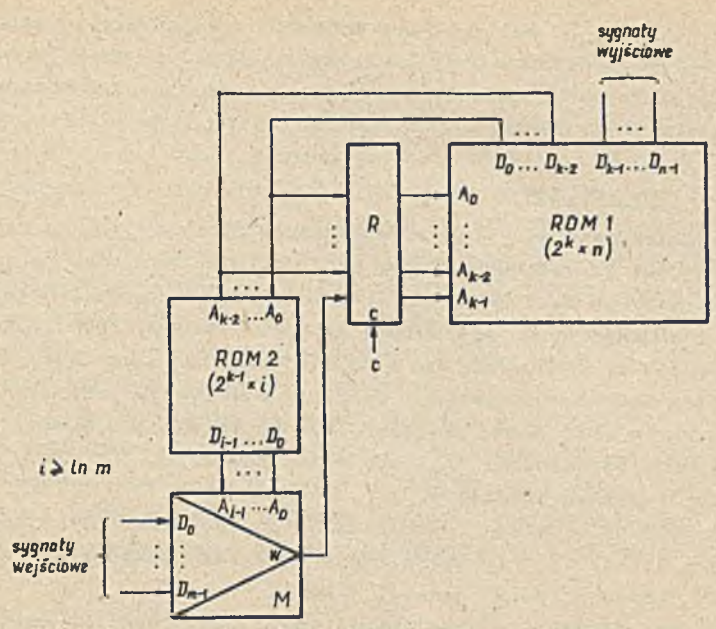
Ograniczając się jedynie do rozpatrzenia wierzchołków grafów odpowiadających stanom, w których sprawdzane są dwa sygnały, a więc wierzchołków z trzema lub czterema strzałkami, można zauważyć, że istnieją dwa podstawowe sposoby przekształcenia takiego wierzchołka na dwa równoważne mu wierzchołki z pojedynczą strzałką wyjściową. Sposoby te pokazane są na rys. 7a i b, na których rozpatrzono stan 1, w którym sprawdza się dwa sygnały a i b, przy założeniu, że ab ≠ 11. W zależności od warunków opisujących przejścia pomiędzy stanami można otrzymać dwie różne postacie grafu zawierającego wierzchołki tylko z dwoma strzałkami wyjściowymi. Natomiast rys. 7c przedstawia wierzchołek o czterech strzałkach wyjściowych. Widać, że równoważny mu graf z wierzchołkami posiadającymi tylko wierzchołki z dwiema strzałkami wyjściowymi powstał jako kombinacja konfiguracji z rys. 7a i 7b.

3.2. Wielkość układu

Maksymalne wymiary układów budowanych wg struktur pokazanych na rys. 1 są uzależnione od aktualnych możliwości nabycia względnie realizacji pamięci stałych i multiplekserów.

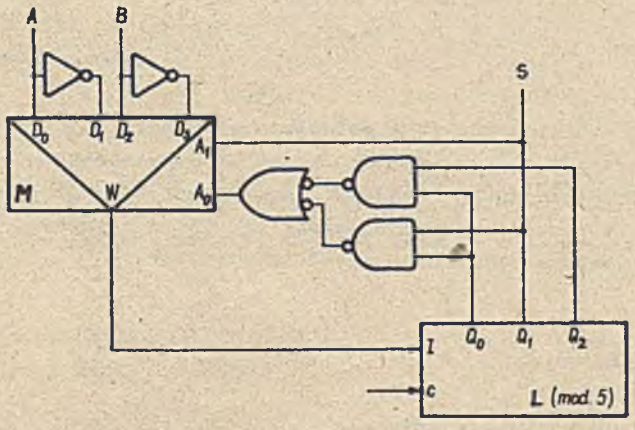
W zasadzie pamięci stałe nie stanowią już obecnie ograniczenia. Pojemności pojedynczych elementów scalonych, będących w produkcji seryjnej, wynoszą 4k w organizacji zazwyczaj 512x8, przy czym w najbliższym czasie oczekuje się wzrostu tej pojemności do 16k, czy nawet 64k. Te pojedyncze elementy pamięci stałej można łączyć w większe jednostki, otrzymując wszystkie pojemności wymagane z punktu widzenia zastosowań.

Nieco bardziej złożonym zagadnieniem jest sprawa realizacji multiplekserów. Największe produkowane multipleksery posiadają 4 wejścia adresowe. Istnieją możliwości powiększania ich wielkości przez wykorzystanie wejścia odblokowującego wg ogólnie znanych zasad. Ponieważ jednak każde dodatkowe wejście adresowe multipleksera powoduje podwojenie liczby tych elementów, możliwości rozszerzenia wielkości multiplekserów nie są zbyt wielkie i praktycznie kończą się na 6 liniach adresowych, czemu odpowiadają 4 elementy multipleksera. Z drugiej strony liczba sygnałów wejściowych automatu, sprawdzanych w multiplekserze i decydujących o przejściach w punktach rozgałęzień grafu przejść nie jest zbyt duża i z reguły nie przekracza kilkunastu do kilkudziesięciu wielkości. Lepiej wobec tego przyjąć liczbę wejść multipleksera nie mniejszą od liczby sygnałów wejściowych, natomiast za pomocą dodatkowej pamięci stałej ROM2 dokonać przekodowania właściwego adresu stanu następnego, pobieranego z głównej pamięci ROM1 na adres potrzebny do sterowania multipleksera. Koncepcję tę pokazuje rys. 8. Dzięki temu rozwiązaniu ograniczenia w stosowaniu proponowanej struktury automatu, wynikające z wielkości multipleksera, maleją. Rozmiary układu są wtedy ograniczone maksymalną liczbą sygnałów wejściowych.



Rys. 8. Optymalne wykorzystanie multipleksera w automacie mikroprogramowanym

która nie może przewyższać liczby wejść multipleksera. Przyjmując multiplekser o 6 liniach adresowych jako maksymalny, oznacza ograniczenie zastosowań do układów o liczbie sygnałów wejściowych nie większej niż 64.



Rys. 9. Schemat automatu z rys. 6 ze zredukowanym multiplekserem

Na rysunku 9 pokazano zastosowanie tego sposobu redukcji liczby wejść multipleksera w układzie z rys. 6 (sterowanie światłami ostrzegawczymi na przejeździe kolejowym). Ponieważ w układzie są tylko cztery sygnały wejś-

ciowe: A, $\bar{A}$, B, $\bar{B}$, multiplekser może być 4-wejściowy. Natomiast pamięć stała, która przekodowywuje 3-bitowy stan układu na 2-bitowy sygnał adresowy multipleksera, może być zrealizowana za pomocą pokazanego na rysunku układu kombinacyjnego.

3.3. Inne problemy

Zasadnicze struktury układów pokazane na rys. 1 mogą być modyfikowane w zależności od zastosowań, np. możliwe jest tworzenie struktur mieszanych, w których podstawę jest licznik z możliwością równoległego wpisu. Automat realizuje wtedy zasadniczo program liniowy, który może być modyfikowany poprzez równoległy wpis do licznika. Stosowanie takiej struktury jest uzasadnione w sytuacji, gdy tych modyfikacji jest niewiele; w przeciwnym razie układ modyfikujący ulega zbytniej rozbudowie. Równocześnie funkcjonowanie całości staje się mniej czytelne, a przy projektowaniu częściej popełnia się błędy.

W przykładzie układu sterowania przygotowywaniem mieszanki w zbiorniku (rys. 2, 3 i 4) sygnały wyjściowe nie zostały zakodowane, stąd stosunkowo długie, 9-bitowe, słowo pamięci stałej.

Z punktu widzenia minimalizacji pojemności pamięci stałej celowe jest kodowanie stanów sygnałów wyjściowych - w skrajnym przypadku stany bitów, określające następny adres pamięci, mogą określać w postaci zakodowanej stany sygnałów wyjściowych. Zmniejszenie pojemności pamięci stałej odbywa się wtedy kosztem powiększenia układów zewnętrznych, dekodujących sygnały wyjściowe. Dlatego decyzja o sposobie zakodowania sygnałów wyjściowych nie jest prosta i musi uwzględnić różne czynniki. Istotną rolę odgrywa tutaj długość słów produkowanych pamięci stałych. Dla przykładu w zadaniu ze zbiornikiem (rys. 4) celowe byłoby zmniejszenie liczby bitów słowa pamięci stałej o jeden, biorąc pod uwagę to, że typowe pamięci stałe mają słowa 8-bitowe.

W artykule pominięto zupełnie zagadnienie doboru częstotliwości sygnału zerowego c, wyznaczającego moment zmiany stanu automatu oraz związane z tym problemy synchronizacji pracy urządzeń współpracujących z automatem. Sprawy te również nie są zupełnie oczywiste i proste, lecz nie mają istotnego wpływu na omawiane w artykule struktury.

4. PODSUMOWANIE

Przedyskutowane w artykule możliwości realizacji układów sekwencyjnych, nawiązując do znanej koncepcji mikroprogramowanej struktury automatów, przyjmuję jako podstawę aktualną bazę elementową, na którą składają się pamięci stałe i elementy scalone średniej skali integracji.

Zasadniczą fazę procesu projektowania jest sprecyzowanie sposobu funkcjonowania układu, który będzie uwzględniał zarówno powiązania pomiędzy

sygnałami wyjściowymi i wejściowymi automatu jak i specyfikę zastosowanych w układzie podzespołów. Ten sposób funkcjonowania można nazwać programem działania automatu i zapisać w formie grafu przejść (lub sieci działań). Graf przejść niekoniecznie musi określać minimalną liczbę stanów; liczbę tę zwiększają jeszcze przekształcenia wierzchołków grafu z kilkoma strzałkami na wierzchołki z dwiema strzałkami wyjściowymi. Jednakże zazwyczaj liczba stanów automatu nie ma wpływu na pojemność pamięci stałej, co wynika ze skokowych zmian tej pojemności z potęgą liczby 2, a więc próby minimalizacji grafu są istotne tylko w sytuacjach, w których pojemność pamięci przekracza nieznacznie potęgę liczby 2. To samo dotyczy kodowania sygnałów wyjściowych. Czasem problem minimalizacji stanów automatu może wynikać z ograniczeń czasowych.

Praktyka projektowania automatów mikroprogramowania uczy, że dużą rolę przy ich powstawaniu odgrywa intuicja, doświadczenie i zdolności projektanta. Pełne sformalizowanie tego zagadnienia nie wydaje się aktualnie możliwe. Określenie jednak pewnych zasadniczych struktur tych automatów i zasad ich stosowania może ułatwić konstruowanie tego rodzaju układów.

LITERATURA

[1] Fletcher W.I., Despain A.M.: Simplify sequential circuit designs with programmable ROMs, Electronic Design, Nº 14, ss. 70-72.

[2] Kenny R.: Microprogramming Simplifies Control System Design. Computer Design, February 1975, ss. 96-98.

[3] Perkowski M., Rydzewski A., Misiurewicz P.: Teoria układów logicznych. Zagadnienia wybrane. Wyd. Politechniki Warszawskiej, Warszawa 1977.

[4] Praca zbiorowa pod red. J. Siwińskiego: Zbiór zadań z układów przełączających. Wyd. II poprawione. Skrypt uczelniany Politechniki Śląskiej, nr 656, Gliwice 1976.

[5] Siwiński J.: Układy przełączające w atuomatyce. WNT, Warszawa 1968.

[6] Traczyk W.: Układy cyfrowe automatyki. WNT, Warszawa 1974.

[7] Tudruj M.: Mikroprogramowane sterowanie w maszynach cyfrowych - aktualne problemy. Materiały z Sympozjum "Organizacja maszyn cyfrowych i mikroprogramowanie". T.I. PWN, Warszawa 1976.

[8] Yong A.: Digital Sequencers. Digital Design, May 1977, ss. 74-78.

ПРОЕКТИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ СХЕМ
В ВИДЕ МИКРОПРОГРАММНЫХ АВТОМАТОВ

Р е з ю м е

В работе представлены две основные структуры последовательных схем, построенных в виде микропрограммных автоматов. Используя граф переходов в качестве основной формы описания действия этих автоматов, автор показывает возможность преобразования их так, чтобы можно было осуществить в предла-

гаемой структуре с постоянным запоминающим устройством. Рассмотрена также возможность выступления ошибок в графе переходов. Представлена область применения предлагаемых структур с точки зрения применяемых в них элементов постоянное запоминающее устройство, мультиплексор . Показывается ряд проблем, связанных с проектированием микропрограммного автомата, например, кодирование выходных сигналов или оптимальное использование мультиплексора. Затронутые в работе вопросы иллюстрируются на примере трех простых последовательных схем.

DESIGN OF SEQUENTIAL CIRCUITS
AS MICRIPROGRAMMED SYSTEMS

S u m m a r y

In the paper two basic structures of sequential circuits built out as the microprogrammed systems have been presented. A transition diagram has been proposed to describe the system. It has been shown how to transform the transition diagram into a form which corresponds to the structure with a read only memory. The possible errors in the transition diagram planning have been discussed. The application range of the structures proposed in the paper has been defined, taking into account the features of the main elements (read only memory, and a multiplexer). Some problems which influence the design of the microprogrammed system have been considered, e.g. coding of output signals, or how to make the best of a multiplexer. The paper contains three examples of simple sequential circuits.