

5.1. Sisteme de conducere cu stări finite în logică cablată

5.1.1. Reducerea numărului de stări ale automatelor finite

În general, pentru realizarea unei anumite dependențe impuse prin tema de proiectare între semnalele de ieșire și cele de intrare există mai multe structuri de automate finite care se comportă identic.

Acste automate finite diferă între ele atât prin numărul de stări cât și prin tranzitia dintre aceste stări. Pentru a obține o structură minimală de circuite secvențiale care să implementeze aceste automate (echivalente între ele întrucât realizează această dependență intrare-ieșire) este necesar ca în prealabil să se caute stări echivalente ale automatului finit și să se înlătăruască printr-o singură stare.

Pentru a înțelege procesul de reducere al numărului de stări la automatele finite trebuie enunțate în prealabil două reguli:

(1) Două stări x_i , x_j ale unui automat finit sunt echivalente dacă aplicând la intrarea automatului aflat în starea x_i sau x_j o secvență de semnale de intrare identică și de lungime arbitrară se obține la ieșirea automatului secvență de semnale identice.

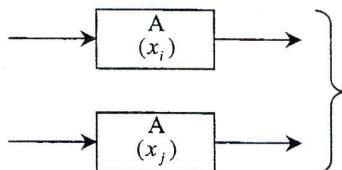


Figura 5.1

Pentru a defini modul în care două stări sunt echivalente, dăm altă definiție:

(2) Două stări x_i și x_j ale unui automat sunt stări k -echivalente dacă pentru orice secvență de semnale de intrare cu lungime egală cu k aplicată la intarea automatului aflat în starea x_i sau x_j se obține la ieșire aceeași secvență de semnale. Această noțiune de k -echivalență a stărilor partajează stările în clase de echivalență.

Determinarea claselor de echivalență se face recursiv: 2 stări echivalente k sunt și $(k+1)$ echivalente dacă stările lor succesoare pentru orice semnal aplicat la intrare sunt la rândul lor k echivalente.

Afirmarea e corectă pentru că dacă aplicăm la intrarea unor stări k -echivalente (deci care nu pot fi diferențiate, identificate prin secvența de intrare de lungime k) un același semnal de intrare și automatului, vor trece într-o stare viitoare, tot k -echivalență, înseamnă că secvența de semnale de la intrare pentru care stările inițiale nu pot fi identificate, va avea lungimea $(k+1)$.

Partajarea în clase de echivalență se face identificând într-o primă etapă stările 1-echivalente. Acestea sunt stările care nu pot fi identificate pentru lungimea secvenței de intrare $k=1$, adică sunt stările care la semnale de intrare identice oferă semnale de ieșire identice.

În continuare se caută stările 2-echivalente. Pentru aceasta se consideră grupurile de stări 1-echivalente și se analizează starea lor succesoare.

Dacă pentru intrări identice, starea lor următoare sunt 1-echivalente, înseamnă că stările nu sunt de fapt nu numai 1-echivalente și chiar 2-echivalente pentru că nici următorul semnal de intrare nu le va putea diferenția.

Partajarea în clase de echivalență continuă până când partiția $(n+1)$ e identică cu partiția n .

Când $P_n \equiv P_{n+1}$ spunem că stările $(n+1)$ echivalente sunt stări echivalente și procedăm la înlocuirea lor printr-o singură stare.

Considerăm două stări pentru identificare:

S.v. S.p.	u_1	u_2	y_1	y_2
A	A	C	0	0
B	B	D	0	0
C	E	C	0	1
D	F	D	0	1
E	E	A	1	1
F	F	B	1	1

-> Automat Mealy

=>

S.v. S.p.	u_1	u_2	y_1	y_2
α	α	β	0	0
β	γ	β	0	1
γ	γ	α	1	1

S.v. S.p.	u_1	u_2	y
A	A	B	0
B	C	B	0
C	A	C	0
D	F	D	0
E	H	E	0
F	G	F	0
G	A	D	1

-> Automat Moore

=>

S.v. S.p.	u_1	u_2	y
α	α	α	0
δ	β	δ	0
β	γ	β	0
γ	α	δ	1

- automatul
redus
echivalent

$$P_1 = (A, B, C, D, E, F)(G, H)$$

Figura 5.2

$$P_2 = (A, B, C, D)(E, F)(G, H)$$

$$P_3 = (A, B, C)(D)(E, F)(G, H)$$

$$P_4 = (A, B, C)(D)(E, F)(G, H)$$

$$P_4 \equiv P_3 \Rightarrow (A, B, C) \rightarrow \alpha$$

(5.1)

$$D \rightarrow \delta$$

$$(E, F) \rightarrow \beta$$

$$(G, H) \rightarrow \gamma$$

5.1.2. Sinteză circuitelor sevențiale asincrone cu un număr mare intrări

Metoda de sinteză ce va fi prezentată în continuare este cunoscută sub denumirea de metoda ordinogramei și cuprinde următoarele etape:

- construirea diagramei stărilor;
- codificarea stărilor;
- proiectarea schemei logice;

În cazul circuitelor cu un număr mare de intrări sinteza după cum se observă începe direct cu construirea unei diagrame a stărilor care se consideră minimă (fără a mai fi minimizată).

În cadrul acestei metode nu se mai urmărește reducerea numărului de stări. Întrucât majoritatea aplicațiilor industriale intră în această categorie în continuu se va prezenta, pe baza unui exemplu, această metodă.

Exemplu: Sinteză schemei de comandă a unui cărucior.

Să se proiecteze instalația de comandă pentru un cărucior care se deplasează între două poziții delimitate de limitatoarele LS și LD. Căruciorul este acționat de un motor comandat prin două contactoare MS și MD. Instalația are două regimuri de

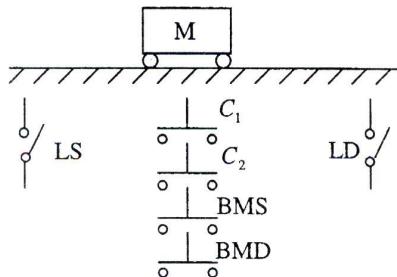


Figura 5.3

a) Construirea diagramei stărilor pentru regimul AUTOMAT

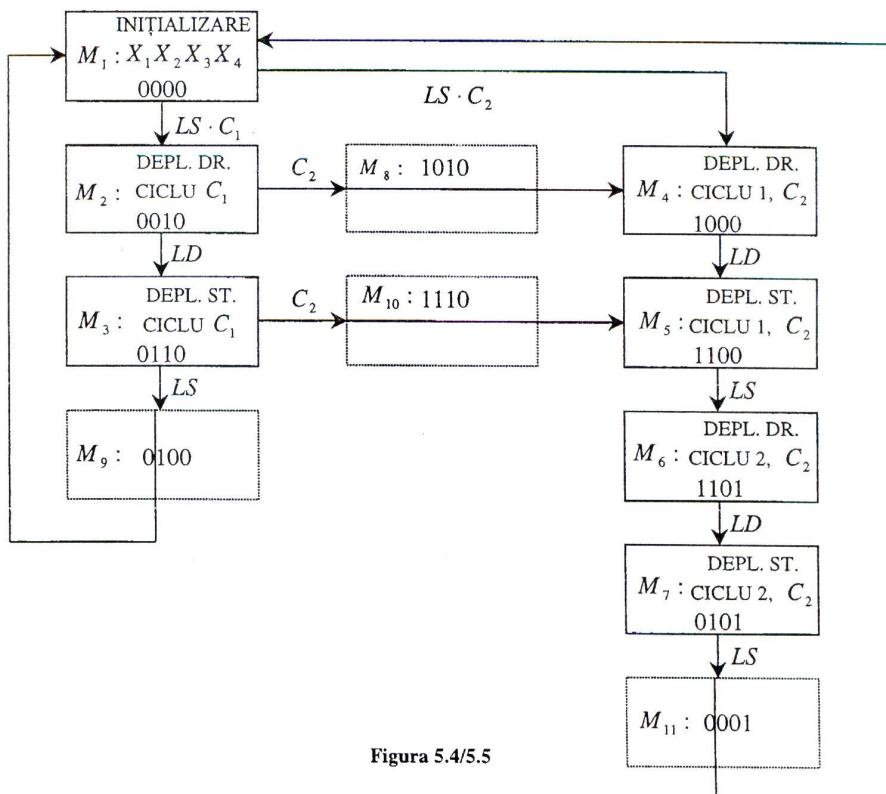


Figura 5.4/5.5

funcționare care se aleg cu un întreupător AUT./MAN. În regimul automat căruciorul nu pornește decât dacă se află inițial în poziția extremă stânga, adică LS=1. Căruciorul trebuie să execute un ciclu dus-întors la acționarea butonului C_1 și două cicluri la acționarea butonului C_2 . Dacă se acționează butonul C_2 în timp ce se execută un ciclu inițiat de C_1 , se vor executa două cicluri.

b) Codificarea stărilor

Cele 7 stări ale diagramei din figura 5.4 pot fi codificate folosind 3 variabile de stare. Această codificare ar conduce însă la apariția curselor critice deoarece nu toate stările între care au loc tranziții ar putea fi codificate adiacent.

Pentru a codifica toate stările între care au loc tranziții cu coduri adiacente trebuie să introducă câteva stări suplimentare, ceea ce conduce la necesitatea utilizării a 4 variabile de stare. Diagrama de stare obținută împreună cu codurile alese este reprezentată în figura 5.5.

Observație: trebuie menționat că problema codificării stărilor circuitelor secvențiale în general (asincrone și sincrone) nu are soluție unică și că nu există o metodă care să garanteze o soluție optimă.

În cazul de față aceste folosim pentru variabilele de stare diagrame Veitch de 8 variabile ($x_1, x_2, x_3, x_4, LD, LS, C_1, C_2$), deci cu 256 celule. Lucrul cu aceste diagrame ar fi foarte dificil.

O metodă care conduce la utilizarea unor diagrame cu dimensiuni mai mici constă în introducerea variabilelor în interiorul tabelelor. Diagramele astfel obținute se numesc diagrame VID (variabile incluse în diagramă). Pentru o diagramă Veitch cu k variabile numărul celulelor este 2^k ; prin includerea în diagramă a “p” variabile de intrare, dimensiunea diagramei se reduce, numărul celulelor devenind 2^{k-p} .

x_4			
M_1	M_2	-	M_{11}
M_9	M_3	-	M_7
M_5	M_{10}	-	M_6
M_4	M_8	-	-

x_1

x_3

x_2

Figura 5.36

x_4			
$LS \cdot C_1$	C_2	*	0
0	C_2	*	0
1	1	*	LD
1	1	*	*

x_1

x_3

x_4			
0	LD	*	0
0	1	*	LS
1	1	*	1
LD	0	*	*

x_1

x_3

x_2

		x_4				
$LS \cdot C_1$		1	*	0		
0		\bar{LS}	*	0		
x_1	0	0	*	0	x_2	
	0	0	*	*		
		x_3				

		x_4					
$LS \cdot C_1$		0	0	*	0		
0		0	0	*	1		
x_1	LS	0	0	*	1	x_2	
	0	0	*	*	*		
		x_3					

Figura 5.7

Pentru exemplul dat variabilele de intrare LS, LD, C_1 și C_2 se includ în diagrame rezultând diagrame VID de 4 variabile. Pentru întocmirea diagramelor VID este mai ușor dacă se utilizează matricea stărilor. Matricea stărilor este o diagramă care indică modul cum sunt codificate stările (fig. 5.6). Pe baza matricei stărilor și a analizei tranzițiilor din diagrama stărilor se construiesc diagramele VID ale stărilor următoare (fig. 5.7).

În continuare se descrie modul de completare a diagramelor VID pentru starea M_1 (pentru implementare cu porți logice).

Din starea M_1 se trece, cu semnalul $LS \cdot C_1$ în starea M_2 , iar cu semnalul $LS \cdot C_2$ în starea M_4 . În lipsa acestor semnale circuitul rămâne în starea M_1 .

În continuare se analizează fiecare variabilă de stare:

- variabila de stare x_1 devine 1 dacă se aplică $LS \cdot C_2$ și rămâne 0 în toate celelalte cazuri, deci expresia valorii viitoare a variabilei de stare x_1 este $LS \cdot C_2$, expresie care se introduce în celula stării M_1 pentru x_1 .
- variabila de stare x_2 rămâne pe 0 la trecerea din M_1 în M_2 sau M_4 . Punem 0 în celula M_1 pentru x_2 .
- variabila de stare x_3 devine 1 dacă se aplică $LS \cdot C_1$ și rămâne 0 în caz contrar. Se trece $LS \cdot C_1$ în celula M_1 pentru x_3 .
- variabila de stare x_4 rămâne 0 la trecerea din M_1 în M_2 sau M_4 . Se pune 0 în celula M_1 pentru x_4 .

Din diagrama VID se scriu ecuațiile de stare utilizând următoarele reguli (pentru forma normală disjunctivă):

- celulele cu 1 logic și * se grupează ca pentru o diagramă Veitch obișnuită.
- se consideră pe rând expresiile logice conținute în celulele diagramei. Acestea sunt funcții de variabilele de intrare. Celulele care conțin aceeași expresie se grupează cu celule care conțin 1 sau * și cu cele ale căror expresii iau valoarea logică 1 atunci când expresia considerată ia valoarea 1.
- se face SAU logic între termenii astfel obținuti.

$$\begin{aligned}
 X_1 &= x_1 \cdot \bar{x}_4 + C_2 \cdot x_3 + LS \cdot C_2 \cdot \bar{x}_2 \cdot \bar{x}_4 + LD \cdot x_1 \\
 X_2 &= x_1 \cdot x_2 + x_2 \cdot x_3 + LD \cdot x_1 \cdot \bar{x}_3 + LD \cdot \bar{x}_1 \cdot x_3 + \bar{LS} \cdot x_2 \cdot x_4 \\
 X_3 &= \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + LS \cdot C_1 \cdot \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4 + \bar{LS} \cdot \bar{x}_1 \cdot x_3 \\
 X_4 &= x_2 \cdot x_4 + LS \cdot x_1 \cdot x_2 \cdot \bar{x}_3
 \end{aligned} \tag{5.2}$$

Aplicând aceste reguli se obțin ecuațiile 5.2. Variabilele de ieșire MS și MD depind numai de starea circuitului secvențial. Expresiile lor se obțin din diagramele Veitch obișnuite (fig. 5.8). Pentru regimul automat ieșirile sunt:

		x_4	
		x_2	x_3
x_1			
0	1	*	0
0	0	*	0
0	0	*	1
1	1	*	*

		x_4	
		x_2	x_3
x_1			
0	0	*	0
0	1	*	1
1	1	*	0
0	0	*	*

Figura 5.8

Schema circuitului secvențial asincron realizat cu porți logice este dată în figurile 5.9 și 5.10.

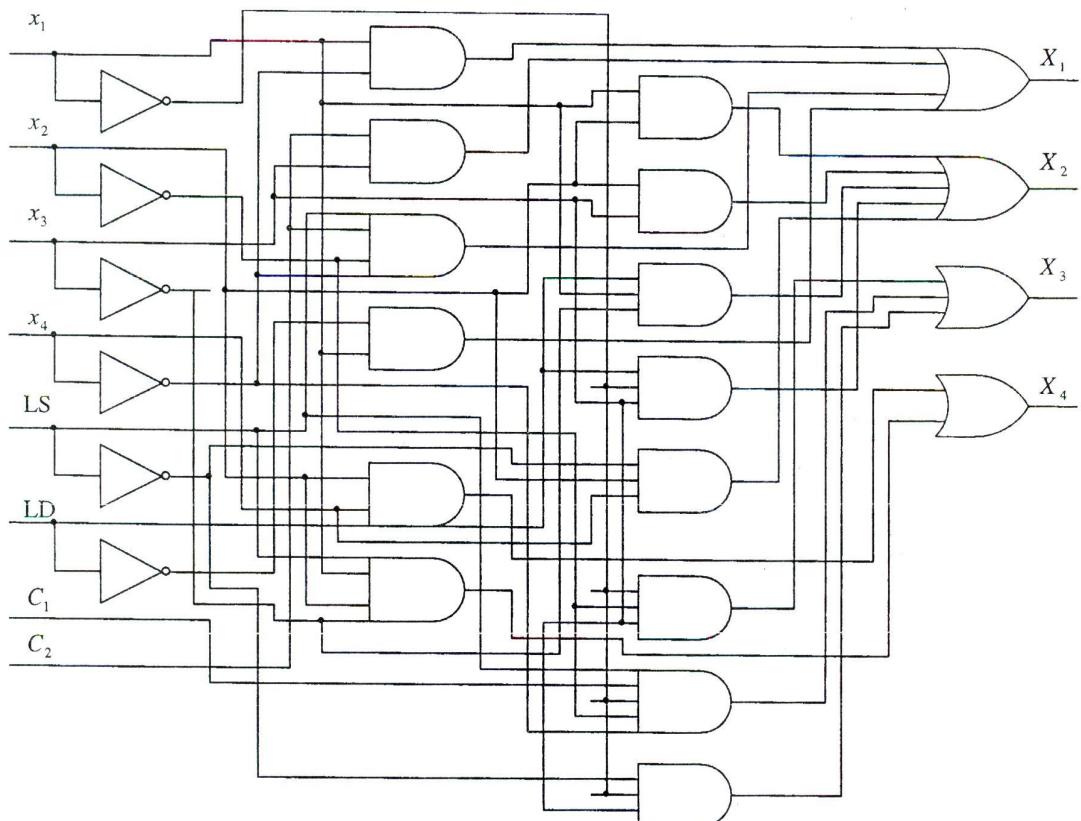


Figura 5.9

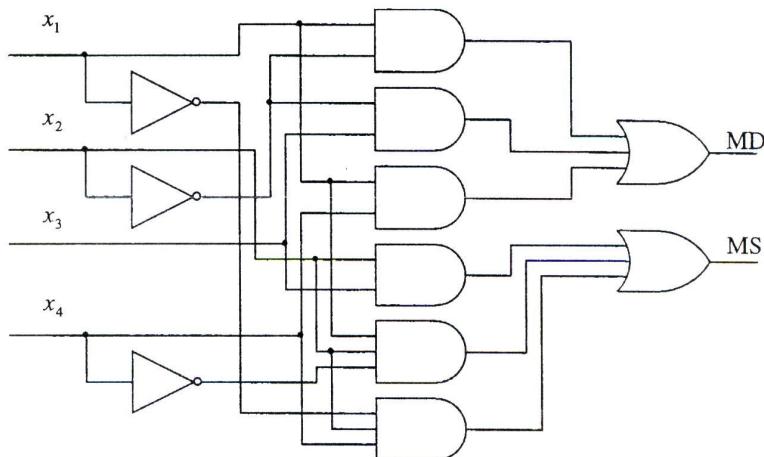


Figura 5.10

5.1.3. Implementarea circuitelor secvențiale sincrone cu bistabile și porti

La codificarea stărilor circuitelor secvențiale sincrone se pot urma aceleași reguli ca și la circuitele secvențiale asincrone. Două stări între care are loc o tranziție provocată de o intrare asincronă vor fi codificate adiacent. Prin intrare asincronă se înțelege o intrare care își poate modifica valoarea în orice moment de timp. O intrare este sincronă dacă își modifică valoarea doar la aplicarea impulsului de sincronizare (pe unul din fronturi). Circuitele de memorie de tip sincron (bistabile de tip D sau J-K) impun anumite restricții semnalelor aplicate la intrări pentru a avea garanția că se produce o singură schimbare de stare. De aceea semnalele asincrone trebuie să condiționeze schimbarea valorii unei singure variabile de stare. În caz contrar se poate produce o tranziție într-o altă stare decât cea dorită datorită timpilor de răspuns diferiți. În cazul în care diagrama de stare nu permite o codificare care să respecte condiția de mai sus se sincronizează semnalul de intrare.

Toate semnalele LS, LD, C_1 și C_2 sunt asincrone. Condiția ca în fiecare stare o variabilă de intrare să condiționeze schimbarea valorii unei singure variabile de stare este respectată dacă toate stările între care au loc tranziții se codifică adiacent. Se poate deci utiliza codificarea de la sinteza circuitului secvențial asincron.

Sinteza circuitului secvențial sincron diferă în acest caz doar la etapa implementării. Se vor folosi bistabile de tip J-K. În acest scop se completează diagramele VID pentru funcțiile de excitație. Acestea se obțin din matricea stărilor, diagrama stărilor și tabelul excitațiilor bistabilelor J-K (tabelul 5.1). Pentru simplificare în continuare se folosesc următoarele notări: $A = x_1$; $B = x_2$; $C = x_3$; $D = x_4$.

Q_n	Q_{n+1}	Acțiunea circ.	Simbol	Funcții de excitație ale bistabilelor		
				J	K	D
0	0	$0 \rightarrow 0$	0	0	*	0
0	1	$0 \rightarrow 1$	I	1	*	1
1	0	$1 \rightarrow 0$	0	*	1	0
1	1	$1 \rightarrow 1$	1	*	0	1

Tabelul 5.1

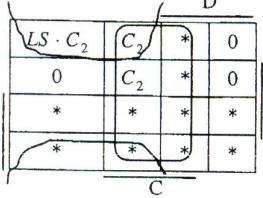
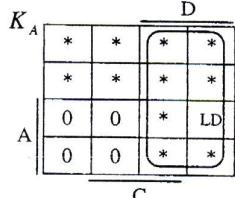
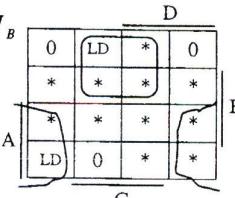
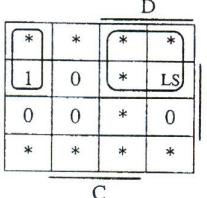
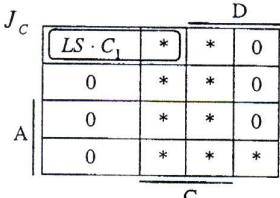
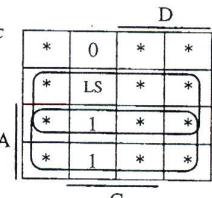
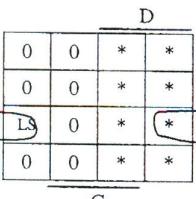
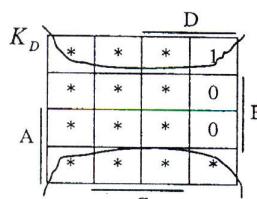
J_A		K_A		J_B	
K_B		J_C		K_C	
J_D		K_D			

Figura 5.11

$$J_A = C_2 \cdot C + LS \cdot C_2 \cdot \bar{B} \cdot \bar{D}$$

$$K_A = LD \cdot D$$

$$J_B = LD \cdot A \cdot \bar{C} + LD \cdot \bar{A} \cdot C$$

$$K_B = LS \cdot \bar{A} \cdot D + \bar{A} \cdot \bar{C} \cdot \bar{D} \quad (5.3)$$

$$J_C = LS \cdot C_1 \cdot \bar{A} \cdot \bar{B} \cdot \bar{D}$$

$$K_C = A + LS \cdot B$$

$$J_D = LS \cdot A \cdot B \cdot \bar{C}$$

$$K_D = \bar{B}$$

Variabilele de ieșire MS și MD depind numai de starea circuitului secvențial. Expresiile lor au fost obținute în exemplul anterior.

$$MD = A \cdot \bar{B} + \bar{B} \cdot C + A \cdot D$$

$$(5.4)$$

$$MS = B \cdot C + A \cdot B \cdot \bar{D} + \bar{A} \cdot B \cdot D$$

Schema electrică principală a automatului este prezentată în figura 5.12.

5.1.4. Sinteză circuitelor secvențiale sincrone cu sincronizarea unor intrări asincrone

Restricția ca două stări între care au loc tranziții să fie codificate adiacent nu este necesară dacă semnalele care provoacă tranziția sunt sincrone. Sincronizarea semnalelor asincrone se realizează simplu cu un bistabil de tip D sau cu un bistabil J-K transformat într-un bistabil de tip D (figura 5.13 și 5.14).

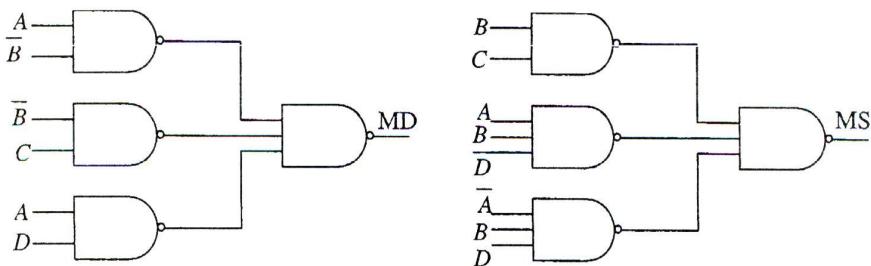
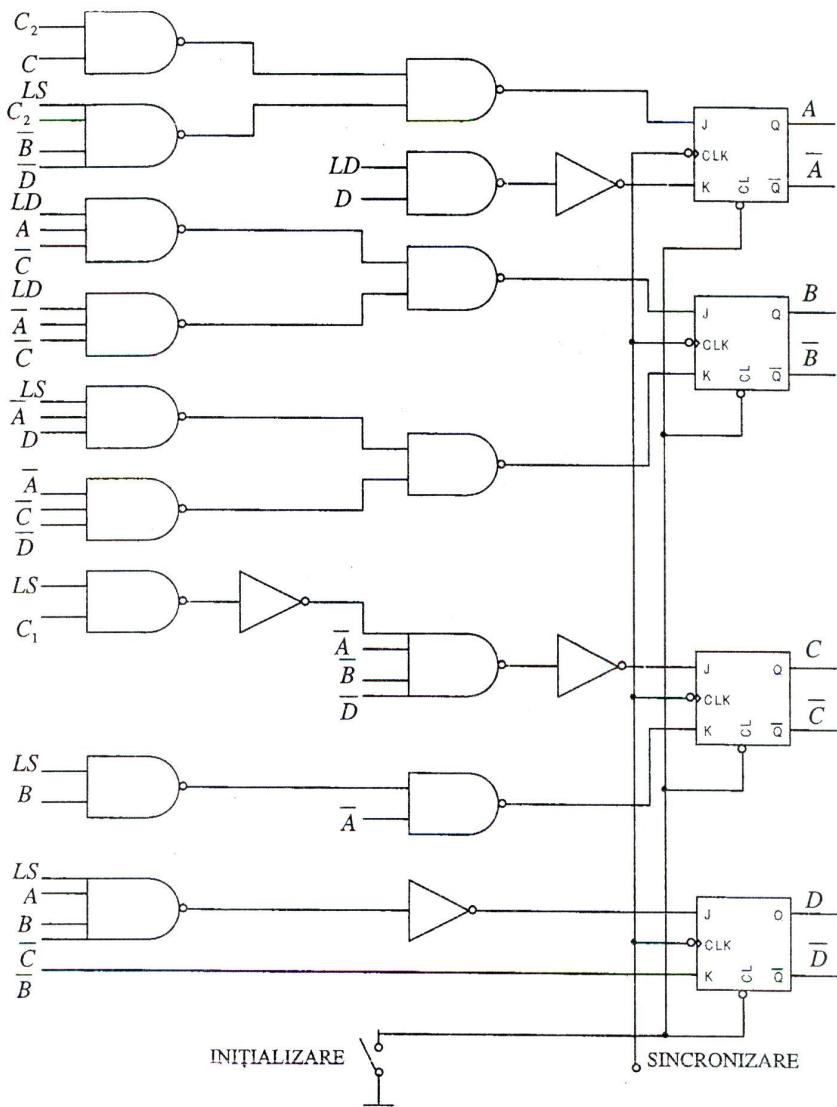


Figura 5.12

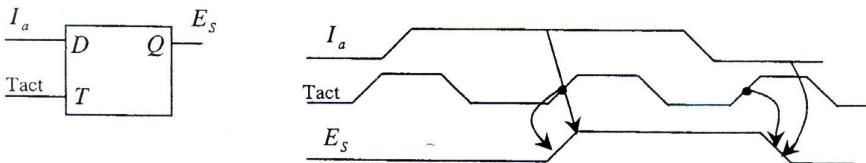


Figura 5.13

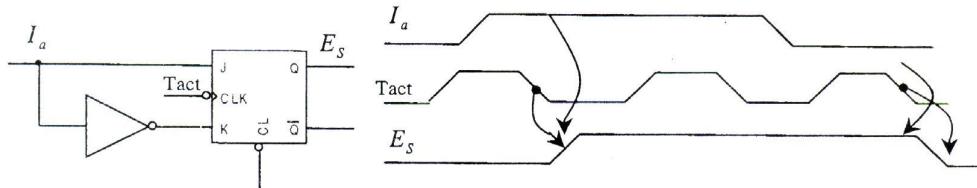


Figura 5.14

Folosind aceste observații putem reduce destul de mult schemele de implementare.

5.1.5. Utilizarea multiplexoarelor și a numărătoarelor la sinteza circuitelor secvențiale sincrone

Utilizarea numărătoarelor cu încărcare paralelă și a multiplexoarelor simplifică și mai mult sinteza circuitelor secvențiale sincrone. Multiplexoarele folosite vor fi de tipul SN74151. Numărătorul sincron cu încărcare paralelă SN74163 va fi descris în cele ce urmează. Simbolul de reprezentare și notațiile terminalelor sunt date în figura 5.15.

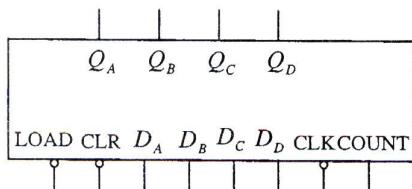


Figura 5.15

Dacă la terminalul de încărcare LOAD se aplică nivel logic scăzut în numărător se introduc sincron (la frontul descrescător al impulsului de tact CLK) datele aplicate la intrările D_A , D_B , D_C și D_D .

Pentru adunarea pe zero a celor patru ieșiri Q_A , Q_B , Q_C și Q_D se aplică semană logic 0 la intrarea de initializare CLR.

Dacă la intrarea de numărare COUNT se aplică semnal logic 1 conținutul numărătorului se incrementează pe frontul descrescător al impulsului de tact.

Pentru ca numărătoarele binare sincrone cu încărcare paralelă să poată fi utilizate la memorarea stărilor unui circuit secvențial se impune condiția ca diagrama stărilor să nu conțină mai mult de două ramificații, condiție ce poate fi respectată prin introducerea unor stări suplimentare (fig. 5.16).

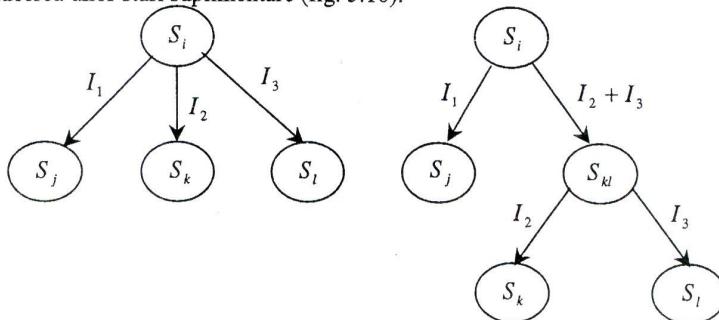


Figura 5.16

Codificarea stărilor se face astfel încât o stare următoare se obține prin incrementarea conținutului numărătorului (COUNT=1), iar cealaltă stare următoare prin încărcare paralelă (LOAD=0).

Reluând exemplul dat la începutul acestui capitol se consideră diagrama stărilor din figura 5.17.

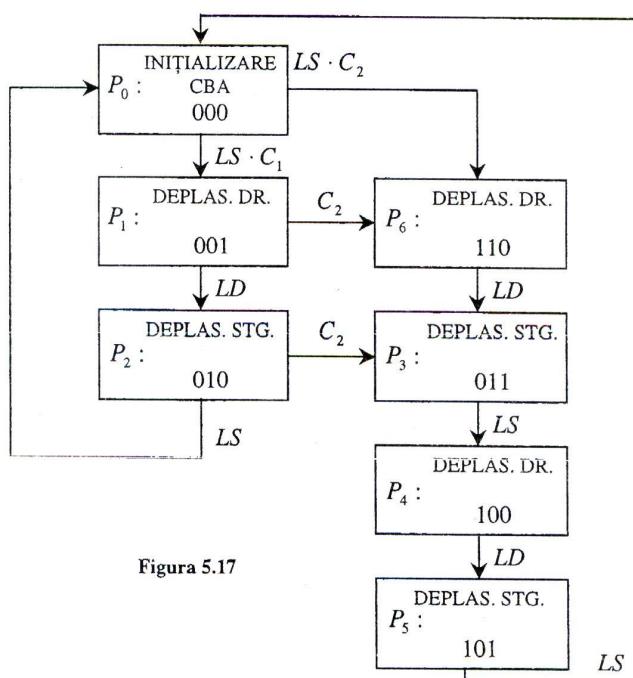


Figura 5.17

Nu sunt necesare stări suplimentare deoarece diagrama nu conține mai mult de două ramificații.

Semnalele care se aplică la intrările de încărcare (LOAD) și numărare (COUNT) vor fi sincronizate, prin aceasta sincronizându-se de fapt toate semnalele de intrare. Deci nu se mai pune problema codificării adiacente a stărilor între care au loc tranziții provocate de intrări asincrone.

Având codificată diagrama stărilor se scriu funcțiile logice care dău semnalele ce se aplică la intrările numărătorului.

- COUNT – condițiile necesare pentru efectuarea tranzitiei conform secvenței principale de numărare.
- LOAD – condițiile necesare pentru efectuarea salturilor, adică a tranzitiei în afara secvenței principale de numărare.
- D_A, D_B, D_C, D_D - codul stării la care se face saltul realizat cu ajutorul stării din care se face saltul.
- CLR – condițiile pentru inițializare.

În exemplul dat secvența principală de numărare este $P_0, P_1, P_2, P_3, P_4, P_5$. Salturi există între stările $P_0 \rightarrow P_6$; $P_1 \rightarrow P_6$; $P_2 \rightarrow P_0$; $P_5 \rightarrow P_0$; $P_6 \rightarrow P_3$.

Scrierea funcțiilor logice se poate face direct din diagrama stărilor.

Numărătorul trebuie incrementat atunci când trebuie să se efectueze o tranziție din secvența principală de numărare. Acest tip de tranziții se efectuează în starea P_0 dacă se aplică $LS \cdot C_1 = 1$, în starea P_1 dacă $LD=1$, în starea P_2 dacă $C_2 = 1$, în starea P_3 dacă $LS=1$, în starea P_4 dacă $LD=1$.

$$COUNT = P_0 \cdot LS \cdot C_1 + P_1 \cdot LD + P_2 \cdot C_2 + P_3 \cdot LS + P_4 \cdot LD \quad (5.5)$$

Numărătorul trebuie încărcat în paralel atunci când trebuie să se efectueze tranziții care nu sunt codificate în secvență binară. Acest tip de tranziții se efectuează în starea P_0 dacă $LS \cdot C_2 = 1$, în starea P_1 dacă $C_2 = 1$, în starea P_2 dacă $LS=1$, în starea P_5 dacă $LS=1$ și în starea P_6 dacă $LD=1$.

$$LOAD = P_0 \cdot LS \cdot C_2 + P_1 \cdot C_2 + P_2 \cdot LS + P_5 \cdot LS + P_6 \cdot LD \quad (5.6)$$

Intrările de date trebuie să conțină codul stării la care se efectuează saltul. În starea P_0 intrările D_C , D_B și D_A trebuie să asigure tranziția în starea P_6 , adică să fie 110. Similar în P_1 trebuie să fie 110 (P_6), în P_2 să fie 000 (P_0), în P_5 să fie 000 (P_0) și în P_6 să fie 011 (P_3).

$$\begin{aligned} D_A &= P_6 \\ D_B &= P_0 + P_1 + P_6 \\ D_C &= P_0 + P_1 \end{aligned} \quad (5.10)$$

Funcțiile logice mai complexe cum sunt LOAD și COUNT pot fi realizate cu multiplexoare. Variabilele de stare A, B, C se folosesc drept variabile de adresare, iar

intrările drept intrări de date. Pentru obținerea funcțiilor D_A , D_B și D_C se poate folosi un decodificator. Schema logică a circuitului este prezentată în continuare.

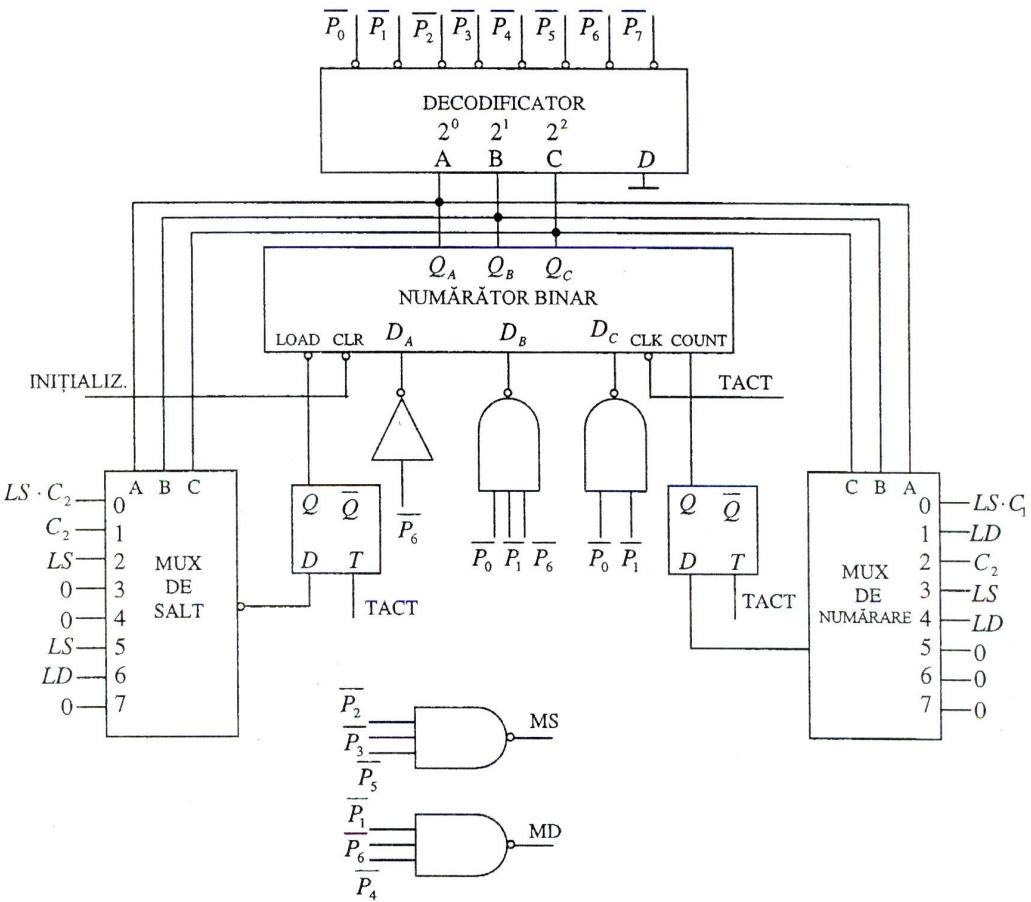


Figura 5.18

Observație: Starea P_7 nu este folosită și dacă circuitul ajunge în mod accidental în ea el rămâne "agățat" în P_7 . Scoaterea lui din P_7 nu se poate face decât prin inițializare. O soluție des folosită constă în utilizarea stărilor nefolosite la generarea semnalului de inițializare. Pentru exemplul dat ecuația semnalului de inițializare este:

$$\overline{CLR} = \overline{P}_7 \cdot \overline{INIT} \quad (5.11)$$

unde $INIT$ este semnalul de inițializare manuală.

5.1.6. Utilizarea memoriilor fixe programabile (PROM) la sinteza circuitelor secvențiale sincrone

Utilizarea memoriilor fixe programabile (PROM) împreună cu alte circuite integrate (registre, numărătoare, multiplexoare, etc.) permite realizarea unor circuite secvențiale sincrone caracterizate printr-o flexibilitate sporită.

În continuare vor fi prezentate două structuri de circuite secvențiale sincrone realizate cu memorii programabile.

Observație: Indiferent de structura adoptată memoria PROM îndeplinește, în general, următoarea funcție: aplicându-se la intrările de adresare ale memoriei codul stării prezente, la ieșiri va apărea codul stării următoare ce se obține prin salt și, eventual, funcțiile de ieșire.

În figura 5.19 este prezentată schema logică a circuitului secvențial de comandă a deplasării căruciorului realizat cu numărător, multiplexoare și memorie PROM. Acest circuit lucrează după diagrama stărilor din figura 5.17 și a fost obținut prin înlocuirea decodificatorului și a porților logice care generează funcțiile D_A , D_B , D_C , MS și MD ale automatului din figura 5.18 cu o memorie PROM. Semnalele care se aplică la intrările de date ale multiplexoarelor sunt identice pentru circuitele secvențiale din figurile 5.18 și 5.19.

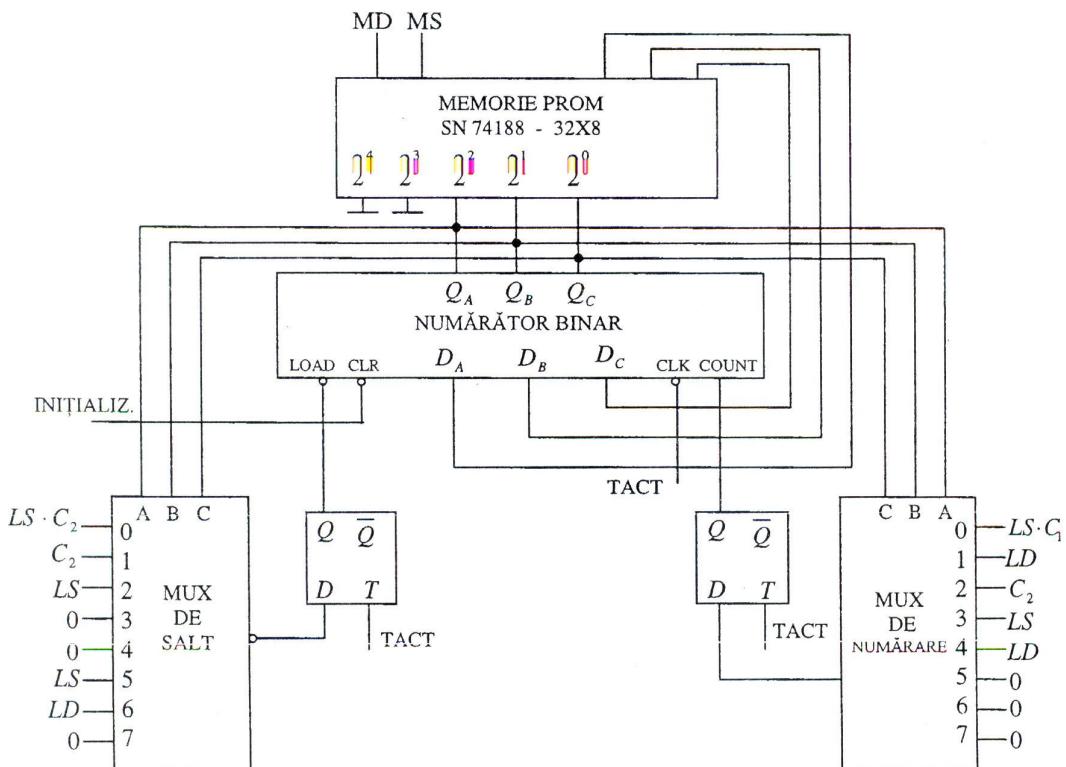


Figura 5.19

Observație: Întrucât diagrama stărilor din figura 5.17 are numai 7 stări codificate cu 3 variabile de stare memoria cea mai mică de care ar fi nevoie ar trebui

să aibă 8 cuvinte de 3 biți. Întrucât nu se produc memorii cu o capacitate atât de mică se poate folosi memoria SN74188 care este organizată în 32 cuvinte de 8 biți. O astfel de memorie poate genera atât starea următoare (3 biți) cât și funcțiile de ieșire MS și MD (2 biți).

Intrările de adresare ale memoriei cu ponderile 8 și 16 sunt legate la masă deci se pot adresa numai primele 8 cuvinte din memorie.

În exemplul dat memoria este incomplet folosită. Folosirea incompletă a memoriei este un fenomen obișnuit datorat numărului limitat de tipuri de memorie produse în serie.

Programarea memoriei este dată în tabelul următor în care s-a notat cu * conținutul celulelor neutilizate. Tabelul se completează direct din diagrama stărilor. Pentru fiecare stare, la adresa selectată de codul acesta se memorează codul stării următoare care se obține prin salt (deci nu prin incrementarea numărătorului).

ADRESA				CONTINUTUL					
-	-	-	-	MD	MS	C	B	A	
0	0	0	0	x	x	x	0	0	1
0	0	0	1	x	x	x	1	0	1
0	0	1	0	x	x	x	0	1	0
0	0	1	1	x	x	x	0	1	x
0	1	0	0	x	x	x	1	0	x
0	1	0	1	x	x	x	0	1	0
0	1	1	0	x	x	x	1	0	0
0	1	1	1	x	x	x	x	x	x

Tabelul 5.2