7 SISTEME SECVENȚIALE SINCRONE

7.1 Registre

Registrele sunt structuri numerice formate din bistabile care comută sincron (pe intrările de ceas ale lor se aplică acelaşi semnal de CLOCK) şi care au un scop comun. Scopul este fie acela de a deplasa biții de date într-o anumită direcție, fie de a-i memora temporar între două fronturi ale ceasului. Deplasarea datelor se poate face de la stânga la dreapta, de la dreapta la stânga, sau bidirecțional, iar registrele care fac aceste operații se numesc **registre de deplasare** sau **registre serie**. Memorarea datelor se face cu **registre de stocare** sau **registre paralel**.

Un registru de 3 biți pentru deplasarea spre dreapta a datelor este reprezentat în figura 7.1. Pe frontul activ al ceasului, valoarea logică de pe intrarea A este memorată în bistabilul 1 şi apare la ieşirea Q_1, valoarea anterioară a ieşirii Q_1 apare la ieşirea Q_2, iar valoarea anterioară a ieşirii Q_2 apare la ieşirea Q_3. Valoarea anterioară a ieşirii Q_3 se pierde.

Tabelul tranzițiilor din figura 7.2 ilustrează toate tranzițiile posibile care au loc în circuit. Ansamblul celor 3 coloane din stânga tabelului indică fiecare dintre cele 8 stări prezente posibile $Q_1Q_2Q_3$, iar celelalte coloane indică tranziția în starea următoare a sistemului $Q_1^+Q_2^+Q_3^+$, în cele două situații: când intrarea A este 0 logic sau 1 logic.

Diagrama stărilor este reprezentată în figura 7.3. Fiecare nod al grafului, reprezentat printr-un cerc cu eticheta $Q_1Q_2Q_3$, este o stare a sistemului, iar cele 16 arce orientate,

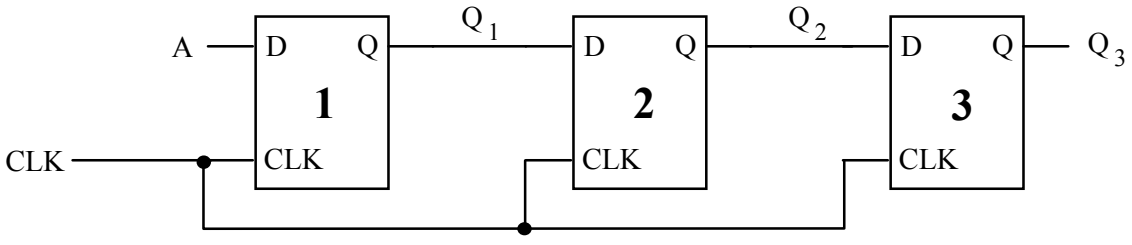


Fig. 7.1 *Registru de deplasare spre dreapta de 3 biți*

Q_1	Q_2	Q_3	Q_1^+ Q_2^+ Q_3^+					
			A = 0			A = 1		
0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	1	1	0	1
0	1	1	0	0	1	1	0	1
1	0	0	0	1	0	1	1	0
1	0	1	0	1	0	1	1	0
1	1	0	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1

Fig. 7.2 *Tabelul tranzițiilor pentru registrul serie de 3 biți*

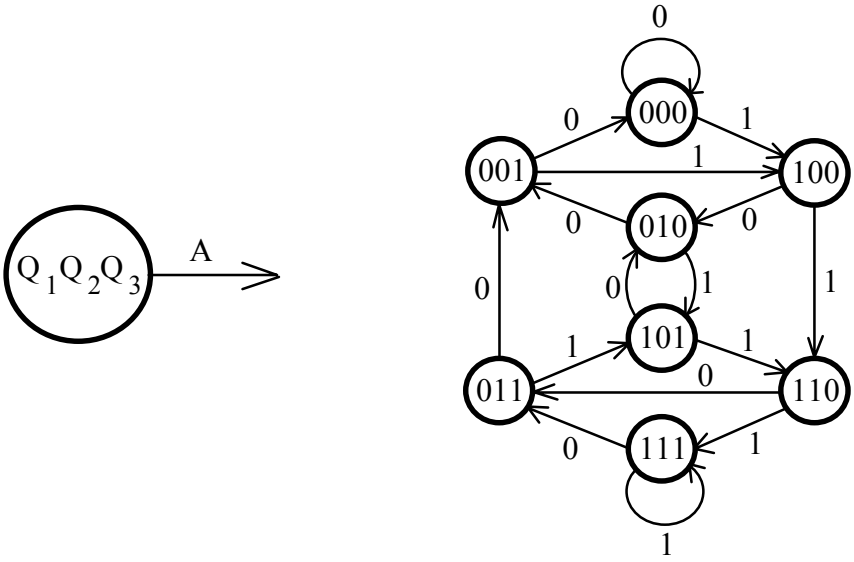


Fig. 7.3 *Diagrama stărilor pentru registrul serie de 3 biți*

etichetate cu valoarea logică a intrării A, sugerează toate tranzițiile posibile de la o stare la alta a circuitului.

Registrul paralel realizează "stocarea temporară a unor configurații binare într-o zonă "uşor" accesibilă prelucrării. Registrul este o memorie al cărei conținut are o dinamică foarte puternică. Registrul este memoria zonelor de viteză maximă dintr-un sistem digital de prelucrare" ([Ştefan, 1984]).

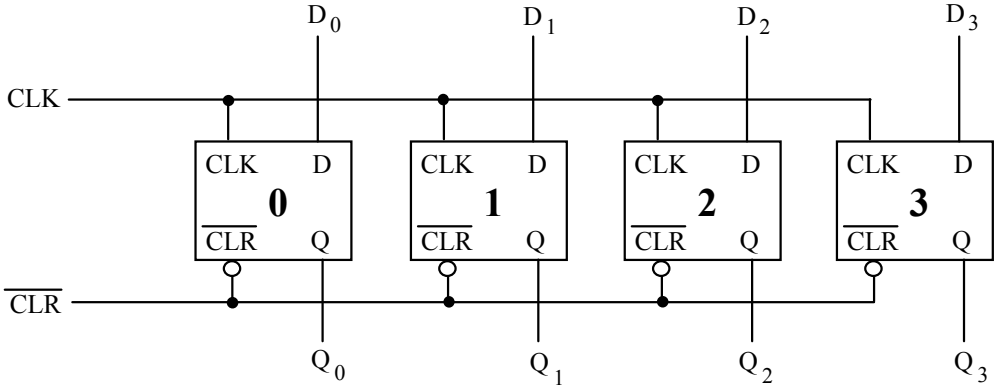


Fig. 7.4 *Registru paralel de 4 biți*

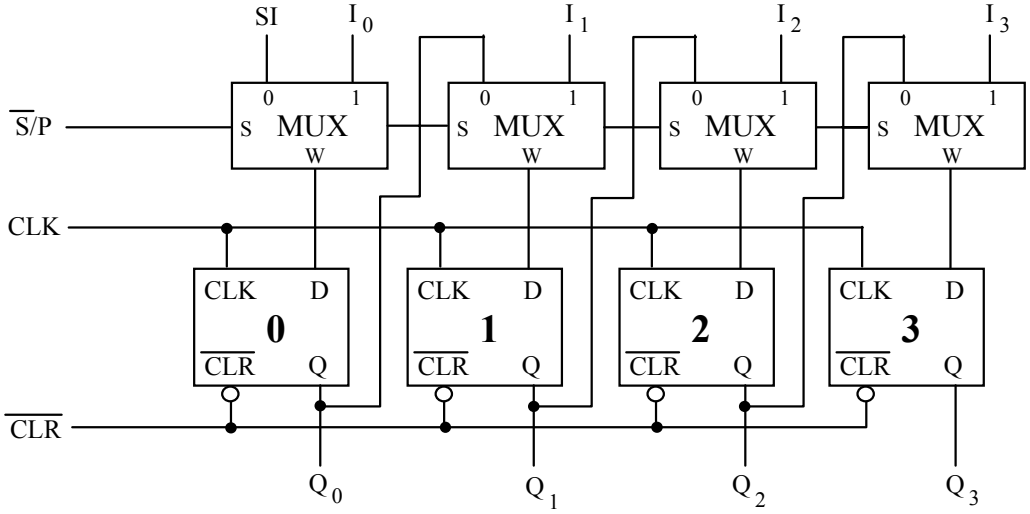


Fig. 7.5 *Registru serie-paralel de 4 biți*

Registrul paralel de 4 biți este reprezentat în figura 7.4. Intrările de date $D_0D_1D_2D_3$ se aplică simultan pe intrările bistabilelor din structură, iar ieşirile bistabilelor $Q_0Q_1Q_2Q_3$ sunt disponibile la ieşirile circuitului. Prin activarea pe 0 logic a intrării asincrone $\overline{CLR}$ (Clear) se resetează toate bistabilele din structură.

Cele două tipuri de registre prezentate sunt utilizabile în aplicații în care transferul datelor se face numai serie, sau numai paralel. De multe ori este necesară însă trecerea de la transferul serie la cel paralel, sau invers. Pentru a rezolva această necesitate a fost conceput **registrul serie-paralel**. Un registru serie-paralel de 4 biți este prezentat în figura 7.5.

Bistabilul de tip D din structura registrului serie-paralel trebuie să primească date la intrare din două surse: bistabilul D anterior, la deplasarea serie, sau exteriorul registrului, la încărcarea paralel. Este deci necesară multiplexarea celor două surse. Dacă intrarea $\overline{S}/P = 0$, atunci intrările notate cu 0 ale multiplexoarelor sunt conectate la ieşirile W: intrarea SI (serial input) se aplică la intrarea primului bistabil D_0; Q_0 se conectează la D_1, Q_1 se conectează la D_2, iar Q_2 se conectează la D_3. Avem o configurație de registru serie cu deplasare spre dreapta. Dacă intrarea $\overline{S}/P = 1$, atunci intrările notate cu 1 ale multiplexoarelor sunt conectate la ieşirile W: I_0 se aplică la intrarea D_0, I_1 se conectează la D_1, I_2 se conectează la D_2, iar I_3 se conectează la D_3. Deci o configurație de registru paralel.

7.2 Numărătoare sincrone

Numărătoarele sincrone sunt structuri numerice care comută sincron şi care parcurg o diagramă a stărilor circulară, adică un drum continuu şi închis printre stările sale. Bistabilele sunt de obicei de tip T, iar intrările lor, care pot da cele două valori logice în funcție de tranziția realizată de sistem , sunt stabilite de o logică combinațională.

Evident că funcția de numărare poate fi realizată şi cu sisteme de ordin inferior, cum ar fi bistabilele de tip D, circuitul având în acest caz o buclă globală introdusă peste elementele de memorie din structură. Funcția de numărare poate fi realizată şi cu sisteme de ordin superior, în situații ce se pot dovedi justificate numai de contexte foarte particulare. De exemplu, un calculator ar putea fi utilizat la executarea unor operații de numărare.

Algoritmul folosit pentru sinteza unui numărător sincron parcurge următoarele etape:

1. se construieşte tabelul tranzițiilor, în care se trece fiecare stare prezentă a numărătorului şi starea următoare care îi corespunde (starea fiecărui bistabil se trece pe câte o coloană pe verticală).
2. se examinează stările fiecărui bistabil i, Q_i şi Q_i^+, şi se trec în tabel, în coloane adiacente, valorile funcțiilor de excitație pentru tipul respectiv de bistabil (de exemplu J_i şi K_i).
3. se obțin ecuațiile algebrice ale funcțiilor de excitație, de obicei prin minimizarea acestor funcții, folosind metodele cunoscute de la implementarea funcțiilor binare.
4. se implementează schema logică a numărătorului, şi, eventual, se verifică prin analiză funcționarea corectă a circuitului obținut.

Exemplul 7.1

Ne propunem să facem sinteza unui numărător sincron modulo 5, care numără crescător. Vom folosi bistabile JK şi pe urmă bistabile D. Pentru numărarea celor 5 stări, de la 0 la 4, sunt necesare 3 bistabile. Diagrama stărilor este prezentată în figura 7.6.

În tabelul tranzițiilor din figura 7.7 s-au trecut stările prezente şi cele viitoare ale sistemului. S-au luat în considerare numai tranzițiile indicate în diagrama stărilor. Celelalte tranziții au fost ignorate, înlocuind valoarea lui Q_i^+ cu simbolul x-don't care. Scopul nostru este să găsim cel mai simplu circuit care realizează tranzițiile indicate. Analiza circuitului sintetizat va permite completarea diagramei stărilor sau a tabelului tranzițiilor cu aceste tranziții, care nu au fost luate în considerare în etapa de sinteză a circuitului.

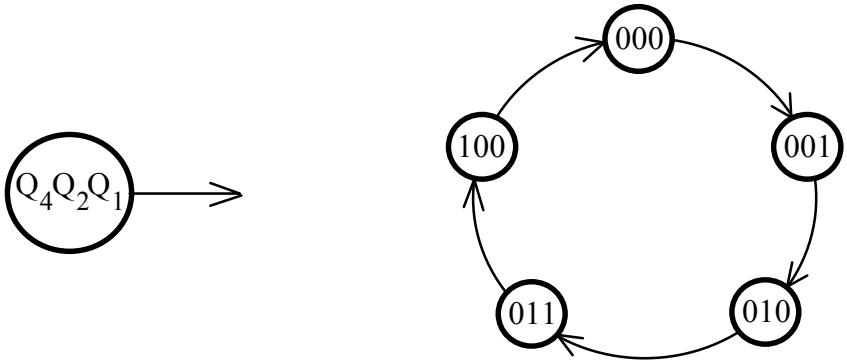


Fig. 7.6 *Diagrama stărilor pentru numărătorul crescător modulo 5*

Q_4	Q_2	Q_1	Q_4^+	Q_2^+	Q_1^+	J_4	K_4	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	0	x	1	0	x	0	x
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

Fig. 7.7 *Tabelul tranzițiilor pentru numărătorul crescător modulo 5*

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

⟶

Q	Q^+	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Fig. 7.8 *Cele două reprezentări pentru tabelul de tranziții al bistabilului de tip JK*

Coloanele funcțiilor de excitație J_i şi K_i au fost completate cu ajutorul tabelului de tranziții al bistabilului JK din figura 6.13. El a fost scris sub o altă formă, în figura 7.8.

Explorând valorile funcțiilor J_i şi K_i din tabel (i = 1, 2, 4), constatăm că putem scrie $K_4 = K_1 = 1$. Pentru minimizarea celor 4 funcții rămase se construiesc diagramele Veitch-Karnaugh şi rezultă ecuațiile algebrice din figura 7.9. Cu ajutorul ecuațiilor obținute se implementează schema logică a circuitului, care este reprezentată în figura 7.10.

Analiza circuitului se poate face cu ajutorul schemei logice şi a tabelului din figura 7.8, sau cu ajutorul ecuației de stare a bistabilului JK dedusă în paragraful 6.5.

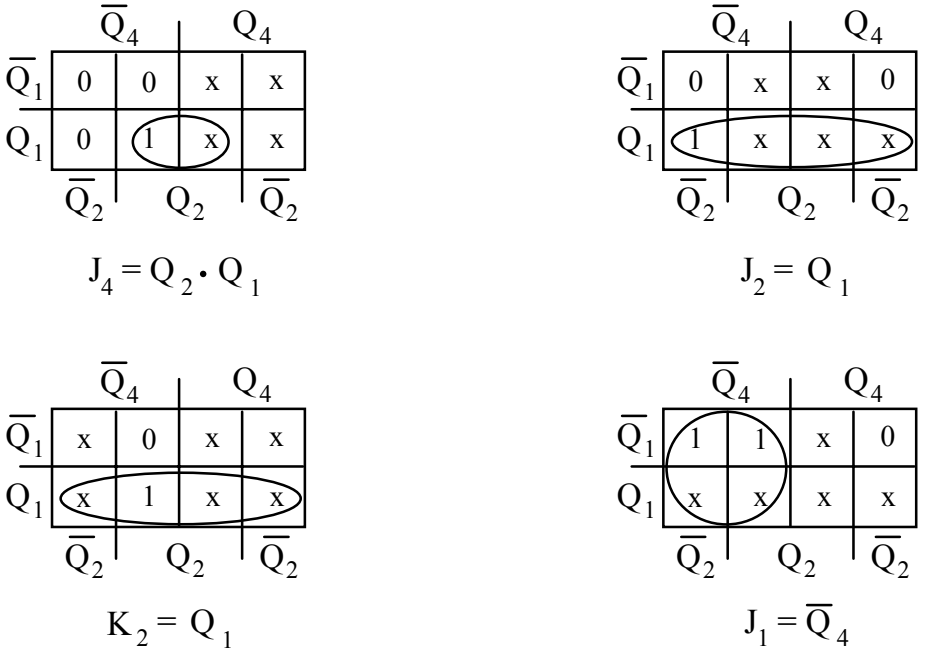


Fig. 7.9 *Minimizarea funcțiilor de excitație pentru bistabilele JK*

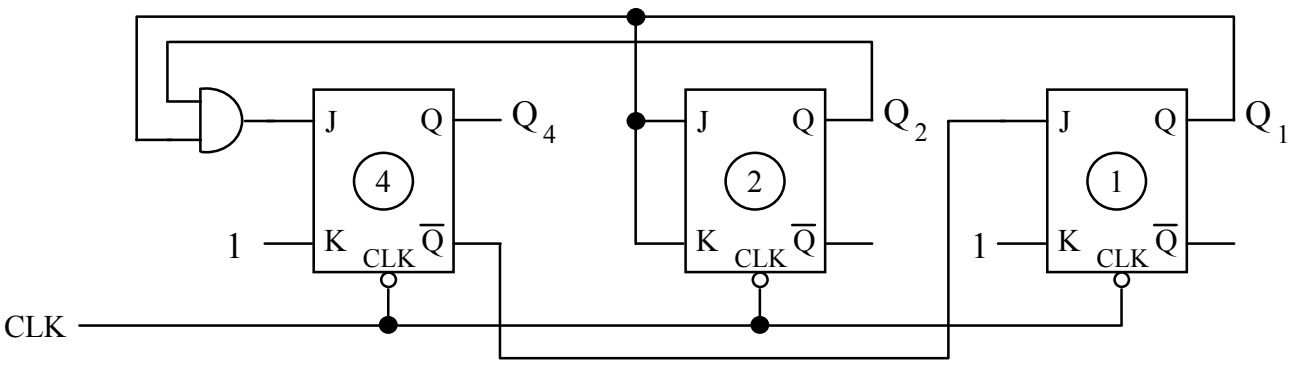


Fig. 7.10 *Schema logică a numărătorului sincron modulo 5 cu bistabile JK*

Considerând că starea prezentă a circuitului din figura 7.10 este $Q_4Q_2Q_1$ = 101, atunci J_4 = 0, K_4 = 1 şi conform tabelului lui JK, Q_4^+ = 0. În aceeaşi stare prezentă avem $Q_4Q_2Q_1$ = 101, J_2 = 1 şi K_2 = 1, deci Q_2^+ = 1, adică complementul stării prezente. În sfârşit, ultimul bistabil are intrările J_1 = 0 şi K_1 = 1, şi starea lui următoare este Q_1^+ = 0.

Să verificăm tranziția de mai sus şi cu ajutorul ecuației de stare a bistabilului JK: $Q_i^+ = J_i \cdot \overline{Q_i} + \overline{K}_i \cdot Q_i$. Pentru bistabilul 4 putem scrie: $Q_4^+ = J_4 \cdot \overline{Q_4} + \overline{K}_4 \cdot Q_4 = Q_2 \cdot Q_1 \cdot \overline{Q_4}$. Dacă introducem valorile stării prezente $Q_4Q_2Q_1$ = 101, rezultă Q_4^+ = 0. Pentru al doilea bistabil $Q_2^+ = J_2 \cdot \overline{Q_2} + \overline{K}_2 \cdot Q_2 = Q_1 \cdot \overline{Q_2} + \overline{Q_1} \cdot Q_2 = Q_1 \oplus Q_2$. Rezultă de aici Q_2^+ = 1. În sfârşit, $Q_1^+ = J_1 \cdot \overline{Q_1} + \overline{K}_1 \cdot Q_1 = \overline{Q_4} \cdot \overline{Q_1}$, deci starea lui următoare este Q_1^+ = 0. Diagrama completă a stărilor este reprezentată în figura 7.11.

Schema logică din figura 7.12 reprezintă aceeaşi structură implementată cu bistabile de tip D. Funcțiile D_4, D_2 şi D_1 se obțin prin minimizarea funcțiilor Q_4^+, Q_2^+ şi respectiv Q_1^+ din tabelul dat în figura 7.7, ecuația de stare a bistabilului D fiind $Q_i^+ = D_i$. Se poate observa că circuitul combinațional rezultat este implementat cu număr minim de porți şi este mai complicat decât cel găsit la sinteza cu bistabile de tip JK. Acest lucru era de aşteptat, deoarece gradul de structurare superior al bistabilului JK simplifică logica combinațională necesară pentru efectuarea tranzițiilor impuse. Lăsăm în seama cititorului efectuarea analizei circuitului şi completarea diagramei stărilor.

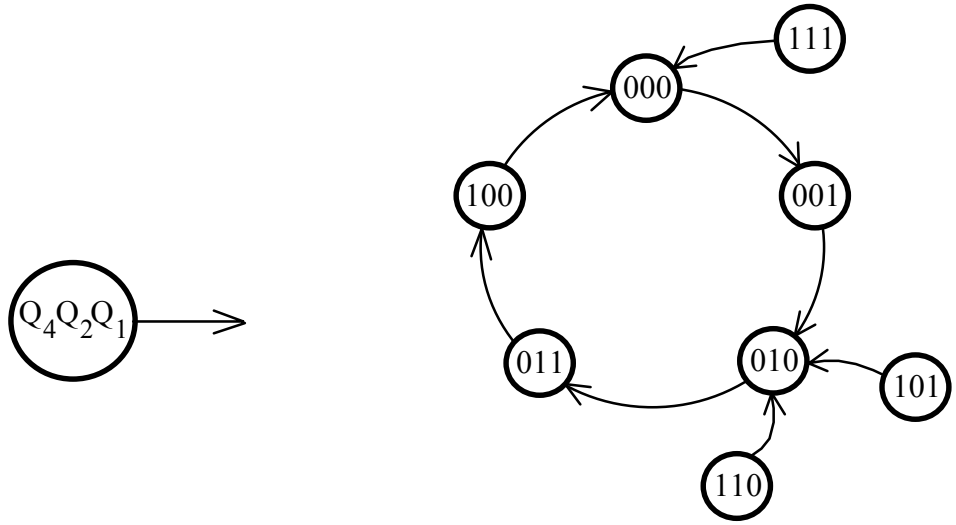


Fig. 7.11 *Analiza completă a circuitului oferă toate tranzițiile posibile*

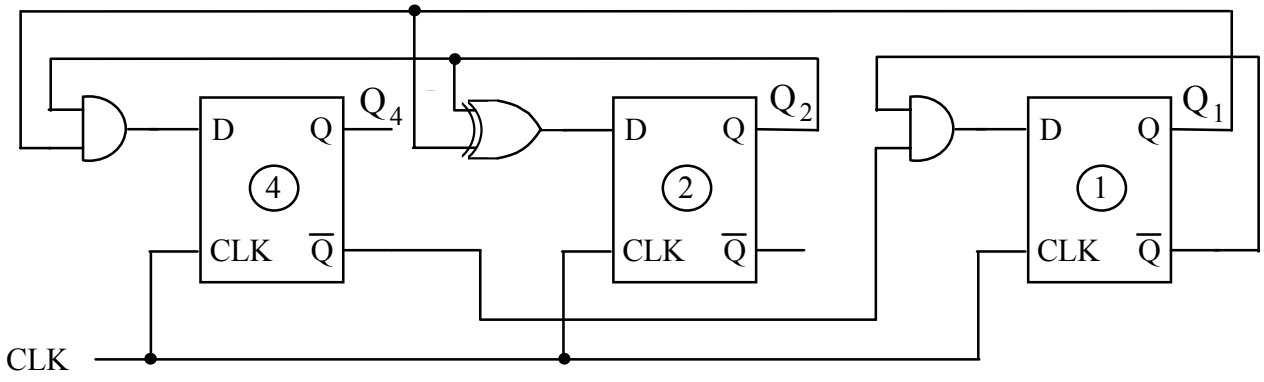


Fig. 7.12 *Schema logică a numărătorului sincron modulo 5 cu bistabile D*

7.3 Automate cu stări finite

Registrele, numărătoarele şi chiar simplul bistabil de tip JK sunt **automate cu stări finite**. Un automat cu stări finite se defineşte formal prin cvintuplul $A=(X,Y,Q,\delta,\lambda)$, unde semnificaţia celor cinci elemente este următoarea:

- $X=\{x_1,x_2,...,x_n\}$ reprezintă mulţimea finită a configuraţiilor binare de intrare,
- $Y=\{y_1,y_2,...,y_r\}$ reprezintă mulţimea finită a configuraţiilor binare de ieşire,
- $Q=\{q_1,q_2,...,q_p\}$ reprezintă mulţimea finită a configuraţiilor binare de stare,
- $\delta: X\times Q\to Q$ este funcţia de tranziţie a stărilor,
- $\lambda: X\times Q\to Y$ este funcţia de tranziţie a ieşirilor.

Bistabilul de tip JK are 2 intrări de date notate cu J şi K, deci mulţimea X are 4 elemente: $X=\{00,01,10,11\}$. Registrul serie din figura 7.1 are o intrare de date notată cu A şi mulţimea X are în acest caz numai 2 elemente: $X=\{0,1\}$. Cele 3 bistabile ale registrului stabilesc elementele mulţimii stărilor: $Q=\{000,001,010,011,100,101,110,111\}$. În toate exemplele de până acum nu am avut alte ieşiri decât cele ale bistabilelor, deci mulţimea Y este vidă.

Dacă funcţia de tranziţie a ieşirilor depinde atât de starea sistemului cât şi de intrări, adică $\lambda: X\times Q\to Y$, atunci automatul este **de tip Mealy**, iar dacă ieşirea depinde numai de starea internă a sistemului, adică $\lambda: Q\to Y$, atunci automatul este **de tip Moore.**

În acest capitol ne ocupăm de **automatele finite sincrone**, adică cele la care funcţiile de tranziţie δ şi λ sunt calculate la momente de timp determinate de un semnal de ceas. Frontul activ al ceasului comandă actualizarea valorilor funcţiilor de tranziţie, calculul lor fiind încheiat înainte de apariţia următorului front activ de ceas. Spaţiul timpului este discret şi este format din mulţimea multiplilor perioadei semnalului de ceas.

Există posibilitatea separării unui automat finit în două blocuri funcţionale, aşa cum se poate vedea în figura 7.13. Această separare nu este obligatorie, dar este deosebit de utilă în proiectare. Aceasta, deoarece subsistemul de memorare poate fi definit foarte simplu, ca un registru paralel cu un număr suficient de bistabile, în timp ce subsistemul de prelucrare este format dintr-o logică combinaţională mai complexă şi mai greu de definit.

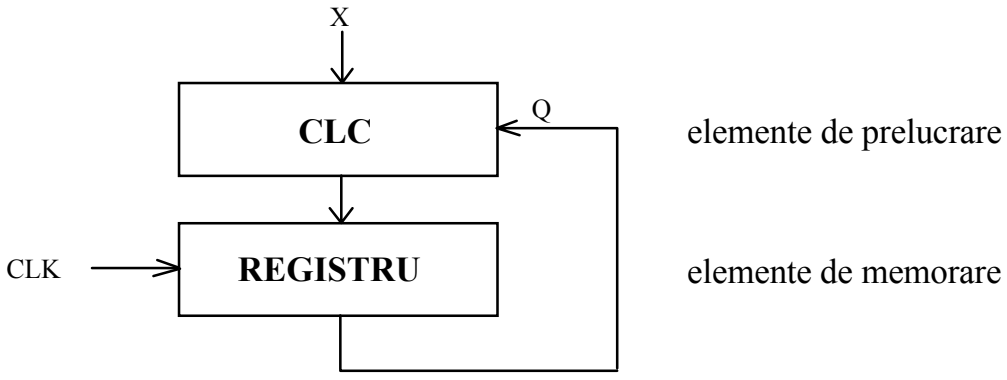


Fig. 7.13 *Separarea funcţională a unui automat finit*

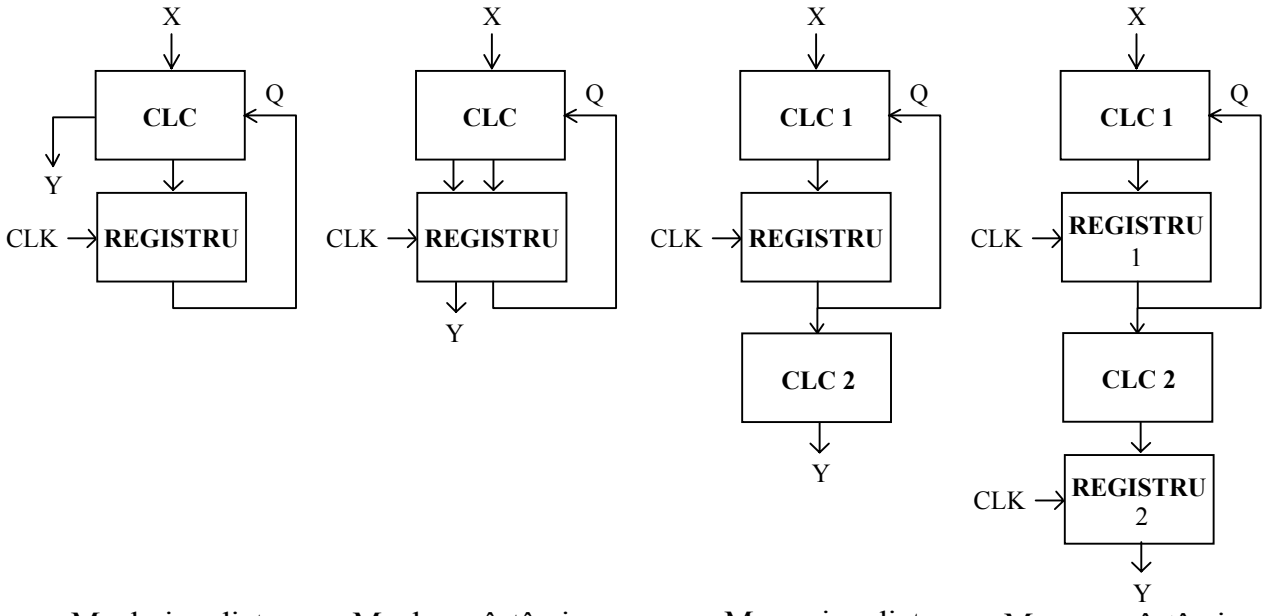


Fig. 7.14 *Structuri fundamentale de automate finite*

Automatele se pot clasifica în **imediate**, dacă ieşirile lor se modifică imediat ce se modifică intrările sau stările, şi **cu întârziere**, dacă ieşirile se modifică cu o întârziere de o perioadă de ceas, față de modificarea intrărilor sau a stărilor. Cele patru structuri fundamentale de automate finite rezultate prin combinarea tipurilor prezentate mai sus sunt date în figura 7.14.

Orice automat Mealy cu întârziere poate fi transformat într-un automat Moore imediat şi, invers, orice automat Moore imediat poate fi transformat într-un automat Mealy cu întârziere. Exemplul din figura 7.15 arată că transformarea unui automat Moore în automat Mealy este mai simplă, deoarece în afara mulțimilor X şi Y se conservă şi spațiul stărilor Q. La transformarea unui automat Mealy în automat Moore este posibil ca numărul de stări să difere, deoarece pentru fiecare ieşire y_i diferită avem nevoie de o nouă stare.

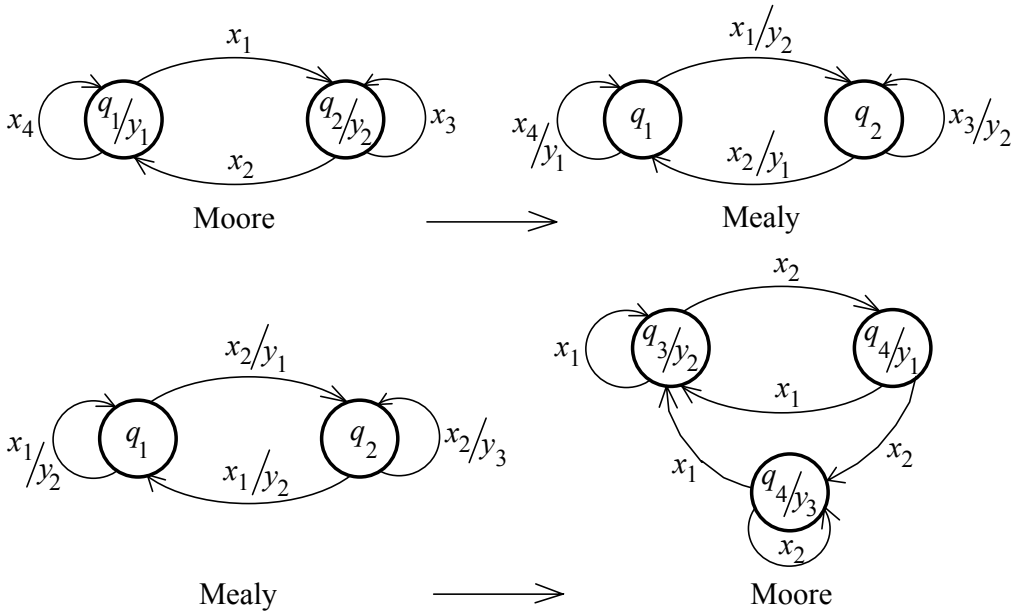


Fig. 7.15 *Transformarea automatelor finite*

Problema fundamentală care se pune la sinteza automatelor finite este o **problemă de optimizare**. Proiectantul nu poate acționa asupra numărului de intrări şi ieşiri din sistem, deoarece ele sunt date prin specificațiile de proiectare, dar are deplina libertate de a acționa asupra mulțimii stărilor sistemului. Prin **reducerea numărului de stări** scade numărul elementelor de memorie, deci a bistabilelor din structură, şi se reduce numărul funcțiilor de excitație, cu alte cuvinte complexitatea circuitului combinațional.

La fel de importantă este şi **codificarea stărilor** rămase după reducere. O codificare care respectă anumite principii poate genera un circuit combinațional foarte apropiat de optim, în timp ce o codificare întâmplătoare poate genera un circuit mult prea complicat.

7.4 Reducerea numărului de stări

Pentru reducerea numărului de stări vom căuta stările care nu se deosebesc prin comportare, adică stări care generează aceleaşi stări viitoare şi aceleaşi ieşiri, pentru aceeaşi secvență de date aplicate la intrare. Două stări de acest fel vor fi numite **stări echivalente**.

Vom spune că două stări q_i şi q_j ale unui automat $A=(X,Y,Q,\delta,\lambda)$ sunt echivalente, $q_i \equiv q_j$, dacă $\lambda(q_i,x^k)=\lambda(q_j,x^k)$ şi $\delta(q_i,x^k)=\delta(q_j,x^k)$ pentru orice secvență finită posibilă de intrare x^k.

Se pot defini şi **automate echivalente**, cu condiția ca ele să fie **complet definite**, adică pentru orice stare prezentă şi pentru orice intrare să existe o stare următoare, deci tranziția respectivă să fie definită de la început.

Două automate complet definite $A_1(X,Y,Q_1,\delta_1,\lambda_1)$ şi $A_2(X,Y,Q_2,\delta_2,\lambda_2)$ sunt echivalente dacă şi numai dacă pentru orice stare $q_i \in Q_1$ există o stare $q_j \in Q_2$, astfel încât $q_i \equiv q_j$, şi pentru orice stare $q_j \in Q_2$ există o stare $q_i \in Q_1$, astfel încât $q_j \equiv q_i$. Prin urmare, dacă le privim din exterior, cele două automate echivalente nu pot fi diferențiate funcțional.

Exemplul 7.2

Dacă considerăm automatele descrise prin tabelele din figura 7.16 şi aplicăm secvența de intrare 0, 1, 1, 0 automatului A_1 aflat în stările q_1 sau q_3 şi automatului A_2 aflat în starea s_1, obținem: $\lambda_1(q_1,(0,1,1,0))=\lambda_1(q_1,0),\lambda_1(\delta_1(q_1,0),1),\lambda_1(\delta_1(\delta_1(q_1,0),1),1),$ $\lambda_1(\delta_1(\delta_1(\delta_1(q_1,0),1),1),0)=\lambda_1(q_1,0),\lambda_1(q_3,1),\lambda_1(q_2,1),\lambda_1(q_2,0)=0,1,0,1$ ca ieşiri pentru q_1.

	starea următoare/ieşire	
starea prezentă	$x=0$	$x=1$
q_1	$q_3/0$	$q_2/1$
q_2	$q_1/1$	$q_2/0$
q_3	$q_1/0$	$q_2/1$

automatul A_1

	starea următoare/ieşire	
starea prezentă	$x=0$	$x=1$
s_1	$s_1/0$	$s_2/1$
s_2	$s_1/1$	$s_2/0$

automatul A_2

Fig. 7.16 *Două automate finite echivalente*

Pentru starea q_3 obținem: $\lambda_1(q_3,(0,1,1,0))=\lambda_1(q_3,0),\lambda_1(q_1,1),\lambda_1(q_2,1),\lambda_1(q_2,0)=0,1,0,1$, iar pentru starea s_1 avem: $\lambda_2(s_1,(0,1,1,0))=\lambda_2(s_1,0),\lambda_2(s_1,1),\lambda_2(s_2,1),\lambda_2(s_2,0)=0,1,0,1$. Observăm că aceeaşi secvență de intrare generează aceleaşi ieşiri pentru cele 3 stări considerate. Deşi nu putem încerca toate secvențele de intrări posibile, putem afirma că cele 3 stări sunt echivalente: $q_1 \equiv q_3 \equiv s_1$. La fel se arată că $q_2 \equiv s_2$ şi rezultă că cele două automate sunt echivalente, adică: $A_1 \equiv A_2$.

Dacă partiționăm stările unui automat $A=(X,Y,Q,\delta,\lambda)$ în **clase de echivalență** şi luăm din fiecare clasă un singur reprezentant, vom obține un **automat redus**, $A_r=(X,Y,Q_r,\delta_r,\lambda_r)$, care este un automat cu număr minim de stări, echivalent cu cel inițial.

Partiționarea mulțimii stărilor în clase de echivalență se face folosind **metoda tabelului implicațiilor**. Tabelul conține un număr de compartimente egal cu numărul tuturor perechilor posibile de stări. Aceste compartimente sunt aranjate într-un tabel triunghiular, în care liniile sunt definite de stările automatului fără prima stare, iar coloanele, de stările automatului fără ultima stare. Fiecare compartiment conține rezultatul comparării a două stări, adică:

- simbolul $\times$ dacă stările din perechea respectivă sunt evident neechivalente
- simbolul $\surd$ dacă stările din perechea respectivă sunt evident echivalente
- implicațiile privind echivalența succesorilor, dacă stările din perechea respectivă sunt 1-echivalente, adică $\lambda(q_i,x^k)=\lambda(q_j,x^k)$.

Clasele de echivalență se stabilesc grupând perechile de stări echivalente în baza proprietății de **tranzitivitate** a relației de echivalență.

Exemplul 7.3

Ne propunem să partiționăm în clase de echivalență stările automatului descris în figura 7.17 şi să construim tabelul tranzițiilor pentru automatul redus. Tabelul implicațiilor este reprezentat în figura 7.18.

Observăm că stările q_1 şi q_2 sunt evident neechivalente, pentru că generează ieşiri diferite pentru intrarea x_1, în timp ce stările q_7 şi q_8 sunt evident echivalente, pentru că generează stări următoare şi ieşiri absolut identice.

	starea următoare/ieşire			
starea prezentă	x_1	x_2	x_3	x_4
q_1	$q_1/0$	$q_3/0$	$q_4/0$	$q_6/1$
q_2	$q_4/1$	$q_2/0$	$q_5/0$	$q_7/1$
q_3	$q_2/0$	$q_5/1$	$q_3/0$	$q_4/1$
q_4	$q_2/1$	$q_4/0$	$q_5/0$	$q_8/1$
q_5	$q_5/1$	$q_2/0$	$q_4/0$	$q_7/1$
q_6	$q_4/0$	$q_5/1$	$q_6/0$	$q_6/1$
q_7	$q_1/0$	$q_6/1$	$q_7/1$	$q_4/0$
q_8	$q_1/0$	$q_6/1$	$q_7/1$	$q_4/0$

Fig. 7.17 *Un exemplu de automat finit complet specificat*

q_2	×						
q_3	×	×					
q_4	×	$q_7 \equiv q_8$	×				
q_5	×	$q_4 \equiv q_5$	×	$q_2 \equiv q_5$ $q_2 \equiv q_4$ $q_7 \equiv q_8$			
q_6	×	×	$q_2 \equiv q_4$ $q_4 \equiv q_6$	×	×		
q_7	×	×	×	×	×	×	
q_8	×	×	×	×	×	×	√
	q_1	q_2	q_3	q_4	q_5	q_6	q_7

Fig. 7.18 *Tabelul triunghiular al implicaţiilor*

Observăm că există şi echivalenţe condiţionate: starea q_2 este echivalentă cu starea q_4 numai dacă stările q_7 şi q_8 sunt echivalente, condiţie îndeplinită după cum am stabilit mai sus. Starea q_3 este echivalentă cu starea q_6 numai dacă starea q_2 este echivalentă cu starea q_4, şi acest lucru este adevărat, şi dacă starea q_4 este echivalentă cu starea q_6. Observăm însă din tabel că stările q_4 şi q_6 sunt cert neechivalente, deci concluzia este că stările q_3 şi q_6 nu sunt echivalente.

Prin gruparea perechilor de stări echivalente rezultă următoarele clase de echivalenţă: $\{q_7, q_8\}$, $\{q_2, q_4, q_5\}$ pentru că starea q_2 este echivalentă cu starea q_4, iar q_4 este echivalentă cu q_5 dacă starea q_2 este echivalentă cu starea q_5. Stările care nu apar în aceste clase formează, singure, câte o clasă de echivalenţă: $\{q_1\}$, $\{q_3\}$, $\{q_6\}$. Partiţia în clase de echivalenţă are forma $\mathrm{P} = \{\{q_1\}, \{q_2, q_4, q_5\}, \{q_3\}, \{q_6\}, \{q_7, q_8\}\}$.

Dacă alocăm fiecărei clase câte o stare a automatului redus, atunci automatul redus are numai 5 stări: $q_1^r \equiv q_1$, $q_2^r \equiv q_2 \equiv q_4 \equiv q_5$, $q_3^r \equiv q_3$, $q_4^r \equiv q_6$, $q_5^r \equiv q_7 \equiv q_8$. Tabelul tranziţiilor şi al ieşirilor pentru automatul redus, tabel reprezentat în figura 7.19, se construieşte folosind tabelul iniţial din figura 7.17 şi relaţiile de echivalenţă între stări. Automatul redus este un automat echivalent, cu număr minim de stări, numit şi **automat redus minim**.

	starea următoare/ieşire			
starea prezentă	x_1	x_2	x_3	x_4
q_1^r	$q_1^r/0$	$q_3^r/0$	$q_2^r/0$	$q_4^r/1$
q_2^r	$q_2^r/1$	$q_2^r/0$	$q_2^r/0$	$q_5^r/1$
q_3^r	$q_2^r/0$	$q_2^r/1$	$q_3^r/0$	$q_2^r/1$
q_4^r	$q_2^r/0$	$q_2^r/1$	$q_4^r/0$	$q_4^r/1$
q_5^r	$q_1^r/0$	$q_4^r/1$	$q_5^r/1$	$q_2^r/0$

Fig. 7.19 *Tabelul tranziţiilor şi al ieşirilor pentru automatul redus*

În cazul automatelor **incomplet specificate**, pentru anumite intrări nu sunt definite stările următoare sau ieşirile, deci nu mai putem vorbi de stări echivalente. Dacă totuşi două stări q_i şi q_j ale unui automat incomplet specificat respectă condițiile de echivalență, în toate cazurile în care sunt specificate stările următoare şi ieşirile, atunci cele două stări se numesc **stări compatibile**, şi se notează $q_i \sim q_j$.

Determinarea **claselor de compatibilități** este o problemă mai dificilă decât determinarea claselor de echivalență, pentru că relația de compatibilitate nu are proprietatea de tranzitivitate. Ea este numai reflexivă şi simetrică. Prin urmare, pentru ca o stare să fie compatibilă cu stările unei clase de compatibilități, ea trebuie să fie compatibilă cu fiecare dintre stările clasei respective, în parte.

Partiționarea mulțimii stărilor în clase de compatibilități se face tot cu ajutorul tabelului implicațiilor. Tabelul conține compartimente aranjate triunghiular ca în exemplul anterior. Fiecare compartiment conține rezultatul comparării a două stări:

- simbolul $\times$ dacă stările din perechea respectivă sunt evident incompatibile
- simbolul $\surd$ dacă stările din perechea respectivă sunt evident compatibile
- pentru celelalte compartimente se scriu condițiile de compatibilitate care se impun pentru ca stările comparate să fie compatibile.

Exemplul 7.4

Ne propunem să partiționăm în clase de compatibilități stările automatului descris prin tabelul din figura 7.20 şi să construim tabelul tranzițiilor şi al ieşirilor pentru automatul redus. Tabelul implicațiilor este reprezentat în figura 7.21.

Se observă că incompatibilitatea stărilor q_2 şi q_5, respectiv a stărilor q_3 şi q_6, determină incompatibilitatea perechilor de stări $\{q_1, q_7\}$, respectiv $\{q_3, q_4\}$ și $\{q_4, q_5\}$, care le implică. Urmărind şi celelalte compartimente din tabelul implicațiilor obținem următoarele perechi de stări compatibile: $\{q_1, q_2\}$, $\{q_1, q_3\}$, $\{q_2, q_3\}$, $\{q_2, q_4\}$, $\{q_1, q_5\}$, $\{q_3, q_5\}$, $\{q_2, q_6\}$, $\{q_4, q_6\}$, $\{q_3, q_7\}$ şi $\{q_5, q_7\}$.

Dacă analizăm acum primele trei perechi de mai sus observăm că starea q_1 este compatibilă cu stările q_2 şi q_3, care la rândul lor sunt compatibile între ele. Concluzia este că cele trei stări formează împreună o clasă de compatibilități: $\{q_1, q_2, q_3\}$.

	starea următoare/ieşire			
starea prezentă	x_1	x_2	x_3	x_4
q_1	$q_3/0$	-/-	$q_4/1$	$q_2/-$
q_2	-/-	$q_2/1$	$q_6/-$	$q_3/1$
q_3	$q_2/0$	$q_6/1$	-/-	-/-
q_4	-/-	$q_3/1$	$q_2/0$	$q_3/-$
q_5	$q_1/0$	$q_6/1$	$q_4/-$	-/0
q_6	$q_3/1$	$q_2/-$	-/0	$q_3/1$
q_7	$q_2/0$	$q_4/-$	-/1	$q_5/0$

Fig. 7.20 *Automat finit incomplet specificat*

q_2	$q_2 \sim q_3$ $q_4 \sim q_6$					
q_3	$q_2 \sim q_3$	$q_2 \sim q_6$				
q_4	×	$q_2 \sim q_3$ $q_2 \sim q_6$	$q_3 \sim q_6$			
q_5	$q_1 \sim q_3$	×	$q_1 \sim q_2$	$q_2 \sim q_4$ $q_3 \sim q_6$		
q_6	×	√	×	$q_2 \sim q_3$	×	
q_7	$q_2 \sim q_3$ $q_2 \sim q_5$	×	$q_4 \sim q_6$	×	$q_1 \sim q_2$ $q_4 \sim q_6$	×
	q_1	q_2	q_3	q_4	q_5	q_6

Fig. 7.21 *Tabelul triunghiular al implicațiilor automatului incomplet specificat*

Dacă procedăm la fel şi cu celelalte perechi de stări vom obține toate compatibilitățile posibile: $\{q_1, q_2, q_3\}$, $\{q_1, q_3, q_5\}$, $\{q_2, q_4, q_6\}$ şi $\{q_3, q_5, q_7\}$. Dacă facem reuniunea acestor clase de compatibilități rezultă o acoperire a mulțimii stărilor automatului. Din acest motiv le vom numi **compatibilități maxime**.

Un automat redus, pe care îl notăm cu A_r, se poate obține din automatul dat A, dacă pentru fiecare clasă de compatibilități maxime C_i din A se găseşte o stare q_i^r din A_r, care să fie compatibilă cu fiecare din stările clasei C_i. Vom spune că starea q_i^r acoperă clasa de compatibilități C_i. Pentru a găsi o acoperire minimă vom determina clasele de **compatibilități prime** şi vom alege un număr minim dintre acestea. Compatibilitățile prime sunt compatibilități care se pot forma cu stările unui automat incomplet specificat şi care nu sunt excluse de alte compatibilități ale aceluiaşi automat. O clasă de compatibilități C_j dintr-o acoperire dată poate fi exclusă de o altă clasă C_i, dacă $C_j \subset C_i$, iar clasa exclusă implică mai multe clase de compatibilități, adică $I_{C_i} \subseteq I_{C_j}$. Compatibilitățile maxime sunt şi prime pentru că nu sunt incluse în nici o

Compatibilități maxime (C_i)	(C_1) $\{q_1, q_2, q_3\}$	(C_2) $\{q_1, q_3, q_5\}$	(C_3) $\{q_2, q_4, q_6\}$	(C_4) $\{q_3, q_5, q_7\}$
Mulțimi implicate (I_{C_i})	$\{q_2, q_6\}$ $\{q_4, q_6\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_1, q_2\}$ $\{q_4, q_6\}$

Fig. 7.22 *Clase de compatibilități maxime şi implicațiile lor*

Compatibilități prime (C_j)	(C_5) $\{q_1, q_2\}$	(C_6) $\{q_2, q_3\}$	(C_7) $\{q_1, q_3\}$	(C_8) $\{q_1, q_5\}$	(C_9) $\{q_3, q_5\}$	(C_{10}) $\{q_2, q_6\}$	(C_{11}) $\{q_3, q_7\}$
Mulțimi implicate (I_{C_j})	$\{q_2, q_3\}$ $\{q_4, q_6\}$	$\{q_2, q_6\}$	$\{q_2, q_3\}$	$\{q_1, q_3\}$	$\{q_1, q_2\}$	Φ	$\{q_4, q_6\}$

Fig. 7.23 *Clase de compatibilități prime şi implicațiile lor*

altă clasă de compatibilități. Tabelul din figura 7.22 ilustrează **mulțimile compatibilităților implicate** de clasele de compatibilități maxime, iar cel din figura 7.23 - mulțimile compatibilităților implicate de clasele de compatibilități prime. Observăm că prin eliminarea unei stări din clasa C_1 se formează clasele C_5, C_6 şi C_7; clasa C_2 formează clasele noi C_8 şi C_9; clasa C_3 formează numai clasa C_{10} (clasele $\{q_2, q_4\}$ şi $\{q_4, q_6\}$sunt excluse de C_3), iar clasa C_4 formează numai clasa C_{11}. Clasa C_{10}, pentru care mulțimea implicațiilor este vidă, nu mai poate genera alte clase de compatibilități prime.

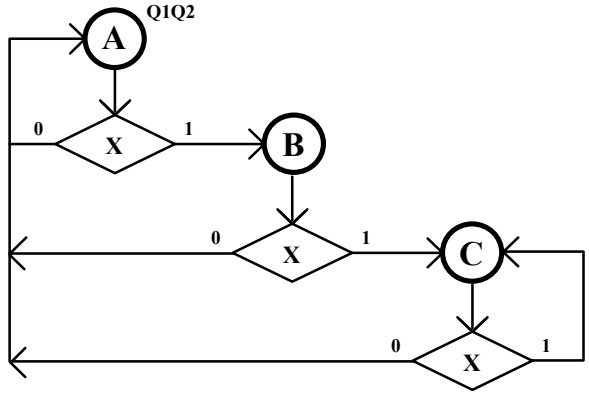
Clasa de **compatibilități prime esențială** este C_3 pentru că starea q_4 nu mai este conținută de nici o altă clasă. Această clasă implică pe $\{q_2, q_3\}$ sau pe $\{q_1, q_2, q_3\}$, căci $\{q_2, q_3\} \subset \{q_1, q_2, q_3\}$. Mai adăugăm la cele două clase de compatibilități maxime C_3 şi C_1, clasa C_4, care implică o clasă deja luată în considerare. Acoperirea minimă în exemplul nostru este formată din clasele C_1, C_3 şi C_4 ([Pop, 1986]).

7.5 Codificarea stărilor

Problema fundamentală în sinteza automatelor finite este codificarea optimă a stărilor. O codificare aleatoare a stărilor automatului redus poate oferi o soluție corectă, dar complexitatea elementelor de prelucrare, adică a logicii combinaționale, ar putea fi mult prea mare. Dacă automatul redus are *m* stări, atunci o codificare binară minimală se poate face folosind coduri cu *r* biți, conform relației $2^{r-1} < m \le 2^r$. Problema este foarte complicată pentru că numărul de codificări distincte posibile este:

$$N_C = \frac{2^r!}{(2^r - m)!},$$

şi este foarte greu de spus care dintre ele este cel mai bun! În figura 7.24 este dată organigrama unui automat finit cu *m* = 3 stări, notate cu *A*, *B* şi *C*, şi o intrare, notată cu *X*. Codificarea stărilor se face cu 2 biți, care sunt furnizați de ieşirile a două bistabile de tip D. În figură sunt prezentate două din cele 24 de soluții distincte posibile pentru codificarea stărilor. Expresiile funcțiilor obținute sugerează diferența de complexitate dintre soluții.



dacă A = 01 , B = 10 şi C = 11:

$$Q_1^+ = x \quad \text{şi} \quad Q_2^+ = \bar{x} + Q_1$$

dacă A = 01 , B = 00 şi C = 11:

$$Q_1^+ = x \cdot (\overline{Q}_0 + Q_1)$$

$$Q_2^+ = \bar{x} + Q_1 + \overline{Q}_2$$

Fig. 7.24 *Două codificări posibile pentru un automat finit*

starea prezentă	starea următoare / ieşire			
	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
q_1	$q_1 / 0$	$q_2 / 0$	$q_3 / 1$	$q_2 / 1$
q_2	$q_2 / 0$	$q_3 / 1$	$q_1 / 0$	$q_3 / 1$
q_3	$q_3 / 0$	$q_1 / 1$	$q_1 / 1$	$q_1 / 1$

Fig. 7.25 *Exemplu de tabel al tranziţiilor şi ieşirilor*

starea prezentă $(Q_1 Q_2)$	starea următoare / ieşire $(Q_1^+ Q_2^+ / y)$			
	$x_1x_2 = 00$	$x_1x_2 = 01$	$x_1x_2 = 11$	$x_1x_2 = 10$
$a_1 b_1$	$a_1 b_1 / 0$	$a_2 b_2 / 0$	$a_3 b_3 / 1$	$a_2 b_2 / 1$
$a_2 b_2$	$a_2 b_2 / 0$	$a_3 b_3 / 1$	$a_1 b_1 / 0$	$a_3 b_3 / 1$
$a_3 b_3$	$a_3 b_3 / 0$	$a_1 b_1 / 1$	$a_1 b_1 / 1$	$a_1 b_1 / 1$

Fig. 7.26 *Tabelul binar al tranziţiilor şi ieşirilor*

Exemplul 7.5

Pentru a vedea influenţa codificării asupra expresiilor stării următoare şi ale ieşirii, considerăm automatul redus minim specificat prin tabelul din figura 7.25. Se codifică cele 3 stări q_i ale automatului cu 2 cifre binare $a_i b_i$, adică $a_i, b_i \in \{0,1\}$, iar $i = 1,2,3$. Cu această codificare a stărilor rezultă tabelul din figura 7.26.

Dacă folosim acum următoarea notaţie simbolică:

$$Q_i^k = \begin{cases} \overline{Q_i} & \text{pentru } k = 0 \\ Q_i & \text{pentru } k = 1 \end{cases}$$

atunci expresiile stărilor următoare ale variabilelor binare de stare devin:

$$Q_1^+ = a_1\left[Q_1^{a_1} Q_2^{b_1} \overline{x}_1 \overline{x}_2 + Q_1^{a_3} Q_2^{b_3} \overline{x}_1 x_2 + \left(Q_1^{a_2} Q_2^{b_2} + Q_1^{a_3} Q_2^{b_3}\right) x_1 x_2 + Q_1^{a_3} Q_2^{b_3} x_1 \overline{x}_2\right] +$$
$$+ a_2\left[Q_1^{a_2} Q_2^{b_2} \overline{x}_1 \overline{x}_2 + Q_1^{a_1} Q_2^{b_1} \overline{x}_1 x_2 + Q_1^{a_1} Q_2^{b_1} x_1 \overline{x}_2\right] + a_3\left[Q_1^{a_3} Q_2^{b_3} \overline{x}_1 \overline{x}_2 + Q_1^{a_2} Q_2^{b_2} \overline{x}_1 x_2 + \right.$$
$$\left. + Q_1^{a_1} Q_2^{b_1} x_1 x_2 + Q_1^{a_2} Q_2^{b_2} x_1 \overline{x}_2\right] = a_1 E_1 + a_2 E_2 + a_3 E_3 \quad \text{şi}$$
$$Q_2^+ = b_1 E_1 + b_2 E_2 + b_3 E_3$$

Aceste expresii se simplifică dacă cifrele binare din faţa parantezelor pătrate au valoarea 0 sau dacă expresiile din paranteze se pot minimiza. Numărul codurilor formate din multe zerouri fiind limitat, trebuie să urmărim mai ales în ce condiţii se pot minimiza expresiile din paranteze. În parantezele simple apar termeni generaţi de stările prezente, care au aceeaşi stare următoare pentru aceleaşi intrări, sau acelaşi **succesor**. Pentru ca doi astfel de termeni să fie absorbiţi de un termen cu o variabilă mai puţin, trebuie ca cei doi termeni să difere printr-o singură variabilă, deci codurile stărilor care le-au generat să fie adiacente. Deci, o **primă regulă de codificare** care trebuie respectată la codificarea stărilor, afirmă că stările care au aceiaşi succesori, pentru aceleaşi intrări, trebuie să primească coduri adiacente.

În exemplul considerat stările q_2 şi q_3 generează pentru $x_1x_2=11$ starea următoare q_1, deci stările q_2 şi q_3 trebuie să primească coduri adiacente.

Reluăm expresiile stărilor următoare scrise mai sus şi constatăm că ele pot fi scrise sub forma:

$$Q_1^+ = Q_1^{a_1}Q_2^{b_1}\left(a_1\overline{x}_1\overline{x}_2 + a_2\overline{x}_1x_2 + a_2x_1\overline{x}_2 + a_3x_1x_2\right) + Q_1^{a_2}Q_2^{b_2}\left(a_1x_1x_2 + a_2\overline{x}_1\overline{x}_2 + \right.$$
$$\left. + a_3\overline{x}_1x_2 + a_3x_1\overline{x}_2\right) + Q_1^{a_3}Q_2^{b_3}\left(a_1\overline{x}_1x_2 + a_1x_1x_2 + a_1x_1\overline{x}_2 + a_3\overline{x}_1\overline{x}_2\right)$$

Expresia lui Q_2^+ se scrie sub o formă asemănătoare. Observăm acum că expresiile se simplifică cel mai mult dacă cifrele binare diferite de zero din parantezele corespondente stau pe lângă aceiaşi termeni cu intrări adiacente. Rezultă imediat o **a doua regulă de codificare**, care afirmă că succesorii unor stări care se află pe coloane cu intrări adiacente, primesc coduri adiacente.

Conform acestei reguli de codificare a stărilor rezultă că perechile de stări q_2 şi q_3, q_1 şi q_3, respectiv q_1 şi q_2 trebuie să primească coduri adiacente.

Expresia ieşirii se poate scrie sub forma:

$$y = \overline{x}_1x_2\left(Q_1^{a_2}Q_2^{b_2} + Q_1^{a_3}Q_2^{b_3}\right) + x_1x_2\left(Q_1^{a_1}Q_2^{b_1} + Q_1^{a_3}Q_2^{b_3}\right) +$$
$$+ x_1\overline{x}_2\left(Q_1^{a_1}Q_2^{b_1} + Q_1^{a_2}Q_2^{b_2} + Q_1^{a_3}Q_2^{b_3}\right)$$

Se poate observa şi aici că stările prezente care au în corespondență, pentru aceleaşi intrări ieşiri identice, trebuie să aibă coduri adiacente. Această concluzie formează **a treia regulă de codificare** a stărilor.

Rezultă deci că perechile de stări q_1 şi q_3, şi q_2 şi q_3 primesc coduri adiacente.

Regulile formulate până acum stabilesc numai condițiile de adiacență între codurile stărilor. Pentru a simplifica expresiile stărilor următoare este importantă şi forma codurilor adiacente. Astfel, stările care apar mai des ca stare următoare ar trebui să primească coduri cu mai multe zerouri şi astfel dispar parantezele care conțin mai mulți termeni. **A patra regulă de codificare** a stărilor, recomandă atribuirea codurilor cu mai multe zerouri stărilor care apar o singură dată pe mai multe coloane.

Vom atribui codul 00 stării q_3. Dacă ținem seamă de aceste recomandări enunțate sub forma unor reguli, atunci alocăm codurile 01 pentru starea q_1 şi 10 pentru starea q_2. Condiția de adiacență pentru stările q_1 şi q_2 nu s-a putut respecta.

starea prezentă (Q_1Q_2)	starea următoare / ieşire ($Q_1^+Q_2^+$ / y)			
	$x_1x_2=00$	$x_1x_2=01$	$x_1x_2=11$	$x_1x_2=10$
00	00 / 0	01 / 1	01 / 1	01 / 1
01	01 / 0	10 / 0	00 / 1	10 / 1
11	- / -	- / -	- / -	- / -
10	10 / 0	00 / 1	01 / 0	01 / 1

Fig. 7.27 *Tabelul tranzițiilor şi al ieşirilor rezultat în urma codificării propuse*

Cu aceste coduri se construieşte tabelul tranziţiilor şi al ieşirilor din figura 7.27. Funcţiile de excitaţie Q_1^+, Q_2^+ şi ieşirea y se calculează prin minimizare cu ajutorul diagramelor Veitch-Karnaugh, iar expresiile obţinute sunt cele de mai jos:

$$Q_1^+ = Q_1\bar{x}_1\bar{x}_2 + Q_2\bar{x}_1x_2 + Q_2x_1\bar{x}_2$$
$$Q_2^+ = Q_2\bar{x}_1\bar{x}_2 + \overline{Q}_1\overline{Q}_2\bar{x}_1x_2 + \overline{Q}_1\overline{Q}_2x_1\bar{x}_2 + \overline{Q}_2x_1x_2$$
$$y = \overline{Q}_2\bar{x}_1x_2 + \overline{Q}_1x_1x_2 + x_1\bar{x}_2$$

Deşi par complicate, alte codificări ale stărilor generează probabil expresii mai complicate. Costul global al circuitului combinaţional este de 13 porţi şi 41 intrări. Alegerea codurilor 00, 01 şi 10 pentru stările q_1, q_2 şi respectiv q_3 oferă o soluţie cu 14 porţi şi 44 intrări, iar codurile 11, 10 şi 01 pentru aceeaşi ordine a stărilor furnizează un circuit cu 16 porţi şi 46 intrări. Lăsăm în sarcina cititorului să verifice şi alte coduri.

Regulile prezentate aici sunt mai mult nişte recomandări practice şi nu garantează cu certitudine găsirea codurilor optime. O recomandare suplimentară, pe care o putem numi **a cincea regulă de codificare**, indică ca stările circuitului între care există tranziţii să primească coduri adiacente. În acest fel, la fiecare tranziţie a sistemului comută un singur bistabil ([Pop, 1986]).

Dificultatea codificării stărilor este ilustrată şi în lucrarea [Almaini, 1995]. Autorii lucrării propun rezolvarea problemei cu ajutorul unui **algoritm genetic**, care oferă o soluţie bună, de obicei foarte apropiată de optimul global. Pentru minimizarea funcţiilor binare se foloseşte algoritmul ESPRESSO, iar rezultatele obţinute sunt de cele mai multe ori mai bune decât cele oferite de algoritmii determinişti convenţionali.

7.6 Exemple de proiectare

Pentru a face sinteza unui sistem secvenţial sincron vom porni de la tema de proiectare sau de la caietul de sarcini impus, care de multe ori poate fi o descriere în limbaj natural a problemei. Pentru început trebuie să determinăm un mod de reprezentare formală a problemei folosind o diagramă a stărilor sau un tabel de tranziţii. Probabil că aceasta este etapa cea mai dificilă a operaţiei de sinteză, deoarece nu se face pe o bază algoritmică.

După ce am construit tabelul tranziţiilor nu mai rămâne decât să facem reducerea numărului de stări, dacă acest lucru este posibil, să codificăm stările rămase, să facem sinteza funcţiilor binare de excitaţie şi de ieşire şi să implementăm schema logică a circuitului. Toate aceste operaţii se pot face relativ uşor pentru că reprezintă o aplicare directă a algoritmilor cunoscuţi la rezolvarea problemei.

Există două clase mari de sisteme secvenţiale sincrone: sisteme cu **comportament asincron**, la care intrările se pot modifica în orice moment de timp, fără nici o legătură cu semnalul de ceas, şi sisteme cu **comportament sincron**, la care intrările se modifică numai pe frontul activ al ceasului. La sistemele cu comportament asincron frecvenţa ceasului este de obicei mult mai mare decât frecvenţa de modificare a intrărilor, deci starea metastabilă, chiar dacă apare, nu poate să dureze mult. Ieşirile unui astfel de sistem nu depind de duratele semnalelor de intrare ci numai de ordinea în care acestea se modifică.

Exemplul 7.6

Vom prezenta procedeul de sinteză a unui **discriminator de sens de rotație**, care este un sistem cu comportament asincron.

Caietul de sarcini este reprezentat schematic în figura 7.28. Circuitul are două intrări, notate cu x_1 şi x_2, şi o ieşire, notată cu y, care are valoarea logică 0 dacă arborele se roteşte în sens orar, şi valoarea logică 1 în caz contrar. Arborele este împărțit în 4 sectoare izolate între ele, conectate alternativ la constantele logice 0 şi 1. Distanța dintre cele două contacte elastice conectate la intrările x_1 şi x_2 este mai mică de un sfert din circumferința secțiunii arborelui.

Pentru a înțelege mai bine funcționarea circuitului în cele două situații se recomandă reprezentarea formelor de undă în timp, sau cronogramele. Admitem că frecvența semnalului CLK este mult mai mare decât cea cu care se modifică intrările x_1 şi x_2. Dacă arborele se roteşte în sens orar, atunci secvența de modificare a intrărilor este $x_1x_2 = = 00 \to 01 \to 11 \to 10 \to 00$. Am presupus că sistemul trece într-o nouă stare pentru fiecare nouă combinație binară aplicată pe intrări. Aceste stări au fost numerotate cu 1, 2, 3 şi 4. Din starea 4 se trece din nou în starea 1, deoarece secvența se repetă. Dacă arborele se roteşte în sens contrar celui orar, atunci secvența de modificare a intrărilor este $x_1x_2 = 00 \to 10 \to 11 \to 01 \to 00$. Noile stări care apar au fost numerotate cu 5, 6, 7 şi 8. Se poate observa că starea 8 nu este identică cu starea 1 pentru că, deşi intrările sunt identice, ieşirile sunt diferite.

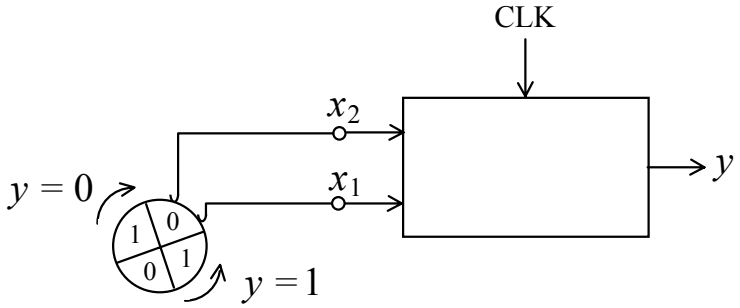


Fig. 7.28 *Caietul de sarcini pentru discriminatorul de sens de rotație*

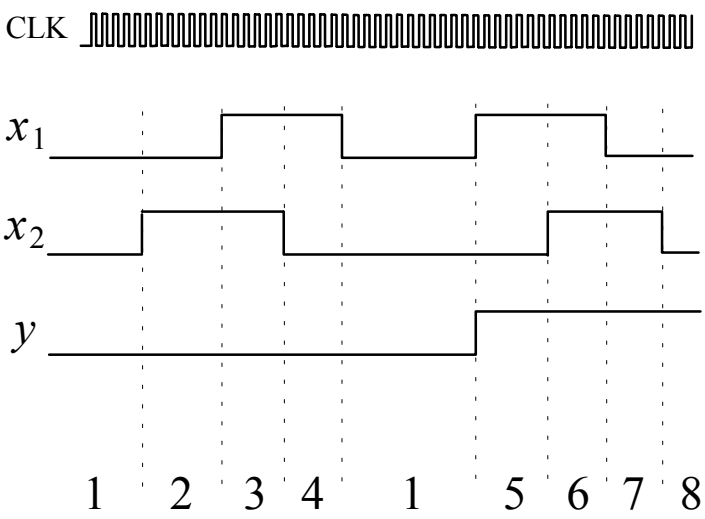


Fig. 7.29 *Formele de undă care explică funcționarea circuitului*

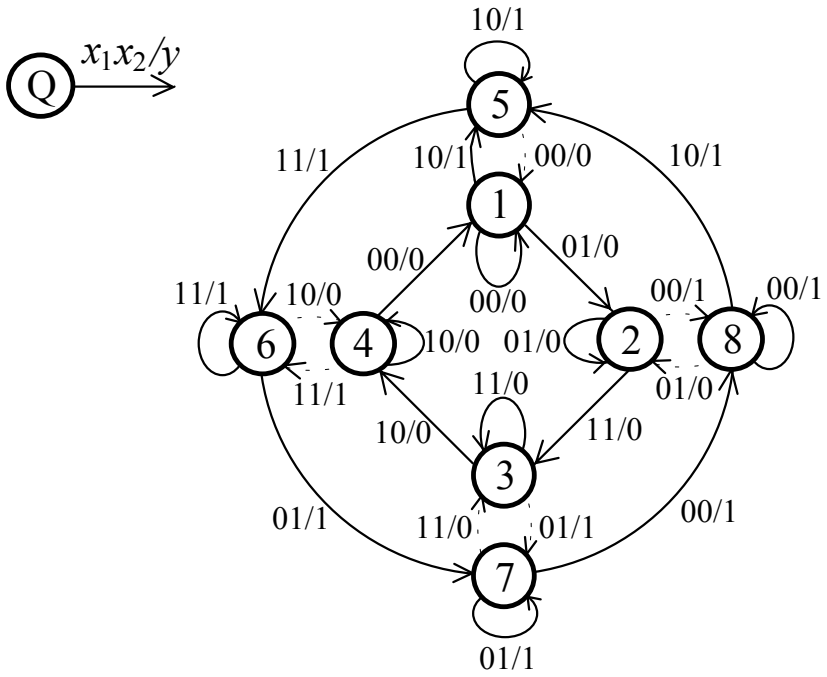


Fig. 7.30 *Diagrama stărilor pentru discriminatorul de sens de rotație*

Formele de undă ne ajută să înțelegem funcționarea sistemului, dar ele nu permit vizualizarea tuturor tranzițiilor posibile între stări. Diagrama stărilor, reprezentată în figura 7.30, arată toate aceste tranziții. Tranzițiile care lipsesc din formele de undă sunt aici reprezentate cu linii punctate. Stările sistemului sunt noduri ale grafului, fiind reprezentate prin cercuri etichetate, iar arcele, care sugerează tranzițiile, au ca etichete valorile intrărilor şi ale ieşirii la momentul respectiv.

Se poate observa că după ce sistemul îşi schimbă starea, noua stare este păstrată timp de mai multe perioade de ceas, atât timp cât intrările nu se modifică din nou. Sistemul "buclează" în starea respectivă. O astfel de stare se numeşte **stare total stabilă**, şi observăm că toate stările sistemului sunt total stabile. Tranziția de la o stare stabilă la alta se face după modificarea intrărilor, sincron cu frontul activ al ceasului, după cel mult o perioadă a semnalului de ceas. Dacă există cel puțin o stare care nu este total stabilă, atunci sistemul are comportament sincron.

Tabelul tranzițiilor este reprezentat în figura 7.31. Cele 4 coloane ale tabelului sunt definite de combinațiile binare ale intrărilor x_1 şi x_2, iar cele 8 linii sunt definite de

q \ x_1x_2	q^+, y			
	00	01	11	10
1	1,0	2,0	-,-	5,1
2	8,1	2,0	3,0	-,-
3	-,-	7,1	3,0	4,0
4	1,0	-,-	6,1	4,0
5	1,0	-,-	6,1	5,1
6	-,-	7,1	6,1	4,0
7	8,1	7,1	3,0	-,-
8	8,1	2,0	-,-	5,1

Fig. 7.31 *Tabelul tranzițiilor şi al ieşirilor*

q \ x_1x_2	00	01	11	10
1	1,0	2,0	4,1	1,1
2	2,1	2,0	3,0	1,1
3	2,1	3,1	3,0	4,0
4	1,0	3,1	4,1	4,0

q^+, y

stări compatibile: $1 = 5 \longrightarrow 1$
$2 = 8 \longrightarrow 2$
$3 = 7 \longrightarrow 3$
$4 = 6 \longrightarrow 4$

Fig. 7.32 *Tabelul redus al tranzițiilor şi al ieşirilor*

Q_1Q_2 \ x_1x_2	00	01	11	10
00	00,0	01,0	10,1	00,1
01	01,1	01,0	11,0	00,1
11	01,1	11,1	11,0	10,0
10	00,0	11,1	10,1	10,0

$Q_1^+Q_2^+, y$

Fig. 7.33 *Tabelul tranzițiilor şi al ieşirilor după codificarea stărilor*

stările sistemului. În tabel vom trece starea următoare q^+ şi ieşirea y. Automatul este incomplet definit, pentru că nu toate tranzițiile au sens. De exemplu, din starea prezentă 1, pentru $x_1x_2 = 00$ se rămâne în aceeaşi stare, iar ieşirea este în 0 logic. Pentru $x_1x_2 = 01$ se face tranziția în starea 2, iar ieşirea rămâne în 0 logic. Pentru $x_1x_2 = 10$ se face tranziția în starea 5, iar ieşirea se modifică în 1 logic. Combinația $x_1x_2 = 11$ nu este posibilă fizic, deci aici nici tranziția şi nici ieşirea nu sunt definite.

Pentru reducerea stărilor se observă că stările 1 şi 5 generează aceleaşi stări următoare şi aceleaşi ieşiri pe coloanele pentru care sunt definite, deci sunt stări compatibile. La fel, perechile de stări 2 şi 8, 3 şi 7, 4 şi 6 sunt compatibile. Înseamnă că tabelul redus, după cum se poate observa în figura 7.32, are numai 4 stări şi o ieşire.

Pentru codificarea stărilor, atribuim coduri adiacente stărilor care au aceiaşi succesori, conform primei reguli de codificare, şi obținem tabelul tranzițiilor din figura 7.33. Starea 1 a primit codul 00, starea 2 codul 01, starea 3 codul 11, iar starea 4 a primit codul 10.

Tabelul din figura 7.33 este descompus în 3 diagrame Veitch-Karnaugh şi obținem următoarele forme minime pentru funcțiile de excitație ale bistabilelor de tip D, Q_1^+ şi Q_2^+: $Q_1^+ = D_1 = x_1x_2 + x_2Q_1 + x_1Q_1$, $Q_2^+ = D_2 = \overline{x}_1x_2 + \overline{x}_1Q_2 + x_2Q_2$, iar pentru ieşirea y avem: $y = x_1\overline{Q}_1\overline{Q}_2 + \overline{x}_2\overline{Q}_1Q_2 + \overline{x}_1Q_1Q_2 + x_2Q_1\overline{Q}_2$. Schema logică este dată în figura 7.34.

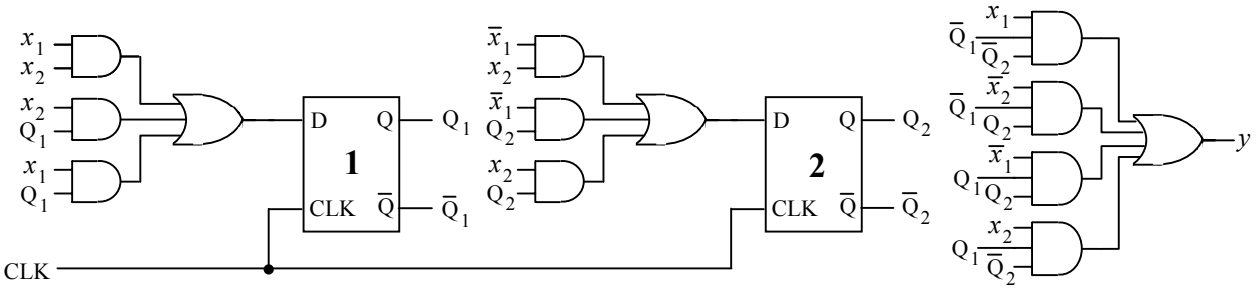


Fig. 7.34 *Schema logică a discriminatorului de sens de rotație*

O altă schemă logică echivalentă se poate obține cu o altă codificare a stărilor. Dacă folosim, de exemplu, codul 1 prin *m*, unde *m* este numărul de stări, atunci nu vom obține numărul minim de bistabile, deoarece această codificare binară a stărilor nu este minimală, dar vom obține **structuri combinaționale iterative**, adică structuri care se repetă, şi acest lucru ar putea fi avantajos dacă utilizăm structuri programabile.

Dacă pentru starea 1 alegem codul 0001, pentru starea 2 codul 0010, pentru starea 3 codul 0100, iar pentru starea 4 codul 1000, atunci noul tabel al tranzițiilor şi al ieşirii este reprezentat în figura 7.35. Din acest tabel putem scrie direct expresiile funcțiilor de excitație pentru bistabile de tip D şi expresia ieşirii:

$$Q_1^+ = D_1 = x_1x_2\left(Q_1 + Q_4\right) + x_1\bar{x}_2\left(Q_1 + Q_2\right)$$
$$Q_2^+ = D_2 = \bar{x}_1x_2\left(Q_1 + Q_2\right) + x_1x_2\left(Q_2 + Q_3\right)$$
$$Q_3^+ = D_3 = \bar{x}_1\bar{x}_2\left(Q_2 + Q_3\right) + \bar{x}_1x_2\left(Q_3 + Q_4\right)$$
$$Q_4^+ = D_4 = \bar{x}_1\bar{x}_2\left(Q_1 + Q_4\right) + x_1\bar{x}_2\left(Q_3 + Q_4\right)$$
$$y = \bar{x}_1\bar{x}_2\left(Q_2 + Q_3\right) + \bar{x}_1x_2\left(Q_1 + Q_2\right) + x_1x_2\left(Q_1 + Q_4\right) + x_1\bar{x}_2\left(Q_3 + Q_4\right)$$

Obținem schema logică din figura 7.36, dacă alegem o implementare cu multiplexoare şi porți. Lăsăm în seama cititorului varianta care foloseşte demultiplexoare şi porți.

Observăm şi pe acest exemplu cum coduri distincte generează structuri distincte, care sunt însă reprezentate prin grafuri sau tabele identice ([Mange, 1987]).

$Q_1Q_2Q_3Q_4$ \ x_1x_2	00	01	11	10
	$Q_1^+Q_2^+Q_3^+Q_4^+, y$			
0001	0001,0	0001,0	1000,1	0001,1
0010	0010,1	0010,1	0100,0	0001,1
0100	0010,1	0100,1	0100,0	1000,0
1000	0001,0	0100,1	1000,1	1000,0

Fig. 7.35 *Tabelul tranzițiilor şi al ieşirii pentru codul 1 prin m*

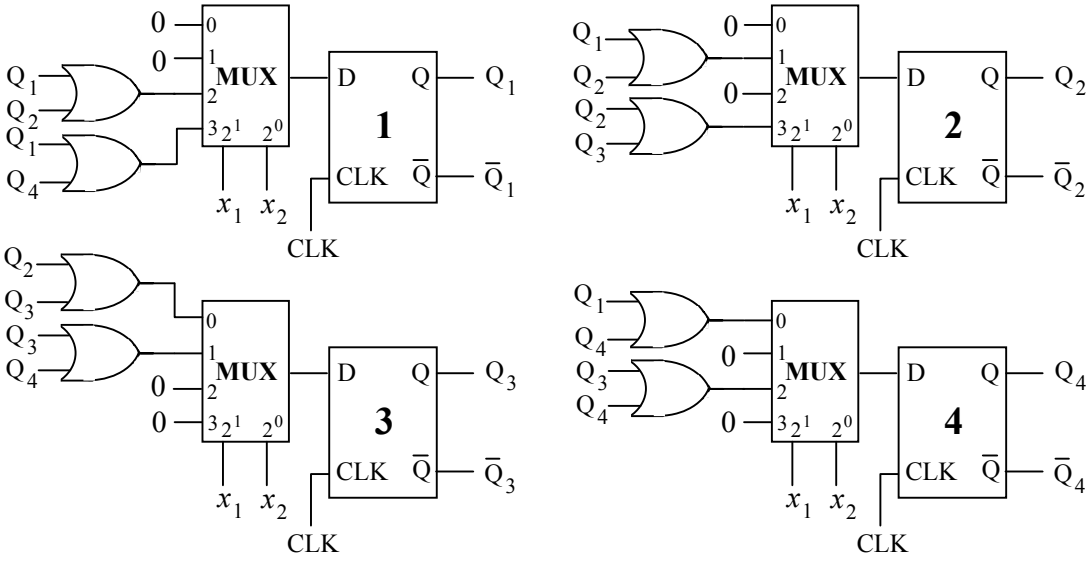


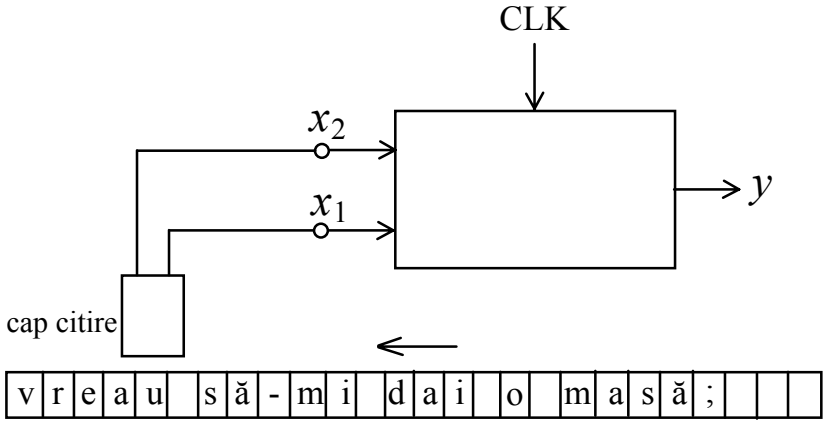
Fig. 7.36 *O implementare cu multiplexoare cu 2 intrări de selecție*

Metoda generală de sinteză pentru sistemele cu comportament sincron este aceeaşi ca la sistemele cu comportament asincron. De această dată intrările se modifică în sincronism cu semnalul de ceas şi nu toate stările sistemului sunt total stabile. Ieşirile depind atât de duratele semnalelor de intrare, cât şi de ordinea de succesiune a lor.

Exemplul 7.7

Vom prezenta procedeul de sinteză a unui **sistem care analizează un text**, caracter cu caracter, şi recunoaşte acele cuvinte care au terminația "să". Sistemul este sincron şi are un comportament sincron.

Caietul de sarcini este reprezentat schematic în figura 7.37. Sistemul are un cap de citire care analizează la fiecare semnal de ceas un caracter nou. Rezultatul citirii este furnizat la cele două intrări ale circuitului, notate cu x_1 şi x_2, iar ieşirea, notată cu y, trece în 1 logic pe ceasul imediat următor detecției unui cuvânt care se termină cu caracterele *s* şi *ă*. Semnificația combinațiilor binare de pe intrări este dată în figură.



x_1	x_2	semnificația
0	0	s-a citit caracterul "s" (s)
0	1	s-a citit caracterul "ă" (a)
1	1	alt caracter, inclusiv cratima (l)
1	0	semn de punctuație (p)

Fig. 7.37 *Caietul de sarcini pentru detectorul de secvență*

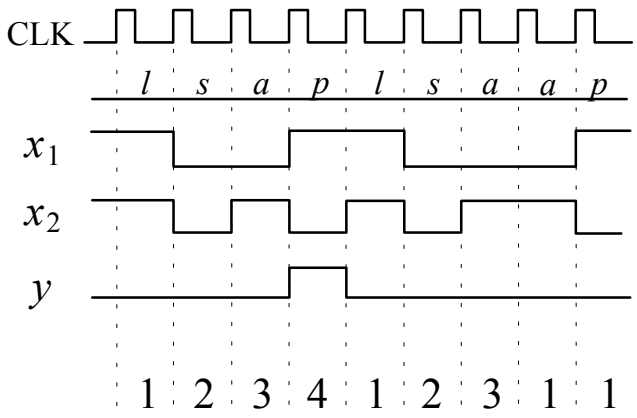


Fig. 7.38 *Formele de undă pentru detectorul de secvență*

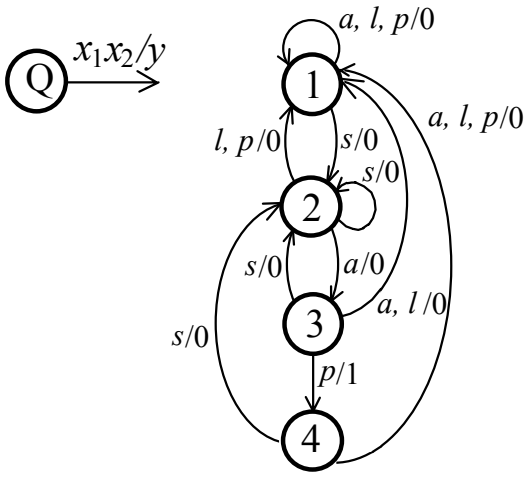


Fig. 7.39 *Diagrama stărilor*

x_1x_2 / Q	Q⁺, y			
	00	01	11	10
1	2,0	1,0	1,0	1,0
2	2,0	3,0	1,0	1,0
3	2,0	1,0	1,0	4,1
4	2,0	1,0	1,0	1,0

x_1x_2 / Q	Q⁺, y			
	00	01	11	10
1	2,0	1,0	1,0	1,0
2	2,0	3,0	1,0	1,0
3	2,0	1,0	1,0	1,1

stări echivalente: 1 = 4

Fig. 7.40 *Tabelul tranzițiilor şi reducerea stărilor*

Formele de undă care evidențiază funcționarea acestui circuit sunt date în figura 7.38. Se observă că o secvență de forma $s \to a \to p$ activează ieşirea prin 1 logic pe durata unei perioade de ceas, în timp ce secvența $s \to a \to a \to p$, sau oricare alta, nu activează semnalul de ieşire.

Diagrama stărilor este dată în figura 7.39, unde, pentru a uşura înțelegerea, s-au folosit ca etichete pe arce nu combinațiile binare ale intrărilor x_1 şi x_2, ci notațiile *s*, *a*, *l* şi *p*, conform caietului de sarcini.

Figura 7.40 reprezintă tabelul tranzițiilor dedus din diagrama stărilor şi tabelul redus, după ce s-a stabilit că stările 1 şi 4 sunt echivalente (se observă că prima linie din tabel este identică cu ultima). Evident că tabelului redus îi corespunde o diagramă redusă a stărilor. Mai observăm că sistemul are o stare care nu este total stabilă. Stării prezente 3 din tabelul redus nu îi corespunde nici o stare următoare 3, pentru nici una din combinațiile binare posibile pe intrări. Prezența acestei stări indică comportamentul sincron al sistemului.

Dacă se face o codificare binară minimală a stărilor şi starea 1 primeşte codul 00, starea 2 codul 01, iar starea 3 codul 11, atunci, prin descompunerea tabelului redus în 3 diagrame Veitch-Karnaugh şi minimizarea funcțiilor Q_1^+, Q_2^+ şi *y* obținem expresiile:

$$Q_1^+ = D_1 = \overline{x}_1 x_2 \overline{Q}_1 Q_2 \,,\; Q_2^+ = D_2 = \overline{x}_1 \overline{x}_2 + \overline{x}_1 x_2 \overline{Q}_1 Q_2 \,,\;\; \text{şi} \;\; y = x_1 \overline{x}_2 Q_1$$

Lăsăm în seama cititorului implementarea schemei logice pentru acest circuit, precum şi folosirea codului 1 prin 3 pentru implementarea schemei logice cu multiplexoare sau cu demultiplexoare.

O variantă foarte interesantă şi simplă, de implementare a sistemului din exemplul anterior, este implementarea cu registre de deplasare.

Vom spune că o succesiune de *k* stări de intrare este o **secvență de intrare determinantă** de lungime *k*, dacă ieşirea generată de această secvență este independentă de starea internă inițială a sistemului secvențial sincron cu comportament sincron.

Sistemul secvențial sincron are **memorie finită** dacă orice secvență de intrare de lungime egală sau mai mare decât *k* este determinantă, unde *k* este un întreg fix caracteristic sistemului.

Metoda de sinteză cu registre de deplasare presupune parcurgerea următoarelor etape:

1. se pune în evidență secvența de intrare determinantă de lungime *k* pentru care $y = 1$
2. pentru un sistem cu *n* intrări, se introduc în paralel *n* registre de deplasare de *k*-1 biți fiecare
3. se implementează logica combinațională care generează ieşirea.

Exemplul 7.8

Sistemul implementat în exemplul 7.7 are o secvență de intrare determinantă de lungime $k = 3$ şi orice secvență de intrare terminată prin $s \to a \to p$ sau $x_1x_2 = 00 \to 01 \to 10$ produce la ieşire $y = 1$, indiferent de starea internă inițială a sistemului.

Sistemul are $n = 2$ intrări, deci se introduc 2 registre de deplasare de câte 2 biți fiecare. Secvența de intrare va fi memorată în registre. După primul ceas, datele de pe intrări sunt memorate la ieşirile bistabilelor de la intrare, iar după al doilea ceas, ele vor fi memorate la ieşirile celorlalte două bistabile. Deci $x_1(t)x_2(t) = 00$. Datele care apar mai târziu vor fi memorate la ieşirile bistabilelor de la intrare, deci $x_1(t+1)x_2(t+1) = 01$, iar ultimele date care apar se regăsesc pe intrări: $x_1(t+2)x_2(t+2) = 10$. Conexiunile din figura 7.41 generează în această situație la ieşirea porții ŞI funcția $y(t+2) = 1$. Pentru orice altă secvență $y(t+2) = 0$.

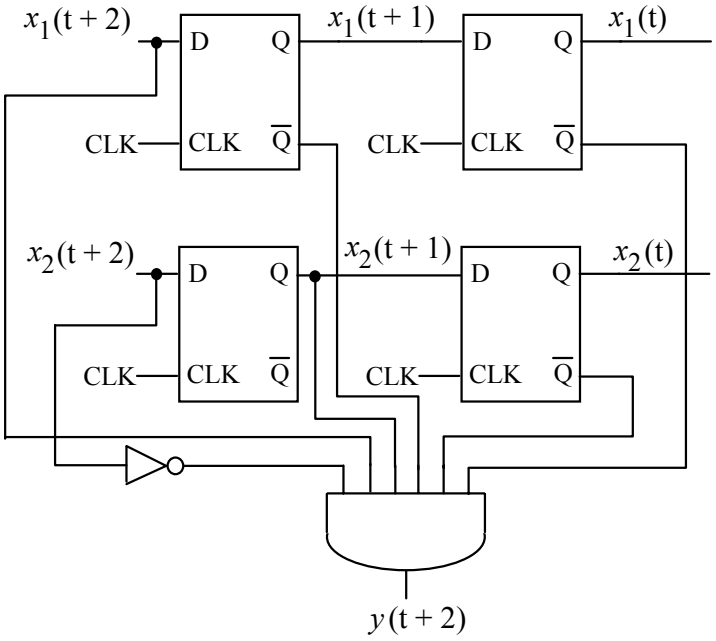


Fig. 7.41 *O soluție cu registre de deplasare*

Probleme

7.1 Să se facă sinteza cu bistabile JK a numărătorului sincron definit prin secvența: $Q_4Q_2Q_1 = 111 \to 100 \to 011 \to 001 \to 000 \to 111$. Să se facă analiza completă a circuitului obținut. Să se repete problema folosind bistabile D şi să se compare soluțiile obținute în cele două cazuri.

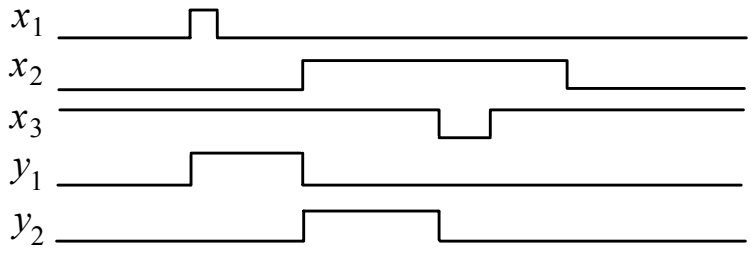
7.2 Să se proiecteze un sistem numeric cu 2 intrări, care nu se modifică simultan, şi o ieşire, care capătă valoarea logică 1 la sfârşitul secvenței de intrare $x_1x_2 = 00 \to 01 \to 11$ şi îşi păstrează valoarea pentru orice variație a intrărilor, până la apariția secvenței $x_1x_2 = 11 \to 10 \to 00$, la sfârşitul căreia capătă din nou valoarea logică 0.

([Mange, 1987])

7.3 Să se proiecteze un sistem numeric care să asigure funcționarea automată a barierelor la trecerea peste calea ferată. Sistemul are două intrări, date de stările unor contacte amplasate de o parte şi de alta a şoselei, şi o ieşire care comandă ridicarea sau coborârea barierelor.

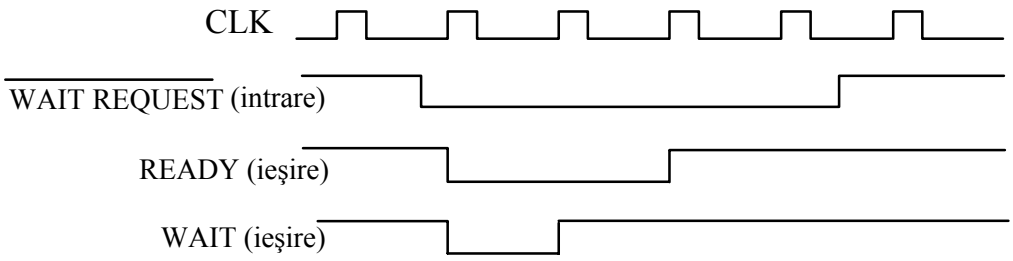
([Mange, 1987])

7.4 Să se proiecteze un sistem numeric secvențial cu 3 intrări şi 2 ieşiri, care are un comportament descris de formele de undă din figura de mai jos:



([Mange, 1987])

7.5 Să se facă sinteza unui circuit de generare a stărilor de WAIT pentru microprocesoarele 8080, 8085 şi Z80, folosind formele de undă din figura de mai jos. Încercați şi o soluție de implementare cu registre de deplasare.



7.6 Să se implementeze un numărător sincron cu 6 stări, fără coduri precizate, folosind structura de registru serie din figura 7.1. În acest scop se foloseşte diagrama stărilor din figura 7.3, şi se stabileşte un drum închis care parcurge 6 stări, intrarea *A* fiind generată de o funcție binară conform tranzițiilor dorite.

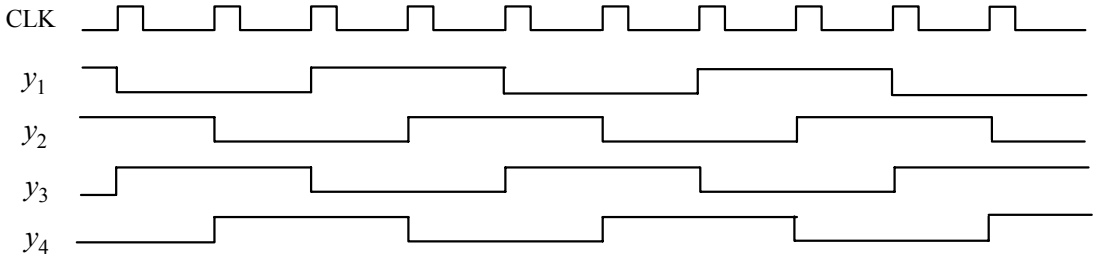
7.7 Să se proiecteze un sistem numeric secvențial cu 2 intrări, notate cu x_1 şi x_2, şi o ieşire y, care detectează orice secvență de 4 stări succesive de intrare care verifică relația $x_1 = x_2$. La detectarea acestei secvențe ieşirea y capătă valoarea logică 1, şi o păstrează atât timp cât $x_1 = x_2$.

([Mange, 1987])

7.8 Să se proiecteze un numărător sincron programabil care admite unul dintre următoarele moduri de funcționare: blocat în starea de repaus, numărător cu 2 stări, numărător cu 3 stări sau numărător cu 4 stări.

([Mange, 1987])

7.9 Să se proiecteze un circuit destinat comenzii unui motor pas cu pas, care să genereze formele de undă din figura de mai jos.



7.10 Proiectați un circuit secvențial sincron cu o intrare şi o ieşire. Ieşirea trebuie să devină 1 ori de câte ori secvența de intrare primită este 1100 sau 0011, iar în rest rămâne 0.

([Wilkinson, 2002])

7.11 Proiectați un generator de impulsuri care generează un puls cu lățimea de n perioade de ceas, unde n este intrarea unui circuit pe trei linii $(0 < n \le 7)$.

([Wilkinson, 2002])

7.12 Să se proiecteze un numărător sincron programabil cu două intrări x_1 şi x_2. Dacă $x_1 = 0$ numărătorul este modulo 3, iar dacă $x_1 = 1$ numărătorul este modulo 4. Dacă $x_2 = 0$ circuitul numără în sens crescător, iar dacă $x_2 = 1$ circuitul numără în sens descrescător.

([Friedman, 1986])

7.13 Un sistem secvențial sincron cu o singură intrare trebuie să detecteze orice secvență formată din cel puțin 2 nivele de 1 logic sau din cel puțin 4 nivele de 0 logic. În oricare din cele două situații ieşirea trece în 1 logic şi îşi păstrează valoarea până la următoarea modificare a intrării. Să se implementeze schema logică a acestui sistem.

7.14 Să se facă sinteza unui sistem secvențial sincron care filtrează datele de pe singura intrare, în scopul eliminării tranzițiilor care au durata unei perioade de ceas. Ieşirea îşi modifică valoarea logică dacă intrarea are o valoare logică contrară timp de cel puțin 2 perioade de ceas.