4 SISTEME SECVENȚIALE

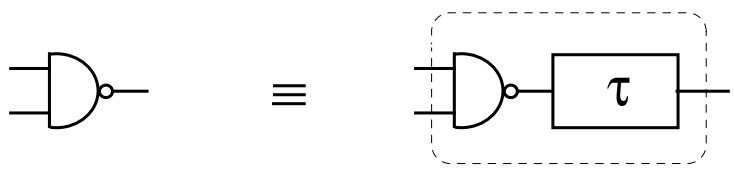
Aplicațiile din acest capitol îşi propun să prezinte circuite secvențiale sincrone şi asincrone cu un nivel de structurare superior față de cel al circuitelor prezentate în capitolul anterior. Pentru sinteza şi analiza acestor circuite există algoritmi, în timp ce un circuit de impuls este folosit atunci când este cazul, prin simpla introducere a lui în sistem.

4.1 Considerații teoretice

4.1.1 Sisteme secvențiale asincrone

Circuitele prezentate în capitolul anterior erau circuite secvențiale, adică circuite cu porți, care conțineau bucle de reacție, conexiuni de la ieşirile sistemului la intrări. Funcționarea acestor circuite nu poate fi explicată decât dacă ținem seama de timpii de propagare prin porți, condiție ignorată de obicei la sistemele combinaționale. Modelul logic asincron, care este folosit pentru analiza şi sinteza acestor sisteme, echivalează poarta logică reală cu o poartă logică ideală(realizează aceeaşi funcție logică, dar timpul de propagare este nul) şi un element de întârziere, aşa cum se poate vedea în figura 4.1.

Funcționarea latch-ului cu inversoare, prezentat în capitolul anterior, este uşor explicată folosind modelul logic asincron. Bistabilele de tip D sau JK au în componență structuri de porți cu numeroase bucle interne, deci blocuri asincrone, deşi pe ansamblu, ele au fost astfel construite încât momentul de timp la care se modifică ieşirile să fie



poarta reală poarta ideală + element de întârziere

Fig. 4.1 *Modelul logic asincron*

un front al semnalului de ceas. Dacă însă bistabilele componente ale unui sistem secvențial complex nu primesc acelaşi semnal de ceas, deci nu comută simultan, atunci spunem că sistemul secvențial este asincron.

Un astfel de sistem asincron este numărătorul asincron, sau divizorul de frecvență. Bistabilele din numărătoarele asincrone funcționează într-un singur fel, şi anume, ca divizoare de frecvență prin 2. Prin urmare, bistabilele din structură, fie că sunt de tip D sau de tip JK, sunt transformate în bistabile de tip T, având intrarea de date T conectată la 1 logic. Schimbându-şi starea pe fiecare front activ al ceasului, ele divizează prin 2 frecvența semnalului aplicat pe intrarea de ceas.

Figura 4.2 reprezintă schema unui divizor de frecvență de 4 biți realizat cu bistabile T. Dacă acceptăm că bistabilele comută pe frontul descrescător al ceasului, atunci se obțin formele de undă prezentate în figura 4.3. La scara de timp la care s-a făcut simularea funcționării nu se observă nimic deosebit şi diagrama stărilor pare a fi cea ideală. Dacă privim însă printr-o "lupă de timp" porțiunea selectată, vom obține formele de undă din figura 4.4. Frontul descrescător al semnalului CLK produce comutarea primului bistabil , iar ieşirea lui, notată cu Q_1, se modifică cu o mică întârziere, dată de timpul de propagare a informației prin porțile din structura bistabilului. Semnalul Q_1 este ceasul pentru bistabilul 2, iar semnalul Q_2 se modifică şi el cu o mică întârziere față de Q_1, şi aşa mai departe. Deci pe lângă cele 16 stări permanente, există un număr de stări tranzitorii.

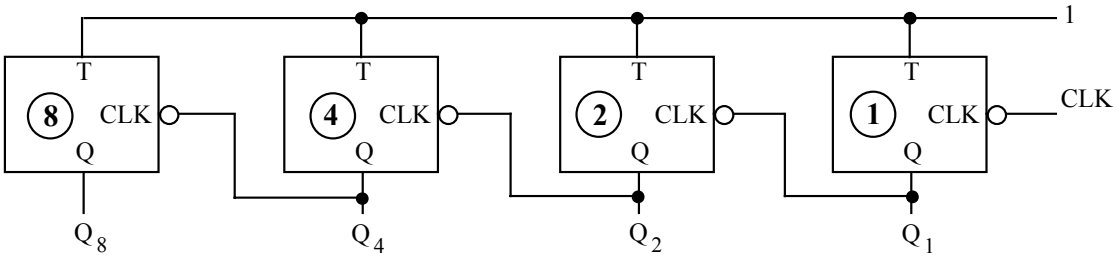


Fig. 4.2 *Divizor de frecvență de 4 biți*

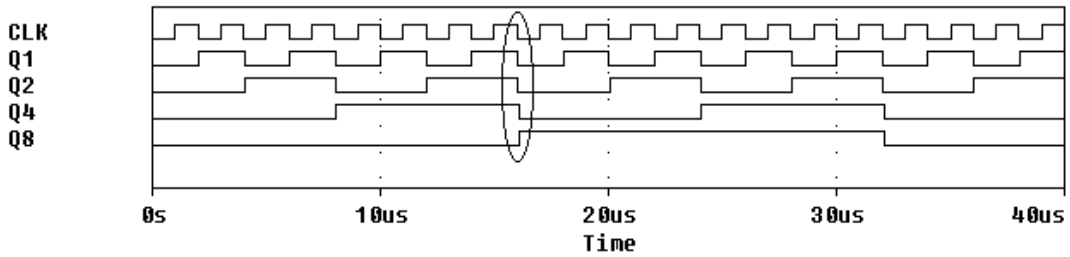


Fig. 4.3 *Forme de undă pentru divizorul de frecvență de 4 biți*

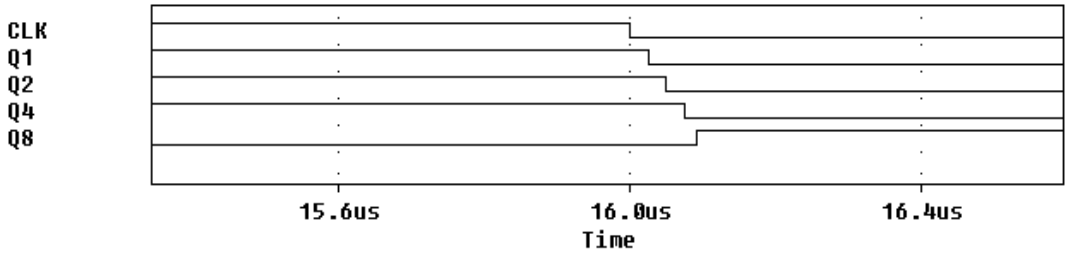
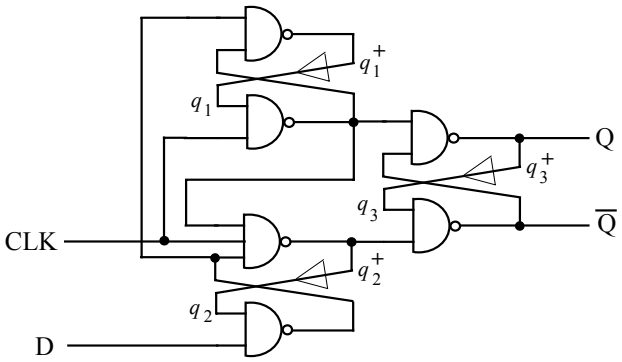


Fig. 4.4 *Forme de undă văzute printr-o "lupă de timp"*

Având în vedere funcționarea lor, analiza şi sinteza numărătoarelor asincrone nu ridică probleme deosebite. În schimb, analiza şi sinteza sistemelor asincrone cu porți logice este dificilă, pentru că timpul nu este discretizat ca la sistemele sincrone. Starea următoare nu apare după o perioadă de ceas, ci poate apare după un multiplu de τ, τ fiind timpul de propagare prin poartă.

Circuitul din figura 4.5, de exemplu, poate fi analizat prin stabilirea numărului minim de bucle. Considerând porțile ideale şi singurele întârzieri fiind introduse de cele 3 bucle ale latch-urilor $\overline{S}\,\overline{R}$, tabelul tranzițiilor este cel alăturat circuitului din figură. Tabelul conține curse(starea următoare diferă prin cel puțin 2 biți față de starea prezentă), dar ele nu sunt critice(cursa critică apare dacă există posibilitatea ca stările următoare să fie diferite, adică să avem o funcționare imprevizibilă). Tabelul tranzițiilor poate fi redus prin eliminarea stărilor care nu sunt total stabile (neîncercuite).



CLK D / $q_1\,q_2\,q_3$	$q_1^+q_2^+q_3^+$ 00	01	11	10
000	010	010	(000)	(000)
001	011	011	000	000
010	(010)	110	110	000
011	(011)	111	111	000
100	010	010	111	111
101	011	011	111	111
110	010	(110)	111	111
111	011	(111)	(111)	(111)

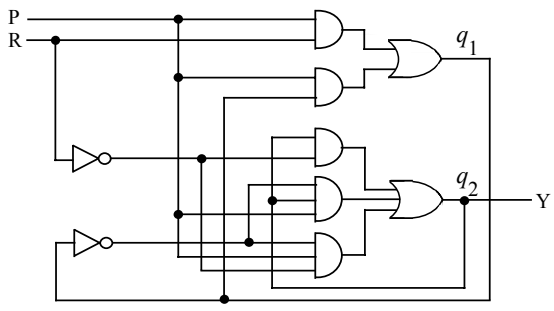
Fig. 4.5 *Structura unui bistabil de tip D şi tabelul tranzițiilor*

Pentru sinteza unui sistem secvențial asincron, codificarea stărilor din tabelul redus este importantă. Pentru eliminarea curselor critice, putem fi obligați să introducem stări suplimentare. Exemplul din figurile 4.6 şi 4.7 elimină complet cursele prin codificarea adiacentă a stărilor între care au loc tranziții.

P R / q	q^+ 00	01	11	10	Y
IDLE	(IDLE)	(IDLE)	RES	PLS	0
PLS	(PLS)	IDLE	RES	(PLS)	1
RES	IDLE	IDLE	(RES)	(RES)	0

P R / q	q^+ 00	01	11	10	Y
IDLE	(IDLE)	(IDLE)	RES	PLS	0
PLS	(PLS)	IDLE	RESA	(PLS)	1
RESA	–	–	RES	–	–
RES	IDLE	IDLE	(RES)	(RES)	0

Fig. 4.6 *Tabelul redus şi tabelul fără curse*



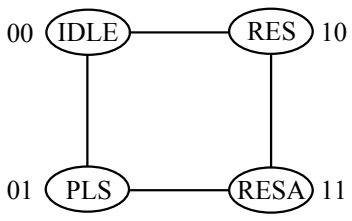


Fig. 4.7 *Schema logică a circuitului şi diagrama de adiacență a stărilor*

4.1.2 Sisteme secvențiale sincrone

Analiza şi sinteza sistemelor sincrone este mult mai comodă pentru proiectant, pentru că putem ignora întârzierile logicii combinaționale, cu condiția ca duratele stărilor, date de frecvența ceasului, să fie semnificativ mai mari decât întârzierile prin porți. Timpul se transformă astfel dintr-o mărime continuă într-una discretă, perfect controlabilă prin semnalul de ceas aplicat sistemului.

Sistemele secvențiale sincrone, ca şi cele asincrone de altfel, pot fi uşor tratate folosind teoria automatelor cu stări finite. Un automat cu stări finite se defineşte formal prin cvintuplul $A = (X, Y, Q, \lambda, \delta)$, unde entitățile componente au următoarea semnificație:

$X = \{x_1, x_2, ..., x_n\}$ - mulțimea configurațiilor binare de intrare,

$Y = \{y_1, y_2, ..., y_r\}$ - mulțimea configurațiilor binare de ieşire,

$Q = \{q_1, q_2, ..., q_p\}$ - mulțimea configurațiilor binare de stare,

$\lambda : X \times Q \rightarrow Q$ - funcția de tranziție a stărilor,

$\delta : X \times Q \rightarrow Y$ - funcția de tranziție a ieşirilor.

Datorită faptului că mulțimile X, Y şi Q sunt finite, circuitul se numeşte automat cu stări finite. Spațiul timpului nu apare explicit în descrierea de mai sus. El este discret şi este format din mulțimea numerelor întregi care semnifică multiplul de T, unde T este perioada după care se comandă o nouă modificare în circuit. Funcțiile de tranziție se pot defini şi reprezenta prin tabele de tranziții, grafuri, sau organigrame. Schema din figura 4.8 reprezintă un automat finit ale cărui ieşiri apar cu întârziere deoarece sunt trecute prin memorie. Ele pot fi obținute şi imediat, de la ieşirea CLC-ului.

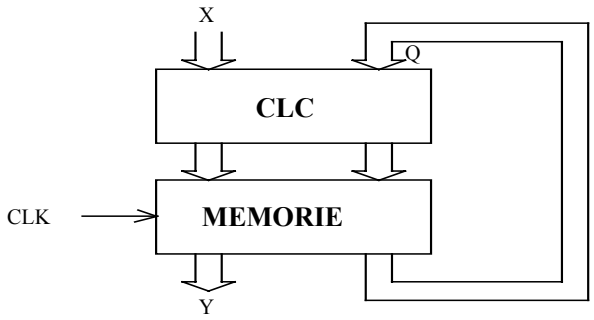


Fig. 4.8 *Structura generală a unui automat finit*

Fără a fi absolut necesară, această separare în două blocuri funcționale, logica combinațională şi blocul de memorare, este deosebit de utilă în proiectare. Aceasta, deoarece blocul de memorare poate fi definit foarte simplu, ca un registru paralel cu un număr suficient de bistabile, în funcție de codificarea adoptată pentru stările sistemului, iar blocul de prelucrare este format dintr-o logică combinațională mai complexă şi mai greu de definit. Efortul de proiectare este orientat spre sinteza circuitului combinațional.

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

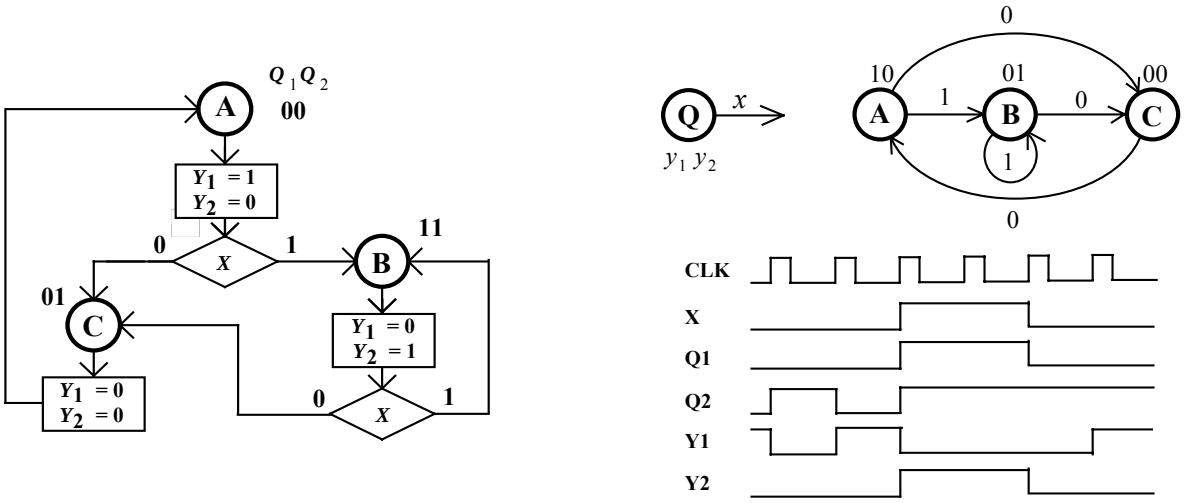


Fig. 4.9 *Exemplu de automat finit şi câteva moduri de reprezentare*

elementelor de memorie, deci a bistabilelor din structură, şi se reduce numărul funcțiilor de excitație, cu alte cuvinte complexitatea circuitului combinațional.

Poate că cea mai importantă chestiune este codificarea stărilor rămase după reducere. Dacă variabilele de intrare sunt sincrone, adică se modifică în funcție de semnalul de ceas, atunci 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. Dar în orice situație, circuitul obținut va funcționa corect. Dacă însă variabilele de intrare sunt asincrone, adică se pot modifica oricând în timp, apare din nou problema întâlnită la cursele critice, iar rezolvarea ei se face prin alocarea unor coduri adiacente stărilor care urmează testării variabilei asincrone (principiul dependenței reduse de o variabilă). La nevoie, se pot introduce şi aici stări suplimentare.

Sistemul din figura 4.9 are o singură intrare notată cu x, să presupunem deocamdată că este sincronă, două ieşiri notate cu y_1 şi y_2, şi un număr de trei stări notate cu A, B şi C. O codificare binară minimală a stărilor se poate face folosind numai 2 biți, adică două circuite bistabile, având ieşirile Q_1 şi Q_2.

4.1.2 Hazard

Prin hazard sau risc, înțelegem posibilitatea de modificare neaşteptată a ieşirii, pentru o durată foarte mică de timp, datorită întârzierilor prin porți. El poate fi static, dacă are loc o singură comutare a ieşirii la modificarea unei variabile, sau dinamic, dacă apar comutări multiple datorită unei singure tranziții a unei intrări. Structurile combinaționale cu două nivele de logică, care pot fi implementate într-o structură programabilă PLD, nu generează hazard dinamic.

În sistemele secvențiale sincrone nu se pune de obicei problema analizei hazardului, pentru că frontul activ al ceasului se aplică după ce logica combinațională a livrat valorile logice aşteptate ale funcțiilor de excitație. În sistemele asincrone însă, această analiză a hazardului şi găsirea unor modalități de eliminare a lui, este absolut necesară.

4.2 Demonstrații practice

Considerațiile asupra alimentării panoului logic, formulate în primul capitol, rămân valabile şi aici. Panoul logic conține circuitele integrate **MMC 4013** şi **MMC 4027**, studiate în capitolul anterior. Pentru implementarea automatului finit prezentat mai sus, se foloseşte un panou care conține, conform figurii 4.14, 4 circuite integrate CMOS: **MMC 4013**, **MMC 4052**(2 buc.) şi **MMC 4011**. Circuitul MMC 4052 conține câte 2 multiplexoare, dar foaia lui de catalog nu a mai fost dată, deoarece conexiunile conform schemei din figură sunt deja realizate. Porțile ŞI-NU ale circuitului MMC 4011 sunt folosite pentru generarea semnalului de CLK, prin apăsarea butonului cu revenire de pe panou. Biții de stare şi de ieşire sunt vizualizați prin intermediul a 4 LED-uri. Starea 1 logic este semnalizată prin aprinderea LED-ului corespunzător, iar cea de 0 logic prin stingerea lui. **Montajul se poate alimenta cu orice tensiune continuă, cuprinsă între 5 şi 15 Vcc**.

Un alt panou logic, destinat implementării aceluiaşi automat cu bistabile de tip D şi memorie, este implementat după schema din figura 4.16. Panoul conține două circuite de memorie: **82S147**, care este o memorie PROM şi **MMN 2114**, o memorie RAM statică. Placa mai conține circuitele **MMC 4013**, **MMC 4011** şi **MMC 4066**, ultimul având în structură 4 comutatoare CMOS necesare pentru înscrierea, respectiv citirea datelor din memoria RAM. Vizualizarea stării şi a ieşirii se face tot cu ajutorul a 4 LED-uri. Despre memorii vom discuta în capitolul următor, dar circuitul este deja cablat şi utilizarea lui aici se poate face fără a avea prea multe cunoştințe despre memorii. **Montajul se alimentează cu tensiunea de 5 Vcc**, datorită prezenței circuitelor de memorie. De fapt, tensiunea sursei este de circa 5,6 - 5,7 V din cauza diodei serie de protecție la alimentare inversă. Din acest motiv **se măsoară cu voltmetrul tensiunea de 5 Vcc între catodul diodei de protecție şi masă.**

4.2.1 Se implementează numărătorul asincron din figura 4.1, folosind bistabile de tip D. Rezultă circuitul din figura 4.10. Se verifică funcționarea montajului. Se pot vizualiza stările tranzitorii? Modificați circuitul pentru a obține un numărător asincron cu 16 stări care să numere în sens descrescător. Repetați implementările folosind bistabile de tip JK şi verificați funcționarea corectă a circuitelor.

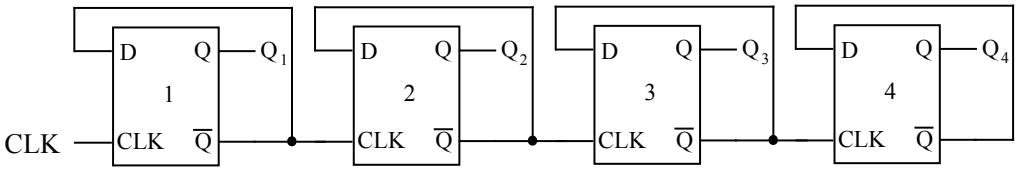


Fig. 4.10 *Schema logică a divizorului de frecvență cu bistabile de tip D*

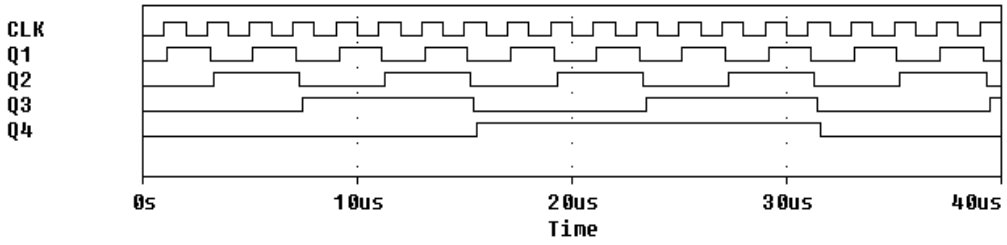


Fig. 4.11 *Forme de undă pentru circuitul din figura 4.10* □

4.2.2 Se realizează numărătorul sincron din figura 4.12, care realizează tranzițiile $Q_1Q_2 = 00 \to 10 \to 11 \to 01 \to 00$. Sinteza circutului se face cu ajutorul tabelului tranzițiilor din figură. Folosind tabelul tranzițiilor pentru bistabilul de tip JK se deduc funcțiile de excitație pentru fiecare dintre cele două bistabile. De ce spunem că este un numărător sincron? Refaceți circuitul pentru a modifica sensul de numărare şi verificați funcționarea lui. Puteți construi acelaşi numărător, dar folosind bistabile de tip D? Care dintre cele două soluții este mai avantajoasă şi de ce?

Q_1	Q_2	Q_1^+	Q_2^+	J_1	K_1	J_2	K_2
0	0	1	0	1	x	0	x
1	0	1	1	x	0	1	x
1	1	0	1	x	1	x	0
0	1	0	0	0	x	x	1

$J_1 = \overline{Q_2}$ $K_1 = Q_2$

$J_2 = Q_1$ $K_2 = \overline{Q_1}$

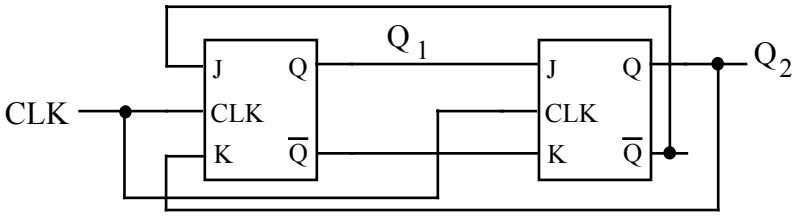


Fig. 4.12 *Schema logică a unui numărător sincron cu bistabile de tip JK*

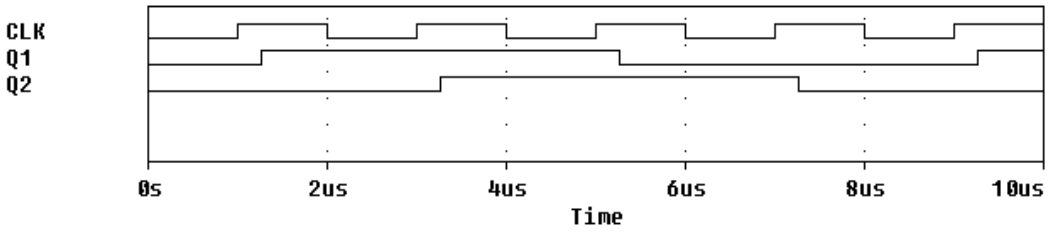


Fig. 4.13 *Forme de undă pentru circuitul din figura 4.12* □

4.2.3 Se foloseşte panoul logic cu bistabile D şi multiplexoare pentru implementarea automatului finit reprezentat în figura 4.9. Schema logică a circuitului este dată în figura 4.14. Se verifică funcționarea corectă a montajului prin urmărirea tuturor tranzițiilor posibile din organigramă sau tabel. Figura 4.15 reprezintă tabelul tranzițiilor, tabel ce a fost construit pe baza tranzițiilor din organigramă şi care este foarte util pentru sinteza schemei logice a circuitului.

Simbolul *d* poate fi 0 sau 1 logic(don' t care). S-a folosit această notație pentru a evita confuzia cu semnalul de intrare, notat aici cu X. Implementarea memoriei se face cu doi bistabili de tip D care comută sincron. Circuitul logic combinațional are intrările X, Q_1 şi Q_2 şi trebuie să genereze la ieşire funcțiile binare $Q_1^+ = D_1$ şi $Q_2^+ = D_2$. Nu este necesară minimizarea funcțiilor binare D_1 şi D_2 pentru că toți termenii canonici sunt disponibili la ieşirile multiplexoarelor. Reacția de la ieşirea memoriei la intrarea în CLC se realizează prin conectarea ieşirilor Q_1 şi Q_2 la intrările de selecție ale multiplexoarelor. Conexiunile la intrările multiplexoarelor se realizează în funcție de alegerea conexiunilor de reacție.

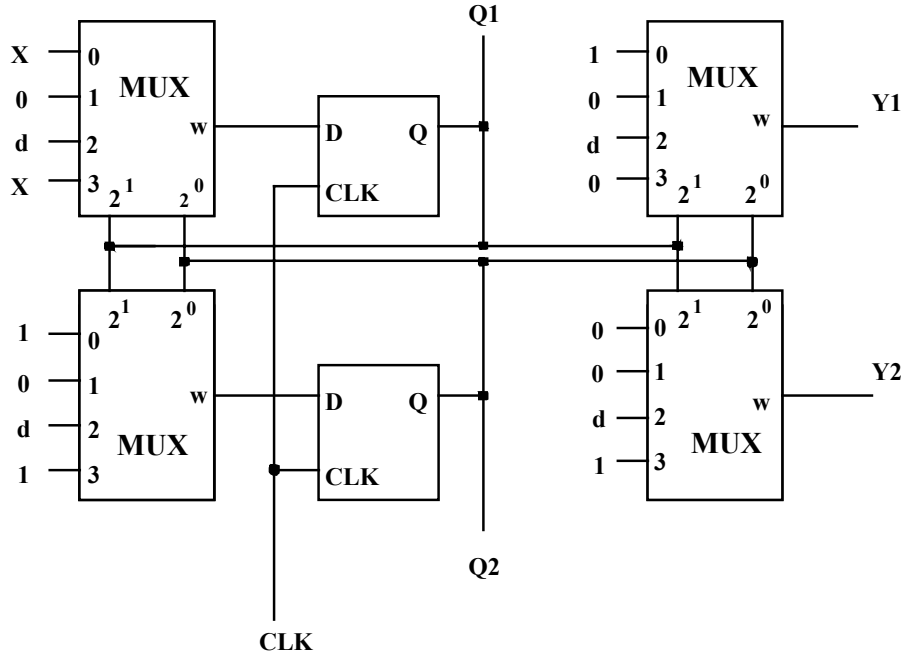


Fig. 4.14 *Schema logică a automatului implementat cu multiplexoare*

X	Q_1	Q_2	Q_1^+	Q_2^+	Y_1	Y_2
0	0	0	0	1	1	0
1	0	0	1	1	1	0
d	0	1	0	0	0	0
0	1	1	0	1	0	1
1	1	1	1	1	0	1

Fig. 4.15 *Tabelul tranzițiilor*

Dacă privim tabelul tranzițiilor observăm că pentru starea prezentă A, codificată prin $Q_1Q_2 = 00$, starea următoare este fie B(codul $Q_1Q_2 = 11$), fie C(codul $Q_1Q_2 = 01$). Decizia este luată în funcție de valoarea logică a variabilei de intrare X. Se vede că pentru stările B şi C valoarea lui Q_2 este o constantă, $Q_2 = 1$, iar valoarea lui Q_1 este dependentă de valoarea lui X: pentru $X = 0$, $Q_1 = 0$, iar pentru $X = 1$, $Q_1 = 1$. Deci putem spune că $Q_1 = X$. Celelalte două multiplexoare au prima intrare(notată cu 0) conectată la valorile logice ale ieşirilor în starea prezentă A, adică $Y_1 = 1$ şi $Y_2 = 0$. Acelaşi raționament se face şi pentru celelalte stări. □

4.2.4 Se foloseşte panoul logic cu bistabile D şi memorii. Cele două circuite integrate, memoria PROM **82S147** şi memoria RAM **MMN 2114**, sunt destinate implementării logicii combinaționale a automatului finit. Blocul de memorie din figura 4.8 este un registru destinat memorării stării curente şi este realizat, ca şi până acum, cu cele două bistabile de tip D. Memoria de tip ROM este un circuit combinațional, care furnizează la ieşiri informația conținută în harta memoriei (tabelul din figura 4.17), în timp ce memoria RAM este un emulator de ROM, adică un circuit care simulează prezența unei memorii ROM în circuit. Memoria RAM nu conține o informație utilă la cuplarea alimentării şi de aceea în primul rând trebuie să înscriem datele din tabel în memorie (vezi subcapitolul 5.1).

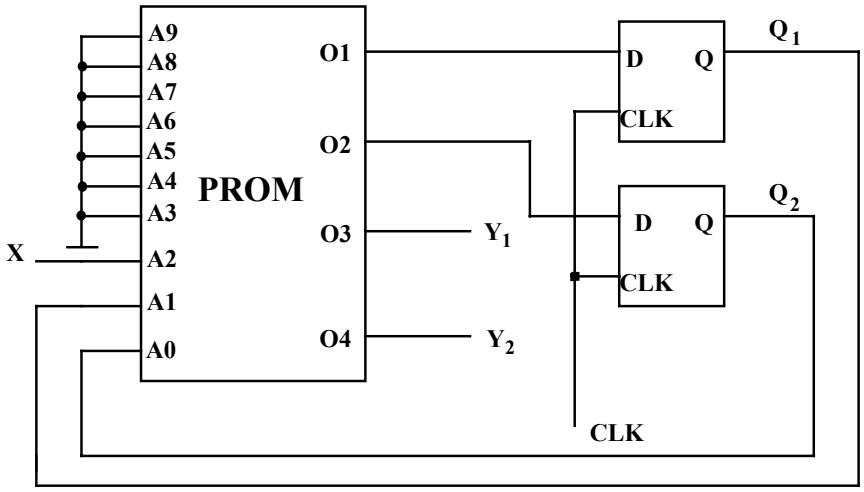


Fig. 4.16 *Schema logică a automatului implementat cu memorie PROM*

A2	A1	A0	O1	O2	O3	O4
X	Q_1	Q_2	Q_1^+	Q_2^+	Y_1	Y_2
0	0	0	0	1	1	0
0	0	1	0	0	0	0
0	1	0	d	d	d	d
0	1	1	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	0
1	1	0	d	d	d	d
1	1	1	1	1	0	1

Fig. 4.17 *Harta memoriei pentru schema logică din figura 4.16*

Circuitul 82S147 este realizat în tehnologie Schottky TTL şi conține 512 cuvinte de câte 8 biți. Memoria RAM MMN 2114 este realizată în tehnologie NMOS şi conține 1024 cuvinte de câte 4 biți. Memoria PROM din figura 4.16 este un circuit generic de 1024 cuvinte (10 linii de adresare $A_9...A_0$, care sunt intrări) de câte 4 biți (4 linii de date $O_4...O_1$, care sunt ieşiri). Automatul conceput foloseşte numai 6 dintre cele 1024 cuvinte, conform tabelului de tranziții din figură.

Verificarea funcționării circuitului cu memorie PROM se face imediat, prin simpla verificare a tranzițiilor din tabel. Comentați cuplajul TTL – CMOS.

Pentru circuitul cu memorie RAM, trebuie să introducem de la început datele necesare în memorie. În acest scop, pentru a asigura adresa 0 de start a memoriei la cuplarea alimentării (A2 = A1 = A0 = 0), comutatorul S2 are pârghia înspre comutatoarele S1 şi S3(în jos), iar comutatorul S3 în dreapta (vezi fig. 4.18). Intrarea X este conectată la masă.

Cele 4 comutatoare notate cu S1 sunt destinate introducerii celor 4 biți pe cuvânt în memorie. Dacă pârghia este în jos, bitul corespunzător este pe 0 logic, iar dacă este în sus, pe 1 logic. După fixarea cuvântului dorit prin poziționarea celor 4 comutatoare, acesta este introdus în memorie prin ridicarea şi coborârea înapoi a pârghiei comutatorului S2 (activarea şi dezactivarea semnalului WE - WRITE ENABLE). Pe urmă se modifică adresa pentru introducerea unui nou cuvânt prin acționarea în cele două sensuri a comutatorului S3 (pentru a genera cele 2 fronturi ale semnalului de ceas CLK - CLOCK). Pentru introducerea datelor în memorie urmărim harta memoriei din figura 4.17. Dacă în locul comutatoarelor S2 şi S3 există butoane cu revenire, atunci ele sunt acționate o singură dată.

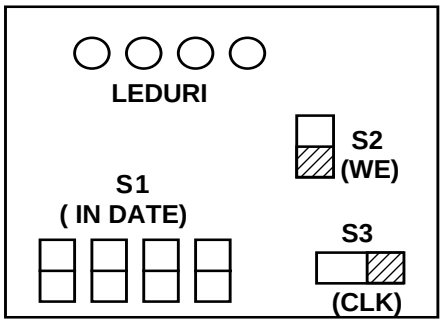


Fig. 4.18 *Amplasarea comutatoarelor şi a LED-urilor pe panou*

Datele se introduc de la comutatoarele S1, de la stânga la dreapta, în ordinea dată în tabel. Pentru înscrierea celor 6 cuvinte utile în memorie se parcurg următorii paşi:

a) se fixează comutatoarele S1 pe poziţiile **1001**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)

b) se fixează comutatoarele S1 pe poziţiile **0000**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)
se mută X pe 1 logic (borna de +5Vcc)

c) se fixează comutatoarele S1 pe poziţiile **1011**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)

d) se fixează comutatoarele S1 pe poziţiile **0111**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)
se mută X pe 0 logic (borna de masă)

e) se fixează comutatoarele S1 pe poziţiile **0101**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)
se mută X pe 1 logic (borna de +5Vcc)

f) se fixează comutatoarele S1 pe poziţiile **0000**
se acţionează în sus şi în jos comutatorul S2 (WE)
se acţionează stânga - dreapta comutatorul S3 (CLK)
se mută X pe 0 logic (borna de masă)

Din acest moment datele sunt stocate în memorie şi se verifică funcţionarea corectă a automatului, urmărind tranziţiile din organigramă. Întreruperea alimentării duce la pierderea informaţiei din memorie şi se impune repetarea operaţiilor de scriere a celor 6 cuvinte în memoria RAM. □

4.2.5 Comparaţi cele două tipuri de implementări prezentate: cu multiplexoare şi cu memorie. Ce avantaje şi dezavantaje prezintă fiecare dintre cele două soluţii? Propuneţi şi alte modalităţi de implementare a automatului şi schiţaţi schemele logice pentru fiecare caz în parte (folosiţi pentru implementarea blocului de memorie fie bistabile de tip JK, fie un registru, fie un numărător). □

4.2.6 Ce înseamnă codificare care urmăreşte principiul dependenței reduse față de o variabilă? În cazul nostru variabila X poate fi asincronă? Alegeți o altă codificare a stărilor şi arătați sub formă tabelară harta memoriei precum şi operațiunile necesare pentru înscrierea datelor în memorie. Verificați funcționarea circuitului obținut. □

4.3 Probleme rezolvate

4.3.1 Să se analizeze circuitul din figura 4.19 şi să se reprezinte formele de undă şi diagrama stărilor. Să se stabilească numărul de stări netranzitorii *P* şi raportul de divizare *N*.

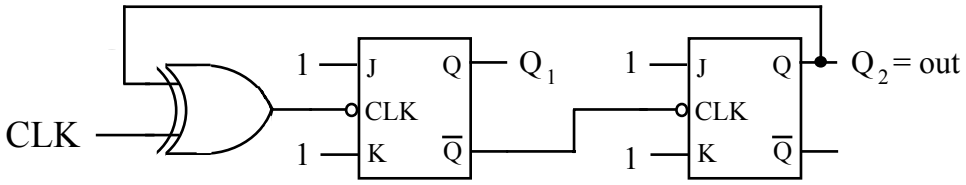


Fig. 4.19 *Schema logică a unui divizor de frecvență realizat cu bistabile de tip JK*

Rezolvare:

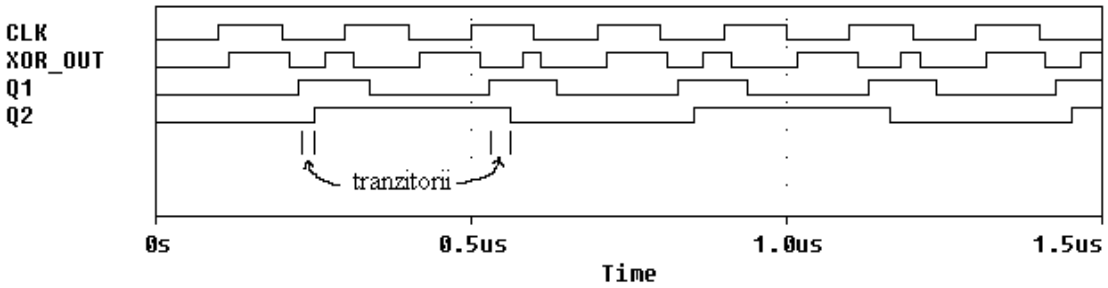


Fig. 4.20 *Forme de undă pentru circuitul din figura 4.19*

Analiza circuitului se face prin desenarea formelor de undă, ținând seamă de întârzierile prin porți şi bistabile. Figura 4.20 prezintă analiza PSpice a circuitului. Stările tranzitorii se datorează timpilor de propagare prin bistabile. Diagrama stărilor pentru circuitul analizat este: $Q_1Q_2 = 00 \to \underline{10} \to 11 \to 01 \to \underline{11} \to 10 \to 00$, unde stările tranzitorii sunt subliniate. Rezultă imediat $P = 4$ şi $N = 3$. □

4.3.2 Să se proiecteze un circuit basculant cu 3 bucle, unde Δ reprezintă întârzierea proprie circuitelor logice şi traseelor de conexiune. În repaus, intrările I_1, I_2 şi I_3 sunt în 0 logic. Atunci când intrarea I_i trece în 1 logic, ieşirea y_i trece în 1 logic dacă era în 0 logic,

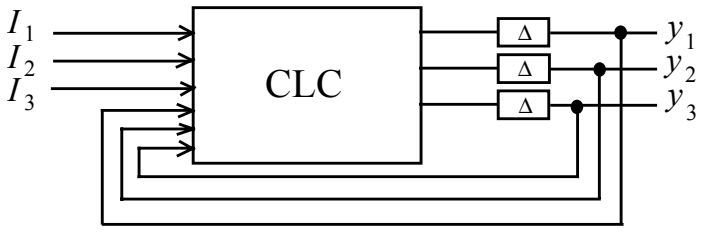


Fig. 4.21 *Un circuit secvențial asincron*

sau îşi păstrează starea dacă era în 1 logic. La revenirea intrării I_i în 0 logic, ieşirile y_i îşi păstrează vechea valoare. Admitem că se modifică o singură intrare, iar comenzile se exclud reciproc, adică $I_1I_2 = I_1I_3 = I_2I_3 = I_1I_2I_3 = 0$.

Rezolvare:

În conformitate cu cerințele problemei, formele de undă ar putea fi cele din figura 4.22. După evidențierea stărilor distincte ale sistemului se construieşte tabelul tranzițiilor şi al ieşirilor din figura 4.23, valorile din tabel fiind starea următoare şi ieşirile $Q^+ / y_1y_2y_3$:

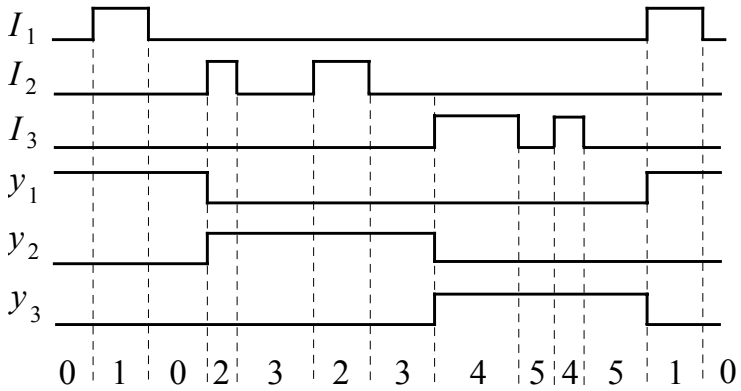


Fig. 4.22 *Stările sistemului deduse din formele de undă*

Q	$I_1I_2I_3$ = 000	001	010	011	100	101	110	111
	$Q^+, y_1y_2y_3$							
0	0,100	4,001	2,010	-	1,100	-	-	-
1	0,100	4,001	2,010	-	1,100	-	-	-
2	3,010	4,001	2,010	-	1,100	-	-	-
3	3,010	4,001	2,010	-	1,100	-	-	-
4	5,001	4,001	2,010	-	1,100	-	-	-
5	5,001	4,001	2,010	-	1,100	-	-	-

Fig. 4.23 *Tabelul tranzițiilor şi al ieşirilor*

Cele 6 stări ale sistemului sunt compatibile două câte două, deci avem în final 3 stări distincte. Pentru prevenirea hazardului şi a curselor critice, două stări succesive trebuie să aibă coduri adiacente(care diferă printr-un singur bit). Cum între fiecare dintre cele 3 stări există tranziții bilaterale, o codificare binară minimală cu 2 biți nu este posibilă. Folosim pentru fiecare stare un cod de 3 biți, astfel încât să existe o corespondență directă între variabilele de stare Q_1, Q_2 şi Q_3 şi ieşirile sistemului y_1, y_2 şi respectiv y_3: $Q_i = y_i$, unde i = 1, 2 şi 3. Atribuim deci, conform formelor de undă stabilite în figura 4.22, stărilor 0 şi 1 codul 100, stărilor 2 şi 3 codul 010, iar stărilor 4 şi 5 codul 001. Pentru a respecta şi condițiile de adiacență, mai introducem starea 000, care este atinsă la fiecare tranziție pentru foarte scurt timp(stare tranzitorie).

Tabelul tranzițiilor stărilor sau al ieşirilor este reprezentat în figura 4.24. Folosind diagrame Veitch-Karnaugh pentru 6 variabile, se minimizează funcțiile Q_i^+, i = 1, 2 şi 3, iar cu ajutorul ecuațiilor obținute se construieşte schema logică a circuitului, care este prezentată în figura 4.25. Aici întârzierile între stări nu sunt date de un ceas extern, ci numai de timpii de propagare prin porți.

$I_1I_2I_3$ / $Q_1Q_2Q_3$	000	001	010	011	100	101	110	111
	$Q_1^+Q_2^+Q_3^+$							
000	-	001	010	-	100	-	-	-
001	001	001	000	-	000	-	-	-
010	010	000	010	-	000	-	-	-
011	-	-	-	-	-	-	-	-
100	100	000	000	-	100	-	-	-
101	-	-	-	-	-	-	-	-
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	-	-	-

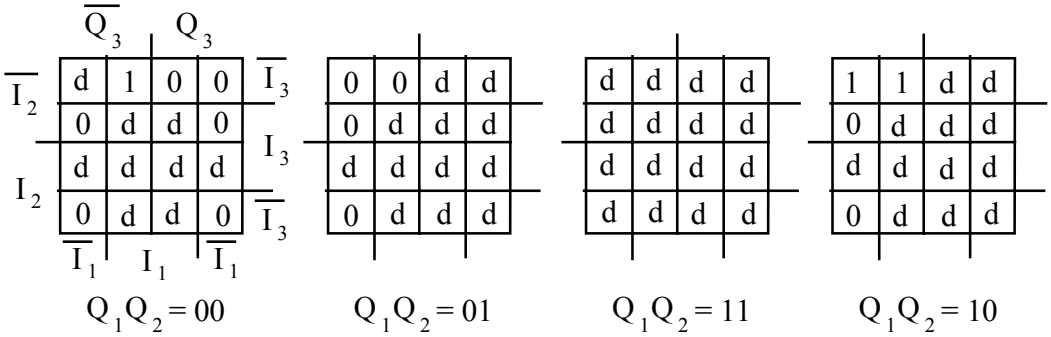


Fig. 4.24 *Tabelul tranziţiilor şi diagramele Karnaugh pentru funcţia* Q_1^+

Dacă grupăm zerourile din diagrama prezentată în figura de mai sus, obţinem expresia lui $\overline{Q_1^+}$: $\overline{Q_1^+} = I_2 + I_3 + Q_2 + Q_3$, sau $Q_1^+ = \overline{I_2 + I_3 + Q_2 + Q_3}$. În mod asemănător rezultă şi celelalte ecuaţii: $Q_2^+ = \overline{I_1 + I_3 + Q_1 + Q_3}$ şi $Q_3^+ = \overline{I_1 + I_2 + Q_1 + Q_2}$.

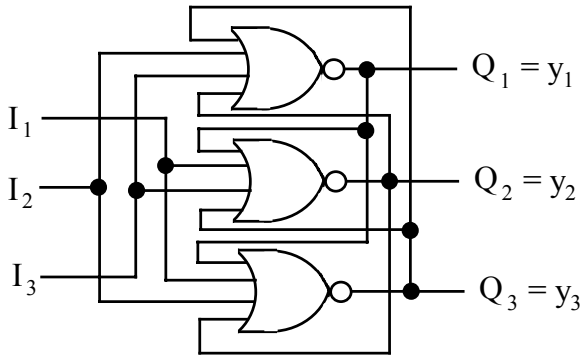


Fig. 4.25 *Schema logică a circuitului*

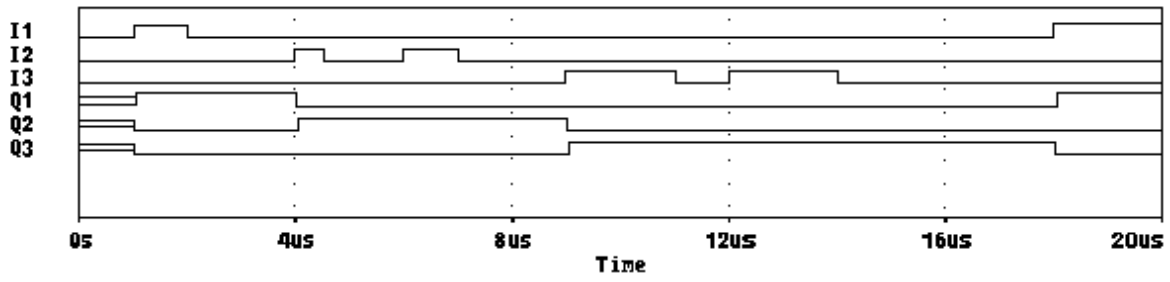


Fig. 4.26 *Formele de undă obţinute prin simulare PSPICE*

Figura 4.26 verifică corectitudinea sintezei prin analiză, folosind o simulare PSPICE. Formele de undă coincid cu cele din figura 4.22, cu excepția stării inițiale, în care toate intrările sunt la 0 logic. Ieşirile circuitului au o valoare logică necunoscută, pentru că circuitul nu a fost inițializat. În circuitele reale există cu siguranță la ieşiri una dintre cele două valori logice posibile, 0 sau 1. Aceeaşi problemă apare şi la sistemele secvențiale sincrone cu bistabile integrate. □

4.3.3 Să se analizeze sistemul secvențial din figura 4.27, reprezentând cronograma, tabelul tranzițiilor şi diagrama stărilor acestuia. Arătați că circuitul este un oscilator şi stabiliți perioada de oscilație.

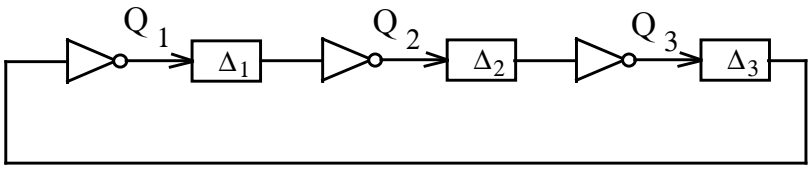


Fig. 4.27 *Schema logică a circuitului*

Rezolvare:

Dacă ținem seama de întârzierile prin porți, rezultă formele de undă din figura 4.28. Semnalele la ieşirile porților sunt periodice, cu perioada $T = 2 \cdot (\Delta_1 + \Delta_2 + \Delta_3)$. Tabelul tranzițiilor şi diagrama stărilor sunt prezentate în figura 4.29.

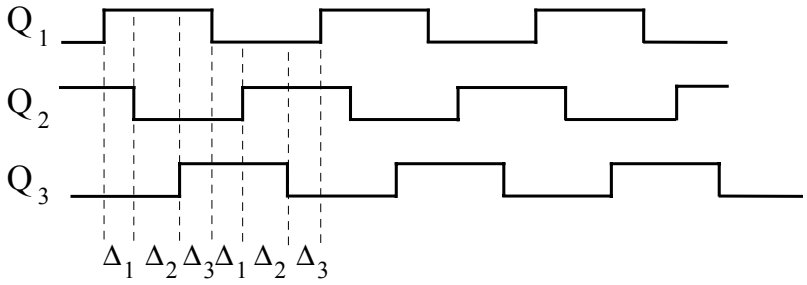


Fig. 4.28 *Cronograma circuitului din figura 4.27*

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$
000	-
001	011
010	110
011	010
100	101
101	001
110	100
111	-

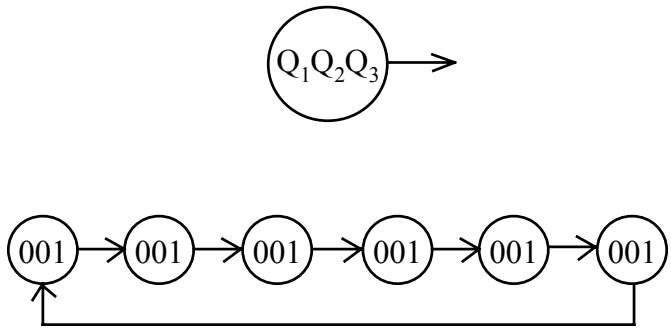


Fig. 4.29 *Tabelul tranzițiilor şi diagrama stărilor* □

4.3.4 Să se proiecteze un bistabil D cu basculare dublă pe front, respectiv care basculează pe ambele fronturi ale semnalului de ceas.

Rezolvare:

Circuitul căutat este un sistem secvențial asincron, implementat cu porți, care trebuie să basculeze ca un bistabil D la aplicarea unui front crescător, sau descrescător, pe intrarea de ceas. Notăm cu C intrarea de ceas şi cu D intrarea de date, iar cu Q ieşirea. Ca orice bistabil, trebuie să aibă şi o ieşire $\overline{Q}$, dar deocamdată ea nu ne preocupă, pentru că o putem obține cu un simplu inversor de la ieşirea Q. Formele de undă care descriu funcționarea circuitului sunt date în figură.

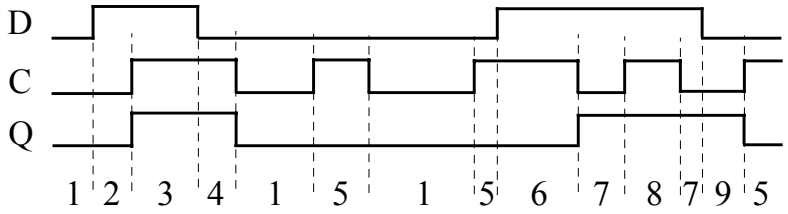


Fig. 4.30 *Formele de undă şi stările sistemului*

Q \ D C	00	01	11	10
1	1,0	5,0	-	2,0
2	-	-	3,1	2,0
3	-	4,1	3,1	-
4	1,0	4,1	-	-
5	1,0	5,0	6,0	-
6	-	-	6,0	7,1
7	-	-	8,1	7,1
8	-	-	8,1	7,1
9	9,1	5,0	-	-

Q^+, y

Fig. 4.31 *Tabelul tranzițiilor şi al ieşirii*

Tabelul tranzițiilor din figura 4.31 sugerează că sistemul are 4 stări distincte. Respectând condiția ca două stări între care au loc tranziții să aibă coduri adiacente, alegem următoarele coduri binare: stările compatibile 1 şi 2 primesc codul 00, stările compatibile 3 şi 4, codul 10, stările compatibile 5 şi 6, codul 01, iar stările compatibile 7, 8 şi 9 primesc codul rămas 11. Evident că sunt posibile şi alte codificări binare.

Cu aceste coduri, rezultă tabelul redus al tranzițiilor din figura 4.32. Se descompune în 3 diagrame Veitch-Karnaugh şi se deduc expresiile minime pentru funcțiile Q_1^+, Q_2^+ şi Q_3^+.

$Q_1 Q_2$ \ D C	00	01	11	10
00	00,0	01,0	10,1	00,0
01	00,0	01,0	01,0	11,1
11	11,1	01,0	11,1	11,1
10	00,0	10,1	10,1	-

$Q_1^+Q_2^+$, y

Fig. 4.32 *Tabelul redus al tranzițiilor şi codificarea stărilor*

Rezultă următoarele ecuații:

$$Q_1^+ = D \cdot Q_1 + C \cdot Q_1 \cdot \overline{Q}_2 + \overline{C} \cdot D \cdot Q_2 + \overline{C} \cdot Q_1 \cdot Q_2 + C \cdot D \cdot \overline{Q}_2$$
$$Q_2^+ = D \cdot Q_2 + Q_1 \cdot Q_2 + C \cdot \overline{D} \cdot \overline{Q}_1$$
$$y = Q_1^+$$

Dacă implementăm acum sistemul folosind numai două nivele de logică (porți ŞI şi porți SAU), obținem un circuit care conține 14 porți logice, iar analiza circuitului prin simulare PSpice generează formele de undă din figura 4.33. Verificați dacă circuitul funcționează corect şi comentați apariția "spike"-ului la momentul de 13μs pe traseul lui Q_2.

Ce se întâmplă dacă dorim să folosim un circuit cu număr mai mic de porți? Ecuațiile de mai sus se pot transforma aşa cum se vede mai jos şi rezultă un circuit care conține numai 10 porți logice. Verificați prin simulare PSPICE funcționarea circuitului şi comentați rezultatele obținute.

$$Q_1^+ = D \cdot Q_1 + (C \oplus Q_2) \cdot (D + Q_1)$$
$$Q_2^+ = (D + Q_1) \cdot Q_2 + C \cdot \overline{D} \cdot \overline{Q}_1$$
$$y = Q_1^+$$

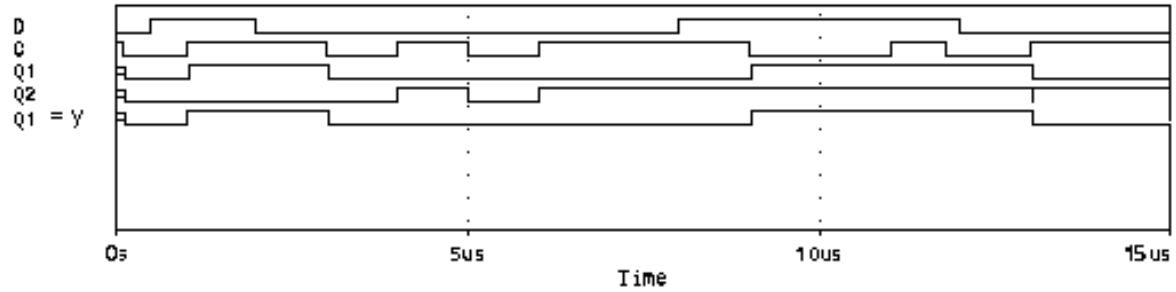


Fig. 4.33 *Formele de undă obținute prin simulare PSPICE* □

4.3.5 Să se proiecteze un sistem numeric care să asigure funcționarea automată a barierelor la o trecere la nivel peste calea ferată. Sistemul are 2 intrări, notate cu x_1 şi x_2, date de stările unor contacte amplasate de o parte şi de alta a şoselei. Ieşirea y comandă închiderea barierelor.

Rezolvare:

Avem de proiectat un sistem sincron cu comportament asincron, datorită intrărilor, care se pot modifica în orice moment de timp. Frecvența semnalului de ceas CLOCK, este mult mai mare decât rata de modificare a intrărilor. Presupunem că atunci când contactele sunt închise(trece trenul) avem 1 logic pe intrări, iar comanda de închidere a barierelor se dă pentru 1 logic la ieşire. Stările sistemului au fost marcate direct pe formele de undă rezultate prin simulare PSPICE din figura 4.34, unde s-au considerat trenuri scurte sau lungi, care vin dintr-o parte sau cealaltă a barierei.

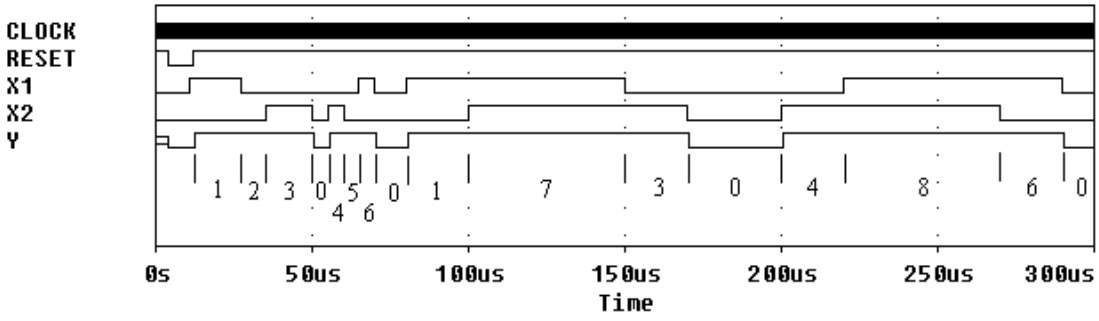


Fig. 4.34 *Sistemul are 9 stări distincte, numerotate de la 0 la 8*

Q^+, y

Q \ x_1x_2	00	01	11	10
0	0,0	4,1	-	1,1
1	2,1	-	7,1	1,1
2	2,1	3,1	-	-
3	0,0	3,1	-	-
4	5,1	4,1	8,1	-
5	5,1	-	-	6,1
6	0,0	-	-	6,1
7	-	3,1	7,1	-
8	-	-	8,1	6,1

Q^+, y

Q \ x_1x_2	00	01	11	10
0	0,0	4,1	-	1,1
1=2=7	2,1	3,1	7,1	1,1
3=6	0,0	3,1	-	6,1
4=5=8	5,1	4,1	8,1	6,1

Codificarea binară a stărilor: $0 \rightarrow 00$, $1 = 2 = 7 \rightarrow 01$, $3 = 6 \rightarrow 11$, $4 = 5 = 8 \rightarrow 10$

Fig. 4.35 *Tabelul tranzițiilor. Reducerea şi codificarea stărilor*

După reducerea stărilor rămân 4 stări distincte, care sunt codificate aşa cum se arată în figura 4.35. Se minimizează funcțiile de excitație şi ieşirea, rezultând ecuațiile care permit implementarea circuitului cu bistabile de tip D şi porți logice:

$$D_1 = \bar{x}_1 x_2 + x_1 Q_1 + Q_1 \overline{Q}_2,$$

$$D_2 = x_1 \bar{x}_2 + x_2 Q_2 + \overline{Q}_1 Q_2, \text{ şi}$$

$$y = Q_1 + Q_2.$$

Ar putea exista probleme în funcționare, dacă ne uităm la tabelul tranzițiilor şi la codurile alese? Observăm că există o situație în care se face o tranziție de la o stare la alta, deşi cele două stări nu au coduri adiacente. Este aceasta o problemă reală şi dacă da, atunci cum ar putea fi rezolvată? □

4.3.6 Un sistem secvențial are 2 intrări şi o ieşire care detectează orice secvență de 4 stări succesive pentru care cele două intrări sunt identice. La detectarea acestei secvențe ieşirea capătă valoarea logică 1 atât timp cât intrările sunt identice. Să se proiecteze sistemul.

Rezolvare:

Avem de proiectat un sistem sincron cu comportament sincron, deoarece modificarea intrărilor este dată de semnalul de ceas. Stările sistemului au fost marcate direct pe formele de undă rezultate prin simulare PSPICE din figura 4.36. Se observă din tabelul tranzițiilor dat în figura 4.37 că sistemul are 4 stări distincte. Codificarea propusă a stărilor generează următoarele expresii pentru funcțiile de excitație şi ieşire: $D_1 = \overline{x_1 \oplus x_2} \cdot (Q_1 + Q_2)$, $D_2 = \overline{x_1 \oplus x_2} \cdot \overline{Q}_1$ şi $y = \overline{x_1 \oplus x_2} \cdot Q_1 \cdot \overline{Q}_2$.

Problema se putea rezolva mai simplu, dacă observam de la început că $x_1 = x_2$ înseamnă $\overline{x_1 \oplus x_2} = 1$ şi obțineam un sistem cu o singură intrare $x = \overline{x_1 \oplus x_2}$.

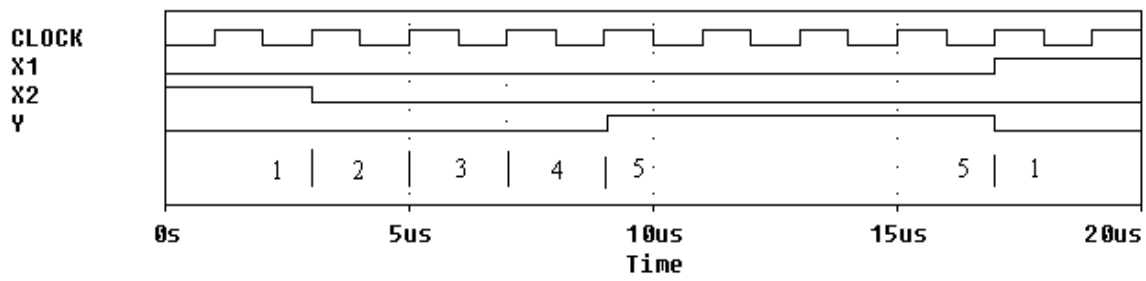


Fig. 4.36 *Sistemul are 5 stări distincte, numerotate de la 1 la 5*

Q \ x_1x_2	00	01	11	10
	Q^+, y			
1	2,0	1,0	2,0	1,0
2	3,0	1,0	3,0	1,0
3	4,0	1,0	4,0	1,0
4	5,1	1,0	5,1	1,0
5	5,1	1,0	5,1	1,0

Sistemul are 4 stări distincte.

Alegem următoarele coduri binare

$1 \longrightarrow 00$

$2 \longrightarrow 01$

$3 \longrightarrow 11$

$4 = 5 \longrightarrow 10$

Fig. 4.37 *Tabelul tranzițiilor. Reducerea şi codificarea stărilor* □

4.3.7 Se consideră automatul finit descris de organigrama din figură. Să se implementeze circuitul folosind bistabile de tip D şi având:

a) număr minim de porți

b) multiplexoare cu 4 căi de intrare

c) memorie ROM de 32 cuvinte a câte 4 biți

d) un circuit combinațional format din memorie ROM de 4 cuvinte a câte 4 biți şi multiplexoare. Reprezentați harta memoriei.

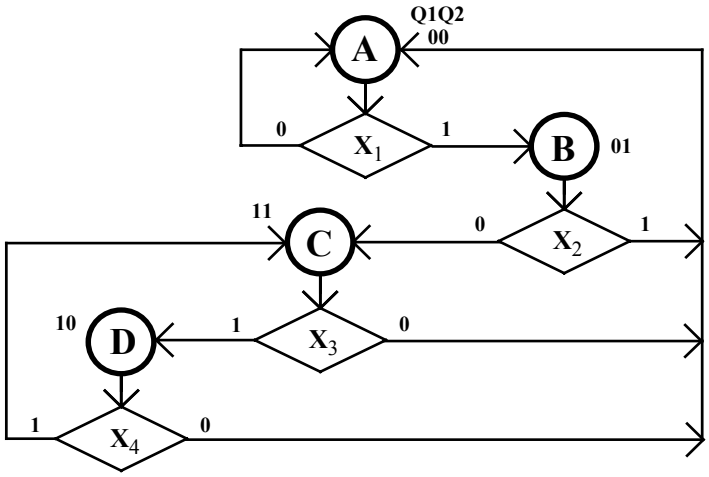


Fig. 4.38 *Organigrama care descrie funcționarea automatului finit*

Rezolvare:

Construim tabelul tranzițiilor urmărind organigrama. Sistemul are 2 bistabile şi 4 intrări, al căror sincronism cu ceasul nu ne preocupă, codurile stărilor fiind deja date. Oricum, dacă variabilele X_1 şi X_3 pot fi şi asincrone, variabilele X_2 şi X_4 sunt obligatoriu sincrone.

X_1	X_2	X_3	X_4	Q_1	Q_2	Q_1^+	Q_2^+
0	x	x	x	0	0	0	0
1	x	x	x	0	0	0	1
x	0	x	x	0	1	1	1
x	1	x	x	0	1	0	0
x	x	0	x	1	1	0	0
x	x	1	x	1	1	1	0
x	x	x	0	1	0	0	0
x	x	x	1	1	0	1	1

Fig. 4.39 *Tabelul tranzițiilor pentru organigrama din figura 4.38*

a) Pentru implementarea cu porți este necesară minimizarea funcțiilor binare $Q_1^+ = D_1$ şi $Q_2^+ = D_2$. Folosim diagramele Veitch-Karnaugh condensate.

b) Urmărind tabelul tranzițiilor se poate face uşor implementarea funcțiilor cu multiplexoare, la fel ca în exemplul prezentat la punctul 4.2.3.

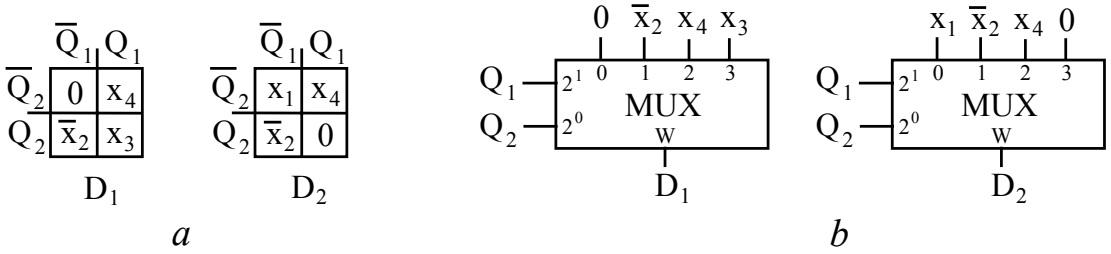


Fig. 4.40 *Soluțiile problemei pentru punctele* a *şi* b

c) Cele 6 variabile necesită folosirea unui număr de 2 cipuri de memorie ROM de 32 cuvinte (2^5), având ieşiri cu 3 stări (figura 4.41.a). Variabila suplimentară, aici x_1, selectează citirea datelor numai de la circuitul selectat. Se poate utiliza şi un singur circuit de memorie de 32 cuvinte a câte 4 biți şi două multiplexoare cu câte 2 intrări. Desenați schema logică şi comparați cele două implementări propuse.

d) O altă soluție de implementare cu memorie este dată în figura 4.41.*b*. Orice soluție care foloseşte un cip de memorie ROM trebuie să aibă şi harta memoriei, care indică conținutul memoriei ROM. Deci soluția din figura 4.41.*b* este completă. Este evident că toate aceste soluții implementează numai logica combinațională a automatului finit. Se vede că cele două bistabile de tip D nu sunt reprezentate în scheme.

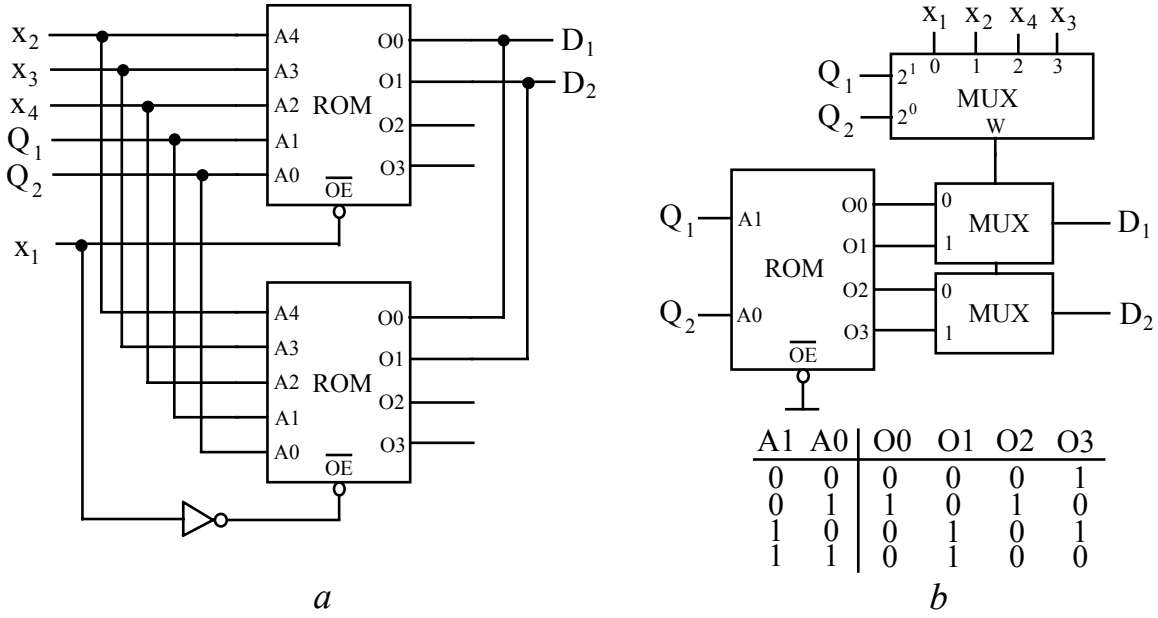


Fig. 4.41 *Soluțiile problemei pentru punctele* c *şi* d □

4.3.8 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. Arătați modul de conectare a circuitului la fiecare dintre cele trei microprocesoare pe 8 biți.

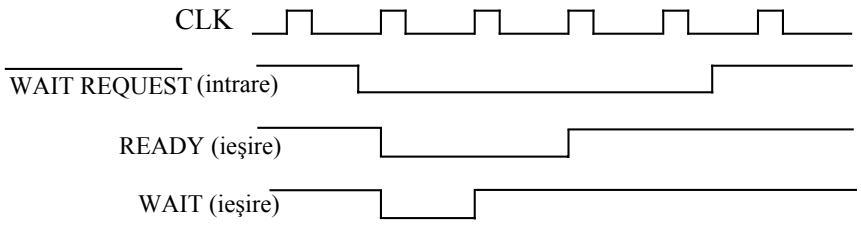
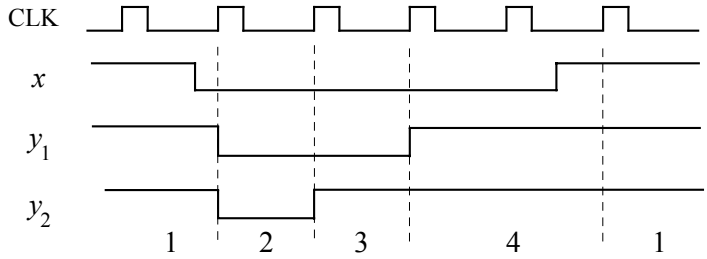


Fig. 4.42 *Formele de undă pentru problema 4.3.8*

Rezolvare:

Semnalul de intrare $\overline{\text{WAIT REQUEST}}$ este considerat asincron, deşi ar putea fi chiar sincron sau sincronizabil cu semnalul de ceas. Pentru simplitate vom nota acest semnal cu *x*, iar ieşirile READY şi WAIT cu y_1 şi respectiv y_2.



Q \ x	0	1
1	2,11	1,11
2	3,00	-
3	4,01	-
4	4,11	1,11

Fig. 4.43 *Stările circuitului şi tabelul tranzițiilor*

Dacă alocăm celor 4 stări codurile Q_1Q_2: 1 = 00, 2 = 01, 3 = 11 şi 4 = 10, atunci obținem următoarele ecuații pentru logica combinațională: $D_1 = Q_2 + Q_1 \cdot \overline{x}$, $D_2 = \overline{Q_1} \cdot \overline{x}$, $y_1 = \overline{Q_2}$ şi $y_2 = Q_1 + \overline{Q_2}$. Desenați schema logică a circuitului.

O soluție alternativă, agreată de multe ori în practică, este implementarea cu registru. Chiar dacă ea introduce un bistabil în plus, se reduce numărul de porți folosite. Funcțiile de ieşire READY şi WAIT sunt generate pentru anumite secvențe ale intrării $\overline{\text{WAIT REQUEST}}$. Atât timp cât intrarea este în 1 logic, READY = WAIT = 1, condiție ce trebuie îndeplinită pentru oricare dintre stările interne $Q_2Q_1Q_0$ = 000, 100, 110 sau 111 ale sistemului. La trecerea intrării în 0 logic, ieşirile $Q_2Q_1Q_0$ ale bistabilelor trec succesiv prin stările 011, 001 şi una dintre stările deja menționate anterior. În starea $Q_2Q_1Q_0 = 011$ ieşirile circuitului sunt $y_1y_2 = 00$, iar în starea $Q_2Q_1Q_0 = 001$ ieşirile sunt $y_1y_2 = 01$. Se construieşte tabelul de adevăr pentru funcțiile de ieşire, având ca variabile ieşirile bistabilelor, se minimizează folosind diagramele Veitch-Karnaugh şi se obține schema logică din figura 4.44. Pentru conectarea acestui circuit la o schemă cu microprocesor trebuie să avem informații de catalog despre microprocesorul utilizat.

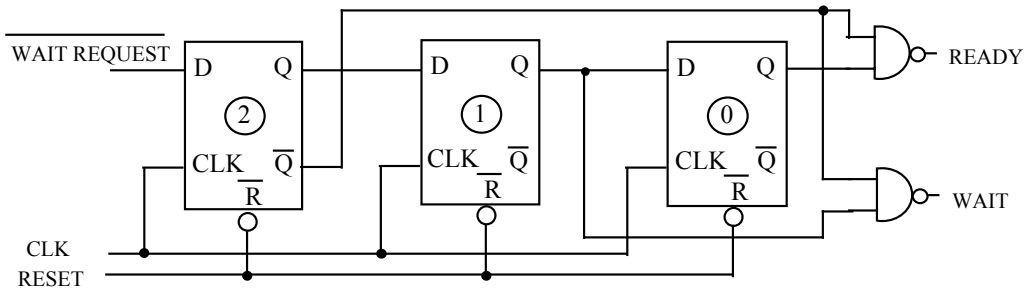


Fig. 4.44 *Schema logică a circuitului implementat cu registru* □