5 STRUCTURI PROGRAMABILE

Aplicațiile din acest capitol îşi propun să prezinte funcționarea circuitelor de memorie ROM(Read Only Memory) şi RAM(Random Access Memory), a structurilor programabile PLD(Programmable Logic Devices), structuri care conțin rețele programabile de porți logice şi bistabile, precum şi a structurilor FPGA(Field Programmable Gate Arrays), rețele complexe de blocuri logice programabile şi resurse de interconectare a lor, care se configurează prin programare pentru o anumită aplicație.

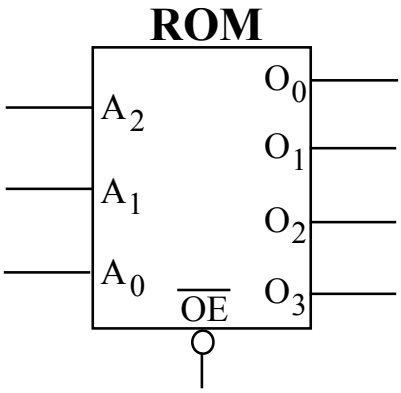
5.1 Considerații teoretice

5.1.1 Memoria ROM

Memoria ROM este un circuit combinațional care stochează permanent date binare, iar această informație poate fi numai citită. Această structură este de obicei definită ca un convertor de cod compus dintr-un decodificator şi un codificator. Vectorul de intrare în decodificator este interpretat ca o adresă, iar datele obținute la ieşirea codificatorului reprezintă informația memorată la adresa respectivă.

În figura 5.1 s-a luat un exemplu de memorie ROM care conține 8 cuvinte de câte 4 biți. O combinație binară care se aplică pe cele 3 intrări de adresă, A_2, A_1 şi A_0, selectează unul dintre cele 8 cuvinte, iar cei 4 biți de date ai cuvântului selectat sunt disponibili la ieşirile O_0, O_1, O_2 şi O_3, cu condiția ca intrarea $\overline{OE}$(Output Enable) să fie activată (în exemplul nostru activarea se face pe 0 logic). Dacă $\overline{OE}=1$ ieşirile memoriei sunt în starea de înaltă impedanță (high Z). Tabelul de adevăr din figură este numai un exemplu care arată o posibilitate de implementare a 4 funcții binare de câte 3 variabile.

Există mai multe tipuri constructive de memorie ROM. Memoriile ROM sunt de obicei considerate cele care sunt încărcate cu date în procesul de fabricație al circuitului integrat, deci care nu sunt programabile de către utilizator. Utilizatorul poate introduce datele lui o singură dată într-o memorie PROM(Programmable ROM), sau de mai multe



A_2	A_1	A_0	$\overline{OE}$	O_0	O_1	O_2	O_3
0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1
0	1	0	0	1	0	1	1
0	1	1	0	0	1	1	1
1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	1	1	0	1	0	0	0
x	x	x	1	high Z	high Z	high Z	high Z

Fig. 5.1 *Memorie ROM de 8 cuvinte de 4 biţi şi harta memoriei*

ori, în memoriile EPROM(Erasable PROM) sau EEPROM (Electrically EPROM), diferenţa fiind dată de modalitatea de ştergere a datelor din memorie. Memoriile EPROM conţin tranzistoare MOS care conţin o poartă suplimentară, izolată de restul circuitului. Această poartă permite stocarea pe termen lung a sarcinii electrice necesare pentru memorarea bitului respectiv de informaţie. Ştergerea se face prin expunere la radiaţii ultraviolete. La memoriile EEPROM izolaţia porţii este mult mai subţire şi sarcina electrică în exces poate fi eliminată prin aplicarea unei tensiuni de polaritate inversă pe poarta tranzistorului care nu este flotantă, deci ştergerea se face pe cale electrică.

Circuitul integrat 82S147, care este o memorie PROM în tehnologie Schottky TTL, a fost utilizat deja în aplicaţiile prezentate în capitolul anterior. Rolul lui era de a implementa logica combinaţională a unui automat cu stări finite (vezi punctul 4.2.4 şi problema 4.3.7).

5.1.2 Memoria RAM

Memoria RAM este un circuit care stochează biţi de informaţie într-o matrice de memorie, la fel ca memoria ROM. Diferenţa constă în faptul că informaţia utilă memorată în RAM trebuie mai întâi să fie "scrisă" acolo, înainte de a fi citită.

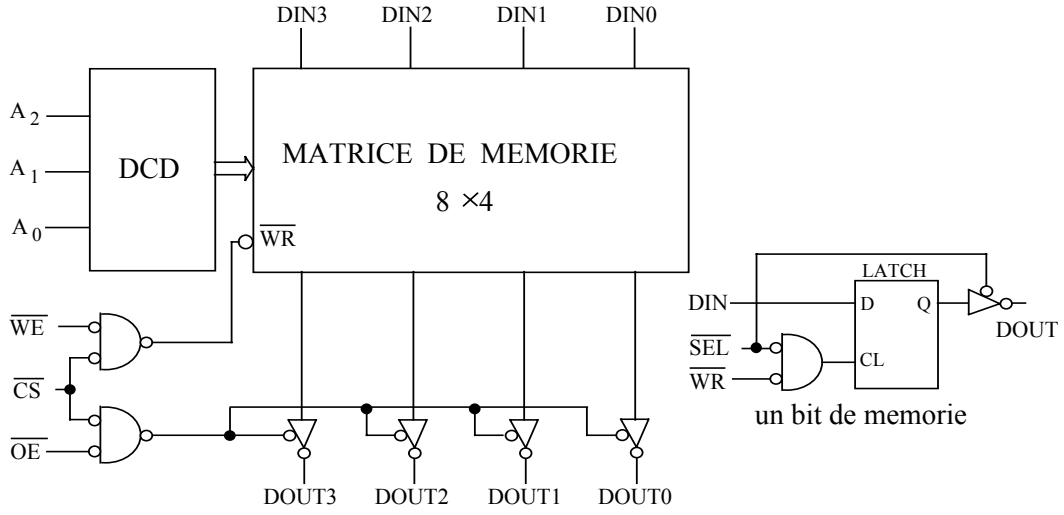


Fig. 5.2 *Structura unei memorii SRAM de 8 cuvinte de 4 biţi*

Există două tipuri constructive de memorie RAM: RAM static sau SRAM, în care biții de date, odată ce au fost înscrişi, sunt memorați atât timp cât circuitul integrat este alimentat cu tensiune, şi RAM dinamic sau DRAM, în care datele memorate trebuie să fie mereu reîmprospătate prin citirea şi apoi rescrierea lor periodică în locațiile respective de memorie, în caz contrar ele pierzându-se definitiv.

Structura unei memorii SRAM este asemănătoare cu cea a unei memorii ROM. Apare în plus semnalul $\overline{WE}$ (Write Enable) care, odată ce este activat pe 0 logic, memorează datele de pe intrările de date la adresa indicată de intrările de adresă. Se poate vedea în figura 5.2 că celula de memorie de un bit conține un latch de tip D, iar memorarea datelor se face pe palierul de 1 logic al ceasului, adică atunci când sunt activate semnalele $\overline{WR}$ şi $\overline{SEL}$, acesta din urmă fiind generat de una din ieşirile decodificatorului liniilor de adresă. Activarea semnalului $\overline{WR}$ este o consecință a activării semnalelor de intrare $\overline{WE}$ şi $\overline{CS}$.

Circuitul integrat MMN 2114, care este o memorie SRAM în tehnologie NMOS, a fost utilizat deja în aplicațiile prezentate în capitolul anterior. Rolul lui era de a emula o memorie ROM care implementa logica combinațională a unui automat cu stări finite (vezi punctul 4.2.4).

5.1.3 Structuri PLD

Structurile PLD conțin porți logice şi, în unele cazuri, circuite bistabile, aranjate în aşa fel încât interconectările dintre componente să poată fi modificate pentru a implementa diverse funcții binare.

Structurile PLD combinaționale conțin numai porți logice cu conexiuni programabile, care permit implementarea comodă a funcțiilor binare reprezentate în formă disjunctivă. Circuitele reprezentative din această categorie sunt **structurile PLA** (**P**rogrammable **L**ogic **A**rrays) şi **structurile PAL** (**P**rogrammable **A**rray **L**ogic). Acestea din urmă sunt marcă înregistrată a firmei AMD.

Un exemplu de circuit PLD secvențial de tip registru este circuitul integrat **GAL16V8**, marcă înregistrată a firmei Lattice Semiconductor, circuit care conține 8 intrări, 8 intrări/ieşiri cu 3 stări şi 8 macrocelule programabile, numite aici **OLMC** (**O**utput **L**ogic **M**acro **C**ell). Schema completă a circuitului este dată în figura 5.3. Terminația QS din codul circuitului integrat provine de la "Quiet Series" şi are în vedere modificarea traseului intern de masă în scopul reducerii zgomotelor. Matricea de porți ŞI conține 2048 de conexiuni programabile, iar porțile SAU au conexiuni fixe (vezi figura 5.4). Matricea de conexiuni programabile permite conectarea oricărei intrări numerice, în formă directă sau negată, la orice termen produs. Fiecare macrocelulă programabilă mai conține câte 10 conexiuni programabile, care stabilesc modul de lucru. În sfârşit, un număr de 64 de conexiuni programabile, grupate în 8 octeți, stabilesc o semnătură digitală a utilizatorului, adică cel care programează circuitul integrat, pentru secretizarea hărții de conexiuni şi evitarea multiplicării neautorizate a unui produs care conține astfel de circuite.

Schema internă a unei macrocelule programabile este dată în figura 5.4. Ieşirea porții SAU poate fi complementată sau nu cu poarta SAU-EXCLUSIV şi aplicată direct la intrarea porții cu 3 stări de la ieşire printr-un canal de multiplexor, sau poate fi aplicată la intrarea unui bistabil D, a cărui ieşire se poate trimite la exterior, sau returna spre matricea de

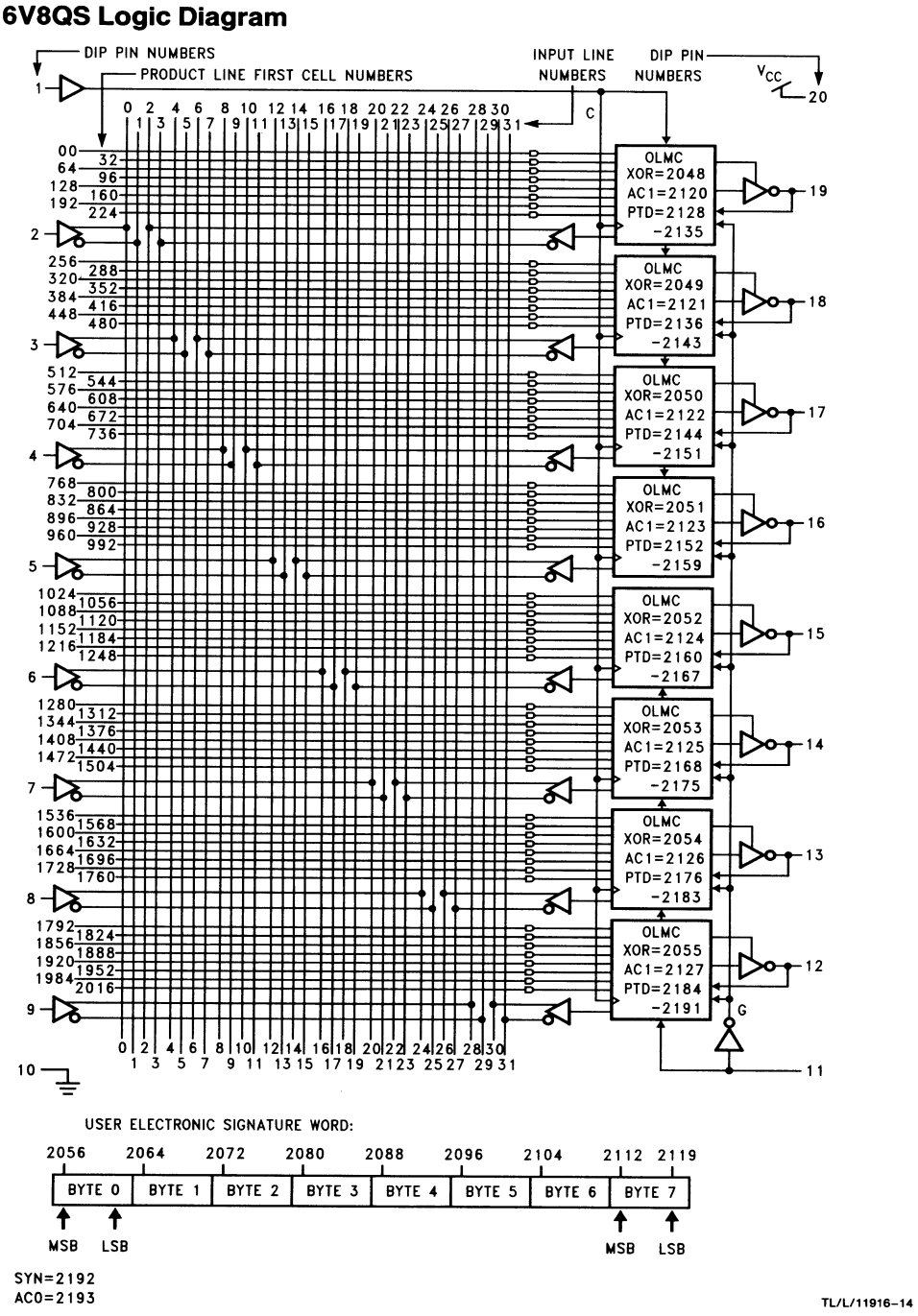


Fig. 5.3 *Structura circuitului GAL 16V8*

comutatoare programabile de la intrare. Fluxul datelor prin circuit este stabilit cu ajutorul unor multiplexoare cu intrări de selecție programabile. Există legături între macrocelule, pe de o parte, pentru transfer de date, iar pe de altă parte, comutatorul AC0 se conectează fie la 0 logic, fie la 1 logic pentru toate macrocelulele din structură. Comutatoarele AC1, XOR şi PTD se programează independent, pentru fiecare celulă în parte, după necesități. Proiectantul nu este obligat să cunoască structura internă a circuitului integrat, decât dacă stabileşte manual harta de conexiuni, lucru foarte plictisitor şi cu mare şansă de eroare. Există programe care realizează automat harta de conexiuni, pornind de la ecuațiile furnizate de proiectant şi folosind un model software al circuitului PLD folosit.

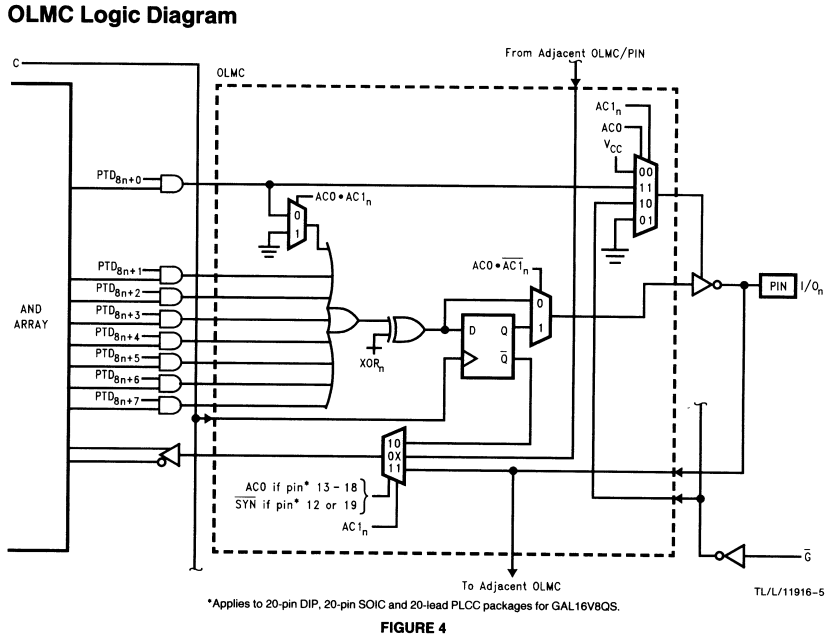


Fig. 5.4 *Structura unei macrocelule programabile din circuitul GAL 16V8*

5.1.4 Structuri FPGA

Arhitectura unui circuit FPGA este prezentată în figura 5.5. Există trei elemente constructive de bază care se repetă ori de câte ori este necesar în structură: blocul logic, blocul de intrare-ieşire, şi resursele de interconectare ale blocurilor, de fapt matrici de comutatoare programabile, numite şi switchbox-uri. Blocul logic poate conține sute sau mii de porți logice şi poate fi configurat diferit în funcție de aplicație. Realizarea interconexiunilor permite o utilizare superioară a resurselor logice față de structurile PLD.

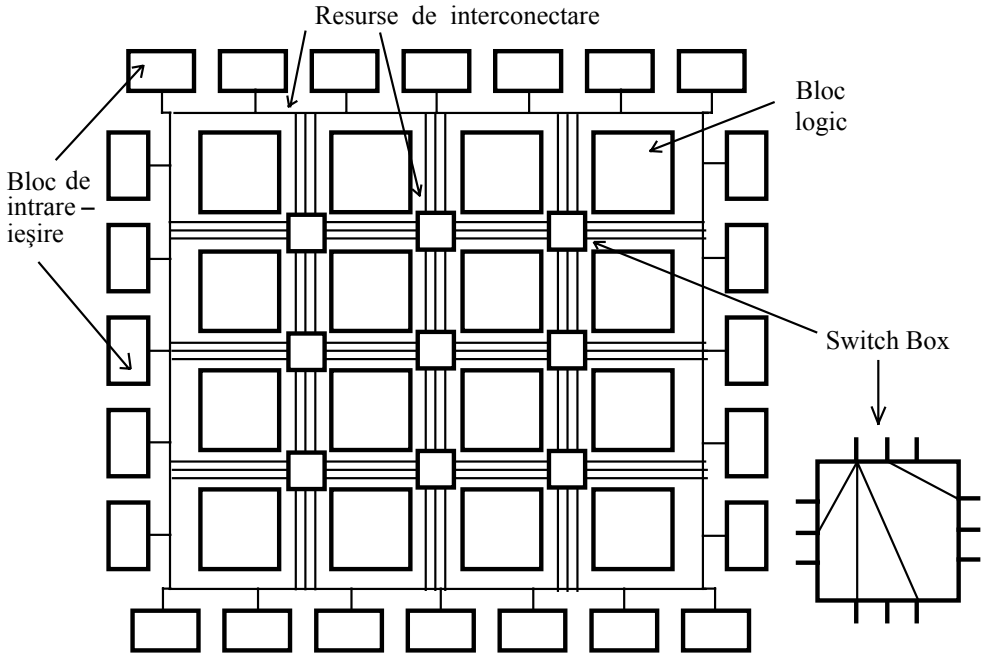


Fig. 5.5 *Arhitectura FPGA*

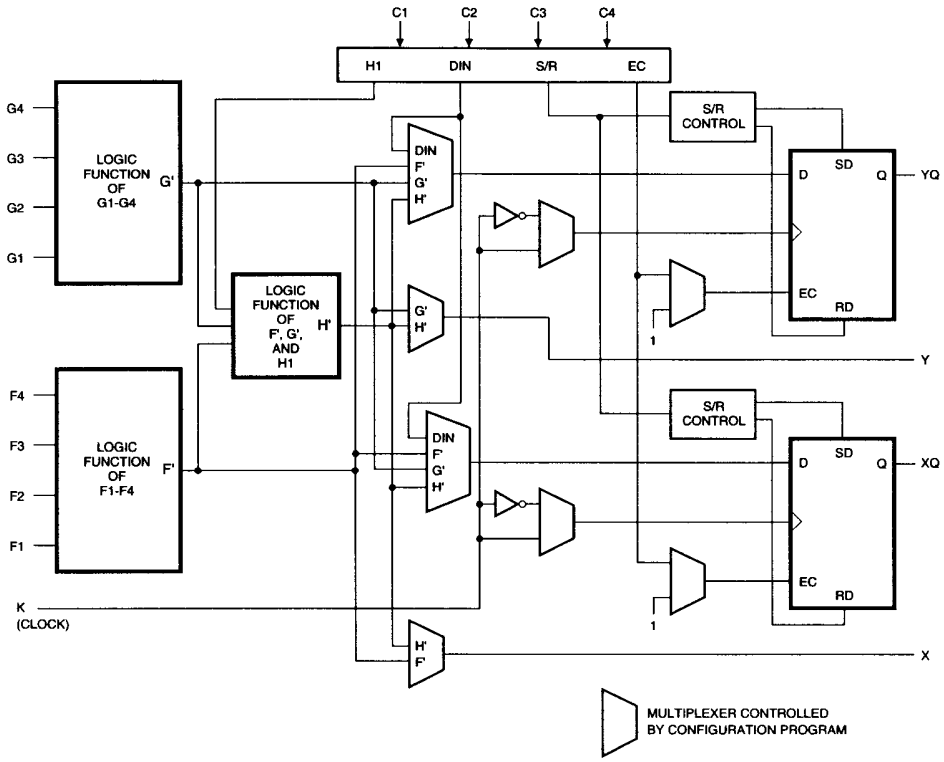


Fig. 5.6 *Schema simplificată a blocului logic la circuitul XC4000*

Structura blocului logic al circuitului XC4000 produs de firma Xilinx este dată în figura 5.6. Modulele F şi G sunt generatoare de funcţii binare programabile cu câte 4 intrări, iar împreună cu modulul H, care este tot un generator de funcţii binare, permit implementarea unor funcţii cu 9 variabile independente. Blocul logic mai conţine o logică combinaţională de selecţie şi 2 bistabile de tip D pentru stocarea rezultatelor date de generatoarele de funcţii. Ieşirile generatoarelor de funcţii se pot utiliza independent de ieşirile elementelor de stocare de tip registru.

Fiecare bloc de intrare/ieşire controlează un singur pin al circuitului integrat şi se poate configura ca port de intrare, port de ieşire sau port bidirecţional. Semnalele de intrare se pot aplica direct sau prin bistabile de intrare. Semnalele de ieşire, care se pot inversa în interiorul blocului, se pot conecta direct la ieşirea pinului sau la bistabilul de ieşire. Matricea de cuplare programabilă, sau switchbox-ul, este alcătuită din conexiuni programabile care permit realizarea oricărei configuraţii posibile de conexiuni.

5.2 Demonstraţii practice

Consideraţiile asupra alimentării panoului logic, formulate în primul capitol, rămân valabile şi aici. Panourile logice folosite în aplicaţii, fie au surse proprii de alimentare (programatorul de memorii EPROM sau sistemul de dezvoltare cu FPGA), fie se alimentează cu o tensiune nominală de 5Vcc (panoul logic cu memorie RAM sau panoul cu GAL), în cazul în care acestea din urmă nu sunt şi ele prevăzute cu surse proprii de alimentare.

5.2.1 Se conectează programatorul de memorii EPROM la portul paralel al unui calculator. Se porneşte calculatorul şi apoi se alimentează şi programatorul de la sursa proprie de alimentare, care furnizează toate tensiunile necesare pentru citirea şi programarea unei memorii EPROM de tipul 2716 sau 2732. Memoria 2716 are 2Kcuvinte de câte 8 biți, iar memoria 2732 are 4Kcuvinte de câte 8 biți.

Se lansează aplicația "Programator EPROM", care deschide fereastra reprezentată în partea stângă a figurii 5.7. Din lista de opțiuni se selectează tipul memoriei EPROM, din butoanele radio operația dorită, iar offset-ul reprezintă adresa de la care se începe citirea sau programarea memoriei. Operația de elaborare a fişierului sursă deschide o a doua fereastră, dată în partea dreaptă a figurii 5.7, prin care se introduce într-un fişier valoarea conținută de fiecare adresă, de la offset +1 şi până la offset + număr de locații. În exemplul dat, prin apăsarea butonului "Save", se introduce numărul binar 01101001 la adresa 5, offset-ul fiind 0. Fişierul construit în acest mod va fi folosit la operația de programare a memoriei.

Introduceți în soclu o memorie EPROM de tipul 2716, după ce ați oprit mai întâi tensiunea de alimentare a programatorului, şi apoi citiți conținutul ei. Toate locațiile sunt programate? Dacă nu, atunci introduceți în locațiile imediat următoare offset-ului 10 cuvinte identice alese de dumneavoastră. Citiți din nou memoria pentru a verifica că datele au fost înscrise corect. Opriți tensiunea de alimentare a programatorului şi apoi alimentați din nou programatorul. Verificați dacă datele înscrise s-au păstrat. Consultați foaia de catalog şi observați cronogramele ciclurilor de citire şi programare.

Repetați operațiile de mai sus pentru o memorie EPROM de tipul 2732. Nu am precizat exact codurile memoriilor EPROM, pentru că ele depind de producător: I 2716, dacă circuitul este fabricat de INTEL, MMN 2716 dacă este fabricat de "Microelectronica" Bucureşti, sau K573РФ2 (echivalent cu 2716) dacă este fabricat în Rusia. Dacă aveți la dispoziție o lampă cu ultraviolete, încercați să ştergeți memoriile EPROM şi verificați dacă acest lucru s-a realizat (toți biții sunt poziționați pe 1 logic).

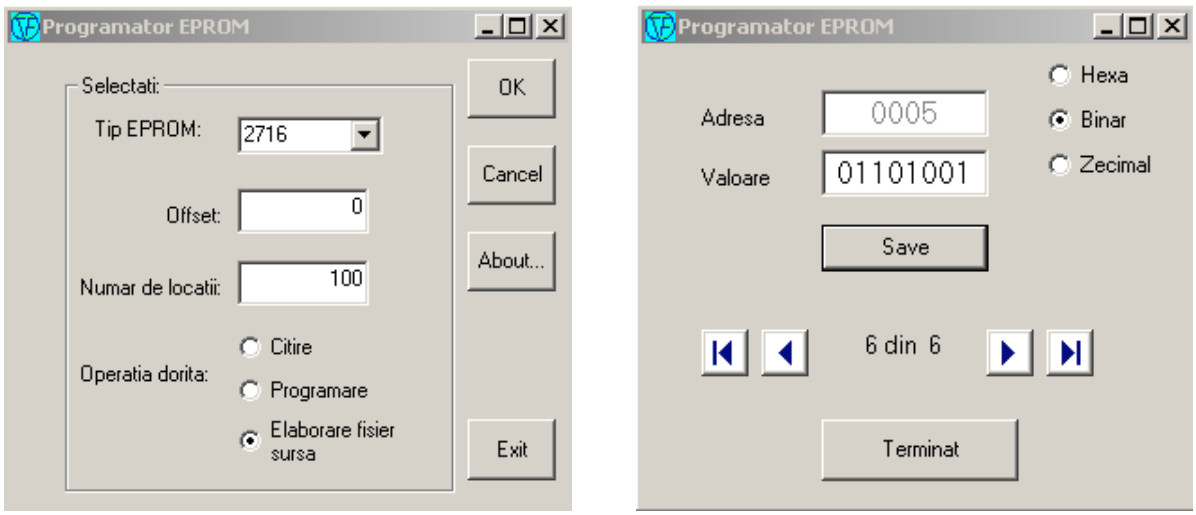


Fig. 5.7 *Două ferestre ale aplicației "Programator EPROM"* □

5.2.2 Se studiază memoria RAM static de tipul 2114, care are o capacitate de 1K de câte 4 biți. Panoul logic folosit deja la automatele cu memorie RAM are un circuit integrat MMN 2114. Se reia scrierea datelor în memorie, aşa cum se arată la pagina 68,

şi se justifică operaţiile care se fac. Se citeşte memoria, măsurând de această dată ieşirile memoriei şi se verifică dacă datele introduse au fost memorate corect. Se întrerupe tensiunea de alimentare câteva secunde şi, după revenirea ei, se citesc din nou datele de la adresele respective. Ce constataţi? Justificaţi răspunsul şi indicaţi o soluţie pentru păstrarea datelor chiar şi după decuplarea temporară a tensiunii de alimentare.

Se repetă operaţiile de mai sus pentru panoul logic care conţine o memorie RAM static de tipul **2102**, cu o capacitate de 1K de câte 1 bit şi linii separate pentru intrarea şi ieşirea bitului de date. O schemă simplificată a panoului logic este dată în figura 5.8. De această dată adresarea locaţiilor de memorie se face cu ajutorul a două numărătoare de câte 5 biţi (**MMC 4024**), care pot fi uşor poziţionate în orice stare dorită cu ajutorul unui ceas manual, realizat cu comutator mecanic şi latch. Astfel se poate fixa orice adresă din spaţiul de 1024 adrese disponibile. Datele se introduc pe linia de intrare printr-un comutator cu două poziţii, iar linia de ieşire arată conţinutul memoriei prin starea unui LED. Semnalul $R/\overline{W}$ este generat tot cu un latch.

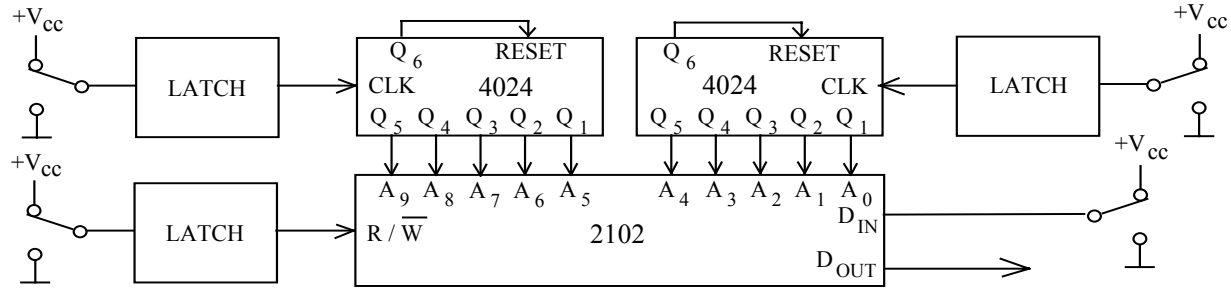


Fig. 5.8 *Schema simplificată a panoului logic cu memorie RAM 2102* □

5.2.3 Se introduce în soclul programatorului de circuite GAL un circuit GAL16V8 şi se conectează programatorul la portul paralel al unui calculator. Se porneşte calculatorul şi apoi se alimentează şi programatorul de la sursa proprie de alimentare. Se lansează programul freeware "galprog2", care permite prin intermediul unei interfeţe grafice selecţia portului paralel, a tipului de GAL, încărcarea datelor din fişier şi salvarea lor în fişierul JEDEC, care conţine harta conexiunilor programabile,

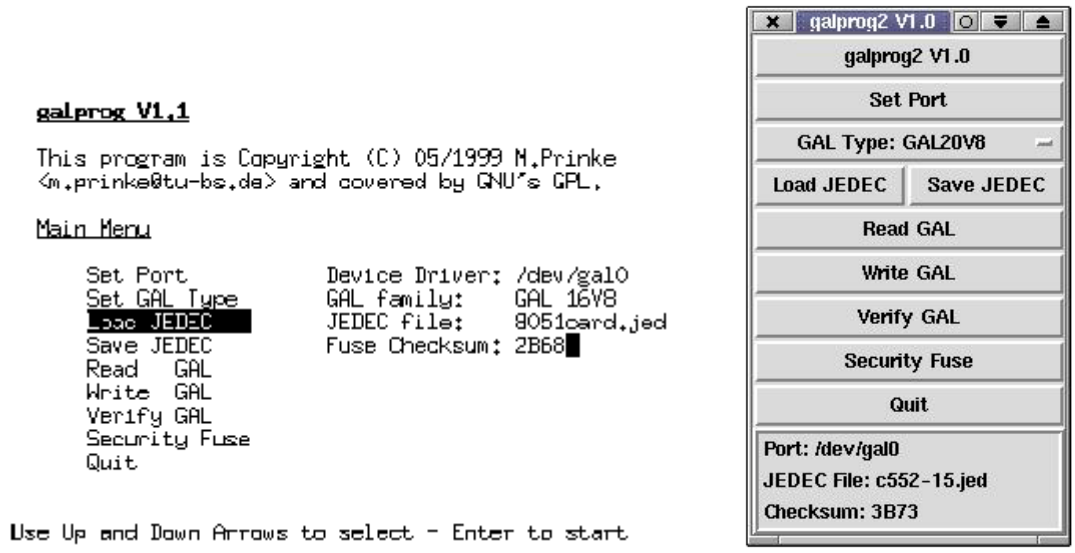


Fig. 5.9 *Două interfeţe pentru programarea circuitelor GAL 16V8 şi GAL20V8*

precum şi citirea stării conexiunilor unui GAL, dacă acest lucru este permis de fuzibilul destinat secretizării datelor, programarea conexiunilor sau verificarea stării lor după ce circuitul integrat a fost programat.

Fişierul JEDEC diferă de la un circuit la altul. Este un fişier text care indică harta conexiunilor şi care se obține de obicei automat, folosind un compilator care transformă ecuațiile boolene ale circuitului într-o matrice de 0 şi 1. Sigur că el poate fi generat şi manual, având structura internă a circuitului folosit, dar această operație chinuitoare va fi făcută pentru circuitul PAL16R4 în capitolul următor.

Fişierul GAL.JED conține harta conexiunilor pentru automatul finit descris în figura 4.9 din capitolul anterior. Intrarea X este pinul 2 al circuitului integrat, stările Q_1 şi Q_2 se obțin la ieşirile primelor două macrocelule, iar ieşirile Y_1 şi Y_2 se obțin la pinii 17 şi respectiv 16 ai circuitului integrat. Se citeşte harta conexiunilor din GAL şi se verifică corespondența cu conținutul fişierului GAL.JED. Se verifică practic funcționarea circuitului, verificând toate tranzițiile din organigramă. □

5.2.4 Placa XS40-V1.2 conține un circuit FPGA de tipul **XC4010XL**, alimentat la 3,3V, care conține 20000 porți logice şi 1120 de bistabile. Se alimentează de la o sursă externă de 9V şi se conectează la calculator prin portul paralel. Proiectarea circuitului se poate face fie cu un editor grafic, fie în limbaj VHDL, folosind un software specializat, de exemplu versiunea de evaluare de la "Xilinx Foundation Package". După analiza circuitului prin simulare, configurația circuitului este stocată în memoria RAM internă, prin portul paralel. Figura 5.10 prezintă o vedere de ansamblu a plăcii şi un exemplu de conexiuni. Ieşirea VGA permite cuplarea directă la monitor.

Se implementează cu ajutorul editorului grafic un numărător cu 16 stări, ale cărui ieşiri se pot vizualiza pe LED-urile afişajului, se obține NETLIST-ul şi se programează placa XS40-V1.2. Se verifică funcționarea circuitului astfel obținut.

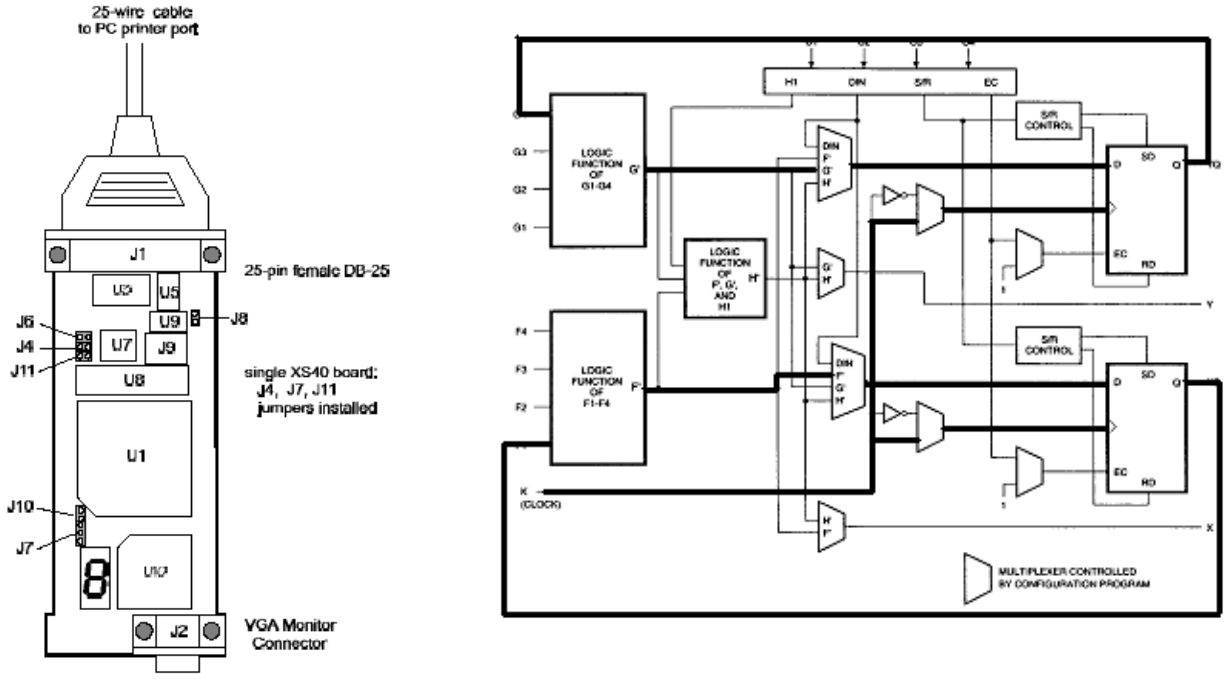


Fig. 5.10 *Vedere de sus a plăcii XS40 V1.2 şi conexiuni realizate într-un bloc logic din circuitul FPGA XC 4010XL, care implementează un numărător cu 16 stări* □

5.1 Probleme rezolvate

5.3.1 Să se implementeze cu memorie ROM un generator al funcției *sinus*, ştiind că argumentul funcției variază între 0 şi π/2 cu paşi de π/512. Se cere rezultatul cu 4 zecimale exacte în cod BCD.

Rezolvare:

Patru cifre zecimale în cod BCD se reprezintă pe 16 biți, deci cuvintele memoriei sunt structurate pe 16 biți. Numărul de cuvinte de memorie N este dat de raportul dintre interval şi numărul de paşi, deci: $N = \frac{\pi}{2} \cdot \frac{512}{\pi} = 256$. Deci memoria necesară are 256 cuvinte a câte 16 biți.

Dacă folosim, de exemplu, cipuri de 256 cuvinte a câte 8 biți, atunci schema logică ar fi cea din figura 5.11.

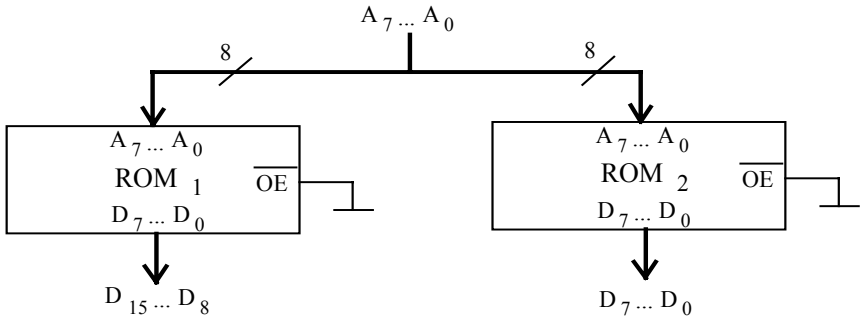


Fig. 5.11 *Extinderea numărului de biți pe cuvânt*

Un fragment din harta memoriei pentru întregul sistem (cele două cipuri împreună) este prezentat în figura 5.12.

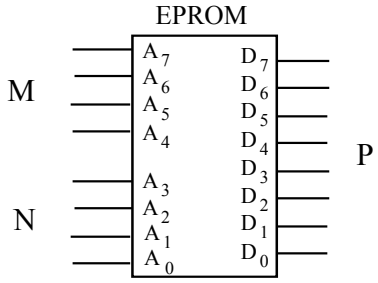
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			.....													.....							
1	1	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

Fig. 5.12 *Harta memoriei pentru generatorul funcției* sinus □

5.3.2 Folosind o memorie EPROM să se implementeze un circuit care realizează înmulțirea $M \times N = P$, unde M, N şi P sunt exprimate în cod BCD, iar M şi N sunt cuprinse între 0 şi 9.

Rezolvare:

Operanzii M şi N se reprezintă pe câte 4 biți, deci memoria trebuie să aibă 8 biți de adresă. Valoarea maximă a produsului P este $9 \times 9 = 81$, număr care se reprezintă în binar pe 7 biți, deoarece $2^6 < 81 < 2^7$. În cod BCD numărul P se reprezintă însă pe 8 biți, fiind format din două cifre BCD. El se obține la ieşirile de date ale memoriei. Deci, memoria are 256 cuvinte de câte 8 biți fiecare. Figura 5.13 reprezintă schema logică şi un fragment din harta memoriei.



A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				.....							.....				
0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0
0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0
				.....							.....				
0	1	1	1	0	1	0	1	0	0	1	1	0	1	0	1
				.....							.....				
1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1

Fig. 5.13 *Circuit de înmulțire cu memorie EPROM şi un fragment din harta memoriei* □

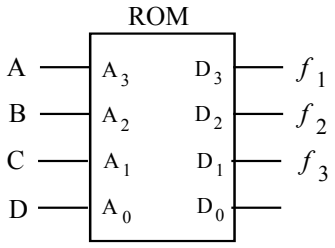
5.3.3 Să se implementeze cu memorie ROM şi apoi cu o structură PLA următorul set de funcții binare:

$$f_1 = P_1 + P_3 + P_4 + P_8 + P_{12} + P_{14} + P_{15}$$
$$f_2 = P_0 + P_1 + P_5 + P_8 + P_{12}$$
$$f_3 = P_2 + P_4 + P_5 + P_8 + P_{13} + P_{15}$$

Rezolvare:

Implementarea funcțiilor binare cu memorii ROM presupune stabilirea capacității memoriei necesare, alegerea tipului potrivit de memorie din oferta disponibilă în catalog şi stabilirea hărții memoriei. De obicei se urmăreşte utilizarea unui număr minim de circuite integrate, chiar dacă memoria ROM rămâne parțial nefolosită.

În exemplul considerat, termenul canonic de rang maxim este P_{15}, deci sunt necesare 4 intrări de adrese. Pentru cele 3 funcții binare sunt necesare 3 ieşiri de date. Folosim o memorie de 16 cuvinte de câte 4 biți. Soluția cu memorie ROM este prezentată în figura 5.14. Implementarea cu o structură PLA este prezentată în figura 5.15.



A_3	A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1	0	X
0	0	0	1	1	1	0	X
0	0	1	0	0	0	1	X
0	0	1	1	1	0	0	X
0	1	0	0	1	0	1	X
0	1	0	1	0	1	1	X
0	1	1	0	0	0	0	X
0	1	1	1	0	0	0	X

A_3	A_2	A_1	A_0	D_3	D_2	D_1	D_0
1	0	0	0	1	1	1	X
1	0	0	1	0	0	0	X
1	0	1	0	0	0	0	X
1	0	1	1	0	0	0	X
1	1	0	0	1	1	0	X
1	1	0	1	0	0	1	X
1	1	1	0	1	0	0	X
1	1	1	1	1	0	1	X

Fig. 5.14 *Implementarea sistemului de 3 funcții binare cu memorie ROM*

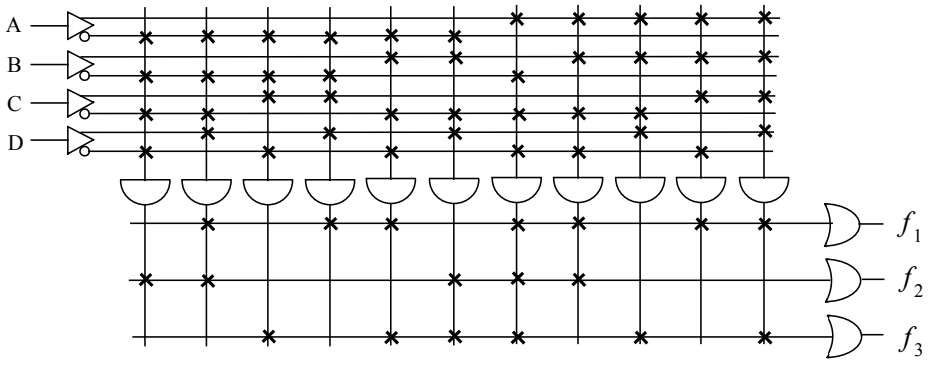


Fig. 5.15 *Implementarea sistemului de 3 funcții binare cu PLA* □

5.3.4 Să se implementeze o memorie EPROM de $2K \times 32$, folosind o memorie EPROM de $64K \times 1$. Se pot utiliza circuite auxiliare SSI/MSI şi se presupune că dispunem de un semnal de ceas cu o perioadă ceva mai lungă decât timpul de acces la memoria de $64K \times 1$. Care este timpul de acces al memoriei de $2K \times 32$?

Rezolvare:

Capacitatea totală a memoriei rămâne neschimbată, se modifică numai organizarea ei. O soluție este prezentată în figura 5.16, unde spațiul adreselor este împărțit în două, cei 5 biți mai puțin semnificativi fiind generați intern, cu ajutorul unui numărător în inel comandat de semnalul de ceas. La fiecare perioadă de ceas, un alt bit este memorat în registru, astfel încât, după 32 perioade de ceas dispunem de un cuvânt de 32 biți la ieşirea registrului. Timpul de acces al memoriei este de 32 de ori mai mare decât perioada ceasului. Presupunem că timpul de acces la memoria ROM este suficient de mare pentru a evita orice probleme legate de metastabilitate.

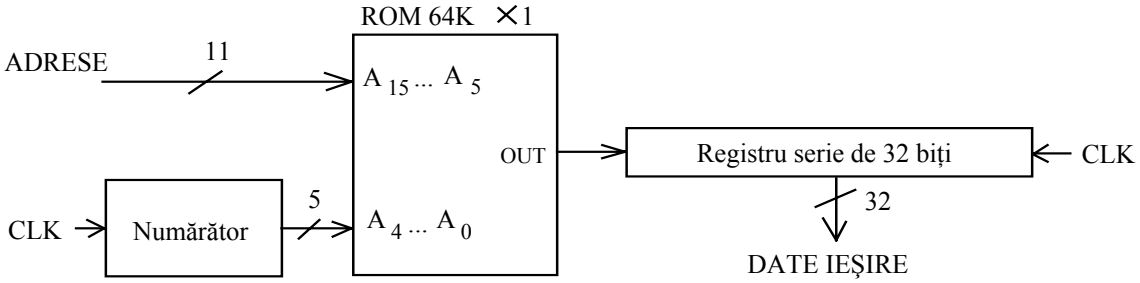


Fig. 5.16 *Schema simplificată a unei memorii de* $2K \times 32$ □

5.3.5 Să se proiecteze un numărător binar reversibil pentru controlerul unui lift într-o clădire cu 20 etaje, folosind un singur circuit PAL16R6. Numărătorul trebuie să aibă o intrare care permite numărarea şi o intrare care stabileşte sensul de numărare. La numărarea înapoi trebuie să se blocheze în starea 1, iar la numărarea înainte trebuie să se blocheze în starea 20. În orice sens de numărare se sare peste starea 13. Reprezentați diagrama stărilor şi ecuațiile scrise în ABEL.

Rezolvare:

Circuitul integrat PAL16R6 are 8 intrări, 2 intrări/ieşiri programabile şi 6 bistabile de tip D având ieşirile cu 3 stări. Matricea de 64×32 comutatoare programabile permite implementarea oricărei funcții binare. Pentru cele 20 stări sunt necesare 5 bistabile. Dacă notăm cu E intrarea care permite numărarea (activă pe 1 logic) şi cu S intrarea de sens (S = 1 crescător, iar S = 0 descrescător), atunci diagrama stărilor este cea din figura 5.17. Detalii despre limbajul ABEL sunt date în curs. O implementare completă de automat finit cu PAL16R4 este dată în capitolul următor.

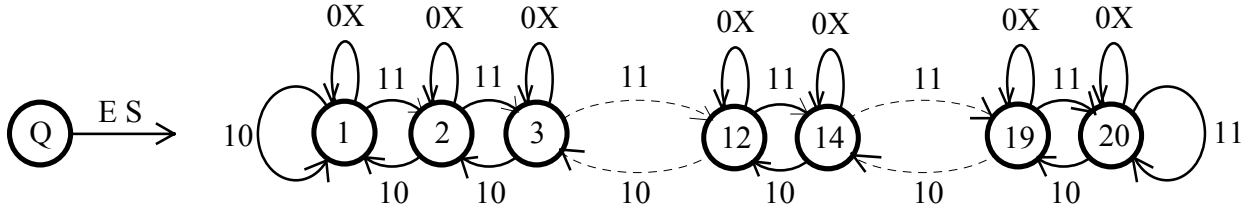


Fig. 5.17 *Diagrama stărilor pentru numărătorul din problema 5.3.5* □