UNIVERSITATEA TEHNICĂ DIN TIMIŞOARA

FACULTATEA DE ELECTRONICĂ ŞI TELECOMINICAŢII

Ing. MARIUS CRIŞAN

CONTROLUL OPERAŢIILOR DE PROCESARE ÎN SISTEMELE NUMERICE COMPLEXE DE MĂSURARE

TEZĂ DE DOCTORAT

Conducător ştiinţific,

Prof.dr.ing. EUGEN POP

TIMIŞOARA

1993

C U P R I N S

INTRODUCERE

Foarte multe preocupări sînt îndreptate, în prezent, spre găsirea de noi soluţii în vederea îmbunătăţirii fiabilităţii sistemelor numerice. Este un fapt stabilit deja că un factor determinant în fiabilitatea sistemelor digitale îl au defecţiunile cu mod de manifestare tranzitoriu sau intermitent, a căror frecvenţă de apariţie este de cel puţin un ordin de mărime mai mare ca a defecţiunilor permanente. Problematica detecţiei unor asemenea tipuri de erori presupune verificarea sistemului în timp real, sau concurent cu execuţia sarcinilor de aplicaţie.

Teza de doctorat abordează aspectul creşterii fiabilităţii sistemelor numerice complexe de măsurare, pe baza efectuării controlului în timp real a operaţiilor de procesare. Lucrarea se întinde pe 10 capitole, fiind anexată şi o listă bibliografică.

Capitolul 1 tratează problematica specifică fiabilităţii echipamentelor numerice, clasificîndu-se principalele tipuri de defecţiuni şi modul lor de manifestare. În capitolul 2 se prezintă o sinteză a soluţiilor de bază în realizarea sistemelor tolerante la defecţiuni, ca punct de plecare al prezentei lucrări. Capitolul 3 abordează sintetic problematica utilizării redundanţei informaţionale la detecţia şi corecţia erorilor. În capitolul 4 se prezintă un studiu asupra realizării controlului operaţiilor aritmetice şi logice pe baza codurilor cu resturi, evidenţiindu-se principalele dezavantaje ale metodei. În capitolul 5 se arată modalitatea originală de a organiza fluxul informaţional în vederea controlului unitar al operaţiilor de procesare. Capitolul 6 detaliază teoretic o metodă originală propusă de a controla prin semnături operaţiile aritmetice şi logice. În capitolul 7 se propune o structură originală a unui sistem de control în timp real a operaţiilor de procesare. Capitolul 8 detaliază aspectele legate de sinteza blocului de control a operaţiilor aritmetice şi logice, simulare a corectei funcţionări precum

şi de modelare a erorilor, verificîndu-se premizele teoretice admise. În capitolul 9 se evaluează performanţele de fiabilitate implicate de soluţia de control peopusă în lucrare, rezultînd avantajele utilizării acesteia. Capitolul 10 conţine concluziile şi reliefarea contribuţiilor originale ale autorului.

Teza de doctorat a fost realizată sub îndrumarea competentă, eficientă şi deosebit de generoasă a conducătorului ştiinţific, domnul prof.dr.ing. Eugen Pop, faţă de care autorul îşi exprimă întreaga sa gratitudine şi consideraţie.

Alese mulţumiri se cuvin a fi aduse domnului prof.dr.ing. Mircea Vlăduţiu, cu care autorul a colaborat îndeaproape un număr însemnat de ani şi ale cărui sugestii şi recomandări i-au deschis posibilitatea abordării domeniului de cercetare ştiinţifică şi finalizării lucrării.

Autorul aduce de asemenea mulţumiri absolvenţilor Ileana Dinescu, Ecaterina Keszeg şi Gabriel Dochian pentru ajutorul acordat la realizarea părţii aplicative a lucrării.

CAPITOLUL 1

PROBLEMATICA SPECIFICĂ FIABILITĂŢII SISTEMELOR NUMERICE

1.1 FIABILITATEA SISTEMELOR NUMERICE

Creşterea spectaculoasă a vitezei de operare şi a complexităţii sistemelor numerice a determinat utilizarea acestora în procese economico-industriale sau ştiinţifice din ce în ce mai complexe. Aceasta a impus într-un mod obiectiv, ca o consecinţă, asigurarea unor valori superioare pentru indicatorii de fiabilitate [1], [2].

Într-o primă delimitare putem clasifica procedeele de creştere a fiabilităţii ca specifice celor trei mari faze de realizare a unui produs: proiectarea logică, proiectarea tehnologică şi realizare constructivă.

Realizarea constructivă constă în ultimă instanţă în ansamblarea optimă a sistemului numeric, începînd de la nivelul componentelor pe plachete pînă la conexiunile între module sau echipamente. După asamblare sistemul este supus încercărilor electrice şi mecano-climatice în vederea fiabilizării. Efectul de îmbătrînire accelerată ca urmare a unui proces de tip "burn-in" este determinat de uzura fizică a componentelor, datorată în principal solicitărilor de temperatură şi potenţial electric. Dintre acestea temperatura joacă rolul dominant. Metodele de îmbătrînire accelerată, aplicate prin testoare specializate au condus la o creştere semnificativă a calităţii şi fiabilităţii în ultimii ani [13],[16],[17],[18].

Proiectarea tehnologică constă în principal din specificarea tipului componentelor de circuit utilizate şi alegerea soluţiei de asamblare constructivă. Aceasta presupune etape premergătoare de proiectare, realizată asistat de calculator, constînd în partiţionarea circuitelor integrate dedicate sau pentru aplicaţii specifice (ASIC's), amplasarea componentelor pe plachete, determinarea traseelor de interconectare şi interconectarea la nivel de sertar sau echipament. Optimizarea proiectării sub aspectul diminuării influenţelor perturbative reprezintă măsura specifică de creştere a fiabilităţii pentru această fază.

În faza de proiectare fiabilitatea sistemului poate fi

abordată în două moduri: prin evitarea defectării sistemului, respectiv prin toleranţa la defectări a sistemului.
Evitarea defectării sistemului presupune adoptarea unor tehnici de verificare a proiectării logice, alegerea unor regimuri de funcţionare facile pentru componentele sistemului, utilizarea de componente fiabile [71]. Aceste măsuri, deşi conduc la o creştere a fiabilităţii, nu pot garanta însă funcţionarea corectă pentru situaţia în care apare o defecţiune în sistem. Din acest punct de vedere această manieră de proiectare tradiţională poate fi numită şi intoleranţă la defectări. Este evident că utilizarea unui sistem numeric intolerant la defectări în aplicaţii care înglobează valori deosebite este foarte riscantă. Nivelul de fiabilitate necesar în astfel de aplicaţii trebuie să fie net superior celui asigurat de o proiectare tradiţională. A doua manieră de proiectare va asigura toleranţa la defectări a sistemului, făcînd posibilă operarea chiar atunci cînd în structura sa apar una sau mai multe defecţiuni. În esenţă, toleranţa la defectări constă dintr-o suplimentare a părţii hardware şi/sau software a sistemului, care va acţiona în momentul apariţiei unui defect în vederea menţinerii parametrilor de funcţionare, fie prin mascarea defecţiunilor, fie prin detecţia acestora şi reconfigurarea corespunzătoare a sistemului.

Performanţele de fiabilitate ale unui sistem pot fi apreciate cantitativ prin măsuri deterministe sau probabiliste. Un scop deterministic pentru un sistem tolerant la defectări poate fi acela că nici o defecţiune singulară poate cauza căderea sistemului.

Efectele unei strategii de proiectare tolerante la defectări asupra fiabilităţii sistemului pot fi exprimate în felul următor:

$$R_{sistem} = Prob(\text{fără defect}) + Prob(\text{operare corectă}/\text{defect}) - Prob(\text{defect}). \quad (1.1)$$

Primul termen reprezintă probabilitatea de a nu apare nici un defect. Ea este maximizată de proiectarea intolerantă la defectări, menţionată anterior. Dacă Prob (fără defect) poate fi făcută suficient de mare, se poate atinge un deziderat de fiabilitate pentru un sistem fără a fi necesare strategii de toleranţă la defectări.

Efectele toleranţei la defectare asupra fiabilităţii sînt

reprezentate de cel de-al doilea termen din (1.1), care este probabilitatea de a apare un defect fără a cădea sistemul, calculată asupra tuturor defectelor posibile. Prob (operare corectă/defect), referitoare la acoperirea mecanismului de toleranţă la defectări, reprezintă probabilitatea condiţională ca un sistem să continue să opereze corect în prezenţa unui defect particular. Fiecare termen de acoperire este ponderat de probabilitatea de apariţie a defectului corespunzător. Astfel pentru o proiectare eficientă a sistemului, raportat la cost, mecanismele de toleranţă la defectări trebuie orientate către defecţiunile cele mai probabile de a apare. De remarcat că dacă probabilităţile de defect sînt mari, un sistem poate fi capabil de a tolera tot setul de defecţiuni date şi încă să nu fie suficient de fiabil pentru aplicaţia respectivă. Detecţia automată a defecţiunilor, diagnoza, reparaţia şi mecanismele de acoperire pot reduce sau elimina oprirea funcţionării, îmbunătăţind disponibilitatea.

Proiectantul de sistem tolerant la defectări trebuie să ia de asemenea în considerare performanţele, complexitatea, costul, gabaritul şi alte restricţii, toate care sînt afectate de redundanţa şi strategiile de toleranţă la defectări utilizate. Aceste costuri trebuie apreciate în raport cu consecinţele materiale sau umane implicate de căderea sistemului, care sînt dificil de precizat.

Prezentarea pe larg a modului de obţinere a indicatorilor de fiabilitate şi disponibilitate este arătată în [20],[63].

1.2. OBIECTIVELE TESTĂRII

Aşa cum am precizat, produsele complexe, în clasa cărora se încadrează şi echipamentele numerice, au asociate fiabilităţii noţiunile de mentenabilitate şi disponibilitate, care completează imaginea calităţii. Evenimentul care afectează valoarea acestor indicatori este defectarea, constînd în ultimă instanţă din pierderea capacităţii unui produs de a-şi îndeplini funcţia pentru care a fost realizat. Restabilirea funcţionabilităţii are loc în urma acţiunii, definite prin termenul de mentenanţă, avînd ca prim scop diagnoza defecţiunii. Aceasta, presupunînd detecţia şi localizarea defecţiunii, se îndeplineşte de regulă în urma aplicării unei serii de operaţii, denumite generic de testare, caracterizate

printr-o metodă şi avînd ca suport un anumit dispozitiv tehnic. În funcţie de eficacitate, testul aplicat poate pune în evidenţă, sau nu, prezenţa unui anumit defect, gradul de rezoluţie al testului influenţînd concretizarea diagnozei, prin localizarea defecţiunii.

Faza premergătoare elaborării unui test se identifică cu necesitatea de a defini în mod precis obiectivele urmărite prin testul respectiv, în sensul defectelor pe care ar trebui să le poată pune în evidenţă. Realitatea însă se confruntă cu o gamă largă de defecte posibile, şi ca urmare idealul de a fi toate detectate de testul elaborat nu poate fi atins. O abordare rezonabilă a acestei probleme constă în considerarea doar a defectelor avînd o apariţie mai probabilă, limitîndu-se astfel obiectivele testării la o dimensiune acceptabilă. Mai mult, vor fi luate în considerare nu defectele propriu-zise, ci numai efectele pe care acestea le implică în comportarea obiectului testat, definindu-se astfel obiectivele testării relativ la anumite modele de defecţiuni [3]. Cu toate acestea pe măsura creşterii complexităţii structurilor logice, problema obiectivelor testării se complică, primind noi dimensiuni. În primul rînd, nu toate tipurile de defecţiuni cu şanse mai mari de apariţie pot fi reprezentate prin modele acceptabile, rezultînd de aici necesitatea abordării separate a fiecărui tip de defecţiune prin secvenţe de test corespunzătoare. În al doilea rînd, utilizarea eficientă a modelelor de defecţiuni presupune cunoştinţe la nivelul structurii, aspect dependent de nivelul, la care se realizează testarea. De exemplu, utilizatorul, familiarizat numai cu comportarea funcţională, va urmări ca obiectiv de testare exersarea tuturor funcţiilor specifice produsului.

Odată elaborat un test, se pune în mod natural problema eficienţei acestuia, uzual definită sub forma gradului de acoperire a defectelor, care reprezintă procentul defectelor selectate, care au fost puse în evidenţă. Deşi de dorit, procentul maxim este dificil de atins, pe de-o parte din cauza faptului că anumite defecţiuni pot fi mascate de stările logice normale din schemă, iar pe de altă parte datorită unei testabilităţi restrînse, ce îngreunează atît specificarea gamei de defecţiuni, cît şi elaborarea testelor.

Admiţînd prezenţa defecţiunilor şi a cauzelor de apariţie a acestora dependente specific de fazele pe care le parcurge un

produs (proiectare, fabricaţie şi exploatare), problematica testării va apare ca urmare diferenţiată, în vederea asigurării eficienţei corespunzatoare.

1.3. TIPURI DE DEFECŢIUNI ŞI MODUL LOR DE MANIFESTARE ÎN SISTEMELE NUMERICE

Cunoaşterea tipurilor de defecţiuni, specifice sistemelor numerice, prezintă o importanţă deosebită în vederea modelării, alegerii strategiei şi metodei de testare cea mai eficace. Problema poate fi abordată din mai multe puncte de vedere, literatura de specialitate oferind mai multe încercări de clasificare a defecţiunilor [1],[2],[4],[5]. Prezentăm în continuare, într-o manieră sintetică, o clasificare după principalele criterii, care ar putea fi luate în cosiderare:

-în funcţie de durată sau persistenţă:
  -permanente sau hard;
  -tranzitorii respectiv intermitente sau soft;

-în funcţie de viteza de manifestare:
  -bruşte;
  -progresive;

-în funcţie de nivelul de apariţie în sistem:
  -hardware ;
  -software ;

-în funcţie de aria de răspîndire:
  -independente sau locale;
  -dependente sau distribuite;

-în funcţie de valoarea logică:
  -logice;
  -parametrice;

-în funcţie de dinamica de manifestare a efectelor:
  -statice;
  -dinamice;

-în funcţie de cauzele defectărilor:
  -datorate proiectării;
  -datorate producţiei;
  -datorate exploatării;

-în funcţie de efectele implicate în sistem:
  -minore;
  -majore;
  -catastrofice;

-în funcţie de gradul de reducere a capacităţii de funcţionare a sistemului:
  -parţiale;
  -totale.

Considerăm de interes, prin prisma operaţiilor de testare, prezentarea tipurilor de defecţiuni mai mult sub aspectul

simptomaticii acestora în cadrul structurii logice şi mai puţin în raport cu identificarea lor prin metode statistice pentru studiul fiabilistic.

Relativ la un sistem numeric, vom accepta prin termenul de defecţiune sau defect o imperfecţiune materială sau fizică cauzată de un eveniment de defectare şi care determină modificarea unei variabile logice sau parametru funcţional, faţă de valoarea admisă iniţial. Astfel, simptomul, sau efectul apariţiei unui defect, îl reprezintă eroarea, fiind de cele mai multe ori singura indicaţie despre existenţa defectului în sistem.

Luînd în considerare criteriile de clasificare prezentate mai sus, apare evident că aceleaşi defecţiuni primare vor putea fi regăsite în clasificări după criterii diferite.

În vederea analizei arhitecturilor tolerante la defectări este util de a clasifica defecţiunile corespunzător cu independenţa şi persistenţa lor.

Defecţiunile pot fi atît dependente cît şi independente. Defecţiunile dependente provin de la o defecţiune comună tuturor variantelor. Defecţiunile dependente se manifestă ca erori similare şi conduc la avarii de mod-comun, în timp ce defecţiunile independente , de regulă, cauzează erori distincte şi avarii separate. Defecţiunile sînt clasificate ca hard sau soft bazat pe persistenţa lor. O asemenea distincţie este uzuală în hardware unde natura unei defecţiuni, din acest punct de vedere, este importantă pentru toleranţa la defectări. O componentă afectată de o defecţiune hard trebuie pasivizată după detecţia defecţiunii, în timp ce o componentă afectată de o defecţiune soft poate fi reutilizată după restabilire. Cu alte cuvinte, o defecţiune hard necesită procesarea erorii şi tratamentul defecţiunii, în timp ce o defecţiune soft reclamă numai procesarea erorii. O defecţiune permanentă este o defecţiune hard tipică şi o defecţiune temporară (atît tranzitorie cît şi intermitentă) este o defecţiune soft tipică [54], [55], [56]. La nivelul părţii software a unui sistem de calcul, termenul de defecţiune reprezintă mai mult un abuz de limbaj, neputînd fi asociat unei imperfecţiuni de natură fizică.

În concluzie, acceptăm următoarele semnificaţii pentru defecţiunile hard şi soft: o defecţiune software de tip soft are o probabilitate neglijabilă de recurenţă şi sistemul poate fi restabilit, în timp ce o defecţiune software de tip hard este

recurentă în operarea normală şi sistemul nu mai poate fi restabilit.

1.3.1. DEFECŢIUNI HARD

Defecţiunile hard se identifică de fapt cu defecţiunile fizice din sistem şi pot afecta valorile atît ale variabilelor logice, cît şi ale parametrilor statici şi dinamici. În cazul în care variabila logică afectată de defect va avea o valoare admisă pentru una din cele două stări logice, ne vom afla în situaţia unei defecţiuni logice. Celelalte cazuri, caracterizate prin degradarea mărimilor specifice pentru curent, tensiune şi timp se încadrează în clasa defecţiunilor parametrice. Prin urmare, procesul de testare poate fi defalcat corespunzător, în testare funcţională (TF), respectiv testare parametrică (TP).

În cele ce urmează vom detalia prezentarea defecţiunilor hard.

A. Defecţiuni logice. Privită în ansamblu, această clasă de defcţiuni conduce la lipsa datelor sau date eronate şi se datorează întеruperilor, scurtcircuitelor, atît în interiorul, cît şi exteriorul circuitelor integrate, imperfecţiunilor fizice sau conceptuale în cadrul nivelelor de realizare a unui produs.

A1. Puneri pe 0 (PP-0), respectiv pe 1 (PP-1) la nivelul nodurilor de circuit (se mai utilizează şi o notaţie s-a-0, respectiv s-a-1, provenită din literatura de limbă engleză).

A2. Defecţiuni tip scurtcircuit.

Defecţiunile tip scurcircuit sînt cauzate de punţi nedorite, care apar cu precădere în faza de execuţie a lipiturilor, între conductoarele imprimate ale plachetei, putînd provoca la rîndul lor şi alte defecţiuni.

În ultimă instanţă, diferitele tipuri de scurtcircuite au drept efect alterarea comportării statice, logice a schemelor, fapt pentru care unii autori le încadrează împreună cu defectele logice propriu-zise (blocări la 0, respectiv 1 logic) în familia mai largă a defectelor de curent continuu [10] sau a defectelor logice [11]. Insistînd asupra malfuncţionării logice determinată de scurtcircuitele dintre conductoare de semnal, subliniem dependenţa acesteia de dominanţa valorii logice specifică circuitelor utilizate, caracteristică în strînsă legătură cu tehnologia de integrare apelată la implementarea schemelor de pe plachetă [10],[11].

Pe lîngă defecţiunile logice de tip static, prezentate anterior, mai există o clasă de defecţiuni avînd un mod dinamic de manifestare, sub formă de impulsuri logice, pe care le enumerăm în continuare.

A3. Impulsuri logice eronate ("glitches"). Sub acest nume se definesc de regulă acele impulsuri care apar ca urmare a unor erori de proiectare. Spre exemplu, dacă la proiectarea unui numărător tip modulo aducerea în starea iniţială pe linia de resetare, la finele numărării, se face printr-o logică adiţională, timpul de propagare prin această logică va determina apariţia unui impuls logic eronat la ieşire.

A4. Impulsuri parazite ("spikes"). Aceste impulsuri apar, în general, datorită cuplajelor capacitive între liniile unei magistrale, în momentul tranziţiei unor semnale cu fronturi crescătoare abrupte. Impulsul parazit, prezent pe linia perturbată un scurt timp, poate determina comanda eronată a altor diapozitive logice cuplate pe linie.

A5. Implusuri eronate datorate semnalelor cu fronturi diferite ("races"). Acestea sînt similare cu cele descrise anterior şi iau naştere cînd semnalele cu fronturi diferite comandă acelaşi circuit logic.

A6. Oscilaţii. Dacă inductivitatea la una dintre intrările unei porţi este de valoare ridicată, frontul ascuţit al unui semnal aplicat la intrare poate cauza oscilaţii, a căror amplitudine să depăşească de cîteva ori nivelul de prag al porţii, determinînd impulsuri eronate la ieşire. Oscilaţii similare pot apare şi datorită fenomenului de reflexie, evitarea acestora realizîndu-se prin restricţii impuse fazei de proiectare tehnologică.

B. Defecţiuni parametrice. Majoritatea acestor aşa numite defecţiuni parametrice determină, la o exploatare de durată apariţia de neconcordanţe şi din punct de vedere funcţional. Sub acest aspect, parametrii de interes sînt, pe de-o parte, cei statici, constînd din tensiuni şi curenţi de intrare, respectiv ieşire, precum şi din curenţi de alimentare, iar pe de altă parte, cei dinamici, constînd, în esenţă, din timpii de propagare, respectiv de tranziţie. Aceste defecţiuni au drept efect alterarea comportării dinamice a schemelor, fapt pentru care sînt întîlnite şi sub denumirea de defecte de curent alternativ [10]. Ele pot fi provocate de deficienţe multiple ale procesului tehnologic.

În continuare, se impun cîteva precizări referitoare la modul de manifestare a defecţiunilor de tip hard în cazul sistemelor de calcul cu microprocesor, avînd în vedere structura acestora orientată pe magistrale [5]. Distingem trei mari clase de defecţiuni, cu diferite posibilităţi de apariţie :

1) Linii ale magistralei de date scurtcircuitate. Ca urmare, microprocersorul devine incapabil de a citi instrucţiunile corespunzătoare din ROM, sistemul fiind complet dezactivat.

2) Linii ale magistralei de adrese scurtcircuitate. Aceasta va determina imposibilitatea accesului în anumite zone din spaţiul de adresare.

3) Conflict pe magistrala de date. Cauza uzuală a defecţiunii se află la nivelul decodificatorului de adrese, care este activat la o adresă diferită de cea prescrisă.

În ceea ce priveşte defecţiunile care pot apare la nivelul μP, ca prim abonat la magistrală, ele se pot împărţi în : (1) execuţia eronată a anumitor instrucţii, (2) interacţiunea sau defecţiunea la nivelul registrelor interne şi (3) sensibilitatea la anumite secvenţe de instrucţii.

Printre defecţiunile hard cele mai răspîndite la nivelul memoriilor de tip RAM, caracterizate prin densitate mare de integrare, distingem :

a) Defectarea decodificatorului de adrese. Aceasta înseamnă că anumite zone de memorie nu pot fi adresate.

b) Înscrieri multiple. Înscrierea într-o celulă a unei capsule de memorie este însoţită de înscrierea a aceleaşi date în mai multe celule.

c) Sensibilitate la modele de biţi. Înscrierea unei valori într-o celulă x va determina modificarea conţinutului unei alte celule y, chiar dacă celulele au fost adresate corect.

d) Blocarea amplificatoarelor de ieşire. Se manifestă prin tendinţa de a favoriza valoarea unui bit după citirea unui lung şir de biţi similari.

e) Împrospătarea necorespunzătoare.

f) Timp de acces mărit la revenire după înscriere. Timpul de acces poate creşte datorită saturării amplificatorului de intrare/ieşire.

g) Întreruperi şi scurtcircuite în structura internă.

Cauzele care duc la apariţia defecţiunilor de tip hard în memoriile RAM sînt următoarele : metalizare săracă şi

întreruperea legăturilor la paduri; defecţiuni la nivelul stratului de oxidare; contaminare cu ioni.

Defecţiunile ce pot apare în memoriile de tip ROM sînt incluse în cele caracteristice pentru memoriile de tip RAM.

Prezentarea modului de manifestare şi a cauzelor de apariţie a defecţiunilor hard poate fi adîncită din mai multe puncte de vedere, evident în funcţie şi de gradul de complexitate a sistemului numeric implementat. Aceasta, însă, prezintă interes doar pentru defecţiunile cu probabilitate mai mare de apariţie, corespunzător cu etapa de realizare, în care se află produsul. O prezentare detaliată a fost realizată de autor în [44].

1.3.2. DEFECŢIUNI SOFT

Defecţiunile de tip soft le vom defini, de fapt, prin erorile ce apar în cadrul programului. Din punctul de vedere al caracterului nepermanent de manifestare o modalitate de clasificare ar fi următoarea:

1. Instrucţiuni eronate. Prezenţa lor determină ca programul să nu fie executat în maniera pentru care a fost conceput. Aceeaşi situaţie apare şi datorită omisiunii unor instrucţii, adresare incorectă, etc.

2. Defecţiuni latente. Fie o parte dintr-un program, unde o operaţie de adunare este urmată de o instrucţie de salt necondiţionat. Dacă instrucţia de salt (cod C3) este schimbată din greşeală cu o instrucţie de salt condiţionat "jump-non-zero" (cod C2), programul se va desfăşura în modul aşteptat atît timp cît rezultatul adunării registrului cu acumulatorul este diferit de zero. În cazul în care rezultatul este zero, saltul nu se va efectua şi programul va continua cu execuţia altor instrucţiuni, ceea ce evident nu se doreşte. Erorile se pot datora instrucţiunilor greşite în program sau chiar defecţiunilor de tip hard. Din acest motiv defecţiunile de tipul menţionat ar putea fi clasificate printre cele hard. Aşa cum se poate observa, singura diferenţă dintre cele două instrucţii de salt o constituie bitul cel mai puţin semnificativ.

3. Defecţiuni datorate timpului. Spre exemplu, în unele programe de comunicaţie între procesor şi diverse echipamente periferice, între care există diferenţe de viteză, rata de transmisie este controlată prin software necorespunzător.

Deşi defecţiunile soft ca şi cele hard, cauzează pierderea datelor, dispozitivul de memorie, în speţă de tip RAM, nu suferă o defectare fizică. Recitirea celulei defecte va asigura aceeaşi dată eronată, în schimb după reînscrierea datei celula va funcţiona corect.

Aceste erori soft la nivelul memoriilor se pot datora perturbaţiilor prin diafonie şi pe barele de alimentare.

O sursă principală a erorilor de tip soft la memoriile RAM se datorează particulelor alfa, care influenţează sarcina celulei, conducînd la modificarea valorii binare a acesteia. Fenomenul s-a intensificat pe măsura creşterii densităţii de integrare a memoriilor, prin reducerea atît a mărimii cît şi a sarcinii celulelor. Dacă în trecut suportul de încapsulare era singura sursă de particule alfa care puteau conduce la erori, dimensiunile reduse de astăzi sînt sensibile şi la radiaţia cosmică [13], [14]. O ameliorare în această direcţie a fost totuşi obţinută, recent, prin realizarea unor ecrane protective din material semiconductor, aplicate direct pe suprafaţa activă a dispozitivului [15].

Din cauza caracterului aleator de manifestare punerea în evidenţă a acestor tipuri de defecţiuni este extrem de dificilă. Mai mult decît atît, rata defecţiunilor tranzitorii depăşeşte cel puţin cu un ordin de mărime pe cea a defecţiunilor permanente. Impotriva lor deosebit de eficace devine utilizarea tehnicilor de testare concurentă, de corecţie a erorilor pe bază de coduri [76],[79],[87], sau de proiectare tolerantă la defecţiuni. Acesta este domeniul în care orientăm cercetarea în prezenta lucrare.

1.3.3. MODELE DE DEFECŢIUNI

Cîteva dintre modelele mai uzitate, sînt prezentate în continuare.

a. Modelul de defecţiune "punere pe" ("stuck-at")

Este cel mai răspîndit model de defecţiune şi a apărut odată cu primele familii de circuite logice. Modelul de "punere pe" şi-a păstrat utilitatea, chiar cu creşterea complexităţii tehnologiei de integrare a componentelor, pe de-o parte datorită simplităţii, iar pe de altă parte din cauza eficienţei, dovedită de experienţă, pe care o au testele dedicate defecţiunilor de "punere pe" în detecţia şi a altor tipuri de defecţiuni.

Cu toate acestea, există situaţii, datorate apariţiei circuitelor MOS şi a utilizării dispozitivelor de tip RAM, în care validitatea acestui tip de model de defecţiune trebuie reconsiderată.

b. Modelul de defecţiune "întrerupere" ("open circuit")

Defecţiunile de tip întrerupere în tehnologia TTL sînt, în general, acoperite de modelul de defecţiune "punere pe". Ca urmare, considerarea separată a acestui tip de defecţiune nu mai este necesară. Totuşi, o dată cu apariţia tehnologiilor MOS, această premiză nu se mai menţine valabilă, implicînd astfel utilizarea unui model de defecţiune specific.

c. Modelul de defecţiune "punte" ("bridging")

Acesta acoperă, de fapt, defecţiunile de tip scurtcircuit atît la nivelul circuitelor integrate cît şi al plachetei.

d. Modelul de defecţiune de timp ("timing")

În principiu, evitarea utilizării circuitelor logice de tip asincron, în cazul unui proiect, conduce la minimizarea riscului de apariţie a erorilor datorate parametrilor de timp. Totuşi, în situaţia în care comportarea dinamică a schemei este dependentă strict de valorile acestor parametri, utilizarea unui model de defecţiune de timp apare ca necesară, deşi inserarea acestui tip de defecţiune într-un circuit funcţional corect, sau simularea devine extrem de dificilă.

CAPITOLUL 2

SISTEME NUMERICE TOLERANTE LA DEFECTĂRI

2.1. ELEMENTE ALE STRATEGIILOR DE TOLERANŢĂ LA DEFECTĂRI

Obiectivele propuse în prezenta lucrare sînt orientate cu precădere spre tratarea erorilor de tip tranzitoriu sau intermitent. O modalitate de abordare a acestei problematici o constituie testarea concurentă, ca un prim pas spre realizarea unor structuri tolerante la defectări.

Toleranţa la defectări într-un sistem digital se realizează prin redundanţa la nivel hardware, software, informaţional şi /sau computaţional [62],[65],[73]. O astfel de redundanţă poate fi implementată în configuraţii statice, dinamice sau hibride. O strategie de toleranţă la defectări poate include unul sau mai multe din următoarele elemente:

- Mascarea. Corecţia dinamică a erorilor generate.
- Detecţia. Detecţia unei erori ca simptom al unei defecţiuni.
- Izolarea. Prevenirea propagării erorii peste anumite graniţe definite.
- Diagnoza. Identificarea modulului defect responsabil pentru eroarea detectată.
- Reparaţia/reconfigurarea. Eliminarea sau înlocuirea unei componente defecte sau mecanism prin şuntarea lui.
- Restabilirea. Corecţia sistemului la o stare acceptabilă pentru a continua funcţionarea.

Pentru o operare pe termen scurt ultrafiabilă, cînd nu există timp disponibil pentru diagnoza şi reparaţia off-line, se prevede o configuraţie statică sau pasivă de elemente pentru a masca un număr maxim dat de defecţiuni.

Redundanţa dinamică, pe de altă parte, implică comutarea de module sau reorientarea canalelor de comunicaţie la apariţia defecţiunilor. Componentele defecte sînt detectate, diagnozate şi reparate sau înlocuite.

În soluţia hibridă, o configuraţie statică de bază maschează un număr de defecţiuni, în timp ce modulele defecte sînt detectate şi înlocuite în această configuraţie. Redundanţa hibridă este convenabilă pentru aplcaţii de termen lung

ultrafiabile în care probabilitatea defectărilor multiple este mare.

Aplicaţiile de înaltă disponibilitate nu impun în mod necesar operarea neîntreruptă fără erori, deşi bazele de date şi alte resurse critice trebuie protejate. În asemenea cazuri, se preferă detecţia şi izolarea erorilor în cadrul modulelor înlocuibile, în loc de mascare. Funcţionarea sistemului este apoi oprită pentru a opera diagnoza, reconfigurarea şi restabilirea.

În continuare, se indică sintetic aceste strategii reliefîndu-se mecanismul tolerării defectelor [21], [57], [58], [64], [66], [67], [72], [75].

A. Diagnosticarea defectărilor şi înlocuirea componentelor defecte.

A.1. Cu structură redundantă selectivă : diagnosticarea elementelor se face individual pentru fiecare componentă.

A.1.1. Fără elemente funcţionale de rezervă : are loc autodiagnosticarea defectelor şi oprirea funcţionării sistemului.

A.1.2. Se face apel la elementele funcţionale de rezervă.

A.1.2.1. Rezervă activată.

A.1.2.2. Rezervă neactivată.

A.2. Cu structură redundantă masivă : diagnosticarea se face prin compararea funcţiilor de ieşire ale elementelor funcţionale.

A.2.1. Cu posibilitatea de diagnosticare a elementelor defecte.

A.2.1.1. Formarea unor elemente perechi prin aplicarea unei rezerve la apariţia defectului : sistem "duplex dinamic".

A.2.1.1.1. Rezervă activată.

A.2.1.1.2. Rezervă neactivată.

A.2.1.2. Continuarea funcţionării numai a elementului în stare de funcţionare fără a mai utiliza elementul defectat: degradarea securităţii sistemului (sistem "duplex-simplex").

A.2.2. Fără posibilitatea de diagnosticare a elementelor defecte: la apariţia defectărilor se utilizează elemente de rezervă.

A.2.2.1. Se utilizează elemente perechi de rezervă: "sistem N-duplex".

A.2.2.1.1. Rezervă activată.

A.2.2.1.2. Rezervă neactivată.

A.2.2.2 Continuarea funcţionării pe o singură unitate funcţională (se face apel la o singură rezervă); se produce degradarea securităţii sistemului.

A.2.2.2.1. Rezervă activată.

A.2.2.2.2. Rezervă neactivată.

A.2.3. Fără utilizarea elementului de rezervă la apariţia defectării.

A.2.3.1. Sistemul îşi întrerupe automat funcţionarea la detectarea unei defectări: sistem "duplex" cu oprire.

B. Mascarea defectărilor.

B.1. Cu structură redundantă masivă.

B.1.1. Redundanţă modulară cu N unităţi: sistem NMR.
B.1.1.1. Cu detecţia defectărilor.
B.1.1.1.1. Eliminarea elementului defect: "self purging system".
B.1.1.1.2. Cu oprirea funcţionării sistemului la apariţia a (N-1)/2 defecte.
B.1.1.1.3. Continuarea funcţionării pe o singură unitate: "TMR simplex".
B.1.1.2. Fără detecţia defectărilor.
B.1.2. Mascarea intrinsecă a defectărilor.
B.1.2.1. Utilizarea de structuri redundante de tip "quading logic".
B.1.2.2. Utilizarea de circuite cu securitate intrinsecă de tipul "fail safe".
B.2. Cu structură redundantă selectivă: utilizarea de coduri corectoare de erori.
B.2.1. Cu detecţia elementelor defecte şi oprirea sistemului atunci cînd a fost atins un număr apriori stabilit.
B.2.2. Fără detecţia elementelor defecte.

C. Hibridă.
C.1. Utilizare de coduri corectoare de erori şi diagnosticarea elementelor defecte: se înlocuieşte elementul defect cu un element de rezervă.
C.1.1. Rezervă activată.
C.1.2. Rezervă neactivată.

În continuare vom detalia cîteva aspecte semnificative.

1. Detecţia, mascarea şi corecţia erorilor. Complexitatea structurii unui sistem numeric influenţează capacitatea de a distinge erorile de valorile corecte. Erorile care apar în elementele de memorare a datelor, cum ar fi registrele şi memoriile, sau în timpul transmisiei datelor pe magistrale sau legături de reţea, sînt mai uşor de detectat ca erorile provenind din modulele care generează sau transformă datele. Mascarea sau corectarea erorilor este mai dificilă deoarece necesită copii multiple ale unui element sau alt tip de redundanţă, astfel încît datele corecte să poată fi extrase din informaţia redundantă. Detecţia şi corecţia erorilor poate fi concurentă cu operarea normală a sistemului sau poate fi executată off-line în timpul intervalelor dedicate testării.

2. Coduri detectoare şi corectoare de erori. Teoria codurilor reprezintă mecanismul cel mai larg dezvoltat în vederea detecţiei şi corecţiei erorilor în sistemele numerice, în general implicînd o redundanţă mai mică decît alte scheme de detecţie şi corecţie [68].

Codurile detectoare şi corectoare de erori diferă într-o gamă largă privitoare la proprietăţile de detecţie şi corecţie, complexitaea codificării şi decodificării şi eficienţa codului [78],[79],[80],[85]. O trecere în revistă a principalelor tipuri

de coduri utilizate se va face în următorul capitol.

3. Logica de autoverificare. Au fost realizate deja cîteva circuite experimentale VLSI, avînd implementate în întregime circuite de autoverificare. Fiecare circuit de autoverificare are intrări şi ieşiri codificate, de regulă in forma a 2 biţi, prin care pentru fiecare linie există două cuvinte de cod valide şi două în afara codului. Un circuit este calificat ca sigur la defectare, dacă, pentru oricare defecţiune specificată din circuit, acesta nu va genera nicodată la ieşire un cod incorect, atunci cînd se stimulează la intrare un cuvînt de cod corect. Pe de altă parte, un circuit autotestabil va da la ieşire un cuvînt din afara codului pentru cel puţin un cuvînt de cod la intrare la fiecare defecţiune posibilă. Un circuit total autotestabil are ambele proprietăţi de sigur la defectare şi autotestare şi prin urmare nici o defecţiune internă nu poate converti o intrare eronată într-o ieşire validă şi cel puţin o intrare normală va detecta fiecare defecţiune internă posibilă.

În ultimii ani s-a manifestat un interes crescînd pentru tehnicile de testare de tip "built-in" (BIT) [24]. Prin aceste tehnici se adaugă suficientă logică suplimentară la un modul sau circuit integrat ca să nu mai fie nevoie de un echipament extern pentru testare.

O tendinţă modernă de proiectare este de a utiliza blocuri de autoverificare care să asigure detecţia concurentă a erorilor în circuitele integrate. Astfel, au fost dezvoltate module de autoverificare atît pentru reţele logice cît şi pentru unităţi centrale de calcul, contribuind la dezvoltarea circuitelor VLSI tolerante la defectări [26], [27], [28], [29], [60], [61], [74], [81], [82].

4. Redundanţa modulară în vederea detecţiei şi mascării erorilor. Frecvent, pentru circuite care generează sau transformă informaţia, redundanţa modulară este singura soluţie disponibilă pentru detecţia şi corecţia erorilor [86]. În figura 2.1 se arată cea mai simplă soluţie pentru detecţia erorilor. Ieşirile a două
module identice sînt comparate, la fiecare tact, evidenţiindu-se

defectarea unuia dintre ele prin formarea semnalului corespunzător de indicare a erorii [19],[21]. Sistemul cu această structură detectează prezenţa unui defect în sistem (atît timp cît nu intervine acelaşi tip de defect simultan în

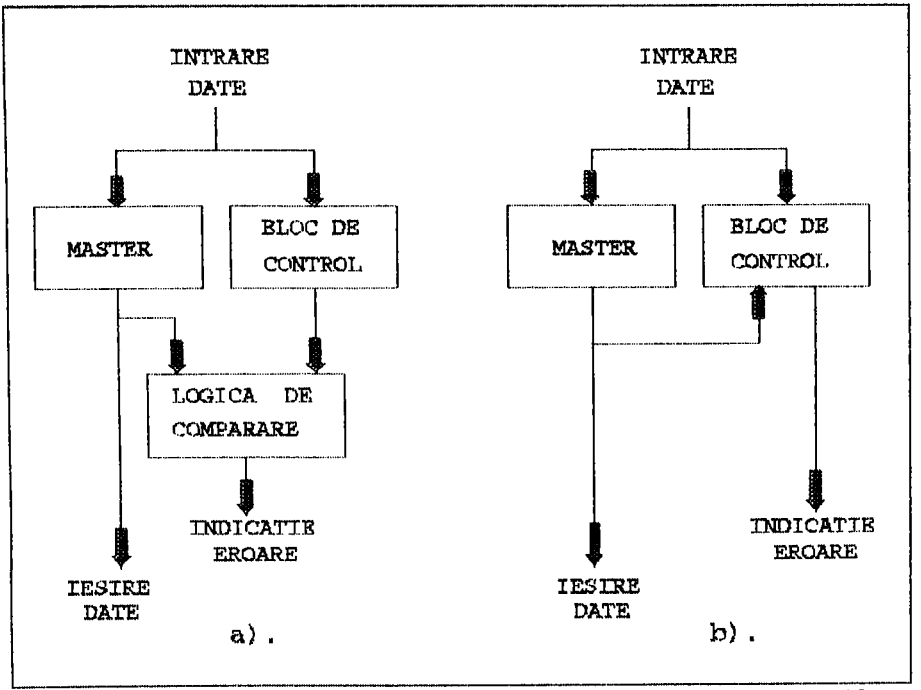

Fig. 2.1. Redundanţa modulară în vederea detecţiei erorilor: a - cu logică de comparare externă; b - cu logică de comparare internă circuitului integrat.

cele două module funcţionale), dar nu poate identifica modulul defect. Pentru diagnoză este necesară oprirea sistemului. În figura 2.1b se prezintă varianta în care compararea se realizează în interiorul circuitului integrat. De menţionat faptul că dublarea unui sistem nu aduce întotdeauna o creştere a fiabilităţii acestuia.

Pentru a asigura o funcţionare continuă se utilizează dispozitive suplimentare de detecţie a erorii, permiţînd autoverificarea modulelor duplicate. Figura 2.2a ilustrează faptul că atunci cînd un modul semnalizează o eroare, el poate fi dezactivat în timp ce celălalt modul continuă să furnizeze informaţia corectă, mascînd de fapt defecţiunea. În figura 2.2b se prezintă un sistem duplex configurat în două perechi de module care se autoverifică, ca în figura 2.1.

Funcţionarea neîntreruptă este adesea asigurată prin votul majoritar al ieşirilor a trei sau mai multe module identice, mascînd defecţiunile minorităţii. În mod extensiv s-a utilizat soluţia redundanţei triple modulare, în sisteme ultrafiabile aerospaţiale şi industriale, cu majoritatea de două voturi din

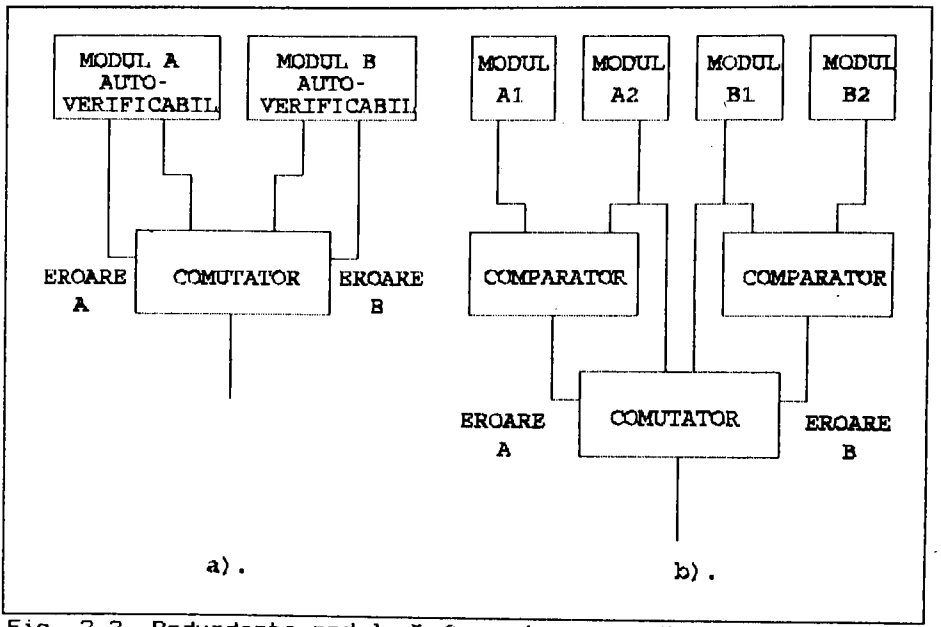

Fig. 2.2. Redundanţa modulară în vederea mascării erorilor: a - prin două module autoverificabile; b - prin două perechi de module care se autoverifică.

trei, mascîndu-se defecţiunile la nivelul unui singur modul. O acoperire suplimentară a defectelor poate fi obţinută cu N module dispuse într-o configuraţie hibridă redundantă modular, în care modulele defecte sînt înlocuite în cadrul unui nucleu configurat triplu modular. Configuraţia hibridă redundantă modular poate masca căderile tuturor modulelor, în afară de două, comparativ cu simpla majoritate din sistemele de votare.

O problemă esenţială în redundanţa modulară este asigurarea sincronizării modulelor replicate. Au fost studiate pe larg scheme de generatoare de tact şi alte măsuri de sincronizare [30], cîteva circuite VLSI recente incluzînd configuraţia pentru sistemul duplex.

Proiectarea sistemelor tolerante la defectări prin utilizarea redundanţei a fost abordată şi sub aspectul unei teorii generale. Aceasta permite proiectarea structurilor tolerante la defectări la nivel de sistem, la nivel de poartă, sau la ambele nivele [31].

5. Izolarea defecţiunilor. Pentru a proteja resursele critice ale sistemului şi minimiza timpul de restabilire, erorile trebuie limitate la modulul în care au apărut. In mod

tipic, graniţele de limitare a erorilor sînt definite ierarhic, cu erorile limitate la nivelul cel mai de jos a modulului înlocuibil sau reparabil. Graniţe suplimentare sînt stabilite în jurul subsistemelor ce conţin aceste module.

Graniţele de limitare pot fi stabilite în două moduri: Fiecare modul îşi poate verifica propriile ieşiri sau fiecare poate verifica informaţia primită. Cea mai uzuală soluţie este de a cere ca fiecare modul să verifice informaţia primită şi să corecteze sau să blocheze datele eronate la nivelul interfeţei.

Dacă un modul este responsabil pentru propria ieşire, devine necesară o graniţă de limitare a erorilor. La interfaţa dintre modul şi magistrala sistem sau canal de comunicaţie va fi plasat un circuit de detecţie sau corecţie a erorilor, împreună cu un circuit capabil să dezactiveze ieşirea modulului. Dacă nu este posibilă corecţia erorii, modulul defect trebuie izolat pentru a preveni propagarea erorii, de regulă fiind efectiv oprit. Un dezavantaj în această configuraţie este acela că interfaţa cu modulul adesea nu poate proteja sistemul împotriva defectărilor propriilor circuite ale interfeţei.

6. Reconfigurare şi reparare. Un sistem poate fi reparat atît prin înlocuirea modulului defect cu unul de rezervă sau prin reconfigurarea structurii sistemului sau redistribuirea sarcinilor de lucru pe celelalte module din sistem [77]. Înlocuirea unui modul restabileşte sistemul la deplina capacitate de funcţionare, însă necesită module redundante neutilizate în funcţionarea normală.

Multe strategii de reconfigurare utilizează toate componentele sistemului în acest sens. La apariţia unei defecţiuni capacitatea sistemului se degradează prin redistribuirea sarcinilor de lucru între celelalte resurse. Redundanţa sistemului se reduce afectînd pe mai departe toleranţa la defectări. Sistemul de calcul al navetei spaţiale este un exemplu al acestei strategii [19].

Unităţile înlocuibile pot fi "calde" sau "reci", pentru o rezervă caldă nefiind necesară iniţializarea cînd se comută în sistem. La proiectarea sistemului trebuie apreciat costul rezervelor neutilizate în raport cu timpul de iniţializare, atunci cînd se decide între rezerve calde sau reci.

Dacă un modul defect nu este înlocuit, capacitatea de operare a sistemului se degradează ca urmare a redistribuirii sarcinilor între resursele rămase active.

7. Restabilirea sistemului. Dacă o eroare nemascată s-a produs prin sistem sau dacă hardware-ul sau software-ul sistemului a fost reconfigurat, este necesară o perioadă de restabilire pentru a corecta sistemul. Timpul între apariţia şi detecţia unei erori determină totalul daunelor şi lungimea perioadei de restabilire.

Majoritatea schemelor de restabilire a sistemului refac capacitatea de operare la o stare prealabilă corectă sau la un punct de restabilire. Un procesor este adus înapoi la un punct de restabilire prin refacerea registrelor şi memoriilor la starea salvată, invalidarea memoriilor (cashe) şi forţarea datelor din acestea de a fi restabilite din memoria globală. Datele globale sînt protejate, de regulă, prin protocoale redundante care permit actualizarea sau refacerea în urma unei defecţiuni. În sistemele multiprocesor cu memorie partajată, datele globale şi listele de sarcini care trebuie executate sînt păstrate în memoria partajată, permiţînd procesoarelor să continue automat cu sarcinile din listă, pe măsură ce procesoarele defecte sînt dezactivate. Această soluţie ajută, de asemenea, de a echilibra sarcinile pe procesoarele individuale [77].

2.2. ARHITECTURI HARDWARE SI SOFTWARE TOLERANTE LA DEFECTĂRI

Aplicaţiile critice în timp real ale sistemelor numerice reclamă posibilitaţi de toleranţă la defectere nu numai pentru componentele hardware dar şi pentru componentele software [32]. Cu toate că în literatură toleranţa la defectări a părţii software a fost puţin tratată, această problemă devine din ce în ce mai serioasă pe măsură ce software-ul creşte în complexitate şi ca urmare nu poate fi cu certitudine validat.

Cele mai bine documentate tehnici pentru toleranţa erorilor de proiectere software sînt soluţia "recovery block" (RB) şi "N- version programming" (NVP). Într-o proiectare diversificată, diferitele sisteme produse dintr-o specificaţie comună se vor numi "variante". O proiectare diversificată are cel puţin două variante şi în plus un element de decizie, care monitorizează rezultatele execuţiei variantei, asigurînd condiţiile iniţiale şi intrările. Specificaţia comună trebuie să adreseze explicit punctele de decizie, adică trebuie să

stabilească cînd se iau decizii şi pe ce date se bazează (datele procesate de elementul de decizie) [33]. În prima soluţie menţionată anterior, variantele sînt denumite alternative şi elementul de decizie este un test de acceptanţă, care este aplicat secvenţial rezultatelor alternative. Dacă rezultatele primei alternative nu satisfac testul de acceptanţă, se va executa a doua alternativă. În cea de-a doua soluţie, variantele sînt denumite versiuni şi elementul de decizie este un vot bazat pe toate rezultatele versiunilor.

Arhitecturile hardware tolerante la defectări echivalente cu RB şi NVP sînt cele cu rezervare pasivă şi respectiv redundanţă N-modulară. O a treia soluţie la toleranţă la defectări hardware, redundanţa activa dinamică, este de asemenea foarte răspîndită, în special cînd se bazează pe componente autoverificabile, dar nu este descrisă în literatură ca o tehnică generică pentru toleranţa la defectări software. Cu toate acestea au fost dezvoltate în trecut programe de autoverificare. Un astfel de program rezultă din adăugarea redundanţei la un program astfel încît să-şi poată verifica comportarea dinamică în timpul execuţiei. O componentă software de autoverificare constă atît dintr-o variantă cît şi dintr-un test de acceptanţă sau din două variante şi un algoritm de comparare.

Toleranţa la defectări este asigurată de funcţionarea paralelă a cel puţin două componente autoverificabile, o componentă asigurînd serviciul sau rezultatele aplicaţiei, în timp ce cealaltă componentă rămîne rezervă caldă. Cînd componenta caldă cade, rezerva continuă să asigure serviciul. Procesarea erorii se face astfel prin detecţia erorii şi posibilitatea comutării rezultatelor. Această soluţie poartă numele de "N self-checking programming" (NSCP). Cînd o componentă de autoverificare software se bazează pe asocierea a două variante, numai o variantă îndeplineşte funcţiile aşteptate, în timp ce cealaltă acţionează ca un test de acceptanţă extins. Fiecare componentă autoverificabilă în NSCP este responsabilă pentru a determina dacă un rezultat este acceptabil, în vreme ce hotărîrea de acceptabilitate în NVP este comună. De asemenea, fiecare test de acceptanţă asociat cu o variantă, sau fiecare algoritm de comparare asociat cu o pereche de variante, poate fi acelaşi sau poate fi în mod

specific derivat dintr-o specificaţie comună pentru fiecare variantă sau pereche de variante. Ca în NVP execuţia paralelă a componentelor necesită un mecanism de a asigura consistenţa de intrare.

Majoritatea sistemelor pentru aplicaţii critice în timp real nu implementează, de fapt, nici o soluţie de restabilire la nivel de bloc, nici o soluţie NVP, ci se bazează pe o autoverificare software. De exemplu, o soluţie în [33] se bazează pe execuţia paralelă a două variante care opresc operarea cînd compararea rezultatelor lor pune în evidenţă o eroare.

Un alt considerent important în restabilirea sistemului este noţiunea de variabile locale şi globale pentru componente. În general, restabilirea din eroare implică ca programul dintre două puncte de decizie să fie de tip procedură, astfel ca activarea şi comportarea lui să nu depindă de vreo stare internă. Cu alte cuvinte, toate datele necesare trebuie să fie date globale.

Privitor la arhitecturile care tolerează atît defecţiunile hardware cît şi software se disting două niveluri de toleranţă la defectări : arhitecturi care tolerează o defecţiune şi arhitecturi care tolerează două defecţiuni consecutive.

Dintre factorii implicaţi în proiectarea tolerantă de defectare, doi sînt deosebiţi de importanţi: numărul de variante şi nivelul la care se aplică toleranţa la defectare.

În afară de considerentele de ordin economic, numărul de variante pentru o metodă dată de toleranţă la defectări este direct legat de numărul de defecţiuni care trebuie tolerate. Natura soft sau hard a defecţiunilor software afectează semnificativ arhitectura numai atunci cînd aceasta trebuie să tolereze mai mult de o defecţiune. Trebuie remarcat aici că, o arhitectură care tolerează o defecţiune hard poate de asemenea să tolereze, în mod teoretic, o secvenţă infinită de defecţiuni soft, considerînd că nu există coincidenţe de defecţiuni.

Relaţia dintre probabilitatea unor asemenea defecţiuni şi numărul de variante este dificil de stabilit, datorită unei multitudini de factori cu influenţe contrare.

CAPITOLUL 3

DETECŢIA ŞI CORECŢIA ERORILOR PE BAZĂ DE CODURI

Codificarea informaţiilor din sistemele de calcul a găsit o largă varietate de aplicaţii, inclusiv pentru memorii de mare viteză şi de masă şi chiar pentru procesoare.

Aplicarea teoriei codurilor în calculatoare dă naştere la o disciplină distinctă de codificare în transmiterea informaţiei, impunîndu-se satisfacerea a trei cerinţe restrictive: viteză, putere consumată şi arie de integrare.

3.1. CONTROLUL ERORILOR ÎN MEMORIILE DE MARE VITEZĂ

Memoriile de tip "cashes" de mare viteză şi memoriile principale sînt susceptibile de erori soft. Datorită acestui fapt s-au utilizat în proiectare coduri detectoare şi corectoare de erori. Pentru ca un cod să fie util la memoriile de mare viteză, structura sa trebuie să permită codificarea şi decodificarea paralelă rapidă.

Coduri detectoare şi corectoare de erori la nivel de bit. În memoriile de mare viteză, cel mai des utilizate sînt codurile detectoare de erori singulare şi corectoare de erori duble (SEC-DED codes). Aceasta se justifică deoarece multe memorii semiconductoare RAM sînt organizate pe un bit de date şi prin urmare orice defecţiune într-o capsulă se manifestă ca un bit eronat în cuvîntul de date.

Coduri Hsiao. Se consideră 2 vectori de lungime k, de pondere impară. Se numeşte ponderea unui vector numărul componentelor sale diferite de zero. De remarcat că suma a doi vectori de lungime k cu pondere impară este un vector de lungime k cu pondere pară. Pe baza acestei proprietăţi se poate construi un cod SEC-DED cu k biţi de control.

Cercetările arată că utilizarea codurilor SEC-DED cu coloane de pondere impară are două avantaje practice: simplitatea codificării/decodificării şi probabilitatea scăzută de decodificare eronată. Aceste coduri sînt, prin urmare, larg utilizate, spre exemplu în sistemele IBM, Cray şi Tandem. Circuitele integrate comerciale disponibile pentru detecţia şi corecţia paralelă a erorilor sînt bazate pe astfel

de coduri [22],[23],[87].

Codificarea în cazul erorilor multiple. Erorile soft, în memoriile de mare densitate, pot apare alături de alte erori hard existente, dînd naştere la erori multiple care nu sînt corectabile cu codurile SEC-DED.

Metoda directă de a corecta erorile multiple este de a utiliza coduri corectoare de erori multiple. Dintre acestea, o răspîndire mare au căpătat codurile corectoare de erori duble, cel mai cunoscut fiind codul Bose-Chaudhuri-Hocquenghem (BCH). În [37] se prezintă o metodă de realizare a decodificatoarelor rapide pentru codurile BCH şi Reed-Solomon, destinate corecţiei erorilor multiple. Aceste coduri reclamă însă de două ori mai mulţi biţi de control ca în cazul codurilor SEC-DED şi prin urmare decodificarea devine mai complexă.

În vederea soluţionării problemelor menţionate s-au propus tehnici care utilizează extensii ale codurilor SEC-DED. Aceste tehnici utilizează "corecţia la ştergere" pentru erorile a căror locaţie este deja cunoscută apriori. Cunoaşterea locaţiei permite unui cod cu distanţa 4 (cod SEC-DED) de a corecta pînă la trei ştergeri. Într-o altă metodă (address skewing) erorile multiple de la aceeaşi adresă sînt dispersate ca erori singulare la diferite adrese. Erorile singulare pot fi apoi corectate [38]. O altă tehnică pentru corecţia erorilor multiple (read-retry-technique) utilizează cicluri repetate de citire pentru a elimina erorile software. Un exemplu este circuitul comercial DP 8400 [22].

Pe baza codurilor SEC-DED au fost propuse circuite care realizează detecţia şi localizarea concurentă a defecţiunii în condiţii normale de intrare, asigurîndu-se o indicaţie separată internă a defecţiunii [39].

Coduri detectoare şi corectoare de erori la nivel de byte. În anumite aplicaţii este recomandabil de a avea un cod corector de erori capabil de a corecta/detecta atît erori la nivel de byte cît şi la nivel de bit.

Coduri corectoare de erori la nivel de byte. Codul, cunoscut sub numele de cod SbEC, este capabil de a corecta toate erorile de b biţi la nivelul unui singur byte [36].

Deoarece codurile SbEC nu garantează detecţia erorilor duble la nivel de bit, împrăştiate pe distanţa de doi octeţi, aceste coduri nu sînt utilizate în sistemele de calcul. Se preferă utilizarea codurilor corectoare de b biţi dintr-un

byte şi de b biţi din 2 octeţi, denumite coduri SbEC-DbED. Codurile Reed-Solomon sînt o clasă generală de coduri de orice distanţă d peste GF(q), din care, ca un caz aparte, se pot deriva codurile SbEC-DbED de distanţă 4 peste $GF(2^{}$).

O clasă de coduri, avînd lungimile de cod şi de byte arbitrare, afost propusă de Kaneda şi Fujiwara în [36].

Un tip interesant de cod SbEC-SbED este codul modularizat. În [36] se dă un exemplu de cod modularizat S4EC-D4ED (80,64) utilizat în sistemul Fujitsu-380/383. Un alt exemplu de cod SED-DED (80,64) în $GF(2^{})$ este prezentat în [41].

Coduri SEC-DED detectoare de erori la nivel de byte. O extensie în continuare, o reprezintă clasa de coduri care detectează erori de un singur byte şi corectează erori singulare şi detectează erori duble la nivel de bit. Codurile SEC-DED-SbED pot fi atractive, din moment ce implică numai o mică mărire a redundanţei. Aceste coduri au fost pe larg studiate, dar un optim care să satisfacă pentru orice lungime arbitrară a lui b încă nu a fost găsit.

3.2 CONTROLUL ERORILOR ÎN MEMORIILE DE MASĂ

Problemele caracteristice cu benzile şi discurile magnetice şi cu discurile optice constau în pachete de erori. Codurile R-S detectoare/corectoare de erori, descrise mai sus, intercalate cu alte coduri, împreună cu tehnici de corecţie la ştergere reprezintă o soluţie eficientă de rezolvare a acestor probleme [80].

Coduri pentru memoriile de bandă magnetică. Datorită densităţilor şi vitezelor crescute ale sistemelor de bandă devine necesară utilizarea unor coduri corectoare de erori mai sofisticate. Noile scheme de codificatoare includ aşa numitul "optimal-rectangular code" (ORC) pentru unităţile de bandă de 6250 biţi/inch, 9 piste şi respectiv "adaptive cross-party (AXP) code" pentru unităţi cu densităţi mai mari, 18 piste [36].

Coduri pentru memoriile de disc. Pentru corecţia pachetelor de erori la memoriile de disc cu largă utilizare pînă în prezent au fost codurile Fire, aparţinînd clasei codurilor ciclice. În sistemele de disc recente aceste coduri au fost înlocuite cu codul R-S intercalat. În afară de acestea

se mai utilizează şi alte tehnici importante de restabilire, pentru creşterea fiabilităţii, cum ar fi saltul peste o zonă defectă, şi suplimentarea blocului de date şi recitirea.

În cazul discurilor optice digitale este necesară corecţia atît a pachetelor de erori cît şi a erorilor aleatoare.

Sistemul de stocare digital denumit "compact disc ROM" (CD- ROM) are aproape o capacitate de 580 Mbyte pentru un disc. Aceste sisteme utilizează o dublă codificare R-S SbEC şi un cod CRC ("cyclic redundancy check") suplimentar faţă de un nou cod propus, şi anume "cross interleaved R-S code" (CIRC). Cele două coduri R-S utilizate sînt (26,24) şi (45,43) peste GF(2). În felul acesta, datele sînt de fapt cvadruplu codificate. Dacă se include, de asemenea, un CRC atunci datele sînt codificate de 5 ori.

Alte sisteme optice răspîndite pentru memoriile de masă ale calculatoarelor sînt "write-once read-many optical disk" (CD-WORM) şi "writable or erasable optical disk". Aceste memorii utilizează codul R-S cu distanţă mare (distanţa 17) şi o lungime a cuvintului de cod de 120-140 octeţi.

3.3. CODURI DE CONTROL A ERORILOR UNIDIRECŢIONALE

Erorile unidirecţionale se definesc ca fiind o clasă de erori în care modul de manifestare a erorii se presupune a fi numai de forma 1 la 0 sau 0 la 1 într-un cuvînt de cod particular. Totuşi nu există informaţii prealabile cu privire la care tip de eroare se aşteaptă apariţia. Prin urmare, într-o transmisie particulară, receptorul poate recepţiona două cuvinte succesive cu două tipuri diferite de erori, dar fiecare cuvînt individual conţine numai erori de tipul 1 la 0 sau 0 la 1. Se face aici o distincţie între aceste tipuri de erori unidirecţionale şi erorile asimetrice, unde toate cuvintele de cod se presupun a avea acelaşi tip de erori, cunoscut în prealabil.

Codurile dezvoltate special pentru erorile unidirecţionale au primit numai recent o atenţie mai mare [51],[52],[53]. Una din propunerile care se fac în dezvoltarea acestor coduri este aceea că erorile induse de defecţiunile tranzitorii şi intermitente sînt limitate numai la un mic număr de erori simetrice, iar restul reprezintă un număr

nelimitat de erori unidirecţionale. În consecinţă, cercetarea a fost orientată spre dezvoltarea unor coduri care să poată corecta t erori simetrice şi detecta toate erorile unidirecţionale. Presupunerea de aici constă în faptul că aceste defecţiuni cauzează un mic număr de defecţiuni simetrice care necesită a fi corectate, în timp ce defecţiunile care cauzează un număr nelimitat de erori unidirecţionale trebuie detectate [36].

3.4. CONTROLUL ERORILOR DE PROCESARE

Controlul erorilor de procesare reprezintă subiectul prezentei lucrări. Soluţiile utilizate în prezent, puţine la număr, au la bază redundanţa masivă.

În figura 3.1 este ilustrat un concept teoretic al corecţiei erorilor în procesoare, avînd la bază principiul duplicării
[42],[43]. Sistemul utilizează patru perechi procesor-memorie.

Fiecare procesor este pe 16 biţi, în timp ce memoriile sînt

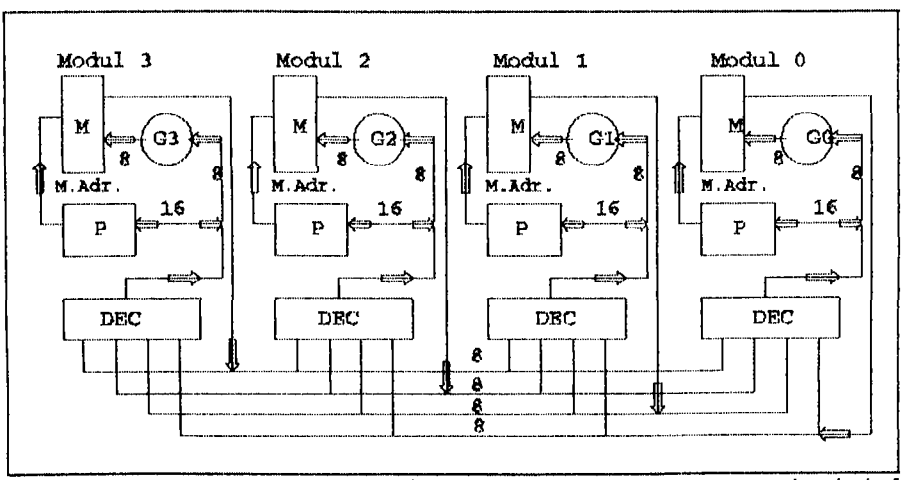

Fig. 3.5. Corecţia erorilor de procesare pe principiul duplicării.

organizate pe 8 biţi. Prin urmare, redundanţa pentru procesoare este de ordinul 4, în timp ce pentru memorii este de ordinul 2. Ieşirea de 16 biţi a fiecărui procesor este

codificată cu o versiune de cod R-S (4,2) peste GF(2), rezultînd un cod de 32 de biţi. Acest cod poate corecta

oricare byte singular de 8 biţi. Ieşirea oricărui procesor este codificată prin codificatoare separate. Fiecare codificator produce numai 8 biţi din cuvîntul de cod de 32 de biţi. Memoria, asociată cu procesorul i, stochează al i-lea byte şi, prin urmare, codificatorul pentru procesorul i produce cei 8 biţi corespunzători cu al i-lea byte. Cu aceste cuvinte, prima memorie a procesorului stochează primul byte, a doua stochează al doilea byte, ş.a.m.d.

La citire, toţi cei patru bytes sînt aduşi din memorie şi decodificaţi prin decodificator, care poate corecta orice eroare a unui singur byte sau orice eroare de 2 biţi în doi octeţi diferiţi. De remarcat că o avarie la un singur procesor sau memorie afectează numai un singur byte în cuvîntul de cod. În felul acesta, sistemul poate tolera avaria oricărei perechi singulare procesor-memorie. În plus, codul poate corecta ştergeri de cîte un byte şi erori singulare la nivel de bit. Astfel, se poate îndepărta pentru reparaţii o pereche procesor-memorie, sistemul continuînd să funcţioneze în baza posibilităţii de corecţie la ştergere. În continuare, orice eroare ce apare la nivel de un bit va putea fi corectată.

Această tehnică pentru controlul erorilor de procesare prezintă în primul rînd dezavantajul unei redundanţe foarte mari. Un alt dezavantaj este legat de faptul că, din moment ce liniile de adresă ale memoriilor nu sînt codificate nu poate avea loc corecţia erorilor la operaţia de scriere.

În general, se apreciază că, în viitor, prin utilizarea unor soluţii neconvenţionale, prin coduri de control, se va putea asigura un control eficient al erorilor de procesare [36]. O astfel de soluţie este propusă în prezenta lucrare.

În acest sens, abordarea problemei controlului prin coduri a erorilor de procesare va porni de la nivelul operaţiilor aritmetice şi logice, ceea ce se va dezvolta în capitolul următor. Codurile utilizate sînt o generalizare a codurilor cu resturi şi prezintă mari asemănări cu codurile ciclice [34],[44],[47].

CAPITOLUL 4

CONTROLUL OPERAŢIILOR ARITMETICE ŞI LOGICE PRIN CODURI CU RESTURI

4.1 NOŢIUNI TEORETICE

Două numere a şi b sînt congruente după modulul p sau modulo p, dacă diferenţa lor a-b este divizibilă prin numărul întreg p:

$$(a-b)=p*q, q \text{ număr întreg} \qquad (4.1)$$

ceea ce se notează

$$a \equiv b \bmod p. \qquad (4.2)$$

Dacă b=r este restul împărţirii numărului a prin p, se scrie:

$$a \equiv r \bmod p. \qquad (4.3)$$

Această congruenţă înseamnă că a este egal cu suma dintre restul r şi un multiplu întreg q de modulo p (sau un multiplu întreg q de baze p) :

$$a=r + q*p. \qquad (4.4)$$

Deoarece r este întotdeauna mai mic decît p, ţinînd seama de diferitele valori ale lui a, acest rest nu poate avea decît p valori distincte: 0, 1, 2,...,p-1.

În continuare se enumeră cîteva proprietăţi ale congruenţelor [88]. Se consideră următorul sistem de congruenţe avînd toate aceeaşi bază:

$$\begin{aligned} a_1 &\equiv r_1 \bmod p \ , \\ &\cdots\cdots\cdots \\ a_n &\equiv r_n \bmod p \ . \end{aligned}$$

1. Cînd se adună congruenţe cu acelaşi modulo, rezultatul este o congruenţă avînd tot acelaşi modulo:

$$\sum_{i=1}^{n} a_i \equiv \left(\sum_{i=1}^{n} r_i\right) \bmod p. \qquad (4.5)$$

2. Corolar. Dacă o congruenţă se scade din alta cu acelaşi modulo, rezultatul este o congruenţă tot de acelaşi modulo:

$$a_1 - a_2 = (r_1 - r_2) \bmod p. \qquad (4.6)$$

3. Corolar. Termenii dintr-o congruenţă pot fi trecuţi dintr-o parte în alta, dacă se schimbă semnul.

4. Cînd se înmulţesc congruenţe de acelaşi modulo, rezultatul este o congruenţă tot de acelaşi modulo:

$$\prod_{i=1}^{n} a_i = (\prod_{i=1}^{n} r_i) \bmod p. \qquad (4.7)$$

5. Corolar. Dacă ambele părţi ale unei congruenţe sînt ridicate la aceeaşi putere sau înmulţite cu aceeaşi constantă, se obţine o congruenţă cu aceeaşi bază.

6. Congruenţele sînt tranzitive. Dacă a≡b şi b≡c, atunci şi a≡c.

7. Dacă numărul întreg, restul şi modulo ale unei congruenţe sînt împărţite cu un factor comun, se obţine tot o congruenţă.

8. Dacă numărul întreg şi restul unei congruenţe sînt împărţite printr-un număr care este factor prim de modulo, se obţine tot o congruenţă.

În general, controlul corectitudinii informaţiei într-un sistem numeric se va putea realiza apelînd la o informaţie redundantă, rezultată în urma codificării informaţiei de bază [89]. Astfel, considerînd sistemul compus din mai multe blocuri Y_i, informaţia redundată se formează prin determinarea legii de codificare C_i în raport cu informaţia de bază. Prelucrarea sau modificarea informaţiei de bază ca urmare a unor operaţii A_i implică determinarea unor legi K_i (care nu coincid cu legile A_i)- coincidenţa se observă numai la dublarea automatelor de control- de modificare a informaţiei de control. Aceste transformări trebuie să se producă astfel încît să se obţină ca rezultat îndeplinirea ambelor operaţii autonome, A_i peste informaţia de bază şi K_i peste informaţia de control, avînd codurile corespunzătoare mutual cu legea iniţială C_i.

Dacă notăm cu I_{bi} şi I_{cbi} informaţia de bază, respectiv de control din blocul Y_i, prin I_{bi}/A_i informaţia de bază după îndeplinirea asupra ei a operaţiei A_i, iar prin I_{bi}/K_i informaţia de control după transformarea ei prin legea K_i atunci pentru realizarea funcţiei de control trebuie să se

îndeplinească următoarea condiţie:

$$\begin{array}{l} I_{bi}=(C_i)I_{kbi}\ , \\ I_{bi}/A_i=(C_i)I_{kbi}/K_i, \end{array} \qquad (4.8)$$

unde simbolul $\equiv(C_i)$ semnifică coincidenţa după legea C_i .

Comparaţia ce se efectuează după (4.8) reprezintă legea logică de bază în vederea organizării dispozitivelor de control a sistemelor numerice. Construcţia dispozitivelor de control pentru diferite operaţii A_i se reduce la căutarea acelei operaţii K_i asupra informaţiei de control care va îndeplini a doua relaţie din (4.8) pentru legea codificării redundante C_i .

Cercetările în acest domeniu sînt orientate în vederea găsirii de noi legi de transformare K_i care să conducă la simplificarea controlului [45], [34].

Structura unui dispozitiv de control a informaţiei de bază presupune următoarele:

- biţi suplimentari pentru reprezentarea, memorarea şi transmiterea informaţiei I_{kbi} şi I_{kbi}/K_i;
- convertor al codurilor de control pentru îndeplinirea asupra informaţiei de control a operaţiei K_i ;
- schema de convoluţie şi comparare (schema de control) pentru încercarea îndeplinirii operaţiei $\equiv(C_i)$ corespunzător informaţiei de bază şi redundante.

Prin urmare, pentru controlul corectitudinii transformărilor informaţiei de bază este necesar să se îndeplinească următoarele operaţii:

- să se formeze un cod de control al rezultatului cu ajutorul îndeplinirii operaţiei K_i asupra biţilor de control pe durata unui timp de conversie t_{conv} . De regulă, această operaţie în timp coincide cu îndeplinirea operaţiei de bază A_i asupra cuvîntului iniţial. De aceea, în continuare, pentru simplitatea exemplului timpul de conversie, atît pentru informaţia de bază, cît şi pentru cea de control, îl luăm egal;
- să se formeze cu ajutorul convoluţiei codul de control din noul cuvînt care reprezintă rezultatul operaţiei A_i pe timpul t_{conv}.
- se compară între ele codul de control format din rezultatul operaţiei K_i şi codul de control care se obţine cu ajutorul convoluţiei rezultatului operaţiei A_i pe timpul de comparaţie

t_{comp}.

Prin urmare, timpul total de control este evaluat, de regulă:

$$t_{control}=t_{conv}+t_{comp}.$$

În situaţia asigurării cotrolului neîntrerupt al blocului care execută un program, adică pentru organizarea controlului tuturor operaţiilor în blocul dat Y_i este necesar să existe o astfel de lege C_i de codificare a informaţiei care să permită îndeplinirea condiţiei (4.8). În caz general, aceasta înseamnă că pentru fiecare bloc trebuie să se îndeplinească $K_{i1},\ldots,K_{iri}$ diferite operaţii asupra informaţiei de control, corespunzător celor $1,\ldots,w_i$ operaţii ale blocului. De aceea, alegerea tipului de lege C_i va trebui să se facă pentru fiecare caz în parte de bloc funcţional.

Apare evident, din analiza relaţiei (4.8), o distincţie clară între organizarea controlului circuitelor "pasive", adică de transmisie şi stocare a informaţiei şi a celor "active", care conduc la transformarea informaţiei de bază, în urma unor operaţii aritmetice şi/sau logice.

Controlul circuitelor "pasive" se poate realiza:

- după modul (pentru diferite metode de determinare a caracteristicii de control);
- cu utilizarea codurilor detectoare şi corectoare de erori (cod Hamming, cod ciclic etc).

Funcţiile de control a dispozitivalor pentru controlul acestor circuite conduce la utilizarea numai a primei coincidenţe din condiţia (4.8), astfel că informaţia de intrare coincide cu cea de ieşire pentru circuitul controlat.

Pentru primul caz, există două feluri de control după modul: al numărului şi al cifrei. Prin controlul numărului codul de control r_i sau cuvîntul care se compară reprezintă în sine restul de la împărţirea numărului A la modulul de control p. Cu alte cuvinte r_i în acest caz reprezintă cel mai mic rest al numărului A după modulul p. De aceea controlul numărului se numeşte control după rest sau după reziduuri:

$$A=r_i \bmod p \qquad (4.9)$$

Prin controlul cifrei codul de control r'_i a numărului N se

determină ca rest de la împărţirea sumei cifrelor numărului la modulul de control d (pentru claritate s-a notat modulul de control al numărului cu p, iar cel al cifrei cu d). Prin aceasta, în caz general, numărul A poate fi reprezentat în orice sistem de numeraţie în baza b:

$$\sum_{i=0}^{n-1} a_i \equiv r'_i \bmod d, \qquad (4.10)$$

unde $a_i=0,1,2,\ldots,b-1$ este valoarea ordinului i a numărului A, n este numărului ordinului numărului A în sistemul de numeraţie b. În prezent pentru controlul informaţiei "pasive" s-au răspîndit cîteva cazuri particulare de control după modul şi anume codul de control cifric al paritǎţii (imparităţii) numărului de unităţi binare după modul, respectiv controlul după resturi prin modulele 3 şi 7 [45],[44],[89].

Controlul paritǎţii (imparităţii) unităţilor binare dintr-un cuvînt exprimat în sistemul binar de numeraţie (b=2) reprezintă de fapt cazul particular de control cifric pentru care codul de control al numărului este egal cu suma cifrelor numărului A după modulul d=b.

Caracteristica de control a paritǎţii P_a a numărului A prin metoda de control cifric reprezintă în sine suma după un modul oarecare a cifrelor numărului A:

$$P_a=\left(\sum_{i=0}^{n-1} a_i\right) \bmod d. \qquad (4.11)$$

În scopul detecţiei erorii ce afectează numărul A se propune un cod de control al numărului pe baza unei metode numerice sub forma [89]:

$$Q_a \equiv (d-1-P_a) \bmod d. \qquad (4.12)$$

În situaţia în care informaţia relativă la numărul A împreună cu codul de control Q_a se pierde, ca urmare a erorii, rezultă:

$$\left[\left(\sum_{i=0}^{n-1} a_i\right) \bmod d\right]_{a'} \equiv 0 \bmod d$$

şi $Q_{a,r}=0$.

Schema de control formează codul de control Q'_a al numărului A, în conformitate cu (4.11) şi (4.12), sub forma:

$$Q'_a=[d-1-(\sum_{i=0}^{n-1}a_i)\bmod d]\bmod d=d-1-0=d-1. \qquad (4.13)$$

Pentru orice modul de control (d>1), $Q'_a\neq 0$ şi prin urmare $Q'_a\neq Q_a$. Prin urmare, la pierderea informaţiei numărului împreună cu caracteristica de control, necoincidenţa codului Q'_a şi Q_a va fi semnalizată de schema de comparaţie printr-un semnal de eroare.

Pentru numere binare, cu modulul de control d=b=2, determinarea codului de control conduce la calculul de imparitate a numărului comun de semne binare în cuvînt:

$$Q_a=[1-(\sum_{i=0}^{n-1}a_i)\bmod 2]\bmod 2. \qquad (4.14)$$

În ceea ce priveşte controlul după resturi, caracteristica de control r_a a numărului A reprezintă restul de la împărţirea numărului dat la modulul de control p:

$$r_a=A \bmod p. \qquad (4.15)$$

Utilizarea acestei metode de control în cazurile practice, conduce la elaborarea unor scheme de convoluţie pentru calculul codurilor de control, evitîndu-se astfel efectuarea de fiecare dată a operaţiei de împărţire propriu-zisă.

Şi în acest caz, controlul poate fi efectuat în două moduri:

1. Controlul numărului după modulul p. Prin metoda de control a numărului restul se determină în concordanţă cu următoarea dependenţă (numărul A se exprimă într-un sistem de numeraţie cu baza b):

$$r_a=(\sum_{i=0}^{n-1}a_i b^i)\bmod p=\sum_{i=0}^{n-1}[a_i(b^i)\bmod p]\bmod p. \qquad (4.16)$$

În sistemul de numeraţie binar codul de control este:

$$r_a=\sum_{i=0}^{n-1}[a_i(2)\bmod p]\bmod p. \qquad (4.17)$$

unde $a_i=0;1$ este valoarea celui de-al i-lea bit a numărului binar. Relaţia conduce de fapt la calculul unităţilor codului numărului ţinînd cont de ponderea bitului. Valoarea factorului $2^i \bmod p$, numit coeficient de pondere a bitului, depinde de poziţia bitului $i=0,1,\ldots,n-1$ şi modulul de control p.

În vederea detecţiei pierderii informaţiei numărului împreună cu caracteristica de control, se recomandă utilizarea unei mărimi suplimentare, într-o manieră asemănătoare cu relaţia (12) de la controlul cifric [89]:

$$S_a \equiv (p-1-r_i) \bmod p. \qquad (4.18)$$

2. Controlul cifrei după modulul $d=p=b^n-1$ ("combinat"). Prin controlul numărului schema de convoluţie formează codul de control ţinînd cont de ponderea fiecărui bit al numărului. Există o posibilitate de a nu calcula ponderea bitului, rezultînd o schemă de convoluţie mai simplă, cu condiţia ca resturile de la împărţirea numărului A să coincidă cu restul de la împărţirea sumei cifrelor numărului A. Problema este de a găsi un asemenea modul de control.

Dacă numărul binar A de n biţi se reprezintă în sistemul de numeraţie b^n

$$A=\sum_{i=0}^{k-1} c_i (b^n)^i, \qquad (4.19)$$

unde $k=n/m$ este numărul de cifre a numărului A în sistemul de numeraţie b^n, iar c_i este cifra de ordinul m a numărului A în sistemul de numeraţie b^n ($c_i=0,1,\ldots,b^{n-1}$), atunci prin modulele $p=b^n-1$ ($p=3;7$), restul de la împărţirea numărului A la modulul p este:

$$r_a \equiv [\sum_{i=0}^{k-1} c_i (b^n)^i] \bmod (b^n-1) \equiv (\sum_{i=0}^{k-1} c_i) \bmod (b^n-1), \qquad (4.20)$$

deoarece $(b^n)^i \equiv 1 \bmod (b^n-1)$.

Prin urmare, pentru obţinerea restului de la împărţirea numărului A la modulul $p=b^n-1$ este suficient de a face suma după modul a tuturor celor k cifre c_i de ordinul m, fără a socoti ponderea lor. Prin aceasta, în concordanţă cu cele prezentate anterior (rel.4.10), se obţine caracteristica de control a

cifrei. În scopul utilizării caracteristicii de control după (4.20), controlul cifrei şi controlul după resturi a numărului sînt identice (cazul controlului "combinat").

În cazul controlului blocurilor "active" informaţia de bază şi cea redundantă iniţială, trebuie să îndeplinească prima relaţie din (4.8), iar rezultatul operaţiei asupra informaţiei de bază şi redundantă trebuie să îndeplinească a doua condiţie din (4.8). Pentru toate operaţiile aritmetice şi logice A_i există o operaţie K_i asupra informaţiei de control (redundante) prin determinarea restului după modulul p, pe baza metodei de control după număr, sau a unei variante combinate.

În ceea ce priveşte operaţia de împărţire exprimată prin

$$Q=\frac{X-R}{Y} \tag{4.21}$$

are loc următoarea dependenţă

$$r_x=[(r_y * r_q)(\mathrm{mod}\ p)+r_r](\mathrm{mod}p), \tag{4.22}$$

unde r_x, r_y, r_q şi r_r sînt resturile de la împărţirea la modulul de control p a deîmpărţitului X, împărţitorului Y, cîtului Q şi restului R.

4.2. CONTROLUL OPERAŢIILOR LOGICE PRIN CODURI CU RESTURI

În vederea controlului operaţiei logice SAU $C=A \vee B$, a operaţiei logice ŞI $C=A \wedge B$ şi a operaţiei logice SAU-EXCLUSIV $C=A \oplus B$, se pot utiliza următoarele relaţii de control:

$$r_V=(r_a+r_b-r_A)\mathrm{mod}\ p, \tag{4.23}$$

$$r_A=(r_a+r_b-r_V)\mathrm{mod}\ p, \tag{4.24}$$

$$r_\oplus=(r_a+r_b-2r_A)\mathrm{mod}p, \tag{4.25}$$

unde r_V, r_A . $r_\oplus$ reprezintă codurile de control ale rezultatului operaţiilor logice SAU(V), ŞI($\wedge$), respectiv SAU-EXCLUSIV($\oplus$).

Operaţia de inversare a unui număr binar cu n biţi (în care se inversează toţi biţii inclusiv cel de semn) descrisă prin

$$\bar{A} = (2^n - 1) - A$$

se poate verifica prin următoarea dependenţă:

$$r_{\bar{A}} \equiv (r_0 - r_A) \bmod p, \tag{4.26}$$

unde r_0 este un parametru ce depinde de numărul de biţi n şi modulul de control p:

$$r_0 \equiv (2^n - 1) \bmod p.$$

Pentru operaţia logică de deplasare cu un bit (în acelaşi timp cu deplasarea numărului va trebui să se deplaseze şi codul lui de control) sînt valabile următoarele reguli de control:

-pentru deplasarea spre stînga fără pierderea informaţiei, adică $A^* = 2A$

$$r_{A^*} = 2r_A \bmod p; \tag{4.27}$$

-pentru deplasarea spre stînga cu pierderea informaţiei bitului cel mai semnificativ

$$r_{A^*} = [2r_A - a_n + \alpha_n (2^n - 1)] \bmod (2^n \mp 1), \tag{4.28}$$

unde a_n este bitul cel mai semnificativ al numărului după deplasare, iar α_n este bitul cel mai semnificativ al codului de control care se pierde prin deplasare;

-pentru deplasarea spre dreapta fără pierderea informaţiei

$$r_{A_*} = r_{A^*} = \frac{1}{2} r_A; \tag{4.29}$$

-pentru deplasarea spre dreapta cu pierderea informaţiei bitului cel mai puţin semnificativ

$$r_{A_*} = \left(\frac{1}{2} r_A \mp a_1 * 2^{n-1} \pm \alpha_1 * 2^{n-1}\right) \bmod (2^n \mp 1), \tag{4.30}$$

unde a_1 este bitul cel mai puţin semnificativ după deplasare, iar α_1 este bitul cel mai puţin semnificativ al codului de control ca urmare a deplasării la dreapta;

-pentru rotaţia la dreapta, prin care bitul cel mai puţin semnificativ nu se pierde ci ocupă locul bitului cel mai

semnificativ, adică

$$A^{-}=\frac{A-a_1}{2}+a_1*2^n,$$

codul de control este:

$$r_e=[\frac{1}{2}r_e+a_1(r_e\mp 2^{n-1})\pm a_1*2^{n-1}]\bmod(2^n\mp 1), \qquad (4.31)$$

unde r_{en} este un coeficient de podere a bitului cel mai semnificativ care ajunge prin rotaţie în poziţia bitului cel mai puţin semnificativ.

În cel de al doilea caz al controlului după modul al cifrei sînt valabile următoarele reguli privitoare la operaţiile logice:

1.Numărul de unităţi binare N_V din codul sumei logice a două numere este egal cu suma unităţilor binare în codurile numerelor iniţiale minus numărul unităţilor binare N_A din codul produsului logic al celor două numere:

$$N_V=N_a+N_b-N_A \qquad (4.32)$$

În realitate, dacă toate unităţile binare în numerele A şi B se află în poziţii care nu coincid atunci numărul de unităţi N_V în codul sumei logice va fi egal cu suma N_a+N_b. În schimb,dacă unităţile în codurile numerelor A şi B se găsesc în aceleaşi poziţii binare, în N_V trebuie introdusă numai o unitate din două. De aceea din (N_a+N_b) se scade N_A unităţi binare, corespunzător numărului de unităţi binare în poziţiile din A şi B care coincid.

2.Nuărul unităţilor binare N_A din codul produsului logic a două numere este egal cu suma unităţilor în codurile numerelor iniţiale minus numărul unităţilor N_V în codul sumei logice a celor două numere:

$$N_A=N_a+N_b-N_V \qquad (4.33)$$

3.Numărul unităţilor binare N_e din codul rezultatului operaţiei SAU-EXCLUSIV este egal cu suma unităţilor în codurile celor două numere minus dublul numărului de unităţi N_A în codul produsului logic al celor două numere:

$$N_{\oplus} = N_a + N_b - 2N_{\wedge}. \qquad (4.34)$$

Dacă toate unităţile în codurile numerelor A şi B se găsesc în poziţii binare diferite, atunci $N_{\oplus}$ va fi egal cu suma $(N_a + N_b)$. Dacă în codurile numerelor A şi B unităţile binare ocupă aceleaşi poziţii, nici o unitate, în acest caz, nu trebuie să fie conţinută în suma $N_{\oplus}$. Prin urmare, din $(N_a + N_b)$ se elimină dublul numărului de unităţi binare care se găsesc în poziţii similare, ceea ce este echivalent cu a scădea $2N_{\wedge}$.

Trecerea de la expresiile pentru număr la expresiile pentru resturile lor, considerînd un modul oarecare, se obţine prin:

$$N_{\vee} \bmod d \equiv (N_a + N_b - N_{\wedge}) \bmod d, \qquad (4.35)$$

$$N_{\wedge} \bmod d \equiv (N_a + N_b - N_{\vee}) \bmod d, \qquad (4.36)$$

$$N_{\oplus} \bmod d \equiv (N_a + N_b - 2N_{\wedge}) \bmod d. \qquad (4.37)$$

Exprimăm egalităţile (4.35)-(4.37) prin codul de control Q, care însoţeşte informaţia de bază şi este păstrat în dispozitive suplimentare de memorare. Avem în vedere că:

$$\begin{aligned} N_a \bmod d &\equiv (d-1-Q_a) \bmod d, \\ N_b \bmod d &\equiv (d-1-Q_b) \bmod d, \\ N_{\vee} \bmod d &\equiv (d-1-Q) \bmod d, \\ N_{\wedge} \bmod d &\equiv (d-1-Q) \bmod d, \\ N_{\oplus} \bmod d &\equiv (d-1-Q_{\oplus}) \bmod d, \end{aligned} \qquad (4.38)$$

unde $Q_{\vee}$, $Q_{\wedge}$ şi $Q_{\oplus}$ sînt codurile de control ale rezultatului operaţiilor $\vee$, $\wedge$ şi $\oplus$ asupra numerelor A şi B. Înlocuind aceste valori în (4.35)-(4.37), în final, după transformări, se obţin următoarele relaţii de control:

$$Q_{\vee} \equiv (Q_a + Q_b - Q_{\wedge}) \bmod d, \qquad (4.39)$$

$$Q_{\wedge} \equiv (Q_a + Q_b - Q_{\vee}) \bmod d, \qquad (4.40)$$

$$Q_{\oplus} \equiv (Q_a + Q_b - 2Q_{\wedge} - 1) \bmod d. \qquad (4.41)$$

După cum se vede din relaţiile (4.39)-(4.41) pentru

controlul corectitudinii efectuării oricărei din cele trei operaţii logice A∨B, A∧B şi A⊕B este necesar de a se îndeplini şi alte operaţii logice suplimentare (∧, ∨ şi ⊕ în mod corespunzător) asupra aceloraşi numere iniţiale A şi B.

De exemplu, pentru controlul operaţiei de sumă logică a două numere A şi B este necesar, în concordanţă cu relaţia (4.39), de a parcurge următoarele etape:

-efectuarea operaţiilor de înmulţire logică A∧B;

-cu ajutorul unei scheme de convoluţie se formează codul de control al produsului logic

$$Q_{A}=(d-1-N_{A} \bmod d) \bmod d;$$

-într-un sumator se calculează codul de control aşteptat Q_v al rezultatului operaţiei de adunare logică, după relaţia (4.39);

-se efectuează operaţia de adunare logică propriu-zisă A∨B;

-schema de convoluţie determină codul de control Q'_v obţinut din suma logică A∨B;

-se compară între ele codurile de control aşteptat Q_v şi calculat Q'_v:

$$Q'_v=Q_v \bmod d,$$

şi dacă nu coincid se generează un semnal de eroare.

În afară de cele trei operaţii logice de bază, a căror control a fost dezvoltat mai sus, în prelucrarea logică a informaţiei mai intervin şi operaţiile de inversare şi deplasare.

Operaţia de inversare are loc pentru toţi biţii cuvîntului, inclusiv bitul de semn. În felul acesta, dacă n este numărul biţilor cuvîntului A în care sînt conţinute N_A unităţi binare, atunci numărul de unităţi $N_{\bar{A}}$ în codul numărului inversat $\bar{A}$ este

$$N_{\bar{A}}=n-N_A. \qquad (4.42)$$

Dacă se are în vedere că $N_A \bmod d=(d-1-Q_A) \bmod d$ relaţia (4.42) devine:

$$(d-1-Q_{\bar{A}}) \bmod d=(n-d+1+Q_A) \bmod d.$$

Prin urmare caracteristica de control a numărului inversat

este:

$$Q_s = [2(d-1) - n - Q_a] \bmod d = -(n+2+Q_a) \bmod d = \\ = [\mu d - (n+2+Q_a)] \bmod d, \qquad (4.43)$$

unde μ este un număr întreg, care îndeplineşte condiţia:

$$|\mu - d - 1| < |n+2+Q_a| \leq |\mu - d| .$$

În felul acesta pentru controlul operaţiei de inversare trebuie parcurse următoarele etape:

-îndeplinirea operaţiei de inversare propriu-zisă, $\bar{A}$;

-determinarea pe baza schemei de convoluţie a codului de control Q'_s din rezultatul $\bar{A}$;

-în sumator se calculează codul de control după relaţia (4.43);

-se compară între ele codurile Q_s şi Q'_s, generîndu-se semnal de eroare în caz de necoincidenţă.

Operaţia logică de deplasare presupune deplasarea informaţiei de bază fie la stînga, fie la dreapta cu un bit şi va fi analizată în trei cazuri: deplasare fără pierderea informaţiei, deplasare cu pierderea informaţiei şi rotaţie.

În primul caz, notînd cu $A^*(\bar{A})$ numărul A deplasat la stînga (dreapta), iar prin $Q^*(\bar{Q})$ codul de control aferent, este adevărată relaţia

$$Q = \bar{Q} \quad ; \quad \bar{Q} = Q, \qquad (4.44)$$

deoarece numărul de unităţi binare în numărul $A^*(\bar{A})$ rămîne nemodificat ($N_A = N_{\bar{A}} = N_{A^*}$).

În cazul al doilea, deplasarea la stînga (dreapta) cu pierderea informaţiei presupune că bitul cel mai semnificativ (cel mai puţin semnificativ) iese din limita lungimii cuvîntului. Dacă $A^*_p(\bar{A}^*_p)$ reprezintă numărul după deplasare la stînga (dreapta) cu pierderea informaţiei, iar $Q_p(\bar{Q}^*_p)$ codul de control corespunzător al numărului deplasat, atunci:

$$N_{A^*_p} = N_A - a_{n-p}, \\ N_{\bar{A}^*_p} = N_{\bar{A}} - a_{1-p}, \qquad (4.45)$$

unde a_n şi a_1 reprezintă în mod corespunzător valoarea bitului

cel mai semnificativ şi mai puţin semnificativ pînă la îndeplinirea deplasării.

Codul de control $Q^{-}_{p}(Q^{+}_{p})$ al numărului A deplasat la stînga (dreapta), pe baza relaţiei (4.12) este:

$$\begin{aligned} Q_p^{-} &= [d-1-(N_A-a_n) \bmod d] \bmod d, \\ Q_p^{-} &= [d-1-(N_A-a_1) \bmod d] \bmod d. \end{aligned} \tag{4.46}$$

Deoarece $(d-1-N_A \bmod d) \bmod d$ este caracteristica de control a numărului A înainte de deplasare, obţinem următoarea dependenţă între codurile de control ale numărului A, înainte şi după deplasarea la stînga (dreapta) cu pierderea informaţiei:

$$\begin{aligned} \overline{Q}_p &= (Q_A+a_n) \bmod d, \\ \overline{Q}_p &= (Q_A+a_1) \bmod d. \end{aligned} \tag{4.47}$$

În cel de al treilea caz, rotaţia presupune transportul ciclic al rangului cel mai semnificativ (de semn) în poziţia celui mai semnificativ. Prin aceasta, chiar dacă în rangul cel mai semnificativ a fost 1, acesta nu s-a pierdut şi deci sînt valabile relaţiile:

$$\begin{aligned} N_A &= N_{A'}, \\ Q_A &= Q_{A'}. \end{aligned} \tag{4.48}$$

Astfel, caracteristica de control a numărului A după rotaţie rămîne neschimbată.

4.3. CONTROLUL OPERAŢIILOR ARITMETICE PRIN CODURI CU RESTURI

Operaţiile aritmetice se referă la adunarea, scăderea, înmulţirea şi împărţirea numerelor.

O primă abordare a problematicii controlului operaţiilor aritmetice face apel la proprietăţile exprimate prin (4.5)-(4.7) şi utilizate de codurile cu resturi.

În calitate de exemplu, în figura 4.1 se prezintă schema unui sumator, în care controlul operaţiei de adunare se face după modul. În sumator se adună operanzii a_1 şi a_2. În continuare, se formează suma şi cel mai mic rest pozitiv după modulul p şi se formează restul sumei (b_1+b_2) după modulul p.

Semnalul de eroare apare în cazul în care $(a_1+a_2) \bmod p \neq (b_1+b_2) \bmod$

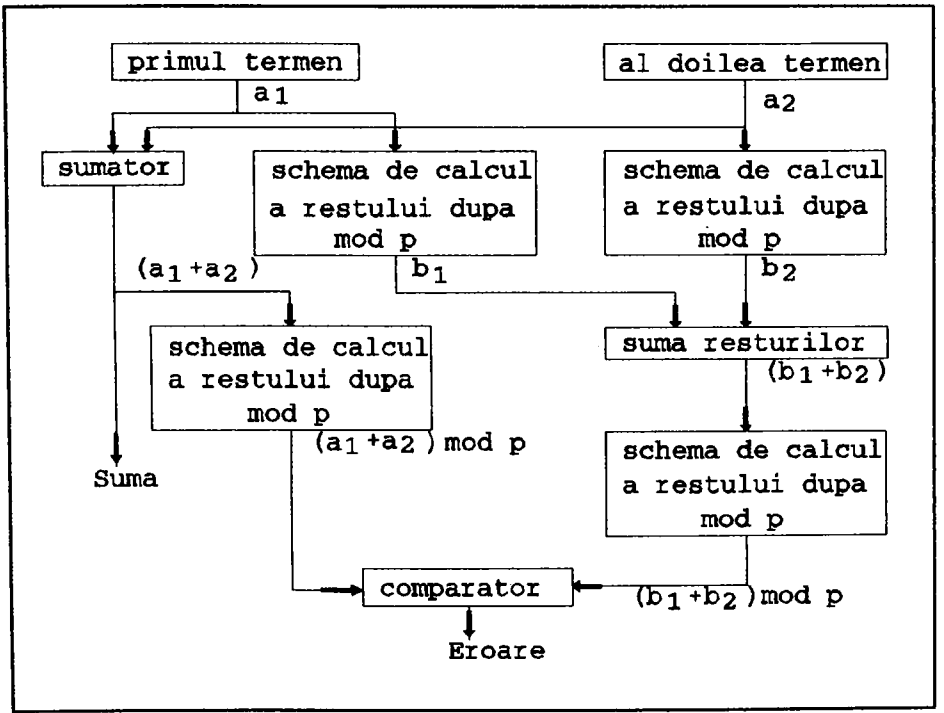

Fig.4.1. Schema sumatorului cu controlul operatiei de adunare dupa modul.

p.

În continuare, se prezintă pentru fiecare operaţie aritmetică dependenţa dintre numărul de unităţi binare din codurile operanzilor şi numărul de unităţi binare din codul rezultatului, introducîndu-se regula de control. Numerele se consideră în valoare absolută reprezentate în codul binar fără semn.

1.Operaţia de adunare. Utilizînd proprietăţile codului binar se poate formula următoarea regulă privitoare la adunarea numerelor binare.

Numărul de unităţi binare în codul termenilor este egal cu suma dintre numărul de unităţi în codul rezultatului şi numărul de unităţi al transportului.

Dacă notăm cu a şi b termenii binari, cu c rezultatul adunării, iar cu N_a, N_b şi N_c numărul de unităţi binare în codurile acestor numere, această regulă se exprimă:

$$N_a + N_b = N_c + (N_{tr})_{a+b}, \tag{4.49}$$

unde $(N_{tr})_{a+b}$ reprezintă numărul unităţilor de transport ce apar la adunarea celor două numere.

Trecînd de la relaţia pentru numerele unităţilor binare la relaţia pentru resturile lor după modul, se obţine:

$$(N_a + N_b) \bmod d = [N_c + (N_{tr})_{a+b}] \bmod d, \tag{4.50}$$

2.Operaţia de înmulţire. Dacă notăm cu a şi b cei doi factori, cu c rezultatul înmulţirii, iar cu N_a, N_b şi N_c numerele de unităţi binare corespunzătoare deînmulţitului, înmulţitorului şi produsului, atunci este adevărată următoarea relaţie:

$$N_a * N_b = N_c + \sum_{j=1}^{N_b} (N_{tr})_{(a*b)_j}, \tag{4.51}$$

unde $\Sigma_{j=1}^{N_b}(N_{tr})_{(a*b)_j}$ este suma unităţilor binare de transport care apar prin adunarea parţială, din procesul de înmulţire a numerelor a şi b în j tacturi, unde $j=1,2,\ldots,N_b$ este numărul de unităţi din codul înmulţitorului.

Astfel se poate formula următoarea regulă:

Produsul numărului de unităţi binare în codurile factorilor este egal cu suma dintre numărul unităţilor în codul produsului şi numărul comun de unităţi de transport care apar în procesul de înmulţire, prin adunare parţială.

Trecînd de la relaţia pentru numerele unităţilor binare la relaţia pentru resturile lor după modul, se obţine:

$$(N_a * N_b) \bmod d = [N_c + \sum_{j=1}^{N_b} (N_{tr})_{(a*b)_j}] \bmod d, \tag{4.52}$$

3.Operaţia de împărţire. Dacă notăm cu X deîmpărţitul, cu Y împărţitorul, cu Z cîtul şi cu W restul, atunci se poate scrie:

$$Z = \frac{X - W}{Y} \quad \text{sau} \quad X = Y * Z + W. \tag{4.53}$$

Prin urmare, operaţia de împărţire se poate reprezenta ca fiind compusă din operaţiile de înmulţire şi adunare. Aplicînd regula de control pentru adunare (4.49) termenilor Y, Z şi W se obţine:

$$N_{yz}+N_{\dot{q}}=N_{x}+(N_{tr})_{yz+w} \qquad (4.54)$$

unde $(N_{tr})_{yz+w}$ reprezintă numărul de unităţi binare de transport, care apar la adunarea numerelor Y*Z şi W, N_x este numărul de unităţi în codul deîmpărţitului, N_w este numărul de unităţi în codul restului şi N_{yz} este numărul de unităţi în codul produsului Y*Z.

În cotinuare, pentru determinarea numărului de unităţi N_{yz} se aplică relaţia (4.51):

$$N_{\dot{y}}+N_{z}=N_{yz}+\sum_{j=1}^{N}(N_{tr})_{(yz)_j}^{z} \qquad (4.55)$$

de unde rezultă:

$$N_{yz}=N_{\dot{y}}+N_{z}-\sum_{j=1}^{N}(N_{tr})_{(yz)_j}^{z} \qquad (4.56)$$

Înlocuind această valoare în (4.54) se obţine în final:

$$N_{\dot{y}}+N_{z}+N_{w}=N_{x}+[\sum_{j=1}^{N}(N_{tr})_{(yz)_j}^{z}+(N_{tr})_{yz+w}]. \qquad (4.57)$$

Astfel, se poate enunţa următoarea regulă de control pentru operaţia de împărţire:

Suma dintre produsul numerelor de unităţi binare ale împărţitorului şi cîtului şi numărul de unităţi binare ale restului este egală cu suma dintre numărul de unităţi în codul deîmpărţitului şi toate unităţile de transport care apar la efectuarea operaţiilor de înmulţire dintre împărţitor şi cît şi adunare la rest.

Din relaţia (4.57) se obţine relaţia de interdependenţă a resturilor după modul, sub forma:

$$(N_{\dot{y}}+N_{z}+N_{w})\bmod d\equiv[N_{x}+\sum_{j=1}^{N}(N_{tr})_{(yz)_j}^{z}+(N_{tr})_{yz+w}]\bmod d. \qquad (4.58)$$

Pentru efectuarea controlului operaţiei de împărţire cu ajutorul relaţiei (4.57) trebuie parcurse următoarele etape:

-determinarea numărului de unităţi binare în codurile deîmpărţitului, împărţitorului, cîtului şi restului N_x, N_y, N_z şi N_w;

-efectuarea operaţiei de înmulţire a împărţitorului cu cîtul Y*Z, determinîndu-se prin aceasta numărul tuturor unităţilor de transport

$$\sum_{k=1}^{N} (N_{tr})_{(w*)_k};$$

-adunarea codului produsului Y,Z cu codul restului W şi determinarea numărului unităţilor de transport $(N_{tr})_{yz+w}$;

-adunarea tuturor unităţilor de transport;

-efectuarea operaţiei de multiplicare $N_x * N_y$;

-încercarea îndeplinirii egalităţii (4.57).

După cum se vede din relaţia (4.57) controlul operaţiei de împărţire prin coduri cu resturi se prezintă sub o formă foarte complicată, deoarece este necesar să se îndeplinească două operaţii (adunare şi înmulţire) asupra numerelor.

Rezultatul (Y*Z+W) obţinut în procesul de control reprezintă în sine deîmpărţitul X. Prin urmare, pentru simplitate nu mai este necesară operaţia suplimentară de calcul a unităţilor de transport corespunzător cu (4.57), ci se compară între ele valorile Y*Z+W şi X după (4.49).

Regulile pentru controlul operaţiilor aritmetice prezentate mai sus sînt corecte dacă numerele se consideră în valori absolute, fără semn. În cazul reprezentării numerelor cu semn, într-o manieră asemănătoare se dezvoltă relaţiile de control în funcţie de tipul reprezentării.

Să considerăm spre exemplu operaţia de adunare. Pentru început, se prezintă dependenţa între caracteristica de control a numărului negativ reprezentat în mărime şi semn şi caracteristica corespunzătoare reprezentării în complement faţă de 1. Notăm caracteristica de control a numărului negativ în complement faţă de 1 cu F.

Dacă în reprezentarea prin mărime şi semn a numărului negativ A (care conţine bitul de semn) există k unităţi binare şi n-k zerouri, atunci în reprezentarea prin complement faţă de

1 se vor găsi k-1 zerouri şi (n-k)+1 unităţi, deoarece bitul de semn nu se inversează.

Caracteristica de control a codului în reprezentarea în mărime şi semn (care însoţeşte numărul la transmisie şi memorare) este egală cu:

$$Q_s \equiv (d-1-k \bmod d) \bmod d. \tag{4.59}$$

În mod analog se determină caracteristica de control a codului în reprezentarea prin complement faţă de 1:

$$\begin{aligned} F_s &\equiv [d-1-(n-k+1) \bmod d] \bmod d \equiv \\ &\equiv [(d-n-2) \bmod d + k \bmod d] \bmod d. \end{aligned} \tag{4.60}$$

Exprimînd din (4.59) pe k mod d în funcţie de Q_s

$$k \bmod d \equiv (d-1-Q_s) \bmod d, \tag{4.61}$$

şi înlocuind în (4.60) se obţine:

$$\begin{aligned} F_s &\equiv [(d-n-2) \bmod d + (d-1-Q_s) \bmod d] \bmod d \equiv \\ &\equiv (d-3-n-Q_s) \bmod d. \end{aligned} \tag{4.62}$$

Dacă notăm (d-3-n)=k, (valoare constantă pentru un sistem de calcul, avînd mărimea cuvîntului de n biţi şi modulul de control d fixat), în final se obţine următoarea dependenţă între caracteristica de control a codului în reprezentare prin număr şi semn şi caracteristica de control a codului în reprezentarea prin complement faţă de 1:

$$F \equiv (K-Q) \bmod d. \tag{4.63}$$

În concluzie, avînd dat un număr A reprezentat prin mărime şi semn, se determină mai întîi caracteristica de control după modulul dat d, cu relaţia (4.59). În continuare, în funcţie de mărimea cuvîntului n se determină caracteristica F_s după relaţia (4.62). Pentru numărul Ă reprezentat în complement faţă de 1 se determină F'_s după relaţia (4.60), care trebuie să coincidă cu F_s.

În felul acesta determinarea caracteristicii de control F_s a codului Ă, în reprezentarea prin complement faţă de 1 a numărului A în funcţie de caracteristica de control Q_s a codului în reprezentarea prin număr şi semn este doar o simplă corecţie

adună lui Q. (relaţia (4.63)).

Aceeaşi problemă se pune în continuare de a găsi dependenţa între caracteristica de control Q în reprezentarea prin număr şi semn şi caracteristica de control D în reprezentarea prin complement faţă de 2 a unui număr negativ. Codul în complement faţă de 2 A* a numărului negativ se obţine din Ã prin adăugarea unei unităţi la bitul cel mai puţin semnificativ. Prin aceasta, în caz general, poate să apară un transport şi ca urmare determinarea caracteristicii de control a codului în complement faţă de 2, din caracteristica de control Q a codului In reprezentarea prin număr şi semn, se face mult mai complicat decît în cazul codului în complement faţă de 1 (relaţia (4.62)), deoarece trebuie calculat numărul unităţilor de transport. Din acest motiv, pentru controlul prin cod a operaţiei de adunare şi scădere este mai convenabilă reprezentarea numerelor prin complement faţă de 1. Similar se procedează şi în celelalte cazuri.

S-a evidenţiat în acest capitol posibilitatea de control a operaţiilor logice şi aritmetice pe baza codurilor cu resturi. A rezultat în general că pentru controlul fiecărui tip de operaţie este necesar de a efectua un număr mare de operaţii suplimentare, ceea ce complică foarte mult structura dispozitivului de control.

Unul din elementele esenţiale ale structurii redundante implicate de utilizarea acestor coduri, indiferent de operaţia aritmetică sau logică executată, îl constituie schema de formare a restului. O analiză detaliată, orientată înspre codul generat pe baza modulului 3, a modalităţilor de sinteză a schemelor de formare a restului precum şi a capacităţii de detecţie a defecţiunilor la coduri cu resturi a fost întreprinsă de autor, în colaborare, în [45], [46] şi [47]. Complexitatea schemelor de formare a resturilor devine deosebit de ridicată, în situaţia sistemelor numerice complexe actuale. În plus, utilizarea codurilor cu resturi pentru controlul operaţiilor aritmetice şi logice presupune prezenţa unor dispozitive UAL suplimentare, de complexitate ridicată. În ansamblu, redundanţa mare implicată de utilizarea codurilor cu resturi a determinat o limitare severă în aplicarea acestei metode de control.

CAPITOLUL 5

ORGANIZAREA FLUXULUI INFORMAŢIONAL PENTRU CONTROLUL OPERAŢIILOR DE PROCESARE

În prezentul capitol se iniţiază o analiză originală a posibilităţii de a defini fluxul de informaţie în vederea controlului operaţiilor de procesare, problematică iniţiată de autor în [101].

Sistemele destinate controlului echipamentelor numerice trebuie să pună în evidenţă toate tipurile de defecţiuni atît defecţiunile de tip hard cît şi cele de tip soft. Prin urmare, apare necesar un control exhaustiv al tuturor blocurilor echipamentului, soluţie care implică un preţ de cost ridicat. Deoarece unele blocuri utilizează verigi comune de informaţie, pentru asigurarea unui control neîntrerupt nu este necesar de a controla fiecare bloc în parte.

Într-un sistem numeric, prin execuţia instrucţiunilor unui program, informaţia poate fi departajată într-o serie de trasee autonome neîntrerupte. Un traseu informaţional este numit neîntrerupt dacă între toate verigile acestuia există o legătură fermă de informaţie în toate etapele programului. În aceste trasee informaţia de bază şi cea redundantă, prin îndeplinirea fiecărei instrucţiuni, fie că rămîne constantă, fie că se modifică după reguli bine determinate. Autonomia traseului informaţional constă în aceea că circulaţia informaţiei de bază şi prin urmare şi a informaţiei redundante nu depinde de starea informaţiei din alte trasee.

În structura tipică a unui sistem de calcul circulaţia informaţiei poate fi departajată în următoarele trasee autonome neîntrerupte:

-traseul de date;
-traseul de comenzi;
-traseul de adrese;
-traseul codului operaţiei;
-traseul de comandă a operaţiei;
-traseul de comandă a adreselor microprogramului;

-traseul de comandă a mersului programului.

În figura 5.1 s-a prezentat traseul informaţional al datelor.

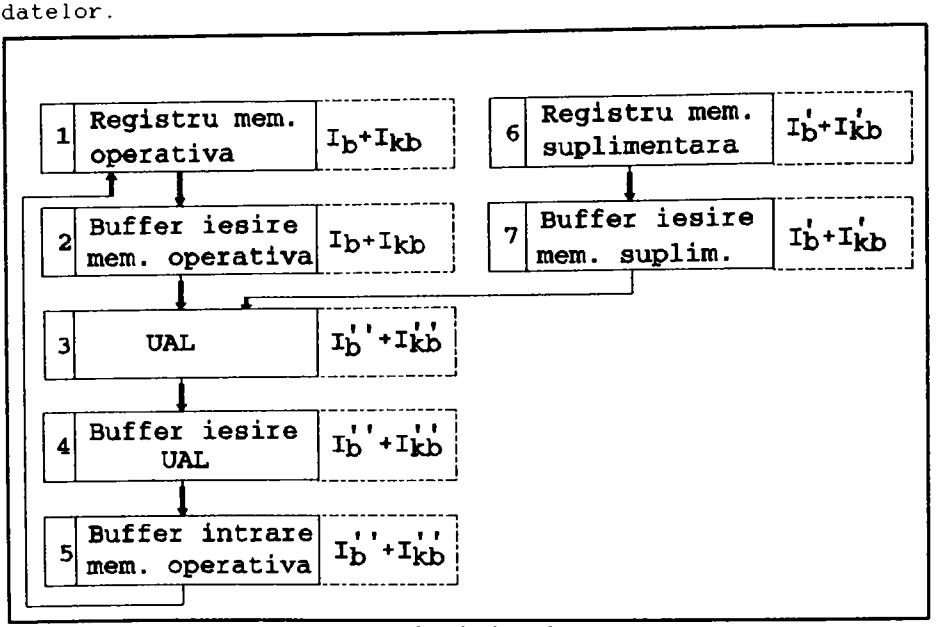

Fig.5.1.Traseul informaţional al datelor

Pe figură s-a specificat şi posibilitatea organizării controlului, arătîndu-se informaţia de bază şi redundantă utilizată pentru control (în dreptunghi punctat)..

În figura 5.2 se arată traseul informaţional al unităţii de comandă. Controlul corectitudinii citirii codului operaţiei se poate realiza cu ajutorul informaţiei redundante I_k .

În figura 5.3 este reprezentat traseul informaţional al generării adreselor pentru date şi instrucţiuni şi modalitatea de control pe baza informaţiei de redundante.

Controlul traseului codului operaţiei (blocurile 1-3 din fig. 5.2) se bazează pe concatenarea operaţiilor în n grupe specifice şi utilizarea informaţiei de control I_k , înscrisă în celule suplimentare ale memoriei şi registrului de instrucţiuni.

Controlul traseului de comandă a operaţiei se realizează cu ajutorul concatenării semnalelor de comandă în n grupe şi compararea codului obţinut I'_k sau I''_k cu informaţia redundantă înscrisă în rangurile suplimentare ale registrului de

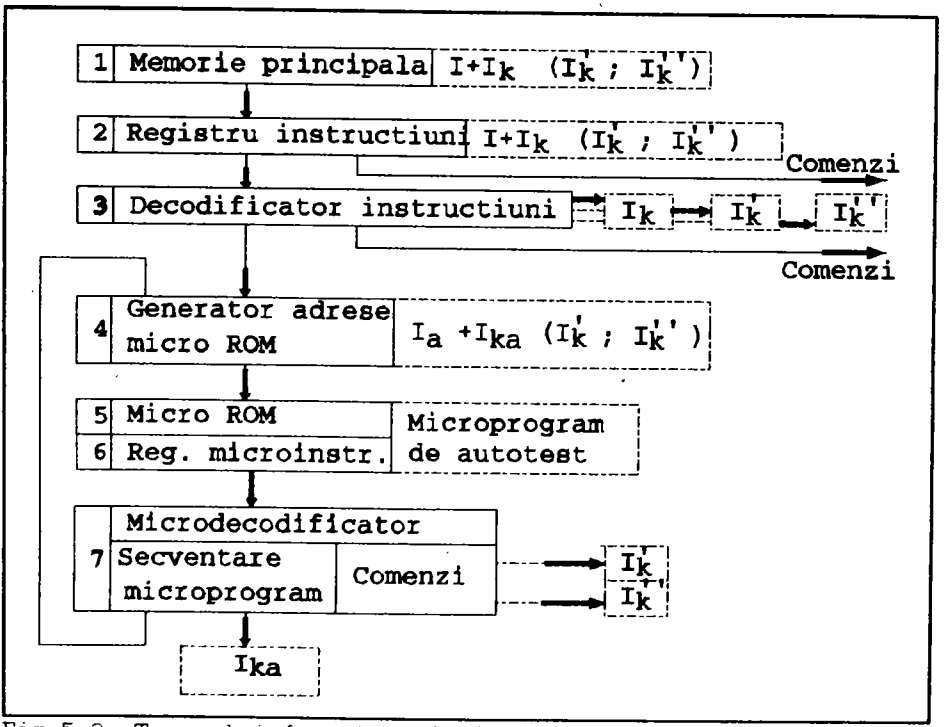

Fig.5.2. Traseul informational al unitatii de comanda

instrucţiuni.

Corectitudinea selecţiei comenzilor în execuţia unei operaţii (blocurile 4-7 din fig. 5.2), adică transmiterea de comandă a adreselor microprogramului se realizează prin verificarea informaţiei redundante I_{rr}.

În figura 5.4 se arată posibilitatea de a controla mersul programului prin divizarea operaţiilor calculatorului în mai multe grupe. În celule suplimentare de memorie se înscrie odată cu codul celei de a n-a operaţii, informaţia de control $(I_r)_{n+1}$ sau $(I')_{n+1}$ şi $(I'')_{n+1}$ ce caracterizează comenzile pentru execuţia celui de al (n+1) lea pas de program.

După cum se observă din figurile prezentate (5.1-5.4) controlul traseelor informaţionale poate fi organizat în toate blocurile traseului, sau în numai câteva verigi din el.

Este important de specificat că pentru controlul anumitor blocuri nu este necesară introducerea informaţiei redundante. Astfel, în figura 5.4 în numărătorul de program (blocul 1) lipseşte informaţia de control, realizată cu un control

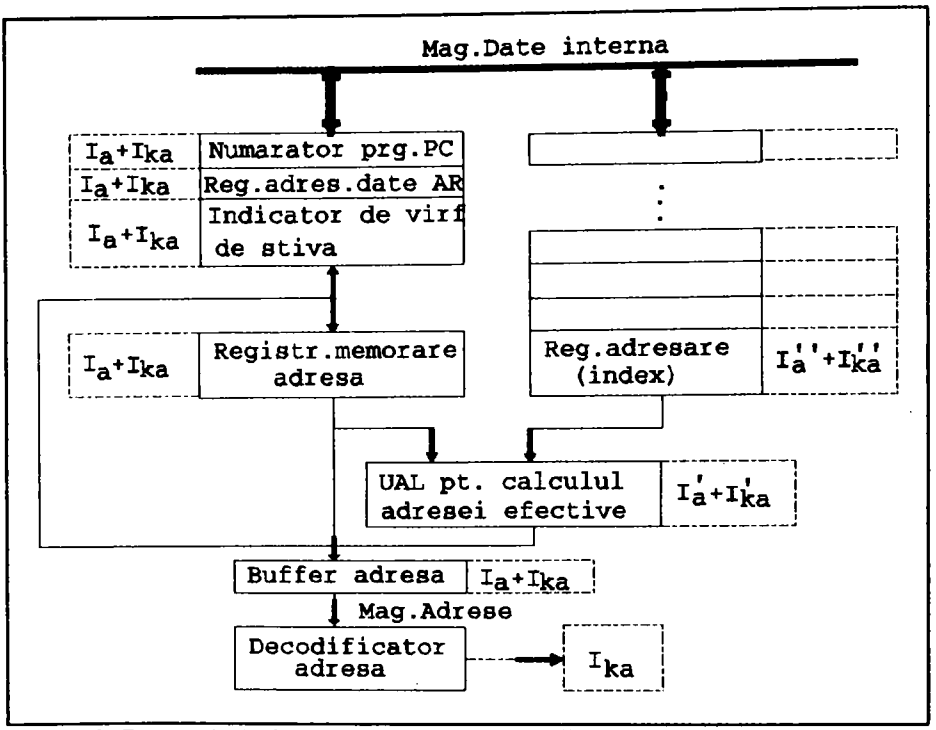

Fig.5.3.Traseul informational al generării adreselor

indirect prin mersul programului şi existenţa elementelor hardware de circuit.

Unele dispozitive sau elemente de structură au anumite verigi comune ale traseului de informaţie. Natural, controlul unor astfel de blocuri poate fi organizat în mod diferit în limitele fiecărui traseu informaţional în parte. Alegerea celei mai bune variante de control, în acest caz, se face ţinînd cont de necesităţile de a detecta şi eventual localiza anumite tipuri de defecţiuni.

Un fapt evident, care trebuie menţionat, este acela că într-o serie de trasee informaţia nu se prezintă neîntrerupt, în sens literal, ci se face referire la existenţa unei informaţii consistente de bază (prin urmare poate exista şi informaţie redundantă) pentru toate buclele de circuit. De exemplu, în traseul de comandă al mersului programului (fig. 5.4) în blocurile 1-2 la al n-lea pas de program se găseşte informaţia de bază (I_{a}), care nu se mai transmite mai departe. In blocul 4

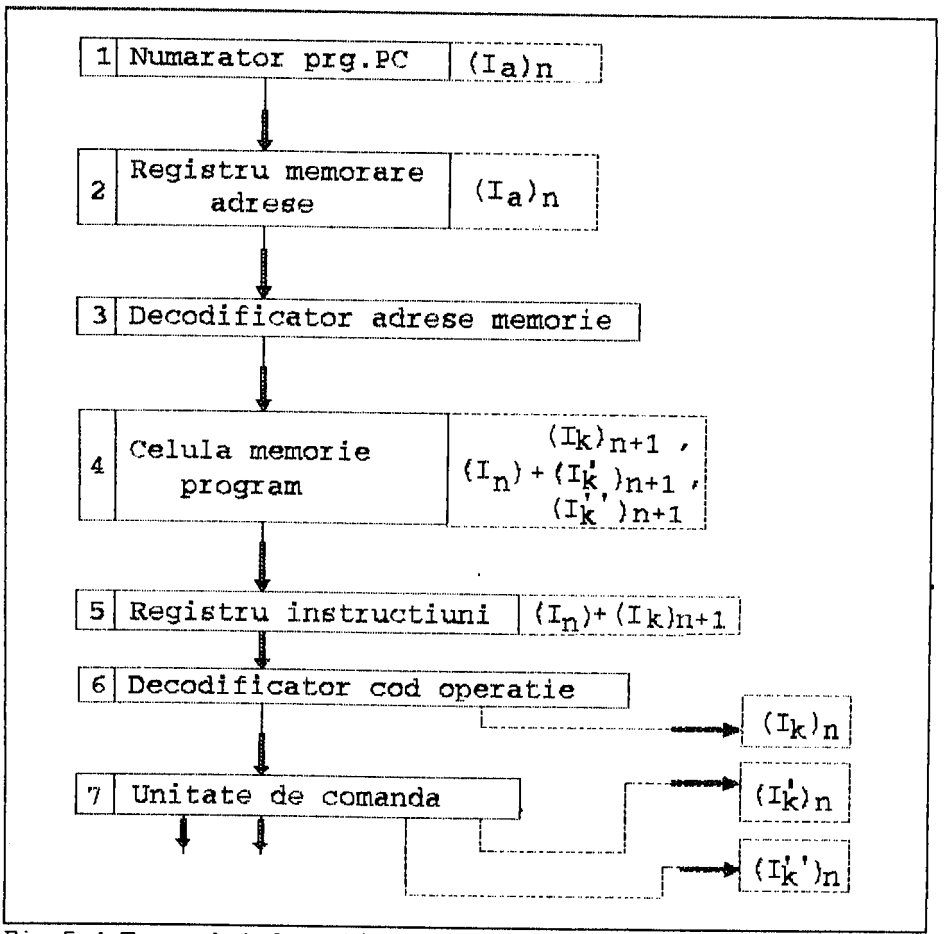

Fig.5.4.Traseul informaţional al derulării programului

se aplică o nouă informaţie de bază $(I)_n$ corespunzătoare informaţiei de bază $(I_a)_n$ din blocurile 1-2. În acelaşi timp, în blocul 4 se ataşează informaţia de control $(I_k)_{n+1}$, care caracterizează comenzile pentru execuţia pasului n+1 al programului. În continuare informaţia de bază $(I)_n$ se converteşte în celelalte blocuri în semnale de comandă pentru îndeplinirea
operaţiilor. Pe baza analizei informaţiei de ieşire a blocului 6 (sau a blocului 7) dispozitivul de control formează informaţia de testare $(I'_k)^{test}_n$ (sau $(I'_k)^{test}$, $(I''_k)^{test}_n$)
care se compară cu informaţia de control $(I_k)_n$ ($(I'_k)_n$ $(I''_k)_n$) ce apare în blocul 4 la pasul n-1. Prin prezenţa unei defecţiuni la

numărătorul de program la al n-lea pas nu se va mai transmite comanda n-1, ci o alta oarecare şi în caz general, în decodificatorul codului operaţiei se va alege o altă cale de operare asupra datelor (blocurile 3-7, fig.5.2). În consecinţă dispozitivul de testare va furniza o informaţie diferită de cea înscrisă în celulele biţilor suplimentari ai codului celei de a n-a operaţie. Prin urmare blocurile 1-7 pot fi privite efectiv ca un traseu de informaţie unitară, cu toate că informaţia de bază din blocurile 1-2, 4-5, 7 diferă total din punct de vedere logic, fiind adresa codului operaţiei, codul operaţiei şi respectiv semnalele de comandă a execuţiei operaţiei.

Controlul informaţiei într-un sistem numeric poate fi organizat în două moduri: concentrat şi distribuit. Un control concentrat presupune realizarea controlului pentru fiecare traseu informaţional (sau numai într-o parte a traseului). De exemplu, se poate controla informaţia numai în unele trasee de date. Evident că între informaţia de bază şi cea redundantă există o dependenţă mutuală, controlul efectuîndu-se în anumite puncte, numite de control, o singură dată. În cazul controlului distribuit, informaţia se testează de mai multe ori, corespunzător punctelor de control dintr-un traseu informaţional.

Noţiunea de punct de control presupune specificarea nu numai a locului fizic din circuit ci şi a momentului cînd se realizează acest control. De exemplu, în porţiunea de traseu din fig.5.1 corespunzătoare registrului memoriei operative, efectuarea controlului trebuie specificată în funcţie de momentul de citire sau scriere a informaţiei, cu alte cuvinte înainte sau după prelucrarea informaţiei în sistemul de calcul. În felul acesta se face distincţie între testarea unor circuite diferite.

CAPITOLUL 6

CONTROLUL OPERAŢIILOR DE PROCESARE PRIN SEMNĂTURI

După cum s-a scos în evidenţă în capitolul 5, dacă se reuşeşte asigurarea unui flux informaţional de control unitar, în structura sistemului numeric, atunci este posibilă verificarea tuturor blocurilor prin care circulă şi se prelucrează informaţia de bază, chiar dacă traseul acesteia este discontinuu. Mai mult, dacă verificarea corectei funcţionări a sistemului ar putea fi făcută concurent cu funcţionarea acestuia s-ar putea acoperi şi defecţiunule de tip tranzitoriu, sau intermitent, preponderente ca frecvenţă de manifestare faţă de cele permanente. Remarcăm în literatură un interes deosebit legat de testarea concurentă şi de găsirea de noi metode de control [48],[49],[50],[93],[95], [113],[114],[115],[116], soluţiile propuse abordînd doar parţial cele două aspecte specificate anterior.

În prezentul capitol se propune o generalizare originală a abordării controlului sistemelor numerice, astfel încît să fie îndeplinit atît deziderатul legat de unicitatea fluxului informaţiei de control, cît şi cel legat de concurenţă. Metoda propusă, abordată de autor în [102], are în vedere asigurarea controlului concurent al tuturor operaţiilor de procesare dintr-un sistem numeric, prin aceasta înţelegînd controlul unitar atît al operaţiilor aritmetice şi logice, cît şi al celor de transfer şi memorare.

6.1. CONTROLUL OPERAŢIILOR LOGICE PRIN SEMNĂTURI

Considerînd pentru început relaţiile de control după modul a operaţiilor logice pentru număr, (4.23)-(4.25), respectiv pentru cifră, (4.39)-(4.41), se observă că acestea sînt echivalente în cazul în care codurile de control sînt binare, de lungime n şi operaţiile se fac în aritmetica modulo 2. Prin urmare, în vederea controlului, spre exemplu, a operaţiilor logice SAU respectiv ŞI rezultă:

$$(A \vee B) = A \oplus B \oplus (A \wedge B), \qquad (6.1)$$

$$(A \wedge B) = A \oplus B \oplus (A \vee B), \qquad (6.2)$$

relaţiile fiind echivalente.

Operaţia de adunare modulo 2 stînd la baza relaţiilor de control, operaţia logică SAU-EXCLUSIV, evident că se va verifica prin ea însăşi. În cazul în care se urmăreşte controlul printr-o procedură distinctă, ţinînd cont de (6.1), sau (6.2), se poate scrie:

$$(A \oplus B) = (A \wedge B) \oplus (A \vee B). \qquad (6.3)$$

În cazul operaţiilor logice care implică un singur operand, poate fi dedusă aceeaşi echivalenţă. Astfel, pentru controlul operaţiei de inversare rezultă:

$$\overline{A} = A \oplus \alpha_n, \qquad (6.4)$$

unde $\alpha_n = (11\ldots11)_n$ este cuvîntul avînd toţi cei n biţi la valoarea logică 1.

Controlul operaţiilor logice de deplasare spre stînga (dreapta) se reduce la verificarea parităţii. Astfel, pentru cazul deplasării spre stînga (dreapta), fără pierderea informaţiei, sînt adevărate relaţiile:

$$P(A^*) = P(A), \qquad (6.5)$$

$$P(A^*) = P(A). \qquad (6.6)$$

Pentru operaţiile logice de deplasare spre stînga (dreapta), cu pierderea informaţiei, rezultă în mod evident:

$$P(A^*) = P(A) \oplus a_{n-1}, \qquad (6.7)$$

$$P(A^*) = P(A) \oplus a_0. \qquad (6.8)$$

Relaţiile de control (6.5)-(6.8) pot fi generalizate pentru ambele situaţii, fără pierderea şi cu pierderea informaţiei, sub forma:

$$A^{\leftarrow}(A_p^{\leftarrow}) = A \oplus H(A^{\leftarrow}), \qquad (6.9)$$

$$A^{\rightarrow}(A_p^{\rightarrow}) = A \oplus H(A^{\rightarrow}), \qquad (6.10)$$

unde $H(A^{\leftarrow})$ şi respectiv $H(A^{\rightarrow})$ sînt cuvintele de lungime n, de control a parităţii. Rangurile binare ale acestora rezultă în felul următor:

$$h_i(A^{\leftarrow}) = (a_{i-1} \oplus a_i),\ i=2,\ldots,n,\ h_1(A^{\leftarrow}) = a_1, \qquad (6.11)$$

$$h_i(A^{\rightarrow}) = (a_i \oplus a_{i+1}),\ i=1,\ldots,n-1,\ h_n(A^{\rightarrow}) = a_n. \qquad (6.12)$$

Operaţiile logice de rotaţie spre stînga (dreapta) nu conduc la pierderea informaţiei şi prin urmare paritatea rămîne neschimbată. În consecinţă rezultă:

$$A^{\leftarrow} = A \oplus H(A^{\leftarrow}), \qquad (6.13)$$

$$A^{\rightarrow} = A \oplus H(A^{\rightarrow}), \qquad (6.14)$$

unde $H(A^{\leftarrow})$, respectiv $H(A^{\rightarrow})$ sînt cuvintele de control a parităţii pentru operaţiile de rotaţie spre stînga, respectiv spre dreapta. Rangurile binare ale acestora rezultă astfel:

$$h_i(A^{\leftarrow}) = (a_{i-1} \oplus a_i),\ 1=\overline{2,n},\ h_1(A^{\leftarrow}) = a_1 \oplus a_n, \qquad (6.15)$$

$$h_i(A^{\rightarrow}) = (a_i \oplus a_{i+1}),\ i=\overline{1,n-1},\ h_n(A^{\rightarrow}) = a_n \oplus a_1. \qquad (6.16)$$

În ceea ce priveşte operaţia logică de coincidenţă, controlul acesteia se va face, ţinînd cont de (6.4) prin relaţia:

$$(A \odot B) = A \oplus B \oplus c_n. \qquad (6.17)$$

Avînd în vedere relaţiile de control prezentate, propunem următoarea teoremă:

Teorema 6.1. Fie A şi B două valori binare de lungime n. Pentru orice operaţie logică $\diamond$, care implică pe A şi B, există o funcţie binară H(A,B) şi o operaţie logică $\circ$, prin intermediul carora poate fi executat controlul operatiei , dupa relatia

$$(A \diamond B) = A \circ B \circ H(A,B). \qquad (6.18)$$

Demonstraţie: $A \diamond B$ fiind o funcţie booleană, ea poate fi reprezentată printr-un polinom Zhegalkin, sau polinom mod 2, [91], în variabilele A şi B. Astfel, pentru fiecare funcţie logică există un set complet de reprezentări, printre care există şi forma:

$$(A \diamond B) = A \oplus B \oplus H(A,B). \qquad (6.19)$$

unde H(A,B) este o funcţie binară oarecare. Apare, prin urmare, că operaţia logică $\circ$, care stă la baza controlului este adunarea modulo 2, iar funcţia binară H(A,B) are sens prin relaţia:

$$H(A,B) = (A \diamond B) \oplus A \oplus B. \qquad (6.20)$$

H(A,B) se va numi funcţie de control a operaţiei logice $\diamond$.

Relaţia de control generală (6.18), respectiv (6.19), în forma ei concretizată, este valabilă, prin particularizare, şi pentru cazul operaţiilor logice ce implică un singur operand.

Relaţia de control (6.19) este deosebit de avantajoasă pentru dezvoltarea ulterioară pe bază de coduri, prin faptul că la efectuarea controlului intervin operanzii A şi B.

Singura problemă care mai trebuie specificată este cea a determinării funcţiilor logice de control H(A,B). Forma acestor funcţii rezultă din relaţia (6.20), precum şi din examinarea relaţiilor (6.1),(6.2),(6.4),(6.11),(6.12),(6.15),(6.16) şi (6.17), pentru toate operaţiile logice considerate.

Pe baza celor prezentate, rezultă ca atractivă soluţia propusă de a generaliza printr-o singură metodă controlul operaţiilor de procesare dintr-un sistem numeric, aceasta însemnînd controlul atît al operaţiilor din circuitele "pasive", de transfer şi memorare, cît şi al celor din circuitele "active", care determină modificarea informaţiei prin operaţii aritmetice şi/sau logice.

Cu referire la operaţiile logice, propunem teorema:

Teorema 6.2. Fie G(x) un polinom generator de grad k al unui cod ciclic şi fie R(A), R(B), $R(A \diamond B)$ şi R[H(A,B)] informaţia redundantă de control în cod ciclic a operanzilor A şi B, a rezultatului operaţiei logice $A \diamond B$ şi respectiv a

funcţiei de control H(A,B). În aceste condiţii este valabilă dependenţa:

$$R(A \diamond B) = R(A) \oplus R(B) \oplus R[H(A,B)]. \qquad (6.21)$$

Demonstraţie: Într-adevăr, dacă se codifică în cod ciclic operanzii A şi B, rezultatul operaţiei logice (A◇B) şi funcţia de control H(A,B) în baza relaţiei (6.19), rezultă:

$$x^k(A \diamond B) \oplus R(A \diamond B) = x^k A \oplus R(A) \oplus x^k B \oplus R(B) \oplus x^k H(A,B) \oplus R[H(A,B)],$$

ceea ce conduce la valabilitatea relaţiei (6.21).

Informaţia redundantă de control prin cod ciclic mai poartă numele de semnătură şi prin urmare, utilizînd notaţia sig, relaţia (6.21) se mai poate scrie:

$$sig(A \diamond B) = sigA \oplus sigB \oplus sigH(A,B). \qquad (6.22)$$

Un fapt remarcabil este acela că relaţia (6.22) este valabilă atît pentru semnăturile care se formează în serie cît şi în paralel. În cazul semnăturilor de tip paralel, în registrele de formare a semnăturii valorile operanzilor se introduc în paralel, în ordinea în care apar la execuţia operaţiilor. Semnăturile rezultatelor, în acest caz, se vor forma tot în paralel, în mod continuu. Verificarea îndeplinirii relaţiei (6.22) se va face pentru fiecare operaţie în parte, după fiecare tact de încărcare, nefiind necesară iniţializarea registrelor de formare a semnăturilor pentru fiecare operaţie, ca în cazul semnăturii formate serial.

Efectuînd controlul operaţiilor logice prin semnături, avînd la bază semnăturile operanzilor A şi B, se asigură posibilitatea de a controla prin aceeaşi metodă şi operaţiile de transfer şi memorare. În felul acesta, metoda propusă împlineşte dezideratul de a realiza unicitatea controlului operaţiilor de procesare. Astfel, fiecărui cuvînt de informaţie îi va fi ataşată, în permanenţă, o semnătură bine determinată de control, indiferent de transformările care au loc asupra informaţiei utile, în urma oricăror operaţii de procesare.

Particularizînd relatia (6.22) pentru cazul operatiilor logice considerate, rezulta relatiile de control prin semnaturi. Astfel, pentru opera iile logice SAU, respectiv I rezult :

$$sig(A \vee B) = sigA \oplus sigB \oplus sig(A \wedge B), \qquad (6.23)$$

$$sig(A \wedge B) = sigA \oplus sigB \oplus sig(A \vee B). \qquad (6.24)$$

Pentru operaţiile SAU-EXCLUSIV şi COINCIDENŢĂ sînt valabile relaţiile de control:

$$sig(A \oplus B) = sigA \oplus sigB, \qquad (6.25)$$

$$sig(A \odot B) = sigA \oplus sigB \oplus sig\alpha_n \qquad (6.26)$$

După cum se vede din (6.23)-(6.26) pentru controlul corectitudinii îndeplinirii oricăreia din operaţiile logice $A \vee B$, $A \wedge B$, $A \bullet B$, asupra operanzilor A şi B este necesar să se îndeplinească în mod corespunzător şi alte operaţii logice. Acestea, conform cu regula generală a controlului prin semnături (6.22) pot fi interpretate ca operaţii de formare a funcţiilor de control a celor doi operanzi, ca parte componentă a datelor de control pentru operaţiile logice $\diamond$.

De remarcat că pentru controlul operaţiilor de COINCIDENŢĂ şi SAU-EXCLUSIV funcţia de control (semnătura) degenerează într-o constantă. Aceasta arată generalitatea proprietăţilor de controlabilitate a operaţiilor. Coincidenţa şi suma modulo 2 a două variabile apar ca funcţii liniare şi funcţiile de control $H(A,B)=\alpha_n=(11\ldots11)_2$, pentru COINCIDENŢĂ, respectiv negata acesteia, $H(A,B)=(00\ldots00)_2$, se pot considera ca aparţinînd acestor operaţii. De aici rezultă importanţa practică de a organiza controlul prin semnături a operaţiei de transfer. Dacă funcţia logică de controlat este liniară, atunci funcţia de control poate fi oricare funcţie liniară cunoscută, cu valori corespunzătoare alese ale constantelor.

Operaţiile logice care implică un singur operand vor fi, de asemenea, controlate prin semnături. Pentru operaţia de deplasare la stînga (dreapta), cu şi fără pierderea informaţiei, rezultă următoarea regulă de control prin semnături:

$$sigA^{*} = sigA \oplus H(A^{*}), \qquad (6.27)$$

$$sigA^{\prime} = sigA \oplus sigH(A^{\prime}), \qquad (6.28)$$

unde $H(A^{-})$ respectiv $H(A)$ se determină pe baza relaţiilor (6.11) respectiv (6.12). Pentru realizarea controlului este necesar un dispozitiv combinaţional mod 2, care să sintetizeze pe $H(A^{-})$ respectiv $H(A^{-})$, avînd n intrări şi n-1 ieşiri.

În mod similar, rotaţia spre stînga (dreapta) va fi controlată prin:

$$sig\bar{A}^{-}=sigA\oplus sigH(\bar{A}^{-}), \qquad (6.29)$$

$$sig\bar{A}=sigA\oplus sigH(A), \qquad (6.30)$$

unde funcţiile de control $H(A^{-})$ respectiv $H(A)$ vor fi sintetizate printr-un dispozitiv combinaţional corespunzător, avînd la bază relaţia (6.15) respectiv (6.16).

În fine operaţia de INVERSIUNE va fi controlată prin relaţia:

$$sig\bar{A}=sigA\oplus sig_{1}. \qquad (6.31)$$

6.2. CONTROLUL OPERAŢIILOR ARITMETICE PRIN SEMNĂTURI

Abordarea problematicii controlului operaţiilor aritmetice prin semnături se face într-o manieră similară cu cea de la controlul operaţiilor logice.

Astfel, avînd în vedere forma generală (6.5) a controlului prin semnături a operaţiilor logice, se poate reformula pentru operaţiile aritmetice următoarea afirmaţie: dacă pentru operanzii A şi B semnăturile acestora sînt sub forma sig A şi sig B, atunci există o caracteristică mutuală de control oarecare, corespunzătoare acestor operanzi, avînd semnătura sig $H(A*B)$.

Problema constă în determinarea caracteristicii mutuale polinomiale $H(A*B)$ a operanzilor A şi B, corespunzătoare fiecărui tip de operaţie aritmetică.

Pentru cazul operaţiei de adunare se are în vedere validitatea egalităţii (4.49). Avînd la bază relaţia (4.49) şi ţinînd cont de particularităţile de îndeplinire a operaţiilor de adunare a numerelor cu semn în reprezentarea prin complement faţă de 1 şi prin complement faţă de 2, se poate formula

următoarea regulă:

Prin adunarea a două numere binare, suma numărului de unităţi în codurile termenilor adunării (ce conţin bitul de semn) este egală cu suma unităţilor în codul rezultatului şi a tuturor unităţilor care apar prin transport (inclusiv unităţile transportului secundar la adunarea în complement faţă de 1).

Este evident că această regulă este corectă în limitele utilizării unui singur cod. Astfel, dacă adunarea în UAL se realizează în reprezentarea prin complement faţă de 1 a numerelor negative, atunci este necesar de a opera cu caracteristica de control a acestui tip de reprezentare atît asupra termenilor, cît şi asupra unităţilor de transport şi a rezultatului operaţiei de adunare (dacă este negativ).

Notînd cu N_a, N_b, N_c numărul de unităţi în codul termenilor şi a rezultatului operaţiei (se ţine cont de bitul de semn), cu N_{tr} numărul de unităţi de transport în reprezentarea prin număr şi semn, iar cu $N^{c1}{}_a(N^{c2}{}_a)$, $N^{c1}{}_b(N^{c2}{}_b)$, $N^{c1}{}_c(N^{c2}{}_c)$ şi $N^{c1}{}_{tr}(N^{c2}{}_{tr})$ aceleaşi mărimi în reprezentarea prin complement faţă de 1 (complement faţă de 2), se enunţă următoarele reguli de bază:

$$N_a + N_b = N_c + N_{tr},$$

$$N^{c1}{}_a + N^{c1}{}_b = N^{c1}{}_c + N^{c1}{}_{tr},$$

$$N^{c2}{}_a + N^{c2}{}_b = N^{c2}{}_c + N^{c2}{}_{tr}.$$

Trecînd de la relaţiile pentru număr la relaţia de control prin semnături se obţine forma generală:

$$sig(A+B) = sigA \oplus sigB \oplus sigH(A+B), \qquad (6.32)$$

unde codul H(A+B) se va forma astfel încît să conţină unităţi binare în acele două poziţii în care provin unităţile de transport din adunarea celor două numere A şi B. Cu alte cuvinte, caracteristica mutuală polinomială va reprezenta cuvîntul de transport rezultat la adunarea numerelor.

Determinarea practică a caracteristicii mutuale H(A+B) a unor operanzi de n biţi se reduce la sinteza unui dispozitiv combinaţional a cărui funcţie de ieşire, pentru cazul reprezentării în complement faţă de 2, are forma:

$$h_{1/i>2}=(a_{i-1}\wedge b_{i-1})\bigvee_{j=1}^{i-2}[(a_j\wedge b_j)\bigwedge_{k=j+1}^{i-1}(a_k\vee b_k)],i=3,\ldots,n,h_1=0,h_2=a_1b_1. \quad (6.33)$$

În mod analog, pentru operaţia de scădere, relaţia de control prin semnături apare sub forma:

$$sig(A-B)=sigA\oplus sigB\oplus sigH(A-B), \quad (6.34)$$

unde caracteristica mutuală polinomială reprezintă cuvîntul de împrumut rezultat din operaţia A-B.

Dispozitivul combinaţional de generare a caracteristicii mutuale H(A-B) va avea următoarea funcţie de ieşire:

$$h_{i/i>2}=(\overline{a_{i-1}}\wedge b_{i-1})\bigvee_{j=1}^{i-2}[(\overline{a_j}\wedge b_j)\bigwedge_{k=j+1}^{i-1}(a_k\wedge b_k\vee\overline{a_k}\wedge\overline{b_k})],i=3,\ldots,n,h_1=0,h_2=\overline{a_1}b_1.$$

(6.45)

În dezvoltarea relaţiilor (6.33) şi (6.35) s-a considerat cazul reprezentării numerelor prin complementul faţă de 2. Spre deosebire de cazul controlului prin coduri cu resturi, în care era mai convenabilă reprezentarea prin complement faţă de 1, sinteza dispozitivului combinaţional de generare a caracteristicii mutuale H la controlul prin semnături, în afară de faptul că e mult mai simplă, nu implică complicaţii deosebite în funcţie de tipul reprezentării numerelor. Astfel, relaţii asemănătoare pentru caracteristica de control H pot fi dezvoltate fără dificultate şi pentru cazul reprezentării numerelor prin complement faţă de 1 sau prin mărime şi semn.

În ceea ce priveşte operaţia de înmulţire, se are în vedere ca punct de plecare validitatea egalităţii (4.51). Dacă exprimăm cu A(x), B(x) şi D(x) deînmulţitul, înmulţitorul şi respectiv produsul, într-o formă polinomială, obţinem următoarea formulă:

$$A(x)\cdot B(x)+T(x)=D(x), \quad (6.36)$$

respectiv următoarea relaţie de control prin semnaturi:

$$sig(A(x)\ B(x))\ sigT(x)\ sigH(x)\ sigD(x), \quad (6.37)$$

unde A(x)*B(x) reprezint multiplicarea polinomial , realizat

într-un dispozitiv de convoluţie corespunzător, T(x) este polinomul transportului ce intervine în adunarea parţială din procesul de înmulţire realizat în UAL, iar H(x) este caracteristica mutuală a operaţiei de adunare (6.36).

Un avantaj al utilizării relaţiei de control (6.36) pentru operaţia de înmulţire îl reprezintă faptul că determinarea polinomului T(x) nu implică accesul în UAL, cum ar apare la prima vedere, ci se poate face convenabil în dispozitivul de convoluţie.

Înmulţirea numerelor în UAL se pote face în codul reprezentării prin număr şi semn, prin complement faţă de 1 sau prin complement faţă de 2.

Dacă înmulţirea numerelor se realizează în reprezentarea prin număr şi semn, rezultatul este egal cu produsul mantiselor numerelor, iar semnul se determină adunînd semnele termenilor după modulul 2.

Notînd numerele prin A şi B, mantisele lor cu A' şi B' , iar prin C şi C' în mod corespunzător produsul numerelor şi al mantiselor, este adevărată relaţia:

$$N_{A'}*N_{B'}=N_{C'}+\sum_{j=1}^{N'} (N_{B'}) \quad (6.38)$$

Pe baza acestei relaţii rezultă regula de conversie a caracteristicii de control a produsului, ţinînd cont de toate cazurile posibile de combinare a semnelor numerelor:

(i). Ambele numere A şi B sînt pozitive, produsul C fiind de asemenea pozitiv, adică $a_n=0$, $b_n=0$ (n-lea bit de semn), $c_{2n-1}=0$. Prin urmare

$$N_A=N'_A \quad , \quad N_B=N'_B \quad , \quad N_C=N'_C .$$

(ii). Unul din factori este negativ, celălalt pozitiv, rezultatul negativ:

a). $a_n=1$, $b_n=0$, $c_{2n-1}=1$

$N_{A'}=N_A-1$, $N_{B'}=N_B$, $N_{C'}=N_C-1$.

b). $a_n=0$, $b_n=1$, $c_{2n-1}=1$, în mod analog cu cazul a).

(iii). Ambii factori sînt negativi, rezultatul fiind pozitiv:

$a_n=1$, $b_n=1$, $c_{2n-1}=0$,

$N_{a'}=N_a-1$, $N_{b'}=N_b-1$, $N_{c'}=N_c$.

Astfel, pentru controlul operaţiei de înmulţire realizată cu numere reprezentate prin mărime şi semn trebuie să se îndeplinească următoarele acţiuni succesive:

-se analizează semnul factorilor pentru concretizarea relaţiei (6.38) care stă la baza controlului;

-se determină numărul unităţilor de transport T(x) şi a caracteristicii aferente de control H(x);

-se încearcă îndeplinirea relaţiei de control necesară (6.37).

În continuare, în capitolul 8 se prezintă o modalitate de sinteză originală a unui bloc de control pentru operaţiile logice şi aritmetice tratate în acest capitol.

CAPITOLUL 7

ORGANIZAREA CONTROLULUI OPERAŢIILOR DE PROCESARE

Avînd stabilită metoda de verificare a operaţiilor de procesare, în baza celor prezentate anterior, propunem în continuare o variantă originală de structurare a unui sistem de control a operaţiilor de procesare.

Utilizarea sistemelor bazate pe microprocesoare de uz general a crescut foarte mult, simţindu-se nevoia tot mai imperioasă de aplicare şi în domenii cu siguranţă ridicată în funcţionare. Astfel de aplicaţii tipice ar consta în controlul de zbor în situaţii dinamice instabile, protecţia sistemelor în industria nucleară şi petrochimică, controlul traficului feroviar şi controlul echipamentelor în fabricile automatizate.

Proiectarea unor sisteme destinate acestor aplicaţii se confruntă cu o problemă, generată de faptul că microprocesoarele comerciale gîndite pentru produse de masă şi de larg consum, nu satisfac cerinţele unor aplicaţii de siguranţă ridicată. De asemenea, probleme apar în acest sens şi la nivelul plachetelor bazate pe asemenea microprocesoare, care oferă o bază inconsistentă pentru toleranţa la defecţiuni, iar acoperirea defecţiunilor pe baza rutinelor de test "built-in" este foarte nesigură.

Problema poate fi rezolvată pe două căi:

-Abordarea unei proiectări dedicate a unui sistem corespunzător tolerant la defecţiuni;

-Găsirea unei structuri tolerante la defecţiuni, avînd la bază configuraţia unui procesor de uz general.

Varietatea de soluţii propuse se referă în general la scheme de monitorizare on-line a operaţiilor de transfer şi de detecţie a erorilor procesorului.

În mod tradiţional, redundanţa masivă, cum ar fi duplicarea sau triplicarea şi codurile detectoare şi corectoare de erori sînt utilizate în controlul on-line. Redundanţa masivă reprezintă o cale foarte eficientă pentru detecţia erorilor tranzitorii, însă este foarte costisitoare şi, de asemenea, poate reduce fiabilitatea sistemului, cînd se aplică exhaustiv.

Soluţiile recente utilizează o combinaţie hardware/software în vederea monitorizării on-line a operaţiilor procesoarelor de uz general [48],[49],[93]. Alte soluţii oferă facilităţi complexe pentru testabilitatea la nivel de capsulă a procesoarelor VLSI [74],[94] sau pentru autoverificarea operaţiilor de procesare [95].

În continuare, se prezintă o soluţie originală, propusă pentru monitorizarea şi controlul operaţiilor de procesare. În figura 7.1 se prezintă schema bloc de organizare a controlului circulaţiei şi prelucrării informaţiei într-un procesor. Schema prezentată a avut în vedere posibilitatea realizării integrate, într-o structură VLSI, dar şi o realizare exterioară procesorului poate fi luată în considerare.

Principiul care stă la baza acestei soluţii îl reprezintă controlul informaţional pe bază de semnături. Circulaţia informaţiei şi prelucrarea ei este determinată de arhitectura procesorului, prin setul de instrucţiuni al acestuia.

Instrucţiunile unui procesor pot fi grupate în: transferuri de date, operaţii aritmetice şi logice, transfer al comenzii, controlul procesorului şi operaţii speciale. În aceste condiţii, fiecărui cuvînt de informaţie i se ataşează o semnătură de control. Această semnătură va fi utilizată pentru controlul operaţiilor de transfer atît în interiorul procesorului, cît şi în dialogul cu memoria externă. În ciclul de scriere, pe magistrala de date a procesorului se va furniza cuvîntul de date împreună cu semnătura lui. În blocul generator de semnătură SIGGEN se formează o nouă semnătură, corespunzătoare cuvîntului de date prezent pe magistrală, care se compară cu semnătura ataşată datelor în registrele procesorului. Evident că şi memoria externă va trebui să conţină un bloc auxiliar pentru biţii de control ai semnăturii. În ciclul de citire, cuvîntul pe magistrala de date va fi prezent alături de semnătura de control din memorie, care se va compara cu o nouă semnătură formată în SIGGEN. În felul acesta se verifică corectitudinea operaţiilor de transfer, atît între procesor şi memoria externă, cît şi în interiorul procesorului, între registre. Controlul este aplicabil şi instrucţiunilor, evident în condiţia ca fiecărui cuvînt instrucţiune să-i fie ataşată semnătura corespunzătoare în memoria de program. Redundanţa implicată de controlul

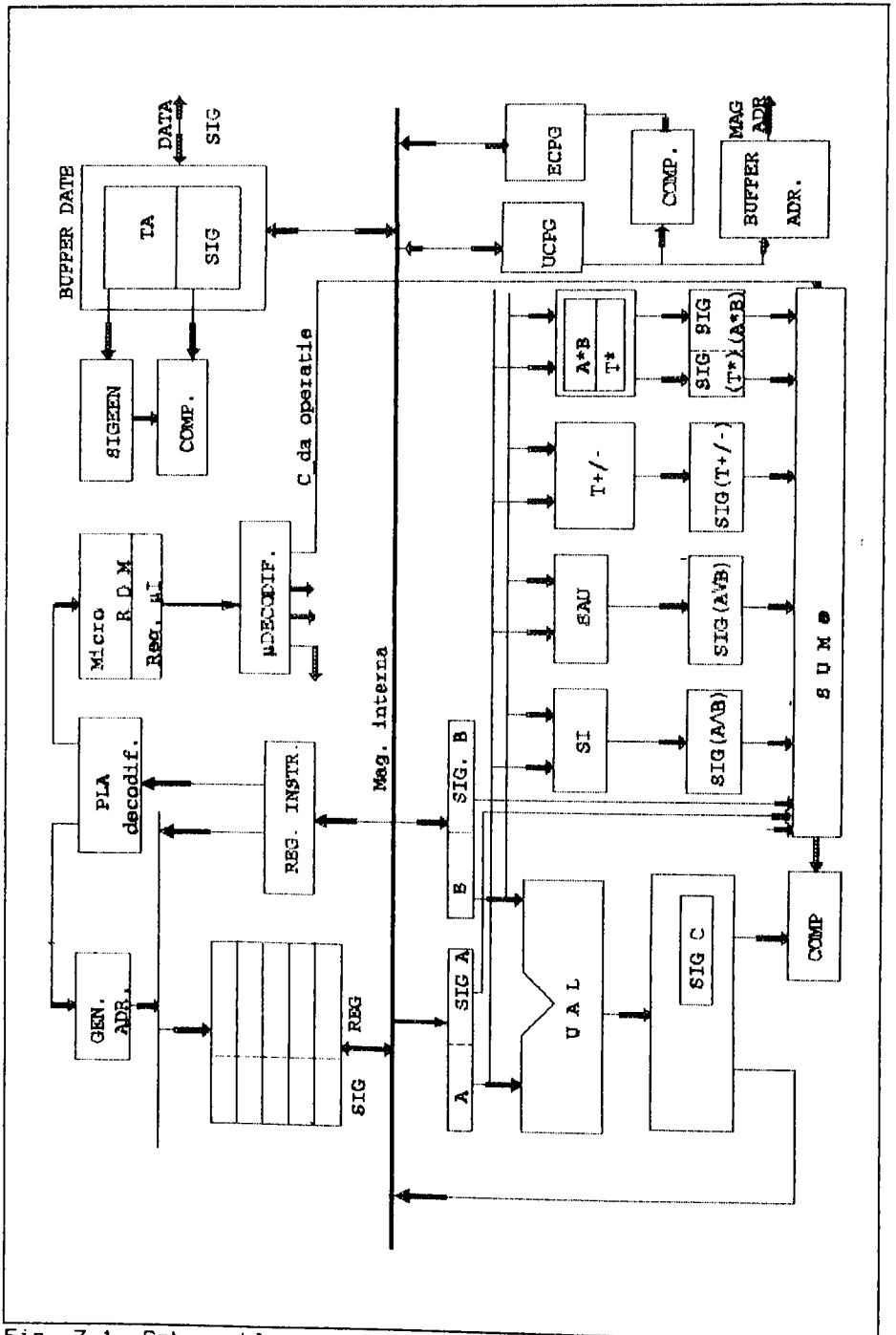

Fig. 7.1. Schema bloc a controlului operațiilor de procesare.

operaţiilor de transfer constă în numărul de biţi suplimentari necesari pentru semnătura de control, corespunzător fiecărui cuvînt de informaţie. În afară de blocul auxiliar al memoriei externe şi registrele interne ale procesorului vor trebui să aibă o lungime corespunzător mai mare.

O altă problemă care trebuie soluţionată constă în determinarea semnăturii de control pentru un cuvînt de date care se modifică ca urmare a prelucrării de către procesor, într-o operaţie aritmetică sau logică. Aceasta implică verificarea corectitudinii funcţionării UAL. Avînd în vedere cele prezentate în cap.6, controlul prin semnături a operaţiilor aritmetice şi logice necesită pe lîngă semnăturile operanzilor, care sînt deja disponibile, formarea caracteristicii mutuale de control şi a semnăturii acesteia pentru fiecare tip de operaţie în parte. În acest scop au fost prevăzute blocurile ŞI, SAU, T+ şi T*. Semnătura de control a operaţiei aritmetice sau logice rezultă la ieşirea sumatorului modulo 2, unde se compară cu semnătura formată în blocul SIGC, din cuvîntul rezultat la ieşirea UAL.

Soluţiei propuse, i se mai poate adăuga un emulator de comandă a programelor ECPG, care să genereze adrese corespunzătoare, în vederea comparaţiei, cu cele generate de unitatea de comandă a programelor UCPG. În acest fel se controlează on-line şi traseul informaţional de generare a adreselor de către procesor.

De asemenea, strategia de autotest se poate dezvolta convenabil pe baza considerentului că dacă microinstrucţiunile pot controla multe blocuri hardware, apare rezonabilă utilizarea lor în vederea autotestului [94]. Vectorii de test necesari sînt generaţi dintr-un ROM de test, rezultatele testelor fiind comprimate paralel într-un bloc de generare a semnăturii, conectat la magistrala internă.

CAPITOLUL 8

BLOCUL DE CONTROL AL OPERAȚIILOR ARITMETICE ȘI LOGICE

8.1 SINTEZA BLOCULUI DE CONTROL

Pornind de la rezultatele obținute în capitolul precedent sinteza blocului de control a operațiilor aritmetice și logice a fost orientată înspre posibilitatea integrării pe scară foarte largă. În acest sens, pentru a reduce aria de integrare la minim, toate modulele funcționale ale blocului de control au fost sintetizate sub formă de rețele logice regulate, având la bază porți ȘI-NU. Admițînd, spre exemplu o tehnologie NMOS, realizarea în continuare a integrării blocului de control se poate aborda cu ușurință.

Fără a reduce din generalitate, exemplificarea sintezei blocului de control a fost făcută pentru o lungime a cuvintelor operanzilor de 4 biți, pentru a facilita prezentarea metodei de control propuse. Evident, extinderea controlului la un număr mai mare de biți, pentru alte aplicații, nu va implica vreo problemă deosebită. Avînd în vedere organizarea de principiu a controlului operațiilor aritmetice și logice prezentată în fig. 7.1, schema bloc de control a fost conturată ca în fig. 8.1.

Prin SIGA, SIGB, respectiv SIGH, s-au reprezentat modulele de formare a semnăturii, pe 4 biți, pentru operanzii A și B, respectiv pentru cuvîntul caracteristicii de control H. Relațiile de control generale dezvoltate în capitolul 6 pentru operațiile aritmetice și logice nu depind de modul de formare a semnăturii, respectiv de metoda de comprimare a informației utilizată. Astfel, modulele de formare a semnăturii pot fi atît de tip serial cît și paralel [98]. În [97] și [100] autorul prezintă un studiu, în colaborare, asupra formării paralele și seriale a semnăturilor. Alegerea uneia sau alteia dintre metode depinde în principal de doi factori. Unul dintre ei îl reprezintă redundanța informațională implicată de formarea semnăturii. Astfel, la comprimarea serială redundanța poate fi variabilă în funcție de numărul de ranguri ale registrului de deplasare cu reacție utilizat. În schimb la comprimarea paralelă redundanța este de 100%, numărul rangurilor registrului de comprimare fiind egal cu numărul de biți ai cuvîntului de

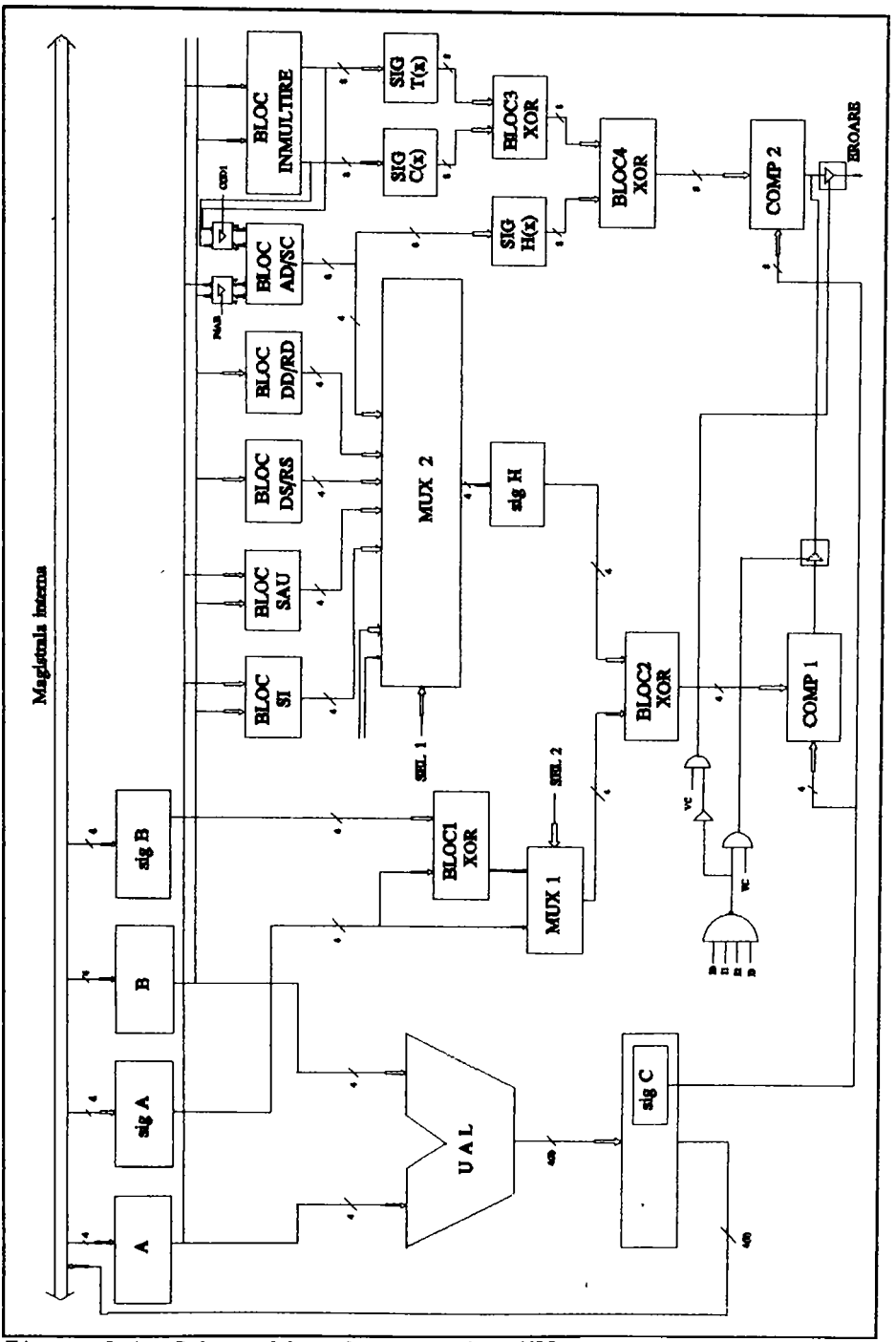

Figura 8.1. Schema bloc de control a UAL.

informaţie. Cel de al doilea factor îl constituie viteza de comprimare, evident corespunzător mai mare în cazul comprimării seriale. Mai trebuie subliniat şi avantajul semnăturii paralele privitor la posibilitatea încărcării datelor într-un flux continuu, nefiind necesară iniţializarea de fiecare dată a registrului de comprimare, ca în cazul semnăturii seriale. În ceea ce priveşte capacitatea de detecţie a erorilor, ea este aceeaşi pentru ambele metode, evident la aceeaşi redundanţă. Prin urmare, optarea pentru una dintre soluţiile de formare a semnăturii trebuie făcută în funcţie de mărimea cuvintelor din UAL şi de viteza acestuia. Astfel, de exemplu, timpul de formare a semnăturii seriale trebuie să se încadreze în intervalul de timp în care UAL efectuează operaţia. Evident, în acest caz, tactul de comandă a registrului de comprimare poate fi ales la valoarea maximă permisă de tehnologie. Numărul de biţi ai semnăturii va fi dictat de alegerea unui polinom generator pentru o acoperire convenabilă a defecţiunilor [44]. În cazul variantei de sinteză prezentate s-a ales soluţia comprimării paralele, pentru a pune în evidenţă performanţele maxime ale blocului de control.

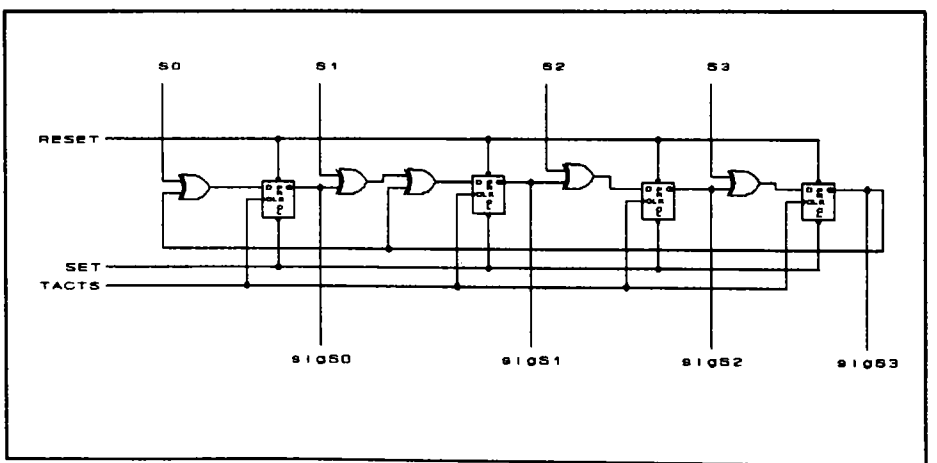

Fig. 8.2. Registru de formare a semnăturii pentru $G(x)=x^4+x+1$.

În fig. 8.2 s-a reprezentat schema de principiu a registrului de formare a semnăturii, pentru polinomul generator $G(x)=x^4+x+1$, sintetizat cu bistabile de tip D. Sinteza sub formă de reţea logică regulată a registrului de comprimare a pornit de la structura bistabilului din fig. 8.3. Astfel, în fig. 8.4 este prezentată schema completă de formare a semnăturii, comună

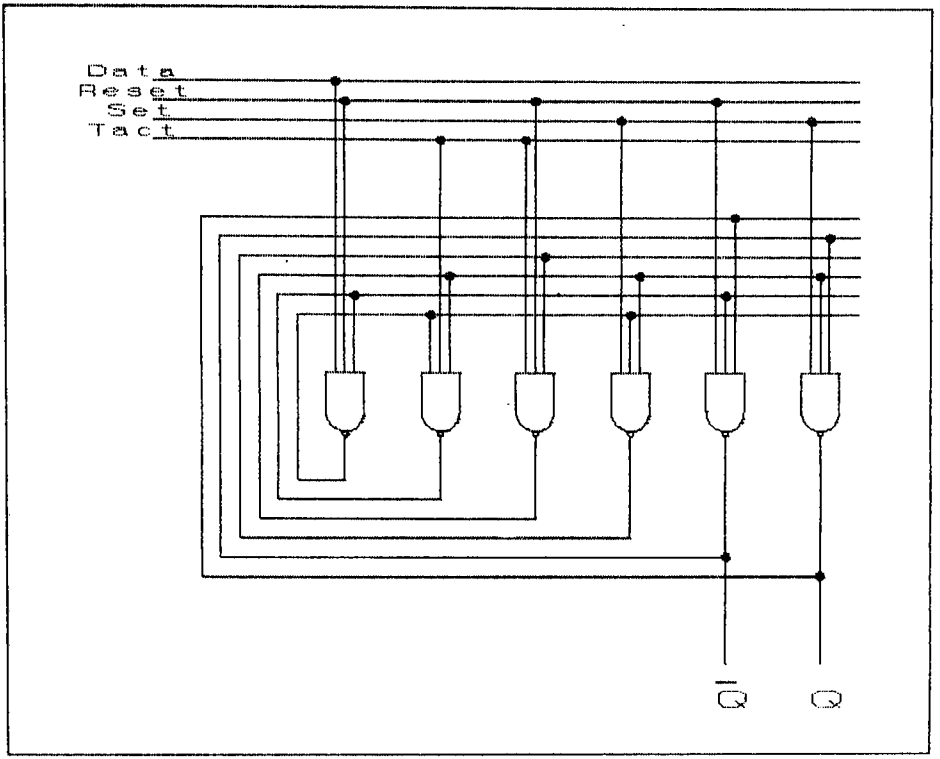

Fig. 8.3. Bistabil D sub formă de reţea logică regulată.

pentru operanzii A şi B, precum şi pentru cuvîntul caracteristicii de control H. Formarea semnăturii este condiţionată prin semnalul de tact de către codul operaţiei CODOP, pentru toate operaţiile cu excepţia celei de înmulţire, unde controlul se organizează într-un bloc separat. Blocul ŞI pentru formarea caracteristicii de control a operaţiei SAU, respectiv blocul SAU pentru formarea caracteristicii de control a operaţiei ŞI sînt prezentate în fig. 8.5, respectiv fig. 8.6.

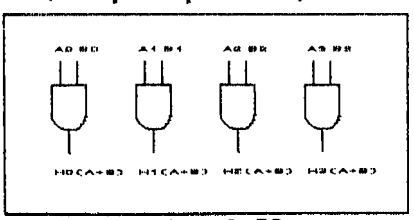

Fig. 8.5. Blocul ŞI.

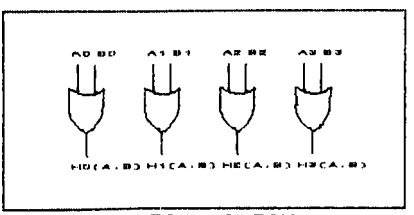

Fig. 8.6. Blocul SAU.

Pentru controlul corectitudinii îndeplinirii operaţiei logice SAU-EXCLUSIV (A⊕B) s-a optat pentru utilizarea relaţiei (6.25),

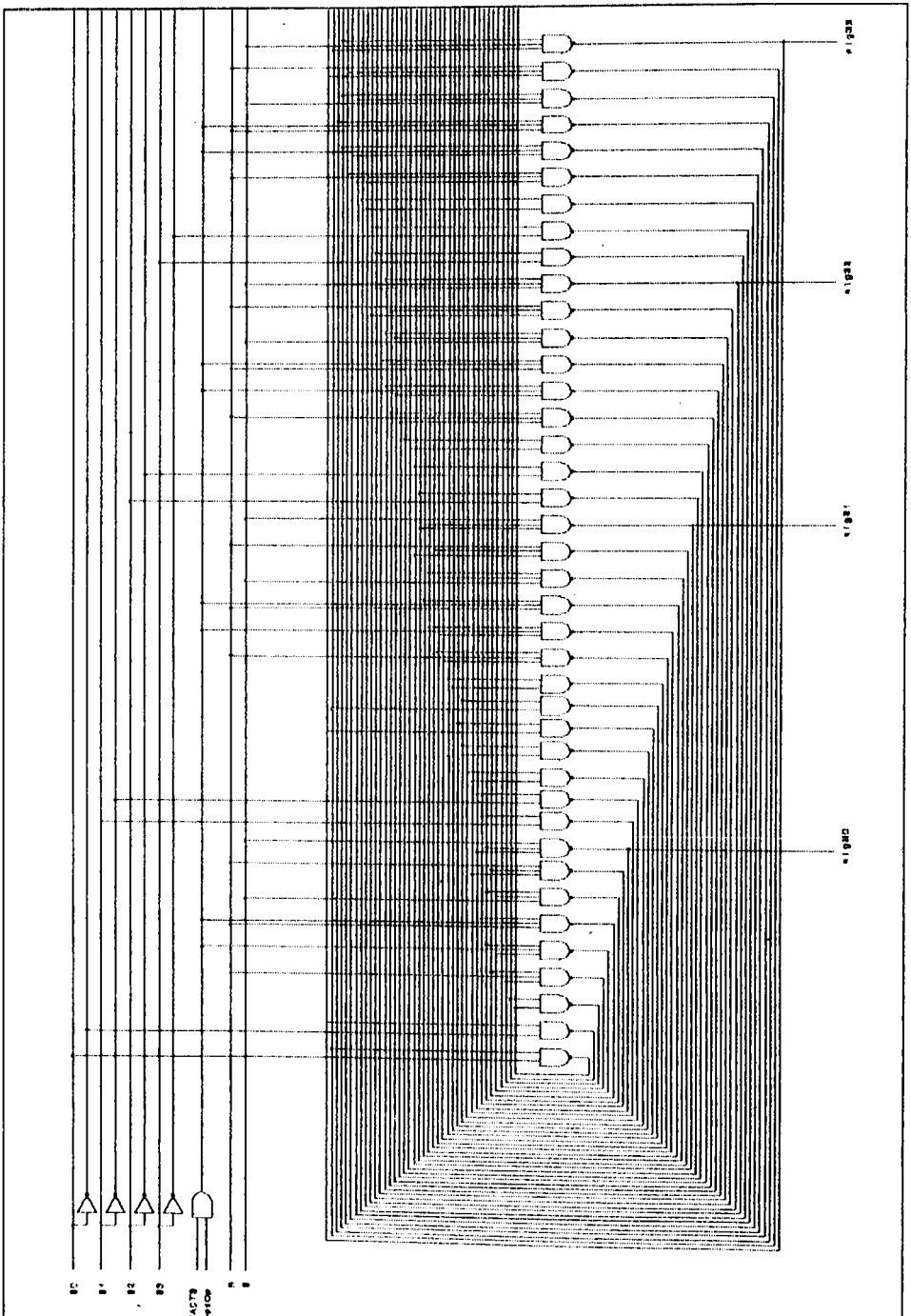

Fig. 8.4. Schema de formare a semnăturii pe 4 biţi.

exploatîndu-se avantajul formării semnăturii paralele. Astfel, relaţia (6.25) mai poate fi scrisă sub forma:

$$sig(A \oplus B) = sigA \oplus sigB \oplus sig\bar{\alpha}_n \quad (8.1$$

unde $\bar{\alpha}_n=(00...00)_n$ este cuvîntul avînd toţi cei n biţi la valoarea logică 0. În felul acesta se observă simetria perfectă între relaţiile (6.26) şi (8.1), arătînd încă o dată generalitatea proprietăţilor de controlabilitate a operaţiilor pentru metoda propusă.

Blocul SAU-EXCLUSIV a fost sintetizat, pentru 4 biţi, conform cu fig. 8.7, fiind identic atît pentru BLOC1-XOR (fig. 8.1), necesar pentru execuţia operaţei A⊕B, cît şi pentru BLOC2-

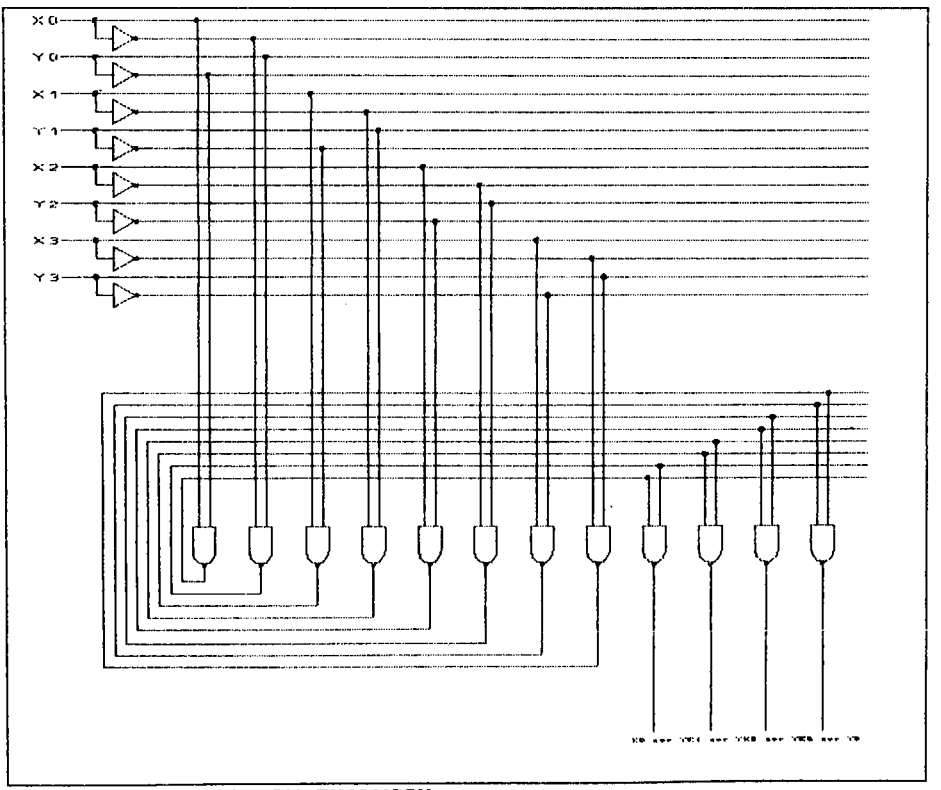

Fig. 8.7. Blocul SAU-EXCLUSIV.

XOR (fig. 8.1), utilizat pentru formarea informaţiei de control a operaţiilor aritmetice şi logice, conform cu relaţiile deduse.

Avînd în vedere structura relaţiilor de control a

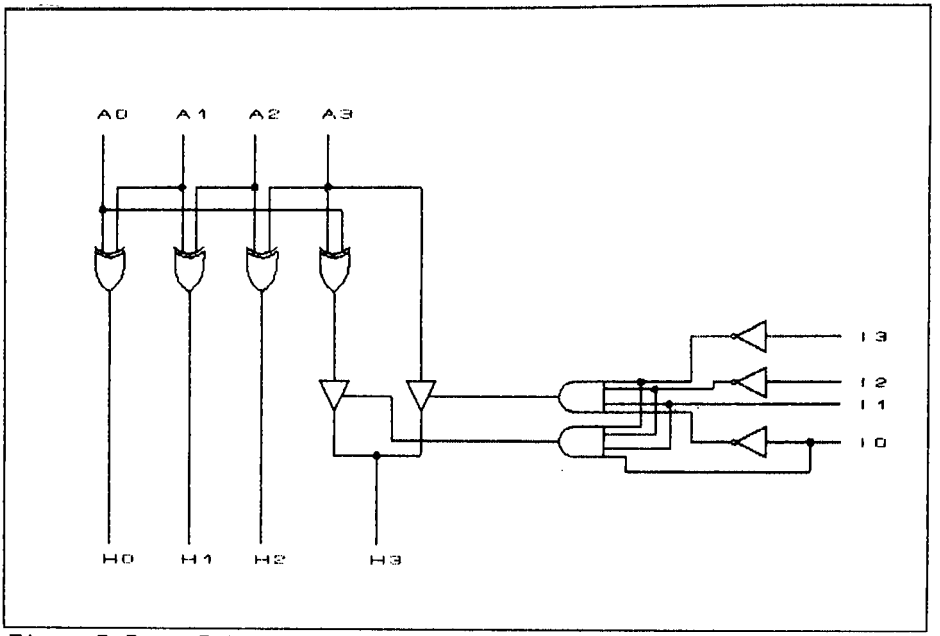

Fig. 8.8a. Schema de principiu pentru modulul de control a operației deplasare/rotație dreapta.

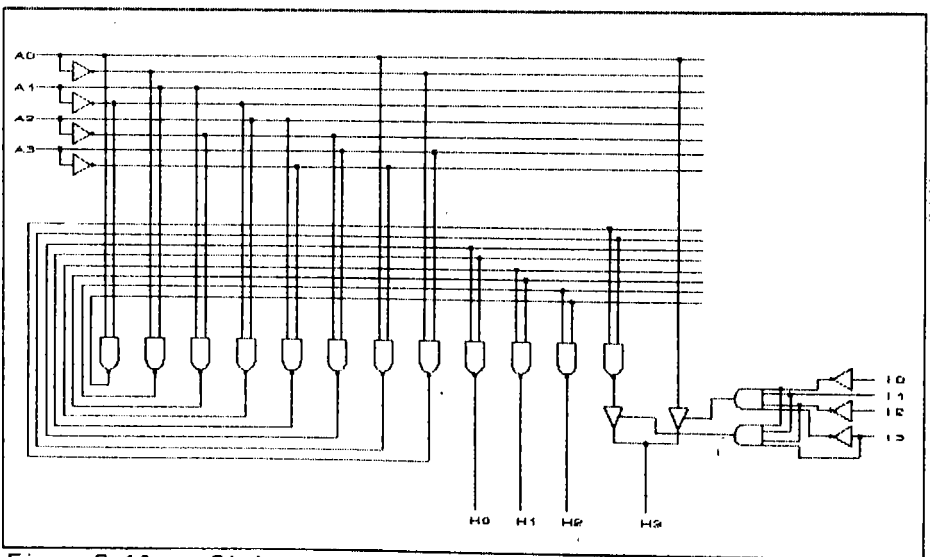

Fig. 8.10. Si-t-za modu`u`ui de control a operați_i ep asare/rotație stînga.

operațiilor de deplasare și rotație (6.27)-(6.30), s-a optat pentru sinteza comună în același bloc a controlului deplasării

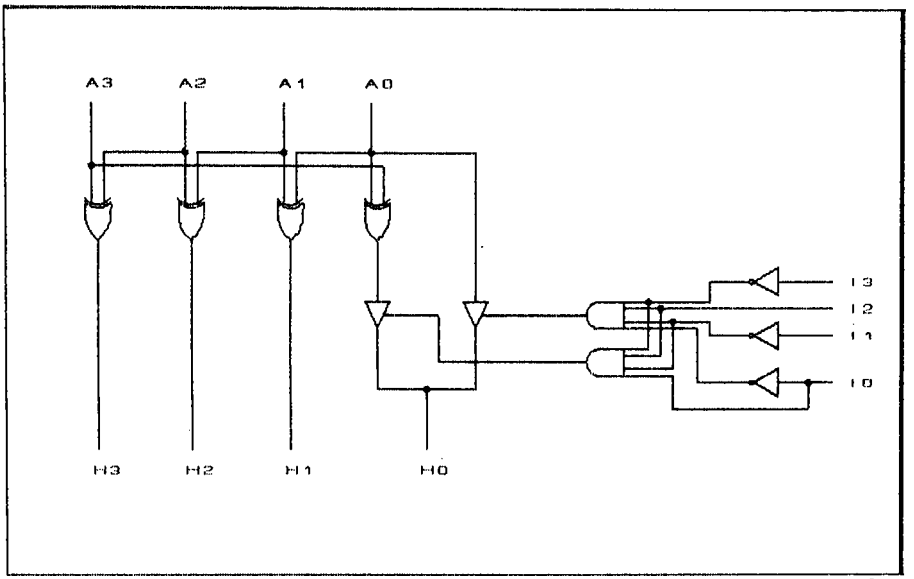

Fig. 8.8b. Schema de principiu pentru modulul de control a operaţiei deplasare/rotaţie stînga.

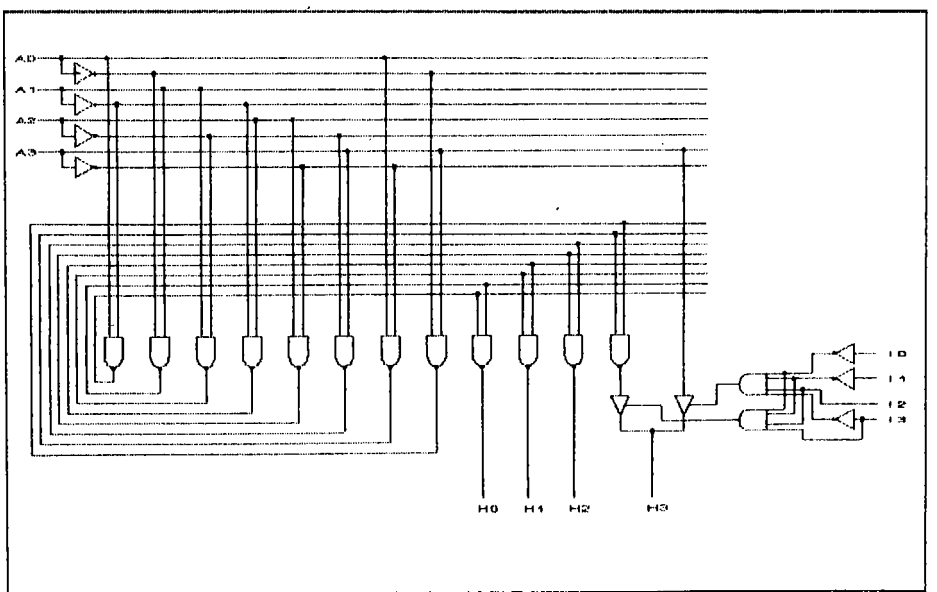

Fig. 8.9. Sinteza modulului de control a operaţiei deplasare/ rotaţie dreapta.

şi rotaţiei, utilizînd corespunzător circuite tri-state. În fig. 8.8a şi b este arătată schema de principiu pentru blocurile de

control a operaţiilor de deplasare/rotaţie dreapta şi stînga.

Sinteza sub formă de reţea logică regulată a acestor blocuri apare în fig. 8.9, pentru deplasare/rotaţie dreapta, respectiv în fig. 8.10, pentru deplasare/rotaţie stînga.

Pentru controlul operaţiilor de coincidenţă şi inversiune nu a fost necesar de a prevedea blocuri separate.

Caracteristicile mutuale H(A+B) şi H(A-B) necesare pentru controlul operaţiilor de adunare şi scădere, avînd la bază relaţiile (6.33) şi (6.35) sînt obţinute într-un singur bloc, datorită simetriei celor două relaţii, prin utilizarea de circuite tri-state validate de codurile operaţiilor de adunare şi scădere. Blocul AD/SC este reprezentat în fig. 8.11. Acest

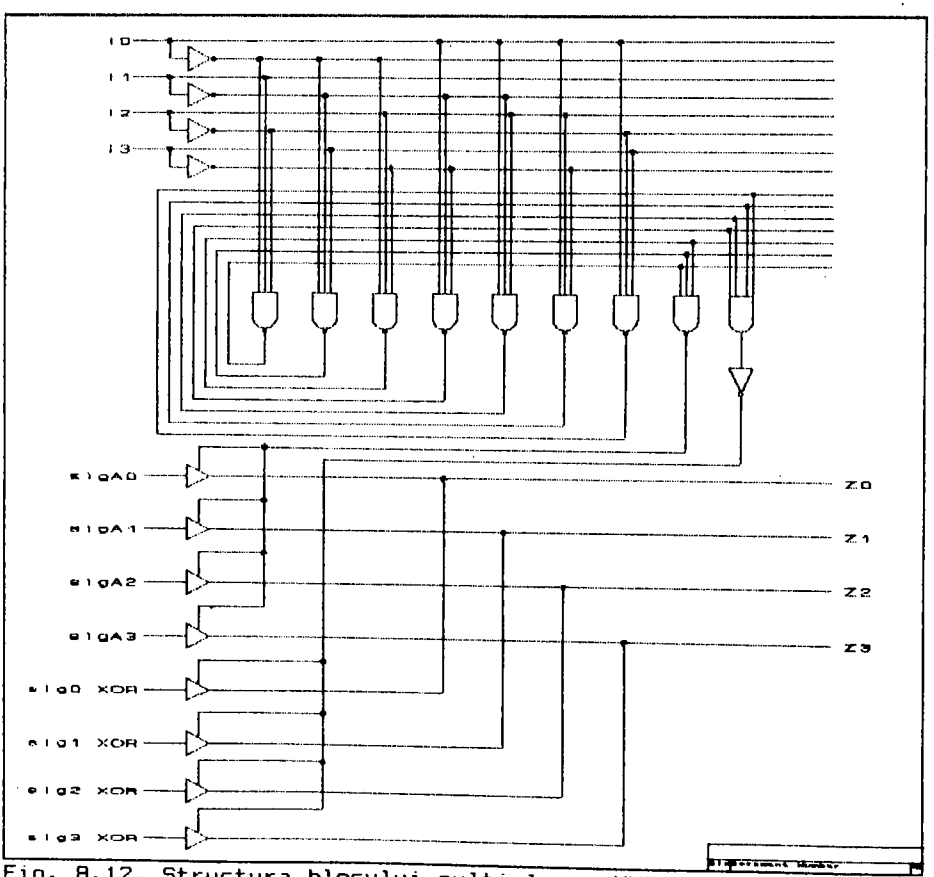

Fig. 8.12. Structura blocului multiplexor MUX1.

bloc a fost sintetizat şi pentru posibilitatea formării caracteristicii mutuale la adunările pe 8 biţi, care intervin la

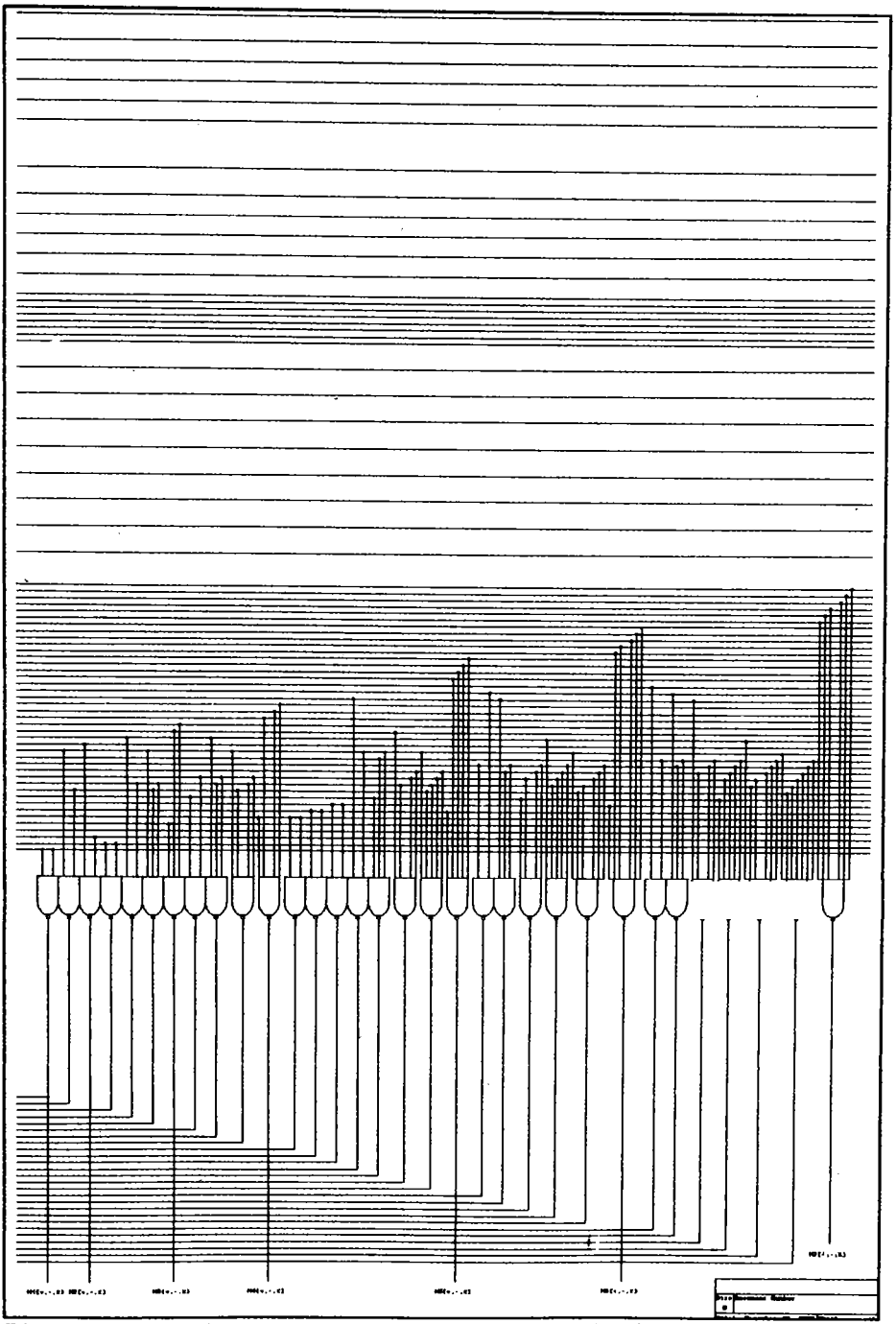

Fig. 8.11. Structura blocului de control (AD/SC) pentru operaţia de adunare/scădere.

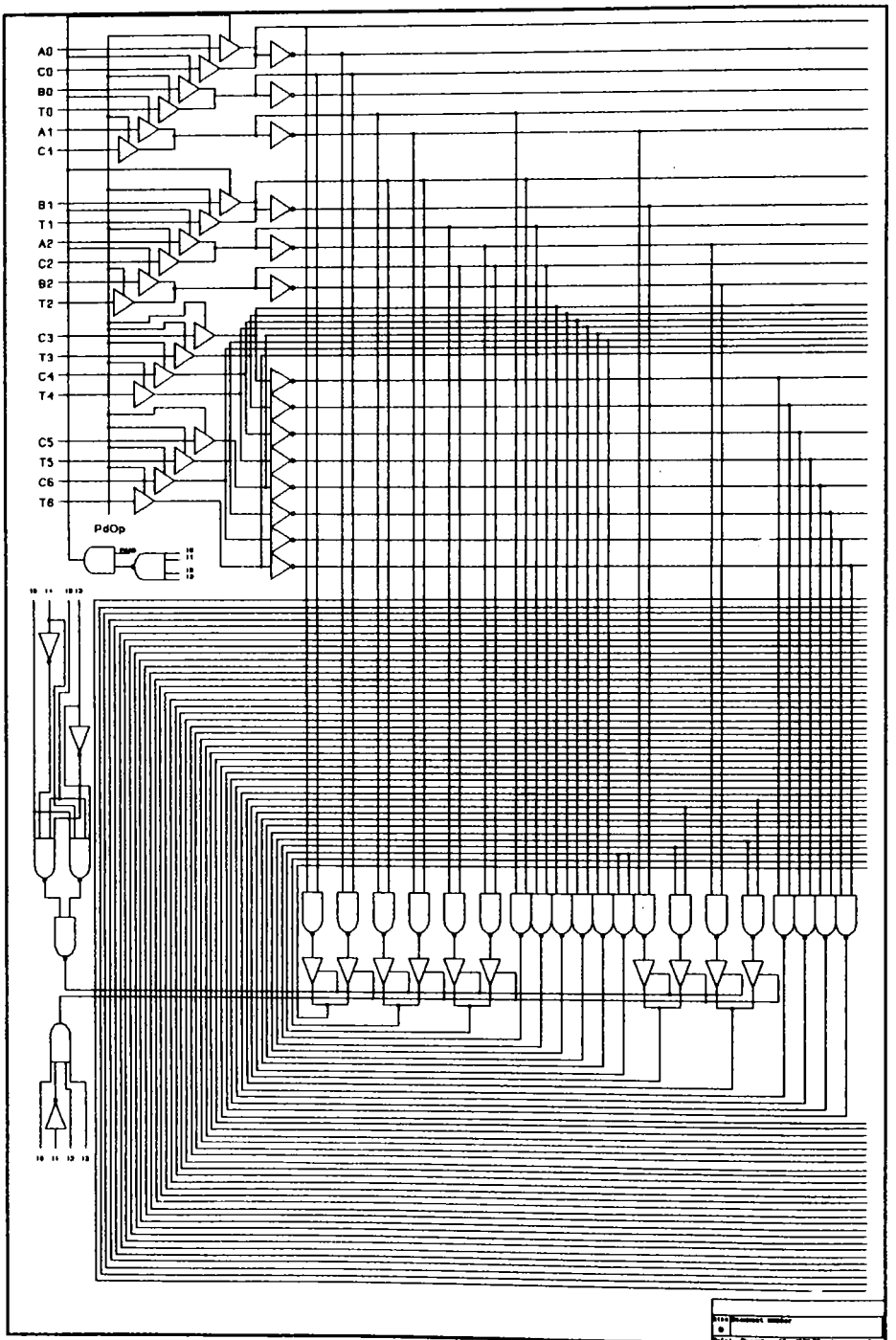

Fig. 8.11. (continuare)

controlul operaţiei de înmulţire. Selecţia între operaţiile pe 4 biţi şi cele pe 8 biţi se face după codul operaţiei.

Schema de control a fost prevăzută cu un bloc multiplexor MUX1, în vederea selecţiei între semnătura operandului A (sigA) şi rezultatul operaţiei SAU-EXCLUSIV între semnăturile operanzilor A şi B (sigA⊕sigB). Schema multiplexorului MUX1 este prezentată în fig. 8.12.

În vederea formării semnăturii sigH, pentru caracteristicile mutuale ale operaţiilor ŞI, SAU, SAU-EXCLUSIV, COINCIDENŢĂ, INVERSIUNE, DEPLASARE/ROTAŢIE, ADUNARE/SCĂDERE a fost sintetizat un bloc multiplexor MUX2, reprezentat în fig. 8.13. Pentru selecţia multiplexoarelor s-au utilizat codurile operaţiilor din cuvîntul de instrucţie I_0 ... I_3, care au fost stabilite, prin convenţie, conform tab. 8.1.

Tab. 8.1. Codificarea operaţiilor.

TIP OPERAŢIE	COD OPERAŢIE I_0	I_1	I_2	I_3
SAU	1	0	0	0
ŞI	1	1	0	1
SAU-EXCLUSIV	1	0	0	1
COINCIDENŢĂ	1	1	1	0
DEPLASARE STÎNGA	0	1	0	0
ROTAŢIE STÎNGA	0	1	0	1
DEPLASARE DREAPTA	0	0	1	0
ROTAŢIE DREAPTA	0	0	1	1
INVERSARE	0	1	1	0
ADUNARE	1	0	1	0
SCĂDERE	1	0	1	1
ÎNMULŢIRE	1	1	1	1

Ecuaţiile celor două multiplexoare sînt:

$$z_1 = sigA \cdot \overline{I_3} + (sigA \oplus sigB) \cdot I_3, \qquad (8.2)$$

$$z_2 = \text{Ş}I \cdot I_3 \overline{I_2} I_1 I_0 + SAU \cdot \overline{I_3} \overline{I_2} \overline{I_1} I_0 + STG \cdot \overline{I_0} \overline{I_2} I_1 + DR \cdot \overline{I_0} I_2 \overline{I_1} + AS \cdot I_0 I_2 \overline{I_1} + \alpha_n \cdot I_2 I_1 \overline{I_3} + \overline{\alpha_n} \cdot \overline{I_1} \overline{I_2} I_3$$

(8.3)

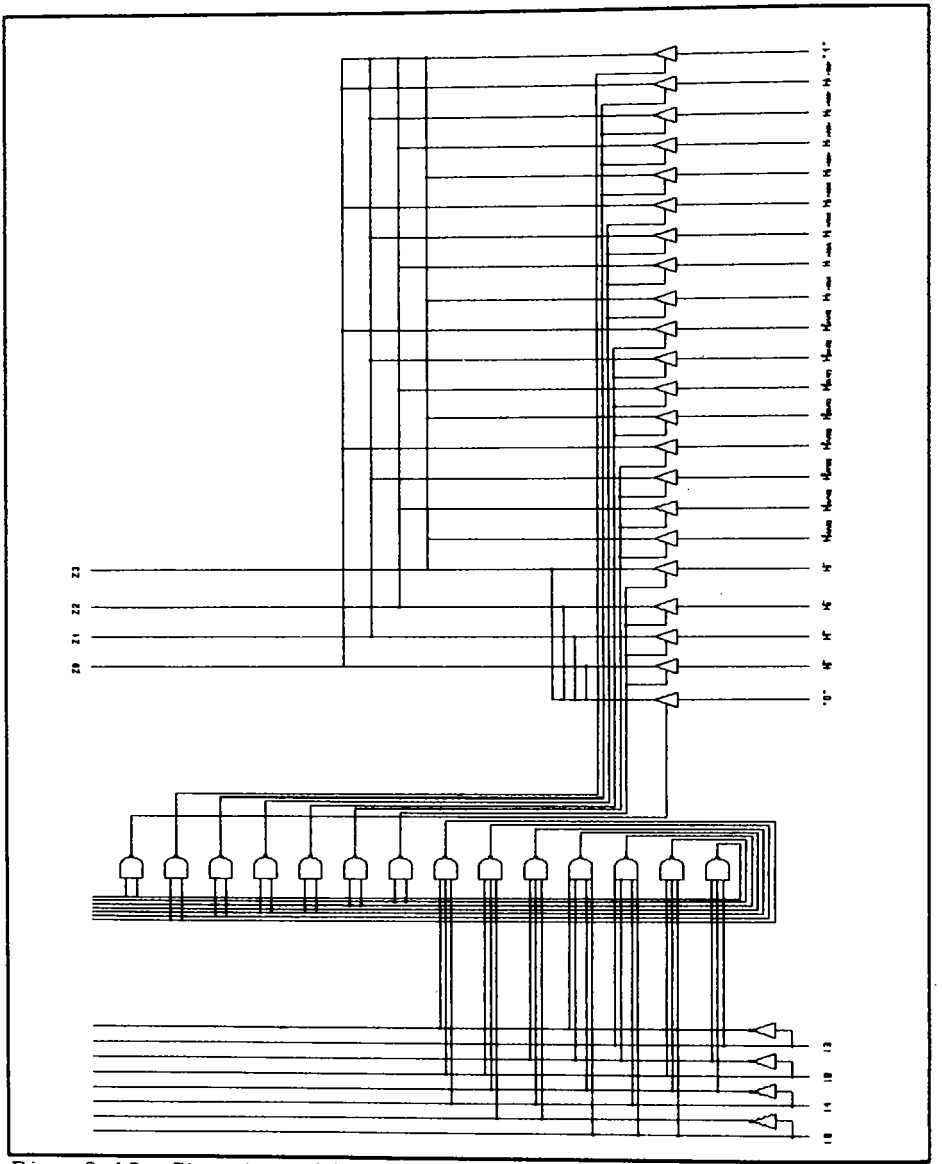

Fig. 8.13. Structura blocului multiplexor MUX2.

Compararea informaţiei de control formată la ieşirea celui de al doilea bloc SAU-EXCLUSIV (fig. 8.1) cu semnătura rezultatului operaţiei din UAL, sigC, se realizează pe 4 biţi, bit cu bit, în modulul comparator COMP1. În fig. 8.14 se prezintă schema modulului comparator. Rezultatul comparării,

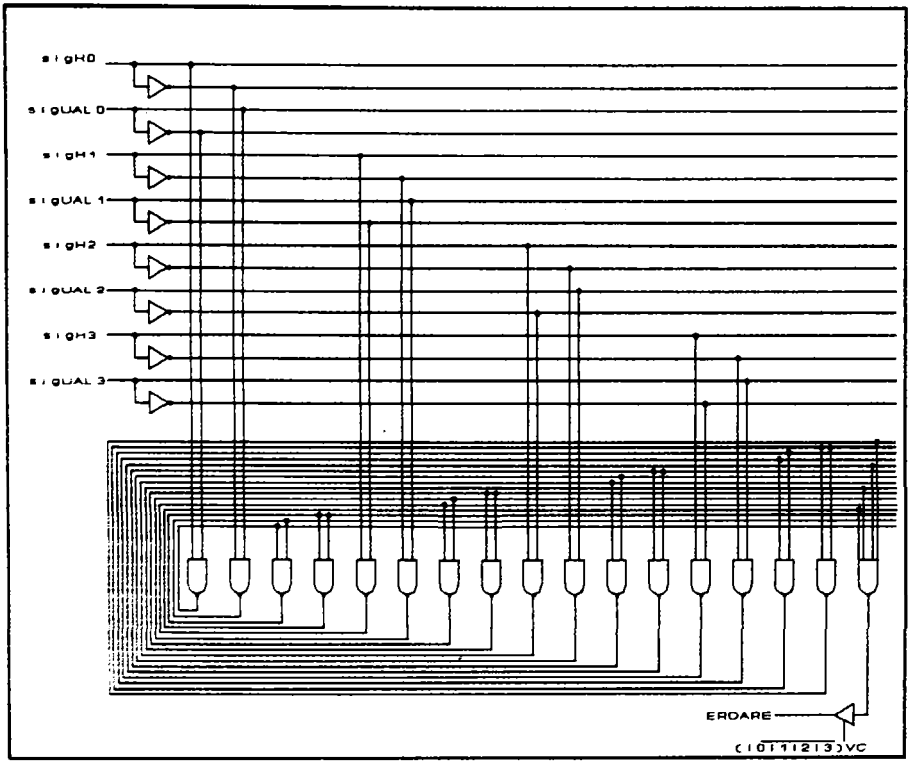

Fig. 8.14. Modulul comparator.

reprezentat de semnalul EROARE, este validat la ieşirea modulului prin linia VC. În ceea ce priveşte operaţia de înmulţire, compararea informaţiei de control, formată într-un cod separat, cu rezultatul din UAL se face pe 8 biţi într-un al doilea modul comparator, COMP2 (fig. 8.1), sintetizat într-o manieră similară. În funcţie de codul operaţiei este validată ieşirea unuia, sau a altuia dintre comparatoare.

Execuţia operaţiei de înmulţire, având la bază relaţia (8.9), impune o structură, formată din mai multe module, reprezentată în fig. 8.15. Astfel, este necesar un registru de încărcare paralelă şi de serializare a deînmulţitului, realizat conform schemei din fig. 8.16. Deînmulţitul se aplică serial unui dispozitiv de înmulţire polinomială, iar înmulţitorul este încărcat paralel. Ieşirea acestui dispozitiv este de asemenea serială, iar rezultatul înmulţirii polinomiale, $A(x) \cdot B(x) \cdot R(x)$, este obţinut după 7 tacturi. Acest timp este asigurat de un numărător modulo 7. În cazul utilizării a 16

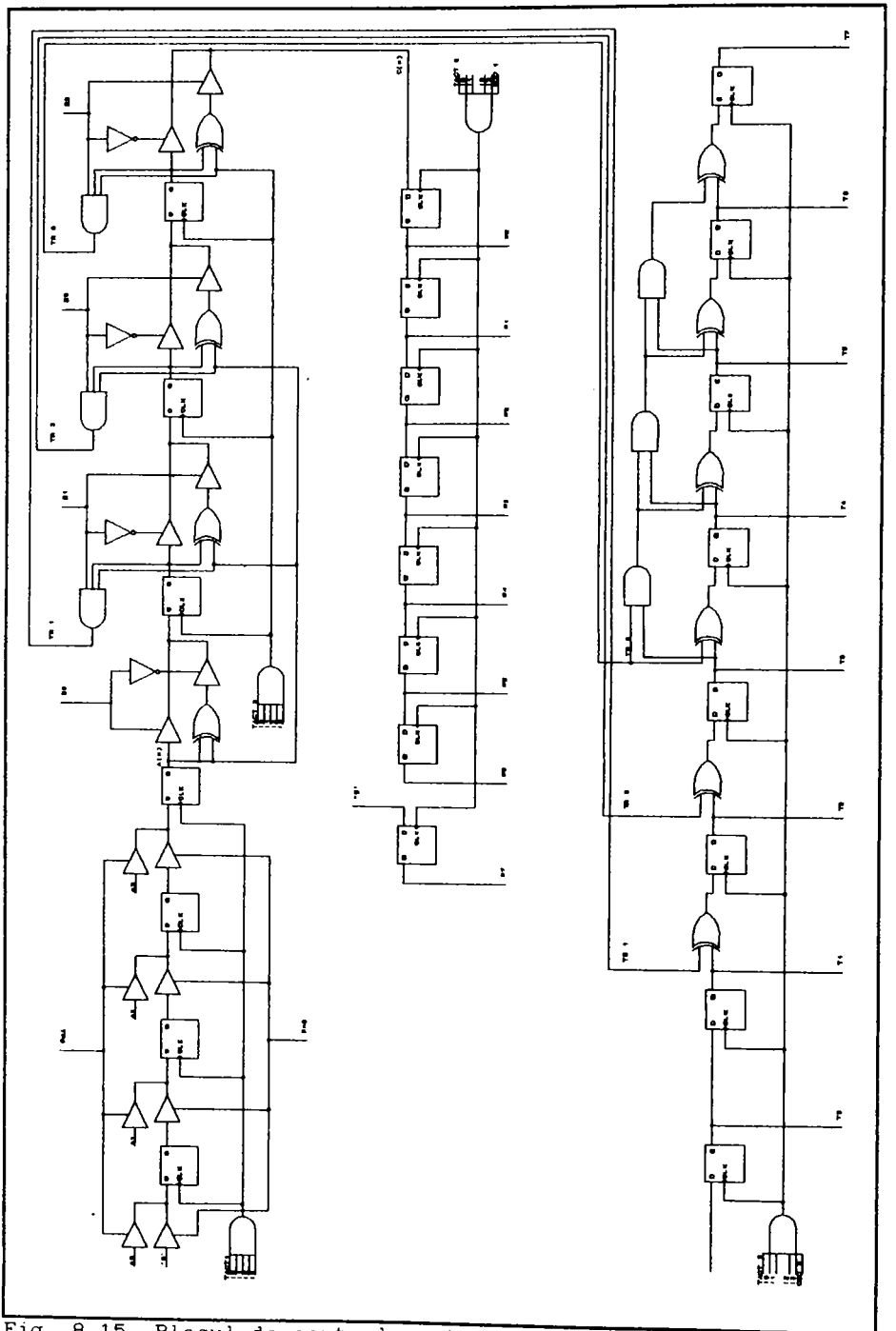

Fig. 8.15. Blocul de control pentru operaţia de înmulţire.

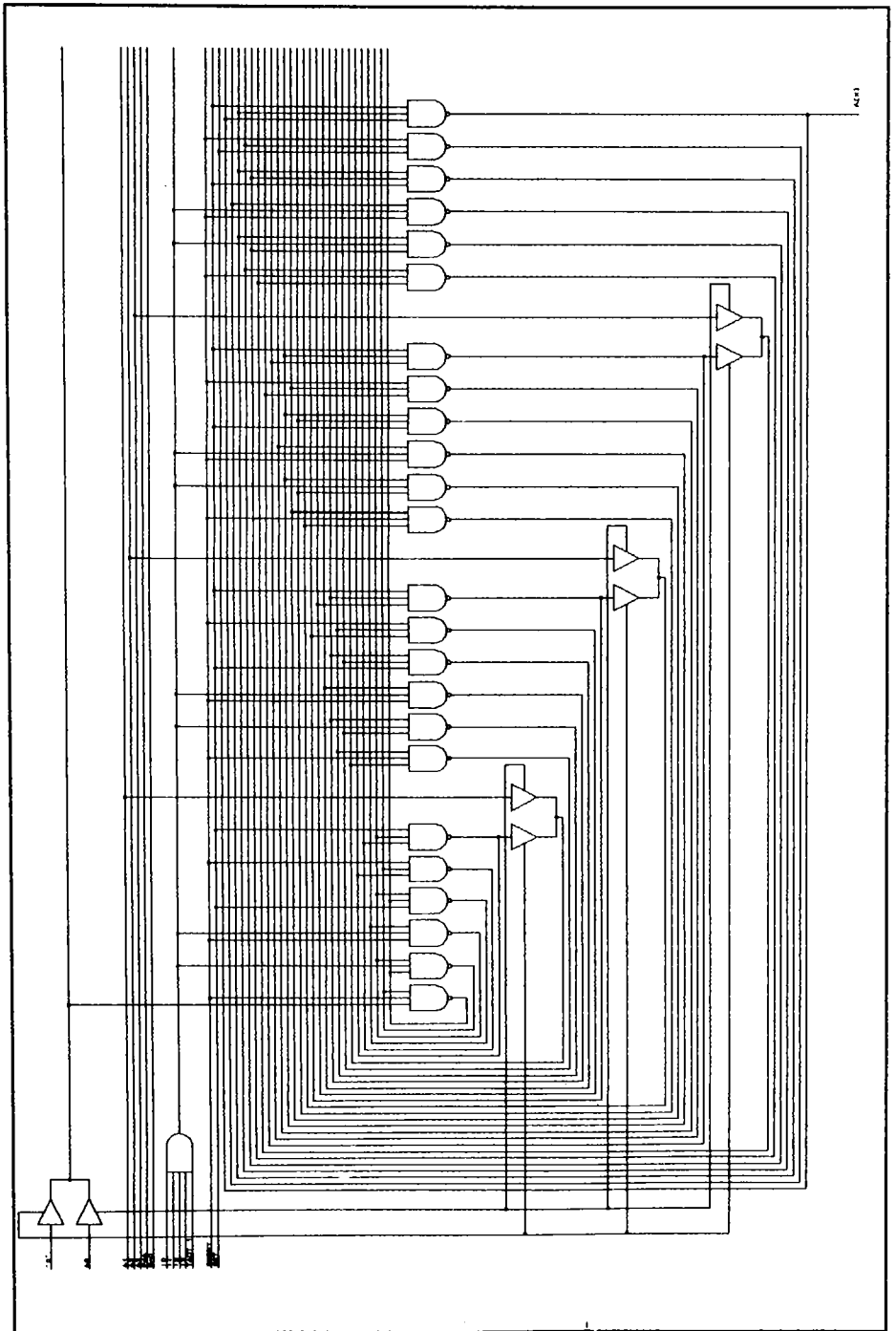

Fig. 8.16. Sinteza registrului de serializare.

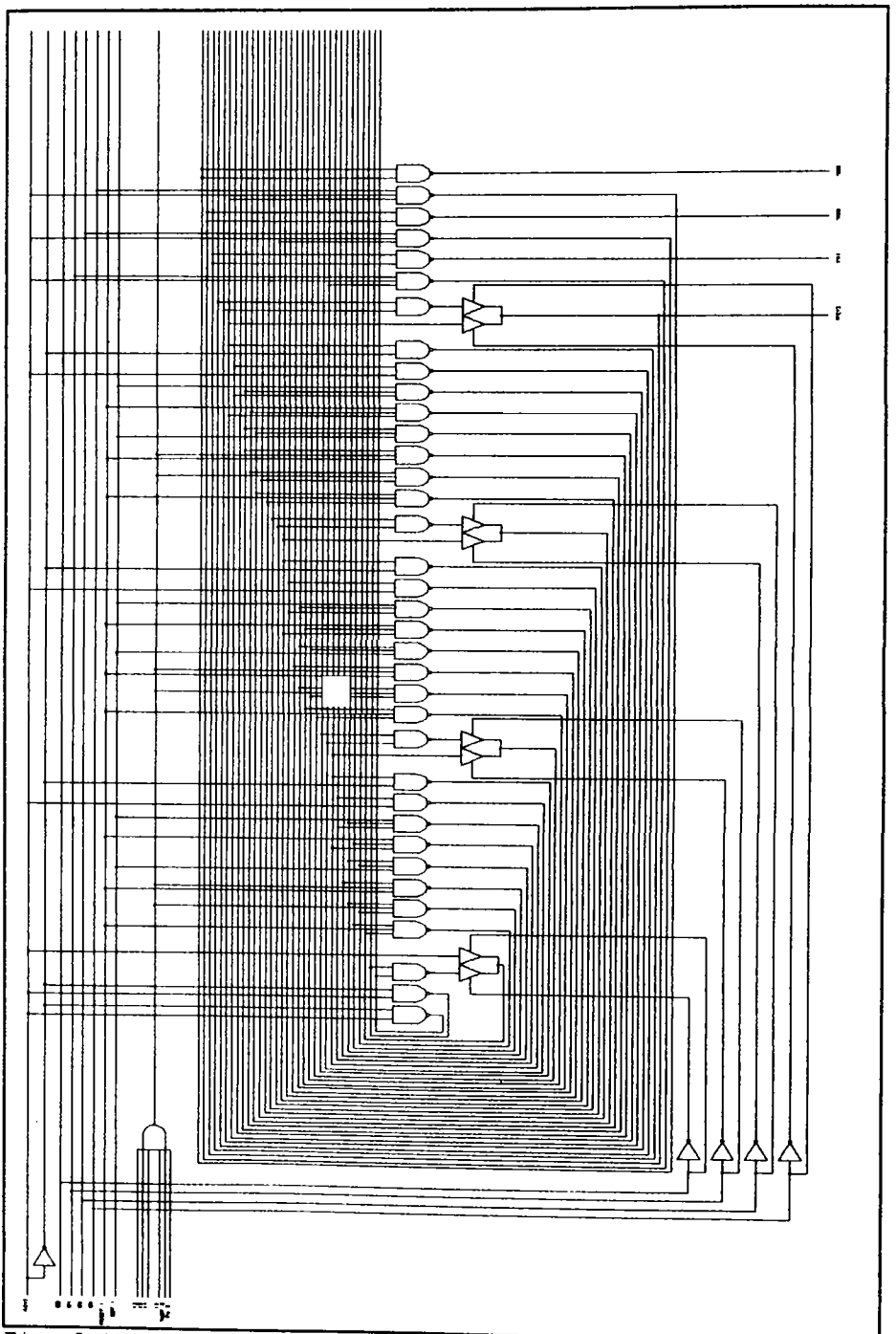

Fig. 8.17. Structura modulului de înmulțire polinomială.

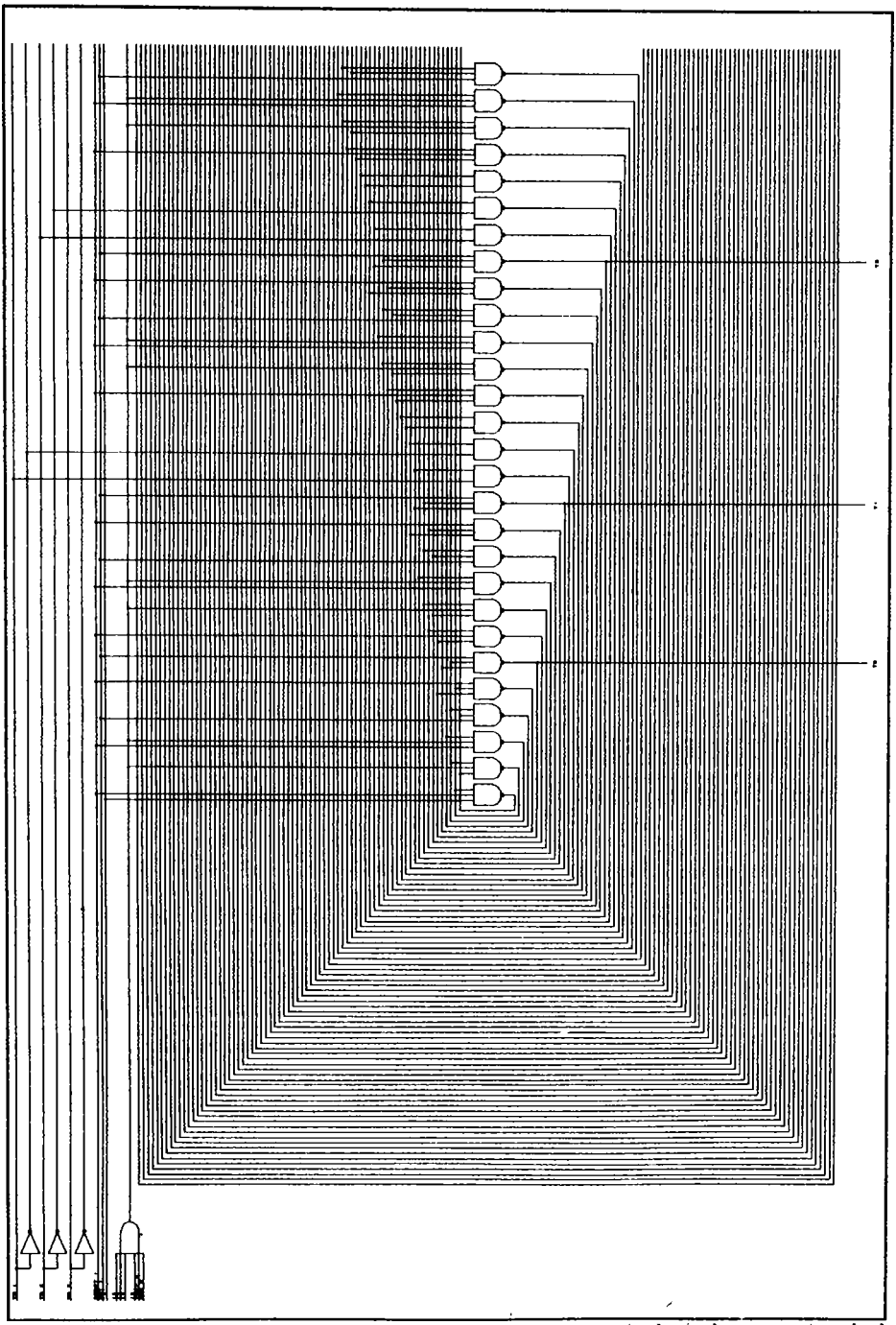

Fig. 8.18. Modulul de determinare a transportului la controlul operaţiei de înmulţire.

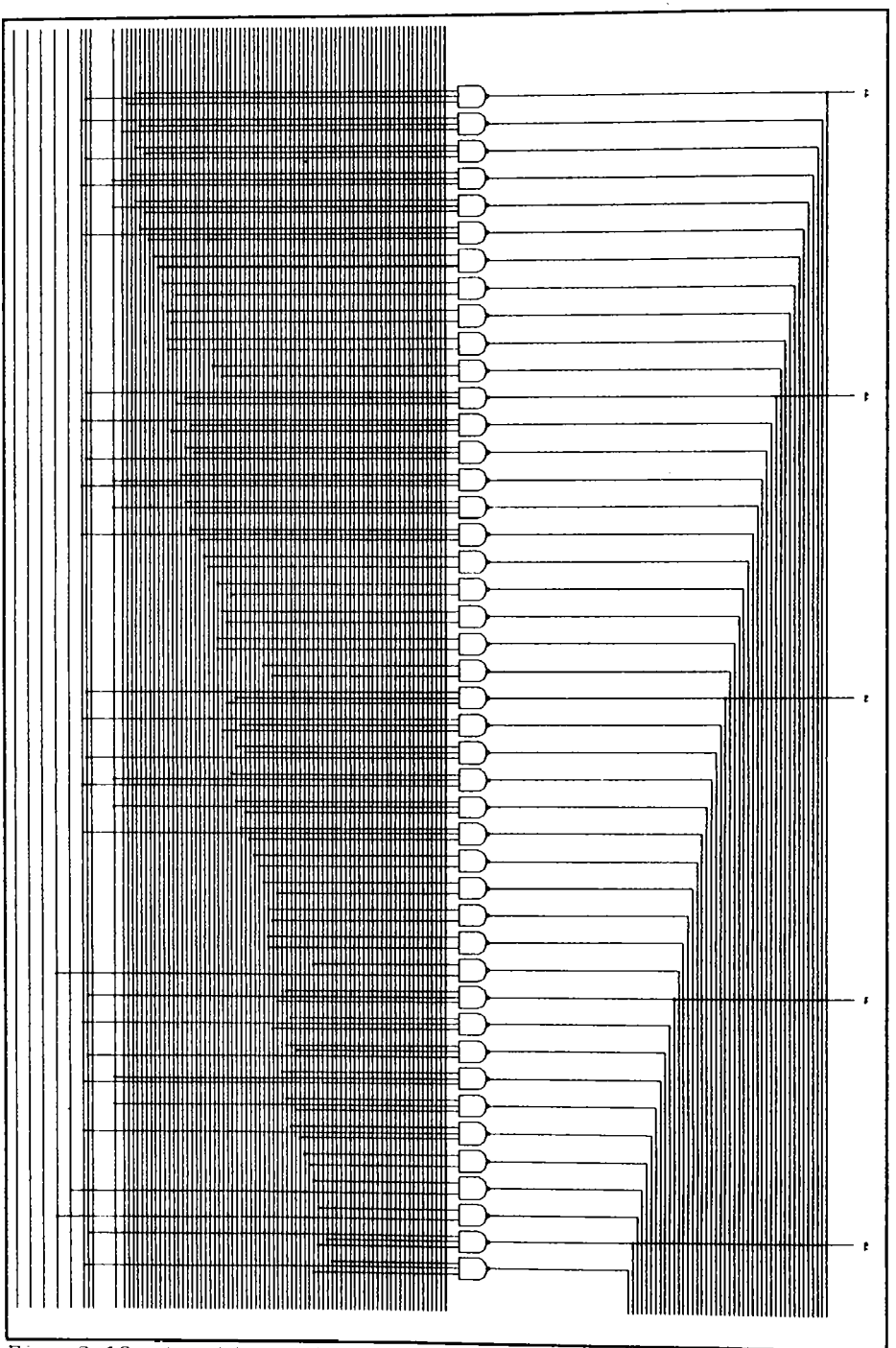

Fig. 8.18. (continuare)

biţi de date, înmulţirea polinomială va fi obţinută după 31 tacturi. În fig. 8.17 se arată modalitatea de sinteză a dispozitivului de înmulţire polinomială. În paralel cu dispozitivul de înmulţire polinomială, funcţionează dispozitivul de calcul al transportului T(x), care se obţine după 4 tacturi, timp controlat de un numărător modulo 4. Pentru cazul cuvintelor de date de 16 biţi, transportul se va obţine după 16 tacturi. Dispozitivul de calcul a transportului este prezentat în fig. 8.18. Rezultatul C(x), al înmulţirii polinomiale este încărcat serial într-un registru de memorare, arătat în fig. 8.15. Ultimul bistabil din acest registru, reprezentînd rangul c.m.s., este izolat de celelalte bistabile şi conţine întotdeauna valoarea logică "0". Aceasta, deoarece din înmulţirea polinomială C(x) va avea maximum 7 cifre
binare (0...6), iar în schimb rezultatul înmulţirii aritmetice

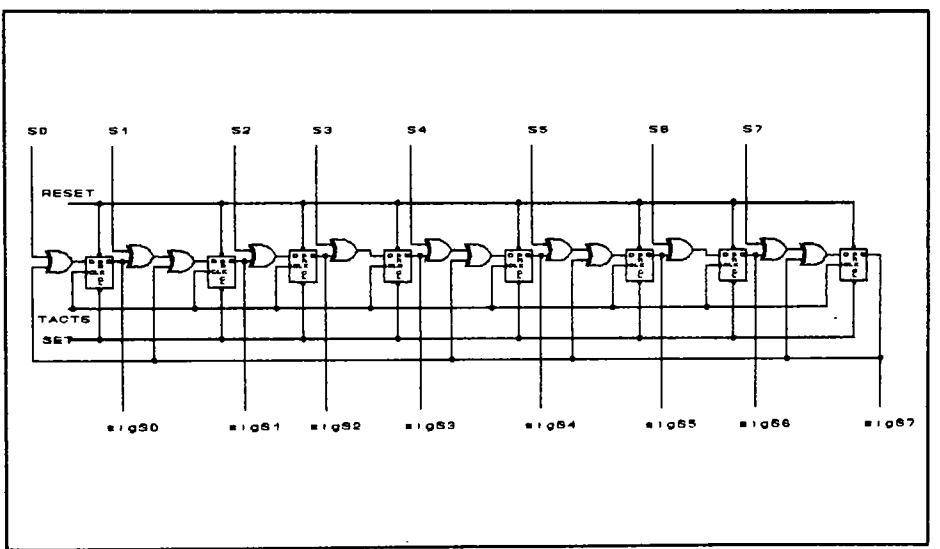

Fig. 8.19. Registru de comprimare pe 8 biţi.

finale a două numere pe 4 biţi poate să aibă maximum 8 cifre binare (0...7). Odată obţinute rezultatele C(x) şi T(x), acestea se adună în blocul AD/SC, selectat pentru operaţiile pe 8 biţi, în vederea formării caracteristicii mutuale H(x). Semnăturile aferente lui C(x), T(x) şi respectiv H(x) se formează în registre de comprimare paralelă pe 8 biţi. Polinomul generator

utilizat este $G(x)=x^8+x^7+x^3+x^2+x+1$, iar în fig. 8.19 apare schema de principiu a registrului de comprimare. Sinteza acestui registru s-a realizat în mod analog ca în cazul registrului de comprimare pe 4 biţi (fig. 8.4). În fig. 8.1 s-au reprezentat cele trei module de formare a semnăturilor pe 8 biţi, SIGC(x), SIGT(x) şi SIGH(x). De asemenea, mai sînt reprezentate două module SAU-EXCLUSIV, pe 8 biţi, necesare pentru concretizarea informaţiei de control după relaţia (6.37).

8.2. SIMULAREA FUNCŢIONĂRII BLOCULUI DE CONTROL

Avînd în vedere complexitatea ridicată a structurii blocului de control a operaţiilor aritmetice şi logice, verificarea corectei funcţionări a acestuia nu s-a putut aborda printr-un prototip experimental. Prin urmare, punerea la punct a blocului de control şi verificarea valabilităţii metodei de control propuse au fost abordate prin simulare pe calculator. Această soluţie se aliniază tendinţei moderne de proiectare [69],[70],[84], cu atît mai mult cu cît sinteza blocului de control a fost orientată spre integrarea pe scară foarte largă.

8.2.1. PROGRAMUL DE SIMULARE

În vederea simulării, a fost realizat un program modular în limbajul TurboC. Partea de început cuprinde directivele de includere a fişierelor header, ce conţin funcţiile de bibliotecă utilizate în program. Urmează prototipurile funcţiilor definite în program, iar în continuare sînt definite structurile şi declaraţiile variabilelor globale. Funcţiile utilizate urmează după corpul programului principal.

În programul principal, după iniţializarea variabilelor se apelează funcţia de culegere (), care crează ecranul de culegere date.

S-a definit o matrice M de noduri şi stări, fiecărui element al matricii, care corespunde unui moment de timp, i se alocă dinamic un număr de octeţi egal cu numărul nodurilor din schema simulată. Numărul nodurilor este obţinut la introducerea datelor, prin incrementare. Într-o matrice L sînt conţinute elementele tipurilor de porţi, iar într-o matrice E elementele de tip stimul.

In matricea M fiecare element corespunde unui moment de timp şi conţine starea tuturor nodurilor existente în schemă. Stările sînt simbolizate corespunzător pentru cele trei stări

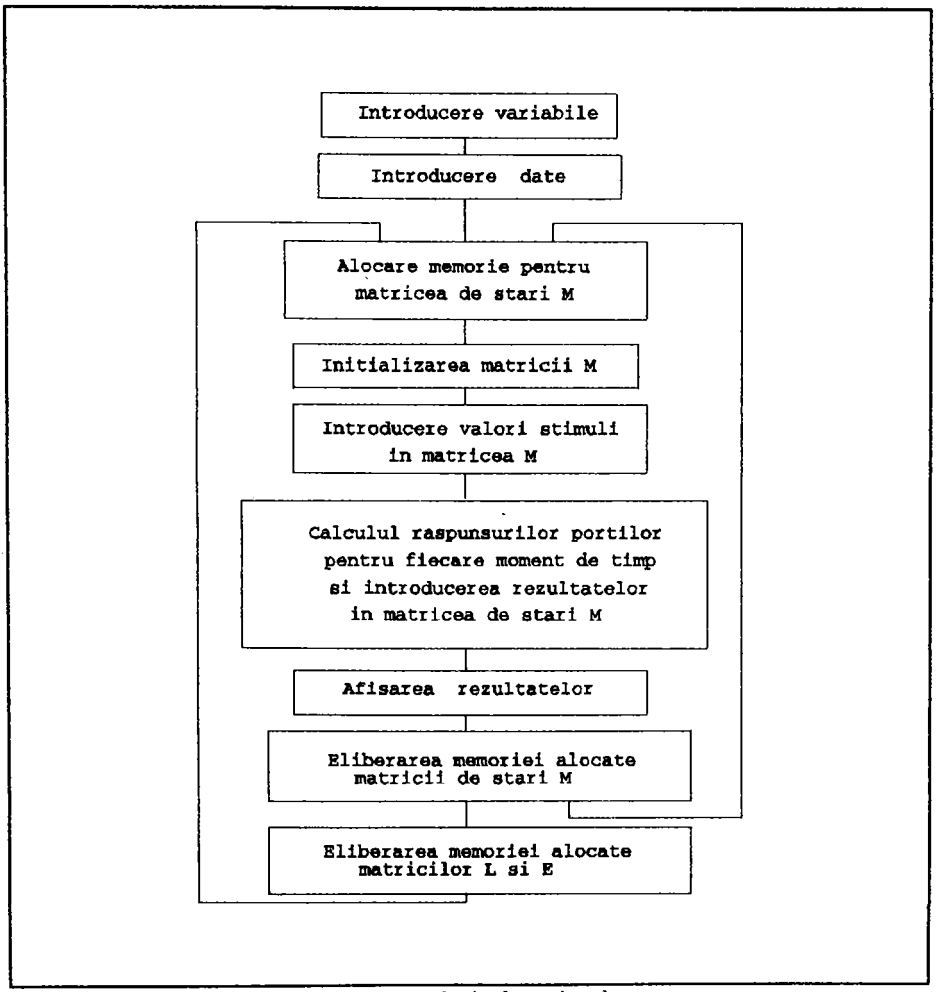

Fig. 8.20. Structura programului de simulare.

logice,

starea a treia "x", sau valoare necunoscută, nivelul logic "1" şi nivelul logic "0". Iniţial, toţi octeţii corespunzători nodurilor din schemă din matrice sînt aduşi în starea "x".
Valorile nodurilor corespunzătoare fiecărui stimul sînt trecute

Pentru fiecare moment de timp se calculează răspunsul fiecărei porţi, valoarea obţinută fiind înscrisă în octetul corespunzător fiecărui nod de ieşire din poartă. După terminarea calculelor, se procedează la vizualizarea nodurilor printr-o funcţie aferentă. După afişare are loc eliberarea memoriei alocată dinamic. Întîrzierea dată de propagarea pe o poartă s-a adoptat pentru poarta logică ŞI-NU, care stă la baza sintezei dispozitivului de control, la valoarea medie de 10 ns. Evident, în cazul concret al tehnologiei de integrare utilizate, această valoare urmează a fi modificată.

Din funcţia de afişare există posibilitatea de a reveni asupra datelor introduse, sau în ecranul de introducere pentru o nouă schemă. De asemenea, se pot încărca scheme salvate anterior.

Structura programului este prezentată schematic în fig. 8.20.

8.2.2. EXEMPLIFICAREA SIMULĂRII FUNCŢIONĂRII BLOCULUI DE CONTROL

Abordarea procesului se simulare a corectitudinii funcţionării blocului de control a operaţiilor aritmetice şi logice s-a făcut pe module funcţionale, pentru o mai bună controlabilitate şi flexibilitate a fazei de punere la punct. Odată fiecare modul simulat şi verificat în parte, s-a procedat la simularea funcţională a interconectării modulelor. Astfel, pentru asigurarea continuităţii fluxului informaţional între module, semnalele de la ieşirea unui modul au constituit intrări pentru modulul interconectat următor. În felul acesta, se pot aborda scheme cu complexitate oricît de mare.

Exemplificăm, în continuare, fazele de simulare pentru o serie de operaţii aritmetice şi logice, care se consideră derulate una după alta, într-o ordine oarecare şi executate corect de o unitate aritmetică şi logică ipotetică. Lungimea operanzilor A şi B implicaţi în aceste operaţii este de 4 biţi. În tab. 8.2 se prezintă tipul operaţiilor implicate precum şi codificarea acestora.

Verificarea corectitudinii funcţionării blocului de control a UAL se face prin compararea semnăturii (sigC) rezultatului

corect al operaţiei respective cu informaţia de control aferentă. În cazul în care blocul de control a funcţionat corect, semnalul EROARE de la ieşirea modulului comparator va rămîne invalidat. Configuraţia de date a operanzilor A şi B a fost aleasă încercînd să se acopere toate combinaţiile binare posibile.

Admiţînd operaţiile exemplificate la simulare ca derulîndu-

Tab. 8.2. Operaţiile simulate.

Nr. Op.	TIP OPERAŢIE	COD OPERAŢIE I_0	I_1	I_2	I_3
1	A∧B	1	1	0	1
2	A∨B	1	0	0	0
3	A⊕B	1	0	0	1
4	$\bar{A}$	0	1	1	0
5	A⊙B	1	1	1	0
6	A* (depl. dr.)	0	0	1	0
7	A* (depl. stg.)	0	1	0	0
8	A* (cu pierd. inform.)	0	1	0	0
9	A* (rot. dr.)	0	0	1	1
10	A* (rot. stg.)	0	1	0	1
11	A+B	1	0	1	0
12	A+B (cu depăşire)	1	0	1	0
13	A−B	1	0	1	1
14	A×B	1	1	1	1
15	A×B	1	1	1	1

se una după alta, semnăturile sigA, sigB, sigH şi sigC se formează în paralel prin încărcarea în secvenţă a fluxurilor de date corespunzătoare. Spre exemplu, în registrul de formare a sigB vor fi încărcaţi toţi operanzii B, în ordinea în care apar la execuţia instrucţiunilor. Modalitatea de formare a semnăturilor este prezentată teoretic în fig. 8.21 pentru sigA, fig. 8.22 pentru sigB, fig. 8.23 pentru sigH şi respectiv în fig. 8.24 pentru sigC, pentru toate operaţiile implicate, cu excepţia ultimelor două, care fiind de înmulţire vor fi tratate pe 8 biţi.

Detaliem în continuare operaţiile exemplificate în

simulare. Avînd în vedere derularea continuă în timp a tuturor operaţiilor, cronogramele aferente funcţionării tuturor modulelor blocului de control vor fi prezentate pe grupe de cîte patru operaţii.

1. OPERAŢIA LOGICĂ ŞI.

Aceasta este prima operaţie considerată a fi executată şi prin urmare se presupune că în blocul de control toate modulele au fost iniţializate prin semnalul RESET.

Operanzi: A=1010, B=1001.

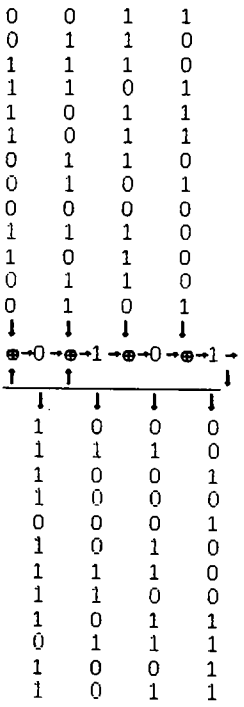

Fig. 8.21. Formarea semnăturii sigA.

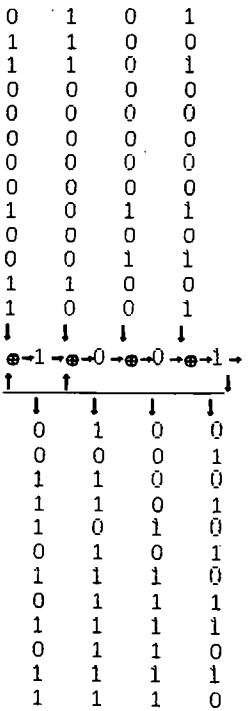

Fig. 8.22. Formarea semnăturii SigB.

Semnăturile operanzilor: sigA=1010, sigB=1001, (fig. 8.21, 8.22)

Caracteristica mutuală H (relaţia (6.2)): H=AVB=1011 (în blocul SAU).

Semnătura caracteristicii mutuale: sigH=1011, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=0011.

Informaţia de control la ieşirea BLOC2-XOR:

IC=(sigA⊕sigB)⊕sigH=1000.

Rezultatul UAL: C=A∧B=1000.

Semnătura rezultatului: sigC=1000, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

2. OPERAŢIA LOGICĂ SAU.

Operanzi: A=0110, B=0011.

Semnăturile operanzilor: sigA=0001, sigB=0010, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.1)): H=A∧B=0010 (în

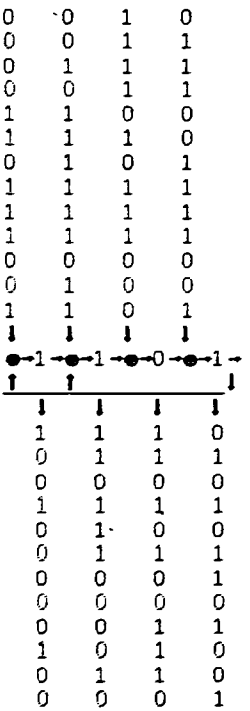

Fig. 8.23. Formarea semnăturii sigH.

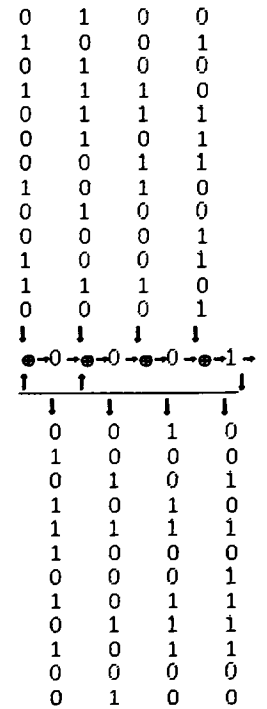

Fig. 8.24. Formarea semnăturii sigC.

blocul ŞI).

Semnătura caracteristicii mutuale: sigH=0111, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=0011.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=0100.

Rezultatul UAL: C=A∨B=0111, (fig. 8.24).

Semnătura rezultatului: sigC=0100.

Ieşire COMP1: EROARE=IC⊕sigC=0.

3. OPERAŢIA LOGICĂ SAU-EXCLUSIV.

Operanzi: A=0101, B=1100.

Semnăturile operanzilor: sigA=0111, sigB=1000, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.25)): $H=\bar{\alpha}_n=0000$, n=4 (la ieşirea blocului multiplexor MUX2). În acest caz s-a procedat la controlul operaţiei SAU-EXCLUSIV după relaţia (8.1), aşa cum s-a specificat anterior.

Pe de altă parte, controlul aceleiaşi operaţii poate fi organizat pe baza relaţiei (6.3). Astfel, rezultă $A\vee B=1101$ şi $A\wedge B=0100$.

Semnătura caracteristicii mutuale: sigH=1110, (fig. 8.23). Trebuie specificat aici că această semnătură rezultă în registrul de formare a semnăturilor în urma încărcării lui H=0000.

Pentru cea de a doua variantă a controlului rezultă $sig(A\vee B)=0011$ şi $sig(A\wedge B)=0010$.

Ieşire BLOC1-XOR: $sigA\oplus sigB=1111$.

Informaţia de control la ieşirea BLOC2-XOR: $IC=(sigA\oplus sigB)\oplus sigH=0001$.

Aceeaşi informaţie de control se va obţine şi în cea de a doua variantă de control: $IC=sig(A\vee B)\oplus sig(A\wedge B)=0001$.

Rezultatul UAL: $C=A\oplus B=1001$.

Semnătura rezultatului: sigC=0001, (fig. 8.24).

Ieşire COMP1: $EROARE=IC\oplus sigC=0$.

4. OPERAŢIA LOGICĂ DE INVERSIUNE.

Operanzi: A=0111, B=0000. În acest caz, precum şi în cazul tuturor operaţiilor ce implică un singur operand, cel de al doilea operand, în această situaţie B, este considerat nul. Astfel, se indică necesitatea de a încărca vectorul nul în registrul de comprimare paralelă, formîndu-se şi pentru celălalt operand o semnătură. În felul acesta se asigură continuitatea fluxului informaţional de control şi respectarea generalităţii relaţiei de control (6.22).

Semnăturile operanzilor: sigA=1001, sigB=0011, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.4)): $H=\alpha_n=1111$, n=4 (la ieşirea blocului multiplexor MUX2).

Semnătura caracteristicii mutuale: sigH=0000, (fig. 8.23).

Ieşire BLOC1-XOR: $sigA\oplus sigB=1010$.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1010.

Rezultatul UAL: C=$\bar{A}$=1000.

Semnătura rezultatului: sigC=1010, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

Controlul operaţiilor 1-4, prezentate mai sus, poate fi urmărit în continuare pe diagramele de timp rezultate în urma simulării funcţionării blocului de control. Astfel, în fig. 8.25 se arată modul de formare a semnăturii pentru operanzii A.

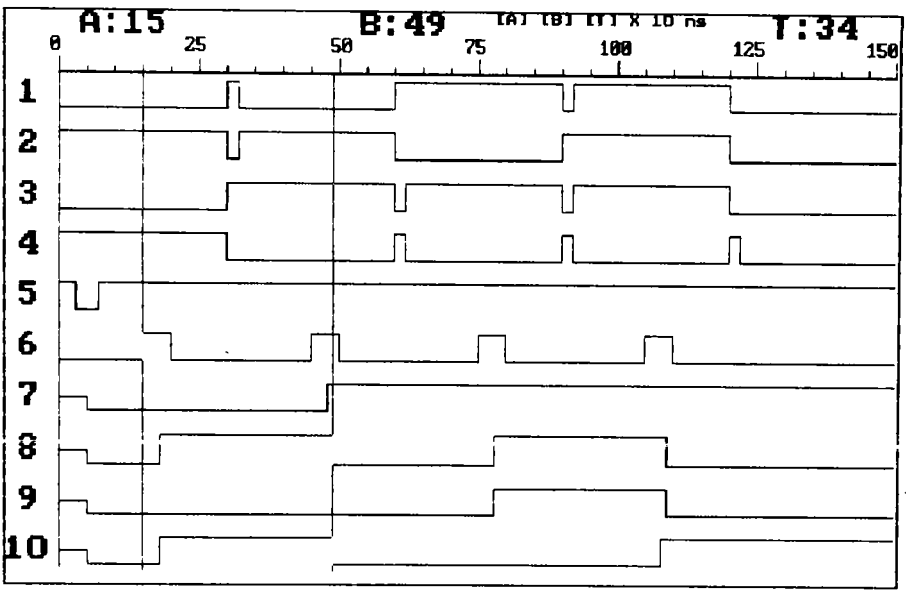

Fig. 8.25. Formarea semnăturii operanzilor A (operaţiile 1-4). 1-A_0, 2-A_1, 3-A_2, 4-A_3, 5-RESET, 6-TACTS, 7-$sigA_0$, 8-$sigA_1$, 9-$sigA_2$, 10-$sigA_3$.

În mod similar, formarea semnăturilor pentru operanzii B este arătată în fig. 8.26. Corectitudinea funcţionării se poate confirma prin compararea cu semnăturile operanzilor, sigA şi sigB, calculate teoretic conform fig. 8.21, respectiv fig. 8.22.

Se observă pe diagrame că modificarea valorilor operanzilor s-a considerat a avea loc din 300 ns în 300ns. De asemenea, se poate urmări, de exemplu, că formarea primei semnături după RESET are loc la 180 ns, iar a doua se formează la 490 ns.

Funcţionarea blocurilor ŞI/SAU se poate urmări prin cronogramele din fig. 8.27, unde se observă operanzii A şi B de

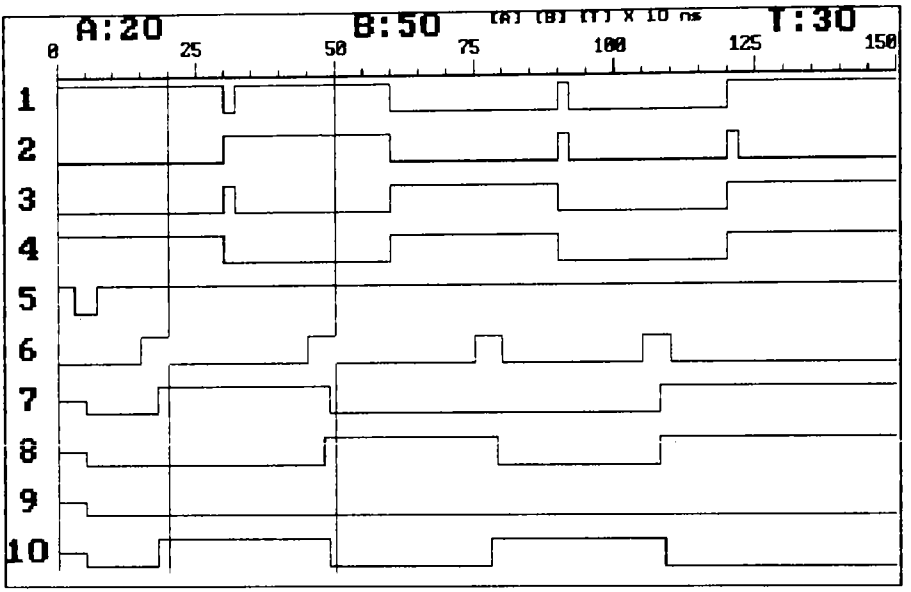

Fig. 8.26. Formarea semnăturii operanzilor B (operaţiile 1-4).
1-B_0, 2-B_1, 3-B_2, 4-B_3, 5-RESET,
6-TACTS, 7-sigB_0, 8-sigB_1, 9-sigB_2, 10-sigB_3.

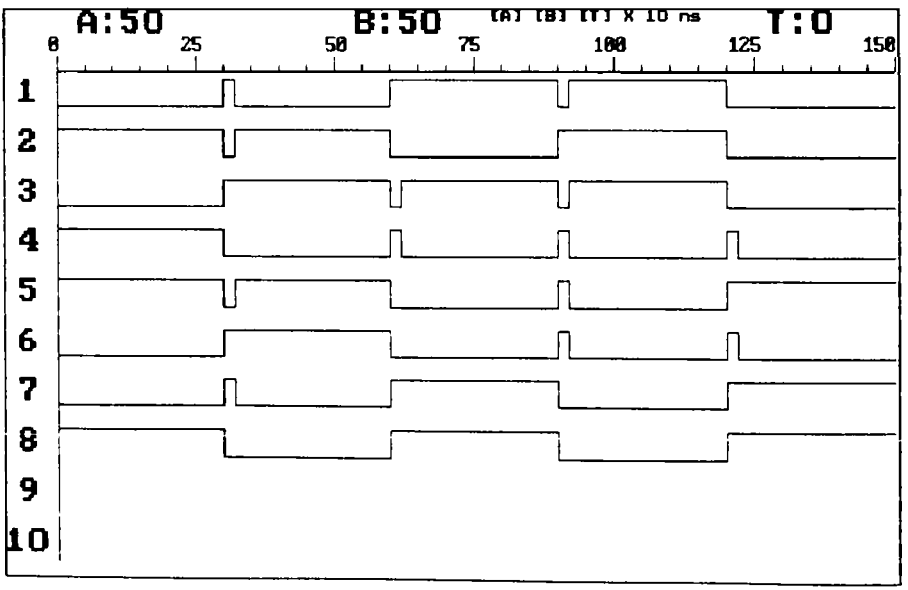

Fig. 8.27. Intrările blocurilor ŞI/SAU (operaţiile 1-4).
1-A_0, 2-A_1, 3-A_2, 4-A_3,
5-B_0, 6-B_1, 7-B_2, 8-B_3.

la intrare, respectiv fig. 8.28, în care se vede formarea caracteristicilor mutuale pentru operaţia SAU, H(A∨B) în blocul ŞI, respectiv ŞI, H(A∧B) în blocul SAU.

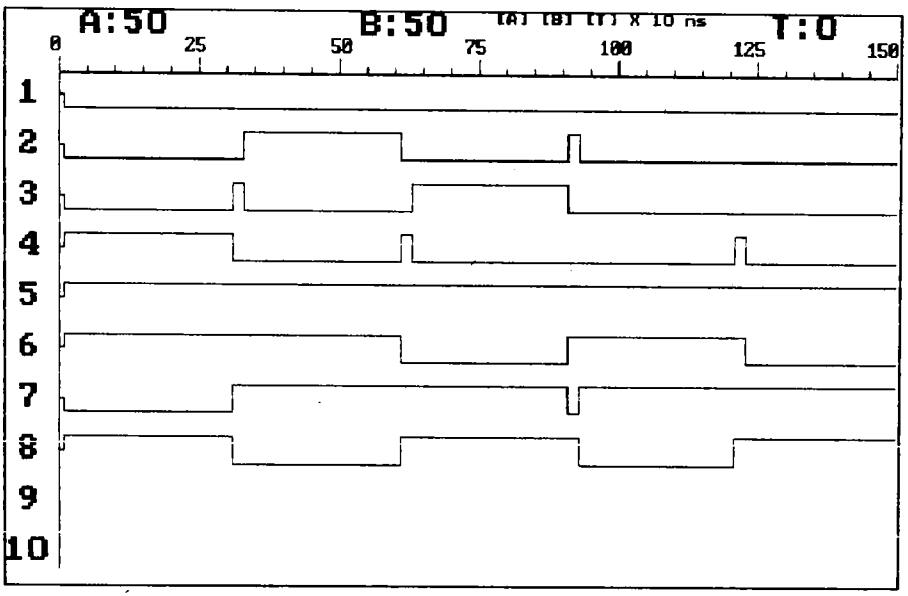

Fig. 8.28. Formarea caracteristicilor mutuale pentru operaţiile SAU/ŞI.
1-H_0(A∨B), 2-H_1(A∨B), 3-H_2(A∨B), 4-H_3(A∨B),
5-H_0(A∧B), 6-H_1(A∧B), 7-H_2(A∧B), 8-H_3(A∧B).

Executarea operaţiilor SAU-EXCLUSIV între semnăturile celor doi operanzi, în BLOC1-XOR, se observă cu ajutorul fig. 8.29, în care apar datele de intrare în bloc, precum şi fig. 8.30, unde se vizualizează semnalele la ieşire.

În continuare, vom urmări funcţionrea primului bloc multiplexor, MUX1. Astfel, în fig. 8.31 se reprezintă semnalele de selecţie, formate din codurile operaţiilor, precum şi semnalele de validare a porţilor tri-state de la ieşire, Z_0-Z_3.

În fig. 8.32 şi respectiv fig. 8.33 se observă semnalele de intrare precum şi cele validate la ieşirea multiplexorului, corespunzător derulării operaţiilor 1-4.

Pentru cel de al doilea multiplexor, MUX2, funcţionarea se poate urmări într-un mod similar. La intrările multiplexorului sînt prezente semnalele de la ieşirile blocurilor ŞI/SAU, (fig.

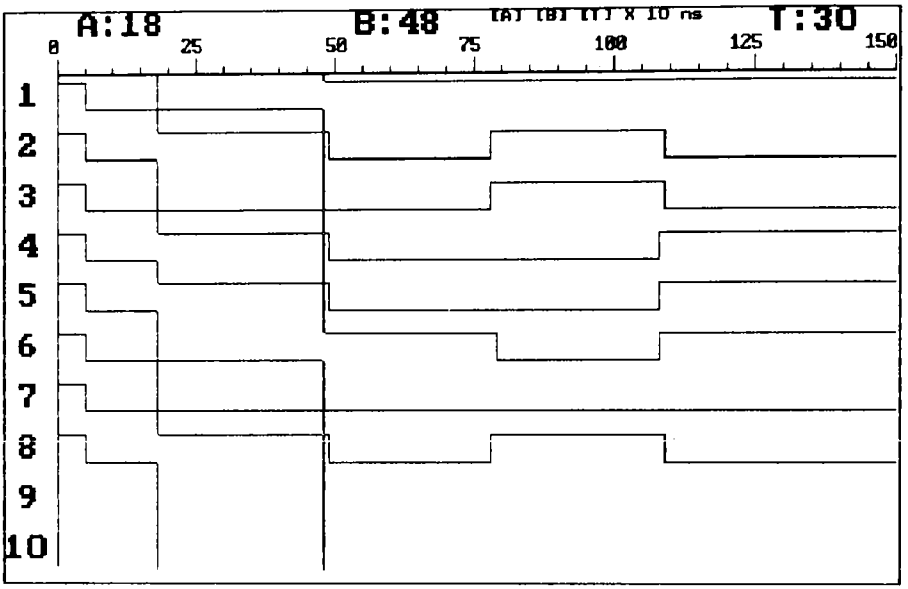

Fig. 8.29. Semnalele de intrare în BLOC1-XOR (operaţiile 1-4).
1-sigA_0, 2-sigA_1, 3-sigA_2, 4-sigA_3,
5-sigB_0, 6-sigB_1, 7-sigB_2, 8-sigB_3.

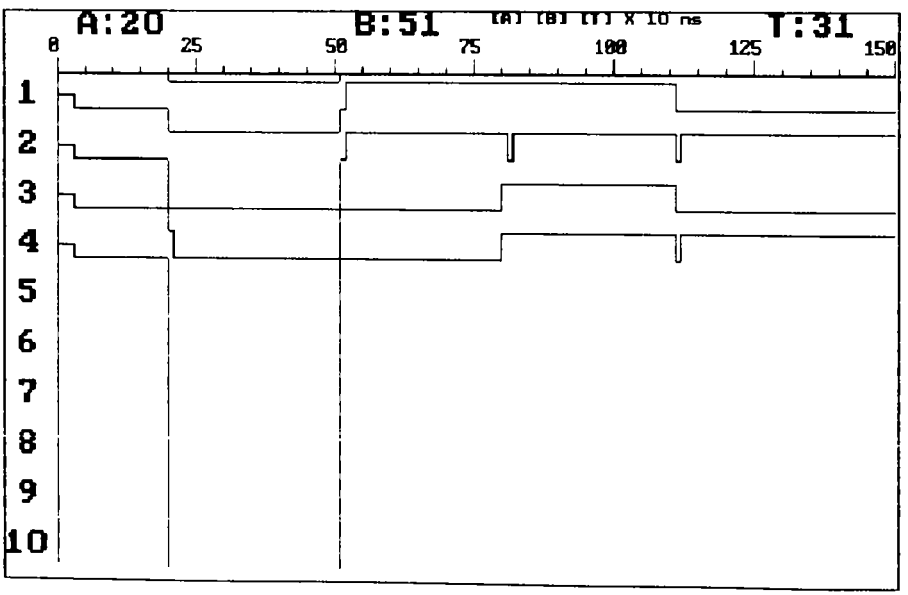

Fig. 8.30. Semnalele de ieşire din BLOC1-XOR (operaţiile 1-4).
1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1,
3-sigA_2⊕sigB_2, 4-sigA_3⊕sigB_3.

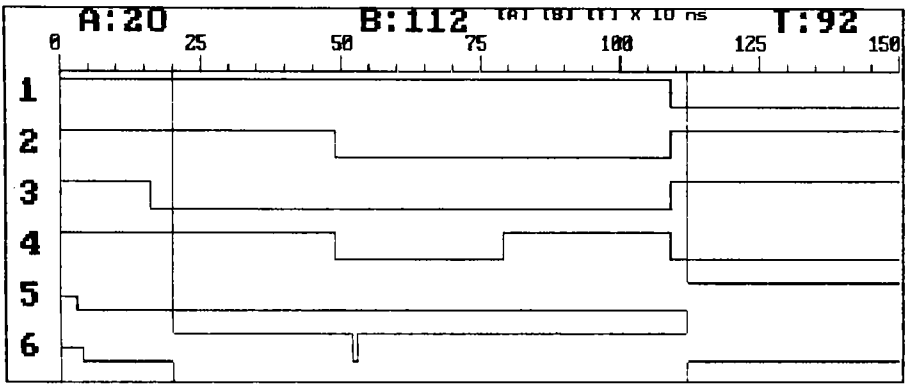

Fig. 8.31. Selecţia blocului MUX1 (operaţiile 1-4).
1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-validare sigA,
6-validare ieşire BLOC1-XOR.

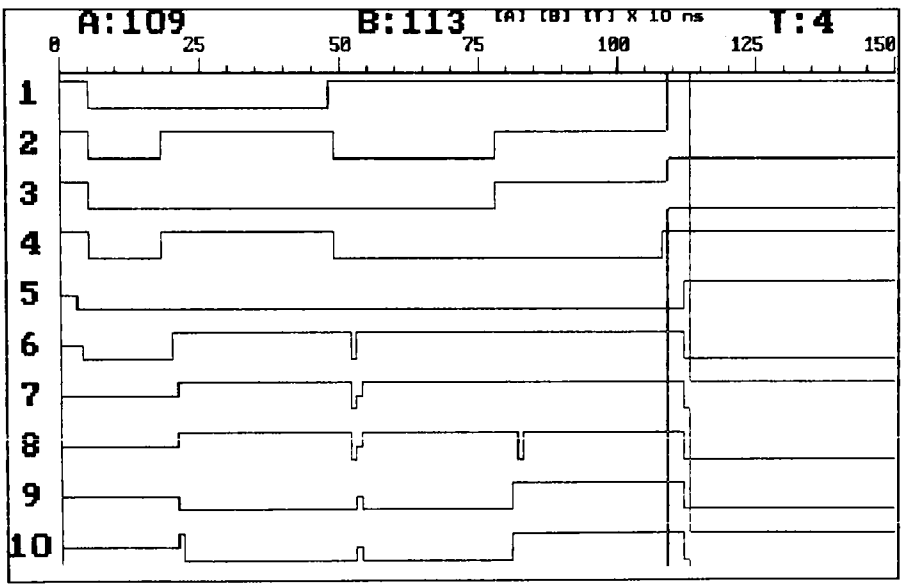

Fig. 8.32. Intrările şi ieşirile activate pentru sigA.
1-$sigA_0$, 2-$sigA_1$, 3-$sigA_2$, 4-$sigA_3$, 5-validare sigA,
6-validare ieşire BLOC1-XOR, 7-Z_0, 8-Z_1, 9-Z_2, 10-Z_3.

8.28). În fig. 8.34 se arată modul de activare a ieşirii pentru caracteristica mutuală a operaţiei ŞI.

Pe cronogramele reprezentate, se poate vedea activarea caracteristicii mutuale în intervalul de timp 40-330 ns.

Semnalele de comandă constituite din codurile operaţiilor,

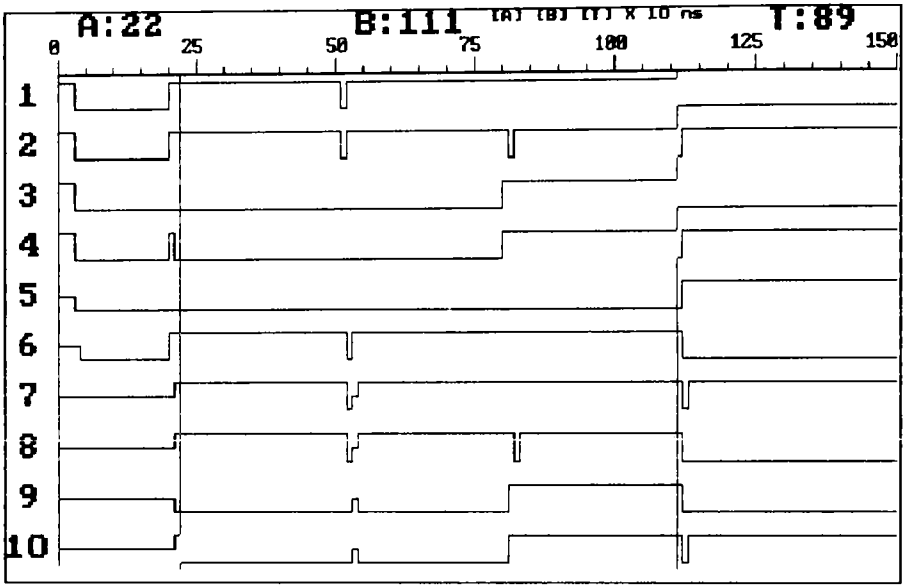

Fig. 8.33. Intrările şi ieşirile activate pentru BLOC1-XOR. 1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1, 3-sigA_2⊕sigB_2, 4-sigA_3⊕sigB_3, 5-validare sigA, 6-validare ieşire BLOC1-XOR, 7-Z_0, 8-Z_1, 9-Z_2, 10-Z_3.

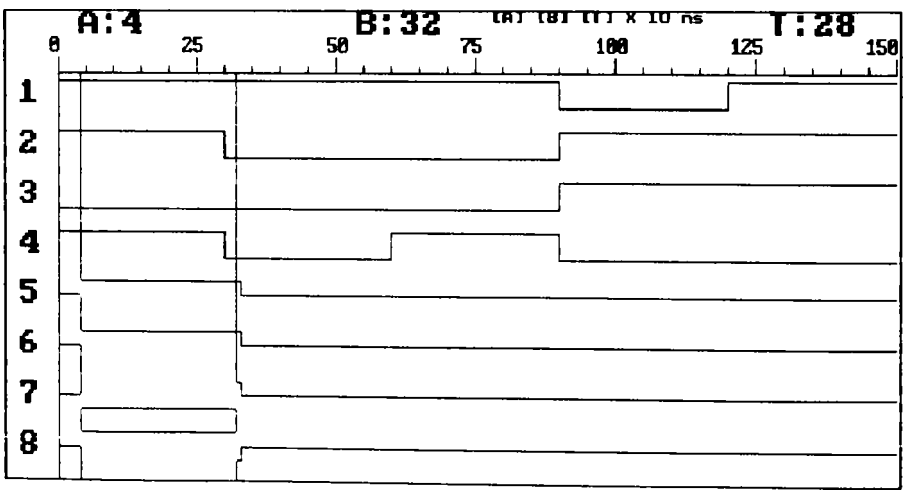

Fig. 8.34. Activarea caracteristicii mutuale a operaţiei ŞI. 1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-$H_0(A \wedge B)_{out}$, 6-$H_1(A \wedge B)_{out}$, 7-$H_2(A \wedge B)_{out}$, 8-$H_3(A \wedge B)_{out}$.

$I_0 \ldots I_3$, apar active cu un avans de 160 ns faţă de semnalele de comandă ale multiplexorului MUX1, în felul acesta compensîndu-se

întîrzierile în formarea caracteristicilor mutuale de control şi asigurîndu-se sincronismul datelor la intrarea în BLOC2-XOR.

În mod asemănător, în fig. 8.35 se arată activarea ieşirii multiplexorului pentru caracteristica mutuală a operaţiei SAU.

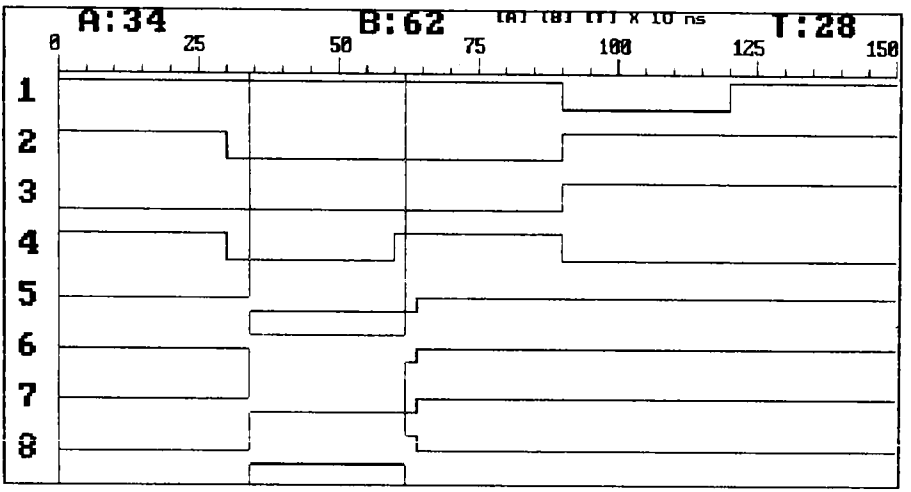

Fig. 8.35. Activarea caracteristicii mutuale a operaţiei SAU.
$1\text{-}I_0$, $2\text{-}I_1$, $3\text{-}I_2$, $4\text{-}I_3$,
$5\text{-}H_0(A \vee B)_{out}$, $6\text{-}H_1(A \vee B)_{out}$, $7\text{-}H_2(A \vee B)_{out}$, $8\text{-}H_3(A \vee B)_{out}$.

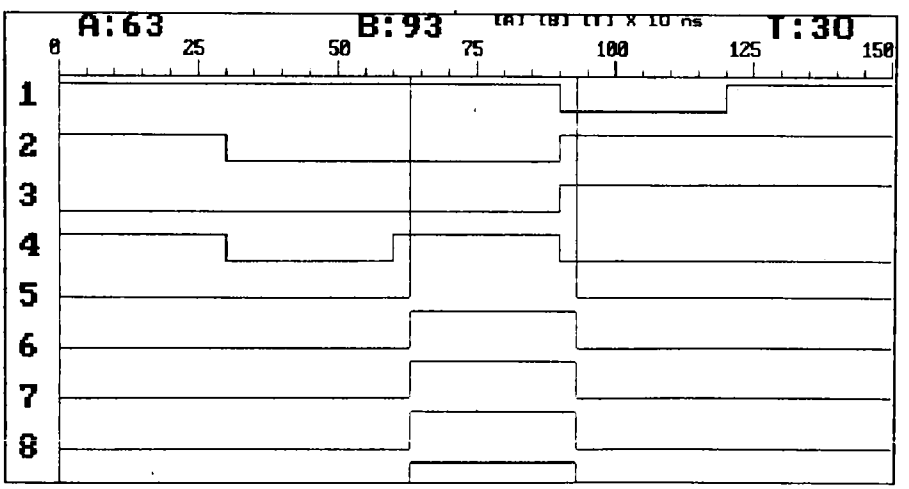

Fig. 8.36. Activarea caracteristicii mutuale a operaţiei SAU-EXCLUSIV
$1\text{-}I_0$, $2\text{-}I_1$, $3\text{-}I_2$, $4\text{-}I_3$,
$5\text{-}H_0(A \oplus B)_{out}$, $6\text{-}H_1(A \oplus B)_{out}$, $7\text{-}H_2(A \oplus B)_{out}$, $8\text{-}H_3(A \oplus B)_{out}$.

Caracteristica mutuală este prezentă la ieşire în intervalul 340-620 ns.

Pentru celelalte două operaţii, SAU-EXCLUSIV şi INVERSIUNE,modul de activare a ieşirii se observă în fig. 8.36 şi respectiv fig. 8.37.

Caracteristica mutuală a operaţiei SAU-EXCLUSIV este $\bar{\alpha}_n$=0000 şi apare activată în intervalul 630-930 ns.

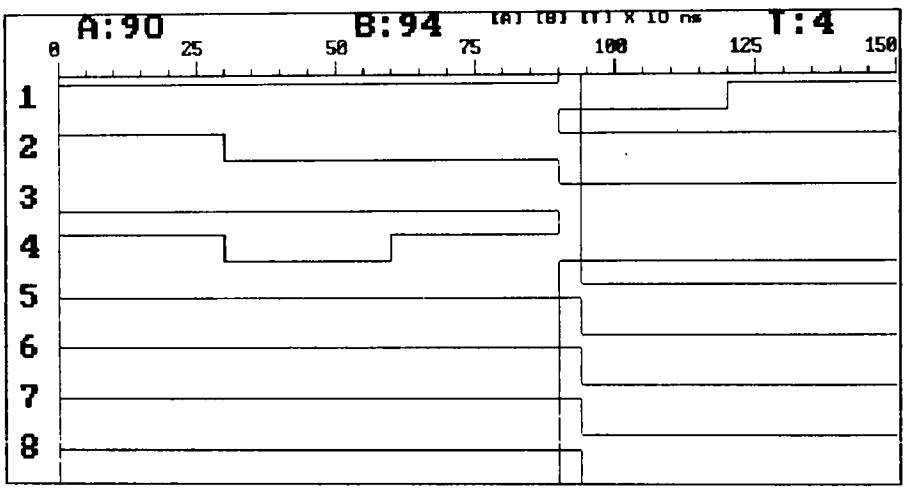

Fig. 8.37. Activarea caracteristicii mutuale a operaţiei de INVERSIUNE
$1\text{-}I_0$, $2\text{-}I_1$, $3\text{-}I_2$, $4\text{-}I_3$,
$5\text{-}H_0(\bar{A})_{out}$, $6\text{-}H_1(\bar{A})_{out}$, $7\text{-}H_2(\bar{A})_{out}$, $8\text{-}H_3(\bar{A})_{out}$.

La momentul de timp de 940 ns se poate observa validarea la ieşirea multiplexorului a caracteristicii mutuale α_n=1111.

Modalitatea de formare a semnăturilor caracteristicilor mutuale şi funcţionarea modulului SIGH, pentru primele patru operaţii a fost simulată conform cu fig. 8.38.

Comparînd rezultatul obţinut prin simulare, pentru sigH, cu cel calculat teoretic, conform cu fig. 8.23, se verifică corectitudinea fluxului informaţional pînă în acest punct.

În continuare, informaţia prezentă la intrarea celui de-al doilea bloc SAU-EXCLUSIV, BLOC2-XOR, apare conform cu fig. 8.39.

Informaţia de control, IC, pentru operaţiile 1-4, se formează la ieşire şi este prezentată în fig. 8.40.

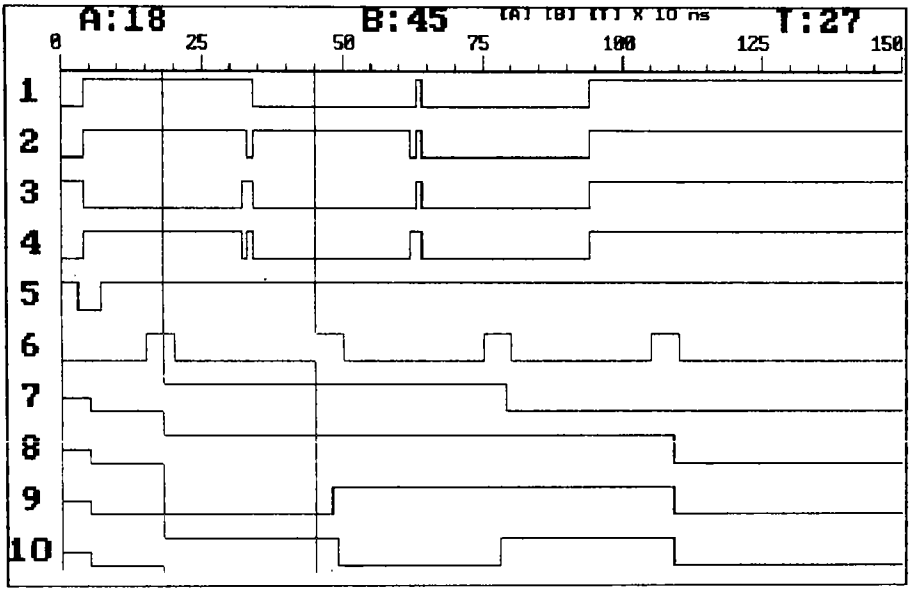

Fig. 8.38. Formarea semnăturii caracteristicilor mutuale H (operaţiile 1-4).
1-H_0, 2-H_1, 3-H_2, 4-H_3, 5-RESET, 6-TACTS, 7-sigH_0, 8-sigH_1, 9-sigH_2, 10-sigH_3.

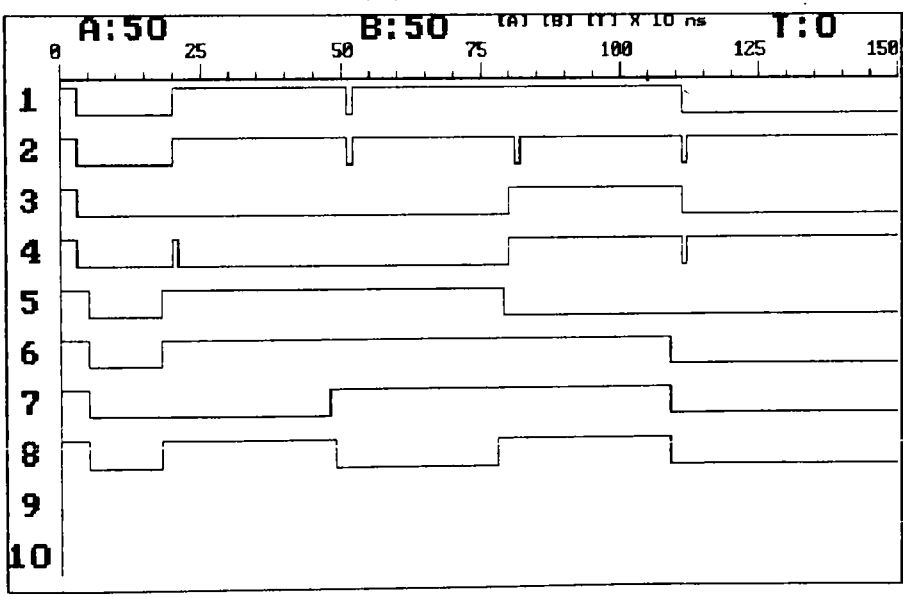

Fig. 8.39. Informaţia la intrarea BLOC2-XOR (operaţiile 1-4).
1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1, 3-sigA_2⊕sigB_2, 4-sigA_3⊕sigB_3, 5-sigH_0, 6-sigH_1, 7-sigH_2, 8-sigH_3.

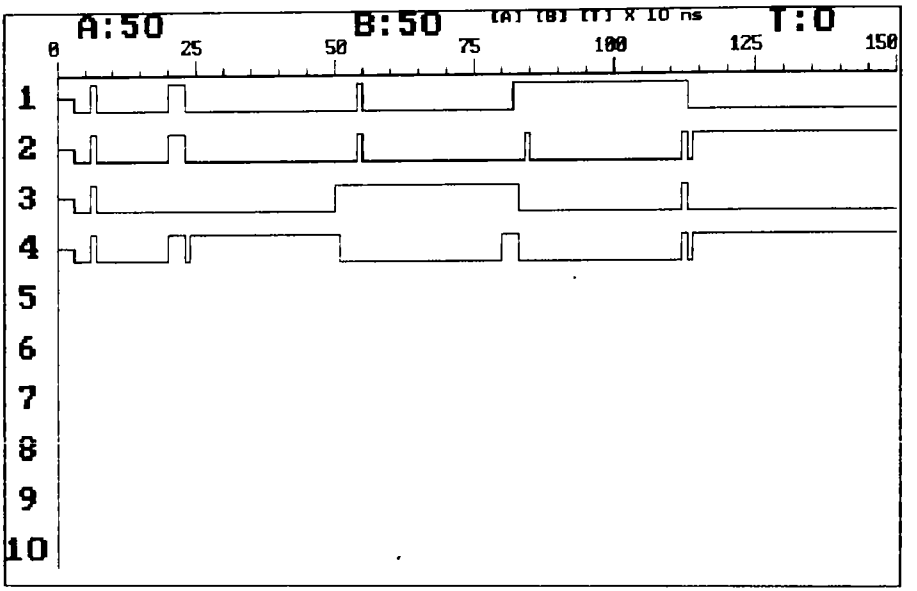

Fig. 8.40. Informaţia de control pentru operaţiile 1-4.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3.

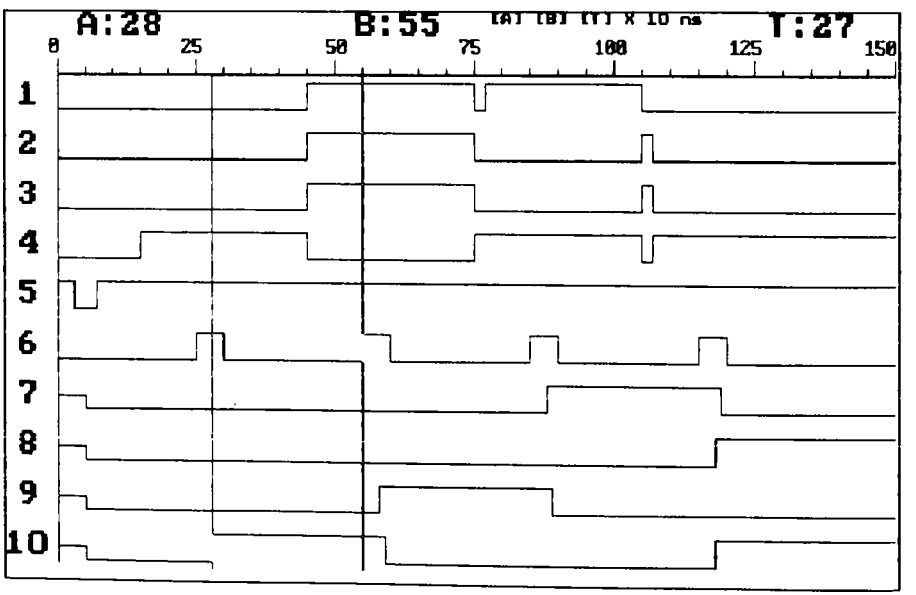

Fig. 8.41. Formarea semnăturii rezultatului din UAL (operaţiile 1-4).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

Pe de altă parte, la ieşirea UAL se formează semnătura rezultatului, în modulul sigC. Funcţionarea acestui modul este simulată în fig. 8.41. Corectitudinea funcţionării se poate constata prin comparare cu rezultatele teoretice din fig. 8.24.

În continuare, fiind disponibile ambele informaţii, sigC precum şi de control, IC, se activează modulul comparator COMP1. Funcţionarea modulului este exemplificată în fig. 8.42. Se

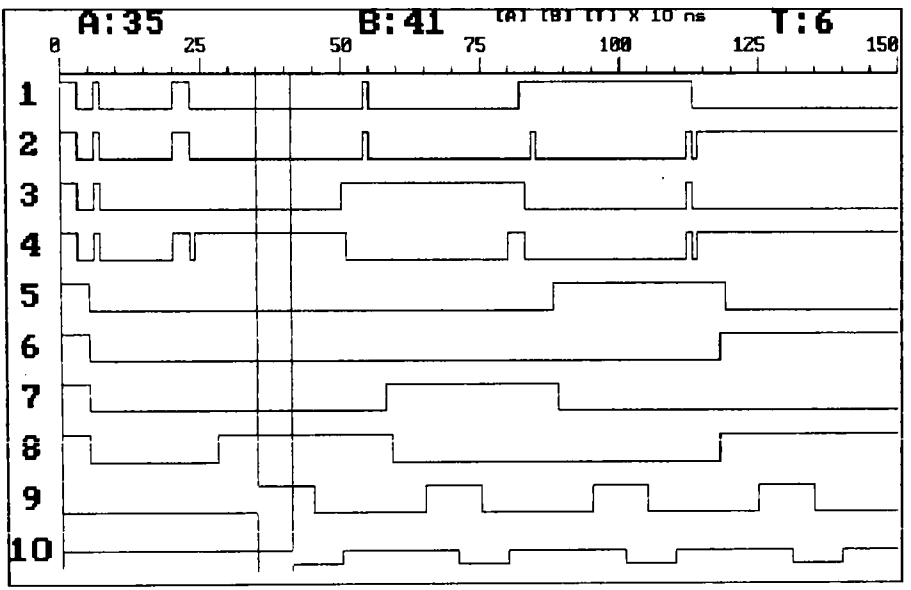

Fig. 8.42. Funcţionarea modulului comparator COMP1.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$,
6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

observă mărimile aplicate la intrarea comparatorului, precum şi rezultatul comparării, reprezentat prin semnalul EROARE. Validarea comparării este realizată prin semnalul VC (fig. 8.14). Starea logică "0" a semnalului EROARE, pe durata activării, indică faptul că operaţia s-a efectuat corect.

5. OPERAŢIA LOGICĂ DE COINCIDENŢĂ.

Operanzi: A=0000, B=1101.

Semnăturile operanzilor: sigA=0001, sigB=1011, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.17)): $H=\alpha_n=1111$ (la ieşirea blocului multiplexor MUX2).

Semnătura caracteristicii mutuale: sigH=1111, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1010.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=0101.

Rezultatul UAL: C=A⊙B=0010, (fig. 8.24).

Semnătura rezultatului: sigC=0101.

Ieşire COMP1: EROARE=IC⊕sigC=0.

6. OPERAŢIA LOGICĂ DEPLASARE DREAPTA.

Operanzi: A=1010, B=0000 (operandul B este considerat nul pentru asigurarea continuităţii fluxului informaţional la formarea semnăturii).

Semnăturile operanzilor: sigA=1000, sigB=0101, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.12)): H=1111 (în blocul DD/RD).

Semnătura caracteristicii mutuale: sigH=0010, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1101.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1111.

Rezultatul UAL: C=A*=0101, (fig. 8.24).

Semnătura rezultatului: sigC=1111.

Ieşire COMP1: EROARE=IC⊕sigC=0.

7. OPERAŢIA LOGICĂ DEPLASARE STÎNGA.

Operanzi: A=0110, B=0000 (operandul B este considerat nul pentru asigurarea continuităţii fluxului informaţional la formarea semnăturii).

Semnăturile operanzilor: sigA=0101, sigB=1010, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.11)): H=1010 (în blocul DS/RS).

Semnătura caracteristicii mutuale: sigH=1110, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1111.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=0001.

Rezultatul UAL: C=A*=1100, (fig. 8.24).

Semnătura rezultatului: sigC=0001, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

8. OPERAŢIA LOGICĂ DEPLASARE STÎNGA (cu pierderea informaţiei).

Operanzi: A=1101, B=0000 (operandul B este considerat nul pentru asigurarea continuităţii fluxului informaţional la formarea semnăturii).

Semnăturile operanzilor: sigA=0111, sigB=0111, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.11)): H=0111 (în blocul DS/RS).

Semnătura caracteristicii mutuale: sigH=1000, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=0000.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1000.

Rezultatul UAL: C=A*=1010, (fig. 8.24).

Semnătura rezultatului: sigC=1000, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

Controlul operaţiilor 5-8, prezentate mai sus, poate fi urmărit în continuare pe diagramele de timp rezultate în urma simulării funcţionării blocului de control. Astfel, în fig. 8.43 se arată modul de formare a semnăturii pentru operanzii A.

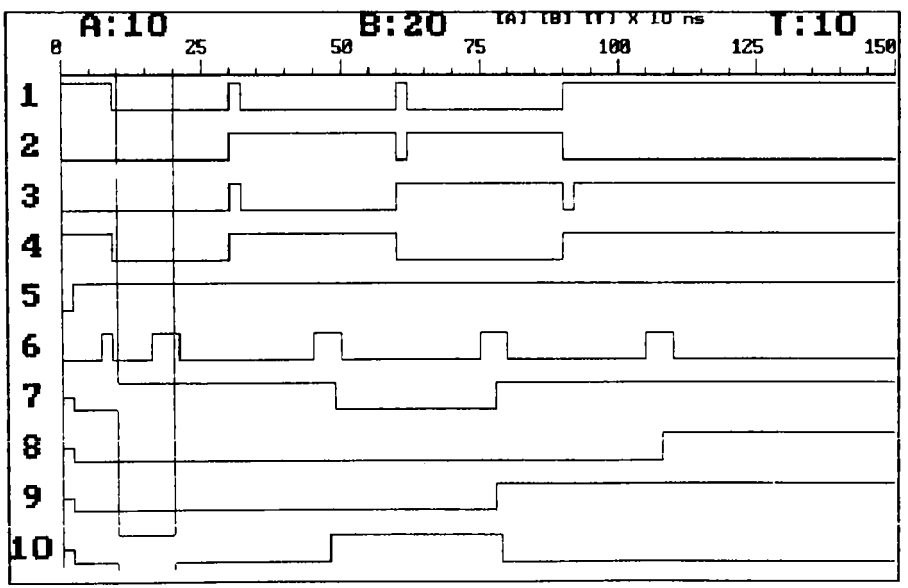

Fig. 8.43. Formarea semnăturii operanzilor A (operaţiile 5-8).
1-A_0, 2-A_1, 3-A_2, 4-A_3, 5-RESET,
6-TACTS, 7-sigA_0, 8-sigA_1, 9-sigA_2, 10-sigA_3.

Deoarece simulatorul a fost conceput pentru afişarea pe pagini de 1500 ns, la simularea controlului operaţiilor 5-8 şi respectiv a celorlalte în continuare, registrele de formare a semnăturilor au fost reiniţializate cu valoarea ultimei semnături calculate pentru operaţia precedentă. Astfel, printr-un tact suplimentar, care la funcţionarea neîntreruptă evident nu mai apare, s-a încărcat în registrul de formare a semnăturii sigA valoarea 1001, validă la momentul de 100 ns. După aplicarea impulsului de tact normal, semnătura se formează la momentul de 200 ns, adică faţă de origine la momentul de 1500+200 ns. Prin urmare, toate reprezentările, care vor apare la simularea controlului operaţiilor 5-8, vor avea ca referinţă momentul de timp de 1500 ns.

În mod similar, formarea semnăturilor pentru operanzii B este arătată în fig. 8.44. Corectitudinea funcţionării se poate confirma prin compararea cu semnăturile operanzilor, sigA şi sigB, calculate teoretic conform fig. 8.21, respectiv fig. 8.22.

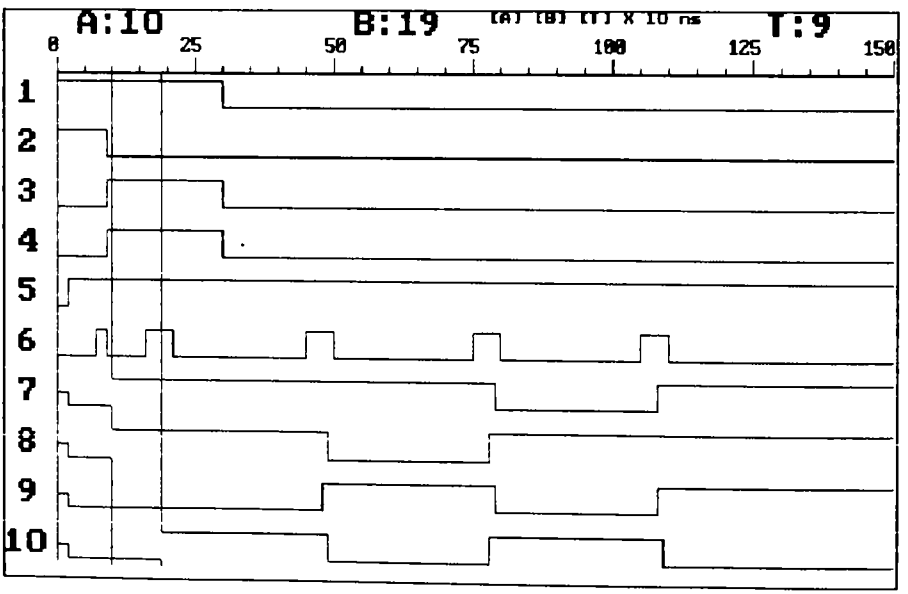

Fig. 8.44. Formarea semnăturii operanzilor B (operaţiile 5-8).
1-B_0, 2-B_1, 3-B_2, 4-B_3, 5-RESET,
6-TACTS, 7-$sigB_0$, 8-$sigB_1$, 9-$sigB_2$, 10-$sigB_3$.

Se observă pe diagramă, conform celor menţionate anterior,

iniţializarea registrului cu valoarea anterioară sigB=0011.

Executarea operaţiilor SAU-EXCLUSIV între semnăturile celor doi operanzi, în BLOC1-XOR, se observă cu ajutorul fig. 8.45, în care apar datele de intrare în bloc, precum şi fig. 8.46, unde se vizualizează semnalele la ieşire.

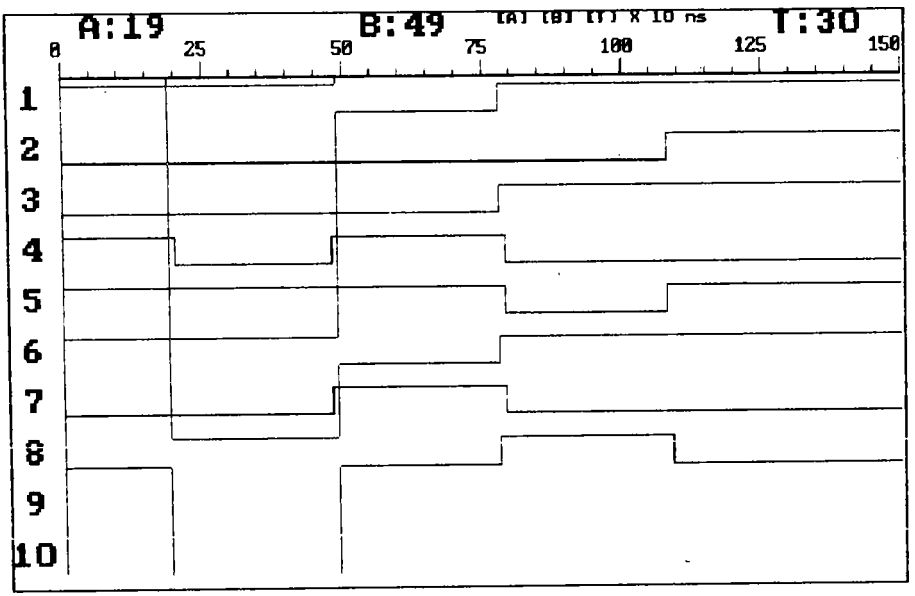

Fig. 8.45. Semnalele de intrare în BLOC1-XOR (operaţiile 5-8).
1-sigA_0, 2-sigA_1, 3-sigA_2, 4-sigA_3,
5-sigB_0, 6-sigB_1, 7-sigB_2, 8-sigB_3.

Formarea caracteristicii mutuale pentru operaţia de COINCIDENŢĂ nu implică probleme deosebite, fiind reprezentată de valoarea α_4=1111, la intrarea blocului MUX2.

În ceea ce priveşte formarea caracteristicilor mutuale ale operaţiilor de deplasare, aceasta se va urmări prin simularea funcţionării blocurilor DD/RD şi DS/RS. Astfel, în figura 8.47 sînt reprezentate codurile operaţiilor 5-8 şi semnalele rezultate pentru selecţia liniei H_3, în vederea formării caracteristicii mutuale pentru operaţia de DEPLASARE (v. fig. 8.9). Reamintim că blocul DD/RD este utilizat şi pentru operaţia de ROTAŢIE.

În fig. 8.48 rezultă formarea caraçteristicilor mutuale pentru operanzii A, în blocul DD/RD. Interes prezintă caracteristica validă pe intervalul 330-610 ns, corespunzătoare

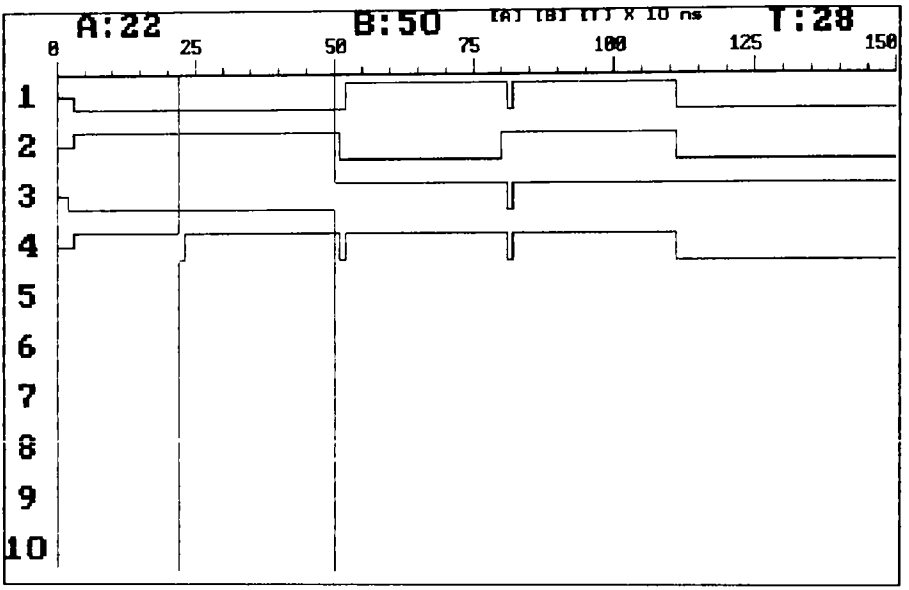

Fig. 8.46. Semnalele de ieşire din BLOC1-XOR (operaţiile 5-8).
1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1,
3-sigA_2⊕sigB_2, 4-sigA_3⊕sigB_3.

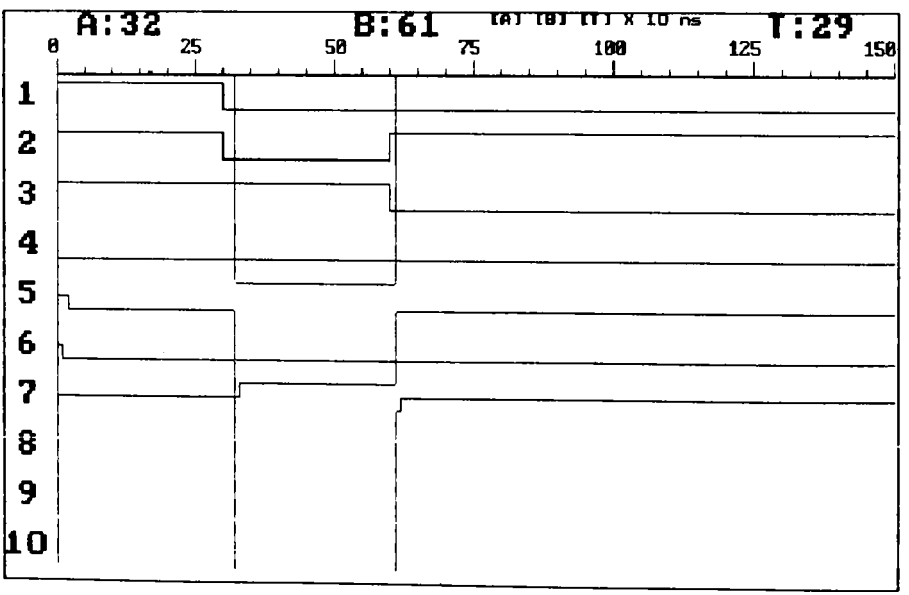

Fig. 8.47. Semnalele de intrare şi comandă ale blocului DD/RD (operaţiile 5-8).
1-I_0, 2-I_1, 3-I_2, 4-I_3,
5-sel.depl.H_3, 6-sel.rot.H_3, 7-H_3.

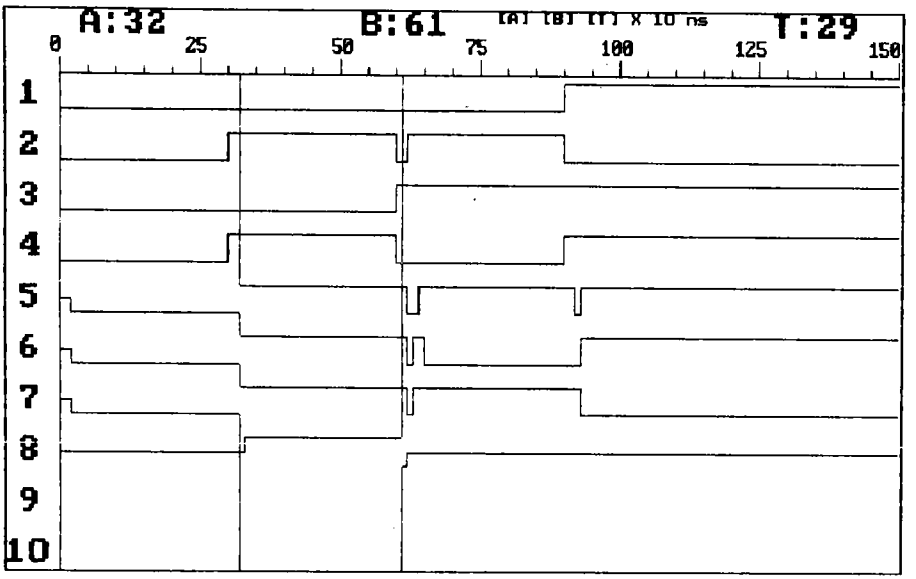

Fig. 8.48. Formarea caracteristicii mutuale pentru operaţia DEPLASARE DREAPTA.
$1-A_0$, $2-A_1$, $3-A_2$, $4-A_3$,
$5-H_0(A^*)$, $6-H_1(A^*)$, $7-H_2(A^*)$, $8-H_3(A^*)$.

operaţiei nr. 6 DEPLASARE DREAPTA.

În mod similar, pentru operaţiile nr. 7 şi 8 de DEPLASARE STÎNGA, în fig. 8.49 se arată codurile operaţiilor de la intrare şi semnalele de selecţie ale liniei H_0, în blocul DS/RS.

Formarea caracteristicilor mutuale ale celor două operaţii de DEPLASARE STÎNGA se observă în fig. 8.50. Astfel, caracteristica mutuală pentru operaţia nr. 7 este validă în intervalul 650-910 ns, iar pentru următoarea operaţie începînd de la 950 ns.

Funcţionarea blocului multiplexor MUX2 se va urmări în continuare. La intrările multiplexorului sînt prezente semnalele de la ieşirile blocurilor DD/RD şi DS/RS, (fig. 8.48 şi fig. 8.50). De asemenea, la intrarea multiplexorului este adusă şi mărimea α_4=1111. În fig. 8.51 se arată modul de activare a ieşirii pentru caracteristica mutuală a operaţiei COINCIDENŢĂ.

Pe cronogramele reprezentate, se poate vedea activarea caracteristicii mutuale în intervalul de timp 40-330 ns.

În mod asemănător, în fig. 8.52 se arată activarea ieşirii

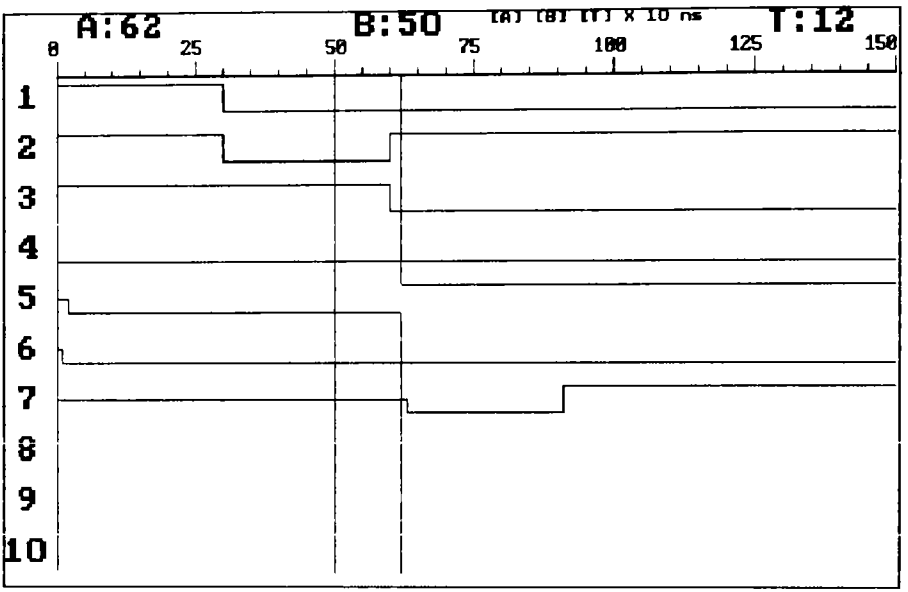

Fig. 8.49. Semnalele de intrare şi comandă ale blocului DS/RS (operaţiile 5-8). 1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-sel.depl.H_0, 6-sel.rot.H_0, 7-H_0.

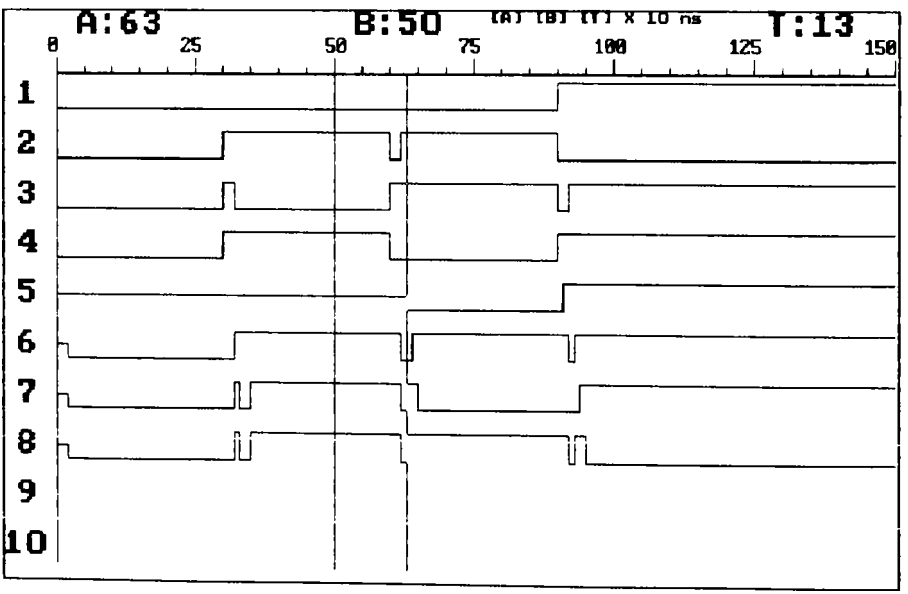

Fig. 8.50. Formarea caracteristicii mutuale pentru operaţiile de DEPLASARE STÎNGA.
1-A_0, 2-A_1, 3-A_2, 4-A_3,
5-$H_0(A^*)$, 6-$H_1(A^*)$, 7-$H_2(A^*)$, 8-$H_3(A^*)$.

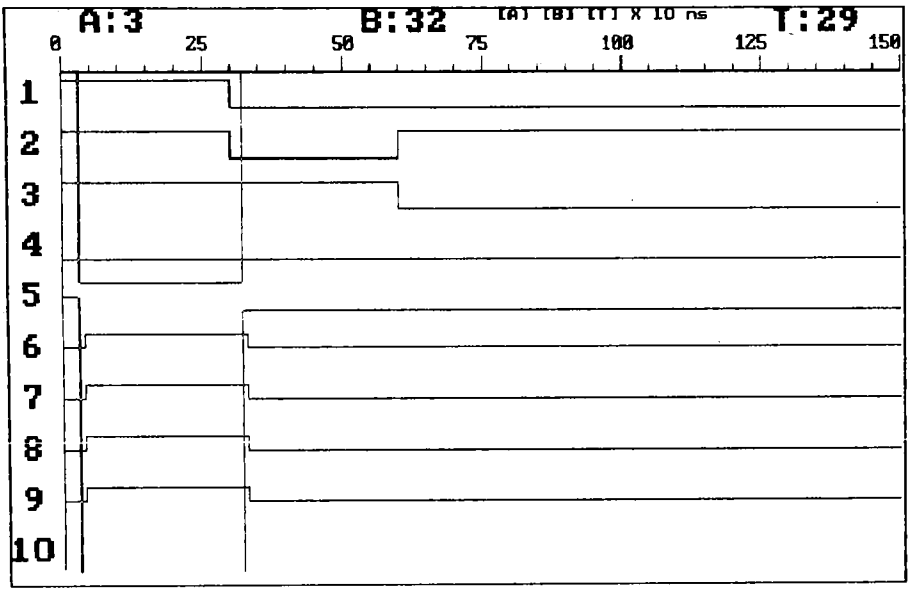

Fig. 8.51. Activarea caracteristicii mutuale a operaţiei COINCIDENŢĂ. 1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-sel. ⊙ 6-$H_0(A \odot B)_{out}$, 7-$H_1(A \odot B)_{out}$, 8-$H_2(A \odot B)_{out}$, 9-$H_3(A \odot B)_{out}$.

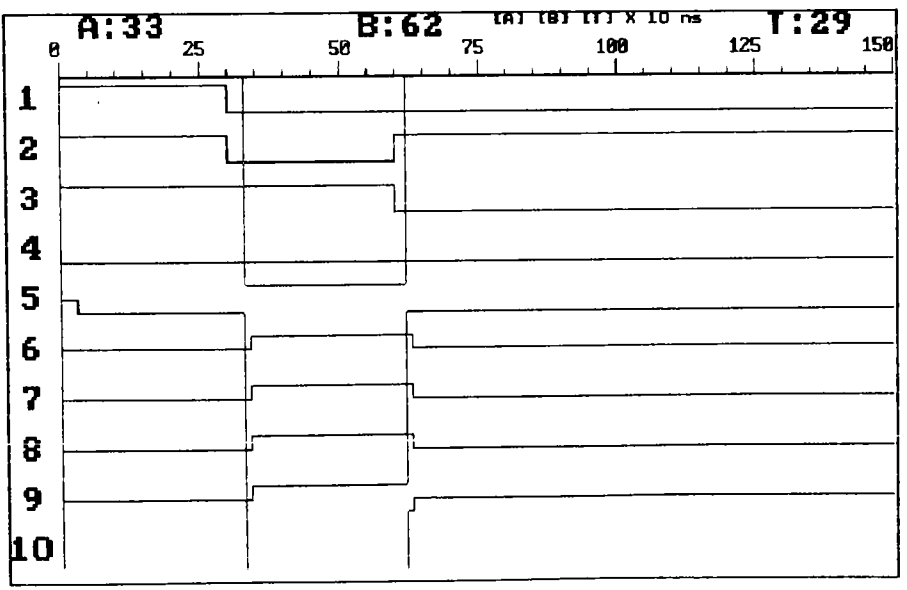

Fig. 8.52. Activarea caracteristicii mutuale a operaţiei DEPLASARE DREAPTA. 1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-sel.depl.dr. 6-$H_0(A^*)_{out}$, 7-$H_1(A^*)_{out}$, 8-$H_2(A^*)_{out}$, 9-$H_3(A^*)_{out}$.

multiplexorului pentru caracteristica mutuală a operaţiei DEPLASARE DREAPTA.

Caracteristica mutuală este prezentă la ieşire în intervalul 340-620 ns.

Pentru celelalte două operaţii de DEPLASARE STÎNGA, modul de activare a ieşirii se observă în fig. 8.53.

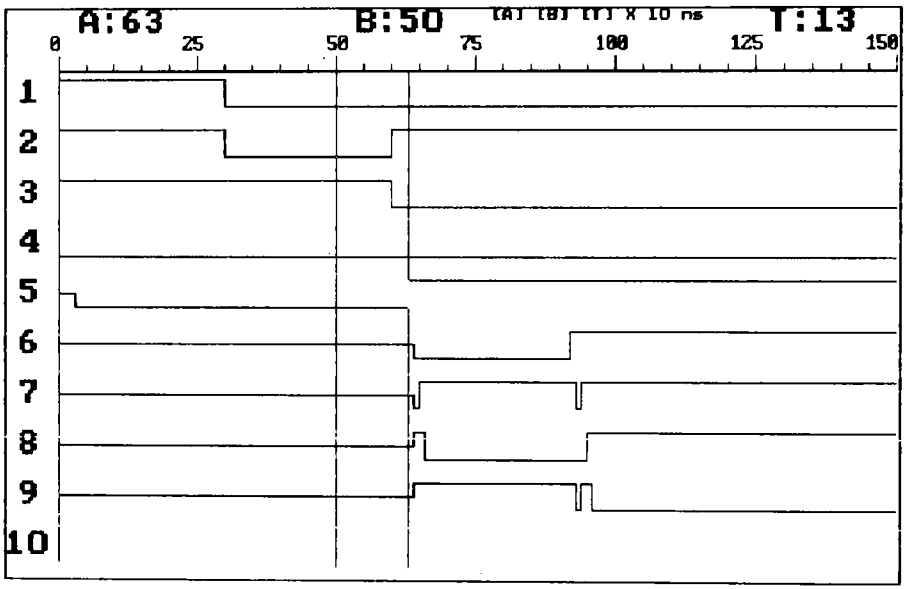

Fig. 8.53. Activarea caracteristicii mutuale a operaţiei de DEPLASARE STÎNGA.
1-I_0, 2-I_1, 3-I_2, 4-I_3, 5-sel.depl.stg.
6-$H_0(A^*)_{out}$, 7-$H_1(A^*)_{out}$, 8-$H_2(A^*)_{out}$, 9-$H_3(A^*)_{out}$.

Caracteristica mutuală a primei operaţii de DEPLASARE STÎNGA apare activată în intervalul 660-920 ns, iar pentru cea de-a doua operaţie începînd de la 960 ns.

Modalitatea de formare a semnăturilor caracteristicilor mutuale şi funcţionarea modulului SIGH, pentru operaţiile 5-8 a fost simulată conform cu fig. 8.54. Registrul a fost iniţializat cu valoarea sigH=0000, calculată pentru operaţia nr. 4.

Comparînd rezultatul obţinut prin simulare, pentru sigH, cu cel calculat teoretic, conform cu fig. 8.23, se verifică corectitudinea fluxului informaţional pînă în acest punct.

În continuare, informaţia prezentă la intrarea celui de-al

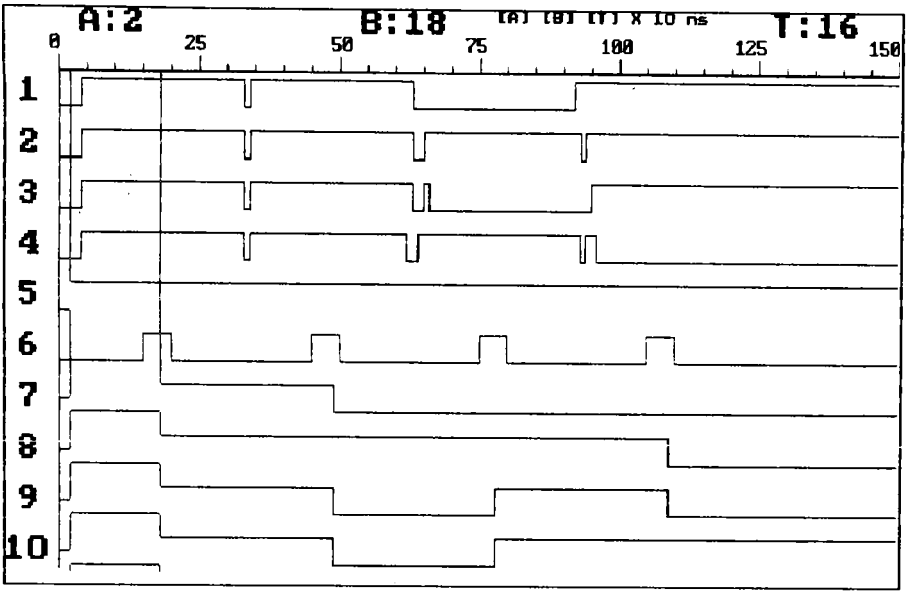

Fig. 8.54. Formarea semnăturii caracteristicilor mutuale H (operaţiile 5-8).
1-H_0, 2-H_1, 3-H_2, 4-H_3, 5-RESET,
6-TACTS, 7-sigH_0, 8-sigH_1, 9-sigH_2, 10-sigH_3.

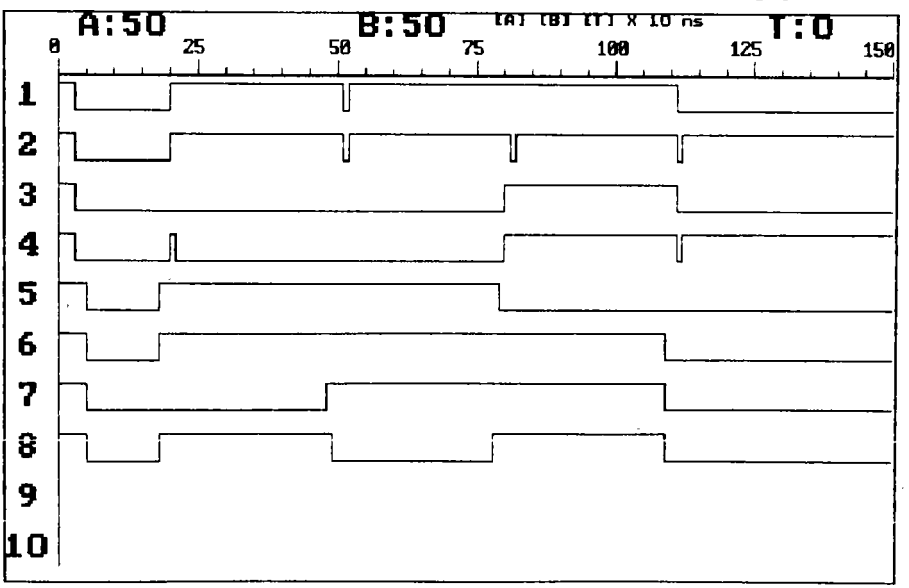

Fig. 8.55. Informaţia la intrarea BLOC2-XOR (operaţiile 5-8).
1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1, 3-sigA_2⊕sigB_2,
4-sigA_3⊕sigB_3, 5-sigH_0, 6-sigH_1, 7-sigH_2, 8-sigH_3.

doilea bloc SAU-EXCLUSIV, BLOC2-XOR, apare conform cu fig. 8.55.

Informaţia de control, IC, pentru operaţiile 5-8, se formează la ieşire şi este prezentată în fig. 8.56.

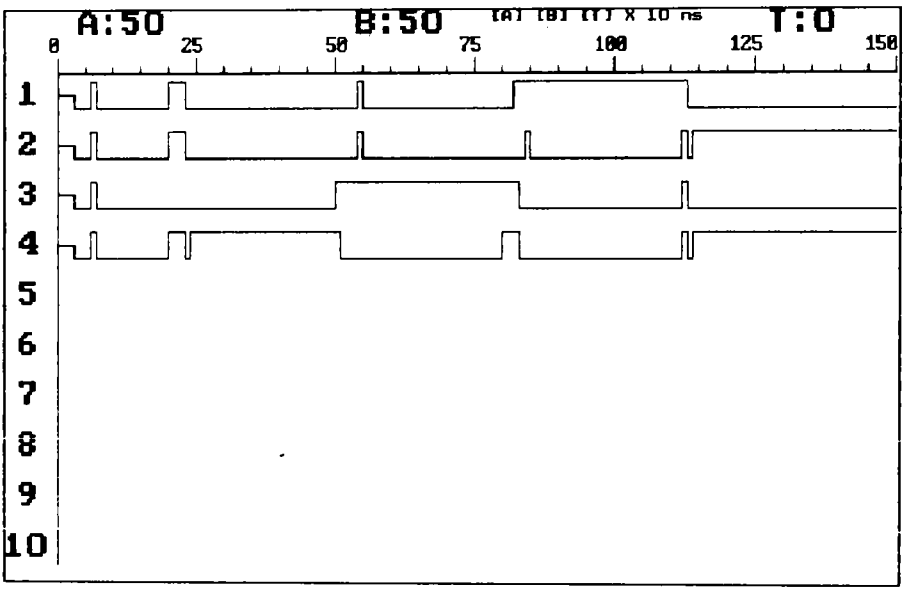

Fig. 8.56. Informaţia de control pentru operaţiile 5-8.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3.

La ieşirea UAL se formează semnătura rezultatului, în modulul sigC. Funcţionarea acestui modul este simulată în fig. 8.57. Corectitudinea funcţionării se poate constata prin comparare cu rezultatele teoretice din fig. 8.24.

Ca şi în cazul formării semnăturilor paralel în celelalte registre, pentru asigurarea continuităţii simulării, registrul de formare a sigC a fost iniţializat cu ultima semnătură formată, cea corespunzătoare operaţiei nr. 4.

În continuare, fiind disponibile ambele informaţii, sigC precum şi de control, IC, se activează modulul comparator COMP1. Funcţionarea modulului este exemplificată în fig. 8.58. Se observă mărimile aplicate la intrarea comparatorului, precum şi rezultatul comparării, reprezentat prin semnalul EROARE. Validarea comparării este realizată prin semnalul VC (fig. 8.14). Starea logică "0" a semnalului EROARE, pe durata activării,

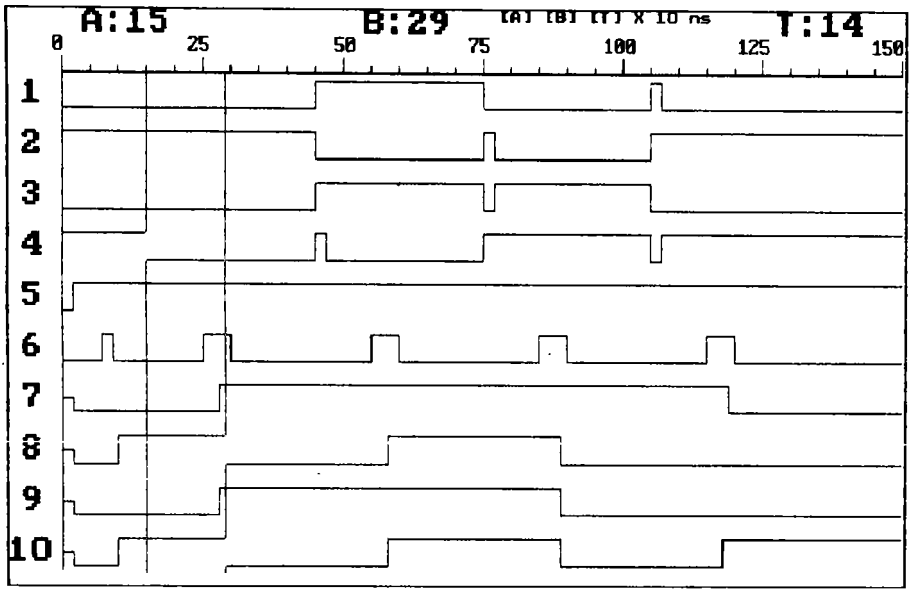

Fig. 8.57. Formarea semnăturii rezultatului din UAL (operaţiile 5-8).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

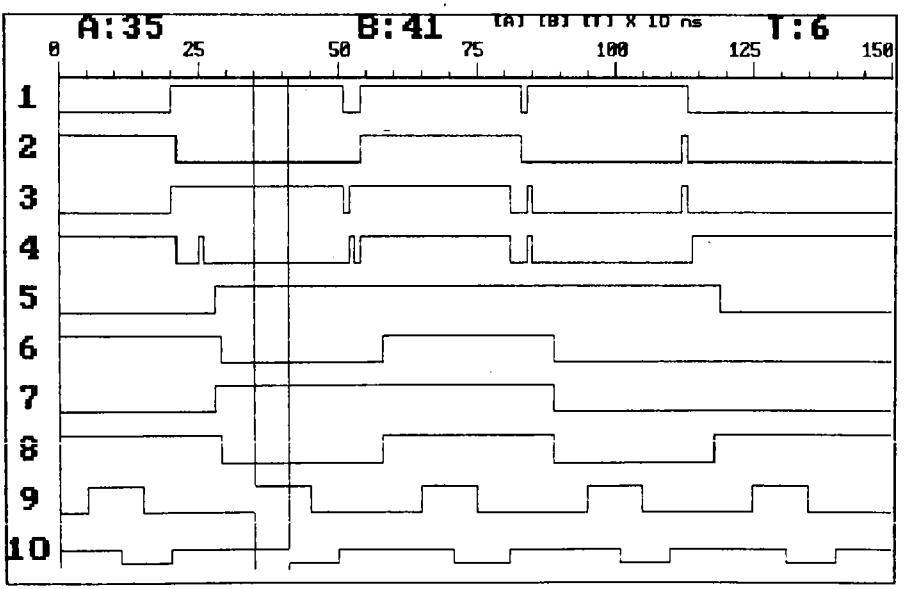

Fig. 8.58. Funcţionarea modulului comparator COMP1.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

indică faptul că operaţia s-a efectuat corect.

9. OPERAŢIA LOGICĂ DE ROTAŢIE DREAPTA.

Operanzi: A=1101, B=0000 (operandul B este considerat nul pentru asigurarea continuităţii informaţiei la formarea semnăturii).

Semnăturile operanzilor: sigA=0011, sigB=1110, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.16)): H=0011 (în blocul DD/RD).

Semnătura caracteristicii mutuale: sigH=0000, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1101.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1101.

Rezultatul UAL: C=Å'=1110, (fig. 8.24).

Semnătura rezultatului: sigC=1101.

Ieşire COMP1: EROARE=IC⊕sigC=0.

10. OPERAŢIA LOGICĂ DE ROTAŢIE STÎNGA.

Operanzi: A=1011, B=0000 (operandul B este considerat nul pentru asigurarea continuităţii informaţiei la formarea semnăturii).

Semnăturile operanzilor: sigA=1101, sigB=1111, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.15)): H=1100 (în blocul DS/RS).

Semnătura caracteristicii mutuale: sigH=1100, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=0010.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1110.

Rezultatul UAL: C=Å'=0111, (fig. 8.24).

Semnătura rezultatului: sigC=1110.

Ieşire COMP1: EROARE=IC⊕sigC=0.

11. OPERAŢIA ARITMETICĂ DE ADUNARE (fără depăşire).

Operanzi: A=0111 (+7), B=1011 (-5). Operanzii sînt consideraţi în reprezentarea prin complement faţă de 2.

Semnăturile operanzilor: sigA=1110, sigB=0110, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.33)): H=1110 (în blocul AD/SC).

Semnătura caracteristicii mutuale: sigH=0101, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1000.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=1101.

Rezultatul UAL: C=A+B=0010 (+2), (fig. 8.24).

Semnătura rezultatului: sigC=1101, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

12. OPERAŢIA ARITMETICĂ DE ADUNARE (cu depăşire).

Operanzi: A=0110 (+6), B=0011 (+3), în complement faţă de 2.

Semnăturile operanzilor: sigA=1001, sigB=1111, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.33)): H=1100 (în blocul AD/SC).

Semnătura caracteristicii mutuale: sigH=0110, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=0110.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=0000.

Rezultatul UAL: C=A+B=1001 (+9, dar reprezentat negativ; fig. 8.24).

Semnătura rezultatului: sigC=0000, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

13. OPERAŢIA ARITMETICĂ DE SCĂDERE.

Operanzi: A=1100 (-4), B=1010 (-6), în complement faţă de 2.

Semnăturile operanzilor: sigA=1101, sigB=0111, (fig. 8.21, 8.22).

Caracteristica mutuală H (relaţia (6.35)): H=0100 (în blocul AD/SC).

Semnătura caracteristicii mutuale: sigH=1000, (fig. 8.23).

Ieşire BLOC1-XOR: sigA⊕sigB=1010.

Informaţia de control la ieşirea BLOC2-XOR: IC=(sigA⊕sigB)⊕sigH=0010.

Rezultatul UAL: C=A-B=0010 (+2).

Semnătura rezultatului: sigC=0010, (fig. 8.24).

Ieşire COMP1: EROARE=IC⊕sigC=0.

Controlul operaţiilor 9-13, prezentate mai sus, poate fi urmărit, de asemenea, pe diagramele de timp rezultate în urma simulării funcţionării blocului de control. Astfel, în fig. 8.59 se arată modul de formare a semnăturii pentru operanzii A.

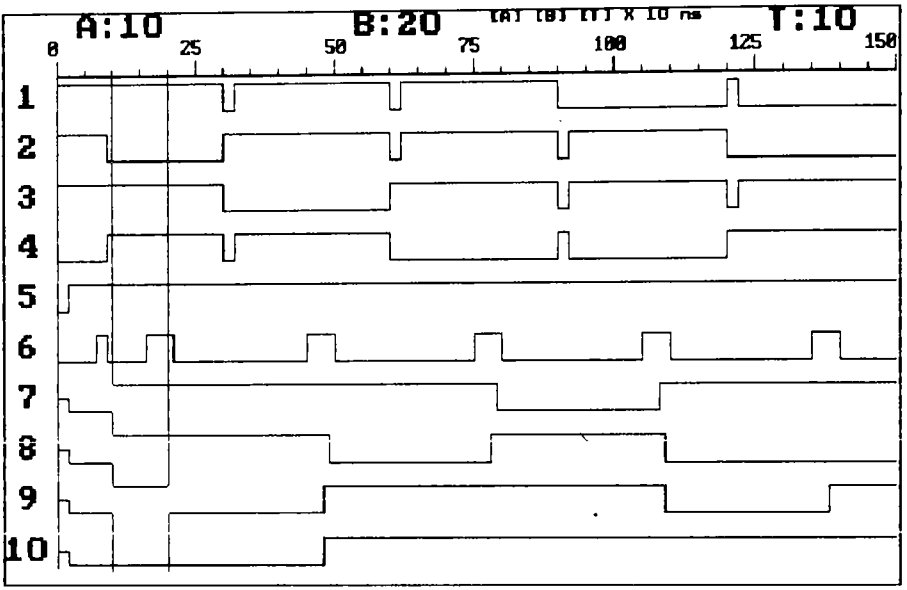

Fig. 8.59. Formarea semnăturii operanzilor A (operaţiile 9-13). 1-A_0, 2-A_1, 3-A_2, 4-A_3, 5-RESET, 6-TACTS, 7-sigA_0, 8-sigA_1, 9-sigA_2, 10-sigA_3.

Registrele de formare a semnăturilor au fost reiniţializate cu valoarea ultimei semnături calculate pentru operaţia precedentă, din motivele explicate anterior. Astfel, s-a încărcat în registrul de formare a semnăturii sigA valoarea 0111, validă la momentul de 100 ns. După aplicarea impulsului de tact normal, semnătura se formează la momentul de 200 ns, adică faţă de origine la momentul de 3000+200 ns. Prin urmare, toate reprezentările, care vor apare la simularea controlului operaţiilor 9-13, vor avea ca referinţă momentul de timp de 3000 ns.

În mod similar, formarea semnăturilor pentru operanzii B este arătată în fig. 8.60. Corectitudinea funcţionării se poate confirma prin compararea cu semnăturile operanzilor, sigA şi sigB, calculate teoretic conform fig. 8.21, respectiv fig. 8.22.

Se observă pe diagramă, conform celor menţionate anterior, iniţializarea registrului cu valoarea anterioară sigB=0111.

Executarea operaţiilor SAU-EXCLUSIV între semnăturile celor doi operanzi, în BLOC1-XOR, se observă cu ajutorul fig. 8.61, unde se vizualizează semnalele la ieşire.

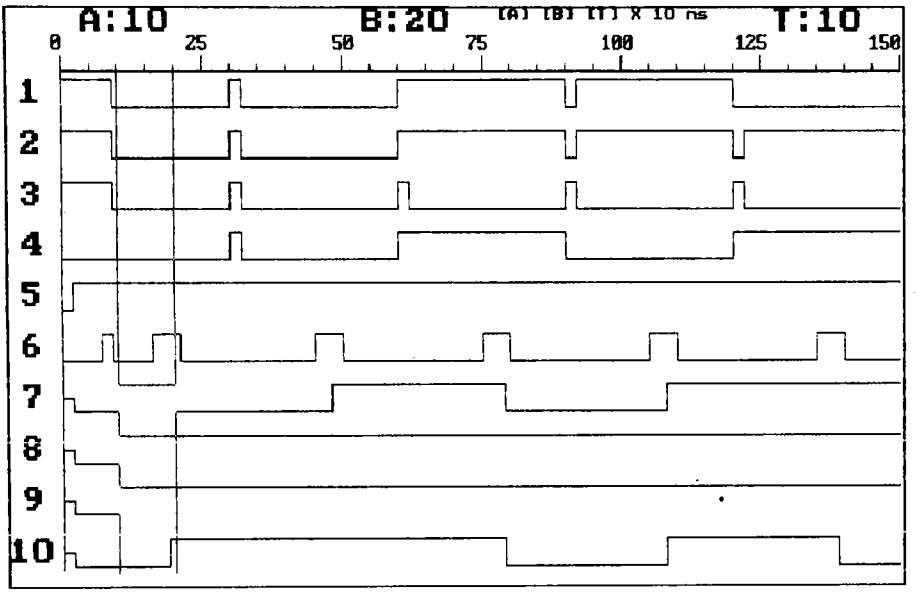

Fig. 8.60. Formarea semnăturii operanzilor B (operaţiile 9-13).
1-B_0, 2-B_1, 3-B_2, 4-B_3, 5-RESET,
6-TACTS, 7-sigB_0, 8-sigB_1, 9-sigB_2, 10-sigB_3.

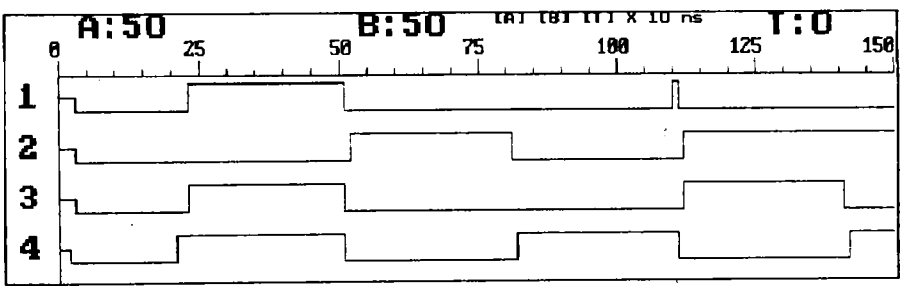

Fig. 8.61. Semnalele de ieşire din BLOC1-XOR (operaţiile 9-13).
1-sigA_0⊕sigB_0, 2-sigA_1⊕sigB_1,
3-sigA_2⊕sigB_2, 4-sigA_3⊕sigB_3.

Formarea caracteristicilor mutuale ale operaţiilor de rotaţie, se va urmări prin simularea funcţionării blocurilor DD/RD şi DS/RS. Astfel, în figura 8.62 sînt reprezentate valorile operanzilor A, semnalele rezultate pentru selecţia liniei H_3, în vederea formării caracteristicii mutuale pentru operaţia de ROTAŢIE DREAPTA (v. fig. 8.9), precum şi cuvîntul rezultat al caracteristicii de control. Interes prezintă caracteristica

validă pe intervalul 40-320 ns, corespunzătoare operaţiei nr. 9 ROTAŢIE DREAPTA.

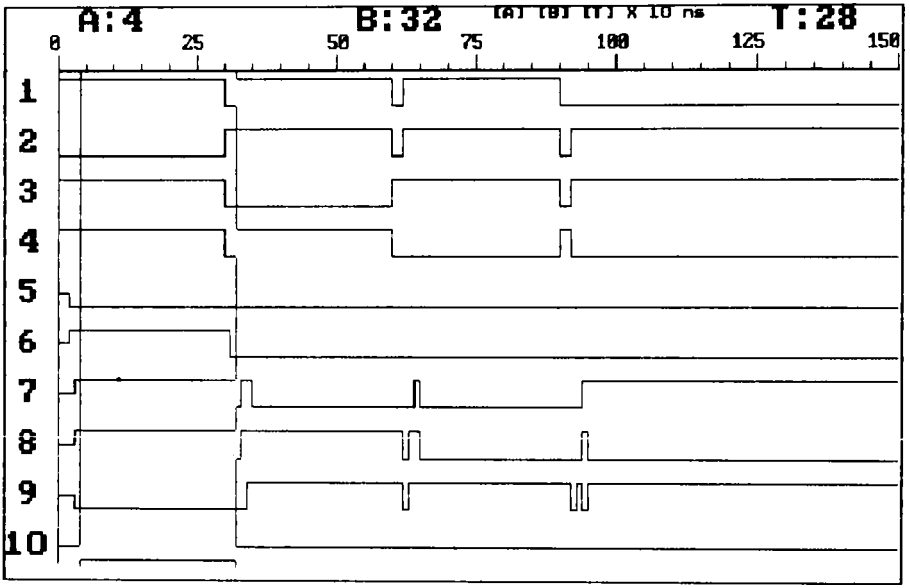

Fig. 8.62. Formarea caracteristicii mutuale pentru operaţia DEPLASARE DREAPTA.
1-A_0, 2-A_1, 3-A_2, 4-A_3, 5-sel.depl.H_3, 6-sel.rot.H_3, 7-H_0(Å*), 8-H_1(Å*), 9-H_2(Å*), 10-H_3(Å*).

În mod similar, pentru operaţia nr. 10 de ROTAŢIE STÎNGA, în fig. 8.63 se arată valorile operanzilor A de la intrare, semnalele de selecţie ale liniei H_0, în blocul DS/RS (fig. 8.10) şi caracteristica formată la ieşire. Astfel, caracteristica mutuală pentru operaţia nr. 10 este validă în intervalul 360-620 ns.

Carcteristicile mutuale de control pentru operaţiile aritmetice se formează în blocul AD/SC (fig. 8.11). Mărimile de intrare, constituite de valorile operanzilor A şi B şi de codurile operaţiilor sînt reprezentate în fig. 8.64.

Ieşirile blocului AD/SC sînt vizualizate în fig. 8.65. Linia H_0 nu a mai fost reprezentată, aceasta fiind întotdeauna de valoarea logică "0". Blocul fiind utilizat atît pentru adunare cît şi pentru scădere, selecţia funcţionării pentru o anumită operaţie se face prin două semnale aferente (fig. 8.11). Aceste

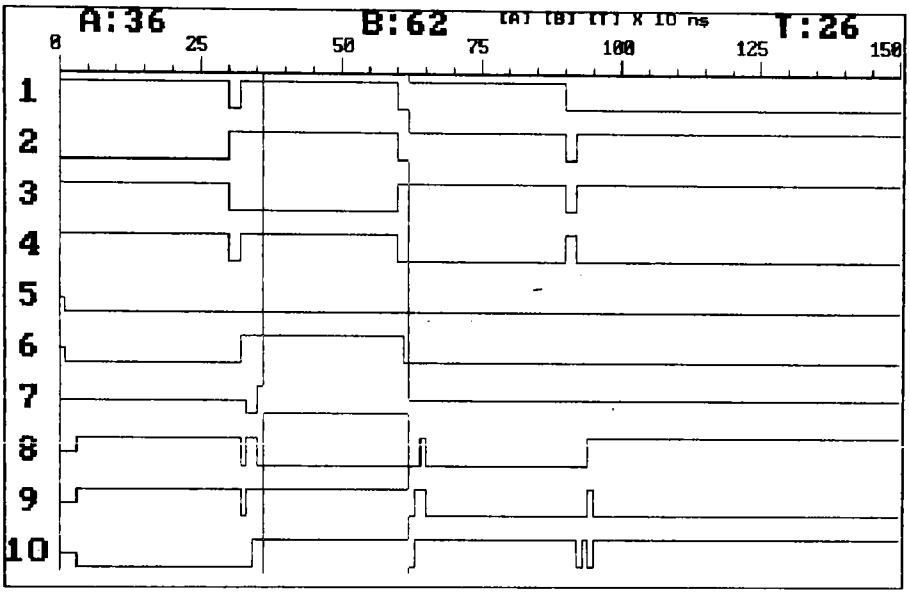

Fig. 8.63. Formarea caracteristicii mutuale pentru operaţia de ROTAŢIE STÎNGA.
1-A_0, 2-A_1, 3-A_2, 4-A_3, 5-sel.depl.H_0, 6-sel.rot.H_0, 7-H_0(Å$^+$), 8-H_1(Å$^+$), 9-H_2(Å$^+$), 10-H_3(Å$^+$).

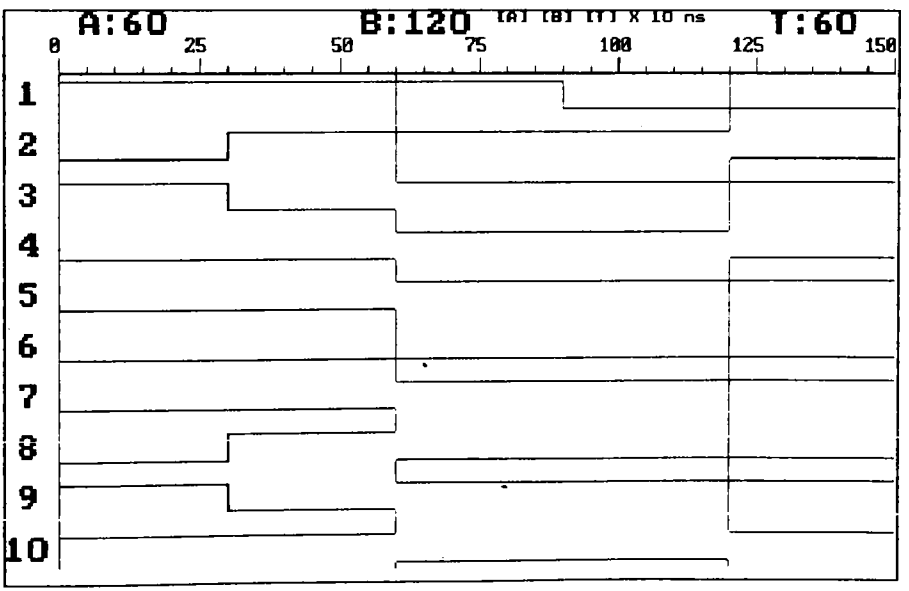

Fig. 8.64. Semnalele de intrare în blocul AD/SC (operaţiile 9-13). 1-A_0, 2-A_1, 3-A_2, 4-B_0, 5-B_1, 6-B_2, 7-I_0, 8-I_1, 9-I_2, 10-I_3.

semnale sînt, de asemenea, reprezentate în fig. 8.65.

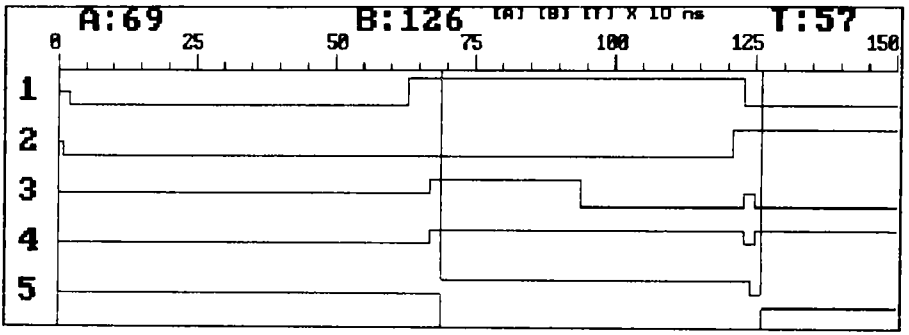

Fig. 8.65. Semnalele de ieşire şi de comandă în blocul AD/SC (operaţiile 9-13).
1-sel.adun.(4 sau 8 biţi), 2-sel.scăd.,
3-H_1, 4-H_2, 5-H_3.

Se observă formarea caracteristicilor mutuale ale celor două operaţii de adunare în intervalul 690-1230 ns. Caracteristica mutuală pentru operaţia de scădere apare validă la 1260 ns.

Funcţionarea blocului multiplexor MUX2 se va urmări în continuare. La intrările multiplexorului sînt prezente semnalele de la ieşirile blocurilor DD/RD şi DS/RS, (fig. 8.62 şi fig. 8.63). De asemenea, la intrarea multiplexorului sînt aduse şi caracteristicile mutuale ale operaţiilor de adunare şi scădere din blocul AD/SC (fig. 8.65). În fig. 8.66 se arată modul de activare a ieşirii, prin semnale de selecţie, pentru caracteristicile mutuale ale operaţiilor de ROTAŢIE DREAPTA şi STÎNGA.

Pe cronogramele reprezentate, se poate vedea activarea caracteristicii mutuale în intervalul de timp 50-330 ns pentru ROTAŢIE DREAPTA şi, respectiv în intervalul 370-630 ns pentru ROTAŢIE STÎNGA.

În mod asemănător, în fig. 8.67 se arată activarea ieşirii multiplexorului pentru caracteristica mutuală a operaţiilor de ADUNARE şi SCĂDERE.

Caracteristicile mutuale sînt prezente la ieşire în intervalul 700-1240 ns, pentru cele două operaţii de adunare şi, respectiv, începînd cu 1270 ns, pentru operaţia de scădere.

Modalitatea de formare a semnăturilor caracteristicilor

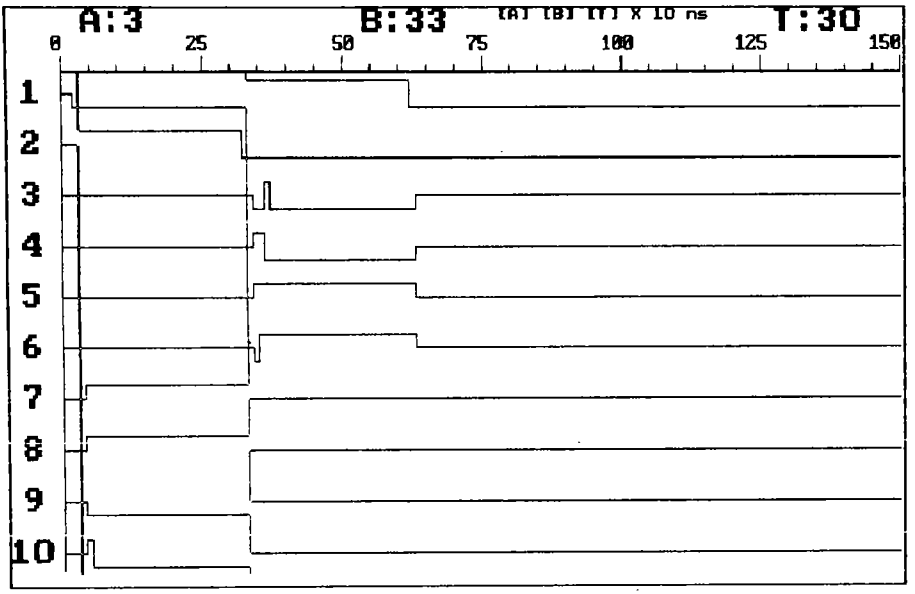

Fig. 8.66. Activarea caracteristicilor mutuale ale operaţiilor de ROTAŢIE DREAPTA şi STÎNGA.
1-sel.rot.stg., 2-sel.rot.dr.,
$3\text{-}H_0(Å^+)_{out}$, $4\text{-}H_1(Å^+)_{out}$, $5\text{-}H_2(Å^+)_{out}$, $6\text{-}H_3(Å^+)_{out}$
$7\text{-}H_0(Å^+)_{out}$, $8\text{-}H_1(Å^+)_{out}$, $9\text{-}H_2(Å^+)_{out}$, $10\text{-}H_3(Å^+)_{out}$.

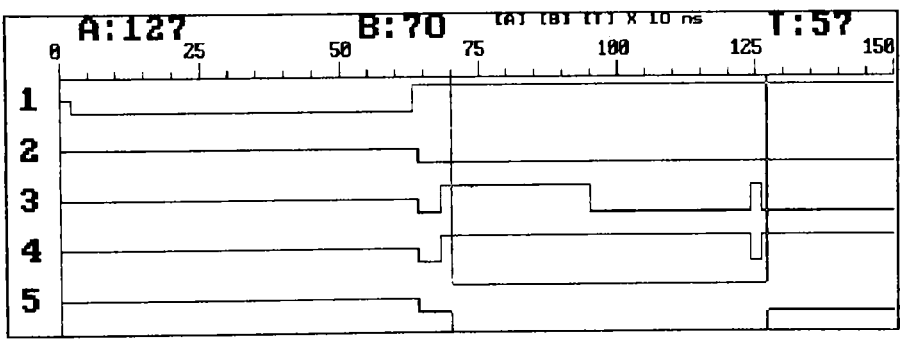

Fig. 8.67. Activarea caracteristicilor mutuale ale operaţiilor de ADUNARE şi SCĂDERE.
1-sel.ad/sc, $2\text{-}H_0(+/-)_{out}$, $3\text{-}H_1(+/-)_{out}$, $4\text{-}H_2(+/-)_{out}$,
$5\text{-}H_3(+/-)_{out}$.

mutuale şi funcţionarea modulului SIGH, pentru operaţiile 9-13 a fost simulată conform cu fig. 8.68. Se observă iniţializarea registrului de comprimare cu valoarea sigH=1000 calculată pentru

operaţia nr. 8.

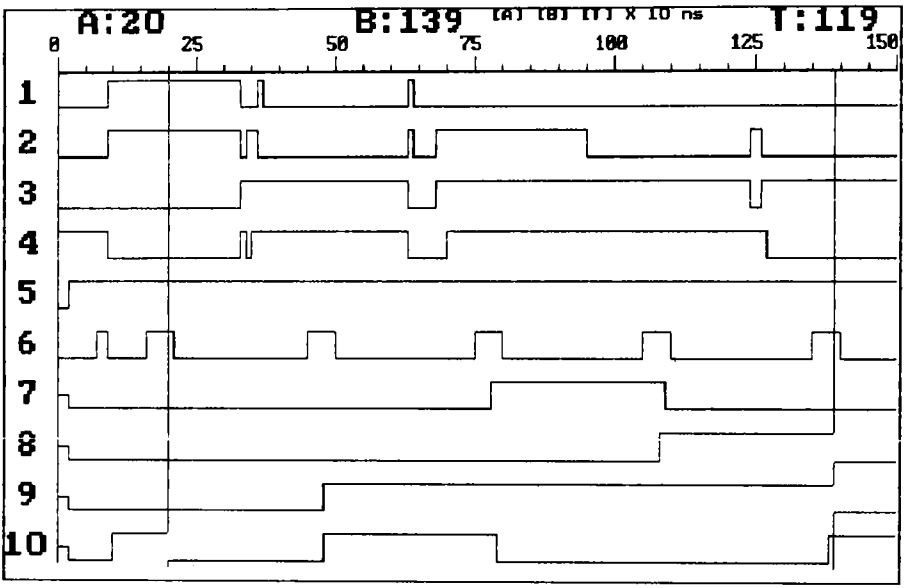

Fig. 8.68. Formarea semnăturii caracteristicilor mutuale H (operaţiile 9-13).
1-H_0, 2-H_1, 3-H_2, 4-H_3, 5-RESET, 6-TACTS, 7-sigH_0, 8-sigH_1, 9-sigH_2, 10-sigH_3.

Comparînd rezultatul obţinut prin simulare, pentru sigH, cu cel calculat teoretic, conform cu fig. 8.23, se verifică corectitudinea fluxului informaţional pînă în acest punct.

Informaţia prezentă la intrarea celui de-al doilea bloc SAU-EXCLUSIV, BLOC2-XOR, apare conform cu fig. 8.68 şi fig. 8.61. Informaţia de control, IC, pentru operaţiile 9-13, se formează la ieşire şi este prezentată în fig. 8.69.

La ieşirea UAL se formează semnătura rezultatului, în modulul sigC. Funcţionarea acestui modul este simulată în fig. 8.70. Corectitudinea funcţionării se poate constata prin comparare cu rezultatele teoretice din fig. 8.24.

Ca şi în cazul formării semnăturilor paralel în celelalte registre, pentru asigurarea continuităţii simulării, registrul de formare a sigC a fost iniţializat cu ultima semnătură formată, cea corespunzătoare operaţiei nr. 8.

În continuare, fiind disponibile ambele informaţii,

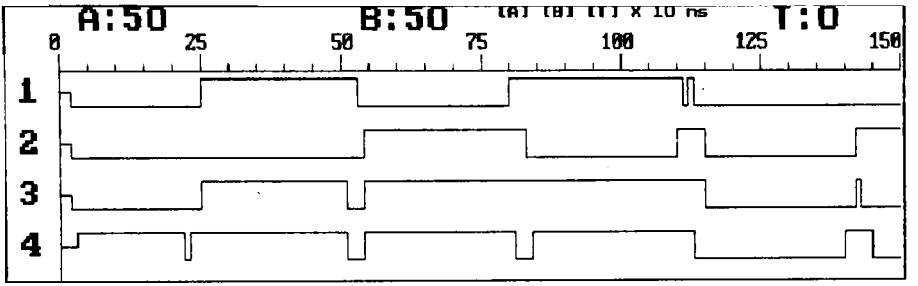

Fig. 8.69. Informaţia de control pentru operaţiile 9-13.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3.

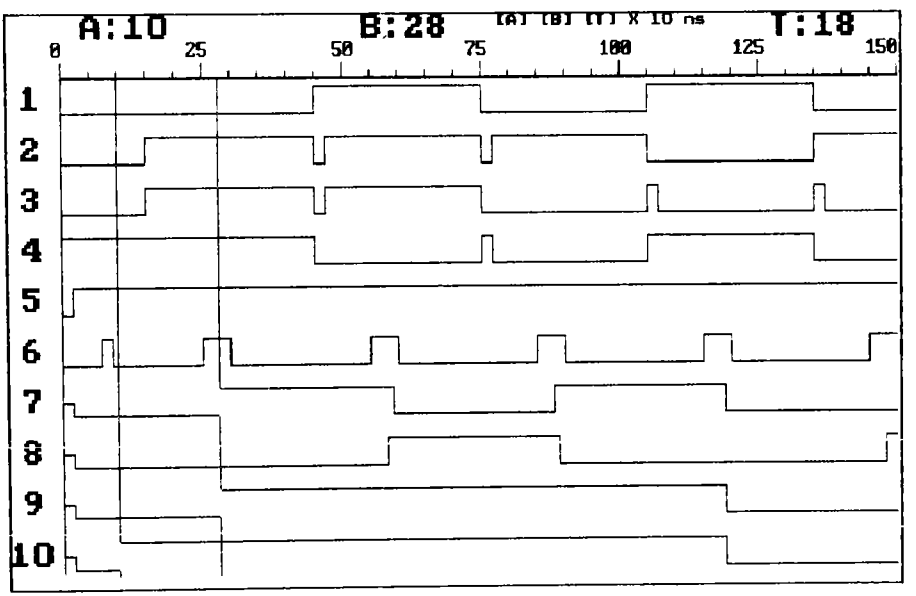

Fig. 8.70. Formarea semnăturii rezultatului din UAL (operaţiile 9-13).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

sigC precum şi de control, IC, se activează modulul comparator COMP1. Funcţionarea modulului este exemplificată în fig. 8.71, pentru operaţiile 9-12 şi în fig. 8.72, pentru operaţiile 12-13. Se observă mărimile aplicate la intrarea comparatorului, precum şi rezultatul comparării, reprezentat prin semnalul EROARE. Validarea comparării este realizată prin semnalul VC (fig. 8.14). Starea logică "0" a semnalului EROARE, pe durata activării,

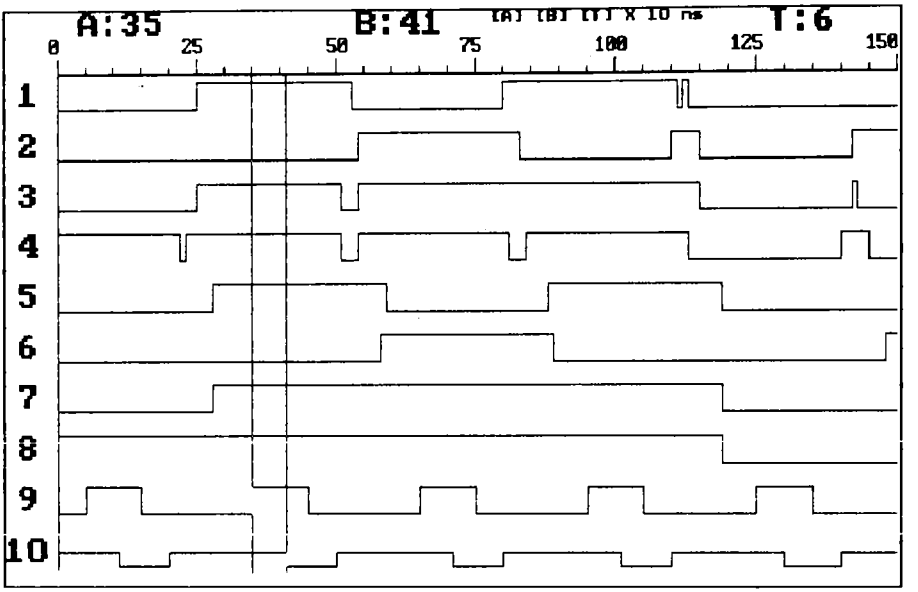

Fig. 8.71. Funcţionarea modulului comparator COMP1 (operaţiile 9-12). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$, 6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

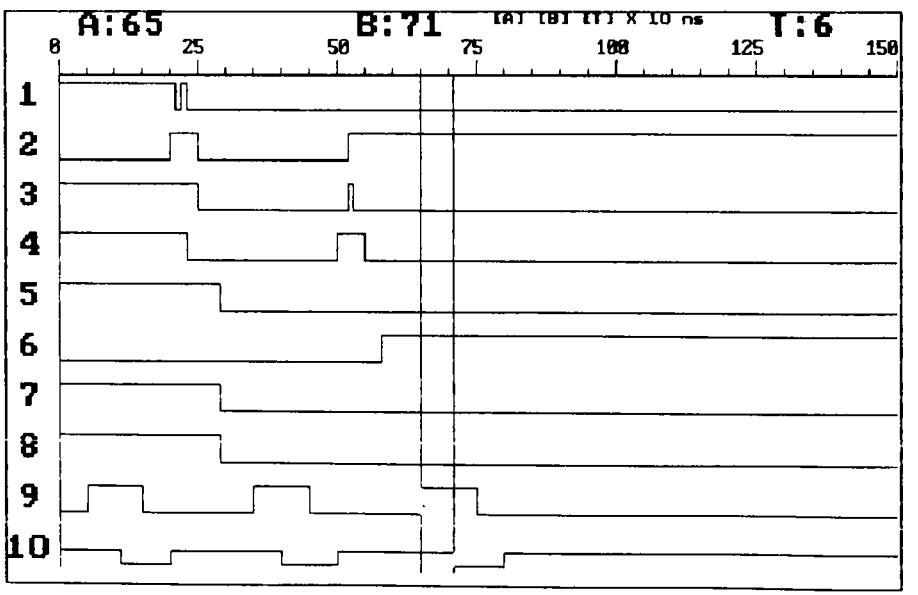

Fig. 8.72. Funcţionarea modulului comparator COMP1 (operaţiile 12-13). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$, 6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

indică faptul că operaţia s-a efectuat corect.

14. OPERAŢIA ARITMETICĂ DE ÎNMULŢIRE (exemplul nr. 1).

Operanzi: A=1111 [$A(x)=x^3+x^2+x+1$], B=1111 [$B(x)=x^3+x^2+x+1$].

Produsul polinomial: A*B=01010101, ($A(x)*B(x)=x^6+x^4+x^2+1$, blocul de înmulţire polinomială, fig. 8.17).

Polinomul de transport: T=10001100, ($T(x)=x^7+x^3+x^2$, blocul de formare al transportului, fig. 8.18).

Semnătura produsului polinomial: sig(A*B)=01010101, ($sig[A(x)*B(x)]=x^6+x^4+x^2+1$, blocul de formare a semnăturii pe 8 biţi, fig. 8.19).

Semnătura transportului: sigT=10001100, ($sigT(x)=x^7+x^3+x^2$, fig. 8.19).

Caracteristica mutuală H (relaţia (6.36)): H=00111000, ($H(x)=x^5+x^4+x^3$, în blocul AD/SC pentru lungimea operanzilor de 8 biţi).

Semnătura caracteristicii mutuale: sigH=00111000, ($sigH(x)=x^5+x^4+x^3$, fig. 8.19).

Ieşire BLOC3-XOR: sig(A*B)⊕sigT=11011001.

Informaţia de control la ieşirea BLOC4-XOR, (rel. 6.37): IC=[sig(A*B)⊕sigT]⊕sigH=11100001.

Rezultatul UAL: C=A×B=11100001

Semnătura rezultatului: sigC=11100001, (fig. 8.19).

Evident, configuraţia semnăturii rezultatului este aceeaşi cu a rezultatului, deoarece este prima semnătură formată pe 8 biţi, după iniţializare. Aceeaşi observaţie este valabilă şi pentru semnăturile pe 8 biţi, formate anterior, ale produsului polinomial A(x)*B(x), transportului T(x) şi respectiv, caracteristicii mutuale H(x).

Ieşire COMP2: EROARE=IC⊕sigC=0.

15. OPERAŢIA ARITMETICĂ DE ÎNMULŢIRE (exemplul nr. 2).

Operanzi: A=0101 [$A(x)=x^2+1$], B=1010 [$B(x)=x^3+x$].

Produsul polinomial: A*B=00100010, ($A(x)*B(x)=x^5+x$, blocul de înmulţire polinomială, fig. 8.17).

Polinomul de transport: T=00010000, ($T(x)=x^4$, blocul de formare al transportului, fig. 8.18).

Semnătura produsului polinomial: sig(A*B)=10001000, ($sig[A(x)*B(x)]=x^7+x^3$, blocul de formare a semnăturii pe 8 biţi, fig. 8.19).

Semnătura transportului: sigT=10111011,

(sigT(x)=$x^7+x^5+x^4+x^3+x+1$, fig. 8.19).

Caracteristica mutuală H (relaţia (6.36)): H=00000000, (H(x)=0, în blocul AD/SC pentru lungimea operanzilor de 8 biţi).

Semnătura caracteristicii mutuale: sigH=01110000, (sigH(x)=$x^6+x^5+x^4$, fig. 8.19).

Ieşire BLOC3-XOR: sig(A*B)⊕sigT=00110011.

Informaţia de control la ieşirea BLOC4-XOR, (rel. 6.37): IC=[sig(A*B)⊕sigT]⊕sigH=01000011.

Rezultatul UAL: C=A×B=00110010

Semnătura rezultatului: sigC=01000011, (fig. 8.19).

Ieşire COMP2: EROARE=IC⊕sigC=0.

Simularea funcţionării blocului de înmulţire, pentru operaţiile nr. 14 şi 15, va putea fi urmărită pe cronograme, în cele ce urmează. Astfel, în fig. 8.73 este simulată funcţionarea modulului de serializare (fig. 8.15 şi fig. 8.16), a operanzilor A ai celor două operaţii. Ieşirea serie a datelor a fost notată cu A(x).

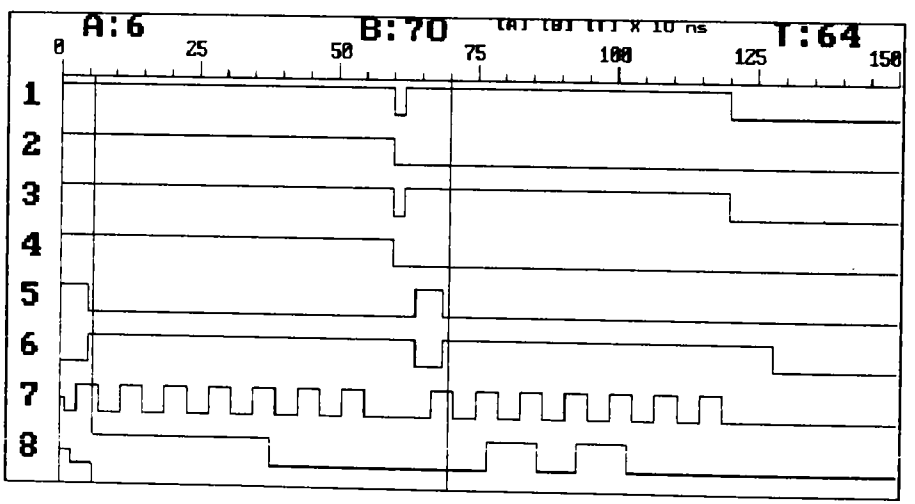

Fig. 8.73. Funcţionarea modulului de serializare (operaţiile 14-15).
1-A_0, 2-A_1, 3-A_2, 4-A_3,
5-PdA, 6-Pn0, 7-TACT1, 8-A(x).

Modificarea valorilor operanzilor s-a considerat a avea loc după un interval de 600 ns. Semnalul TACT1 a fost adoptat la o perioadă de 80 ns, pentru funcţionarea modulului fiind necesare

7 impulsuri de tact. Au fost, de asemenea, reprezentate şi cele două semnale de comandă, PdA, pentru încărcarea paralelă şi Pn0, pentru comanda serializării.

Înmulţirea polinomială între operanzii A, încărcaţi serial sub forma A(x), şi operanzii B, încărcaţi paralel (fig. 8.15 şi fig. 8.17), este realizată conform diagramei din fig. 8.74.

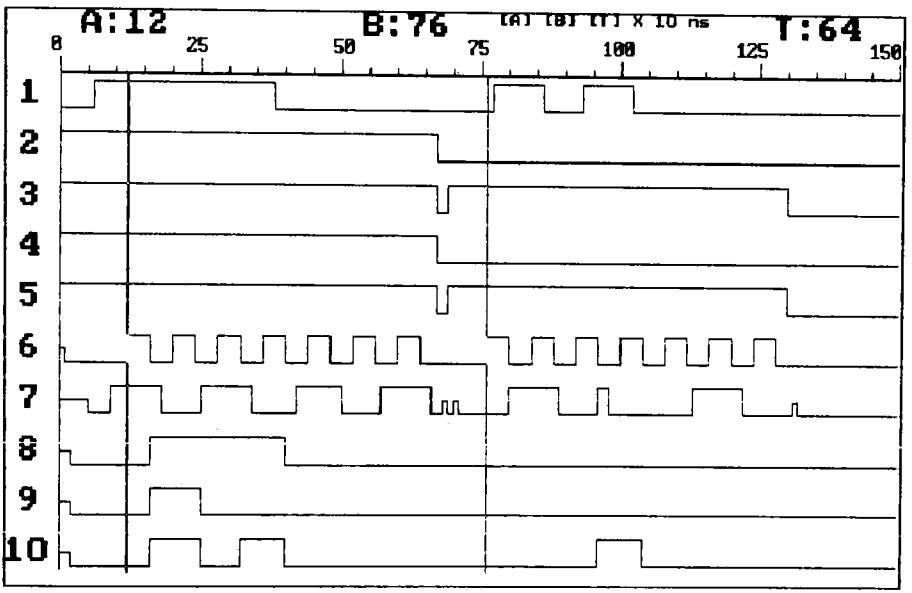

Fig. 8.74. Funcţionarea modulului de înmulţire polinomială, P(x)=A(x)*B(x), (operaţiile 14-15).
1-A(x), 2-B_0, 3-B_1, 4-B_2, 5-B_3,
6-TACT2, 7-P(x), 8-TR1, 9-TR2, 10-TR3.

Prin TR1, TR2 şi TR3 au fost notate cele trei linii de date necesare la formarea cuvîntului de transport al operaţiei de înmulţire. Semnalul TACT2 are aceeaşi perioadă ca şi TACT1, dar apare cu o întîrziere de 90 ns.

În continuare, rezultatul înmulţirii polinomiale este memorat într-un registru (fig. 8.15), comandat de acelaşi semnal TACT2. Funcţionarea acestui registru se poate observa în fig. 8.75.

La comanda semnalului TACT3, care apare cu o întîrziere de 140 ns faţă de TACT1, se formează cuvîntul de transport, în registrul transportului prezentat în fig. 8.15 şi fig. 8.18.

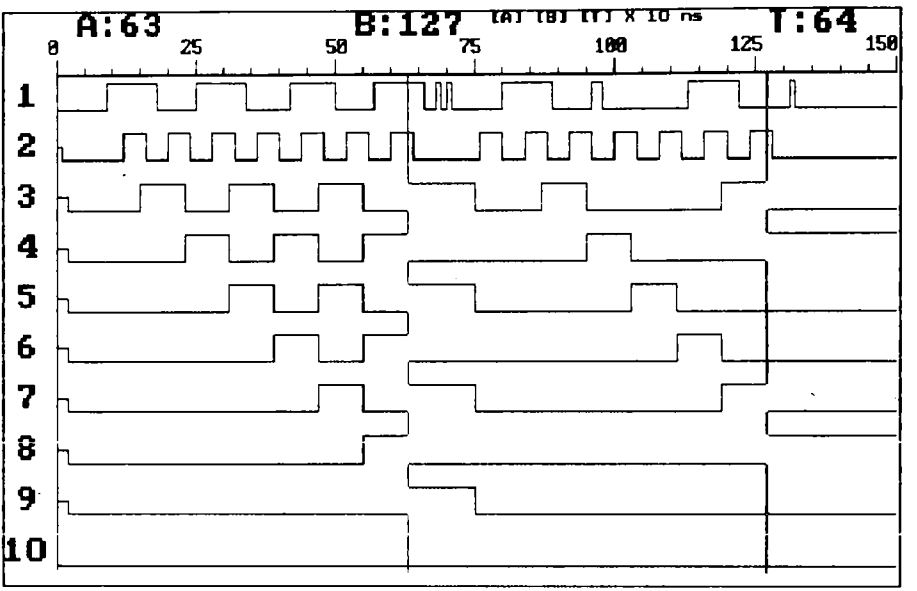

Fig. 8.75. Funcţionarea registrului de memorare a rezultatului P(x)=A(x)*B(x), (operaţiile 14-15).
1-P(x), 2-TACT2, 3-P_0, 4-P_1, 5-P_2, 6-P_3, 7-P_4, 8-P_5, 9-P_6, 10-P_7.

Mărimile de intrare în registru, TR1, TR2 şi TR3, sînt prezentate în fig. 8.76. Funcţionarea în timp se poate urmări pe diagrama din fig. 8.77. La formarea cuvîntului de transport sînt necesare doar 4 impulsuri de tact.

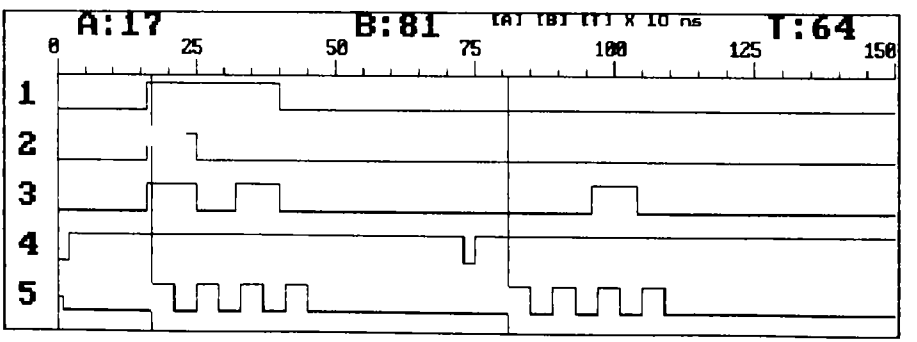

Fig. 8.76. Mărimile de intrare în registrul transportului (operaţiile 14-15).
1-TR1, 2-TR2, 3-TR3, 4-RESET3, 5-TACT3.

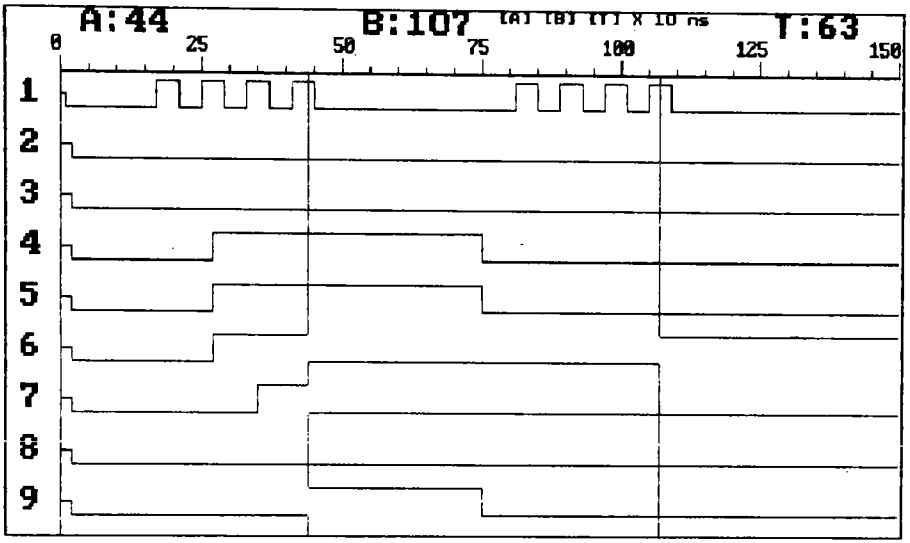

Fig. 8.77. Formarea cuvîntului de transport (operaţiile 14-15).
1-TACT3, 2-T_0, 3-T_1, 4-T_2,
5-T_3, 6-T_4, 7-T_5, 8-T_6, 9-T_7.

Determinarea caracteristicii mutuale a adunării rezultatului înmulţirii polinomiale, P(x)=A(x)*B(x), la cuvîntul de transport se realizează în blocul AD/SC, care este comandat pentru funcţionarea pe 8 biţi. Semnalul de comandă PdOp, precum şi formarea acestei caracteristici pot fi vizualizate pe cronograma din fig. 8.78. Semnalul de comandă PdOp devine activ spre sfîrşitul perioadei active a operanzilor A şi B, în acest caz în intervalul 640-740 ns, pentru prima operaţie de înmulţire şi, respectiv, în intervalul 1280-1380 ns, pentru cea de-a doua.

Semnăturile pe 8 biţi ale caracteristicilor mutuale, pentru cele două operaţii de înmulţire, se realizează în maniera reprezentată în fig. 8.79. Semnalul de tact, TACTS, pentru comanda registrului de comprimare, este activ la momentele 720 ns şi, respectiv, 1350 ns.

Acelaşi impuls de tact este utilizat şi pentru formarea semnăturilor produsului polinomial şi transportului. Astfel, în fig. 8.80 se arată modul de formare a semnăturii pentru produsul polinomial P(x)=A(x)*B(x).

În mod similar, în fig. 8.81 apare formarea semnăturii pentru cuvîntul de transport T(x).

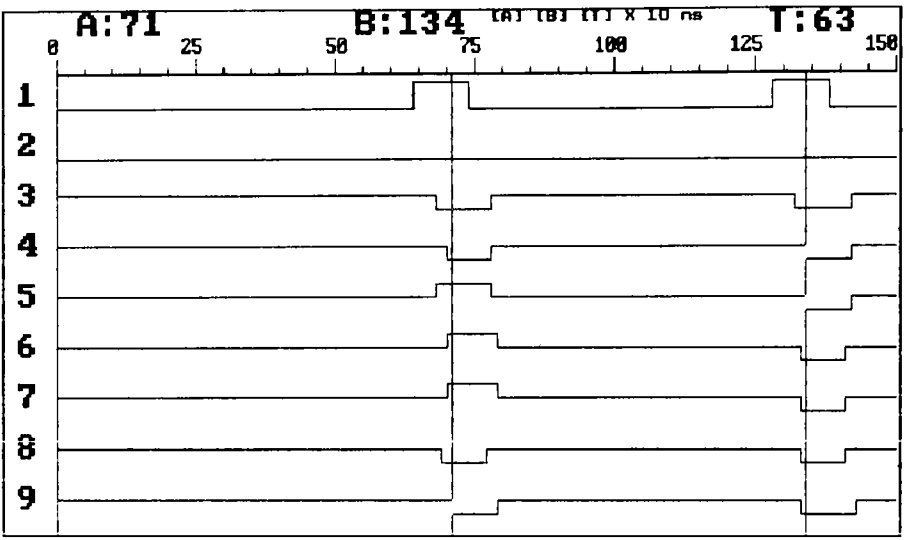

Fig. 8.78. Formarea caracteristicii mutuale (operaţiile 14-15).
1-PdOp, 2-H_0, 3-H_1, 4-H_2,
5-H_3, 6-H_4, 7-H_5, 8-H_6, 9-H_7.

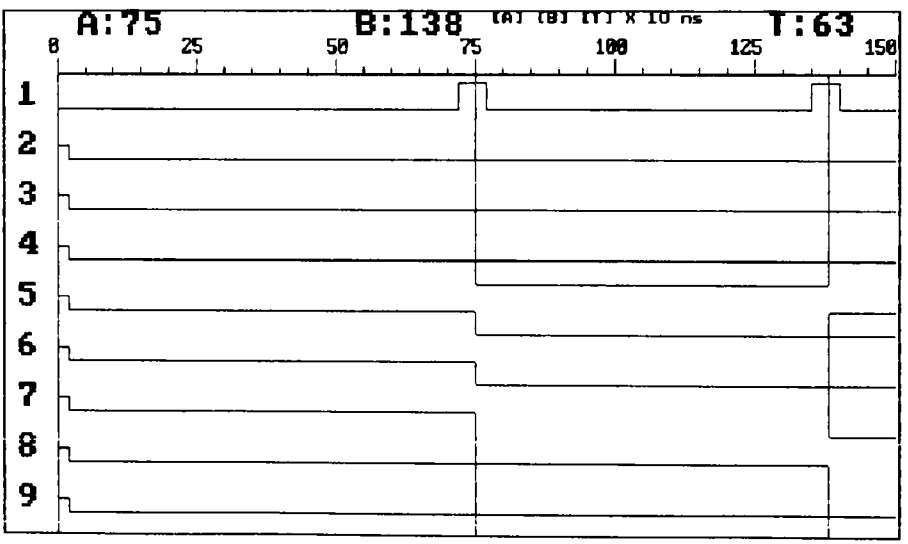

Fig. 8.79. Formarea semnăturii caracteristicii mutuale (operaţiile 14-15).
1-TACTS, 2-sigH_0, 3-sigH_1, 4-sigH_2, 5-sigH_3,
6-sigH_4, 7-sigH_5, 8-sigH_6, 9-sigH_7.

Între semnăturile produsului polinomial şi transportului

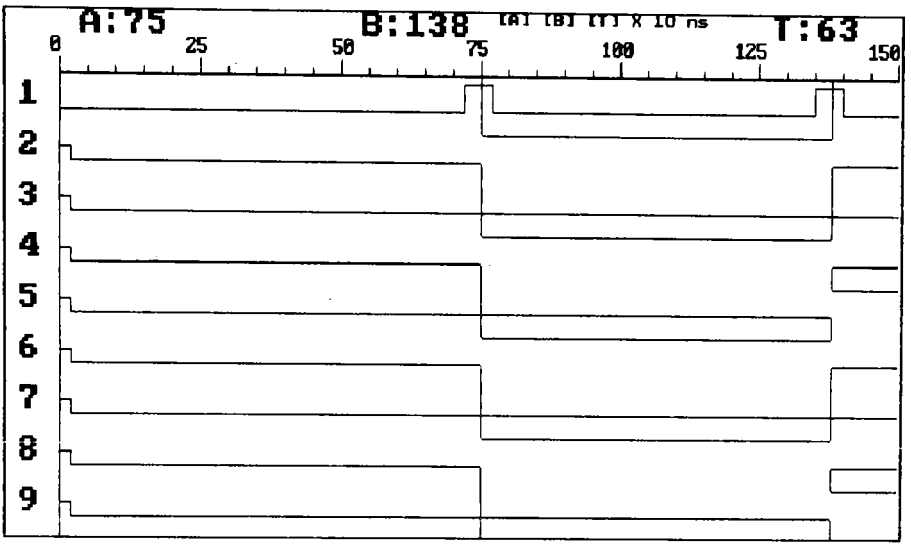

Fig. 8.80. Formarea semnăturii produsului polinomial P(x)=A(x)*B(x) (operaţiile 14-15).
1-TACTS, 2-sigP_0, 3-sigP_1, 4-sigP_2, 5-sigP_3, 6-sigP_4, 7-sigP_5, 8-sigP_6, 9-sigP_7.

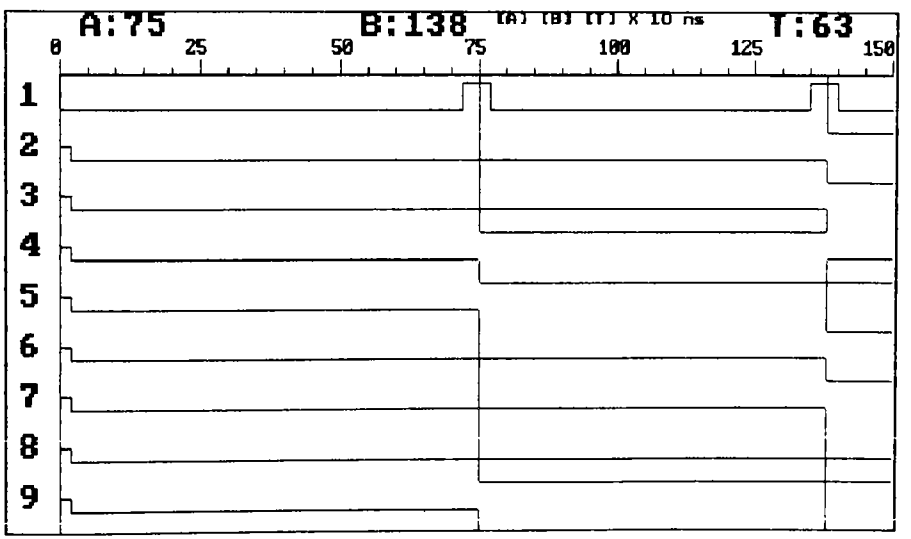

Fig. 8.81. Formarea semnăturii cuvîntului de transport T(x) (operaţiile 14-15).
1-TACTS, 2-sigT_0, 3-sigT_1, 4-sigT_2, 5-sigT_3, 6-sigT_4, 7-sigT_5, 8-sigT_6, 9-sigT_7.

urmează a se executa operaţia logică SAU-EXCLUSIV, în BLOC3-XOR (fig. 8.1). Funcţionarea acestui bloc este reprezentată în diagrama de simulare din fig. 8.82.

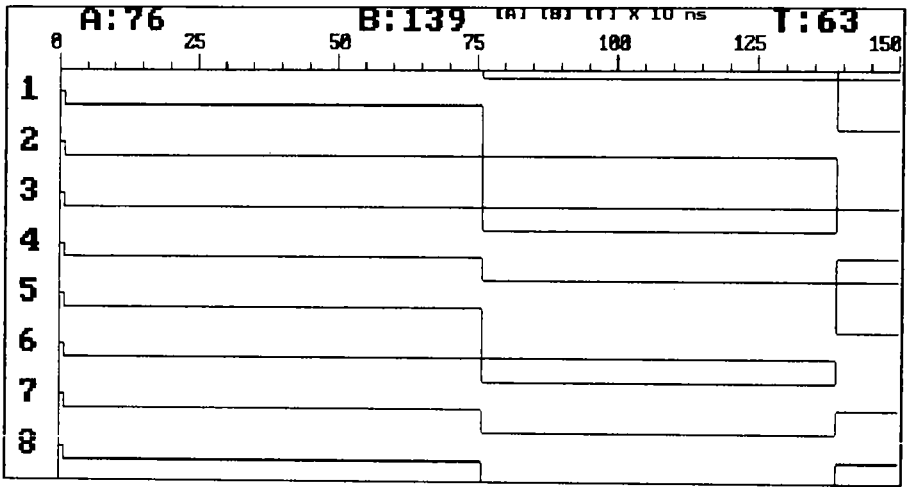

Fig. 8.82. Semnalele de ieşire din BLOC3-XOR (operaţiile 14-15).
1-sigP_0⊕sigT_0, 2-sigP_1⊕sigT_1,
3-sigP_2⊕sigT_2, 4-sigP_3⊕sigT_3,
5-sigP_4⊕sigT_4, 6-sigP_5⊕sigT_5,
7-sigP_6⊕sigT_6, 8-sigP_7⊕sigT_7.

În fine, informaţia de control IC, se formează la ieşirea următorului modul, BLOC4-XOR (fig. 8.1), în urma operaţiei SAU-EXCLUSIV între ieşirea din BLOC3-XOR şi semnătura sigH a caracteristicii mutuale. În fig. 8.83 se arată modul de formare a informaţiei de control pentru cele două operaţii de înmulţire considerate.

La ieşirea UAL se formează semnătura pe 8 biţi a rezultatului, în modulul sigC. Pe baza mărimilor aplicate la intrare, reprezentate în fig. 8.84, funcţionarea acestui modul este simulată în fig. 8.85. Corectitudinea funcţionării se poate constata cu uşurinţă, pe baza celor prezentate anterior la descrierea celor două operaţii de înmulţire.

În continuare, fiind disponibile ambele informaţii, sigC precum şi de control, IC, se activează modulul comparator COMP2, (fig. 8.1), pentru realizarea comparării pe 8 biţi. Funcţionarea modulului este exemplificată în fig. 8.86, pentru operaţiile 14-

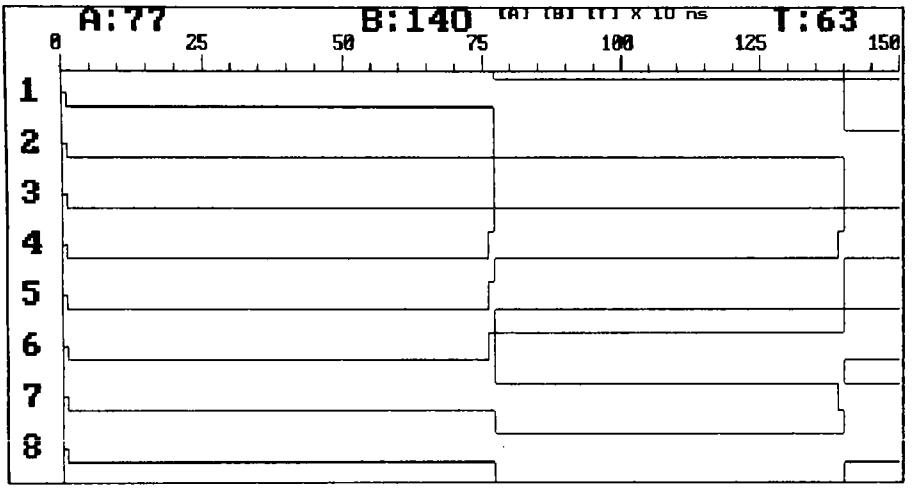

Fig. 8.83. Informaţia de control pentru operaţiile 14-15.
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3,
5-IC_4, 6-IC_5, 7-IC_6, 8-IC_7.

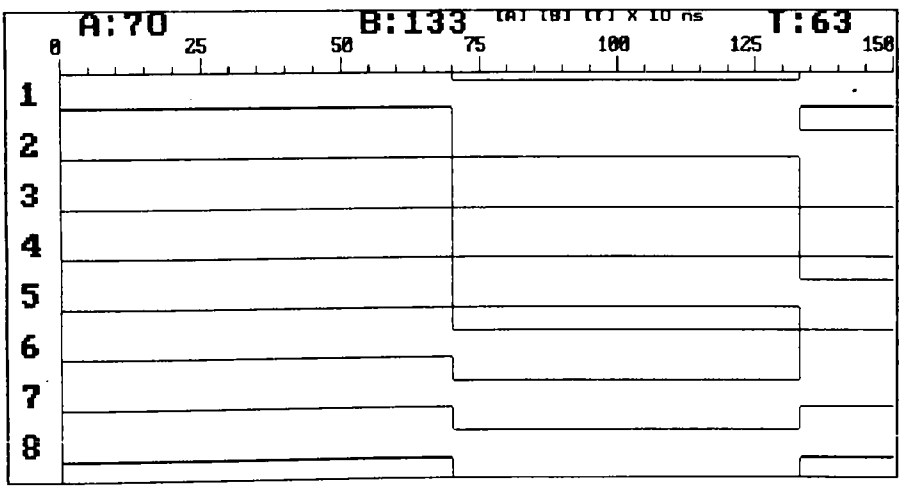

Fig. 8.84. Rzultatul din UAL pentru operaţiile 14-15.
1-C_0, 2-C_1, 3-C_2, 4-C_3,
5-C_4, 6-C_5, 7-C_6, 8-C_7.

15.

Se observă rezultatul comparării, reprezentat prin semnalul EROARE. Validarea comparării este realizată prin semnalul VC activ în intervalele 800-860 ns şi 1420-1480 ns. Starea logică

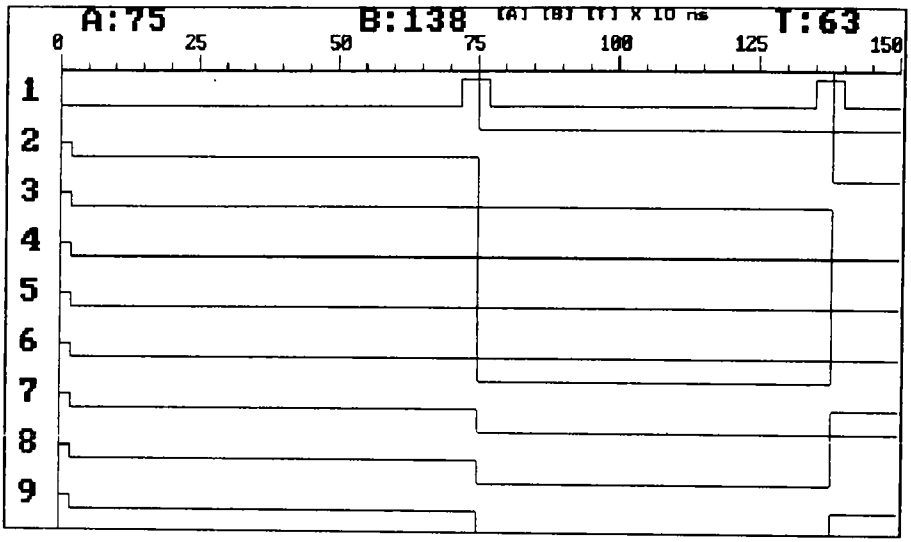

Fig. 8.85. Formarea semnăturii rezultatului din UAL (operaţiile 14-15).
1-TACTS, 2-sigC_0, 3-sigC_1, 4-sigC_2, 5-sigC_3, 6-sigC_4, 7-sigC_5, 8-sigC_6, 9-sigC_7.

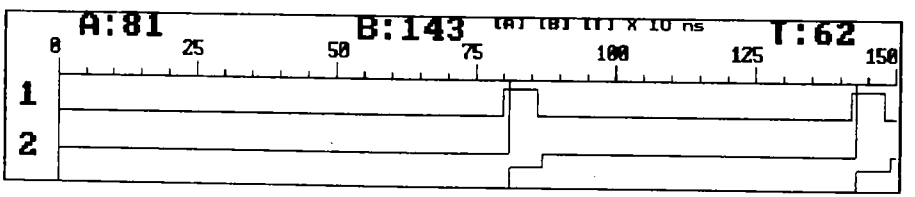

Fig. 8.86. Funcţionarea modulului comparator COMP2 (operaţiile 14-15).
1-VC, 2-EROARE.

"0" a semnalului EROARE, pe durata activării, indică faptul că operaţia s-a efectuat corect.

Funcţionarea întregului bloc pentru controlul operaţiei de înmulţire necesită o serie de semnale de comandă, care au fost în parte vizualizate în diagramele anterioare de simulare. O reprezentatre unitară a acestor semnale se poate urmări în fig. 8.87. şi fig. 8.88.

Performanţele deosebit de ridicate realizate de blocul de control a operaţiilor de înmulţire se pot uşor constata din analiza diagramelor de simulare, reprezentate mai sus. Astfel,

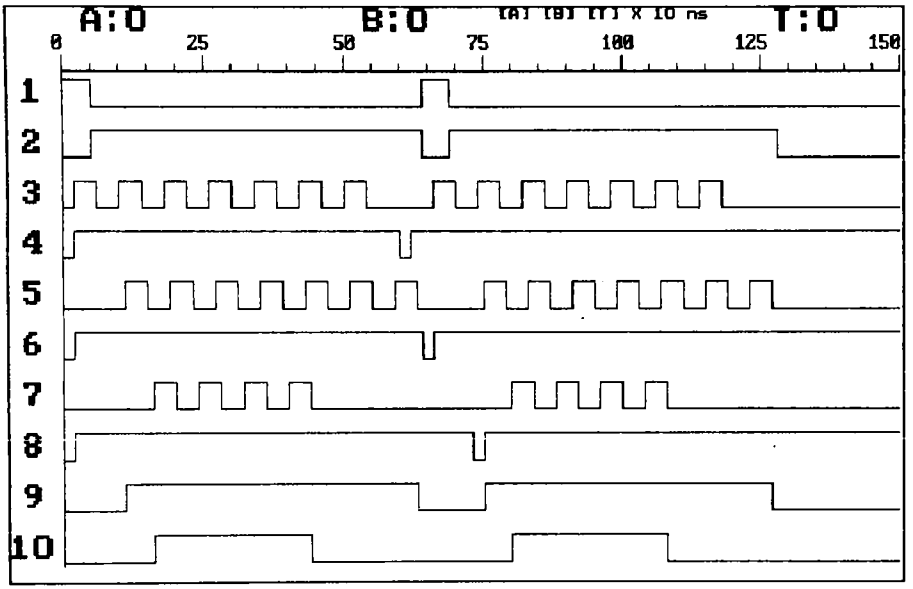

Fig. 8.87. Semnalele de comandă ale blocului de control a operaţiilor de înmulţire (operaţiile 14-15).
1-PdA, 2-Pn0, 3-TACT1, 4-RESET1 (serializare), 5-TACT2, 6-RESET2 (înmulţire polinomială), 7-TACT3, 8-RESET3 (transport), 9-COD1 (fig.8.15), 10-COD2 (fig.8.15).

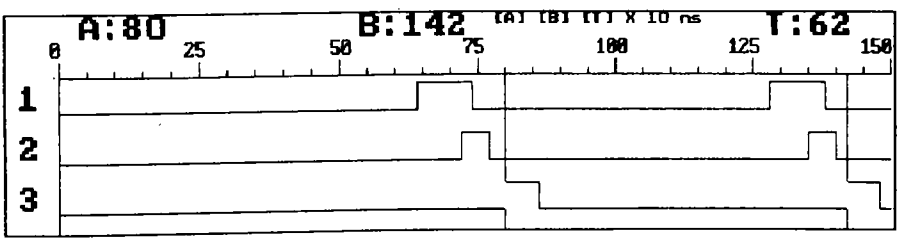

Fig. 8.88. Semnalele de comandă ale blocului de control a operaţiilor de înmulţire (operaţiile 14-15)- continuare.
1-PdOp (bloc AD/SC), 2-TACTS, 3-VC.

în ansamblu, durata de realizare a controlului unei operaţii de înmulţire se poate considera de la momentul iniţializării registrului de serializare (fig. 8.87), pînă la momentul

activării semnalului de comparare VC (fig. 8.88), ceea ce reprezintă un total de 780 ns. La intrarea blocului comparator, COMP2, informaţia de control este disponibilă după 750 ns. Considerînd tactul principal al sistemului de calcul, ca fiind la baza generării celorlalte semnale de tact, TACT1, TACT2, respectiv TACT3, durata totală acoperitoare de control este echivalentă cu 10 tacturi maşină. Comparativ, durata de execuţie a operaţiei de înmulţire într-o UAL, pe 4 biţi, avînd o structură tip Intel 8086, este estimată la 41-45 de tacturi.

În ceea ce priveşte performanţele de control ale celorlalte tipuri de operaţii aritmetice şi logice, acestea se încadrează în totalitate sub limita de execuţie a operaţiilor respective în UAL, aşa cum se poate observa din diagramele de simulare. Astfel, pentru controlul operaţiei de adunare, timpul maxim de formare a informaţiei de control este de 170 ns, ceea ce se află sub limita celor 3 tacturi necesare pentru execuţia operaţiei în UAL. Pentru operaţiile de tip deplasare/rotaţie, care se execută în UAL în timpul cel mai scurt de 2 tacturi, informaţia de control este formată în timpul de 100 ns, încadrîndu-se în intervalul de execuţie. Reamintim că aceste valori rezultă în urma procesului de simulare în care s-a adoptat valoarea de 10 ns, ca timp mediu de propagare pe o poartă logică ŞI-NU. Evident că în funcţie de tehnologia utilizată la integrare, la valori mai mici ale timpului de propagare pe o poartă, performanţele dispozitivului de control vor putea creşte considerabil.

În tab. 8.3 se reprezintă comparativ, pentru diferite lungimi ale operanzilor, duratele de execuţie şi de control ale operaţiei de înmulţire şi a celorlalte tipuri de operaţii, avînd ca referinţă microprocesorul Intel 8086. Se observă, aşadar, că performanţele dispozitivului de control realizat se încadrează în timpul de execuţie a tuturor tipurilor de operaţii aritmetice şi logice, care au loc în UAL. Prin urmare, degradarea performanţelor de viteză ale unui UAL, controlat prin această metodă, se datorează doar timpului de formare a semnăturii rezultatului operaţiei, la ieşirea UAL. În cazul utilizării metodei de formare paralelă a semnăturilor, timpul de întîrziere este doar de 1 tact.

Tab. 8.3. Evaluarea performanţelor dispozitivului de control.

OPERAŢIE ARITMETICĂ SAU LOGICĂ	TIMP DE CALCUL UAL [nr. tacturi]	TIMP DE CONTROL [nr. tacturi]
ÎNMULŢIRE (4 biţi)	41-45	9-10
ÎNMULŢIRE (8 biţi)	70-77	17-18
ÎNMULŢIRE (16 biţi)	118-133	33-34
OPERAŢII CU UN OPERAND	2	<2
OPERAŢII CU DOI OPERANZI	3	<3

8.3. MODELAREA ERORILOR

În paragraful anterior am pus în evidenţă corectitudinea funcţionării blocului de control al operaţiilor aritmetice şi logice, prin simularea a cîtorva cazuri semnificative. În continuare, vom aborda problema modelării capacităţii de detecţie a erorilor ce pot apare la execuţia operaţiilor în UAL. În ceea ce priveşte capacitatea de detecţie a metodei de control, exprimată probabilistic, remarcăm valoarea ei ridicată, specifică metodei de comprimare bazată pe generarea biţilor de control separaţi de cei utili la coduri ciclice [59],[99].

Modelarea erorilor va fi exemplificată pentru fiecare dintre operaţiile considerate anterior, prin aceeaşi metodă de simulare. Astfel, pentru fiecare din valorile rezultat ale operaţiilor în cauză s-a considerat cazul defavorabil cel mai probabil al erorilor singulare, distribuite aleator în toate rangurile binare. Evident, într-o manieră similară poate fi abordată problema şi pentru modelarea erorilor de ordin superior. Reamintim că detecţia erorilor multiple este corelată de modalitatea de alegere a tipului de polinom generator [34],[44], utilizat la formarea semnăturii şi, prin urmare, este independentă de structura propusă a blocului de control. Remarcăm prin aceasta avantajul gradului ridicat de generalitate a metodei de control a operaţiilor aritmetice şi logice la alegerea polinomului generator al codului ciclic.

Rangul binar în care apare eroarea va fi specificat în mod

distinct, valorile eronate ale rezultatelor din UAL putîndu-se compara cu cele corecte prezentate în paragraful anterior.

1. OPERAŢIA LOGICĂ ŞI.

Rezultatul UAL: C=A∧B=1010

↑ eroare

Corespunzător valorii eronate a rezultatului la ieşirea din UAL, se formează o altă semnătură, care este vizualizată în fig. 8.89. În figură s-au arătat, de fapt, toate semnăturile formate pentru operaţiile 1-4.

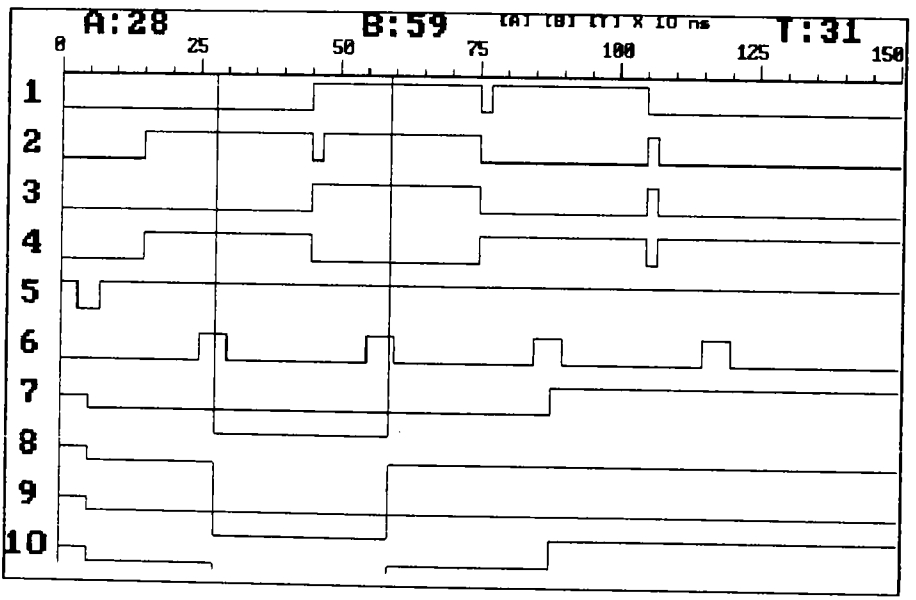

Fig. 8.89. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 1).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

Datorită faptului că semnăturile se formează paralel, şi celelalte semnături, corespunzătoare operaţiilor 2-4, vor fi diferite de situaţia corectă, chiar dacă rezultatul din UAL a reieşit corect. Prin urmare, se observă gradul ridicat de punere în evidenţă a erorilor. Evident, în cazul apariţiei unei erori, acest fapt se semnalizează unităţii de decizie, urmînd a se reiniţializa registrele de formare a semnăturilor, în vederea restabilirii capacităţii de control.

Detecţia erorii este realizată în modulul comparator, COMP1,

activîndu-se semnalul EROARE. Funcţionarea comparatorului şi detecţia erorii sînt reprezentate în fig. 8.90. Diagrama se poate analiza prin analogie cu cea reprezentată în fig. 8.42, pentru cazul funcţionării fără defecţiuni.

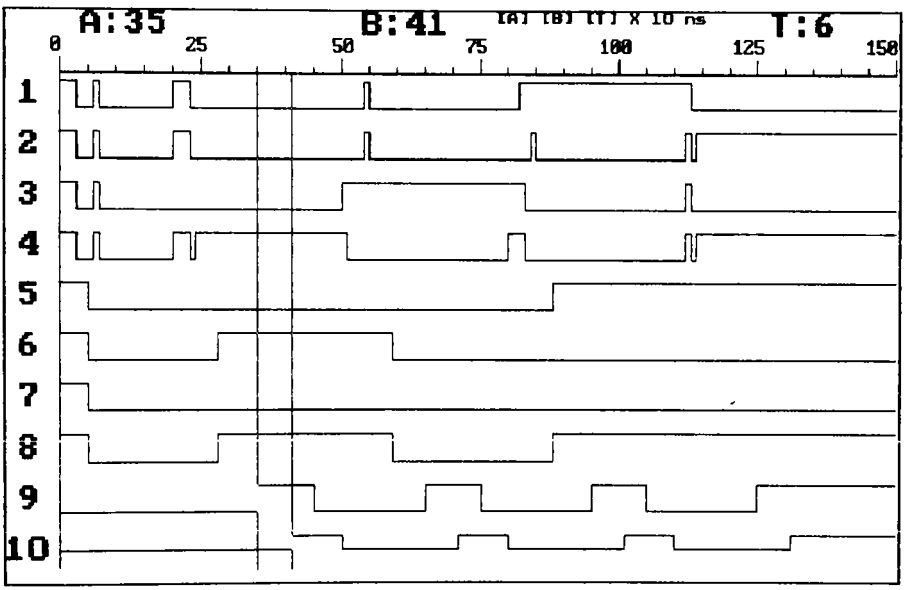

Fig. 8.90. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 1).
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$, 6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

2. OPERAŢIA LOGICĂ SAU.

Rezultatul UAL: C=A∨B=0110
↑ eroare

Semnătura rezultatului eronat este formată conform cronogramei din fig. 8.91.

Detecţia erorii se realizează în COMP1, conform cu fig. 8.92.

În acest caz, semnalul EROARE este activat începînd cu a doua operaţie exemplificată, execuţia primei operaţii fiind considerată corectă.

3. OPERAŢIA LOGICĂ SAU-EXCLUSIV.

Rezultatul UAL: C=A⊕B=0001
↑ eroare

Formarea semnăturii rezultatului eronat şi detecţia erorii

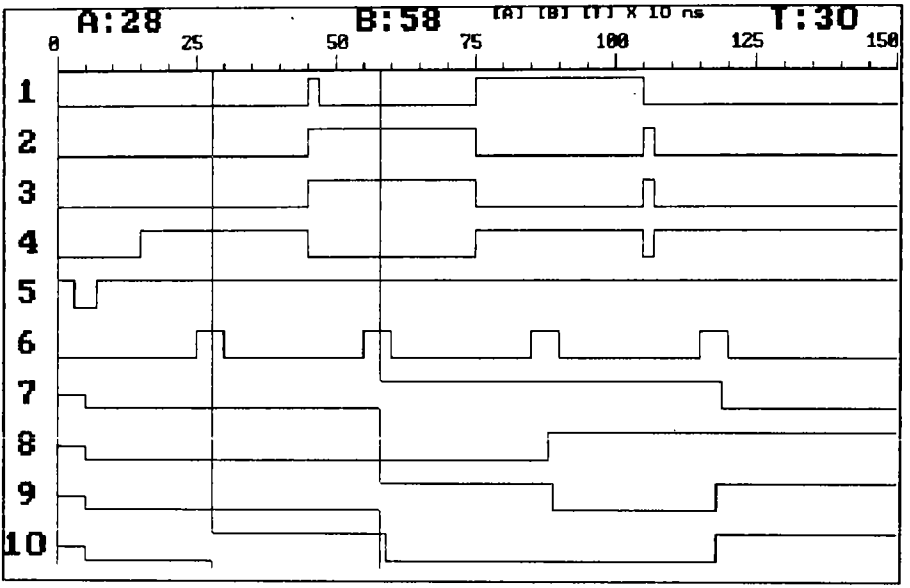

Fig. 8.91. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 2). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

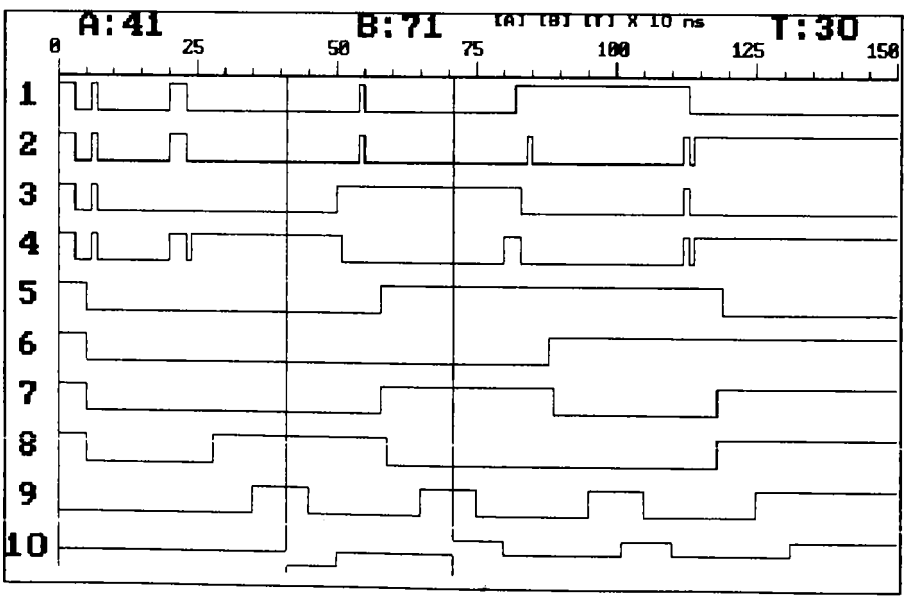

Fig. 8.92. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 2). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

sînt prezentate în fig. 8.93 şi, respectiv, în fig. 8.94.

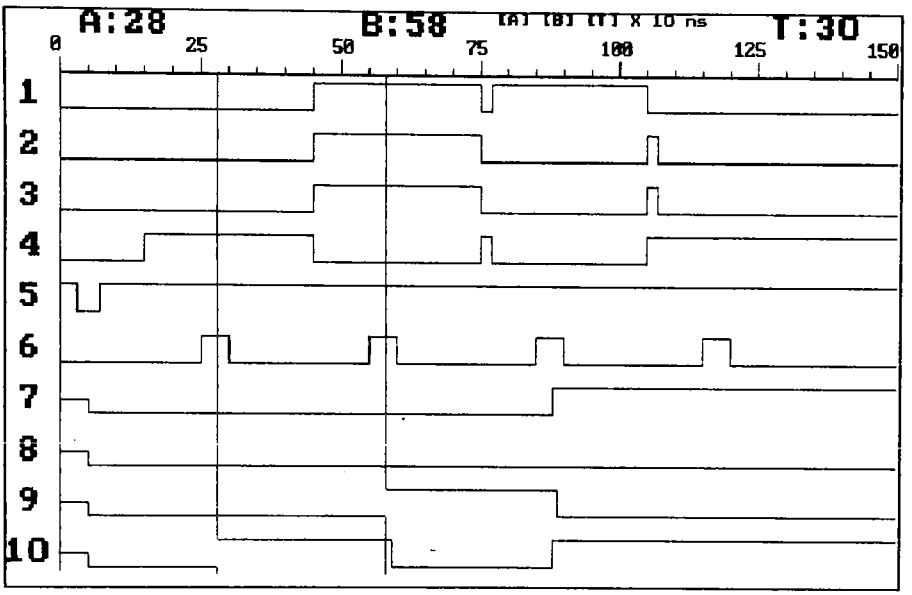

Fig. 8.93. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 3). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

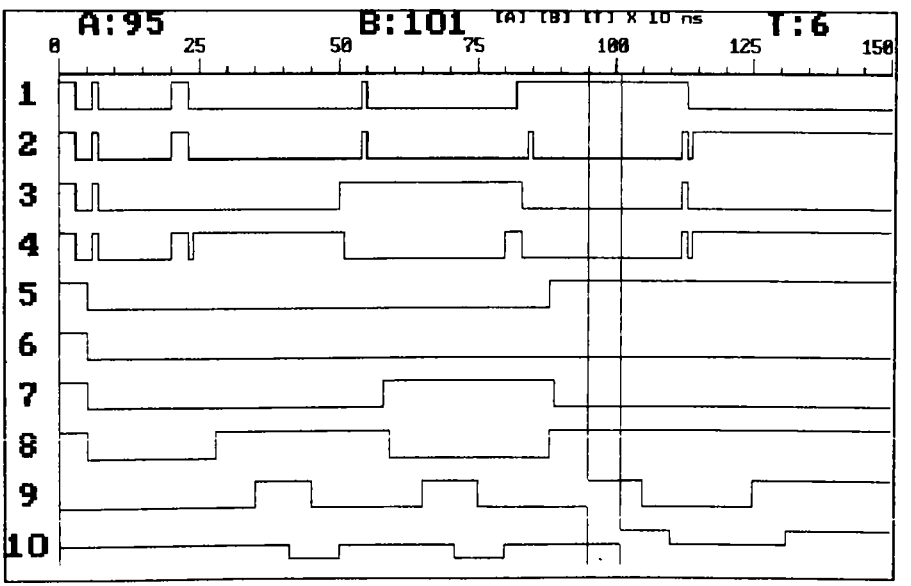

Fig. 8.94. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 3). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

4. OPERAŢIA LOGICĂ DE INVERSIUNE.

Rezultatul UAL: C=$\bar{A}$=1100

↑ eroare

Corespunzător acestui rezultat se formează semnătura pentru comparaţie, reprezentată în fig. 8.95.

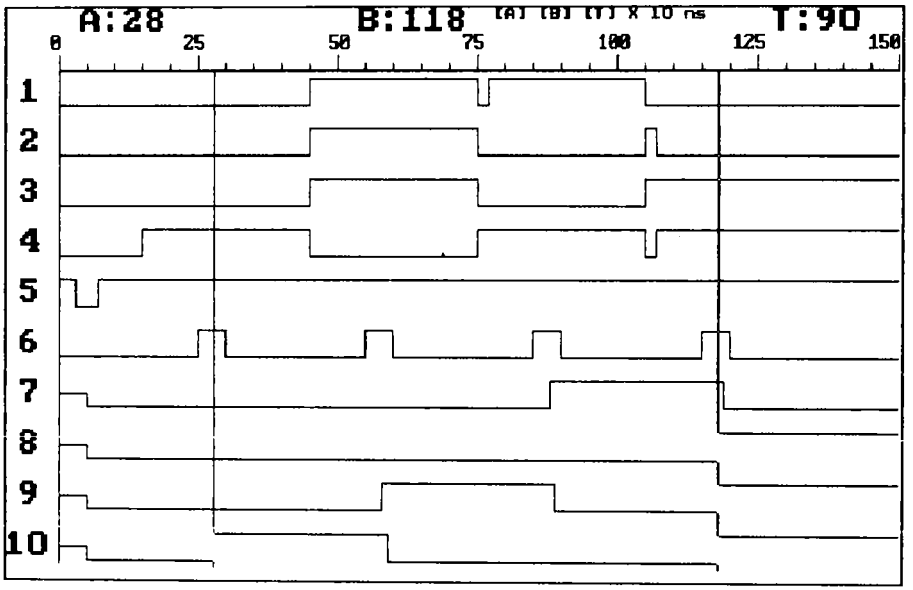

Fig. 8.95. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 4).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET,
6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

Rezultatul comparării cu informaţia de control şi detecţia erorii se observă în fig. 8.96.

5. OPERAŢIA LOGICĂ DE COINCIDENŢĂ.

Rezultatul UAL: C=A⊙B=0000

↑ eroare

Formarea semnăturii rezultatului eronat şi detecţia erorii sînt prezentate în fig. 8.97 şi, respectiv, în fig. 8.98.

Ca şi în cazul primelor patru operaţii eroarea s-a considerat doar la nivelul rezultatului unei singure operaţii (operaţia nr. 5), însă pentru claritatea prezentării s-a arătat formarea semnăturilor şi detecţia erorii, care se propagă, şi pentru celelalte operaţii, nr. 6-9, chiar dacă acestea s-au considerat executate corect.

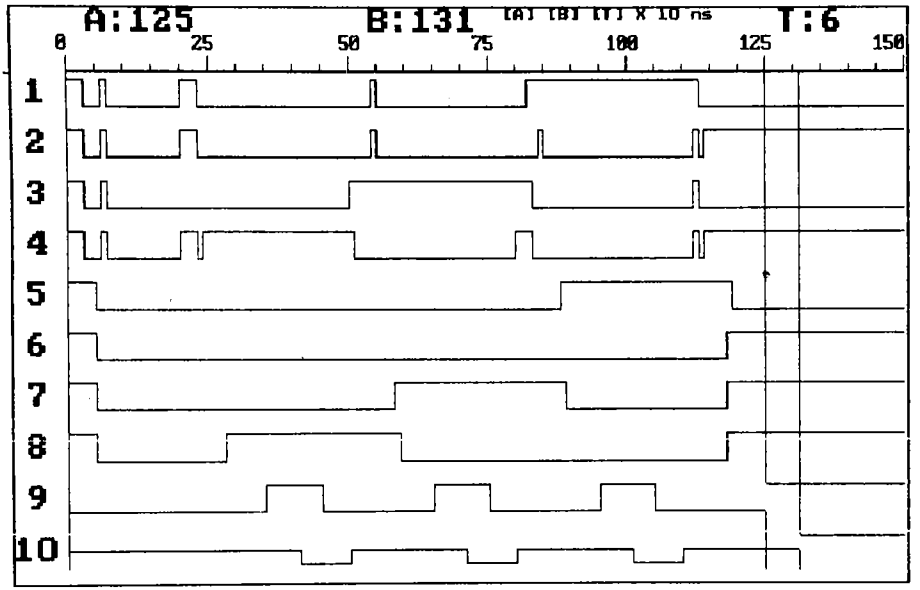

Fig. 8.96. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 4). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$, 6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

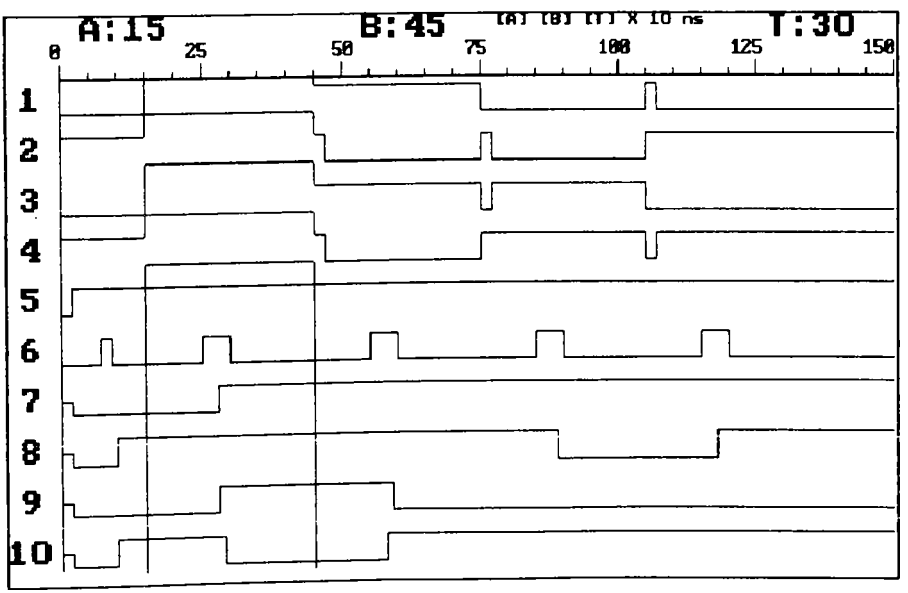

Fig. 8.97. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 5). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-$sigC_0$, 8-$sigC_1$, 9-$sigC_2$, 10-$sigC_3$.

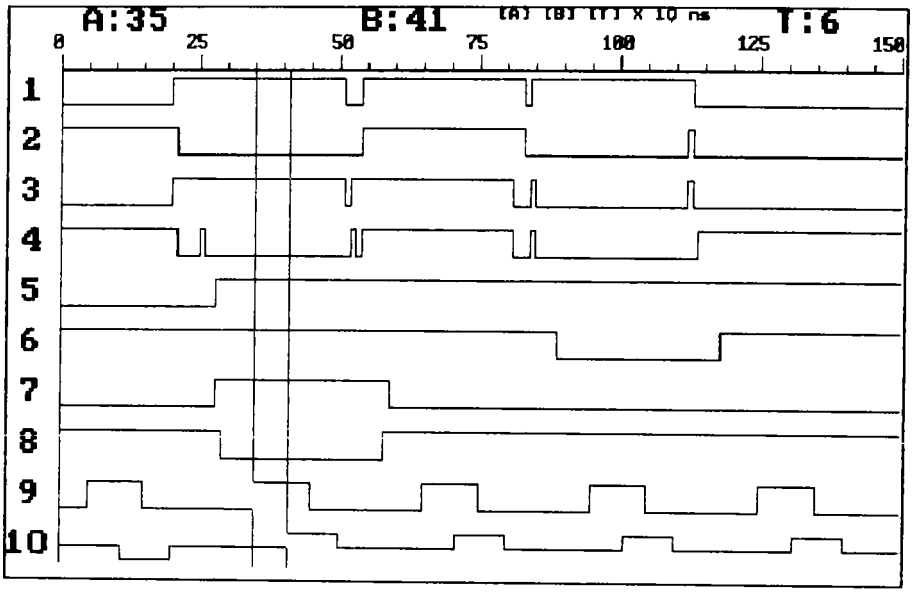

Fig. 8.98. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 5).
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

6. OPERAŢIA LOGICĂ DEPLASARE DREAPTA.

Rezultatul UAL: C=A*=1101
↑ eroare

Corespunzător acestui rezultat se formează semnătura pentru comparaţie, reprezentată în fig. 8.99.

Rezultatul comparării cu informaţia de control şi detecţia erorii se observă în fig. 8.100.

7. OPERAŢIA LOGICĂ DEPLASARE STÎNGA.

Rezultatul UAL: C=A*=1101
↑ eroare

Eroarea manifestată determină o semnătură reprezentată în fig. 8.101, care diferă de cea rezultată în urma controlului, semnalizîndu-se apariţia erorii, conform celor arătate în fig. 8.102.

8. OPERAŢIA LOGICĂ DEPLASARE STÎNGA (cu pierderea informaţiei).

Rezultatul UAL: C=A*=1110
↑ eroare

În mod similar, semnătura rezultatului afectat de eroare şi funcţionarea modulului comparator, cu detecţia erorii sînt

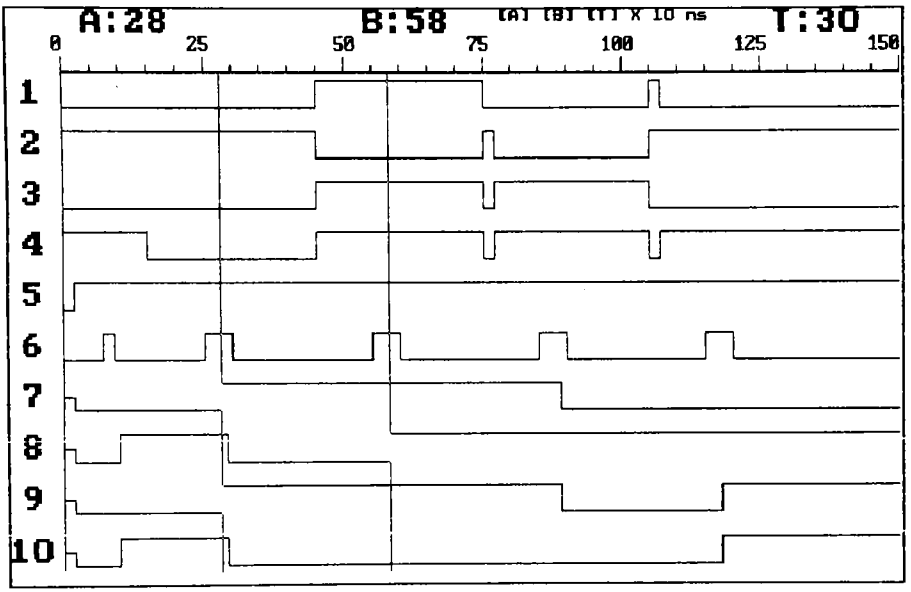

Fig. 8.99. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 6). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

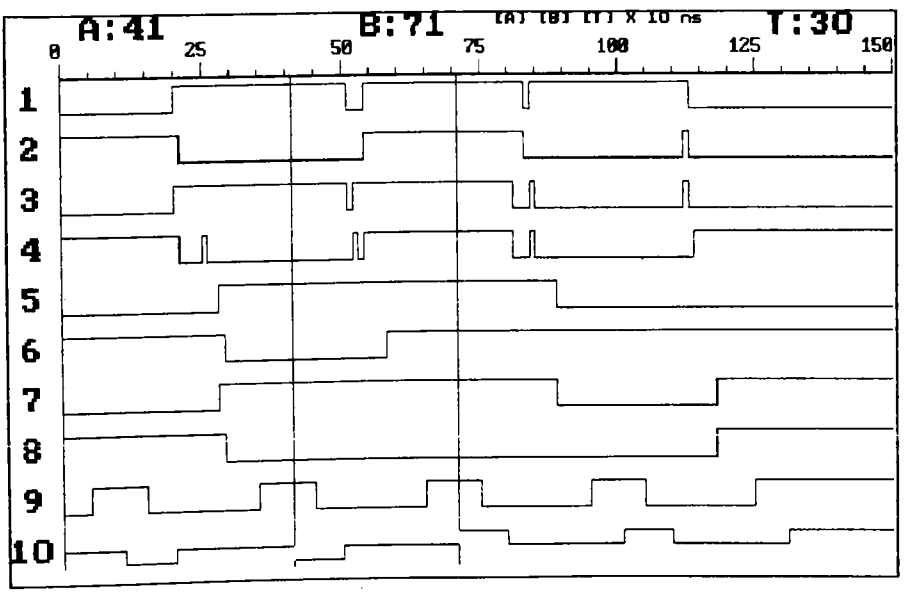

Fig. 8.100. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 6). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

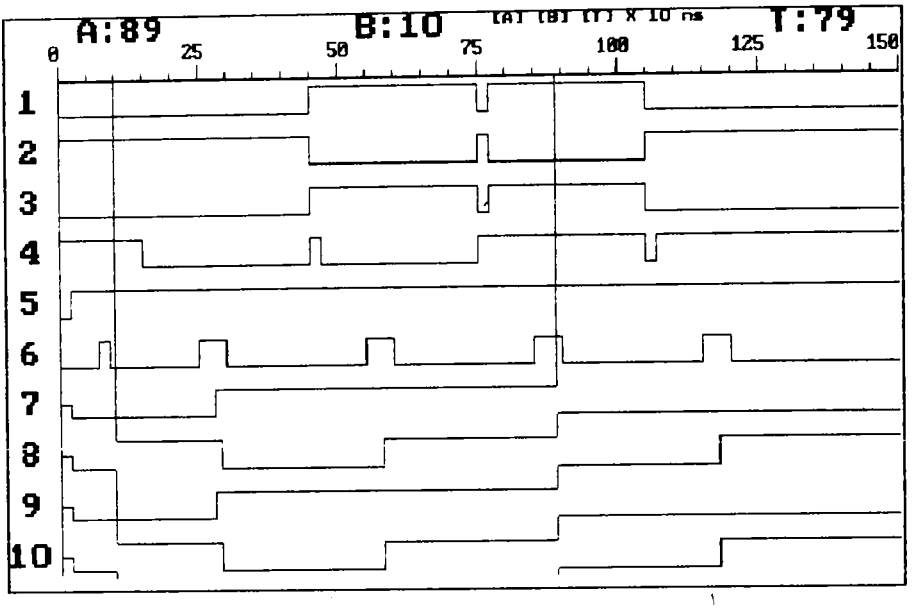

Fig. 8.101. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 7). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

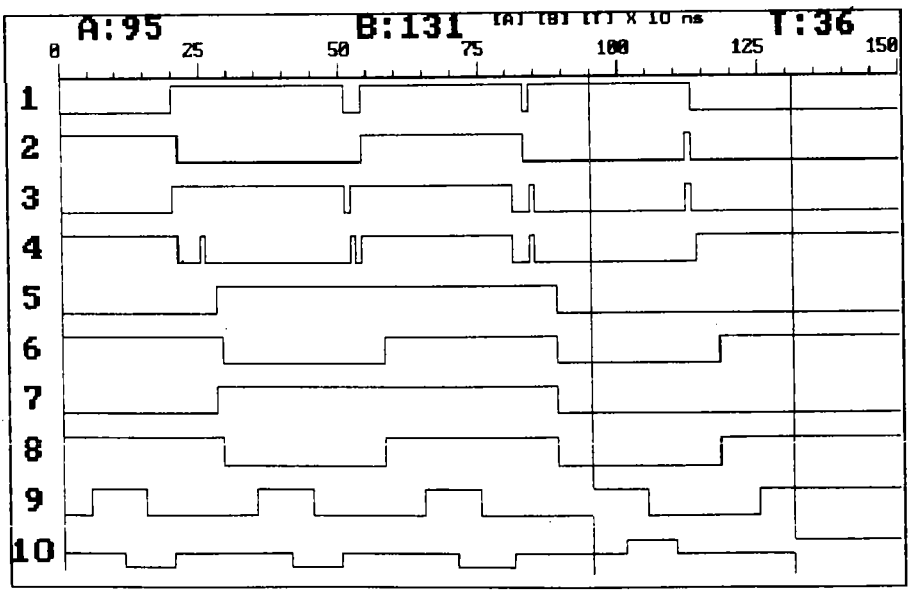

Fig. 8.102. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 7). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

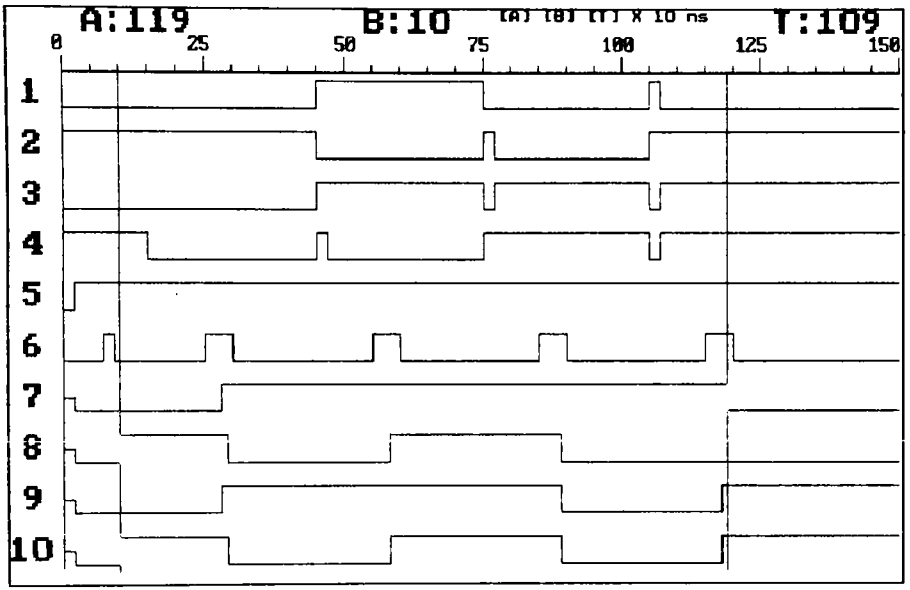

Fig. 8.103. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 8). 1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

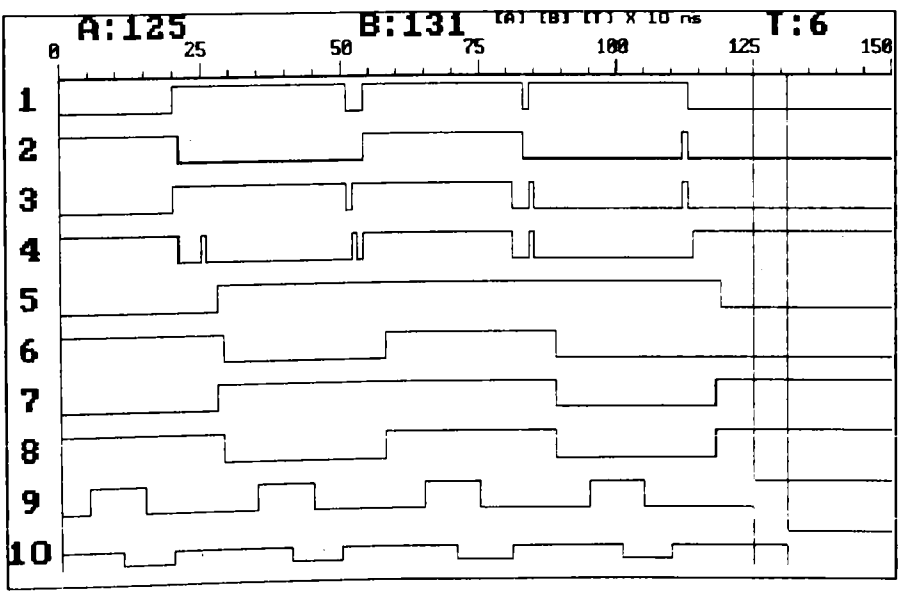

Fig. 8.104. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 8). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

simulate în cronogramele din fig. 8.103 şi, respectiv, fig. 8.104.

9. OPERAŢIA LOGICĂ ROTAŢIE DREAPTA.

Rezultatul UAL: C=Å*=0110

↑ eroare

În fig. 8.105 se vizualizează formarea semnăturii pentru rezultatul eronat din UAL, iar în fig. 8.106 se arată modul de detecţie a erorii şi activarea semnalului EROARE de către modulul COMP1.

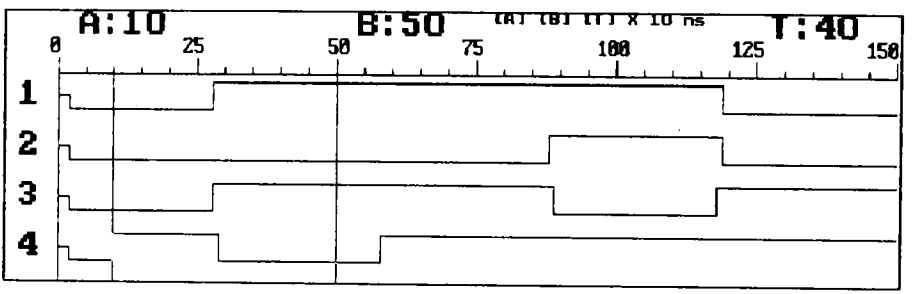

Fig. 8.105. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 9).
1-sigC_0, 2-sigC_1, 3-sigC_2, 4-sigC_3.

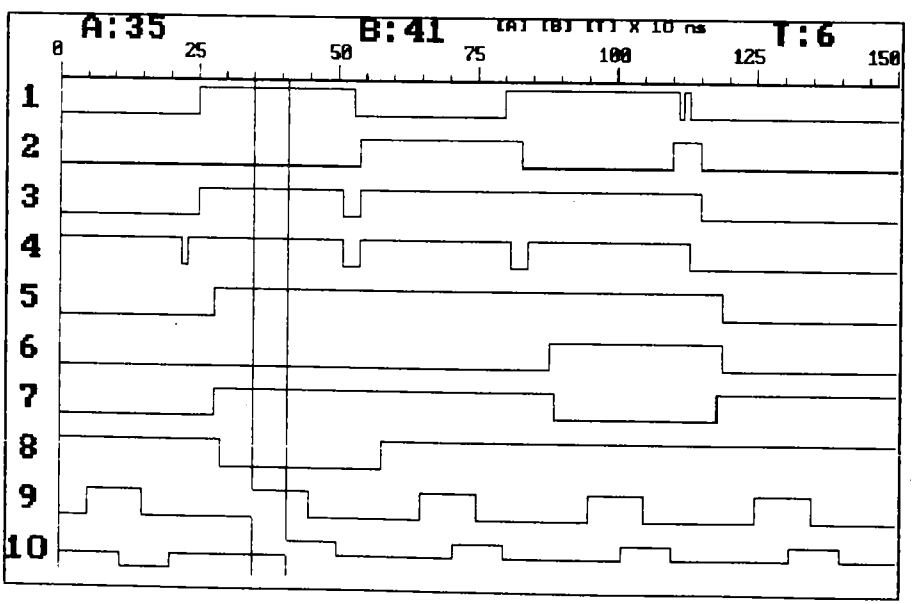

Fig. 8.106. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 9). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

10. OPERAŢIA LOGICĂ ROTAŢIE STÎNGA.

Rezultatul UAL: C=Å*=1111

↑ eroare

În acest caz, formarea semnăturii pentru rezultatul eronat din UAL este reprezentată în fig. 8.107, iar în fig. 8.108 se arată modul de detecţie a erorii şi activarea semnalului EROARE de către modulul COMP1.

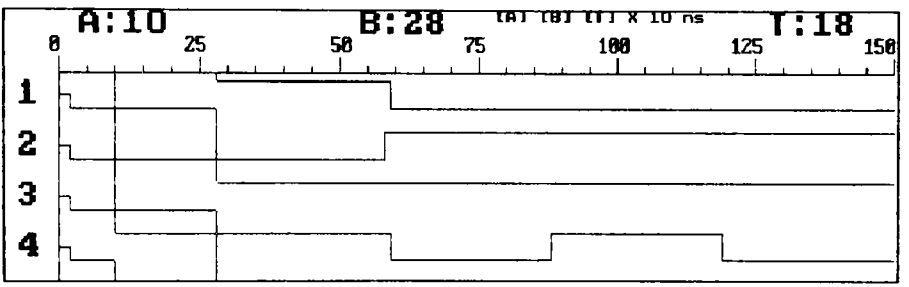

Fig. 8.107. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 10).
1-sigC_0, 2-sigC_1, 3-sigC_2, 4-sigC_3.

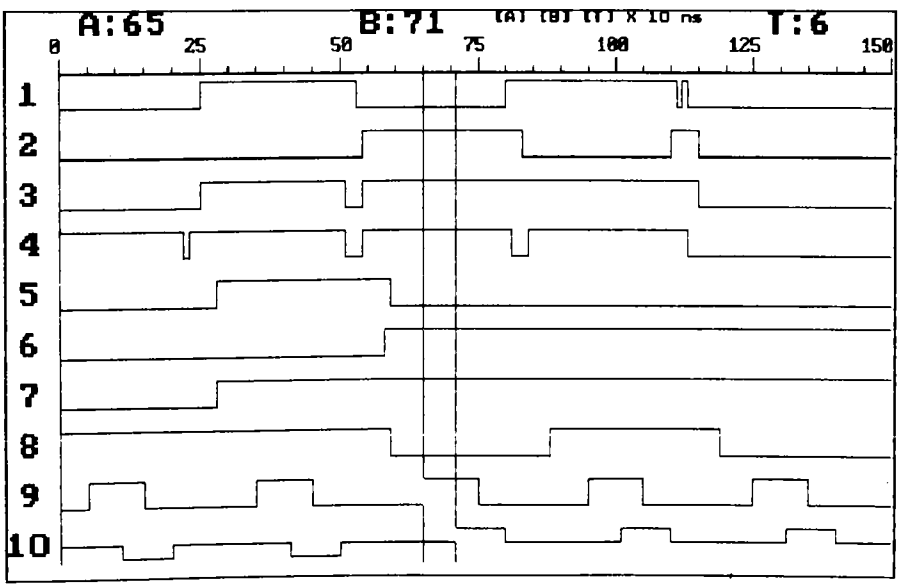

Fig. 8.108. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 10).
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

11. OPERAŢIA ARITMETICĂ DE ADUNARE.

Rezultatul UAL: C=A+B=0110

↑ eroare

Corespunzător acestui rezultat se formează semnătura pentru comparaţie, reprezentată în fig. 8.109.

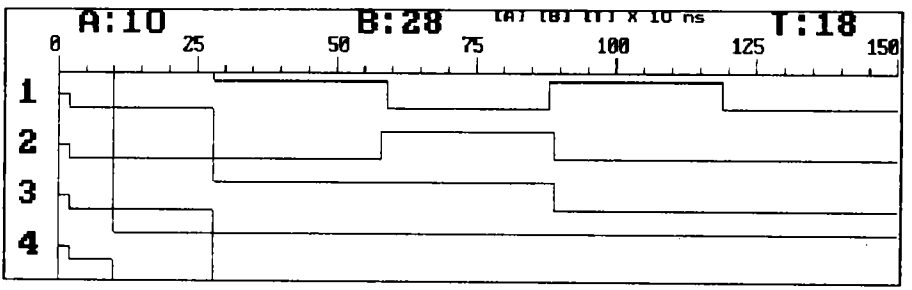

Fig. 8.109. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 11).
1-sigC_0, 2-sigC_1, 3-sigC_2, 4-sigC_3.

Rezultatul comparării cu informaţia de control şi detecţia erorii se observă în fig. 8.110.

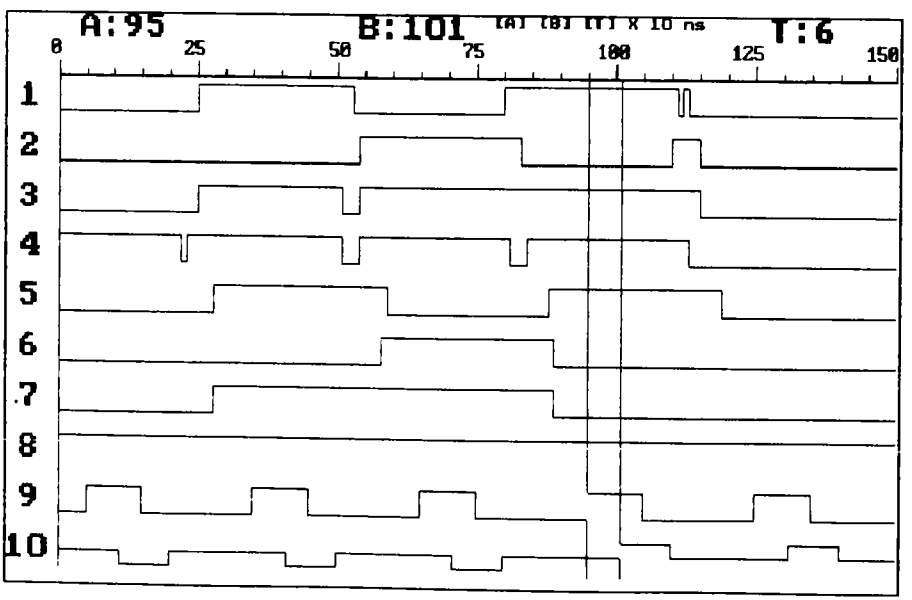

Fig. 8.110. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 11). 1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

12. OPERAŢIA ARITMETICĂ DE ADUNARE (cu depăşire).

Rezultatul UAL: C=A+B=1000

↑ eroare

În mod similar, semnătura rezultatului afectat de eroare şi funcţionarea modulului comparator, cu detecţia erorii sînt simulate în cronogramele din fig. 8.111 şi, respectiv, fig. 8.112.

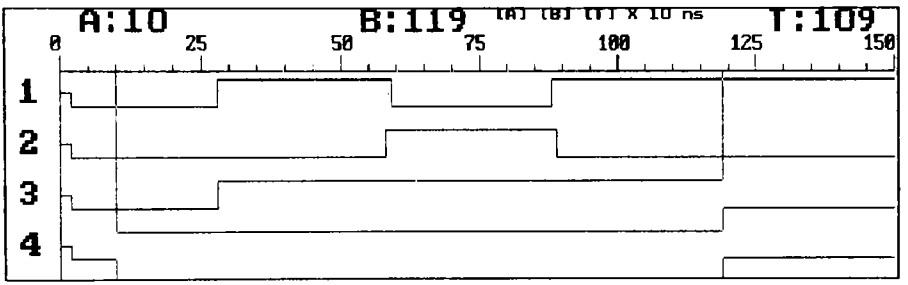

Fig. 8.111. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 12).
1-sigC_0, 2-sigC_1, 3-sigC_2, 4-sigC_3.

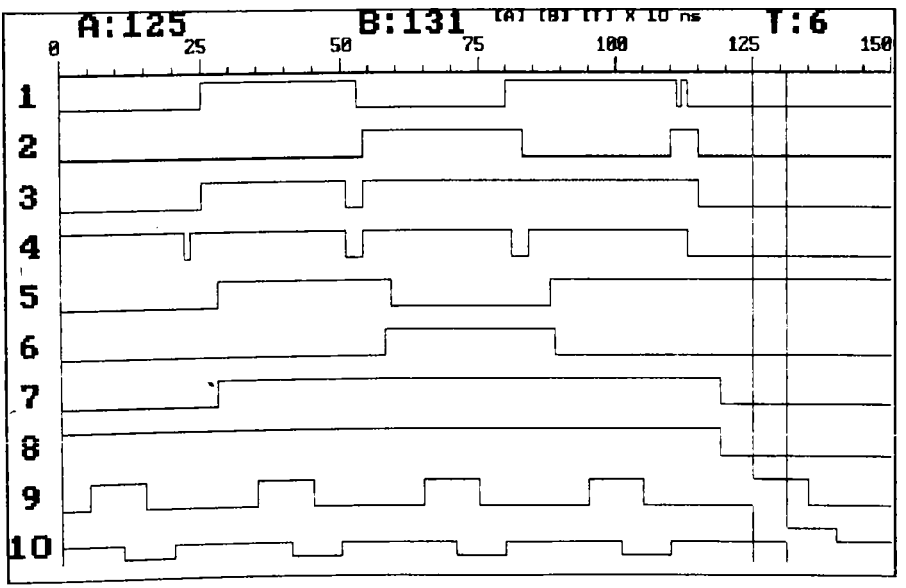

Fig. 8.112. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 12).
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-sigC_0, 6-sigC_1, 7-sigC_2, 8-sigC_3, 9-VC, 10-EROARE.

13. OPERAŢIA ARITMETICĂ DE SCĂDERE.

Rezultatul UAL: C=A-B=0000

↑ eroare

În fig. 8.113 se vizualizează formarea semnăturii pentru rezultatul eronat din UAL, iar în fig. 8.114 se arată modul de detecţie a erorii şi activarea semnalului EROARE de către modulul COMP1.

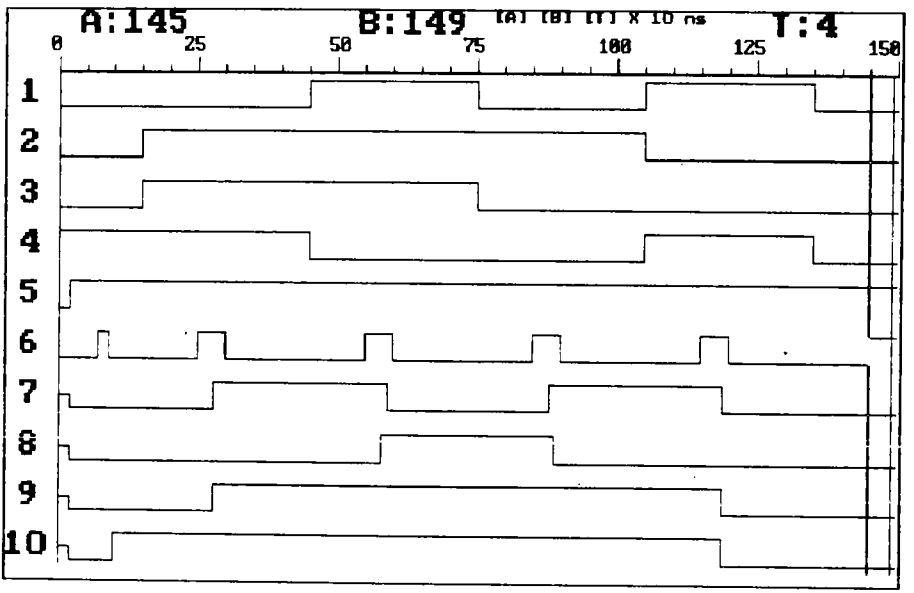

Fig. 8.113. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 13).
1-C_0, 2-C_1, 3-C_2, 4-C_3, 5-RESET, 6-TACTS, 7-sigC_0, 8-sigC_1, 9-sigC_2, 10-sigC_3.

14. OPERAŢIA ARITMETICĂ DE ÎNMULŢIRE (exemplul nr. 1).

Rezultatul UAL: C=A×B=11000001

↑ eroare

În acest caz, formarea semnăturii pe 8 biţi pentru rezultatul eronat din UAL este reprezentată în fig. 8.115, iar în fig. 8.116 se arată modul de detecţie a erorii şi activarea semnalului EROARE de către modulul COMP2.

Configuraţia acestor semnături se poate compara cu cea de la funcţionarea corectă, din fig. 8.85.

Ca şi în situaţiile anterioare activarea semnalului EROARE apare şi pentru operaţia următoare, nr. 15, deoarece odată

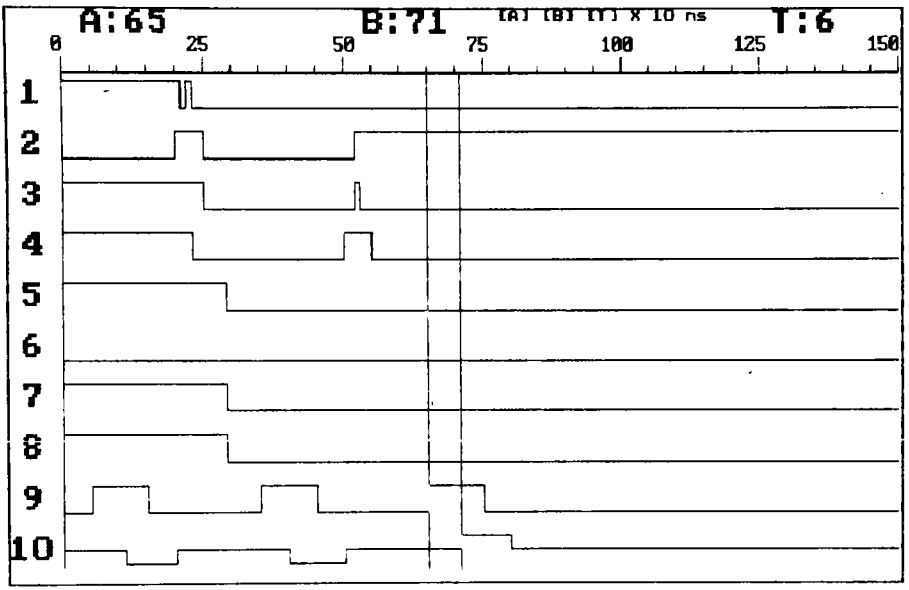

Fig. 8.114. Detecţia erorii în modulul comparator COMP1 (operaţia nr. 13).
1-IC_0, 2-IC_1, 3-IC_2, 4-IC_3, 5-$sigC_0$, 6-$sigC_1$, 7-$sigC_2$, 8-$sigC_3$, 9-VC, 10-EROARE.

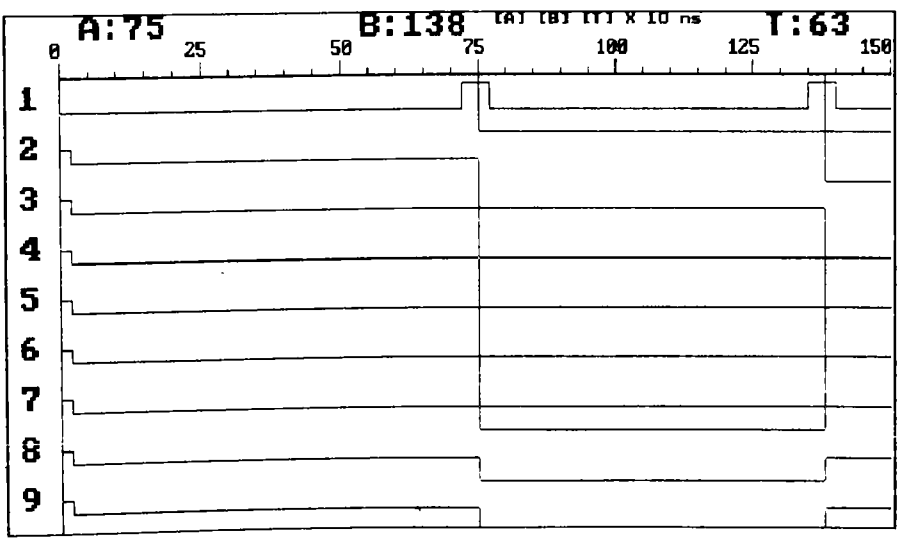

Fig. 8.115. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 14).
1-TACTS, 2-$sigC_0$, 3-$sigC_1$, 4-$sigC_2$, 5-$sigC_3$, 6-$sigC_4$, 7-$sigC_5$, 8-$sigC_6$, 9-$sigC_7$.

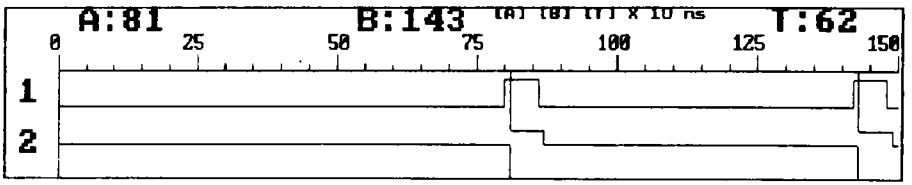

Fig. 8.116. Detecţia erorii în modulul comparator COMP2 (operaţia nr. 14).
1-VC, 2-EROARE.

prezentă eroarea în fluxul informatic, aceasta conduce la semnături diferite şi pentru celelalte operaţii.

15. OPERAŢIA ARITMETICĂ DE ÎNMULŢIRE (exemplul nr. 2).

Rezultatul UAL: C=A×B=00110110
↑ eroare

În mod similar, formarea semnăturii pe 8 biţi pentru rezultatul eronat din UAL este reprezentată în fig. 8.117, iar în fig. 8.118 se arată modul de detecţie a erorii şi activarea semnalului EROARE de către modulul COMP2.

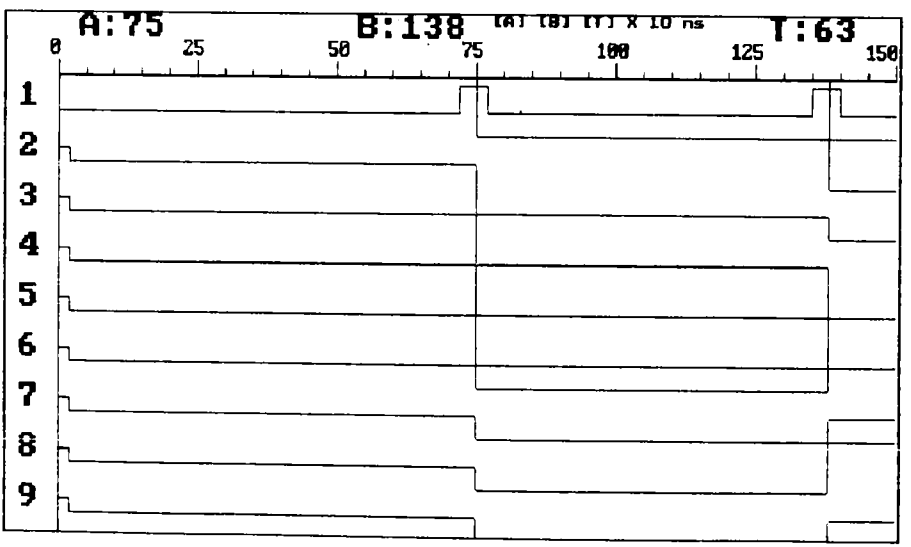

Fig. 8.117. Formarea semnăturii rezultatului eronat din UAL (operaţia nr. 15).
1-TACTS, 2-sigC_0, 3-sigC_1, 4-sigC_2, 5-sigC_3, 6-sigC_4, 7-sigC_5, 8-sigC_6, 9-sigC_7.

În acest caz, activarea semnalului EROARE apare pentru a

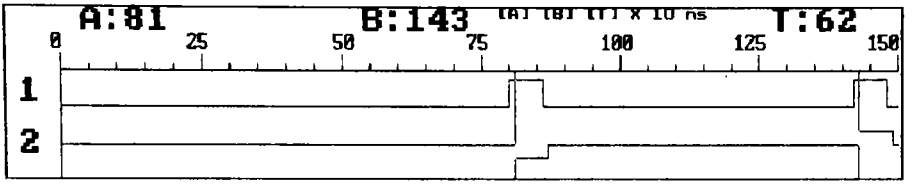

Fig. 8.118. Detecţia erorii în modulul comparator COMP2 (operaţia nr. 15).
1-VC, 2-EROARE.

doua operaţie de înmulţire, prima operaţie fiind considerată corect executată.

În concluzie, din exemplele prezentate de modele de erori, în care s-au considerat atît puneri pe "0", cît şi puneri pe "1", cu o distribuţie uniformă în toate rangurile cuvîntului rezultat din UAL, reiese în mod clar capacitatea de control ridicată a operaţiilor aritmetice şi logice pentru metoda şi dispozitivul propus, gradul de acoperire al erorilor fiind cel specific analizei de semnături. În [99] autorul întreprinde un studiu, în colaborare, asupra capacităţii de detecţie a defecţiunilor la comprimarea bazată pe generarea biţilor de control la coduri ciclice.

CAPITOLUL 9

PERFORMANŢELE DE FIABILITATE ALE SISTEMULUI

9.1. INDICATORI DE FIABILITATE

În general, evaluarea performanţelor de fiabilitate pentru un sistem se realizează prin stabilirea unei expresii analitice şi respectiv a valorilor numerice ale indicatorilor de fiabilitate ai sistemului, cum este de regulă funcţia de fiabilitate R(t), [112].

În literatură, sînt definiţi o serie de indicatori de fiabilitate aplicaţi diferitelor arhitecturi de sisteme [1], [2]. Pentru cazul sistemelor reconfigurabile, sau autotestabile, care generează semnal de EROARE sau STOP, indicatorii de fiabilitate standard nu se consideră adecvaţi [103], propunîndu-se definirea unor indicatori specifici. Caracteristic acestor sisteme autotestabile şi tolerante la defectări este faptul că au două tipuri de ieşiri: o ieşire de DATE şi cealaltă indicatoare de defect, care poate declanşa oprirea sistemului.

Se definesc, în continuare următorii indicatori specifici de fiabilitate, care caracterizează complet un sistem autotestabil:

1. Securitatea sistemului indică probabilitatea ca la ieşire să nu apară o informaţie eronată nedetectată, ceea ce se exprimă prin:

S=1-Prob(informaţie eronată la ieşire şi semnal EROARE neactivat).

2. Fiabilitatea sistemului reprezintă probabilitatea ca sistemul să funcţioneze corect pe o durată de timp alocată şi se exprimă sub forma:

R=Prob(informaţie corectă la ieşire şi semnal EROARE neactivat).

3. Disponibilitatea sistemului este probabilitatea de a fi disponibil la un moment dat, sau probabilitatea ca semnalul de EROARE să nu fi fost activat):

A=Prob(semnalul EROARE nu a fost activat).

Această definiţie a disponibilităţii are în vedere doar condiţia de a nu se activa semnalul EROARE şi diferă, astfel, de

definiţia standard, în care se subînţelege că sistemul funcţionează corect dacă este disponibil [21]. În cazul sistemelor autotestabile informaţia este corectă la ieşire dacă securitatea sistemului este totală.

4. Credibilitatea sistemului indică probabilitatea ca informaţia la ieşire să fie corectă, cu condiţia ca sistemul să fie disponibil:

C=Prob (informaţie corectă la ieşire/sistem disponibil).

9.2. FUNCŢIA DE FIABILITATE A SISTEMULUI AUTOTESTABIL

Se admite sistemul autotestabil, cu modelul structural indicat în fig. 9.1.

Notăm cu $R_f(t)$ funcţia de fiabilitate a modulului funcţional, cu $R_r(t)$ funcţia de fiabilitate a modulului de control redundant, iar cu $R_c(t)$ pe cea a modulului comparator. Ipoteza adoptată, de regulă în literatura de specialitate, privind neapariţia concomitentă a doua defectări de acelaşi tip în modulul funcţional şi în cel redundant este satisfăcută în acest caz prin însăşi structura total diferită a modulului de control. Se adoptă notaţiile F şi FN pentru evenimentele de funcţionare corectă, respectiv funcţionare defectuoasă a modulului funcţional, R şi RN, respectiv C şi CN, pentru aceleaşi evenimente privitoare la modulul de control redundant şi respectiv comparator.

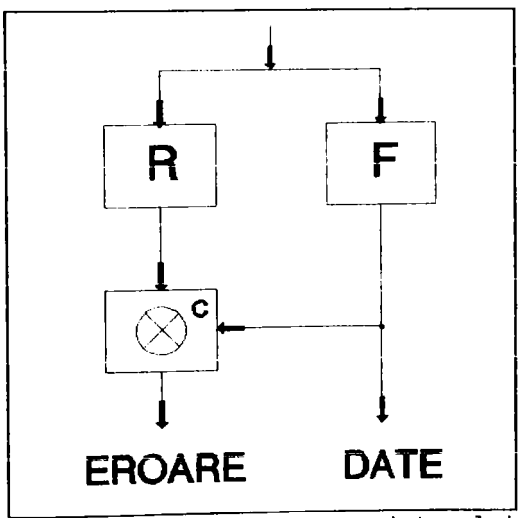

fig. 9.1. Structura s'semului autotestabil.

Sistemul autotestabil de acest tip prezintă două stări în care nu apare semnalul EROARE şi informaţia pe canalul DATE este eronată. Acestea sînt atunci cînd modulul funcţional şi comparatorul sînt defecte şi atunci cînd modulul funcţional,

modulul de control şi comparatorul sînt defecte. Avînd în vedere definiţiile anterioare, rezultă relaţia de calcul a funcţiei de securitate:

$$S(t)=1-[P(FN\wedge R\wedge CN)+P(FN\wedge RN\wedge CN)]=$$
$$1-R_r(t)[1-R_f(t)][1-R_c(t)]-[1-R_f(t)][1-R_r(t)][1-R_c(t)]=$$
$$R_f(t)+R_c(t)-R_f(t)R_c(t). \qquad (9.1)$$

Starea în care sistemul este disponibil este unic determinată în situaţia în care nu se generează semnalul EROARE, iar cele două module, funcţional şi de control, sînt în stare de bună funcţionare, informaţia la ieşire fiind corectă. Prin urmare, funcţia de fiabilitate este de forma:

$$R_{SA}=P(F\wedge R\wedge C)=R_f(t)\cdot R_r(t)\cdot R_c(t). \qquad (9.2)$$

Calculul funcţiei de credibilitate se abordează determinînd mai întîi stările în care informaţia este corectă. Ele sînt FARAC, sau FACACN. Pe de altă parte, stările în care sistemul este disponibil, adică nu apare semnalul EROARE, sînt: FARAC, sau FARACN, sau FARNACN, sau FNARACN, sau FNARNACN. Atfel, credibilitatea sistemului apare sub forma:

$$\frac{P(F\wedge R\wedge C)+P(F\wedge R\wedge CN)+P(F\wedge RN\wedge CN)}{P(F\wedge R\wedge C)+P(F\wedge R\wedge CN)+P(F\wedge RN\wedge CN)+P(FN\wedge R\wedge CN)+P(FN\wedge RN\wedge CN)}=$$
$$=\frac{R_f(t)R_r(t)R_c(t)+R_f(t)R_r(t)[1-R_c(t)]+R_f(t)[1-R_r(t)][1-R_c(t)]}{R_f(t)R_r(t)R_c(t)+R_f(t)R_r(t)[1-R_c(t)]+R_f(t)[1-R_r(t)][1-R_c(t)]+}$$
$$+[1-R_f(t)]R_r(t)[1-R_c(t)+[1-R_f(t)][1-R_r(t)][1-R_c(t)] \qquad (9.3)$$

9.2.1. CALCULUL FUNCŢIILOR DE FIABILITATE ALE MODULELOR

În marea majoritate a cazurilor practice, analizele de fiabilitate se bazează pe ipoteza simplificatoare a independenţei defectărilor elementelor din sistem. În anumite situaţii se impune o analiză mai aprofundată a fiabilităţii în ipoteza considerării dependenţei defectărilor elementelor din sistem.

Metodele uzuale de analiză a fiabilităţii sînt însă laborioase, în cazul considerării dependenţei defectărilor componentelor sistemului, implicînd astfel o abordare bazată pe anumite modele matematice [20].

Lucrarea de faţă nu îşi propune abordarea unor asemenea metodologii, acestea constituind un studiu în sine şi, prin urmare, în continuare se vor utiliza metode clasice de analiză a fiabilităţii previzionale pe baza calculului probabilităţilor. Scopul acestor calcule este de a scoate în evidenţă performanţele soluţiei propuse de autotestare, care utilizează un modul de verificare structural diferit de cel funcţional.

Considerînd modulele componente în mod individual, fără rezervare, structura acestora se poate considera serie, din punct de vedere fiabilistic. Prin urmare, relaţia dintre evenimentul constituit de funcţionarea corectă a modulului M şi evenimentele de funcţionare corectă a elementelor componente, E_i (i=1,2,...,n) este:

$$M=E_1 \wedge E_2 \wedge \ldots \wedge E_n . \tag{9.4}$$

Relaţia generală de calcul a fiabilităţii va fi:

$$R_{M(n)}=P(M)=P(E_1 \wedge E_2 \wedge \ldots \wedge E_n), \tag{9.5}$$

prin $R_{M(n)}$ înţelegîndu-se probabilitatea de funcţionare fără defecţiuni a sistemului, avînd structura serie formată din n elemente.

Considerînd evenimentele independente, se obţine:

$$R_{M(n)}=\prod_{i=1}^{n} P(E_i)=\prod_{i=1}^{n} R_i , \tag{9.6}$$

unde $R_i=P(E_i)$ este funcţia de fiabilitate a elementului i.

În cazul distribuţiei exponenţiale, rezultă:

$$R_{M(n)}=\exp\left(-\sum_{i=1}^{n}\lambda_i \cdot t\right), \tag{9.7}$$

unde λ_i este intensitatea de defectare a elementului i.

9.2.2. EVALUAREA PERFORMANŢELOR SISTEMULUI AUTOTESTABIL

Analizînd în mod comparativ complexitatea modulului de control redundant cu cea a unor module UAL convenţionale [104-110], s-a putut estima redundanţa implicată de soluţia de control propusă. Astfel, a rezultat o valoare de 78% din totalul de porţi necesare la sinteza unei UAL convenţionale. Considerînd lungimea operanzilor pe 4 biţi, conform studiului executat, a rezultat necesar pentru formarea informaţiei de control, fără blocurile de formare a semnăturii, un număr de 342 de porţi logice. În cazul cel mai defavorabil, al redundanţei maxime implicate de formarea în paralel a semnăturilor, rezultă un număr suplimentar de 287 de porţi. Astfel, se poate aprecia redundanţa determinată de modulul de control la un număr de 629 de porţi. Dat fiind liniaritatea demonstrată a soluţiei de control, se poate estima spre exemplu complexitatea modulului de control, pentru lungimea operanzilor de 16 biţi, la un total de 2516 de porţi. Această valoare obţinută este condiţionată de realizarea modulului de control în variantă integrată pe aceeaşi pastilă de Si cu microprocesorul ce urmează a fi controlat, beneficiindu-se de unicitatea unui bloc de comandă.

Luînd în considerare şi numărul de porţi necesare pentru sinteza blocurilor de formare în paralel a semnăturilor operanzilor şi a rezultatului format în UAL rezultă un necesar de 834 de porţi, ceea ce pentru operanzi cu lungimea de 16 biţi, conduce la un număr de 3336 de porţi. Ţinînd cont, spre exemplu, de totalul de 23000 de porţi ale microprocesorului MC 68000, rezultă o redundanţă de aproximativ 14%, ceea ce se poate considera acceptabil, ţinînd cont că prin formarea semnăturilor operanzilor se pot controla şi operaţiile de transfer, atît interne cît şi externe microprocesorului (v. par. 7.1).

Spre comparaţie, în [49] este prezentat un modul de monitorizare pentru microprocesorul MC 68000, bazat pe un alt principiu de control (SIS-streams instruction signatured), care a necesitat un total de 3947 de porţi şi 5435 octeţi de memorie, adică o redundanţă de aproximativ 17%.

În ceea ce priveşte blocul comparator, acesta necesită un total de 51 de porţi, respectiv 204 porţi pentru cazul operanzilor de 16 biţi.

Corespunzător datelor prezentate în [1],[111] s-a putut aprecia valoarea medie pentru intensitatea de defectare a unei porţi integrate pe scară largă la valoarea $\lambda_{med}=0,01\cdot10^{-6}\ h^{-1}$.

În aceste condiţii, pe baza rel. (9.7) se pot determina valorile de fiabilitate pentru fiecare modul al sistemului autotestabil. Astfel, pentru modulul funcţional (UAL), modulul redundant de control şi respectiv modulul comparator rezultă următoarele relaţii de calcul ale fiabilităţii:

$$R_f=\exp(-3225\cdot0,01\cdot10^{-6}\cdot t), \qquad (9.8)$$

$$R_r=\exp(-2516\cdot0,01\cdot10^{-6}\cdot t), \qquad (9.9)$$

$$R_c=\exp(-204\cdot0,01\cdot10^{-6}\cdot t). \qquad (9.10)$$

În fig. 9.2 se reprezintă valorile de fiabilitate ale celor

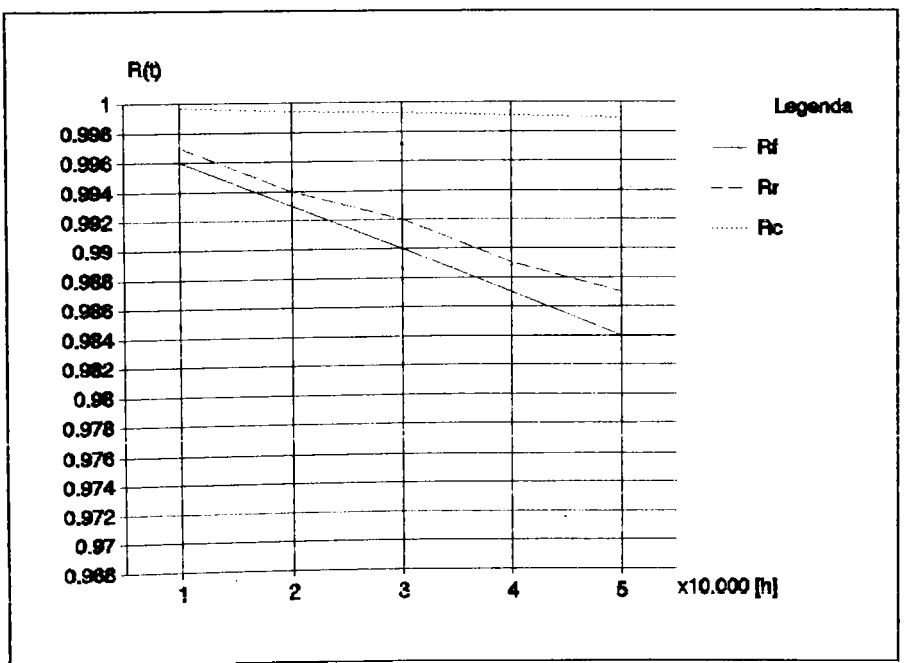

Fig. 9.2. Funcţiile de fiabilitate ale modulelor sistemului autotestabil.

trei module. Variaţia în timp este exponenţială, cu rata de defectare a fiecărui modul proporţională cu numărul de porţi componente.

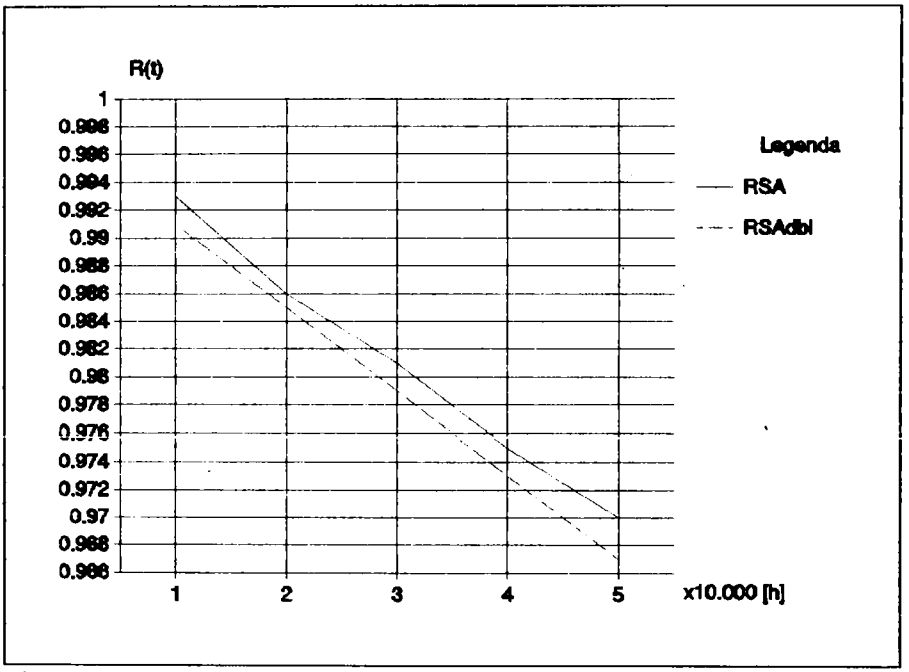

Fig. 9.3. Funcţiile de fiabilitate ale sistemului autotestabil prin modulul de control propus, RSA, respectiv prin dublare, RSA_{dbl}.

Analiza performanţelor de fiabilitate ale sistemului autotestabil propus se va face comparativ faţă de soluţia clasică a dublării modulului funcţional, prin evaluarea valorilor indicatorilor de fiabilitate definiţi anterior. Astfel, în fig. 9.3 se reprezintă valorile funcţiei de fiabilitate în cele două cazuri. Prin RSA s-a reprezentat funcţia de fiabilitate a sistemului autotestabil prin blocul de control propus în lucrare, iar prin RSA_{dbl} funcţia de fiabilitate a sistemului autotestabil prin dublare. Valorile RSA_{dbl} au fost determinate pornind de la aceeaşi relaţie (9.2), în care evident R_r are aceeaşi valoare cu R_f. Performanţele superioare ale funcţiei de fiabilitate pentru structura propusă de sistem autotestabil se datorează redundanţei

mai mici implicate, faţă de soluţia cu dublare.

Comparînd valorile obţinute pentru RSA cu cele ale fiabilităţii modulului funcţional, R_f, se constată o scădere a valorilor funcţiei de fiabilitate pentru sistemul autotestabil. Acest fapt este normal, fiind de aşteptat în toate cazurile care implică structuri redundante. Performanţele de a asigura o informaţie corectă la ieşire, ale sistemelor autotestabile, se evaluează însă prin ceilalţi indicatori de fiabilitate definiţi.

În fig. 9.4 sînt reproduse valorile indicatorilor de fiabilitate, reprezentaţi prin securitate, S(t) şi credibilitate, C(t), pentru cele două soluţii.

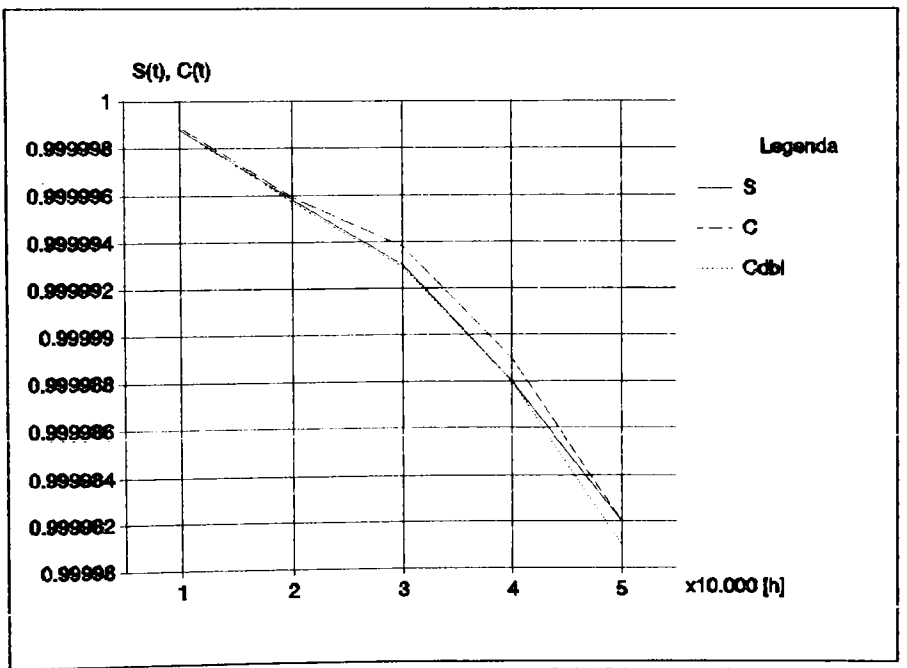

Fig. 9.4. Valorile indicatorilor credibilitate şi securitate pentru soluţia propusă, respectiv pentru cea cu dublare.

Trebuie remarcat faptul că valorile indicatorului de securitate sînt aceleaşi pentru ambele soluţii. Aceasta apare evident dacă se ţine seama de definiţia adoptată anterior şi de rel. (9.1). În schimb, se observă superioritatea soluţiei de control propusă prin valorile mai mari ale indicatorului de

credibilitate, C, faţă de cele ale soluţiei cu dublare, C_{dbl}. Astfel, sistemul autotestabil propus asigură aceeaşi securitate a informaţiei la ieşire, dar cu o credibilitate mai mare, cu alte cuvinte cu o încredere mai mare acordată informaţiei disponibile la ieşire.

CAPITOLUL 10

CONCLUZII

Cele prezentate în lucrare s-au referit la o soluţie originală propusă pentru realizarea unui procesor cu facilităţi de autoverificare. Acest procesor va constitui elementul de bază în dezvoltarea unui sistem tolerant la defecţiuni. Cea mai simplă configuraţie a unui astfel de sistem ar putea fi constituită din două asemenea procesoare.

Noutatea soluţiei propuse şi unul din avantajele majore ale acesteia constă pe de-o parte în controlul on-line, prin semnături, a circulaţiei informaţiei în interiorul procesorului (v. fig. 7.1) şi pe de altă parte în controlul prin semnături a operaţiilor aritmetice şi logice, executate în UAL. În felul acesta s-a reuşit îndeplinirea unui mare deziderat al testării, şi anume asigurarea continuităţii fluxului informaţional de control atît în interiorul cît şi în exteriorul procesorului, simplificîndu-se şi generalizîndu-se soluţia de control, pe bază de semnături.

În continuare, se prezintă spre comparaţie soluţiile de autotestare utilizate în cazul a două microprocesoare de 32 biţi, Motorola MC 68020 şi Digital Equipment Vax [94]. La 68020 soluţia se bazează pe conceptul de partiţionare şi utilizarea microinstrucţiunilor pentru a implementa proiectarea structurală pentru testabilitate, printr-o logică de control complicată şi generarea testelor funcţionale pentru blocurile procesorului. La procesorul Vax se exploatează caracteristicile inerente ale structurii, prevăzîndu-se circuite adiţionale pentru testabilitate, avînd o strategie de testare asemănătoare. Întregul circuit este partiţionat în blocuri hardware, iar fiecărui bloc i se aplică testele cele mai potrivite corespunzător anumitor criterii. În alegerea configuraţiilor circuitelor adiţionale de testare se urmăreşte o optimizare, avîndu-se în vedere aria de integrare implicată şi degradarea performanţelor procesorului. În această direcţie a proiectării pentru testabilitate a microprocesoarelor, o variantă alternativă la cele două soluţii discutate mai sus, pe baza conceptului 'scan-design', este prezentată în [94]. Nici una din aceste

soluţii nu asigură însă capacitatea de autotestare on-line a procesorului. Toate cele trei soluţii sînt similare în ceea ce priveşte strategiile de bază de testare, şi anume partiţionarea întregului circuit în blocuri hardware şi aplicarea celui mai potrivit set de vectori de test, potrivit cu anumite criterii.

O monitorizare pe bază de semnături a fluxului informaţional vehiculat de procesor este abordată în [49], însă soluţia este complicată, sub aspectul intervenţiilor atît la nivel hardware cît şi software. De asemenea, capacitatea de detecţie a erorilor pentru întreg blocul de control este relativ scăzută, de 82%, mult inferioară capacităţii de detecţie a erorilor pentru soluţia propusă în prezenta lucrare, care este cea tipică de la comprimarea bazată pe generarea biţilor de control separaţi de cei utili la coduri ciclice (de exemplu 99,998% pentru o semnătură pe 16 biţi).

Soluţia propusă reprezintă, de asemenea, o abordare structurală a strategiei de testare. Avantajul soluţiei îl constituie faptul că se realizează, în realitate, o verificare, prin comparare în timp real, a corectitudinii totalităţii informaţiei vehiculată de procesor. Astfel, pentru fiecare cuvînt al informaţiei, rezultate fie în urma unei operaţii de transfer, fie în urma unei operaţii aritmetice sau logice, se formează o semnătură aferentă utilizată pentru control. Evident, o verificare asemănătoare se poate face într-o soluţie dublată, ca cea prezentată în [95], în care se compară informaţia a două procesoare, ce funcţionează în paralel. Dezavantajele unei asemenea soluţii sînt legate de dificultăţile întîmpinate în sincronizarea celor două procesoare, de numărul mare de semnale de test, ca şi de costul ridicat. Suluţia dublării, deşi simplă nu este acceptabilă în cele mai multe cazuri şi pentru faptul că nu poate asigura detectarea defecţiunilor identice simultane-cum ar fi cele de concepţie-ale celor două module. Pentru o funcţionare tolerantă la defecţiuni însă, ansamblul mai solicită o schemă de localizare a defecţiunilor, sau rularea de programe de autotest, pentru a determina care din procesoare este defect. În acest sens, soluţia propusă oferă avantajul că procesoarele reprezintă de fapt module autoverificabile şi prin urmare, o variantă dualã, tolerantă la defecţiuni, nu mai necesită circuite adiţionale de localizare a defecţiunilor. Un alt avantaj al

soluţiei este acela că, odată cu controlul operaţiilor de transfer interne procesorului, se realizează şi controlul corectitudinii informaţiei pe magistrala de date externă şi implicit şi a corectei funcţionări a memoriei.

În ceea ce priveşte aria de integrare suplimentară implicată, aceasta depinde de: (1) numărul rangurilor suplimentare ale registrelor de date interne, pentru biţii de control, (2) realizarea schemelor de generare a semnăturii şi (3) realizarea schemelor de generare a caracteristicii mutuale de control. Aceste scheme au fost implementate printr-o structură logică regulată, de tip PLA şi prin urmare se estimează că aria suplimentară implicată de această soluţie ar fi minimă. Privitor la performanţele dinamice ale procesorului controlat, s-a demonstrat că acestea nu vor fi afectate, deoarece verificarea se execută on-line într-un timp mai scurt ca cel de procesare, la viteza normală a procesorului.

În prezentarea soluţiei s-a presupus integrarea circuitelor adiţionale în aceeaşi capsulă procesoare. O variantă alternativă ar putea fi realizată în forma a două capsule, cum este spre exemplu varianta prezentată în [83]. O capsulă ar fi ocupată de procesorul propriu-zis, iar cealaltă capsulă ar fi coprocesorul de control, care ar opera numai cu semnăturile corespunzătoare datelor de pe magistrală.

Un alt aspect, care trebuie menţionat se referă la metoda de codificare aleasă. Abordarea pînă în prezent a problematicii controlului on-line a operaţiilor aritmetice şi logice este foarte sărac reprezentată în literatură, iar realizările practice moderne în acest sens sînt ca şi inexistente. Soluţiile actuale de verificare a operaţiilor din UAL, prezentate în literatură, se referă doar la testarea UAL prin programe de test şi nicidecum la controlul on-line. Principial controlul on-line al operaţiilor de procesare ar putea fi efectuat similar şi după modul, prin resturi, avînd în vedere principiile teoretice prezentate în cap.4. Soluţia de a utiliza coduri cu resturi necesită realizarea unor scheme de formare a resturilor, în locul celor de formare a semnăturilor, precum şi dispozitive UAL suplimentare. Schemele de formare a resturilor însă, prezentate în [44],[89],[96], au o complexitate ridicată, motiv pentru care sînt puţin răspîndite în practică. Ele sînt mult mai complicate decît cele de formare

a semnăturii prezentate în lucrare, de unde rezultă avantajul variantei de codificare propusă pentru implementarea structurii de control.

Concluziv, contribuţiile originale ale autorului pe capitole sînt următoarele:

1. Sistematizarea principalelor aspecte prezentate în literatură asupra problematicii fiabilităţii echipamentelor numerice, privită prin prisma operaţiilor de testare.

1.3. O împărţire sintetică în două clase, hard şi soft, a tipurilor de defecţiuni şi analiza modurilor lor de manifestare în echipamentele numerice.

2.1. Analiza detaliată şi sistematică a principalelor strategii pentru asigurarea toleranţei la defectări a sistemelor numerice.

3. Abordarea sintetică a problematicii redundanţei informaţionale în vederea detecţiei şi corecţiei erorilor şi clasificarea codurilor de control pentru:

3.1. Memorii de mare viteză;

3.2. Memorii de masă;

3.3. Controlul erorilor unidirecţionale;

3.4. Operaţii de procesare.

4. Analiza şi aprofundarea posibilităţilor de utilizare a codurilor cu resturi la controlul operaţiilor aritmetice şi logice. Reiese complexitatea aplicării practice a acestei metode de control.

5. Propunerea unei noi metode de testare în timp real a procesoarelor. Se arată modalitatea de organizare a fluxului informaţional în vederea controlului operaţiilor de procesare. Se configurează următoarele trasee informaţionale:

-Traseul informaţional al datelor;

-Traseul informaţional al unităţii de comandă;

-Traseul informaţional al generării adreselor;

-Traseul informaţional al derulării programului.

Se evidenţiază faptul că deoarece anumite blocuri din structura procesorului utilizează în comun aceleaşi trasee informaţionale, pentru a asigura o verificare completă nu mai este necesară testarea fiecărui bloc separat, ca în metodele clasice, ci doar efectuarea controlului asupra unui lanţ informaţional continuu. Noua metodă propusă rezolvă deziderатul

verificării complete a procesoarelor în timpul funcţionării acestora, printr-un control unitar, atît asupra operaţiilor de transfer şi stocare cît şi asupra operaţiilor aritmetice şi logice.

6. Propunerea unei noi soluţii de control a operaţiilor aritmetice şi logice pe bază de semnături. Astfel, se rezolvă dificila problemă a verificării blocurilor "active" din structura procesoarelor, care conduc la transformarea informaţiei de bază prin operaţii aritmetice şi/sau logice. Prin introducerea semnăturilor şi la controlul circuitelor "pasive", care transferă şi stochează informaţia, se asigură unicitatea controlului operaţiilor de procesare. Utilizarea pentru control a semnăturilor are marele avantaj al capacităţii ridicate de detecţie a defecţiunilor, specifică pentru codificarea în cod ciclic. De asemenea, soluţia propusă are avantajul că asigură posibilitatea alegerii efectuării controlului atît prin semnături formate serial, cît şi paralel. Soluţia propusă poate constitui baza unor cercetări viitoare privind realizarea şi a corecţiei erorilor la operaţiile aritmetice şi logice.

6.1. Deducerea relaţiilor de control prin semnături pentru principalele operaţii logice. Relaţiile rezultate sînt liniare constînd din egalitatea dintre semnătura rezultatului şi suma modulo 2 a semnăturilor operanzilor cu semnătura unei caracteristici mutuale de control, specifice operaţiei în cauză.

6.2. Deducerea relaţiilor de control prin semnături pentru operaţiile aritmetice de adunare, scădere şi înmulţire. Se definesc caracteristicile mutuale de control pentru operaţiile de adunare şi scădere, astfel încît relaţiile de control prin semnături să rezulte de asemenea liniare. Se dezvoltă o relaţie liniară pentru controlul operaţiei de înmulţire, pe baza utilizării caracteristicii mutuale a operaţiei de adunare. Relaţia de control dedusă are avantajul că asigură convenabil sinteza modulului de control a operaţiei de înmulţire fără a fi necesar accesul în blocul UAL. Relaţiile dezvoltate permit cu uşurinţă extinderea controlului şi asupra operaţiei de împărţire.

7. Elaborarea unei structuri originale de sistem de control pentru operaţiile de procesare. Structura propusă asigură unicitatea controlului prin semnături a informaţiei vehiculată atît în interiorul cît şi în exteriorul elementului de procesare.

Structura propusă asigură controlul în timp real al funcţionării procesorului, fără a afecta performanţele de viteză ale acestuia. Prin utilizarea acestei structuri de control procesorul devine capabil de a se autotesta în timpul funcţionării normale, facilitîndu-se de asemenea, prin aceleaşi semnături, şi controlul dispozitivelor externe de memorare şi de intrare/ieşire.

8. Realizarea sintezei unui bloc de control a operaţiilor aritmetice şi logice, verificarea corectei funcţionări şi a capacităţii de control, confirmîndu-se rezultatele teoretice şi ipotezele originale expuse de autor.

8.1. Propunerea unei variante moderne de sinteză a blocului de control a operaţiilor aritmetice şi logice, pe baza unei reţele logice regulate, în vederea integrării pe scară foarte largă. Rezultă avantajul deosebit al utilizării şi realizării blocului de control propus, în aceeaşi capsulă procesoare. Se demonstrează astfel posibilitatea realizării unui procesor autotestabil, ce constituie nucleul oricăror structuri tolerante la defecţiuni.

8.2. Verificarea corectei funcţionări a blocului de control a operaţiilor aritmetice şi logice prin simulare şi demonstrarea practică a generalităţii metodei de control prin semnături a operaţiilor de procesare.

8.2.1. Elaborarea unui program de simulare în limbaj Turbo C în vederea modelării şi verificării exhaustive a funcţionării blocului de control propus.

8.2.2. Exemplificarea simulării funcţionării blocului de control pe baza unor modeluri pentru fiecare tip de operaţie aritmetică şi logică. Modelurile exemplificate sînt alese astfel încît să se asigure o repartiţie uniformă a ponderilor binare în cuvintele de informaţie. Se reprezintă cronogramele tuturor semnalelor care intervin în funcţionarea blocului de control. Se demonstrează că performanţele dinamice ale structurii de control sînt mult superioare timpului de execuţie a operaţiilor aritmetice şi logice, în felul acèsta nefiind afectată viteza de operare a procesorului ca urmare a efectuării controlului.

8.3. Verificarea capacităţii de detecţie a erorilor pentru blocul de control propus, pe baza unor modeluri de eroare simulate. Ponderea biţilor de eroare în cuvintele de date are, de asemenea, o repartiţie uniformă. Se arată modalitatea de

detecţie a erorilor, prin activarea unui fanion de eroare, pentru fiecare tip de operaţie aritmetică şi logică. Metoda de control propusă avînd la bază relaţii liniare între semnături are avantajul că păstrează capacitatea ridicată de detecţie a erorilor prin coduri ciclice.

9. Evaluarea performanţelor de fiabilitate ale unui sistem autotestabil, asigurate prin utilizarea blocului de control propus. Se utilizează indicatori specifici de fiabilitate, definiţi pentru sisteme autotestabile.

9.2.2. Se evaluează complexitatea blocului de control propus, rezultînd o redundanţă mai mică decît în cazul aplicării metodei de autocontrol prin dublare, sau a altor metode de autocontrol în timp real, prezentate în literatură. Aceasta, în condiţiile în care se soluţionează şi problema dificilă a controlului operaţiei de înmulţire. Se dovedeşte avantajul metodei şi a blocului de control propuse, prin valorile superioare obţinute pentru indicatorii de fiabilitate.

În final, se poate afirma că soluţia propusă în lucrare pentru abordarea controlului unitar prin semnături a operaţiilor aritmetice şi logice şi a operaţiilor de transfer, rezolvă în ansamblu problema autotestării în timp real a procesoarelor, aducînd o contribuţie importantă la perspectiva realizării de sisteme numerice integrate tolerante la defecţiuni.

BIBLIOGRAFIE

1. Geber, T. ş.a.,"*Fiabilitatea şi mentenabilitatea sistemelor de calcul*", Editura Tehnică, Bucureşti, 1984

2. Cătuneanu, V.N., Bacivarof, I.C., "*Fiabilitatea sistemelor de telecomunicaţii*", Editura Militară, Bucureşti, 1985

3. Bennetts, R.G.,Maunder, C.,Morris, D.,"*Proceedings of Automatic Testing and Test Instrumentation, Tutorial*", Network Events Ltd., 1983.

4. Bouwens, A.J., Koldenhof, H., Rijsdijk, F., "*Digital Instrument Course - Part 5. Logic Analyzers*",N. V. Philips Gloeilampen fabriken, Test ans Measuring Departament, Eindhoven, 1981.

5. Medich, Cathy I., "*Component Level Self-Diagnosis in a Digital System*", Teză de doctorat, Massachussets Institute of Technology, 1979

6. Chang, H.I., Manning, E., Metze, C., "*Fault Diagnosis of Digitals Systems*", Wiley-Interscience, New York, 1970

7. Abramovici, M.,Breuer, M.A., "*Fault Diagnosis on Effect-Cause Analysis: An Introduction*", Proceedings of the 17-th Design Automation Conference p,17-24.

8. Ardelean, I., Giuroiu, H., Petrescu, L.L., "*Circuite integrate CMOS. Manual de utilizare*", Editura Tehnică, Bucureşti, 1986.

9. Vernon, N.B., "*Testing CMOS Digital Logic Systems*". Application Report, Membrain Ltd.

10. Muehldorf, E.I., Savkar, A.D., "*LSI Testing-An Overview*", IEEE Transaction on Computers, January, 1981, p.1-17.

11. Breuer, M.A., Friedman, A.D., "*Diagnosis and Reliable Design of Digital Systems*", Computer Science Press Ins.

12. Iosupovicz, A., "*Optimal Detection of Bridge Fualts and Stuck At Faults in Two-Level Logic*", IEEE Transaction on Computers, May 1978, p.452-455.

13. Scheiber, S.F., "*Dollars to Doughnuts-or , Testing the World's Cheapest Complex Parts*", Test and Measurement World, October 1985

14. Scheiber, S.F., "*Memory Testing - New Technologies, New Challenges, New Solutions*", Test and Measurement World, October 1984

15. Kookootsedes, G., "*Using High Purity Coatings for Alpha Particle Protection*", Microelectronic Manufacturing and Testing, July 1987.

16. Ost, G., "*The Anatomy of Burn-In-An Analysis*", Electronic

Production, February 1985, p.27-31.

17. Persons, R., "*Is There a Future for Semiconductor-Device Burn-In?*", Microelectronic Manufacturing and Testing, August 1987, p.36-37

18. Schell, M.,Osteros, L., " *Automated Burn-In Board Tester Requirements and ROI*", Microelectronic Manufacturing and Testing, August 1987, p.28-29

19. Nelson, P.V., "*Fault-Tolerant Computing: Fundamental Concepts*", Computer, Vol.23, No.7, July 1990, p.19-25

20. Geist, R., Trivedi, k., "*Reliability Estimation of Fault-Tolerant Systems: Tools and Techniques*", Computer, Vol.23, No.7, July 1990, p.52-61

21. Cătuneanu, V.M., Bacivarof, A., "*Structuri electronice de înaltă fiabilitate. Toleranţa la defectări*", Editura Militară, Bucureşti, 1989.

22. Gaston, C., " *Le circuit detecteur et correcteur d'erreurs simples et doubles* ", Electronique Industrielle, No.20, 15 sept. 1981, p.41-50.

23. Bazes, M., Farrell, L., May, B., Mebel, M., " *Keep memory design simple yet cull-single -bit errors*", Electronic Design, Sept. 30 1981, p. 195-201.

24. McCluskey, E.J., Bozorgui-Nesbat, S., "*Design for Autonomous test*", IEEE Transactions on Computers, Vol.C-30, No.11, Nov. 1981, p.866-874.

25. McCluskey E.J., "*Design Techniques for Testable Embedded Error Checkers*", Computer, Vol.23, No.7, July 1990, p.84-88.

26. Nicolaidis, M., Courtois, B., "*Self-Checking logic arrays* ", Microprocesors and Microsystems, Vol.13, No.4, May 1989, p. 281-289.

27. Halbert, M. P., "*Selfchecking computer module based on the Viper microprocessor*", Microprocessors and Microsystems, Vol.12, No.5, June 1988,p.264-270.

28. Koren, I., Singh, A. D., "*Fault Tolerance in VLSI Circuits*", Computer, Vol.23, No.7, July 1990, p. 73-83.

29. Lea, R. M., Bolouri, H. S., "*Fault lolerance : step towards WSI*", IEEE proceedings, Vol. 135 Pt.E., No.6, Nov.1988, p.289-297

30. McConnel, S. R., Siewiorek, D.P., "*Synchronization and Voting*", IEEE Transactions on Computers, Vol. C-30, No.2, Febr. 1981, p. 161-164.

31. Barbour, A.E., Wojcik, A.S., "*A General, Constructive Approach to Fault Tolerant Design Using Redundancy*", IEEE Transactions on Computers, Vol. 38,No.1, Ianuary 1989, p.15-29.

32. Kim, K. H., Welch, H.O., "*Distributed Execution of Recovery Blocks: An Approach for Union Treatment of Hardware and Software Faults in Real Time Aplications*", IEEE Transactions on Computers, Vol.38, No.5, May 1989, p.626-636.

33. Laprie, J.C., Arlat, J., Beounes, C., Kanoun, K., "*Definition and Analysis of Hardware and Software-Fault-Tolerant Architectures* ", Computer Vol.23, No.7, July 1990, p.39-51.

34. Cullman, G., "*Coduri detectoare şi corectoare de erori*" (trad. limba franceză),Ed.Tehnică, Bucureşti 1972.

35. Ionescu, D., "*Codificare şi coduri*", Ed. Tehnică, Bucureşti, 1981.

36. Fujiwara, E., Pradham, D.K., "*Error-Control Coding in Computer*" ,Vol.23, No.7, July 1990, p.63-72.

37. Okano, H., Imai, H., "*A Construction Method of High-Speed Decoders Using ROM's for Bose-Chaudhuri-Hocquenghem and Reed-Solomon Codes*" , IEEE Transactions on Computers, Vol. C-36, No.10, October 1987, p.1165-1171.

38. Kanai, T., "*An Improvement of Reability of Memory System with Skewing Reconfiguration*", IEEE Transactions on Computers, Vol.C-30, No.10, October 1981, p. 811-812.

39. Gaitanis, N., "*TSC-error C/D circuits for SEC/DED product codes* ", IEEE Proceedings, Vol. 135, Pt.E.,No.5, Sept. 1988, p.253-258.

40. Senguota, A., Chattopadhyay, D.K., Palit, A., Bandyopadhyay, A.K., Choudhury, A.K., "*Realization of Fault-Tolerant Machines-Linear Code Application*", IEEE Transactions on Computers, Vol.c-30, No.3, March 1981, p.237-240.

41. Dao, T.T.," *SEC/DED Nonbinary Code for* Fault-Tolerant Byte-Organized Memory Implemented with Quaternary Logic", IEEE Transactions on Computers, Vol.C-30, No.9, Sept. 1981, p.662-666.

42. Pradham, D>K>, Reddy, S.M., "*Error Control Techiques for Logic Processors*", IEEE Transactions on Computers, Vol.C-21, No.12, Dec. 1972, p.1331-1337.

43. Krol, T., "*N,K Concept Fault Tolerance*", IEEE Transactions on Computers, Vol.C-35, No.4, Apr.1986, p.339-349.

44. Vlăduţiu, M., Crişan, M., "*Tehnica testării echipamentelor automate de prelucrare a datelor*", Ed.Facla, Timişoara, 1989.

45. Vlăduţiu, M., Crişan, M., "*Sinteza schemelor de formare a resturilor pentru controlul operaţiilor aritmetice şi logice*", Al 7-lea Simpozion Naţional TEF-87, Băile erculane 29-31 oct. 1987, p.317-324.

46. Vlăduţiu, M., Crişan, M., "*Asupra impactului performanţă/cost, la controlul operaţiilor aritmetice prin coduri cu resturi*", Al 7-lea Simpozion Naţional TEF-87, Băile Herculane,

29-31 oct.1987, p.327-334.

47. Vlăduţiu, M., Crişan. M., "*Asupra capacităţii de detecţie a defecţiunilor la coduri cu resturi*", Al 7-lea Simpozion Naţional TEF-87, Băile Herculane, 29-31 oct. 1987, p.334-340.

48. Lu, D.I., "*Watchdog processors and stuctural integrity checking* ", IEEE Transactions on Computers, Vol.C-31, July 1982, p.681-685.

49. Schuette, M.A., Shen, I.P., "*Processor control flow monitoring using signatured instruction streams*", IEEE transactions on Computers, Vol.C-36, No.3, March 1987, p.264-275.

50. Tupkalo, V. N., *"Kontrol logiceskih operaţii na osnove ispolzovania signatur,"* Avtomatizirovannîe sistemî upravlenia i pribori avtomatiki, Vîşa şcola, Harcov, 1988, p.70-77.

51. Bose, B., Pradham, D.K., "*Optimal Unidirectional Error Detecting/Correcting Codes*", IEEE Transactions on Computers, Vol.C-31, No.6, June1982, p.564-568.

52. Nikolos, D., Gaitanis, N., Philekypron, G., "*Systematic t-error Correcting/All Unidirectional Error Detecting Codes*", IEEE Transactions on Computers, Vol. C35, No.5, May 1986, p.394-402.

53. Kundu, S., Reddy, S.M., "*On Symmetric Error Correcting Codes and All Unidirectional Error Detecting Codes*", IEEE Transactions on Computers, Vol. C-39, No.6, June 1990.

54. Malaiya, Y.K., Su, S.Y.H., "*Reliability Measure of Hardware Redundancy Fault-Tolerant Digital Systems with intermitent Faults*," IEEE Trans. Comput., vol. C-30, p.600-604, August 1981.

55. Koren, I., Su, S.Y.H., "*Reliability analysis of N-modular redundancy systems with intermitent and permanent faults*," IEEE Trans. Comput., vol. C-28, p.514-520, July 1979.

56. Su, S.Y.H., Koren, I., Malaiya, Y.K., "*A continous parameter Markov model and detection procedures for intermitent faults*," IEEE Trans. Comput., vol. C-27, p.567-570, June 1978.

57. Losq, J., "*A highly efficient redundancy scheme: Self purging redundancy*," IEEE Trans. Comput., vol. C-25, p.569-578, June 1976.

58. Su, S.Y.H., DuCasse, E., "*A harware redundancy reconfiguration scheme for tolerating multiple module failures*," IEEE Trans. Comput., vol. C-29, p.254-248, March 1980.

59. Smith, J.E., "*Measures of effectiveness of fault signature analysis*," IEEE Trans. Comput., vol. C-29, p.510-514, June 1980.

60. Williams, T.W., "*VLSI Testing*," Computer, p.126-136, Oct. 1984.

61. Barzilai, Z., Savir, J., Markovsky, G., Smith, M.G., "*VLSI*

self-testing based on syndrome techniques," IEEE Trans. Comput., vol. C-30,p.996-1000, Dec. 1981.

62. Avižienis, A., "*Fault Tolerant Systems*," IEEE Trans. Comput., vol. C-25, Dec. 1976.

63. Costes, A., Landrault, C., Laprie, J.C., "*Reliability and Availability Models for Maintained Systems Featuring Hardware Failures and Design Faults*," IEEE Trans. Comput., vol. C-27, p.548-560, June 1978.

64. Freeman, H., Metze, G., "*Fault Tolerant Computers Using 'Dotted Logic' Redundancy Techniques*," IEEE Trans. Comput., vol. C-21, August 1972.

65. Kime, C.R., "*Fault-Tolerant Computing. An Introduction and a Perspective*," IEEE Trans. Comput., col. C-24, May 1975.

66. Takaoko, T., Mine, H., "*N-Fail Safe Logical Systems*," IEEE Trans. Comput., vol. C-20, May 1971.

67. Avižienis, A., Gilley, G.G., Mathur, F.P., Rennels, D.A., Rohr, J.A., Rodin, D.K., "*The STAR Computer: An Investigation of the Theory and Practice of Fault Tolerant Computer Design*," IEEE Trans. Comput., vol. C-20, Jan. 1971.

68. Larsen, R.W., Reed, R.S., "*Redundancy by Coding versus Redundancy by Replication*," IEEE Trans. Comput., vol. C-21, Febr. 1972.

69. Armstrong, D.B., "*A deductive Method for Simulating Faults in Logic Circuits*," IEEE Trans. Comput., vol. C-20, May 1971.

70. Szigenda, S.A., Thompson, E.W., "*Modelling and Digital Simulation for Design Verification and Diagnostic*," IEEE Trans. Comput., vol. C-25, p.1242-1253, Dec. 1976.

71. Williams, T.W., "*Design for Testability. A survey*," IEEE Trans. Comput., vol. C-31, p.2-15, Jan. 1982.

72. Wakerly, J.F., "*Microcomputer Reliability Improvement Using Triple Modular Redundancy*," Proceedings of the IEEE, vol.64, June 1976.

73. Avižienis, A., Kelly, J., "*Fault Tolerance by Design Diversity: Concepts and Experiments*," Computer, vol.17, p.67-80, August 1984.

74. Johnson, D., "*The Intel 432: A VLSI Architecture for Fault-Tolerant Computer Systems*," Computer, vol.17, p.40-48, August 1984.

75. Avižienis, A., "*The N-Version Approach to Fault- Tolerant Systems*," IEEE Trans. Software Engineering, vol. SE-11, p.1491-1501, Dec. 1985.

76. Chen, C.L., "*Byte-Oriented Error-Correcting Codes for Semiconductor Memory Systems*," IEEE Trans. Comput., vol. C-35,

p.646-648, July 1986.

77. Chean, M., Fortes, J.A.B., "*A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays*," Computer, vol.23, p.55-69, Jan. 1990.

78. Goldberg, J., Levitt, K.N., Wensley, J.H., "*An organization for high survival memory*," IEEE Trans. Comput., vol. C-23, p.693-705, July 1974.

79. Kaneda, S., Fujiwara, E., "*Single byte error correcting-double byte error detecting codes for memory systems*," IEEE Trans. Comput., vol. C-31, p.596-602, July 1982.

80. Shao, H.M., Truong, T.K., Deutsch, L.J., Yuen, J.H., Reed, I.S., "*A VLSI design of a pipeline Reed-Solomon decoder*," IEEE Trans. Comput., vol. C-34, p.393-403, May 1985.

81. Sedmark, R.M., Liebergot, H.I., "Fault-tolerance of a general purpose computer implemented by a very large scale integration," IEEE Trans. Comput., vol. C-29, p.492-500, June 1980.

82. Schwab, T.E., Yan, S.S., "*An algebraic model of fault-masking logic circuits*," IEEE Trans. Comput., vol. C-32, p.809-825, Sept. 1983.

83. Peterson, C.B., ş.a., "*Two chips endow 32-bit processor with fault-tolerant architecture*," Electronics, 7 April 1983.

84. Rose, C., "*Refinements for hardware-based logic simulation*," Electronic Engineering, vol.61, p.91-99, Sept. 1989.

85. Pradhan, D.K., "*A new class of error-correcting/deleting codes for fault-tolerant computer applications*," IEEE Trans. Comput., vol. C-39, p.471-481, June 1980.

86. Snyder, F.G., "*Modular fault tolerance keeps computer systems reliable*," Electronic Design, 14 May, 1981, p.163-167.

87. Hunt, D., Tyson, T.J., "*Error detection and correction using SN54/74LS630 or SN54/74LS631*," Microprocessors and Microsystems, vol. 13, p.473-480, Sept. 1989.

88. Chu,Y., " *Bazele proiectării calculatoarelor numerice*", Ed. Tehnică, Bucureşti 1968.

89. Uşakova, G.N., " *Apparatnîi kontrol i nadejnosti speţializirovanîh EVM*",Sovetskoe Radio, Moskva, 1969.

90. Verner, V.D., Vorobev, N.N. ş.a., *"Microproţessorî. Sredstva sopriajenia"*, Kn. 2, Vîsş. sk., Minsk, 1987.

91. Yablonsky, S.V., *"Introduction to discrete mathematics,"* Mir Publishers, Moscow, 1989.

92. Seelers Jr., F.F., Hsiao, H.Y., Bearnson, L.W., *"Error detecting logic for digital computers,"* McGraw-Hill, 1968.

93. Iyergar, V.S., Kinney, L.L., *"Current fault detection in microprogrammed control units"*,IEEE Trans. Comput., vol.c-34, p.810-821, sept. 1985.

94. Nozuyama, Y., Nishimura, A., Iwamura, I., *"Design for testability of a 32-bit TRON microprocessor"*, Microprocessors and Microsystems vol.13, no.1, p.17-27, January/Febr. 1989.

95. Halbert, P.M., *"Selfchecking computer module based on the Viper microprocessor"*, Microprocessors and Microsystems, vol.12, no.5, p.264-270, June 1988.

96. Putinţev, N.D., *"Apparatnîi kontrol upravliaiuşcih ţifrovîh vîcislitelnih maşin"*, Izdatelstvo Sovetskoe Radio, Moşkva, 1966.

97. Vlăduţiu, M., Crişan, M., *"Program de determinare a restului prin comprimare paralelă a datelor"*, Al 7-lea Simpozion Naţional Tehnologie Electronică şi Fiabilitate, Herculane, 1987, p.324-327.

98. Krivulia, G.F., Taranov, V.B., *"Strukturî i pogreşnosti preobrazovania parallelnih signaturnîh analizatorov"*, Avtomatizirovannîe sistemî upravlenia i pribori avtomatiki, Vîşa şcola, Harcov, 1988.

99. Vlăduţiu, M., Crişan, M., *"Detecţia defecţiunilor la comprimarea bazată pe generarea biţilor de control la coduri ciclice"*, Al 6-lea Simpozion Naţional de Tehnologii şi Echipamente de Testare Automată, Cluj-Napoca, 1987, p.102-108.

100. Vlăduţiu, M., Crişan, M., *"Asupra comprimării seriale a informaţiei"*, Simpozionul Naţional de Calculatoare şi Conducerea Automată a Proceselor, Timişoara, 1988, p.163-167.

101. Crişan, M., *"Data flow organization for processor checking,"* Buletinul Ştiinţific şi Tehnic al Universităţii Tehnice din Timişoara, Tom 37(51), Electrotehnica, 1992.

102. Crişan, M., *"Signature control of arithmetic and logic operations,"* Buletinul Ştiinţific şi Tehnic al Universităţii Tehnice din Timişoara, Tom 37(51), Electrotehnica, 1992.

103. Meyer, J.F., *"On Evaluation the Performability of Degradable Computing Systems"*, IEEE Trans. Comput., vol.c-29, p.720-731, August 1980.

104. Moralee, D., *"Microprocessor architectures:ten years of development"*, Electronics & Power, nr.3, p.214-221, 1981.

105. Alexandridis, N., *"Microprocessor System Design Concepts"*, Computer Science Press, 1984.

106. Veronis, A., *"Microprocessors, Hardware and Applications"*, Prentice Hall Inc., Englewood Cliffs, N.J., 1984.

107. Triebel, W.A., Singh, A., *"16-bit Microprocessors. Architecture, Software and Interface Techniques"*, Prentice Hall Inc., Englewood Cliffs, N.J., 1985.

108. Eccles, W.J., "*Microprocessor Systems. A 16-bit Approach*", Addison Wesley Publishing Co., 1985.

109. Wakerly, J.F., "*Microcomputer Architecture and Programming*", John Wiley & Sons, N.Y., 1981.

110. * * * "*The TTL Data Book for Design Engineers*", Texas Instruments, 1977.

111. Cătuneanu, V.M. (coordonator), "*Materiale pentru electronică*", E.D.P., Bucureşti, 1982.

112. Cătuneanu, V.M., Mihalache, A., "*Bazele teoretice ale fiabilităţii*", Ed. Academiei, Bucureşti, 1983.

113. Mahmood, A., McCluskey, E.J., "*Concurent Error Detection Using Watchdog processor-A Survey*", IEEE Trans. Comput., vol.c-37, p.160-174, Febr. 1988.

114. Yarmolik, V.N., "*Fault diagnosis of Digital Circuits*", John Wiley & Sons, 1990.

115. Rao, T.R.N., Fujiwara, E., "*Error-Control Coding for Computer Systems*", Prentice Hall Inc., 1989.

116. Fujiwara, H., "*Logic Testing and Design for Testability*", MIT Press, England, 1990.