MINISTERUL EDUCAȚIEI ȘI ÎNVĂȚĂMÎNTULUI
Institutul Politehnic "Traian Vuia" Timișoara
Facultatea de electrotehnică

ing. DECEA STRATULAT

STRUCTURI DE CALCULATOARE REALIZATE CU MICROPROCESOARE CU POSIBILITATE DE AUTOTESTARE

TEZĂ DE DOCTORAT

CONDUCĂTOR ȘTIINȚIFIC:
Prof.dr.ing. ALECLA PĂTRESCU

1986

P R E F A Ţ Ă

Extinderea automatizării tuturor sectoarelor de activitate este determinată de dezvoltarea fără precedent a electronicii şi tehnicii de calcul, atît pe plan teoretic cît şi în domeniul tehnologic. În conformitate cu prevederile Congresului al XIII-lea al Partidului Comunist Român cu privire la dezvoltarea economico-socială a României în cincinalul 1985-1990 şi orientării de perspectivă pînă în anul 2000, activitatea în acest domeniu va creşte în viitorul cincinal cu 30-35 la sută, existînd premise ca industria electronicii să pătrundă masiv în întreaga activitate economico-socială, fiind orientată spre dezvoltarea cu precădere a producţiei de componente electronice, mijloace de automatizare, echipamente de electronică industrială şi tehnică de calcul.

Lucrarea de faţă se înscrie în preocupările generale de ridicare a productivităţii muncii, de creştere a fiabilităţii şi coeficientului de disponibilitate a sistemelor de calcul, punînd responsabilitatea detectării şi localizării defectelor chiar sistemului, prin metode de autotestare.

Autotestarea implică atît elaborarea unor concepte noi în proiectarea sistemelor de calcul cît şi elaborarea unor sisteme de operare adecvate.

În lucrare sînt abordate aspectele teoretice privind proiectarea unor sisteme de calcul autotestabile şi posibilitatea realizării practice a unor astfel de sisteme prin utilizarea unor structuri multiprocesor.

Pe perioada elaborării tezei, am beneficiat de îndrumarea atentă, exigentă şi competentă a regretatului prof.dr.ing. Alexandru Negoiţan, care cu înţelegere şi răbdare mi-a conturat domeniul, contribuind, în mod esenţial, la pregătirea mea profesională în calitate de doctorand, conducător de colectiv de număr şi conducător de doctorat. Puternica sa personalitate, contribuţiile de cea mai înaltă valoare pe care şi le-a adus la dezvoltarea tehnicii româneşti de calcul, fac din profesorul Al.Negoiţan o figură de neuitat pentru multe generaţii de ingineri specialişti în calculatoare electronice. Datorez ca

aceste rînduri să fie considerate drept omagiu adus memoriei Domniei-sale.

În perioada finalizării tezei am beneficiat de sprijinul deosebit de competent, de sfaturile şi observaţiile riguroase şi exigente ale conducătorului ştiinţific prof.dr.ing.Mircea Petrescu. Analiza atentă a materialului tezei, observaţiile şi recomandările formulate au fost elemente care au contribuit într-o manieră decisivă în etapa finală de elaborare, motiv pentru care îmi exprim întreaga stimă şi consideraţie faţă de persoana Domniei-sale, împreună cu cele mai sincere mulţumiri.

Mulţumesc în mod deosebit, şefului de catedră, conf.dr. ing.Grigore Stănăşel pentru sprijinul real de care am beneficiat în catedră la realizarea acestei lucrări, pentru discuţiile şi observaţiile care le-a formulat, pentru înţelegerea arătată în etapa de finalizare a tezei.

Aduc mulţumiri de asemenea tov.prof.dr.ing.Pop Vasile pentru discuţiile frumoase purtate, pentru încurajările susţinute pe care le-a adresat şi pentru sprijinul îndelungat şi constant pe care l-am primit din partea Domniei-sale.

Doresc să mulţumesc colegului şef lucr.dr.ing.Mircea Vlădeţiu pentru sprijinul constant, pentru discuţiile şi observaţiile care le-a formulat pe toată perioada elaborării tezei, pentru baza materială şi bibliografică pe care mi-a pus-o la dispoziţie.

Mulţumesc în mod deosebit colegului şi prietenului asist. ing.Ioan Hoş pentru observaţiile formulate, pentru sprijinul moral pe care mi l-a acordat pe întreg parcursul elaborării tezei, pentru înţelegere şi încurajările constante care m-au încurajat.

Doresc de asemenea să mulţumesc colegului şef de lucr.ing. Radu Stoinescu pentru sprijinul său, discuţiile şi observaţiile formulate pe perioada elaborării tezei.

Mulţumesc în final tuturor colegilor din catedra de automatică, celor care într-o manieră sau alta mi-au permis să-mi finalizez în bune condiţii lucrarea.

Cuprinsul tezei de doctorat

1. PROBLEMATICA TESTARII STRUCTURILOR DE CALCUL REALIZATE CU CIRCUITE LSI SI VLSI

1.1. Introducere

Dezvoltarea tot mai accentuată a tehnologiilor moderne a condus la introducerea pe scară largă a automatizărilor proceselor de producţie, robotizării avansate, cu implicarea tot mai frecventă a tehnicii de calcul, fără de care progresul tehnic actual devine greu de imaginat.

Utilizarea tehnicii de calcul în toate domeniile vieţii social-economice a fost posibilă datorită reducerii preţului de cost, creşterii performanţelor şi îmbunătăţirea fiabilităţii sistemelor numerice. Dar îmbunătăţirea performanţelor şi a fiabilităţii a condus la realizarea unor sisteme de calcul tot mai complexe, a căror eventuale defectări pot determina o serie de consecinţe dintre cele mai grave.

Este suficient să se amintească sistemele numerice de comandă şi control din misiuni de cercetare spaţială, din domeniul energiei nucleare, în sisteme de navigaţie şi comutare, reactoare chimice, elevatoare, sisteme de ridicat, în prelucrarea automată a informaţiei, pentru a înţelege necesitatea de asigurare împotriva unor evenimente nedorite.

Exemple de incidente grave datorate unor sisteme tehnice de mare complexitate sînt relativ numeroase [211] , ele fiind caracterizate de o mare disproporţie între cauză şi efect şi de un grad mare de imprevizibilitate.

Probleme deosebit de complexe apar în domeniul unor tehnici noi, în primul rînd domeniul electronicii şi al tehnicii de calcul, care utilizează materiale şi tehnologii în legătură cu care nu există suficientă experienţă, iar dinamismul acestor domenii exclude practic posibilitatea constituirii viitoare a unor metode de proiectare, general valabile, care să garanteze în mod absolut conservarea calităţii produselor pe un timp îndelungat.

Documentele Partidului Comunist Român acordă o atenţie deosebită ridicării continue a nivelului tehnic şi calitativ sporirii gradului de competitivitate al produselor românesti. În raportul prezentat la cel de-al XIII-lea Congres

al partidului, tovarășul Nicolae Ceaușescu, secretarul general al Partidului Comunist Român președintele Republicii Socialiste România, sublinia:"Pînă în 1990, circa 95 la sută din produsele românești trebuie să fie din punct de vedere tehnic și calitativ la nivelul celor existente pe plan mondial, iar cel puțin într-un procent de 2-5 la sută să realizăm produse cu asemenea parametri tehnici și calitativi care să situeze România pe primul loc în lume" [1].

In lumina documentelor de partid și stat, de un mare interes economic îl constituie realizarea unor produse cu un grad ridicat de complexitate, fiabile, cu un consum de materii prime și energie scăzut, și cu perspective de utilizare într-un domeniu cît mai larg de aplicații. In acest context sistemele de calcul realizate cu circuite integrate pe scară largă și foarte largă s-au impus; implicînd soluții tehnice noi de abordare a unor structuri de calcul, noi cercetări în domeniul arhitecturilor de sisteme de calcul, datorită unor avantaje față de sistemele de calcul convenționale. Se pot aminti cîteva din aceste avantaje:

- costurile de fabricație ale produsului sînt mai mici;

- timpul și costul dezvoltării unor soluții originale sînt mai mici, deoarece proiectarea și realizarea structurilor cu microprocesoare se la bază metode mai sistematice și mai organizate ;

- flexibilitatea structurilor cu microprocesoare micșorează timpul de răspuns al producătorului la noile cerințe ale beneficiarului, conducînd astfel la o creștere a duratei de viață activă a produsului ;

- capacitatea funcțională este mai mare la un volum și preț de cost scăzut datorită, în primul rînd, gradului de integrare ;

- fiabilitatea sistemului va fi mai bună prin utilizarea unor componente cu un grad de integrare pe scară largă și foarte largă ;

- puterea de calcul a microprocesoarelor poate fi utilizată și pentru diferite metode de testare și autotestare, ceea ce permite reducerea operațiilor de întreținere și reparații.

In lucrare se propune un model de structură de calcul cu posibilitatea de autotestare, structură ce poate fi im-

plementată relativ ușor cu noile componente de circuite integrate. Sistemul rezultat ca urmare a modelului teoretic propus este axat pe o arhitectură multiprocesor (multimicroprocesor) cu modalitatea de interconexiune de așa manieră încît să asigure autotestarea sistemului.

După ce se prezintă unele aspecte legate de problematica testării unor componente LSI și ISPI și se conturează dificultatea testării sistemelor numerice realizate cu astfel de componente, în capitolul 2 se prezintă modelul unui sistem numeric cu posibilități de autotestare. În această direcție sînt abordate sistemele autotestabile cu posibilitatea punerii în evidență a unor defecte multiple. Problematica este analizată din punctul de vedere al sistemelor autotestabile fără reperații. Modelul de diagnoză ce se analizează poate pune în evidență atît defecte permanente cît și defecte intermitente cu precizarea unor particularități de natură să permită localizarea modulului defect. Modelul ce se prezintă face o localizare a defectelor la nivel de unități funcționale, urmînd ca prin metode cunoscute 77 să se testeze, eventual, pînă la nivel de componentă.

În continuare, problemele teoretice ale modelului unui sistem autotestabil sînt simulate pe calculator și prin programele ALKSIM, ALMASIM și ALKSIT se validează valabilitatea aspectelor teoretice. În sensul că sindroamele, pe baza cărora se efectuează analiza autodiagnozei, sînt unic definite pentru fiecare situație de defect din sistem, ceea ce permite atribuirea cîte unui sindrom, fiecărui defect în parte.

În capitolul 3 se propun un număr de algoritme pentru localizarea unității (unităților) defecte, pe baza sindromului obținut în cursul rutinei de test. Algoritmele sînt verificate pe calculator.

În capitolul 4 se analizează unele metode de testare a circuitelor secvențiale și sînt prezentate cîteva metode de proiectare pentru îmbunătățirea testabilității sistemelor de calcul. Metodele de proiectare au fost analizate pe nivel de complexitate, și anume : la nivel de circuit integrat, la nivel de bloc și la nivel de sistem. Pe baza aspec-

telor analizate se poate trage concluzia că îmbunătăţirea diagnozei prin metode de proiectare duce la creşterea complexităţii sistemului cât şi a preţului de cost.

In capitolul 5 se prezintă o structură realizată cu două microprocesoare (8080 şi Z80) şi un analizor de semnături paralel, conectat conform principiilor enunţate în capitolul 2.

Cele trei unităţi funcţionale satisfac condiţiile de realizare a unei structuri de calcul cu posibilităţi de autotestare. Structura a fost astfel concepută încât să aibă o aplicaţie generală. Fiecare microsistem este prevăzut cu magistrale independente, ceea ce permite celor două microsisteme să lucreze împreună sau independent. Procedura de test este prevăzută a se executa fie concurent, când un microsistem nu are nevoie poate executa o rutină de test, fie prin întreruperea executării rutinii şi abordarea unor proceduri de autotest complect. In cursul detectării unui defect s-a prevăzut ca o parte din rutinile microsistemului defect să fie preluat de sistemul ce funcţionează corect.

Semnalizarea defectului se va face la panoul de comandă.

1.2. Problematica testării structurilor de calcul realizate cu circuite integrate LSI şi ISPS

1.2.1. Diagnoza în tehnica numerică

In cadrul diagnozei se pune problema detecţiei şi localizării defectelor, dar cum între cele două aspecte există o strânsă interdependenţă se obişnuieşte să fie reunite sub numele de diagnoză tehnică (fig.1.1). 77, 86

Diagnoza tehnică cuprinde totalitatea măsurilor care permit (pe cât posibil fără demontarea aparaturii) determinarea proprietăţilor şi posibilităţilor.

şi/sau a stării sistemelor tehnice; incluziv evaluarea acestora în funcţie de condiţiile de exploatare, cît şi localizarea componenteler defecte ale sistemului.

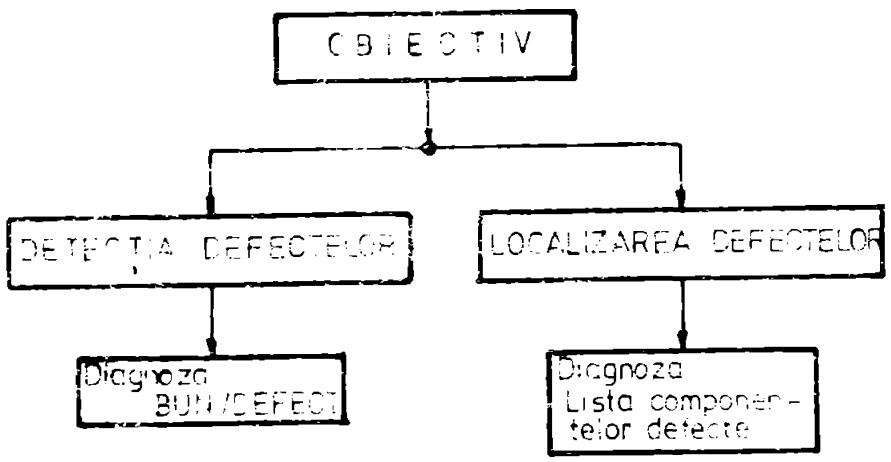


Fig.1.1

Sarcinile diagnozei în tehnica numerică sînt:

1. Perfecţionarea şi dezvoltarea de procedee şi algoritmi de diagnoză prin:
   - desvoltarea unor modele de defecte corespunzătoare;
   - realizarea unor algoritmi şi programe de test performante pentru automatele de test, în special sub aspectul diagnozei sistemelor cu LSI;
   - utilizarea intensivă a unor date statistice de analiză a defectelor ;
   - folosirea posibilităţiilor de autotestare;
   - optimizarea sistemelor pe baza teoriei informaţiei.
2. Perfecţionarea tehnico-constructivă:
   - proiectarea la nivel de circuit integrat cu facilităţi de testare ;
   - proiectarea la nivel de subansambluri cu facilităţi de testare ;
   - proiectarea la nivel de sistem cu facilităţi de testare ;
   - elaborarea unor documentaţii de întreţinere în paralel cu documentaţia de proiectare şi cu realizarea produsului.
   - formularea unor principii de realizare a unor

produse cu facilităţi de testare cît mai bună.

3. Obţinerea şi prelucrarea imaginilor defectelor:
   - obţinerea de dicţionare de semnături de defecte;
   - obţinerea de dicţionare de sindroame de defecte.

4. Proiectarea şi construirea de mijloace de testare.

5. Dezvoltarea unor procedee practice pentru determinarea algoritmică a seturilor de teste pentru sisteme complexe realizate cu circuite integrate.

6. Creşterea calificării personalului din activitatea de diagnosticare.

In principiu problematica testării se întîlneşte la două nivele: la nivelul punerii în fabricaţie şi la nivelul exploatării.

Soluţiile preconizate diferă,dependent de nivelul pentru care se asigură verificarea.

In ceea ce priveşte nivelul exploatării deziderатul urmărit constă în obţinerea pentru echipament a unei valori cît mai mari pentru coeficientul de disponibilitate [182,46] [55] :

$$K_D = \frac{T}{T + T_D} \qquad (1.1)$$

unde:

$$T_D = T_S + T_{AM} + T_{AS} + T_R \qquad (1.2)$$

cu: T - timpul mediu de funcţionare fără defecte;
T_D- timpul mediu de nefuncţionare a sistemului;
T_S- timpul de semnalizare (durata dintre apariţia defectului şi descoperirea prezenţei defectului);
T_{AM} - timpul de anunţare ;
T_{AS} - timpul de aşteptare (pînă la începerea remedierii)
T_R - timpul de remediere a defecţiunii.
T_R poate fi exprimat prin:

$$T_R = T_{LD} + T_{iD} + T_C \qquad (1.3)$$

unde:

T_{LD} - timpul de localizare a defectului;
T_{iD} - timpul de înlăturarea defectului ;
T_C - timpul de control a corectei funcţionări după depanare.

O valoare cît mai mare a coeficientului de disponibilitate rezultă prin mărirea lui T sau micşorarea lui T_D.

Micşorarea lui T_D, lăsînd deoparte timpii administrativi (T_{AM},T_{AD}) poate fi făcută prin termenii T_S şi T_R.

Referitor la componentele T_{LD} şi T_{iD}, ale timpului T_M, acestea dau informaţii mai ales asupra soluţiilor adoptate la nivel de diagnoză. Pentru reducerea acestor timpi, în literatura de specialitate sînt prevăzute în principiu două căi, care pot să nu se excludă una pe alta:

1. Elaborarea şi dezvoltarea unor algoritmi de diagnoză.
2. Proiectarea subansamblelor cu facilităţi de testare

In situaţia cînd sistemul lucrează într-un mediu sau o aplicaţie unde el practic trebuie să lucreze fără posibilitatea intervenţei umane, pentru executarea operaţiilor de întreţinere şi reparaţii, autotestarea sistemului este singura posibilitate de a controla exactitatea executării misiunii, [16] .

In aplicaţii în care T_S devine critic sistemul trebuie să fie dotat pe lîngă o redondenţă corespunzătoare aplicaţiei şi cu posibilitatea de autotestare, pentru a cunoaşte în fiecare moment starea de bună funcţionare a modulelor conectate în sistem, [17,183] .

In cazul structurilor de calcul realizate cu microprocesoare introducerea posibilităţilor de autotestare nu implică o creştere exagerată a costului întregului sistem, tocmai datorită facilităţilor pe care le posedă microprocesoarele, care împreună cu o memorie fixă îşi pot autogenera stimuli de test (programe de test) rămînînd ca un bloc suplimentar să evalueze corectitudinea funcţionării microprocesorului, [15] .

1.2.2. Strategia folosită la testarea structurilor de calcul realizate cu circuite integrate LSI şi VLSI.

Componentele integrate pe scară largă au din punct de vedere funcţional o complexitate ce corespunde unor grupe întregi de elemente primare. Ele nu pot testa cu metode aplicate la grupe cu elemente integrate pe scară mică sau mijlocie deoarece de multe ori structura internă nu este

cunoscută utilizatorului şi pe de altă parte nu există acces la nivel de componentă, [7,9] . Extinderea algoritmilor şi metodelor de detecţie a defectelor folosite pentru circuite SSI şi MSI ar duce la creşterea exagerată a timpului de testare; astfel exemplu pentru o memorie ROM trebuie verificat $n.2^n$ stări, unde n este numărul de biţi a cuvîntului de ieşire şi n este numărul de intrări (pentru 2708 rezultă 8.2^{16} stări). Pentru un circuit secvenţial sînt necesare a se verifica $n.2^{n+g}$, unde n şi n au aceeaşi semnificaţie ca mai sus şi g reprezintă numărul de stări interne (pentru 8080 rezultă $10x2^{16+117}=3x10^{39}$ stări , [90,84] .

In cazul utilizării componentelor LSI trebuie făcută o distincţie clară între diagnoza elaborată de producător şi cea elaborată de utilizator.

In procesul de fabricaţie trebuie efectuate teste asupra parametrilor şi funcţiilor,care pun în evidenţă un defect în procesul tehnologic cît mai din timp. In cazul acesta se pot elabora teste şi la nivel de porţi prin utilizarea adaptoarelor de tip ac care se poziţionează pe suprafaţa de contact a cristalelor. Pentru această fază s-ar putea, principial, aplica aceleaşi metode de test ca şi în cazul schemelor realizate cu circuite integrate SSI şi MSI.

In cazul testelor efectuate la nivelul utilizatorului sînt de semnalat următoarele aspecte:

- Circuitele integrate LSI şi VLSI îşi pierd din transparenţă, ceea ce determină ca procedeele de testare structurală să nu mai fie adecvate. In plus, avîndu-se în vedere că structura internă nu este cunoscută întotdeauna utilizatorului, testarea prin activarea căilor şi găsirea de defecte de punere pe zero, respectiv de punere pe unu nu mai este adecvată. In acest caz s-ar putea trece de la metoda activării unei căi la cea a activării modulelor.

- Nu există procedee algoritmice pentru stabilirea vectorilor de stimuli de test.

- Testarea completă a tuturor funcţiilor cu toate combinaţiile de date posibile nu mai este adecvată, datorită volumului mare de informaţii ce trebuie utilizat şi a timpului de testare prohibit. Din acest motiv stabilirea testului are un pronunţat caracter euristic.

- Datorită caracterului complex şi funcţionării date, nemodificabilă nici pentru scopuri de diagnoză, a circuitelor interne,nu sînt aplicabile metodele de testare cvasistatice, folosite în cazul circuitelor SSI şi MSI. În acest caz fiind necesare metode de test dinamice.

Problemele prezentate duc la concluzia că o testare structurală este neaplicabilă fiind indicată o testare funcţională. Astfel metodele de testare se vor orienta tot mai mult pe software.

Din punct de vedere al testabilităţii sînt în acest sens convenabile structurile orientate pe magistrale. Prin aceasta se realizează un acces simplu la module, ceea ce permite introducerea şi extragerea de informaţii de diagnoză, pentru a genera teste şi a evalua răspunsurile [84,90]

In fig.1.2 se prezintă un sistem orientat pe magistrală. Modulele 1,2...n reprezintă unităţi constructive oarecare (atît cu LSI cît şi altele). Legătura la magistrala externă trebuie să prezinte o anumită standarizare pentru fiecare structură de microsistem, în timp ce magistrala internă este specifică modulului.

Conectarea şi deconectarea unui modul la magistrala externă se face prin activarea informaţiilor de comandă şi/sau de adrese.

Luînd în consideraree toate aspectele prezentate s-ar desprinde următoarele cerinţe pentru testarea unei structuri de calcul realizată cu microprocesoare:

- Dezvoltarea şi utilizarea de procedee care să permită o comprimare a informaţiilor necesare detecţiei şi diagnozei fără a pierde informaţii utile acestora,prin analiza semnalelor de recunoaştere [197] .

- Elaborarea de metode, care permit diagnoza fără cunoaşterea structurilor intime ale capsulelor LSI respectiv VLSI atît la stabilirea procedurilor de diagnoză cît şi la aplicarea lor [207].

- Folosirea şi dezvoltarea unor metode de testare dinamică.

- Utilizarea unor metode de analiză privind posibilităţile de diagnoză şi pe baza acestora stabilirea unor structuri optime pentru testare [172,107,192,194] .

- Aplicarea intensă a structurilor autotestabile [174, 175,179,45,47].

- Folosirea programării modulare |126, 55| .

- Elaborarea automatelor de test universale programabile, pentru testarea grupelor de elemente şi a elementelor constructive, [98,23,24,89] .

- Proiectarea şi realizarea de aparatură de test uşor de manevrat, pentru utilizarea pe teren [37,69,70,73] .

1.3. Metode de diagnoză a structurilor de calcul realizate cu microprocesoare.

Pe baza situaţiilor de test descrise, mulţi autori se ocupă de proceduri de diagnoză pentru circuite LSI şi structuri de calcul realizate cu elemente LSI [90,84,127,26,159, 108, 88,192,69,39,41,212] . Dintre toate procedurile de test s-au dovedit utile exclusiv metodele funcţionale, care pot fi clasificate:

1. Metode de comparare ;
2. Metode de generare algoritmică a stimulilor;
3. Metode de test prin memorarea răspunsurilor ;
4. Metode de autodiagnoză.

1.3.1. Metode de comparare.

Numite şi procedee de referinţă prin care modulul de testat se compară cu un modul de referinţă ce funcţionează în paralel. Atît în modulul de testat cît şi în cel etalon se introduc aceleaşi informaţii, iar semnalele de ieşire se compară. Pentru ca diagnoza să fie edificatoare este necesară o sincronizare perfectă a celor două module, sau cel puţin compararea rezultatelor să se facă sincron. Generarea unui nou şir de stimuli de test se va face doar după o comparare pozitivă.

Metoda prezintă o serie de avantaje:

- se poate aplica oricărui tip de circuit integrat.
- este eficientă în producţie.
- se poate realiza şi o testare în timp real.
- implementarea unui automat de testare nu este relativ dificilă, deoarece datele de ieşire nu trebuie să fie memorate într-o memorie pentru comparare.

Referitor la vectorii de test, aceştia trebuie generaţi de aşa manieră încît toate defectele corespunzătoare modelului de defecte să fie recunoscute şi să genereze date critice.

In funcţie de modul de sincronizare se poate realiza o comparare pe semnalul de tact sau se pot folosi procedee de comprimare a datelor.

In fig.1.5 se prezintă principial structura unui automat de testat bazat pe metoda comparării.

Acest procedeu necesită un număr de puncte de test adecvat pentru a realiza o precizie a diagnozei corespunzătoare. Vectorul de diagnoză generat trebuie să conţină stimuli suficienţi pentru obţinerea unor informaţii dorite despre unitatea testată. Pe de altă parte dacă numărul de puncte de test este prea mare, compararea şi evaluarea testului va influenţa asupra vitezei de generare a stimulilor de test.

Metoda prezintă următoarele limitări:

- depinde de unitatea etalon, din acest motiv stimulii de test sînt puţin flexibili. O modificare în structura unui bloc (modul) duce la modificarea stimulilor de test.

- este utilă numai în teste funcţionale.

- trebuie asigurat un sincronism perfect între unitatea etalon şi cea de testat.

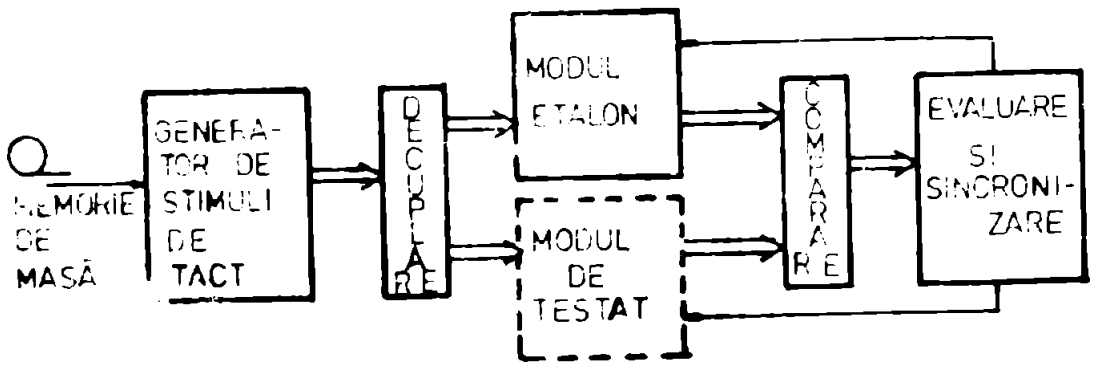


Fig.1.5

- dacă numărul de puncte de test este prea mare blocul de evaluare va limita viteza de lucru.

Metoda comparării este eficientă acolo unde se testează un număr mare de unităţi de acelaşi tip (structură).Este

indicată la testarea microprocesoarelor a unor unități de comandă, a memoriilor, a unor circuite de intrare/ieșire.

1.3.2. Metode de generare algoritmică a stimulilor de test

Metoda de generare algoritmică a stimulilor de test este indicată atît la testarea microprocesoarelor cît și a memoriilor. In principiu [68] metoda se bazează pe considerentul că instrucțiile unui microprocesor sînt unic definite, în sensul că rezultatul corect se obține după executarea unei anumite instrucții în conjuncție cu un operand. Totuși nu este totdeauna posibil să se vadă la bornele de ieșire rezultatul execuțării unei singure instrucții. Pentru aceasta trebuie executate o serie de instrucții înainte de a se interpreta rezultatul de la ieșire.

Pe de altă parte se utilizator îl interesează testarea microprocesorului pentru aplicațiile proprii. Din acest motiv diagnoza este orientată cu precădere spre o testare funcțională ceea ce poate duce la reducerea timpului de testare. Metoda din acest punct de vedere trebuie să asigure o flexibilitate de programare sporită.

Metoda de generare algoritmică a stimulilor de test numită și metoda recunoașterii modelelor de biți, comportă înregistrarea pe o memorie a unor rezultate de test corecte, modulul de testat este pus să execute un anumit program de test al căror rezultate sînt comparate cu cele corecte.

Principalul element al metodei îl reprezintă generatorul de stimuli. Acesta este realizat, în principiu, dintr-o memorie locală tampon de mare viteză în care se stochează instrucțiile și *operanzii*.

Metoda de generare a stimulilor de test constă în introducerea în registul tampon (T), a generatorului de stimuli, a primei instrucții de testat (fig.1.4).

Generatorul de stimuli, pe baza analizei instrucției va depune în registul A codul instrucției, dacă aceasta generează un rezultat la ieșire, sau va forma o adresă, în cazul că sînt necesare secvențe de instrucții pentru obținerea unui rezultat. Adresa din registul de adresă (A) va indica codul instrucției următoare necesare în secvența de

instrucţii, care este trimisă de memoria tampon în registul T. Procedura se repetă pînă la rularea întregului algoritm de testat.

Operanzii diferitelor instrucţii sînt generaţi din exterior de către utilizator fiind incorporaţi în programul de test, împreună cu rezultatele corecte ce trebuie să se obţină. Din acest motiv utilizatorul trebuie să cunoască la perfecţie atît funcţionarea microprocesorului cît şi aplicaţia pentru care îi este destinat. Metoda exclude o testare dinamică.

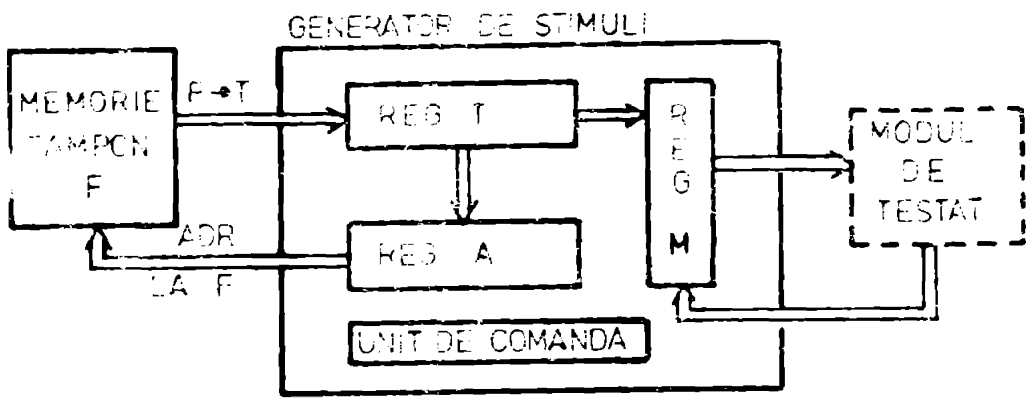


Fig.1.4

Metoda necesită un număr mare de programe de test, pentru fiecare aplicaţie în parte. Diagnoza pe această cale este făcută în bună parte de producător garantînd calitatea produsului şi din punct de vedere funcţional.

1.3.3. Metode de test prin memorarea răspunsurilor.

Aceste tehnici de diagnoză implică două metode de test şi două modalităţi de generarea stimulilor. Fiecare metodă memorează şi execută programe de diagnoză scrise foarte diferit şi anume: prin emulare şi prin simulare.

Cu această tehnică se obţine un program de emulare sau simulare rezident într-o memorie de masă. Programul de diagnoză se aplică microprocesorului de testat şi comparat răspunsul cu cel corect.

1.3.3.1. Procedeu prin emulare.

Implică utilizarea unui dispozitiv de referinţă la stabilirea condiţiilor de test, dar care nu mai este necesar la testarea propriu-zisă, ceea ce determină ca viteza de

testare să nu mai fie limitată de circuitul de referinţă. Testul este conceput pentru a diagnosa funcţionarea în mediul natural al microprocesorului (sau a circuitului testat)

Strategia procedeului de emulare constă în :

1. Crearea unui cadru natural de funcţionare.

2. Încărcarea programului de diagnoză în sistemul de test, translatat în cod maşină şi apoi aplicat circuitului etalon.

3. Executarea programului de diagnosticare de către dispozitivul de referinţă şi memorarea răspunsurilor la fiecare ciclu de bază, prin realizarea unui tabel de adevăr.

4. Executarea programului de diagnosticare de către dispozitivul de testat şi comparat răspunsurile cu valorile din tabelele de adevăr obţinute în paşii anteriori.

Condiţiile naturale de funcţionare a microprocesorului sînt asigurate de un testor automat care are rolul de a prelua din microsistem partea de memorie şi periferice (fig. 1.5). În acest sens în testor 64.K cuvinte sînt rezervate, ca reprezentînd imaginea memoriei microsistemului şi memorează instrucţiile utilizatorului. Fiecare cuvînt al testo-

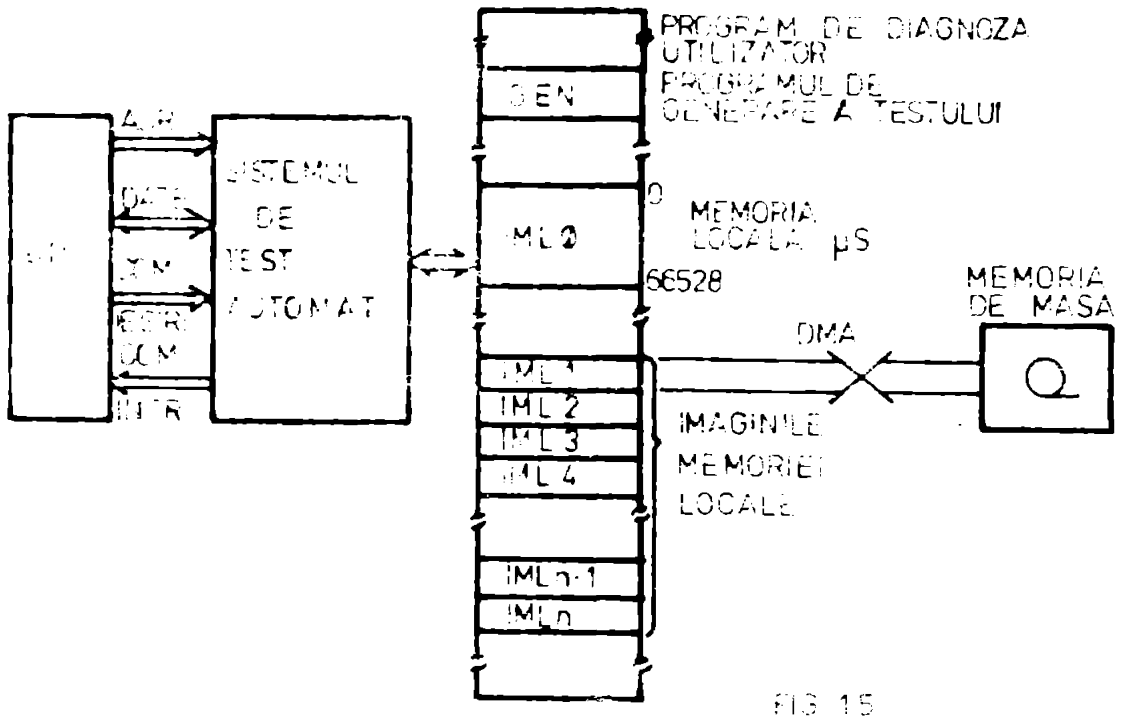


Fig.1.5

rului este format din mai mulţi biţi, fiind utilizaţi pentru controlul forţărilor la masă, simularea întreruperilor, eliberarea datelor de la periferice sau chiar pentru simularea modului de lucru DMA.

Testorul este prevăzut cu un program de generare a tes-

tului, care pe baza programului de diagnoză, specific unei aplicații, crează tabele de adevăr IMLI pentru fiecare stare și fiecare instrucție executată de dispozitivul etalon. Procesul de învățare și înregistrare a stimulilor și a răspunsurilor microprocesorului este determinat de programul de diagnoză sub controlul programului de generare a testului.

Programul de test va fi rulat în continuare pe microprocesorul de verificat și răspunsurile vor fi comparate cu cele corecte.

Prin procedeul de emulare se poate experimenta, cu stimulii de test, funcționarea în condițiile cele mai defavorabile, pentru o aplicație specifică la care este utilizat microprocesorul. Se poate de asemenea studia sensibilitatea lui la diferite seturi de instrucții sau coduri. Mai mult sînt permise atît teste funcționale cît și parametrice. Programele de diagnoză sînt relativ ușor de modificat, ceea ce permite un grad ridicat de flexibilitate prin existența sistemului de test automat.

Pe de altă parte costul echipamentului de test este ridicat. Pentru fiecare tip de microprocesor trebuie un program de generare a testului diferit.

1.3.3.2. Procedeul prin simulare.

Simularea implică utilizarea unui sistem de calcul puternic și cunoașterea în detaliu a structurii circuitului simulat. Simularea se poate face atît la nivel de poartă cît și la nivel de blocuri (funcțională). În primul caz există mai multe informații utile, dar timpul de simulare este foarte mare, în cazul unor scheme complexe. De asemenea trebuie cunoscut detaliile constructive. Simularea funcțională permite o vedere mai de ansamblu, este mai simplă, dar aplicarea ei nu este directă, datorită lipsei de universalitate a metodelor folosite pentru descrierea unui modul oarecare. Simularea modulului este obținută prin evaluarea unei funcții, ceea ce produce în practică avantajul unei descrieri mai concise. În schimb, precizia obținută este în general limitată, deoarece nu se ia în considerare corect propagarea anumitor defecte. În general, pentru simularea unui sistem, sau scheme complexe se combină simularea la

nivel de poartă cu simularea funcțională. Fiecare parte din sistem se simulează în funcție de necesitățile diagnozei.Și în cadrul procedeului de simulare se determină prin program stările de ieșire ca rezultat al datelor de intrare, obținându-se tabele de adevăr care vor fi comparate cu stările de ieșire a modulului de testat. Procedeul prin simulare permite simularea tuturor circuitelor unui sistem a,anumitor părți din sistem și chiar simularea întregului sistem.

1.3.4. Metode de autodiagnoze

Este metoda cea mai la îndemîna utilizatorului, deoarece nu necesită hard suplimentar, asistență de la fabricant și echipamente de laborator.

Diagnoza folosește o strategie de auto-test, astfel încît, dacă toate instrucțiile sînt executate corect,programul ajunge la o adresă de sfîrșit și indică condiția de trecere. Dacă apar erori, diagnoza se termină la o locație de memorie ce indică eroarea.

Microprocesorul trebuie să lucreze într-o configurație minimă, care să cuprindă memorii ROM, RAM, magistralele de date, adrese și comenzi și eventual unele periferice de introducere a informațiilor. Programul de diagnoză se încarcă într-o memorie RAM, sau există deja într-o memorie ROM. La testarea microprocesorului se rulează acest program. El trebuie să utilizeze cît mai multe instrucții posibile, într-o secvență defavorabilă și în cazul cel mai bun, cu date defavorabile registrelor interne.

Folosirea tehnicii de auto-testare în sisteme cu un singur microprocesor prezintă unele dezavantaje.

- erorile multiple și complementare pot fi mascate și deci nedetectate.

- adevărata cauză a defectului poate rămîne nediagnosticată.

- programele de diagnosticare lungi trebuie rulate pînă la sfîrșit, chiar și în cazul că defectul este repede pus în evidență,rezultînd un timp de test lung,fără a fi necesar.

- unele condiții externe, cum sînt întreruperile, nu pot fi testate în condiții defavorabile.

Principalele avantaje ale auto-testării sînt:

- sistemul lucrează în cadrul natural.

- avertizarea operatorului asupra unei posibile probleme de operare.

- asigurarea operatorului că sistemul funcţionează chiar dacă în aparenţă are probleme de utilizare a lui în special datorită complexităţii.

- autotestarea indică în mod cert cînd sistemul funcţionează corect sau nu.

Metoda devine foarte utilă prin adăugarea unui hard suplimentar în partea de evaluare.

1.4. Modalităţi de diagnosticare a defecţiunilor din microsisteme prin tehnici de comprimare a datelor

In ultimul timp s-au dezvoltat metode de diagnosticare a defecţiunilor din microsisteme prin tehnici de comprimare a şirurilor de date cum ar fi: numărarea de tranziţii, numărarea de unu-ri, analiza de semnături. Prin aceste metode se urmăreşte simplificarea modalităţii de depanare.

Cel mai potrivit mod de testare ar fi cu analiza de semnături [69,70] , ce asigură o probabilitate de eroare foarte mică şi punerea în evidenţă a defectelor dinamice. Se pot analiza o multitudine de condiţii intermediare şi de ieşire, iar microsistemul este testat la viteza de lucru normală. Astfel fiecărui mod i se atageză un simbol, care descrie buna funcţionare a circuitului. Depanatorul compară simbolul afişat de analizorul de semnături cu cel corect, trăgînd concluzii asupra funcţionării circuitului testat.

1.4.1. Tehnica de testare cu analiza de semnături.

In principiu un analizor de semnături (AS) conţine un registru de deplasare de 16 biţi cu linii de reacţii obţinute de la ieşirea anumitor rangiri ale registrului (fig.1.6). Tehnica de comprimare a datelor [69,197,152] constă în faptul că informaţiile binare, serie, culese de la modul de testat prin intrarea DI, sînt convertite într-o semnătură binară şi comparată cu semnătura corectă din modul respectiv. Deplasarea în registrul de deplasare se face sincron cu un semnal de tact, într-un inter-

val de timp,determinat de semnalele START şi STOP, numită fereastră [70,37] .

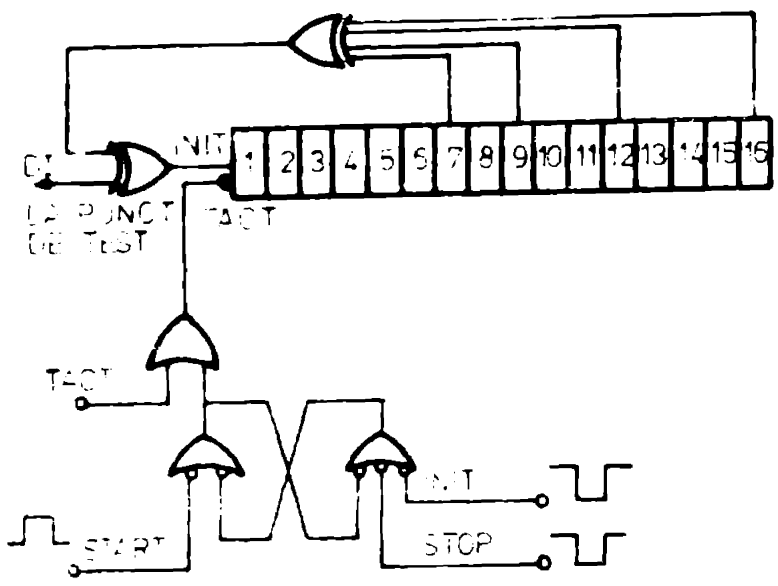


Fig.1.6

Una din avantajele AS faţă de alte tehnici de testare constă în faptul că există posibilitatea detectării erorilor legate timp. Pentru a avea această posibilitate, datele trebuiesc introduse în analizor în mod sincron cu tactul circuitului.In acest scop se pot alege şi una din fazele tactului sau un semnal de comandă pentru care informaţiile de intrare sînt stabile. Alegerea acestui punct de preluare a tactului este ugurată prin posibilitatea analizorului de a selecta fronturile tactului. Pentru circuite asincrone alegerea tactului pentru analizor se va prelua de la tactul circuitului care generează secvenţa la intrările de test.

In mod asemănător şi semnalele de START şi STOP, care determină intervalul de timp temporal în care analizorul citeşte datele de la intrare, au posibilitatea alegerii fronturilor active.

Informaţiile de intrare se culeg în diferitele noduri ale sistemului ce conţin semnături caracteristice.

In cazul determinării unei semnături false se urmăresc semnăturile nodurilor pînă la sursa de generare a erorilor.

Caracteristicile AS îi determină flexibilitatea, permiţîndu-i utilizarea la numeroase dispozitive numerice, de la circuite combinaţionale şi secvenţiale, la memorii şi microprocesoare.

AS se bazează pe tehnica generării unor vectori de stimuli la intrarea circuitului. Modul de generare a stimulilor de test depinde de circuitul de testat. Pentru circuite combinaţionale şi memorii ROM se poate utiliza un numărător ce poate genera semnale pentru testarea exhaustivă a acestor

dispozitive (fig.1.7.a şi b). In cazul memoriilor ROM semnalele de START şi STOP pot fi bitul cel mai semnificativ al numărătorului. Datele se culeg fie de la ieşire (ROM) sau de la ieşire şi/sau puncte intermediare pentru circuite combinaţionale.

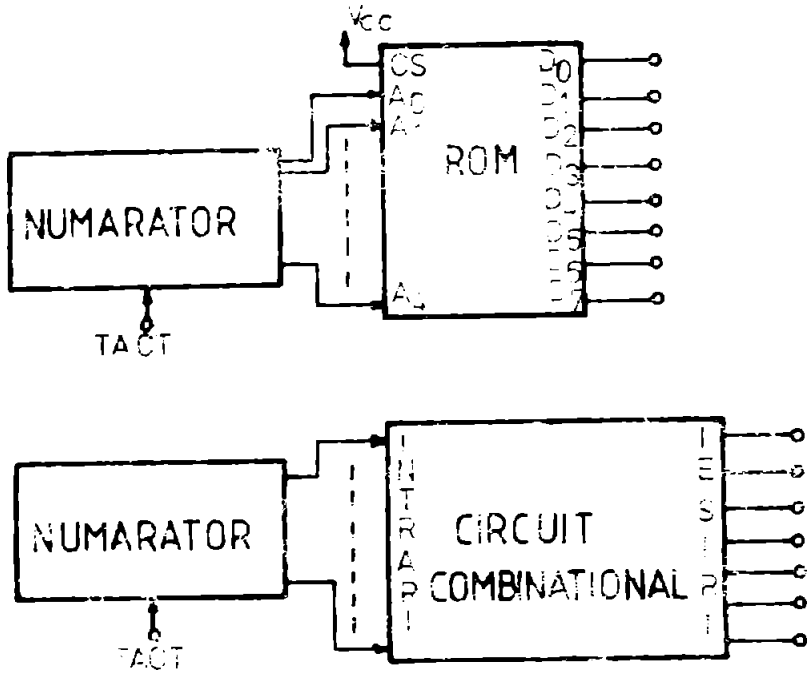


Fig.1.7

In cazul că dispozitivul lucrează într-o structură autotestabilă atunci se poate genera convenabil stimuli de test. Pentru generarea stimulilor de test către structurile realizate cu microprocesor, se poate asigura baclarea adresei pe întreg spaţiu de adresare, asigurînd astfel şi stimuli corespunzători pentru celelalte circuite din sistem.

In cazul unor circuite cu reacţie (secvenţial, microsisteme) se indică întreruperea buclei de reacţie.

Dezavantajul AS constă în faptul că nu poate furniza indicaţii asupra defectului în sensul că caracterele particulare ale unei semnături sînt neutilizabile pentru determinarea calitativă a defectului. Semnăturile oferă doar indicaţii asupra prezenţei sau absenţei defectului în modul respectiv.

1.4.2. Testarea microsistemelor cu analiza de semnături.

Pentru sistemele bazate pe microprocesoare, pentru fiecare defecţiune, localizarea componentei defecte în structura magistralelor microprocesorului este particulară.

O defecţiune în sistem poate determina, datorită reacţiei formată de magistrala între microprocesor şi memorie, propagarea acestei defecţiuni în tot sistemul, rezultînd

intrări eronate şi în consecinţă ieşiri eronate.

Dificultăţile determinate de reacţie, în depanarea, pot fi soluţionate prin suprimarea însăşi a reacţiei. Cea mai sigură metodă de întrerupere a magistralei de date este cea mecanică. Utilizarea comutatoarelor, călăreţilor, implantarea într-un soclu de circuite integrate, a tampoanelor de magistrale sau a amplificatoarelor de magistrale, sînt toate tehnici bune de izolare. O alternativă ar consta în utilizarea unor circuite cu trei-stări (TS). Programul de test asigură întreruperea trecerii prin magistrala de date cu ajutorul întrerupătorului S_1, conectat cu o instrucţie de mascare a întreruperilor.

După deconectarea legăturilor de reacţie, urmează aplicarea stimulilor de test. Cel mai simplu mod de obţinere constă în forţarea executării de către microprocesor a unor cicluri identice (aceeaşi instrucţie), prin care se realizează numai incrementarea numărătorului de adrese. În acest fel magistrala de adrese a microprocesorului parcurge întregul spaţiu de adrese, asigurînd aplicarea unui semnal de tip tabel de adevăr pe magistrală.

Alegerea instrucţiei care va fi executată de microprocesor nu este critică. În principiu, orice instrucţie care asigură incrementarea numărătorului de adrese şi trecerea microprocesorului în ciclu de aducere a instrucţiei poate fi utilizată.

Pentru crearea unor facilităţi de testare, se pot adăuga cîteva componente care să asigure testarea execuţiei unei anumite instrucţii. După cum se observă în fig.1.9 se conectează rezistenţe de sarcină spre Vcc pe linii de date. Apoi, se conectează diode spre masă printr-un comutator ce asigură "rularea buclată" a microprocesorului. Aceste conexiuni se execută spre partea magistralei de date conectată spre microprocesor. Prin deconectarea magistralei de date, microprocesorul va bucla executarea aceleaşi instrucţii. În fig.1.9 se arată modul, cum instrucţia MOVA,(01111111) forţează microprocesorul 8080 de a intra în modul de funcţionare buclat. Acest mod de funcţionare este în general valabil şi pentru alte tipuri de microprocesoare [197] .

Pentru semnalele de START şi STOP se va alege linia de

adresă, iar pentru ÎACT semnalul DĂBLĂ sau oricare semnal

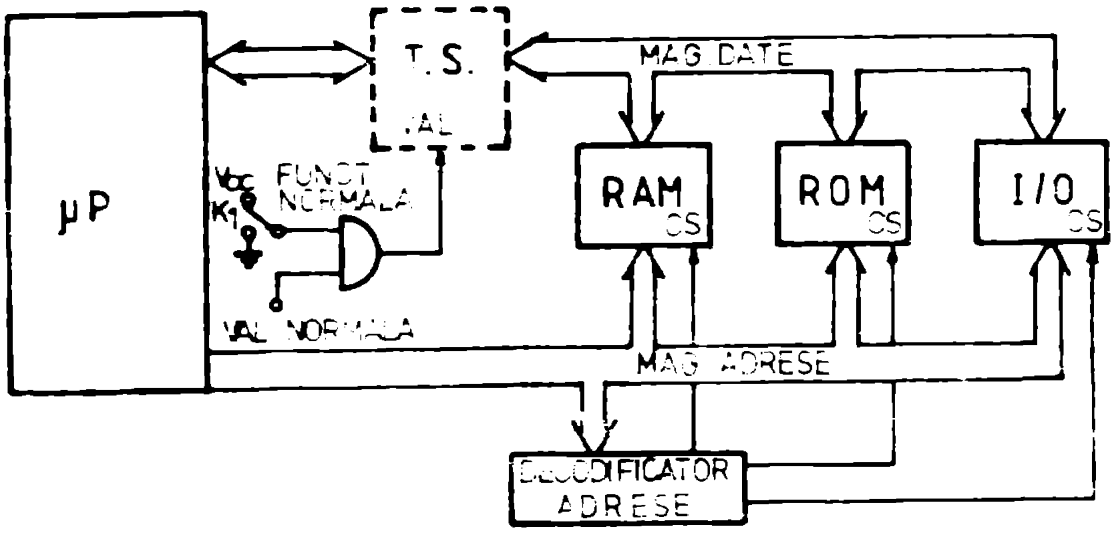


Fig.1.8

de comandă în timpul căruia liniile de adresă sînt stabile.

Fig.1.9

După verificarea magistralei de adresă şi a decodificatorului de adresă se vor conecta pe rînd memoriile ROM prin mentinerea în continuare a magistralei de date întrerupte între memoriile ROM şi microprocesor. Apoi se iau semnăturile de pe magistrala de date, verificîndu-se toate cuvintele din primul ROM.

Dacă semnăturile sînt corecte se adaugă al doilea ROM (validîndu-se CS) şi se vor culege din nou semnăturile de pe magistrala de date, dar numai datele culese de la ROM1 şi ROM2.

Se procedează la fel pentru toate memoriile ROM.

Prin metoda prezentată pînă la acest moment se cunoaşte că microprocesorul poate să funcţioneze în ciclul său de adresă corect şi din defecţiunile statistice ale microprocesorului probabil că şi celelalte funcţii ale sale sînt bune. Magistrala de date nu este afectată de defecţiuni, memoriile ROM sînt bune, iar circuitul de decodificare a adreselor funcţionează corect.

Pentru verificarea memoriilor RAM există multe algoritme de test [210] , ce au rolul de a testa:

- logica de adresare şi decodificare ;
- posibilitatea de înscriere şi citire a fiecărei celule de memorie ;
- interdependenţa fiecărei celule de memorie cu cele vecine (atît logic cît şi tehnologic).

Datorită timpului de lucru al programului de test cît şi condiţiile climatice ale mediului se mai includ probleme legate de domeniu de folosire, reîmprospătare, timpi de acces, etc. Algoritmii de testare a memoriilor semiconductoare sînt enumeraţi în [213] .

Scopul testării unei memorii RAM din cadrul unui microsistem constă în controlul magistralei de date a sistemului de la microprocesor spre memoria RAM, care trebuie să fie excitată într-o manieră controlată pentru a determina defecţiunile.

Metode mai folosite sînt:

1) Memorarea conţinutului memoriei ROM, în memoria RAM, aceasta asigură o testare oarecum aleatoare a posibilităţilor de memorare a celulelor sale.

2) Memorarea unor şiruri formate din zerouri şi unităţi consecutive (tablă de şah), apoi a complementului acestuia ; aceasta verifică posibilitatea memorării unui zero sau unu în fiecare celulă de memorie.

3) Pornind de la o memorie completată numai cu zerouri se deplasează o unitate prin locaţiile de memorie. În continuare se completează memoria numai cu unuri şi se deplasează cîte un zero prin locaţiile de memorie; astfel se verifică dacă există conectări pe un nivel logic şi, în plus,absenţa scurtcircuitelor între celulele de memorie.

4) Poziţionarea pe o diagonală a matricei de memorie numai cu unităţi. În continuare se deplasează tot conţinutul memoriei cu o poziţie pe linii în mod ciclic. Prin citirea de durată a acestor informaţii se testează amplificatoarele de citire asupra efectelor de inerţie.

5) Mutarea cîte unei unităţi (sau zero) într-o locaţie vecină şi înapoi, după care se controlează celula modificată, dacă conţinutul s-a menţinut. Prin această metodă se verifică partea de adresare şi semnalele de comandă.

6) Memorarea fiecărei adrese a memoriei RAM la adresele respective;se verifică posibilitatea adresării corecte a

fiecărei locații de memorie.

Verificarea circuitelor de intrare/ieșire (I/E) urmărește pe de-o parte controlul magistralei de date de la porturi spre microprocesor și pe de altă parte magistrala de date de la microprocesor spre porturile de I/E. Aceste circuite se testează în regimul de rulare buclată a microprocesorului. Astfel semnăturile pot fi prelevate la aceste porturi și de la perifericele asociate lor.

Fereastra START-STOP se deschide, de obicei, cu semnalul de adresă A15. Ca tact se folosește semnalul IORQ.

Porturile de ieșire nu pot fi excitate prin rularea buclată. În acest caz o metodă va fi conectarea ieșirilor unor porturi la intrările porturilor de testat.

Folosirea analizei de semnături la testarea microsistemelor prezintă o serie de avantaje amintite, dar cum s-a constatat necesită un mod de proiectare adecvat, o diagnoză de tip funcționează sau nu funcționează dispozitivul (util la beneficiar), dînd indicații reduse asupra caracterelor particulare ale unei erori. Fiind inutilizabile pentru determinarea calitativă a defectului. Necesită asistarea permanentă a personalului de deservire service. Prin introducerea într-o structură de calcul autotestabilă, considerăm că această metodă poate fi dezvoltată și utilizată cu un randament superior.

1.5. Metoda de diagnoză secvențială.

În cazul în care microsistemul este implementat pe o plachetă, care cuprinde și alte circuite LSI (memorii RAM, ROM, circuite I/E, etc.), ce se caracterizează prin existența unor magistrale comune, la care au acces majoritatea circuitelor LSI din sistem. Fiecare din aceste dispozitive ridică ele însăși probleme dificile la testare fig.1.10.

Prin magistrală se fac toate comunicările cu lumea externă în mod bidirecțional. Datorită acestui fapt, plus starea de impedanță ridicată, aparatura de test nu poate distinge informațiile de intrare de cele de ieșire precum și care din dispozitivele conectate la magistrală este defect, fără a recurge la informații suplimentare.

Organizarea pe magistrală duce la comportarea între-

gului sistem identic cu un circuit secvențial cu bucle de reacție uriașe, unde informația de pe magistrala de date, la un moment dat, poate modifica adresa la momentul-următor. Cînd pe magistrală apare un defect, el se propagă rapid și devine nediagnosticabilă.

Această situație poate fi prevenită prin punerea în evidență a primei defecțiuni. Identificarea acestui defect cere utilizarea unei memorii de referință a fiecărui cuvînt care este așteptat pe magistrală. Prin organizarea pe magistrale comune diagnoza poate fi simplificată, dacă se cunoaște starea microprocesorului la momentul detectării primului defect [90,84,58].

Pentru rezolvarea aspectelor prezentate, se utilizează algoritmul de diagnoză secvențial, cum este cel prezentat în fig.1.11. Dacă starea microprocesorului, la detectarea primului defect, nu a fost un ciclu de citire, atunci fie microprocesorul fie circuitul de acces direct la memorie poate să introducă erori. Cunoscînd starea actuală a sistemului, respectiv dispozitivul care deține controlul magistralei se va cunoaște modulul defect. Dacă starea microprocesorului, la prima defecțiune a fost un ciclu de citire, atunci un canal al magistralei pornind de la un beneficiar este defect. Acest dispozitiv specific poate fi identificat utilizînd informațiile de adresă și datele de stare. Fiind primul circuit care furnizează date eronate pe magistrală pentru variabile de intrare corecte. În cazul în care conectarea unui dispozitiv la magistrala se face printr-o logică combinațională intermediară, prin algoritmii corespunzători se poate detecta defectul la un modul specific sau dispozitiv. Problema se reduce la conectarea dispozitivului direct la magistrală.

Stările inițiale ale fiecărui dispozitiv conectat la magistrala trebuie cunoscute.

Algoritmul secvențial se poate aplica presupunînd că magistrala de date este fără defect și poate fi implementat pe orice testor capabil să recunoască toate ciclurile sistemului și să se oprească pe primul defect detectat prin comparare.

Această metodă necesită accesul secvențial a testorului la o memorie de masă pentru memoria de referință sau o memorie rapidă de capacitate mare. Se poate utiliza, în unele

cazuri şi un microsistem de referinţă pentru a micşora capacitatea memoriei testorului.

La nivel de sistem nu este indicată generarea stimulilor de test ca în cazul modelelor cunoscute la componente, deoarece modelarea atît a funcţiilor cît şi mecanismul de defectare al circuitelor LSI este foarte complexă.

In concluzie trebuie menţionat că există multe metode de diagnoză a structurilor de calcul realizate cu componente LSI dar nu şi o sistematizare bine formulată şi o teorie unitară. Din acest motiv problematica testării în acest domeniu este direcţională în mare măsură spre rezolvarea unor cazuri, mai degrabă particulare, decît generale.

Pentru testarea sistemelor de calcul sînt necesare elaborarea unor metode noi specifice structurilor realizate cu circuite integrate pe scară largă, şi utilizarea unor testoare cu facilităţi de test corespunzătoare. Această direcţie conduce la o creştere a preţului produsului final testat. Mai mult prin metodele de test prezentate se rezolvă doar aspectul că după testare, sistemul numeric este în stare de funcţionare, neputîndu-se preciza problemele legate de siguranţa în funcţionare a acestuia şi mai precis cînd sistemul se defectează. La rezolvarea acestui aspect proiectarea unor sisteme de calcul cu posibilităţi de autotestare, poate să-şi aducă o contribuţie hotărîtoare în aplicaţii care impun cunoaşterea permanentă a stării de funcţionare a sistemului.

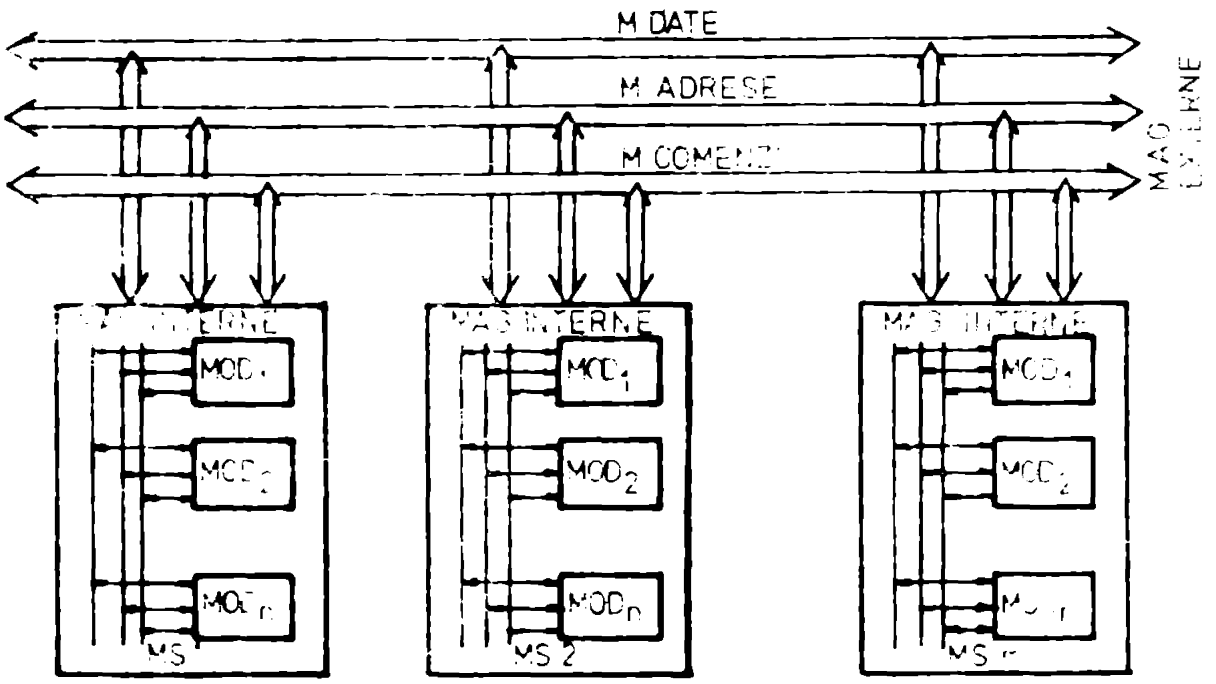
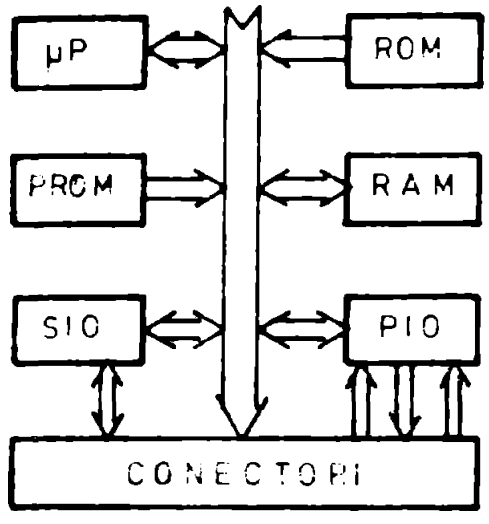
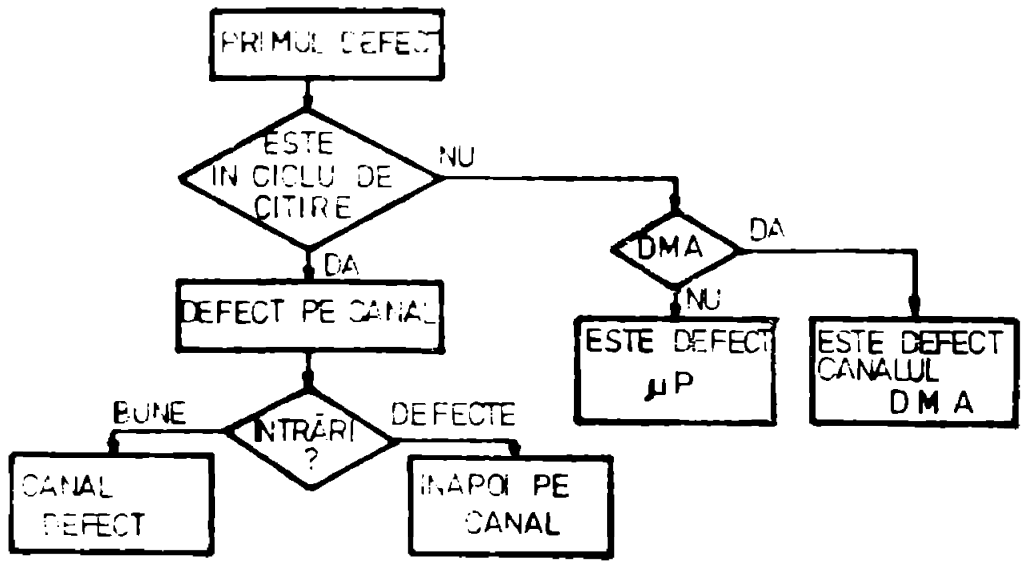


FIG 12


µP
ROM
PROM
RAM
SIO
PIO
CONECTORI


PRIMUL DEFECT
ESTE IN CICLU DE CITIRE
NU
DA
DMA
DA
NU
DEFECT PE CANAL
ESTE DEFECT µP
ESTE DEFECT CANALUL DMA
BUNE
INTRĂRI ?
DEFECTE
CANAL DEFECT
INAPOI PE CANAL

CAPITOLUL 2

MODELUL DE DIAGNOZĂ A SISTEMELOR NUMERICE CU POSIBILITATI DE AUTOTESTARE

2.1. Conceptul autotestării sistemelor numerice.

Folosirea tot mai frecventă a circuitelor integrate pe scară largă şi foarte largă în construcţia sistemelor de calcul a determinat o schimbare radicală atît în domeniul tehnologiei, a arhitecturii sistemelor numerice, cît şi în strategiile adoptate privind tehnicile de diagnoză a erorilor. Privite din acest punct de vedere, la proiectarea unui sistem numeric, se impune a se ţine cont de următoarele trei caracteristici de bază : îmbunătăţirea performanţelor, reducerea preţului de cost şi creşterea fiabilităţii sistemului [162]. O proiectare ideală a unui sistem numeric ar conduce la îmbunătăţirea unui factor să se facă cu menţinerea constantă a celorlalţi doi factori.

In cazul proiectării unui sistem numeric cu circuite integrate pe scară mică la medie, creşterea numărului de componente pentru îmbunătăţirea performanţelor sau a creşterii fiabilităţii, duce inevitabil la creşterea semnificativă a costului sistemului. Prin introducerea circuitelor LSI şi LSI problema diferă puţin, în sensul că rata de creştere a preţului de cost per funcţie adăugată este nesemnificativă. Acest lucru se justifică în sensul că la un moment bine determinat nu se utilizează totalitatea funcţiilor pe care le poate produce un circuit LSI sau LSI. Deci se realizează o utilizare mai eficientă a funcţiilor unui circuit integrat şi prin adăugarea unui număr de componente suplimentare, preţul de cost al sistemului, ar creşte în mod nesemnificativ, faţă de valoarea iniţială, la presupunerea că s-a îmbunătăţit performanţa, sau a crescut fiabilitatea sistemului.

Rezultă că în cazul proiectării sistemelor numerice cu circuite integrate LSI şi VLSI se pot prevedea circuite suplimentare, pe care să le acopere sistemul, fără a creşte în mod exagerat preţul de cost. In această direcţie pot fi prevăzute şi metode adecvate de îmbunătăţire a procesului de diagnoză a întregului sistem. In [130] s-a pus bazele teoretice şi experimentale a unei noi metode de proiectare a unui sistem, prin care sistemul este partiţionat în blocuri funcţionale de aşa manieră încît detectarea defectelor, din sistem, să poată fi făcută prin el însuşi. Noua concepţie, care plasează responsabilitatea detectării erorilor chiar sistemului este cunoscută prin denumirea de autodiagnoză, care se rezolvă prin metode de autotestare.

Domeniile în care un sistem numeric trebuie să lucreze cu siguranţă sporită sînt: sisteme de comutare (telefonie), sisteme de navigaţie (aeriene, spaţială, maritime, feroviare etc.), centrale atomice, reactoare chimice, sisteme de ridicat [214].

Defectele şi erorile în sistemele numerice au fost împărţite în mai multe tipuri [214]. Considerînd cauzele posibile de defectare a componentelor sau programelor, se pot distinge trei categorii diferite de defecte (fig.2.1):

1. Defecte de echipament datorită mecanismului de defectare fizică sau chimică.

2. Proiectarea eronată a echipamentelor şi a programelor datorită înţelegerii greşite, interpretarea greşită, deciziilor eronate, etc. cauzată de om.

3. Defectarea programelor datorită defectelor sporadice în partea de echipament, numită şi defect de "străvire a programului".

Pentru a permite o funcţionare sigură a sistemelor numerice s-au impus două strategii:

1. eliminarea totală a mecanismului de defectare. In această situaţie se tinde spre sisteme perfecte fără defecte, numite "sisteme ideale".

2. proiectarea unui sistem imperfect, din punct de vedere al defectelor, la care se adoptă unele mecanisme de tolerare a defectelor şi aplicarea unor metode de întreţinere specifice pentru prevenirea defectelor, numite "sisteme tolerante la defecte".

Deoarece sistemele numerice sînt relativ complexe, strategia cea mai aplicată este a doua prin proiectarea unor sisteme numerice redondante, sau prin îmbunătăţirea metodelor de diagnosă, inclusiv a autodiagnozei.

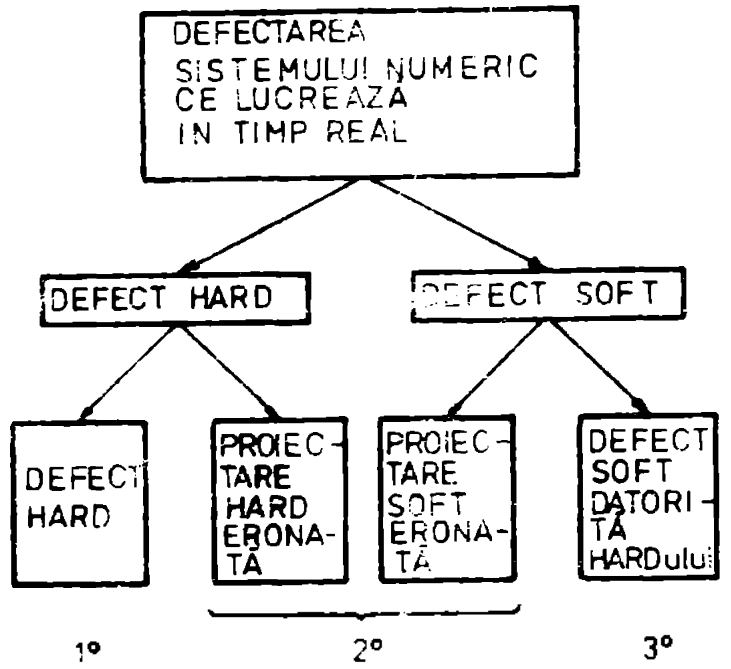


Fig.2.1

Ceea ce este comun celei de a doua strategii constă în faptul că pentru de intrare şi ieşire deobicei nu este redondantă, impunînd ca aceste blocuri să fie proiectate după prima metodă. Trebuie să existe în acest sens tehnici adecvate de verificare şi corectare a informaţiilor, metode sistematizate de testare statistică şi de analiză a stimulilor,care să garanteze că datele sînt recepţionate corect de partea numerică, respectiv rezultatele ajung corect la destinaţie.

Prin autotestarea unui sistem numeric se va înţelege procesul prin care componentele, modulele, blocurile sistemului sînt în stare să-şi genereze setul de stimuli de test şi să evalueze corect răspunsurile unităţilor existente.

Se observă că definiţia este foarte largă şi se impun anumite precizări.

In primul rînd se pune automat întrebarea cum poate să-şi genereze sistemul singur testele şi să-şi evalueze răs-

pemantile dacă amumite componente ale sale sînt defecte. Aceste consideraţii ne vor clarifica în capitolul de faţă cînd se vor da condiţiile necesare şi suficiente ca un sistem să se autotesteze.

In al doilea rînd se pune întrebarea dacă prin autotestare se va detecta doar prima componentă găsită defectă sau autotestarea conduce la determinarea tuturor elementelor defecte. Pentru precizări suplimentare se va considera următoarea denumire :

Dacă într-un sistem, în urma aplicării procedurii de test, sînt identificate toate componentele defecte, printr-o singură trecere a testului, atunci autotestarea este fără reparaţii.

Dacă într-un sistem, în urma aplicării procedurii de test, sînt identificate toate componentele defecte, prin înlănţuirea succesivă a elementelor găsite defecte, atunci autotestarea este cu reparaţii, sau secvenţială.

In acest caz, prima componentă găsită defectă se înlocuieşte, se reia testul, găsindu-se o a doua componentă defectă. Procedeul continuă pînă la determinarea tuturor elementelor defecte.

Metoda autotestării sistemelor numerice devine din ce în ce mai răspîndită datorită unor avantaje evidente:

1. Introducerea ei poate reduce semnificativ (dacă absenţa defectelor a sistemului nu este avansată) necesitatea unor aparaturi de testat, care de obicei nu sînt prezente la beneficiar, reducînd astfel costul de întreţinere.

2. Timpul de obţinere a unei diagnoze corecte se reduce, ceea ce va determina creşterea coeficientului de disponibilitate.

3. Sistemul este testat la frecvenţa de lucru normală.

Trecerea sistemului în regim de autotest poate fi făcută dinamic în cadrul programului, atunci cînd sistemul, sau părţi ale sistemului, în anumite momente, nu este utilizat, evitîndu-se astfel un control permanent al funcţionării lui corecte [64].

Cele cîteva avantaje amintite sînt net superioare faţă de creşterea spaţiului ocupat în memorie (ROM) pentru programele de test şi introducerea unor componente suplimentare care să facă posibilă autotestarea. Reducerea la minim a părţii de hardare suplimentar, necesară unei structuri autotestabile, impune o

nouă concepţie de testare. Se va renunţa la testarea exhaustivă a sistemului în favoarea testării funcţionale.

2.2. Diagnoza sistemelor autotestabile (SAT) în prezenţa defectelor permanente

Problema autodiagnozei unui sistem a fost abordată din punct de vedere teoretic, pentru prima dată în lucrarea [130] In continuare au apărut o serie de alte lucrări [22,86,100, 114,116,140,142] care au dezvoltat ideile lui Preparata[130] privind autodiagnoza unui sistem.

Pe baza modelelor prezentate în literatură, se va dezvolta în continuare un model de diagnoză generalizat, care permite diagnoza unui sistem cu posibilităţi de autotestare în cazul apariţiilor unor defecte multiple. Modelul teoretic se va generaliza atît în cazul prezenţei unor defecte permanente cît şi în cazul prezenţei unor defecte intermitente. Se va constata în final că modelul permite detecţia şi localizarea tuturor defectelor, indiferent de natura lor, permanentă sau intermitentă, dacă secvenţele de test sînt rulate în mod corespunzător. Prin modelul generalizat, ce se va prezenta, se poate obţine o diagnoză corectă sau în cazul cel mai defavorabil a diagnoză incompletă, dar niciodată o diagnoză incorectă.

2.2.1. Modelul teoretic de diagnoză a SAT în prezenţa DP.

Analiza teoretică a diagnozei unui sistem numeric cu posibilităţi de autotestare în cazul prezenţei unui defect permanent se va face pe baza teoriei grafurilor [34,94,149].

In acest sens se consideră că un sistem S se descompune în n unităţi de sine stătătoare, notate cu setul (fig.2.2)

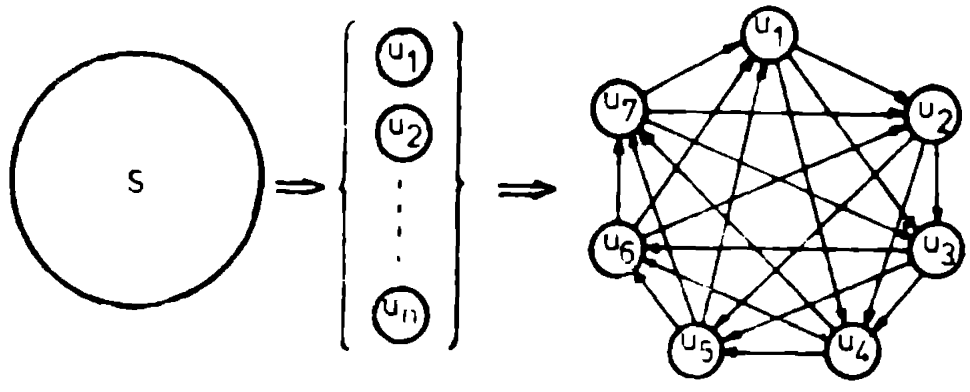


Fig.2.2

$$U = \{u_1, u_2, u_3 \dots u_n\} \qquad (2.1)$$

care să satisfacă următoarea ipoteză:

Ipoteza 2.1.

1. Fiecare unitate $u_i \in U$ să poată genera stimuli de test pentru alte unități din sistem, $u_j \in U$, pentru i=1,2...n şi j = 1,2...n.

2. Fiecare unitate $u_i \in U$ să poată evalua corect răspunsul unității $u_j \in U$ la stimulii de test aplicați.

Se presupune că o unitate a sistemului astfel descompus, nu mai poate fi divizată mai departe în alte subunități, mai simple, care să îndeplinească condițiile enunțate în ipoteza 2.1.

In continuare, sistemului S i se atașează o matrice de conexiuni de test, T. Matricea $T=(t_{ij})$ este definită în felul următor:

Definiția 2.4

$$t_{ij} = \begin{cases} 1 \text{ dacă unitatea } u_i \text{ testează unitatea } u_j \\ 0 \text{ în caz contrar.} \end{cases}$$

pentru: i = 1,2...n
j = 1,2...n (2.2)

Pe baza setului de unități U şi a matricei T sistemul se poate reprezenta sub forma unui graf G=(U,T), unde nodurile grafului sînt unitățile $u_i \in U$, iar arcele de la nodul u_i la nodul u_j există dacă şi numai dacă elementul t_{ij} a matricei T are valoarea "1".

Aplicarea setului de test, prin intermediul legăturilor de test, $t_{ij} \in T$, este însoțită de o evaluare a unității u_j de către unitatea u_i, prin faptul că unitatea u_i specifică dacă u_j este defectă sau nu. Ca măsură de exprimare a acestei evaluări se va asocia un răspuns, notat cu a_{ij}, fiecărui element $t_{ij} \in T$, unde $a_{ij} \in \{0,1\}$. Referitor la valoarea lui a_{ij} se presupune următoarea ipoteză:

Ipoteza 2.2.

1) Dacă $t_{ij} \in T$, şi u_i nu este defectă, atunci $a_{ij}=0$ implică că u_j nu este defectă.

2) Dacă $t_{ij} \in T$, şi u_i nu este defectă, atunci $s_{ij}=1$, implică că u_j este defectă.

3) Dacă $t_{ij} \in T$ şi u_i este defectă, atunci $s_{ij}=x$, indiferent de comportarea unităţii u_j ; unde $x \in \{0,1\}$.

Definiţia 2.2. Un vector de răspunsuri de test, s_{ij}, este numit sindrom [13,22,114,116,216,121,217,18].

Deoarece răspunsul $s_{ij} \in \{0,1\}$, rezultă că un sindrom este constituit dintr-un şir de cifre binare. Numărul de cifre binare dintr-un sindrom corespunde cu numărul de conexiuni de test $t_{ij} \in T$, pentru care $s_{ij} = 1$. In mod evident se constată că un sindrom ce are un număr mare de cifre binare va conţine o cantitate de informaţii mai mare, referitoare la diagnoză. Pe de altă parte un sindrom de lungime prea mare va determina un timp de obţinere mai mare, un număr mai mare de teste şi în fine, o procedură de diagnoză mai complicată.

Definiţia 2.3. Un sistem S, este diagnosticabil dacă şi numai dacă fiecare sindrom, ce corespunde unei situaţii de defect din sistem, diferă de toate celelalte sindroame pe care le poate produce sistemul.

Problema ce se pune, constă în faptul de a vedea dacă pe baza unui sindrom se poate realiza o diagnoză corectă, a unui sistem.

Intr-o primă fază se presupune că defectele, care pot să apară în sistem, sînt permanente.

Se vor defini în continuare următoarele seturi de unităţi.

Definiţia 2.4: Prin aplicaţia multivocă exprimată de relaţia :

$$\Gamma(u_i) = \{u_j \text{ / pentru } t_{ij} \in T\} \qquad (2.3)$$

se înţelege mulţimea de unităţi u_j testate de unitatea u_i, pentru cazul că există legătură de test de la u_i la u_j ($t_{ij} \in T$).

Definiţia 2.5: Prin aplicaţia multivocă inversă exprimată de relaţia :

$$\Gamma^{-1}(u_i) = \{u_j \text{ / pentru } t_{ij} \in T\} \qquad (2.4)$$

se înțelege mulțimea de unități u_j ce testează unitatea u_i, pentru cazul că există legătură de test de la u_j la u_i ($t_{ji} \in T$).

Definiția 2.6: Prin aplicație multivocă exprimată de relația :

$$\Gamma(U_1) = \bigcup_{u_i \in U_1} \Gamma(u_i) - U_1 \qquad (2.5)$$

se înțelege mulțimea reunită al tuturor unităților u_j ce sînt testate de unitățile $u_i \in U_1$, și care nu fac parte din mulțimea U_1.

Definiția 2.7.: Prin aplicație multivocă inversă dată de relația :

$$\Gamma^{-1}(U_1) = \bigcup_{u_i \in U_1} \Gamma^{-1}(u_i) - U_1 \qquad (2.6)$$

se înțelege mulțimea reunită al tuturor unităților u_j ce testează unitățile $u_i \in U_1$ și care nu fac parte din mulțimea U_1.

Avînd în vedere cele prezentate sistemul poate fi reprezentat sub forma unui graf : $G = (N, A)$, unde nodurile N ale grafului reprezintă unitățile sistemului, iar arcele A sînt reprezentate prin legăturile de test între unități.

Pentru stabilirea condițiilor necesare și suficiente ca un sistem, descompus în n unități, să fie autotestabil se pleacă de la precizarea că nu toate unitățile sînt defecte simultan, deoarece în acest caz avem de-a face cu un defect catastrofal. Problema care se pune este de a determina numărul maxim de unități ce pot fi defecte la un moment dat și care pot fi detectate.

Fie t_p numărul de unități maxim defecte, ce pot fi depistate simultan în urma procesului de diagnoză și n numărul de unități din sistem.

În acest caz se poate enunța următoarea teoremă :

Teorema 2.1: Un sistem S cu n unități este diagnosticabil fără reparații, pentru un număr maxim de t_p defecte, dacă și numai dacă sînt îndeplinite simultan următoarele trei condiții:

c1) $n \geq 2t_p + 1$

c2) $\Gamma^{-1}(u_i) \geq t_p$

c3) $\bigcup_{j=i+1}^{i+t_p} u_j \subseteq \Gamma^{-1}(u_i)$

pentru: $i = 1, 2 \ldots n$.

Prin cele trei condiţii ale teoremei 2.1 se specifică că numărul de unităţi funcţional corecte trebuie să fie cu cel puţin o unitate mai mare decît unităţile defecte (c1). In condiţia c2) se arată că fiecare unitate este testată de cel puţin t_p alte unităţi, iar prin condiţia c3) se precizează că două unităţi nu se testează reciproc.

Demonstrarea teoremei se face presupunînd, pe rînd, că una din condiţii nu se îndeplineşte, dar sistemul este diagnosticabil pentru t_p defecte.

Diagnoza sistemului se va face pe baza interpretării răspunsurilor unităţii testate de către unitatea ce testează, Setul răspunsurilor formează un sindrom.

Demonstrarea primei condiţii se face presupunînd că ea nu este îndeplinită, dar sistemul rămîne diagnosticabil.

Fie un sistem S diagnosticabil pentru care avem :

$$n \geq 2t_p+1 \qquad (2.7)$$

Pentru $n=2t_p$ se poate imagina că sistemul se poate descompune în două submulţimi de unităţi, fiecare formată din t_p unităţi :

Fie mulţimile U_1 şi U_2 cu $|U_1| \geq t_p$ şi $|U_2| \geq t_p$ şi $U_1 \cap U_2$ adică:

$$U_1 \cup U_2 = \emptyset \qquad (2.8)$$

unde mulţimea U_1 conţine unităţile defecte şi U_2 conţine unităţile funcţional corecte (fig.2.3 a). Rezultatele testelor vor conduce la obţinerea sindroamelor indicate în fig.2.3.a. Toate unităţile bune ce vor testa alte unităţi bune vor da pe $s_{ij}=0$, respectiv $s_{ij}=1$ pentru cazul cînd vor testa unităţi defecte. La rîndul lor, toate unităţile defecte vor genera un sindrom oarecare $s_{ij}=x$, conform ipotezei 2.2. Dacă se consideră că U_1 este mulţimea unităţilor corecte şi U_2 este mulţimea unităţilor defecte,rezultatele testului sînt specificate în fig.2.3.b.

Se constată că cele două cazuri de diagnoză nu pot fi distinse, neputînd preciza care este setul de unităţi corecte, respectiv defecte, deoarece valoarea lui x din fig.2.3 poate lua oricare valoare binară; x 0,1 . Deci pot exista situaţii cînd rezultatul testului pentru cele două cazuri să

fie identice.

Diagnoza nu este posibilă, decît cu implicarea unei condiţii necesară : $n \geq 2t_p+1$.

Se pune problema dacă condiţia este şi suficientă.

Se presupune că se introduce în sistem cel puţin o unitate corectă în plus faţă de numărul de unităţi defecte. Această unitate u_i realizează cel puţin un test asupra unei unităţi din U_1 sau U_2, atunci, pentru cele două cazuri prezentate mai sus, există cel puţin un element al celor două sindroame care să difere. În cazul prezentat în fig. 2.4.a, toate elementele $s_{ij}=1$, iar în fig.2.4.b, toate $s_{ij}=0$, pentru $u_j \in U_1$.

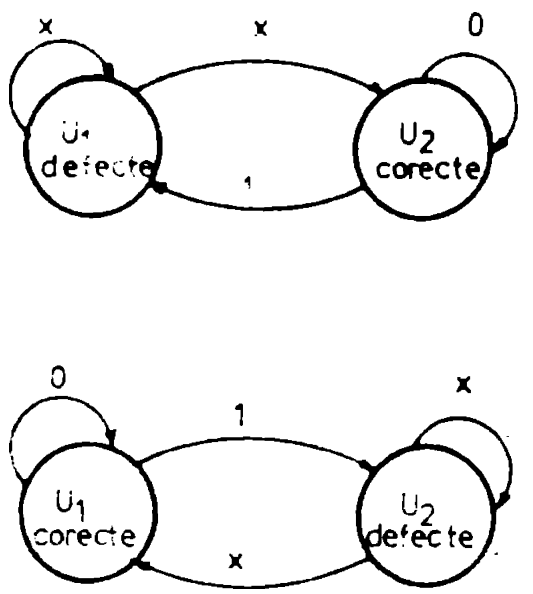


Fig.2.3

Rezultă că este suficient ca cel puţin t_p+1 unităţi să fie funcţional corecte ca să se poată găsi cele t_p unităţi defecte, ceea ce demonstrează prima condiţie a teoremei.

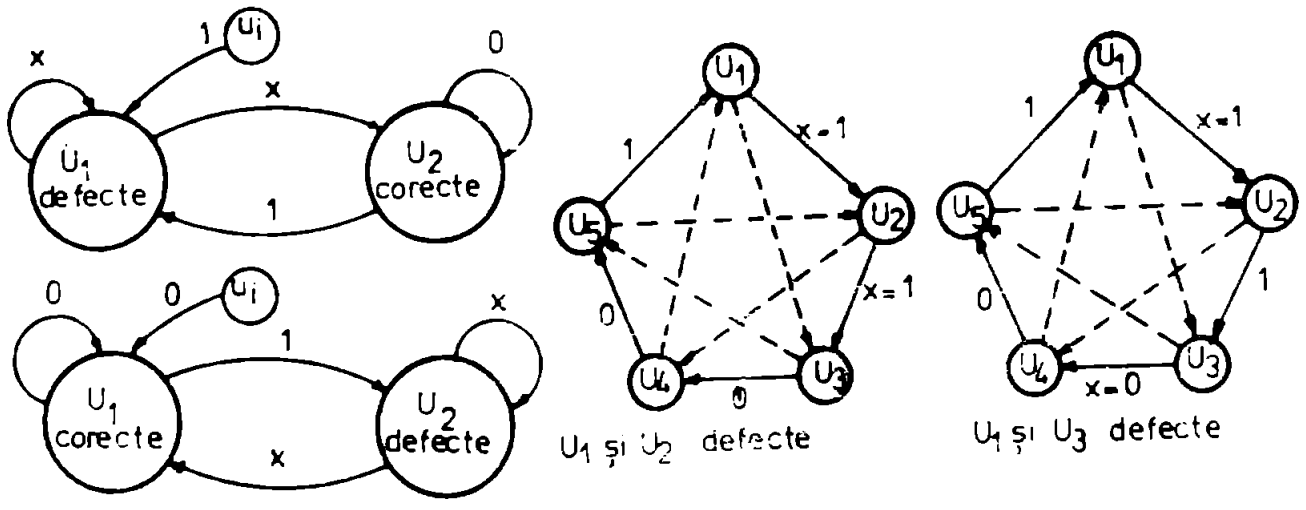
Referitor la demonstrarea condiţiei c2) se consideră că sistemul este diagnosticabil şi că unităţile $u_1, u_2 \ldots u_k$ sînt toate unităţi care testează o anumită unitate u_i.

Pentru cazul că $k > t_p$ şi dacă unităţile $u_1, u_2 \ldots u_k$ sînt toate defecte, rezultatele testelor, celor k unităţi, pot lua valori arbitrare. Deci nu se poate distinge dacă u_i este defectă sau nu. Presupunem în continuare că mulţimea de unităţi $u_2, u_3 \ldots u_k, u_i$ testează unitatea u_j. Dacă şi u_i a fost defectă (situaţie posibilă deoarece $k > t_p$) de asemenea nu se poate preciza starea unităţii u_j. Există în acest caz două situaţii ce nu se pot distinge.

Pentru $k=t_p$ şi presupunem că aceleaşi k unităţi, defecte testează unitatea u_i. În prima fază unitatea u_i nu este distinsă (dar este funcţional corectă, din ipoteza că numărul de unităţi defecte nu poate depăşi t_p unităţi). În faza a doua, datorită faptului că u_i este o unitate funcţional corectă va evalua corect

comportarea unității u_j, ceea ce duce la o situație diferită de cea din prima fază.

Pentru ilustrare se consideră un sistem S format din



n=5 unități și t_p=2, iar fiecare unitate este testată de k=1 ($k < t_p$) alte unități. Din figura 2.5 se poate constata ușor că cele două cazuri nu se disting, atunci cînd variabila x ia valorile posibile indicate. Prin adăugarea a unui test suplimentar pentru fiecare unitate (linia punctată din fig.2.5) atunci $k=t_p=2$, iar sindromul obținut în cele două cazuri prezintă valorile din tabelul 2.1.

Tabel 2.1

unități defecte	SINDROMUL OBTINUT									
	a_{12}	a_{13}	a_{23}	a_{24}	a_{34}	a_{35}	a_{45}	a_{41}	a_{51}	a_{52}
1	2	3	4	5	6	7	8	9	10	11
u_1, u_2	x	x	x	x	0	0	0	1	1	1
u_1, u_3	x	x	1	0	x	x	0	1	1	0

Se constată, în această situație, că cele două sindroame diferă fără prin elementul de pe ultima coloană, ceea ce permite să se facă distincția între cele două situații posibile de defect în sistem.

Rezultă că și această condiție este necesară și suficientă.

Se presupune în continuare că există o structură în care două unități se testează reciproc, iar sindromul rămîne diagnosabil fără reparații.

Fie trei seturi de unități U_1, U_2 și U_3, unde U_2 este

setul de unităţi defecte. Se mai presupune că $|U_2| \leqslant t_p$.

Se consideră că există cel puţin o unitate în setul U_1 şi U_2 care se testează reciproc. O structură posibilă de acest fel se prezintă în fig.2.6.

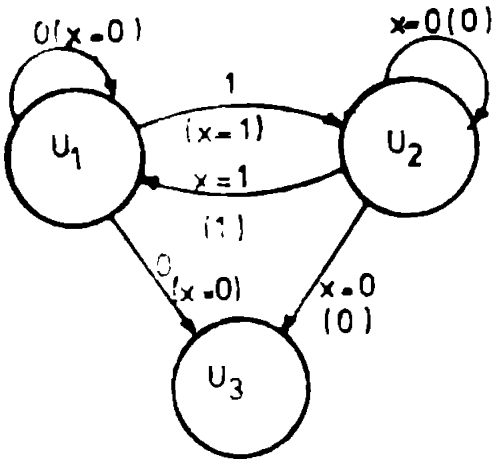


Fig.2.6.

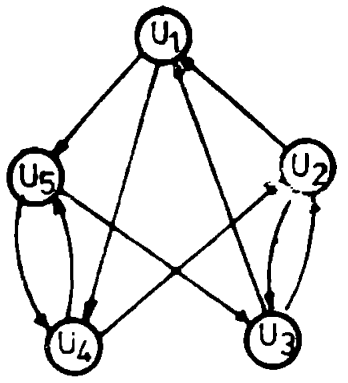


Fig.2.7

Rezultatele testelor sînt specificate pe figură. Dacă se consideră o altă împărţire posibilă a unităţilor din sistem în care unităţile din setul U_3 rămîn nemodificate. Mulţimile U_1 şi U_2 îşi modifică între ele unităţile ce se testează reciproc. În această situaţie rezultatele testelor specificate în fig.2.6 (în paranteză) nu se deosebesc de situaţia anterioară, ceea ce implică respectarea şi celei de a treia condiţie a teoremei. Un caz particular se prezintă în fig.2.7, unde dacă unităţile u_1 şi u_5 sînt defecte sau u_1 şi u_4, se poate obţine acelaşi sindrom prezentat în tabelul 2.2.

Tabel 2.2

unităţi defecte	SINDROM OBTINUT									
	a_{14}	a_{15}	a_{21}	a_{23}	a_{32}	a_{31}	a_{42}	a_{45}	a_{54}	a_{53}
1	2	3	4	5	6	7	8	9	10	11
u_1, u_2	x	x	x	x	1	1	1	0	0	0
u_1, u_3	x	x	1	1	x	x	0	0	0	1
u_1, u_4	x	x	1	0	0	1	x	x	1	0
u_1, u_5	x	x	1	0	0	1	0	1	x	x

Se observă că există posibilitatea, avînd în vedere că $x = \{0,1\}$ ca sindroamele de pe linia 3 şi 4 a tabelului 2.2 să fie identice, ceea ce face să nu se deosebească cele două cazuri posibile de defect.

Teorema 2.1 specifică condiţiile restrictive pe care trebuie să le îndeplinească un sistem autotestabil pentru găsirea unui număr maxim de defecte t_p din totalul de n unităţi existente în sistem. Cele t_p unităţi defecte vor putea să fie determinate simultan printr-o procedură corespunzătoare, dacă şi numai dacă structura de interconexiuni de test respectă cele trei condiţii ale teoremei de mai sus.

Pe baza celor demonstrate în teorema 2.1 se obţine următorul enunţ.

Coroler 2.1: Într-un sistem S, cu n unităţi, şi diagnosabil pentru t_p defecte, există pentru fiecare situaţie de defect un sindrom unic definit.

Dacă în sistem sînt îndeplinite condiţiile din teorema 2.1 rezultă că nu există o situaţie în care două cazuri de defect să prezinte acelaşi sindrom.

Numărul de sindroame posibile într-un sistem sînt:

$$NS_p = C_n^{t_p} \cdot 2^{t_p} \cdot t_p + C_n^{t_p-1} \cdot 2^{t_p(t_p-1)} + \ldots + C_n^2 \cdot 2^{t_p} \cdot 2 + C_n^1 \cdot 2^{t_p} \cdot 1 \qquad (2.11)$$

care pot fi generate de un număr de situaţii de defect din sistem obţinut prin relaţia:

$$ND_p = C_n^{t_p} + C_n^{t_p-1} + \ldots C_n^1 \qquad (2.12)$$

Pentru n=7 şi t_p=3 ; rezultă NS_p = 19.320 şi ND=63.

Lungimea unui sindrom în cazul că fiecare unitatee este testată de alte t_p unităţi este dată de relaţia:

$$LS_p = nxt_p \qquad (2.13)$$

Pentru n=7 şi t_p=3 ; rezultă LS_p=21.

Se constată că dintr-un număr mare de informaţii binare posibile (2^{LS_p}) doar o parte relativ mică asigură un sindrom unic definit. Restul informaţiilor binare sînt redondante, existînd posibilitatea ca unele combinaţii binare să nu se poată produce niciodată în cazul unor defecte permanente.

2.2.2. Alegerea unei structuri optime pentru SAT.

In paragraful de faţă se va studia structura de interconexiuni optimă ce trebuie să existe între unităţile care se testează, plecînd de la faptul că numărul de unităţi din sistem este n şi pot exista cel mult t_p unităţi defecte, care satisfac condiţiile din teorema 2.1.

Definitia 2.11. Un sistem S, diagnozabil fără reparații se zice că este optim interconectat dacă $n=2t_p+1$ și fiecare unitate este testată exact de t_p unități.

In general există mai multe structuri de interconectare optime pentru sistemul dat.

Definitia 2.12. Un sistem S se zice că este simetric interconectat, dacă există o legătură de test de la unitatea u_i spre u_j și dacă $\min(j-i)=\delta$ (modulo n) pentru j și i= 1,2,3...n. In acest caz structura de interconexiuni se va nota cu D_{δ,t_p} [130].

Definitia 2.13. Un sistem S, se zice că este asimetric interconectat, dacă unitatea u_i testează t_p unități oarecare și pentru care δ ia valori diferite la fiecare unitate.

Un exemplu din fiecare tip de structură se arată în fig 2.8 (sistem simetric) D_{1,t_p} respectiv fig.2.9 (sistem asimetric).

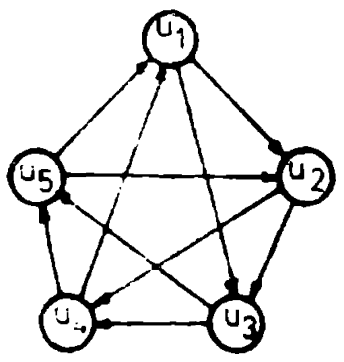


Fig.2.8

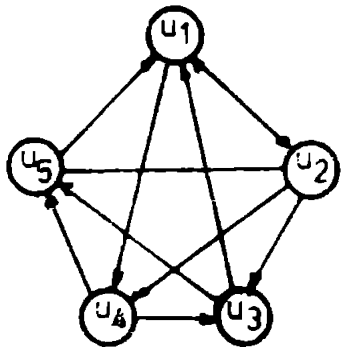


Fig.2.9

Lema 2.1.: Dacă un sistem S, admite o structură de interconexiuni D_{1,t_p} atunci el este diagnozabil fără reparații pentru t_p defecte [130].

Demonstrarea se face: plecînd de la o buclă ciclică, care cuprinde unitățile: $u_1,u_2...u_n$. Se consideră o secvență de $\beta < t_p$ unități defecte, care sînt cuprinse între două unități corecte.

Fie unitățile defecte: $u_i,u_{i+1},...u_{i+\beta-1}$. Deoarece:

$$[(i+\beta-1)+1]-(i-1)=\beta+1\leq t_p \qquad (2.14)$$

rezultă că structura D_{1,t_p} are o legătură care conectează unitățile u_{i-1} cu $u_{i+\beta}$ ambele funcționînd corect, prin ipoteză. In acest fel se va găsi întotdeauna o legătură(buclă)

între δ unităţi care sînt corecte ; pentru $\delta \geq t_p+1$ .Deci identificînd unităţile corecte se poate identifica şi unităţile (β la număr) care sînt defecte în sensul că fiecare unitate defectă este testată de cel puţin o unitate corectă şi deci identificată.

In cazul cînd cele $\beta = t_p$ unităţi sînt consecutiv defecte avem:

Fie unităţile defecte: $u_i, u_{i+1}, \ldots u_{i+t_p-1}$, aceasta implică că unităţile $u_{i+t_p}, u_{i+t_p+1}, \ldots u_{i-1}$, formează un şir de t_p+1 unităţi consecutiv corecte. Mai mult $a_{j,j+1}=0$ pentru $j=i+t_p$, $i+t_p+1 \ldots i-2$ şi $a_{i-1,i}=1$.

Dacă se presupune că unitatea u_{i-1} este defectă, în acest caz toate unităţile u_j pentru $j=i+t_p$, $i+t_p+1, \ldots i-2$, sînt corecte, ceea ce determină un total de

$$(i+t_p-1) - (i-2) = t_p+1 \qquad (2.15)$$

unităţi defecte, ce contrazice ipoteza referitoare la numărul maxim de defecte; deci şi unitatea u_{i-1} este corectă, rezultînd un total de

$$(i+t) - (i-1) = t_p+1 \qquad (2.16)$$

Deoarece u_{i-1} este fără defect şi cum aceasta testează unităţile defecte $u_i, u_{i+1}, \ldots u_{i+t_p-1}$ va rezulta că lema este demonstrată în sensul că cel puţin o unitate ştiută bună testează t_p unităţi defecte. Conform definiţiei 2.11 structura D_{1,t_p} este optimă şi conform definiţiei 2.12 structura D_{1,t_p} este simetrică.

Teorema 2.2. Dacă într-un sistem S se foloseşte structura D_{δ,t_p} astfel ca $(\delta,n) = 1$ (δ şi n sînt relativ prime) atunci S este diagnozabil fără reparaţii şi D_{δ,t_p} este o structură optimă.

Pentru demonstrarea teoremei se pleacă de la considerentul că o structură D_{δ,t_p} este simetrică şi conform [34,149] unde se arată că într-un graf finit G, cu $2n+1$ vîrfuri, se pot forma n cicluri hamiltoniene disjuncte.

Prin ciclu hamiltonian se înţelege ciclu care întîlneşte fiecare vîrf o singură dată (cu excepţia vîrfului iniţial care coincide cu vîrful final) [34].

Conform lemei 2.1, dacă într-un sistem există o buclă ciclică care cuprinde unităţile $u_1, u_2 \ldots u_n$, atunci se poate

identifica cel puţin o unitate bună, care să testeze cele t_p unităţi defecte, ceea ce conduce la concluzia că şi structura D_{δ,t_p} este optimă şi diagnozabilă.

Teorema 2.3. Dacă într-un sistem S, există cel puţin un ciclu format din t_p+1 unităţi, care conţine toţi termenii $a_{ij}=0$, atunci cele t_p+1 unităţi sînt funcţional corecte şi sistemul poate fi identificat.

Dacă se respectă condiţiile teoremei 2.1, atunci fiecare unitate u_i testează alte t_p unităţi. Avînd în vedere că două unităţi nu se testează reciproc rezultă că fiecare unitate testată va testa la rîndul ei cel puţin o unitate diferită de cele t_p unităţi testate de u_i. Fie unitatea u_j care testează aceleaşi unităţi ca şi u_i plus o unitate în plus, astfel:

unitatea u_i va testa unităţile $u_{i+1}, u_{i+2} \cdots u_{i+t_p}$

unitatea u_{i+1} va testa unităţile $u_{i+2}, u_{i+3} \cdots u_{i+t_p+1}$

---

unitatea u_{i+t_p} va testa unităţile $u_{i+t_p+1}, u_{i+t_p+2} \cdots u_{i+2t_p}$

unitatea u_{i+t_p+1} va testa unităţile $u_{i+t_p+2}, u_{i+t_p+3} \cdots u_{i+2t_p}$ (2.17)

unde: $i+2t_p+1 = i$.

deoarece şirul de unităţi $u_{i+1}, u_{i+2} \cdots u_{i+t_p}$ pot testa unitatea u_{i+t_p+1}, care la rîndul ei poate testa unitatea u_i rezultă că în acest lanţ de unităţi ce se testează există cel puţin t_p+1 unităţi:

$$(i+t_p+1) - i = t_p+1 \qquad (2.18)$$

Dacă în urma testelor, toate cele t_p+1 unităţi au fost găsite că prezintă pe $a_{ij}=0$ atunci pot exista doar două alternative: sau toate cele t_p+1 unităţi sînt defecte, ceea ce contrazice ipoteza, sau cele t_p+1 unităţi sînt corecte. Prin identificarea unităţilor corecte se pot identifica şi unităţile defecte, deci sistemul a fost diagnozat.

Corolar 2.2. Într-un sistem S autodiagnosabil există cel puţin n cicluri ce conţin t_p+1 unităţi.

Fiecare ciclu poate începe cu oricare din cele n unităţi, deci este evident că pot exista n cicluri format din t_p+1 unităţi conform teoremei 2.3.

Ideal ar fi ca în fiecare structură autodiagnozabilă să se poată identifica ciclu de t_p+1 unităţi funcţional corecte ceea ce ar duce la simplificarea diagnozei. Acest lucru însă nu se întîmplă la toate structurile, ceea ce impune creşterea informaţiilor redondante din sindromul sistemului.

Rezultă că structura optimă de diagnoză este cea simetrică cu un număr de $n=2t_p+1$ unităţi care satisfac condiţiile din teorema 2.1.

Diagnoza optimă se obţine pentru o structură optimă şi care satisface condiţiile teoremei 2.3.

Structurile care nu satisfac teorema 2.3 sînt acelea prin care cele t_p unităţi defecte testează aceeaşi unitate corectă, în această situaţie cele n cicluri de t_p+1 unităţi vor conţine cel puţin o unitate defectă, datorită restricţiei impuse ca două unităţi să nu se testeze reciproc.

2.2.3. Metodă de identificare a SAToptim diagnozabile

Pentru identificarea unui sistem autotestabil optim diagnozabil trebuie găsite structurile şi situaţiile de defect din sistem care permit obţinerea a cel puţin unui ciclu format din t_p+1 unităţi funcţional corecte. Un algoritm de identificare a SAToptim diagnozabile se dă mai jos.

Algoritm 2.1

1. Se formează matricea $MIT=(a_{ij})$ cu relaţia:

$$a_{ij}=\begin{cases}0 - \text{dacă unitatea } u_i \text{ corectă testează unitatea } u_j \text{ corectă.}\\ 1 - \text{dacă unitatea } u_i \text{ corectă testează unitatea } u_j \text{ defectă.}\\ x - \text{dacă unitatea } u_i \text{ este defectă şi testează unitatea } u_j.\end{cases} \tag{2.19}$$

2. Se formează vectorul $\underline{L}=(l_j)$, cu relaţia:

$$l_j = a_{1j} \cap a_{2j} \cap \dots \cap a_{nj} \tag{2.20}$$
pentru $j=1,2...n$

3. Se formează vectorul $\underline{C}=(c_i)$ cu relaţia:

$$c_i = a_{i1} \cap a_{i2} \cap \dots \cap a_{in} \tag{2.21}$$
pentru $i=1,2...n$

4. Se formează vectorul $\underline{V} = (v_k)$ cu relaţia:

$$v_k = l_j \cup c_i \text{ , pentru } i=j=k \qquad (2.22)$$
$$k=1,2...n$$

5. Dacă există cel puţin t_p+1 elemente al vectorului $\underline{V}$ care are valoarea 0 ($v_k=0$) atunci structura respectivă este optim diagnozabilă, deoarece unităţile u_k pentru care $v_k=0$ aparţin unui ciclu care conţine t_p+1 unităţi corecte.
6. STOP.

Vectorul linie $\underline{L} = (l_j)$ exprimă unităţile găsite corecte de cel puţin o unitate ce o testează. Valoarea unui element l_j este

$$l_j = \begin{cases} 1 - \text{dacă unitatea } u_j \text{ este găsită defectă de toate unităţile ce o testează.} \\ 0 - \text{dacă unitatea } u_j \text{ este găsită corectă de cel puţin o unitate ce o testează.} \\ x - \text{dacă unitatea } u_j \text{ este testată numai de unităţi defecte.} \end{cases} \qquad (2.23)$$

Vectorul coloană $\underline{C} = (c_i)$ determină dacă o unitate a găsit sau nu cel puţin o unitate corectă. Valoarea unui element c_i este

$$c_i = \begin{cases} 1 - \text{dacă nici o unitate nu este găsită corectă de către unitatea } u_i \\ 0 - \text{dacă cel puţin o unitate este găsită corectă de către unitatea } u_i. \\ x - \text{dacă unitatea } u_i \text{ este defectă.} \end{cases} \qquad (2.24)$$

Vectorul $\underline{V} = (v_k)$ arată dacă o unitate u_i este găsită cel puţin o dată corectă de altă unitate, atunci se consideră că testul lui u_i este corect. La rîndul ei dacă u_i determină că u_j este corectă se poate aprecia că testul lui u_j este corect. In final dacă unitatea u_k găsită anterior corectă determină că u_i este corectă se obţine un ciclu format din unităţi care prin testare au fost considerate bune. Dacă numărul de unităţi din ciclu este t_p+1 atunci sau găsit unităţile funcţional corecte conform teoremei 2.3. Pentru exemplificarea algoritmului se prezintă două sisteme; unul optim diagnozabil (fig.2.10),res-

pectiv diagnozabil (fig.2.11). Exemplu a fost dat pentru n=7, t_p=3. Ciclu obținut este format din unitățile u_2, u_4, u_5, u_7 (fig.2.10). Unitățile defecte din cele două exemple au fost marcate pe figură.

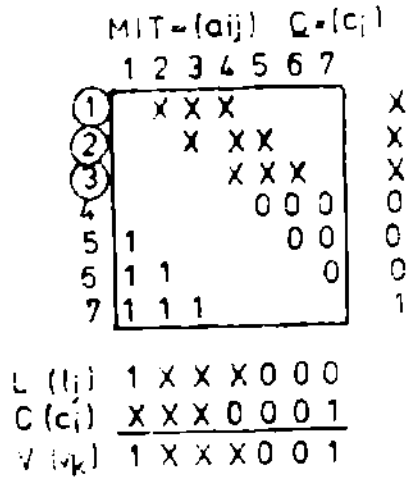


Fig.2.10

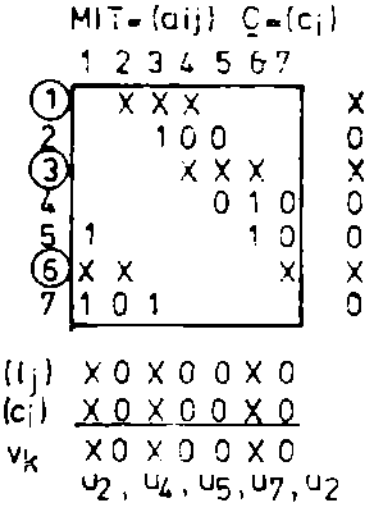


Fig.2.11

In capitolul 3 se dau două metode de diagnoză pentru cazul unor structuri de SAT oarecare.

2.3. Diagnoza SAT în prezența defectelor intermitente.

2.3.1. Defectele intermitente în cadrul SAT.

Definiția 2.14. Prin defecte intermitente, într-un sistem se înțeleg acele defecte care pot fi active la un moment dat, determinînd funcționarea eronată a sistemului, sau pot fi inactive la alt moment, ceea ce permite sistemului să funcționeze corect [154].

Defectele intermitente pot fi puse în evidență atunci cînd sînt active. Pentru activarea unui defect intermitent sînt necesare tehnici speciale de test care sînt studiate în literatură pentru circuite combinaționale [19,96,97] respectiv pentru circuite secvențiale [38,154,155]. In principiu metoda de punere în evidență a defectelor intermitente constă din repetarea unor teste (stimuli de test) de un număr de ori determinat prin metode statistice și probabilistice. Stimulii de test se aplică cu scopul de a face defectul intermitent activ. Tratarea unui defect intermitent activ se face cu procedee cunoscute pentru defecte permanente.

Pentru tratarea teoretică a defectelor intermitente în

cazul unor circuite cunoscute se foloseşte cel mai frecvent modelul Markov [154].

In cadrul unui sistem S descompus în n unităţi funcţionale se pune problema dacă apar defecte intermitente, modelul SAT pentru defecte permanente poate să pună în evidenţă acest tip de defecte.

In literatura de specialitate s-a încercat să se analizeze aspectele legate de defectele intermitente într-un sistem presupus diagnozabil pentru defecte permanente [114,116]. Trebuie precizat că analiza defectelor intermitente s-a făcut pe unele modele de diagnoză particulare.

In acest paragraf se va căuta găsirea unui model de structură generală care să permită şi diagnoza defectelor intermitente pentru toate cazurile, cu precizarea ca defectul să fie activ cel puţin o dată în cadrul aplicării procedurilor de test.

Presupunem că procedurile de test au fost bine alese, astfel ca defectul intermitent să fie activ cel puţin o dată. Modul de obţinere a procedurilor de test nu face obiectul acestei lucrări.

Existenţa unor defecte intermitente adaugă o nouă dimensiune la autodiagnoza defectului, în sensul că pot exista situaţii cînd o unitate defectă, testată insuficient este determinată ca bună, care la rîndul ei evaluează incorect o unitate bună. Pentru precizarea termenului se defineşte:

Definiţia 2.15. Prin diagnoză incorectă vom înţelege acea diagnoză care identifică ca defectă o unitate corectă.

Definiţia 2.16. Prin diagnoză incompletă vom înţelege acea diagnoză care nu identifică toate unităţile defecte, datorită unor proceduri de test necorespunzătoare.

Pentru punerea în evidenţă a faptului că o unitate a fost găsită defectă cel puţin p dată, în cazul defectelor intermitente, se va utiliza un sindrom caracterizat prin elementele sale definite cu relaţia:

$$s_{ij} = \begin{cases} 0 - \text{dacă toate unităţile } u_i \text{ sau } u_j \text{ sînt funcţional corecte.} \\ x - \text{dacă una din unităţile } u_i \text{ sau } u_j \text{ sînt defecte.} \end{cases}$$

unde: $x = \{0,1\}$ (2.25)

Valoarea lui x va fi "1" dacă cel puţin o dată unitatea u_j este găsită defectă de unitatea u_i în cadrul procesului de test.

Se poate constatat că în cazul unui test corect, sindromul unui defect intermitent aparţine sindromului de defecte permanente.

Numărul de sindroame intermitente ce pot exista într-un sistem cu t_i defecte este dată de relaţia:

$$NSi = C_n^{t_i} \cdot 2^{t_i(t_i+1)} + \sum_{j=1}^{t_i-1} (t_i-j)\, C_n^{t_i-1} \cdot 2^{t_i(t_i-1+1)} + \sum_{j=1}^{t_i-2} (t_i-j) + \ldots + C_n^1 \cdot 2^{2 \cdot t_i} \qquad (2.26)$$

Spre exemplu pentru $t_i=3$ şi n=7, rezultă NSi =207.335.

Numărul de sindroame intermitente este foarte mare şi dificil de analizat pentru obţinerea unei diagnoze corespunzătoare. Din acest motiv nu se vor lua în considerare decît sindroamele intermitente ce aparţin sindroamelor permanente:

$$S_i \in S_p \qquad (2.27)$$

Se pune problema dacă prin această limitare se poate realiza o diagnoză corectă, sau dacă nu există posibilitatea ca două cazuri de defect intermitent să genereze acelaşi sindrom $S_i \in S_p$ ceea ce ar duce la o diagnoză incorectă.

În cazul defectelor intermitente la fiecare set de test s-ar putea obţine alt sindrom, lucru care nu se întîmplă în cazul defectelor permanente. Din acest motiv se va stabili condiţia necesară şi suficientă pentru ca un SAT să poată diagnoza defectele intermitente pe baza sindromului de defecte permanente, ceea ce reduce substanţial numărul de sindroame posibil de analizat, dar care va introduce unele restricţii suplimentare în structură. În continuare se va considera sindromul de defecte permanente ca sindrom de bază.

Pe baza observaţiilor făcute rezultă că strategia urmărită în diagnoza defectelor intermitente este:

1. unităţile defecte sînt diagnozate pe baza sindromului de bază.

2. un set de unităţi defecte U_1 este unic definit de un sindrom S_i dacă $S_i(U_1) \in S_p(U_1)$.

3. Se admite şi o diagnoză incompletă.

2.3.2. Modelul de diagnoză a SAT în prezenţa Di

Se va determina condiţia necesară şi suficientă ca un sistem să fie autotestabil în cazul cînd unităţile din sistem sînt defecte intermitent.

Se presupune că sistemul S s-a descompus în n unităţi funcţionale care satisfac condiţiile din ipoteza 2.1. şi teorema 2.1.

Teorema 2.4: Un sistem S, cu n unităţi este diagnozabil fără reparaţii pentru t_i defecte intermitente dacă se îndeplineşte condiţia ca fiecare unitate să fie testată de cel puţin o unitate funcţional corectă.

Pentru demonstrarea teoremei se presupune că unităţile din sistem se pot împărţi în trei seturi U_1, U_2 şi U_3.

U_1 reprezintă setul de unităţi defecte, iar numărul de unităţi din set este $|U_1| \leq t_i$.

U_2 reprezintă un set de unităţi corecte cu $|U_2| \leq t_i$.

U_3 reprezintă setul de unităţi corecte rămase, unde:

$$U_3 = U - (U_2 \cup U_1) \qquad (2.28)$$

iar : $$U_1 \cap U_2 = \emptyset \qquad (2.29)$$

Pentru demonstrarea condiţiei necesare, presupunem că toate unităţile din setul U_2 sînt testate numai de unităţi defecte din setul U_1, deci

$$\Gamma^{-1}(U_2) \subseteq U_1 \qquad (2.30)$$

ceea ce implică că toate unităţile din setul U_3 vor testa numai unităţi din U_1:

$$\Gamma(U_3) \subseteq U_1 \qquad (2.31)$$

faptul că nu există legătură de test de la U_3 la U_2 poate duce la situaţia obţinerii unui sindrom în care toate răspunsurile testelor, $a_{ij}=0$, pentru $u_i \in U_3$ şi $u_j \in U_1$. Acest lucru se poate întîmpla în cazul unor testări insuficiente. În plus dacă toate răspunsurile $a_{ij}=1$, pentru $u_i \in U_1$ şi $u_j \in U_2$ şi $a_{ij}=0$ pentru u_i şi $u_j \in U_1$, situaţie posibilă deoarece unităţile din U_1 sînt defecte, ceea ce poate conduce la orice rezultat, deci şi la rezultatul presupus. Rezultatul testelor indică că sin-

drumul obţinut corespunde unei sătmţii reale de defecte permanente, care va indica că mulţimea unităţii U_2 este cel defect. Diagnoza obţinută prin neîndeplinirea condiţiei din teorema 2.4 este incorectă, ceea ce implică faptul că dacă unităţile din sistem sînt testate numai de unităţi defecte diagnoza defectelor intermitente nu este posibilă.

Demonstrarea condiţiei suficiente se va face pe baza următorului raţionament : Dacă U_1 este mulţimea de unităţi defecte, cu $|U_1| \geq t_1$, şi dacă în mod eronat se găseşte cel puţin o unitate corectă ca fiind diagnosticată defectă, va rezulta că mulţimea unităţii defecte este altul şi anume U_2 cu $|U_2| \geq t_1$, dar :

$$U_1 \cup U_2 = \emptyset \qquad (2.32)$$

rezultă că U_2 este mulţimea de unităţi diferite de U_1. Pentru două mulţimi de unităţi diferite li defecte, ar trebui să se obţină sindroame diferite, în caz contrar nu se poate preciza care din cele două mulţimi de unităţi este defectă.

Dacă fiecare unitate este testată de cel puţin o unitate bună, atunci :

$$\Gamma^{-1}(U_2) \subseteq U_1 \qquad (2.33)$$

care în mod automat implică :

$$\Gamma(U_3) \subseteq U_1 \qquad (2.34)$$

ceea ce înseamnă că cel puţin o unitate din U_3 testează cel puţin o unitate din U_2. In acest caz dacă U_3 testează unităţi din U_1 şi U_2, atunci pentru a fi determinată mulţimea U_2 ca defectă, este necesar ca cel puţin un răspuns $s_{ij}=1$ pentru $u_i \in U_3$ şi $u_j \in U_2$. Dar răspunsul nu poate fi niciodată "1", deoarece s-a presupus că U_1 este defectă iar U_2 nu va fi diagnosticat de către U_3 (mulţime de unităţi corecte) ca fiind defect. Deci orice sindrom ce indică unităţile din U_1 ca fiind defecte va fi diferit de sindromul ce corespunde situaţiei cînd U_2 este defect. Analiza unor sindroame diferite pentru cazuri diferite conduce la identificarea stării unităţilor din sistem.

In cazul că răspunsul $s_{ij}=0$ pentru toate sau unele unităţi $u_i \in U_3$ şi $u_j \in U_1$ conduce la concluzia că unităţile

u_j respective nu au fost testate corespunzător şi sînt găsite bune. Diagnoza în acest caz este incompletă datorită procedurii de test.

Răspunsul $s_{ij}=1$ pentru $u_i \in U$ şi $u_j \in U_2$ va conduce la un sindrom $S_i \in S_p$.

Într-o structură diagnosticabilă pentru defecte intermitente şi permanente există o legătură exprimată de următoarea teoremă :

Teorema 2.5: Un sistem S este diagnosticabil pentru t_i defecte intermitent dacă şi numai dacă este optim diagnosticabil pentru acelaşi număr de defecte permanente.

Pentru demonstrarea teoremei se pleacă de la faptul că un sistem optim diagnosticabil conţine cel puţin un ciclu format din t_p+1 unităţi corecte ce formează o buclă de test cu toate răspunsurile $s_{ij}=0$. Dacă se poate demonstra că cele t_p unităţi defecte dintr-un sistem diagnosticabil sînt testate de cel puţin o unitate corectă atunci s-a verificat afirmaţia din teorema 2.5.

Se va descompune sistemul în două mulţimi de unităţi , unde U_1 este mulţimea de unităţi corecte cu un număr de unităţi :

$$n_1 = t_p+1 \qquad (2.35)$$

şi U_2 mulţimea de unităţi defecte cu un număr de unităţi :

$$n_2 = t_p \qquad (2.36)$$

Cunoscînd faptul că numărul de legături de test spre o unitate este t_p şi cum o unitate nu se testează pe sine însăşi rezultă că cele t_p unităţi defecte sînt testate de cel puţin o unitate corectă. Dacă $t_i=t_p$ s-a demonstrat că sistemele care au cel puţin o buclă de test cu toate răspunsurile $s_{ij}=0$ este diagnosticabilă pentru defecte intermitente. Trebuie precizat că acest lucru se poate întîmpla şi în situaţia unei diagnoze incomplete, cînd se vor găsi corecte unităţi insuficient testate.

In cazul SAT pentru defecte permanente care nu sînt optim diagnosticabile, diagnoza defectelor intermitente nu este posibilă. Pentru a face şi aceste sisteme optim diagnosticabile propunem un algoritm, prin care se vor schimba ordinea de aranjare a unităţilor, de aşa manieră, încît sistemul să poată fi diagnosticat şi pentru defecte intermitente.

Algoritmul 2.2:

1. i = 1
2. $U_i^* = u_i x t_p - (t_p - 1)$
3. i = i+1
4. Dacă $i \leq n$; atunci: se trece la pasul 2.
   altfel: se trece la pasul 5.
5. STOP

Toate operaţiile cu indici se vor face în modulo n. Practic algoritmul 2.2 conduce la o structură de interconexiuni de test în care unităţile, din noul aranjament, să testeze cel puţin o unitate diferită faţă de cazul anterior. Dacă iniţial o unitate a fost testată de t_p unităţi defecte, în noul aranjament cel puţin o unitate diferită de cele t_p unităţi defecte va testa unitatea respectivă, şi această unitate nu poate fi decît o unitate corectă.

Spre exemplu dacă un sistem prezintă o structură în care unităţile au fost notate în ordinea crescătoare: $u_1, u_2, u_3 \ldots u_7$ pentru n=7, va prezenta următorul sindrom:

$$a_{12}, a_{13}, a_{14}, a_{25}, a_{24}, a_{25}, a_{34}, a_{35}, a_{36} \ldots a_{71}, a_{72}, a_{73} \qquad (2.37)$$

Prin aplicarea algoritmului 2.2 ordinea de aranjare a unităţilor este: $u_1, u_4, u_7, u_3, u_6, u_2, u_5$, cu sindromul:

$$a_{12}, a_{14}, a_{17}, a_{47}, a_{43}, a_{46}, a_{73}, a_{76}, a_{72} \ldots a_{51}, a_{54}, a_{57} \qquad (2.38)$$

Practic unitatea u_1 va testa unitatea u_7 în loc de u_2, unitatea u_2 va testa unitatea u_1 în locul unităţii u_3, ect.

Corolar 2.3. Oricare sistem care este diagnozabil fără reparaţii pentru t_i defecte este cel puţin diagnozabil fără reparaţii pentru t_p defecte permanente.

Pentru demonstrare se presupune că U_1 este setul de unităţi defecte şi se pune problema dacă pe baza structurii propuse de teorema 2.4 şi 2.5 este posibil să se găsească un alt sistem U_2 care este defect. Acest lucru este posibil numai dacă $U_2 \subset U_1$, ceea ce corespunde unei diagnoze incomplecte. Deci cel puţin t_p defecte permanente sînt detectate.

2.3.3. Relaţia dintre t_p şi t_i în cadrul SAT.

Condiţia din teorema 2.4 impune o structură restrictivă

pentru interconexiunile sistemului, în sensul că fiecare unitate trebuie să fie testată de cel puţin o unitate corectă. Cum în cazul SAT prezentat pentru defecte permanente o unitate este testată de t_p unităţi, rezultă că un sistem care este diagnozabil fără reparaţii pentru defecte permanente nu este sigur că poate fi diagnozabil pentru t_i defecte intermitente datorită unor condiţii restrictive suplimentare, şi anume ca sistemul să fie optim diagnosat conform teoremei 2.5.

În acest paragraf se consideră un sistem care este diagnosat pentru t_p defecte propunînd de a obţine limita t_i de unităţi ce pot fi defecte intermitent, respectînd structura propusă pentru defecte permanente.

Definiţia 2.17. Se va nota cu $\lfloor X \rfloor$ cel mai mare întreg mai mic decît X.

Definiţia 2.18. Se va nota cu $\lceil X \rceil$ cel mai mic întreg mai mare sau egal cu X.

Teorema 2.6. În oricare sistem S, cu n unităţi, care satisface condiţiile din teorema 2.1, numărul de unităţi ce pot fi defecte intermitente este dată de relaţia:

$$\left\lfloor \frac{2t_p+1}{3} \right\rfloor \leq t_i \leq (t_p-1) \qquad (2.39)$$

Demonstraţia va fi formată din două părţi. Partea întîia va conţine găsirea limitei inferioare, iar partea a doua găsirea limitei superioare.

Un sistem care îndeplineşte condiţiile din teorema 2.1 va avea fiecare unitate testată de cel puţin alte t_p unităţi.

Considerăm sistemul format din trei seturi de unităţi: U_1, U_2, U_3, unde U_1 este setul de unităţi defecte.

Cele trei seturi de unităţi satisfac relaţiile:

$$|U_1| \leq t_p \qquad (2.40)$$

$$|U_2| \leq t_p \qquad (2.41)$$

$$U_1 \cap U_2 = \emptyset \qquad (2.42)$$

$$U_3 = U-(U_1 \cup U_2) \qquad (2.43)$$

Dacă numărul de unităţi din U_1 este n_1, în acest caz numărul minim de legături de test din cadrul setului U_1 este:

$$m_1^1 = \frac{n_1(n_1 - 1)}{2} \qquad (2.44)$$

Această valoare implică un număr minim de legături de test de la setul U_1 spre celelalte unități ale sistemului. Cum fiecare unitate din setul U_1 testează cel puțin t_p alte unități, rezultă că numărul cel mai mic de legături de test de la U_1 este:

$$m_1 = n_1 \cdot t_p - \frac{n_1(n_1-1)}{2} \qquad (2.45)$$

Dacă n_2 este numărul cel mai mic de unități din setul U_2 necesar pentru a testat toate unitățile n_1 din setul U_1 va rezulta că se poate obține numărul minim de unități din setul U_2 ce testează unitățile din U_1 ca fiind egal cu:

$$m_2^1 = n_1 \cdot n_2 = n_1 \cdot t_p - \frac{n_1(n_1-1)}{2} \qquad (2.46)$$

sau:

$$n_2 = t_p - \frac{n_1-1}{2} \qquad (2.47)$$

cum n_2 trebuie să fie o mărime întreagă rezultă că:

$$|U_2|_{min} = n_{2\,min} = \left\lceil t_p - \frac{n_1-1}{2} \right\rceil \qquad (2.48)$$

Este posibil să se obțină o structură a sistemului care să fie divizat în subsetul U_1 și U_2 cu valorile de mai sus. Se observă că dacă crește valoarea lui n_1 valoarea minimă a lui n_2 va scădea (2.48). Numărul maxim de unități din setul U_1 și U_2 ($\max(n_1,n_2)$) pentru care se asigură îndeplinirea condiției din teorema 2.4 pentru un număr minim de unități defecte intermitente se obține când $n_1=n_2$:

$$n_1 = n_2 = t_p - \frac{n_1-1}{2} \qquad (2.49)$$

și:

$$n_1 = \frac{2\,t_p + 1}{3} \qquad (2.50)$$

Dacă mărimea $\frac{2t_p+1}{3}$ este o valoare întreagă atunci:

$$\max(n_1,n_2) = \frac{2t_p+1}{3} \qquad (2.51)$$

dacă $\frac{2t_p+1}{3}$ nu este o mărime întreagă atunci:

$$n_1 = \left\lfloor \frac{2t_p+1}{3} \right\rfloor \qquad (2.52)$$

şi

$$n_2 = \left\lceil \frac{2t_p+1}{3} \right\rceil \qquad (2.53)$$

Cum U_1 este setul de unităţi defecte rezultă că:

$$t_{i\ min} = n_1 \geq \left\lfloor \frac{2t_p+1}{3} \right\rfloor \qquad (2.54)$$

Pentru stabilirea limitei superioare a mărimilor t_i se ţine cont de corolarul 2.3 care stabileşte că t_i nu poate fi mai mare decît t_p, atunci cînd structura a fost proiectată pentru diagnoza unor defecte permanente. Rezultă că există cel puţin o unitate u_i care este testată de exact t_p unităţi Fie unitatea $u_i \in U_2$ şi setul $|U_1| = t_p$ unităţi care testează unitatea u_i. Rezultă că unitatea u_i este testată de t_p unităţi defecte (U_1 setul de unităţi defecte), ceea ce determină să se facă afirmaţia că valoarea lui t_i nu poate fi mai mare decît t_p rezultă că

$$t_{i\ max} < t_p \qquad (2.55)$$

sau

$$t_{i\ max} \leq (t_p-1) \qquad (2.56)$$

Teorema este demonstrată:

$$t_{i\ min} \leq t_i \leq t_{i\ max} \qquad (2.57)$$

pentru cazul că sistemul a fost construit să diagnoseze t_p defecte permanente.

In cazul că în sistem se poate realiza o reconfigurare a conexiunilor de test conform algoritmului 2.2, prin care fiecare unitate este testată de cel puţin o unitate diferită faţă de cele t_p unităţi iniţiale se obţine un sistem ce va avea practic cel puţin t_p+1 legături de test din care t_p sînt operaţionale la un moment dat, ceea ce face ca sistemul să nu mai fie optim interconectat, dar devine optim diagnozat.

Teorema 2.7. In orice sistem S, cu n unităţi, care este

diagnozabil pentru t_p defecte permanente şi care admite cel puţin o legătură de test suplimentară faţă de valoarea t_p, numărul de unităţi ce pot fi defecte intermitente este dată de relaţia:

$$\left\lfloor \frac{2t_p+3}{3} \right\rfloor \leq t_i \leq t_p \qquad (2.58)$$

Pentru obţinerea limitei inferioare a lui t_i se consideră că un sistem este diagnozabil pentru t_p defecte dacă se îndeplinesc condiţiile din teorema 2.1, ceea ce înseamnă că indiferent de modul de realizare a interconexiunilor de test, sistemul are active, în cadrul unui test, t_p legături spre fiecare unitate. Rezultă că numărul maxim de legături de test rămîn:

$$m = n.t_p \qquad (2.59)$$

pentru ambele structuri de test, cu deosebirea că cele două structuri de test diferă prin faptul că fiecare unitate testează cel puţin o unitate diferită. O structură care ar ţine cont de toate interconexiunile din sistem, luînd în considerare sistemul iniţial şi cel reconfigurat ar conduce la un număr de legături de test totale:

$$m = n(t_p+1) \qquad (2.60)$$

Pe baza celor arătate la determinarea limitei inferioare la teorema 2.6 rezultă că numărul cel mai mic de unităţi din setul U_2 necesar pentru a testa toate unităţile n_1 din setul U_1 va fi egal cu:

$$n_2 = (t_p+1) - \frac{n_1-1}{2} \qquad (2.61)$$

şi cum n_2 trebuie să fie o mărime întreagă rezultă că:

$$|U_2|_{min} = n_{2\ min} = \left\lceil t_p - \frac{n_1-1}{2} \right\rceil \qquad (2.62)$$

Deoarece n_1 descreşte cu creşterea lui n_2, numărul maxim de unităţi din setul U_1 şi U_2, care să asigure îndeplinirea condiţiei din teorema 2.6, pentru un număr minim de unităţi defecte intermitent se obţine cînd $n_1 = n_2$, ceea ce conduce la obţinerea valorii lui n_1 şi n_2:

$$n_1 \geq \left\lfloor \frac{2t_p+3}{3} \right\rfloor \qquad (2.63)$$

şi

$$n_2 = \left\lceil \frac{2t_p+3}{3} \right\rceil \qquad (2.64)$$

respectiv a limitei inferioare pentru t_i:

$$t_{i\ min} = n_1 \geq \left\lfloor \frac{2t_p+3}{3} \right\rfloor \qquad (2.65)$$

Pentru stabilirea limitei superioare, se face afirmaţia evidentă că t_i nu poate fi mai mare decît t_p. Deoarece sistemul este constituit din două structuri de interconexiuni în care, în final, fiecare unitate este testată de mai mult de t_p alte unităţi, rezultă că fiecare unitate este testată de cel puţin o unitate corectă, ceea ce conform teoremei 2.4 conduce la concluzia că sistemul este diagnozabil pentru t_i defecte deci:

$$t_{i\ max} = t_p \qquad (2.66)$$

Se face precizarea că dacă sistemul este proiectat pentru a diagnoza defecte intermitente atunci structura poate fi aleasă de la început ca să îndeplinească condiţia din teorema 2.4, fără a fi necesară o reconfigurare a structurii de interconexiuni.

2.4. Generalizarea diagnozei în SAT

Într-un sistem S, pot apărea simultan defecte permanente şi defecte intermitente. Se pune problema dacă prin folosirea unei structuri de interconexiuni de test adecvate nu se poate generaliza diagnoza sistemului pentru toate tipurile de defecte.

Pentru început se va pleca tot de la un SAT pentru defecte permanente, încercînd să găsim condiţiile ca acest sistem să fie diagnozabil pentru defecte permanente, defecte intermitente, cît şi pentru defecte permanente şi intermitente apărute simultan în sistem.

Se va adopta aceaşi strategie ca la defectele intermitente, luîndu-se în considerare doar acele sindroame ce aparţin la un sindrom permanent, acceptîndu-se şi diagnoza incompletă. O diagnoză incompletă poate să apară în cazul unei testări insuficiente a unităţilor ce sînt defecte intermitent.

Strategia utilizată la diagnoza tuturor defectelor se va baza pe următoarele consideremte:

1) Unităţile defecte sînt diagnozate numai pe baza sindro-

mului asociat defecteler permanente.

2) Un set de unităţi defecte U_1, este unic definit de un set de sindroame, care este asociat cu setul de sindroame S_t ce aparţine sindroamelor de defecte permanente: $S_t \in S_p(U_1)$.

3) Structura de interconexiuni de test trebuie să permită şi diagnoza cazurilor extreme: $t=t_p$, pentru $t_i=0$, respectiv $t=t_i$, pentru $t_p=0$.

2.4.1. Modelul generalizat de diagnoză a SAT.

Pentru determinarea condiţiilor necesare şi suficiente ca un sistem să fie diagnozat atît pentru defecte permanente cît şi pentru defecte intermitente, plecînd de la o structură a unui SAT pentru defecte permanente, se enunţă următoarea teoremă:

<u>Teorema 2.8:</u> Un sistem S, format dintr-un set de n unităţi (U) este diagnozabil fără reparaţii pentru t defecte, dacă pentru oricare set de unităţi defecte $U_2 \subseteq U$, există un alt set $U_1 \subseteq U$, pentru $U_1 \cap U_2 = \emptyset$ şi $|U_1| \leq t$, $|U_2| \leq t$, care satisfac condiţia de mai jos

c1) Fiecare unitate să fie testată de cel puţin o unitate funcţional corectă.

Condiţia din teoremă este satisfăcută în două situaţii:

a) cînd fiecare unitate testează exact t alte unităţi, iar în setul de unităţi defecte U_2 nu există nici o unitate care să testeze oricîte unităţi corecte, ceea ce implică:

$$\Gamma^{-1}(U_1) \not\subseteq U_2 \qquad (2.67)$$

şi:

$$\Gamma^{-1}(U_3) \not\subseteq U_2 \qquad (2.68)$$

unde: U_1 şi U_3 reprezintă două seturi de unităţi corecte, pentru:

$$U_3 = U - (U_2 \cup U_1) \qquad (2.69)$$

b) Dacă există totuşi o unitate corectă testată de t unităţi defecte, atunci pentru îndeplinirea condiţiei c1) se impune ca fiecare unitate să testeze t+1 alte unităţi. În acest caz:

$$|\Gamma(U_3)| > t+t_i - |U_1| \qquad (2.70)$$

unde:

$$t = t_i + t_p \qquad (2.71)$$

In acest caz se asigură că fiecare unitate să fie testată de cel puţin o unitate corectă.

Pentru demonstrarea condiţiei necesare se consideră că în sistem nu se îndeplineşte condiţia din teoremă, dar sistemul ar rămîne diagnozabil pentru t defecte:

$$\Gamma^{-1}(U_1) \subseteq U_2 \qquad (2.72)$$

şi

$$|\Gamma(U_3)| \leq t+t_i-|U_1| \qquad (2.73)$$

ceea ce conduce la o structură posibilă ca cea din fig.2.12,

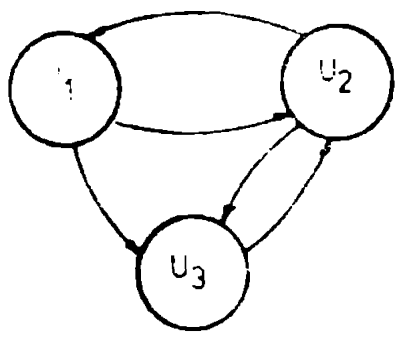


Fig.2.12

unde se poate constata că setul de unităţi U_1 este testat de unităţi aparţinînd lui U_2; acest lucru implică automat ca unităţile din setul U_3 să testeze numai unităţi din U_2:

$$\Gamma(U_3) \subseteq U_2 \qquad (2.74)$$

Se consideră că setul U_2 este divizat în setul unităţilor defecte permanent (U_{21}) şi setul unităţilor defecte intermitent (U_{22}) astfel ca:

$$|U_{21}| \leq t - |U_1| \qquad (2.75)$$

$$|U_{22} \cap \Gamma(U_3)| \leq t_i \qquad (2.76)$$

Împărţirea în cele două subseturi de unităţi a setului U_2 specifică că cel mult t_i unităţi din U_{22} sînt testate de unităţi din U_3 care testează cel mult $t+t_i-|U_1|$ unităţi din setul U_2 (relaţia 2.75).

Pe baza presupunerii iniţiale că U_2 este setul unităţilor defecte, unde subsetul U_{22} este format din unităţi defecte intermitent şi U_{21} din unităţi defecte permanent, unde:

$$|U_2| = |U_{21}| + |U_{22}| \leq t, \text{ pentru } t=t_p+t_i \qquad (2.77)$$

rezultă o situaţie posibilă cu următoarele răspunsuri de test:

s_{ij}=0 pentru toate testele t_{ij}, unde:

- $u_i \in U_1$, $u_j \in U_1$;

- $u_i \in U_1$, $u_j \in U_3$; determinat de faptul că U_1 şi U_3 este setul de unităţi corecte.

- $u_i \in U_{22}$, $u_j \in U_{22}$;

- $u_i \in U_3$, $u_j \in U_{22}$,

- $u_i \in U_{22}$, $u_j \in U_3$; determinat de faptul că U_{22} este setul de unităţi defecte intermitent.

s_{ij} = 1 pentru toate testele t_{ij}, unde:

- $u_i \in U_3$, $u_j \in U_{21}$, determinat de faptul că U_{21} este setul de unităţi defecte permanent şi U_3 este setul de unităţi corecte.

- $u_i \in U_{22}$, $u_j \in U_1$

- $u_i \in U_{22}$, $u_j \in U_{21}$; determinat de faptul că U_{22} este setul de unităţi defecte intermitent.

Pe baza sindromului obţinut rezultă că setul U_{21} şi U_1 este setul de unităţi defecte, unde:

$$|U_{21} \cup U_1| = |U_{21}| + |U_1| \leq t \qquad (2.78)$$

conform relaţiei (2.75), setul $(U_{21} \cup U_1)$ este un set ce satisface ipoteza din teoremă (2.76), ceea ce implică o diagnoză incorectă. Condiţia ca o unitate să fie testată de cel puţin o unitate corectă deci trebuie satisfăcută, sau dacă nu, diagnoza nu poate pune în evidenţă defectele intermitente pentru cazul extrem t_i=t.

Pe baza celor arătate rezultă condiţia necesară ca un sistem să fie diagnozat pentru t defecte.

Pentru demonstrarea condiţiei suficiente,vom considera că se îndeplinesc condiţiile din teoremă, dar sindromul nu este unic definit. Presupunem că sindromul aparţine la două situaţii de defect.

Fie U_2 şi U_4 două seturi de unităţi defecte, unde $|U_2| \leq t$ şi $|U_4| \leq t$, cu $U_2 \neq U_4$. Notăm cele două sindroame obţinute cu $S_2 \in S_p(U_2)$ şi $S_4 \in S_p(U_4)$.

Deoarece $U_2 \neq U_4$ poate exista situaţia ca $U_4 \not\subset U_2$, presupunem că:

$$U_{21} = U_2 \cap U_4 \qquad (2.79)$$

$$U_1 = U_4 - U_{21} \tag{2.80}$$

este clar că:

$$U_1 \cap U_2 = \emptyset$$

și

$$U_4 = U_1 \cup U_{21} \tag{2.81}$$

o astfel de împărțire se arată în fig.2.13. Deoarece sindromul nu este unic definit rezultă că

$$S_i \in (S_p(U_4) \cap S_p(U_2)) \tag{2.82}$$

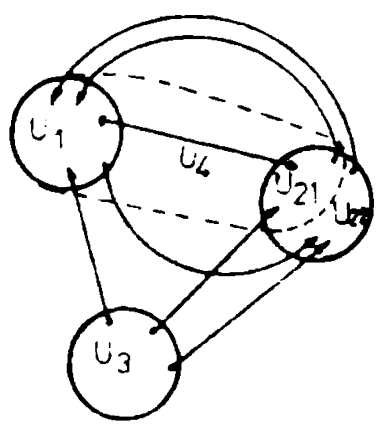


Fig.2.13

Dar în urma testului se poate obține următoarele răspunsuri: $a_{ij}=1$, pentru $u_i \in U_3$ și $u_j \in U_4$(U_4 conține subsetul U_{21} ce este defect permanent, respectiv $a_{ij}=0$ pentru $u_i \in U_3$ și $u_j \in U_2$(U_2 conține subsetul U_{22} care este defect intermitent). Acest sindrom posibil contrasice presupunerea că:
$S_i \in (S_p(U_4) \cap S_p(U_2))$.

Dacă în schimb avem situația:

$$\Gamma^{-1}(U_1) \subseteq U_2 \tag{2.82}$$

ceea ce implică că:

$$\Gamma(U_3) \subseteq U_2 \tag{2.83}$$

din relația (2.81) avem:

$$|U_{21}| = |U_4| - |U_1| \leq t - |U_1| = t_p \tag{2.84}$$

și din relația (2.73)

$$|\Gamma(U_3)| > t+t_i - |U_1| \geq t_p+t_i = t \tag{2.85}$$

Relația (2.85) indică că setul de unități U_3 testează toate unitățile din subsetul U_{21}, toate unitățile din subsetul U_{22} și cel puțin o unitate din setul U_1. Acest fapt conduce la următoarele răspunsuri de test:

$a_{ij} = 0$ pentru cel puțin un caz, unde $u_i \in U_3$, $u_j \in U_4$ (setul U_1 este inclus în setul U_4) și

$a_{ij}=1$ pentru $u_i \in U_3$ și $u_j \in U_{21}$ (U_{21} este inclus în setul U_2)
Acest sindrom de asemenea nu aparține la$(S_p(U_2) \cap S_p(U_4))$. Mai

mult relaţia (2.85) face ca relaţia (2.83) să nu mai fie valabilă, deci:

$$\Gamma(U_3) \not\subseteq U_2 \qquad (2.86)$$

care la rîndul ei conduce la concluzia că şi

$$\Gamma^{-1}(U_1) \not\subseteq U_2 \qquad (2.87)$$

Concluzia este că dacă fiecare unitate este testată de o unitate corectă sindromul este unic definit, în sensul că nu va fi indicată ca defectă o unitate corectă.

S-a presupus pentru demonstrarea condiţiei suficiente că ar putea să fie indicată ca defectă o unitate corectă. In continuare facem presupunerea că sindromul obţinut indică tot setul U_4 ca defect în locul setului U_2. U_4 este ales cu relaţiile (2.79) şi (2.81).

Se face precizarea că în situaţia cînd setul U_2 este setul de unităţi defecte, răspunsurile a_{ij} pot avea numai valoarea "0" pentru $u_i \in U_3$ şi $u_j \in U_1$.

Presupunem că U_4 ar putea fi indicat ca setul de unităţi defecte, dar din condiţia:

$$|\Gamma(U_3)| > t \qquad (2.88)$$

şi din relaţia (2.80) se obţine că:

$$|U_{21}| = |U_4| - |U_1| \leq t_p$$

ceea ce conduce la concluzia că subsetul U_{21} conţine cel mult t_p unităţi defecte permanent, mai mult din relaţia (2.85) rezultă că:

$$|\Gamma(U_3) \cap U_{22}| \geq t_i$$

subsetul U_{22} conţine mai multe unităţi defecte decît mărimea t_i. Cum U_2 este setul de unităţi defecte înseamnă că în subsetul U_{22} se găseşte cel puţin o unitate defectă permanent. In acest caz se va obţine cel puţin un răspuns $a_{ij}=1$, pentru $u_i \in U_3$ şi $u_j \in U_{22}$ şi t_p răspunsuri $a_{ij}=1$, pentru $u_i \in U_3$ şi $u_j \in U_2$. Pe de altă parte sindromul lui U_4 poate conţine cel puţin un răspuns $a_{ij}=0$, pentru $u_i \in U_3$ şi $u_j \in U_4$, datorită faptului că setul U_3 testează mai mult de t unităţi, din care cel mult t_i unităţi din U_{22} iar restul de unităţi testate pot aparţine lui U_{21}, care are cel mult t_p unităţi. Rezultă că U_3 va testa cel puţin o unitate co-

rectă, pentru care se obţine un răspuns a_{ij}=0. Dar această unitate nu poate aparţine decît setului U_1, respectiv U_4,ceea ce conduce la concluzia că U_4 va conţine cel puţin o unitate corectă şi care este găsită ca atare. Deci setul de unităţi U_4 nu poate fi un set de unităţi defecte.

S-a demonstrat în două etape că setul U_4 nu poate conţine numai unităţi defecte, respectiv chiar dacă ar conţine unităţi defecte sindromul obţinut este unic definit pentru fiecare situaţie de test în parte. Concluzia ce se obţine este că un sistem în care fiecare unitate este testată de cel puţin o unitate corectă este diagnozabil pentru t defecte aerecare.

In acest caz sindromul va conţine pentru toate testele $t_{ij} \in$ răspunsuri a_{ij}=0 pentru $u_i \in (U-U_2)$ şi $u_j \in (U-U_2)$ şi a_{ij}=1 pentru $u_i \in (U-U_2)$ şi $u_j \in (U_2)$ cu condiţia ca rutina de test să permită activarea defectelor intermitente cel puţin odată.

Teorema 2.8. este valabilă şi pentru cazurile extreme.

Pentru t_i=0 şi t_p=t atunci condiţiile teoremei 2.8 sînt evidente.

Pentru t_i=t_p şi t_p=0 atunci condiţia ca fiecare unitate să fie testată de cel puţin o unitate corectă a fost demonstrată în teorema 2.4, dacă această condiţie se obţine doar pentru:

$$|\Gamma(U_3)| > t_i+t-|U_1|$$

atunci pentru t_p=0 rezultă că:

$$|\Gamma(U_3)| > 2t_i - |U_1| \qquad (2.89)$$

şi cum: $|U_1| \leq t = t_i$ se obţine:

$$|\Gamma(U_3)| > t_i \qquad (2.90)$$

relaţia (2.90) implică că setul de unităţi corecte U_3 trebuie să testeze mai mult de t_i unităţi. Dar acest lucru implică că fiecare unitate corectă trebuie să testeze mai mult de t_i unităţi, ceea ce evident conduce la îndeplinirea condiţiei ca fiecare unitate să fie testată de o unitate corectă.

2.3.2. Relaţiile între tipul defectelor în cadrul unui SAT

Pentru stabilirea relaţiilor între tipul defectelor care pot apărea într-un sistem a cărui structură a fost organizată

să diagnozeze t_p defecte se dă următoarea teoremă:

<u>Teorema 2.9</u>: Un sistem S, care este diagnozabil pentru t_p defecte permanente, și în care există t_p legături de test, numărul de unități ce pot fi defecte se obține cu relația:

$$\left\lfloor \frac{2t_p+1}{3} \right\rfloor \leqslant t \leqslant t_p \qquad (2.91)$$

Fie un sistem diagnozabil pentru t_p defecte permanente, în care fiecare unitate este testată de t_p alte unități:

$$\left| \Gamma^{-1}(u_i) \right| = t_p \qquad (2.92)$$

pentru $u_i \in U$ și i=1,2...n.

Pentru demonstrarea limitei superioare se presupune definit setul de unități U_2 și U_3 prin relațiile:

$$U_2 = \Gamma^{-1}(u_i) \qquad (2.93)$$

$$U_3 = U - \{u_i\} - \Gamma^{-1}(u_i) \qquad (2.94)$$

este evident că:

$$\{u_i\} \notin \Gamma^{-1}(u_i) \qquad (2.95)$$

ce rezultă pe baza definirii lui $\Gamma^{-1}(u_i)$.

In cazul că $t=t_p$ și $t_i=0$ este îndeplinită relația

$$\left| \Gamma(U_3) \right| > t+t_i - \left| \{u_i\} \right|$$

sau:

$$\left| \Gamma(U_3) \right| > t_p+t_i - \left| \{u_i\} \right| \qquad (2.96)$$

Relația (2.96) poate să fie satisfăcută numai pentru $t_i=0$.

In cazul că $t_i > 0$ rezultă că $t < t_p$. Astfel limita superioară a lui t este

$$t \leqslant t_p \qquad (2.97)$$

Pentru $t_p=0$, avem $t=t_i$ și din teorema 2.6 se obține limita inferioară a lui t, care este:

$$t \geqslant \left\lfloor \frac{2t_p+1}{3} \right\rfloor \qquad (2.98)$$

Teorema este demonstrată și:

$\left\lfloor \frac{2t_p+1}{3} \right\rfloor \leqslant t \leqslant t_p$, pentru $t=t_p+t_i$

Ca o concluzie la cele prezentate referitoare la structura unor sisteme autotestabile se poate preciza că un sistem poate diagnoza t_p defecte permanente simultan, dacă sînt îndeplinite cele trei condiţii din teorema 2.1 şi anume:

c1) $n \geq 2t_p+1$

c2) $|\Gamma^{-1}(u_i)| \geq t_p$, fiecare unitate să fie testată de cel puţin t_p alte unităţi.

c3) Două unităţi nu se pot testa reciproc.

Iar în cazul posibilităţii de apariţie a defectelor intermitente mai apare o condiţie:

c4) Fiecare unitate să fie testată de cel puţin o unitate corectă.

Dacă structura a fost concepută să determine numai defecte permanente, atunci pentru a se găsi şi defecte intermitente este necesar ca structura să fie prevăzută cu posibilităţi de reconfigurare a legăturilor de test. Un algoritm posibil de reconfigurare a structurii sistemului este dat prin algoritmul 2.2. Prin această reconfigurare a structurii se păstrează numărul de legături de test, în schimb fiecare unitate îşi schimbă vecinul, ceea ce *implică* ca un sistem care nu a fost optim diagnozat, să devină optim diagnozat.

In capitolul 3 se vor da două metode de diagnoză a unui sistem autotestabil, plecîndu-se de la sindromul obţinut în urma efectuării testelor în sistem.

2.5. Simularea unor structuri de SAT

Pe baza condiţiilor necesare şi suficiente ca un sistem să poată deveni autotestabil s-au conceput mai multe programe, în FORTRAN, prin care s-au simulat astfel de structuri cu scopul de a verifica că sindromul ce se obţine este unic definit pentru fiecare situaţie de defect în parte.

Programele au fost concepute pentru cazul general şi anume se consideră un sistem S, format din n unităţi care satisfac condiţiile din ipoteza 2.1.

Simularea structurilor de SAT s-a făcut pentru mai multe cazuri.

2.5.1. Simularea unor SAT pentru defecte permanente.

Pentru simularea unor SAT a căror structură pot diagnoza t_p

defecte permanente, s-a luat în considerare cazul unor structuri simetrice, respectiv a unor structuri de interconexiuni asimetrice.

Pentru primul caz, programul de simulare SIMSIM (simularea structurilor simetrice) are ca date de intrare numărul de unităţi n şi numărul maxim de defecte din sistem, t_p. Programul a fost conceput să respecte prescripţiile din paragraful 2.2. Modul de realizare a interconexiunilor de test nu sînt necesare a fi indicate, deoarece structura sistemului a fost presupusă simetrică.

Cu ajutorul programului SIMSIM se obţin toate sindroamele de test, pentru toate cazurile de defect din sistem. Se poate constata că rezultatele obţinute, respectînd condiţiile din teorema 2.1, generează cîte un sindrom specific fiecărui caz în parte. Pentru exemplificare s-a luat n=7 şi t_p=3. In programul de simulare SIMSIM (P2.1) variabila X aparţine mulţimii {0,1} şi indică un răspuns de test nesemnificativ. Trebuie precizat că variabilele X = {0,1} din sindrom, introduc date redondante, fără a determina informaţii utile.

Programul de simulare SIMASIM (simularea structurilor asimetrice) P2.2, are ca date de intrare pe lîngă n şi t_p şi matricea interconexiunilor de test MCONMX = (t_{ij}), datorită structurii asimetrice a sistemului. Rezultatul obţinut, pentru fiecare situaţie de defect, confirmă faptul că sindromul este unic definit.

In P2.2 s-au adoptat valorile n=7 şi t_p=3.

2.5.2. Simularea unor SAT pentru defecte intermitente.

Programul de simulare pentru defecte intermitente a unor SAT, SIMSIT (P.2.3) a fost conceput să respecte condiţia din teorema 2.4, unde interconexiunea de test suplimentară se introduce prin aplicarea algoritmului 2.2. Se constată că numărul de elemente din fiecare sindrom creşte cu C_n^1. Prin legăturile de test suplimentare se asigură respectarea condiţiei din teorema 2.4 pentru toate cazurile de diagnoză, pentru $n \geq 2t_i+1$

Rezultatele obţinute în P2.3. confirmă faptul că sindromul este unic definit şi pentru defecte intermitente. Indiferent de sindromul obţinut în urma testelor se poate constata că diagnoza este corectă, atît în cazul unor teste incomplete

se poate obține o diagnoză incompletă, dar niciodată o diagnoză incorectă.

2.5.3. Simularea unor SAT pentru t defecte.

În cazul că în sistem apar defecte intermitente și permanente, simultan, se impune respectarea condiției din teorema 2.6. Având în vedere că astfel de structuri, pentru cazul cel mai defavorabil, când $t = t_i$, sînt identice cu structurile unor SAT diagnozabile pentru defecte intermitente, programul de simulare utilizat este același ca și în cazul simulării SAT pentru defecte intermitente, SIMSIT (P.2.3).

CAPITOLUL 3

METODE DE DIAGNOZA PENTRU STRUCTURI MULTIPROCESOR CU POSIBILITATI DE AUTOTESTARE

3.1. Diagnoza structurilor multiprocesor autotestabile

In situaţia cînd structura unui sistem poate fi descompusă în n unităţi distincte care satisfac condiţiile impuse de ipoteza 2.1, iar modul de organizare între unităţi respectă condiţiile de diagnoză automată propuse în capitolul 2 pentru diagnoza defectelor permanente:

1. $n \geq 2t+1$
2. Fiecare unitate este testată de cel puţin alte t unităţi.
3. Două unităţi nu se vor testa reciproc;şi condiţia suplimentară necesară diagonsi defectelor intermitente[174]
4. Fiecare unitate trebuie să fie testată de cel puţin o unitate funcţional corectă.

In acest caz o astfel de structură devine autotestabilă, generînd sindroame unic definite pentru fiecare situaţie de defect. Un sindrom se va obţine în urma generării unei rutine de test.

Pe baza sindromului obţinut conform [172] diagnoza nu este terminată, deoarece, în continuare, acesta trebuie comparat cu fiecare din sindroamele existente într-un dicţionar de sindroame. Dicţionarul de sindroame este constituit din totalitatea situaţiilor de defect din sistem, exprimată sub forma unui sindrom corespunzător fiecărui caz în parte.

Pentru comparare aceasta implica o unitate de comandă redondantă, care să asigure găsirea situaţiei de defect din sistem, specificîndu-se în acest fel unităţile defecte respectiv unităţile funcţional corecte,şi o unitate de memorie asociată dicţionarului de sindroame.

Numărul de situaţii de defecte din sistem este dat de relaţia:

$$N_d = C_n^1 + C_n^2 + C_n^3 + \ldots + C_n^t \qquad (3.1)$$

pentru:

$$n \geq 2t+1 \qquad (3.2)$$

unde: n reprezintă numărul de unităţi din sistem şi t numărul maxim de unităţi ce pot fi detectate defecte în sistem.

Pentru aceste N_d situaţii de defect sistemul poate genera un număr de sindroame.

$$N_s = C_n^t \cdot 2^{t \cdot t} + C_n^{t-1} \cdot 2^{t(t-1)} + C_n^{t-2} \cdot 2^{t(t-2)} + \ldots C_n^1 \cdot 2^{t \cdot 1} \qquad (3.3)$$

având un număr de cifre binare:

$$L_s = n \times t \qquad (3.4)$$

Memoria asociată dicţionarului de sindroame trebuie să fie prevăzută cu o capacitate de N_s cuvinte de L_s biţi, ceea ce implică, la o căutare secvenţială în memorie, un timp de diagnoză inacceptabil de mare. In plus memoria trebuie să fie astfel concepută încît să nu introducă erori în procesul de diagnoză.

Pentru a elimina neajunsurile legate de timpul de diagnoză mare şi capacitatea memoriei de asemenea relativ mare, în literatură s-au produs mai multe metode de determinare a unităţilor defecte, plecîndu-se de la sindromul defectelor.

Metodele prezentate în literatură prezintă dezavantajul că sînt asociate unor structuri particulare. Astfel în [3] se propune o metodă de diagnoză pentru structuri de procesoare interconectate sub forma unei reţele de cuburi n Booleene. Meyer şi Masson analizează în [110] un algoritm de diagnoză pentru structuri multiprocesor cu posibilităţi de interconectare simetrice. Pentru arhitecturi asimetrice se analizează în [113] un algoritm incomplect, dar care prezintă cîteva proprietăţi utile privind conceptul de defect implicat.

Abordarea probabilistică a diagnozei sistemelor numerice este pe larg prezentată în lucrările [71,111], iar în lucrările [6,22,100,140,142,157] problematica diagnozei este analizată pe baza unor modele de diagnoză ce impun un număr

mult mai mare de condiţii restrictive, care în practică sînt greu de realizat.

În continuare se vor prezenta două metode de diagnoză pentru determinarea unităţilor defecte într-o structură multimicroprocesor (multiprocesor) cu posibilităţi de autotestare. Metodele ce se vor analiza prezintă avantajul că pot fi aplicate pentru orice interconexiune între unităţile sistemului în cazul că unităţile din sistem satisfac condiţiile prezentate în capitolul 2. Algoritmii pot fi uşor implementaţi în aplicaţii în care unul sau mai multe unităţi u_i sînt realizate cu microprocesoare. Algoritmul mai poate fi implementat relativ simplu prin logică cablată sau printr-o unitate microprogramată cu un număr redus de componente faţă de restul sistemului. Logica redondantă introdusă pentru implementarea algoritmilor, fiind simplă, poate să fie realizată cu fiabilitate ridicată. În plus sistemul de diagnoză nu foloseşte dicţionarul de sindroame; diagnoza se face pe baza sindromului obţinut în urma unei rutine de test.

Pentru creşterea facilităţii de diagnoză se pot implementa algoritmii ce se vor propune folosind conceptul de diagnoză concurentă, propus în [164]. Metoda este utilă în sisteme ce se pot diviza într-un număr mare de unităţi, existînd posibilitatea ca acele unităţi din sistem care nu au de îndeplinit o anumită sarcină să se grupeze într-un subsistem, care să execute o rutină de test. Subsistemul devine astfel autotestabil, independent de celelalte unităţi, putînd fi considerat ca hardcorul întregului sistem, pînă la îndeplinirea condiţiilor de autotestabilitate a unui alt subsistem. Prin această procedură de diagnoză concurentă se asigură o utilizare eficientă a resurselor sistemului, cu o creştere a responsabilităţilor părţii de soft. Metoda poate fi aplicată doar în sisteme cu un număr de zeci de unităţi, care satisfac condiţiile enunţate mai sus, atît pentru întregul sistem cît şi pentru subsistemele ce se formează.

3.2. Metoda de diagnosă a defectelor prin utilizarea conceptului de defect implicat

Se consideră un sistem S format din n unităţi; $u_1, u_2 \ldots u_n$ şi interconexiunile de test date de matricea $T=(t_{ij})$, reprezentat sub forma unui graf: G=(N,T)(fig.3.1).

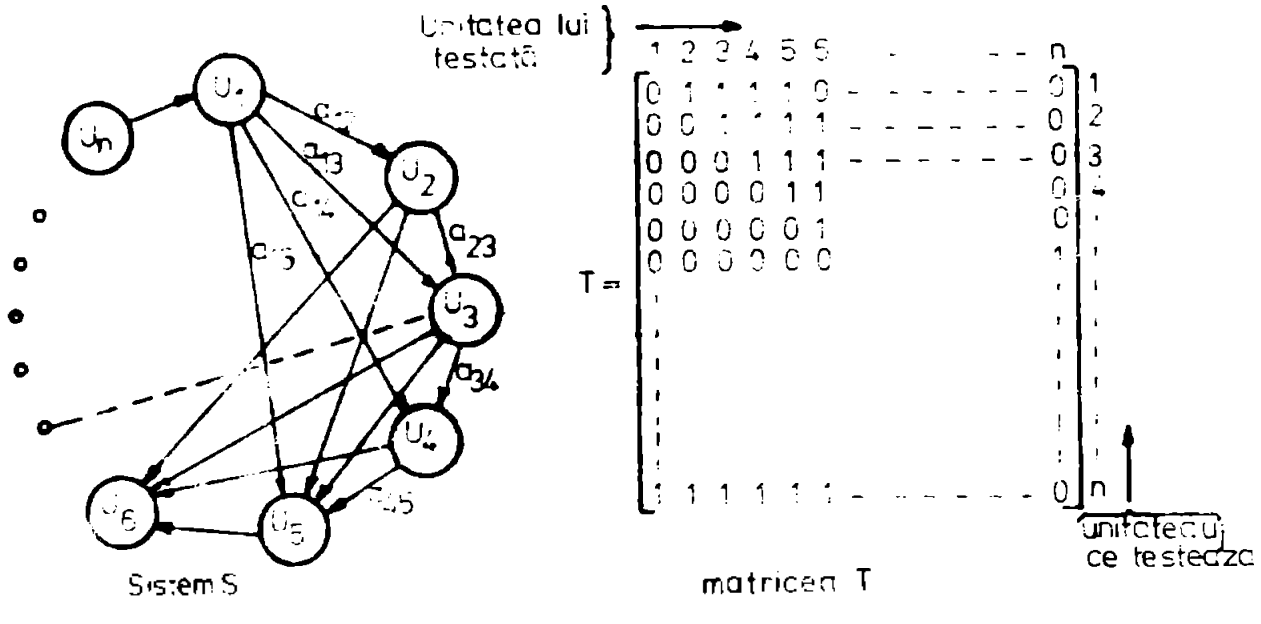


Fig.3.1

Răspunsul la un test este dat de matricea $A=(a_{ij})$ pentru i=1,2...n şi j=i+1,i+2...i+t pentru structuri simetrice, respectiv $j=b_1,b_2 \ldots b_t$, unde:

$$b_1 = \{1,2 \ldots n\} - \{i\}$$
$$b_2 = \{1,2 \ldots n\} - \{i,b_1\}$$
$$b_3 = \{1,2 \ldots n\} - \{i,b_1,b_2\}$$
$$\vdots$$
$$b_t = \{1,2 \ldots n\} - \{i,b_1,b_2 \ldots b_{t-1}\}$$

pentru structuri asimetrice.

Dacă sistemul S se poate diviza în două subsisteme B şi D unde subsistemul B conţine setul de unităţi funcţional corecte iar subsistemul D conţine setul de unităţi defecte, atunci problema diagnozei este rezolvată. Pentru identificarea celor două subsisteme se pleacă de la sindromul conţinut în matricea A, sindrom care este unic definit pentru o situaţie de defect dată. Găsirea celor două subseturi de unităţi determină terminarea procesului de diagnoză. Se poate constata relativ simplu că

pentru fiecare sindrom posibil din sistem îi corespunde o partiţie (B şi D), ceea ce face dificil de a găsi pentru sindromul obţinut partiţia căreia îi corespunde. Metoda căutării secvenţiale constituie o soluţie, dar prezintă, cum s-a văzut o serie de dezavantaje.

Metoda ce se va prezenta foloseşte conceptul de defect implicat [115] .

Definiţia 3.1.: Setul de unităţi implicate cu defecte, notat cu $L(u_i)$ a unităţii u_i (cu referire la un sindrom dat) este setul tuturor unităţilor din S pe care unitatea u_i le consideră defecte, presupunînd că unitatea u_i nu este defectă.

Definiţia 3.2.: O partiţie (B,D) a sistemului S este corectă (pentru un sindrom dat) dacă:

1) u_i este în subsetul B şi a_{ij}=0 implică că u_i este în subsetul B ;

2) u_i este în subsetul B şi a_{ij}=1 implică că u_i este în subsetul D.

Deci partiţia (B,D) a sistemului S este asociată la un sindrom dat, dacă presupunem că toate modulele din B sînt funcţional corecte şi toate modulele din D sînt defecte.

În cazul utilizării conceptului de defecte implicate rezultă următoarele corolare obţinute fără dificultate:

Corolar 3.1: Dacă o partiţie (B,D) a sistemului S este asociată unui sindrom dat, atunci: $L(B) \subseteq D$, cînd $L(B) = \{ u_i$ în S / pentru u_i aparţine lui B $\}$.

Corolar 3.2: Dacă o partiţie (B,D) a sistemului S este asociată unui sindrom dat atunci:

$D \subseteq \{u_i$ / pentru u_i aparţinînd la $L(u_j)$

Corolar 3.3: Dacă o partiţie (B,D) a sistemului S satisface relaţia:

$L(B) = D$

atunci (B,D) este o partiţie unic asociată unui sindrom dat.

Corolarul 3.1 specifică că toate defectele implicate, determinate de unităţile lui funcţional corecte ($u_i \in B$) sînt incluse în setul de unităţi defecte (D), conform definiţiei 3.2 punctul 2.

Corolarul 3.2 precizează că setul unităţilor defecte

sînt incluse în setul unităţilor implicate ca defecte. Ceea ce este evident deoarece în setul unităţilor implicate ca defecte sînt incluse atît unităţile defecte specificate de unităţile funcţional corecte cît şi unităţile propriuzis defecte

Corolarul 3.3 este evident, în sensul că dacă toate unităţile defecte sînt determinate de unităţile corecte, este clar că partiţia este cea căutată, pentru un sindrom dat.

In continuare se introduce conceptul de defect implicat invers.

Definiţia 3.3.: Setul de unităţi implicate ca defecte inverse, notat cu $L^{-1}(u_i)$, a unităţii u_i (cu referire la un sindrom dat) este setul tuturor unităţilor din sistemul S care consideră defectă unitatea u_i, presupunînd că unităţile respective sînt funcţional corecte.

Pe baza definiţiei 3.3 poate rezulta următoarea lemă:

Lema 3.1: Dacă $L^{-1}(u_i)=\emptyset$, pentru $u_i \in S$, atunci $u_i \in B$.

Prin această lemă se specifică că orice u_i care este evaluată corectă de toate cele t unităţi ce o testează, atunci unitatea respectivă aparţine setului de unităţi corecte (B).

Demonstraţia este simplă, în sensul că, unitatea u_i poate fi testată de unităţi corecte sau defecte. In cazul că unitatea u_i este testată de o unitate defectă, aceasta va putea să indice că unitatea u_i este defectă, şi în acest caz setul de unităţi $L^{-1}(u_i)$ nu mai este vid ($L^{-1}(u_i) \neq \emptyset$), ceea ce demonstrează *lema*.

Trebuie precizat că reciproca nu este adevărată. Adică unităţile funcţional corecte nu întotdeauna au setul $L^{-1}(u_i) = \emptyset$.

Algoritmul propus este destinat de a permite realizarea unei partiţii (B,D) a sistemului S, pentru un sindrom dat, de aşa manieră încît partiţia să fie unic definită pentru sindromul respectiv, cunoscînd faptul că sindromul este unic definit pentru o situaţie de defect dată.

Identificarea modulelor defecte şi funcţional corecte pe baza unui sindrom dat nu este posibil de realizat fără precizări suplimentare. In cazul căutării secvenţiale a sindromului prin dicţionarul de sindroame se preciza totalitatea sindroamelor ce ar putea să apară în sistem.

In cazul metodei ce se propune, precizările sînt făcute de definiţii şi cele trei corolare, pe baza cărora se defineş-

te următorul algoritm.

Algoritmul 3.1

1. $D = \emptyset$
   $B = \emptyset$

2. $K = 1$
   $D_K = \{ u_i$ / pentru $u_i \in L(u_i)\}$
   $B_K = \overline{D}_K$, pentru $D_K \subseteq S$

3. Dacă: $B_K \neq 0$;
   atunci: $D = D \cup \{L(u_i)$/pentru toate $u_i \in B_K\}$;
   $B = B \cup \{u_i$ / pentru toate $u_i \in B_K\}$;
   $D_{K+1} = D_K - D$;
   $B_{K+1} = B_K - B$;
   se trece la pasul 4.
   altfel: $B_{K+1} = D_K$;
   $D_{K+1} = B_K$;
   se trece la pasul 4.

4. $K = K+1$

5. Dacă: $D_K \cup B_K = \emptyset$;
   atunci: se trece la pasul 9.
   altfel: se trece la pasul 6.

6. $k_K = \max\{| L^{-1}(u_i) \cap B_K|$,/ pentru toate $u_i \in B_K\}$.

7. Dacă: $h_K = 0$;
   atunci: se trece la pasul 5.
   altfel: se trece la pasul 8.

8. $D_K = D_{K-1} \cup \{ u_i$ / pentru toate unitățile care au pe $|L^{-1}(u_i) \cap B_K| = h_K\}$,
   $B_K = B_{K-1} - D_K$, se trece la pasul 4.

9. STOP.

In fig.3.2 se prezintă ordinograma de implementare a algoritmului 3.1.

In faza inițială partiția (B,D) a sistemului S este vidă, nu se cunosc unitățile defecte și funcțional corecte La pasul următor, a algoritmului se repartizează mulțimii D_K toate unitățile din sistem găsite implicit defecte,iar în mulțimea B_K se repartizează unitățile care nu sînt spe-

cificate ca implicit defecte. In acest fel în D_K ar putea să fie atît unităţi propriu-zis defecte, cît şi unităţi funcţional corecte, pe cînd în mulţimea B_Δ se vor repartiza doar unităţi corecte.

Dacă mulţimea B_Δ conţine cel puţin o unitate corectă u_i atunci acea unitate (sau unităţi) pot fi trecute în partiţia (B) conform lemei 3.1. Pe de altă parte unităţile corecte vor genera o diagnoză corectă, ceea ce implică că toate unităţile găsite defecte de unitatea u_i (unităţile) corectă se vor repartiza partiţiei (D), conform corolarului 3.1. In continuare din mulţimile iniţiale D_K şi B_Δ se vor elimina unităţile găsite defecte, respectiv corecte şi se trece la pasul următor(4) Dacă mulţimea B_K este vidă atunci rezultă că nu se poate preciza comportarea unităţilor din sistemul S.

In această situaţie setul de unităţi D_Δ va conţine atît unităţi defecte cît şi unităţi corecte. Si cunoscînd că numărul de unităţi corecte depăşeşte cu cel puţin o unitate numărul de unităţi defecte putem face afirmaţia că este mai probabil ca unităţile din setul D_Δ să fie funcţional corecte, decît defecte, ceea ce permite transferul unităţilor din setul D_Δ în setul B_Δ.

In pasul patru se trece la iteraţia următoare, iar în continuare se verifică dacă în cele două mulţimi B_Δ şi D_Δ mai sînt unităţi a căror comportări nu o cunoaştem. Dacă nu mai sînt unităţi, atunci procesul de repartizare a unităţilor în cele două partiţii s-a încheiat, identificîndu-se în acest fel unităţile corecte şi defecte din sistem pe baza sindromului dat.

In cazul că mai sînt unităţi neidentificate în cele două multimi se trece la pasul următor (6), unde pentru fiecare unitate din mulţimea B_K se caută unităţile care le testează. Unităţile din B_Δ testate şi găsite defecte de cel mai mare număr de unităţi din B_K, presupuse că nu sînt defecte, vor fi trecute în mulţimea D_Δ, (pasul 8). Procedura continuă pînă cînd nici o unitate din B_K nu mai este găsită defectă, ceea ce implică că $h_K=0$ şi prin revenire la pasul 3, unităţile corecte respectiv unităţile implicit găsite defecte sînt repartizate partiţilor corespunzătoare. Procedura continuă iterativ pînă se satisface relaţia L(B)=D şi conform

corolarului 3.3, partiţia (B,D) obţinută devine unie asociată sindromului dat.

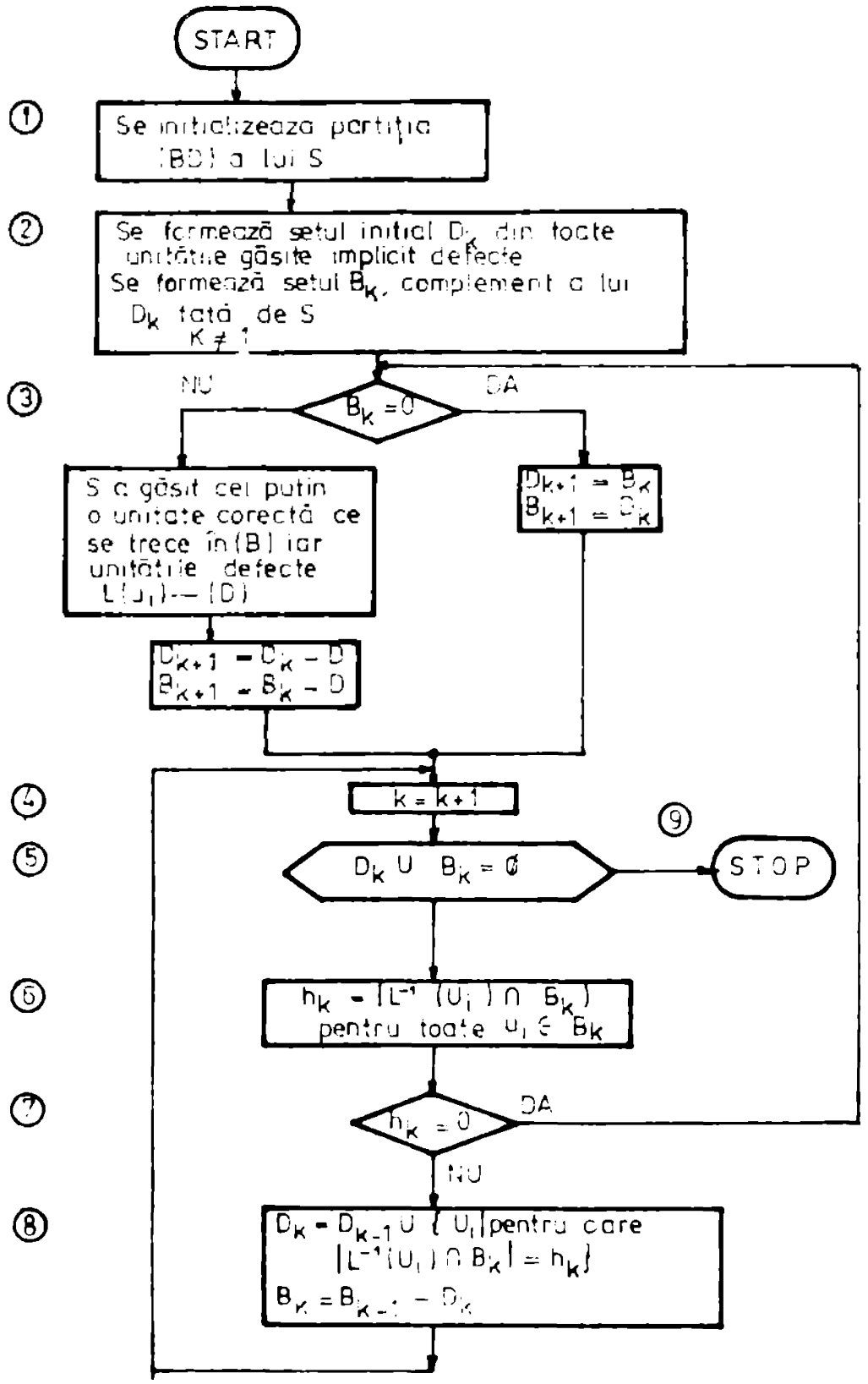


Fig.3.2

Pentru a demonstra că algoritmul 3.1 este corect se va considera următoarea teoremă:

Teorema 3.1. Dacă într-un sistem S sînt îndeplinite condiţiile de autotestabilitate, pentru defecte permanente:

c1) $n \geq 2t+1$

c2) Fiecare unitate este testată de cel puţin t alte unităţi.

c3) Două unităţi nu se testează reciproc,

atunci există cel puţin o unitate funcţional corectă în submulţimea $B_A \subseteq S$ care să satisfacă relaţia:

$$L^{-1}(u_i) = \emptyset$$

după un număr oarecare de iteraţii ale algoritmului 3.1. Demonstrarea teoremei se va face discutînd cazul cel mai defavorabil care apare în sistem. Se consideră că toate cele t unităţi defecte testează aceeaşi unitate corectă, ceea ce implică că va exista cel puţin o unitate corectă care va teste toate cele t unităţi defecte. In acest caz relaţia:

$$h_K = \max \{ | L^{-1}(u_i) \cap B_K | , / \text{ pentru toate } u_i \in B_K \} \qquad (3.5)$$

va da un maxim sigur pentru cel puţin o unitate defectă şi poate să mai dea un maxim pentru cazul cînd toate unităţile defecte indică că unitatea corectă testată de ele este defectă. Acest lucru implică că din mulţimea B_K se elimină respectiv se adaugă în D_A cel puţin o unitate defectă şi cel mult acelaşi număr de unităţi funcţional corecte. Cazul cel mai defavorabil devine cînd există paritate între unităţile defecte şi corecte eliminate.

In continuare, la iteraţia următoare procesul se repetă, astfel că în final în mulţimea B_K rămîne cel puţin o unitate pentru care se îndeplineşte relaţia:

$$h_{K_i} = | L^{-1}(u_i) \cap B_K | = 0 \qquad (3.6)$$

şi din condiţia c1 : $n \geq 2t+1$ rezultă că unitatea (unităţile) pentru care relaţia (3.6) este adevărată nu poate fi decît o unitate (unităţi) funcţional corecte, adică:

$$L^{-1}(u_i) = \emptyset \text{ , pentru } u_i \in B_K.$$

Dacă se găseşte cel puţin o unitate u_i corectă, aceasta va determina la rîndul ei cel puţin o unitate defectă : $u_j \in L(u_i)$. Numărul maxim de unităţi rămase în sistem neiden-

tificate după prima trecere prin algoritm devine:

$$n-1 \geq 2(t-1) + 1 \qquad (3.7)$$

sau:

$$n \geq 2(t-1) + 2$$

In următoarea trecere prin algoritm se vor găsi cel puţin două unităţi funcţional corecte, iar numărul maxim de unităţi rămase neidentificate sînt:

$$n-2 \geq 2\left[(t-1) - 2\right] + 2 \qquad (3.8)$$

sau:

$$n \geq 2(t-3) + 4$$

Procedura poate continua pînă la găsirea tuturor unităţilor corecte respectiv defecte.

In cazul cel mai defavorabil submulţimea B_k va conţine pentru fiecare iteraţie K un număr de unităţi date de relaţia recursivă:

$$\begin{aligned} &\text{pentru } K = 1 , \quad |B_1| = n \geq 2t+1 \\ &\text{pentru } K = 2 , \quad |B_2| = n-2 \geq 2(t-2) + 1 \\ &\text{pentru } K = 3 , \quad |B_3| = n-4 \geq 2(t-4) + 1 \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.9) \\ &\text{pentru } K=t+1, \quad |B_k| = n-2t \geq 2(t-2t) + 1 \\ &\text{sau } n \geq 1, \end{aligned}$$

In urma procesului iterativ dat de relaţia (3.9) cel puţin o unitate corectă va fi identificată, şi anume unitatea (unităţile) ce rămîn în submulţimea B_k cînd $L^{-1}(u_i)=\emptyset$.

Teorema 3.1 confirmă că există cel puţin o unitate care este găsită corectă după parcurgerea algoritmului 3.1. La rîndul ei unitatea u_i, corectă, va indica ca defectă cel puţin o unitate defectă. Algoritmul se va aplica în continuare unui subsistem ce prezintă acelaşi caracteristic ca şi cel iniţial.

In final după cel mult t treceri prin algoritm vor fi identificate toate unităţile ceea ce permite obţinerea partiţiei (B,D) corespunzătoare sindromului dat.

Prin implementarea algoritmului într-o unitate de comandă redondantă se elimină necesitatea alocării de memorie dicţionarului de sincrome. In cazul că sistemul este realizat într-o structură multimicroprocesor, algoritmul poate

fi prelucrat de unul dintre microprocesoarele funcţional co recte.

3.3. Metoda de diagnoză a defectelor prin utilizarea conceptului de matrice de incidenţă

Conform celor arătate în capitolul 2, orice sistem S se poate descompune în n unităţi : $u_1, u_2, \ldots u_n$, a căror proprietăţi, şi mod de interconexiune au fost prezentate, astfel încît sistemul să genereze un sindrom unic definit, care să caracterizeze o situaţie de defect, din sistem, şi numai una.

Pe baza sindromului obţinut, în urma executării unei rutine de test, şi a matricei de interconexiune de test $T=(t_{ij})$ ne propunem să obţinem o a doua metodă de diagnoză a defectelor bazată pe conceptul de matrice de incidenţă.

3.3.1. Matricea de legătură şi matricea de incidenţă.

Matricea de legătură este definită să exprime într-o formă sintetică sindromul obţinut pentru o interconexiune a sistemului dată.

Definiţia 3.4. Prin matricea de legătură, vom defini o matrice sintetică a interconexiunilor de test şi a rezultatelor (sindroamelor) acestora, notată: MIRET = (a_{ij}), unde elementele (a_{ij}) ale matricei au valorile exprimate de relaţia (3.10)

$$a_{ij} = \begin{cases} 0 - \text{dacă unitatea } u_i \text{ testează unitatea } u_j \text{ şi consideră că aceasta este fără defect.} \\ 1 - \text{dacă unitatea } u_i \text{ testează unitatea } u_j \text{ şi consideră că aceasta este defectă.} \\ * - \text{dacă unitatea } u_i \text{ nu are legătură de test directă cu unitatea } u_j. \end{cases}$$

$$a_{ii} = 0$$

pentru : $i = 1,2\ldots n$

$j = 1,2\ldots n$ (3.10)

În acest fel o linie i din matricea MIRET reprezintă unitatea u_i care testează iar coloana j reprezintă unitatea j de testat, iar elementele matricei exprimă rezultatele testelor.

Pe baza testelor considerate bune, din matricea MIRET se va defini o matrice de incidenţă a unui test complect, notată

cu MATEST = (r_{ij}).

Definiția 3.5. Matricea de incidență, MATEST = (r_{ij}) va avea următoarele valori pentru elementele sale [174,212] :

$$r_{ij} = \begin{cases} 1 - \text{dacă unitatea } u_i \text{ consideră că unitatea } u_j \text{ este} \\ \quad \text{defectă, chiar dacă unitatea } u_i \text{ apreciază sta-} \\ \quad \text{rea unității } u_j \text{ prin intermediul altei unități.} \\ 0 - \text{în caz contrar.} \end{cases}$$

pentru: i = 1,2...n
j = 1,2...n (3.11)

Fiecare vector $\underline{r_i}$ de incidență conține stările unităților u_j care sînt testate de către unitatea u_i direct sau prin intermediul altei unități u_x, apreciată corectă de către unitatea u_i.

Vectorul $\underline{r_i}$ se formează iterativ după relația următoare

$$\underline{r_i} = \begin{cases} \underline{r_i^1} = \underline{e_i} \\ \underline{r_i^k} = \underline{r_i^{k-1}} \cup \left(\bigcup_{\ell=1}^{m} e_\ell \cap \overline{r_{i\ell}^{k-1}} \right) \\ \underline{r_i} = \underline{r_i^k} \end{cases}$$

pentru: j = 1,2...n
i = 1,2...n (3.12)

Vectorul $\underline{r_i}$ se obține în n pași. Valoarea finală a vectorului $\underline{r_i}$ este atinsă atunci cînd

$$\underline{r_i^k} = \underline{r_i^{k+1}} \qquad (3.13)$$

unde: $k \leq n$ reprezintă numărul iterației.

Matricea MATEST = (r_{ij}) se obține cu următorul algoritm:

Algoritmul 3.2:

1. Se formează matricea MINET = (e_{ij}), după relația (3.10).
2. Se determină primul vector $\underline{r_i}$ (i=1, j=1,2...n) în mod iterativ prin:
   a) $\underline{r_i^1} = \underline{e_i}$; linia i a matricei MINET, pentru: j=1,2...n.
   b) $\underline{r_i^k} = \underline{r_i^{k-1}}$ SAU toate liniile $\underline{e_j}$ ale matricei MINET, care au fost marcate prin valoarea binară "0" în vectorul

$\underline{r_i}$ de la pasul 2.a.

c) reluarea ecuației de la pasul 2b, până când r_i^k și r_i^{k+1} nu mai produc nici o schimbare în forma vectorului $\underline{r_i}$.

3. i = i+1. Dacă i ≤ n:

atunci: se trece la pasul 2.

altfel: se trece la pasul 4.

4. STOP.

Cu ajutorul algoritmului 3.2 se obține matricea MATEST = (r_{ij}) prin care se specifică modul de transmitere a informațiilor de diagnoză prin întreaga structură a sistemului. Fiecare element r_{ij} reprezintă concluzia unității u_i referitoare la starea modulului u_j(j=1,2...n), chiar dacă unitatea nu este testată direct de u_i.

3.3.2. Detecție defectelor cu ajutorul matricei de incidență.

Plecînd de la algoritmul 3.2 de obținere a matricei de incidență ne propunem să demonstrăm că această matrice poate fi utilizată la detecția defectelor într-o structură autotestabilă și că diagnoza ce se obține reflectă corect situația din sistem.

Pentru sistemul S analizat ce satisface următoarea ipoteză

Ipoteza 3.1

c1) un sistem S se poate descompune în n unități.

c2) Fiecare unitate poate diagnoza alte t unități.

c3) Fiecare unitate este diagnozată la rîndul ei de t unități.

c4) $n \geq 2t+1$

c5) Două unități nu se testează reciproc.

În această situație se poate scrie următoarea teoremă:

Teorema 3.2: În matricea de incidență, există cel puțin n-t vectori $\underline{r_i}$ care sînt identici.

Demonstrația teoremei se va face pe baza observației după care a fost obținută matricea de incidență MATEST. Se va ține cont că numărul de unități funcțional corecte trebuie să depășească cel puțin cu o unitate numărul de unități defecte; condiția c4 din ipoteza 3.1. Se pleacă de la vectorul $\underline{a_i}$ ce formează în prima iterație vectorul $\underline{r_i}$. Cum unitatea u_i testează

t alte unităţi şi presupunînd că u_i este fără defect, rezultă că diagnoza acestor t unităţi este corectă. In urma diagnozei pot apărea două cazuri diferite.

Cazul a). Dacă toate unităţile t diagnozate sînt găsite defecte, de către unitatea u_i, rezultă că vectorul $\underline{r_i} = \underline{a_i}$ iar procedura de diagnoză efectuată de unitatea u_i s-a încheiat. Celelalte n-t unităţi care nu sînt testate de u_i sînt corecte conform punctului 4 a ipotezei 3.1.

Cazul b). Dacă cel puţin o unitate u_k este găsită fără defect de unitatea u_i (presupusă corectă) rezultă că diagnoza acestei unităţi, u_k, este acceptată ca bună şi se adaugă prin reuniune la diagnoza unităţii u_i, în pasul 2b a algoritmului 3.2. La rîndul ei unitatea u_k testează cel puţin o altă unitate, diferită de unităţile testate de u_i şi această unitate u_j, va fi găsită defectă sau corectă. In situaţia că u_j este defectă procedura de formare a vectorului r_i s-a încheiat. Dacă u_j este găsită funcţional corectă de către unitatea u_k, aceasta se va găsi şi ea în una din cele două cazuri. Procedura de obţinere a vectorului r_i se termină cînd sînt găsite fie cele t defecte (dacă sînt) fie cînd $r_i^k = r_i^{k+1}$, ceea ce înseamnă că vectorul r_i este plin ($j=1,2...n$) şi că unitatea u_i a adăugat prin reuniune diagnozele tuturor unităţilor considerate de către ea ca funcţional corecte.

Conform punctului 4 din ipoteza 3.1, care specifică că numărul de unităţi corecte trebuie să depăşească cu cel puţin o unitate pe cele defecte rezultă acum în mod evident că numărul de vectori r_i, ce sînt asociaţi unităţilor corecte sînt n-t. Mai mult cei n-t vectori r_i vor fi identici, deoarece în mod sigur unităţile corecte, prin diagnoza lor, vor reflecta situaţia reală din sistem.

Se pune în continuare întrebarea dacă cei n-t vectori r_i ce sînt identici nu cumva aparţin la două sau mai multe seturi de vectori. Pentru aceasta enunţăm următoarea teoremă:

Teorema 3.1. Intr-un sistem S, cu $n \geq 2t+1$ unităţi, unde t este numărul maxim de defecte ce pot fi localizate, există numai un singur set de vectori $V(r_1, r_2...r_m)$, în matricea de incidenţă, ce conţine un număr $m \geq n-t$ vectori identici.

Demonstrarea teoremei se face presupunînd contrariu. Se

consideră că există două seturi de vectori cu un număr m_1 respectiv m_2 de vectori identici ce satisfac condiţiile:

$$m_1 \geq n-t \qquad (3.14)$$

$$m_2 \geq n-t \qquad (3.15)$$

prin adunarea celor două relaţii (3.14) şi (3.15) se obţine:

$$m_1 + m_2 \geq 2n-2t \qquad (3.16)$$

dar din modul de structurare a sistemului trebuie ca:

$$n \geq m_1 + m_2 \qquad (3.17)$$

din relaţiile (3.16) şi (3.17) rezultă că:

$$n \geq m_1+m_2 \geq 2n-2t \qquad (3.18)$$

din condiţia c4, ipoteza 3.1

$$n \geq 2t+1 \qquad (3.19)$$

se obţine:

$$n-1 \geq 2t \qquad (3.20)$$

înlocuind pe 2t în relaţia (3.18) rezultă:

$$n \geq 2n-2t = 2n-n+1 = n+1 \qquad (3.21)$$

ceea ce conduce în mod evident la concluzia că nu pot fi două seturi de vectori cu un număr $m \geq n-t$ de vectori identici.

Pe baza teoremei 3.2 şi 3.3 s-a constatat că în matricea de incidenţă se obţin n-t vectori identici şi acest set de vectori este unic.

Pentru a constata dacă matricea de incidenţă MATRIX reflectă corect situaţia de diagnoză din sistem şi dacă pe baza acestei matrici se pot determina unităţile defecte respectiv cele funcţional corecte se enunţă următoarea teoremă:

Teorema 3.4: Într-un sistem S, cu $n \geq 2t+1$ unităţi, unde t este numărul maxim de defecte ce pot fi localizate simultan, matricea de incidenţă, obţinută cu algoritmul 3.2, va pune în evidenţă toate unităţile defecte din sistem.

Pentru a demonstra teoreme şi implicit valabilitatea algoritmului 3.2, se fac următoarele precizări:

- numărul de unităţi fără defecte sînt cel puţin n-t (punctul 4 din ipoteza 3.1).

- numărul de vectori r_i identici sînt cel puţin n-t (din

teorema 3,2).

- numărul seturilor de vectori ce conţin n-t vectori identici este unu (din teorema 3.3).

Rezultă că există un număr n-(n-t) vectori $\underline{r}_i$ care diferă de cei n-t vectori identici şi cum:

$$n-t > n-(n-t) = t \qquad (3.22)$$

rezultă că cei n-t vectori identici nu pot corespunde decît unităţilor corecte, iar cei t vectori $\underline{r}_i$ diferiţi vor corespunde unităţilor defecte.

Vectorul $\underline{r}_i$ rezultant constituie o reuniune a diagnozelor tuturor unităţilor u_i considerate corecte de către unitatea u_i. Dacă unitatea u_i este cu adevărat corectă rezultă că $\underline{r}_i$ reprezintă reuniunea diagnozelor a celor n-t unităţi sigur corecte, ceea ce face ca vectorul $\underline{r}_i$ să reprezinte diagnoza corectă a unităţii u_i asupra stărilor celorlalte u_j unităţi, pentru j=1,2...n. Mai mult cum sînt n-t unităţi corecte vor rezulta că toate aceste unităţi au vectori $\underline{r}_i$ identici conform teoremei 3.2.

Dacă unitatea u_i este defectă va rezulta că vectorul corespunzător $\underline{r}_i$ obţinut în urma reuniunii de diagnoze incorecte, reflectă o situaţie de diagnoză falsă. Astfel de diagnoze incorecte pot fi generate de t vectori $\underline{r}_i$.

Concluzia este că algoritmul 3.2 de obţinere a matricei de incidenţă generează cel puţin:

$$n-t \geqslant t+1 \qquad (3.23)$$

vectori $\underline{r}_i$ care reflectă în mod sigur situaţia de diagnoză corectă.

Pentru ca procedura de diagnoză să fie completă trebuie găsite n-t vectori identici. Procedura de comparare a vectorului $\underline{r}_i$ fiecare cu fiecare este relativ simplă dar necesită un timp de găsire nejustificat de mare, în cazul cînd numărul de unităţi este mare. Pentru simplificarea procedurii de căutare a unităţilor defecte se propune următorul algoritm:

Algoritmul 3.3:

1. Se iniţializează vectorul $\underline{SUM} = (v_j)$ cu zero, pentru $j = 1,2...n$.
2. Se iniţializează j=1.
3. Se determină $v_j = v_j + r_{ij}$, pentru i=1,2...n şi r_{ij}=1.

4. Dacă $v_j > t$;

atunci: indicele j indică unitatea u_j defectă
şi se trece la pasul 5.
altfel: unitatea u_j este corectă şi se
trece la pasul 5.

5. j = j+1

6. Dacă $j \leq n$:

atunci: se trece la pasul 3.
altfel: se trece la pasul 7.

7. STOP.

Pentru a demonstra că algoritmul 3.3 permite determinarea corectă a stării unităţilor se va enunţa următoarea teoremă:

Teorema 3.5: Intr-un sistem S, cu $n \geq 2t+1$ unităţi, în care numărul maxim de defecte simultane nu depăşeşte t unităţi, o unitate u_j este defectă dacă şi numai dacă valoarea $v_j > t$, unde v_j este un element al vectorului SUM obţinut prin algoritmul 3.3.

Demonstrarea teoremei se face pe baza consecinţelor ce rezultă din teoremele 3.2, 3.3 şi 3.4.

Se face constatarea că dacă cei t+1 = n-t vectori r_i identici (teorema 3.2) formează un set de vectori unic (teorema 3.3) care reprezintă diagnoza corectă (teorema 3.4) atunci elementii r_{ij}, a vectorului r_i, care sînt egali cu unu, vor indica unităţile j defecte. In matricea MATEST=(r_{ij}) vor exista deci, t coloane (j) pentru care numărul de "1" va fi cel puţin t+1. Dacă fiecărei coloane din matricea MATEST îi atasează un contor (v_j) care să crească cu fiecare de "1" găsit pe coloana respectivă, va rezulta că prin simpla comparare a elementului vectorului SUM = $(v_1, v_2 \ldots v_n)$ se poate determina unităţile defecte dacă:

$$v_j \geq t+1 > t \qquad (3.24)$$

respectiv unităţile funcţional corecte dacă:

$$v_j \leq t \qquad (3.25)$$

Utilizarea matricei de incidenţă la determinarea diagnozei într-un sistem elimină, ca şi la metode utilizării defectului implicat, necesitatea unui dispozitiv de diagnoză care să utilizeze o memorie redondantă, de mare capacitate. Totodată cele două metode sînt relativ simplu de implementat prin hard. Mai

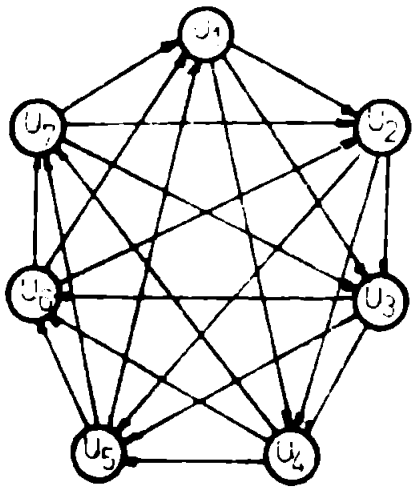
mult, în sistemele cu structură multimicroprocesor, algoritmii propuşi pot fi prelucraţi de unul sau mai multe microprocesoare ce funcţionează sigur corect. Implementarea prin hard sau soft a algoritmului propus depinde de modul şi gradul în care proiectantul sistemului doreşte să finalizeze structura autotestabilă. În capitolul 5 se propune o soluţie posibilă.

3.4. Implementarea algoritmilor de diagnoză.

În acest paragraf se vor prezenta rezultatele experimentale privind implementarea algoritmilor de diagnoză prin simularea unor structuri de sisteme autotestabile (SAT).

3.4.1. Implementarea metodelor de diagnoză prin simularea unor SAT.

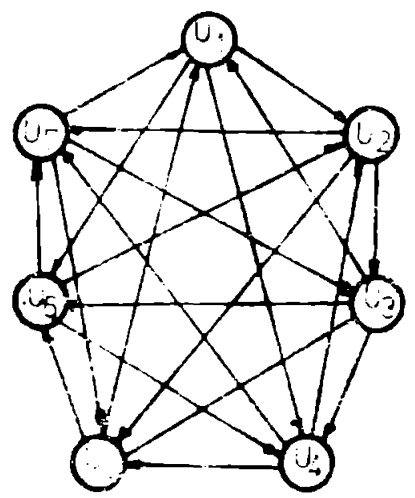
Pentru implementarea celor două metode de diagnoză se propun două structuri de SAT, care satisfac condiţiile de testabilitate stabilite în capitolul 2. Cele două SAT sînt reprezentate în fig. 3.3 şi respectiv 3.4, pentru cazul unor



MCONEX

	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	0	0	1	1	1	0	0
3	0	0	0	1	1	1	0
4	1	0	0	0	1	1	1
5	1	0	0	0	0	1	1
6	1	1	0	0	0	0	1
7	1	1	1	0	0	0	0

Fig. 3.3



MCONEX

	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0
2	0	0	1	0	1	0	1
3	1	0	0	1	0	1	0
4	0	1	0	0	1	0	1
5	1	0	1	0	0	1	0
6	0	1	0	1	0	0	1
7	1	0	1	0	1	0	0

Fig. 3.4

interconexiuni de test simetrice sau asimetrice împreună cu

matricea conexiunilor de test. Pentru exemplificarea modului de diagnoză a unui SAT, cele două metode de diagnoză; metoda de diagnoză prin utilizarea matricei de incidenţă (MDMI) şi metoda de diagnoză prin utilizarea defectelor implicate(MDDI) se prezintă în paralel. Fiecare din cele două metode de diagnoză sînt astfel reprezentate încît să se observe în mod sugestiv principalele etape parcurse pentru obţinerea rezultatelor finale; unităţile defecte, respectiv unităţile funcţional corecte.

Pentru cazul de diagnoză a unei SAT ce se poate descompune în şapte unităţi funcţionale, cu proprietăţile specificate; numărul maxim de unităţi defecte simultan ce sînt detectate, la un moment dat, nu pot depăşi valoarea t=3. Trebuie precizat că pentru anumite cazuri particulare există posibilitatea detestării şi a unui număr de defecte mai mare decît t unităţi.

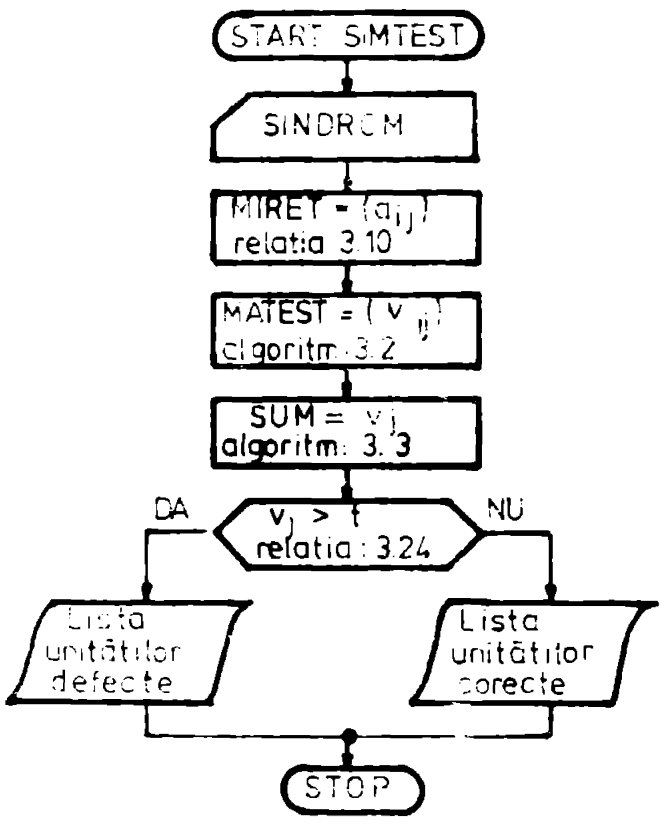


Fig.3.5

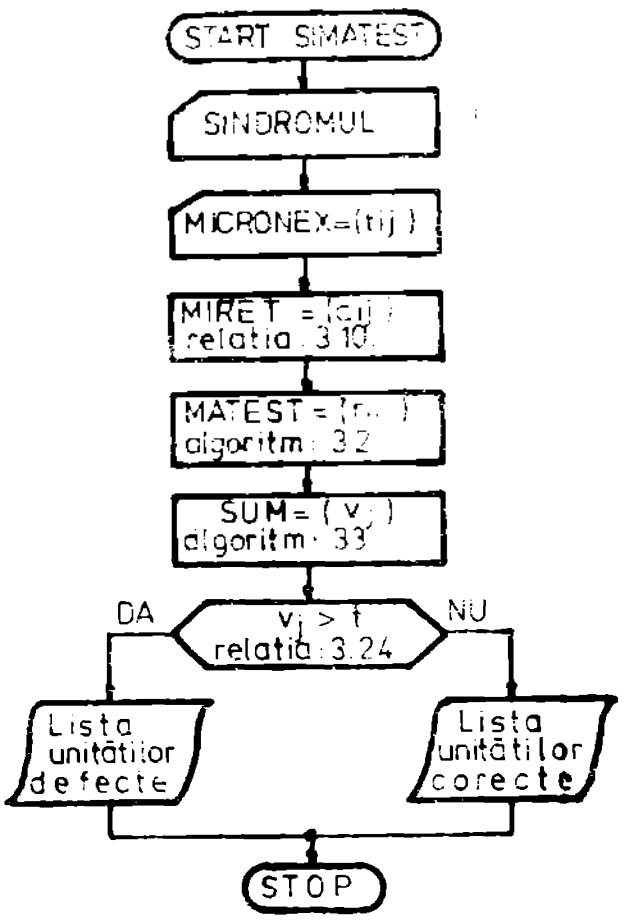


Fig.3.7

Mărimea t este numărul de unităţi maxim ce pot fi sigur detectate.

In cazul prezenţei a unor defecte permanente în fig.3.6 şi 3.8 se arată principalele etape ale celor metode de diagnoză, prezentate în paralel, pentru cazul unui SAT simetric, respectiv SAT

DIAGNOZA UNUI SISTEM CU STRUCTURA INTERCONEXIUNILOR DE TEST SIMETRICE IN CAZUL DEFECTELOR PERMANENTE

DIAGNOZA PRIN UTILIZAREA MATRICEI DE INCIDENTA

DIAGNOZA PRIN UTILIZAREA DEFECTELOR IMPLICATE

NUMARUL UNITATILOR DIN SISTEM: 7

NUMARUL MAXIM DE UNITATI DEFECTE DIN SISTEM: 3

UNITATILE DEFECTE : u_1, u_2, u_5

SINDROM POSIBIL:

$c_{12}\ c_{13}\ c_{14}\ c_{23}\ c_{24}\ c_{25}\ c_{34}\ c_{35}\ c_{36}\ c_{45}\ c_{46}\ c_{47}\ c_{56}\ c_{57}\ c_{54}\ c_{67}\ c_{61}\ c_{62}\ c_{71}\ c_{72}\ c_{73}$

X X X X X X 0 1 0 1 0 0 X X X 0 1 1 1 1 0

X = {0,1}

SINDROM OBTINUT:

1 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0

MIREZ = (a_{ij})

	1	2	3	4	5	6	7
①	·	1	0	·	·	·	·
②	·	·	0	1	0	·	·
3	·		·	0	1	0	·
4	·	·		·	1	0	0
⑤	0	·		·	·	1	1
6	1	1	·	·	·	·	0
7	1	1	0	·	·	·	·

MATRICE = (r_{ij})

$r_1^1 = a_1 = .\ 1\ 0\ 1\ .\ .\ .$

$r_1^2 = r_1^1\ a_2 = .\ 1\ 0\ 1\ 1\ 0\ .$

$r_1^3 = r_1^2\ a_3\ a_6 = 1\ 1\ 0\ 1\ 1\ 0\ 0$

$r_1^4 = r_1^3\ a_5\ a_6\ a_7 = 1\ 1\ 0\ 1\ 1\ 0\ 0$

$r_1^4 = r_1^3\ r_1 = 1\ 1\ 0\ 1\ 1\ 0\ 0$

	1	2	3	4	5	6	7
1	1	·	0	1	1	0	0
2	1	1	0	1	1	1	1
3	1	1	0	0	1	0	0
4	1	1	0	0	1	0	0
5	1	1	0	1	1	1	1
6	1	1	0	0	1	0	0
7	1		0	0	1	0	0

SUM= 7,7,0,3,7,2,2

UNITATI GASITE DEFECTE

u_1, u_2, u_5

1. $L(u_1)=\{u_2,u_4\}$ $L^{-1}(u_1)=\{u_6,u_7\}$
   $L(u_2)=\{u_4\}$ $L^{-1}(u_2)=\{u_1,u_6,u_7\}$
   $L(u_3)=\{u_5\}$ $L^{-1}(u_3)=\emptyset$
   $L(u_4)=\{u_5\}$ $L^{-1}(u_4)=\{u_1,u_2\}$
   $L(u_5)=\{u_6,u_7\}$ $L^{-1}(u_5)=\{u_3,u_4\}$
   $L(u_6)=\{u_1,u_2\}$ $L^{-1}(u_6)=\{u_5\}$
   $L(u_7)=\{u_1,u_2\}$ $L^{-1}(u_7)=\{u_5\}$
2. $D_1=\{u_1,u_2,u_4,u_5,u_6,u_7\}$
   $B_1=\{u_3\}$
3. $B=\{u_3\}$
   $D=L(u_3)=\{u_5\}$
4. $D_2=\{u_1,u_2,u_4\}$
   $B_2=\{u_6,u_7\}$
5. $h_2(6)=|L^{-1}(u_6)\cap B_2|=0$
   $h_2(7)=|L^{-1}(u_7)\cap B_2|=0$
6. $B=B\cup B_2=\{u_3,u_6,u_7\}$
   $D=D\cup L(u_6)\cup L(u_7)$
   $=\{u_1,u_2,u_5\}$
7. $D_3=\emptyset$
   $B_3=\{u_4\}$
8. $B=\{u_3,u_4,u_6,u_7\}$
9. $D=\{u_1,u_2,u_5\}$

UNITATI GASITE DEFECTE

u_1, u_2, u_5

Fig.3.6

DIAGNOZA UNUI SISTEM CU STRUCTURA INTERCONEXIUNILOR DE TEST ASIMETRICA IN CAZUL DEFECTELOR PERMANENTE

DIAGNOZA PRIN UTILIZAREA MATRICEI DE INCIDENTA

DIAGNOZA PRIN UTILIZAREA DEFECTELOR IMPLICATE

NUMARUL DE UNITATI DIN SISTEM : 7

NUMARUL MAXIM DE UNITATI DEFECTE DIN SISTEM : 3

UNITATI DEFECTE : 1, 2, 3

SINDROM:

a_{14} a_{15} a_{21} a_{23} a_{25} a_{27} a_{31} a_{34} a_{36} a_{42} a_{45} a_{47} a_{51} a_{53} a_{56} a_{62} a_{64} a_{67} a_{71} a_{72} a_{75}

X X X X X X 1 0 0 1 1 0 1 0 0 1 1 0

SINDROM OBTINUT:

0 1 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 1 0

MIRAT

	1	2	3	4	5	6	7
①		0		1		0	
②			0		1		1
③	1			1		0	
4		1			0		0
5			1			0	
6				1	0		0
7			1		0		

MATEST

	1	2	3	4	5	6	7
1	1	1	1	1	1	0	1
2	1	1	1	1	1	0	1
3	1	1	1	1	0	0	1
4	1	1	1	0	0	0	0
5	1	1	1	0	0	0	0
6	1	1	1	0	0	0	0
7	1	1	1	1	0	0	0

SUM = {7, 7, 7, 3, 2, 0, 2}

UNITATI GASITE DEFECTE

u_1, u_2, u_3

1. $L(u_1)=\{u_4\}$ $L^{-1}(u_1)=\{u_3,u_5,u_7\}$
   $L(u_2)=\{u_5,u_7\}$ $L^{-1}(u_2)=\{u_4,u_6\}$
   $L(u_3)=\{u_1,u_4\}$ $L^{-1}(u_3)=\{u_5,u_7\}$
   $L(u_4)=\{u_2\}$ $L^{-1}(u_4)=\{u_1,u_3\}$
   $L(u_5)=\{u_1,u_3\}$ $L^{-1}(u_5)=\{u_2\}$
   $L(u_6)=\{u_2\}$ $L^{-1}(u_6)=\emptyset$
   $L(u_7)=\{u_1,u_3\}$ $L^{-1}(u_7)=\{u_2\}$
2. $D_1=\{u_1,u_2,u_3,u_4,u_5,u_7\}$
   $B_1=\{u_6\}$
3. $B=\{u_6\}$
   $D=\{u_2\}$
4. $D_2=\{u_1,u_3,u_4\}$
   $B_2=\{u_5,u_7\}$
5. $h_2(5)=0$
   $h_2(7)=0$
6. $B=\{u_5,u_6,u_7\}$
   $D=\{u_1,u_2,u_3\}$
7. $D_3=\emptyset$
   $B_3=\{u_4\}$
8. $B=\{u_4,u_5,u_6,u_7\}$
   $D=\{u_1,u_2,u_3\}$

UNITATI GASITE DEFECTE

u_1, u_2, u_3

Fig.3.8

asimetric, împreună cu ordinograma de principiu (fig.3.6 şi fig.3.7) în cazul utilizării ADMI. Pentru ADDI ordinograma din fig.3.2 descrie algoritmul folosit.

În fig.3.6 se prezintă şi modul de obţinere a vectorului $\underline{r_1}$ a matricei MATEST.

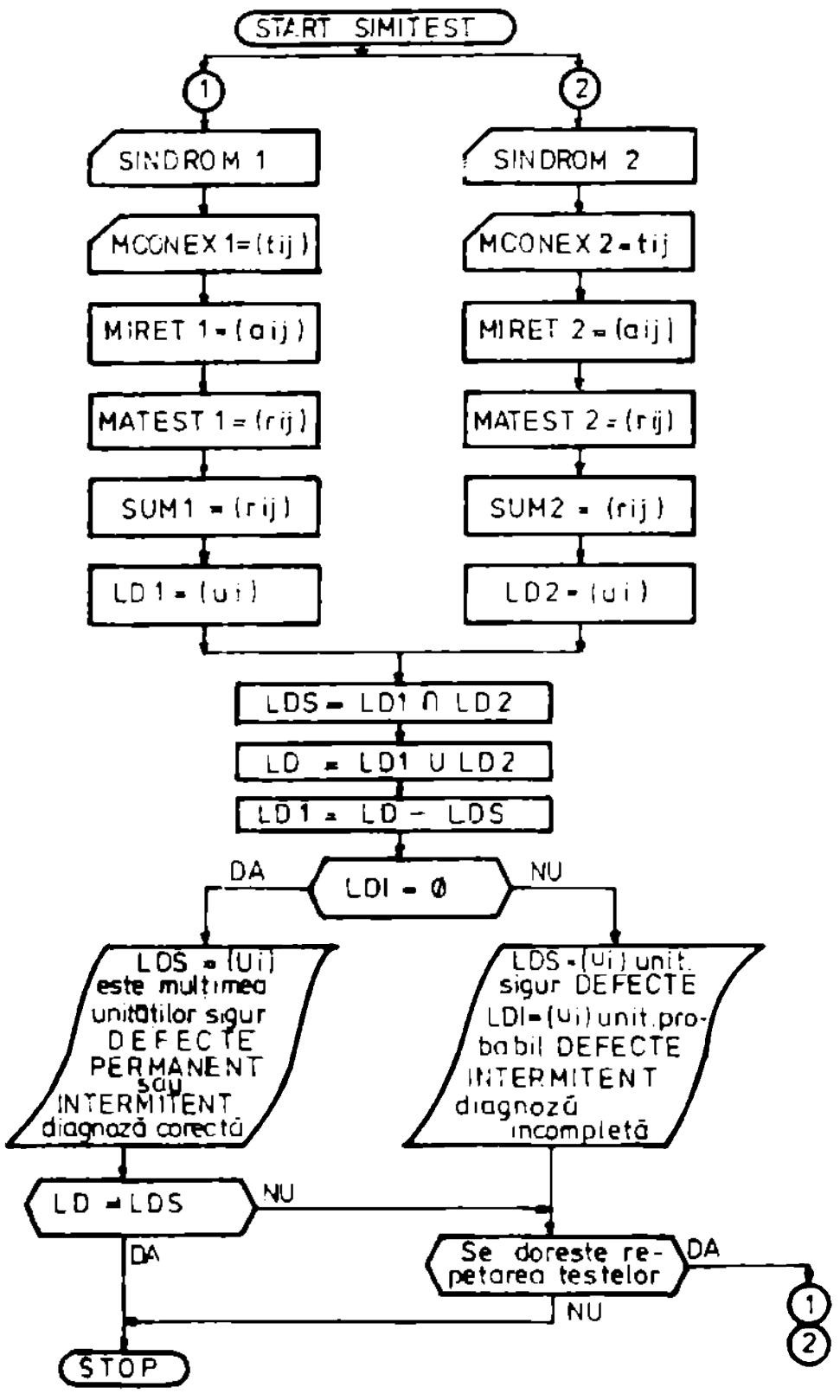


Fig.3.9

Simularea metodelor de diagnosă pentru SAT în situaţia că în sistem există defecte permanente şi intermitente se

descrie prin ordinograma din fig.3.9. Diagnoza se va executa în două faze. In prima fază pentru structura inițială a conexiunilor de test se obține o diagnoză parțială, care specifică unitățile găsite defecte în această fază. In faza următoare se realizează o schimbare a interconexiunilor de test, conform algoritmului 2.2 Setul de unități defecte obținut în urma diagnozei din etapa a doua se vor intersecta cu setul de unități defecte determinate în prima fază a diagnozei. Rezultatul intersecției determină unitățile sigur defecte (permanent sau intermitent). Prin eliminarea unităților sigur defecte pot să rămînă unități asupra cărora metodele de diagnoză nu pot da un răspuns ferm. Ceea ce se poate preciza este că din setul de unități rămase, sigur una sau mai multe unități sînt defecte intermitente, care nu au fost testate în mod corespunzător. In acest caz se obține o diagnoză incompletă dar nu incorectă (fig.3.10).

Etapa următoare ar consta fie în înlocuirea unităților sigur defecte cu unități funcțional corecte, fie repetarea unor rutine de test asupra unităților presupuse defecte.

Modul de rezolvare depinde de la caz la caz.

Metodele de diagnoză s-au verificat prin simularea unor SAT în FORTRAN.

Listingul celor trei rutine de simulare: SIMTEST, SIMATEST și SIMITEST, pentru diagnoza unor SAT simetrice, asimetrice, în cazul prezenței unor defecte permanente, respectiv diagnoza unor SAT în prezența unor defecte intermitente, sînt prezentate în programele PR 3.1; PR 3.2 și PR 3.3, împreună cu rezultatele obținute pentru un caz oarecare.

Programele sînt generale putîndu-se verifica metodele de diagnoză pentru un n oricît de mare.

In această etapă a cercetării nu ne putem pronunța asupra uneia dintre cele două metode, deoarece în funcție de sindromul obținut timpul de răspuns cel mai favorabil înclină spre o metodă sau alta.

DIAGNOZA UNUI SISTEM IN CAZUL UNOR DEFECTE PERMANENTE SI INTERMITENTE

DIAGNOZA PRIN UTILIZAREA MATRICEI DE INCIDENTA

DIAGNOZA PRIN UTILIZAREA DEFECTELOR IMPLICATE

UNITATI DEFECTE PERMANENTE : u_1, u_2

UNITATI DEFECTE INTERMITENT: u_3

SINDROM

c_{13}	c_{14}	c_{23}	c_{24}	c_{25}	c_{34}	c_{35}	c_{36}	c_{45}	c_{46}	c_{47}	c_{56}	c_{57}	c_{67}	c_{72}
1	0	1	0	1	0	1	0	0	0	0	0	0	1	0

MIRET

MIRET 1

	1	2	3	4	5	6	7
①	·	1	0	1	·	·	·
②	·	·	0	1	0	·	·
③	·	·	·	1	0	0	·
4	·	·	·	·	0	0	0
5	1	·	·	·	·	0	0
6	1	1	·	·	·	·	0
7	1	1	0	·	·	·	·

1. $L(u_1) = \{u_2, u_4\}$ $L^{-1}(u_1) = \{u_5, u_6, u_7\}$
   $L(u_2) = \{u_4\}$ $L^{-1}(u_2) = \{u_1\}$
   $L(u_3) = \{u_4\}$ $L^{-1}(u_3) = \emptyset$
   $L(u_4) = \emptyset$ $L^{-1}(u_4) = \{u_1, u_2, u_3\}$
   $L(u_5) = \{u_1\}$ $L^{-1}(u_5) = \emptyset$
   $L(u_6) = \{u_1, u_2\}$ $L^{-1}(u_6) = \emptyset$
   $L(u_7) = \{u_1, u_2\}$ $L^{-1}(u_7) = \emptyset$

MATEST

MATEST 1

	1	2	3	4	5	6	7
1	1	1	0	1	0	0	0
2	1	1	0	1	0	0	0
3	1	1	0	1	0	0	0
4	1	1	0	1	0	0	0
5	1	1	0	1	0	0	0
6	1	1	0	1	0	0	0
7	1	1	0	1	0	0	0

2. $D_1 = \{u_1, u_2, u_4\}$
   $B_1 = \{u_3, u_5, u_6, u_7\}$

3. $B = \{u_3, u_5, u_6, u_7\}$
   $D = \{u_1, u_2, u_4\}$

UNITATI GASITE DEFECTE IN PRIMA ETAPA:

u_1, u_2, u_4

SINCRON

$c_{16}c_{47}c_{43}c_{67}c_{43}c_{46}c_{73}c_{76}c_{72}c_{36}c_{32}c_{35}c_{62}c_{65}c_{61}c_{25}c_{21}c_{24}c_{51}c$ c

1 0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0

MIRET

MIRET 2

	4	7	3	6	2	5
3	1	0	1	·	·	·
4		0	·	0	·	·
7		·	0	0	1	·
3	·	·		1	0	1
6	·	·		·	1	0
2	1	1	·		·	0
5	1	0	0	·	1	·

MATEST

MATEST 2

	1	4	7	3	6	2	5
1	1	1	0	1	0	1	0
4	1	0	0	1	0	1	0
7	1	0	0	1	0	1	0
3	1	1	0	0	1	0	1
6	1	0	0	1	0	1	0
2	1	1	0	0	1	1	1
5	1	0	0	1	0	1	0

1. $L(u_1)=\{u_3,u_4\}$ $L^{-1}(u_1)=\{u_2,u_5,u_6\}$
   $L(u_4)=\{u_3\}$ $L^{-1}(u_4)=\{u_1,u_2\}$
   $L(u_7)=\{u_2\}$ $L^{-1}(u_7)=\emptyset$
   $L(u_3)=\{u_5,u_6\}$ $L^{-1}(u_3)=\{u_1,u_4\}$
   $L(u_6)=\{u_1,u_2\}$ $L^{-1}(u_6)=\{u_3\}$
   $L(u_2)=\{u_1,u_4\}$ $L^{-1}(u_2)=\{u_6,u_7\}$
   $L(u_5)=\{u_1\}$ $L^{-1}(u_5)=\{u_3\}$
2. $D_1=\{u_1,u_4,u_3,u_6,u_2,u_5\}$
   $B_1=\{u_7\}$
3. $B=\{u_7\}$
   $D=\{u_2\}$
4. $D_2=\{u_1,u_4,u_3,u_6,u_5\}$
   $B_2=\emptyset$
5. $h_2(u_1)=2$
   $h_2(u_4)=1$
   $h_2(u_3)=2$
   $h_2(u_6)=1$
   $h_2(u_5)=1$
6. $D_3=\{u_1,u_3\}$
   $B_3=\{u_4,u_6,u_5\}$
7. $h_3(u_4)=0$
   $h_3(u_6)=0$
   $h_3(u_5)=0$
8. $B=\{u_4,u_5,u_6,u_7\}$
   $D=\{u_1,u_2,u_3\}$

UNITATI GASITE DEFECTE IN ETAPA A DOUA:

u_1, u_2, u_3

$LD1=\{u_1,u_2,u_4\}$
$LD2=\{u_1,u_2,u_3\}$
$LD3=\{u_1,u_2\}$
$LD=\{u_1,u_2,u_3,u_4\}$
$LD1=\{u_3,u_4\}$

UNITATI SIGUR DEFECTE : $LDS=\{u_1,u_2\}$

DIAGNOZA INCOMPLECTA: nu se poate precisa starea unităţilor u_3 şi u_4, ceea ce implică testarea suplimentară a acestor unităţi.

Fig.3.10

CAPITOLUL 4

DIAGNOZA UNITĂŢILOR FUNCŢIONALE

4.1. Metode de diagnoză a unităţilor funcţionale

Plecînd de la structura unui sistem S, descompus în n unităţi funcţionale, care satisfac ipoteza 2.1, se poate obţine un sistem autotestabil, dacă interconexiunile de test respectă structura propusă în capitolul 2. În urma experimentului de test rezultă un sindrom. Pe baza sindromului, care exprimă situaţia de defect din sistem, se poate ajunge, cu ajutorul algoritmilor de diagnoză din capitolul 3, la situaţia cînd una sau mai multe unităţi, din sistem, sînt indicate ca defecte. Se pune problema în continuare de a găsi o metodă de testare a acestor unităţi funcţionale, pentru localizarea defectului. În principiu fiecare unitate funcţională este constituită să realizeze o anumită sarcină, pentru care a fost prevăzută în sistem, plus să aibe posibilitatea de a genera stimuli de test spre una sau mai multe unităţi funcţionale din sistem şi să poată evalua răspunsul unităţii testate. Practic o astfel de unitate funcţională este constituită dintr-o unitate de comandă prevăzută cu o unitate de memorie. În cadrul unei structuri multiprocesor, aceste unităţi pot fi realizate dintr-un microprocesor şi o memorie. Dintre toate blocurile unui sistem de calcul, cele care s-au bucurat de o atenţie sporită în domeniul testării au fost blocurile de memorie. Acestea prezintă o particularitate, care le fac mai uşor testabile, şi anume, regularitatea structurii interne [182, 210, 212] .

Unitatea de comandă dintr-un sistem numeric, indiferent cum este realizată, prezintă dezavantajul că diferă de la un sistem la altul, ceea ce impune găsirea unor metode particulare de testare. În plus datorită complexităţii acestora devine foarte dificil de abordat problematica testării dintr-un punct de vedere general. Problema găsirii unor metode general valabile pentru testarea unităţilor de comandă, s-au cel puţin pentru un număr cît mai mare de cazuri particulare, a consti-

tuit o preocupare majoră a proiectanţiilor de sisteme numerice, fără a se ajunge la un consens în acest domeniu.

Prin unitate de comandă vom înţelege orice bloc care generează o secvenţă de comenzi (microcomenzi), spre alte blocuri, necesare pentru executarea unui algoritm de funcţionare [15]. In principiu orice dispozitiv de comandă poate fi privit ca un circuit secvenţial. In literatură s-au desprins două concepţii privind testarea unităţiilor de comandă: testarea pe blocuri mici şi adoptarea unor tehnici de proiectare care să faciliteze testarea acestora prin metode de autoverificare [201].

Prima metodă constă în partajarea UC în subansamble care pot fi testate prin procedee adecvate circuitelor secvenţiale [30 , 155, 77, 212] . Divizarea în subansamble devine necesară în special în cazul reţelelor de circuite secvenţiale asincrone mari pentru care soluţiile generale şi practice, pentru testare, încă nu şi-au găsit rezolvarea [115]. Obţinerea experimentului de test pentru circuitele secvenţiale, prezintă la ora actuală încă o problemă complexă, iar metodele prezentate nu acoperă un domeniu general valabil. Mai mult metodele de generare automată a stimulilor de test folosesc algoritmi complexi, ce sînt greu de implementat pe calculator.

A doua metodă constă în adoptarea unor tehnici de proiectare care să faciliteze o testare mai eficientă a unităţiilor de comandă, prin adăugarea unor elemente hard suplimentare. In acest domeniu există o diversitate de soluţii care preconizează modificări la toate nivelele unui sistem. Metoda poate conduce la creşterea necesarului de componente pînă aproape la dublu [30]

4.2. <u>Testarea pe bloguri mici</u>

In principiu această metodă constă în elaborarea vectorilor stimuli de test pentru determinarea experimentului de test la scheme secvenţiale. Raportat la problema similară pentru scheme combinaţionale [4, 5, 23, 30, 65, 77, 212, 97], elaborarea vectorilor de stimuli de test pentru schemele secvenţiale prezintă o creştere în complexitate datorată în esenţă următoarelor aspecte:

- aducerea schemei într-o stare cunoscută, acceptată drept iniţială,

- creşterea numărului de teste prin multiplicare cu doi, corespunzător fiecărui element de memorare din cadrul schemei

secvenţiale.

- stările interne, ce reprezintă funcţiile de memorare, nu sînt observabile şi nici direct controlabile.

In timp ce pentru testarea unui defect în circuitele combinaţionale este suficient aplicarea unui singur test şi urmărirea rezultatului, la circuitele secvenţiale, pentru testarea unui defect este necesară o secvenţă de test şi o analiză corespunzătoare a acestora [74, 75, 91, 118, 131, 139, 155, 163, 100].

In fig.4.1. se reprezintă modelul lui Huffman pentru circuite secvenţiale sincrone, dacă semnalul de tact este prezent, respectiv asincrone, dacă semnalul de tact nu este ataşat schemei [58,77].

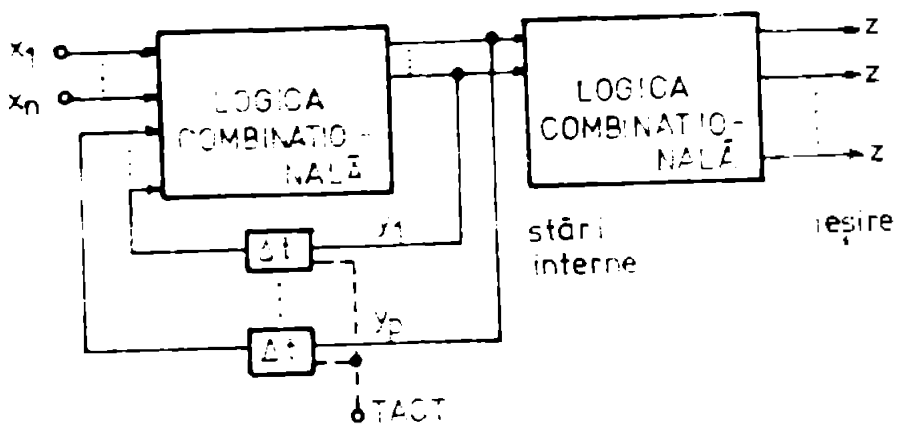


Fig.4.1

In general la testarea circuitelor secvenţiale asincrone se pot utiliza aceleaşi metode folosite la testarea circuitelor secvenţiale sincrone, cu observaţia că în acest caz apar o serie de probleme suplimentare care trebuiesc soluţionate în cazul testării circuitelor secvenţiale asincrone:

- apariţia hazardurilor statice şi dinamice;

- imposibilitatea repetării aceluiaşi semnal de intrare, deoarece circuitul este conceput să răspundă numai la schimbarea semnalelor de intrare.

- apariţia unor căi critice.

- posibilitatea apariţiei oscilaţiilor între două stări.

Pentru depăşirea acestor probleme, la realizarea experimentului de test se introduc unele restricţii suplimentare, faţă de testarea circuitelor secvenţiale sincrone, şi anume:

- tabelul de adevăr, al circuitului co ect, constituie unica legătură.

- defectele să nu fie ciclice.

- numărul de stări să nu crească ca rezultat al apariţiei unui defect.

Procedurile cunoscute în literatură abordează în manieră diferită soluţionarea aspectelor amintite, legate de testarea circuitelor secvenţiale. În acest sens procedurile de test se pot împărţi:

a) clasa procedurilor algoritmice

b) metode de identificare a maginilor.

c) metode de testare prin simulare.

4.2.1. Proceduri de testare algoritmică.

In principiu metodele de testare algoritmică constau în sensitivitatea căilor de acces a informaţiilor de la intrare spre ieşire şi transmiterea defectului din punctul de apariţie spre bornele de ieşire a schemei spre a putea fi identificat. Cele mai cunoscute metode de testare algoritmică sînt:

a) Metoda algoritmului D-extinsă

b) Metoda derivatelor temporale

c) Metoda tabelelor de adevăr.

4.2.1.1. Metoda algoritmului D-extinsă.

Este o metodă de sensitivizare a căilor multiple. Metoda a fost preluată de la circuitele combinaţionale şi extinsă la circuitele secvenţiale [58, 77, 212].

Testarea unui circuit secvenţial, format dintr-un circuit combinaţional şi o memorie de stări(Fig.4.1) se poate face prin deschiderea buclei de reacţie şi înlocuirea memoriei de stare printr-un circuit combinaţional la momentul t-1. Ieşirile circuitului combinaţional la momentul t-1 sînt considerate pseudoieşiri (P_s), care devin, pentru circuitul combinaţional următor la momentul t, pseudointrări (P_e).In fig.4.2 se arată un circuit secvenţial considerat ca o mulţime semifinită de celule combinaţionale.

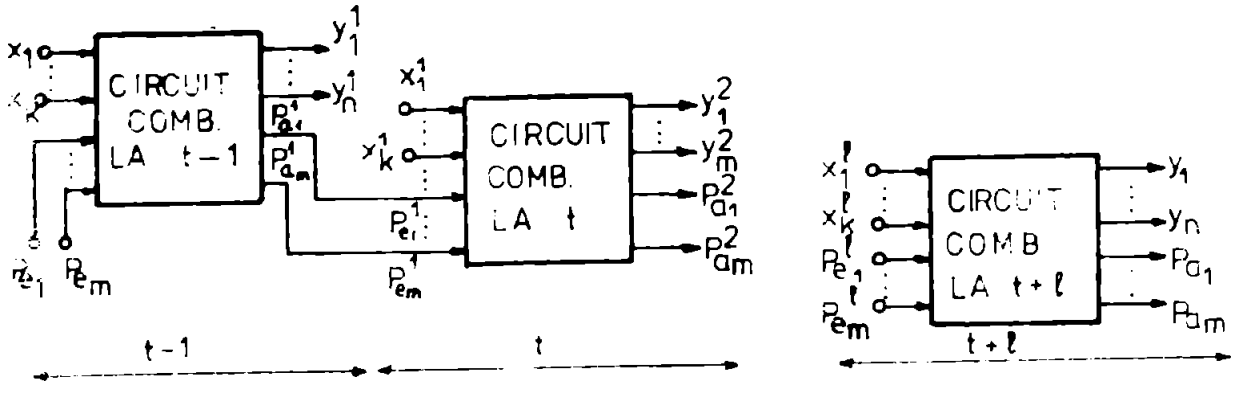


Fig.4.2

unde:

$$P_{a_i}^{t-1} = P_{e_i}^{t} \quad \text{pentru } i = 1,2...m \qquad (4.1)$$

Un defect singular în circuitul secvențial devine un defect multiplu în mulțimea celulelor combinaționale; cîte un defect singular în fiecare celulă combinațională.

Problema găsirii unei secvențe de test pentru un astfel de circuit este echivalentă cu testarea unui circuit combinațional cu defecte multiple. In acest caz se consideră ca intrări diferite aceleași intrări, dar la momente diferite de timp.

Pentru ℓ celule combinaționale va rezulta următoarea secvență de test obținută iterativ în timp [212].

$$T_\ell = (x_1...x_k)^{t-(\ell+1)} \quad (x_1...x_k)^{t-\ell} \quad ...(x_1...x_k)^{t} \qquad (4.2)$$

Vectorii $(x_1...x_k)$ la fiecare moment se obțin pe baza algoritmului D [77], care detectează un anumit defect considerat, în sensul că defectul se va propaga prin celulele combinaționale spre o ieșire observabilă.

Dacă nu se găsește nici o secvență de test pentru un defect considerat, se mărește numărul de celule combinaționale, luat inițial în considerare, mărind astfel și gradul defectului multiplu. Se reia determinarea secvenței de test pentru noua împărțire. Procesul se continuă iterativ pînă se găsesc secvențele de test care să acopere toate defectele.

Algoritmul D (Roth) astfel conceput poate fi aplicat atît circuitelor secvențiale sincrone cît și asincrone, unde

tectul se consideră ca variabilă logică la momentele stabilite. Pentru a elimina hazardurile, în urma aplicării seturilor de test obţinuţi prin deschiderea buclei, este indicat a se verifica stimuli de test prin simulare.

Dezavantajul procedurii constă în faptul că se poate aplica doar circuitelor secvenţiale de dimensiuni mici şi nu poate garanta lungimea minimă a secvenţelor de test.

O extindere a metodei algoritmului D se prezintă în [58] Metoda presupune reprezentarea elementelor secvenţiale sub forma unor elemente pseudocombinaţionale. Acestea pe lîngă intrările obişnuite mai posedă şi pseudointrări, constituite din stările iniţiale ale elementelor secvenţiale.

Prin aplicarea metodei de sensitivizare a căilor logice ale circuitului pseudocombinaţional se obţin printr-un procedeu specific metodei, secvenţe de test finale formate dintr-o secvenţă de iniţializare şi din secvenţa de propagare a defectului de-a lungul căilor logice.

Metoda a fost aplicată pe circuite secvenţiale sincrone integrate pe scară medie şi mică (MSI, SSI).

4.2.1.2. Metoda derivatelor temporale.

Metoda a fost concepută pentru scheme combinaţionale, fiind extinsă şi la scheme secvenţiale prin introducerea variabilei de timp în plus [58, 77, 153, 212].

In principiu această metodă pleacă de la considerentul că fiecare circuit poate fi descris printr-o ecuaţie booleană de forma:

$$y = f(x_1, \ldots x_k) \qquad (4.3)$$

In cazul testării, se impune stabilirea seturilor de test care pot detecta schimbări ale intrărilor la ieşiri. Acest lucru poate fi exprimat prin derivarea funcţiei în raport cu variabilele de intrare:

$$\frac{dy}{d\underline{x}} = f(\underline{x}) + f(\bar{\underline{x}}) \qquad (4.4)$$

Derivata unei funcţii, y după variabila x_k este "1" dacă la o modificare a lui x_k şi y se modifică, ceea ce se poate exprima cu relaţia:

$$\frac{dy}{dx_k} = f(x_k=0) \oplus f(x_k=1) \qquad (4.5)$$

Derivînd pe y funcţie de toate variabilele de intrare se obţine setul de teste:

$$T_s(\underline{x}) = \left\{(\underline{x},\underline{dx}) / \text{pentru } (dx_k \frac{dy}{dx_k} = 1)\right\} \qquad (4.6)$$

unde: dx_k este derivata lui x_k obţinută prin fixarea lui x_k=1 şi apoi x_k=0.

$\frac{dy}{dx_k}$ este derivata lui y funcţie de variabila x_k conform relaţiei (4.5).

In felul acesta se testează comportarea circuitului în raport cu fiecare variabilă de intrare în parte.

Metoda poate fi generalizată, în sensul că funcţia de ieşire poate să fie exprimată funcţie de variabilele intermediare, pentru determinarea defectelor pe liniile interne ale schemei [101] .

$$y = f\left\{\underline{Y}\left[\underline{k}\ (\underline{x})\right]\right\} \qquad (4.7)$$

unde: g şi k sînt variabile intermediare iar x variabila de intrare.

În acest caz derivata funcţiei poate fi exprimată:

$$\frac{dy}{d\underline{x}} = \frac{dy}{dg}\ \frac{dg}{d\underline{k}}\ \frac{d\underline{k}}{d\underline{x}} \qquad (4.8)$$

Testarea circuitului funcţie de variabilele intermediare se face asemănător ca pentru variabilele de intrare,fiind condiţionat testul de posibilitatea modificării acestor variabile intermediare.

Cu ajutorul derivatelor booleene, se pot testa atît circuite combinaţionale cît şi secvenţiale. Pentru circuite secvenţiale trebuie ţinut cont de variabila de timp. Acest lucru se realizează prin introducerea derivatelor temporale.

Relaţia (4.6) devine în acest caz:

$$T_s(\underline{x}) = \left\{(\underline{x},\underline{dx}) / \text{pentru } (dx_k^{(m)} \frac{dy}{dx_k} = 1)\right\} \qquad (4.9)$$

unde: $dx_k^{(m)}$ este modificarea lui x_k la momentul de timp m.

Pentru simplificarea procedurii de test este de dorit de a duce circuitul în starea iniţială.

Dezavantajul procedurii constă în faptul că nu se poate aplica circuitelor secvenţiale de dimensiuni mari.

4.2.1.3. Metoda tabelelor de adevăr.

Metoda tabelelor de adevăr [212] se bazează pe obținerea secvențelor de test plecînd de la tabelele de adevăr ale circuitelor elementare (porți, bistabile, etc.) din care este constituit circuitul secvențial. Diagnoza cuprinde două părți: Prima parte constă dintr-un procedeu de diagnoză, prin care se pun în evidență defectele din schemă și faza a doua constă dintr-un procedeu de transport pentru a conduce vectorul de test de la intrare la circuitul elementar de testat, respectiv răspunsul circuitului spre o ieșire primară.

Metoda tabelelor de adevăr se desfășoară iterativ începînd cu testarea acelor circuite care au ieșirile identice cu ieșirile primare. Se continuă cu circuitele care au ieșirile identice cu intrările circuitelor testate anterior. Procedura continuă pînă la găsirea unui set de test complect care să fie transparent de la intrare spre ieșire.

Cu această metodă nu se obțin, în general, seturi de test minime, deoarece vectorii de intrare necesari pentru stabilirea procedurii de diagnoză sînt aleși euristic.

Dezavantajul major al metodei constă în faptul că nu se poate aplica unor circuite secvențiale mari, iar setul de test nu este minim.

4.2.2. Metode de identificarea mașinilor.

Caracteristici metodei, spre deosebire de metodele algoritmice, este că se urmărește punerea în evidență a mașinii bune în raport cu toate mașinile defecte, cu același număr de stări, intrări și ieșiri. Secvențele de testare ce se obțin pot detecta și defecte multiple.

Cele mai reprezentative metode de identificarea mașinilor sînt: [58, 77] :

a) Metoda Poage-Mc Cluskcy
b) Metoda Kenmie
c) Metoda Hsieh
d) Metoda Ad-hoc.

4.2.2.1. Metoda Poage-Mc Cluskey

Caracteristica acestei metode este determinarea experimentului de test prin comparerea efectuată, pentru fiecare defect în

parte, a tabelelor de tranziţii. Una corespunzătoare funcţionării schemei fără defect, pentru maşina bună, şi cealaltă corespunzătoare schemei în care s-a inserat defectul; maşina defectă.

Procesul de comparare, în cadrul metodei Poage-Mc Cluskey, constă dintr-o operaţie de înmulţire a elementelor tabelelor de tranziţii. Se pleacă de la propoziţiile Poage, elaborate atît pentru ieşirile observabile cît şi pentru ieşirile de reacţie a schemei. Pe această bază se elaborează,într-o a doua etapă, tabelele de tranziţii, corespunzătoare maşinii bune, respectiv a maşinii defecte. In continuare, bazat pe regula de înmulţire, specifică procedurii, se obţine aşa-numitul tensl-produs al tabelelor de tranziţii.

Procedura consideră că se pleacă dintr-o stare cunoscută, admisă ca iniţială, în care schema a ajuns prin forţarea unor semnale de iniţializare asincrone.

Procedura se repetă pentru fiecare defect în parte în aşa fel încît un experiment de testare complect, pentru detecţia defectelor logice din schema secvenţială sincronă, constituie înlănţuirea experimentelor de testare elaborate pentru fiecare defect în parte. Intre fiecare experiment de test se intercalează activarea intrărilor asincrone, de aducere a schemei în starea iniţială.

Experimentul de testare obţinut conduce la solicitări foarte mari, atît prin prisma efortului implicat de generarea automată, cît şi prin prisma volumului mare de memorie, ceea ce face ca această metodă să nu fie eficientă pentru scheme secvenţiale practice.

4.2.2.2. Metoda Hennie

Punerea în evidenţă a maşinilor defecte în raport cu maşina bună se face prin căutarea în tabelul de tranziţii corespunzătoare maşinii bune. Metoda are avantajul că secvenţele de test sînt aplicate doar pe tabela de tranziţii a maşinii bune, spre deosebire de procedura Poage-Mc.Cluskey la care se operează concomitent cu două tabele de tranziţii, unul pentru maşina bună, iar al doilea pentru maşina defectă.

Soluţionarea experimentului de test, în cadrul metodei Hennie se face prin înlănţuirea a trei subsecvenţe de vec-

tori stimuli de test aplicați la intrările primare:

a) Subsecvențe de inițializare a schemei

b) Subsecvențe de identificare a stărilor interne

c) Subsecvența de verificare a tranzițiilor.

In vederea aducerii mașinii într-o stare inițială,printr-un procedeu de căutare pe arborele stărilor interne, se determină așa numita secvență de aducere ("homing-sequence") [77]

Pentru identificarea stărilor interne, se caută tot pe arborele de stări interne, modificată față de cea pentru determinarea stării inițiale. Se investighează faptul dacă mașina posedă sau nu o așa-numită secvență de distingere. Prin prisma acestei secvențe, mașinile secvențiale se împart în unele care posedă și altele care nu posedă secvență de distingere. Se menționează că experimentul de testare este mult simplificat pentru acele mașini secvențiale care posedă secvențe de distingere și care acoperă de fapt majoritatea acestor mașini.

In cazul în care mașina secvențială nu posedă secvențe de distingere, procedura Hennie implică investigări suplimentare, pe arborele stărilor interne asociat mașinii, în vederea elaborării unor secvențe de caracterizare, respectiv a secvențelor de localizare. Intercalarea acestora în cadrul experimentului de testare duce la mărirea numărului de vectori de stimuli de test, cu implicații atît în ceea ce privește elaborarea acestor vectori, a creșterii duratei de execuție a verificării, cît și în ceea ce privește spațiul de memorie.

O dată testate stările interne ale mașinii secvențiale se trece la secvența de verificare a tranzițiilor, în care se înlocuiesc secvențe care determină trecerea prin fiecare stare internă, cu fiecare dintre vectorii de intrare posibili.

Dezavantajul metodei constă în faptul că procedeul de investigare, pentru identificarea mașinii defecte este complex, iar experimentul de verificare nu ia în considerare circuitul real. In plus se cere o tabelă de tranziții a circuitului real, care este dificil de determinat pentru circuite complexe cu multe stări. Secvențele de test generate sînt lungi și deci pentru circuite secvențiale cu multe stări, această metodă este practic dificil de utilizat.

4.2.2.3. Metoda Hsieh

Procedura Hsieh aduce îmbunătăţiri procedurii Hennie pentru maşinile ce nu au secvenţă de distingere a stărilor. Ca şi procedura Hennie, experimentul de test este format din două subsecvenţe, excluzînd partea de iniţializare.

a) Subsecvenţa de validare, se aplică o secvenţă de intrare/ieşire şi se verifică dacă este validă pentru circuitul testat,verificîndu-se dacă se pot atinge toate stările.

b) Subsecvenţa de diagnosticare verifică toate tranziţiile maşinii folosind o secvenţă de intrare/ieşire.

Prin secvenţa de intrare/ieşire, asociată fiecărei stări se face distingerea dintre perechile de intrare/ieşire asociate cu stări diferite [77] .

4.2.2.4. Metoda Ad-hoc

Constă din visualizarea tuturor stărilor şi de urmă verificarea tuturor tranziţiilor, bazat în esenţă pe modificarea unui singur bit din vectorul de intrare, astfel încît problema analizei hazardurilor şi oscilaţiilor să poată fi neglijat. Importantă în acest context este procedura consacrată Seshu-Freeman [77].

In prezent nu există o procedură general valabilă de identificare a maşinilor, care să genereze un experiment de test eficient şi sistematic şi care să înlocuiască procedurile algoritmice de sensitivizarea unei căi.

4.2.3. Metode de testare prin simulare.

Simularea circuitelor numerice este o posibilă metodă de generare a secvenţelor de test, pentru diferite tipuri de scheme logice, inclusiv şi pentru circuite secvenţiale. Simularea logică poate fi folosită atît în etapa de proiectare, cît şi în etapa de generare, sau de validare a experimentului de test [15, 31, 36, 160, 212] .

Programele de simulare se pot împărţi în simulatoare bazate pe compilator şi simulatoare bazate pe tabele.

4.2.3.1. Simulator bazat pe compilator.

Aceste simulatoare constau în descrierea circuitului pe baza unui program sursă, într-un limbaj de programare de nivel înalt. Programul se prelucrează programul de simulare, numit

compilator, generează codul de simulare a fiecărui circuit elementar din montaj. Acest tip de simulare este eficace pentru scheme de mărime moderată (sute de părţi)[51, 77, 82, 115,175]

Dezavantajul acestui tip de simulator constă în faptul că nu poate folosi tehnica traseului selectiv. Analizorul secvenţial a lui Seshu[58, 77] este un exemplu tipic de simulator condus de compilator.

Analizorul secvenţial este format dintr-un set de programe capabil să simuleze simultan un circuit ce funcţionează corect şi mai multe circuite defecte. Numărul de circuite defecte simulate depinde de lungimea cuvîntului calculatorului pe care se face simularea [51, 82, 175] . De asemenea permite detectarea sau diagnoza defectelor.

Simulatorul poate avea două regimuri de utilizare. Fie pentru obţinerea dicţionarului de defecte, fie încorporat într-un program de generare a vectorilor de test.

Printre dezavantajele acestui simulator se pot enunţa:lungimea secvenţelor de test ce se obţine nu sînt minime ceea ce duce la creşterea timpului de testare. Circuitele secvenţiale trebuie presupuse într-o stare iniţială cunoscută chiar şi în condiţii de defect. Nu poate detecta toate defectele posibile.

4.2.3.2. Simulator bazat pe tabele.

La acest tip de simulator,descrierea circuitelor este memorată sub formă de tabele de adevăr. Programul de simulare foloseşte aceste tabele ca date. Aceste simulatoare se pot utiliza pentru descrierea unor circuite mari (zeci de mii de porţi) cu utilizarea eficientă a memoriei calculatorului.

Deşi necesită mai mult timp pentru găsirea datelor folosirea metodei traseului selectiv compensează dezavantajul [14, 30 51, 61, 77, 175].

Avantajele majore ale acestui simulator sînt:

- Posibilitatea folosirii tehnicii traseului selectiv. La un moment dat se tratează numai modulele pentru care cel puţin una din intrări şi-a modificat nivelul logic. Deoarece numai o parte din circuitele unei reţele logice sînt active simultan procedeul are ca rezultat o reducere importantă a timpului de simulare.

- Buclele de reacţie sînt introduse în mod natural fără

specificarea explicită şi iniţializarea obligatorie, cum este necesar la simularea prin compilare.

- Există o mare suplete în modificarea ulterioară a programelor. Introducerea unui nou circuit primitiv este simplă şi anume este suficient să se modifice instrucţia de apelare a circuitului şi să se introducă subprogramul corespunzător ce simulează noul tip de modul.

- Marea generalitate a metodei permite tratarea de circuite logice de tipuri variate. Blocurile logice mari sînt tratate mai eficient deoarece memoria centrală este mai bine gestionată.

- Inserarea defectelor se face uşor prin metoda deductivă [14, 30, 61, 203].

- Defectele se determină printr-o singură trecere prin program, spre deosebire de simulatorul prin compilare care necesită cîte o trecere prin program pentru fiecare defect.

Ambele tipuri de simulatoare necesită o analiză la nivel de poartă, deşi uneori mai multe porţi sînt tratate ca o unitate. Programul de simulare actualizează valorile logice de la ieşirile circuitelor la intervale succesive de timp fixate prin program.

Dezavantajul simulării la nivel de poartă constă în faptul că fiecare modul primar trebuie descompus în circuite elementare, ceea ce duce la creşterea numărului de circuite şi deci a timpului de simulare. În plus metoda nu poate fi aplicată unui circuit secvenţial cu stări iniţiale necunoscute, deoarece în acest caz listele de defecte sînt nedeterminate.

În cazul simulării la nivel funcţional, fiecare circuit este tratat ca o "cutie neagră", care prezintă anumite valori la ieşire funcţie de semnalele de la intrare.

În circuitele secvenţiale mari este avantajos să se facă o simulare, pentru o anumită parte din schemă, la nivel de porţi, respectiv pentru alte părţi la nivel funcţional. Defectele sînt în acest caz detectate doar în partea simulată la nivel de porţi.

Deşi simularea funcţională este relativ simplă de conceput, există o dificultate majoră în aplicare, şi anume lipsa unei metode satisfăcătoare de descriere a circuitelor care trebuiesc simulate la nivel funcţional. Metoda de simulare

funcţională este, în prezent, restrictivă la circuite care posedă o funcţie caracteristică bine precizată cum sînt: registrele de deplasare, numărătoarele, decodificatoarele, etc. Simularea funcţională este obţinută prin evaluarea unei funcţii ceea ce produce în practică avantajul unei descrieri mai concise, un timp de execuţie mai redus şi un spaţiu de memorie mai mic. În schimb precizia simulării este în general mai redusă, deoarece nu se ia în seamă corect propagarea în timp a anumitor defecte.

4.3. Tehnici de proiectare

Îmbunătăţirea proprietăţii de testabilitate a unui sistem implică în mod automat găsirea unor soluţii corespunzătoare pentru localizarea şi detecţia defectelor pînă la nivel de componentă. Acest lucru se poate realiza fie prin metode de detecţie şi localizare a defectelor, prezentate mai sus pentru circuite secvenţiale, fie prin adoptarea unor tehnici de proiectare corespunzătoare, care să permită o testare mai simplă.

În prezent nu există o strategie unică şi nici o metodă universal acceptată în domeniul tehnicilor de proiectare pentru îmbunătăţirea testabilităţii. În literatura de specialitate sînt abordate diferite concepte, unele chiar contradictorii privind modalitatea de îmbunătăţire a testabilităţii unui sistem numeric. În acest sens de multe ori se pune problema dacă se justifică efortul de proiectare şi preţul mai ridicat al unor sisteme cu proprietăţi de diagnoză îmbunătăţite, deoarece testarea se face la intervale relativ mari, în comparaţie cu gradul de utilizare. Ceea ce conduce la ideia că costul testării reprezintă doar o mică parte din costul întregului sistem. În [115] se arată, pe baza unor analize referitoare la unele echipamente proiectate fără a se ţine cont de condiţiile de testare, respectiv proiectarea şi realizarea unor echipamente identice din punct de vedere al performanţelor, la care s-au impus unele condiţii de testare că modul de evoluţie a costului de fabricaţie este mai mare la prima categorie (fig.4.3). Mai mult aceste costuri cresc exponenţial cu creşterea complexităţii sistemului. În costurile de fabricaţie s-au inclus şi costul de generare a stimulilor de test, de realizare a echipamentelor de test, timpii de punere la punct al echipamentului,etc.

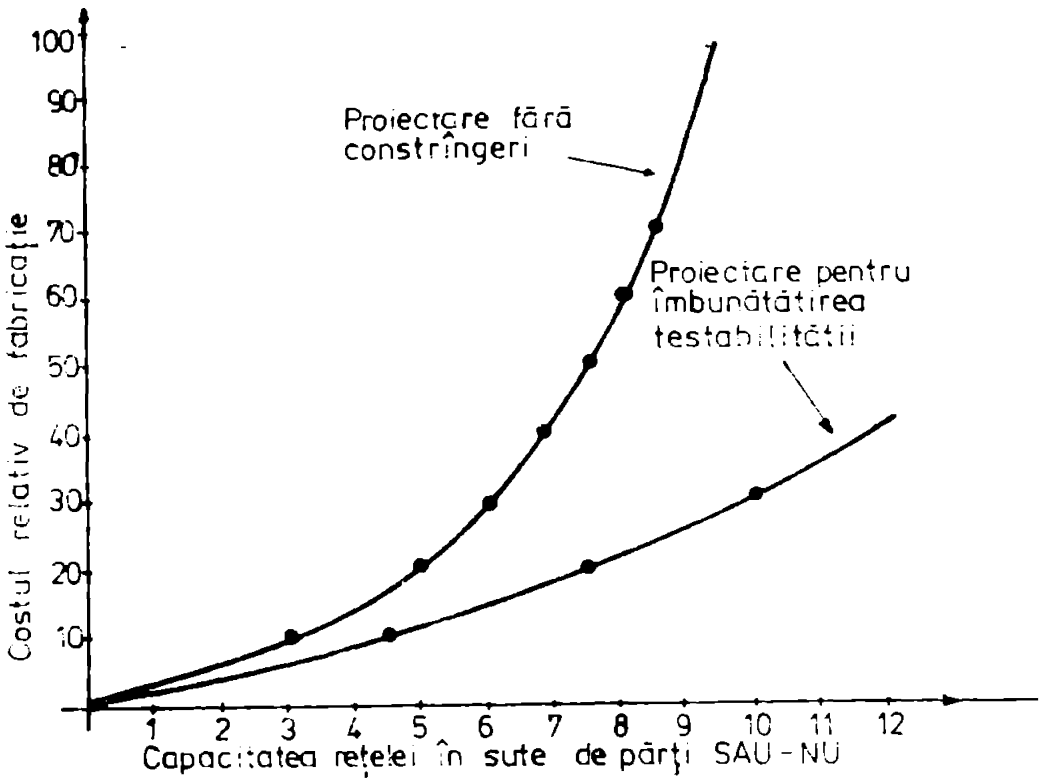


Fig.4.3

Literatura de specialitate este foarte bogată în lucrări ce tratează problematica îmbunătățirii testabilității prin reproiectarea circuitelor în mod corespunzător. Din acest motiv s-a împărțit problematica tratată în literatură pe trei nivele structurale:

a) Tehnici de proiectare la nivel de circuite integrate.

b) Tehnici de proiectare la nivel de blocuri.

c) Tehnici de proiectare la nivel de sistem.

Analiza tehnicilor de proiectare (TP) pe baza celor trei nivele se impune datorită unor particularități ce le fac specifice unor scheme de anumită complexitate. Sînt și unele TP care pot cuprinde mai multe nivele. De exemplu, în prezent nivelul de integrare este atît de mare încît unele tehnici folosite la nivel de blocuri pot fi utilizate și la nivel de circuite integrate.

4.3.1. TP la nivel de circuite integrate.

Prin creșterea gradului de integrare s-a ajuns la concluzia că este posibil și justificabil ca îmbunătățirea proprietății de testabilitate să pleacă de la nivelul cel mai de jos, adică de la circuitul integrat [48, 50, 89, 128, 134, 141, 162, 166].

Se consideră faptul că în prezent un circuit integrat de tip LSI sau VLSI conține un număr atît de mare de componente integrate, încît oricare metodă de diagnoză a defectelor din interiorul unui CI sînt lipsite de orice suport practic, datorită numărului redus de ieșiri observabile a numărului mare de componente electronice integrate și lipsei unei metode de diagnoză general, acceptată pentru un număr mare de tipuri de module. Mai mult, la nivel de utilizator interesează doar condiția dacă un circuit integrat este funcțional sau nu, nefiind necesară o localizare a defectului.

Prin introducerea în interiorul unui CI a unor tehnici de testare, prin reproiectarea circuitului, se urmărește de fapt ca testarea exterioară să nu mai fie necesară sau să se simplifice considerabil. Pentru eliminarea totală a testului din exterior, modulul integrat, trebuie să conțină în interior atît generatorul de stimuli de test cît și circuitele de evaluare a răspunsului, pe lîngă funcția normală de realizat.

In [40, 233] se face o evaluare a metodelor de proiectare a testului în interiorul capsulei (BIT-Built-in Test) pe baza mai multor criterii: performanță, cost, dificultate de proiectare. In ceea ce privește creșterea complexității unui circuit integrat se apreciază că la ora actuală aceasta poate fi cu circa 60 % mai mare decît cea față de un circuit normal. Totodată se apreciază că prin creșterea gradului de integrare complexitatea poate scădea la 37 % ce reprezintă o valoare acceptabilă.

O clasificare a testării unui circuit integrat este dată în (fig.4.4) [233].

Testarea externă a unui CI se poate realiza fie cu echipamente de testare automate, sau prin metode de diagnoză care generează stimuli de test specifici CI, iar răspunsul este comparat cu răspunsul unui circuit etalon. Se mai poate realiza o testare externă prin posibilități de proiectare la nivel de bloc care să asigure transferul vectorilor stimulilor de test simplu, spre circuitul testat, iar evaluarea răspunsurilor să fie realizată de circuitele specializate din cadrul sistemului, sau blocuri reconfigurate corespunzător în faza de testare. Microdiagnoza și testele funcționale, rezidente pe o memorie fixă sau suport extern (disc, bandă), constituie metode frecvent utilizate.

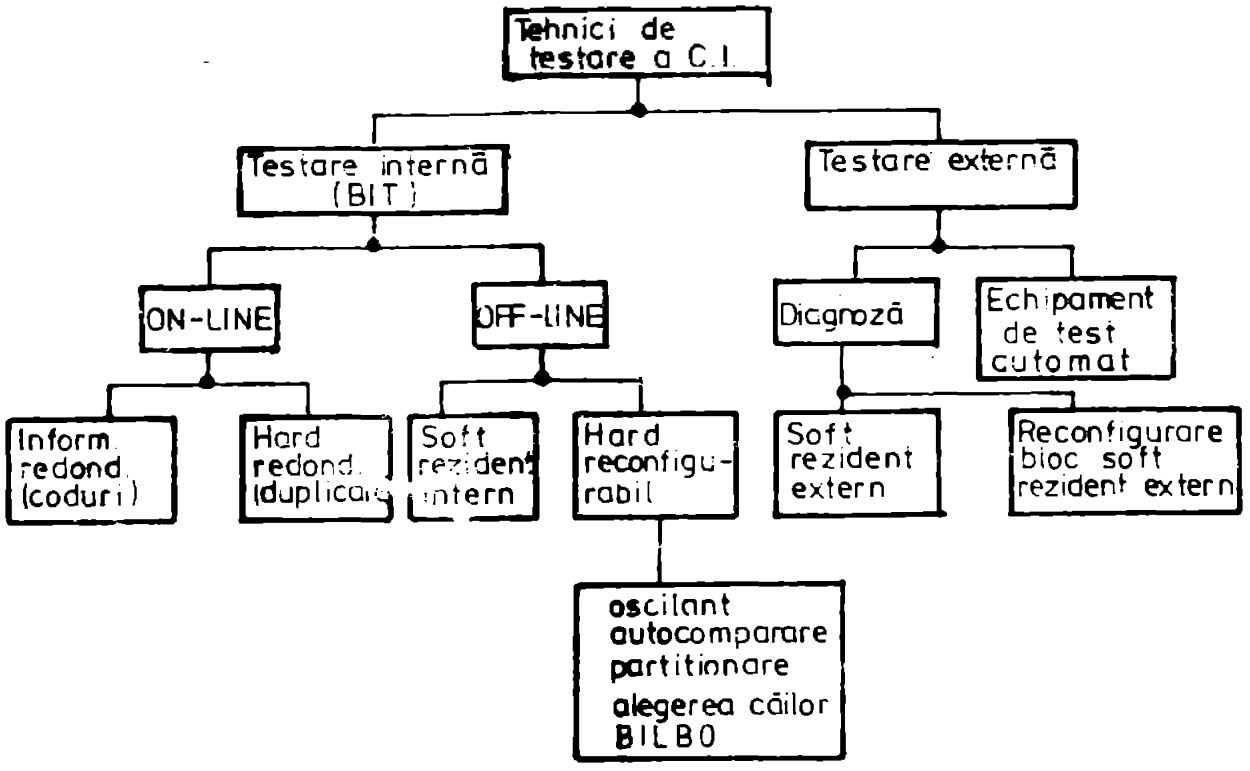


Fig.4.4

Tehnicile de testare internă pot fi cele care permit verificarea funcționării CI în timpul operării normale a acestuia (ON-LINE) respectiv în timpul unei faze speciale de testare (OFF-LINE).

Tehnica de testare internă ON-LINE implică fie o redondanță informațională, prin: utilizarea verificării bitului de paritate, folosirea unor coduri detectoare și corectoare de erori (coduri redondant ciclice, CRC, codul Hamming). La nivelul hardului redondanta se va manifesta prin dublicarea unor părți din circuit și adăugarea unor comparatoare de verificare.

Referitor la tehnica de testare internă OFF-LINE, aceasta implică executarea unor teste atunci cînd sistemul nu este operand, ceea ce permite o autotestare la nivel de circuit integrat. Ca rezultat al autotestului se pot obține: sindromul circuitului, semnătura circuitului, numărul de tranziții executat de circuit. Aceste rezultate se vor compara cu valorile de bună funcționare a modulului, rezultînd un semnal de bună funcționare (sau nu) a circuitului. La nivel de hard circuitul se va reconfigura corespunzător în faza de testare, atunci cînd funcționarea normală încetează.

Cele mai cunoscute tehnici de reconfigurare a circuitului sînt:

- Autooscilant: circuitul se reconfigurează de aşa manieră încît în faza de testare să oscileze dacă este bun şi nu oscilează dacă circuitul este defect. Prin această tehnică reacţia este uşor de realizat, complexitatea circuitului creşte nesemnificativ, se pot verifica circuitele la frecvenţe maximă de lucru, nu sînt necesari stimuli de test [233].

- Autocomparare: circuitul se divide de aşa manieră încît fiecare parte obţinută să producă funcţii elementare identice ce se pot compara. Metoda nu poate fi aplicată la circuite ce nu se pot fi partajate în blocuri identice, comparatorul este conceput să se autoverifice singur [63, 115, 153, 189, 233].

- Partiţionare: circuitul se divide în blocuri, iar stimulii de test pot ajunge la diferitele părţi prin intermediul unui multiplexoare. Metoda de reconfigurarea circulaţiei informaţiei prin intermediul multiplexoarelor se poate aplica la orice circuit, testarea se face rapid, iar reconfigurarea este uşor de realizat [40, 63, 115].

- Alegerea căilor: circuitul implică accesul în diferite puncte cheie, a intrării unui registru de deplasare, care culege semnătura punctului, atunci cînd la intrare se aplică anumiţi stimuli de test. Registrul de deplasare se poate verifica şi el. Metoda este generală, dar necesită un timp lung de testare datorită naturii seriale a testului [40, 115, 233].

BILBO (Built-in logic block observer): în circuit se utilizează două registre de deplasare, unul pentru generarea stimulilor de test şi al doilea pentru a obţine semnătura. Între cei doi regiştrii se află circuitul de testat. Metoda implică un număr mare de componente suplimentare, un timp lung de testare, un număr mare de linii de date interne [40, 115, 233].

În tabelul 4.1 se prezintă principalele caracteristici ale celor cinci metode de proiectare a testelor din interiorul unui circuit integrat.

Dintre metodele prezentate numai metoda BILBO asigură o autotestare internă a circuitului. Celelalte metode permit o simplificare a testării. Metoda autooscilantă permite o testare simplă şi eficientă cu un număr redus de componente suplimentare.

Tabelul 4.1

Nr. crt.	Caracteristici	Autoosc.	Autocomp.	Partition.	Alegere c.	BILBO
1	generarea stimuli de test	int.	ext.	ext.	ext.	int.
2	evaluarea circ.	ext.	int.	ext.	ext.	int.
2	nr.de componente suplimentare	5%	20%	30%	60%	80%
4	acoperirea defectului	100%	100%	100%	100%	100%
5	număr de teste	mic	mediu	mic	mic	mare

Nu ne putem pronunța asupra unei tehnici, avîndu-se în vedere că fiecare metodă se pretează pentru anumit tip de aplicație.

In [162] se apreciază că o reproiectare a circuitelor din componența sistemelor IBM360 care să permită o autoverificare la acest nivel ar duce la o creștere a întregului sistem numai cu 6,5 %, fără o pierdere de viteză, în comparație cu performanțele normale ale lui IBM360.

Tehnica ON-LINE comportă o creștere și mai mare de componente decît tehnica OFF-LINE. Astfel pentru un microprocesor, s-au o memorie, numărul de componente suplimentare poate crește cu 73% în cazul utilizării metodei de verificare a parității, cu 94% în cazul folosirii unor coduri K/n și cu 105% prin dublarea unor blocuri sau căi de acces [48, 50].

Indiferent de tehnica de proiectare a CI se consideră că prin creșterea gradului de integrare se poate rezolva în mod fundamental problema diagnozei prin plasarea responsabilității detectării defectelor în interiorul circuitelor VLSI [162].

4.3.2. TP la nivel de blocuri.

Deoarece generarea automată a stimulilor de test pentru o rețea secvențială nestructurată este o problemă care nu are practic rezolvare, în momentul de față [115], iar circuitele integrate cu posibilități de testare sînt realizate doar la nivel de experiment, sau sînt produse de firmă, necomerciale, se impune adoptarea unor tehnici de proiectare pentru îmbunătățirea testabilității la nivel de bloc funcțional.

Metode posibile de testare a blocurilor funcționale se prezintă în fig.4.5.

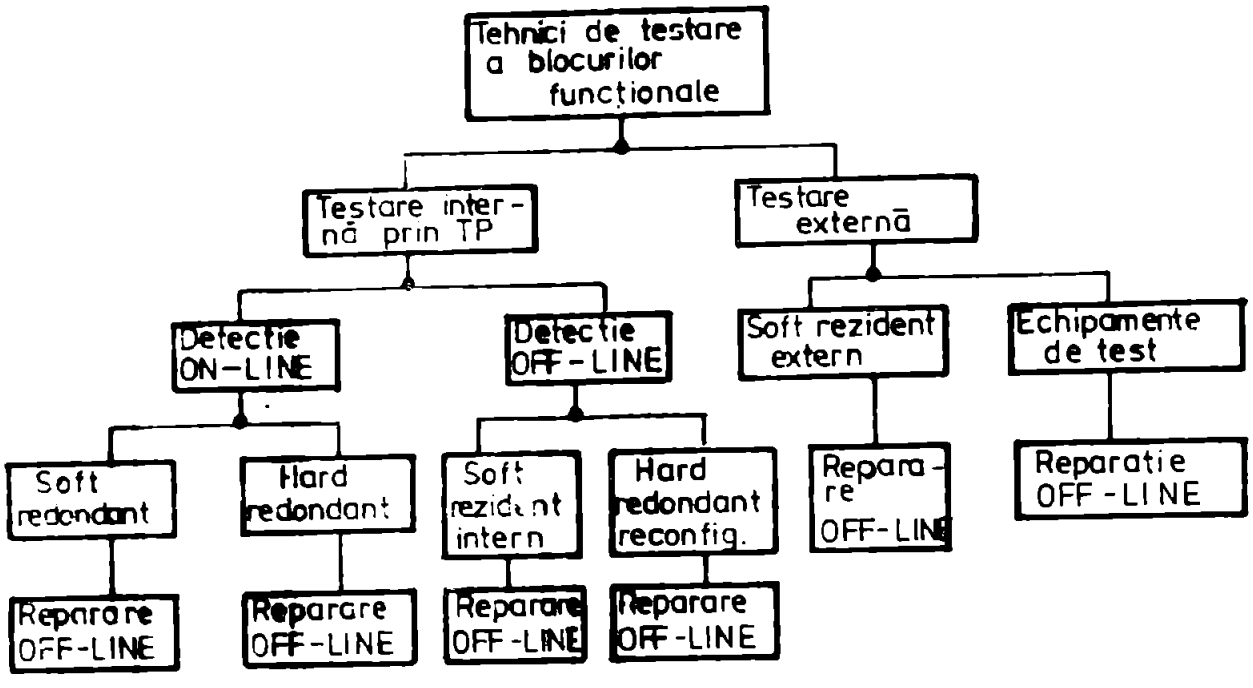


Fig.4.5

Testarea din exterior a unui bloc funcțional cu ajutorul echipamentelor de testat, necesită ca acestea să aibe posibilitatea de a genera stimuli de test și de a evalua răspunsul unității. Generarea stimulilor de test se obține cu ajutorul unor algoritmi și metode pentru scheme combinaționale respectiv secvențiale. Pentru testarea din exterior mai pot fi prevăzute programe de test rezidente pe un suport extern, care sînt rulate în faza de testare. Programele de tip "service" pot fi incluse în această categorie.

Tehnicile de proiectare care permit o testare din interiorul blocului depind de modul cum se realizează detecția și localizarea defectelor. În cazul că detecția se obține în paralel cu funcționarea normală a circuitului atunci se obține o detecție ON-LINE, iar dacă există o fază distinctă de testare, în cadrul funcționării blocului, atunci detecția este OFF-LINE.

Detecția ON-LINE implică fie un soft redondant, fie un hard redondant. Utilizarea unui soft redondant conduce la o redondanță informațională care se realizează prin coduri detectoare și corectoare de erori [2, 11, 12, 44, 48, 59] , verificarea bitului de paritate, dublarea executării unor operații, sau comenzi. Utilizarea unui hard redondant impune duble-

rea unor părţi componente din blocul funcţional, dublarea unor linii de comandă şi date, folosirea unor scheme de mascare a erorilor cum ar fi logica majoritară, cuadrupă, radiară, complementară [77]. Pentru localizarea defectelor se pot implementa programe de teste funcţionale on-line,[212, 144, 146, 147, 157, 179].

TP pentru detecţia OFF-LINE implică din punct de vedere hard o redondanţă care să permită o testare simplă a blocului. In literatura de specialitate se prezintă multe soluţii bune dintre care s-au impus următoarele: [68, 92, 212, 118, 124, 133, 143, 145, 153, 185,].

- Circuite secvenţiale cu secvenţe distincte: s-a văzut că realizarea unui experiment de verificare este considerabil simplificată dacă maşina are secvenţe distincte [77, 115, 190].

- Circuite combinaţionale cu localizarea defectelor:circuitele combinaţionale se realizează de aşa manieră încît să permită o localizare simplă a defectelor. Metoda constă în principiu din realizarea acestor circuite cu ajutorul funcţiei SAU-EXCLUSIV[77, 212].

- Metoda duală: realizarea unor module cu funcţie dublă; funcţionare normală şi funcţionare pentru testare. Prin folosirea unor astfel de module duale se pot realiza scheme foarte complexe [64].

- Utilizarea unor puncte de test suplimentare pentru a uşura testabilitatea blocului funcţional [74, 75, 87, 115, 212]

- Circuite de rang dublu şi multiplu: pentru a uşura diagnoza la circuitele secvenţiale asincrone se preconizează introducerea unor circuite suplimentare care să deschidă bucla de reacţie în timpul testării ceea ce reduce problema generării testului la *nivelul* unui circuit combinaţional [77].

- Multiplexarea căilor de acces a informaţiilor de test. Prin introducerea unor multiplexoare comandate din exterior se poate trimite stimuli de test direct la partea ce se doreşte a fi testată. Culegerea rezultatelor de la ieşirea părţii testate se face cu alt multiplexor. Prin introducerea multiplexoarelor de intrare şi ieşire se simplifică considerabil procedura de test şi procedura de transport a testului [212].

- Utilizarea schemelor de tip BILBO la nivel de bloc [212]

- Autotestarea: dacă blocul funcţional permite descompunerea în unităţi funcţionale conform ipotezei 2.1. Metoda permite localizarea defectelor la nivel de unitate.

TP pentru detecţia OFF-LINE cu soft rezident în interiorul blocului necesită utilizarea unor programe de diagnoză sau microdiagnoză. De obicei se combină metodele hard, care asigură o structură a blocului mai simplu de testat, cu metodele de diagnoză specifice blocului.

Repararea componentei defecte se face din exterior prin oprirea sistemului. In cazul tehnicilor de mascare a defectelor fără posibilitate de detecţie a acestora, repararea se face după defectarea întregului asamblu de mascare a erorii.

4.3.3. TP la nivel de sistem.

La nivel de sistem apariţia unui defect poate crea probleme de natură foarte diferite, care sînt dificil de prevăzut în faza de proiectare. Pentru prevenirea unor efecte nedorite au fost concepute două strategii de realizare a sistemelor numerice: sisteme de calcul intolerante la defecte şi sisteme tolerante la defecte.

Referitor la sistemele intolerante la defecte se face precizarea că pentru a creşte coeficientul de disponibilitate se impune realizarea acestor sisteme cu componente foarte fiabile [229].

Sistemele de calcul tolerante la defecte prezintă particularitatea că pot executa un algoritm chiar şi în prezenţa unui defect. Pentru a evita efectul defectului se foloseşte o redondanţă adecvată la nivel de sistem. Redondanta poate fi temporală, prin repetarea execuţiei unor operaţii, sau fizică printr-o redondanţă hard şi soft.

Arhitectura sistemlor care utilizează metode de proiectare adecvate pentru a face o testare mai simplă şi o funcţionare normală, chiar şi în prezenţa defectelor, se reprezintă în fig.4.6 [127, 133, 198, 225, 226, 231].

Tehnica de tolerare a defectelor se poate diviza în trei clase majore:

1. Sistemele cu detecţia defectelor: Astfel de sisteme sînt de obicei netolerante la defecte, fiind mai utilizate în

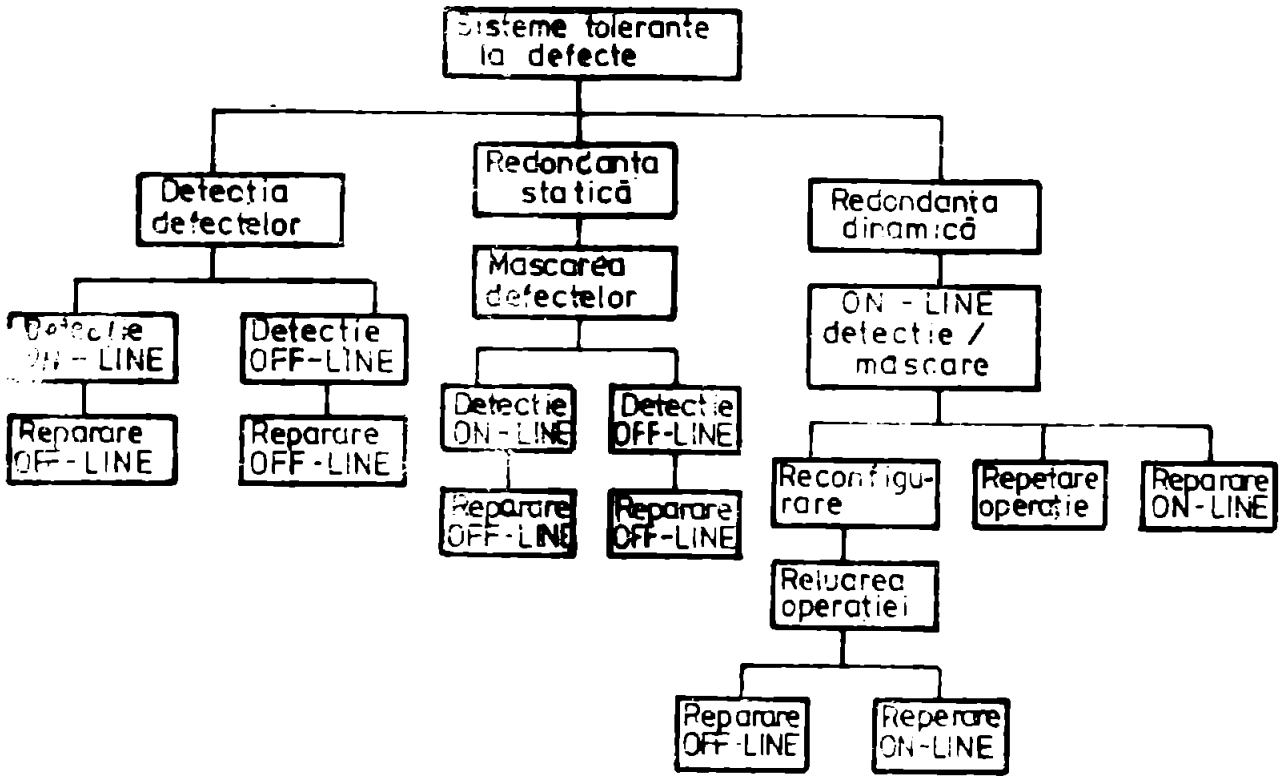


Fig.4.6

arhitectura sistemelor de calcul mici (microcalculatoare, minicalculatoare) care deobicei încorporează metode on-line de detecție a defectelor. Aceste sisteme nu sînt tolerante la defecte, în sensul strict al definiției [229]. Principalele metode folosite în sistemele cu detecția defectelor pentru găsirea defectelor sînt: coduri detectoare și corectoare de erori, verificarea bitului de paritate, verificarea sumei, coduri aritmetice, coduri ciclice, autotestarea, verificarea timpului (Watchdog time), teste funcționale on-line, dublarea unei instrucții iar în caz de nereușită se oprește sistemul (IBM370/168). După oprirea sistemului la detectarea unei erori se trece la rularea unor programe de diagnoză.

Pentru blocul de memorie sînt concepute coduri detectoare de erori pe liniile de date. La liniile de intrare/ieșire se aplică cu prioritate controlul parității. Pentru unitatea centrală se pot prevedea linii duble pentru semnalele de comandă și compararea acestora. (Univac 1100/60) [17, 102, 162, 172].

2. <u>Sistemele cu redondanță statică</u>: Arhitectura unor astfel de sisteme este concepută astfel ca sistemul să funcționeze corect chiar și în prezența unor defecte prin masca-

rea acestora. In astfel de sisteme se foloseşte cu precădere coduri corectoare de erori, sau o redondantă de tip majoritar, într-o configuraţie fixă, care au rolul de a corecta sau izola defectul [45, 122, 183, 184, 191].

Logica de interconectare a modulelor rămîne fixă şi nu necesită nici o intervenţie din exterior. Astfel de sisteme în principiu nu au posibilitatea de a detecta erorile. Efectul defectului este automat neutralizat fără a se specifica dacă a apărut sau nu. Un astfel de mod de operare implică,ca prin acumularea defectelor să se producă o funcţionare incorectă a sistemului. Pentru a elimina acest neajuns sistemele cu redondentă statică sînt prevăzute şi cu tehnici de detecţie a erorilor. Preţul de cost al sistemului în acest fel va creşte, dar într-o rată neglijabilă faţă de preţul unui sistem prevăzut numai cu redondantă statică. [229].

3. <u>Sisteme cu redondantă dinamică</u>: La detectarea unui defect sistemul este reconfigurat prin deconectarea din sistem a componentei defecte sau prin înlocuirea ei cu o componentă de rezervă. In cazul deconectării componentei din sistem puterea de procesare, a acestuia poate scădea. După reconfigurare, efectul erorii trebuie să fie eliminat. Acest lucru se face prin întoarcerea sistemului de operare în punctul ce a precedat defectul şi operarea va fi reluată de la acel punct (rollback).

In situaţia cînd o componentă a fost găsită defectă ea este înlocuită. Înlocuirea se poate face ON-LINE sau OFF-LINE In modul OFF-LINE componenta se înlocuieşte din exterior cu sau fără întreruperea sistemului.

In procedura ON-LINE înlocuirea echivalează cu o reconfigurare practică a sistemului. Mai există posibilitatea ca defectul să fie mascat (prin redondantă statică). In oricare caz componenta se înlocuieşte fără întreruperea sistemului.

Reconfigurarea sistemului se face dinamic la recunoaşterea unui defect, sau preventiv, cînd parametrii componentei scad sub un anumit prag. In cazul cînd în sistem se prevede şi o redondantă statică unitatea defectă este deconectată sau înlocuită dacă sistemul este prevăzut cu metode de detecţie a erorilor on-line [17, 93, 205, 218].

Sistemele cu redondanță dinamică sînt prevăzute cu metode adecvate de diagnoză pentru găsirea defectelor, acestea jucînd un rol important în buna funcționare a sistemului.

Reconfigurarea se poate face automat prin sistemul propriu on-line, sau manual prin off-line. În primul caz timpul de reconfigurare este scurt, iar în al doilea caz oprirea sistemului este mai lungă și poate cere o reinițializare a sistemului. Reconfigurarea on-line îmbunătățește atît coeficientul de disponibilitate cît și fiabilitatea sistemului.

Cele mai cunoscute sisteme cu redondanță dinamică sînt:

a) Sisteme de calcul paralele (fig.4.7,a). În această configurație dacă sistemul de calcul principal se defectează, atunci sistemul al doilea preia sarcina primului sistem. Comutarea unei lucrări de pe un sistem pe altul se face într-un timp relativ lung 15-30 min.

b) Sisteme de calcul paralele cu comutație periferice (fig.4.7,b). De obicei echipamentele periferice sînt scumpe și pentru a nu dota cele două sisteme cu periferice identice, se dublează doar procesorul și unitatea de memorie. Prin această tehnică prețul de cost al întregului sistem se reduce, prin duplarea numai a componentelor critice.

c) Sisteme de calcul paralele cu selector dublu a magistralei de i/o. Perifericele sînt legate direct cu cele două procesoare, ceea ce nu mai necesită timp pentru comutare în caz că procesorul principal este defect (fig.4.7,c). Procesoarele sînt în lucru simultan, ceea ce permite o continuitate în desfășurarea sarcinilor.

d) Sisteme de calcul paralele cu selector dublu a magistralei de i/o și cu magistrale de comunicație directă între procesoare (fig.4.7,d). În momentul defectului, procesorul principal trimite procesorului secundar o serie de date legate de starea sistemului. Cele două procesoare lucrează simultan. În program se introduc punote de verificare. În momentul cînd un procesor ajunge în punctul de verificare, comunică celui de al doilea procesor informații de verificare referitoare la adresă, starea sistemului, rezultatele intermediare, etc. Procesorul al doilea verifică dacă aceste informații coincid cu

informaţiile din propriul punct de funcţionare şi la rîndul lui trimite informaţii de verificare.

Urmărirea funcţionării corecte se face pe baza unui dicţionar de puncte de funcţionare [229].

e) Sisteme de calcul paralele cu selector şi periferice dublate (fig.4.7,e). In astfel de structură există o mai mare flexibilitate de a conecta procesoarele şi perifericele între ele la apariţia unui defect [231].

f) Sisteme de calcul cu procesoare duble cuplate printr-o memorie comună (fig.4.7,f). Procesoarele au comun, memoria şi echipamentele periferice [231].

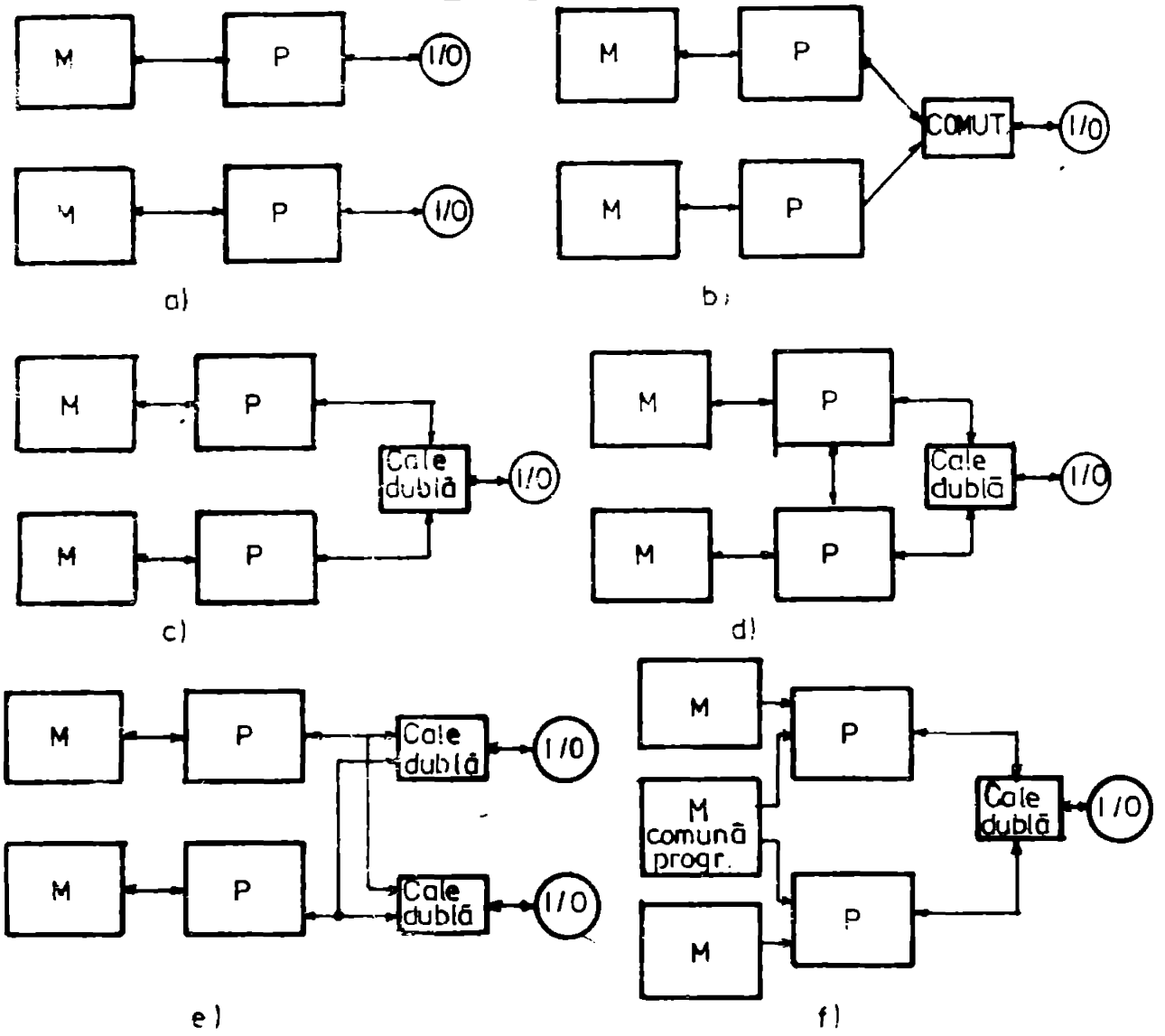


Fig.4.7

Deobicei pentru o cît mai rapidă comutare a sarcinilor de pe un sistem pe altul, cele două sisteme sînt puse să lucreze simultan, unde comunicarea cu lumea exterioară (proces) este realizată doar de o unitate (fig.4.8).

Cînd un defect este detectat în unitatea A atunci comanda va fi preluată de unitatea B.

Din literatură se desprind patru moduri de comutare a sarcinilor de la un sistem la altul.

a) Cu program de diagnoză. Unităţile execută un program de autodiagnoză atunci cînd comparatorul sesizează o diferenţă în funcţionarea celor două unităţi. In caz că eroarea este găsită la unitatea A atunci se generează semnalele de comutare a sarcinilor pe unitatea B. Metoda este folosită de firma BELL [229].

b) Prin descompunerea sistemului în module care sînt prevăzute cu posibilităţi de autoverificare pe fiecare modul sau permit autotestarea întregului sistem, atunci cînd comparatorul sesizează o diferenţă în funcţionare.

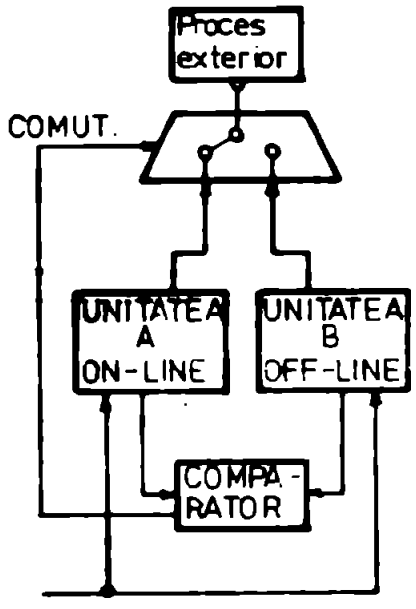


Fig.4.8

c) Prin utilizarea unui comparator de timp (Watchdog time). Unitatea activă generează periodic semnale de verificare. Dacă în funcţionarea acestei unităţi apare o eroare atunci semnalele de verificare nu mai apar, sau apar la momente diferite faţă de semnalul generat de comparatorul de timp, ceea ce determină sistemul să treacă controlul unităţii B.

d) Utilizarea unui arbitru extern de control. In momentul testării se generează un program de test pentru fiecare unitate. Programul de test trebuie să fie conceput să excite complect unităţile A şi B. Rezultatul programului de test este de dorit să fie exprimat sub forma unei constante. Dacă apare o eroare într-o unitate atunci cu o probabilitate foarte mare rezultatul nu va fi cel aşteptat.

Prin compararea rezultatelor cu constanta bună se determină unitatea defectă [229].

Sistemele tolerante la defecte, sau prevăzute cu posibilitatea detectării erorilor prezintă o deosebită importanţă în

aplicații care impun o funcționare a sistemului în timp real fără perturbarea procesului de comandă, chiar și în prezența defectelor. Sistemele cu redondanță statică sînt utilizate în misiuni de scurtă durată și în care intervenția omului nu este posibilă. Aceste sisteme comportă un preț de cost foarte ridicat, față de sistemele care nu sînt tolerante la defecte. Prețul poate fi pînă la 210 % mai mare.

Sistemele cu redondanță dinamică sînt mai ieftine, numai cu 100 % mai scumpe decît sistemele clasice, în schimb necesită un anumit timp din momentul în care a fost depistată eroarea pînă în momentul înlocuirii componentei defecte.

Sistemele cu detectarea erorilor sînt cu cel mult 50 % mai scumpe decît sistemele care nu sînt tolerante la defecte. Ele prezintă particularitatea că defectul nu poate fi înlăturat decît din exteriorul [127, 216].

4.4. Concluzii privind diagnoza blocurilor numerice complexe

Pe baza analizei făcute se poate desprinde concluzia că diagnoza unor blocuri numerice complexe cu un număr mare de componente și funcții, prezintă un grad ridicat de complexitate. Utilizarea metodelor clasice de diagnoză folosite pentru scheme cu un număr redus de componente nu sînt adecvate blocurilor numerice complexe. Mai mult prin utilizarea circuitelor integrate pe scară largă, blocurile prezintă caracteristici speciale care influențează diagnoza.

Principalele probleme care se pun la diagnoza unităților numerice complexe sînt:

- Blocurile numerice pot fi realizate cu circuite a căror densitate de integrare este foarte diferită, cu structura internă puțin cunoscută de către utilizator. Din acest motiv metodele de test orientate pe structură nu pot fi aplicate ;

- Prin trecerea la integrarea pe scară mare și foarte mare circuitele își pierd din transparență. Nu se mai pot aplica procedeele de testare structurală. Testarea tuturor căilor activate și găsirea defectelor de punere pe unu, respectiv de punere pe zero nu mai este posibilă dar nici necesară, deoarece circuite integrate LSI funcțional compatibile, pot fi realizate diferit.

- Datorită numărului mare de funcţii realizate, în timpul diagnozei nu se poate atîpini în întregime procedeul de testare, chiar dacă s-ar utiliza un sistem de calcul pentru simulare.

- Testarea completă a tuturor funcţiilor cu toate combinaţiile de date permise nu mai este posibilă, datorită volumului prea mare de date şi a timpului prohibit. Stabilirea testului are un caracter euristic.

- Datorită caracterului complex şi a funcţionării date, nemodificabil nici pentru scopuri de diagnoză, nu se pot aplica metode cvasistatice, fiind necesare metode de test dinamice.

- La diferite circuite integrate LSI, exemplu microprocesoare sau sisteme complexe, se poate trece de la activarea căilor la activarea modulului. Această posibilitate există doar atîta timp cît este posibil accesul din exterior la aceste module. Dacă gradul de integrare creşte în continuare, ceea ce este sigur, nici această metodă nu mai este posibilă.

Toate problemele prezentate duc la concluzia că o testare structurală este neaplicabilă urmînd a se aplica o testare funcţională. De aici rezultă deplasarea centrului de greutate pe teste realizate prin program. Avantajoase din punct de vedere al testării s-au arătat a fi structurile orientate pe magistrale. In acest fel se realizează alegerea modulelor, la care se permite în mod simplu introducerea şi extragerea informaţiilor de diagnoză.

Prin utilizarea metodelor de proiectare pentru îmbunătăţirea testabilităţii la nivel de circuit integrat bloc sau sistem se poate obţine o flexibilitate sporită la testare şi posibilităţi mai bune de diagnoză dar cu o creştere a complexităţii structurii cu implicaţii asupra preţului de cost. In plus nu se rezolvă în totalitate aspectele legate de localizarea defectelor. Metodele de proiectare pentru îmbunătăţirea testabilităţii nu sînt general valabile, fiind necesare eforturi de adaptare pentru fiecare aplicaţie în parte.

Se apreciază că prin îmbunătăţiri tehnologice testarea funcţională la nivel de circuit integrat ar putea fi rezolvată în cadrul modulului, rămînînd deschisă problema testării la nivel de bloc sau sistem.

Considerăm că în această direcţie realizarea unor structuri autotestabile prevăzute cu programe de test corespunzătoare pot asigura o soluţie optimă. Această orientare este susţinută de faptul că tot mai frecvent se utilizează în prezent, sisteme de calcul multiprocesor, ceea ce permite, printr-o structură de interconexiuni corespunzătoare, ca procesoarele din sistem să fie utilizate şi la diagnoză. In acest fel partea de harware creşte nesemnificativ.

Prin autodiagnoză se pot rezolva majoritatea problemelor ridicate de asigurarea bunei funcţionări a unui sistem, de localizarea părţii din sistem defectă. Blocul defect poate fi testat în continuare off-line, sau on-line. In cazul unei testări on-line părţile nedefecte ale sistemului pot contribui la localizarea modulului sau a circuitului defect, dacă sînt prevăzute programe de test adecvate.

Apreciem că această lucrare contribuie în mod original la rezolvarea unor aspecte teoretice legate de realizarea unor structuri de calculatoare cu posibilităţi de autotestare. Rezolvînd problema găsirii unui optim între numărul de unităţi din sistem, numărul de unităţi defecte detectate şi numărul de interconexiuni între unităţi. Prin tratarea aspectelor legate de diagnoza unităţilor defecte permanente şi intermitente, într-o formă unitară şi originală, lucrarea se încadrează printre puţinele lucrări de specialitate, care abordează şi aprofundează toate problemele legate de găsirea celor mai eficiente structuri autotestabile cît şi de rezolvare practică a autodiagnozei prin implementarea unor metode de diagnoză adecvate.

CAPITOLUL 5

SISTEME MULTIMICROPROCESOR CU POSIBILITATI DE AUTOTESTARE

5.1. Arhitectura SAT

SAT fac parte din categoria sistemelor tolerante la defecte, în sensul celor prezentate în paragraful 4.3. In această direcţie există un număr foarte mare de arhitecturi posibile privind detecţia, mascarea şi repararea defectelor. In general structura unui sistem tolerant la defecte s-a ales în funcţie de ţinta de aplicaţie din care trebuie să facă parte sistemul.

Pe baza analizei făcute [176, 229] rezultă că sistemele autotestabile pot fi împărţite în trei tipuri arhitecturale distincte. Criteriu după care s-a făcut această clasificare derinde de posibilitatea sistemului de a fi divizat în n unităţi funcţionale, care să respecte condiţiile din ipoteza 2.1. In principiu fiecare unitate trebuie să fie capabilă de a genera stimuli de test spre alte unităţi şi să poată evalua răspunsul acestei unităţi. Acest lucru implică ca fiecare unitate să fie prevăzută cu o unitate de memorie proprie, pentru stocarea programelor de test, să aibă posibilitatea de prelucrare a informaţiilor necesare generării stimulilor de test şi să poată evalua funcţionarea altor unităţi pe baza răspunsurilor acestora la stimuli de test. In funcţie de posibilităţile de divizare în unităţi funcţionale sistemele autotestabile pot fi conectate direct, sau *indirect*.

5.1.1. SAT multiprocesor.

Sînt sistemele care pot fi împărţite în n unităţi funcţionale, între care există legături de test, conform celor indicate în capitolul 2. In această categorie ar putea intra reţelele de calculatoare sau sistemele multimicroprocesor [226].

O categorie de sisteme autotestabile multiprocesor conectate direct ar putea fi sistemele complect interconectate prevăzute atît cu detecţie erorilor cît şi cu mascarea şi reperarea defectelor, dacă sistemul este prevăzut cu programe adecvate de test şi o redondanţă statică şi dinamică corespunzătoare. Astfel de sisteme ar prezenta caracteristici foarte bune din punct de vedere al detecţiei defectelor, simplificînd mult programele de test. Sistemul ar putea detecta defecte multiple şi prin reconfigurare, unităţile defecte ar putea fi înlocuite sau deconectate. Prezintă dezavantajul unor interconexiuni directe între module care să faciliteze generarea stimulilor de test şi evaluarea răspunsurilor. Aceasta implică că fiecare procesor trebuie să fie conectat direct cu cel puţin alte t procesoare prin legături dedicate. Inserarea unui procesor suplimentar necesită o mulţime de noi conexiuni. In schimb, modularitatea locală, imunitatea la defecţiuni şi capacitatea de reconfigurare sînt sporite. O astfel de structură poate funcţiona şi la o capacitate de procesoare mai redusă prin simpla deconectare a procesorului sau procesoarelor defecte.

Apariţia conflictelor între procesoare nu este probabilă, iar logica circuitelor de comunicaţie este simplă. Această arhitectură impune tehnici de adresare directă a locaţiilor în care se trimit mesajele. Sistemele de acest tip pot fi concentrate sau distribuite geografic, prima variantă fiind mai răspîndită [220, 234].

Printre alte posibilităţi de interconectare directă a microprocesoarelor se prezintă arhitecturile cu memorie comună şi cele cu magistrală comună.

Arhitecturile cu memorie comună, reprezintă cel mai răspîndit mod de interconectare a procesoarelor şi constă din faptul că procesoarele comunică printr-o memorie comună accesibilă fiecăruia dintre ele. Memoria comună este utilizată mai degrabă ca un mijloc de comunicare şi nu neapărat ca un mijloc de înmagazinare a datelor. Modularitatea locală a sistemului este foarte bună, în sensul că un procesor poate fi adăugat oriunde în cadrul arhitecturii, procesoarele fiind echivalente din punct de vedere topologic. Mărirea capacităţii de transmitere a mesajelor poate fi efectuată prin mărirea dimensiunii memoriei comune. Costul sistemului este în mod fundamental influenţat de

tipul structurii prin care procesoarele au acces la unitatea de memorie. Adăugarea unui procesor duce la creşterea complexităţii structurii căilor de comunicaţie cu memorie. Dacă însă memoria este conectată la toate procesoarele printr-o magistrală comună, modularitatea devine foarte bună, în schimb banda de frecvenţă a magistralei impune o limitare severă a performanţelor sistemului. Complexitatea comunicaţiilor nu este prea mare, efectul defectării unui procesor afectează într-o măsură mică întregul sistem. Pot apărea de asemenea fenomene de distrugere accidentală sau deliberată a informaţiilor depuse în memorie, dacă nu se impune restricţii în accesul procesoarelor la memoria comună. Acest tip de sisteme au apărut datorită necesităţii înmagazinate unor programe şi fişiere de utilizare comună, utilizarea memoriei ca mijloc de comunicaţie apărînd ca un efect secundar.

Eficienţa sistemului nu creşte proporţional cu numărul procesoarelor suplimentare. Cînd memoria este utilizată pentru ambele scopuri menţionate anterior, numărul de procesoare ce pot fi cuplate în mod eficient într-o asemenea structură depăşeşte rar numărul de trei datorită creşterii numărului de conflicte potenţiale în accesul la memoria comună[230].

Arhitecturile cu magistrală comună includ un anumit număr de procesoare conectate la aceeaşi magistrală prin care ele comunică direct. Modularitatea obţinută este foarte bună şi nu depinde de poziţia relativă a procesoarelor adăugate în raport cu cele existente. În ceea ce priveşte numărul de conexiuni la magistrală, modularitatea nu se obţine uşor, necesitînd înlocuirea magistralei sau dublarea ei pentru creşterea eficienţei comunicaţiilor, lucru care duce la modificarea interfeţei cu toate procesoarele. Imunitatea la defecţiuni este foarte bună în raport cu defectarea procesoarelor şi foarte scăzută în raport cu defectarea magistralei. În plus, magistrala reprezintă principala limitare a acestui tip de sisteme. Datorită preţului ridicat al cablurilor se utilizează de obicei magistrale cu transmisie serie. Fiabilitatea sistemului poate fi mărită prin dublarea magistralei [106, 226, 230].

5.1.2. SAU-multiprocesor conectate indirect.

Conectarea între procesoare se face prin intermediul

unei anumite structuri hardware, numită comutator. In literatura de specialitate se amintesc mai multe arhitecturi multiprocesor conectate indirect. Cele mai performante din punct de vedere al imunității la defecte sînt:

Arhitectura stea. Arhitectura stea are multe trăsături comune cu arhitecturile cu conectare directă și căi de comunicație partajată. In ambele cazuri, mesajele trebuie să treacă prin anumite structuri hardware. Modularitatea este bună în ceea ce privește procesoarele, dar slabă în ceea ce privește comutatorul. Imunitatea la defectele comutatorului cît și posibilitatea de reconfigurare sînt reduse datorită faptului că adăugarea unui procesor implică utilizarea unui comutator central mai complex.

Arhitecturi cu magistrală comună centralată de un comutator central. Procesoarele sînt conectate la comutatorul central printr-o magistrală comună partajată printr-o logică adecvată de priorități și control, ceea ce face ca la un moment dat un singur procesor poate deveni controlor de magistrală. Ca atare, procesoarele trebuie să preia controlul magistralei înainte de a transmite mesajele către comutator, care la rîndul lui le retransmite către destinația corectă. Caracteristicile acestei arhitecturi sînt similare cu ale arhitecturii stea cu comutator central. Modularitatea ei este superioară însă arhitecturii stea, deoarece microprocesorul suplimentar trebuie conectat numai la magistrală și nu la comutator. Acest tip de arhitectură nu este prea larg utilizat, datorită faptului că fiabilitatea sistemului este puternic dependentă de magistrala comună cît și de comutatorul central.

Rețele regulate, constau dintr-o matrice de microprocesoare conectate între ele prin conexiuni dedicate în structuri topologice simetrice. Modularitatea acestui tip de arhitectură este mică, deoarece adăugarea unui procesor distruge simetria rețelei. Pentru mărirea puterii de prelucrare trebuie adăugate mai multe procesoare astfel încît să se păstreze simetria rețelei. [227, 230]. Structura este inflexibilă iar complexitatea comunicațiilor nu este prea mare. Imunitatea la defecte este bună și depinde de protocolul de comunicație. Posibilitățile de reconfigurare sînt mici, dat fiind că trebuie menținută simetria rețelei. In cazul apariției defecte se menține buna funcționare în detrimentul simetriei. Arhi-

tecturile regulate deşi foarte elegante sînt dificil de realizat [20]. Cea mai întîlnită este arhitectura ierarhizată tip arbore, în care fiecare procesor comunică numai cu un procesor superior şi cu toţi procesorii ierarhic inferiori.

Reţele neregulate, constau în faptul că fiecare procesor poate fi conectat cu orice număr de vecini prin conexiuni dedicate. Modularitatea locală este foarte bună, căci în orice punct al reţelei se pot introduce atît procesoare, cît şi conexiuni suplimentare. Costul inserării nu este prea mare, putîndu-se reduce la costul unei singure conexiuni suplimentare. Flexibilitatea este foarte mare. Cu cît o structură este mai asimetrică cu atît este mai flexibilă, deoarece între două procesoare date se pot realiza o multitudine de căi de comunicaţie. Dezavantajul imediat constă din complexitatea sporită a protocoalelor de comunicaţie.

Cele mai cunoscute aplicaţii ale structurilor neregulate sînt reţelele de calculatoare distribuite pe spaţii geografice întinse [Iau, 194, 226].

5.1.3. SAT redondante.

In cazul unor sisteme în care nu toate cele n unităţi funcţionale îndeplinesc condiţia de a fi prevăzute cu memorie şi posibilităţi de prelucrare a informaţiei, dar există un număr minim de unităţi (trei) care să îndeplinească condiţiile impuse prin ipoteza 2.1, atunci sistemul se poate împărţi într-un subsistem autodiagnozabil. Subsistemul astfel constituit va forma partea centrală a sistemului (hardcore) din punct de vedere al testabilităţii, de la care se va testa restul componentelor din sistem printr-o metodă de activare a modulelor. Astfel de sisteme pot fi prevăzute cu redondanţă statică pentru mascarea defectelor şi/sau redondanţă dinamică pentru reconfigurarea sistemului, fie prin înlocuirea componentei defecte, fie prin preluarea sarcinilor unităţilor defecte de către unităţile funcţional corecte. Acest lucru se va realiza prin scăderea capacităţii de procesare a sistemului.

In general componentele unui sistem care sînt capabile să-şi autogenereze stimuli de test sînt microprocesoarele în cooperare cu elemente de memorie. Astfel de soluţii cu hardcore este indicată în cazul sistemelor multimicroprocesor,

prevăzute cu un număr limitat de microprocesoare, care se doresc a fi interconectate prin structuri simple şi sigure.

O ultimă categorie de SAT ar fi acele sisteme care nu pot fi divizate într-un număr minim de unităţi funcţionale, care să satisfacă condiţiile de autodiagnoză. In această situaţie este necesară introducerea unor elemente redondante care să preia funcţia de generare şi evaluare a modulelor din sistem. Redondanţa se va manifesta atît la nivel hard cît şi la nivel soft. In acest caz elementele redondante vor trebui să fie realizate cu un minim de componente. Astfel de sisteme implică utilizarea unor metode de proiectare adecvate pentru îmbunătăţirea testabilităţii.

Sistemele autotestabile redondante prezintă dezavantajul că sînt puţin flexibile, fiind destinate unor aplicaţii particulare; sînt relativ scumpe. Nu pot fi adaptate simplu pentru o nouă aplicaţie, ceea ce implică o reconfigurare a structuri atît la nivel de hard cît şi la soft.

O structură autotestabilă multimicroprocesor prezintă în prezent calea cea mai eficientă şi cu cele mai mari perspective de utilizare. Pentru alegerea unei arhitecturi multiprocesor cu posibilităţi de autotestare se va analiza unele structuri multiprocesor pentru adoptarea unor soluţii practice, economice şi performante, care să satisfacă cerinţele obiectivelor urmărite.

5.2. Sisteme multimicroprocesor.

Configuraţiile multimicroprocesor au căpătat în ultima vreme o deosebită importanţă [15b, 220] prin împărţirea sarcinilor între mai multe procesoare ce sînt superioare celor cu un singur calculator central puternic.

Ideea de a utiliza mai mult de un element de prelucrare pentru îmbunătăţirea performanţelor sistemului a precedat apariţia microprocesoarelor, dar abia acum tehnologia permite utilizarea puterii de calcul într-o gamă largă de aplicaţii care nu era pînă acum practică din cauza preţului prohibitiv şi dimensiunilor mari.

Avantajele care apar prin utilizarea sistemelor multimicroprocesor (SMM) sînt [...]; sensibilitatea redusă la perturbaţii, timpul de execuţie a unei sarcini este mult redus permi-

tînd chiar lucru în timp real; fiabilitate sporită (prin redondanţă sau imunitate la defecte); dezvoltarea modulară a sistemelor (spaţială şi funcţională) ; partajarea resurselor (hardware, programe, date, timp); partajarea funcţională a sarcinilor pe procesoare specializate ; un raport cost/performanţă excelent.

Odată cu aceste avantaje, realizarea sistemelor multimicroprocesor ridică o serie de probleme: separarea sarcinilor executate în paralel pe mai multe procesoare; determinarea celor mai eficiente structuri de interconectare a procesoarelor; proiectarea unor mecanisme cît mai adecvate pentru translatarea adreselor logice în adrese fizice; eliminarea interblocărilor care apar cînd un procesor aşteaptă după o resursă alocată altuia; proiectarea unor structuri hardware şi software care să faciliteze imunitatea la defecte a SMM [220, 221, 223, 224].

Rezolvarea problemelor enunţate fac ca concepţia şi strategia aplicării SMM să se dezvolte în mod nesistematic; existînd un număr mare de soluţii bune pentru un anumit tip de aplicaţie.

Pentru realizarea unor sisteme multimicroprocesor trebuie bine definite relaţiile logice între diversele elemente ale sistemului. În acest context, structura logică se referă la modul în care funcţiile de comandă sînt distribuite între diversele elemente ale sistemului. Cele mai frecvente sînt relaţiile de subordonare şi de cooperare. Într-un sistem în care relaţiile sînt de subordonare, elementele sînt structurate ierarhic implicînd relaţii stăpîn-sclav; într-un sistem în care relaţiile sînt de cooperare, elementele sînt echivalente din punct de vedere logic, implicînd relaţii de egal la egal [220, 221].

La nivel de structură fizică SMM implică metode specifice de transfer a informaţiilor şi depinde de aranjamentul comunicaţiilor interprocesor şi de modul de interconectare.

Referitor la comunicaţiile interprocesor, acestea se pot efectua fie printr-o memorie comună, fie printr-o structură de magistrală. În structura cu memorie comună, toate transferurile de date se fac prin memoria comună şi elementele, nu au acces direct unul la celălalt. În cazul structurii cu magistra-

lă, o legătură logică stabilită pe structura de magistrală creează o cale de comunicare între elemente; în cazul cel mai general transferurile de date sînt inițiate și îndeplinite într-o manieră distribuită.

La sistemele cu transferuri mari de date tehnicile menționate nu sînt eficiente datorită conflictelor sporite pentru resursele comune. Problema se complică și mai mult în sistemele microprocesor datorită timpului de acces memorie-procesor și datorită capacităților de transfer limitat între i/E-procesor [220].

Din punctul de vedere al conectării elementelor există multe moduri de a interconecta un număr de n elemente într-un sistem, dar în stabilirea schemei de interconectare apar factori ca fiabilitatea și dezvoltarea. O schemă fiabilă asigură o cale secundară în cazul în care legătura, calea directă, între două elemente se defectează. O schemă de interconectare flexibilă, permite adăugarea de noi elemente fără afectarea structurii existente [226] . Cele patru scheme de interconectare mai importante sînt: cu magistrală comună, stea inel, complect conectate [220]. Celelalte topologii au la bază combinații sau variații ale acestora.

Cele mai indicate scheme de interconexiuni în SAT sînt cele cu magistrală comună sau complect conectate.

În funcție de modul de interconexiune sistemele pot fi clasificate după gradul de cuplare și natura intercomunicațiilor între procesoare. Cuplarea se referă la abilitatea diverselor elemente de a-și partaja resursele, la cele două extreme situîndu-se sistemele: slab cuplate și cele puternic cuplate.

Sistemele slab cuplate, sau rețele de calculatoare, au următoarele caracteristici [135, 226, 290].

- Rețelele conțin un număr de sisteme de calcul independente, care pot fi dispersate pe spații geografice foarte mari.

- Interconectarea între calculatoarele din rețea se face printr-o interfață de comunicare, unde legăturile de comunicație sînt în general legături serie de mare viteză.

- Comunicațiile intercalculator sînt supuse unui protocol rigid .

In general rețeaua este utilizată numai pentru comunicații. Prelucrarea propriu-zisă se face de fapt pe cîte un element al rețelei. In cadrul rețelei oricare utilizator poate folosi facilitățile de calcul ale celorlalte elemente.

Rețelele de calcul cu cerințele lor rigide în ceea ce privește comunicațiile interprocesor , nu sînt direct aplicabile SMM, dar probabil versiuni modificate ale rețelelor de calculatoare vor fi realizate și cu microprocesoare.

Sistemele cuplate puternic, sau SMM propriu-zise se caracterizează:

- Dispun de o memorie comună. In plus, fiecare procesor poate avea o memorie separată de date.

- Dispun de un sistem de operare comun. Un singur sistem de operare controlează și coordonează toate interacțiunile dintre procesoare și procese.

- Dispun de resurse partajate. Resursele sistemului sînt în general partajate între procesoare, ca de altfel și intrările/ieșirile (I/b).

- Sincronizarea interprocesoarelor ce cooperează este absolut necesară.

- Fiecare dintre procesoare prezintă o anumită autonomie

- Sarcinile pe procesoare sînt egal distribuite, în general, iar în caz de supraîncărcare a unui procesor se permite o reconfigurare dinamică a sarcinilor.

Limitarea majoră a sistemelor multiprocesor puternic cuplate constă în posibilitatea apariției conflictelor în accesul la memoria comună. Acest fapt tinde să limiteze superior numărul procesoarelor care pot fi efectiv guvernate de un singur sistem de operare. Cele mai multe configurații procesor-memorie tind să reducă cantitatea conflictelor referitoare la accesul la memoria principală. Cele mai utilizate structuri procesor-memorie sînt:

- cu magistrală comună; toate elementele sistemului sînt conectate la o magistrală comună.

- cu comutator ; elementele sînt conectate la un modul separat, denumit comutator, care poate asigura mai multe conexiuni simultan între perechi de elemente.

-cu memorie multiport, la care fiecare element de memorie are mai multe registre de memorie (port) de acces, și e

conectat la celelalte elemente printr-o magistrală multiplă.

Sistemele care combină cele mai bune calităţi ale sistemelor tare şi slab cuplate sînt mai adecvate în SMM. Aceste structuri moderat cuplate sînt cunoscute sub denumirea de sisteme de microcalculatoare distribuite. In astfel de sisteme, sarcina totală de prelucrare este împărţită de aşa manieră încît să poată fi alocată la o varietate mare de elemente

Sistemele distribuite realizate cu microprocesor se caracterizează prin:

- Fiecare element individual constă dintr-un microprocesor, memorie locală RAM, ROM, şi eventual poate utiliza sau controla periferice.

- Structura sistemului nu este în mod necesar simetrică deoarece procesoarele sale pot să prezinte complexitate diferită.

- Comunicarea interprocesor se face cu precădere prin date. Totuşi în anumite situaţii datele pot conţine comenzi sau includ răspunsuri la cereri specifice.

- Fiecare componentă hard sau soft este croită pentru sarcina specifică ce o îndeplineşte, fiind dedicată sarcinii de îndeplinit.

- Procesoarele pot fi distribuite local(în acelaşi laborator, fabrică, vehicul) sau geografic (sisteme de comunicaţie), iar sarcina de îndeplinit să fie bine cunoscută dinainte, astfel încît funcţiile sistemului să poată fi divizate între elementele individuale de prelucrare.

Comenzile numerice, comanda proceselor, automatizările discrete sînt exemple posibile de aplicare a arhitecturilor multiprocesor distribuite [204, 188].

Luînd în considerare fluxul instrucţiunilor şi datelor, sistemele de calcul au fost divizate în sisteme cu prelucrare serie şi sisteme cu prelucrare paralelă.

In cadrul sistemelor cu prelucrare serie se încadrează:

- Sistemele cu un singur flux de instrucţii şi un singur flux de date - SISD (Single Instructions Stream-Single Data Stream). Un sistem SISD este un sistem clasic (von Neumann) care execută instrucţiunile secvenţial.

Sistemele cu prelucrare paralelă sînt împărţite:

- Sisteme cu mai multe fluxuri de instrucţii şi un sin-

gue flux de date - MISD (Multiple Instruction-Single Data Stream). Au existat contraverse în legătură cu tipul sistemelor ce ar trebui incluse în această clasă. Aceste structuri sînt practic nerealizabile [220].

- Sisteme cu un singur flux de instrucţii şi mai multe fluxuri de date - SIMD, care includ sistemele cu prelucrare paralelă: procesoarele matriciale şi cele asociative. Ele au în general o singură unitate centrală de control, care aduce şi decodifică instrucţiile şi care apoi difuzează controlul elementelor de prelucrare (procesoare).

- Sisteme cu mai multe fluxuri de instrucţii şi mai multe fluxuri de date - MIMD, multiprocesoare propriu-zise. Clasa MIMD, este cea mai generală, conţine sisteme cu mai multe procesoare, fiecare cu propria unitate de comandă.

Clasificarea făcută mai sus de Flynn a fost îmbunătăţită de alţi autori (Braun, White) [220], printr-o diferenţiere a sistemelor MIMD astfel: sisteme puternic cuplate, sisteme slab cuplate şi sisteme moderat cuplate. Acesti autori postulează explicit că MISD este o clasă pur teoretică de calculatoare paralele fără nici o valoare practică.

O clasificare după modul de prelucrare şi de interacţiune este prezentată în fig.3.1 [22 0].

In sistemele SIMD (numite şi sisteme cu procesor paralel), o singură unitate de comandă aduce şi decodifică instrucţiile. Instrucţia este executată de unitatea de comandă însăşi, sau este difuzată către alte elemente de prelucrare (aceeaşi instrucţie este executată de un vector de procesoare asupra unui vector de date).

In cadrul procesoarelor matriciale, instrucţiile manipulează simultan vectori de date, iar capacitatea unităţii de comandă este limitată, fiind cele mai eficiente arhitecturi SIMD din punct de vedere al raportului cost/performanţă. Viteza de execuţie a acestui tip de organizare este foarte mare datorită paralelismului operaţiilor pe diferite fluxuri de date. In acest tip de sisteme, elementele de prelucrare sînt independente, fiecare dispunînd de memorie şi registre proprii, dar conducînd sub controlul unei singure unităţi de comandă.

Procesoarele asociative au acces şi operează asupra datelor prin conţinutul lor nu prin adresă. Ele constituie un

tip de procesoare matriciale în care elementele de prelucrare nu sînt adresate direct. Ele sînt adresate cînd sînt satisfăcute anumite relaţii între conţinutul unui registru încărcat de unitatea de comandă şi datele conţinute în registrele asociative din elementul de prelucrare. Procesoarele astfel selectate primesc următoarea instrucţie din program în timp ce celelalte rămîn inactive.

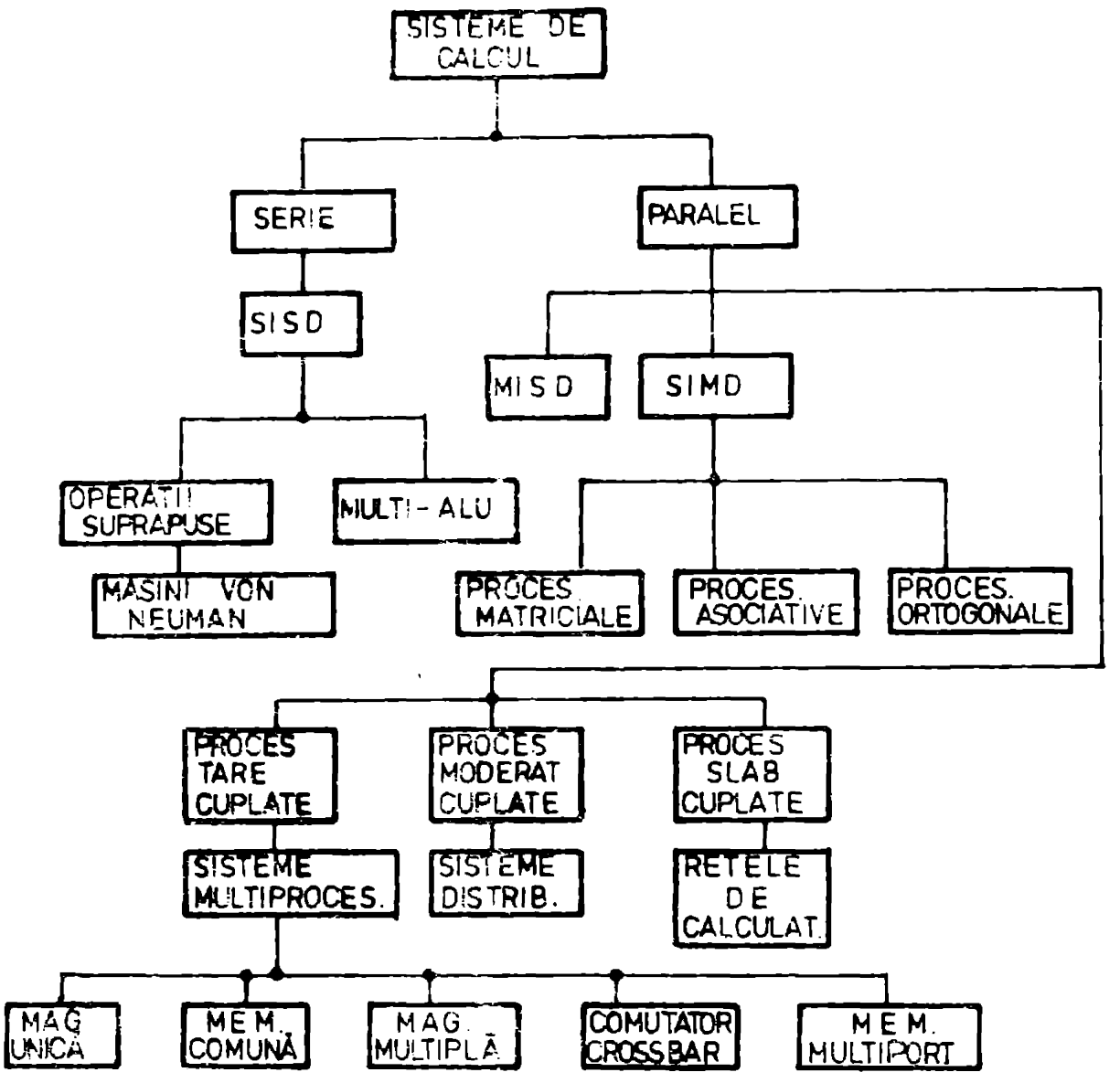


Fig.5.1

Procesoarele ortogonale se caracterizează prin blocul de memorie care poate fi adresată atît direct (cuvînt cu cuvînt pentru un calculator de I/E) cît şi pe octet (pe felii de cîte un octet de mai multe elemente de prelucrare). O astfel de adresare se realizează asociativ.

5.3. Implementarea unui SAI multimicroprocesor.

Pentru implementarea şi realizarea unui sistem de calcul multimicroprocesor cu posibilităţi de autotestare se propun două soluţii. Ambele soluţii implică conceperea unei celule autotestabile, care să reprezinte partea principală,din punct de vedere al testării, a unui sistem multimicroprocesor Celula autotestabilă a fost prevăzută cu următoarele caracteristici:

- Să implice utilizarea unui minim de hard şi soft suplimentar pentru partea de testare.

- Să asigure testarea tuturor componentelor din cadrul celulei, în faza de rulare a programelor de test.

Programele de test vor fi rulate automat, la începutul executării sarcinilor, sau la cererea operatorului.

- Celula autotestabilă va fi concepută sub forma unui sistem distribuit, cu posibilităţi de utilizare într-un număr mare de aplicaţii.

Dacă celula autotestabilă face parte dintr-un sistem de calcul cu un număr mare de componente fazele de testare ar comporta următoarele etape:

a) - testarea nucleului (celulei) autotestabile.

b) - extinderea nucleului.

c) - repetarea pasului b, până la epuizarea blocurilor sistemului.

Etapa a) este cea care ridică cele mai complexe probleme, cu precădere în cazul autotestării, deoarece nu se poate executa o evaluare a răspunsurilor blocului testat şi lua decizia corespunzătoare, de către un bloc în stare de funcţionare incertă.

Celula autotestabilă trebuie să fie realizată din cel puţin trei unităţi funcţionale ce posedă proprietăţile din ipoteza 2.1.

In cazul unui sistem multimicroprocesor cu posibilităţi de autotestare, blocurile funcţionale pot cuprinde un microprocesor şi o memorie fixă ROM, în care sînt depuse programele de test. Microprocesorul îndeplineşte rolul de comandă şi evaluare a testelor. Mai mult microprocesorul poate, împreună cu memoria ROM, să-şi genereze programe de autotest

[212], dar nu poate să evalueze răspunsul. Acest lucru îl va îndeplini altă unitate din cadrul sistemului.

Pentru îndeplinirea condiţiilor din capitolul 2 şi avînd în vedere caracteristicile microprocesoarelor se propun două soluţii de implementare a unei celule autotestabile.

5.3.1. Arhitectura SAT cu trei microprocesoare.

SAT cu trei microprocesoare se prezintă în fig.5.2. O unitate funcţională este concepută să cuprindă un microprocesor (μP) şi o memorie fixă ROM ce conţine programul de test pentru microprocesorul din unitatea respectivă. Evaluarea funcţionării corecte a unei unităţi este făcută de o altă unitate.

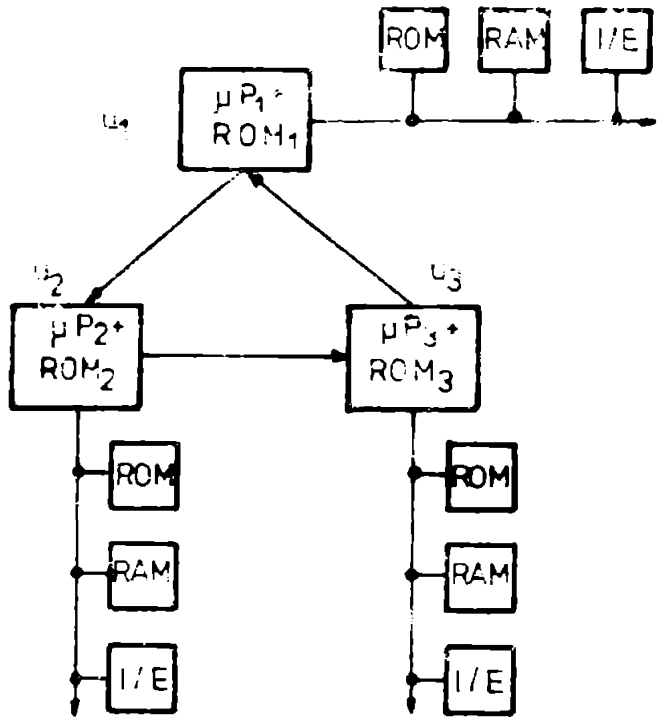


Fig.5.2

Modul de evaluare depinde de programele de test. Acestea trebuie să asigure o excitare corespunzătoare a microprocesorului şi a memoriei ROM. In literatură se dau mai multe soluţii bune pentru programele de test [206, 212, 220]. Strategia de evaluare poate fi concepută:

- unitatea de evaluat să aibe acces la punctele de verificare intermediabile din programul de test. In momentul în care se detectează o eroare se întrerupe programul de test se poziţionează elementul sindromului în mod corespunzător şi se trece la testarea unităţii următoare.

- unitatea de evaluat are acces doar la rezultatul final al testului. In acest caz interconectarea se face mai simplu, dar timpul de testare este mai lung.

SAT cu trei microprocesoare poate fi conceput atît pentru detectarea erorilor dar şi pentru mascarea acestora dacă în timpul funcţionării sistemului cele trei μP sînt conectate să lucreze într-o logică majoritară. Structura unui astfel de sistem este mai complexă, cele trei μP vor executa aceeaşi

sarcină iar în timpul testării sistemul trebuie reconfigurat ca în fig.5.2. Prin reconfigurare şi o detecţie corespunzătoare unitatea defectă poate fi deconectată din sistem, sau înlocuită, dacă se prevede acest lucru.

Dezavantajul unei astfel de structuri constă în faptul că interconectarea unităţilor funcţionale este mai dificil de realizat, în schimb prezintă o flexibilitate sporită în aplicaţii.

5.3.2. Arhitectura SAT cu două microprocesoare.

SAT cu două microprocesoare se prezintă în fig.5.3. Sistemul s-a descompus în două unităţi funcţionale realizate cu microprocesor şi un al treilea element redondant format dintr-un comparator sau un analizor de semnături, care are rolul de a evalua comportarea unităţii u_2, pe baza programului de test rezident în memoria ROM a acestei unităţi. Acest bloc va indica despre unitatea u_2 că este funcţional corectă sau defectă.

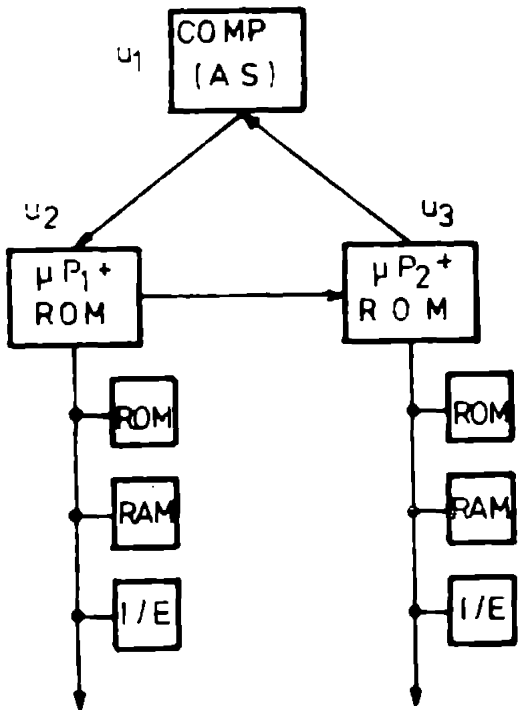


Fig.5.3

Avantajele unei astfel de structuri constă în faptul că interconectarea între unităţi este mai simplă (se face între două microsisteme); unitatea funcţională u_1 este simplă ceea ce implică că şi testarea ei se face mai simplu.

Dacă u_1 este un simplu comparator, atunci nu poate fi folosit decît la detecţia defectelor. Dacă u_1 încorporează un AS atunci prin atagarea unei memorii fixe ROM, cu rolul de a memora dicţionarul de semnături, acest bloc poate fi utilizat şi la localizarea defectelor, dar complexitatea unităţii poate creşte pînă la nivelul unui microsistem.

Pentru implementarea unui sistem autotestabil multimicroprocesor s-a ales varianta cu două microprocesoare.

5.4. Sistem biprocesor cu posibilităţi de autotestare.

5.4.1. Structura sistemului.

Sistemul biprocesor cu posibilităţi de autotestare a fost conceput sub forma unui sistem cu funcţii distribuite format din două microsisteme [175] . Microsistemele s-au realizat cu Z80 respectiv I80&0. Din punct de vedere logic sistemul s-a organizat pe verticală cu funcţia principală pentru microsistemul (μS) Z80 şi de subordonare pentru μS cu I8080. μS realizat cu I8080 are rol de procesor cu I/E fiind specializat, în cadrul sistemului, pe operaţii de I/E.

Fiecare μS este prevăzut cu magistrală proprie, dar cu posibilităţi de interconectare a sistemelor. Comunicaţia între procesoare se face prin intermediul alocării unui spaţiu de memorie comună pentru cele două μS (fig.5.4). Blocul de inter-

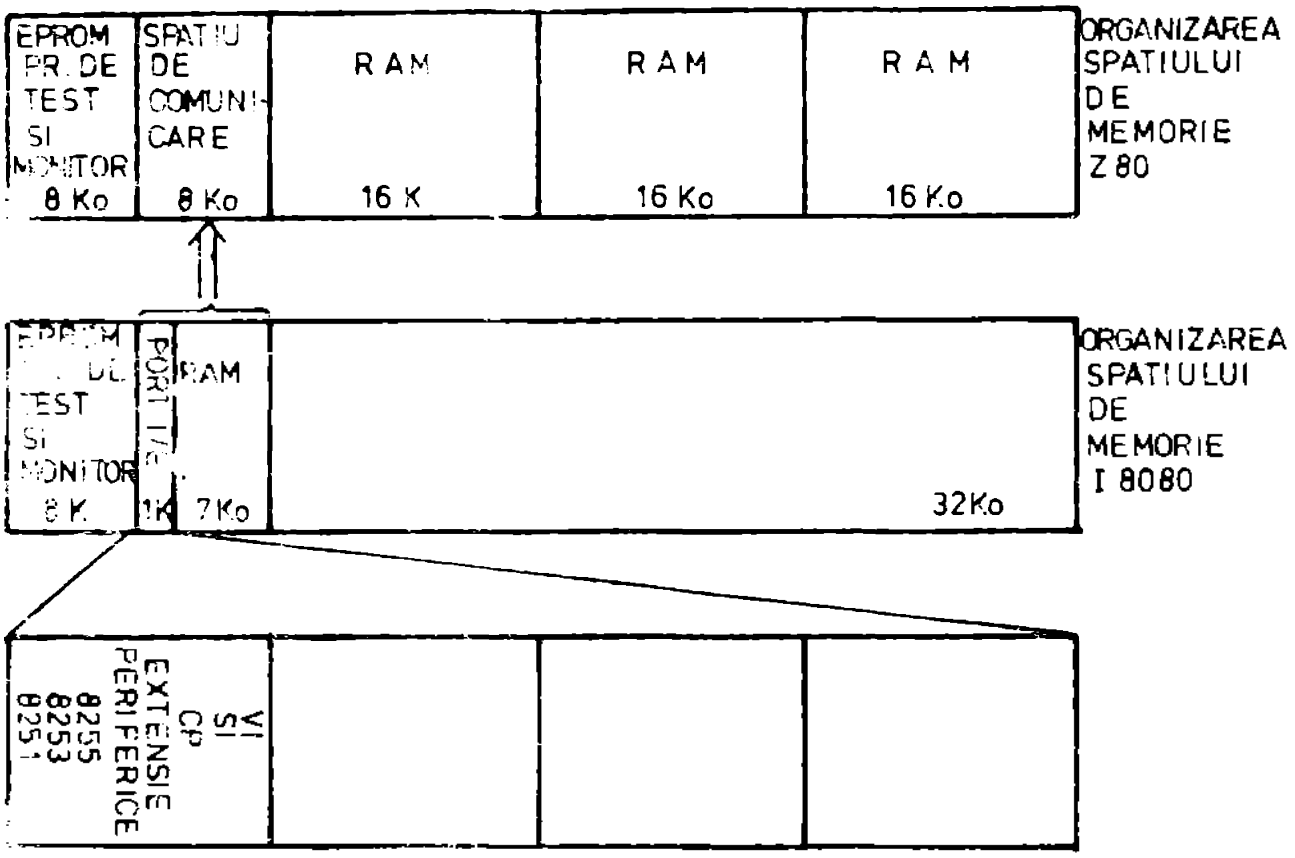


Fig.5.4

conectare a microsistemelor realizează cuplarea magistralelor sistemului Z80 (magistrala de adrese, magistrala de date şi cea de comenzi) la magistralele sistemului 8080, permiţînd prin aceasta accesul microprocesorului Z80 la toate resursele sistemului cu 8080; cu excepţia blocului de memorie în care este rezident sistemul de operare pentru sistemul cu 8080.

La microsistemul realizat cu microprocesorul Z80 memoria este alocată astfel încît programele de test şi monitorul să înceapă de la adresa 0000H. Tinînd cont de acest fapt, primii 8 KO din memoria sistemului este ocupată de o memorie fixă EPROM. Următorii 8 KO vor fi folosiţi ca spaţiu de comunicare între cele două sisteme.

Programele ce permit lucru cu resursele sistemului 8086, (memoria RAM, dispozitivele periferice, adresate ca locaţii de memorie), în cazul conectării magistralelor microsistemelor, sînt rezidente tot în memoria EPROM.

Programul de test este conceput să testeze microprocesorul Z 80, blocul de decodificare şi selecţie, memoria EPROM-monitor. Testarea se va face în colaborare cu unitatea u_1.

In conformitate cu [206] testarea unui microprocesor (fără componente externe) se face prin aplicarea unor stimuli determinaţi (instrucţii, în esenţă) şi evaluarea se face pe baza răspunsului la aceşti stimuli de către un bloc de evaluare/ decizie. Aplicarea stimulilor se face într-o succesiune stabilită de necesitatea verificării unor funcţii ale microprocesorului (decodificarea registrelor, trasferul de date, decodificarea instrucţiilor, prelucrarea datelor, etc.).

Deşi oferă o capacitate bună de detecţie a defectelor, metoda prezintă următoarele dezavantaje: se testează doar microprocesorul, introdus într-un testor şi nu în schema de aplicaţie; este necesară prezenţa unui sistem de generare a stimulilor de test, evaluarea răspunsurilor şi decizie, deci un testor autonom; volumul de memorie ocupat de către programele de test este mare (în [206] se indică 1 KO doar pentru verificarea decodificării registrelor).

Din aceste motive, pentru sistemul biprocesor s-a ales varianta de testare a setului de instrucţii ale microprocesorului pe baza unui program de test ales convenabil, pentru a asigura o testare exhaustivă a setului de instrucţii, pe baza modelului propus în [206, 212], în care generarea stimulilor de test se face de către însuşi blocul de testat prin memoria EPROM, iar evaluarea răspunsurilor se face fie printr-o metodă de comprimare (analiza de semnături) fie prin compararea rezultatului final cu o valoare determinată. Decizia se realizează de către un bloc activat de rezultatul comparării.

O astfel de testare asigură atît verificarea microprocesorului dar şi a memoriei EPROM, a magistralei de date şi adrese.

Dacă în cadrul programului de test AS găseşte că microprocesorul Z 80 este defect, elementul de decizie întrerupe executarea programului de test şi se trece la executarea programului de testare a următoarei unităţi (8080).

După verificarea funcţionării celor trei unităţi se poate determina unitatea defectă. In cazul că Z 80 este defect se întrerupe funcţionarea sistemului, dacă Z 80 a fost găsit că funcţionează corect se va trece la verificarea celorlalte blocuri din sistem, conform ordinogramei din fig.3.3.

Dacă în urma verificării blocurilor sistemului, acestea sînt găsite corecte, atunci se poate intra în monitorul microsistemului cu Z 80.

Sistemul realizat cu 8080 are rolul de procesor de intrare/ieşire ce va recepţiona caracterele alfa numerice de la un terminal (DAF); decodifică comenzile. Transmiterea comenzilor de la 8080 spre sistemul principal are loc prin intermediul blocului VI (vector de întrerupere), adresat de către 8080 în maniera obignuită şi citită de către Z 80 în momentul recunoaşterii cererii de întrerupere, conform celor specificate la descrierea blocului VI.

In afara rutinelor de tratare a operaţiilor de I/E, monitorul 8080 mai cuprinde un program de test şi partea de comandă care realizează prelucrarea primară a comenzilor (eliminarea bitului de paritate din cod şi transformarea în index pentru tabelul de adrese al sistemului Z 80, prin înmulţire cu 2, realizîndu-se astfel adrese multiple de 2). Pentru monitorul sistemului cu 8080 s-au alocat 8 KO de memorie, iar în continuare 8 KO pentru memoria RAM şi dispozitive periferice ce se va suprapune peste spaţiu de comunicare al lui Z 80, permiţînd acestuia accesul la resursele sistemului cu 8080 (adresele fizice ale microsistemului 8080 se vor intercala în spaţiu de adrese fizice ale microsistemului Z 80). Folosirea resurselor μS-8080 de către μS-Z 80 presupune invalidarea memoriei EPROM-ului 8080 (monitorul 8080) fapt ce se poate realiza prin condiţionarea răspunsului microprocesorului 8080 la o cerere de HOLD, semnalul HLDA, cu semnalul de selecţie a memoriei EPROM-8080.

Pentru a ugura dialogul între cele două microsisteme, în momentul interconectării, semnalele de comandă a μS-8080 s-au făcut compatibile cu semnalele de comandă a μP-Z 80.

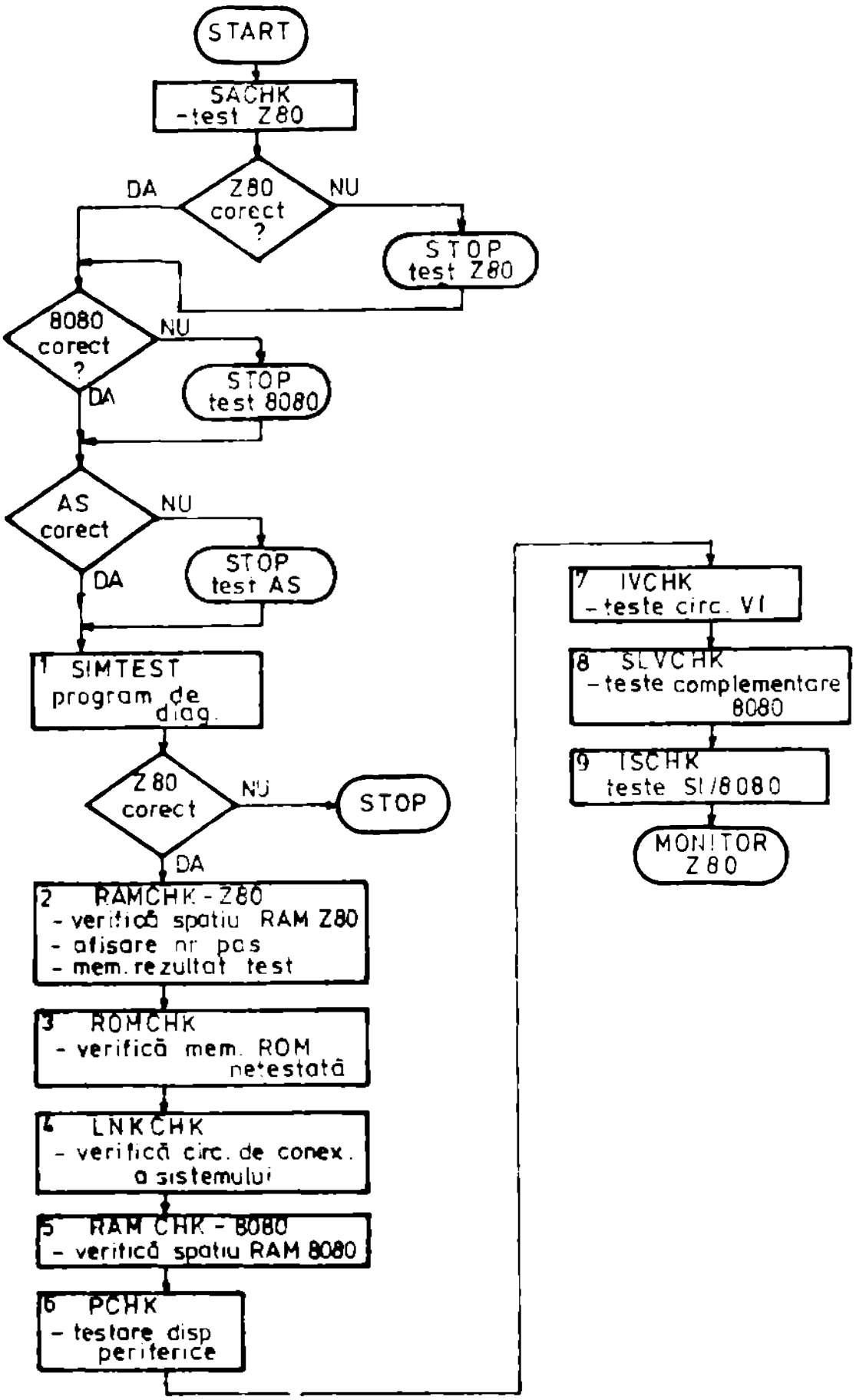


Fig.3.5

Echipamentele periferice comunică cu sistemul 8080, prin intermediul interfeţelor serie şi paralelă.

Sistemul de întreruperi la μS-8080, specializat pe operaţii de intrări/ieşiri, este format din întreruperile provenite de la dispozitivele periferice şi de la μS-Z 80. In cazul existenţei unei defecţiuni numai la sistemul de întreruperi, ce este semnalizată de către Z 80 prin testele făcute asupra acestuia, se inactivează sistemul de întreruperi, rămînd ca 8080 să lucreze cu dispozitivele periferice numai prin program, ceea ce scad performanţele sistemului.

5.4.2. Interconectarea microsistemelor.

5.4.2.1. Schema de conectare a magistralelor.

Această parte a blocului de interconectare a microsistemelor realizează cuplarea magistralelor sistemului Z 80 (magistrala de adrese, de date şi cea de comenzi) la magistralele sistemului 8080, permiţînd prin aceasta accesul microprocesorului Z 80 la toate resursele sistemului cu 8080 (cu excepţia blocului de memorie ce conţine monitorul acestuia).

In acest sens microsistemul cu Z 80 are acces la memoria RAM a μS-8080, atît la scriere cît şi la citire. De asemenea dispozitivele periferice sînt văzute ca locaţii de memorie.

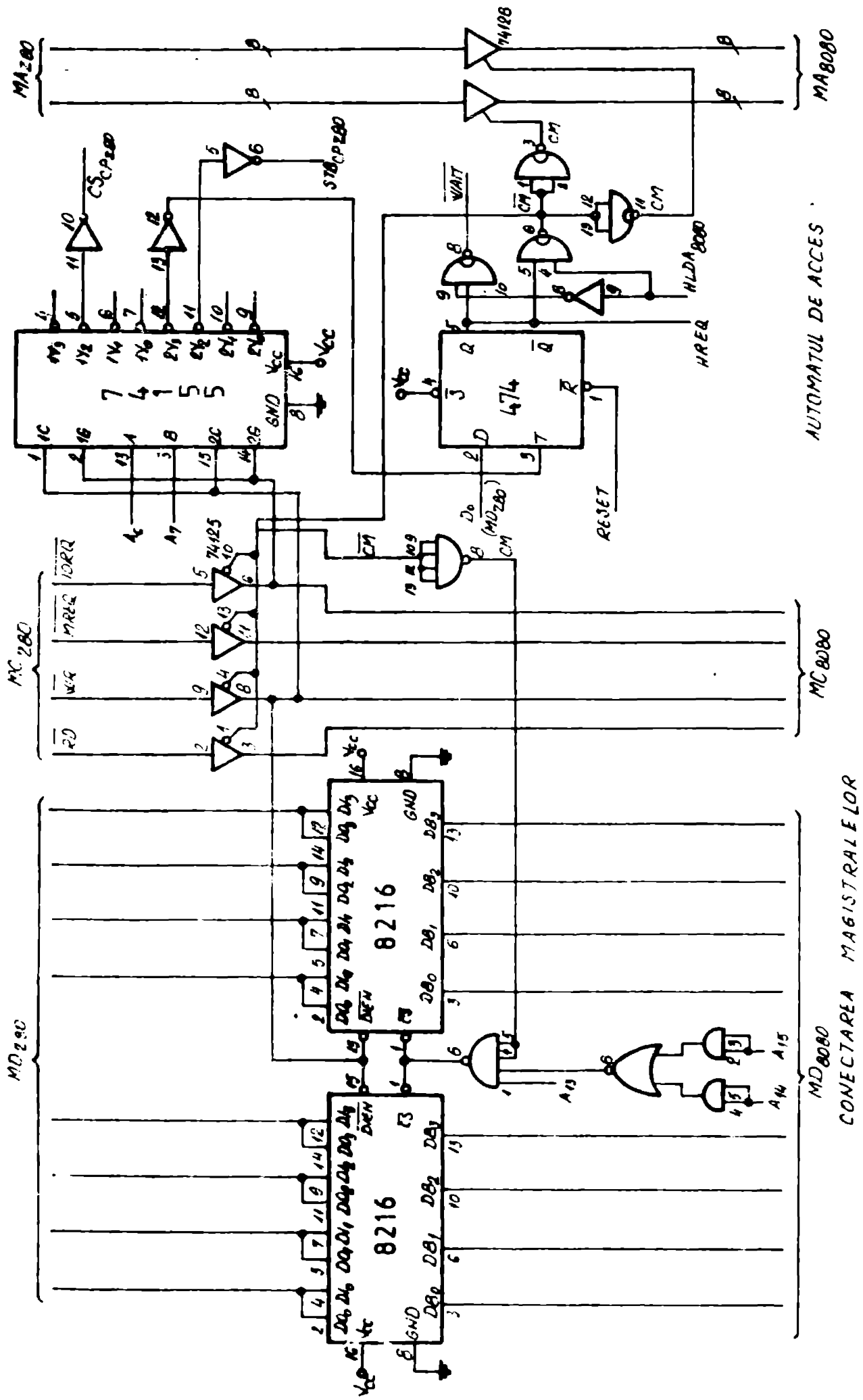
Pentru a asigura o conectare simplă a celor două sisteme, la proiectarea sistemului cu 8080 s-a urmărit generarea unor semnale de comandă compatibile cu Z 80 [175].

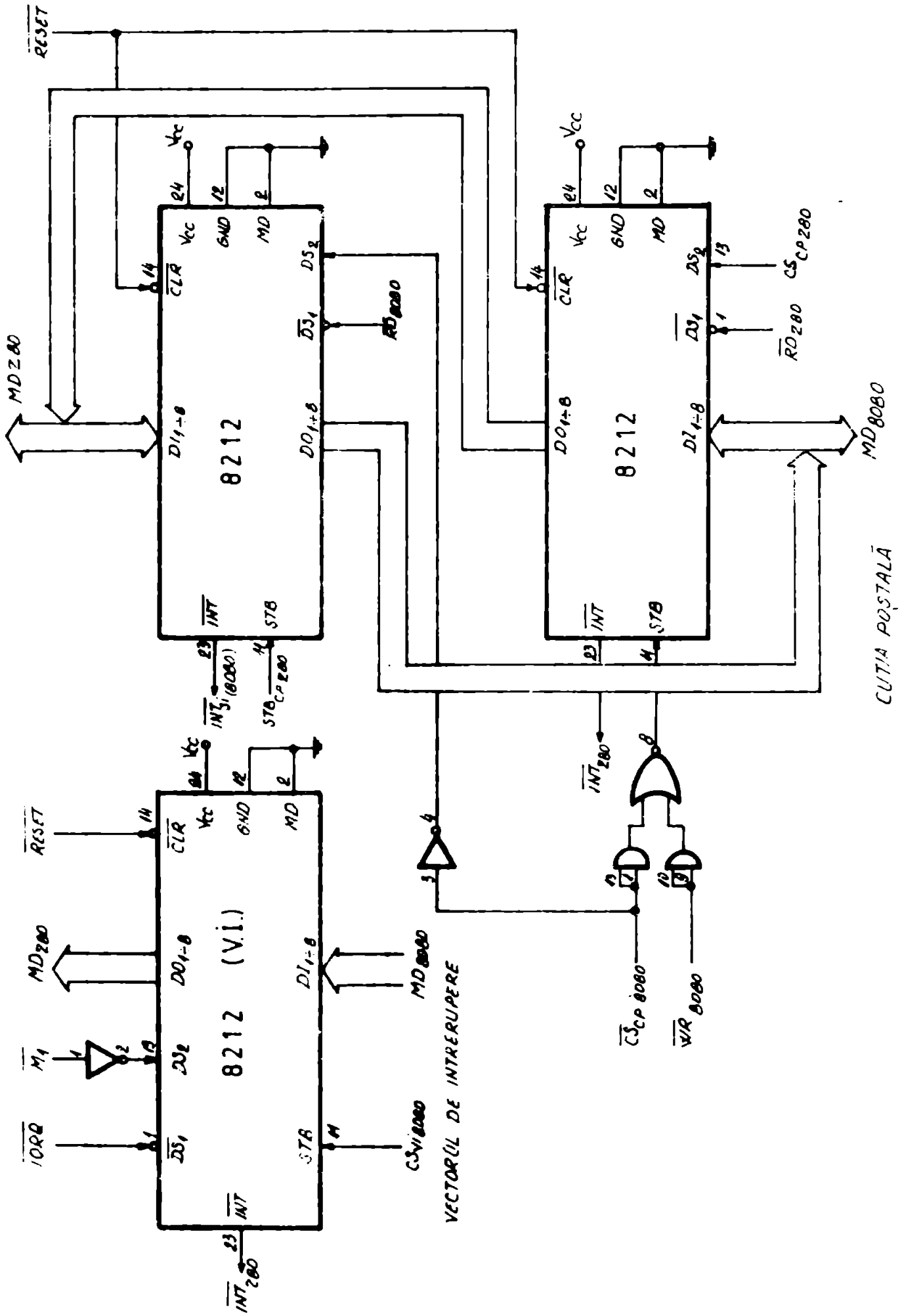
Conectarea semnalelor magistralei de adrese şi magistralei de comenzi se realizează într-un sens, adică de la sistemul cu Z 80 spre sistemul cu 8080 prin circuite tampon, comandate de automatul de acces care va fi prezentat în paragraful următor.

Conectarea magistralei de date se face bidirecţional fiind comandată tot de automatul de acces.

Conectarea magistralelor se face din punct de vedere logic într-un singur sens; de la microsistemul cu Z 80 spre microsistemul cu 8080, sub controlul microprocesorului Z 80 prin automatul de acces.

Nu s-a prevăzut legătura bidirecţională din punct de vedere logic între magistrale din mai multe motive:


MA280
MA8080
74128
CS CPU280
STB CPU280
WAIT
CM
HLDA 8080
HREQ
AUTOMATUL DE ACCES
7415125
8216
474
RESET
MC 280
MC 8080
MD 280
MD 8080
CONECTAREA MAGISTRALELOR


RESET
MDZ80
8212
8212
8212 (V.I.)
VECTORUL DE INTRERUPERE
CUTIA POSTALĂ

1. Automatul de acces s-ar fi complicat foarte mult, prin necesitatea stabilirii unei discipline de priorităţi.

2. Accesul lui 8080 la sistemul cu Z 80 nu se justifică din următoarele motive:

- Sistemul cu 8080, fiind specializat doar pe funcţia de procesor de intrare/ieşire, accesul la sistemul Z 80 nu aduce nici un cîştig din punct de vedere al vitezei de transfer a informaţiei din tamponul lui 8080 spre Z 80 sau invers, deoarece Z 80 dispune de instrucţii de transfer pe blocuri care sînt mult mai puternice ;

- Sistemul cu 8080 nu poate executa programe scrise în cod-obiect pentru Z 80, deci nu ar putea executa programul monitor a lui Z 80, iar scrierea unui monitor special pentru 8080 nu s-ar justifica.

3. Pentru simplificarea schemei s-a realizat reîmprospătarea memoriei dinamice a lui Z 80 doar cu semnale de comandă ale acestuia. Dacă s-ar fi adoptat şi soluţia conectării sistemului cu 8080 la sistemul cu Z 80, ar fi trebuit realizat şi un bloc de reîmprospătare a memoriei dinamice, deoarece sistemul cu 8080 nu poate asigura semnalele necesare reîmprospătării memoriei RAM dinamice.

Scopul pentru care a fost adoptată modul de conectare propus, a celor două sisteme, prin controlul întregului sistem de către microprocesorul Z 80, este creşterea disponibilităţii sistemului, deoarece la defectarea microprocesorului 8080, operaţiile lui să fie preluate de μP-Z 80, ceea ce permite ca sistemul însă să mai poată funcţiona dar cu performanţe mai reduse. Se realizează în acest fel o reconfigurare a structurii sistemului. In plus din punct de vedere al testării procesorului 8080 şi a dispozitivelor periferice asociate acestuia este de dorit existenţa facilităţilor de conectare a sistemului în sensul prezentat.

Prin modul de conectare realizat, rezultă că se acordă o mare importanţă procesorului Z 80, fiindu-i atribuit statut de MASTER în sistem (fig.3.6).

Trebuie făcută observaţia că prin cuplarea la sistemul cu Z 80 a sistemului cu 8080 se realizează practic o dublare a spaţiului de memorie care poate fi adresat. Pentru

aceasta s-a urmărit distribuirea adreselor fizice de memorie şi dispozitive periferice ale fiecărui sistem astfel încît spaţiul total să nu depăşească posibilitatea de adresare a microprocesorului Z 80. Deci adresele fizice ale sistemului cu 8080 se vor intercala în spaţiu de adrese fizice ale sistemului cu Z 80. Practic, din punct de vedere al programatorului, nu este necesară o comutare a adreselor de pe un sistem pe altul după ce s-a dat comanda de conectare a magistralelor [175].

5.4.2.2. Automatul de acces (AAC)

Funcţiile acestuia sînt:

- recepţionarea unei cereri de conexiune a magistralelor din partea lui Z 80 ;
- transmiterea acestei cereri spre procesorul 8080 ;
- ţinerea în aşteptare a procesorului Z 80 pînă la achitarea cererii de către microprocesorul 8080 ;
- activarea semnalului de comandă pentru circuitele de conectare a magistralelor ;
- recepţionarea cererii de separare a magistralelor de la Z 80.

Aceste funcţii se realizează simplu datorită variantei de conectare adoptată, într-un singur sens (*fig.*5.6)[175].

5.4.2.3. Cutia poştală (CP)

Cutia poştală a fost introdusă din motivul creării unei căi de comunicaţie rapidă între cele două microsisteme pentru cantităţi de informaţie redusă, cînd nu s-ar justifica procedura de conectare a sistemelor prin intermediul circuitelor de comutare a magistralelor.

Informaţiile vehiculate de acest bloc vor fi în special comenzi pentru microsistemul 8080 de la μp-Z 80 şi comenzi sau date scurte: 1,2,3 octeţi pentru Z 80. Prin comenzi înţelegîndu-se caracterele funcţionale ale monitorului provenite de la consola sistemului.

Cutia poştală asigură totodată, o cale redondantă de cuplare a celor două sisteme, ceea ce permite o siguranţă sporită în acest domeniu.

Acest bloc cuprinde două posturi de intrare/ieşire para-

lele, conectate ca un registru tampon bidirectional, adică un port de ieşire dinspre sistemul 8080 spre sistemul Z 80, iar celălalt dinspre sistemul cu Z 80 spre sistemul cu 8080 (fig.5.7) [175].

5.4.2.4. Vectorul de întrerupere (VI)

Acest bloc reprezintă o cale şi mai rapidă de transmitere a unor comenzi standard către microsistemul cu Z 80 dinspre microprocesorul 8080 şi se bazează pe caracteristicile procesorului Z 80 de a lucru cu întreruperi vectorizate.

Pe scurt metoda presupune furnizarea, după acceptarea de către procesorul Z 80 a cererii de întrerupere, a unui octet care reprezintă jumătatea mai puţin semnificativ a unei adrese - vectorul de întrerupere. Acest octet va fi preluat de către Z 80 nu printr-o instrucţie de intrare cu adresa portului, ci printr-un ciclu special de recunoaşterea întreruperii specific procesorului Z 80.

In port va fi încărcat un octet de către microprocesorul 8080, după decodificarea unei intrări de la consolă sau în cazul trecerii procesorului 8080 printr-un punct în care trebuie să ceară asistenţa lui Z 80 ; în conformitate cu comanda primită de la consolă, sau starea programului procesorului 8080. Microprocesorul Z 80 va căuta într-un tabel octetul corespunzător care să permită apelarea directă a rutinei de tratare a situaţiei existente în conformitate cu procedura de tratare a întreruperilor procesorului Z 80 în modul 2. Deci portul este utilizat ca periferic cu posibilităţi de vectorizarea întreruperilor pentru Z 80 (fig.5.7) [175].

5.4.3. Unitatea redondantă.

Pentru realizarea unui sistem autotestabil sînt necesare cel puţin trei unităţi funcţionale care să îndeplinească condiţiile din ipoteza 2.1. Cum în cele două microsisteme, microprocesoarele şi memoriile fixe în care sînt depuse programele de test îndeplinesc aceste condiţii; pentru formarea unui SAT mai trebuie introdusă o unitate redondantă. Acest element redondent poate fi un comparator sau o unitate de comprimare a datelor.

In cazul utilizării unui bloc de comparare acesta tre-

buie să îndeplinească următoarele funcţii: să compare rezultatul obţinut în urma executării programului de test de către microprocesorul Z 80 cu un rezultat cunoscut; pe baza comparării să specifice dacă microprocesorul Z 80 şi memoria de test sînt defecte sau nu; să transmită la panoul de comandă, rezultatul comparării; să fie prevăzut cu un comparator de timp pentru a constata dacă programul de test se bucleazâ sau nu; după terminarea programului de test pentru Z 80 să genereze o comandă care să permită iniţierea programului de test pentru 8080.

In cadrul sistemului autotestabil realizat s-a adoptat varianta cu analizorul de semnături, pentru unitatea redondantă, care asigură o flexibilitate sporită în ceea ce priveşte localizarea defectelor la microsistemul Z 80.

Analizorul de semnături este realizat într-o variantă paralelă fiind plasat pe magistrala de date şi pe biţii cei mai semnificativi ai magistralei de adrese ai microsistemului Z 80

Strategia de testare a microsistemului Z 80 este următoarea:

- programele de test sînt rezidente în memoria fixă EPROM începînd de la adresa 00H ;

- La fiecare salt la adresa 00H, sau iniţiere de la panoul de comandă, programele de test sînt activate şi se trece la testarea automată a sistemului ;

- In cadrul analizorului de semnături este decodificată adresa 00H şi în conjuncţie cu semnalul MREQ este declanşat analizorul de semnături. Aceasta este şi semnalul de START pentru AS.

- Tactul AS este constituit de semnalul MREQ ;

- Semnalul de STOP este un semnal de adresă; ultima adresă a programului de test;

- Odată AS declanşat se va pune bistabilul B_{Z80} pe unu şi punerea lui pe zero se face numai dacă semnătura a fost cea corectă;

- Declanşarea AS va determina ca μP-8080 să fie adus în starea HOLD pînă la terminarea programului de test a microsistemului Z 80 ;

- După terminarea programului de test pentru Z 80 se va

face un salt la programul de test pentru 8080 ;

- In timpul executării programului de test pentru 8080 se vor trimite spre microsistemul Z 80, periodic, puncte de verificare. In cazul cá la momentele specificate gi/sau punctele de test nu sînt cele aşteptate se va poziţiona bistabilul B_{8080} pe unu ;

- La terminarea programului de test pentru 8080 se va iniţia un ciclu de autotest pentru AS [182]. Semnătura rezultată va fi trimisă spre microsistemul 8080 care o va compara cu semnătura bună, poziţionîndu-se corespunzător bistabilul B_{AS} ;

- Dacă toate elementele sindromului sînt pe unu sau sistemul cu Z 80 este defect, se opreşte întregul sistem. In caz contrar sistemul poate trece la îndeplinirea sarcinilor pentru care a fost conceput. Pentru remedierea defectelor din sistem este necesară întreruperea funcţionării lui.

Schema bloc a analizorului de semnături este prezentată în fig.5.8.

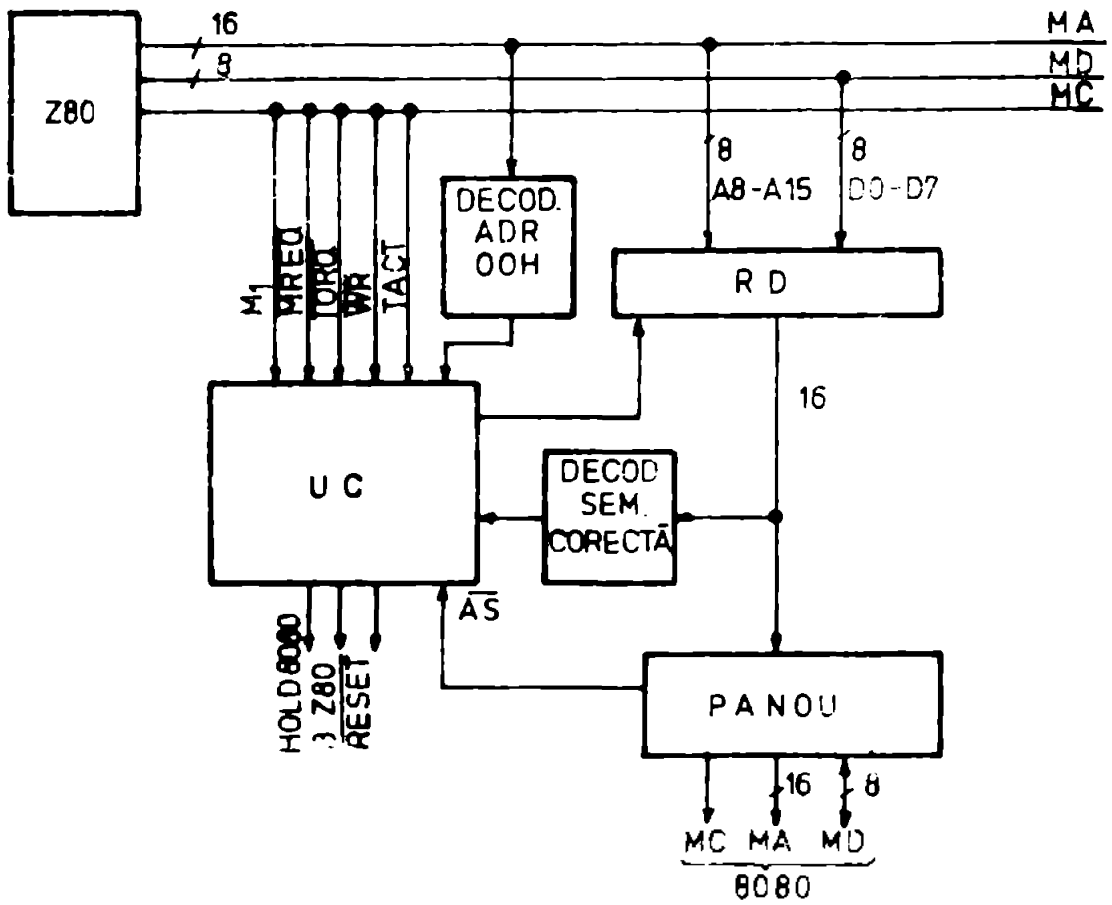


Fig.5.8

El se compune din următoarele blocuri:

a) Registru de deplasare (RD) este prevăzut cu posibilităţi de intrare serie şi paralelă.

Polinomul generator implementat este:[70, 152, 162]:

$$G(x) = x^{16} + x^{15} + x+1 \qquad (5.1)$$

Schema a fost sintetizată după modelul cu sumatoare incluse [2] .

b) Decodificatorul adresei OOH, ce va genera semnalul de START pentru AS.

c) Decodificatorul, "semnătură corectă".

d) unitatea de comandă a AS.

In AS s-a introdus şi un comparator de timp ce va fi declanşat de programul de test. Dacă progfamul de test se bucleazâ (există un defect) după cuanta de timp stabilită pentru el, AS va indica o eroare şi se întrerupe programul.

Toate ieşirile AS se vor afişa la panoul de comandă.

5.4.4. PROGRAMUL MONITOR

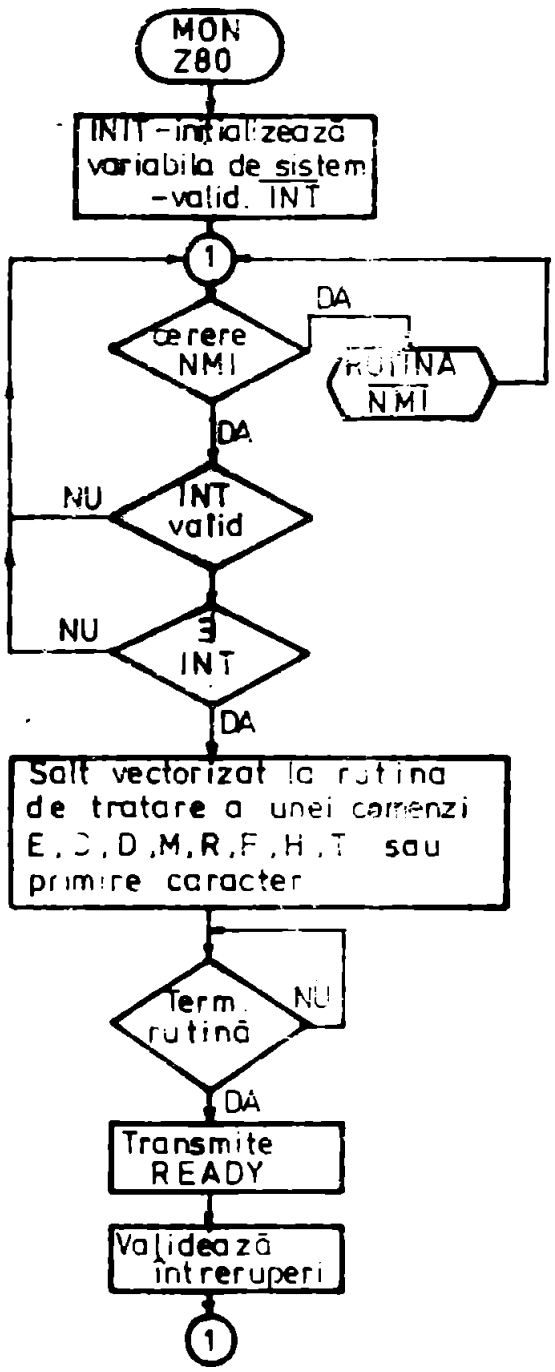
Programul monitor, în cadrul sistemului, are rolul de a îndeplini anumite funcţii ce vor fi introduse de la consola sistemului. Comenzile au fost alese după modelul celor existente la minicalculatoarele din formaţia SOLLIX M-18 [132].

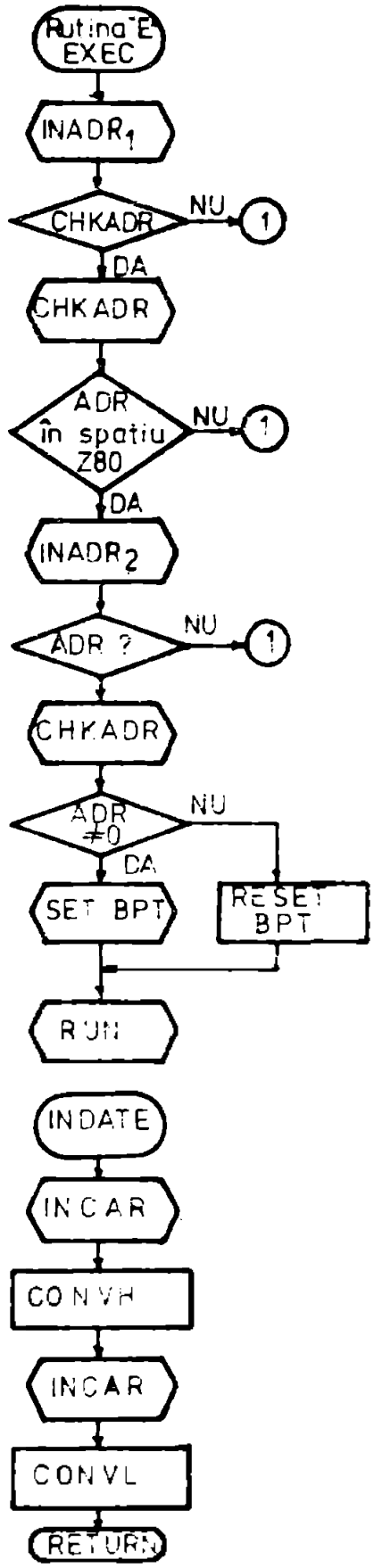
Particularitatea tratării acestor comenzi derivă din structura aleasă pentru sistem (bimicroprocesor), cu microprocesorul 8080 specializat ca procesor de I/E. De asemenea, posibilitatea de vectorizare a întreruperilor oferită de Z 80 permite ramificarea simplă spre rutinele specifice de tratare a comenzilor.

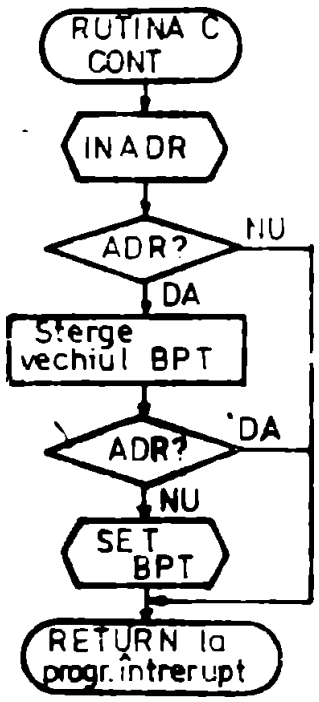
In esenţă, tratarea unei comenzi ar consta din etapele: recepţia caracterului (lor), care reprezintă comanda; decodificarea comenzii ; tratarea comenzii (activarea rutei specifice comenzii).

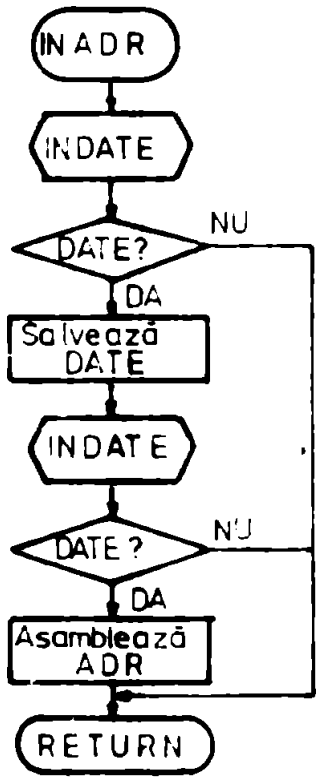
In cadrul sistemului biprocesor, primele două etape sînt executate în sistemele cu 8080, cu precizarea că etapa a doua este mult simplficată (se reduce la operaţii aritmetice simple, fără decizii) şi se suprapune cu operaţii executate în sistemul Z 80 (vectorizarea întreruperilor).

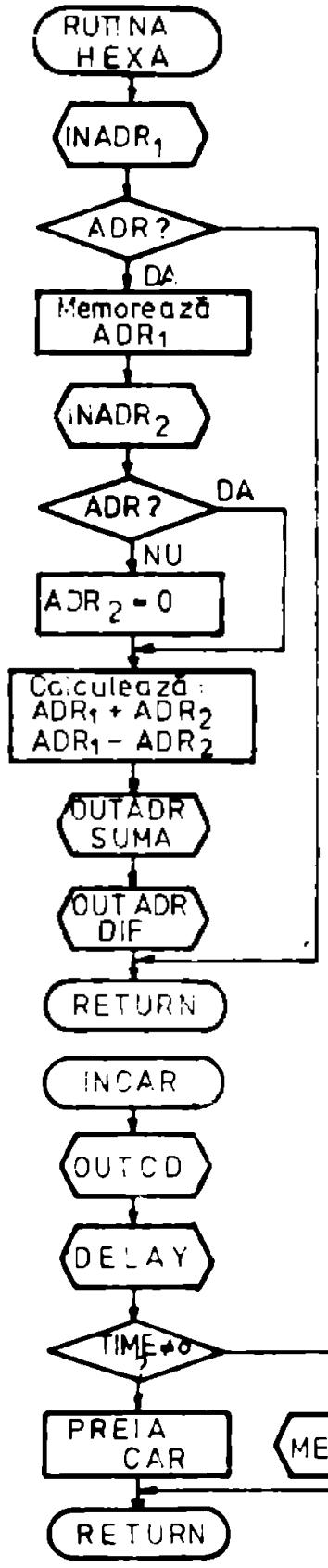
In sistemul principal, cu Z 80, se execută ramificarea

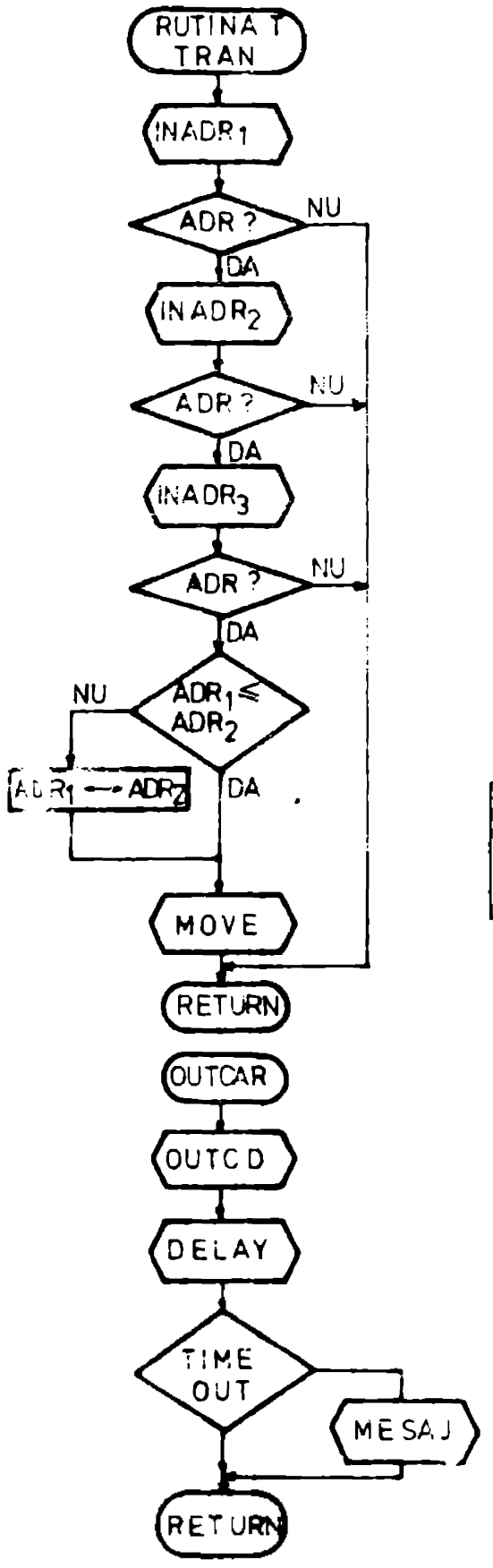

MON Z80
INIT - inițializează variabila de sistem -valid. INT
1
Cerere NMI
DA
RUTINA NMI
DA
NU
INT valid
NU
∃ INT
DA
Salt vectorizat la rutina de tratare a unei comenzi E, C, D, M, R, F, H, T sau primire caracter
Term. rutină
NU
DA
Transmite READY
Validează întreruperi
1

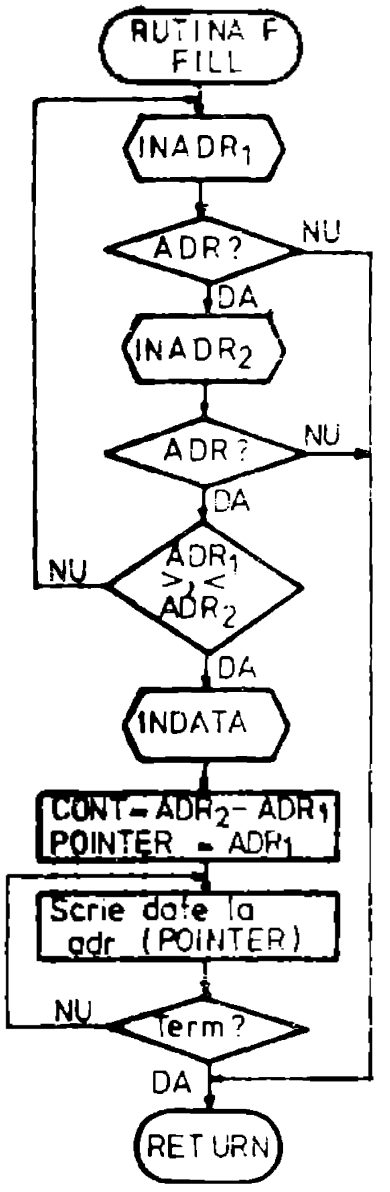

Rutina "E" EXEC
INADR1
CHKADR
NU
1
DA
CHKADR
ADR în spațiu Z80
NU
1
DA
INADR2
ADR ?
NU
1
CHKADR
ADR ≠0
NU
DA
SET BPT
RESET BPT
RUN
INDATE
INCAR
CONVH
INCAR
CONVL
RETURN

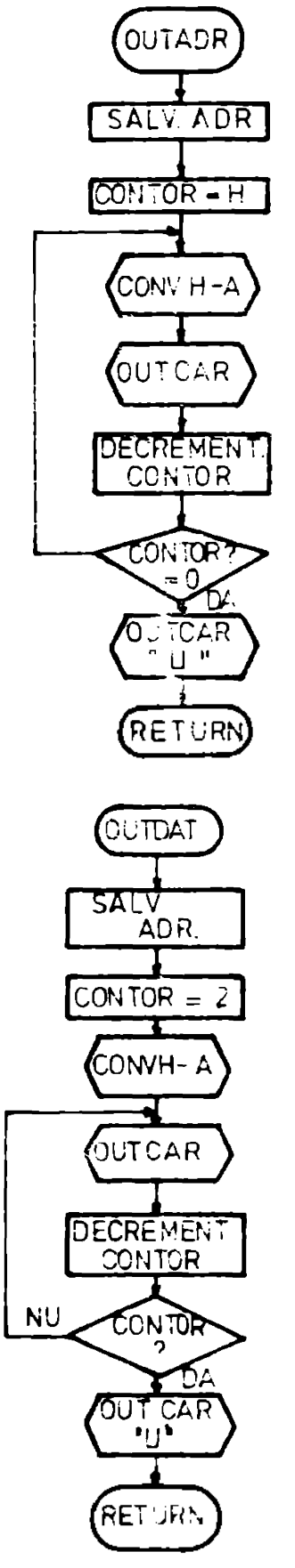

RUTINA C
CONT
INADR
ADR?
NU
DA
Sterge
vechiul BPT
ADR?
DA
NU
SET
BPT
RETURN la
progr. întrerupt

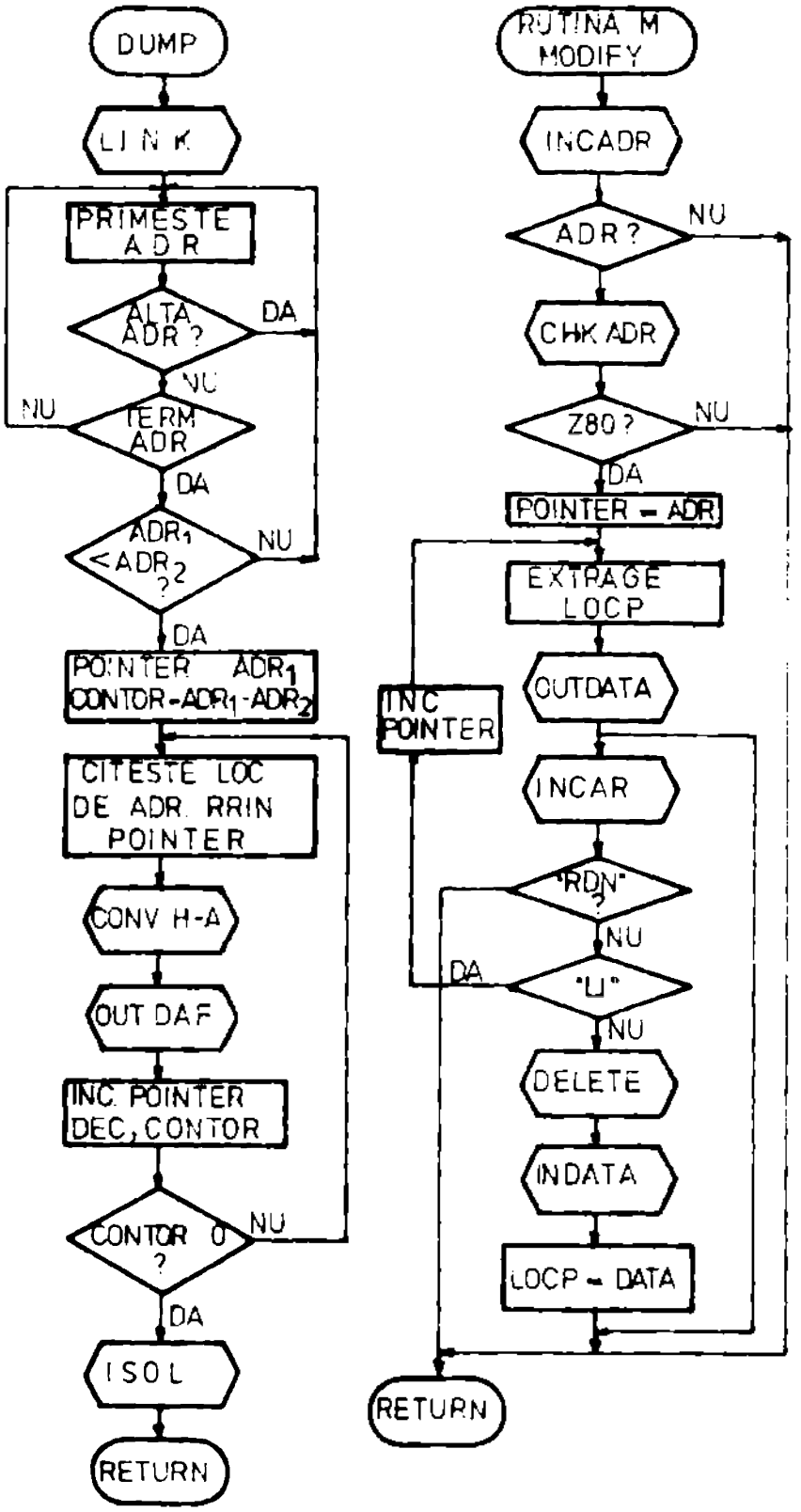

INADR
INDATE
DATE?
NU
DA
Salvează
DATE
INDATE
DATE ?
NU
DA
Asamblează
ADR
RETURN


RUTINA
HEXA
INADR1
ADR?
DA
Memorează
ADR1
INADR2
ADR?
DA
NU
ADR2 = 0
Calculează:
ADR1 + ADR2
ADR1 - ADR2
OUTADR
SUMA
OUT ADR
DIF
RETURN
INCAR
OUTCD
DELAY
TIME≠0
PREIA
CAR
MESAJ
RETURN


RUTINA T TRAN
INADR1
ADR ?
NU
DA
INADR2
ADR ?
NU
DA
INADR3
ADR ?
NU
DA
ADR1 ≤ ADR2
NU
DA
ADR1 ⟷ ADR2
MOVE
RETURN
OUTCAR
OUTCD
DELAY
TIME OUT
MESAJ
RETURN


RUTINA F FILL
INADR1
ADR ?
NU
DA
INADR2
ADR ?
NU
DA
ADR1 > < ADR2
NU
DA
INDATA
CONT = ADR2 - ADR1
POINTER = ADR1
Scrie date la adr (POINTER)
NU
Term ?
DA
RETURN


OUTADR
SALV. ADR
CONTOR = H
CONV H-A
OUTCAR
DECREMENT CONTOR
CONTOR ? = 0
DA
OUTCAR "U"
RETURN
OUTDAT
SALV ADR.
CONTOR = 2
CONVH-A
OUTCAR
DECREMENT CONTOR
NU
CONTOR ?
DA
OUTCAR "J"
RETURN


DUMP
LINK
PRIMESTE ADR
ALTA ADR ?
DA
NU
TERM ADR
NU
DA
ADR1 <ADR2 ?
NU
DA
POINTER ADR1
CONTOR-ADR1-ADR2
CITESTE LOC DE ADR. RRIN POINTER
CONV H-A
OUT DAF
INC. POINTER DEC, CONTOR
CONTOR 0 ?
NU
DA
ISOL
RETURN
RUTINA M MODIFY
INCADR
ADR ?
NU
CHK ADR
Z80 ?
NU
DA
POINTER = ADR
EXTRAGE LOCP
OUTDATA
INC POINTER
INCAR
"RDN" ?
NU
DA
"U"
NU
DELETE
INDATA
LOCP ← DATA
RETURN

spre rutinele specifice diferitelor comenzi (cu eliminarea comenzilor inexistente).

Transmiterea comenzilor de la 8080 spre sistemul principal are loc prin intermediul blocului notat "vector de întrerupere" (VI), adresat de către 8080 în maniera obişnuită şi citit de către Z 80 în momentul recunoaşterii cererii de întrerupere, conform celor specificate la descrierea blocului VI.

5.4.4.1. Comenzile sistemului

Comenzile alese pentru sistemul biprocesor sînt următoarele:

1. E XXXX YYYY ZZZZ <CR>

Acţiunea: lansează execuţia unui program începînd de la adresa XXXX ; dacă parametrul YYYY este prezent, fixează un punct de întrerupere la adresa YYYY ; dacă parametrul ZZZZ este prezent, limitează timpul de execuţie la valoarea ZZZZ * t, unde t este o cuantă de timp programabilă.

2. C XXXX <CR>

Acţiunea: continuă execuţia unui program oprit într-un punct de întrerupere; dacă parametrul XXXX este prezent, fixează un nou punct de întrerupere la adresa XXXX; dacă parametrul lipseşte, nu se modifică punctul de întrerupere, dar dacă adresa are valoarea 0000, se şterge punctul de întrerupere.

3. D XXXX YYYY <CR>

Acţiunea : realizează vidajul memoriei, între adresele XXXX şi YYYY.

4. M XXXX: mm - dd ... <CR>

Acţiunea: afişează conţinutul locaţiilor de memorie, începînd de la adresa XXXX; după fiecare octet afişat, se poate modifica valoarea aflată în memorie la adresa curentă, prin introducerea de la tastatură a simbolului " - ", urmat de data care se doreşte introdusă; dacă se doreşte trecerea la locaţia următoare, se introduce " " (blanc) afişîndu-se noua locaţie, ş.a.m.d.

5. R : AF - rrrr ... <CR>

Acţiunea: afişează valorile memorate în registrele microprocesorului Z 80 şi modifică, analog comenzii "M", aceste

valori cu excepţia registrelor IX, IY, I, R, PC, SP.

6. F, XXXX, YYYY dd <CR>

Acţiunea: iniţializează zona de memorie cuprinsă între adresele XXXX şi YYYY cu valoarea "dd".

7. H XXXX YYYY<CR>

Acţiunea: afişează suma, respectiv diferenţa celor două valori.

8. T XXXX YYYY ZZZZ

Acţiunea: transferă zona de memorie cuprinsă între adresele XXXX şi YYYY la zona care începe cu adresa ZZZZ.

9. P aaaa...<CR> ...<TRM>

Acţiunea: tipăreşte la imprimanta (SM4000) textul introdus de la consolă; comanda se încheie la întîlnirea caracterului "SM" ("sfîrşit mesaj"), transmis de către DAF, la apăsarea tastei "TRM", în modul caracter.

5.4.4.2. Ordinogramele monitorului Z 80.

In continuare se vor prezenta ordinogramele monitorului sistemului cu Z 80, după care s-a scris programul monitor. In figurile 5.9, 5.10, 5.11 şi 5.12 sînt date cele mai importante subrutine de programului monitor pentru Z 80.

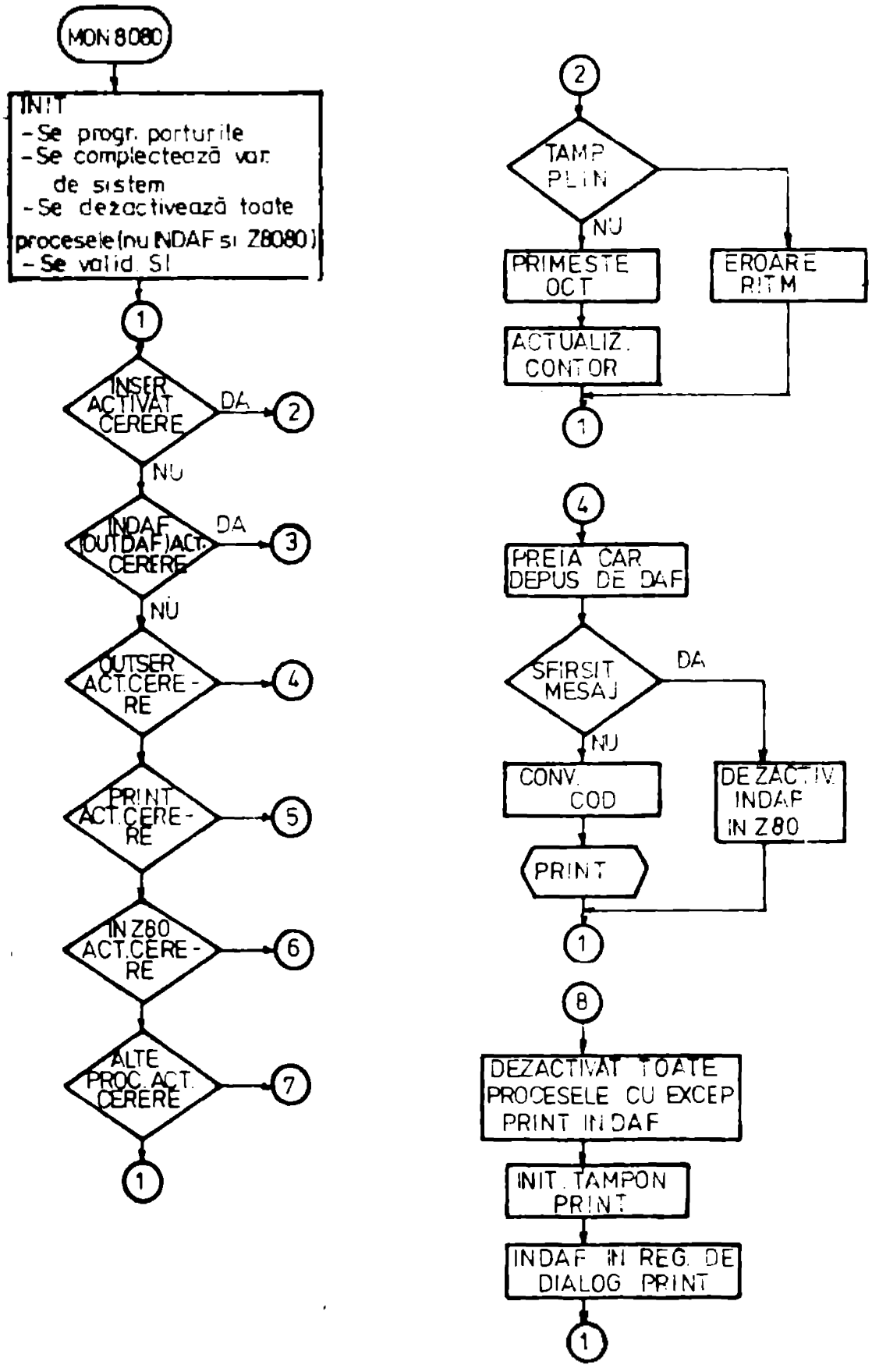
5.4.4.3. Programul monitor pentru 8080

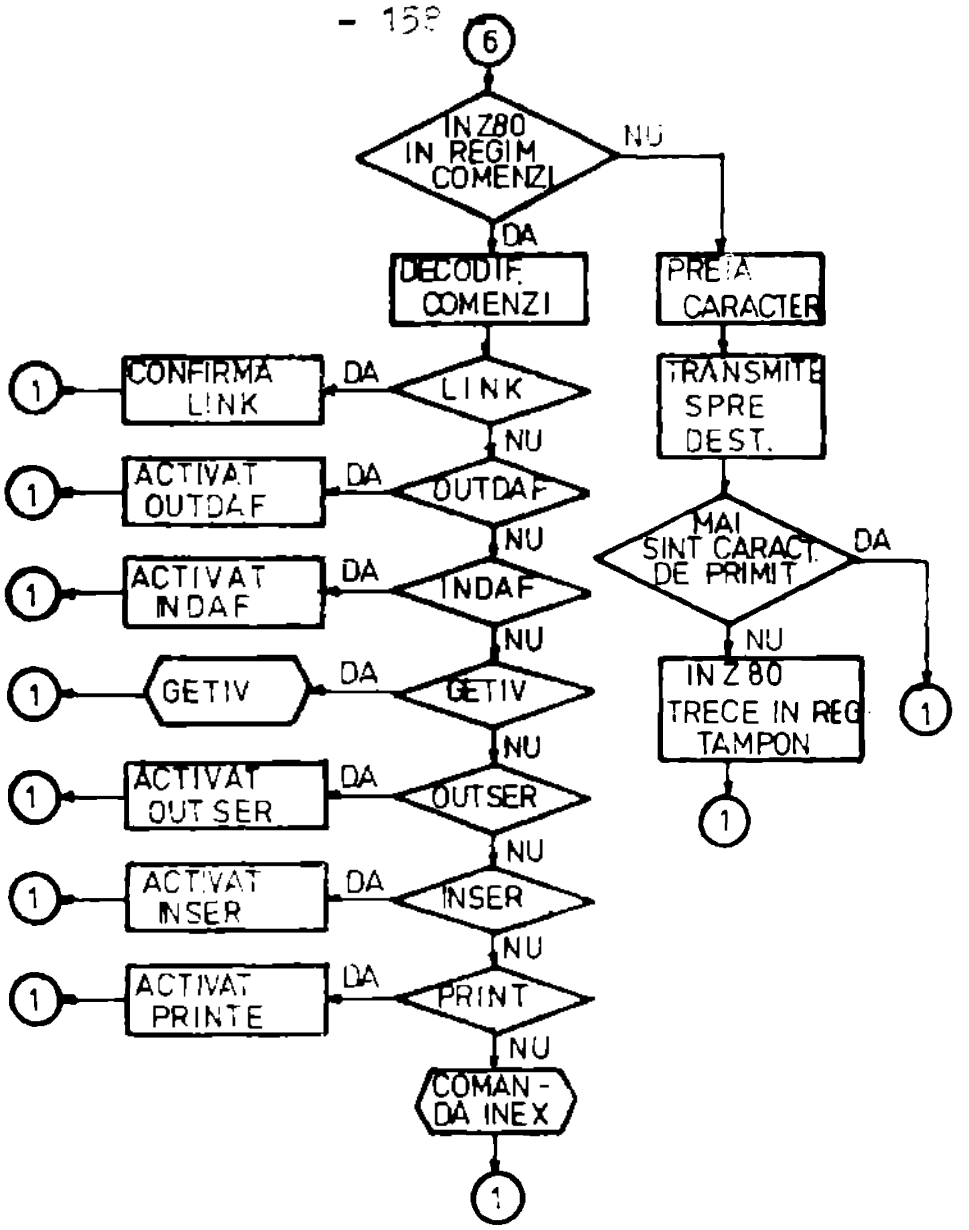
Monitorul 8080 cuprinde rutinele specifice pentru dialogul cu aceste periferice. Sînt conectate la sistem un dispozitiv de afişare DAF 1002, o maşină de scris SM4000 şi o linie serială asincronă (RS232C), fără periferic asociat.

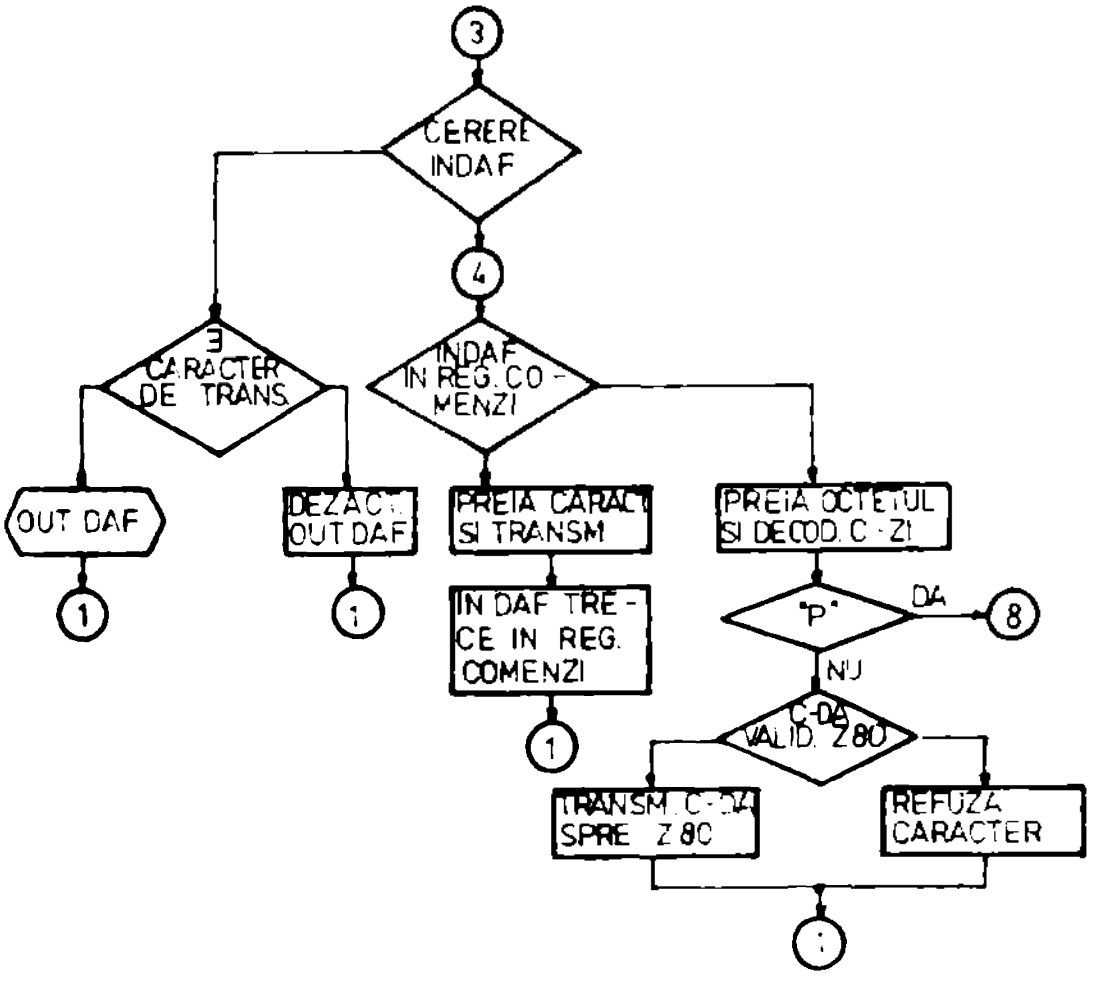
Ordinogramele prezentate în figurile 5.13, 5.14, 5.15 oferă datele necesare înţelegerii funcţionării acestui monitor.

5.5. Concluzii privind realizarea unui sistem biprocesor cu posibilităţi de autotestare

In cadrul lucrării s-a realizat un sistem biprocesor cu posibilităţi de autotestare. Partea aplicativă a urmărit verificarea aspectelor teoretice prezentate. In acest sens s-a realizat în cadrul unui sistem experimentarea atrei modalităţi de interconectare între două microsisteme: interconectarea prin intermediul unei memorii comune, interconectarea prin intermediul unei magistrale comune şi interco-


MON 8080
INIT
- Se progr. porturile
- Se complectează var. de sistem
- Se dezactivează toate procesele (nu NDAF si Z8080)
- Se valid. SI
1
INSER ACTIVAT CERERE
DA
2
NU
INDAF (OUTDAF) ACT. CERERE
DA
3
NU
OUTSER ACT.CERE-RE
4
PRINT ACT.CERE-RE
5
IN Z80 ACT.CERE-RE
6
ALTE PROC. ACT. CERERE
7
1
2
TAMP PLIN
NU
PRIMESTE OCT
ACTUALIZ. CONTOR
EROARE RITM
1
4
PREIA CAR DEPUS DE DAF
SFIRSIT MESAJ
DA
NU
CONV. COD
PRINT
DEZACTIV. INDAF IN Z80
1
8
DEZACTIVAT TOATE PROCESELE CU EXCEP PRINT INDAF
INIT. TAMPON PRINT
INDAF IN REG. DE DIALOG PRINT
1


6
INZ80 IN REGIM COMENZI
NU
DA
DECODIF. COMENZI
PREIA CARACTER
CONFIRMA LINK
DA
LINK
TRANSMITE SPRE DEST.
NU
ACTIVAT OUTDAF
DA
OUTDAF
MAI SINT CARACT. DE PRIMIT
DA
NU
ACTIVAT INDAF
DA
INDAF
NU
GETIV
DA
GETIV
INZ80 TRECE IN REG TAMPON
NU
ACTIVAT OUTSER
DA
OUTSER
NU
ACTIVAT INSER
DA
INSER
NU
ACTIVAT PRINTE
DA
PRINT
NU
COMAN - DA INEX
1


3
CERERE INDAF
4
3 CARACTER DE TRANS.
INDAF IN REG. CO - MENZI
OUT DAF
1
DEZACT. OUT DAF
1
PREIA CARACT. SI TRANSM
IN DAF TRE - CE IN REG. COMENZI
1
PREIA OCTETUL SI DECOD. C - ZI
"P"
DA
8
NU
C-DA VALID. Z80
TRANSM. C-DA SPRE Z80
REFUZA CARACTER
1

nectarea directă între microsisteme prin considerarea celui-lalt microsistem ca un dispozitiv periferic. Prin realizarea celor trei modalităţi de interconectare s-a urmărit şi o siguranţă sporită a sistemului de conexiuni.

Programele de test au urmărit doar punerea în evidenţă a defectării unei unităţi. Nu s-a urmărit o diagnoză la nivel de circuit integrat avîndu-se în vedere că o astfel de problemă a fost tratată în mai multe lucrări de referinţă [115, 77, 23] sau lucrări de specialitate [182, 185, 210] .

CAPITOLUL 6

CONCLUZII.

Scopul acestei lucrări este acela de a-şi aduce contribuţia, într-o manieră originală la concepereа unor arhitecturi multiprocesor autotestabile. In acest sens au fost urmărite următoarele obiective:

1. Analiza şi clasificarea metodelor de testare a unităţilor de prelucrare a datelor.
2. Determinarea condiţiilor necesare şi suficiente ca un sistem numeric să poată fi autotestabil.
3. Elaborarea unor modele de autodiagnoză a sistemelor de calcul.
4. Elaborarea unor metode de diagnoză a sistemelor autotestabile.
5. Verificarea pe un model experimental a unei arhitecturi multimicroprocesor cu posibilităţi de autotestare.
6. Simularea pe calculator a modelelor şi metodelor de diagnoză a sistemelor autotestabile.

6.1. Contribuţii originale

Pornind de la scopul şi obiectivul propus se pot scoate în evidenţă următoarele contribuţii originale ale lucrării:

1. Analiza comparativă a metodelor de testare a microprocesoarelor cît şi a sistemelor de calcul bazate pe microprocesoa-

re, cu punerea în evidenţă a dificultăţilor majore în elaborarea unor strategii de testare şi greutăţiile create în acest domeniu,de complexitatea circuitelor şi schemelor de testat. Din aceste motive la ora actuală acest domeniu tinde să rămînă în urma celorlalte laturi ale tehnicii de calcul ca:proiectarea şi realizarea echip mentelor şi sistemelor de operare pentru tehnica de calcul; dezvoltarea unor noi arhitecturi de sisteme de calcul; utilizarea calculatoarelor în economie şi industrie [206].

2. Pe baza teoriei grafurilor şi a modelului unui sistem autotestabil [130] s-a elaborat un suport teoretic privind determinarea condiţiilor necesare şi suficiente ca un sistem să poată fi autotestabil. In acest scop au fost enunţate nouă teoreme, o lemă , trei corolare şi două algoritme.

3. Punerea în evidenţă a unui defect într-un sistem autotestabil se face pe baza unui sindrom. Sindrom ce trebuie să fie specific unui defect din sistem. In acest sens s-au determinat condiţiile ca un sindrom să fie unic definit pe baza teoremei 2.1 şi a corolarului 2.1.

4. S-a introdus noţiunea de sistem autotestabil optim diagnozabil şi s-au determinat condiţiile ca un sistem să fie optim diagnozabil pe baza teoremei 2.3 şi corolarului 2.2.

5. S-a elaborat un algoritm (2.1) pentru identificarea SAToptim diagnozabile.

6. Pe baza modelului de diagnoză s-au obţinut relaţiile pentru determinarea numărului de sindroame şi de defecte simple şi multiple dintr-un sistem, în cazul apariţiei defectelor permanente.

7. S-a dezvoltat în continuare modelul de diagnoză a SAT şi pentru cazul că în sistem ar apărea defecte intermitente. Defectele intermitente aduc un grad sporit de complexitate în găsirea condiţiilor ca un sistem să poată fi autotestabil. In acest sens s-a stabilit o strategie de testare a sistemelor în prezenţa defectelor intermitente.

8.S-au enunţat condiţiile necesare şi suficiente ca un sistem să fie autotestabil în prezenţa defectelor intermi-

tente şi s-e elaborat un model de diagnoză. Modelul elaborat este superior celui prezentat în [114], fiind mai general. In plus s-au eliminat situaţiile de nedeterminare, ceea ce permite o diagnoză sigură. Modelul de diagnoză din [114] este valabil pentru unele structuri particulare de SAT. Teorema 2.4 şi 2.5 asigură o extindere a modelului de diagnoză elaborat pentru defecte permanente şi pentru cazul defectelor intermitente.

9. S-e elaborat un algoritm (2.2) de transformare a unui sistem diagnosticabil într-un sistem optim diagnosticabil. Prin aceasta sistemul autodiagnosticabil pentru defecte permanente devine autodiagnosticabil şi pentru defecte intermitente. Sindromul obţinut în acest fel este unic definit, chiar în prezenţa defectelor intermitente.

10. Pe baza modelului de diagnoză a SAT s-au determinat relaţiile între numărul de defecte intermitente şi permanente din sistem (teorema 2.6 şi 2.7).

11. Pornind de la modelele de diagnoză propuse s-a elaborat strategia de testare a SAT pentru situaţia când defectele permanente şi intermitente apar simultan. In acest mod s-a generalizat modelul de diagnoză a SAT şi pentru cazul unor defecte nespecificate (intermitente şi/sau permanente). Teorema 2.8 specifiind condiţia necesară şi suficientă ca un sistem să fie autotestabil în prezenţa oricărui tip de defect.

12. Determinarea relaţiilor între defecte în cadrul modelului generalizat.

13. Elaborarea unor programe de simulare a unor structuri de sisteme autotestabile pentru demonstrarea corectitudinii modelelor elaborate. Sindroamele obţinute prin programele de simulare sînt unic definite pentru fiecare caz de defect în parte.

Cele trei programe de simulare au fost realizate pentru structuri simetrice (SIMSIL), structuri asimetrice SIMASIL), cît şi pentru situaţia cînd în sistem sînt prezente defecte intermitente (SIMAIT).

14. Definirea conceptului de defect implicant invers, cu precizarea modului de utilizare în diagnoza SAT.

15. Elaborarea unei metode de diagnoză a defectelor prin utilizarea conceptului de defect implicant şi defect implicant invers (algoritmul 3.1, teorema 3.1).

16. Definirea matricei de legătură și a matricei de incidență în SAT cu precizarea elementelor caracteristice.

17. Elaborarea unui algoritm de obținere a matricei de incidență pe baza matricei de legătură (algoritmul 3.2).

18. Elaborarea unei metode de diagnoză a sistemelor autotestabile plecînd de la matricea de incidență, atît pentru structuri SAT simetrice cît și asimetrice. Se demonstrează cu ajutorul teoremelor 3.2, 3.3 și 3.4 că algoritmul 3.2 împreună cu algoritmul 3.3 permit localizarea unităților defecte în sistemele autotestabile.

19. Cu ajutorul a trei programe de simulare s-au verificat algoritmii de diagnoză pe calculator. Programele au fost scrise în FORTRAN. Algoritmii de diagnoză au fost verificați pe diferite structuri de SAT simulate pe calculator.

20. Analiza comparativă a celor mai cunoscute metode de diagnoză a schemelor secvențiale cu scoaterea în evidență a elementelor caracteristice.

21. Calificarea și analiza comparativă a metodelor de proiectare pentru îmbunătățirea testabilității pe baza datelor publicate în literatură. Clasificarea s-a făcut pe trei nivele structurale: la nivel de circuit integrat, la nivel de bloc și la nivel de sistem.

22. Studiul arhitecturilor sistemelor de calcul multiprocesor, abordat prin prisma cerințelor impuse de realizare a unor sisteme autotestabile și propunerea unor variante de arhitecturi multimicroprocesor cu posibilități de autotestare.

23. Realizarea unui sistem experimental biprocesor cu facilități de autotestare.

Referitor la structura propusă s-a găsit un optim între numărul de unități funcționale din sistem, numărul de unități defecte și numărul de interconexiuni între unități.

6.2. Valoarea aplicativă și direcții de dezvoltare viitoare

Aspectele prezentate în lucrare au fost valorificate în cadrul a 17 lucrări științifice publicate, o temă de contract de cercetare științifică, o inovație. Tot în acest domeniu autorul tezei este coautor a 2 lucrări propuse pentru a fi brevetate ca invenții.

Pentru elaborarea lucrării a fost utilizată o bibliografie ce cuprinde 230 titluri.

Pentru viitor se conturează însă noi perspective de valorificare ţinînd cont de interesul crescînd care se manifestă faţă de sistemele de calcul autotestabile, interes stimulat pe de o parte de proliferarea sistemelor de calcul bazat pe microprocesoare în cele mai diverse domenii, iar pe de altă parte de utilizarea în noi domenii de aplicaţie a sistemelor de calcul autotestabile sau tolerante la defecte.

În acest context, din multiplele direcţiilor posibile de dezvoltare a sistemelor de comandă numerică autotestabile, realizate prin structuri multimicroprocesor, se menţionează următoarele domenii mai importante : aeronautică, misiuni de cercetare spaţială, energia nucleară, sisteme de navigaţie şi comunicaţie, reactoare chimice, sisteme de ridicat, informatică şi robotică.

BIBLIOGRAFIE

1 N.CEAUSESCU - Raport la cel de-al XIII-lea Congres al Partidului Comunist Roman, Editura Politică, Bucureşti, 1984.

2 ANGHELOIU I. - Teoria Codurilor, In Editura militară, Bucureşti, 1972.

3 ARMSTRONG J.R., GRAY F.G. - "Fault Diagnosis in a Boolean n Cube Array of Microprocessors" in IEEE Trans.on Computer, vol.C-30, nr.8, aug.1981,p.587-596.

4 AGARWAL K. VINOD, FUNG S.F.A. - "Multiple Fault Testing of Large Circuits by Single Fault Test Set", IEEE Trans. on Computer, vol.C-30, nov.1981,p.855-865.

5 AGARWAL K.V., MASSON G.M. - "Generic Fault Characterizations for Table Look-up Coverage Bounding", IEEE Trans. Computer, vol.C-24, apr.1980, p.288-299.

6 ALLAN F.J., KAMEDA T., TOIDA S. - "An Approach to the Diagnosability Analysis of [illegible] Systems", IEEE Trans.Computer, vol.C-24, oct.1975, p.1040-1042.

7 ANDERSON R.E. - "Test methods change to meet complex demands", Electronics, 15 apr.1976,p.125-126.

8 AGARWAL V.K., MASSON G.M. - "Recursive Coverage Projections of Test Sets", IEEE Trans.on Computer Vol.C-28, p.865-870, nov.1979.

9 ANDERSON R.E. - "Testing Glossary Reflects Industry usage" Electronics, 27 mai, 1976, p.116-121.

10 ARNOLD J.F. - "The concept of coverage and its Effect on the Reliability Model of a Repairable system", IEEE Trans.on Computer, vol.C-22, mart.1973, p.251-255.

11 ANDERSON D.A., METZE G. - "Design of Self-Checking check Circuits for m-out-of-n Codes", IEEE Trans.on Computer vol.C-22, mart.1973, p.263-269.

12 AVIZIENIS A. - "Arithmetic Codes:Cost and Effectiveness Studies for Applications in Digital Systems Design",IEEE Trans.on Computer,vol.C-20,nov.1971,p.1322-133.

13 ABRAHAM J.A., SIEWIOREK D.P. - "An Algorithm for the Accurate Realiability Evoluation of Triple Modular Redundancy Netwerks", IEEE Trans.on Computer,vol.C-23,iulie 1974, p.682-692.

14 ARMSTRONG D.B.- "A Deductiv Method for Simulating Faults in Logic Circuits, IEEE Trans.on Computer, vol.21,nr.5, mai, 1972, p.464-471.

15 ASPINALL D. - "The microprocessor and its Application"- Cambridge University Press, "Cambridge 1980.

16 AVIZIENIS A. - "Fault tolerant Systems", IEEE Trans.on Comruter, vol.C-25, dec.1976,p.1304-1311.

17 AVIZIENIS A. - "The STAR (self-testing and repairing) com puter. An inveetigation of the theory and practice of fault tolerant computer design", IEEE Trans.on Computer, vol.C-20, nov.1971, p.1312-1321.

18 BARZILAI Z., SAVIR J., MARKOWSKY G., SMITS G., MERLIN - "The Weighted Syndrome Sums Approach to VLSI Testing",in IEEE Trans.on Computer, vol.C-30,nr.12,dec.1981,p.996-1000.

19 BREUER M.A., - "Testing for Intermittent Faults in Digital Circuits" in IEEE Trans.Computer, vol.C-32,mar.1973,p.241-246.

20 BERNHARD R.,-"Computers 1000 times faster than today's supercomputers would benefit vital scrientific applications" in IEEE Spectrum, iul.1982,p.26-31.

21 BUTLER T.JON, .- "Speed-Efficiency-Complexity Tradeoffs in Universal Diagnosis Algorithm" in IEEE Trans.on Computer, vol.C-30, aug.1981, p.590-596.

22 BARSI F., GRANDONI F., MAESTRINI P.-;"A Theory of diagnosability of digital systems", IEEE Trans.Computer vol.C-25, iun.1976, p.585-593.

23 BREUER M.A., FRIEDMAN A.D.- DIAGNOSIS AND RELIABLE DESIGN OF DIGITAL SYSTEMS", Woodland Hills CA: Computer Science Press,1976.

24 BENNETTS R.G.,SCOTT R.V.-"Recent Developments in Theory and Practice of Testable Logic Design",Computer,iun.1976, p.47-63.

25 BLASAU J.-"Systèmes l'aide au diagnostic" in Electronique Industrielle, nr.95, 1985, pag.43-49.

26 BURGLA P.-"System Integration and Testing with Microprocessors-II", in Microprocessors:Fundamentals and Application, IEEE Press, 1977,p.197-201.

27 CHAMAL D.P.,LEU J.S.-"Dynamic Accessibility Testing and Path Length Optimization of Multistage Interconnection Networks" in IEEE Trans.on Computers, Vol.C-34,mart.1985, p.255-267.

28. GARCIA-MOLINA H.KANT J.-"Evaluating Response Time in a Faulty Distributed Computing System",in IEEE Trans.on Computers,vol.C-34,feb.1985,p.101-110.

29 DAVIS N.J.,LEU M.T.Y.,SIEGEL H.J.-"Fault Location Techniques for Distributed Control Interconnection Networks" in IEEE Trans.on Computers,vol.C-34,oct.1985,p.902-910.

30 DAVAUT J.P.-"Simulation: de la Conception au test" in Electronique Industrielle,nr.95,1985,p.59-69.

31 DAHBURA A.T.,MASSON G.M.,YAGLE C.L.-"Self-Implicating Structures for Diagnosable Systems" in IEEE Trans.on Computers,vol.C-34,aug.1985,p.713-729.

32 DAVISON C.-"Pour le test de VLSI: informatique conciviale et temps réel" in Electronique Industrielle,nr.97,1985, p.79-84.

33 HAYES J.-"A Regular Fault-Tolerant Architecture for Interconnection Networks" in IEEE Trans.on Computers, vol.C-34,iulie 1985,p.677-688.

34 IYENGAR V.S.,KINNEY L.L.-"Concurrent Fault Detection in Microprogrammed control Units" in IEEE Trans.on Computers, vol.C-34,sept.1985,p.846-858.

35 BALTAC V.-"Optimizarea sistemelor de operare ale calculatoarelor numerice",Ed.Facla Timișoara,1974.

36 BALTAC V.-coordonator-"Calculatoarele electronice, grafica interactivă și prelucrarea imaginilor",Ed.Tehnică, Bucureşti,1985.

37 KHANCZYK H.,KUHAR A.-"An Approximation Algorithm for Diagnostic Test Scheduling in Multicomputer Systems" in IEEE Trans.on Comp.,sept.1985,p.869-872.

38 CHI-CHANG LIAW, STEPHEN Y.H.SU, KAMAIYA V.K.-"Test-Experiments for Detection and Location of Intermitent Faults in Sequential Circuits", in IEEE Trans.on Computers,vol.C-3o,nr.12,dec.1981,p.989-996.

39 KAWAOKA T.,TAKAHASHI Y.-"Test Procedure Optimization for Layered Protocol Implementations" IEEE Trans.on Computers vo.C-34,ian.1985,p.94-97.

4o Le CLUSKEY E.J., BOZORGUI-NESBAT S.-"Design for Autonomous Test",IEEE Trans.on Computers vol.C-3o,nov.1981, p.866-875.

41 CHIANG A.C.L.Le CASKILL R.-"Two new approaches simplify Testing of microprocessors",Electronics,22 ian.1976.

42 HALT C.S.,SMITH J.E.-"Self-Diagnosis in Distributed Systems" IEEE Trans.on Computers,vol.C-34,ian.1985, p.19-3o.

43 PANTANO G.-"Test in situ diagnostics assistes par systeme expert" in Electroniques Industrielle,nr.lol,1986,p.39-43.

44 CULLMAN G.-"CODURI DETECTOARE SI CORECTOARE DE ERORI", Editura Tehnica,Bucuresti,1972.

45 CHOUZET Y.-"Un microcalculateur a autodetection et autodiagnosis des fautes:Description,mesure de sa surete", notă tehnică,apr.1981,de Centre National de la Recherche Scientifique.

46 CATUNEANU V.M.,MIHALACHE A.-"BAZELE TEORETICE ALE FIABILITATII",Ed.Academiei R.S.R.Bucuresti,1983.

47 CHEN K.Y.,DAKIBI L.-"An Fault Identification in Diagnosable Systems",IEEE Trans.on Computers,vol.C-3o,iunie 1981,p.414-422.

48 CLIFF R.A.-"Acceptable Testing of VLSI Components which Contain Error Correctors",IEEE Trans.on Computers,vol.C-29,feb.198o,p.125-134.

49 COX H.-"A Remark on the Norminiality of Certain Kultiple Fault Detection Algorithms",IEEE Trans.on Computers,vol.

C-29,aug.1980,p.757-759.

50 CROUZET Y., LANDRAULT C.-"Design of Self-Checking MOS-LSI Circuits:Application to a Four-Bit Microprocessor" IEEE Trans.Computer,vol.C-30,iun.1980,p.532-537.

51 CATUNEANU V.M.,MARIOTTI E., CRIOORESCU M.-"Simularea funcționării rețelelor cu circuite logice",în Cercetări în tehnologie, electronică și fiabilitate,Editura D.și Pedagogică,București,1979,p.293-302.

52 CHU W.W.-"A Mathematical Model for Diagnosing System Failures",IEEE Trans.on Electron Computers,vol.EC-16,nr.3 iune 1967,p.327-331.

53 CATUNEANU V.M.,BACIVAROF I.C.-"Fiabilitatea sistemelor de telecomunicații",E.Militară,București,1985.

54 CHECHANU M., STRATULAT M., MIHAILESCU A.-"Prelucrarea analog-numerică a parametrilor sudurii automate"- A 4-a Conferință internațională de Sisteme Automate și Informatice în Industrie, mai 1981,București.

55 CHECHANU M., STRATULAT M., MIHAILESCU A., MOS I.C.-"Sisteme numerice pentru conducerea unor procese de sudare CNETEC, oct.1984,București.

56 CHANG H.Y.,CHAPPEL S.G.-"Deductiv Techniques for Simulating Logic Circuits",Computer nr.3,1975,p.52-55.

57 CHIANG A.C.L.-"Test Schemas for Microprocessor Chips", Computer Design,nr.4,1975,p.67-92.

58 COSMA O.,etc.-"Proiectarea asistată de calculator a sistemelor discrete",Ed.Academiei R.S.R.,București,1984.

59 DODESCU GH.,IONESCU DAN.-"Consideration on the efficiency of redundant coding in cybernetic systems",in Modern Thends in Cybernetics and Systems, vol.II,București,aug. 1975,p.291-301.

60 DANDAPONI R.,REDDY S.R.-"On the Design of Logic Networks with redundancy and Testability Considerations",IEEE Trans.computer vol.C-23,nov.1974,p.1139-1149.

61 Mc DERMOTT R.-"Simulation of Simple Digital Logic Through a Computer-Aided Design System",in Byte,vol.8,ian.1983.

62 DERMAN S.-"Benchmark Testing of Microprocessors" in Microprocessor Basics,Hayden Book Comp.,inc.Rachelle Park,New Jersey,1976.

63 DUERR R.J.-"Data communications testing averview-Digital testing",Computer Design,vol.18,nr.3,mart.1979,p.12-18.

64 DASGUPTA S., HARTMANN C.R.P., RUDOLPH L.D.-"Dual Mode Logic for Function-Independent Fault Testing",IEEE Trans.on Computer,vol.C-29,nov.1980,p.1025-1029.

65 DESCHIZEAUX-"Localisation des pannes dans les grands ensembles logiques".-Revue d'Automatique,Informatique et Recherche operationnelle-Automatique,iulie,1974,p.76-84.

66 DRAGANESCU M., PETRESCU A., STEFAN G.-colectiv de editare "Calculatoarele electronice din generația a cincea",In ed. academiei RSR,Bucuresti,1985.

67 DODESCU GH.,IONESCU D.,POPESCU CR.,POPA I.-"Minicalculatoare.Aplicații,Ed.Tehnică Bucuresti, 1978.

68 EPURE M.ș.a.-"Calculatoarele FELIX-256,IRIS-50,IBM 360/30, 40, Ed.Tehnică,Bucuresti,1974.

69 FIROOZ KAMRAN,-"Board Testing with Signature Analysis",in Hewlett-Packard journal, mart.1979,p.31.

70 FROHWERK A.ROBERT.-"Signature Analysis:A New Digital Field Service Method" in Hewlett-Packard journal,mai 1977,p.2-8.

71 FUJIWARA H., KINOSHITA K.-"Some Existence Theorems for Probabilistically Diagnosable Systems",IEEE Trans.computer, vol.C-27,apr.1978,p.379-384.

72 HAHNBACH W.A.-"Bring up your μP"bit-by-bit"-Test both hardware and software in less time",Electronic Design,19 iul.1976,p.80-85.

73 HARNBACH W.A.-"System Testing with a Logic-State Analyzer" in Microprocessor Basics,Hayden Book Comp.,inc.Rachelle Park, New Jersey,1976.

74 FUJIWARA H., NAGAO Y., SASAO T., KINOSHITA K.-"Easily Testable Sequential Machines with Extra Inputs",IEEE Trans. computer,vol.C-24,aug.1975,p.821-826.

75 FUJIWARA H., INOSHITA K.-"Design of Diagnosable Sequential Machines Utilising Extra Outputs", IEEE Trans.on Computer,vol.C-23,nr.2,feb.1974,p.138-145.

76 FRANCIS R., THIZEL R.-"Real time prototype analysis as a microprocessor design aid", Computer Design,vol.17, dec.1978,p.65-73.

77 FRIEDMAN A.D.,MENON P.R.-"FAULT DETECTION IN DIGITAL CIRCUITS", Englewood Cliffs, New Jork:Pretice-Hall,1971

78 FUJIWARA N., KINOSHITA K.-"Connection assignment for probabilistically diagnosable systems",IEEE Trans.on computer,vol.C-27,mart.1978,p.280-283.

79 FRIEDMAN A.D.-"Feedback in synchronous sequential switching circuits",IEEE Trans.Elect.Computer,vol.EC-15, iun.1966,p.354-364.

80 FRIGENBAUM E.,Mc CORDUCK P.-"La cinquieme génération" in Inter Edition, Paris 1984.

81 FREY N.N.- "Safety an Reliability - Their terms and models of complex systems",-IFAC, 1980,p.3-10.

82 GROVES A. WILLIAM-"Rapid Digital Fault isolation with FASTRACE",in Hewlett-Packard journal, mart.1979,p.8-13.

83 GOLIA J.,CROUZET Y., VERONIAULA M.-"Phisical Versus Logical Fault Models in MOS-LSI Circuits:Impact on Their Testability", IEEE Trans.on Computer,vol.C-30, iun.1980,p.527-531.

84 HANGANUI -"Utilizarea calculatoarelor in procese industriale", Cluj.

85 HILL F.J.,y PETERSON G.R.-"Calculatoare numerice-Hardware structură şi proiectare", Editura Tehnică,Bucureşti, 1980.

86 HAKIMI S.L., AMIN A.T.-"Characterisation of Connection Assignment of Diagnosable Systems",IEEE Trans.Computer, vol.C-23,ian.1974,p.86-88.

87 HAYES J.P.-"On Modifying Logic Networks to improve Their Diagnosability",IEEE Trans.Computer,vol.C-23, ian.1974,p.56-62.

88 HNATEK R.E.-"Test Methods for Microprocessors",in Microprocessor Basics, Hayden Book Comp.,INC,Rochelle Park, New Jersey, 1976.

89 HUSTON R.-"Microprocessor Function Test Generation on the Sentry 600",in Technicall Bulletin, Fairchild Systems nov.1974.

90 HAMILTON G.A.-"Testing LSI Boards can be easy",Systems International, vol.6, nr.10,dec.1978,p.20-21.

91 HAYES J.P.,FRIEDMAN A.D.-"Test Paint Placement to Simplify Fault Detection",IEEE Trans.on Computer,vol.C-23 iulie 1974,p.727-737.

92 HAMILTON G.A.-"Testing Microboards",in Systems International, sept.1978,p.37-38.

93 HOPKINS L.A.jr.,SMITH T.B.-"The architectural elements of a symmetric fault-tolerant multiprocessor",IEEE Trans.on Computer vol.C-24,mai 1975,p.498-505.

94 IONESCU T.-"Grafuri".Aplicații, E.D.P.Bucuresti,1973.

95 JURCA I.-"A Multiprocessor System with Multitasking Facilities", teză doctorat, Delft (Olanda) 1977.

96 KAMAL S., PAGE C.V.-"Intermittent Faults:A model and detection procedure",IEEE Trans.Computer,vol.C-23,iul.1974, p.713-719.

97 KAREL I., KOHOVI Z.-"Diagnosis of Intermittent Faults in Combinational Network",in IEEE Trans.on Computer,vol.C-26 nov.1977,p.1154-1158.

98 KARUNANITHI S., FRIEDMAN A.D.-"Analysis of Digital Systems Using o New Measure of Systems Diagnosis",IEEE Trans. Computer, vol.C-28,feb.1979,p.121-133.

99 KARPOVSKY M.-"An Approach for Error Detection on Error Correction in Distributed Systems Computing Numerical Functions", IEEE Trans.on Computer,vol.C-30,dec.1981, p.947-953.

100 KIME C.R.-"An Analysis Model for Digital Systems Diagnosis IEEE,Trans.Computer,vol.C-19,nov.1970,p.1063-1073.

101 KU C.T., MASSON G.M.-"The Boolean difference and multiple fault analysis",IEEE Trans.on Computer,vol.C-24, iulie 1975,p.691-695.

102 KAMEYAM M., HIGUCHIT -"Design of Dependent-Feilure-Tolerant-Microcomputer System Using Triple-Modular-Redundancy",IEEE Trans.on Comput.vol.C-29,feb.1980,p.202 205.

103 KOREN I., SADEN E.-"A New Approach to the Evaluation of the Reliability of Digital Systems",IEEE Trans.on Computer,vol.C-29,iun.1980.

104 KARPLUS W.J.-"Sisteme de calcul cu divizarea timpului" Ed.Tehnică,Bucuregti, 1970.

105 KORN A.G.-"Microprocesoare, minicalculatoare, Ed.Tehnică, Bucuregti, 1981.

106 LORIN J.- "PARALLELISM IN HARDWARE AND SOFTWARE REAL AND APPARENT CONCURRENCY, Pretince-Hall, New Jork,1970

107 LEPICARD G.-"Conception des ordinateure",in Techniques de L'ingenieur, nr.9, 1971, cap.10,p.H720

108 LEATHERMAN J., BURGER P.-"System Integration and Testing with Microprocessors-I",in Microprocessors:Fundamentals and Applications, 1977, p.193-197.

109 LALICTIS T.A., BRUMETT T.D.-"A Microprocessor-controlled Digital Integrated Circuit (DIC) Test Systems" IEEE Computer, oct.1975,p.60-67.

110 MEYER G.G.L.,MASSON G.M.-"An Efficient Fault Diagnosis Algorithm for Symmetric Multiple Processor Architectures",in IEEE Trans.Comput.vol.C-27,nov.1978,p.1059-1063.

111 MAHESHWARI S.N.,HAKIMI S.L.-"On Models for Diagnosable Systems and Probabilistic Fault Diagnosis", IEEE Trans Computer, vol.C-25, mart.1976,p.228-236.

112 SANKOWSKY G.-"Syndrome-Testability Can be Achived by Circuit Modification",in IEEE Trans.on Computer,vol.C-30, nr.6, aug.1981, p.604-606.

113 MEYER G.G.L.-"A Fault Diagnosis Algorithm for Asymetric

Modular Architectures",IEEE Trans. on Computer,vol.C-30, ian.1981,p.81-83.

114 MALLELA S., MASSONG M.-"Diagnosable Systems for Intermittent Faults",IEEE Trans.Computer,vol.C-27,iun.1978, p.560-566.

115 MUEHLDORF I.E., SAVKAR D.A.-"LSI Logic Testing-An overview",IEEE Trans.on Computer,vol.C-30,ian.1981,p.1-17.

116 MALLELA S., MASSONG E.-"Diagnosis Without Repair for Hibrid Fault Situations",IEEE Trans.on Computer,vol.C-29, iun.1980,p.461-470.

117 MURESAN T., STRUGARU C., STOINESCU R., PETRIU E.-"Microprocesorul 8080 in aplicații",Ed.Facla 1981,Timișoara.

118 NAKI G.E.,SAWIND H.-"Fault-Tolerant Asynchronous Sequential Machines",IEEE Trans.on Computer,vol.C-24,iulie 1974, p.651-657.

119 MILTON L.-"Microprocessor proliferation opens new test-equipment markets-High Technology,apr.1980,p.20-22.

120 MEADOWS E., PARSONS A.J.-"Microprocessors:Essentials,Components and Systems";Pitman,London,1982.

121 MIRESCU P.,ROSU Al.,-"Teoria grafurilor",Ed.Militară, București, 1966.

122 MAISON F.P.-"The NEGRA:A self-repairable computer for highly reliable process",IEEE Trans.on Computer,vol.C-20, nov.1971,p.1382-1393.

123 MOORE A.W.,etc.-"Microprocessor Applications Manual Mc Graw-Hill, New Jork,1975.

124 NELSON BOB -"Error Correction the Hard Way",in Computer Design,vol.20,nr.12,1981,p.187-190.

125 NTAFOR C.S.,HAKIMI L.S.-"On Structural Diagraphs and Program Testing",IEEE Trans.on Computer,vol.C-30,ian.1981,p 67-77.

126 NEESE W.J.-"Microprocessor System Validation and Failure Isolation with Portable Tester",Computer Design,vol.16,nr.9 1977,p.105-111.

127 NG Y.W.,AVIZIENIS A.A.-"A Unified Reliability Model for Fault-Tolerant Computers",IEEE Trans.on Computer,vol.C-29,

128 PETRESCU M.-"Efectul integrării pe scară medie şi largă asupra tehnicilor de sinteză a circuitelor combinaţionale şi secvenţiale",I.P.Bucureşti,1979.

129 PIMENTEL R.J.,LOEFFLER T.M.-"A Real-Time Engine Simulator using Multiple Microcomputers",in IEEE Trans.on Industrial Electronics,vol.IE-30,nr.2,mai 1983,p.117-125.

130 PREPARATA F.P., METZE G., CHIEN R.T.-"On the Connection assignment problem of diagnosable systems",IEEE Trans.Computer vol.C-16,dec.1967,p.848-854.

131 PUTZOLU G.R., ROTH J.P.-"A heuristic algorithm for Testing of asynchronous circuits",IEEE Trans.on Computer, vol.C-20,iun.1971,p.639-647.

132 PETRESCU A., MOISA T., TAPUS N., GAYRAUD A., MOTEZ C.-"MICROCALCULATOARELE FELIX 18, M 188, M 118", Ed.Tehnică,Bucureşti, 1984.

133 PRADHAN D.K.-"O New Class of Error-Correcting/Detecting Codes for Fault Tolerant Computer Applications",IEEE Trans.on Computer,vol.C-29,iun.1980,p.471-481.

134 POP V., STRATULAT M., GROZA V.-"Studiul tehnico-economic privind oportunitatea abordării în MSE a memorilor cu dispozitive semiconductoare cuplate prin sarcină",Protocol nr.6/1986 ITC.

135 PRICE W.L., DAVIES D.W., HASLER D.I., SCIOMIDES C.M.-"Teleinformatica,Reţele de calculatoare şi protocoalele lor,Ed.Tehnică, Bucureşti,1983.

136 PETRESCU A.-"Microprogramare.Principii şi aplicaţii",ed. Tehnică Bucureşti,1977.

137 POP V.-"Bazele logice ale calculatoarelor".-curs vol.I, Litog.I.P.Timişoara,1972.

138 POP V.-"Structura sistemelor de prelucrarea datelor numerice",curs vol.I,II.,Litog.I.P.Timişoara,1982.

139 POP V.-"Analiza şi sinteza dispozitivelor numerice",Litog I.P.Timişoara,1985.

140 RUSSELL J.A.,KIME C.R.-"System Fault Diagnosis:Closure

and Diagnosability with repair",IEEE Trans.Computer,vol.C-24 nov.1975,p.1078-1089.

141 REDDY S.M.-"Easily Testable Realisations for Logic Functions",IEEE Trans.Computer,vol.C-21,nov.1972,p.1183-1188.

142 RUSSELL J.A., KIME C.R.-"System Fault Diagnosis:Masking Expasure,and Diagnosability without repair",IEEE Trans. Computer,vol.C-24,dec.1975,p.1155-1161.

143 RASEK E.-"Generating error correction Codes",Systems International,vol.7,iulie 1979,p.32-36.

144 RAYMOND D.W.-"Component-by-component Testing of Digital Circuits Boards",Computer Design,vol.19,apr.1980,p.129-137.

145 REDDY S.M.-"A class of Linear Codes for Error Control in Byte-per-Card Organized Digital Systems",IEEE Trans.on Computer, mai 1976,p.455-459.

146 ROTH J.P., BOURICIUS W.G., SCHNEIDER P.R.-"Programmed Algorithms to Compute Tests to Detect Between Failures in Logic Circuits",IEEE Trans.Electron Computer,vol.EC-16, oct.1967,p.567-580.

147 REDDY S.M.-"A Note on Self-Checking Chekers",IEEE Trans.on Computer,vol.C-23,oct.1974,p.1100-1002.

148 ROGOJAN AL., POP V., STRUGARU C., STRATULAT M,etc.-"Tester de baterii electrice", Buletinul I.P.T., Tomul 25,fasc.2, 1980.

149 ROSU AL.AL.-"Teoria grafelor.Algoritmi aplicaţii,Ed.Militară,Bucureşti, 1974.

150 ROGOJAN AL.-"Calculatoare numerice,Curs vol.I,II,III,Litog. I.P.Timişoara,1974.

151 STRUGARU C., DICSOV E., BEDROS D., NOVACESCU C., MORUN C., PANESCU D.,-"Microcalculator individual SPECTIM în microcalculatoare şi utilizarea lor în industrie",Simpozionul naţional "Microprocesoare",Timişoara, 1985.

152 SMITH E.,JAMES.-"Measure of the Effectiveness of Fault Signature Analysis",in IEEE Trans.on Computers,vol.C-29, Nr.6,iun.1980,p.510-514.

153 SELLERS F.F., HSIAO M.Y., BEARNSON L.W.-"ERROR DETECTING

LOGIC FOR DIGITAL COMPUTER, New Jork: Mc Graw-Hill,1965.

154 SU S.Y.H., KAREN I., MALAIYA Y.K.-"A Continuous-parameter Markov Model and Detection Procedures for Intermittents Faults",in IEEE Trans.Computer,vol.C-27,iun.1978,p.567-570

155 SAVIR J.-"Detection of Single Intermittent Faults in Sequential Circuits",IEEE Trans.on Computer,vol.C-29,iul.1980, p.673-678.

156 SRIDHAR T., HAYES P.J.-"Design of Easily Testable Bit-Sliced Systems",in IEEE Trans.on Computer,vol.C-30,nr.8, aug.1981,p.563-572.

157 SMITH E.J.-"Universal System Diagnosis Algorithm",IEEE Trans.Computer,vol.C-27,mai 1979,p.374-378.

158 SPATARU AL.-"TEORIA TRANSMISIUNII INFORMATIEI,vol.1,Ed. Tehnică, Bucuresti, 1965.

159 SCHNEIM B.- "A Flexible Approach to Microprocessor Testing", Computer Design, mart.1976,p.67-72.

160 SZYGENDA S.A., THOMPSON E.W.-"Modeling and digital simulation for design verification diagnosis",IEEE on Computer, vol.C-25,dec.1976,p.1242-1253.

161 SWANSON R.-"Matrix technique leads to direct error code implementation" Computer Design, vol.10,nr.8,1980,p.101-106.

162 SEDMAK M.A., LIEBERGOT H.L.,-Fault Tolerance of a General Purpose Computer Implemented by Very Large Scale Integration",IEEE Trans.on Computer,vol.C-30,iun.1980,p.492-500.

163 SALUJA K.L.-"Synchronous Sequential Machines: A Modular and Testable Design, IEEE Trans.on computer,vol.C-29,nov.1980, p.1020-1025.

164 SIMONCINI L., SAHBAN F., FRIEDMAN A.D.-"Design of Self-Diagnosable Multiprocessor Systems with Concurrent Computation and Diagnosis", IEEE Trans. on Computer,vol.C-30, iun.1980,p.540-546.

165 SCHÜLLER H., SANTUELI C.-"The combined role of redundancy and test programs in improving fault tolerance and failure detection",in IFAC,1980.

166 STRATULAT M.-"Tehnica impulsurilor şi circuite de comutare",vol.I şi II, I.P.T.V. Timişoara,1981.

167 STRATULAT M.-"Proiectarea logică a unei interfeţe între calculatorul CETA şi unitatea de bandă magnetică PT-3", referat de doctorat nr.1, 1981.

168 STRATULAT M., CHECEANU M., MIHAILESCU A.-"Inregistrarea şi redarea parametrilor de sudare la instalaţiile automate de sudură", Buletinul de comunicări tehnico-ştiinţifice, oct.1981, Constanţa.

169 STRATULAT M., CHECEANU M., MIHAILESCU A., MOS I.C.-"Sistem de achiziţie conversie şi reproducere a mărimilor analogice",al X-a Sesiune de comunicări tehnice-ştiinţifice, în domeniul automatizărilor, mai 1982, Bucureşti.

170 STRATULAT M., MOS I.C.- "Sistem de achiziţii şi memorare de date pentru urmărirea parametrilor sudurii cu arc electric realizat cu microprocesoare IMS 100. Sesiunea de comunicări tehnico-ştiinţifice "Utilizarea calculatorului în industrie", nov.1983 Timişoara.

171 STRATULAT M., CIORDAS M., HERMAN L., FINGERMAN E.,- Calculator didactic microprogramat", Sesiunea de comunicări tehnico-ştiinţifică,oct.1979 Timişoara.

172 STRATULAT M.-"Structuri de calculatoare cu posibilităţi de autotestare", Simpozionul naţional,"Microprocesoare, minicalculatoare şi utilizarea lor în industrie",nov. 1985, Timişoara.

173 STRATULAT M.-"Simulator logic", Idem.

174 STRATULAT M.-"Diagnosemethode fur autotestfähiger strukturen, idem.

175 STRATULAT M., BOSCU I., BANCIU S., BULTER W.-"Autodiagnostizierbare Rechnerstruktur mit Zwei Mikroprozessoren", idem.

176 STRATULAT M., CHECEANU M., MIHAILESCU A, MOS I.C.-Sistem de inregistrare şi redare a unor mărimi analogice având ca suport de informaţie:banda magnetică (SIR-BM),certificat de inovaţie nr.196, iunie 1985, I.P.Timişoara.

177 [illegible] D.K.-"A Fault-Tolerant Communication architecture for Distributed Systems" IEEE Trans.on Computers,vol.C-31,sept.1982.

178 PRADHAN D.K.-"Fault-Tolerant Multiprocessor Link and Bus Network Architectures", IEEE Trans.on Computers, vol.C-34,ian.1985,p.33-46.

179 PRADHAN D.K.-"Dynamically Restructurable Fault-Tolerant Processor Network Architectures",IEEE Trans-on Computers,vol.C-34,mai 1985,p.434-447.

180 VARGHESE P.K.-"On analytical modeling of intermittent faults in digital systems",in IEEE Transon Computers, vol.C-28,oct.1979mp.706-791.

181 VLADUTIU M., MOS I.C., STRATULAT M., JEBELEAN I.-"Determinarea fiabilităţii parametrice a subansamblurilor electronice din echipamente de sudare şi tăiere",Buletinul de comunicări ştiinţifice "Tehnic 2000",apr.1984, Timişoara.

182 VLADUTIU M.-"Contribuţii la creşterea coeficientului de disponibilitate al echipamentelor automate de prelucrare a informaţiei prin testare neconvenţională",-teză de doctorat, I.P.Bucureşti,1982.

183 VLADUTIU M., STRATULAT M., MOS I.C., TUSA O.-"Aplicarea redondanţei structurale active de restabilire",Buletinul celei de a IX-a sesiuni de comunicări ştiinţifice a cadrelor didactice din Institutul "Mircea cel Bătrîn", iunie,1984,Constanţa.

184 VLADUTIU M.-"Tehnologie de reparaţii şi fiabilitate",Lit. I.P.Timişoara,1981.

185 [illegible]-"A Hypergraph Model for Fault-Tolerant VLSI Processor Arrays", IEEE Trans-on Computers,vol.C-34,iun.1985,p.578-583.

186 ROBINSON J.P.-"Segmented Testing" IEEE Trans-on Computers,vol.C-34,mai 1985,p.467-471.

187 SHEN T.M.,SAVIR J.-"Layout Influences Testability", IEEE Trans.on Computers,mart.1985,p.287-290.

188 [illegible] J.A.-"Distributed Function Microprocessor Architecture" in Microprocessors: Fundamentals and Aplications, IEEE Press,New York, 1977,p.232-239.

189 [illegible] E.J., ANGELL J.B.-"Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic.", IEEE Trans.Computers,vol.C-22,iun.1973, p.46-60.

190 [illegible] E.S., GHAI F.G., THOMPSON A.H.-"Self Diagnosing Cellular Implementations of Finite-State Machines",IEEE Trans-on computers,vol.C-30,dec-1981,p.953-958.

191 [illegible] E.J.-"Microcomputer Reliability Improvment Using Triple-modular Redundancy" in Microprocessors: Fundamentals and Applications, IEEE pres.1977.

192 [illegible] J.H.-"Designer's Guide to: Testing and troubleshooting µP"-based products", EDN,mart.1980.

193 [illegible] J.,KOLFA R.,KRAUSE J.-"Software Error Checking Procedures for Data Communication Protocols", Computer Design vol.18,nr.2,1979,p.122-126.

194 [illegible] J.A.-"Analysis of Multiple-Microprocessor System Architectures", Computer Design, vol.16,1977, p.151-163.

195 [illegible] E.J.-"Partially Self-Checking Circuits and Their Use in Performing Logical Operations",IEEE,Trans on Computers,vol.C-23,iulie 1974,p.658-660.

196 [illegible] J.D.-"L'homme face a l'intelligence artificielle" in Les editions d'organisation,Paris,1984.

197 Implementing Signature Analysis for Production Testing with the HP 3060 A Board Test system", notă aplicativă nr.222-1, Hewlett Packard.

198 [illegible] E.H., [illegible] E.J.-"Functional redundancy to achieve high reliability",IFAC 1980,p.59-64.

199 [illegible] R.-"On a diversified parallel microcomputer system", IFAC,1980,p.81-88.

200 [illegible] G.-"Overview of hardware related safety problems of computer control systems",IFAC 1980,p.169-177.

201 ROBACH C., SAUCIER G.-"Dynamic Testing of control Units" IEEE Trans.on computer,vol.C-27,iulie 1978,p.617-623.

202 ROBACH C., SAUCIER G., LEMMON J.-"Processor Testability and Design Consequences",IEEE Trans.on Computer,vol.C-25 iun.1976,p.645-652.

203 MANON P.R., CHAPPELL S.G.-"Deductive Fault Simulation with Functional Blocks",IEEE Trans.on computer,vol.C-27, aug.1978,p.689-695.

204 MEHRA S.K., WONG J.W., MAJITHIA J.C.-"A Comperativ Study of Some Two-Processor Organisations",IEEE Trans.on Computer,vol.C-29,ian.1980,p.44-49.

205 BEAUDRY M.D.-"Performance-Related Reliability Measures for Computing Systems",IEEE Trans.on Computer,vol.C-27, iun.1978,p.540-547.

206 THATTE S.M., ABRAHAM J.A.- "Test Generation for Microprocessors",IEEE Trans.on computer,vol.C-29,iunie 1980.

207 LAU L.F.-"Test Policy VS Maintenance Policy and System Reliability"IFAC ,1980,p.201-206.

208 SASTRY K.V.,KAIN R.Y.-"On the Performance of Certain multiprocessor Computer Organisations",IEEE Trans.on Computer,vol.C-24,nov.1975,p.1066-1074.

209 AGRAWAL D.VISHWANI.-"An Information Theoretic Approach to Digital Fault Testing",in IEEE Trans.on Computer,vol. C-30,nr.8,aug.1981,p.582-587.

210 DANCAU A.G.-"Probleme ale testării industriale a unităților de memorie RAM cu circuite integrate",Teză de doctorat,Timişoara,1983.

211 DANIELS B.K., ALTKEN A., SMITH I.C.-"Experience with computers on some UK power plants",IFAC,1980,p.11-33.

212 HUBNER D.SCHONHERR E.-"Diagnostic inder Digitaltechnik", VEB Verlag Technik Berlin,1982.

213 DRAGANESCU M.-"A doua revoluție industrială",București, Ed.Tehnică,1980.

214 LAUBER R.-"Introduction into the subject of the workshop in IFAC,1980.

215 PETRESCU M.-"Dezvoltarea stiinței sistemelor cibernetice în România",în Istoria științelor în România,Cibernetica,Ed.Academiei RSR,București,1981.

216 SAVIR J.-"Syndrome-Testing of "Syndrome-Untestable Combinational circuits",IEEE Trans.on Computer,vol.C-30,aug. 1981,p.606-608.

217 SAVIR J.-"Syndrome-testability Design of Combinational circuits",IEEE Trans.Computer,vol.C-29,iun.1980,p.442-451.

218 AVIZIENIS A., KELLY J.P.J.-"Fault Tolerance by Design Diversity:Concepts and Experiments",Computer,aug.1984, p.67-80.

219 Mc.DONALD W.C., SMITH R.W.-"A Flexible Distributed Testbed for Real-Time Applications",Computer,oct.1982,p.25-39.

220 FATHI E.T.,KRIEGER M.-"Multiple Microprocessor Systems", Computer, mart.1983,p.23-32.

221 GOTTLIEB A., SCHWARTZ J.T.-"Network and Algorithms for Very-Large-Scale Parallel Computing",Computer,ian.1982, p.27-36.

222 GEHRINGER E.F., JONES A.K., SEGALL Z.Z.-"The Cm Testbed", Computer,oct.1982,p.40-53.

223 HINDEN R., HAVERTY J., SHELTZER A.-"The DARPA Internal: Interconnecting Heterogeneous Computer Networks with Gateways",Computer,sept.1983,p.38-48.

224 HAYNES L.O., LAU R.L., SIEWIOREK D.P., MIZELL D.W.-"A Survey of Highly Parallel Computing",Computer,ian.1982, p.7-24.

225 JOHNSON D.-"The intel 432:A VLSI Architecture for Fault-Tolerant Computer System",Computer,aug.1984,p.40-48.

226 LILEN H.-"Interfaces pour microprocesseurs et micro-ordinateurs",Ed.Radio,Paris,1983.

227 THORNTON J., CHRISTENSEN G.S.-"Hyperchannel Network Links" Computer,sept.1983,p.50-54.

228 SNYDER L.-"Introduction to the Configureble,Highly Parallel Computer",Computer,ian.1982,p.47-56.

229 [illegible] D.P.-"Architectures of Fault-Tolerant Computers" Computers,aug.1984,p.9-18.

230 BABUTIA I., [illegible] T., [illegible] I., PROSTEAN O.-"Conducerea automată a proceselor", Ed.Facla 1985.

231 SHRIER O.-"Fault-Tolerant Systems in Commercial Applications", Computers,aug.1984,p.19-30.

232 BOWLER K.C., SIEVERS M.W.-"Off-Line, Built-in Test Techniques for VLSI Circuits", Computers,iun.1982,69-82.

```
      INTEGER P(255)
      COMMON /P1/P(255,36)
      READ(105,1)N,IT
    1 FORMAT(2I2)
      NN=N*IT
      CALL STIP(N,IT)
      NRC=1
      PC=1
      I=1
   10 J=1
    2 M(I,J)=0
      J=J+1
      IF(JCLE.N*IT)GO TO 2
      I=I+1
      IT=L-1*(N-(IT-1))
      NR=1
      LO=0
      NR=1
      LT=L*IT+L0
   25 J=1
   26 IF(LCGT.N)GO TO 3
      IF(JCEQ.LT)GO TO 4
   29 J=J+1
      IF(ONE.NE.ON)GO TO 5
   28 IF(JCLE.N*IT)GO TO 6
      IF(MN.LE.IT)GO TO 7
   27 J=1
   23 IF(JONE.(L-1)*IT+1) GO TO 8
    9 M(I,J)=100
      J=J+1
      IF(JCLT.L*IT+1)GO TO 9
      CALL SORT(NCOUNT,PC,N)
   24 WRITE(108,30)NRC,NRC,(M(I,J),J=1,N)
   30 FORMAT(,1X,'*',I5,'*',I7,'*',21(,1GT1)
      NRC=NRC+1
      I=I+1
      IF(I.LE.NCOUNT)GO TO 10
      IP=1
      NR=0
      PO=LO
   18 K=0
   21 L=K+1
   16 J=1
   15 IZ=M(I,J)
      IY=M(L,J)
      IF(IZ.EQ.100)GO TO 11
      IF(IY.EQ.100)GO TO 12
```



```
      IF(IZ.EQ.IY)GO TO 13
      IF(IY.EQ.IZ)GO TO 14
      M(I,J)=0
   22 J=J+1
      IF(JCLE.NN)GO TO 15
      CALL SUB(NRC,I,IP,I,N,P,NN)
      I=I+1
      I=I+1
      NRC=NRC+1
      IF(I.LE.N)GO TO 16
      K=K+1
      IK=I+1
      IF(K.LE.N-1)GO TO 21
      IF(IP.EQ.(NCOUNT-NC))GO TO 17
      IP=IP+1
      PC=PC+1
      PO=M
      CALL SORT(NCOUNT,PC,N)
      NCOUNT=NCOUNT+NCOUNT1
   20 IF(NC.LT.IT)GO TO 18
      GO TO 19
   17 IP=IP+1
      PO=PO+1
      K=0
      IF(PO.LT.N)GO TO 20
      IK=IP+1
      PO=(IK-1)*L+NP
      K=0
      NR=N+1
      GO TO 20
   14 M(I,J)=1
      GO TO 22
   13 M(I,J)=1
      GO TO 22
   12 M(I,J)=100
```

```
      GO TO 25
11    F(I,J)=100
      GO TO 22
10    J=J+1
      IF(J.LE.N+1)GO TO 21
      GO TO 24
7     I=I-1
      GO TO 23
6     IF(MN.LE.I)GO TO 26
      GO TO 27
5     MN=NR
      IC=II-1
```

SIMSIM 01/03/80 11.42.50

```
      I=I+10
      GO TO 28
8     NC=MN+1
      M(I,J)=1
      GO TO 29
3     IF(J.NE.I-M+1)GO TO 29
      GO TO 8
19    STOP
      END
```

SIMSIM 01/03/80 11.43.11

```
      SUBROUTINE STIP(N,IT)
      INTEGER A(36),B(36)
      REAL N2(5)
      DATA N2/10H(*X,      7H(*'*)/
      WRITE(108,2)N,IT
2     FORMAT(50X,'SINDROAMUL DE DETECTIE:',I3,'P(I='*)/20X,
     *'PENTRU N=',I2,'UNITATI SI I=',I2,'DETECTIE')/
     *20X,39('*')//)
      N5=5*N+II
      ENCODE(4,5,N2(5))N5
5     FORMAT(I4)
      WRITE(108,N2)
      K=1
      DO 3 I=1,N
      DO 4 NR=1,IT
      A(K)=I
4     K=K+1
3     CONTINUE
      N1=N*IT
      I=2
      K=1
7     DO 11 NR=1,IT
      B(K)=I
      IC=I+1
      IF(IC.GT.N)I=1
      K=K+1
      IF(IC.GT.N1)GO TO 6
11    CONTINUE
      K=K-IT
      I=B(K)+1
      IF(I.GT.N)I=1
      GO TO 7
6     WRITE(108,8)(A(K),K=1,N1)
8     FORMAT(1X,'*NR * TEST  *UNITATEA DE TESTARE*',36I1,'*')
      WRITE(108,9)(B(K),K=1,N1)
9     FORMAT(1X,'*ORT*UNITATI*UNITATEA TESTATA    *',36I1,'*')
      WRITE(108,N2)
      RETURN
      END
```

SIMSIM 01/03/80 11.43.21

```
      SUBROUTINE SUBT(NCONT,KC,M)
      I=1
      K=1
      DO 1 I=1,KC
1     I=I+1
      I=N+KC+1
2     K=K+1
      J=J+1
      IF(I.EQ.J)GO TO 2
      M=K-1
      RETURN
      END
```

SIMSIM 01/03/80 11.43.28

```
      SUBROUTINE SUBIRC,I,IP,I,N,B,NR)
      INTEGER B(255)
      COMMON /DIAP/255,50
      IF(IA.LT.5)GO TO 2
      B(NRC)=10*B(IP)+I
      GO TO 3
2     B(NRC)=10*IP+I
3     WRITE(108,1)NRC,B(NRC),NR,I,J,IP,IA,NA
```

```
                                                                    SIMSIM      01/03
```

MODULE	BL	TYPE	C	LONGUEUR	0F70
MODULE	EXMDATA	TYPE	P	LONGUEUR	0A60
MODULE	STTP	TYPE	P	LONGUEUR	0140
MODULE	SUPT	TYPE	P	LONGUEUR	0080
MODULE	BL	TYPE	C	LONGUEUR	0F70
MODULE	SUP	TYPE	P	LONGUEUR	0130

```
***** FIN DE COMPILATION (PLUS HAUT NIVEAU D'ERREUR RENCONTRE = 0)

                CENTRE DE CALCUL AL I.P.C.C. TIMISOARA                  SYSTE
0113  SIMSIM      AN = 0050   PP = 0002   DATE = 01/03/80-06
H.D.H = 11H 03M 525   H.FIN = 11H 03M 595   TIME = 00000170
UCP = 00046   MEM = 00015   I.O = 00C00179   IN. = 00C00167

C       LINK NEW
LINK    STARTED

AUCUNE ERREUR A L'EDITION DE LIENS

                CENTRE DE CALCUL AL I.P.C.C. TIMISOARA                  SYSTE
0113  SIMSIM      AN = 0050   PP = 0003   DATE = 01/03/80-06
H.D.H = 11H 03M 595   H.FIN = 11H 04M 159   TIME = 00000270
UCP = 00046   MEM = 00012   I.O = 00000005   IN. = 00000000

C       RUN
        STARTED

                        SINDROMUL DEFECTELOR

                        **********

                PENTRU N= UNITATI SI T= DEFECTE

                **********

***********************************************
*NR * TEST  =UNITATEA DE TESTARE=1112223334445556666777
*CRT=UNITATE=UNITATEA TESTATA    =2345345456567671712123
***********************************************

*   1=      1=                    **=0000000000001010100

*   2=      2=                    100**=0000000000001=10

*   3=      3=                    010100**=000000000001

*   4=      4=                    [redacted]
```

BUPT

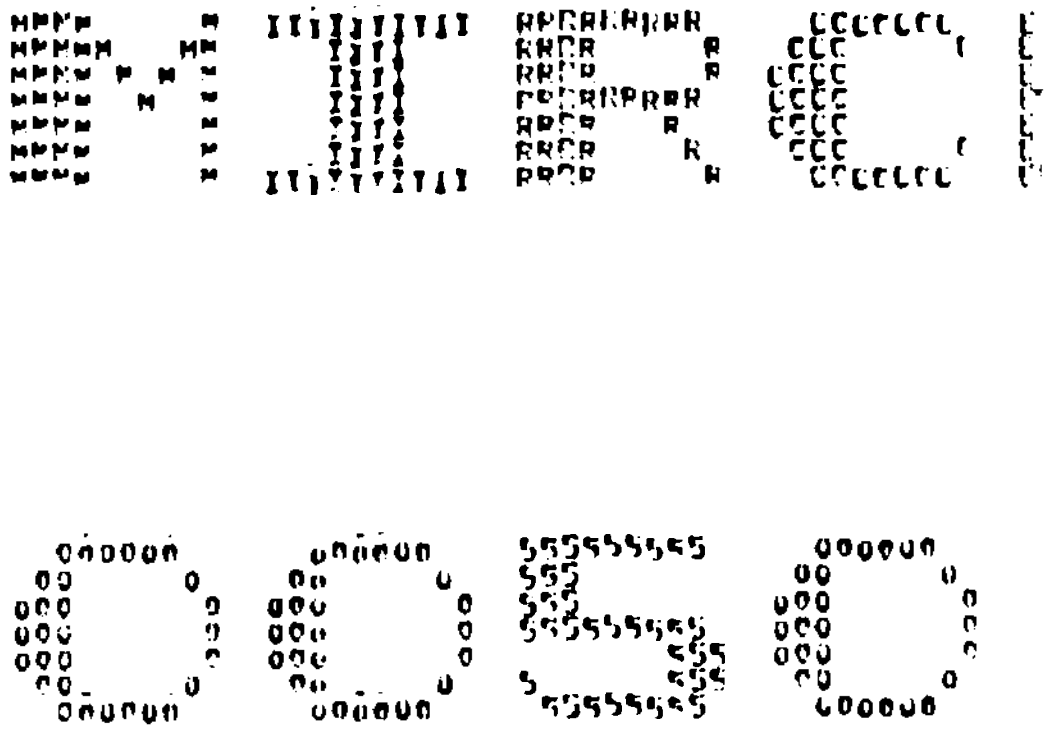
52	257	[illegible]
51	256	10[illegible]010
50	247	101[illegible]10
49	246	101[illegible]010
48	245	101[illegible]010
47	237	1[illegible]0101010
46	236	110[illegible]01
45	235	110[illegible]001
44	234	11[illegible]00000
43	167	[illegible]0000101111
42	157	[illegible]0101010
41	156	[illegible]0111100
40	147	[illegible]0101010
39	146	[illegible]0101010
38	145	[illegible]0111000
37	137	[illegible]100[illegible]0111000
36	136	[illegible]100[illegible]010101
35	135	[illegible]10[illegible]100[illegible]010101
34	134	[illegible]110[illegible]0010101
33	127	[illegible]0000101111
32	126	[illegible]0101010101
31	125	[illegible]0101000[illegible]0111
30	124	[illegible]100[illegible]0010111
29	123	[illegible]0000101111

BUFT

```
* 53*      267*                    100**=0010111110******

* 54*      545*                    011111**********000001

* 55*      546*                    011110**=***100***001

* 56*      547*                    011110**=****010100***

* 57*      556*                    010101**=110******001

* 58*      357*                    010101**=101***100***

* 59*      567*                    010100**=011110******

* 60*      056*                    0010111111*********000

* 61*      457*                    0010111110******100***

* 62*      067*                    0010101011*1110******

* 63*      567*                    0000010111111*********
*STOP*
```

```
CENTRE DE CALCUL AL I.P.T.C. TIMISOARA                    SYSTE
0113  SIMSIM      AN = 0050   PR = 0001   DATE = 01/03/80-06
P.O.H = 11H 44M 15S   H.LIN = 11H 44M 25S   TIME = 00000A7D
LGP = 00046   PLN = 00030   I.O = 00000141   IN = 00000007
```

BUPT



I'.ÂN;ŋUSO.PN;MIRCEA

$FORTRAN

SYMSIT 01/03/86 11.44.46

```
      INTEGER H(255)
      COMMON /PL/P(255,36)
      READ(105,1)N,IT
1     FORMAT(2I2)
      WN=N*IT*N
      CALL STIP(N,IT)
      NR=1
      NC=1
      I=1
10    J=1
2     P(I,J)=0
      J=J+1
      IF(J.LE.(N*IT+N))GO TO 2
      P=I
      L=N+1+(N-(IT-1))
      WN=1
      I=30
      NR=1
      ITL=IT+1-0
25    J=1
26    IF(J.GT.N)GO TO 5
      IF(J.GT.(IT-1))GO TO 4
```

```
C*      J=J+1
        IF(JD.NE.NN)GO TO 5
28      IF(J.LE.N+IT)GO TO 6
        IF(NN.LT.IT)GO TO 7
27      J=1
23      IF(J.NE.(K-1)*IT+1) GO TO 8
9       M(I,J)=100
        J=J+1
        IF(J.LT.K*IT+1)GO TO 9
        CALL SUBT(NCONT,KC,N)
        J=N*IT+I
        M(I,J)=1
        IX=N-(I-1)
        IF(IX.EQ.N)GO TO 40
        J=N*IT+N-IX
        GO TO 41
40      J=N*IT+N
41      M(I,J)=100
20      WRITE(10,30)NRC,NRC,(M(I,J),J=1,NN)
30      FORMAT(/,1X,'*',I3,'*',I7,'*',21X,'*',3011)
        NRC=NRC+1
        I=I+1
        IF(I.LE.NCONT)GO TO 10
        I=1
        NR=0
```



```
        KO=KC
18      K=KO
21      L=L+1
16      J=1
15      IZ=M(I,J)
        IY=M(L,J)
        IF(IZ.EQ.100)GO TO 11
        IF(IY.EQ.100)GO TO 12
        IF(IZ.EQ.1)GOTO 13
        IF(IY.EQ.1)GO TO 14
        M(I,J)=0
22      J=J+1
        IF(J.LE.NN)GO TO 15
        CALL SUBT(NRC,I,IT,J,N,P,NN)
        L=1
        I=I+1
        NRC=NRC+1
        IF(I.LE.N)GO TO 16
        K=K+1
        IF(K.EQ.N)GO TO 21
        IF(IC.LE.(NCONT-KC))GO TO 17
        IK=IK+KC
        KC=KC+1
        KO=KC
        CALL SUBT(NCONT1,KC,N)
        NCONT=NCONT+NCONT1
20      IF(KC.LT.IT)GO TO18
        GO TO 19
17      IK=IK+1
        KO=KO+1
        K=KO
        IF(KO.LT.N)GO TO 20
        IK=IK+1
        KO=KO+(N-1)*L+NR
        K=KO
        NR=NR+1
        GO TO 20
14      M(I,J)=1
        GO TO 22
13      M(I,J)=1
        GO TO 22
12      M(I,J)=100
        GO TO 22
11      M(I,J)=100
        GO TO 22
8       J=J+1
```



```
        IF(J.LE.N+IT)GO TO 24
        GO TO 26
7       ITA=-1
        GO TO 25
6       IF(NN.LT.IT)GO TO 26
        GO TO 27
5       NN=NN
        KO=IT-1
        ITA=IT+10
        GO TO 28
4       NK=NN+1
        M(I,J)=1
        GO TO 29
3       IF(JONE.LT.N+IT)GO TO 24
        GO TO 4
10      STOP
```

```
      SUBROUTINE STIP(N,IT)
      INTEGER A(36),B(36)
      REAL N2(3)
      DATA N2/1HU(*X*,      71(*=*))/
      WRITE(10R,2)N,IT
    2 FORMAT(30X,'SIMULARUL DE EFECTELON',/,30X,20(*=*)//20X,
     *'PENTRU N=',I2,' UNITATI SI T=',I2,' UNITATI'/
     *20X,39(*=*)//)
      N3=36,N*IT-N
      ENCODE(4,5,N2(3))N3
    5 FORMAT(I4)
      WRITE(10R,N2)
      M=1
      DO 3 I=1,N
      DO 4 JR=1,IT
      A(M)=I
    4 M=M+1
    3 CONTINUE
      DO 12 I=2,N
      A(M)=I
   12 M=M+1
      A(M)=1
      N1=N*IT
      N3=N1+N
      I=2
      M=1
    7 DO 11 NR=1,IT
      B(M)=I
      I=I+1
      IF(I.GT.N)I=1
      M=M+1
      IF(M.GT.N)GO TO 6
   11 CONTINUE
      M=M-IT
      I=B(M)+1
      IF(I.GT.N)I=1
      GO TO 7
    6 DO 13 I=1,N
      B(M)=I
   13 M=M+1
      WRITE(10R,8)(A(K),K=1,N3)
    8 FORMAT(1X,'*NR * TEST *UNITATE (I TESTEAZA=',36I1,'*')
      WRITE(10R,9)(B(K),K=1,N3)
    9 FORMAT(1X,'*CRT*UNITATI*UNITATE TESTATA    =',36I1,'*')
      WRITE(10R,N2)
      RETURN
      END
```

SIMSIT 01/03/80 11:45:09

```
      SUBROUTINE SUBINCNT,PC,N)
      I=1
      M=1
      DO 1 I=1,NC
    1 I=A(I)
      I=N-NC+1
    2 M=M+1
      I=I+1
      IF(I.LE.N) GO TO 2
      NCOUT=M/I
      RETURN
      END
```

SIMSIT 01/03/80 11:45:10

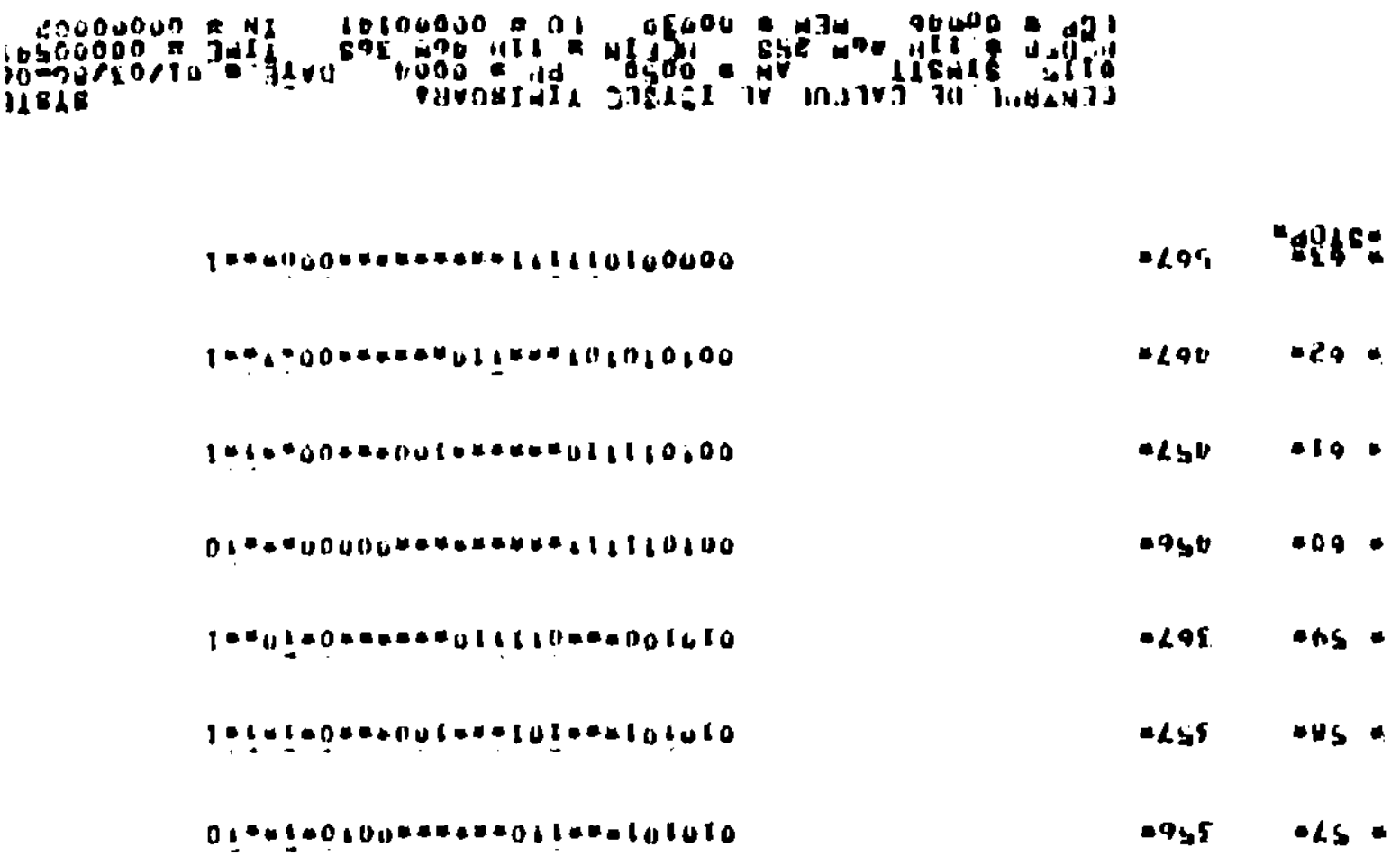
```
      SUBROUTINE SUB(NRC,I,IP,I,N,B,NN)
      COMMON /B1/B(255,36)
      INTEGER B,255)
      IF(I.EQ.1)N1=0 TO 2
      B(NRC)=10*(IP)+1
      GO TO 3
    2 B(NRC)=10*IP+1
    3 WRITE(10R,1)NRC,B(NRC),(B(I,J),J=1,NN)
    1 FORMAT(/,1X,'*',I3,'*',I7,'*',21X,36I1)
      RETURN
      END
```

SIMSIT 01/03/80 11:45:10

MODULE	B1	TYPE	C	LONGUEUR	0F70 (36720)
MODULE	FXMDATA	TYPE	P	LONGUEUR	0030 ([illegible])

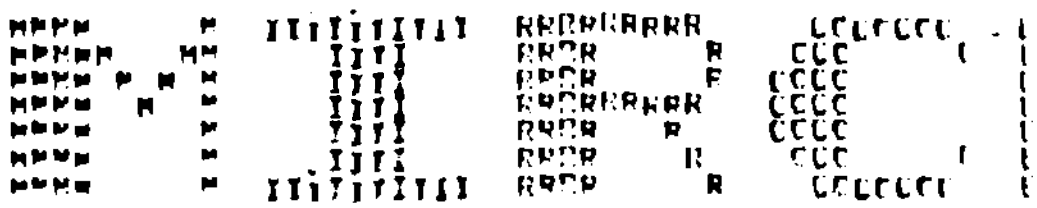
```
                MODULE      STTP            TYPE    P       LONGUEUR    07F
                MODULE      SUBT            TYPE    P       LONGUEUR    0004
                MODULE      P1              TYPE    C       LONGUEUR    0F7A
                MODULE      SUB             TYPE    P       LONGUEUR    0152

***** FIN DE COMPILATION (PLUS HAUT NIVEAU D'ERREUR RENCONTRE = 0)


                CENTRUL DE CALCUL AL ICTCC TIMISOARA                    SYST
                0115  SIMSIT        AN = 0050   PP = 0002   DATE = 01/03/80
                H.DEB = 11H 44M 015   H.FIN = 11H 45M 298   TIME = 0000
                TGP = 00046   MEP = 00013   IO = 0000019C   IN = 00000104

C       LINK NIP
LINK    STARTED

AUCUNE ERREUR A L'EDITION DE LIENS


                CENTRUL DE CALCUL AL ICTCC TIMISOARA                    SYS
                0115  SIMSIT        AN = 0050   PP = 0003   DATE = 01/03/80
                H.DEB = 11H 45M 295   H.FIN = 11H 46M 259   TIME = 0000
                TGP = 00046   MEP = 00013   IO = 00000005   IN = 00000000

C       RUN
        STARTED

                                SINDROMUL DETECTIEI

                PENTRU Nr. UNITATE SI Tr. DEFECTE

*NR * TEST  *UNITATEA CE TESTEAZA*1112223334444555566677778888*
*CRT*UNITATE*UNITATEA TESTATA    *2345678901234567890123456789*
*   1*       1*                  *0000000000000000000000000000*
*   2*       2*                  *0000000000000000000000000000*
*   3*       3*                  *0000000000000000000000000000*
*   4*       4*                  *0000000000000000000000000000*
*   5*       5*                  *0000000000000000000000000000*
*   6*       6*                  *0000000000000000000000000000*
*   7*       7*                  *0000000000000000000000000000*
*   8*      12*                  *0000000000000000000000000000*
```

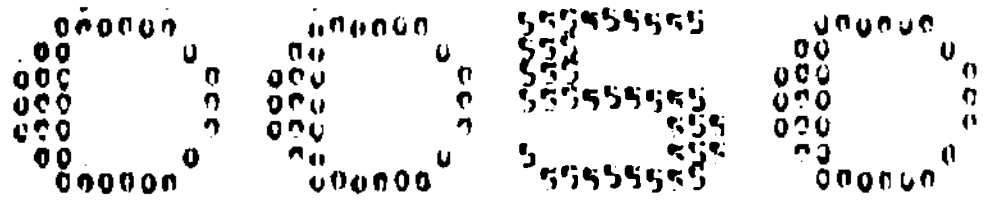
BUPT



BUPT

32	127	[illegible]
33	134	[illegible]
34	142	[illegible]
35	135	[illegible]
36	136	[illegible]
37	137	[illegible]
38	145	[illegible]
39	146	[illegible]
40	147	[illegible]
41	156	[illegible]
42	157	[illegible]
43	167	[illegible]
44	234	[illegible]
45	235	[illegible]
46	236	[illegible]
47	237	[illegible]
48	245	[illegible]
49	246	[illegible]
50	247	[illegible]
51	246	[illegible]
52	257	[illegible]
53	267	[illegible]
54	245	[illegible]
55	246	[illegible]
56	137	[illegible]





"E*T,AN:0050,PN:MIRCEA


```
FORTRAN

                                        SIMTEST    01/03/80  11.40.25


      INTEGER SINDR(36),MIRET(9,9),MATEST(9,9),NSUM(9)
      READ(105,1) N,IT
    1 FORMAT(2I2)
      N1=N*IT
      READ(105,18)(SINDR(J),J=1,N1)
   18 FORMAT(36I1)
      CALL TTPSIND(N,IT,SINDR)
      DO 3 J=1,N
      DO 4 NR=1,IT
      A(K)=I
    4 K=K+1
    3 CONTINUE
      N1=N*IT
      I=2
      K=1
    7 DO 11 NR=1,IT
      H(K)=I
      I=I+1
      IF(I.GT.N)I=1
      K=K+1
      IF(K.GT.N1)GO TO 6
   11 CONTINUE
      K=K-IT
      I=H(K)+1
      IF(I.GT.N)I=1
      GO TO 7
    6 WRITE(105,8)(A(K),K=1,N1)
    8 FORMAT(1X,'*UNITATEA DE TESTARE*',36I1,'*')
      WRITE(105,9)(H(K),K=1,N1)
    9 FORMAT(1X,'*UNITATEA TESTATA   *',36I1,'*')
      WRITE(105,N2)
      WRITE(105,10)(SINDR(K),K=1,N1)
   10 FORMAT(1X,'*SINDROMUL CONSIDERAT*',36I1,'*')
      RETURN
      END

                                        SIMTEST    01/03/80  11.40.53
```

```
      SUBROUTINE TMIRET(PIRET,N)
      INTEGER MIRET(9,9),A(9)
      WRITE(10A,1)
1     FORMAT(//,9X,'MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII:
     *'IRET')
      DO 2 I=1,N
2     A(I)=I
      WRITE(10A,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNITATEA TESTATA=',10I1)
      WRITE(10A,7)
7     FORMAT(10X,2A('*'))
      WRITE(10A,6)
6     FORMAT(10X,'UNIT DE TESTATA=')
      NRC=1
      DO 9 I=1,N
      WRITE(10A,5)NRC,(MIRET(I,J),J=1,N)
5     FORMAT(20X,I2,'=',10I1)
9     NRC=NRC+1
      RETURN
      END
                                        SIMTEST    01/03/80  11:41:02

      SUBROUTINE TIPMATS(MATEST,N)
      INTEGER MATEST(9,9),A(9)
      WRITE(10A,1)
1     FORMAT(//,9X,'MATRICEA TESTELOR PRELUCRATA IN CADRUL UNEI STRUCTU
     *I MULTIPROCESOR: MATEST')
      DO 2 I=1,N
2     A(I)=I
      WRITE(10A,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNIT DE INTRARE=',10I1)
      WRITE(10A,7)
7     FORMAT(10X,2A('*'))
      WRITE(10A,6)
6     FORMAT(10X,'UNIT DE IESIRE =')
      NRC=1
      DO 9 I=1,N
      WRITE(10A,5)NRC,(MATEST(I,J),J=1,N)
5     FORMAT(23X,I2,'=',10I1)
9     NRC=NRC+1
      RETURN
      END
                                        SIMTEST    01/03/80  11:41:10

      SUBROUTINE TIPA(A)
      WRITE(10A,11)
1     FORMAT(/,10X,'UNITATE TESTATA=',I2)
      RETURN
      END
                                        SIMTEST    01/03/80  11:41:10
```

MODULE	PZMDATA	TYPE	P	LONGUEUR	0748 (01896)
MODULE	TIPSTND	TYPE	P	LONGUEUR	0008 (01240)
MODULE	TMIRET	TYPE	P	LONGUEUR	0208 (00820)
MODULE	TIPMATS	TYPE	P	LONGUEUR	0218 (00836)
MODULE	TIPA	TYPE	P	LONGUEUR	0040 (00098)

```
ATION (PLUS HAUT NIVEAU D'ERREUR RENCONTRE = 0)                 11:41:20

RUN DE CALCUL AL ISTELC TIPSUARA                          SYSTEM MOL
  SIMTEST    AN = 0030   PI = 0082   DATE = 01/03/80-060
P = 11H AOM 1A5   ICFIN = 11H 41P 20S   TIME = 00001A20
= 00046   MEM = 00213   IO = 00000151   IN = 00000101   CODE = 000
```

```
LINK      16.50.09    01/03/86  11:42:17.5

  SEGMENT   FYMDATA    NO    1     IMPLANTATION      0
            MODULE    FYMDATA     IMPLANTATION     70
            MODULE    TIPSIND     IMPLANTATION    7C8
            MODULE    TMTREF      IMPLANTATION    CD0
            MODULE    TIPMATS     IMPLANTATION    E08
            MODULE    TIPF        IMPLANTATION   10C0
            MODULE    TDFSYSUT    IMPLANTATION   1130
            MODULE    TDFSYSTM    IMPLANTATION   11A0
            MODULE    TIINIT      IMPLANTATION   1220
            MODULE    IZREAD      IMPLANTATION   16FC
            MODULE    IZTOT       IMPLANTATION   1900
            MODULE    IZENDIOL    IMPLANTATION   19C0
            MODULE    IZSTOF      IMPLANTATION   19F8
            MODULE    IZPRINT     IMPLANTATION   1B10
            MODULE    IZENCODE    IMPLANTATION   1D60
            MODULE    IZERRI      IMPLANTATION   1E30
            MODULE    STOPRUN     IMPLANTATION   1F58
            MODULE    IZIECARI    IMPLANTATION   1FC8
            MODULE    IZIORMA     IMPLANTATION   22F8
            MODULE    IZECRAPI    IMPLANTATION   4908
            MODULE    IZDONNEE    IMPLANTATION   5C70
                                  LONGUEUR DU SEGMENT 5CF0

LINK      16.50.09    01/03/86  11:42:17.5

                      IMPLANTE APRES TRAITEMENT OPTION IMS

  SEGMENT   FYMDATA    NO    1     IMPLANTATION      0
                                  LONGUEUR DU SEGMENT 5C70

LINK      16.50.09    01/03/86  11:42:17.5

     0     ERREUR  EN EDITION DE LIENS
     ADRESSE DE LANCEMENT                          408
     LONGUEUR PLUS GRANDE BRANCHE                 5C70
     LONGUEUR DU PROGRAMME EDITE                  5C70
                  PLUS HAUT NIVEAU D'ERREUR RENCONTRE  N=0   (PAS D'ERREUR)


                  CENTRUL DE CALCUL AL I.P.C.T. TIMISOARA                    SYSTEM 800
                  JOB:  SYSTEST    AN = 0050   PP = 0003   DATE = 01/03/86-060
                  H.ORD = 11H 41M 25S   H.FIN = 11H 42M 22S   TIME = 00000057
                  ICP = 00046   MEM = 00012   IO = 00000048   IN = 00000000  CODE = 000

0          RUN
           STARTED

           SINDROMUL DEFECTIUNII GASITE IN CADRUL UNEI STRUCTURI  MULTIPROCESOR AUTOTESTABILE

 SINDROMUL ESTE PREZENTAT DIN REGISTRUL SINDROAMELOR AVIND URMATOAREA CONFIGURATIE:
 N=0  IT=0
*****************************************************
*UNITATEA DE TESTARE=0123456789ABCDEF0123456789ABCDEF*
*UNITATEA TESTATA   =2345678A9BCDEF0123456789ABCDEF01*
*****************************************************
*SINDROMUL CONSIDERAT=0000000100000000000000010101010*


           MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII; NIPLI


            UNITATEA TESTATA=123456789
            ***************************
            UNIT DE TESTARE=
```

2***1011***

3****0000**

4*****1100*

5*******0000

6*0******000

7*10******0

8*101*****2

9*1010*****

MATRICEA TESTELOR PREFIXATA IN CADRUL UNEI STRUCTURI MULTIPROCESOR: MATEST

UNITATE DE TRATARE:123456789

**********+*+**********

UNITATE DE TESTARE *

1*001011000

2*101011000

3*101011000

4*101011000

5*101001000

6*101010000

7*101011000

8*101011000

9*101011000

UNITATEA DE EXECUTIE 2

UNITATEA DE EXECUTIE 3

UNITATEA DE EXECUTIE 5

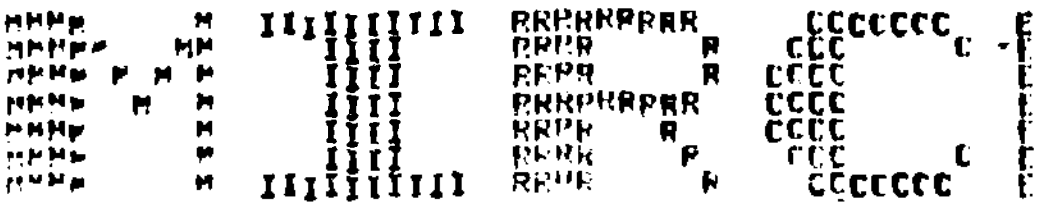
UNITATEA DE EXECUTIE 6

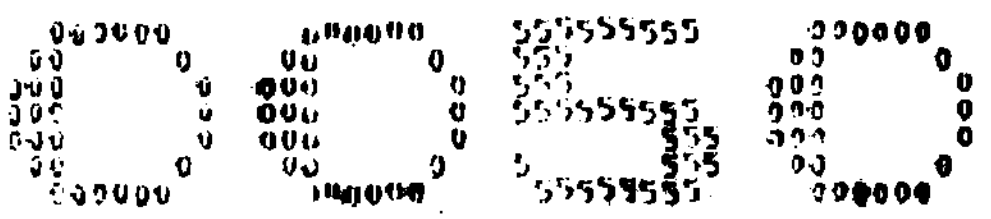
STOP

CENTRUL DE CALCUL AL I.P.T.C.N. TIMISOARA SYSTEM 800
211: SYMTEST AN = 0050 PP = 0004 DATE = 01/03/86-060
P.OFF = 11H 42M 233 RCLIN = 11H 42M 318 TIME = 00000219
LCP = 00006 PLP = 0001? LO = 00000000 TA = 00000003 CODE = 000

ANEXA C - PR.22

SIMATEST





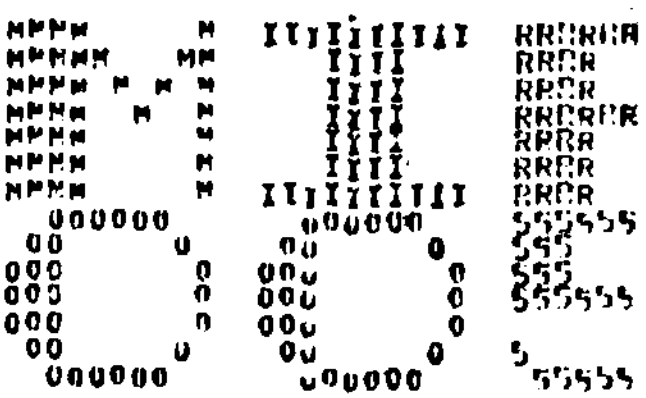
ATLANTIC, AN=0059, PN=MIHAELA

```
PROIECTUL DE CALCUL AL LUCRARII DE LICENTA                    SYSTEM 360
/* SIMATEST   AN = 0059   PN = 0001   DATE = 01/03/86   PRIORITY = 060
// TIME = 11H 37M 12S   TIME LIMIT = 11H 37M 15S   TIME = 00000199
// REGION = 00040K   REM = 00000   I/O = 00000022   IN = 00000002   CODE = 000
// EXEC FORTRAN
1                                                SIMATEST   01/03/86   11.37.37

      INTEGER SINDR(30),MIRE(9,9),MCONEX(9,9),MATLST(9,9),NSUM(9)
      READ(10,1)N,IT
    1 FORMAT(2I2)
      WRITE(108,5)N,IT
    5 FORMAT(2X,'N=',I1,2X,'IT=',I1)
      WRITE(108,20)
   20 FORMAT(2X,'MATRICEA CONEXIUNILOR MCONEX')
      DO 2 I=1,N
      READ(10,3)(MCONEX(I,J),J=1,N)
    3 FORMAT(10I1)
      WRITE(108,4)(MCONEX(I,J),J=1,N)
    4 FORMAT(/,10X,10I2)
    2 CONTINUE
      I=1+IT
      READ(10,18)(SINDR(J),J=1,N1)
   18 FORMAT(30I1)
      WRITE(108,19)(SINDR(J),J=1,N1)
   19 FORMAT(1X,'SINDROAME:',/,30I2)
      DO 9 I=1,N
      DO 7 J=1,N
    7 MIRE(I,J)=100
    9 CONTINUE
      K=1
      DO 8 I=1,N
      DO 8 J=1,N
      IF(MCONEX(I,J).NE.1)GO TO 8
      MIRE(I,J)=SINDR(K)
      K=K+1
    8 CONTINUE
      CALL TRIKL(MIRE,N)
      DO 6 I=1,N
      DO 6 J=1,N
    6 MATLST(I,J)=100
      CONTINUE
      K=1
   25 DO 4 I=1,N
```

```
      IF (MIRET(I,J).EQ.1) GO TO 21
      IF (MIRET(J,I).EQ.1) GO TO 21
      IF (MATEST(I,J).EQ.0) GO TO 12
11    DO 22 K=1,N
      IF (MIRET(I,K).EQ.1) GO TO 21
      IF (MIRET(J,K).EQ.1) GO TO 21
      IF (MATEST(I,K).EQ.0) GO TO 12
21    MATEST(I,K)=1
                                        $MATEST   01/03/86  11:37.37

12    CONTINUE
10    CONTINUE
9     CONTINUE
      IF (NR.NE.0) GO TO 22
      NR=1
      DO 23 I=1,N
      DO 24 J=1,N
24    MIRET(I,J)=MATEST(I,J)
23    CONTINUE
      GO TO 25
22    DO 26 I=1,N
      DO 27 J=1,N
27    IF (MATEST(I,J).GE.100)MATEST(I,J)=0
26    CONTINUE
      CALL TIPMAT(MATEST,N)
      DO 13   K=1,N
      NSUM(K)=0
      DO 14   J=1,N
14    NSUM(K)=NSUM(K)+MATEST(J,K)
13    CONTINUE
      DO 15 K=1,N
      IF (NSUM(K).GT.1) CALL TIPA(K)
15    CONTINUE
      STOP
      END
                                        $MATEST   01/03/86  11:37.52

      SUBROUTINE TMIRET(MIRET,N)
      INTEGER MIRET(9,9),A(9)
      WRITE(108,1)
1     FORMAT(//,9X,'MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII;
     *MIRET')
      DO 2 I=1,N
2     A(I)=I
      WRITE(108,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNITATEA TESTATA=',10I1)
      WRITE(108,7)
7     FORMAT(10X,20('*'))
      WRITE(108,6)
6     FORMAT(10X,'UNIT CE TESTEAZA=')
      NRC=1
      DO 4 I=1,N
      WRITE(108,5)NRC,(MIRET(I,J),J=1,N)
5     FORMAT(24X,I2,'=',10I1)
4     NRC=NRC+1
      RETURN
      END
                                        $MATEST   01/03/86  11:38.00

      SUBROUTINE TIPMAT(MATEST,N)
      INTEGER MATEST(9,9),A(9)
      WRITE(108,1)
1     FORMAT(//,9X,'MATRICEA TESTELOR PRELUCRATA IN CADRUL UNEI STRUCTU
     *I MULTIPROCESOR; MATEST')
      DO 2 I=1,N
2     A(I)=I
      WRITE(108,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNIT DE INTRARE=',10I1)
      WRITE(108,7)
7     FORMAT(10X,20('*'))
      WRITE(108,6)
6     FORMAT(10X,'UNIT DE IESIRE =')
      NRC=1
      DO 4 I=1,N
      WRITE(108,5)NRC,(MATEST(I,J),J=1,N)
5     FORMAT(23X,I2,'=',10I1)
4     NRC=NRC+1
      RETURN
      END
                                        $MATEST   01/03/86  11:38.11

      SUBROUTINE TIPA(K)
      WRITE(108,1)K
1     FORMAT(/,10X,'UNITATE DEFECTA=',I2)
      RETURN
      END
```

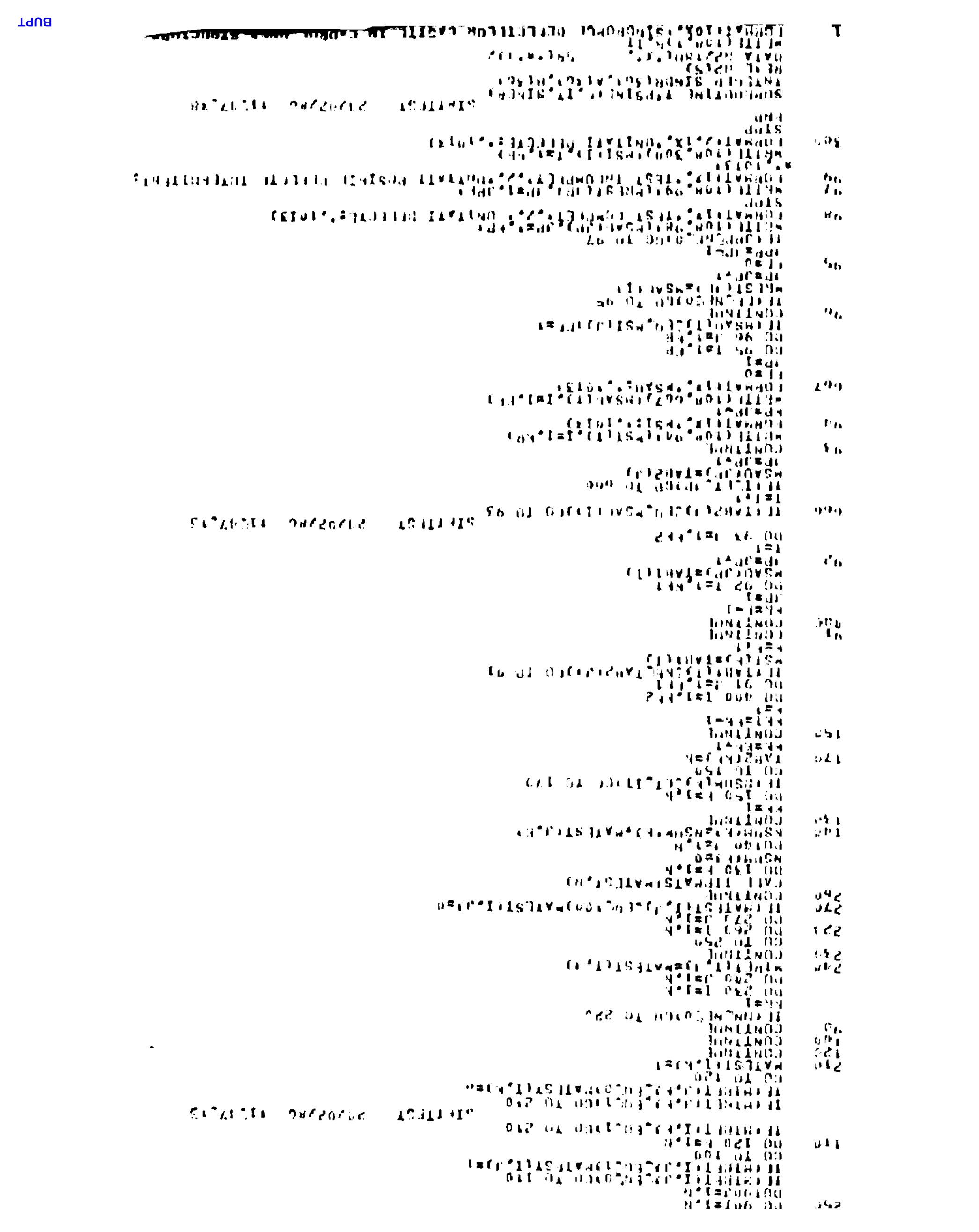
```
STARTED
N=5  IT=2
MATRICEA CONEXIUNILOR MCONEX

          0 0 1 0 1

          1 0 0 1 0

          0 1 0 1 0

          1 0 0 0 1

          0 1 1 0 0
SINDROMUL:
2 0 1 0 0 1 0 0 1 1


        MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII: MINET

         UNITATEA TESTATA=12345
         ***********************
         UNITATI DE TESTARE=
                        1=0,1,4
                        2=0,3,4
                        3=0,1,4
                        4=0,3,4
                        5=0,1,4

        MATRICEA TESTELOR PREVAZUTE IN CADRUL UNEI STRUCTURI MULTIPROCESOR: MATEST

         UNITI DE INTRARE=12345
         **********************
         UNITI DE TESTARE =
                        1=01100
                        2=11100
                        3=10010
                        4=01100
                        5=01100

         UNITATE DEFECTA= 2

         UNITATE DEFECTA= 4
*STOP*

CENTRUL DE CALCUL AL I.P.T.C. TIMISOARA                    SYSTEM D50
0107  SIMATEST   AN = 005C   PN = 0004   DATE = 01/03/80-060
E_TIME = 110 39M 50S   U_TIM = 110 39M 51S   TIME = 00000107
LCP = 0004C   MEM = 00011   LO = 00000002   HI = 00000008   CODE = 000
```

BUFT



JOB SIPTEST.AN;0050.PN;PIRCEA
EXITME STANTFO

```
MTL DE CALCUL AL IST-L. TIMISOARA                    SYSTEM OPO
  SIMITEST   AN = 0050   PR = 0001   DATE = 24/02/80-055
0 = 11H 40M 49S   OCTIN = 11H 47M 06S   TIME = 00000127
= 00065   MEM = 00906   IO = 00000021   IN = 00000001   CODE = 000
21 CR = 01
FORTRAN

                                                SIMITEST   24/02/80   11.47.13
      INTEGER SINDR(36),MIRET(9,9),MATEST(9,9),NS(9),TAB1(9),TAB2(9)
      INTEGER MSI(9),NSAU(9),PP,MCONEX(9,9),MPRST(9)
      READ(105,1) N,IT
    1 FORMAT(2I2)
      NI=N*IT
      READ(105,18)(SINDR(J),J=1,NI)
   18 FORMAT(80I1)
      CALL TTPSIND(N,IT,SINDR)
      DO 3 J=1,N
      DO 4 K=1,N
    4 MIRET(J,K)=1000
    3 CONTINUE
      I=1
      DO 5 J=1,N
      K=I+1
      IF(K.GT.N)K=K-N
      DO 6 M=1,IT
      MIRET(J,K)=SINDR(I)
      K=K+1
      IF(K.LE.N)GO TO 6
      K=K-N
    6 I=I+1
    5 CONTINUE
      CALL TMTRET(MIRET,N)
      DO 16 I=1,N
      DO 7  M=1,N
    7 MATEST(I,M)=0
   16 CONTINUE
      I=1
      J=I
   12 K=I+1
   10 IF(K.GT.N)GO TO 8
      IF(MIRET(J,K).EQ.1)GO TO 9
      K=K+1
   11 K=K+1
      IF(K.LE.N) GO TO 10
      K=1
      GO TO 10
    9 MATEST(I,K)=1
      GO TO 11
    8 I=I+1
      IF(I.LE.N) GO TO 12
      CALL TTPMATS(MATEST,N)
      DO 13  K=1,N
      NSUM(K)=0
      DO 14  J=1,N
   14 NSUM(K)=NSUM(K)+MATEST(J,K)
                                                SIMITEST   24/02/80   11.47.13
   13 CONTINUE
      KK=1
      DO 15 K=1,N
      IF(NSUM(K).GT.1)GO TO 17
      GO TO 15
   17 TAB1(KK)=K
      KK=KK+1
   15 CONTINUE
      KK2=KK-1
      WRITE(108,200)
  200 FORMAT(27,'MATRICEA CONEXIUNILOR MCONEX=')
      DO 20 I=1,N
      READ(105,30)(MCONEX(I,J),J=1,N)
   30 FORMAT(10I1)
      WRITE(108,40)(MCONEX(I,J),J=1,N)
   40 FORMAT(/,10X,10I2)
   20 CONTINUE
      NI=N*IT
      READ(105,180)(SINDR(J),J=1,NI)
  180 FORMAT(80I1)
      WRITE(108,190)(SINDR(J),J=1,NI)
  190 FORMAT(1X,'SINDROAME:',/,36I2)
      DO 60 I=1,N
      DO 710 J=1,N
  710 MIRET(I,J)=100
   60 CONTINUE
      NR=1
      DO 810 I=1,N
      DO 910 J=1,N
      IF(MCONEX(I,J).EQ.1)GO TO 910
      MIRET(I,J)=SINDR(NR)
      NR=NR+1
  910 CONTINUE
  810 CONTINUE
      CALL TMTRET(MIRET,N)
      DO 70 I=1,N
      DO 80 J=1,N
   80  MATEST(I,J)=100
   70 CONTINUE
      NN=1
```



```
      *STRUL SINONIMELOR AVIND LUNGIMEA EGALA CU LUNGIMEA',/,2X,'MAS',I1,
     *2X,'IT=',I1)
      N3=13*N+IT
      ENCODE(4,5,N2(3))N3
5     FORMAT(I4)
      WRITE(10H,42)
      K=1
      DO 5 I=1,N
      DO 4 NP=1,IT
      A(K)=I
      K=K+1
4     CONTINUE
5     CONTINUE
      N1=N*IT
      I=2
      K=1
7     DO 11 NP=1,IT
      B(K)=I
      I=I+1
      IF(I.GT.N)I=1
      K=K+1
      IF(K.GT.N1)GO TO 6
11    CONTINUE
      K=K-IT
      I=B(K)+1
      IF(I.GT.N)I=1
      GO TO 7
6     WRITE(10H,8)(A(K),K=1,N1)
8     FORMAT(1X,'*UNITATEA DE TESTARE=*',36I1,'*')
      WRITE(10H,9)(B(K),K=1,N1)
9     FORMAT(1X,'*UNITATEA TESTATA    =*',36I1,'*')
      WRITE(10H,42)
      WRITE(10H,10)(SINON(K),K=1,N1)
10    FORMAT(1X,'*SINONIMELE CONSIDERATE=*',36I1,'*')
      RETURN
      END
                                                SINTEST   24/02/80  11.47.46
      SUBROUTINE TMIRET(MIRET,N)
      INTEGER MIRET(9,9),A(9)
      WRITE(10H,1)
1     FORMAT(//,9X,'MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII',
     *MIRET')
      DO 2 I=1,N
      A(I)=I
2     WRITE(10H,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNITATEA TESTATA=*',10I1)
      WRITE(10H,7)
7     FORMAT(10X,20('*'))
      WRITE(10H,6)
6     FORMAT(10X,'UNIT DE TESTARE')
      NRC=1
      DO 4 I=1,N
      WRITE(10H,5)NRC,(MIRET(I,J),J=1,N)
5     FORMAT(2X,I2,'*',10I1)
4     NRC=NRC+1
      RETURN
      END
                                                SINTEST   24/02/80  11.47.51
      SUBROUTINE TIPMATS(MATEST,N)
      INTEGER MATEST(9,9),A(9)
      WRITE(10H,1)
1     FORMAT(//,9X,'MATRICEA TESTELOR PRELUCRATA IN CADRUL UNEI STRUCTU
     *I MULTIPROCESOR, MATEST')
      DO 2 I=1,N
2     A(I)=I
      WRITE(10H,3)(A(I),I=1,N)
3     FORMAT(/,10X,'UNIT DE INTRARE=*',10I1)
      WRITE(10H,7)
7     FORMAT(10X,20('*'))
      WRITE(10H,6)
6     FORMAT(10X,'UNIT DE TESIRE *')
      NRC=1
      DO 4 I=1,N
      WRITE(10H,5)NRC,(MATEST(I,J),J=1,N)
5     FORMAT(2X,I2,'*',10I1)
4     NRC=NRC+1
      RETURN
      END
                                                SINTEST   24/02/80  11.47.51

MODULE    SYMDATA         TYPE   P    LONGUEUR:    1210 (04B4R)
MODULE    TIPSIND         TYPE   P    LONGUEUR:    0108 (00120)
MODULE    TMIRET          TYPE   P    LONGUEUR:    0208 (00520)
MODULE    TIPMATS         TYPE   P    LONGUEUR:    0210 (00536)
TATION (PLUS HAUT NIVEAU D'EMPILEMENT PENDANT LE = 01                    11.47.56
TAPE DE CALCUL AL ICTEC. TIPINDARA                        SYSTEM B06
P  SYMTEST   AN = 0050   PP = 0002   DATE = 24/02/80-056
** = 110 47M 08S   HEURE = 110 47M 56S   TYPE = 00002035
L = 00065   REM = 00013   LU = 00000270   ID = 0000325B   CODE = 050
P = 01 CR = 01


24/02/80  11H48M35S
```

```
MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII:  MIT

 UNITATEA TESTATA=1234567
 ***********************
 UNIT DE TESTARE
                  1=**11***1
                  2=1**1*11**
                  3=**0*1*0*
                  4=**1*1***0
                  5=1**1*1**0
                  6=11*1*0**
                  7=*10*1*0*


MATRICEA TESTELOR PREGATITA IN CADRUL UNEI STRUCTURI MULTIPROCESOR: MATEST

 UNIT DE PREPARE=1234567
 ***********************
 UNIT DE TESTE *
                  1=0011001
                  2=1001100
                  3=1101100
                  4=1111000
                  5=1101100
                  6=1101000
                  7=1101000
MSI:  1  2
MSA0:  1  2  3  4
TEST INCOMPLET
MITATI POSIBIL DEFECTE INTERMITENTE:  3  4

UNITATI DEFECTE:  1  2
*STOP*
                CENTRUL DE CALCUL AL ICPIAL TIMISOARA                    SYSTEM B06
                OPER. SIMITEST   AN = 0050   PR = 0004   DATE = 24/02/80-050
                ORA = 11H 48M 29S   ORA = 11H 48M 36S   TIMP = 00000707
                ICP = 00065   MEM = 00013   IO = 00000794   IR = 00000011   CODE = 000
                PR = 01 CR = 01
```

```
  SEGMENT    I?MDATA    NO    1      IMPLANTATION      0
             MODULE     ISMDATA      IMPLANTATION     70
             MODULE     TIPSTRO      IMPLANTATION   1360
             MODULE     TYTEST       IMPLANTATION   1838
             MODULE     TIPMAIS      IMPLANTATION   1A90
             MODULE     TOFSYSOUT    IMPLANTATION   1C50
             MODULE     TOFSYSIN     IMPLANTATION   1CF8
             MODULE     IIINIT       IMPLANTATION   1D50
             MODULE     IIMLAD       IMPLANTATION   2208
             MODULE     IITOI        IMPLANTATION   2428
             MODULE     IIERROIOL    IMPLANTATION   23F8
             MODULE     IIPPTRT      IMPLANTATION   2520
             MODULE     IISTOF       IMPLANTATION   2770
             MODULE     IIERROOL     IMPLANTATION   2AF8
             MODULE     IXLDRI       IMPLANTATION   2B58
             MODULE     STOPPUN      IMPLANTATION   2AB0
             MODULE     IIILCARI     IMPLANTATION   2AE0
             MODULE     IIFOPDA      IMPLANTATION   2DE0
             MODULE     IIFCRARI     IMPLANTATION   0030
             MODULE     IYDOSOUT     IMPLANTATION   0798
                                     LONGUEUR DU SEGMENT 0B1A

LINK       16_50C09     24/02/80   11H48M24S
                        IMPLANT. APRES TRAITEMENT OPTION FMS

  SEGMENT    FIMDATA    NO    1      IMPLANTATION      0
                                     LONGUEUR DU SEGMENT 6398

LINK       16_50C09     24/02/80   11H48M24S
        0      ERREUR  EN FIN DE LISTE
           ADRESSE DE LANCEMENT                    810
           LONGUEUR PLUS GRANDE BRANCHE           6398
           LONGUEUR DU PROGRAMME EDITE            6398
                  PLUS HAUT NIVEAU D'ERREUR RENCONTRE  0.   (PAS D'ERREUR)
     CENTRE DE CALCUL AL ISTEL TYMISOARA                    SYSTEM B06
     0130  SIMITEST   AN = 0050   MN = 0005   DATE = 24/02/80-05'5
     H_DEP = 11H 47M 59S   H_FIN = 11H 48M 20S   TIME = 00000020
     ICP = 00005   MEM = 00012   IO = 00000048   IE = 00000000   CODE = 000
     PE = 01 CC = 01
C
        RUN
        STARTED
        SINDROMUL DETECTIEI GASITE IN CADRUL UNEI STRUCTURI MULTIPROCESOR AUTOTESTABILE

 SINDROMUL ESTE PREZENTAT DIN REGISTRUL SINDROAMELOR AVAND URMATOAREA CONFIGURATIE:
 N=7  I1=5
 ***********************
 *UNITATEA DE TESTARE  *1112225556004455566777
 *UNITATEA TESTATA     *2340567567671712123
 ***********************
 *SINDROMUL  CONSTATAT *1000000000001011111


        MATRICEA INTERCONEXIUNILOR SI REZULTATUL TESTARII:  MITEST

         UNITATEA TESTATA=1234567
         ************************
         UNITATEA DE TESTARE:
                          1=*000****
                          2=**=000**
                          3=***=000*
                          4=****=000
                          5=1***=000
                          6=11*****0
                          7=111*****


        MATRICEA TESTELOR PREZENTATA IN CADRUL UNEI STRUCTURI MULTIPROCESOR: MATEST

         UNITATEA TESTATA=1234567
         ************************
         UNITATEA DE TESTARE *
                          1=0000000
                          2=1110000
                          3=1100000
                          4=1110000
                          5=1110000
                          6=1110000
                          7=0000000
 MATRICEA CONEXIUNILOR MCONEX

            0 0 1 1 0 1 1

            1 0 0 1 1 0 0

            0 1 0 0 1 1 0

            0 0 1 0 0 1 1

            1 0 0 1 0 0 1

            1 1 0 0 1 0 0

            0 1 1 0 0 1 0
SINDROMUL:
1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0
```